



Data Mining in Action

Лекция 3. Линейные модели



На предыдущей лекции

Мы изучили простые методы машинного обучения:

- Оптимальный порог и решающий пень
- Переход от решающего пня к дереву решений
- Метод ближайших соседей
- Наивный Байесовский классификатор

Как они работают?

- Как искать оптимальный порог для разбиения по одному признаку?

Как они работают?

- Как искать оптимальный порог для разбиения по одному признаку?
- Как выбрать оптимальный признак для построения новой вершины в дереве решений?

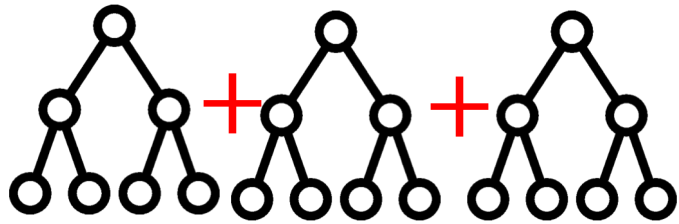
Как они работают?

- Как искать оптимальный порог для разбиения по одному признаку?
- Как выбрать оптимальный признак для построения новой вершины в дереве решений?
- Какие параметры нужно настроить для применения метода ближайших соседей?

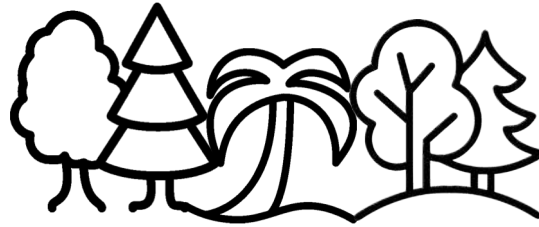
Как они работают?

- Как искать оптимальный порог для разбиения по одному признаку?
- Как выбрать оптимальный признак для построения новой вершины в дереве решений?
- Какие параметры нужно настроить для применения метода ближайших соседей?
- В чем состоит “наивное” предположение Байесовского классификатора?

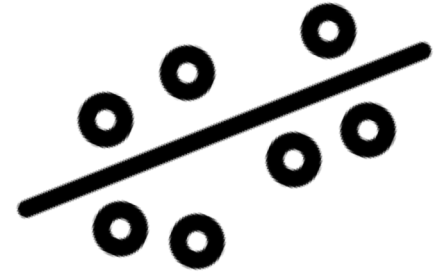
Напоминание: часто используемые методы



Градиентный бустинг



Случайный лес



Линейные модели

План лекции

1. Линейная классификация

2. Обучение линейной модели

3. Борьба с переобучением

4. Линейная регрессия

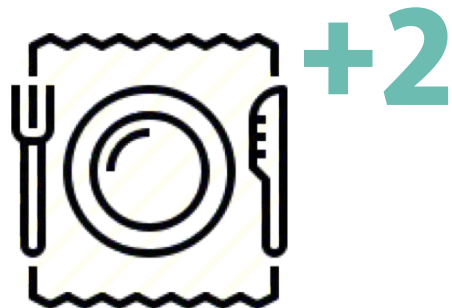
1. Линейная классификация

Пример: выходить из дома или нет

Признаки (1/0):



Вы свободны в
данный момент



Вам хочется
где-то поест



Вам хочется
спать



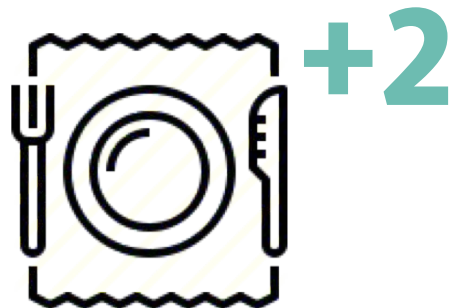
Вам хочется
увидеться с
друзьями

Пример: выходить из дома или нет

Признаки (1/0):



Вы свободны в
данный момент



Вам хочется
где-то поест



Вам хочется
спать



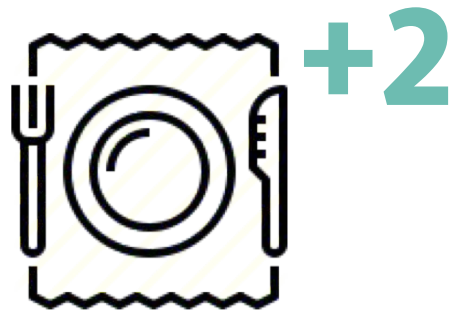
Вам хочется
увидеться с
друзьями

Пример: выходить из дома или нет

Признаки (1/0):



Вы свободны в
данный момент



Вам хочется
где-то поест



Вам хочется
спать

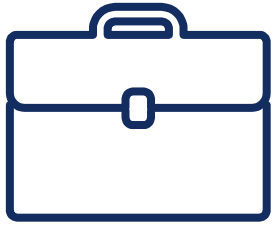


Вам хочется
увидеться с
друзьями

Порог для решающего правила: +1
Если сумма больше – выходим :)

Более серьезный пример: дать ли кредит

Признаки (1/0):



Работоспособный
возраст



Имеет счет в вашем
банке



Много просрочек по
другим кредитам



Просрочек нет, а
кредиты есть

Скоринговые карты

| ПОКАЗАТЕЛЬ | ДИАПАЗОН ЗНАЧЕНИЙ |
|---------------------------|---------------------|
| Возраст заемщика | До 35 лет |
| | От 35 до 45 лет |
| | От 45 и старше |
| Образование | Высшее |
| | Среднее специальное |
| | Среднее |
| Состоит ли в браке | Да |
| | Нет |
| Наличие кредита в прошлом | Да |
| | Нет |
| Стаж работы | До 1 года |
| | От 1 до 3 лет |
| | От 3 до 6 лет |
| | Свыше 6 лет |
| Наличие автомобиля | Да |
| | Нет |

Скоринговые карты

| ПОКАЗАТЕЛЬ | ДИАПАЗОН ЗНАЧЕНИЙ | СКОРИНГ-БАЛЛ |
|---------------------------|---------------------|--------------|
| Возраст заемщика | До 35 лет | 7,60 |
| | От 35 до 45 лет | 29,68 |
| | От 45 и старше | 35,87 |
| Образование | Высшее | 29,82 |
| | Среднее специальное | 20,85 |
| | Среднее | 22,71 |
| Состоит ли в браке | Да | 29,46 |
| | Нет | 9,38 |
| Наличие кредита в прошлом | Да | 40,55 |
| | Нет | 13,91 |
| Стаж работы | До 1 года | 15,00 |
| | От 1 до 3 лет | 18,14 |
| | От 3 до 6 лет | 19,85 |
| | Свыше 6 лет | 23,74 |
| Наличие автомобиля | Да | 51,69 |
| | Нет | 15,93 |

Подбор весов признаков и порога

Почему нельзя продолжать также:

- Сложно настраивать вручную
- Требуется эксперт в области
- Требуется проверка на данных и уточнение весов (эксперт может что-то не учесть)

Подбор весов признаков и порога

Почему нельзя продолжать также:

- Сложно настраивать вручную
- Требуется эксперт в области
- Требуется проверка на данных и уточнение весов (эксперт может что-то не учесть)

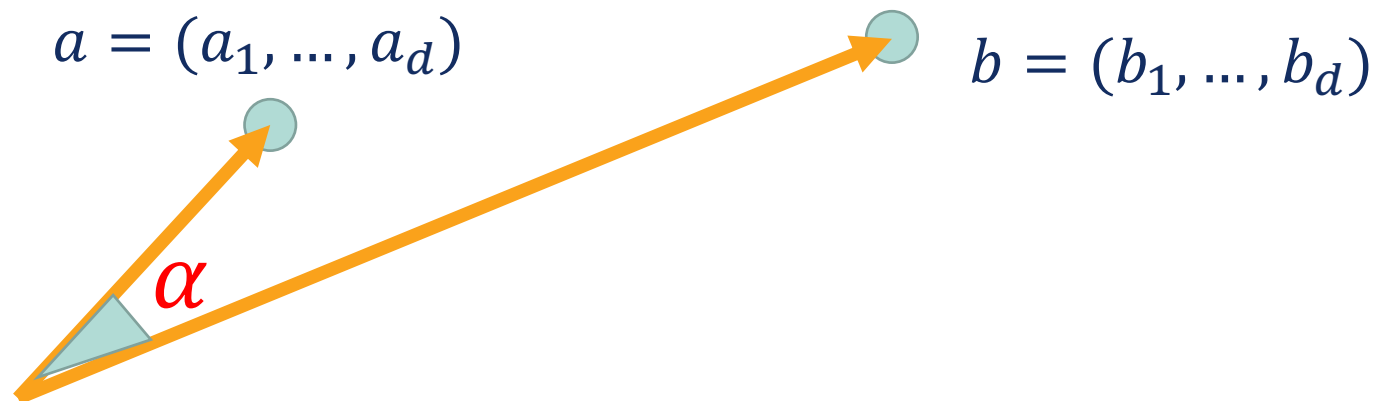
Решение – автоматизируем подбор параметров: придумаем функцию от параметров, которую надо минимизировать, и используем методы численной оптимизации

Формализуем линейный классификатор

$$a(x) = \begin{cases} 1, & \text{если } f(x) > 0 \\ -1, & \text{если } f(x) \leq 0 \end{cases}$$

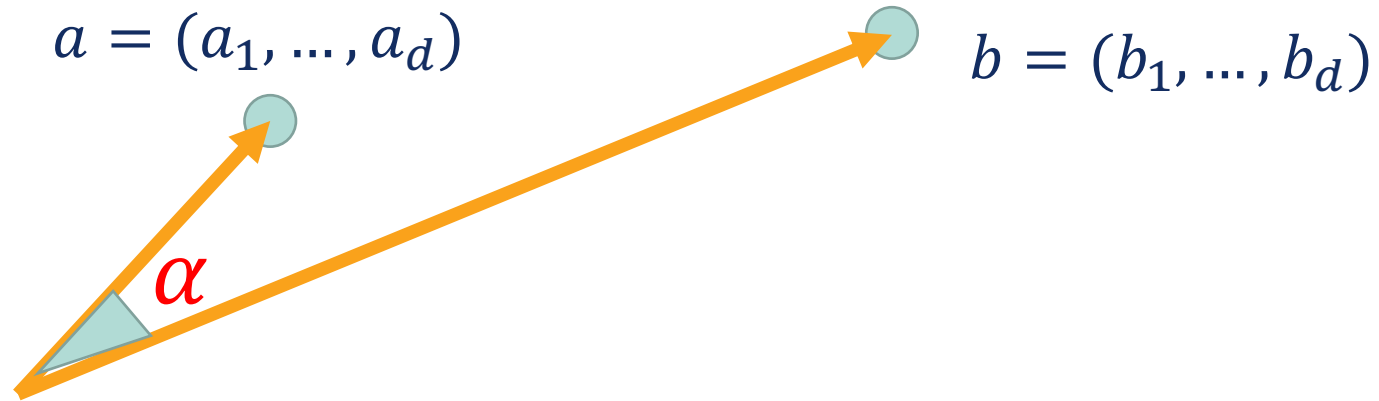
$$f(x) = w_0 + w_1x_1 + \dots + w_dx_d$$

Скалярное произведение векторов



$$\langle a, b \rangle = a_1 \cdot b_1 + \dots + a_d \cdot b_d$$

Скалярное произведение векторов



$$\langle a, b \rangle = a_1 \cdot b_1 + \dots + a_d \cdot b_d$$

$$\langle a, b \rangle = \|a\| \|b\| \cos \alpha$$

Формализуем линейный классификатор

$$a(x) = \begin{cases} 1, & \text{если } f(x) > 0 \\ -1, & \text{если } f(x) \leq 0 \end{cases}$$

$$f(x) = w_0 + w_1 x_1 + \dots + w_d x_d = w_0 + \langle w, x \rangle$$

Как выглядит
код:
применение
модели

Найдите «ошибку»:

```
import numpy as np

def f(x):
    return np.dot(w, x) + w0

def a(x):
    return 1 if f(x) > 0 else 0
```

Как выглядит
код:
применение
модели

Найдите «ошибку»:

```
import numpy as np

def f(x):
    return np.dot(w, x) + w0

def a(x):
    return 1 if f(x) > 0 else 0
```

Как выглядит
код:
применение
модели

```
import numpy as np

def f(x):
    return np.dot(w, x) + w0

def a(x):
    return 1 if f(x) > 0 else -1
```

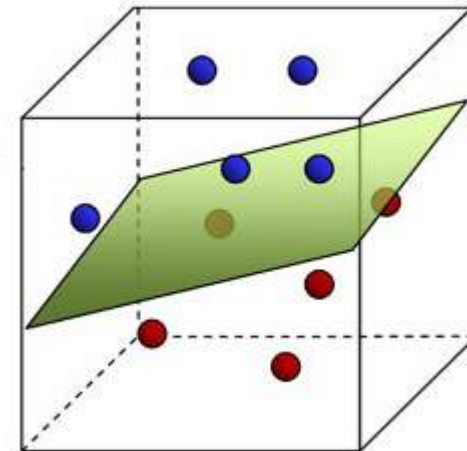
Будьте внимательны с метками класса!

Формализуем линейный классификатор

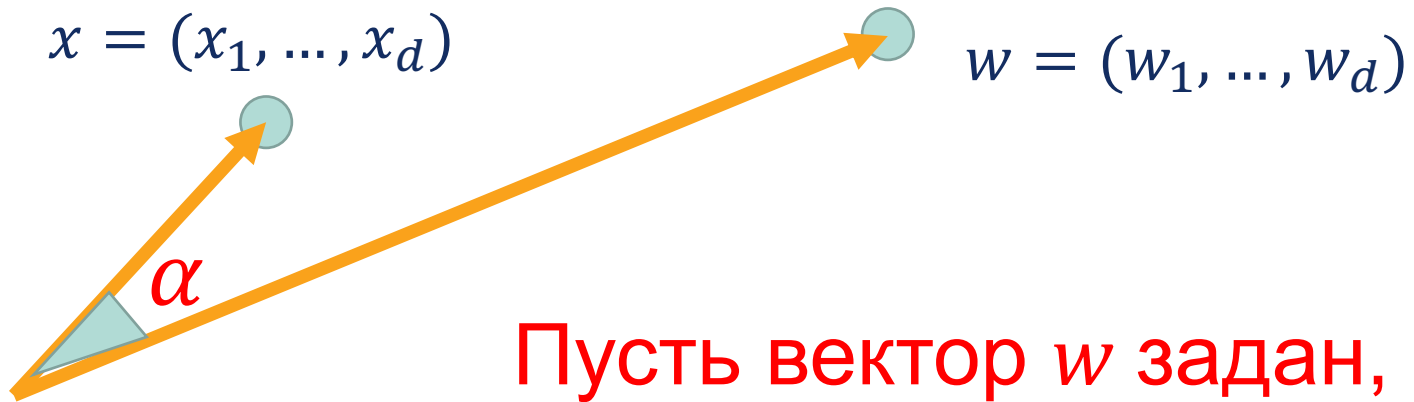
$$a(x) = \begin{cases} 1, & \text{если } f(x) > 0 \\ -1, & \text{если } f(x) \leq 0 \end{cases}$$

$$f(x) = w_0 + w_1x_1 + \dots + w_dx_d = w_0 + \langle w, x \rangle$$

Геометрическая
интерпретация: разделяем
классы плоскостью

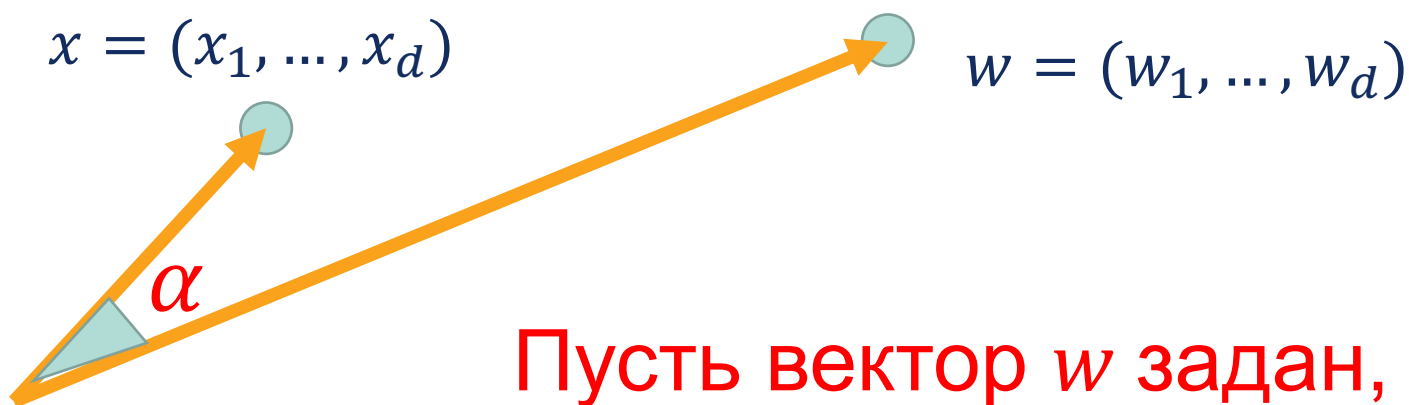


Почему плоскость?



Пусть вектор w задан, какие x удовлетворяют уравнению $\langle w, x \rangle = 0$?

Почему плоскость?



Пусть вектор w задан, какие x удовлетворяют уравнению $\langle w, x \rangle = 0$?

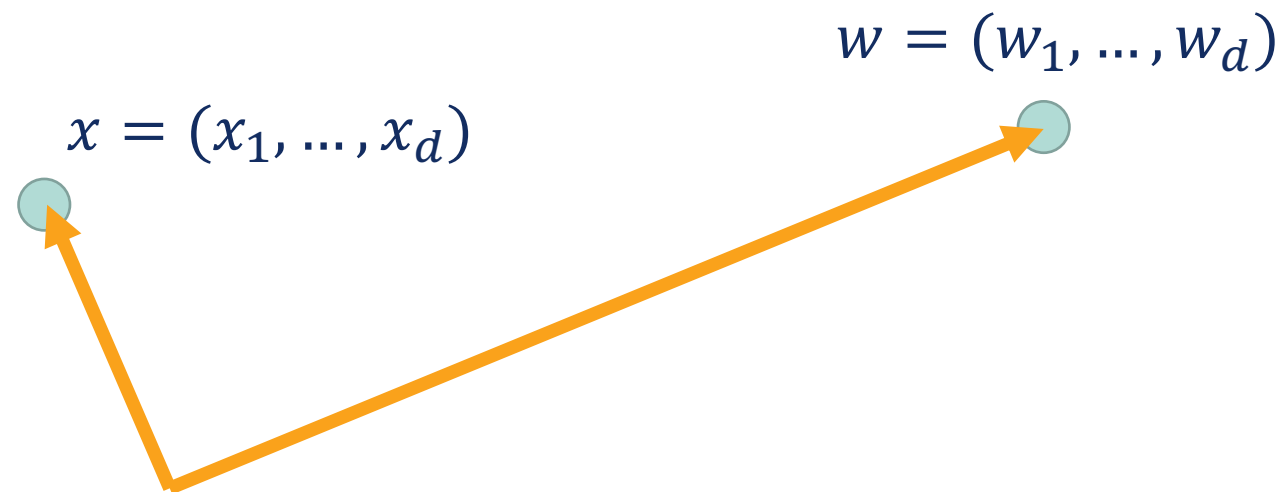
$$\langle w, x \rangle = \|w\| \|x\| \cos \alpha = 0,$$

Значит, если оба вектора не нулевые:

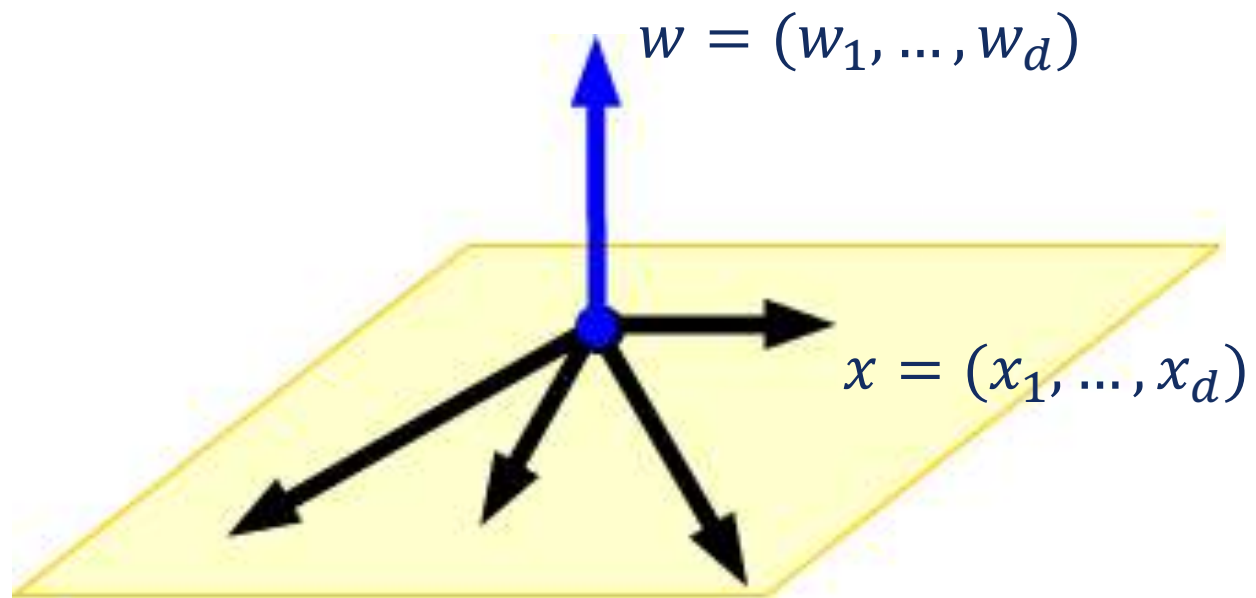
$$\cos \alpha = 0$$

$$\alpha = 90^\circ$$

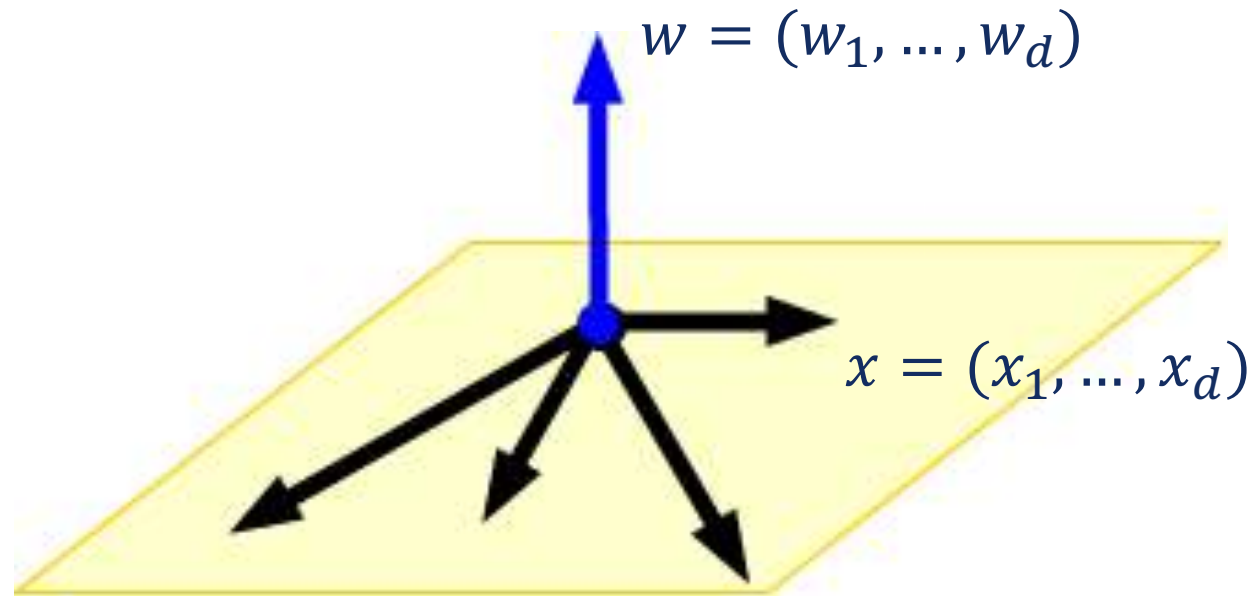
Почему плоскость?



Почему плоскость?



Почему плоскость?



А w_0 - сдвиг плоскости
от начала координат

Формализуем линейный классификатор

$$a(x) = \begin{cases} 1, & \text{если } f(x) > 0 \\ -1, & \text{если } f(x) \leq 0 \end{cases}$$

Если добавляем $x_{(0)} = 1$, то:

~~$$f(x) = w_0 + \langle w, x \rangle$$~~

$$f(x) = \langle w, x \rangle$$

Формализуем линейный классификатор

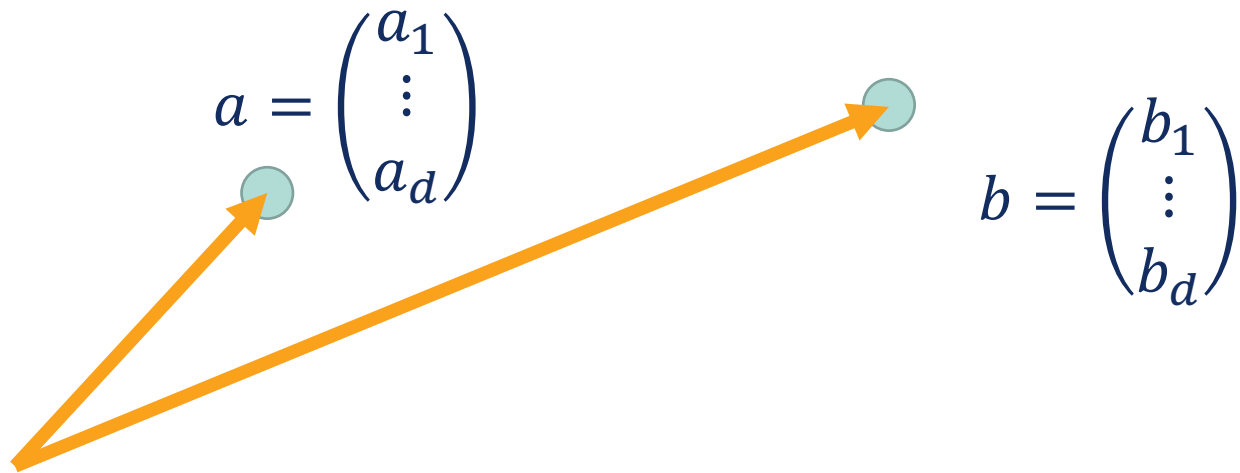
$$a(x) = \begin{cases} 1, & \text{если } f(x) > 0 \\ -1, & \text{если } f(x) \leq 0 \end{cases}$$

Если добавляем $x_{(0)} = 1$, то:

~~$$f(x) = w_0 + \langle w, x \rangle$$~~

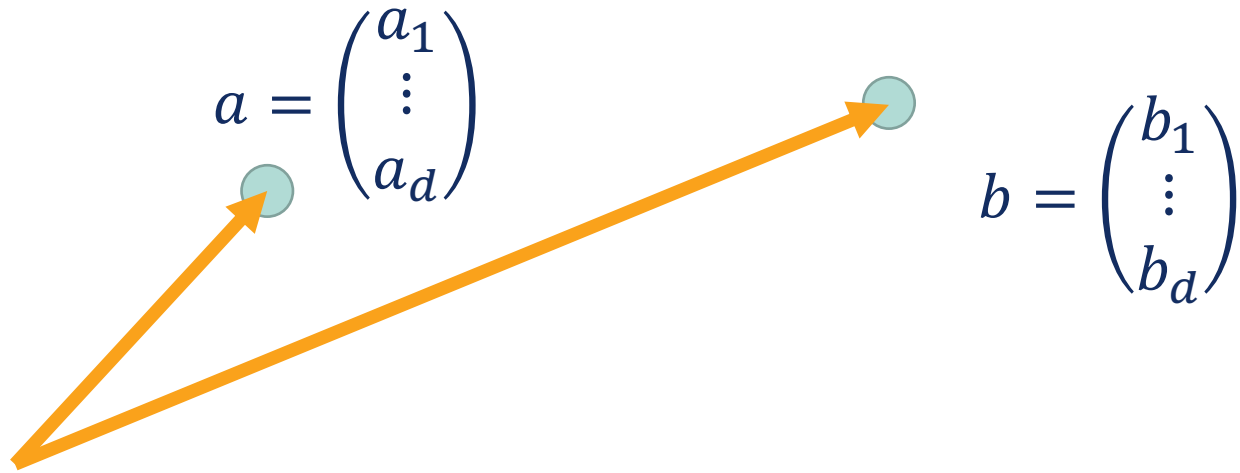
$$f(x) = \langle w, x \rangle = w^T x$$

Строки и столбцы



$$\langle a, b \rangle = (a_1, \dots, a_d) \begin{pmatrix} b_1 \\ \vdots \\ b_d \end{pmatrix}$$

Строки и столбцы



$$\langle a, b \rangle = (a_1, \dots, a_d) \begin{pmatrix} b_1 \\ \vdots \\ b_d \end{pmatrix} = \mathbf{a}^T \mathbf{b}$$

Отступ (margin)

Отступом алгоритма $a(x) = \text{sign}\{f(x)\}$ на объекте x_i называется величина

$$M_i = y_i f(x_i)$$

(y_i - класс, к которому относится x_i)

$$M_i \leq 0 \Leftrightarrow y_i \neq a(x_i)$$

$$M_i > 0 \Leftrightarrow y_i = a(x_i)$$

Функция потерь

$$Q(w) = \sum_{i=1}^{\ell} [M_i(w) \leq 0]$$

Функция потерь

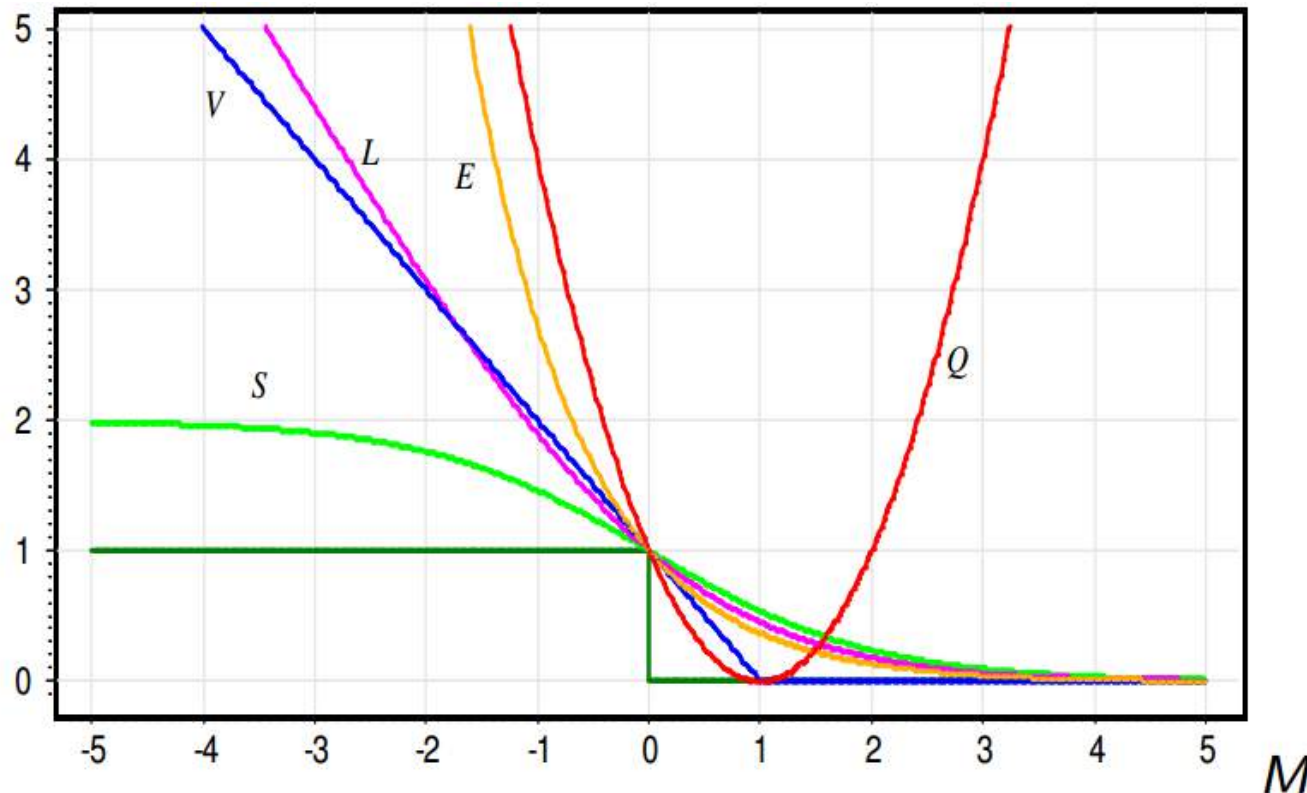
$$Q(w) = \sum_{i=1}^{\ell} [M_i(w) \leq 0] \leq \tilde{Q}(w) = \sum_{i=1}^{\ell} \mathcal{L}(M_i(w)) \rightarrow \min_w;$$

Функция эмпирического
риска

Функция потерь

Функция потерь

$$Q(w) = \sum_{i=1}^{\ell} [M_i(w) \leq 0] \leq \tilde{Q}(w) = \sum_{i=1}^{\ell} \mathcal{L}(M_i(w)) \rightarrow \min_w;$$



$$\begin{aligned} Q(M) &= (1 - M)^2 \\ V(M) &= (1 - M)_+ \\ S(M) &= 2(1 + e^M)^{-1} \\ L(M) &= \log_2(1 + e^{-M}) \\ E(M) &= e^{-M} \end{aligned}$$

2. Обучение модели

Скорость изменения функции

Рассмотрим функцию $f(x) = x^2$

$f(3) = 9$, вопрос: как оценить $f(3.01)$?

Скорость изменения функции

Рассмотрим функцию $f(x) = x^2$

$f(3) = 9$, вопрос: как оценить $f(3.01)$?

$$(3 + 0.01)^2 = 9 + 2 \cdot 3 \cdot 0.01 + 0.0001$$

Скорость изменения функции

Рассмотрим функцию $f(x) = x^2$

$f(3) = 9$, вопрос: как оценить $f(3.01)$?

$$(3 + 0.01)^2 = 9 + 2 \cdot 3 \cdot 0.01 + 0.0001$$

$$(x + \varepsilon)^2 \approx x^2 + 2x\varepsilon$$

Скорость изменения функции

Рассмотрим функцию $f(x) = x^2$

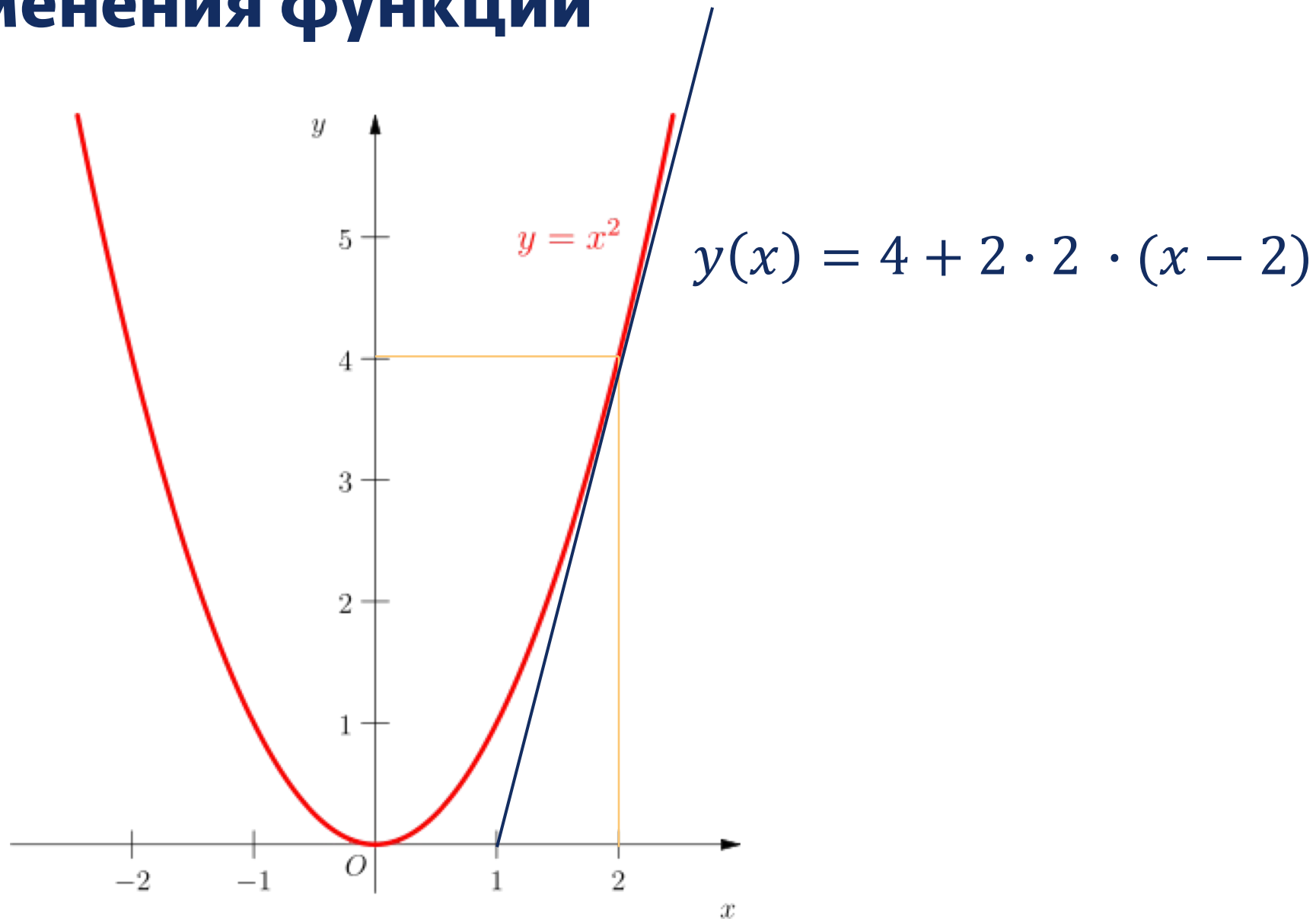
$f(3) = 9$, вопрос: как оценить $f(3.01)$?

$$(3 + 0.01)^2 = 9 + 2 \cdot 3 \cdot 0.01 + 0.0001$$

$$(x + \varepsilon)^2 \approx x^2 + \boxed{2x}\varepsilon$$

$$f'(x) = 2x$$

Скорость изменения функции



Производные простых функций

$$(x^n)' = nx^{n-1}$$

$$(\ln x)' = \frac{1}{x}$$

$$(e^x)' = e^x$$

Сложная функция и chain rule

$$\ln((x + \varepsilon)^n) \approx \ln(x^n + nx^{n-1}\varepsilon) \approx$$

Сложная функция и chain rule

$$\begin{aligned}\ln((x + \varepsilon)^n) &\approx \ln(x^n + nx^{n-1}\varepsilon) \approx \\ &\approx \ln(x^n) + \frac{1}{x^n} nx^{n-1}\varepsilon\end{aligned}$$

Сложная функция и chain rule

$$\begin{aligned}\ln((x + \varepsilon)^n) &\approx \ln(x^n + nx^{n-1}\varepsilon) \approx \\ &\approx \ln(x^n) + \frac{1}{x^n} nx^{n-1}\varepsilon\end{aligned}$$

Вывод: $(g(f(x)))' = g'(f(x))f'(x)$

Когда переменных много

$$f(x, y) = 2x + y^2$$

$$\frac{\partial f}{\partial x} \equiv f'_x = 2$$

$$\frac{\partial f}{\partial y} \equiv f'_y = 2y$$

Градиент функции

$$f(x, y) = 2x + y^2$$

$$\nabla f = \begin{pmatrix} f'_x \\ f'_y \end{pmatrix} = \begin{pmatrix} 2 \\ 2y \end{pmatrix}$$

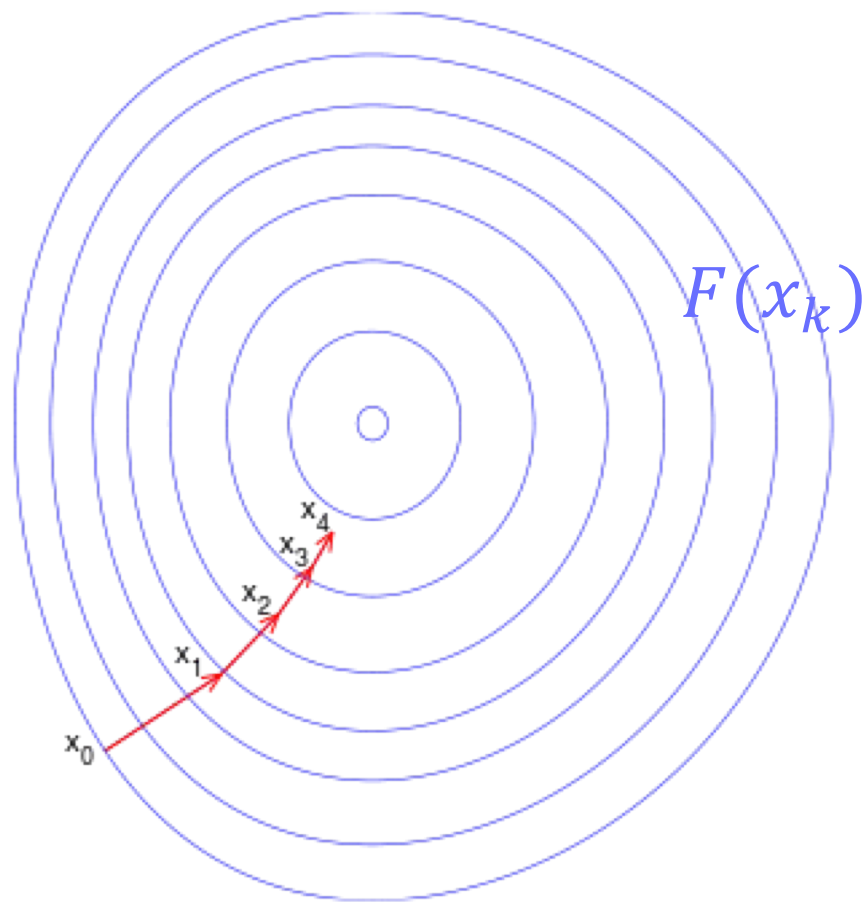
Градиент функции

$$f(x, y) = 2x + y^2$$

$$\nabla f = \begin{pmatrix} f'_x \\ f'_y \end{pmatrix} = \begin{pmatrix} 2 \\ 2y \end{pmatrix}$$

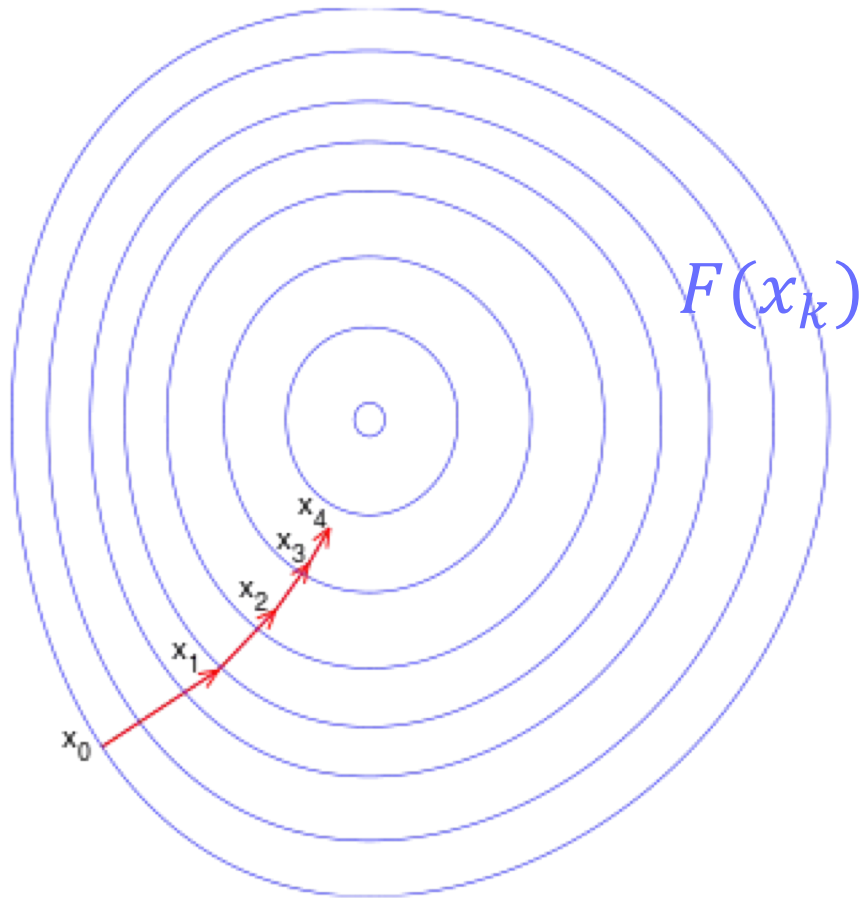
Утверждение: ∇f – направление наискорейшего роста функции, $-\nabla f$ – направление наискорейшего убывания

Градиентный спуск (GD, Gradient Decent)



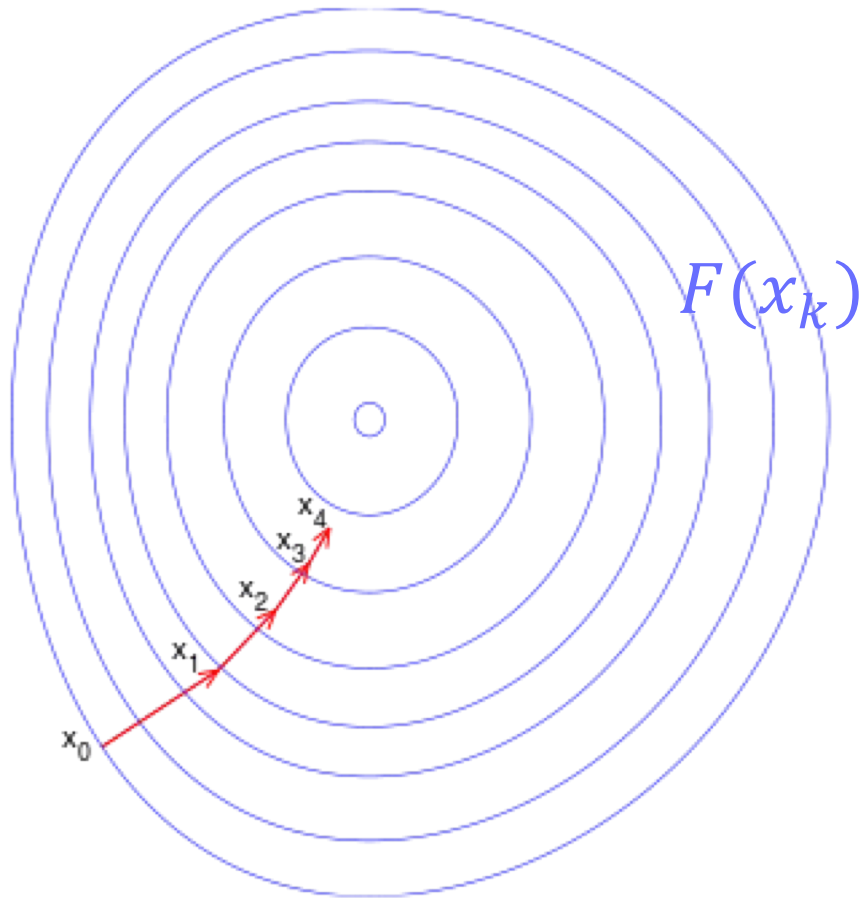
Градиентный спуск (GD, Gradient Decent)

$$x_{k+1} = x_k - \gamma_k \nabla F(x_k)$$



Градиентный спуск (GD, Gradient Decent)

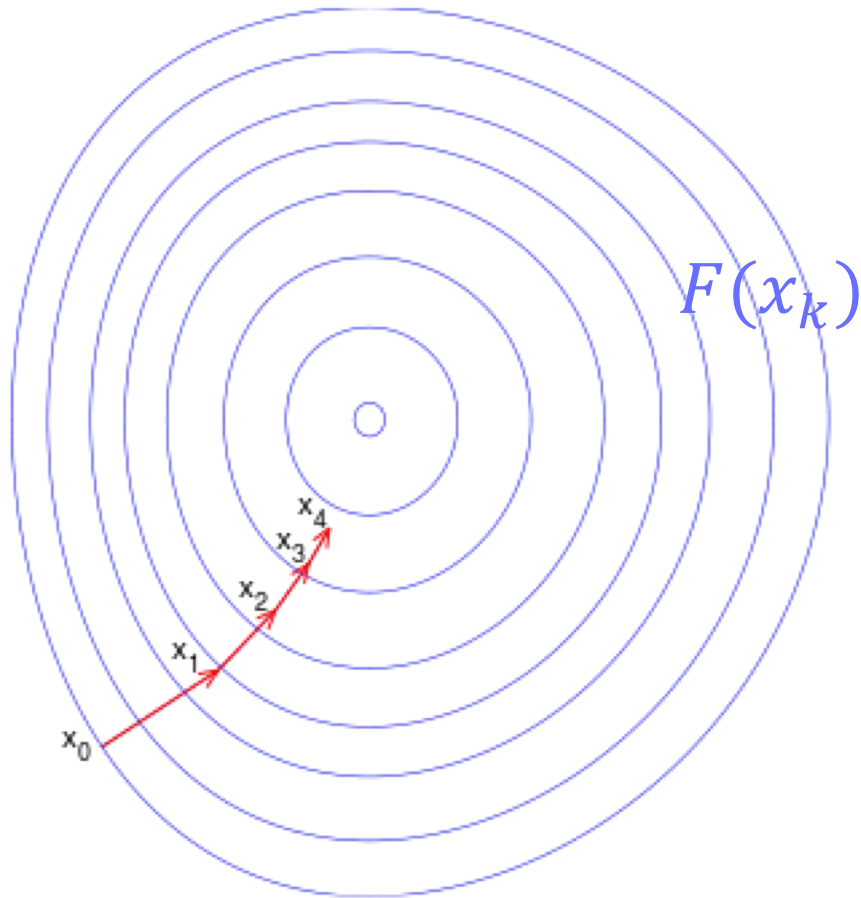
$$x_{k+1} = x_k - \gamma_k \nabla F(x_k)$$



$$\nabla_w \tilde{Q} = \sum_{i=1}^l \nabla L(M_i) = \sum_{i=1}^l L'(M_i) \frac{\partial M_i}{\partial w}$$

Градиентный спуск (GD, Gradient Decent)

$$x_{k+1} = x_k - \gamma_k \nabla F(x_k)$$

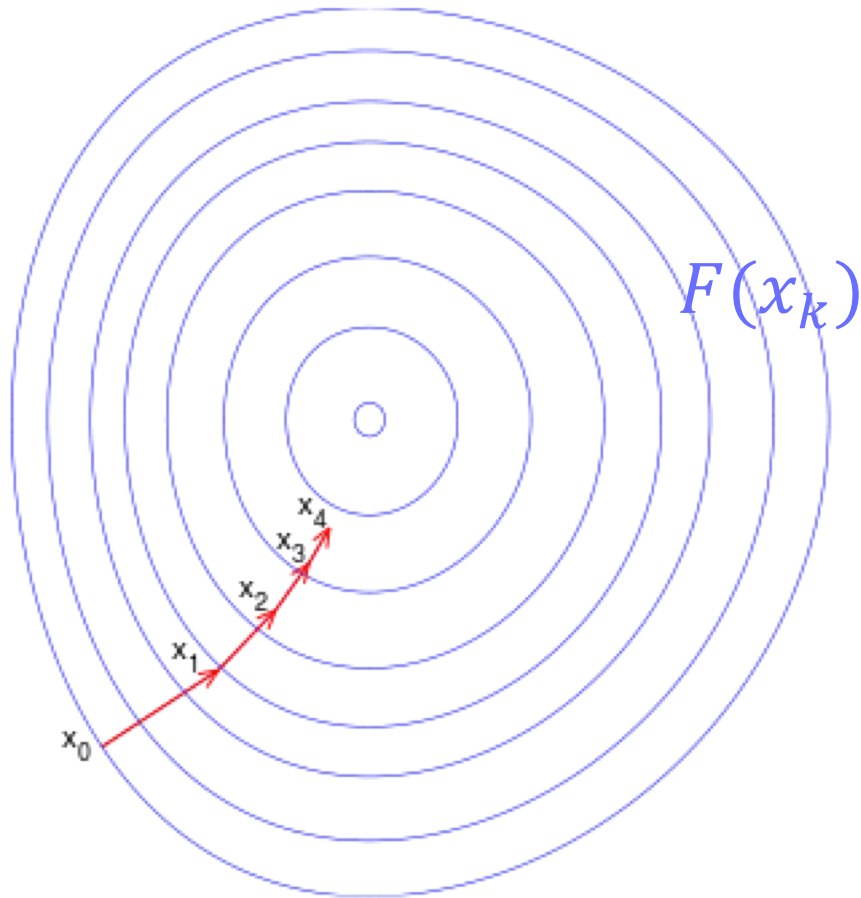


$$\nabla_w \tilde{Q} = \sum_{i=1}^l \nabla L(M_i) = \sum_{i=1}^l L'(M_i) \frac{\partial M_i}{\partial w}$$

$$M_i = y_i \langle w, x_i \rangle \Rightarrow \frac{\partial M_i}{\partial w} = y_i x_i$$

Градиентный спуск (GD, Gradient Decent)

$$x_{k+1} = x_k - \gamma_k \nabla F(x_k)$$



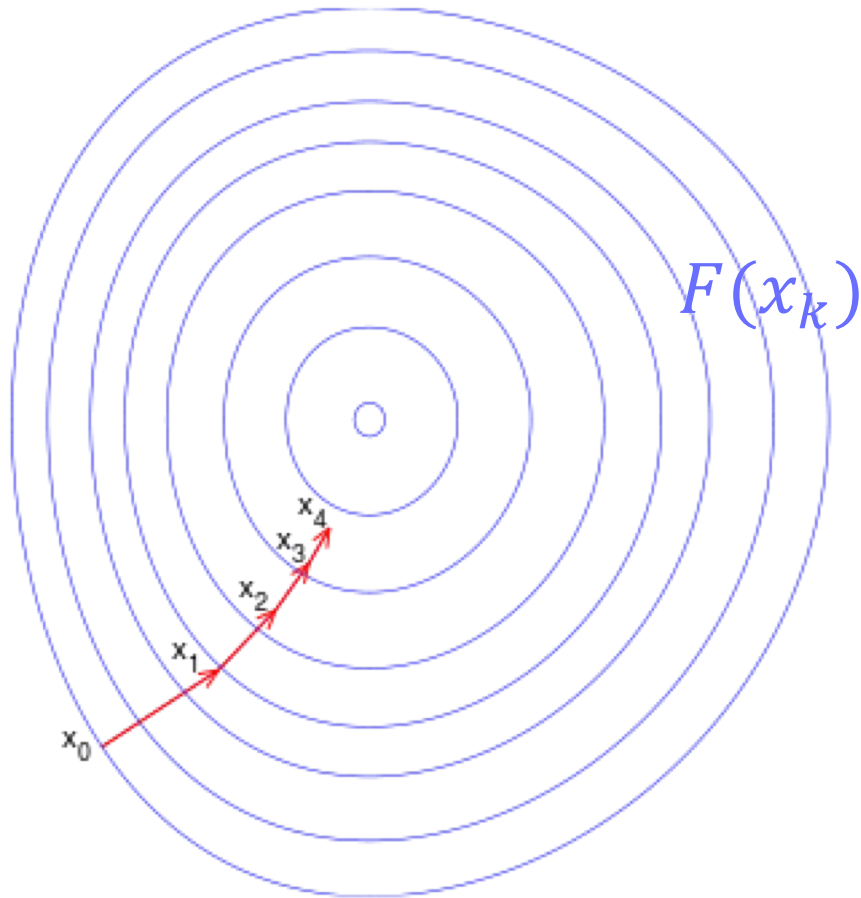
$$\nabla_w \tilde{Q} = \sum_{i=1}^l \nabla L(M_i) = \sum_{i=1}^l L'(M_i) \frac{\partial M_i}{\partial w}$$

$$M_i = y_i \langle w, x_i \rangle \Rightarrow \frac{\partial M_i}{\partial w} = y_i x_i$$

$$\nabla \tilde{Q} = \sum_{i=1}^l y_i x_i L'(M_i)$$

Градиентный спуск (GD, Gradient Decent)

$$x_{k+1} = x_k - \gamma_k \nabla F(x_k)$$



$$\nabla_w \tilde{Q} = \sum_{i=1}^l \nabla L(M_i) = \sum_{i=1}^l L'(M_i) \frac{\partial M_i}{\partial w}$$

$$M_i = y_i \langle w, x_i \rangle \Rightarrow \frac{\partial M_i}{\partial w} = y_i x_i$$

$$\nabla \tilde{Q} = \sum_{i=1}^l y_i x_i L'(M_i)$$

$$w_{k+1} = w_k - \gamma_k \sum_{i=1}^l y_i x_i L'(M_i)$$

Стохастический градиент (SGD)

$$w_{k+1} = w_k - \gamma_k \sum_{i=1}^l y_i x_i L'(M_i)$$

$$w_{k+1} = w_k - \gamma_k y_i x_i L'(M_i)$$

x_i — случайный элемент обучающей выборки

Как выглядит код: обучение модели

```
from random import randint
import numpy as np

def loss(x, y):
    return max([0, 1 - y * f(x)])

def der_loss(x, y):
    return -1.0 if 1 - y * f(x) > 0 else 0.0

def fit(X_train, y_train):
    w = np.random.randn(X_train.shape[1])
    w0 = np.random.randn()
    for k in range(10000):
        rand_index = randint(0, len(X_train) - 1)
        x = X_train[rand_index]
        y = y_train[rand_index]
        step = 0.01
        w -= step * y * x * der_loss(x, y)
        w0 -= step * y * der_loss(x, y)
```

Как выглядит код: обучение модели

```
from random import randint
import numpy as np
```

$$L(M) = \max\{0, 1 - M\} = (1 - M)_+$$

```
def loss(x, y):
    return max([0, 1 - y * f(x)])
```

```
def der_loss(x, y):
    return -1.0 if 1 - y * f(x) > 0 else 0.0
```

```
def fit(X_train, y_train):
    w = np.random.randn(X_train.shape[1])
    w0 = np.random.randn()
    for k in range(10000):
        rand_index = randint(0, len(X_train) - 1)
        x = X_train[rand_index]
        y = y_train[rand_index]
        step = 0.01
        w -= step * y * x * der_loss(x, y)
        w0 -= step * y * der_loss(x, y)
```

Как выглядит код: обучение модели

```
from random import randint
import numpy as np
```

```
def loss(x, y):
    return max([0, 1 - y * f(x)])
```

```
def der_loss(x, y):
    return -1.0 if 1 - y * f(x) > 0 else 0.0
```

```
def fit(X_train, y_train):
    w = np.random.randn(X_train.shape[1])
    w0 = np.random.randn()
    for k in range(10000):
        rand_index = randint(0, len(X_train) - 1)
        x = X_train[rand_index]
        y = y_train[rand_index]
        step = 0.01
        w -= step * y * x * der_loss(x, y)
        w0 -= step * y * der_loss(x, y)
```


Как выглядит код: обучение модели

```
from random import randint
import numpy as np

def loss(x, y):
    return max([0, 1 - y * f(x)])

def der_loss(x, y):
    return -1.0 if 1 - y * f(x) > 0 else 0.0

def fit(X_train, y_train):
    w = np.random.randn(X_train.shape[1])
    w0 = np.random.randn()
    for k in range(10000):
        rand_index = randint(0, len(X_train) - 1)
        x = X_train[rand_index]
        y = y_train[rand_index]
        step = 0.01
        w -= step * y * x * der_loss(x, y)
        w0 -= step * y * der_loss(x, y)
```

Как выглядит код: обучение модели

```
from random import randint
import numpy as np

def loss(x, y):
    return max([0, 1 - y * f(x)])

def der_loss(x, y):
    return -1.0 if 1 - y * f(x) > 0 else 0.0

def fit(X_train, y_train):
    w = np.random.randn(X_train.shape[1])
    w0 = np.random.randn()
    for k in range(10000):
        rand_index = randint(0, len(X_train) - 1)
        x = X_train[rand_index]
        y = y_train[rand_index]
        step = 0.01
        w -= step * y * x * der_loss(x, y)
        w0 -= step * y * der_loss(x, y)
```

Как выглядит код: обучение модели

```
from random import randint
import numpy as np

def loss(x, y):
    return max([0, 1 - y * f(x)])

def der_loss(x, y):
    return -1.0 if 1 - y * f(x) > 0 else 0.0

def fit(X_train, y_train):
    w = np.random.randn(X_train.shape[1])
    w0 = np.random.randn()
    for k in range(10000):
        rand_index = randint(0, len(X_train) - 1)
        x = X_train[rand_index]
        y = y_train[rand_index]
        step = 0.01
        w -= step * y * x * der_loss(x, y)
        w0 -= step * y * der_loss(x, y)
```


Как выглядит код: обучение модели

```
from random import randint
import numpy as np

def loss(x, y):
    return max([0, 1 - y * f(x)])

def der_loss(x, y):
    return -1.0 if 1 - y * f(x) > 0 else 0.0

def fit(X_train, y_train):
    w = np.random.randn(X_train.shape[1])
    w0 = np.random.randn()
    for k in range(10000):
        rand_index = randint(0, len(X_train) - 1)
        x = X_train[rand_index]
        y = y_train[rand_index]
        step = 0.01
        w -= step * y * x * der_loss(x, y)
        w0 -= step * y * der_loss(x, y)
```

Как выглядит код: обучение модели

```
from random import randint
import numpy as np

def loss(x, y):
    return max([0, 1 - y * f(x)])

def der_loss(x, y):
    return -1.0 if 1 - y * f(x) > 0 else 0.0

def fit(X_train, y_train):
    w = np.random.randn(X_train.shape[1])
    w0 = np.random.randn()
    for k in range(10000):
        rand_index = randint(0, len(X_train) - 1)
        x = X_train[rand_index]
        y = y_train[rand_index]
        step = 0.01
        w -= step * y * x * der_loss(x, y)
        w0 -= step * y * der_loss(x, y)
```

Как выглядит код: обучение модели

```
from random import randint
import numpy as np

def loss(x, y):
    return max([0, 1 - y * f(x)])

def der_loss(x, y):
    return -1.0 if 1 - y * f(x) > 0 else 0.0

def fit(X_train, y_train):
    w = np.random.randn(X_train.shape[1])
    w0 = np.random.randn()
    for k in range(10000):
        rand_index = randint(0, len(X_train) - 1)
        x = X_train[rand_index]
        y = y_train[rand_index]
        step = 0.01
        w -= step * y * x * der_loss(x, y)
        w0 -= step * y * der_loss(x, y)
```

Как выглядит код: обучение модели

```
from random import randint
import numpy as np

def loss(x, y):
    return max([0, 1 - y * f(x)])

def der_loss(x, y):
    return -1.0 if 1 - y * f(x) > 0 else 0.0

def fit(X_train, y_train):
    w = np.random.randn(X_train.shape[1])
    w0 = np.random.randn()
    for k in range(10000):
        rand_index = randint(0, len(X_train) - 1)
        x = X_train[rand_index]
        y = y_train[rand_index]
        step = 0.01
        w -= step * y * x * der_loss(x, y)
        w0 -= step * y * der_loss(x, y)
```


Как выглядит код: обучение модели

```
from random import randint
import numpy as np

def loss(x, y):
    return max([0, 1 - y * f(x)])

def der_loss(x, y):
    return -1.0 if 1 - y * f(x) > 0 else 0.0

def fit(X_train, y_train):
    w = np.random.randn(X_train.shape[1])
    w0 = np.random.randn()
    for k in range(10000):
        rand_index = randint(0, len(X_train) - 1)
        x = X_train[rand_index]
        y = y_train[rand_index]
        step = 0.01
        w -= step * y * x * der_loss(x, y)
        w0 -= step * y * der_loss(x, y)
```

$$\gamma_k = \frac{1}{\alpha + k}$$

Как выглядит код: обучение модели

```
from random import randint
import numpy as np

def loss(x, y):
    return max([0, 1 - y * f(x)])

def der_loss(x, y):
    return -1.0 if 1 - y * f(x) > 0 else 0.0

def fit(X_train, y_train):
    w = np.random.randn(X_train.shape[1])
    w0 = np.random.randn()
    for k in range(10000):
        rand_index = randint(0, len(X_train) - 1)
        x = X_train[rand_index]
        y = y_train[rand_index]
        step = 0.01
        w -= step * y * x * der_loss(x, y)
        w0 -= step * y * der_loss(x, y)
```

$$\gamma_k = \frac{1}{\alpha + k}$$
$$\gamma_k = \frac{1}{\sqrt{\alpha + k}}$$

Как выглядит код: обучение модели

```
from random import randint
import numpy as np

def loss(x, y):
    return max([0, 1 - y * f(x)])

def der_loss(x, y):
    return -1.0 if 1 - y * f(x) > 0 else 0.0

def fit(X_train, y_train):
    w = np.random.randn(X_train.shape[1])
    w0 = np.random.randn()
    for k in range(10000):
        rand_index = randint(0, len(X_train) - 1)
        x = X_train[rand_index]
        y = y_train[rand_index]
        step = 0.01
        w -= step * y * x * der_loss(x, y)
        w0 -= step * y * der_loss(x, y)
```

$$\gamma_k = \frac{1}{\alpha + k}$$

$$\gamma_k = \frac{1}{\sqrt{\alpha + k}}$$

$$\gamma_k = (\alpha + k)^{-\beta}$$

Как выглядит код: обучение модели

```
from random import randint
import numpy as np

def loss(x, y):
    return max([0, 1 - y * f(x)])

def der_loss(x, y):
    return -1.0 if 1 - y * f(x) > 0 else 0.0

def fit(X_train, y_train):
    w = np.random.randn(X_train.shape[1])
    w0 = np.random.randn()
    for k in range(10000):
        rand_index = randint(0, len(X_train) - 1)
        x = X_train[rand_index]
        y = y_train[rand_index]
        step = 0.01
        w -= step * y * x * der_loss(x, y)
        w0 -= step * y * der_loss(x, y)
```

$$\gamma_k = \frac{1}{\alpha + k}$$

$$\gamma_k = \frac{1}{\sqrt{\alpha + k}}$$

$$\gamma_k = (\alpha + k)^{-\beta}$$

$$\gamma_k = \tau \beta_k$$

Как выглядит код: обучение модели

```
from random import randint
import numpy as np

def loss(x, y):
    return max([0, 1 - y * f(x)])

def der_loss(x, y):
    return -1.0 if 1 - y * f(x) > 0 else 0.0

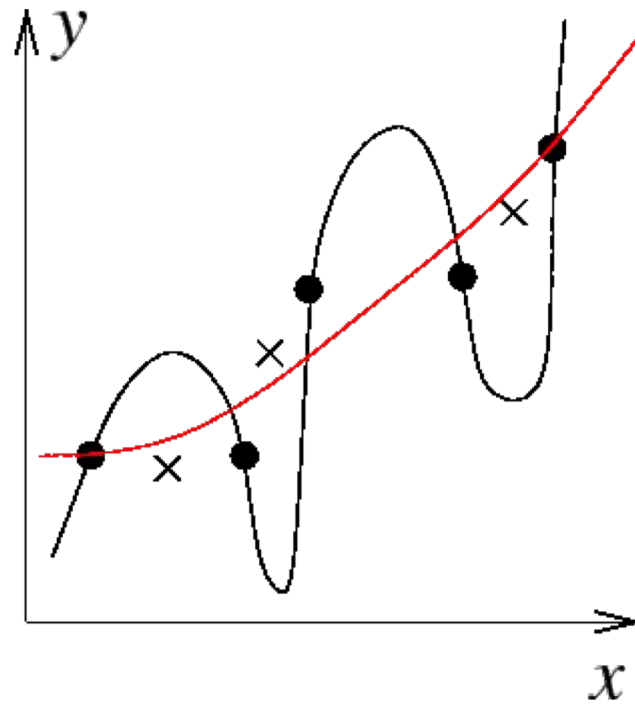
def fit(X_train, y_train):
    w = np.random.randn(X_train.shape[1])
    w0 = np.random.randn()
    for k in range(10000):
        rand_index = randint(0, len(X_train) - 1)
        x = X_train[rand_index]
        y = y_train[rand_index]
        step = 0.01
        w -= step * y * x * der_loss(x, y)
        w0 -= step * y * der_loss(x, y)
```

$$w_{k+1} = w_k - \gamma_k y_i x_i L'(M_i)$$

3. Борьба с переобучением: регуляризация

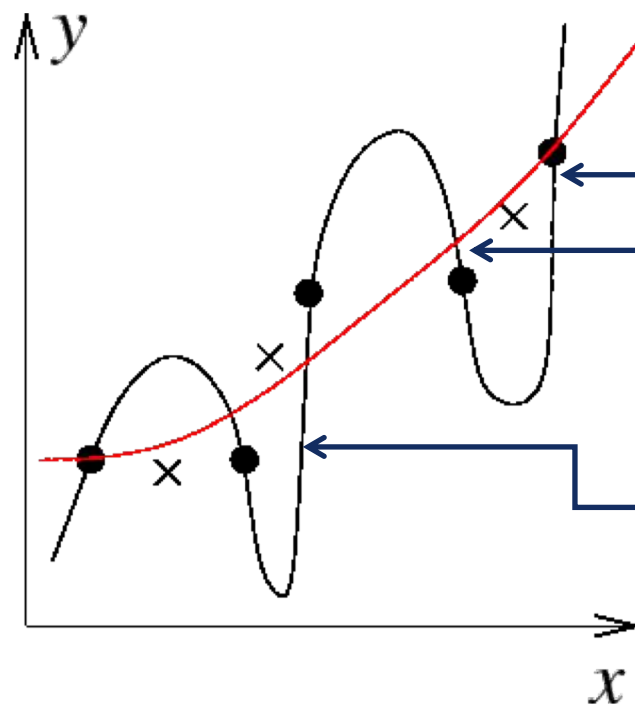
Регуляризация

Переобучение в задаче обучения с учителем:



Регуляризация

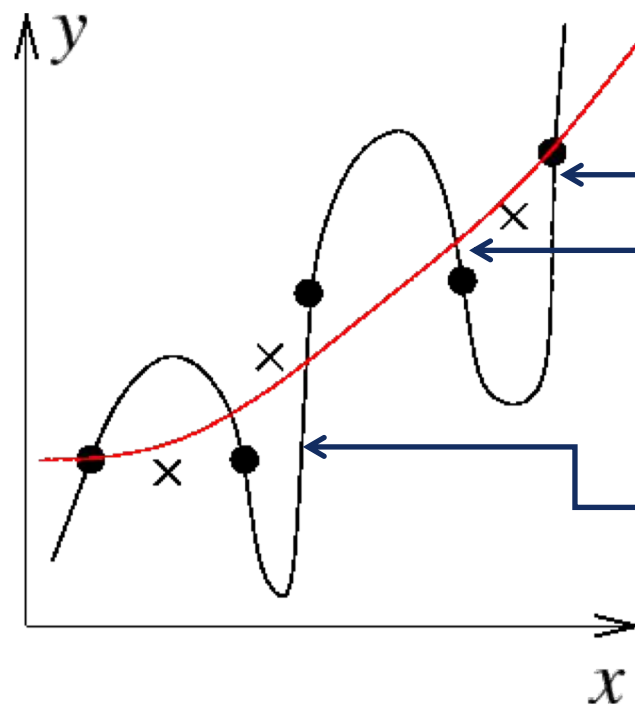
Переобучение в задаче обучения с учителем связано с большими коэффициентами:



Большие коэффициенты делают возможным сильное изменение величины при небольшом изменении признаков

Регуляризация

Переобучение в задаче обучения с учителем связано с большими коэффициентами:

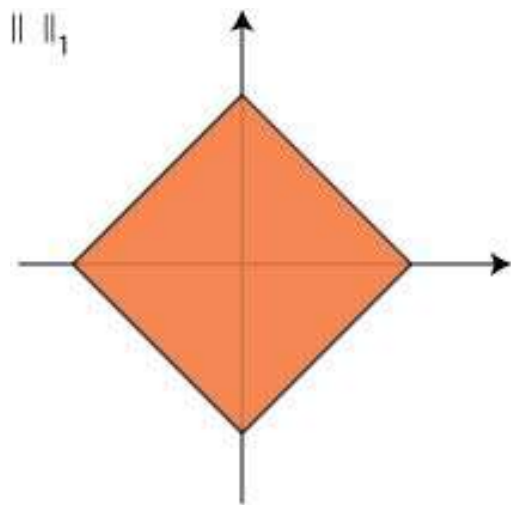


Большие коэффициенты делают возможным сильное изменение величины при небольшом изменении признаков

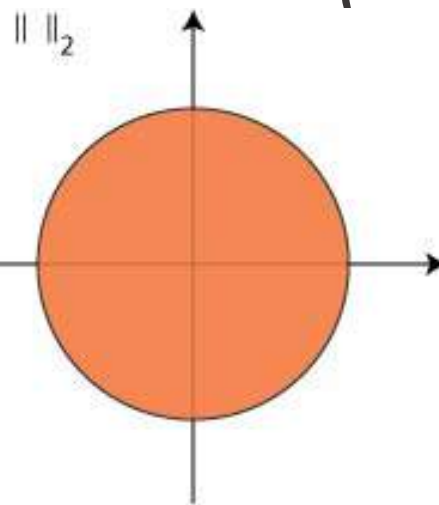
Идея: добавить ограничение на коэффициенты

Регуляризация

$$\left\{ \begin{array}{l} \tilde{Q} = \sum_{i=1}^l L(M_i) \rightarrow \min \\ \sum_{n=1}^d |w_n| \leq \tau \end{array} \right.$$

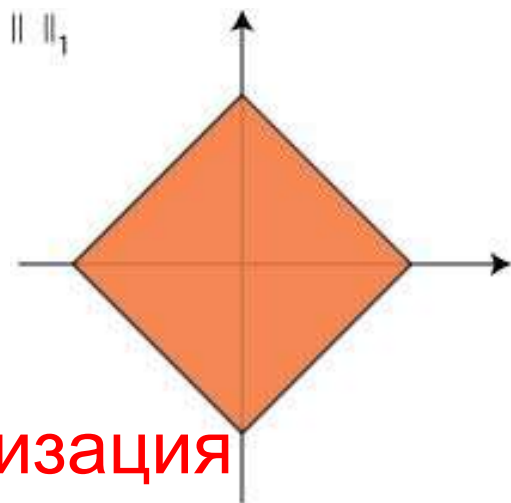


$$\left\{ \begin{array}{l} \tilde{Q} = \sum_{i=1}^l L(M_i) \rightarrow \min \\ \sum_{n=1}^d w_n^2 \leq \tau \end{array} \right.$$



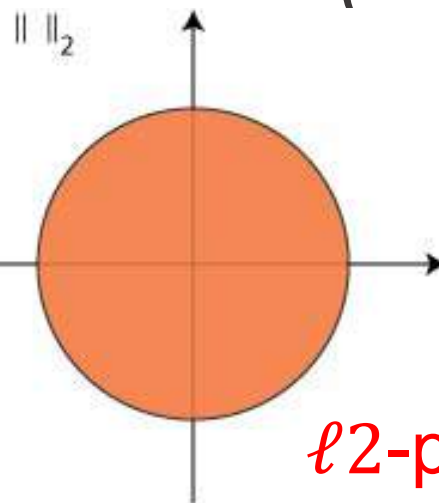
Регуляризация

$$\left\{ \begin{array}{l} \tilde{Q} = \sum_{i=1}^l L(M_i) \rightarrow \min \\ \sum_{n=1}^d |w_n| \leq \tau \end{array} \right.$$



ℓ_1 -регуляризация

$$\left\{ \begin{array}{l} \tilde{Q} = \sum_{i=1}^l L(M_i) \rightarrow \min \\ \sum_{n=1}^m w_n^2 \leq \tau \end{array} \right.$$

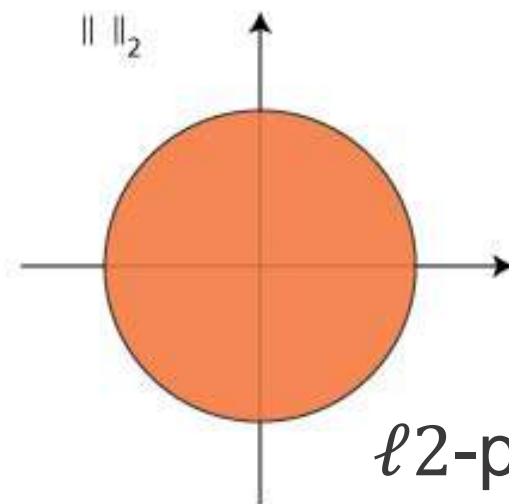
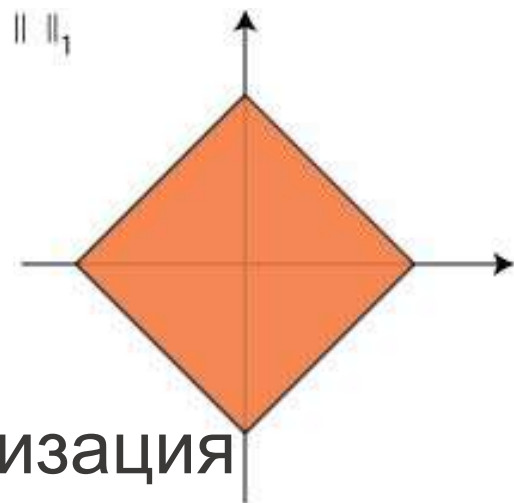


ℓ_2 -регуляризация

Регуляризация

$$\sum_{i=1}^l L(M_i) + \gamma \sum_{n=1}^d |w_n| \rightarrow \min$$

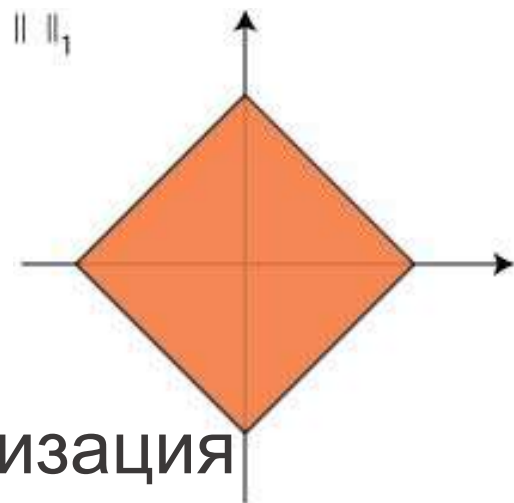
$$\sum_{i=1}^l L(M_i) + \gamma \sum_{n=1}^d w_n^2 \rightarrow \min$$



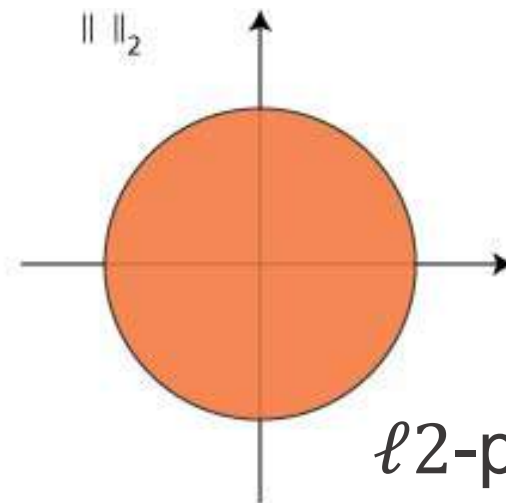
Регуляризация

$$\sum_{i=1}^l L(M_i) + \gamma \sum_{n=1}^d |w_n| \rightarrow \min$$

$$\sum_{i=1}^l L(M_i) + \gamma \sum_{n=1}^d w_n^2 \rightarrow \min$$



$\ell 1$ -регуляризация



$\ell 2$ -регуляризация

Вопрос:

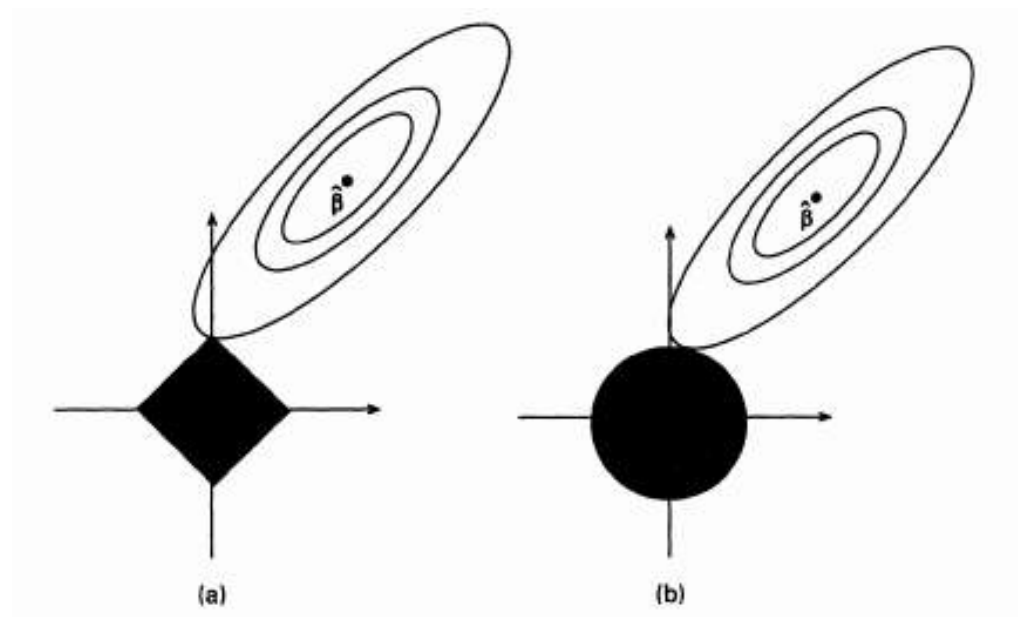
вы заметили, что в регуляризатор не включается вес w_o ?

Различия между ℓ_1 и ℓ_2

- Разреженность – ℓ_1 -регуляризация делает вектор весов более **разреженным** (содержащим больше нулей)
- В случае линейной классификации это означает **отбор признаков**: признаки с нулевыми весами не используются в классификации

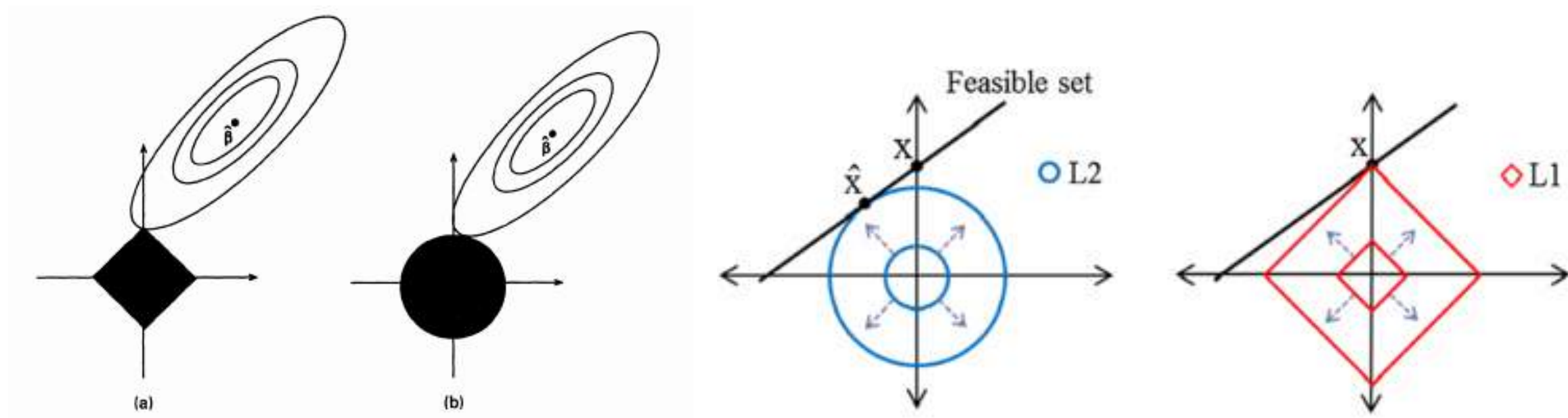
Различия между ℓ_1 и ℓ_2

- Разреженность – ℓ_1 -регуляризация делает вектор весов более **разреженным** (содержащим больше нулей)
- В случае линейной классификации это означает **отбор признаков**: признаки с нулевыми весами не используются в классификации



Различия между ℓ_1 и ℓ_2

- Разреженность – ℓ_1 -регуляризация делает вектор весов более **разреженным** (содержащим больше нулей)
- В случае линейной классификации это означает **отбор признаков**: признаки с нулевыми весами не используются в классификации



Стандартные линейные классификаторы

| Классификатор | Функция потерь | Регуляризатор |
|------------------------------------------------------|---------------------------------------|------------------------------------------------------|
| SVM (Support vector machine, метод опорных векторов) | $L(M) = \max\{0, 1 - M\} = (1 - M)_+$ | $\sum_{k=1}^m w_k^2$ |
| Логистическая регрессия | $L(M) = \log(1 + e^{-M})$ | Обычно $\sum_{k=1}^m w_k^2$ или $\sum_{k=1}^m w_k $ |

Обязательно ли L – функция от отступа?

Пример:

$$y_i \in \{0, 1\} \quad Q = - \sum_{i=1}^{\ell} y_i \ln p_i + (1 - y_i) \ln(1 - p_i) \rightarrow \min_w$$

$$p_i = \sigma(\langle w, x_i \rangle) = \frac{1}{1 + e^{-\langle w, x_i \rangle}}$$

Обязательно ли L – функция от отступа?

Пример:

$$y_i \in \{0, 1\} \quad Q = - \sum_{i=1}^{\ell} y_i \ln p_i + (1 - y_i) \ln(1 - p_i) \rightarrow \min_w$$

$$p_i = \sigma(\langle w, x_i \rangle) = \frac{1}{1 + e^{-\langle w, x_i \rangle}}$$

Упражнение:

Показать, что это та же оптимизационная задача, что и в логистической регрессии

Общий случай

$$Q = \sum_{i=1}^{\ell} L(y_i, f(x_i)) + \gamma V(w) \rightarrow \min_w$$

Функция потерь



Коэффициент
регуляризации



Регуляризатор



Упражнение

Выпишите, как поменяется правило обновления весов признаков в линейном классификаторе с помощью SGD при добавлении регуляризатора

5. Линейные модели в задаче регрессии

Линейные модели в регрессии

$$a(x) = \langle w, x \rangle + w_0$$

Линейные модели в регрессии

$$a(x) = \langle w, x \rangle + w_0$$

$$Q = \sum_{i=1}^l L(y_i, a(x_i)) \rightarrow \min_w$$

Линейные модели в регрессии

$$a(x) = \langle w, x \rangle + w_0$$

$$Q = \sum_{i=1}^l L(y_i, a(x_i)) \rightarrow \min_w$$

$$L(y_i, a(x_i)) = (y_i - a(x_i))^2 \quad L(y_i, a(x_i)) = |y_i - a(x_i)|$$

Линейные модели в регрессии

$$a(x) = \langle w, x \rangle + w_0$$

$$Q = \sum_{i=1}^l L(y_i, a(x_i)) + \gamma V(w) \rightarrow \min_w$$

$$L(y_i, a(x_i)) = (y_i - a(x_i))^2 \qquad L(y_i, a(x_i)) = |y_i - a(x_i)|$$

$$V(w) = \|w\|_{l_2}^2 = \sum_{n=1}^d w_n^2$$

$$V(w) = \|w\|_{l_1} = \sum_{n=1}^d |w_n|$$

Линейные модели в регрессии

$$a(x) = \langle w, x \rangle + w_0$$

$$Q = \sum_{i=1}^l L(y_i, a(x_i)) + \gamma V(w) \rightarrow \min_w$$

Гребневая регрессия
(Ridge regression):

$$V(w) = \|w\|_{l_2}^2 = \sum_{n=1}^d w_n^2$$

LASSO (least absolute
shrinkage and selection
operator):

$$V(w) = \|w\|_{l_1} = \sum_{n=1}^d |w_n|$$

Линейная регрессия

$$a(x) = \langle w, x \rangle + w_0$$

$$Q = \sum_{i=1}^l L(y_i, a(x_i)) \rightarrow \min_w$$

$$L(y_i, a(x_i)) = (y_i - a(x_i))^2$$

А без регуляризатора и с квадратичными потерями получаем привычную нам линейную регрессию

Линейная регрессия

Модель: $y_i \approx \hat{y}_i = \langle w, x_i \rangle + w_0$

Линейная регрессия

Модель: $y_i \approx \hat{y}_i = \langle w, x_i \rangle + w_0$

Если добавить $x_{i0} = 1$:

Линейная регрессия

Модель: $y_i \approx \hat{y}_i = \langle w, x_i \rangle + w_0$

Если добавить $x_{i0} = 1$:

$$y_i \approx \hat{y}_i = \langle w, x_i \rangle$$

Линейная регрессия

Модель: $y_i \approx \hat{y}_i = \langle w, x_i \rangle + w_0$

Если добавить $x_{i0} = 1$:
 $y_i \approx \hat{y}_i = \langle w, x_i \rangle$

$$y_1 \approx \hat{y}_1 = x_1^T w$$

...

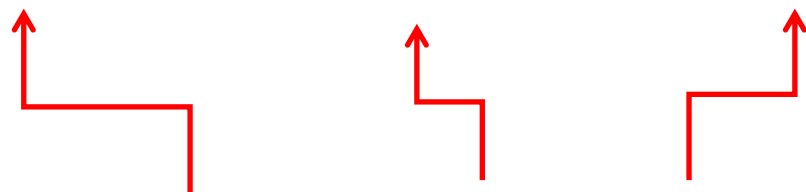
$$y_i \approx \hat{y}_i = x_i^T w$$

...

$$y_l \approx \hat{y}_l = x_l^T w$$

Матричная запись

$$\begin{pmatrix} y_1 \\ y_2 \\ \dots \\ y_l \end{pmatrix} \approx \begin{pmatrix} \hat{y}_1 \\ \hat{y}_2 \\ \dots \\ \hat{y}_l \end{pmatrix} = \begin{pmatrix} x_1^T \\ x_2^T \\ \dots \\ x_l^T \end{pmatrix} w$$


$$y \approx \hat{y} = Fw$$

$$w = \underset{w}{\operatorname{argmin}} \|y - \hat{y}\|^2$$

Веса признаков

$$\frac{\partial (Fw - y)^2}{\partial w} = 2F^T (Fw - y) = 0$$

$$F^T Fw = F^T y$$

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

Если добавить ℓ_2 регуляризацию

$$\frac{\partial (Fw - y)^2 + \gamma w^2}{\partial w} = 2F^T (Fw - y) + 2\gamma w = 0$$

$$(F^T F + \gamma I)w = F^T y$$

$$w = (F^T F + \gamma I)^{-1} F^T y$$

План лекции

1. Линейная классификация

2. Обучение линейной модели

3. Борьба с переобучением

4. Линейная регрессия

Pros & cons

Преимущества:

- легко реализовывать уже обученную модель
- не многим сложнее реализовывать и ее обучение
- быстро работают
- хорошо работают, когда много признаков
- нормально работают, когда мало данных

Недостатки:

- может быть слишком простым для вашей зависимости $y(x)$
- будет плохо работать, если забыть/не суметь отмасштабировать признаки

Библиотеки

- libSVM
- liblinear
- sklearn.linear_models
- Vowpal Wabbit (SGD для онлайн-обучения + Hashing Trick)

Дополнительные темы для изучения

Дополнительные темы: линейные модели

1. Почему l_1 регуляризатор разреживает вектор весов
2. SVM и его вывод из максимизации ширины разделяющей полосы
3. Kernel Trick в SVM и двойственная задача
4. Стратегии построения многоклассовых линейных классификаторов
5. Кросс-энтропия и многоклассовая логистическая регрессия
6. Semi-supervised версии SVM и логистической регрессии

Дополнительные темы: оптимизация

1. Mini-batch SGD
2. Метод Ньютона-Рафсона
3. Квазиньютоновские методы и BFGS
4. Теорема Куна-Таккера и ее геометрическое объяснение, объяснение связи ограничений на веса и добавления штрафа в оптимизационную задачу, двойственная задача в SVM

Дополнительные темы: анализ

1. Запись chain rule в виде $\frac{du}{dx} = \frac{du}{dv} \frac{dv}{dx}$ (связано с понятием дифференциала)

2. Уточнение смысла записи $f(x + \varepsilon) \approx f(x) + f'(x)\varepsilon$:

$$f(x + \varepsilon) = f(x) + f'(x)\varepsilon + o(\varepsilon)$$

(разложение в ряд Тейлора до первого члена)

Дополнительные темы: линейная алгебра

1. Геометрический смысл линейной регрессии (связано с линейной оболочкой, ортогональностью и проекцией на линейное подпространство)
2. Матричная запись линейного классификатора в случае многоклассовой классификации
3. Собственные значения и число обусловленности матрицы, его связь с вычислением обратной матрицы
4. Проблема мультиколлинеарности и связь с регуляризацией

Data Mining in Action

Лекция 2

Группа курса в Telegram:



<https://t.me/joinchat/B1OlTk74nRV56Dp1TDJGNA>