Градиентный спуск

Алгоритм градиентного спуска — это метод оптимизации, который используется в машинном обучении для поиска минимальных значений функции потерь, измеряющей ошибку модели. Идея заключается в том, чтобы итеративно корректировать параметры модели — веса — в направлении отрицательного градиента функции ошибки. Градиент показывает, в каком направлении функция возрастает быстрее всего, а значит, шаг в обратном направлении уменьшит ошибку.

В начале выбираются случайные начальные значения параметров. Затем на каждом шаге вычисляется градиент функции потерь по параметрам, и параметры обновляются по формуле

$$w_{new}=w_{old}-\eta\cdot\nabla L(w_{old}),$$

где η − скорость обучения (learning rate). Процесс повторяется до тех пор, пока изменения функции ошибки не станут минимальными или не будет достигнуто максимальное число итераций.

Итерации следует продолжать, пока не наступает сходимость. Она определяется разными способами, но в данном случае удобно определять как ситуацию, когда векторы весов от шага к шагу изменяются незначительно, то есть норма отклонения вектора весов на текущем шаге от предыдущего не привышает заданное значение ε:

$$||w_{new}-w_{old}||<\varepsilon.$$

Как видно тут используется евклидова (L2) норма:

$$||x|| = \sqrt{\sum_{i=1}^{l} x_i^2}$$

Среднеквадратичная ошибка имеет один минимум и непрерывна на всей области значений (то есть является выпуклой и гладкой), а значит в каждой ее точке можно посчитать частные производные.

Формула вычисления градиента принимает вид

$$\nabla L(w_{old}) = \frac{2}{l} X^T(X\cdot w_{old}-y)$$

Пошаговый алгоритм градиентного спуска следущий:

1. Инициализировать начальные веса (приравнять их нулю или взять значения из нормального распределения)
2. Инициализировать цикл по количеству итераций

   2.1 рассчитать новые веса по формуле $w_{new}=w_{old}-\eta\cdot\nabla L(w_{old})$,

2.2 если разница весов по L2 меньше чем очень маленькое число, то цикл завершить ($\|w_{new} - w_{old}\| < \varepsilon$)

Реализуем алгоритм градиентного спуска

Для решения задачи регрессии будем минимизировать такую функцию потерь как MSE.

```
def mserror(X, w, y):
    y_pred = X.dot(w)
    return (np.sum((y_pred - y)**2)) / len(y)
```

Для нахождения весов будем использовать формулу из метода наименьших квадратов

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

```
normal_eq_w = np.linalg.inv(np.dot(X.T, X)) @ X.T @ Y
```

Инициализируем начальные веса

```
w = np.zeros(n_features)
```

задаем скорость обучения

```
eta = 0.01
```

Задаем количество итераций

```
max_iter = 1e4
```

Задаем разницу весов, при которой алгоритм останавливается

```
min_weight_dist = 1e-8
```

Зададим начальную разницу весов большой, чтобы начался цикл

```
weight_dist = np.inf
```

Ход градиентного спуска

```
# ход градиентного спуска
while weight_dist > min_weight_dist and iter_num < max_iter:
    y_pred = np.dot(X, w)
    dL = 2 / Y.shape[0] * np.dot(X.T, y_pred - Y)
    new_w = w - eta * dL
    #вычисление евклидова расстояния между двумя векторами new_w и w с
использованием нормы L2
    weight_dist = np.linalg.norm(new_w - w, ord=2)
    error = mserror(X, new_w, Y)

    w_list.append(new_w.copy())
    errors.append(error)

    print(f'Iter {iter_num}: error - {error}, weights: {new_w}')

    iter_num += 1
```

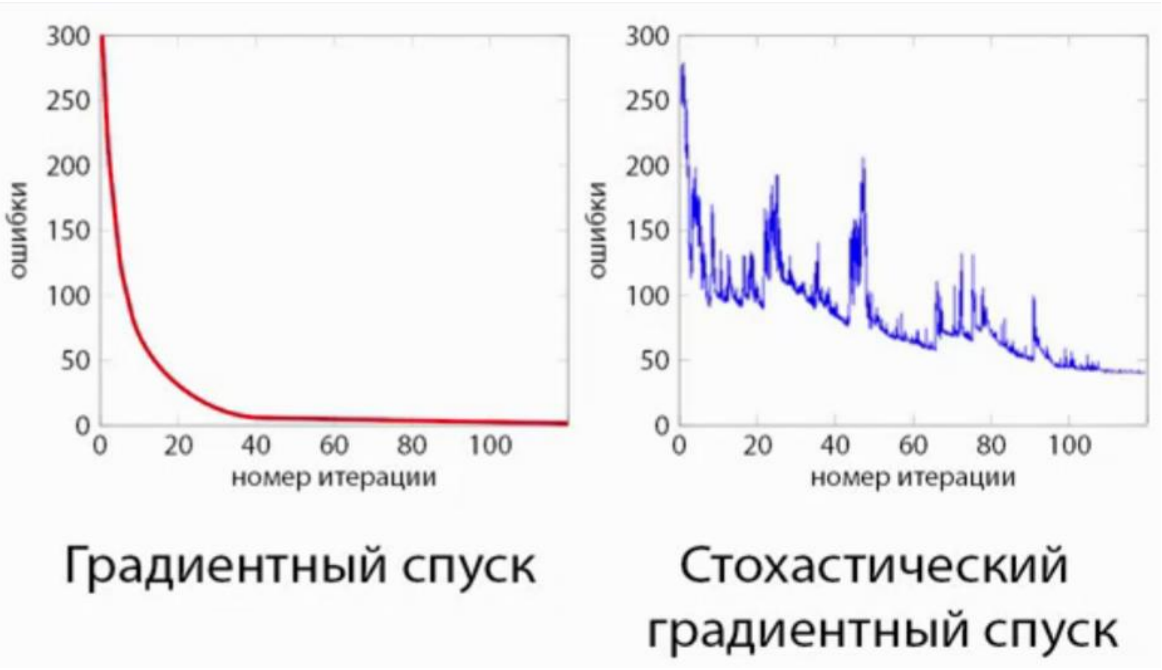
```
    w = new_w

w_list = np.array(w_list)
w_pred = w_list[-1]
```

Градиентный спуск помогает эффективно обучать модели, автоматически находя лучшие параметры для наименьшей ошибки предсказания. Существует несколько вариантов, включая пакетный, стохастический и мини-батч градиентный спуск, которые отличаются объемом данных, используемым на каждой итерации.

Стохастический градиентный спуск

Если в случае градиентного спуска мы стараемся на каждой итерации уменьшить ошибку на всей выборке, и по мере увеличения числа итераций ошибка падает монотонно, то в случае стохастического градиентного спуска мы уменьшаем на каждой итерации ошибку только на одном объекте, но при этом есть вероятность увеличить ее на другом объекте, поэтому график изменения ошибки может получаться немонотонным, и даже иметь пики. То есть на какой-то итерации мы можем даже увеличить ошибку, но при этом в целом по ходу метода ошибка снижается, и рано или поздно мы выходим на нормальный уровень.



Чтобы внести изменения в код приведенный выше и сделать из алгоритма классического градиентного спуска стохастический необходимо в цикле добавить строку, которая случайным образом выберет один объект для обучения

```
    # генерируем случайный индекс объекта выборки
    train_ind = np.random.randint(X.shape[0], size=1)
```

```
    y_pred = np.dot(X[train_ind], w)
    new_w = w - eta * 2 / Y[train_ind].shape[0] * np.dot(X[train_ind].T,
y_pred - Y[train_ind])
```

Переобучение и регуляризация

Одним из знаков, что произошло переобучение модели, или мерой сложности является получение больших по модулю весов при признаках. Например:

```
[ 0.00000000e+00  1.07116178e+03  8.76325216e+02  4.03130196e+01
 -5.61701732e+01 -9.81320836e+00  9.56286199e-02  1.16554743e-01
  6.21060019e-03]
```

Видим веса 2 и 3 порядков в то время как в исходной зависимости ничего подобного не было. Это и говорит нам о том, что в данном случае имеет место переобучение. На этой особенности и основывается метод регуляризации для борьбы с переобучением. Существуют следующие способы борьбы с переобучением:

Метод	Суть и применение
Регуляризация (L1, L2)	Ограничение весов, штраф за сложность модели
Dropout	Случайное отключение нейронов во время обучения
Ранняя остановка	Прекращение обучения при росте ошибки на валидации
Кросс-валидация	Многократное разбиение данных для объективной оценки
Увеличение данных	Сбор новых или аугментация существующих данных
Снижение сложности	Уменьшение числа признаков, слоёв, параметров
Батч-нормализация	Нормализация входов слоёв
Ансамбли	Использование нескольких моделей для повышения обобщающей способности

Ridge-регуляризация, также известная как L2-регуляризация, добавляет к функции потерь модельного обучения штраф, пропорциональный сумме квадратов весов модели. Это ограничение заставляет веса оставаться как можно меньшими, предотвращая чрезмерное увеличение коэффициентов, что помогает снизить эффект переобучения. В отличие от *L1-регуляризации*, которая может занулять веса, L2 постепенно уменьшает их, делая модель более устойчивой и гладкой. Такой подход особенно полезен при наличии большого числа признаков или высокой корреляции между ними, поскольку

он сохраняет вклад всех признаков, но снижает влияние избыточных. Гиперпараметр регуляризации, обычно обозначаемый как λ или alpha, контролирует силу штрафа: чем он выше, тем сильнее уменьшаются веса и упрощается модель. Ridge-регуляризация широко используется в линейных моделях, включая Ridge-регрессию, и легко интегрируется в популярные библиотеки для машинного обучения.

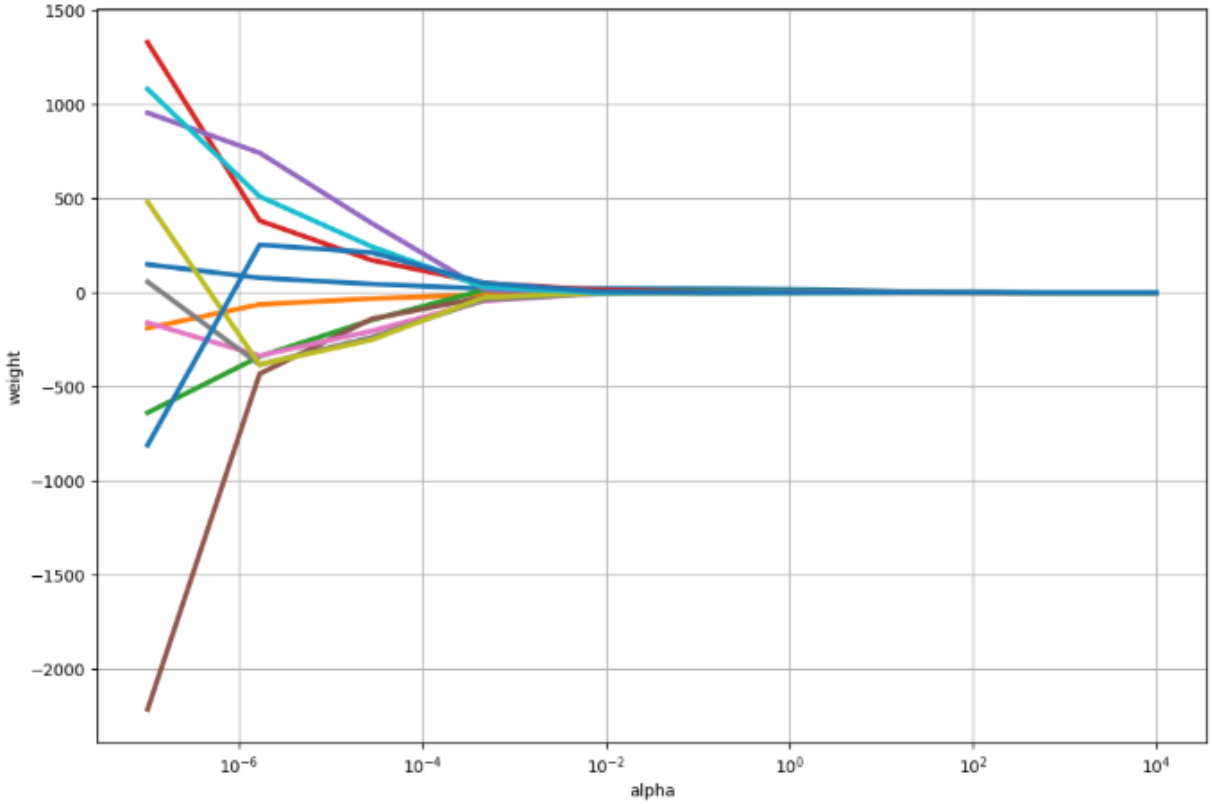
Метод регуляризации заключается в "штрафовании" модели за слишком большие веса путем добавления нового члена к ошибке

$$Q(w, X) + \alpha||w||^2 \to \min_w$$

Так, например, можно поэкспериментировать с коэффициентом регуляризации, взять набор разных коэффициентов и посчитать метрики модели на обучающей выборке и тестовой можно подобрать наилучший коэффициент, чтобы не уйти в переобучение. Ниже представлен набор коэффицентов регуляризации и метрики

	alpha	train_r2	test_r2
0	no	1.000000	-76.478993
1	0.0	0.999901	-18.170492
2	0.000002	0.999682	-0.349948
3	0.000028	0.998399	0.719186
4	0.000464	0.995018	0.956577
5	0.007743	0.993622	0.963986
6	0.129155	0.992731	0.954618
7	2.154435	0.970881	0.960356
8	35.938137	0.732147	0.720257
9	599.48425	0.142472	0.114558
10	10000.0	0.009741	-0.010146

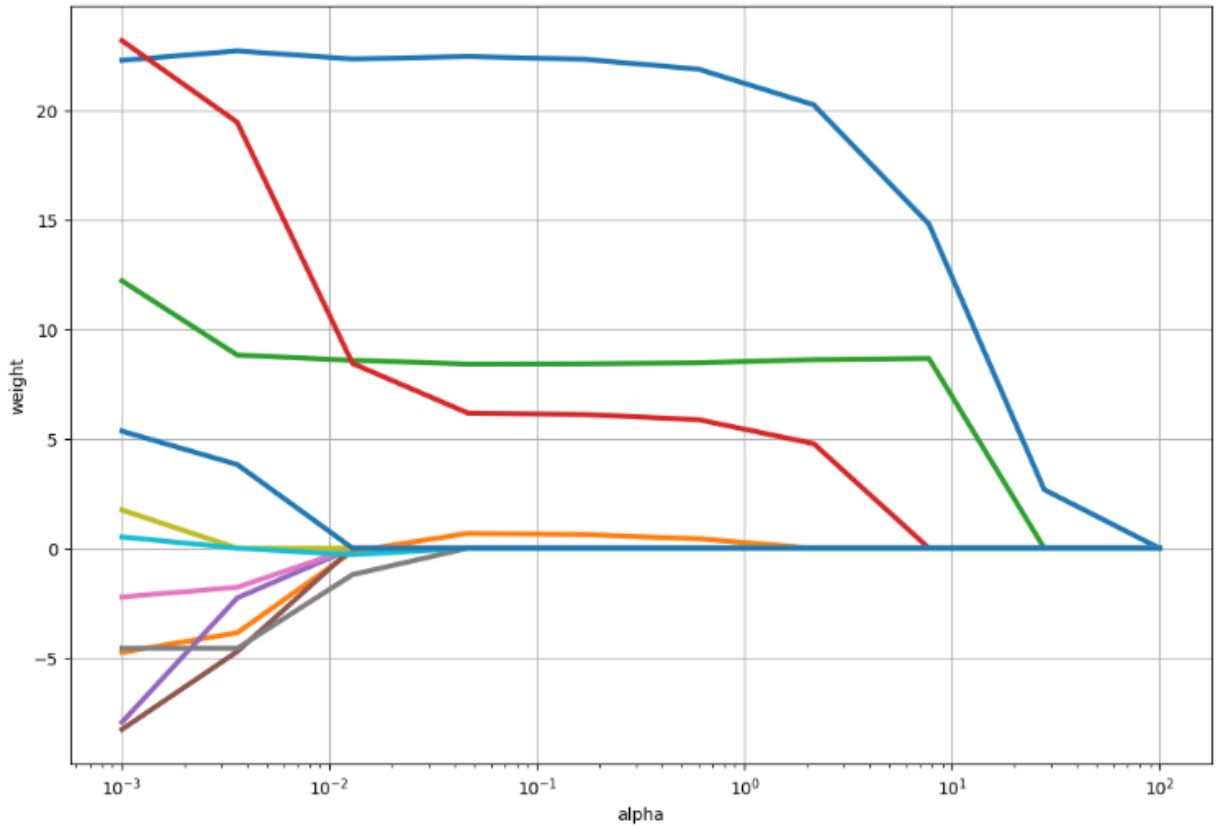
Видно, что при отсутствии регуляризации качество модели оставляло желать лучшего, самое лучшее соотношение метрик находится на строке 5, метрики наиболее близки и они высокие. Так же при высоких коэффициентах регуляризации веса стремятся к нулю, что отражается на метриках. Визуально получим следующее



Описанный выше метод с использованием L2 -нормы вектора весов в качестве регуляризатора называется L2 -регуляризацией. По аналогии существует также L1 -регуляризация, использующая в качестве регуляризатора L1 -норму вектора весов, то есть сумму модулей весов.

$$||w||_1 = \sum_{j=1}^{d} |w_j|.$$

Для *Лассо* регуляризации гарфик будет следующим



L_1 регуляризация зануляет веса некоторых признаков, поэтому может использоваться для отбора признаков.