РУКОВОДСТВО ПО ПОДГОТОВКЕ К DATA SCIENCE ИНТЕРВЬЮ

Автор
Ренат Алимбеков

Первое издание
2021

Статистика и теория вероятности

Что такое центральная предельная теорема? Объясните ее. Почему она важна?

Центральная предельная теорема гласит, что выборочное распределение выборочного среднего приближается к нормальному распределению по мере того, как размер выборки увеличивается, независимо от формы распределения генеральной совокупности

Центральная предельная теорема важна, потому что она используется при проверке гипотез, а также для вычисления доверительных интервалов.

Что такое статистическая мощность?

Статистическая мощность - вероятность отклонения основной (или нулевой) гипотезы при проверке статистических гипотез в случае, когда конкурирующая (или альтернативная) гипотеза верна.

$$Power = P(reject\ Null\ |alternative\ is\ true)$$

Как поступать с отсутствующими данными? Какие методы вы рекомендуете?

Есть несколько способов исправить недостающие данные:

- Удалить строки с отсутствующими данными
- Среднее / Медиана / Мода
- Присвоение уникального значения
- Прогнозирование недостающих значений
- Использование алгоритма, поддерживающего пропущенные значения, например случайный лес

Наилучшим методом является удаление строк с отсутствующими данными, поскольку это гарантирует, что смещение или отклонение не будет добавлено или удалено, и в конечном итоге приведет к созданию надежной и точной модели. Однако это можно рекомендовать только в том случае, если есть достаточно данных и процент пропущенных значений невелик.

Распределения

Нормальное распределение

Нормальное распределение, также известное как распределение Гаусса. Нормальное распределение, вероятно, является самым популярным распределением вероятностей. Это непрерывное распределение в форме колокола, симметричное среднему значению. Функция плотности вероятности для нормального распределения выглядит следующим образом:

$$f(x \mid \mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

где μ - среднее значение, а σ - стандартное отклонение распределения.

Среднее значение (μ): Среднее значение всех точек в сэмпле.

Стандартное отклонение (σ): насколько набор данных отклоняется от среднего значения выборки.

Некоторые характерные особенности нормального распределения заключаются в следующем:

1. Оно симметрично относительно среднего.
2. Оно следует колоколообразной кривой. Каждая кривая колокола не обязательно должна быть нормальным распределением, но каждое нормальное распределение является кривой колокола.
3. Среднее значение, медиана и моды равны.
4. Общая площадь под кривой равна 1.
5. 68,26% данных находится в пределах одного стандартного отклонения от среднего.
6. 95,44% данных находится между двумя стандартными отклонениями среднего.
7. 99,73% данных лежат между тремя стандартными отклонениями среднего.

Нормальное распределение получает свою важность из **Центральной предельной теоремы**, которая гласит, что если мы возьмем достаточно большое количество выборок, их среднее будет следовать нормальному распределению независимо от начального распределения выборки, то есть распределения среднего значения выборок нормально. Важно, чтобы каждый сэмпл не зависел от другого.

Позвольте представить новую переменную с именем **z. Z** - разница между каждым элементом данных и средним значением, деленное на стандартное отклонение. Формула:

$$\mathbf{Z} = \frac{x-\mu}{\sigma}$$

Интересная особенность z заключается в том, что:

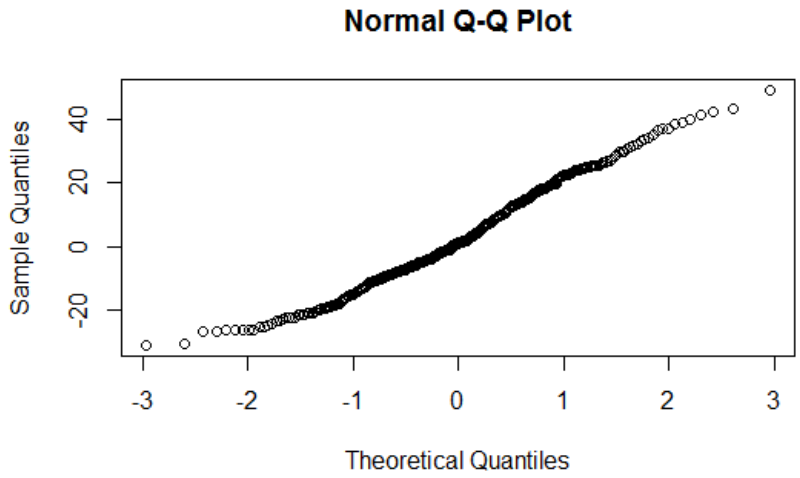
E(z) = 0 (Expected value = 0)

V(Z) = 1 (Variance =1)

Процесс преобразования значений в столбце в соответствующие им значения z называется **стандартизацией**. Кроме того, в данных, которые следует нормальному распределению, даже значения z подчиняются нормальному распределению. Итак, мы можем сказать, что когда, $X \sim N$ (mu, $sigma^2$), это означает, что следуют соответствующие значения z, $Z \sim N$ ($0,1^2$).

Как мы проверяем, соответствует ли переменная нормальному распределению?

1. Постройте гистограмму из выборочных данных. Если вы можете подогнать колоколообразную «нормальную» кривую к гистограмме, то гипотезу о том, что основная случайная величина следует нормальному распределению, нельзя отвергнуть.
2. Проверьте Skewness и Kurtosis выборочных данных. Skewness = 0 и Kurtosis = 3 типичны для нормального распределения, поэтому, чем дальше они от этих значений, тем более ненормальное распределение.
3. Используйте тесты Колмогорова-Смирнова и / или Шапиро-Уилка на нормальность. Они одновременно учитывают асимметрию и эксцесс.
4. Проверьте график квантиля-квантиля. Это диаграмма рассеяния, созданная путем сопоставления двух наборов квантилей друг с другом. На нормальном графике Q-Q точки данных располагаются примерно по прямой линии.

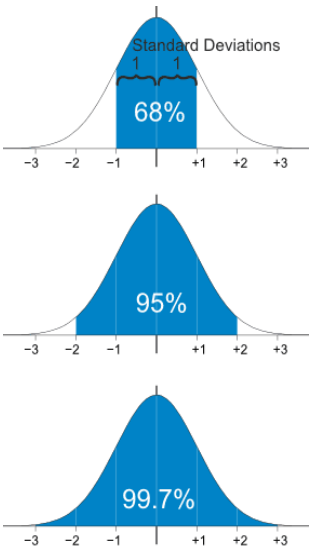


Что такое стандартное нормальное распределение?

Да, это так просто, как и кажется. Это стандартизация («освобождение данных от ограничений какой-либо шкалы») нормального распределения со средним значением $\mu = 0$ и стандартным отклонением $\sigma = 1$. Это означает, что считается любая нормально распределенная кривая со средним значением 0 и стандартным отклонением 1. как стандартное нормальное распределение.

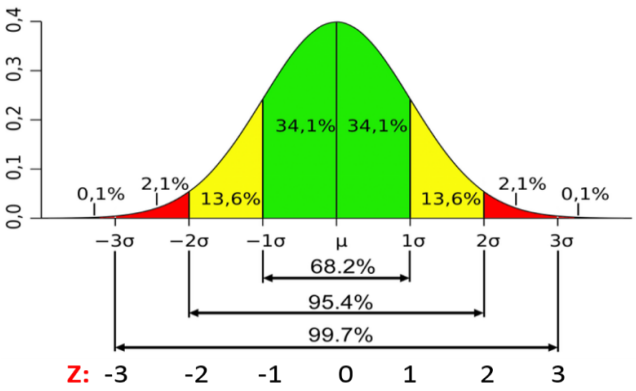
Правило силы 68–95–99,7 для колоколообразной кривой!

Правило 68–95–99.7 говорит о том, как данные распределяются по функции распределения вероятностей (Probability Distribution Function - PDF) нормального распределения.



Источник: Pinterest Normal Rule на Pinterest

PDF используется для указания вероятности того, что случайная величина (x) попадает в определенный диапазон значений нормального распределения, или, простыми словами, PDF - это сглаживание нормально распределенной гистограммы. Это правило объясняет, что когда мы строим кривую функции нормального распределения со средним значением (μ) и стандартным отклонением (σ), то 68% точек данных лежат в диапазоне от -1σ до 1σ, аналогично 95% точек данных лежат между - От 2σ до 2σ, и 99,7% точек данных лежат в диапазоне от -3σ до 3σ.



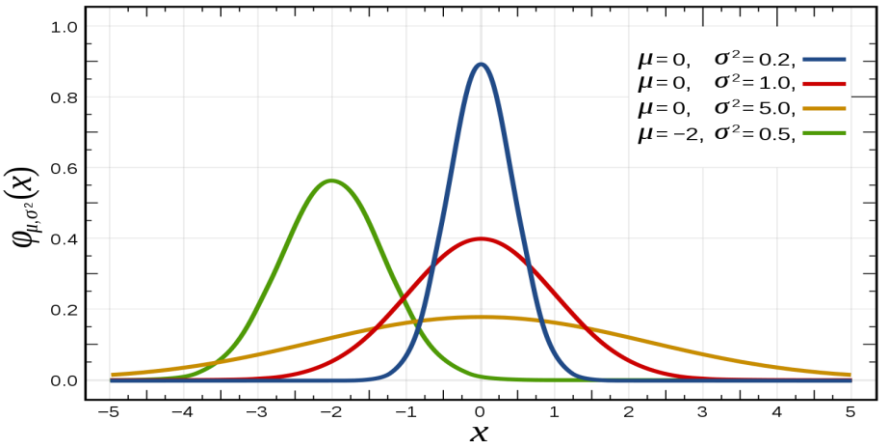
Источник: Sphweb, Характеристики нормального распределения.

На рисунке выше мы можем ясно видеть, что нормальное распределение симметрично разделено на 3 цветные области. Зеленый цвет составляет 68% от полных точек данных. Зеленый + желтый - 95% всех данных, а зеленый + желтый + красный - 99,7% полного набора данных.

? Можно ли сместить или наклонить распределение в одну сторону?

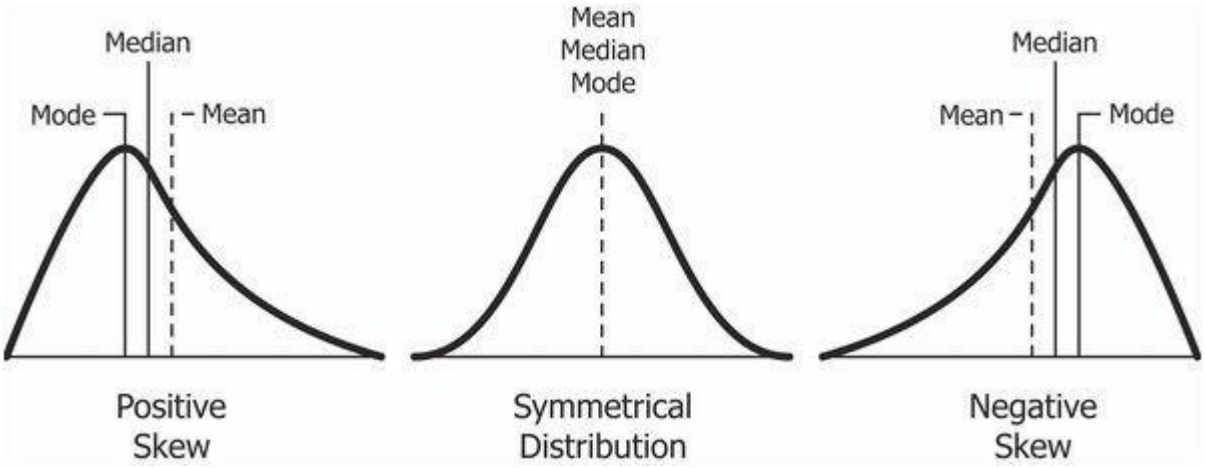
Да, сдвиг (влево или вправо) изменит среднее значение (μ) нормального распределения без изменения его формы. Однако масштабирование («диапазон» от конца до конца) изменит стандартное отклонение (σ) кривой, изменив ее форму.

Примечание: Масштабирование не подразумевает какой-либо пропорциональности с высотой кривой, а просто означает, насколько толстая кривая. Чем больше значение σ, тем толще кривая по оси абсцисс.



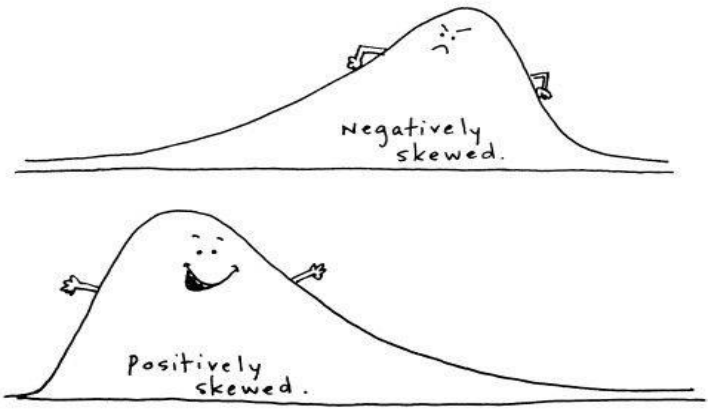
Источник: Википедия Типы нормальных кривых

Мы измеряем перекос распределения, видя его «более толстый хвост», но не его пик (который, очевидно, будет на противоположной стороне от более толстого хвоста).



Источник: Pinterest Перекос Нормального Распределения

- Положительный перекос: когда хвост толще к правому концу, это называется наклонным вправо или положительным перекосом.
- Отрицательный перекос: когда хвост толще к левому концу, он называется скошенным влево или отрицательно скошенным.
- Нет перекоса: когда есть симметрия с обеих сторон кривой без какого-либо дифференцируемого толстого хвоста с одной стороны.



Источник: : Resourceaholic, Положительный и Негативный перекос.

Эй, не волнуйся! Мы можем превратить искаженное нормальное распределение в симметричное нормальное распределение, взяв его логарифмическое нормальное распределение.

Биномиальное распределение

Биномиальное распределение часто называют распределением вероятности подбрасывания монеты. Он измеряет вероятность двух событий, одно из которых происходит с вероятностью p, а другое - с вероятностью 1-p.

- Биномиальное распределение - это дискретное распределение.
- Биномиальное распределение используется для представления вероятности успеха x в n испытаниях при заданной вероятности успеха p в каждом испытании.
- Если распределение удовлетворяет следующим условиям, то такое распределение называется биномиальным распределением:
  1. Должно быть фиксированное количество попыток.
  2. У него должно быть только два возможных исхода.
  3. События должны быть независимыми.
  4. Вероятность успеха и неудачи должна оставаться неизменной.

Распределение Бернулли

- Распределение Бернулли является самым простым распределением среди всех.
- Оно похоже на биномиальное распределение. Единственная разница в том, что требуется только одна попытка, в то время как при биномиальном распределении учитывается n попыток.
- У него есть только два возможных исхода: успех или неудача.
- Рассмотрим случайную величину X только с одним параметром p, который представляет вероятность наступления события.
- Функция плотности:
  - P[X=1]=p
  - P[X=0]=1-p
  - Где X = 1 указывает, что событие произошло, а X = 0 указывает, что событие не произошло.

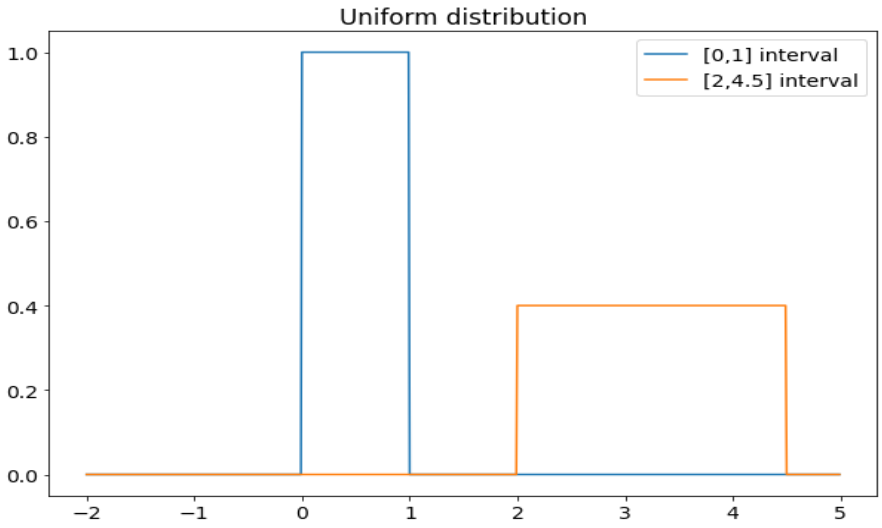
Равномерное распределение

Простейшее распределение вероятностей - это равномерное распределение, которое дает одинаковую вероятность для любых точек набора.

В своей непрерывной форме равномерное распределение между a и b имеет функцию плотности:

$$f(x) = \frac{1}{b-a}$$

А вот как это выглядит:



Как видите, чем шире диапазон, тем ниже распределение. Это нужно, чтобы площадь оставалась равной единице.

- Распределение называется равномерным, если все исходы события имеют равные вероятности.
- Равномерное распределение также называется прямоугольным.
- Ожидаемое значение равномерного распределения не дает нам соответствующей информации.
- Поскольку каждый результат одинаково вероятен, среднее значение и дисперсия не поддаются интерпретации.

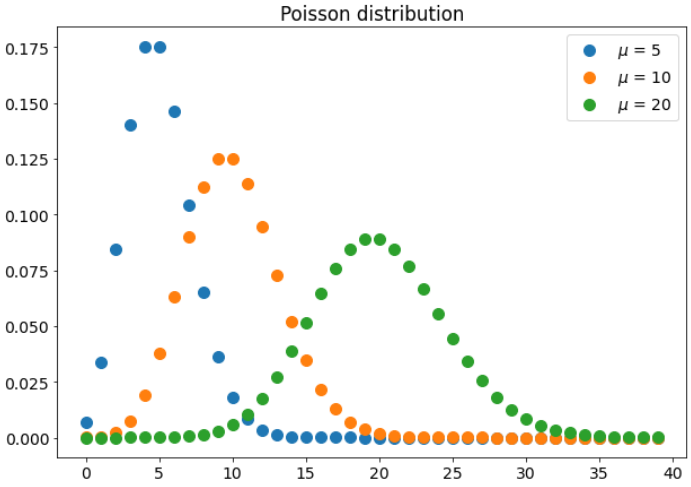
Распределение Пуассона

Распределение Пуассона также называется распределением редких событий. В общем, если у вас есть событие, которое происходит с фиксированной скоростью во времени (т.е. 3 события в минуту, 5 событий в час), вероятность наблюдения числа n событий в единицу времени может быть описана распределением Пуассона, которое имеет эту формулу:

$$f(n, \mu) = \frac{\mu^n}{n!} e^{-\mu}$$

μ - частота событий в единицу времени.

Вот несколько примеров:



Как видите, его форма похожа на распределение Гаусса, а его пик равен μ.

Распределение Пуассона широко используется в физике элементарных частиц, а в науке о данных может быть полезно, описывать события с фиксированной скоростью (например, покупатель, который входит в супермаркет утром).

- Распределение Пуассона - это дискретное распределение вероятностей.
- Распределение Пуассона — это распределение количества, т.е. количества случаев, когда событие произошло в заданный интервал времени.
- Распределение Пуассона можно использовать для прогнозирования вероятности количества успешных событий, которые могут произойти в определенный интервал времени.
- Пример: если в колл-центр поступило 50 звонков за 1 час, то с помощью распределения Пуассона мы можем предсказать вероятность получения 20 звонков в следующие 30 минут.

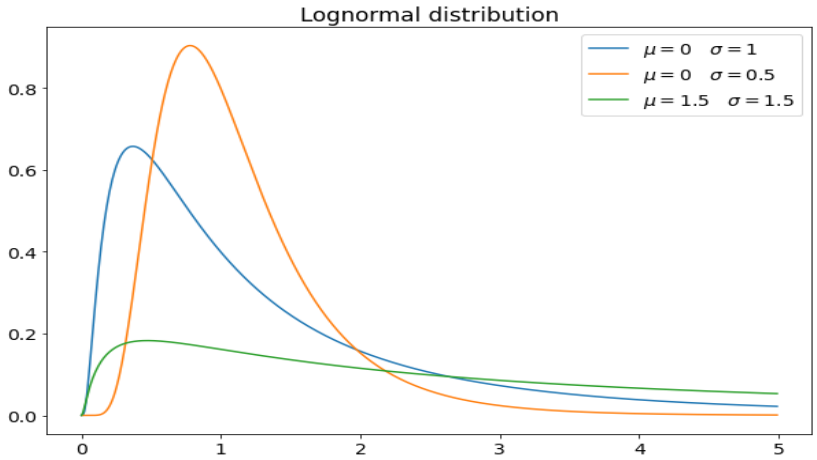
Логнормальное распределение

Если вы возьмете гауссовскую переменную и возведете в степень, вы получите логнормальное распределение, функция плотности вероятности которого:

$$f(x, \mu, \sigma) = \frac{e^{-\frac{(\ln x - \mu)^2}{2\sigma^2}}}{\sqrt{2\pi}\sigma x}$$

μ и σ совпадают с исходным гауссовым распределением.

Несколько примеров:



Логнормальное распределение широко встречается в природе. Артериальное давление следует логнормальному распределению, размеры городов и так далее. Очень интересно использовать геометрическое броуновское движение, которое представляет собой модель случайного блуждания, часто используемую для описания финансовых рынков, особенно в уравнении Блэка-Шоулза для ценообразования.

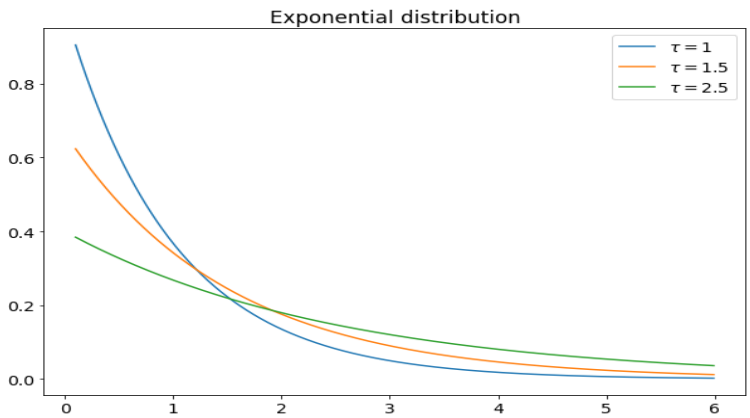
Экспоненциальное распределение

Если у вас есть событие Пуассона, которое происходит с фиксированной скоростью, временной интервал между двумя последовательными появлением этого события распределяется экспоненциально.

Экспоненциальное распределение имеет такую функцию плотности:

$$f(t, \tau) = \frac{1}{\tau} e^{-t/\tau}$$

τ - средний временной интервал между двумя последовательными событиями.



Экспоненциальное распределение используется в физике элементарных частиц и, вообще, если вы хотите перейти от пуассоновского процесса (в котором вы изучаете количество событий) к чему-то более связанному со временем (например, сколько времени проходит между двумя последовательными клиентами, входящими в магазин).

Машинное обучение с учителем

Что такое машинное обучение с учителем?

Случай, когда у нас есть как объекты (матрица X), так и таргеты (вектор y)

Бинараная классификация

Статистическая бинарная классификация. Статистическая классификация - это проблема, изучаемая в машинном обучении. Это тип обучения с учителем, метод машинного обучения, в котором таргеты предопределены, и используется для категоризации новых вероятностных наблюдений по указанным таргетам. Когда есть только два таргета, проблема известна как статистическая бинарная классификация.

Некоторые из методов, обычно используемых для двоичной классификации:

- Деревья решений
- Случайные леса
- Байесовские сети
- Support vector machines (SVM)
- Нейронные сети
- Логистическая регрессия
- Пробит-модель

Каждый классификатор лучше всего подходит только для выбранной области на основе количества наблюдений, размерности вектора признаков, шума в данных и многих других факторов. Например, случайные леса работают лучше, чем классификаторы SVM для трехмерных облаков точек.

Есть много метрик, которые можно использовать для измерения производительности классификатора или предиктора; разные метрики применяются для разных целей. В медицине часто используются чувствительность и специфичность, тогда как при поиске информации предпочтительны точность и охват (presicion и recall). Важное различие заключается между метриками, которые не зависят от того, как часто каждая категория встречается в популяции (распространенность), и метриками, которые зависят от распространенности - оба типа полезны, но имеют очень разные свойства.

Учитывая классификацию конкретного набора данных, существует четыре основных комбинации таргетов фактических данных и предсказанного таргета: истинно положительные TP (правильные положительные присвоения), истинно отрицательные TN (правильные отрицательные присвоения), ложные положительные результаты FP (неправильные положительные назначения) и ложноотрицательные FN (неправильные отрицательные отнесения).

Матрица ошибок

	Состояние положительное	**Состояние отрицательное**
Результат положительный	Истинно положительный (TP)	Ложный положительный (FP)
Результат отрицательный	Ложноотрицательный (FN)	Истинно отрицательный (TN)

- (P) - количество реальных положительных таргетов в данных

- (N) - количество реальных отрицательных таргетов в данных
- (TP) - истинно положительный
- (TN) - истинно отрицательный
- (FP) - ложное срабатывание, ошибка I рода
- (FN) - ложноотрицательный, ошибка II рода

Чувствительность - частота совпадений или истинно положительный показатель (TPR)

Чувствительность измеряет долю правильно идентифицированных положительных результатов (т.е. долю тех, у кого есть какое-либо заболевание, которые правильно идентифицированы как имеющие заболевание).

$$\mathrm{TPR} = \frac{\mathrm{TP}}{\mathrm{P}} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}} = 1 - \mathrm{FNR}$$

Специфичность, селективность, истинно отрицательный показатель (TNR)

Специфичность измеряет долю правильно идентифицированных отрицательных результатов (т. е. долю тех, у кого нет заболевания, которые правильно идентифицированы как не страдающие этим заболеванием).

$$\mathrm{TNR} = \frac{\mathrm{TN}}{\mathrm{N}} = \frac{\mathrm{TN}}{\mathrm{TN} + \mathrm{FP}} = 1 - \mathrm{FPR}$$

Положительная прогностическая ценность (PPV)

$$\mathrm{PPV} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP}} = 1 - \mathrm{FDR}$$

Отрицательная прогностическая ценность (NPV)

$$\mathrm{NPV} = \frac{\mathrm{TN}}{\mathrm{TN} + \mathrm{FN}} = 1 - \mathrm{FOR}$$

Ложноотрицательный показатель (FNR), процент промахов

$$\mathrm{FNR} = \frac{\mathrm{FN}}{\mathrm{P}} = \frac{\mathrm{FN}}{\mathrm{FN} + \mathrm{TP}} = 1 - \mathrm{TPR}$$

Частота выпадений или ложных срабатываний (FPR)

В статистике при выполнении множественных сравнений коэффициент ложноположительных результатов (также известный как коэффициент выпадений или ложных тревог) - это вероятность ложного отклонения нулевой гипотезы для конкретного теста. Частота ложных срабатываний рассчитывается как соотношение между количеством отрицательных событий, ошибочно классифицированных как положительные (ложные срабатывания), и общим количеством фактических отрицательных событий (независимо от классификации).

$$\mathrm{FPR} = \frac{\mathrm{FP}}{\mathrm{N}} = \frac{\mathrm{FP}}{\mathrm{FP} + \mathrm{TN}} = 1 - \mathrm{TNR}$$

Коэффициент ложного обнаружения (FDR)

$$\mathrm{FDR} = \frac{\mathrm{FP}}{\mathrm{FP} + \mathrm{TP}} = 1 - \mathrm{PPV}$$

Коэффициент ложных пропусков (FOR)

$$\mathrm{FOR} = \frac{\mathrm{FN}}{\mathrm{FN} + \mathrm{TN}} = 1 - \mathrm{NPV}$$

Порог распространенности (PT)

$$PT = \frac{\sqrt{TPR(-TNR+1)} + TNR - 1}{(TPR + TNR - 1)}$$

Оценка угрозы (TS) или индекс критического успеха (CSI)

$$\mathrm{TS} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN} + \mathrm{FP}}$$

Точность (ACC)

$$\mathrm{ACC} = \frac{\mathrm{TP} + \mathrm{TN}}{\mathrm{P} + \mathrm{N}} = \frac{\mathrm{TP} + \mathrm{TN}}{\mathrm{TP} + \mathrm{TN} + \mathrm{FP} + \mathrm{FN}}$$

Сбалансированная точность (BA)

$$\mathrm{BA} = \frac{TPR + TNR}{2}$$

Оценка F1 является средним гармоническим из точности и чувствительности

$$\mathrm{F_1} = 2 \cdot \frac{\mathrm{PPV} \cdot \mathrm{TPR}}{\mathrm{PPV} + \mathrm{TPR}} = \frac{2\mathrm{TP}}{2\mathrm{TP} + \mathrm{FP} + \mathrm{FN}}$$

Коэффициент корреляции Мэтьюза (MCC)

$$\mathrm{MCC} = \frac{\mathrm{TP} \times \mathrm{TN} - \mathrm{FP} \times \mathrm{FN}}{\sqrt{(\mathrm{TP} + \mathrm{FP})(\mathrm{TP} + \mathrm{FN})(\mathrm{TN} + \mathrm{FP})(\mathrm{TN} + \mathrm{FN})}}$$

Индекс Фаулкса – Маллоуса (FM)

$$\mathrm{FM} = \sqrt{\frac{TP}{TP + FP} \cdot \frac{TP}{TP + FN}} = \sqrt{PPV \cdot TPR}$$

Информированность или информированность(BM)

$$\mathrm{BM} = \mathrm{TPR} + \mathrm{TNR} - 1$$

Маркировка (МК) или дельтаР (Δр)

$$\mathrm{BM} = \mathrm{TPR} + \mathrm{TNR} - 1$$

Регрессия

? Что такое регрессия? Какие модели можно использовать для решения задачи регрессии?

Регрессия — это часть машинного обучения с учителем. Модели регрессии исследуют взаимосвязь между зависимой (целевой) и независимой (-ыми) переменной (-ями) (предиктором). Вот несколько распространенных регрессионных моделей:

- Линейная регрессия устанавливает линейную зависимость между целью и предиктором. Он предсказывает числовое значение и имеет форму прямой линии.
- Полиномиальная регрессия имеет уравнение регрессии со степенью независимой переменной больше единицы. Это кривая, которая вписывается в точки данных.
- Гребневаярегрессия помогает, когда предикторы сильно коррелированы (проблема мультиколлинеарности). Он штрафует квадраты коэффициентов регрессии, но не позволяет коэффициентам достигать нулей (использует регуляризацию L2).
- Регрессия лассо штрафует абсолютные значения коэффициентов регрессии и позволяет некоторым коэффициентам достигать абсолютного нуля (тем самым позволяя выбирать признаки).

Линейная регрессия

Основная цель линейной регрессии - предсказать некоторую целевую переменную y с помощью некоторой переменной-предиктора X.

Очень простой пример: прогнозирование роста (y) человека на основе его веса (X). Очевидно, что более высокие люди будут весить больше. В этом случае линейная регрессия может использоваться для прогнозирования среднего роста при каждом заданном весе.

? Что такое линейная регрессия? Когда мы ее используем?

Линейная регрессия-это модель, которая предполагает линейную зависимость между входными переменными (X) и единственной выходной переменной (y).

С помощью простого уравнения: **y = B0 + B1*x1 + ... + Bn * xN**

B-коэффициенты регрессии, значения x-независимые переменные, а y-зависимая переменная.

Случай с одной независимой переменной называется простой линейной регрессией. Для более чем одной независимой переменной этот процесс называется множественной линейной регрессией.

Простая линейная регрессия: **y = B0 + B1*x1**

Множественная линейная регрессия: **y = B0 + B1*x1 + ... + Bn * xN**

? Каковы основные предположения линейной регрессии?

Существует несколько предположений линейной регрессии. Если какой-либо из них нарушается, предсказания и интерпретация модели могут оказаться бесполезными или вводящими в заблуждение

1. Линейная связь между объектами и целевой переменной.
2. Аддитивность означает, что влияние изменений одного из признаков на целевую переменную не зависит от значений других признаков. Например, модель для прогнозирования выручки компании имеет два признака-количество проданных товаров a и количество проданных товаров b. Когда компания продает больше товаров a, выручка увеличивается, и это не зависит от количества проданных товаров b. Но, если клиенты, которые покупают a, перестают покупать b, предположение аддитивности нарушается.
3. Признаки не коррелируются (нет коллинеарности), поскольку бывает трудно отделить отдельные эффекты коллинеарных признаков от целевой переменной.
4. Ошибки независимо и одинаково нормально распределены (**y = B0 + B1*x1 + ... + ошибка**):
    1. Отсутствие корреляции между;
    2. Постоянная дисперсия ошибок - гомоскедастичность. Например, в случае временных рядов сезонные паттерны могут увеличивать ошибки в сезонах с более высокой активностью;
    3. Ошибки обычно нормально распределены, иначе некоторые признаки будут иметь большее влияние на целевую переменную, чем на другие. Если распределение ошибок существенно отличается от нормального, доверительные интервалы могут быть слишком широкими или слишком узкими.

Полиномиальная регрессия

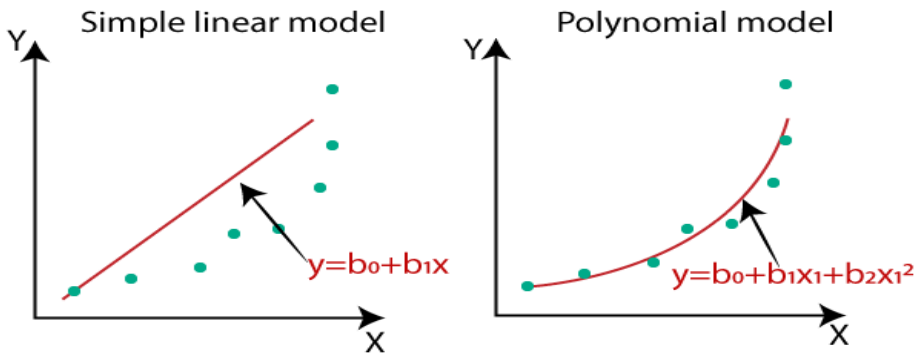
В статистике, полиномиальная регрессия — это форма регрессионного анализа, в которой взаимосвязь между независимой переменной x и зависимой переменной y моделируются как полином n-й степени от x. Полиномиальная регрессия соответствует нелинейной зависимости между значением x и соответствующим условным средним y, обозначенным E(y|x). Хотя полиномиальная регрессия соответствует нелинейной модели данных, как проблема статистической оценки она является линейной в том смысле, что функция регрессии E (y|x) линейна относительно неизвестных параметров, которые оцениваются на основе данных. По этой причине полиномиальная регрессия считается частным случаем множественной линейной регрессии.

? Что такое полиномиальная регрессия?

- Полиномиальная регрессия - это форма регрессионного анализа, в которой взаимосвязь между независимыми и зависимыми переменными моделируется полиномом n-й степени.

- Модели полиномиальной регрессии обычно подбираются с использованием метода наименьших квадратов. Метод наименьших квадратов минимизирует дисперсию из несмещенных оценок коэффициентов при условиях теоремы Гаусса-Маркова.

- Полиномиальная регрессия — это частный случай линейной регрессии, когда мы подбираем полиномиальное уравнение к данным с криволинейной связью между зависимыми и независимыми переменными.



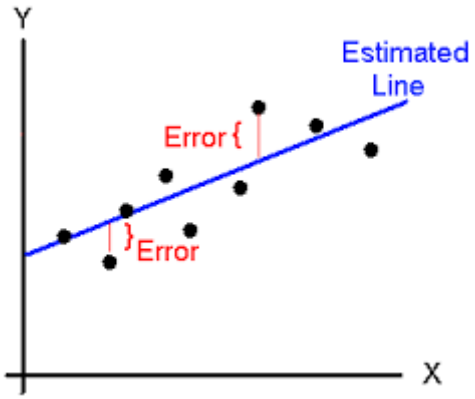
- Полиномиальная регрессия не требует, чтобы отношения между независимыми и зависимыми переменными были линейными в наборе данных. Это также одно из основных различий между линейной и полиномиальной регрессией.

- Полиномиальная регрессия обычно используется, когда точки в данных не захватываются моделью линейной регрессии и линейная регрессия не может четко описать лучший результат.

По мере того, как мы увеличиваем степень в модели, она имеет тенденцию к увеличению производительности модели. Однако увеличение степеней модели также увеличивает риск переобучения или недооценки данных.

Гребневая (ридж)-регрессия (метод регуляризации L2)

Регуляризация — это метод, который помогает преодолеть проблему переобучения в моделях машинного обучения. Она называется регуляризацией, так как помогает поддерживать регулярные или нормальные параметры. Распространенными методами являются регуляризация L1 и L2, широко известные как регрессия лассо и гребневая(ридж) регрессия.



Функция ошибок рассчитывается на основе обучающего набора данных. Когда модель слишком близко подходит к обучающим данным, это называется переобучением. В этом случае модель очень хорошо работает с данными обучения, но очень плохо с данными тестирования. Чтобы нейтрализовать или оптимизировать эту ошибку, выполняется регуляризация, что помогает поддерживать параметр в постоянном или нормальном состоянии.

? Зачем нужна эта регуляризация?

Любое новое большое значение (значение выброса) считается плохим признаком. Новая точка данных определенно немного изменит коэффициенты модели и, таким образом, окажет огромное влияние на модель.

$$= \sum_{k=0}^{n} y - (b + a_k x_k)^2 + \sum_{i=0}^{n} \lambda |a_i|^p$$

где p = 1,2 в зависимости от того, какая регрессия.

p = 1 для Лассо, помогает в выборе признаков за счет уменьшения количества столбцов, когда они не важны. У него может быть несколько решений.

p = 2 для гребневой, он представляет только одно решение и до некоторой степени сходится к проблеме.

К функции потерь добавляется член регуляризации, чтобы предотвратить переобучения и повысить точность прогнозирования.

Здесь мы подробно рассмотрим только гребневую-регрессию.

? Как узнать, когда нужна регуляризация?

Модель первоначально находит наиболее подходящее решение с помощью «Тренировочных данных», но, когда те же коэффициенты используются для предсказания данных тестирования, модель не может предсказывать с хорошей точностью, поскольку решение изменяется с новыми значениями данных.

Сумма квадратов ошибки будет разной для данных обучения и тестирования. Итак, мы применяем несколько методов коррекции или методов регуляризации, чтобы компенсировать функцию потерь. Этот параметр известен как штрафная функция (Penalizing function).

Необходимо использовать Гребневую Регрессию (Ridge Regression), чтобы найти новую модель, которая не соответствует обучающим данным. Другими словами, мы вносим небольшую погрешность в то, как новая строка соответствует данным, и взамен получаем значительное снижение дисперсии. Если начать с немного худшего совпадения, Гребневая Регрессия может обеспечить лучшие долгосрочные прогнозы.

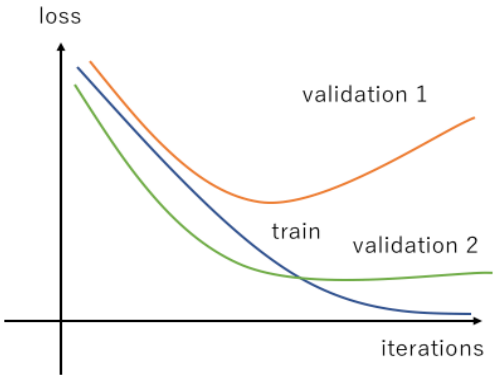
? Почему недостаточно наименьшего квадрата?

При прогнозировании с помощью «наименьших квадратов» y = ax + b, так минимизируется сумма квадратов остатка. Без малейшей погрешности метод обыкновенных наименьших квадратов имеет больший разброс.

При прогнозировании с использованием гребневой регрессии**, y =ax + b + lambda (slope)2**.

Этот дополнительный термин известен как штраф, и лямбда определяет, насколько серьезным будет наказание. Таким образом, мы предпочли бы метод гребеневой регрессии обыкновенным наименьшим квадратам.

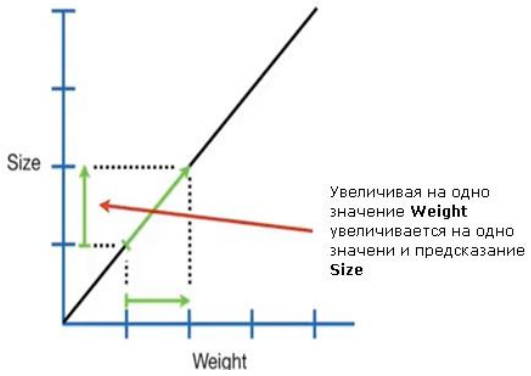
Давайте посмотрим на это на примере ...



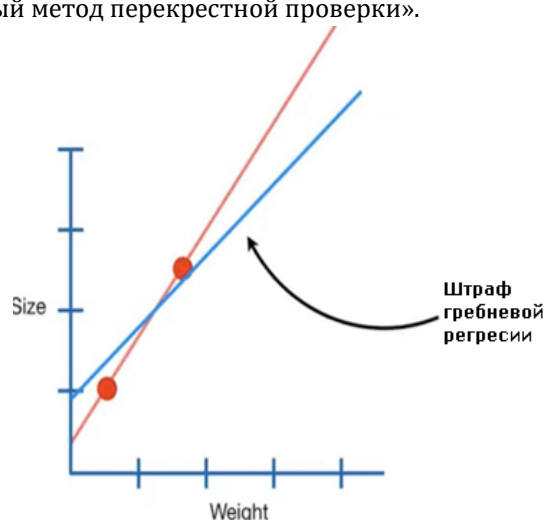
Loss и итерационная кривая

Наклон линии определяет изменение значений «y» или прогноз для каждого единичного изменения значений «x». Если наклон крутой, на каждую единицу изменения x, y изменяется более чем на 1 единицу. В таком случае прогноз для «y»

Если наклон мал, для каждого изменения единицы x, y практически не изменяется. В таких случаях прогнозы менее чувствительны к изменению x.



Когда лямбда = 0, гребневая регрессия аналогична методу суммы наименьших квадратов. По мере увеличения значения лямбда уравнение становится менее чувствительным к значениям x. Чтобы найти оптимальные значения лямбда, которые приводят к наименьшей дисперсии, мы используем «10-кратный метод перекрестной проверки».



Гребневая регрессия также работает, когда у нас есть дискретные переменные, такие как высокое содержание жира, низкое содержание жира и т. д.

Таким образом, в целом гребневая регрессия помогает уменьшить дисперсию за счет уменьшения параметров и уменьшения чувствительности наших прогнозов к ним.

Подробности регрессии LASSO (L1-регуляризация)

Это метод регуляризации, который создает модели при наличии больших моделей при наличии большого количества признаков, что подразумевает:

1. Достаточно большой, чтобы вызвать вычислительные проблемы.

2. Достаточно большой, чтобы усилить тенденцию к переобучению.

Лассо и Гребеневая регрессия очень похожи друг на друга, но в то же время у них есть ключевое различие.

В формуле суммы квадратов остатков,

$$= \sum_{k=0}^{n} (y - (b + ax_i))^2 + \sum_{i=0}^{n} \lambda |a_i|$$

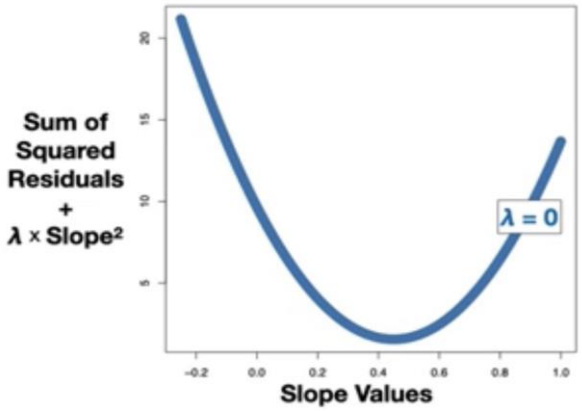
Или, другими словами, ***y =ax + b + lambda (slope)*2.***

Когда значение лямбда равно нулю, сумма квадратов остатков становится обыкновенным методом наименьших квадратов. По мере увеличения значения лямбда параметры в значительной степени уменьшаются, а также иногда уменьшаются до нуля.

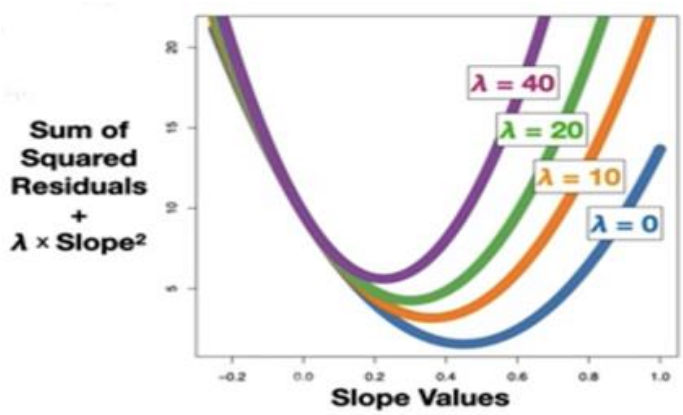
Визуализация Гребеневой регресии и Лассо:

В Гребневой Регрессии:

На графике значений slope и график кривой для различных значений лямбда. Первоначально, когда лямбда равна нулю, slope достигает минимального значения, скажем, 0,45.

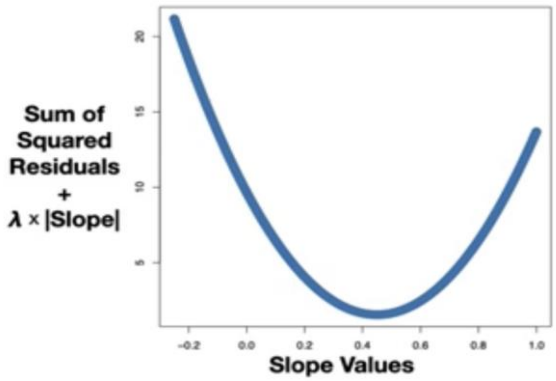


При дальнейшем увеличении значений лямбда slope сходится раньше, и минимальное значение также достигается раньше, чем предыдущее значение. В результате получается оптимальный slope. Даже если мы очень сильно увеличим значение лямбда, мы все равно получим оптимальный slope (> 0).

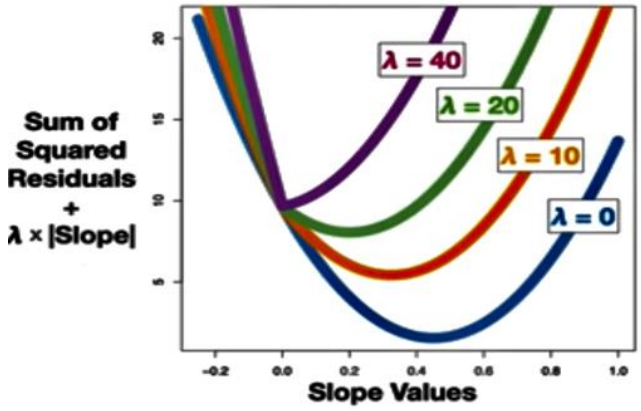


В LASSO Регрессии:

Аналогично построению для штрафа Лассо, мы берем абсолютное значение slope. В регрессии лассо изначально, когда лямбда = 0, мы получаем slope.

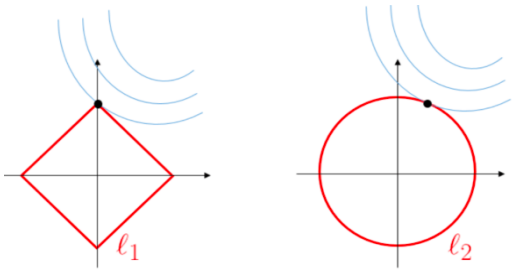


По мере увеличения значения лямбда slope очень быстро сходится, и при получении минимального значения наблюдается резкое изменение slope. Эта функция позволяет модели уменьшить некоторые нежелательные функции до нуля, тем самым помогая снизить сложность модели.



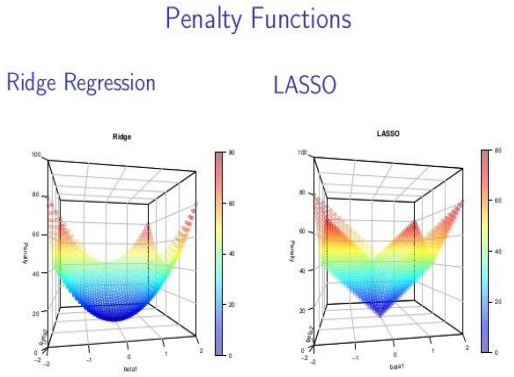
На приведенном ниже рисунке показаны подробности, объясняющие, что происходит в обоих методах регуляризации.

В регрессии Лассо (L1) значения наклона совпадают с x = 0, тогда как в Гребневой регрессии (L2) оно приближается только к нулю.



Какую использовать и когда?

Чтобы ответить на этот вопрос, нам нужно знать размер данных, которые мы, возможно, захотим обработать и спрогнозировать. Если размер данных меньше, то есть меньше признаков, мы можем использовать Гребневую регрессию.



Но когда данные содержат много признаков, скажем несколько миллионов, тогда в этом случае сложность модели возрастает, и, следовательно, использование Лассо Регрессии уменьшит нежелательные признаки и снизит сложность модели.

Какие методы решения линейной регрессии вы знаете?

Чтобы решить линейную регрессию, вам нужно найти коэффициенты бета, которые минимизируют сумму квадратов ошибок.

Метод матричных алгебр: допустим, у вас есть Х-матрица признаков и У- вектор со значениями, которые вы хотите предсказать. Пройдя через матричную алгебру и задачу минимизации, вы получите следующее решение:

$$\beta = (X^T X)^{-1} X^T y$$

Но для решения этой задачи требуется найти обратную величину, что может занять много времени, если вообще возможно. К счастью, существуют такие методы, как сингулярная декомпозиция (SVD) или QR-декомпозиция, которые могут надежно вычислить эту часть (называемую псевдоинверсной) без фактической необходимости находить обратную. Популярная библиотека python ML sklearn использует SVD для решения задач наименьших квадратов.

$$(X^T X)^{-1} X^T$$

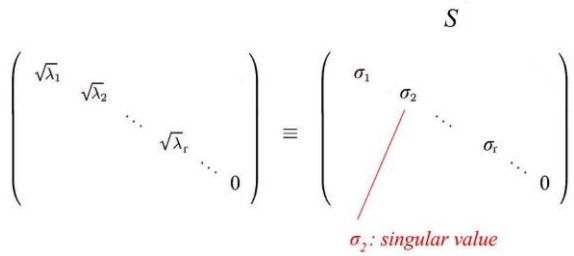
Альтернативный метод: градиентный спуск.

Сингулярное разложение (SVD)

Начнем с самого сложного. SVD утверждает, что любую матрицу A можно факторизовать как:

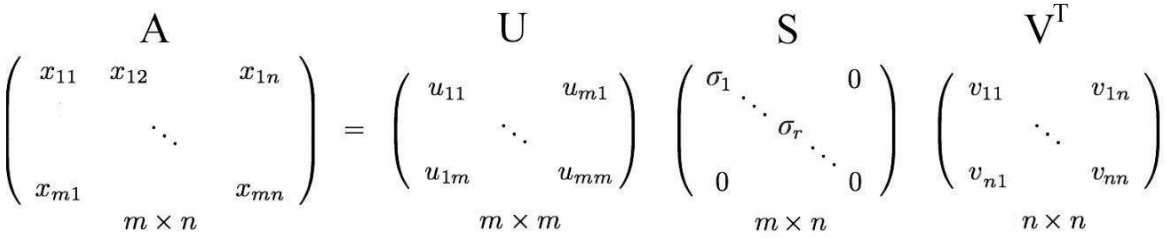
$$A = U S V^T$$

где U и V являются ортогональными матрицами с ортонормированными собственными векторами, выбранных из AA^T и A^TA соответственно. S - диагональная матрица с r элементами, равными корню положительных собственных значений матрицы AA^T или A^T A (в любом случае обе матрицы имеют одинаковые положительные собственные значения). Диагональные элементы состоят из сингулярных значений.

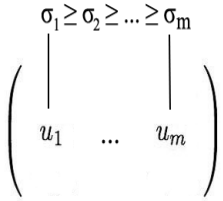


т.е. матрицу размера m × n можно факторизовать как:

$$A_{m\times n} = U_{m\times m} S_{m\times n} V^T_{n\times n}$$



Мы можем организовать собственные векторы в различных порядках для производства U и V. Чтобы стандартизировать решение, мы упорядочиваем собственные векторы так, чтобы векторы с более высокими собственными значениями предшествовали векторам с меньшими значениями.



По сравнению с собственным разложением SVD работает с неквадратными матрицами. U и V обратимы для любой матрицы в SVD, и они ортонормированы, что нам нравится. Без доказательства здесь мы также говорим вам, что особые значения численно более устойчивы, чем собственные значения.

Пример

Прежде чем заходить слишком далеко, давайте продемонстрируем это на простом примере.

$$A = \begin{pmatrix} 3 & 2 & 2 \\ 2 & 3 & -2 \end{pmatrix}$$

Рассчитываем:

$$AA^T = \begin{pmatrix} 17 & 8 \\ 8 & 17 \end{pmatrix}, \qquad A^TA = \begin{pmatrix} 13 & 12 & 2 \\ 12 & 13 & -2 \\ 2 & -2 & 8 \end{pmatrix}$$

Эти матрицы являются как минимум положительно полуопределенными (все собственные значения положительны или равны нулю). Как показано, они имеют одинаковые положительные собственные значения (25 и 9). На рисунке ниже также показаны соответствующие им собственные векторы.

$$AA^T = \begin{pmatrix} 17 & 8 \\ 8 & 17 \end{pmatrix} \qquad A^TA = \begin{pmatrix} 13 & 12 & 2 \\ 12 & 13 & -2 \\ 2 & -2 & 8 \end{pmatrix}$$

eigenvalues: $\lambda_1 = 25, \lambda_2 = 9$ eigenvalues: $\lambda_1 = 25, \lambda_2 = 9, \lambda_3 = 0$

eigenvectors eigenvectors

$$u_1 = \begin{pmatrix} 1/\sqrt{2} \\ 1/\sqrt{2} \end{pmatrix} \quad u_2 = \begin{pmatrix} 1/\sqrt{2} \\ -1/\sqrt{2} \end{pmatrix} \qquad v_1 = \begin{pmatrix} 1/\sqrt{2} \\ 1/\sqrt{2} \\ 0 \end{pmatrix} \quad v_2 = \begin{pmatrix} 1/\sqrt{18} \\ -1/\sqrt{18} \\ 4/\sqrt{18} \end{pmatrix} \quad v_3 = \begin{pmatrix} 2/3 \\ -2/3 \\ -1/3 \end{pmatrix}$$

Сингулярные значения — это квадратный корень из положительных собственных значений, т.е. 5 и 3. Следовательно, композиция SVD:

$$A = USV^T = \begin{pmatrix} 1/\sqrt{2} & 1/\sqrt{2} \\ 1/\sqrt{2} & -1/\sqrt{2} \end{pmatrix} \begin{pmatrix} 5 & 0 & 0 \\ 0 & 3 & 0 \end{pmatrix} \begin{pmatrix} 1/\sqrt{2} & 1/\sqrt{2} & 0 \\ 1/\sqrt{18} & -1/\sqrt{18} & 4/\sqrt{18} \\ 2/3 & -2/3 & -1/3 \end{pmatrix}$$

Логистическая регрессия

Логистическая регрессия — это статистический метод прогнозирования бинарных классов. Переменная результата или цели имеет дихотомический характер, что означает, что существует только два возможных класса. Например, его можно использовать для задач обнаружения рака. Он вычисляет вероятность возникновения события.

Это особый случай линейной регрессии, когда целевая переменная является категориальной по своей природе. Он использует журнал шансов в качестве зависимой переменной. Логистическая регрессия предсказывает вероятность возникновения двоичного события с использованием функции журнала.

Уравнение линейной регрессии:

$$y = \beta 0 + \beta 1X1 + \beta 2X2 + \ldots\ldots + \beta nXn$$

Где y - зависимая переменная, а X1, X2… и Xn - независимые переменные.

Сигмоидная функция:

$$p = 1/1 + e^{-y}$$

Применение сигмовидной функции к линейной регрессии:

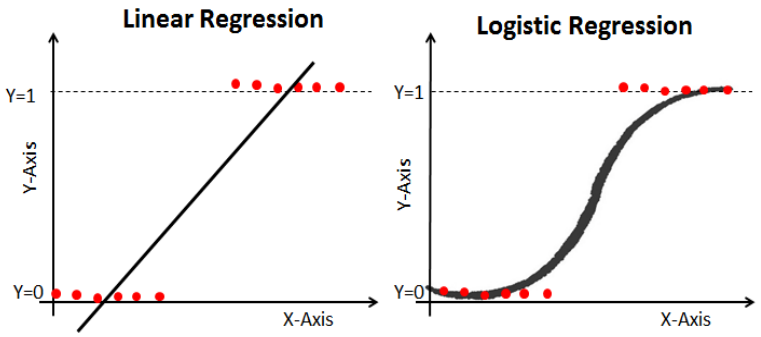
$$p = 1/1 + e^{-(\beta 0 + \beta 1 X 1 + \beta 2 X 2 ..... \beta n X n)}$$

Свойства логистической регрессии

- Зависимая переменная в логистической регрессии соответствует распределению Бернулли.
- Оценка производится по максимальному правдоподобию.
- Нет R square, соответствие модели рассчитывается через Concordance, KS-Statistics.

Линейная регрессия Vs. логистическая регрессия

Линейная регрессия дает вам непрерывный результат, а логистическая регрессия обеспечивает постоянный результат. Примером непрерывного результата является цена дома или цена акций. Примеры дискретных выходных данных — это прогнозирование наличия у пациента рака. Линейная регрессия оценивается с использованием обыкновенных наименьших квадратов (OLS), а логистическая регрессия оценивается с использованием подхода оценки максимального правдоподобия (MLE).



Оценка максимального правдоподобия Vs. Метод наименьших квадратов (Maximum Likelihood Estimation Vs. Least Square Method)

MLE - это метод максимизации «правдоподобия», а OLS - метод аппроксимации с минимизацией расстояния. Максимизация функции правдоподобия определяет параметры, которые с наибольшей вероятностью дадут наблюдаемые данные. Со статистической точки зрения, MLE устанавливает среднее значение и дисперсию, поскольку параметры при определении конкретного параметра могут использоваться для прогнозирования данных, необходимых для нормального распределения.

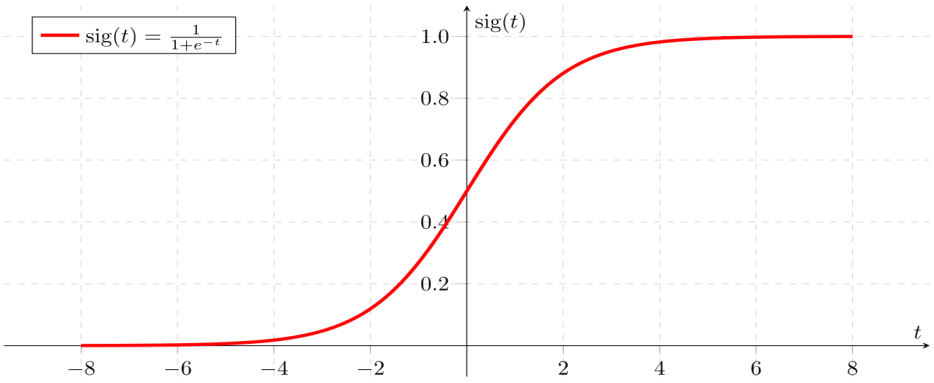
Обычные оценки методом наименьших квадратов вычисляются путем подбора линии регрессии по заданным точкам данных, которые имеют минимальную сумму квадратов преданности (наименьшая квадратичная ошибка). Оба используются для оценки параметров модели линейной регрессии. MLE предполагает совместную функцию массы вероятностей, в то время как OLS не требует каких-либо стохастических предположений для минимизации расстояния.

Сигмоида

Сигмоидальная функция, также называемая логистической функцией, дает S-образную кривую, которая может принимать любое действительное число и отображать его в значение от 0 до 1. Если кривая переходит в положительную бесконечность, прогнозируемый у станет 1, а если кривая уходит в отрицательную бесконечность, прогнозируемый у станет 0. Если выход

сигмовидной функции больше 0,5, мы можем классифицировать результат как 1 или ДА, а если он меньше 0,5, мы можем классифицировать его как 0 или НЕТ. Выход не может. Например: если выход 0,75, мы можем сказать с точки зрения вероятности следующим образом: Вероятность того, что пациент будет болеть раком, составляет 75 процентов.

$$f(x) = \frac{1}{1 + e^{-(x)}}$$



Типы логистической регрессии:

- Порядковая логистическая регрессия: целевая переменная имеет три или более порядковых категорий, например рейтинг ресторана или продукта от 1 до 5.
- Бинарная логистическая регрессия: целевая переменная имеет только два возможных результата, таких как Спам или не спам, Рак или, нет рака.
- Полиномиальная логистическая регрессия: целевая переменная имеет три или более номинальных категорий, таких как предсказание типа вина.

Градиент

Вывод формулы для алгоритма градиентного спуска:
Основная идея метода заключается в том, чтобы осуществлять оптимизацию в направлении наискорейшего спуска, а это направление задаётся антиградиентом $-\nabla f$:

$$x^{[k+1]} = x^{[k]} - \lambda^{[k]} \nabla f(x^{[k]})$$

где $\lambda^{[k]}$ выбирается

- постоянной, в этом случае метод может расходиться;
- дробным шагом, т.е. длина шага в процессе спуска делится на некое число;
- наискорейшим спуском: $\lambda^{[k]} = \arg\min_{\lambda} f(x^{[k]} - \lambda \nabla f(x^{[k]}))$

Алгоритм

Вход: функция $f: \mathbb{R}^n \to \mathbb{R}$

Выход: найденная точка оптимума x

1. Повторять:

2. $x^{[k+1]} = x^{[k]} - \lambda^{[k]} \nabla f(x^{[k]})$, где $\lambda^{[k]}$ выбирается одним из описанных выше способов
3. если выполен критерий останова, то возвращаем текущее значение $x^{[k+1]}$

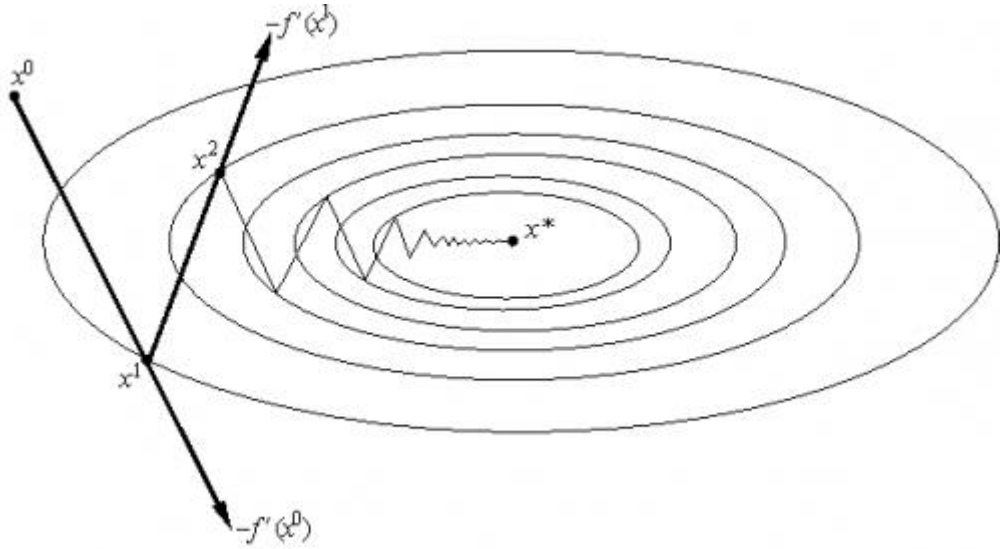
Критерий останова

Критерии остановки процесса приближенного нахождения минимума могут быть основаны на различных соображениях. Некоторые из них:

1. $\|x^{[k+1]} - x^{[k]}\| \leq \epsilon$
2. $\|f(x^{[k+1]}) - f(x^{[k]})\| \leq \epsilon$

Здесь $x^{[k]} \in \mathbb{R}^n$ - значение, полученное после k -го шага оптимизации. ϵ - наперед заданное положительное число.

Геометрическая интерпретация метода градиентного спуска с постоянным шагом. На каждом шаге мы сдвигаемся по вектору антиградиента, "уменьшенному в λ раз":



Что такое градиентный спуск? Как он работает?

Градиентный спуск — это алгоритм, который использует расчетную концепцию градиента, чтобы попытаться достичь локальных или глобальных минимумов. Он работает, принимая отрицательное значение градиента в точке заданной функции и многократно обновляя эту точку с использованием вычисленного отрицательного градиента, пока алгоритм не достигнет локального или глобального минимума, что приведет к тому, что будущие итерации алгоритма будут возвращать значения, которые равны или слишком близки к текущей точке. Он широко используется в приложениях машинного обучения.

Что такое СГД - стохастический градиентный спуск? В чем разница между градиентным спуском и стохастическим градиентным спуском?

Как в градиентном спуске (GD), так и в стохастическом градиентном спуске (SGD) вы обновляете набор параметров итеративным образом, чтобы минимизировать функцию ошибок.

Находясь в GD, вы должны пройти ВСЕ записи в вашем обучающем наборе, чтобы выполнить одно обновление для параметра в определенной итерации, а в SGD, с другой стороны, вы используете ТОЛЬКО ОДНУ или ВСПОМОГАТЕЛЬНУЮ обучающую запись из своего обучающего набора сделать обновление для параметра в конкретной итерации. Если вы используете SUBSET, он называется Minibatch Stochastic Gracent Descent.

Что такое нормальное уравнение?

Нормальные уравнения — это уравнения, полученные путем приравнивания к нулю частных производных суммы квадратов ошибок (наименьших квадратов); нормальные уравнения позволяют оценить параметры множественной линейной регрессии.

Градиентный спуск

Градиентный спуск является итеративным алгоритмом оптимизации для нахождения локального минимума в виде дифференцируемой функции. Идея состоит в том, чтобы сделать повторяющиеся шаги в направлении, противоположном градиенту (или приблизительному градиенту) функции в текущей точке, потому что это направление наиболее крутого спуска. И наоборот, переход в направлении градиента приведет к локальному максимуму этой функции; процедура тогда известна как подъем по градиенту.

Шаги для реализации градиентного спуска

1. Произвольная инициализация значений.
2. Обновить значения.

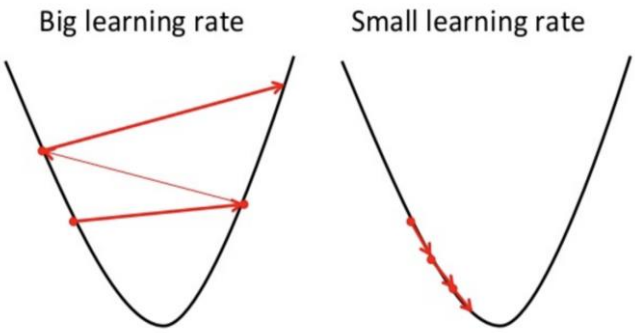
$$weight^{(new)} = weight^{(old)} - constant \frac{\partial J(\Theta)}{\partial weight}$$

3. Повторяйте до тех пор, пока наклон не будет равен 0

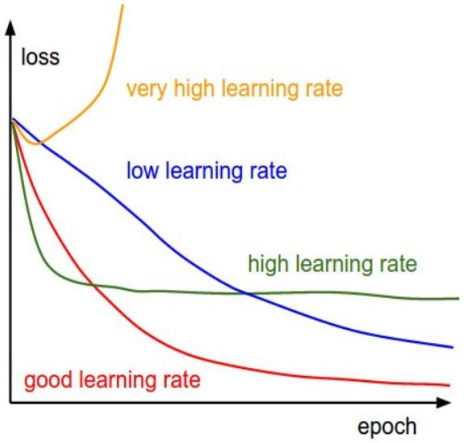
Производная — это значение, полученное из расчетов и рассчитанный как наклон графика в определенной точке. Наклон описывается проведением касательной к графику в точке. Итак, если мы сможем вычислить эту касательную линию, мы сможем вычислить желаемое направление для достижения минимумов.

Шаг обучения (learning rate) должен быть выбран с умом:

1. Если она слишком мала, то модели потребуется некоторое время для обучения.
2. Если он слишком большой, мы можем не достичь минимума проскочив его.



Большой шаг обучения по сравнению с небольшим шагом обучения

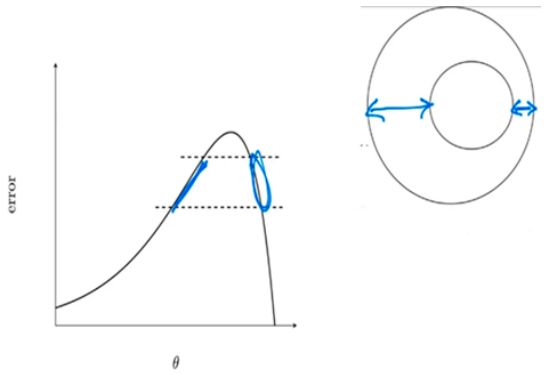


Gradient Descent с разными шагами обучения

Vanilla Gradient descent, однако, не может гарантировать хорошую сходимость по следующим причинам:

- Выбор подходящей скорости обучения может быть проблематичным. Слишком низкая скорость обучения приведет к медленному обучению, а более высокая скорость обучения не приведет нас к глобальному минимуму.

- Еще одно ключевое препятствие, с которым сталкивается Vanilla Gradient Descent, — это избегать попадания в локальные минимумы; эти локальные минимумы окружены холмами одной и той же ошибки, из-за чего Gradient Descent действительно трудно избежать этого.



Контурные карты, визуализирующие пологие и крутые участки кривой

Проще говоря, каждый шаг, который мы делаем в направлении минимумов, имеет тенденцию уменьшать наш наклон, теперь, если мы визуализируем, в крутой области производной кривой будет много, поэтому шаги, предпринятые нашей моделью тоже будут большими, но по мере того, как мы войдем в пологую область наклона, наша производная будет уменьшаться, как и время достижения минимумов.

Градиентный спуск на основе импульса(momentum)

Если мы примем во внимание, что простой градиентный спуск полностью полагается только на вычисления, то есть если есть 10000 шагов, то наша модель попытается реализовать простой градиентный спуск 10000 раз, что, очевидно, потребует слишком много времени и вычислительных ресурсов.

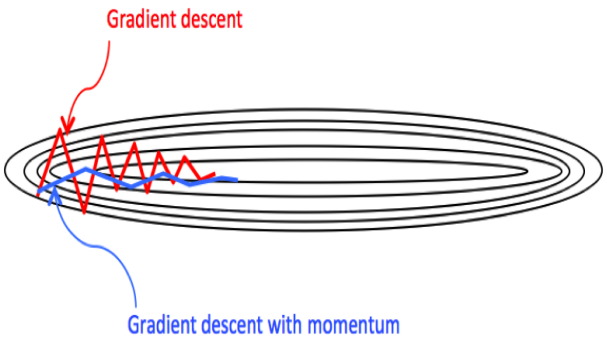
Если говорить простым языком, предположим, что человек идет к своему дому, но он не знает дороги, поэтому он спрашивает направление у прохожего, теперь мы ожидаем, что он пройдет некоторое расстояние, а затем спросит направление, но человек спрашивает направление на каждом шаге, который он делает, что, очевидно, отнимает больше времени, теперь сравнивайте человека с простым градиентным спуском и его цель с минимумами.

Чтобы избежать недостатков ванильного градиентного спуска, ввели градиентный спуск на основе импульса, цель которого состоит в том, чтобы сократить время вычислений, и это может быть достигнуто, когда мы вводим понятие опыта, то есть уверенности, используя предыдущие шаги.

Псевдокод для градиентного спуска на основе импульса:

- обновление = скорость обучения * градиент
- скорость = предыдущее_обновление * импульс
- параметр = параметр + скорость - обновление

Даже в пологой области градиентный спуск на основе импульса делает большие шаги из-за импульса, который он несет.



Но из-за больших шагов он выходит за пределы своей цели на большее расстояние, поскольку он колеблется вокруг минимумов из-за крутого наклона, но, несмотря на такие препятствия, он быстрее, чем vanilla Gradient Descent.

Проще говоря, предположим, что человек хочет достичь пункта назначения, который находится на расстоянии 1200 метров, и он не знает пути, поэтому он решил, что через каждые 250 метров он будет спрашивать направление, а теперь, если он спрашивает направление 5 раз, он проехал

1250 метров, то есть он уже прошел свою цель, и для достижения этой цели ему нужно будет проследить свои шаги назад. Аналогичным образом обстоит дело с GD на основе Momentum, где из-за большого опыта наша модель делает большие шаги, что приводит к превышению и, следовательно, пропуску цели, но для достижения минимума модель должна отслеживать свои шаги.

Ускоренный градиент Нестерова - Nesterov Accelerated Gradient Descent (NAG)

Чтобы преодолеть проблемы градиентного спуска на основе импульса, мы используем NAG, при этом мы сначала движемся, а затем вычисляем градиент, так что если наши колебания выходят за пределы, то он должен быть незначительным по сравнению с Momentum на основе градиентного спуска.

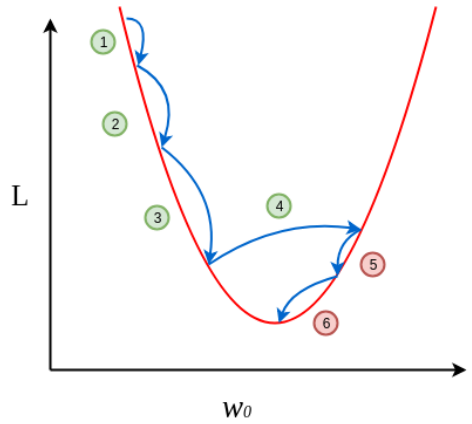
With momentum:

$$update_t = \gamma \cdot update_{t-1} + \eta \nabla w_t$$
$$w_{t+1} = w_t - update_t$$

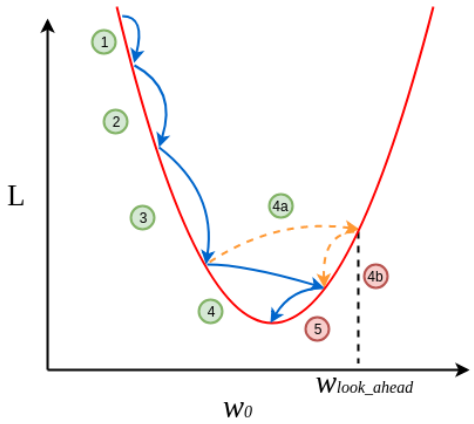
With NAG:

$$w_{look\_ahead} = w_t - \gamma \cdot update_{t-1}$$
$$update_t = \gamma \cdot update_{t-1} + \eta \nabla w_{look\_ahead}$$
$$w_{t+1} = w_t - update_t$$

We follow similar update rule for b_t.



(a) Momentum-Based Gradient Descent



(b) Nesterov Accelerated Gradient Descent

○ $\Longrightarrow \frac{\partial L}{\partial w_0} = \frac{Negative(-)}{Positive(+)}$ ○ $\Longrightarrow \frac{\partial L}{\partial w_0} = \frac{Negative(-)}{Negative(-)}$

Стохастический градиентный спуск

Обучение происходит на каждом примере:

$$w = w - \alpha \nabla_w J(x^i, y^i; w)$$

- Перемешайте набор обучающих данных, чтобы избежать уже существующего порядка данных
- Разделите обучающий набор данных на m примеров

Преимущества:

- Легко помещается в память
- Вычислительная скорость
- Эффективен для большого набора данных

Недостатки:

- Из-за частых обновлений шаги к минимуму очень шумны
- Частые обновления требуют больших вычислительных ресурсов

Пакетный (Batch) градиентный спуск

Это подход, когда мы должны суммировать все примеры для каждого обновления.

$$w = w - \alpha \nabla_w J(w)$$

Преимущества:

- Менее шумные шаги
- Обеспечивает стабильную сходимость GD
- Вычислительная эффективность, поскольку все ресурсы используются не для одной выборки, а для всех обучающих выборок

Недостатки:

- Может потребоваться дополнительная память
- Обработка большой базы данных может занять много времени
- Приблизительные градиенты

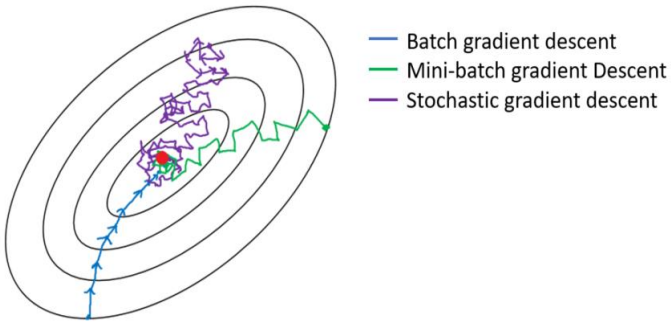
Мини-пакетный градиентный спуск

Вместо того чтобы перебирать все примеры, мини-пакетный градиентный спуск суммирует меньшее количество примеров в зависимости от размера партии.

$$w = w - \alpha \nabla_w J(x^{\{i:i+b\}}, y^{\{i:i+b\}}; w)$$

Преимущества:

- Легко помещается в памяти
- Вычислительная эффективность
- Стабильная погрешность и сходимость



Batch v/s Stochastic v/s Mini Batch Gradient Descent

Что, если мы подадим разряженные данные к градиентному спуску?

В случае разреженных данных мы можем столкнуться с редкими функциями ON(1) и более частыми функциями OFF(0), теперь большая часть обновления градиента будет NULL как производная в большинстве случаев равна нулю, а когда она будет равна единице, шаги будут слишком малы для достижения минимумов.

Для часто используемых функций требуется низкая скорость обучения, но для высоких функций требуется высокая скорость обучения.

Итак, чтобы улучшить нашу модель для разреженных данных о природе, нам нужно выбрать адаптивную скорость обучения.

Функция потерь

Какие метрики для оценки регрессионных моделей вы знаете?

1. Средняя квадратическая ошибка (MSE)
2. Среднеквадратическая ошибка (RMSE)
3. Средняя абсолютная ошибка (MAE)
4. R в квадрате (R^2) или коэффициент детерминации
5. Скорректированный R квадрат (R^2)

Функция потерь

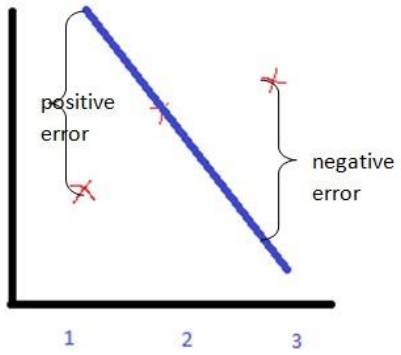
Сумма ошибок (SE)

Начнем с рассмотрения самой простой функции потерь, которая представляет собой не что иное, как сумму ошибок на каждой итерации. Ошибка будет разницей между прогнозируемым значением и фактическим значением. Таким образом, функция потерь будет представлена как:

$$\mathbf{L} = \Sigma(\hat{Y} - Y)$$

$\hat{Y}$ - прогнозируемое значение; Y - фактическое значение

где суммирование идет от n = 1 до N, где N - количество наблюдений в нашем наборе данных. Теперь рассмотрим следующую линию, соответствующую нашим 3 точкам данных:



Конечно, вы могли бы сказать, что это не самая подходящая линия! Но в соответствии с этой функцией потерь эта линия является наиболее подходящей линией, так как ошибка почти 0. Для точки 3 ошибка отрицательна, поскольку прогнозируемое значение ниже. Тогда как для пункта 1 ошибка положительная и почти такой же величины. Для пункта 2 это 0. Сложение всего этого приведет к общей ошибке 0! Но ошибка, безусловно, намного больше. Если ошибка равна 0, тогда алгоритм будет считать, что он сходимся, хотя на самом деле этого не произошло, и преждевременно завершит работу. Это покажет очень меньшее значение ошибки, тогда как на самом деле значение будет намного больше. Итак, как вы можете утверждать, что это неправильная линия? Вы действительно не можете. Вы просто выбрали неправильную функцию потерь.

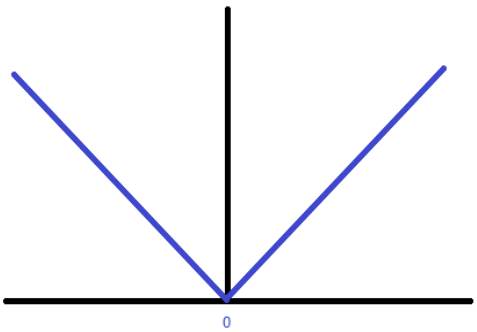
Сумма абсолютных ошибок (SAE):

SE определенно не была той функцией потерь, которую мы хотели бы использовать. Так что давайте немного изменим его, чтобы преодолеть его недостаток. Давайте просто возьмем абсолютные значения ошибок для всех итераций. Это должно решить проблему: верно? Или нет? Вот как будет выглядеть функция потерь:

$$\mathbf{L} = \Sigma(|\hat{Y} - Y|)$$

? **Ошибки не будут компенсировать друг друга и фактически будут складываться. Итак, есть ли потенциальные проблемы с этой функцией?**

Да! Эта функция потерь не дифференцируема при 0. График функции потерь будет:



Ось Y - функция потерь

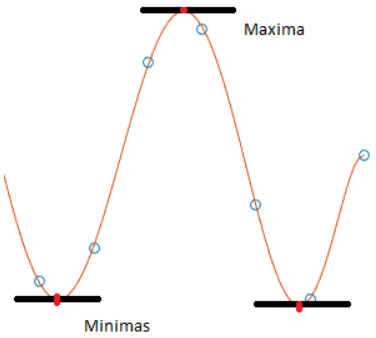
Производная не будет существовать в 0. Нам нужно дифференцировать функцию и приравнять ее к 0, чтобы найти оптимальную точку. А здесь это невозможно. Мы не сможем найти решение.

Сумма квадратов ошибок (SSE):

Так что давайте возьмем квадраты вместо модулей. Функция потерь теперь станет следующей:

$$\mathbf{L} = [\Sigma(\hat{Y} - Y)^2]$$

которая дифференцируема во всех точках и дает неотрицательные ошибки. Но вы можете возразить, почему мы не можем перейти к более высоким порядкам, например, к четвертому порядку или около того. Тогда что, если мы рассмотрим функцию потерь 4-го порядка, которая будет выглядеть так:



Следовательно, его градиент исчезнет в 3 точках. Так что у него также будут локальные минимумы, что не является нашим оптимальным решением. Нам нужно найти точку в глобальном минимуме, чтобы найти оптимальное решение. Так что давайте остановимся на самих квадратах.

MSE (Mean Squared Error) среднеквадратичная ошибка

Теперь представьте, что мы используем SSE в качестве функции потерь. Итак, если у нас есть набор данных, скажем, из 100 точек, наш SSE будет, скажем, 200. Если мы увеличим количество точек данных до 500, наш SSE увеличится, поскольку теперь в сумме квадратов ошибок будет 500 точек данных. Допустим, оно становится 800. Если мы снова увеличим количество точек данных, наш SSE увеличится еще больше. Справедливо? Точно нет!

N	L
100	200
500	800
1000	1200

Ошибка должна уменьшаться по мере того, как мы увеличиваем наши выборочные данные, поскольку распределение наших данных становится все более узким (относится к нормальному распределению). Чем больше у нас данных, тем меньше ошибок. Но в случае с SSE происходит полная противоположность. Вот, наконец, и наш воин - Mean Squared Error:

$$\mathbf{L} = \frac{1}{\mathrm{N}}[\,\Sigma(\hat{\mathrm{Y}} - \mathrm{Y})^2]$$

Мы берем среднее значение SSE. Чем больше данных, тем меньше будет агрегированная ошибка MSE.

N	L	L/N
100	200	2
500	800	1.6
1000	1200	1.2

Как видите, ошибка уменьшается по мере того, как наш алгоритм набирает все больше и больше опыта. Среднеквадратичная ошибка используется в качестве метрики по умолчанию для оценки производительности большинства алгоритмов регрессии, будь то R, Python или даже MATLAB.

Среднеквадратичная ошибка (RMSE):

Единственная проблема с MSE заключается в том, что порядок потери больше, чем порядок данных. Поскольку мои данные имеют порядок 1 и функцию потерь, MSE имеет порядок 2. Таким образом, мы не можем напрямую соотнести данные с ошибкой. Следовательно, мы берем корень MSE - среднеквадратичной ошибки:

$$\mathbf{L} = \sqrt{\frac{1}{\mathrm{N}}[\,\Sigma(\hat{\mathrm{Y}} - \mathrm{Y})^2]}$$

Здесь мы не меняем функцию потерь, и решение остается прежним. Все, что мы сделали, — это уменьшили порядок функции потерь, взяв корень.

Хьюбер Лосс

Лосс Хьюбера сочетает в себе лучшие свойства MSE и MAE (средней абсолютной ошибки). Он квадратичен для меньших ошибок и линейен в противном случае (и аналогично для его градиента). Он идентифицируется по своему дельта- параметру:

$$L_\delta(y, f(x)) = \begin{cases} \frac{1}{2}(y - f(x))^2 & \text{for} |y - f(x)| \le \delta, \\ \delta\,|y - f(x)| - \frac{1}{2}\delta^2 & \text{otherwise.} \end{cases}$$

Потери Хьюбера менее чувствительны или более устойчивы к выбросам в данных, чем MSE. Он также дифференцируем на 0. Это в основном абсолютная ошибка, которая становится квадратичной, когда ошибка мала. Насколько мала эта ошибка, чтобы сделать ее квадратичной, зависит от гиперпараметра (дельта), который можно настроить. Потери Хьюбера приближаются к MAE, когда ~ 0, и MSE, когда ~ ∞ (большие числа).

Смещение (Bias)

Смещение — это величина, на которую прогноз модели отличается от целевого значения по сравнению с данными обучения. Ошибка смещения возникает из-за упрощения допущений, используемых в модели, поэтому целевые функции легче аппроксимировать.

Разброс (Variance)

Разброс описывает, насколько случайная величина отличается от ожидаемого значения. Разброс основан на одной обучающей выборке.

Компромисс смещения и разброса (Bias / Variance Tradeoff)

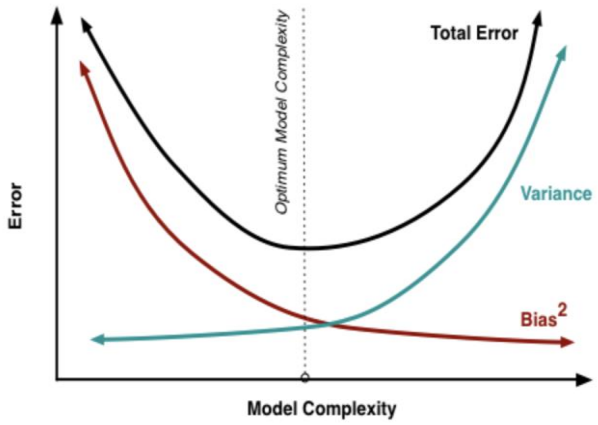
Есть удивительное уравнение, чтобы поймать этот баланс:

$$Error = Bias^2 + Variance + Irreducible\ Error$$

Неприводимая ошибка — это ошибка, которую вы не можете зафиксировать с помощью своей модели, независимо от обстоятельств. Это случайный шум набора данных.

Это уравнение интересно, потому что оно сигнализирует о важной взаимосвязи между смещением и разбросом: **они имеют обратную взаимосвязь.**

Как вы можете видеть на изображении ниже, по мере уменьшения смещения разброс увеличивается и наоборот.



Оптимальная модель - это та, которая сводит к минимуму смещение и дисперсию.

Итак, как мы можем создать оценку для наших моделей, которая должным образом отражает смещение и разброс?

В конце концов, мы видели в приведенном выше примере, что модель с лучшим показателем 2R2 не была лучшей моделью для использования.

Наиболее популярными методами являются **стратегия набора проверки** и **перекрестная проверка.**

Data Science вопросы для интервью

1. Почему в Наивном Байесе мы предполагаем, что функции условно независимы?
2. Sklearn имеет GaussianNB, MultinomialNB, КатегориальныйNB, BernoulliNB → Какую модель вы выберете для данных с категориями, числовыми, двоичными характеристиками?
3. Как реализовать Multinomial Naive Bayes с нуля для текстовых данных и сопоставить результаты с Sklearn MultinomialNB?
4. Как реализовать категориальный наивный байесовский метод с нуля для категориальных данных и сопоставить результаты с помощью Sklearn CategoriesavericalNB?
5. Как реализовать Gaussian Naive Bayes с нуля для числовых данных и сопоставить результаты с помощью Sklearn GaussianNB?
6. Каковы сложности обучения и проверки → временные и пространственные сложности наивного Байеса?
7. Влияет ли на наивный байесовский метод несбалансированные данные, если да, то как это исправить?
8. Как наивный байесовский метод влияет на выбросы?
9. Можно ли интерпретировать наивный байесовский анализ, можем ли мы сказать, какие особенности помогли нам предсказать конкретный класс?
10. Является ли наивный байесовский классификатор линейным, может ли он решить нелинейную границу принятия решений?
11. Как избежать проблем с переобучением или недообучением в наивном байесовском методе?
12. Опишите механизм работы логистической регрессии, что такое функция правдоподобия и как она максимизируется.
13. Разница между регуляризациями L1 и L2, в чем их смысл и как они используются в логарифмической регрессии.
14. Как устроено дерево и лес, по какому принципу растет дерево, как упорядочено и почему. Определение энтропии и критерия Джини.
15. Вопросы о предварительной обработке: что такое тригонометрические функции и когда это может быть полезно; способы восполнения пробелов, способы поиска выбросов и аномалий.
16. Зачем нужен логарифм в логистической регрессии?
17. Могут ли две обученные случайные модели леса иметь один и тот же признак с разной важностью?
18. Какими свойствами должна обладать потеря для повышения?
19. Может ли функция активации быть недифференцируемой?
20. Может ли линейная регрессия переобучаться? Почему?

Телекоммуникационная компания

1. SQL. Есть 2 таблицы в одной 20 строк в других 30 строках. Мы соединяемся, используя внутреннее соединение 1 = 1. Сколько строк будет в итоговой таблице?
2. SQL. Как подсчитать дубликаты?
3. SQL. Как работает левый join?
4. Задача. Есть набор данных, в котором есть 3 столбца: название города, координаты квартиры, стоимость аренды. Объясните порядок прогнозирования стоимости аренды.
5. Что делать с категоричными признаками?
6. Как устранить пропуски в этом наборе данных?
7. Как рассчитать расстояние от центра?
8. Формула расстояния Евклида?
9. Где взять координаты городов?
10. Формула линейной регрессии?
11. Как рассчитываются веса линейной регрессии?
12. Формула линейной регрессии?
13. Опишите алгоритм дерева решений?
14. На каком основании выполняется разделение в первую очередь?
15. Есть прогнозируемые данные об оттоке. Как узнать топ-20 нелояльных клиентов?
16. Какие существуют метрики классификации?

17. Что такое точность?
18. Что такое точность и охват?
19. Как объяснить бизнесу, что означает точность 0,6 и охват 0,5?
20. Как построить AUC-ROC?
21. Формула логистической регрессии?
22. Формула сигмоидной функции и зачем она нужна?
23. Метрики в задачах регрессии?
24. Формула MSE?
25. Что такое R2? Формула и что это значит, если R2 = 0,3?

Компьютерное зрение

1. Основные вещи, такие как классификация, обнаружение, сегментация.
2. Иногда задавались узкие определения вроде архитектуры какой-то сетки, например resnet, resnext, se-resnext и т. д.
3. Как бы вы решили какую-то прикладную задачу (обычно это то, что решают на работе), выбирая метрику, собирая данные и т. д.
4. Какие архитектуры классификации, сегментации, обнаружения вы знаете? (в чем разница между ними)
5. Чем ReLU отличается от сигмовидной? Почему лучше в глубоких сетях? Что является причиной этого?
6. Почему Адам лучше SGD? Какие еще есть алгоритмы обучения и почему они лучше?

Банк

1. В чем разница между метриками регрессии (среднеквадратичная ошибка, средняя абсолютная ошибка, R в квадрате)? Когда использовать?
2. Как связаны коэффицент Джини и AUC-ROC?
3. Когда менять порог классификации?
4. После Upsampling выясняется, что вероятности классов меняются. Как сделать так, чтобы модель была предсказуемой на самом деле?

Интервью - Вопросы от Мирас Амира

1. (startup) Чем бустинг отличается от бэггинга?
2. (startup) Чем xgboost отличается от повышения в sklearn?
3. (startup) Расскажите об алгоритмах выбора признаков
4. (startup) Что такое PCA?
5. (startup) Подскажите, как обучается нейросеть (backprop и т. д.)
6. (startup) На что влияет batch_size в нейронных сетях?
7. (startup) Расскажите о методах оптимизации в нейронных сетях
8. (startup) Как определяется функция потерь для разных алгоритмов?
9. (startup) Как решить проблему обнаружения спама. Какой алгоритм вы бы использовали, какую метрику и т. д.?
10. (media) Как решить проблему с несбалансированными выборками?
11. (media) Алгоритм выдал 0,1 и 0,5 для объектов нулевого класса и 0,2, 0,3, 0,4 для первого класса. Рассчитать ROC-AUC?
12. (media) Что нужно сделать, чтобы алгоритм напрямую оптимизировал ROC-AUC?
13. (media) Что такое декоратор Python?
14. (media) Можно ли скомпилировать код, написанный на интерпретируемом языке?
15. (telecom) Расскажите подробнее о каком-нибудь алгоритме машинного обучения, кроме knn
16. (telecom) Какой метод оптимизации используется в vowpal-wabbit?
17. (can't remember) Как настроить xgboost / random forest? С каких параметров начать, какие параметры на что влияют и т. д.
18. (can't remember) Чем критерий Джини отличается от энтропии в деревьях решений и на что он влияет?

19. (bank) Алгоритм ошибочно принимается за большие значения ответа в задаче регрессии. Что делать?
20. (bank) Мессенджеры еще не появились, люди пишут. У вас есть набор данных, который состоит из строк формата <идентификатор отправителя, идентификатор получателя, дата отправки SMS с точностью до секунд>. Для 50% id-шников мы знаем пол человека, а для остальных пол необходимо предсказать. Как это решить?
21. (scoring) Мы хотим решить проблему классификации: есть ли у пользователя ВКонтакте машина и какого она класса. Какие знаки использовать, какой алгоритм использовать? У нас нет размеченной выборки, как собрать данные для обучающей выборки?
22. (scoring) Некоторым пользователям банк хочет предложить карту для путешественников (например, All Airlines в Тинькофф). Как определить таких пользователей на странице ВКонтакте или Instagram?

Полезные ссылки ссылки и репозитории для подготовки

- https://github.com/chiphuyen/machine-learning-systems-design
- https://github.com/rbhatia46/Data-Science-Interview-Resources
- https://github.com/khangich/machine-learning-interview
- https://github.com/alexeygrigorev/data-science-interviews/
- https://www.springboard.com/blog/machine-learning-interview-questions/
- https://medium.com/analytics-vidhya/test-your-skills-26-data-science-interview-questions-answers-69cb2b223e57
- https://dzone.com/articles/109-data-science-interview-questions-and-answers
- https://interview-mds.ru/
- https://github.com/iamtodor/data-science-interview-questions-and-answers
- https://github.com/academic/awesome-datascience

SQL

- https://www.interviewbit.com/sql-interview-questions/
- https://towardsdatascience.com/crack-sql-interviews-6a5fc90ec763
- https://www.indeed.com/career-advice/interviewing/sql-queries-interview-questions
- https://www.toptal.com/sql/interview-questions

Поведенческие интервью

- https://medium.com/@scarletinked/are-you-the-leader-were-looking-for-interviewing-at-amazon-8301d787815d
- https://github.com/jlevy/og-equity-compensation?fbclid=IwAR3XvGJbYYNzyAAzUR2VifxJyU92epfIbZ619C0qcGrblN635plyVSH7Gew#introduction

Материалы о компании

- https://towardsdatascience.com/the-lyft-data-scientist-interview-a935700c3f8e
- https://towardsdatascience.com/the-netflix-data-scientist-interview-35093d4c20aa

System Design

- https://github.com/donnemartin/system-design-primer

Примеры интервью на YouTube

- https://www.youtube.com/channel/UCcQx1UnmorvmSEZef4X7-6g

Фишки для прохождения Data Science интервью

- https://www.youtube.com/watch?v=3BRLGRqj8ps&t=858s&ab_channel=Galvanize

- https://www.simplilearn.com/tutorials/data-science-tutorial/data-science-interview-questions
- https://www.edureka.co/blog/interview-questions/data-science-interview-questions/
- https://www.edureka.co/blog/interview-questions/data-science-interview-questions/
- https://www.kdnuggets.com/2016/02/21-data-science-interview-questions-answers.html
- Questions divided genre wise - https://www.springboard.com/blog/data-science-interview-questions/
- http://nitin-panwar.github.io/Top-100-Data-science-interview-questions/?utm_campaign=News&utm_medium=Community&utm_source=DataCamp.com
- Questions in all topics, not segregated - https://data-flair.training/blogs/data-science-interview-questions/
- Product Question - https://productmanagerhq.com/amazon-product-manager-interview-questions/