

ДОНСКОЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ  
УНИВЕРСИТЕТ

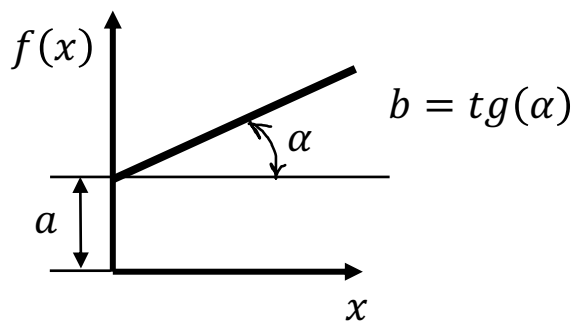
КАФЕДРА АВТОМАТИЗАЦИЯ ПРОИЗВОДСТВЕННЫХ ПРОЦЕССОВ

ЛЕКЦИЯ № 04  
**Линейные модели**

СОСТАВИТЕЛЬ:            КАНД. ТЕХН. НАУК    БЫКАДОР В.С.

# Общие положения линейных моделей

Модели которые можно интерпретировать в терминах прямых или плоскостей обычно называют линейными. Прямая выражается уравнением вида:



$$f(x) = a + b \cdot x$$

где  $a$  - свободный член  
 $b$  - угловой коэффициент

Если  $x$  рассматривать как параметр, то в задачах машинного обучения мало найдётся задач распознавания с одним параметром. В основном, в задачах машинного обучения присутствует несколько параметров, характеризующих объект.

# Общие положения линейных моделей

Пусть  $d$  это количество параметров, характеризующих объект машинного обучения, тогда все признаки можно представить в виде вектора  $\mathbf{x}$  :

$$\mathbf{x} = \begin{pmatrix} x_1 \\ x_2 \\ \dots \\ x_d \end{pmatrix} \Rightarrow f(\mathbf{x}) = a + b_1 \cdot x_1 + b_2 \cdot x_2 + \dots + b_d \cdot x_d = a + \mathbf{b} \cdot \mathbf{x} \quad \text{где } \mathbf{b} = (b_1 \ b_2 \ \dots \ b_d)$$
$$f(\mathbf{x}) = a + \mathbf{b} \cdot \mathbf{x} = 0 \Rightarrow \text{получим плоскость в пространстве } \mathbb{R}^d \text{ (гиперплоскость) } \perp \mathbf{b} .$$

*(для справки)*

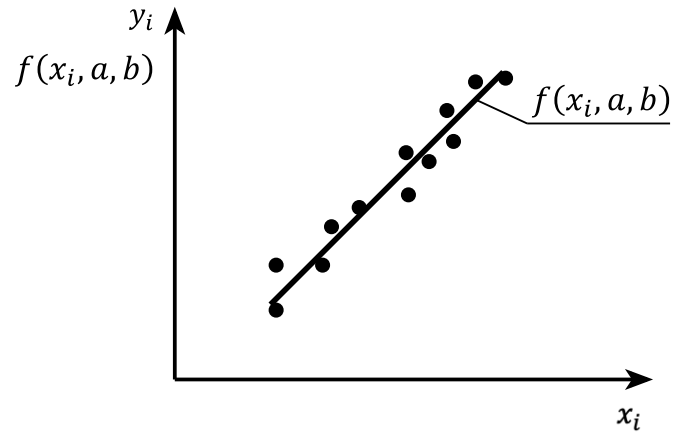
Иногда, для удобства и большей общности, переходят к так называемым однородным координатам, тогда уравнение  $f(\mathbf{x})$  будет иметь вид:

$$f(\mathbf{x}) = \mathbf{b}^\circ \cdot \mathbf{x}^\circ$$
$$\mathbf{b}^\circ = (a \ b_1 \ b_2 \ \dots \ b_d) \quad \mathbf{x}^\circ = \begin{pmatrix} 1 \\ x_1 \\ x_2 \\ \dots \\ x_d \end{pmatrix}$$

# Одномерный метод наименьших квадратов

Одномерный метод наименьших квадратов (МНК) часто используется для регрессии, то есть прогнозирования в линейных моделях.

В общем случае МНК строится на основе минимизации функции вида:



$$Q(a, b) = \left( \sum_{i=0}^n (y_i - f(x_i, a, b))^2 \right) \rightarrow \min$$

где  $y_i$  - экспериментальные данные (тестовые данные);

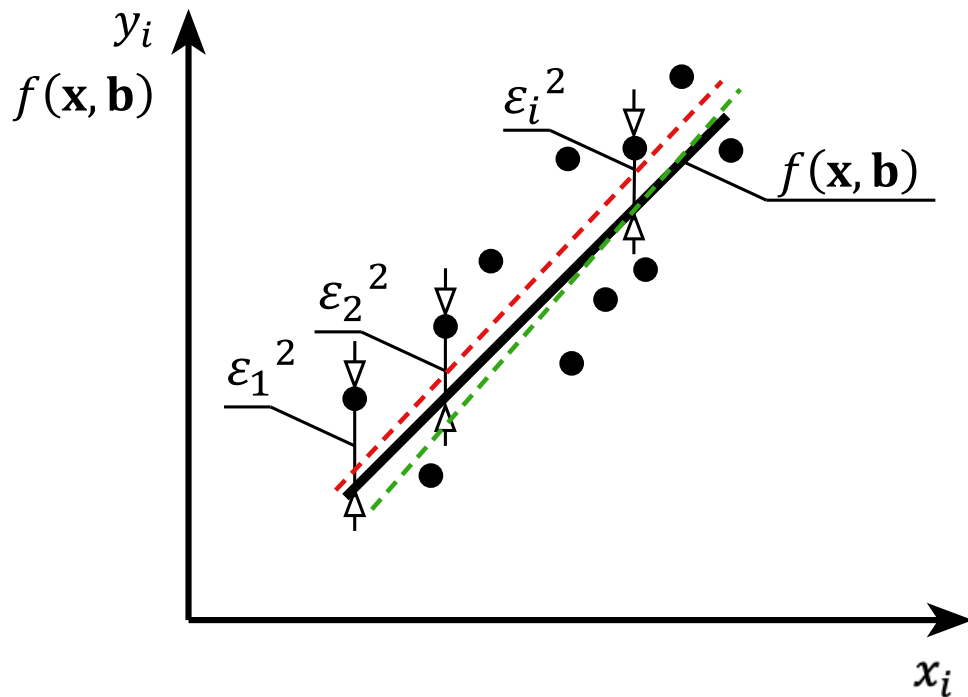
$f(x_i, a, b) = a + b \cdot x_i$  - аппроксимирующая линейная функция с неизвестными коэффициентами  $a$  и  $b$ .

(для справки)

В общем случае,  $f(x_i, a, b)$  может быть любой функцией, не обязательно линейной.  $f(x_i, a, b)$  может быть полиномом  $n$ -ой степени или вообще экспонентой. Но так как мы рассматриваем линейные модели, то функция  $f(x_i, a, b)$  является линейной.

# Одномерный метод наименьших квадратов

Задача МНК найти такие значения неизвестных коэффициентов  $a$  и  $b$ , чтобы сумма квадратов ошибок была минимальной. Только в этом случае аппроксимирующая функция  $f(x_i, a, b)$  будет наилучшим образом описывать экспериментальные данные.



$$Q(a, b) = \left( \sum_{i=0}^n \varepsilon^2 \right) \rightarrow \min$$

где  $\varepsilon = y_i - f(x_i, a, b) = y_i - (a + b \cdot x_i)$  - ошибка между экспериментальными данными и аппроксимирующей функцией.

# Одномерный метод наименьших квадратов

Для минимизации функции  $Q(a, b) \rightarrow \min$  требуется найти частные производные от неё по каждому коэффициенту и приравнять их нулю. После этого необходимо разрешить полученную систему уравнений.

$$\begin{cases} \frac{\partial Q(a, b)}{\partial a} = 0 \\ \frac{\partial Q(a, b)}{\partial b} = 0 \end{cases}$$

$$\begin{cases} \frac{\partial}{\partial a} \left( \sum_{i=0}^n (y_i - (a + b \cdot x_i))^2 \right) = -2 \sum_{i=0}^n (y_i - (a + b \cdot x_i)) = 0 \Rightarrow \hat{a} \\ \frac{\partial}{\partial b} \left( \sum_{i=0}^n (y_i - (a + b \cdot x_i))^2 \right) = -2 \sum_{i=0}^n (y_i - (a + b \cdot x_i)) \cdot x_i = 0 \Rightarrow \hat{b} \end{cases}$$

Получим оценочные значения искомых коэффициентов аппроксимирующей функции.

# Многомерный метод наименьших квадратов

Если признаков  $x_i$  объекта несколько, то необходимо переходить к многомерному методу наименьших квадратов. Для удобства записи будем рассматривать задачу в однородных координатах и использовать матричную нотацию.

Тогда:

$$\mathbf{y} = \mathbf{w} \cdot \mathbf{X}$$

где:

$$\mathbf{y} = \begin{pmatrix} y_1 \\ y_2 \\ \dots \\ y_n \end{pmatrix} \quad \mathbf{w} = \begin{pmatrix} a \\ b_1 \\ b_2 \\ \dots \\ b_d \end{pmatrix} \quad \mathbf{X} = \begin{pmatrix} 1 & x_1 \\ 1 & x_2 \\ 1 & \dots \\ 1 & x_d \end{pmatrix}$$
$$\Downarrow$$
$$\begin{pmatrix} y_1 \\ y_2 \\ \dots \\ y_n \end{pmatrix} = \begin{pmatrix} a \\ b_1 \\ b_2 \\ \dots \\ b_d \end{pmatrix} \cdot \begin{pmatrix} 1 & x_1 \\ 1 & x_2 \\ 1 & \dots \\ 1 & x_d \end{pmatrix}$$

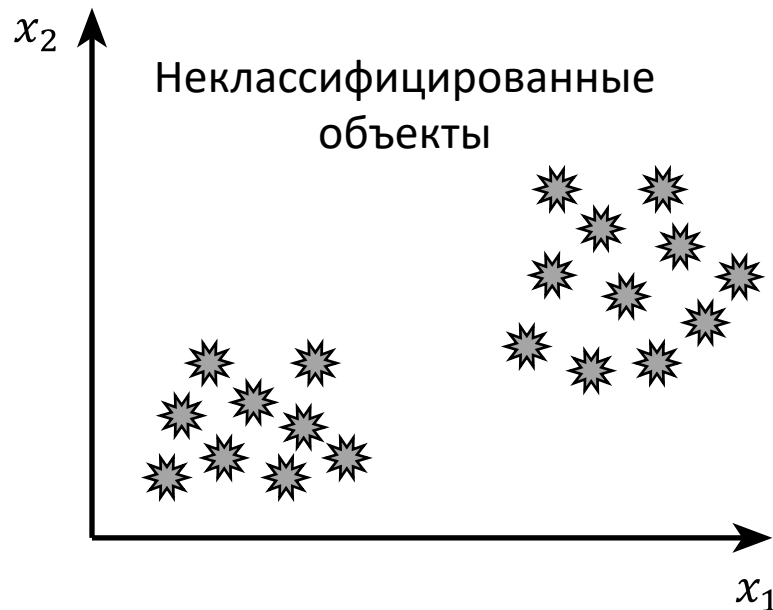
Не вдаваясь в подробности можно записать выражение для нахождения искомых коэффициентов многомерной регрессии:

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

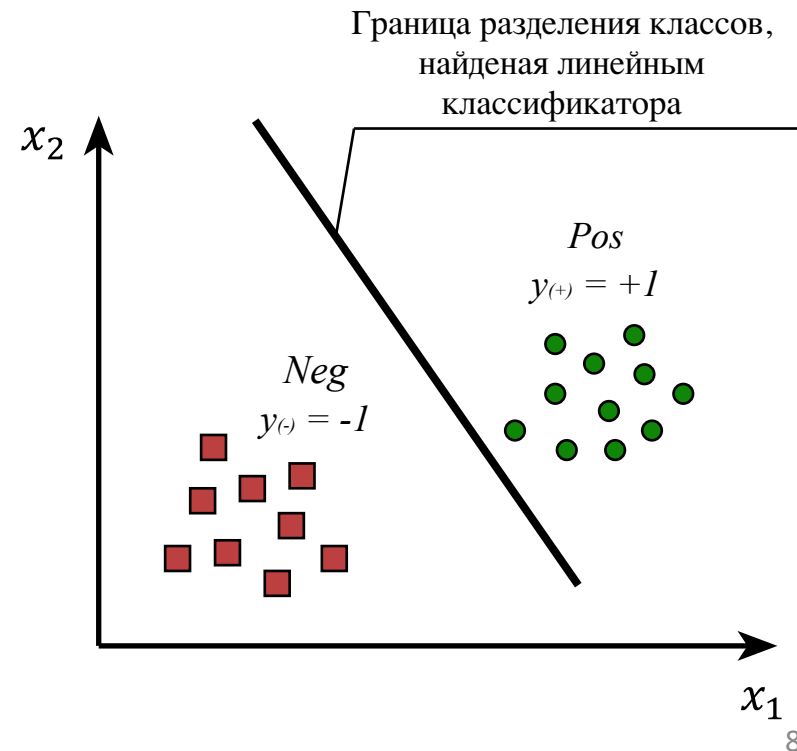
# Применение регрессии по МНК к задаче классификации

Линейную регрессию можно применять и для построения бинарного классификатора (разделение объектов на два класса), если два класса представлять вещественными числами. Для этого объекты одного класса мы можем обозначить как «положительные» - *Pos*, а объекты другого класса обозначить как «отрицательные» - *Neg*.

Объекты *Pos* помечают числом  $y^{(+)} = +1$ , а объекты *Neg* помечают числом  $y^{(-)} = -1$ .



Линейный  
классификатор



# Применение регрессии по МНК к задаче классификации

Линейный классификатор по методу МНК обучается решающей границе:

$$\mathbf{w} \cdot \mathbf{x} = \vartheta$$

где:  $\mathbf{w} = (\mathbf{X}^T \mathbf{X})^{-1} (Pos \mu^{(+)} - Neg \mu^{(-)})$

где:  $\mu^{(+)}, \mu^{(-)}$  -  $d$ -мерные векторы, содержащие средние значения каждого признака для положительных и отрицательных примеров соответственно.

Тогда линейный классификатор будет иметь вид:

$$\hat{y} = \text{sign}(\mathbf{w} \cdot \mathbf{x} - \vartheta)$$

$$\text{sign}(x) = \begin{cases} +1, & \text{если } x > 0 \\ 0, & \text{если } x = 0 \\ -1, & \text{если } x < 0 \end{cases}$$

## МИНУСЫ ТАКОГО ПОДХОДА:

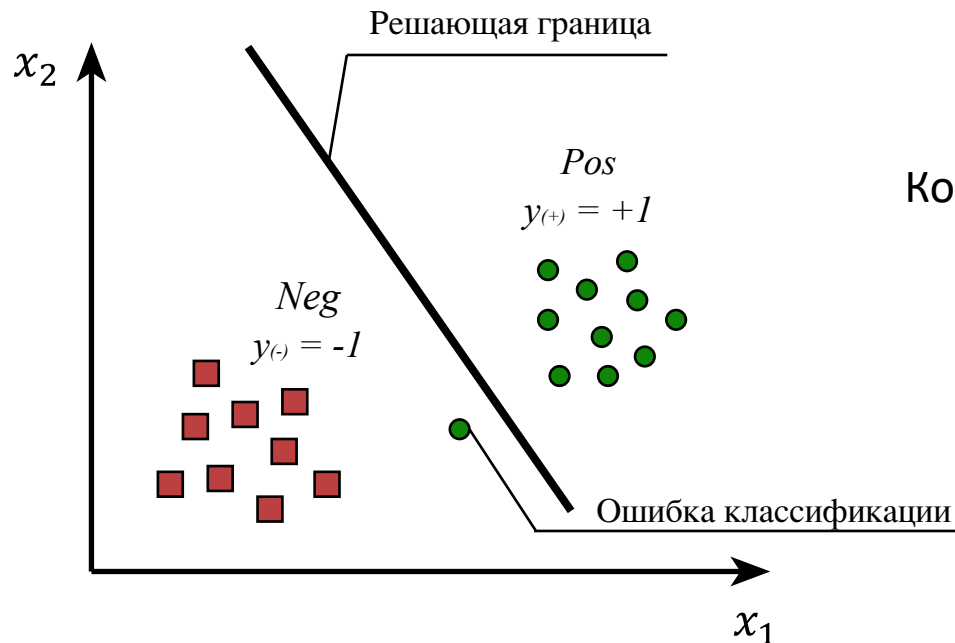
- 1) «Проклятие размерности» => при большом количестве признаков значительно увеличивается время вычисления.
- 2) Требуются специальные математические библиотеки, что не очень подходит при реализации на микроконтроллерах (речь идёт о так называемом малом машинном обучении или машинном обучении на микроконтроллерах, или TinyML).

# Перцептрон в качестве линейного классификатора

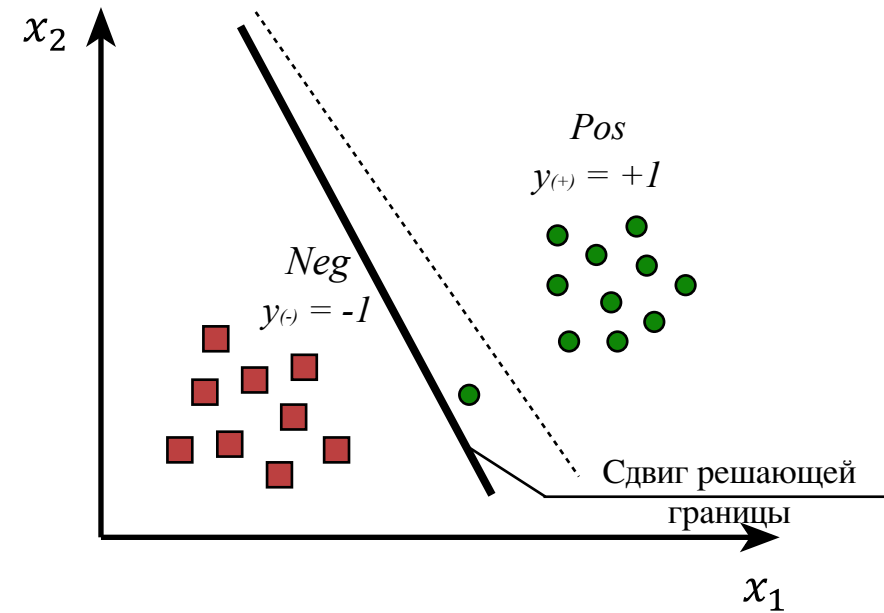
Линейный классификатор, который позволяет наилучшим образом разделить линейно разделимые данные называется **перцептроном**. По сути, перцептрон – это простейшая нейронная сеть.

Перцептрон корректирует вектор весов  $\mathbf{w}$  на основе поступающих входных обучающих данных. Данный метод является методом машинного обучения с учителем.

То есть, если на вход перцептрону поступает ошибочно классифицированный объект, например, в обучающем наборе объект помечен как  $+1$ , а перцептрон его классифицирует как  $-1$ , то перцептрону необходимо пересчитать вектор весовых коэффициентов  $\mathbf{w}$ , так чтобы линейная решающая граница сдвинулась и отнесла обучающий пример к правильному классу.



Коррекция вектора  
весов  $\mathbf{w}$



# Формула коррекции вектора весов перцептрона

$$\mathbf{w}' = \mathbf{w} + \eta \cdot y_i \cdot \mathbf{x}_i$$

где:  $\mathbf{w}$  - текущие значения вектора весов;  
 $\mathbf{w}'$  - новые (скорректированные) значения вектора весов;  
 $\eta$  - скорость обучения  $0 < \eta \leq 1$  ;  
 $y_i$  - метка текущего обучающего объекта  $y_i \in \{+1 \ -1\}$ ;  
 $\mathbf{x}_i$  - вектор признаков текущего обучающего объекта.

# Алгоритм обучения перцептрона как линейного классификатора

**Вход:** помеченные входные данные  $x \in D$ ,  
скорость обучения  $\eta$ .

**Выход:** вектор весов  $w$  для классификатора  $\hat{y} = \text{sign}(w \cdot x)$ .

*/\*Инициализация вектора весов нулями, в общем можно инициализировать и другими значениями\*/*

$w \leftarrow 0$ ;

$converged \leftarrow \text{false}$ ; */\*флаг сходимости алгоритма и завершения главного цикла\*/*

**while**  $converged = \text{false}$  **do**

*/\*коррекция вектора весов  $w$  перцептрона\*/*

$converged \leftarrow \text{true}$ ;

**for**  $i \leftarrow 1$  **in**  $|D|$  **do**

**if**  $y_i \cdot w \cdot x_i \leq 0$  */\*т.е.  $\hat{y} \neq y$ \*/*

**then**

$w \leftarrow w + \eta \cdot y_i \cdot x_i$ ;

$converged \leftarrow \text{false}$ ; */\*меняли  $w$ , значит алгоритм еще не сошёлся\*/*

**end**

**end**

**end**

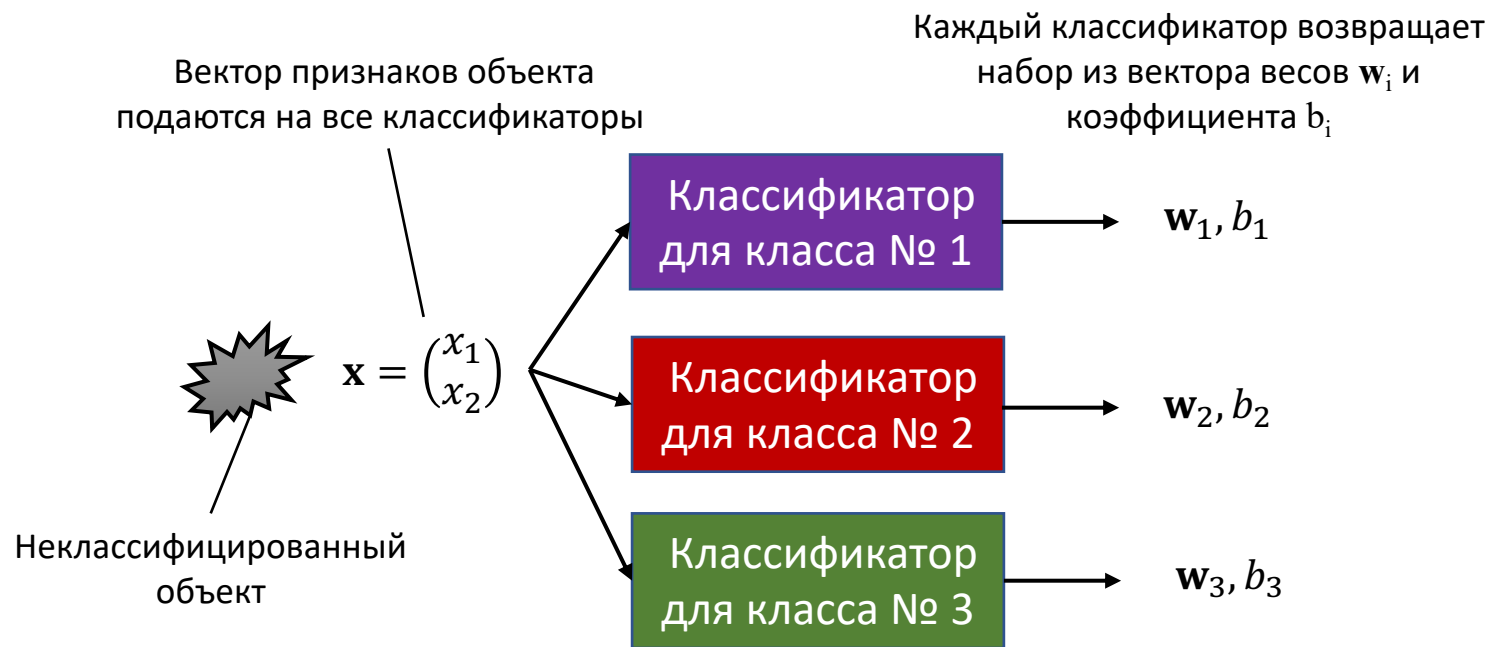
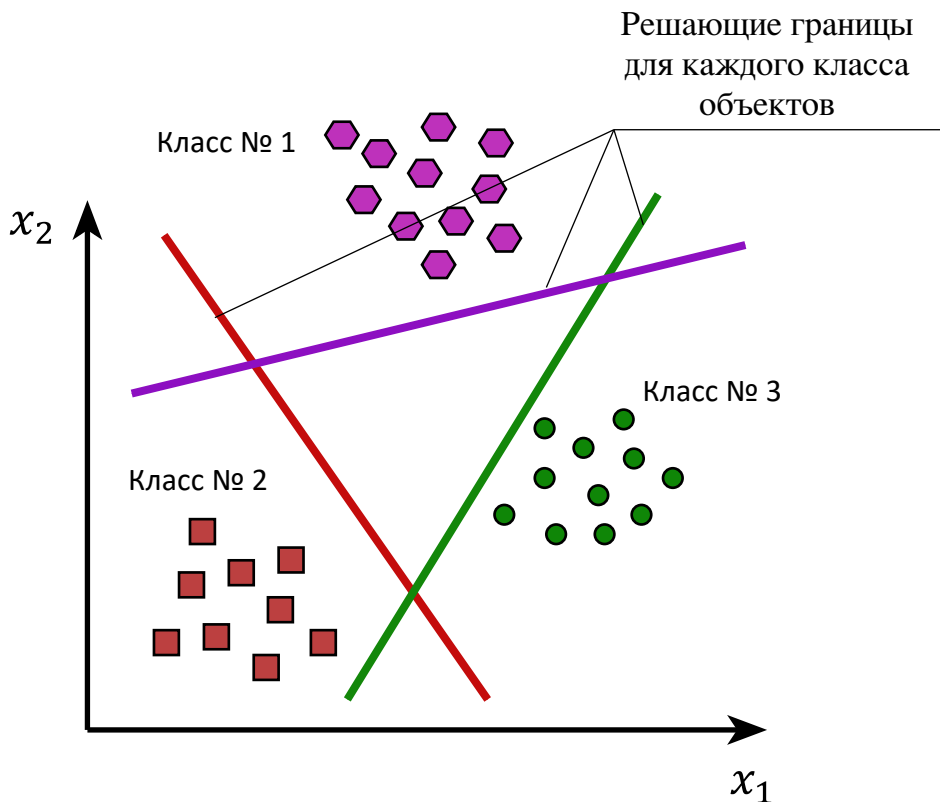
**return**  $w$ ;

Если алгоритм обучения не может обучить перцептрон разделять объекты на два класса, например, данные линейно плохо разделимы, то алгоритм может зациклиться!

# Линейные модели для мультиклассовой классификации

Обще распространенный подход, позволяющий использовать линейные модели не только для бинарной, но и мультиклассовой классификации, называют подходом «**один против остальных**» (**one-vs.-rest**).

В этом подходе строиться бинарная модель для каждого класса, которая пытается отделить этот класс от всех остальных классов => количество создаваемых моделей определяется количеством классов => количество решающих границ равно количеству классов (или моделей).

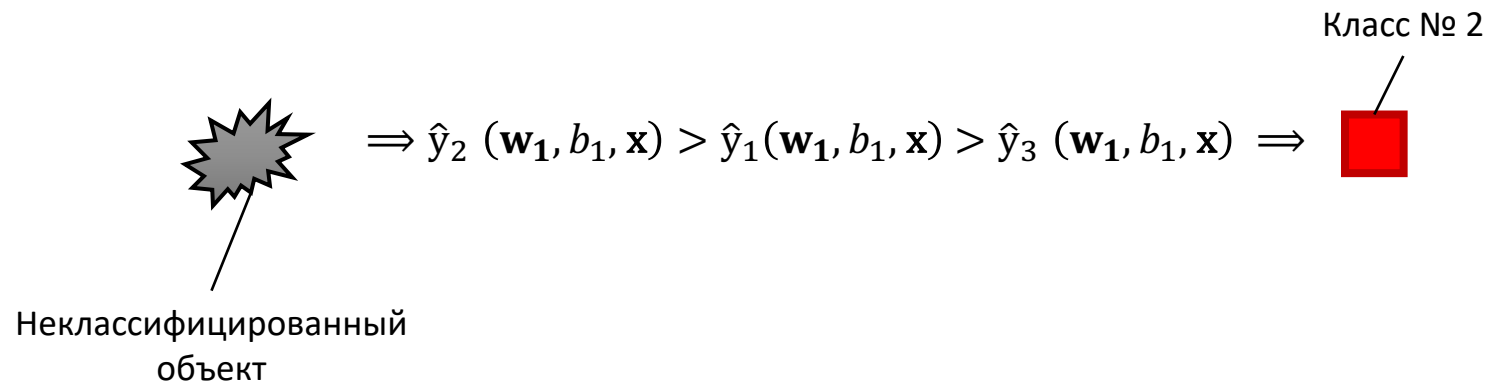


# Линейные модели для мультиклассовой классификации

Далее вычисляется выражение для каждого набора  $\mathbf{w}_i$  и  $b_i$  :

$$\hat{y}_i(\mathbf{w}_i, b_i, \mathbf{x}) = \mathbf{w}_i[0] \cdot x[0] + \mathbf{w}_i[1] \cdot x[1] + \dots + \mathbf{w}_i[d] \cdot x[d] + b_i$$

Классификатор, который выдаст **наибольшее значение** «побеждает» и выдает метку классифицируемому объекту.



# Использованные информационные источники

1. Флах П. Машинное обучение. Наука и искусство построения алгоритмов, которые извлекают знания из данных / пер. с англ. А. А. Слинкина. – М.: ДМК Пресс, 2015. – 400 с.: ил.
2. Мэрфи К. П. Вероятностное машинное обучение: введение / пер. с англ. А. А. Слинкина. – М.: ДМК Пресс, 2022. – 990 с.: ил.
3. Мюллер А., Гвидо С. Введение в машинное обучение с помощью Python. Руководство для специалистов по работе с данными.: Пер. с англ. - СПб.: ООО "Альфа-книга", 2017. - 480 с.: ил.