Линейные классификаторы

Лекция № 8

Лектор: Шевляков Артём

Простое правило классификации

Если объект А описывается признаками $(x_1,x_2,\ldots x_n)$, то он принадлежит классу 1, если

$w_1x_1+w_2x_2+\ldots+w_nx_n+w_0 >0,$

и объект А принадлежит классу -1, если

$w_1x_1+w_2x_2+\ldots+w_nx_n+w_0 <0.$

Числа w_i являются параметрами модели и настраиваются по тренировочной выборке.

(Как вы уже заметили, в теории линейной классификации классы удобнее обозначать через 1 и -1).

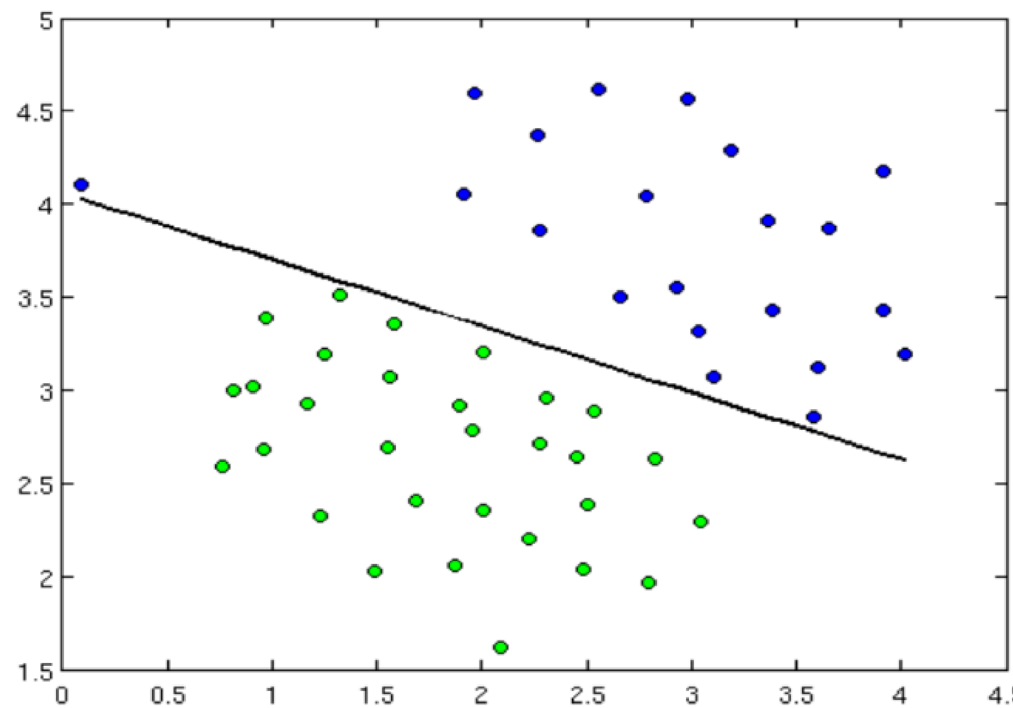
Геометрическая интерпретация

Фактически строится разделяющая гиперплоскость, разделяющая классы.

Например, если признаков 2,

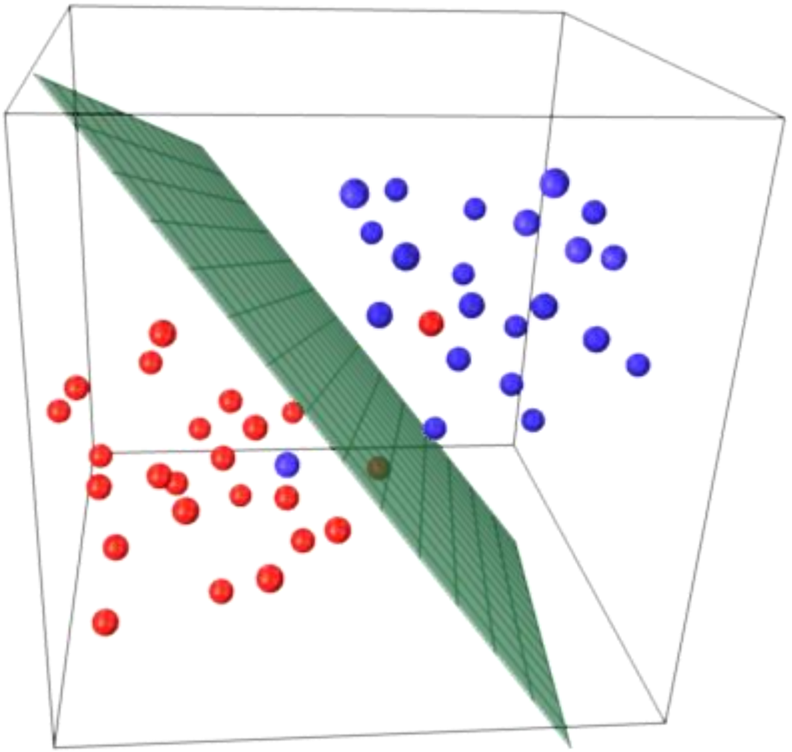
то строится

прямая.



https://habrahabr.ru/company/iticapital/blog/279963/

если признаков 3 штуки, то строится гиперплоскость в пространстве.



https://medium.com/data-science-group-iitr/support-vector-machines-svm-unraveled-e0e7e3ccd49b

Когда линейный классификатор ошибается?

Вспомним правило классификации:

если $w_1x_1+w_2x_2+\ldots+w_nx_n+w_0>0$, то объект относим к классу 1 (иначе к классу -1).

Когда модель допускает ошибку на тренировочной выборке?
Это происходит в 2-х случаях
1) когда $w_1x_1+w_2x_2+\ldots+w_nx_n > 0$, но объект из класса -1
2) когда $w_1x_1+w_2x_2+\ldots+w_nx_n<0$ но объект из класса 1

Когда модель ошибается?

Иными словами, модель ошибается на объекте тренировочной выборки, если

$$M=y(w_1x_1+w_2x_2+\ldots+w_nx_n+w_0)<0$$

здесь y – значение целевого признака объекта.

Величина M называется отступом.

Что минимизировать?

Введем обозначение

$$[M < 0] = \begin{cases} 1, \text{если } M < 0 \\ 0, \text{если } M > 0 \end{cases}$$

тогда для объектов тренировочной выборки нужно минимизировать функцию

$$\sum_{i=1}^{n} [M_i < 0]$$

где M_i – отступ i-го объекта тренировочной выборки,
y_i – истинная метка класса i-го объекта тренировочной выборки.

Что минимизировать?

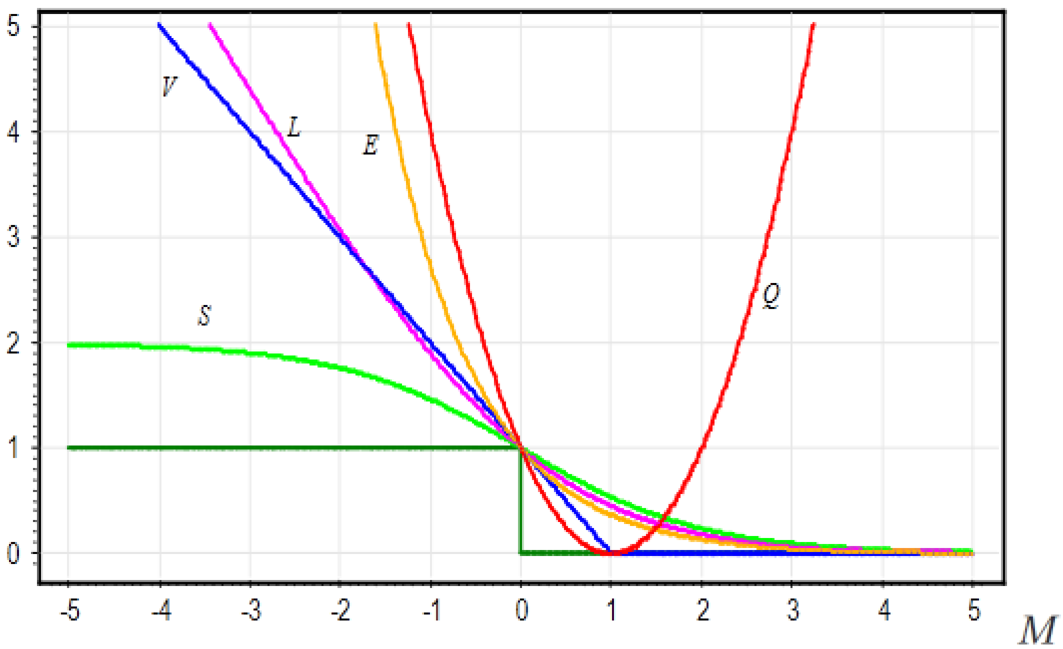
Есть проблема: эта функция

$$\sum_{i=1}^{n} [M_i < 0]$$

не дифференцируемая (нельзя взять производную). Искать минимум таких функций – это мазохизм.

Идея: найти дифференцируемую функцию *f*, которая приближенно равна [M<0]. Желательно также, чтобы выполнялось неравенство [M<0]<f (мажорирование).

И таких функций несколько



Все эти функции мажорируют функцию [M<0] (темно-зеленого цвета)

$$
\begin{aligned}
Q(M) &= (1-M)^2 && \text{— квадратичная;}\\
V(M) &= (1-M)_+ && \text{— кусочно-линейная;}\\
S(M) &= 2(1+e^M)^{-1} && \text{— сигмоидная;}\\
L(M) &= \log_2(1+e^{-M}) && \text{— логистическая;}\\
E(M) &= e^{-M} && \text{— экспоненциальная.}
\end{aligned}
$$

http://ru.learnmachinelearning.wikia.com/wiki

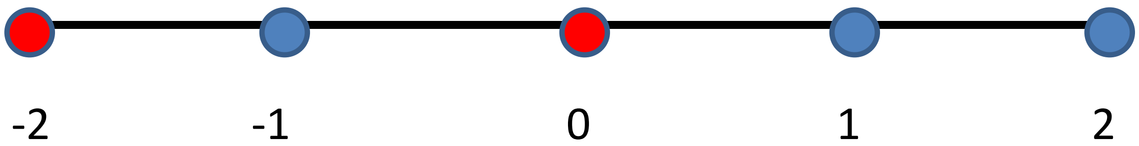
Схема построения линейного классификатора

1. Выбрать мажорирующую [M<0] функцию.
2. Составить выражение для минимизации.
3. Найти точку минимума выражения. Она даст оптимальные значения весов w_i.

Сейчас разберем пример, который всё это демонстрирует.

Пример

Дана тренировочная выборка из 5 объектов.

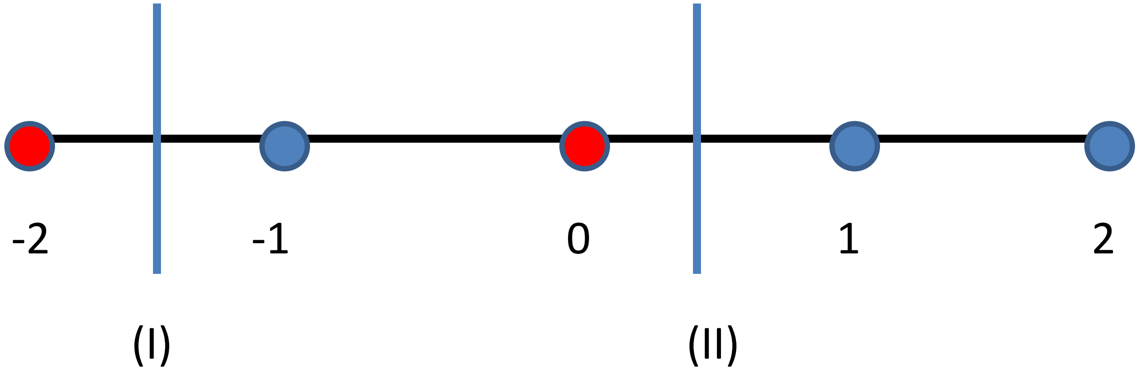


	x	y
A	-2	-1
B	-1	1
C	0	-1
D	1	1
E	2	1

Эта выборка линейно не разделима, то есть нельзя найти линейный классификатор с нулевой ошибкой на тренировочной выборке.
Но все-таки: как должна пройти разделяющая поверхность для этих объектов?

Пример

Очевидно (для человека, но не для машины), что есть два оптимальных варианта для расположения разделяющей поверхности.



	x	y
A	-2	-1
B	-1	1
C	0	-1
D	1	1
E	2	1

Оба варианта допускают 1 ошибку на тренировочной выборке. Посмотрим, что будет построено в результате вычислений…

Расчет примера

Отступы объектов равны:

$M_1=-1(-2w_1+w_0)$,

$M_2=1(-1w_1+w_0)$,

$M_3=-1(0w_1+w_0)$,

$M_4=1(1w_1+w_0)$,

$M_5=1(2w_1+w_0)$,

В качестве мажорирующей функции возьмём $Q(M)=(1-M)^2$.

	x	y
A	-2	-1
B	-1	1
C	0	-1
D	1	1
E	2	1

Расчет примера

Тогда нужно минимизировать выражение:

$L=(1+(-2w_1+w_0))^2+(1-(-1w_1+w_0))^2+$

$(1+1(0w_1+w_0))^2+(1-(1w_1+w_0))^2+(1-(2w_1+w_0))^2=$

$(1-2w_1+w_0)^2+(1+w_1-w_0)^2+(1+w_0)^2+(1-w_1-w_0)^2+(1-2w_1-w_0)^2=$

$10w_1^2+5w_0^2-8w_1-2w_0+5$

Расчет примера

Находим частные производные для

$L=10w_1^2+5w_0^2-8w_1-2w_0+5$

$$\frac{\partial L}{\partial w_1} = 20w_1 - 8, \frac{\partial L}{\partial w_0} = 10w_0 - 2$$

Приравнивая производные к нулю, находим $w_1=0.4$, $w_0=0.2$.

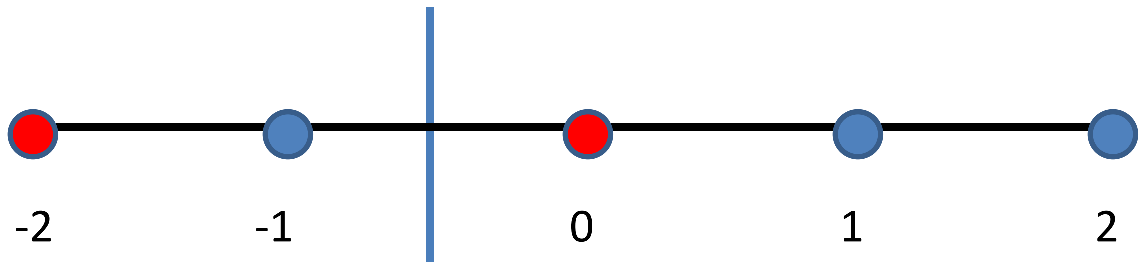
Расчет примера

Следовательно, классификатор будет таким:

если *0.4x+0.2>0, то объект из класса 1 (иначе из класса -1).*

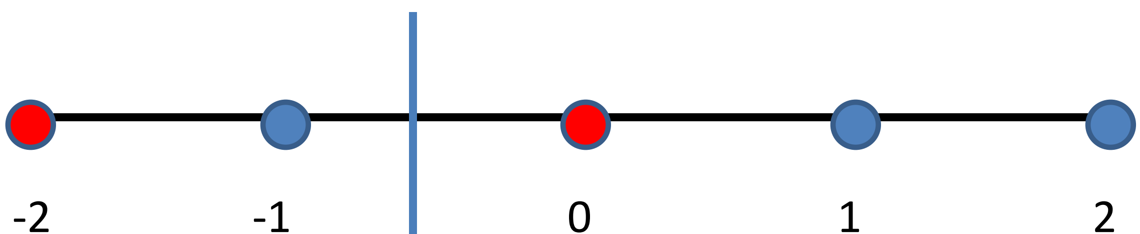
Что эквивалентно*: если x>-0.5, то объект из класса 1 (иначе из класса -1).*

То есть был найден вот такой раздел:



Объяснение такого результата

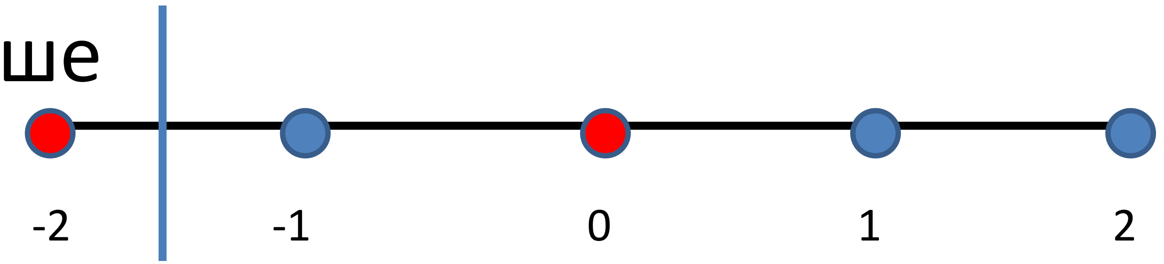
Может показаться, что найденное алгоритмом решение очень плохое, так как допускается 2 ошибки на тренировочной выборке.

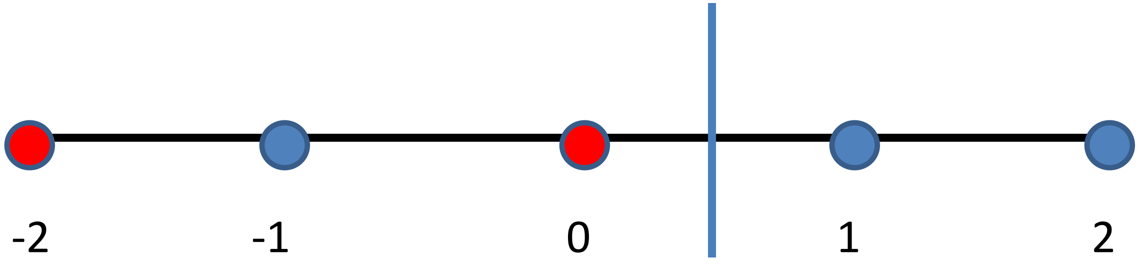


Однако надо понимать, что «под капотом» алгоритма минимизировалось не количество ошибок, а сумма квадратов ошибочных отступов.

Объяснение такого результата

И поэтому полученное решение (с точки зрения теории) не хуже чем обсуждаемые выше





Нахождение минимума функции с помощью градиентного спуска

Как происходит поиск минимума функции

Разобранный ранее пример очень простой, и найти точку минимума (оптимальные значения весов w_i) несложно.

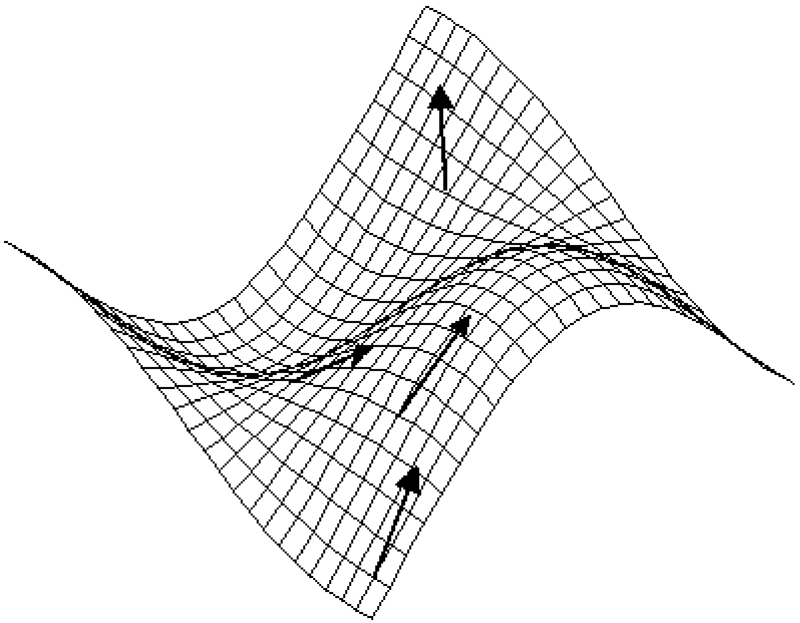
А как быть в общей ситуации?

Как ищут точку минимума функции от нескольких переменных?

Минимум ищут с помощью градиента

Функция от нескольких переменных представляется поверхностью.

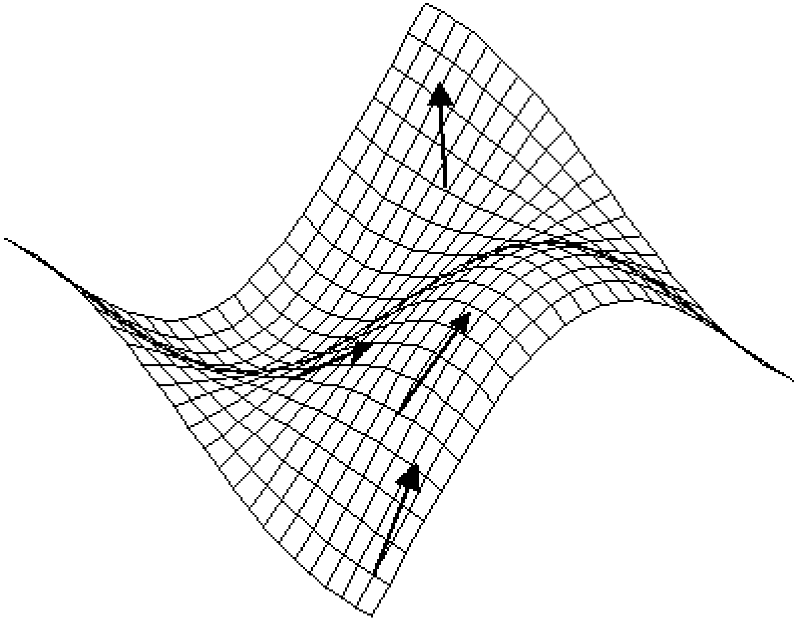
Градиент для точки поверхности – это направление наиболее сильного возрастания функции.



http://citadel.sjfc.edu/faculty/kgreen/vector/block2/pder/node8.html

Идём против градиента!

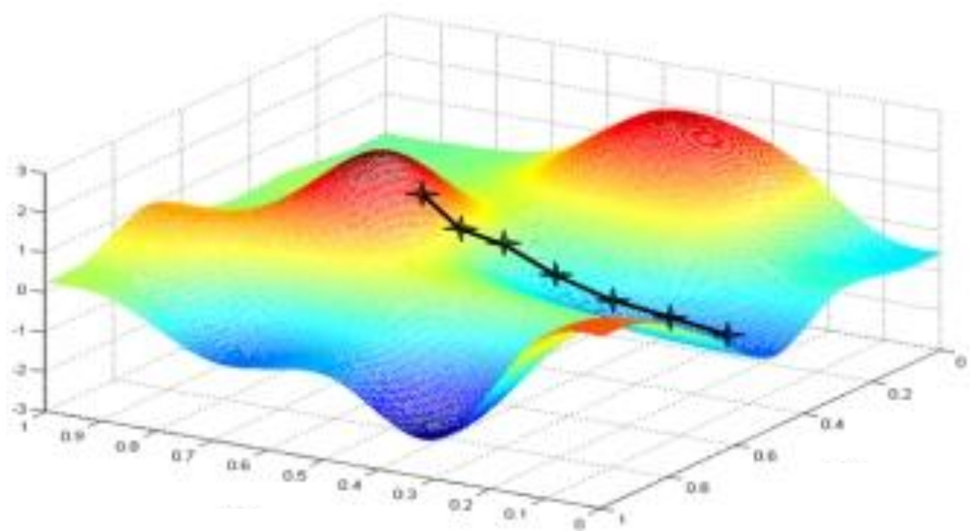
Поскольку у функции ищется минимум, то нужно идти по поверхности в направлении обратном градиенту.



http://citadel.sjfc.edu/faculty/kgreen/vector/block2/pder/node8.html

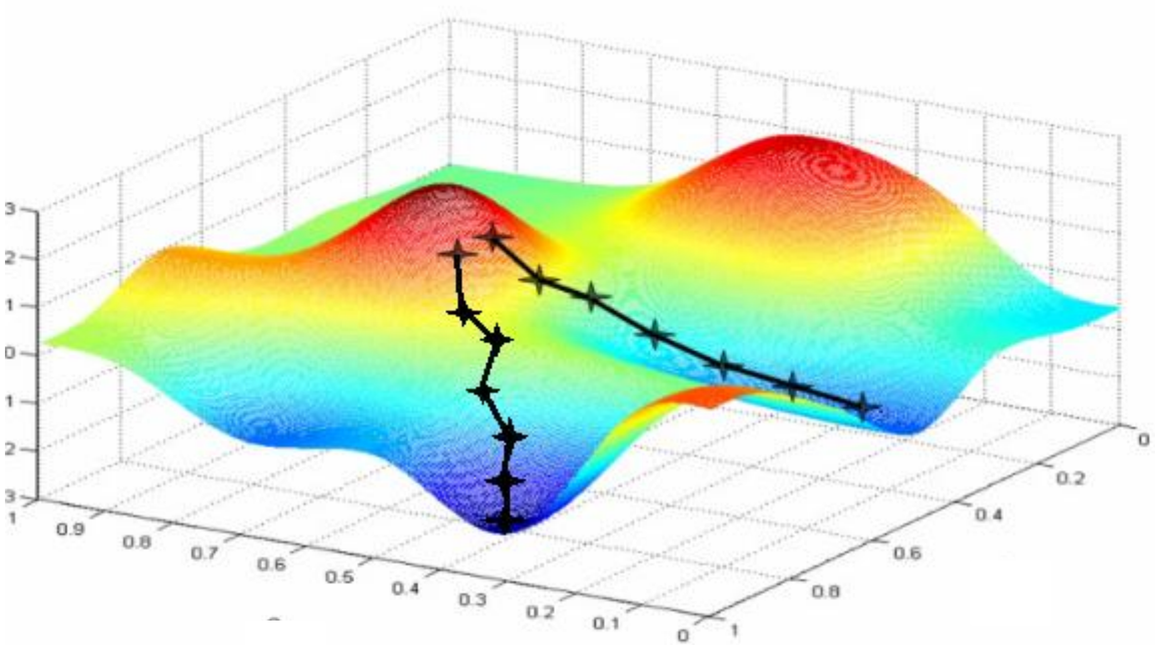
Идём против градиента!

Спуск нужно осуществлять за несколько шагов. В точке минимума градиент станет равен 0 – дальше идти некуда!



Недостаток №1

Первая точка с поверхности выбирается случайно. НО от нее существенно зависит результат!

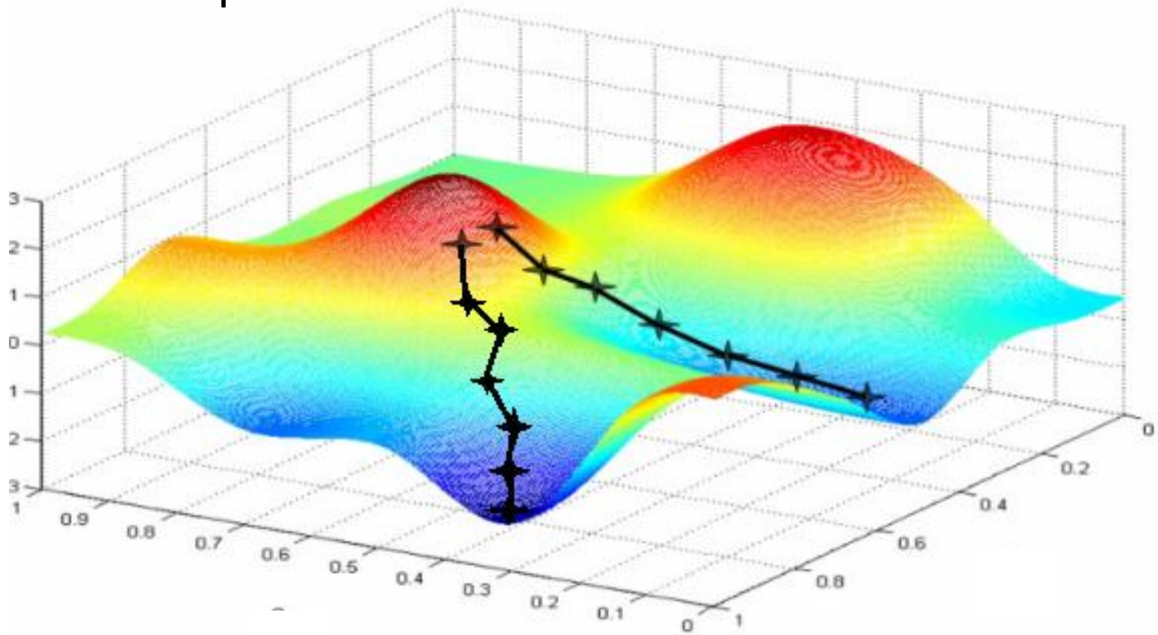


http://blog.datumbox.com/tuning-the-learning-rate-in-gradient-descent/

Недостаток №2

Длина шага называется скоростью обучения.

Слишком маленькая скорость обучения заставляет алгоритм сходиться очень долго и застревать в локальных минимумах, слишком большая — «пролетать» узкие глобальные минимумы или вовсе работать бесконечное время.

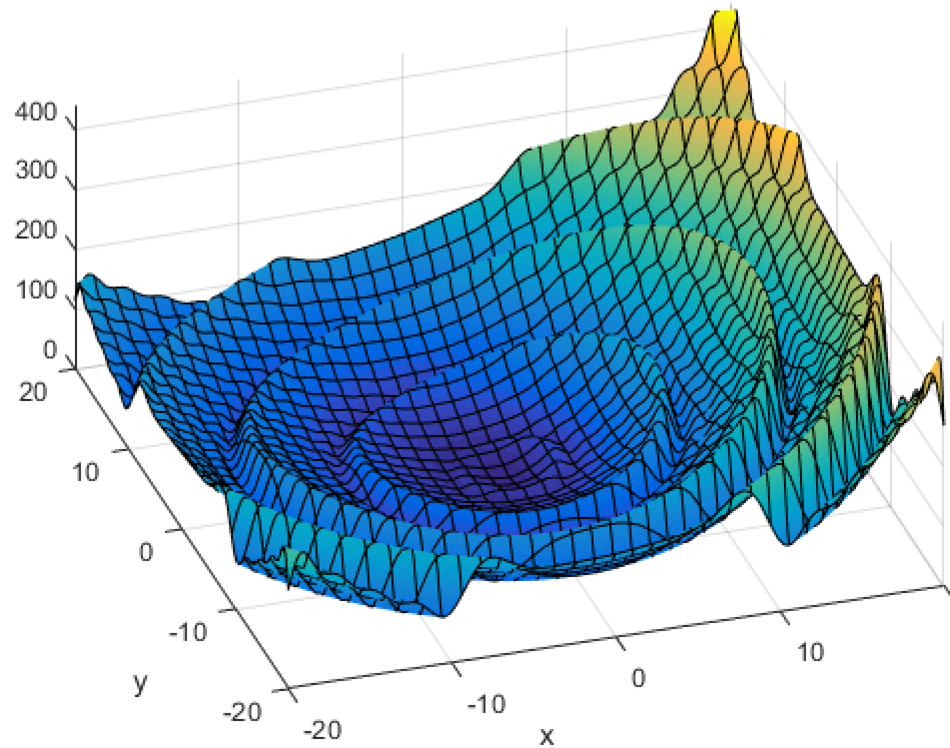


http://blog.datumbox.com/tuning-the-learning-rate-in-gradient-descent/

Существует много модификаций метода «идти против градиента»

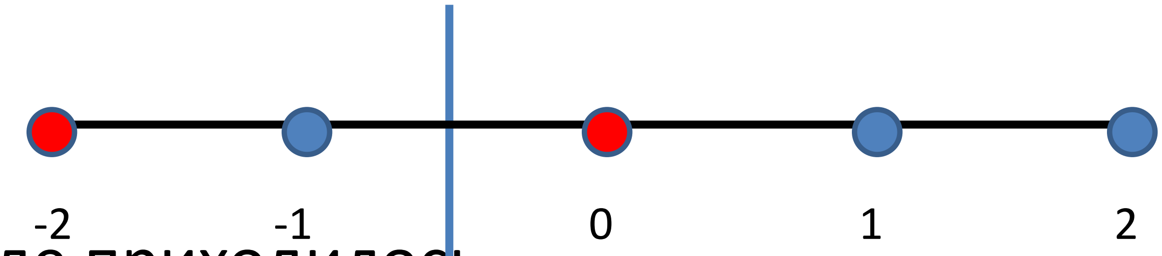
Эти алгоритмы используют разные эвристики, позволяющие добраться до глобального минимума.

Некоторые методы намеренно замедляют процесс спуска – лишь бы не попасть в локальный минимум.

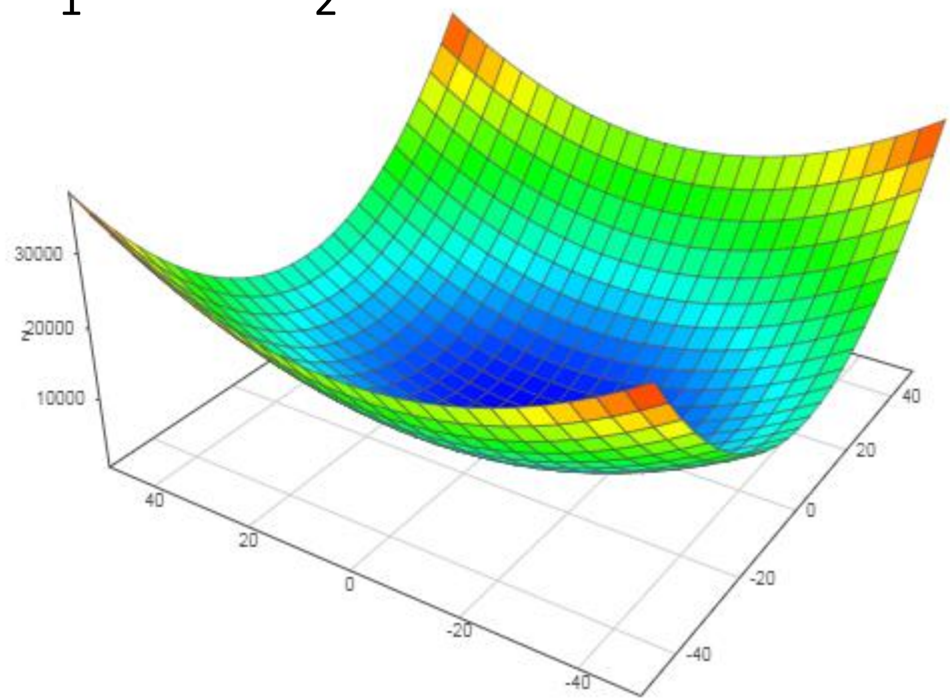


Решение старого примера с помощью градиента.

Ранее был пример:



где приходилось минимизировать выражение $L=10w_1^2+5w_0^2-8w_1-2w_0+5$ (здесь нарисована соотв. ему поверхность)



Идем против градиента

L=10w_1^2+5w_0^2-8w_1-2w_0+5

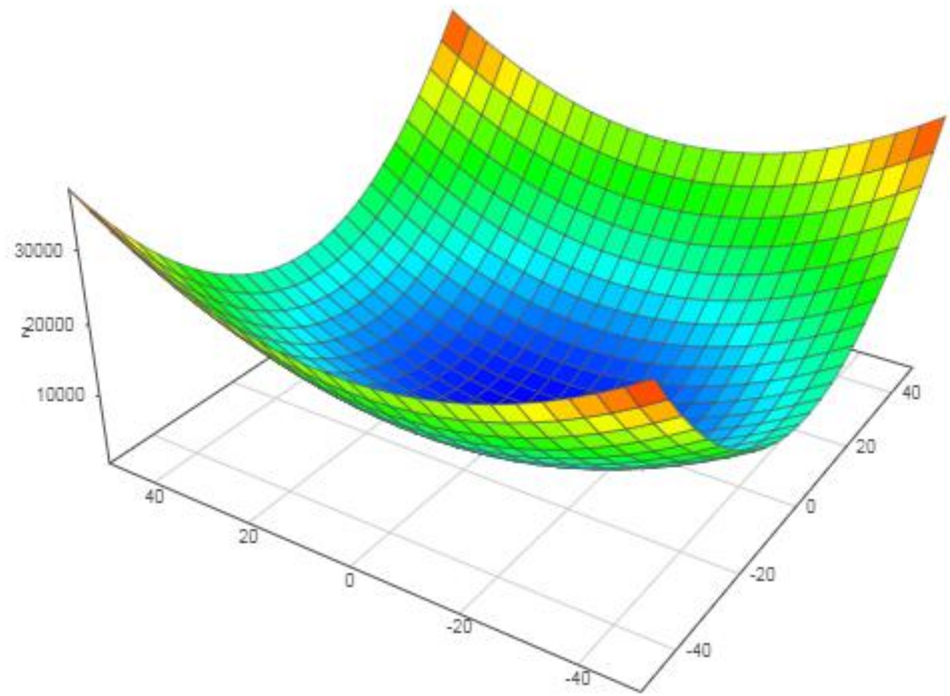
$$\frac{\partial L}{\partial w_1} = 20w_1 - 8, \frac{\partial L}{\partial w_0} = 10w_0 - 2$$

градиент – это вектор

(20w_1-8,10w_0-2)

Пусть (1,1) – начало спуска.

Зададим длину шага 0.05



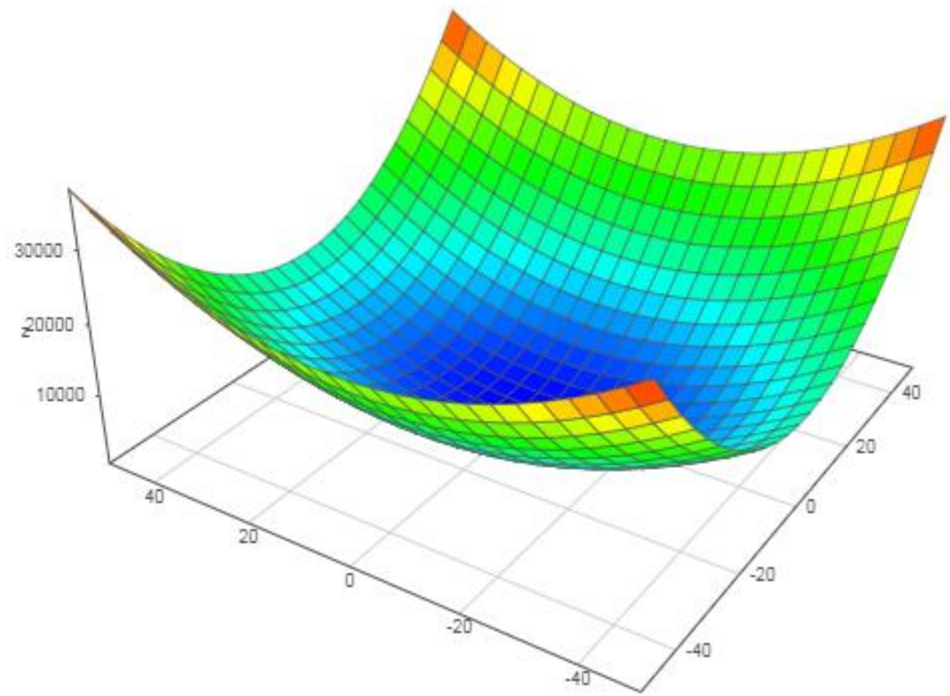
Идем против градиента

Подставляя (1,1) в градиент, получаем *(12,8).* Значит, нужно идти в направлении *(-12,-8). Тогда*

(1,1)+0.05(-12,-8)=*

(0.4,0.6)

Новая точка стала ближе к точке минимума.



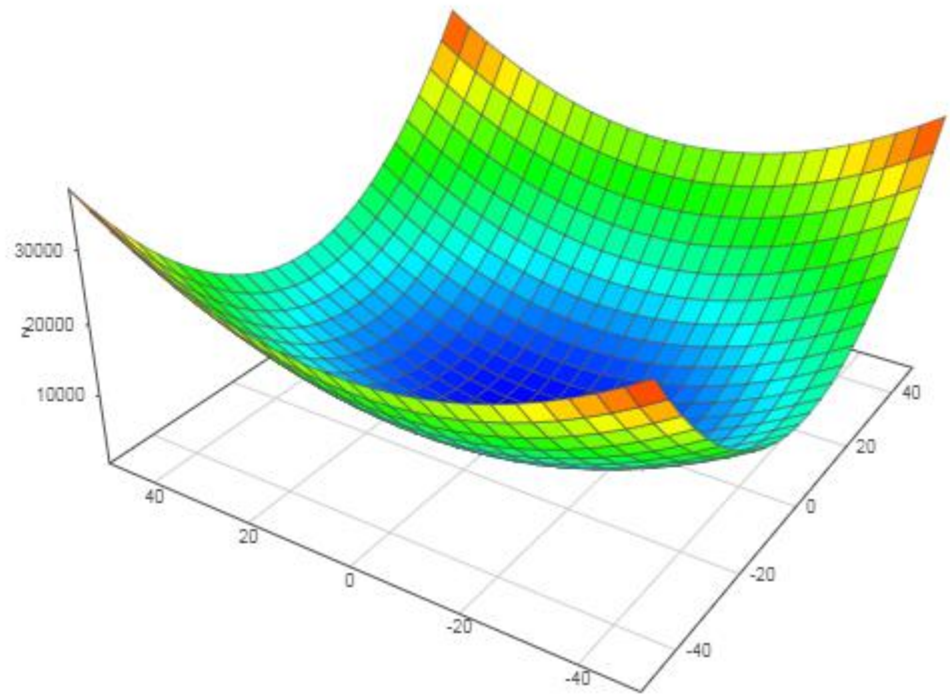
Идем против градиента

2 итерация: подставляя (0.4,0.6) в градиент, получаем (0,4). Значит, нужно идти в направлении *(0,-4). Тогда*

$(0.4,0.6)+0.05*(0,-4)=$

$(0.4,0.4)$

Новая точка стала еще ближе к точке минимума. Процесс продолжается…



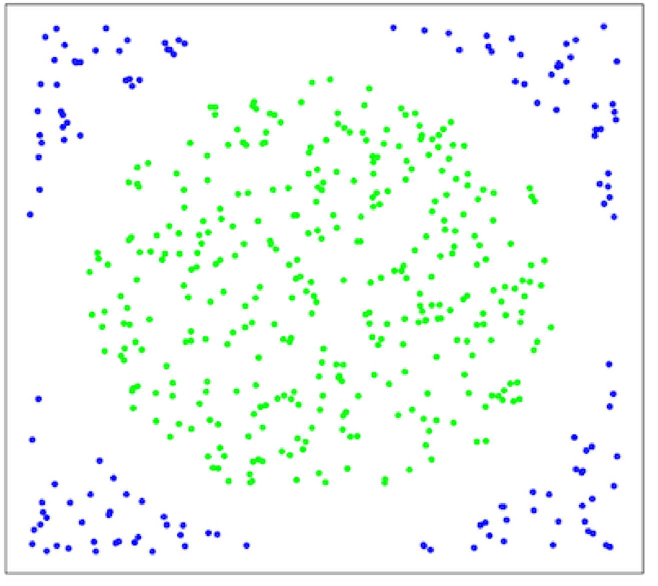
Градиентные методы: резюме

1. Поиск минимума с помощью градиента – это итерационный процесс (нужно задавать максимальное число итераций).
2. Процесс может завести вас не в глобальный, а в локальный минимум.
3. Длина шага – нежная штука. Многие методы модифицируют длину шага на каждой итерации.

Kernel trick

Kernel trick позволяет преобразовать данные так, что классы становятся линейно разделимыми, а затем построить линейный классификатор.

Эта выборка линейно не разделима.

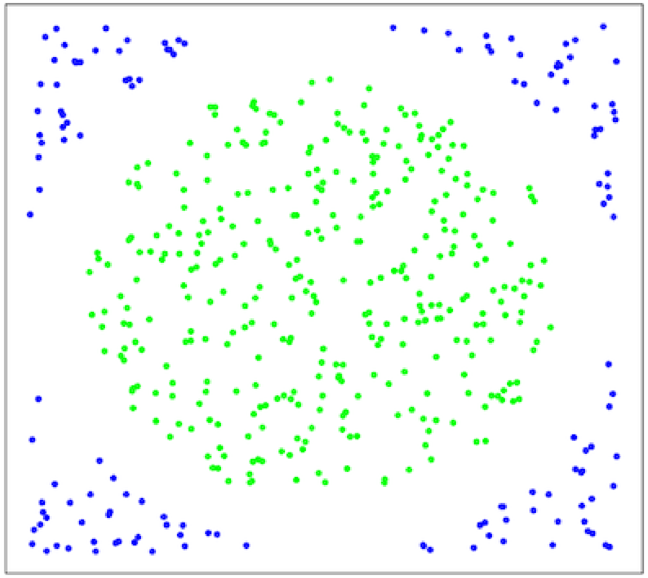


https://habrahabr.ru/post/105220/

Kernel trick

Чтобы классы стали разделимыми, классифицируемые объекты вкладывают в пространство большей размерности.

Как перенести эти объекты в 3-х мерное пространство, чтобы классы стали разделимы?



https://habrahabr.ru/post/105220/

Kernel trick

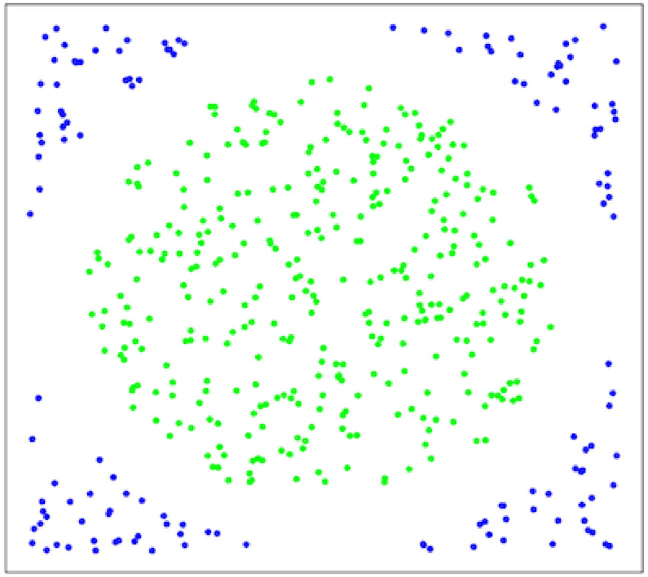
Представьте, что эти точки нарисованы на плёнке, которую можно растягивать.

Вставайте ногами в центр и тяните всеми четырьмя руками за углы.

Есть математические формулы,

которые осуществляют

такое преобразование.

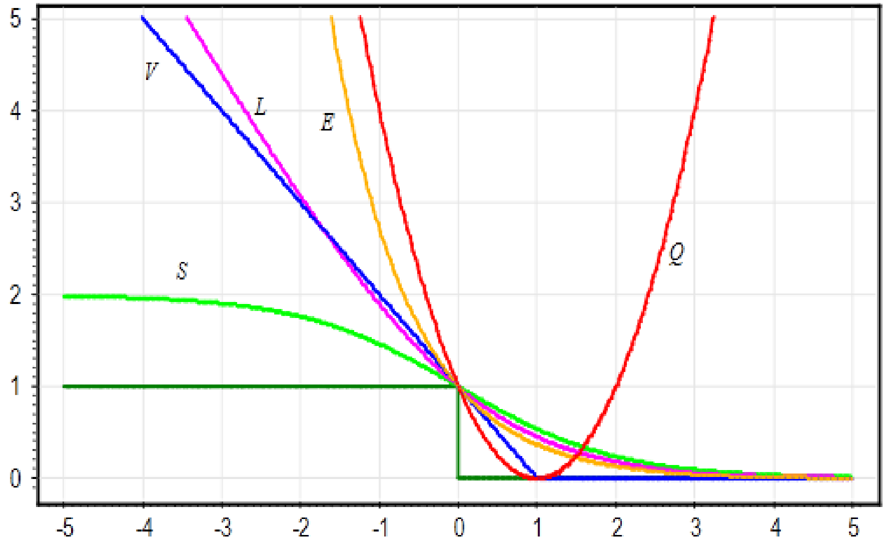


https://habrahabr.ru/post/105220/

Важный тип лин. классификатора: Метод опорных векторов (support vector machine SVM)

Формально SVM – это…

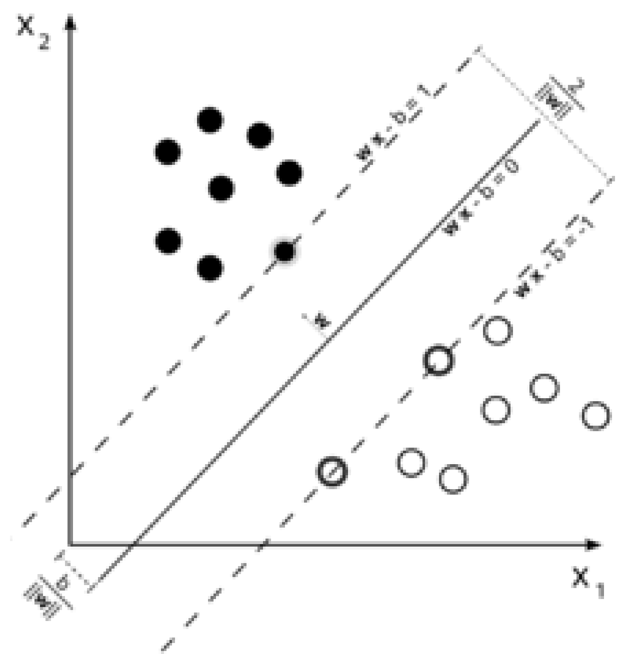
Обычный линейный классификатор, где [M<0] мажорируется с помощью $V(M)=(1-M)_+$



$$Q(M) = (1-M)^2 \quad \text{— квадратичная;}$$
$$V(M) = (1-M)_+ \quad \text{— кусочно-линейная;}$$
$$S(M) = 2(1+e^{M})^{-1} \quad \text{— сигмоидная;}$$
$$L(M) = \log_2(1+e^{-M}) \quad \text{— логистическая;}$$
$$E(M) = e^{-M} \quad \text{— экспоненциальная.}$$

Геометрическая интерпретация SVM

Поиск оптимальных параметров w_i в случае SVM эквивалентен поиску гиперплоскости, разделяющей классы с максимальным отступом.



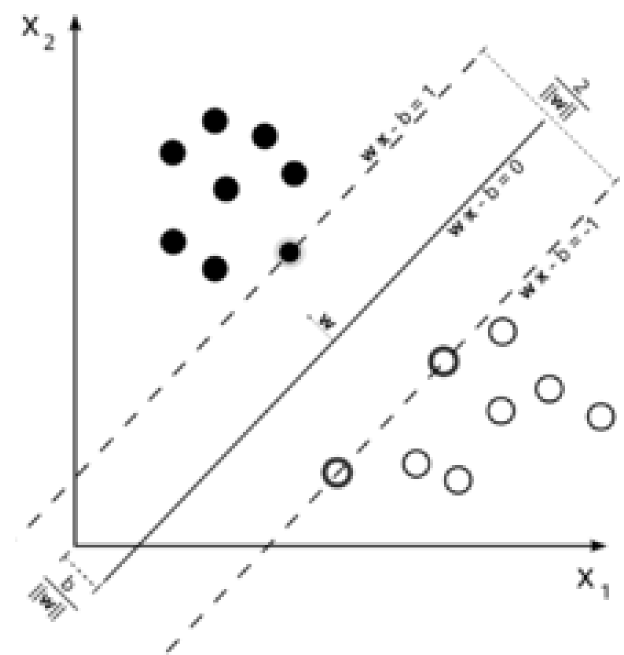
https://en.wikipedia.org/wiki/Support_vector_machine

Геометрическая интерпретация SVM

Если классы не разделяются линейно, то в SVM применяют либо

а) kernel trick

б) вводят величину штрафа за каждый неправильно классифицированный объект



https://en.wikipedia.org/wiki/Support_vector_machine

Достоинства SVM

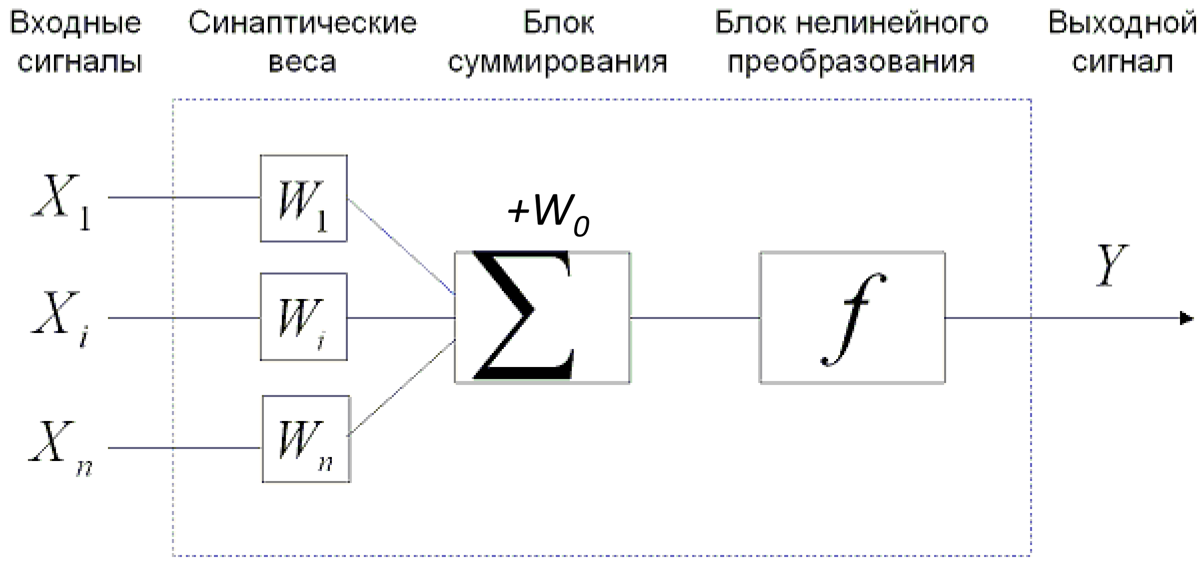
1) Быстрота построения модели. Градиентный спуск гарантировано не зациклится и найдет глобальный минимум.
2) Хорошая устойчивость к выбросам (фактически модель строится по объектам, лежащим вблизи раздела).
3) Доказано, что SVM не слабее любой двуслойной нейросети.

Нейронные сети
(как композиция линейных
классификаторов)

Что такое искусственный нейрон?

Искусственный нейрон очень похож на линейный классификатор. Получая значения признаков объекта $x_1,\ldots x_n$, нейрон выдает ответ

$$f(w_1x_1+\ldots+w_nx_n+w_0)$$

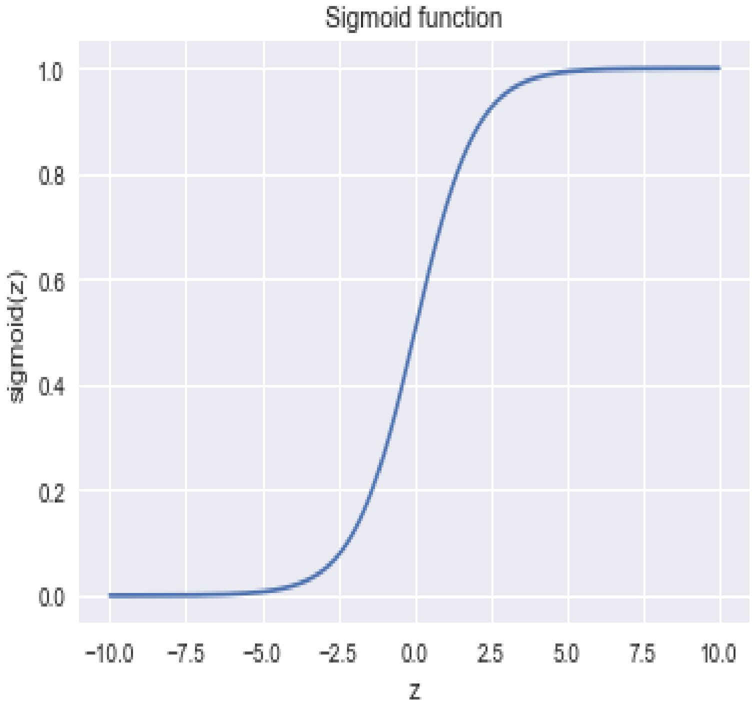


http://nano-e.ucoz.ru/publ/publikacii_studentov_mifi/referaty/osnovnye_paradigmy_nejronnykh_setej/8-1-0-211

Что такое искусственный нейрон?

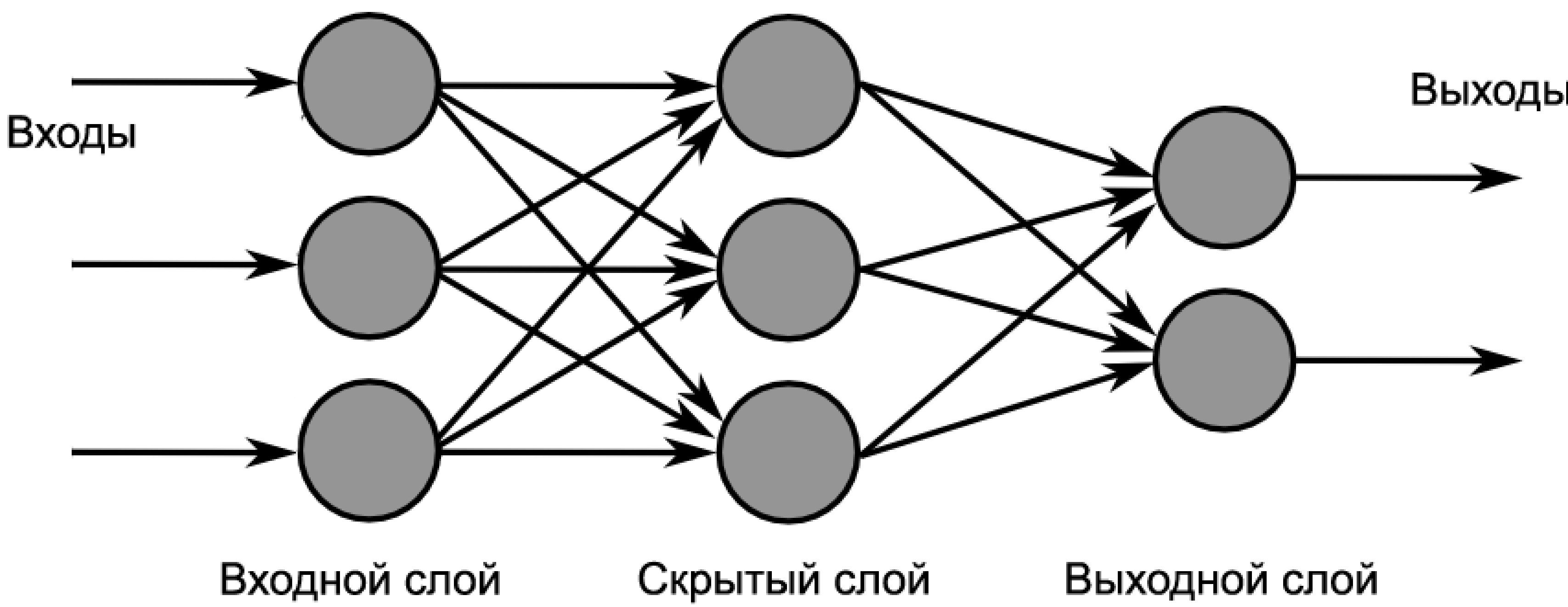
Функция f называется функцией активации. Например (и очень часто), в качестве f берут сигмоид

$$\sigma(z) = \frac{e^z}{e^z + 1}$$



Нейронная сеть = композиция нейронов

Несколько нейронов можно соединить друг с другом так, что выходной сигнал одних нейронов являлся входным сигналом для других. Нейроны в сети могут иметь различные функции активации.



https://neuralnet.info/chapter/%D0%BE%D1%81%D0%BD%D0%BE%D0%B2%D1%8B-%D0%B8%D0%BD%D1%81/

Какие параметры сети задаёт человек?

1. Топология сети, то есть число слоев, число входных и выходных нейронов…
2. Функции активации.

Алгоритм лишь находит оптимальные веса.

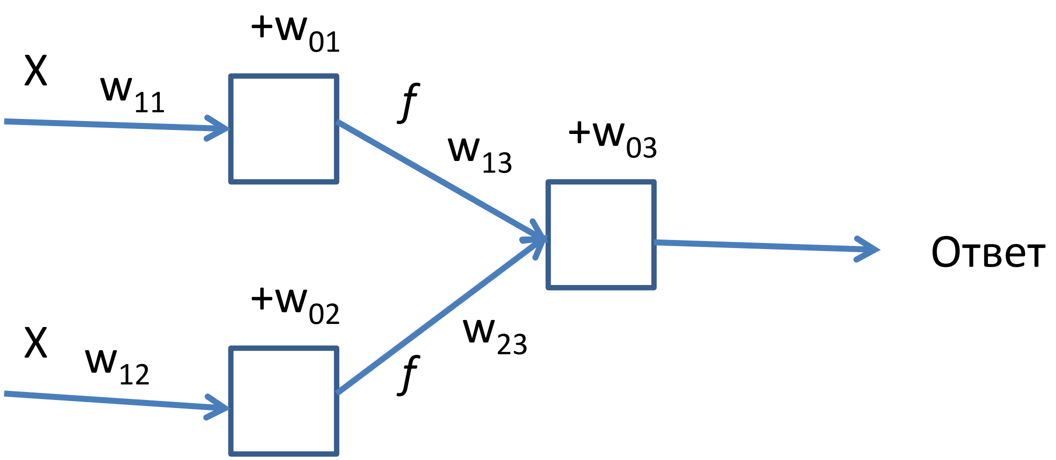
Пример

Будем решать задачу регрессии для данных:

с помощью нейросети вида

(f - сигмоид):

Объект	X	Y
A	-1	1
B	0	0
C	1	1
D	2	4



Пример

Для входа X нейросеть выдаст ответ:

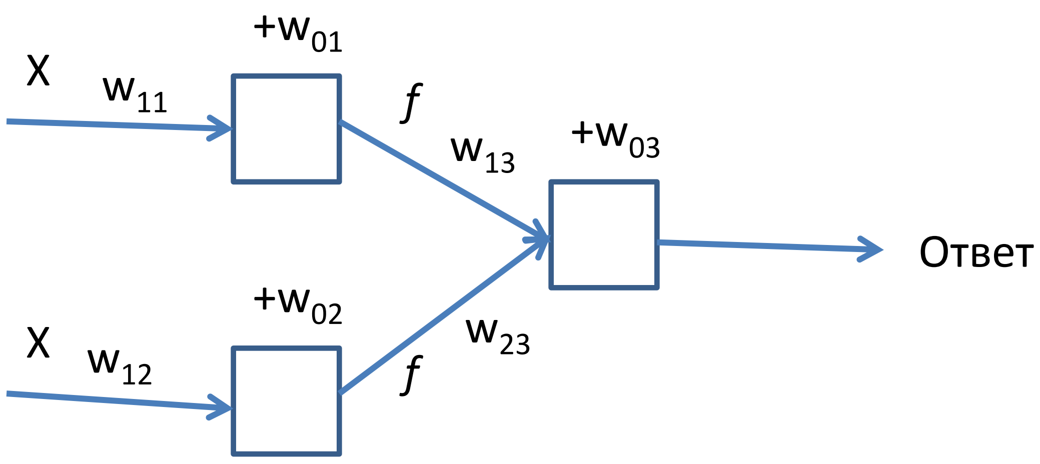
$F = w_{13} f(w_{11}x + w_{01}) + w_{23} f(w_{12}x + w_{02}) + w_{03}$

Тогда, например, квадрат ошибки

для объекта D равен

$(w_{13} f(2w_{11} + w_{01}) + w_{23} f(2w_{12} + w_{02}) + w_{03} - 4)^2$

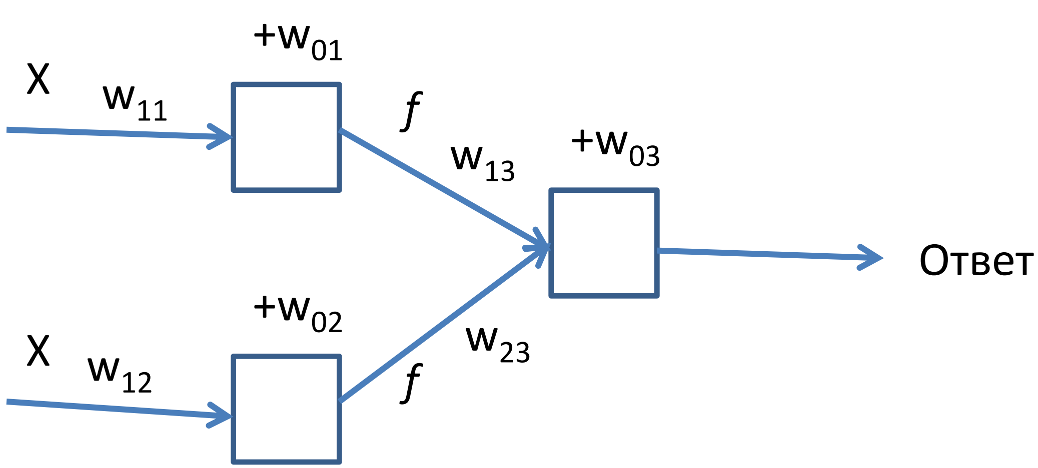
Объект	X	Y
A	-1	1
B	0	0
C	1	1
D	2	4



Пример

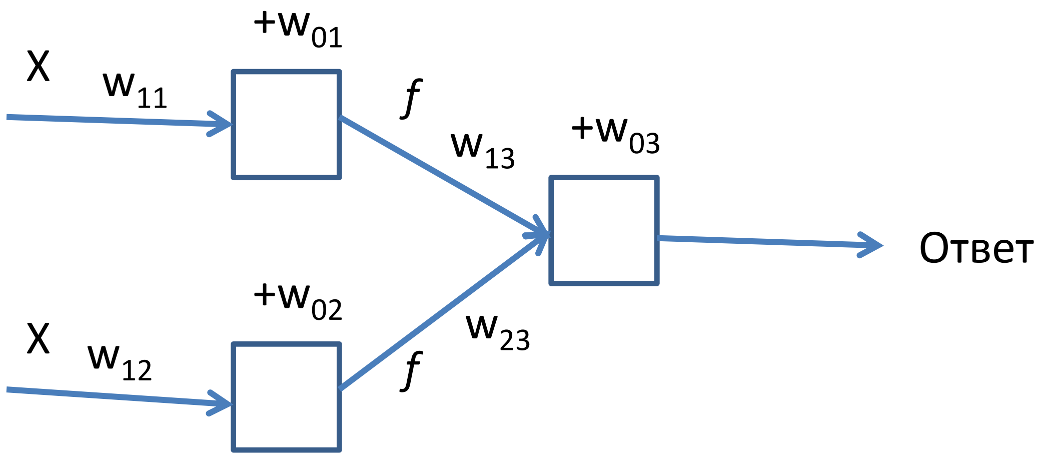
Осталось составить сумму L квадратов ошибок по всем объектам тренировочной выборки, и найти точку минимума этого выражения.

Значения весов w_{ij} в точке минимума дают оптимальные значения параметров сети.



Пример

Точку минимума для L можно найти с помощью градиентного спуска. Одной из адаптаций градиентного для нейронных сетей является алгоритм обратного распространения ошибки.



Нейросети для классификации

Для классификации могут применяться сети с несколькими выходными нейронами.

В данном случае ответ i-го выходного нейрона – уверенность в принадлежность i-му классу.

Виды нейронных сетей

Поскольку топологию сети определяет человек, то возникают различные классы нейронных сетей.

1. Сверточные (предназначены для распознавания изображений).
2. Реккурентные (для видео)
3. …