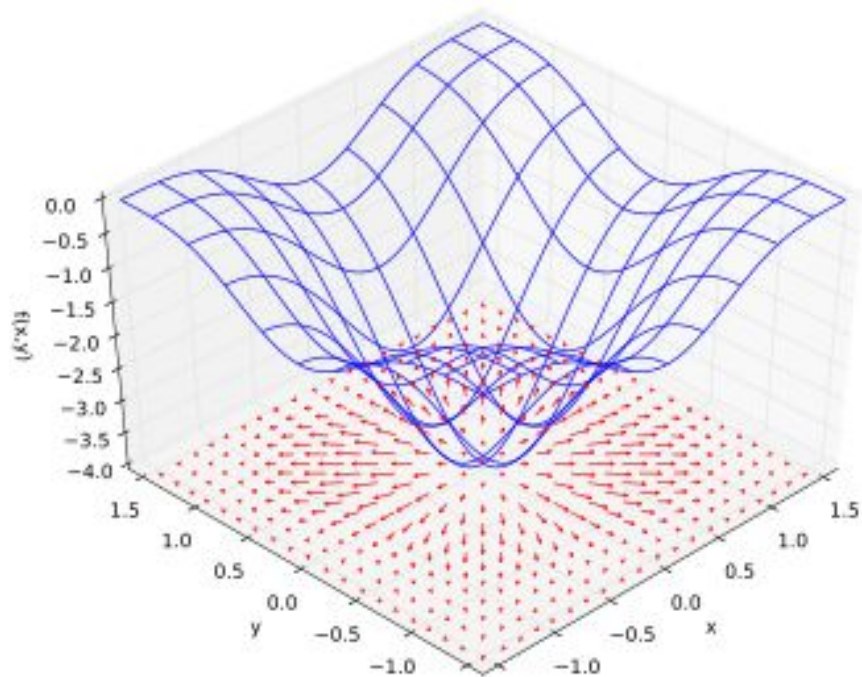


Классификация: продолжение

Маша Шеянова, masha.shejanova@gmail.com

Градиентный спуск

Что такое градиент



Градиент — это вектор, указывающий **направление наибольшего роста функции**.

Градиент находится через *дифференцирование* (это вектор, элементы которого — значения всех возможных *частных производных* в конкретной точке).

Идея

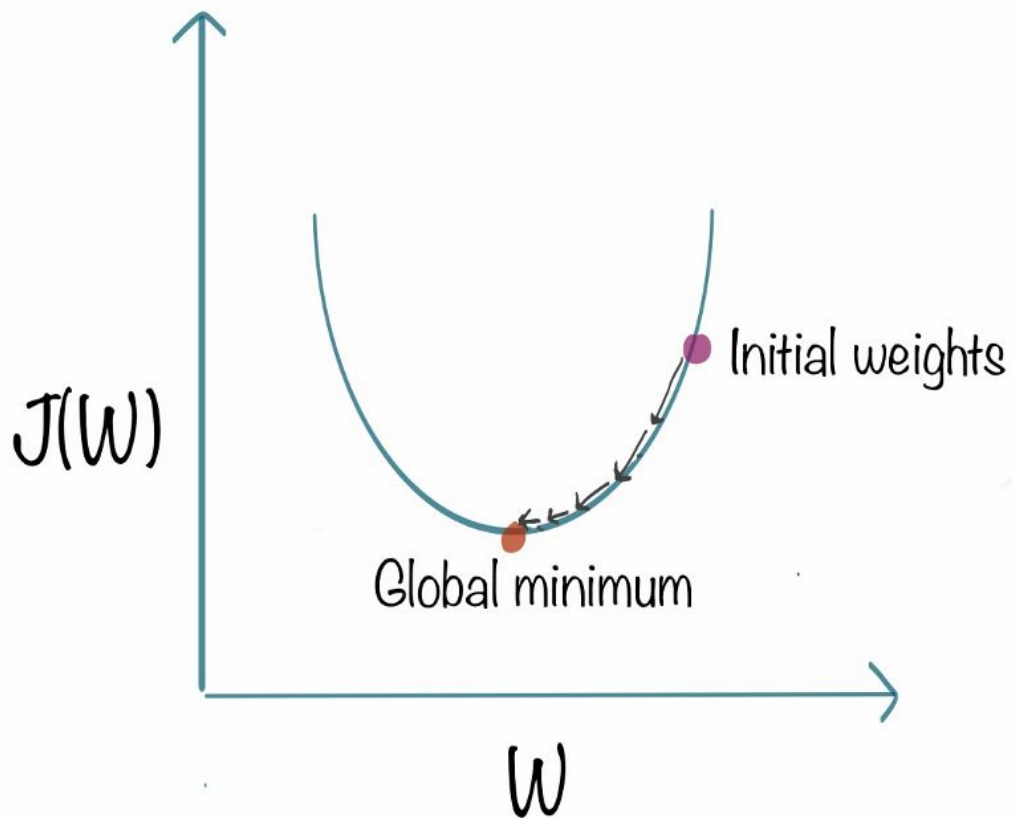
loss function = cost function = error function = функция потерь = $J(W)$

Её мы хотим минимизировать.

Зная градиент мы умеем находить, в каком направлении функция растёт быстрее всего. Но нам нужен минимум функции потерь, а не максимум!

Решение очевидно: найдём градиент и пойдём в обратную сторону.

С какой скоростью? Функция изменяется быстро — с большой, медленно — с маленькой.



Источник картинки — очень понятно про то, как оно работает и какое бывает.

Шаги:

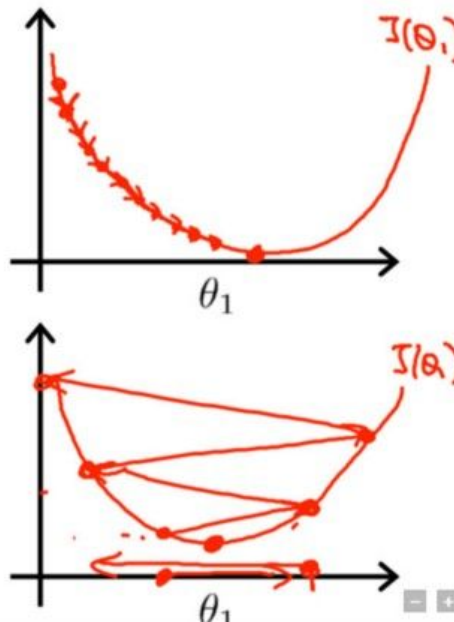
- подобрать случайные коэффициенты
- найти градиент функции потерь в этой точке
- обновить коэффициенты
- повторять, пока мы не начнём кружиться вокруг одной точки

Learning rate

$$\theta_1 := \theta_1 - \alpha \frac{\partial}{\partial \theta_1} J(\theta_1)$$

If α is too small, gradient descent can be slow.

If α is too large, gradient descent can overshoot the minimum. It may fail to converge, or even diverge.



Learning rate — это гиперпараметр, который отвечает за то, как быстро мы изменяем веса при признаках. (отсюда)

Что делать, если хочется разобраться глубже

Непонимание градиентного спуска, в принципе, не мешает вам решать типичные задачи готовыми инструментами. Но может мешать улучшать модель и решать проблемы, если что-то пойдет не так.

Если всё ещё ничего непонятно, keep calm and:

- пройдите небольшой курс по multivariate calculus на khan academy
- посмотрите [вот это видео](#) про градиентный спуск
- прочитайте [эту](#) и [эту](#) статью
- если удастся сформулировать вопросы, feel free to ask

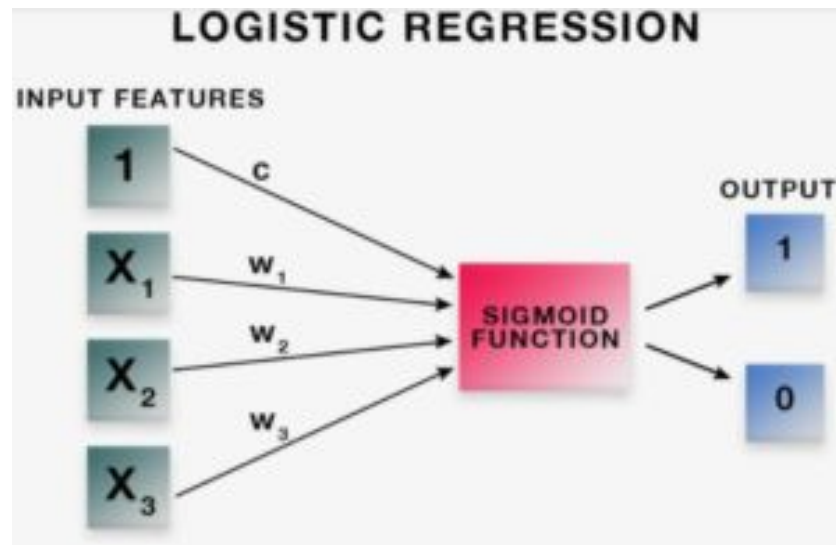
Логистическая регрессия

Для чего

from sklearn.linear_model import LogisticRegression

Несмотря на название, это алгоритм классификации ͸_(ツ)_/͸

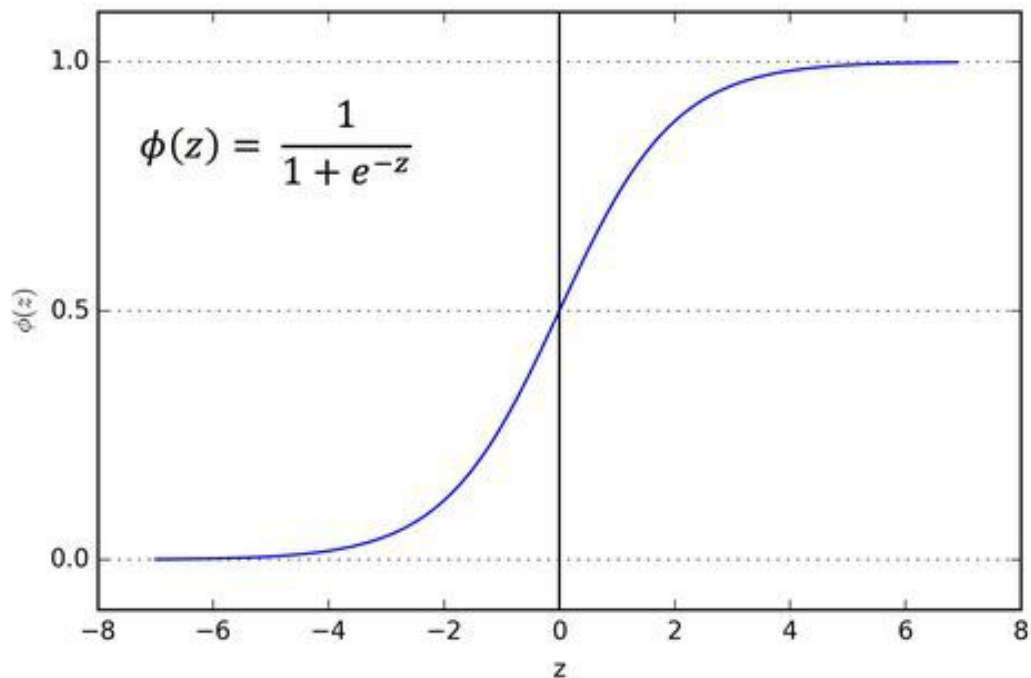
Подбирает веса коэффициентов (w_1 , w_2 , w_3 ...), скармливает функции-сигмоиду, которая принимает значения от 0 (класс 1) до 1 (класс 2).



$$y = \text{logistic} (c + x_1 * w_1 + x_2 * w_2 + x_3 * w_3 + + x_n * w_n)$$

$$y = 1 / 1 + e [- (c + x_1 * w_1 + x_2 * w_2 + x_3 * w_3 + + x_n * w_n)]$$

Sigmoid function

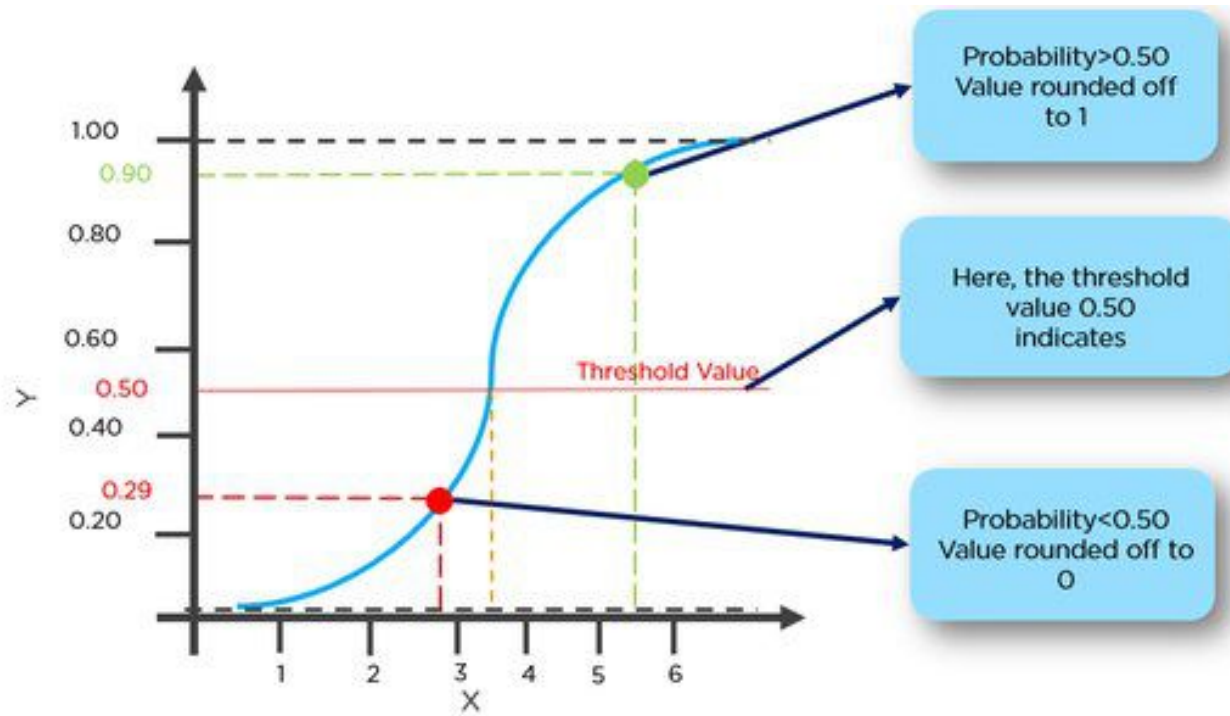


z — сумма признаков,
умноженных на свои
коэффициенты:

$$a * x_1 + b * x_2 + c * x_3 + \dots$$

$\phi(z)$ — вероятность
положительного класса

Что в итоге



Мы решили задачу бинарной классификации

... а что делать, если класса не 2, а, например, 4?

Мультиклассовая классификация и Softmax

Softmax — нормализация вектора вероятностей (= сделать так, чтобы сумма вероятностей была 1).

$$\sigma(\mathbf{z})_j = \frac{e^{z_j}}{\sum_{k=1}^K e^{z_k}} \quad \text{for } j = 1, \dots, K \text{ and } \mathbf{z} = (z_1, \dots, z_K) \in \mathbb{R}^K$$

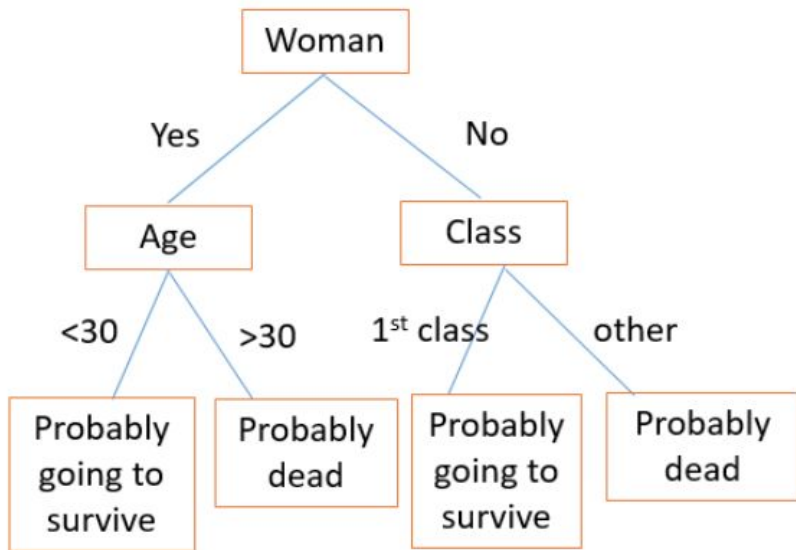
Логистическая регрессия выдаёт вероятность принадлежности к определённому классу vs. все остальные — для каждого класса. Чтобы понять, как какому классу всё-таки объект относится, прогоняем эти вероятности через softmax.

Деревья решений

Как выглядит дерево решений

Источник картинки.

Дерево решений на примере датасета из титанка.



Как строятся деревья?

Сверху-вниз: сначала находим корень, потом в каждом из поддеревьев — новый корень, и так далее.

Как выбираем корень? Вводим “*impurity function*” — насколько плохо классифицирован датасет. *Impurity function* может быть разной, главное — чтобы она была большой, когда разбиение датасета на “кучки”

Когда каждое новое разделение классифицирует датасет чуть лучше, значение *impurity*

Алгоритм

1. Вычислить impurity function для изначального датасета
2. Для каждого признака:
 - a. вычислить impurity function для каждого сплита
 - b. вычислить, насколько текущий атрибут лучше, чем было до него
3. Выбрать атрибут с лучшей разницей в impurity function
4. Повторять, пока мы не захотим остановиться

Энтропия


Как можно измерить насколько распределение "разнородное", или насколько "грязный" датасет?

Взять математическое ожидание количества бит, которое понадобится, чтобы закодировать один из исходов в **оптимальной** кодировке.

Это количество бит — **информация**.

Матожидание информации - **энтропия**.

$$Entropy = - \sum p(X) \log p(X)$$



here $p(x)$ is a fraction of examples in a given class

Information gain

Разница между энтропией до и после разделения. Иными словами, насколько “чище”, “определённое” стали данные.

$$IG(A, S) = H(S) - \sum_{t \in T} p(t)H(t)$$

Where,

- $H(S)$ – Entropy of set S
- T – The subsets created from splitting set S by attribute A such that $S = \bigcup_{t \in T} t$
- $p(t)$ – The proportion of the number of elements in t to the number of elements in set S
- $H(t)$ – Entropy of subset t

Деревья решений в sklearn

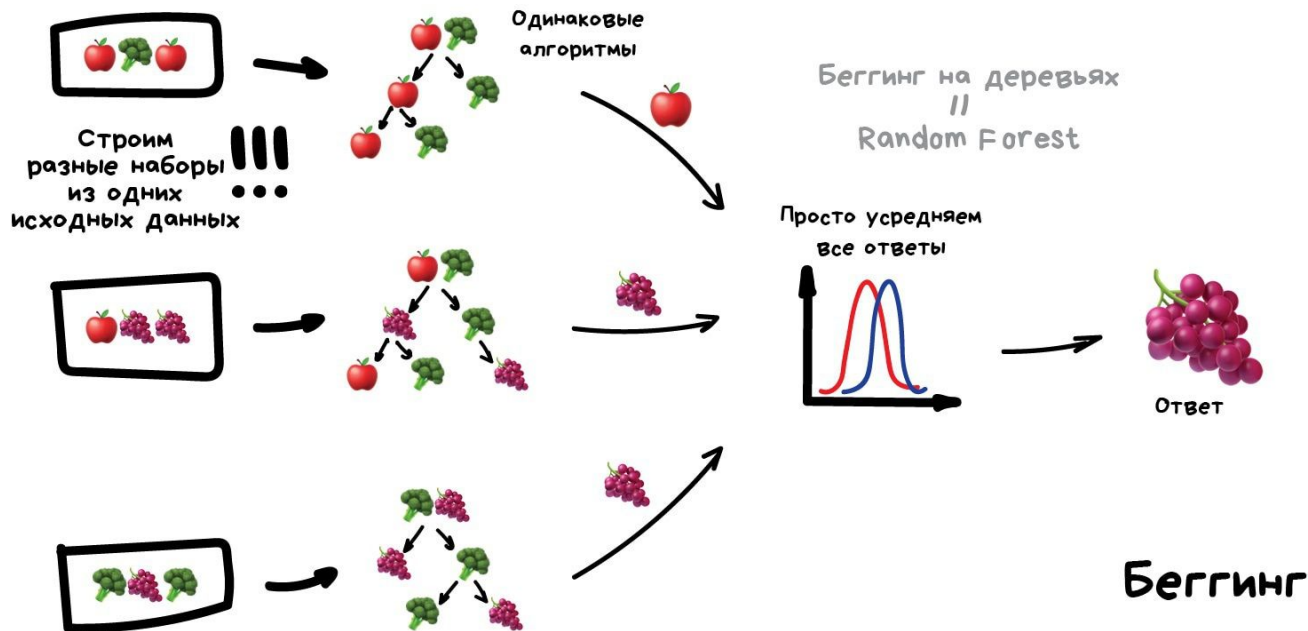
from sklearn.tree import DecisionTreeClassifier

Гиперпараметры:

- criterion — gini (ещё один способ разделить) или entropy
- min_samples_split — сколько должно быть точек данных, чтобы мы продолжили делить
- min_impurity_decrease
- max_depth — максимально возможная высота дерева
- max_leaf_nodes — максимально возможная “ширина” дерева

Случайный лес (random forest)

```
from sklearn.ensemble import RandomForestClassifier
```



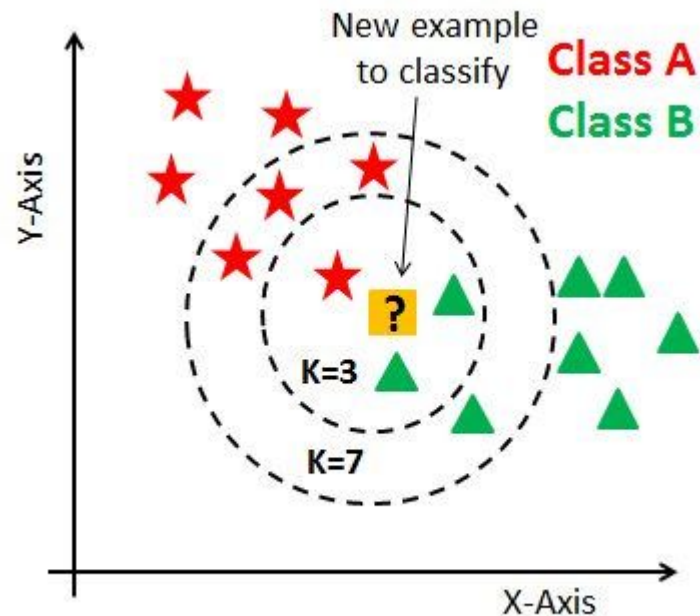
Обучаем один алгоритм много раз на случайных выборках из данных. Потом усредняем ответы.

Данные в случайных выборках могут повторяться.

Другие алгоритмы классификации

kNN: k ближайших соседей

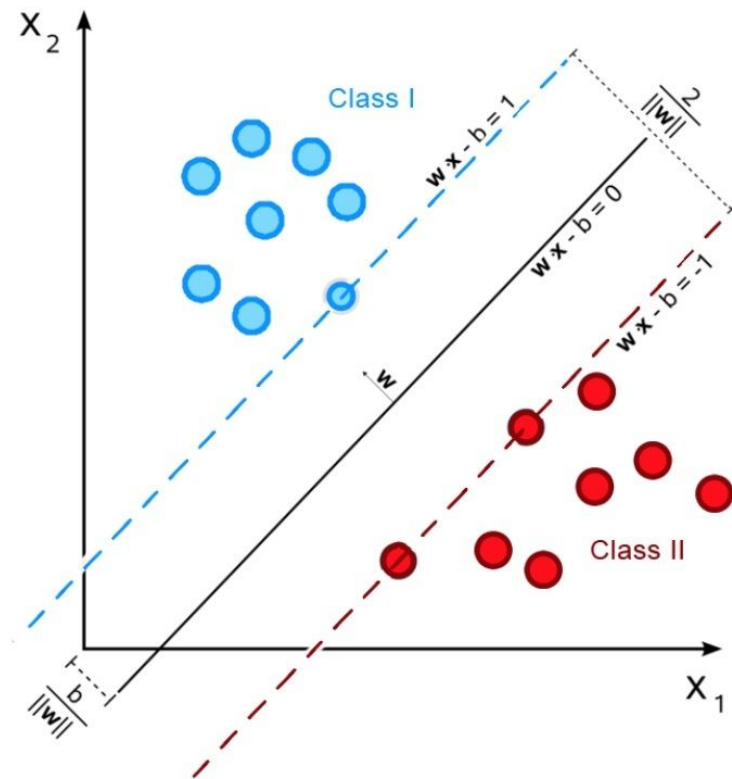
Идея: чтобы понять, к какому классу относится объект, смотрим на класс на какого-то числа близких к нему (в векторном пространстве) объектов.



SVM: support vector machines

Идея: ищем вектор, “идеально”
разделяющий два класса.

“Идеально” — значит, что зазор между
двумя классами максимальный.



Гиперпараметры

Параметры vs гиперпараметры

Параметр — это внутренняя характеристика модели, значение которой может быть выведено из данных. Это, например, коэффициент при признаке “слово *КОТИК*”.

Гиперпараметр — это характеристика, “внешняя” по отношению к модели. Например, k в kNN . Его нельзя вывести из данных при обучении, и надо подбирать как-то отдельно.

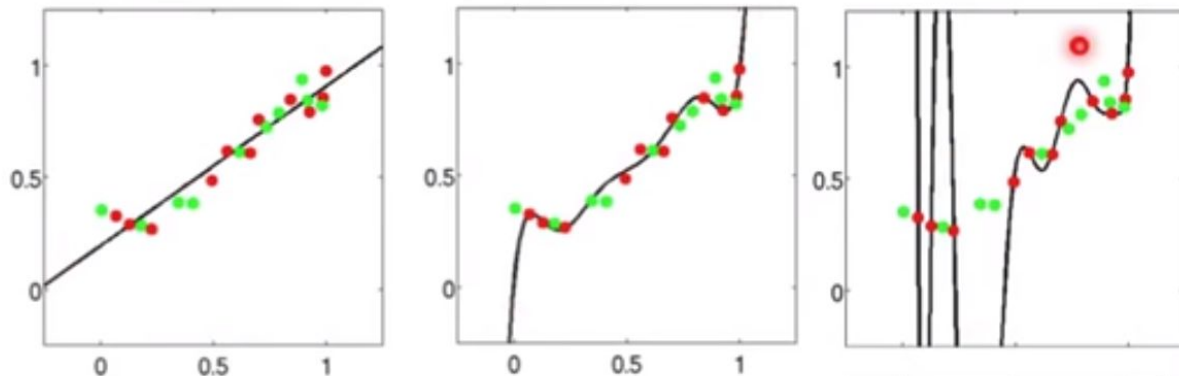
(Определения из [статьи](#)).

пример гиперпараметра: альфа / лямбда

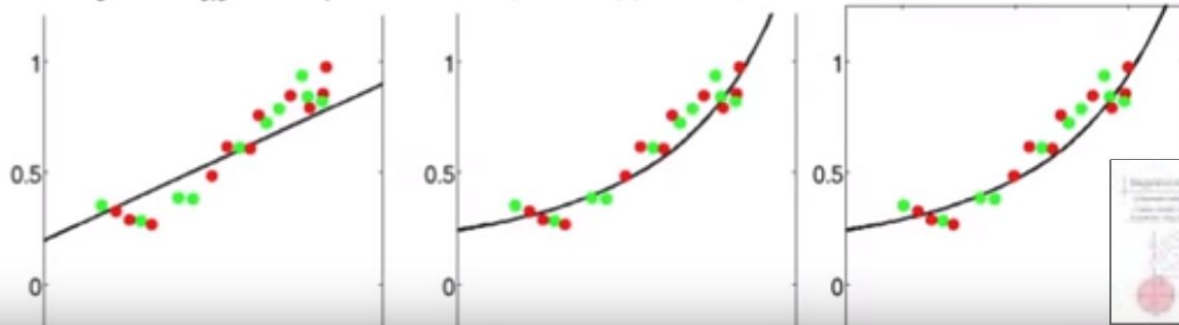
Чем больше
эта величина,
тем больше
значения мы
придаем
штрафу за
сложность
модели.

[Источник.](#)

**Alpha =0
(Unreg)**



Alpha =1



Подбор гиперпараметров

Как подобрать гиперпараметры?

- можно пробовать менять разные варианты руками
- можно перебирать их в цикле

```
for df in range(0, 20):  
    vec = TfidfVectorizer(min_df=df)
```

- а можно использовать Grid Search

```
from sklearn.model_selection import GridSearchCV
```

Заключение

Ресурсы

Почитать (англ):

- [статья про градиентный спуск](#)
- [про learning rate](#)
- [про softmax](#)
- [про Grid Search](#)

Посмотреть (англ):

- [про софтмакс](#)

Домашнее задание: классификация отзывов

Данные: отзывы на фильмы.

Что сделать:

- 3 балла — считать датасет, обучить на нём любой из описанных сегодня классификаторов, измерить качество
- 3 балла — перебрать как минимум 3 классификатора, найти лучший
- 2 балла (*) — попробовать разные гиперпараметры, найти лучший
- бонусный 1 балл за понятный и чистый код :)