

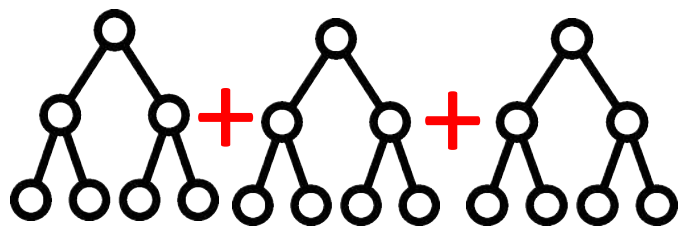
# Data Mining in Action

Лекция 2.

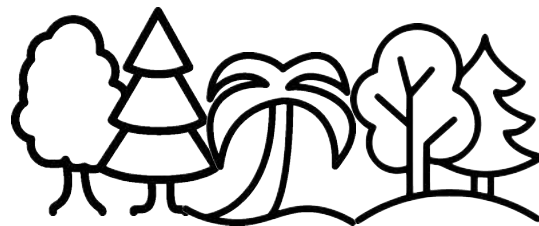
Supervised learning: линейные модели



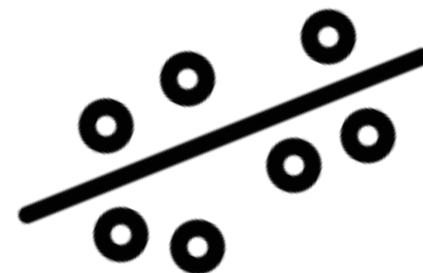
# Напоминание: часто используемые методы



Градиентный бустинг



Случайный лес



Линейные модели

# Обсуждаем сегодня: линейные модели

- I. Линейная классификация и оптимизационная задача в ней
- II. Как настраиваются коэффициенты: SGD
- III. Как бороться с переобучением: регуляризация
- IV. Стандартные линейные классификаторы
- V. О линейных моделях в регрессии

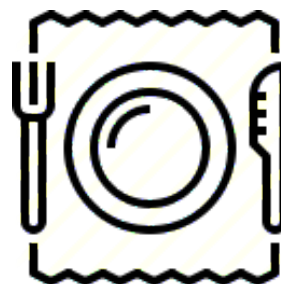
# I. Линейная классификация

# Пример: выходить из дома или нет

Признаки (1/0):



Вы свободны в  
данный момент



Вам хочется  
где-то поехать



Вам хочется  
спать



Вам хочется  
увидеться с друзьями

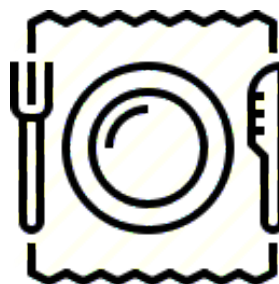


# Пример: выходить из дома или нет

Признаки (1/0):



+1



Вы свободны в  
данный момент

Вам хочется  
где-то поесть

Вам хочется  
спать

Вам хочется  
увидеться с друзьями

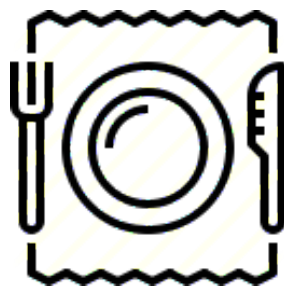
# Пример: выходить из дома или нет

Признаки (1/0):



+1

Вы свободны в  
данный момент



+2

Вам хочется  
где-то поехать



Вам хочется  
спать



Вам хочется  
увидеться с друзьями

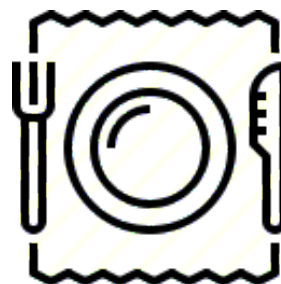
# Пример: выходить из дома или нет

Признаки (1/0):



+1

Вы свободны в  
данный момент



+2

Вам хочется  
где-то поехать



-3

Вам хочется  
спать



Вам хочется  
увидеться с друзьями



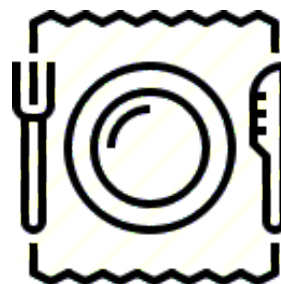
# Пример: выходить из дома или нет

Признаки (1/0):



+1

Вы свободны в  
данный момент



+2

Вам хочется  
где-то поехать



-3

Вам хочется  
спать



+4

Вам хочется  
увидеться с друзьями

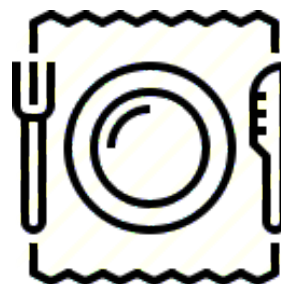
# Пример: выходить из дома или нет

Признаки (1/0):



+1

Вы свободны в  
данный момент



+2

Вам хочется  
где-то поехать



-3

Вам хочется  
спать



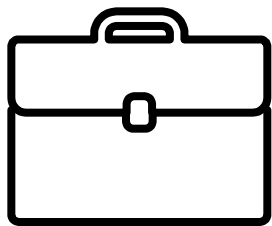
+4

Вам хочется  
увидеться с друзьями

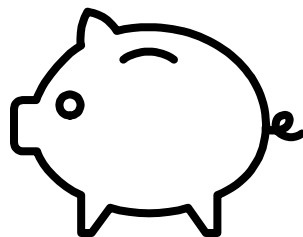
Порог для решающего правила: +1  
Если сумма больше – выходим :)

# Более серьезный пример: дать ли кредит

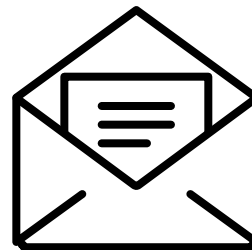
Признаки (1/0):



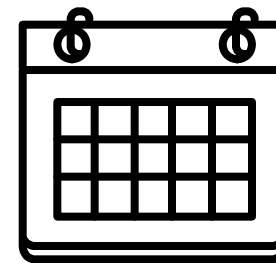
Работоспособный  
возраст



Имеет счет в вашем  
банке



Много просрочек по  
другим кредитам



Просрочек нет, а  
кредиты есть

# Скоринговые карты

ПОКАЗА- ТЕЛЬ	ДИАПАЗОН ЗНАЧЕНИЙ
Возраст заемщика	До 35 лет
	От 35 до 45 лет
	От 45 и старше
Образова- ние	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

ПОКАЗА- ТЕЛЬ	ДИАПАЗОН ЗНАЧЕНИЙ	СКОРИНГ- БАЛЛ
Возраст заемщика	До 35 лет	7,60
	От 35 до 45 лет	29,68
	От 45 и старше	35,87
Образова- ние	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

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

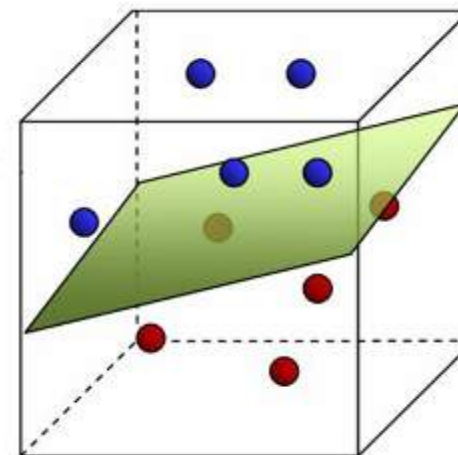
$$f(x) = w_0 + w_1x_1 + \dots + w_dx_d$$

# 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



# 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:

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

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



# 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 0
```

# 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

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

# Функция потерь

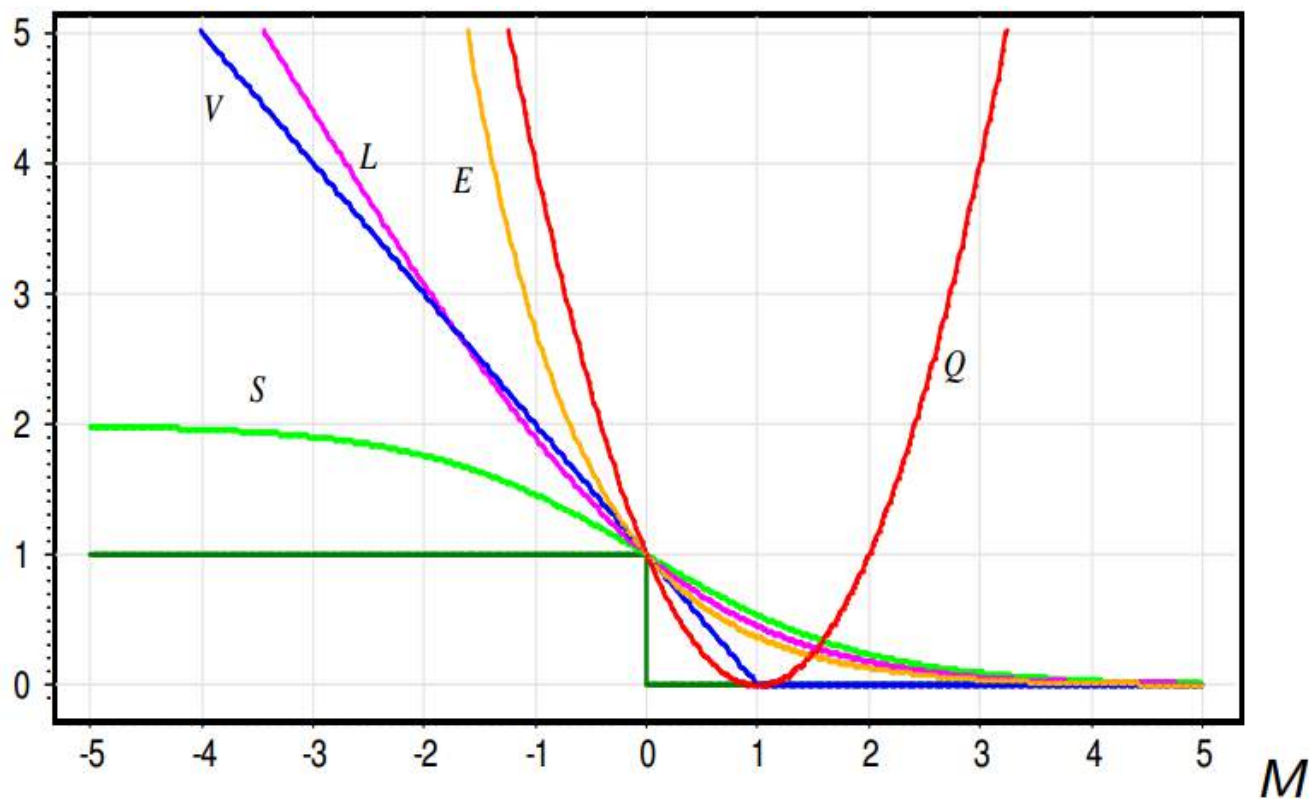
$$Q(w) = \sum_{i=1}^{\ell} [M_i(w) < 0] \leq \tilde{Q}(w) = \sum_{i=1}^{\ell} \mathcal{L}(M_i(w)) \rightarrow \min_w;$$

Функция эмпирического риска

Функция потерь

# Функция потерь

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



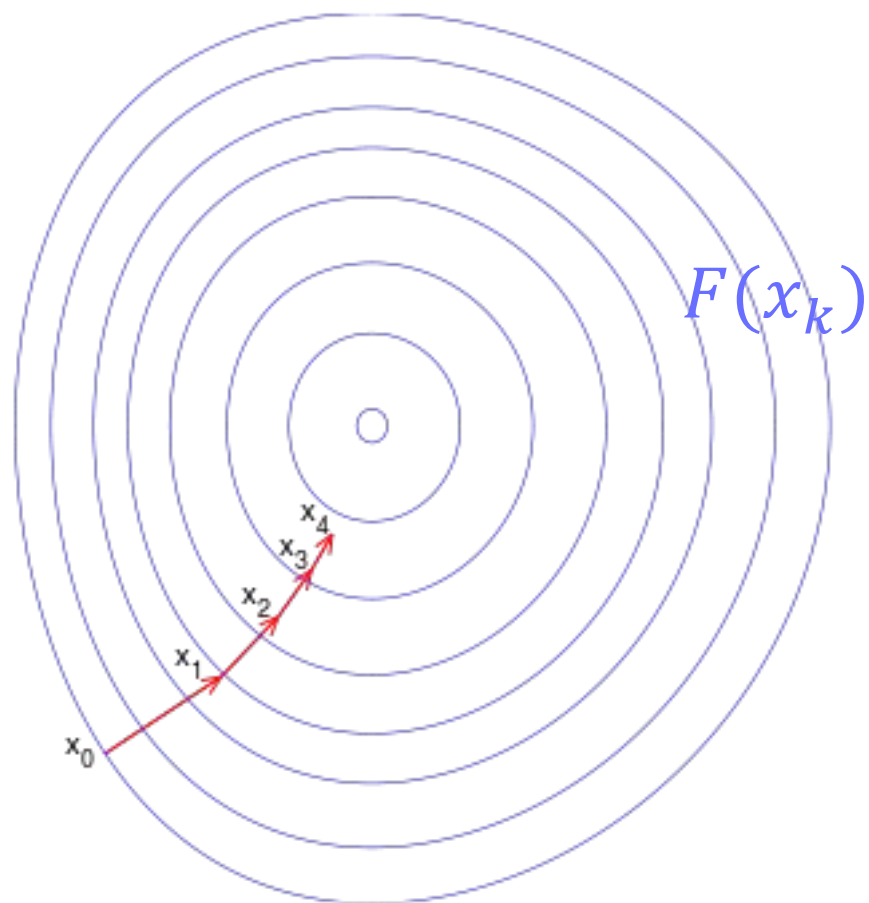
$$\begin{aligned} Q(M) &= (1 - M)^2 \\ V(M) &= (1 - M)_+ \\ S(M) &= 2(1 + e^M)^{-1} \\ L(M) &= \log_2(1 + e^{-M}) \\ E(M) &= e^{-M} \end{aligned}$$



## II. Обучение модели: SGD

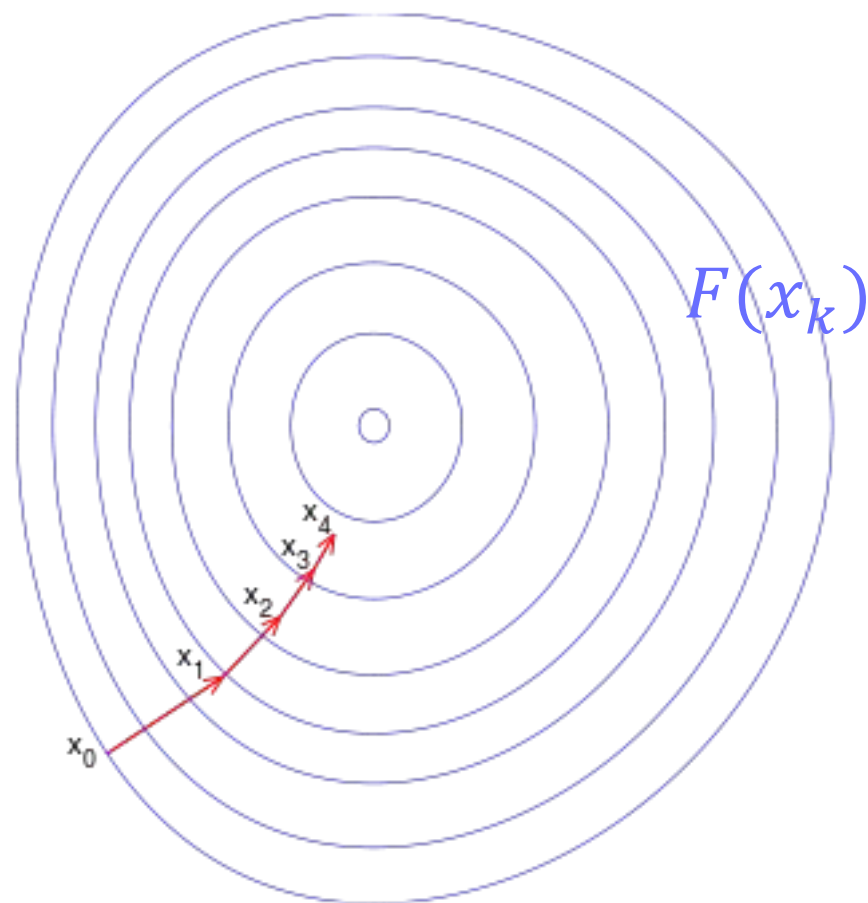
# Градиентный спуск (GD, Gradient Decent)

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



# Градиентный спуск (GD, Gradient Decent)

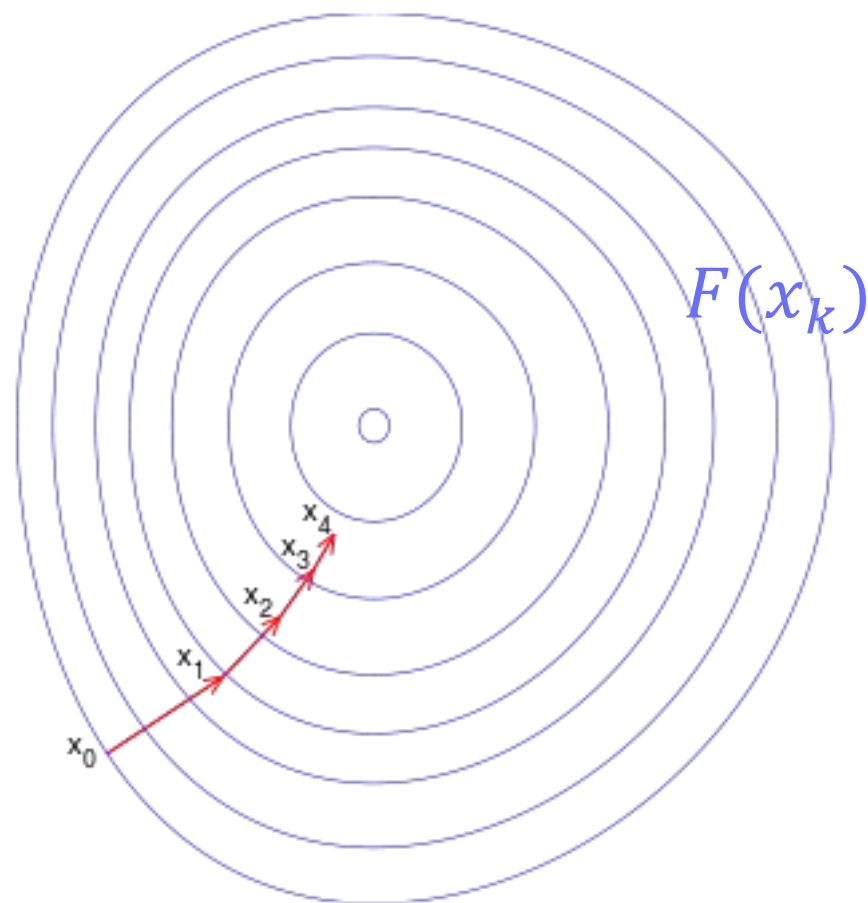
$$x_{k+1} = x_k - \gamma_k \nabla F(x_k)$$



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

# Градиентный спуск (GD, Gradient Decent)

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

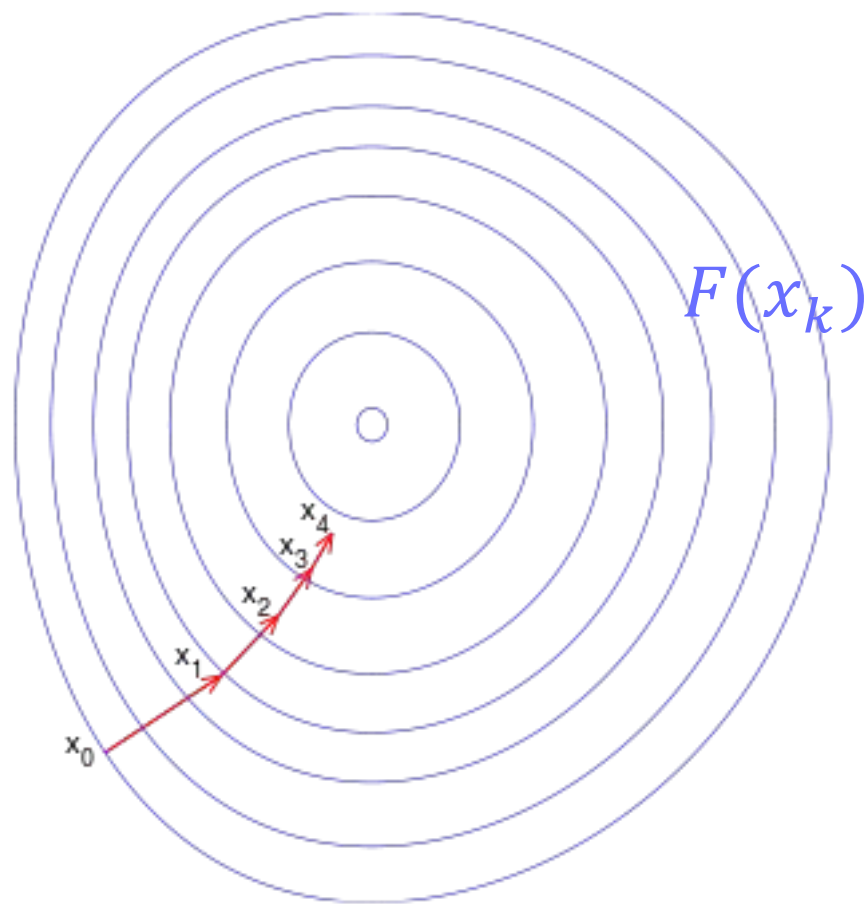


$$\nabla_w \tilde{Q} = \sum_{i=1}^l \nabla L(M_i) = \sum_{i=1}^l L'(M_i) \frac{\partial M_i}{\partial w}$$

$$M_i = y_i \langle w, x_i \rangle$$

# Градиентный спуск (GD, Gradient Decent)

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



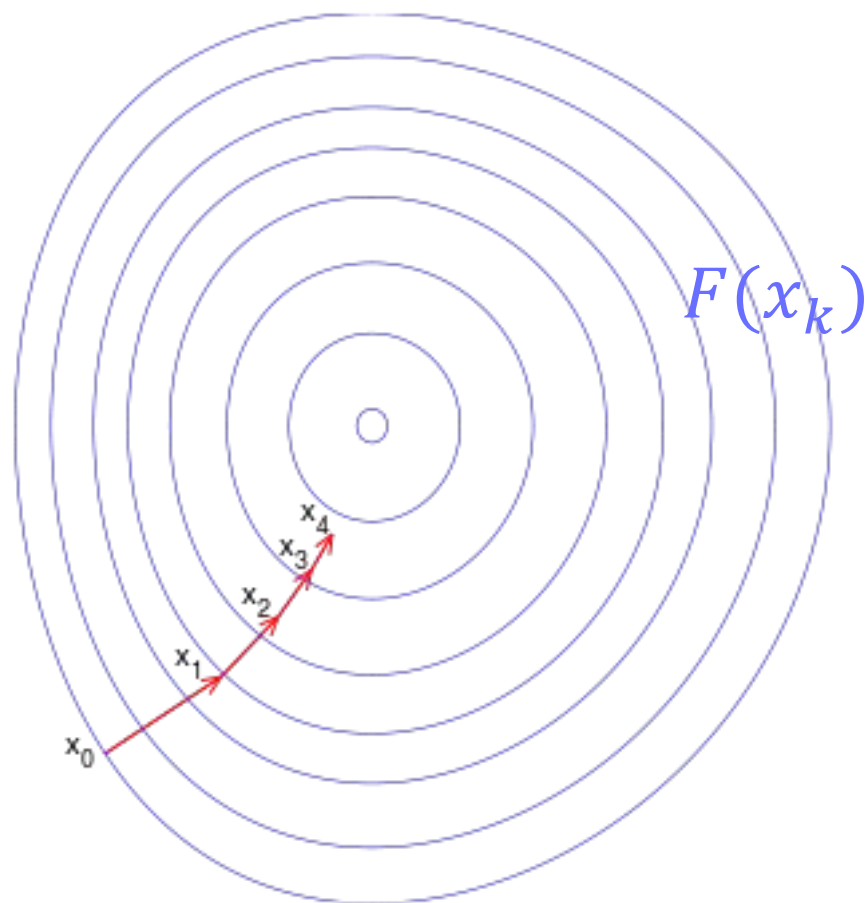
$$\nabla_w \tilde{Q} = \sum_{i=1}^l \nabla L(M_i) = \sum_{i=1}^l L'(M_i) \frac{\partial M_i}{\partial w}$$

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



# Градиентный спуск (GD, Gradient Decent)

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



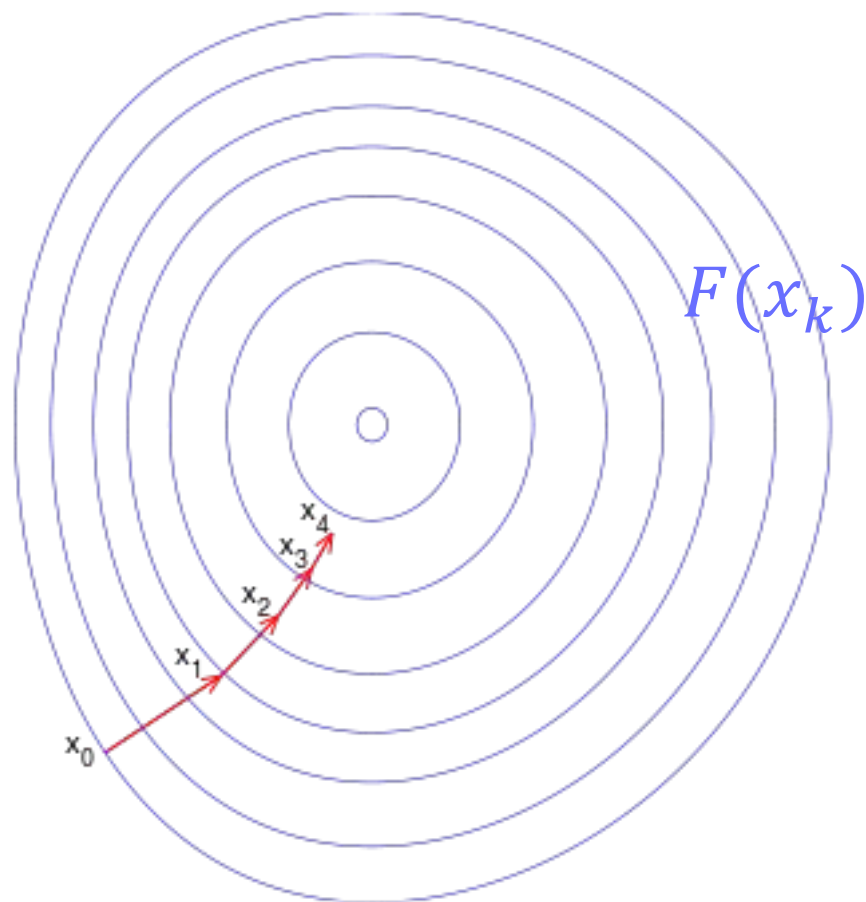
$$\nabla_w \tilde{Q} = \sum_{i=1}^l \nabla L(M_i) = \sum_{i=1}^l L'(M_i) \frac{\partial M_i}{\partial w}$$

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

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

# 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

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

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

def fit(X_train, y_train):
    for k in range(10000):
        rand_index = randint(0, len(X_train))
        x = X_train[rand_index]
        y = y_train[rand_index]

        step = 0.01

        w -= x * step * y * der_loss(x, y)
```



# Как выглядит код: обучение модели

```
from random import randint
```

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

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

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

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



# Как выглядит код: обучение модели

```
from random import randint

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

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

def fit(X_train, y_train):
    for k in range(10000):
        rand_index = randint(0, len(X_train))
        x = X_train[rand_index]
        y = y_train[rand_index]

        step = 0.01

        w -= x * step * y * der_loss(x, y)
```

# Как выглядит код: обучение модели

```
from random import randint

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

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

def fit(X_train, y_train):
    for k in range(10000):
        rand_index = randint(0, len(X_train))
        x = X_train[rand_index]
        y = y_train[rand_index]

        step = 0.01

        w -= x * step * y * der_loss(x, y)
```

# Как выглядит код: обучение модели

```
from random import randint

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

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

def fit(X_train, y_train):
    for k in range(10000):
        rand_index = randint(0, len(X_train))
        x = X_train[rand_index]
        y = y_train[rand_index]

        step = 0.01

        w -= x * step * y * der_loss(x, y)
```



# Как выглядит код: обучение модели

```
from random import randint

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

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

def fit(X_train, y_train):
    for k in range(10000):
        rand_index = randint(0, len(X_train))
        x = X_train[rand_index]
        y = y_train[rand_index]

        step = 0.01

        w -= x * step * y * der_loss(x, y)
```

# Как выглядит код: обучение модели

```
from random import randint

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

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

def fit(X_train, y_train):
    for k in range(10000):
        rand_index = randint(0, len(X_train))
        x = X_train[rand_index]
        y = y_train[rand_index]

        step = 0.01

        w -= x * step * y * der_loss(x, y)
```



# Как выглядит код: обучение модели

```
from random import randint

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

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

def fit(X_train, y_train):
    for k in range(10000):
        rand_index = randint(0, len(X_train))
        x = X_train[rand_index]
        y = y_train[rand_index]

        step = 0.01

        w -= x * step * y * der_loss(x, y)
```

# Как выглядит код: обучение модели

```
from random import randint

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

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

def fit(X_train, y_train):
    for k in range(10000):
        rand_index = randint(0, len(X_train))
        x = X_train[rand_index]
        y = y_train[rand_index]

        step = 0.01

        w -= x * step * y * der_loss(x, y)
```

# Как выглядит код: обучение модели

```
from random import randint

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

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

def fit(X_train, y_train):
    for k in range(10000):
        rand_index = randint(0, len(X_train))
        x = X_train[rand_index]
        y = y_train[rand_index]
```

```
    step = 0.01
```

```
    w -= x * step * y * der_loss(x, y)
```

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



# Как выглядит код: обучение модели

```
from random import randint

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

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

def fit(X_train, y_train):
    for k in range(10000):
        rand_index = randint(0, len(X_train))
        x = X_train[rand_index]
        y = y_train[rand_index]
```

```
    step = 0.01
```

```
    w -= x * 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

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

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

def fit(X_train, y_train):
    for k in range(10000):
        rand_index = randint(0, len(X_train))
        x = X_train[rand_index]
        y = y_train[rand_index]
```

```
    step = 0.01
```

```
    w -= x * 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

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

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

def fit(X_train, y_train):
    for k in range(10000):
        rand_index = randint(0, len(X_train))
        x = X_train[rand_index]
        y = y_train[rand_index]
```

```
    step = 0.01
```

```
    w -= x * 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

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

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

def fit(X_train, y_train):
    for k in range(10000):
        rand_index = randint(0, len(X_train))
        x = X_train[rand_index]
        y = y_train[rand_index]
```

```
        step = 0.01
```

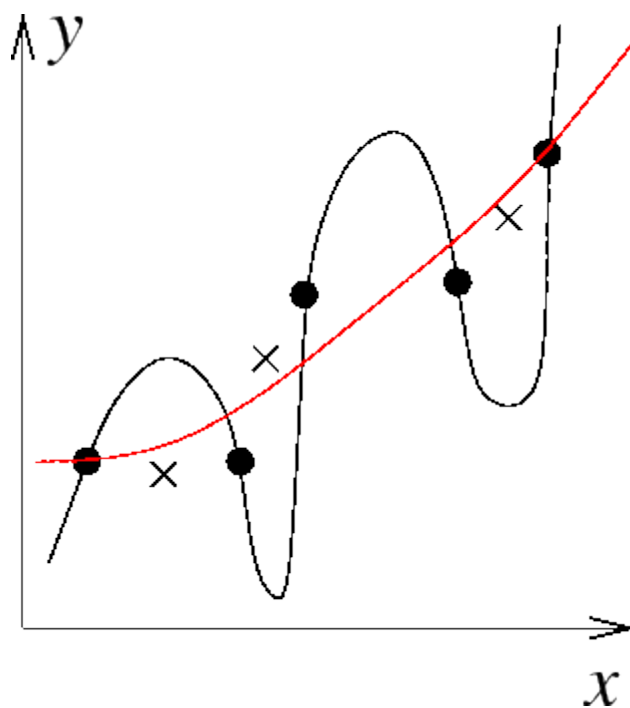
$$w_{k+1} = w_k - \gamma_k y_i x_i L'(M_i)$$

```
        w -= x * step * y * der_loss(x, y)
```

### III. Борьба с переобучением: регуляризация

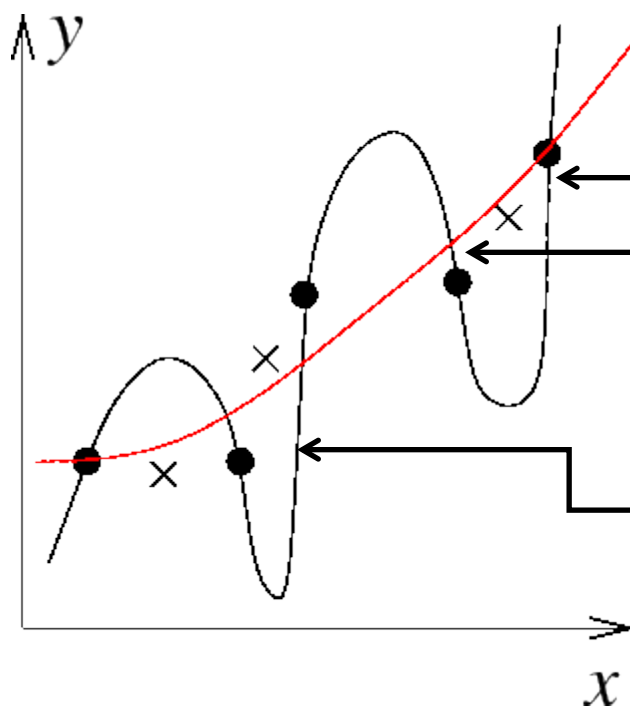
# Регуляризация

Переобучение в задаче обучения с учителем:



# Регуляризация

Переобучение в задаче обучения с учителем связано с большими коэффициентами:

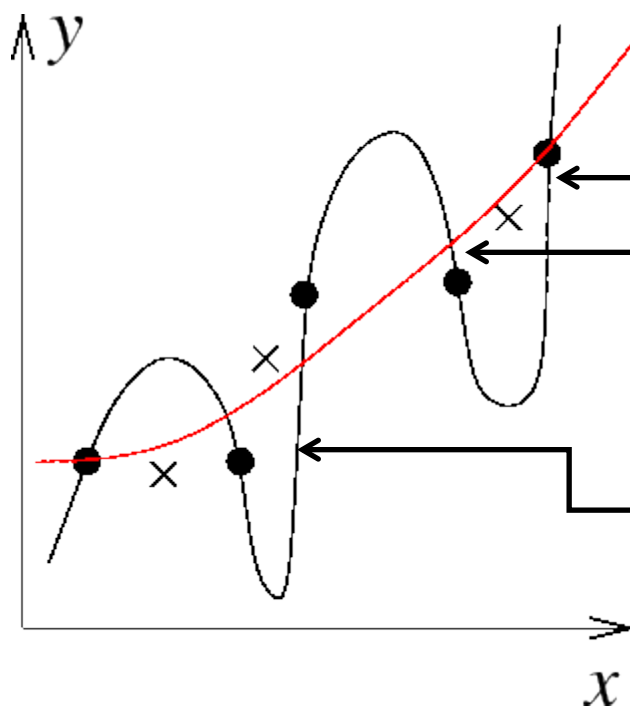


Большие коэффициенты делают возможным сильное изменение величины при небольшом изменении признаков



# Регуляризация

Переобучение в задаче обучения с учителем связано с большими коэффициентами:

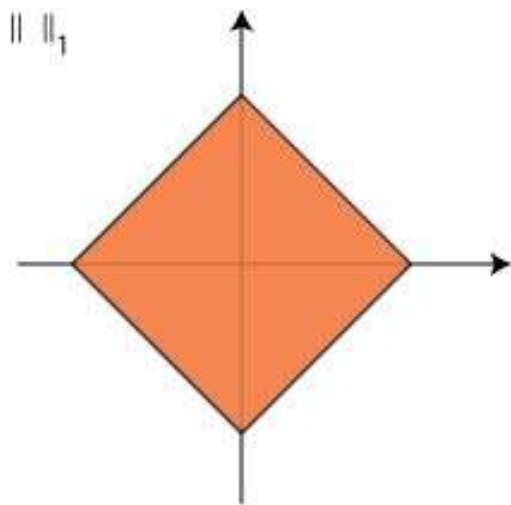


Большие коэффициенты делают возможным сильное изменение величины при небольшом изменении признаков

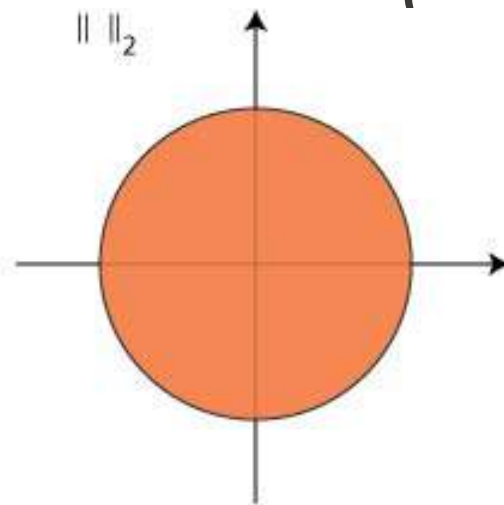
Идея: добавить ограничение на коэффициенты

# Регуляризация

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

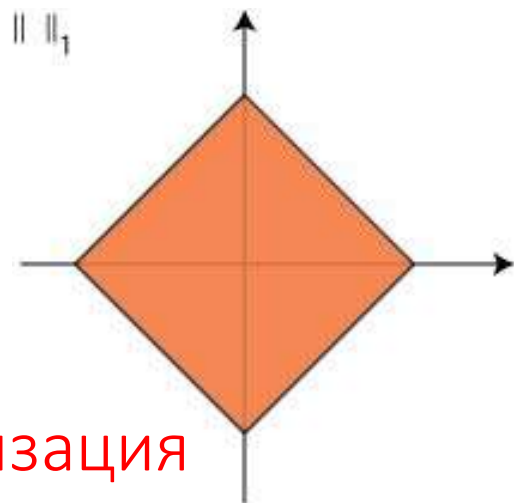


$$\left\{ \begin{array}{l} \tilde{Q} = \sum_{i=1}^l L(M_i) \rightarrow \min \\ \sum_{n=1}^d w_n^2 \leq \tau \end{array} \right.$$



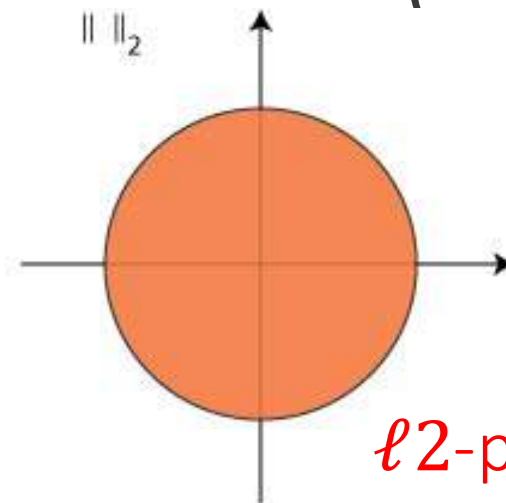
# Регуляризация

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



$\ell_1$ -регуляризация

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

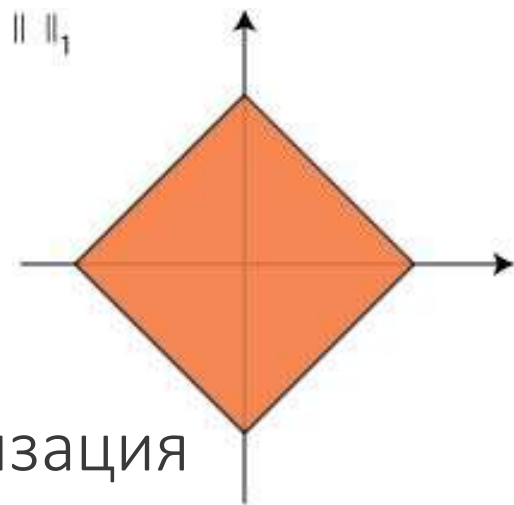


$\ell_2$ -регуляризация

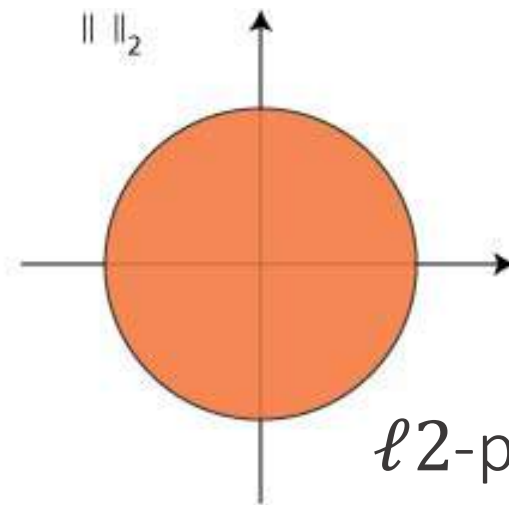
# Регуляризация

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

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



$\ell 1$ -регуляризация

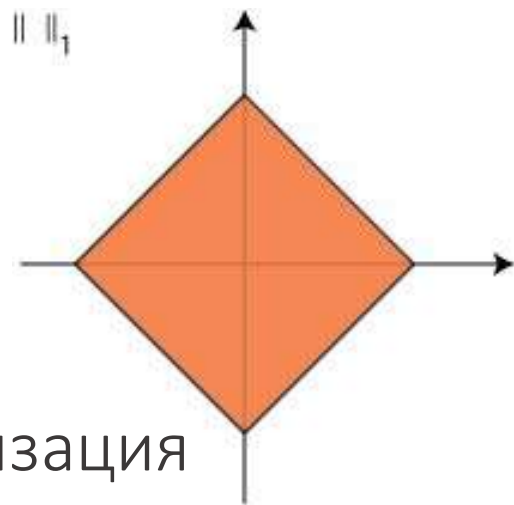


$\ell 2$ -регуляризация

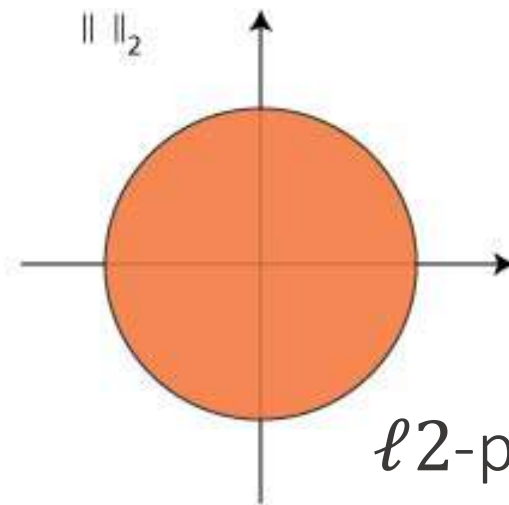
# Регуляризация

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

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



$\ell_1$ -регуляризация



$\ell_2$ -регуляризация

## Вопрос:

вы заметили, что  
в регуляризатор  
не включается вес  
 $w_0$ ?

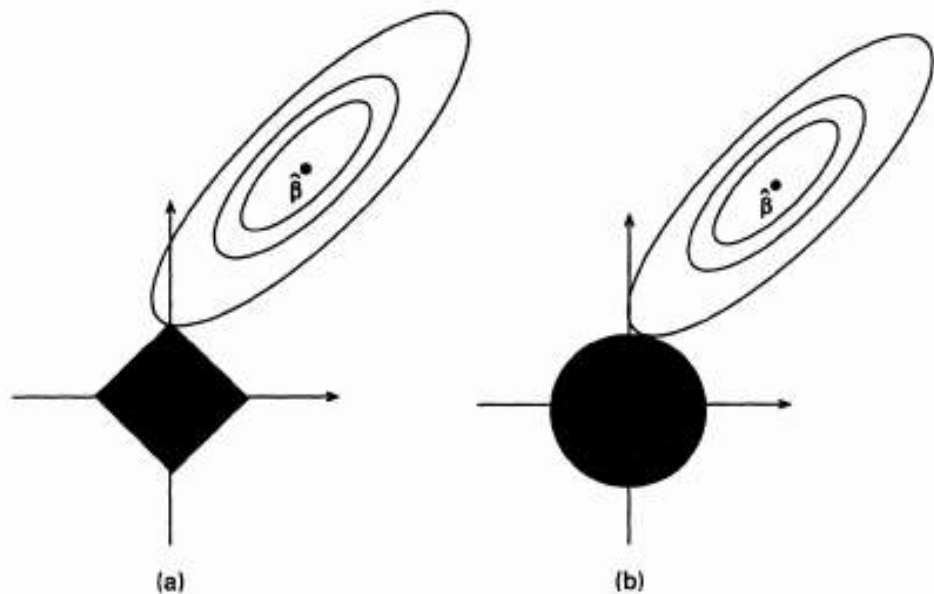


# Различия между $\ell_1$ и $\ell_2$

- Разреженность –  $\ell_1$ -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

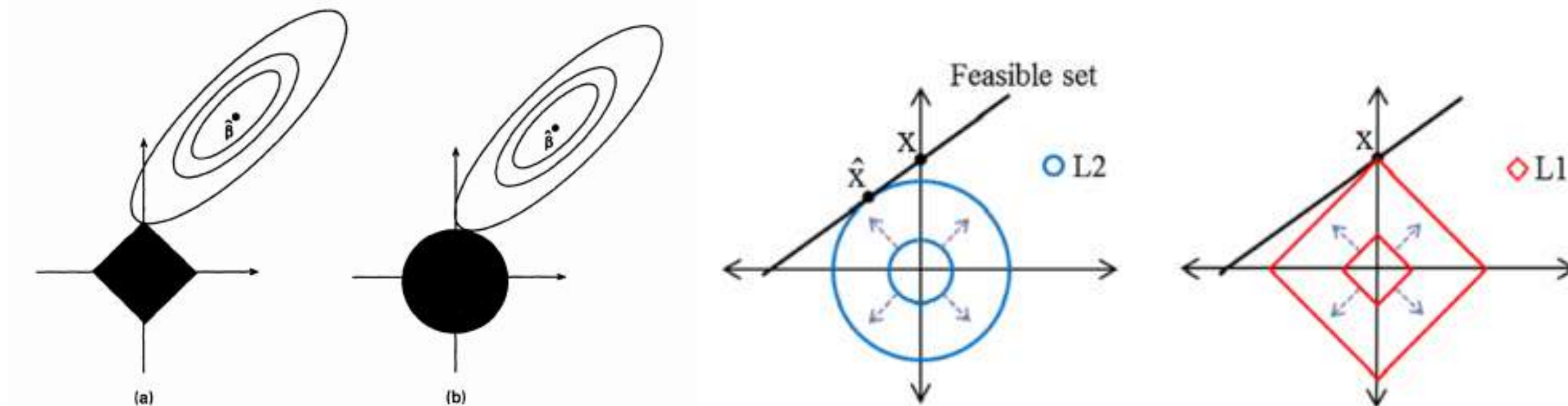
# Различия между $\ell_1$ и $\ell_2$

- Разреженность –  $\ell_1$ -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 $\ell_1$ и $\ell_2$

- Разреженность –  $\ell_1$ -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



# Упражнение

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

## IV. Стандартные классификаторы: SVM и логистическая регрессия



# 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 функция потерь – 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 функция потерь – 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

# Метод опорных векторов (SVM)

Линейный классификатор:

$$a(x) = \text{sign}(\langle w, x \rangle - w_0)$$

Использующий кусочно-линейную функцию потерь и  $\ell_2$ -регуляризатор:



# Метод опорных векторов (SVM)

Линейный классификатор:

$$a(x) = \text{sign}(\langle w, x \rangle - w_0)$$

Использующий кусочно-линейную функцию потерь и  $\ell_2$ -регуляризатор:

$$\sum_{i=1}^l L(M_i) + \gamma \|w\|^2 \rightarrow \min_w$$

# Метод опорных векторов (SVM)

Линейный классификатор:

$$a(x) = \text{sign}(\langle w, x \rangle - w_0)$$

Использующий кусочно-линейную функцию потерь и  $\ell_2$ -регуляризатор:

$$\sum_{i=1}^l L(M_i) + \gamma \boxed{\|w\|^2} \rightarrow \min_w$$

квадратичный  
регуляризатор

# Метод опорных векторов (SVM)

Линейный классификатор:

$$a(x) = \text{sign}(\langle w, x \rangle - w_0)$$

Использующий кусочно-линейную функцию потерь и  $\ell_2$ -регуляризатор:

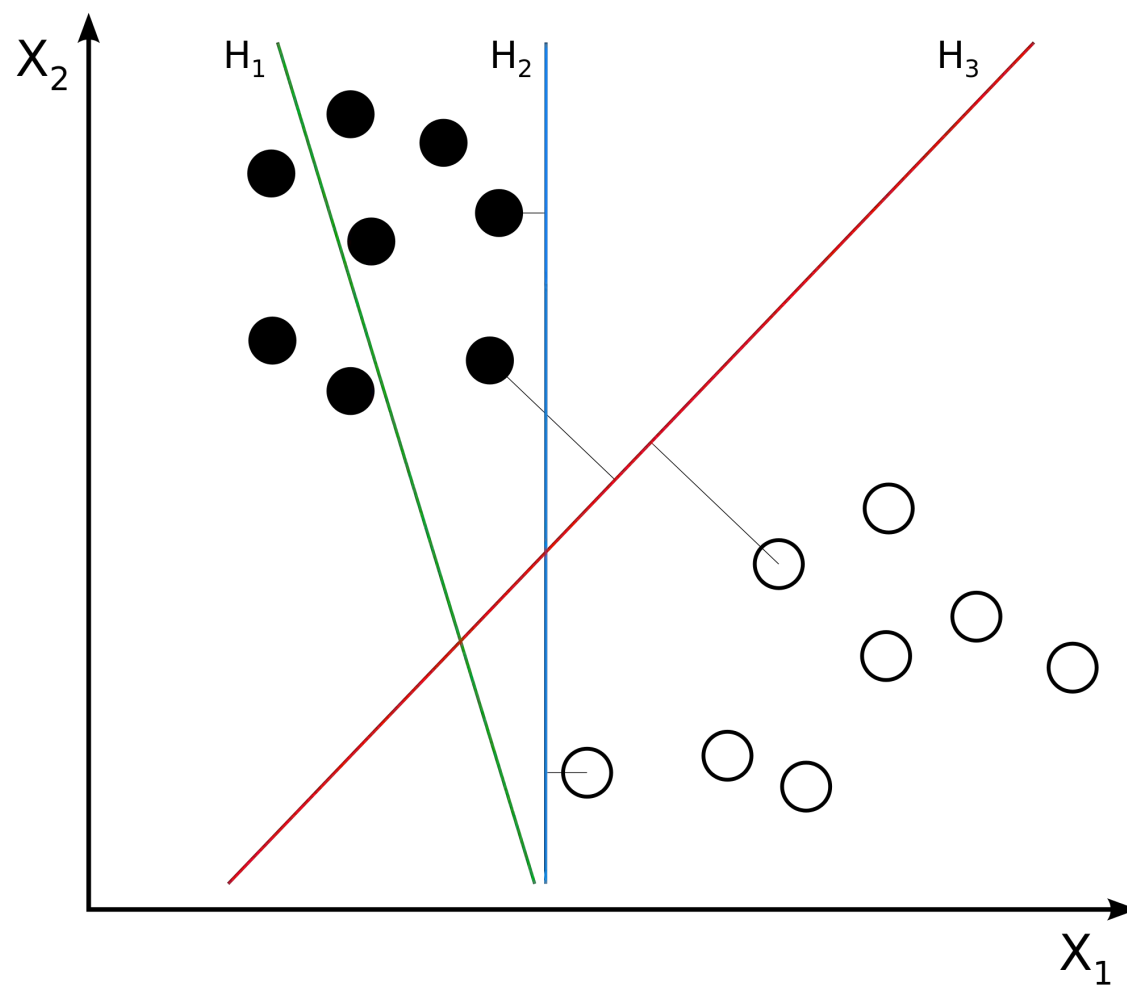
$$\sum_{i=1}^l L(M_i) + \gamma \|w\|^2 \rightarrow \min_w$$

квадратичный  
регуляризатор

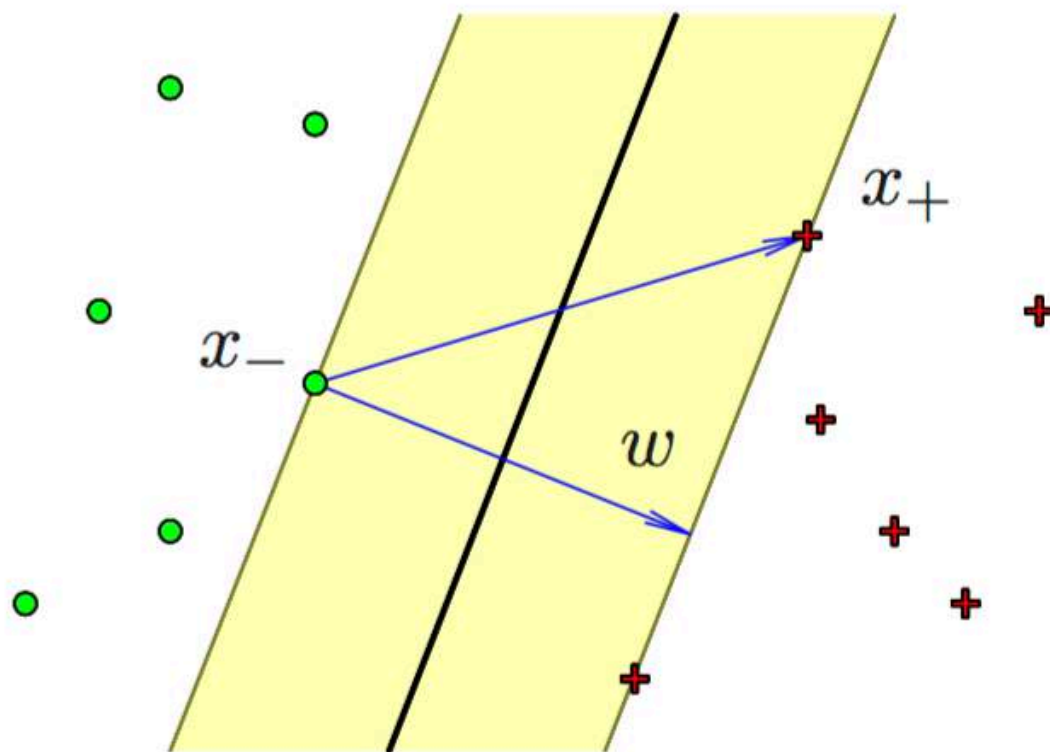
кусочно-линейная функция потерь:

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

# Построение разделяющей гиперплоскости



# Разделяющая полоса

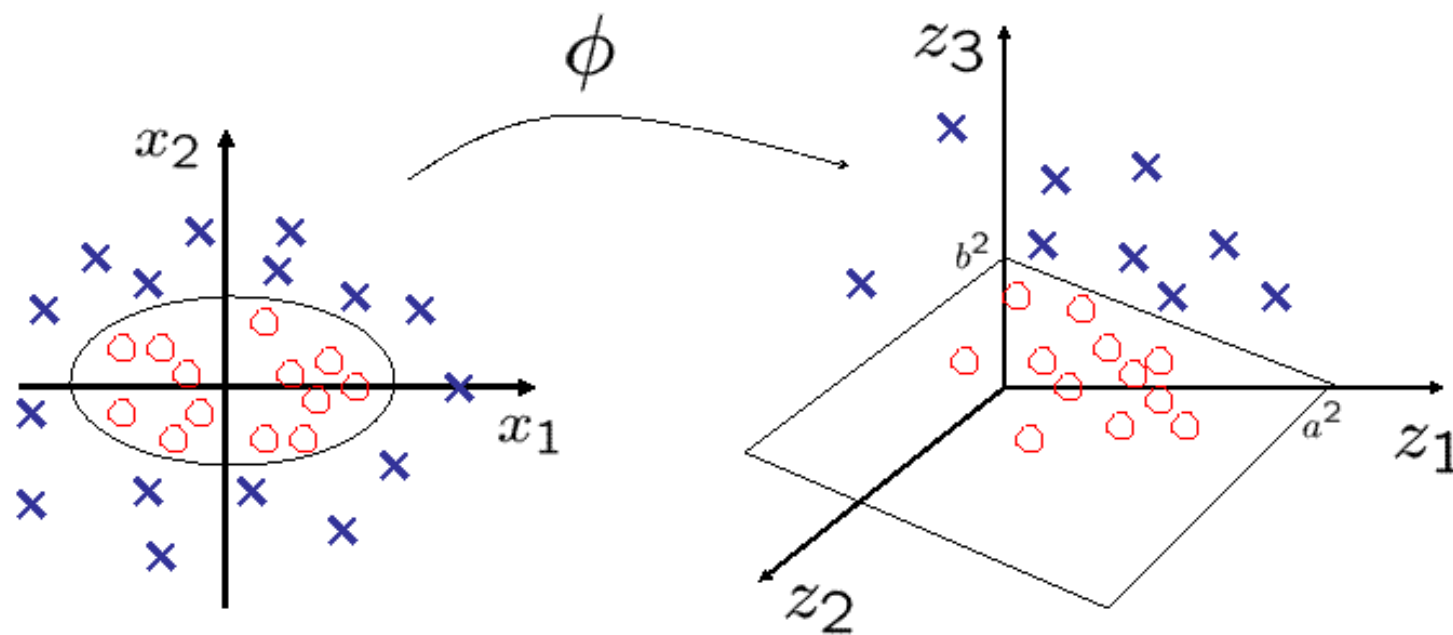




# Ключевые моменты

- Метод опорных векторов – линейный классификатор с кусочно-линейной функцией потерь (hinge loss) и L2-регуляризатором
- Изначально метод был предложен из соображений максимизации зазора между классами
- Позволяет строить нелинейные разделяющие поверхности (об этом дальше)

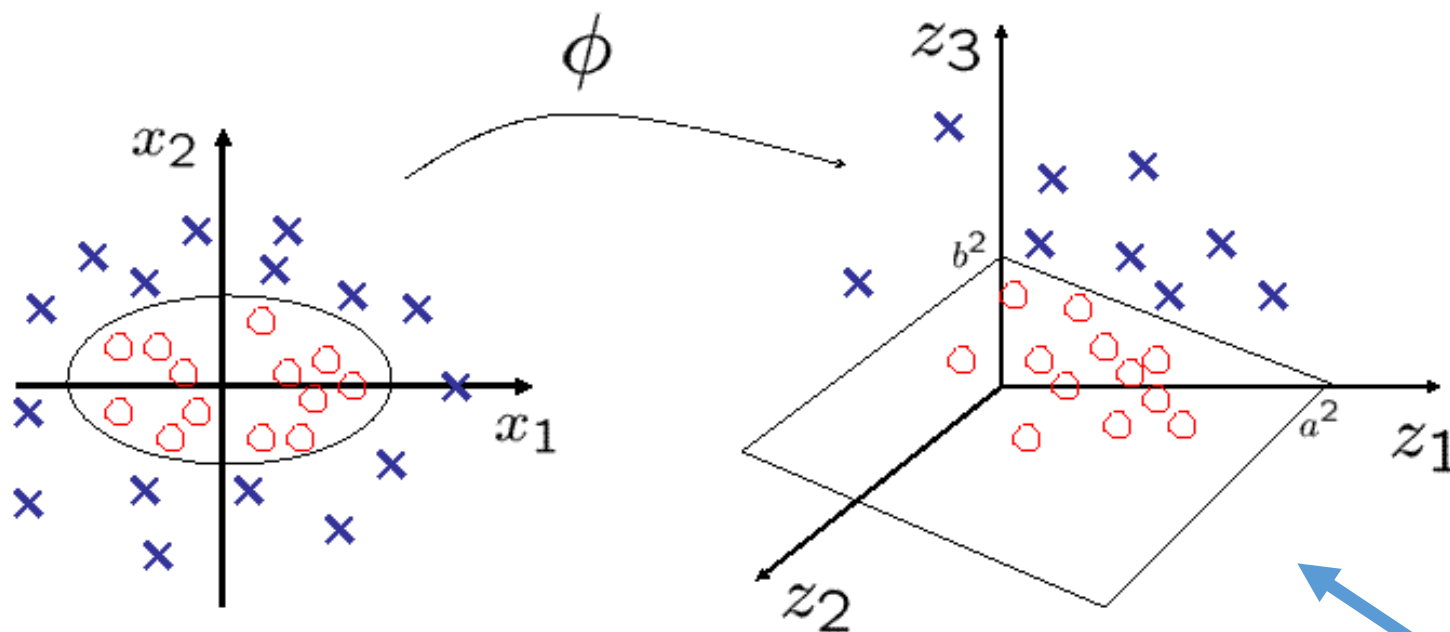
# Добавление новых признаков



$$\phi : (x_1, x_2) \longrightarrow (x_1^2, \sqrt{2}x_1x_2, x_2^2)$$

$$\left(\frac{x_1}{a}\right)^2 + \left(\frac{x_2}{b}\right)^2 = 1 \longrightarrow \frac{z_1}{a^2} + \frac{z_3}{b^2} = 1$$

# Добавление новых признаков



$$\phi : (x_1, x_2) \longrightarrow (x_1^2, \sqrt{2}x_1x_2, x_2^2)$$

$$\left(\frac{x_1}{a}\right)^2 + \left(\frac{x_2}{b}\right)^2 = 1 \longrightarrow \frac{z_1}{a^2} + \frac{z_3}{b^2} = 1$$

Спрямяющее  
пространство

# Kernel Trick

$$\begin{array}{l} x \mapsto \phi(x) \\ w \mapsto \phi(w) \end{array} \implies \langle w, x \rangle \mapsto \langle \phi(w), \phi(x) \rangle$$

# Kernel Trick

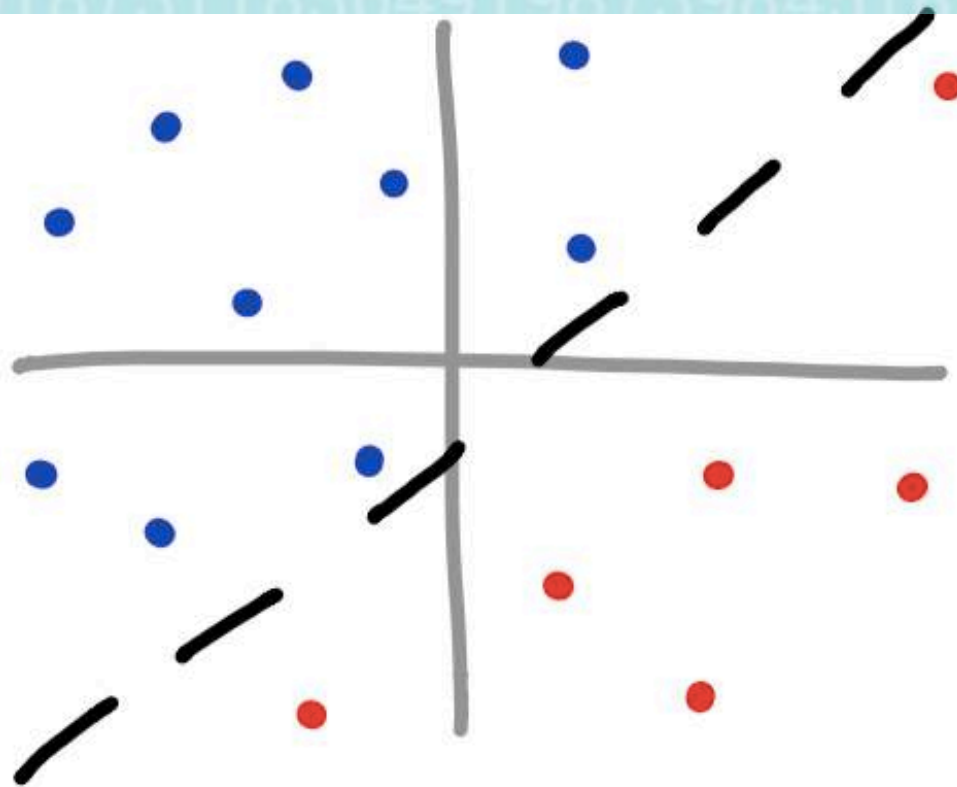
$$\begin{array}{l} x \mapsto \phi(x) \\ w \mapsto \phi(w) \end{array} \Rightarrow \langle w, x \rangle \mapsto \langle \phi(w), \phi(x) \rangle$$

Можно не делать преобразование признаков явно, а вместо скалярного произведения  $\langle w, x \rangle$  использовать функцию  $K(w, x)$ , представимую в виде:

$$K(w, x) = \langle \phi(w), \phi(x) \rangle$$

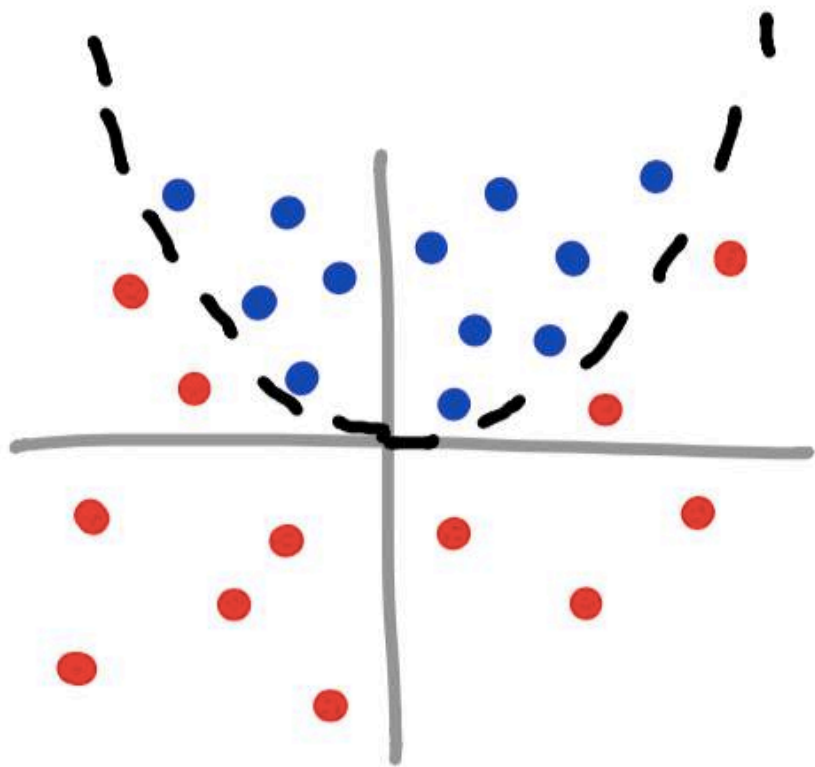


# Линейное ядро



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

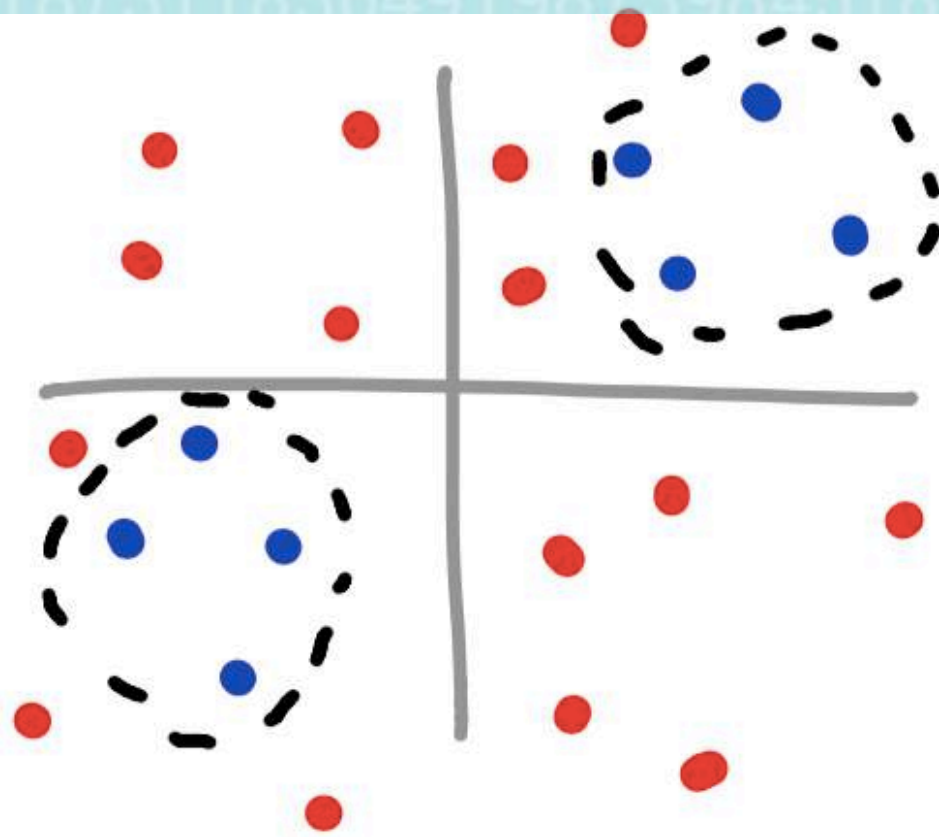
# Полиномиальное ядро



$$K(w, x) = (\gamma < w, x > + r)^d$$

<https://youtu.be/3liCbRZPrZA>

# Радиальное ядро



$$K(w, x) = e^{-\gamma \|w - x\|^2}$$

## V. Линейные модели в 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

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

Если добавить  $x_{i0} = 1$ :

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

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

...

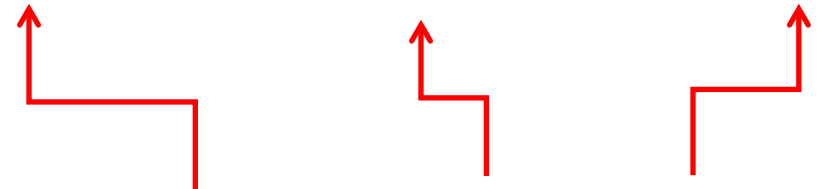
$$y_i \approx \hat{y}_i = x_i^T w$$

...

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

# 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

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

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

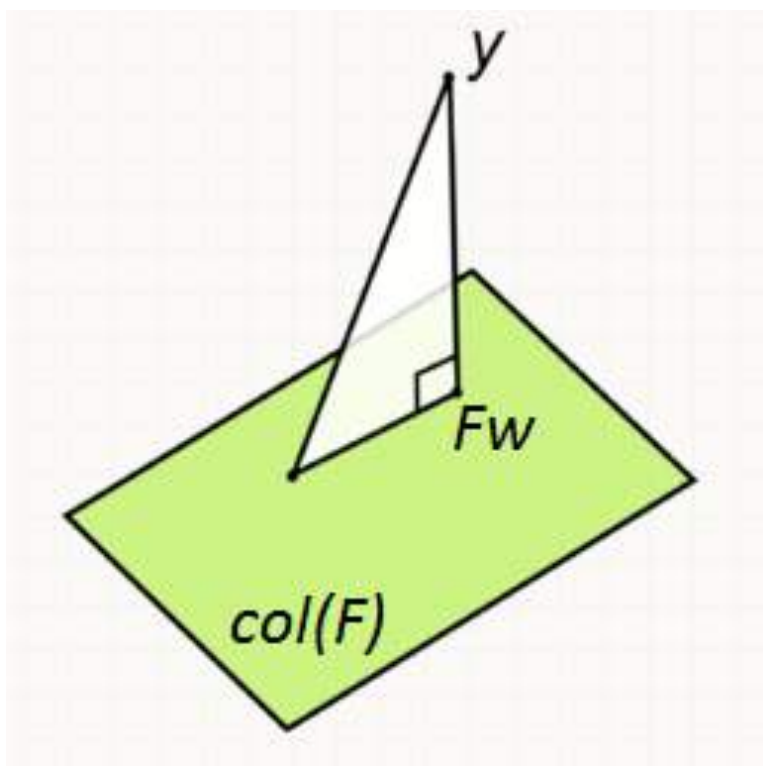
$$w = (F^T F)^{-1} F^T y$$

# Геометрическая интерпретация

$$\begin{pmatrix} y_1 \\ y_2 \\ \dots \\ y_l \end{pmatrix} \approx \begin{pmatrix} \widehat{y}_1 \\ \widehat{y}_2 \\ \dots \\ \widehat{y}_l \end{pmatrix} = (F_{(1)} \quad \dots \quad F_{(m)})w$$

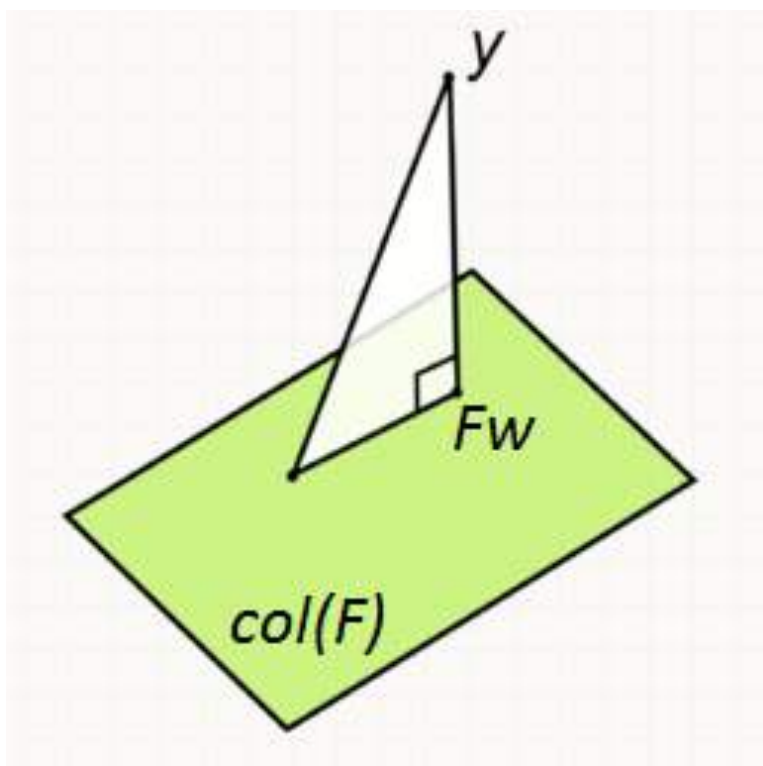
$$y \approx \hat{y} = Fw = w_1 F_{(1)} + \dots + w_m F_{(m)}$$

# Геометрическая интерпретация



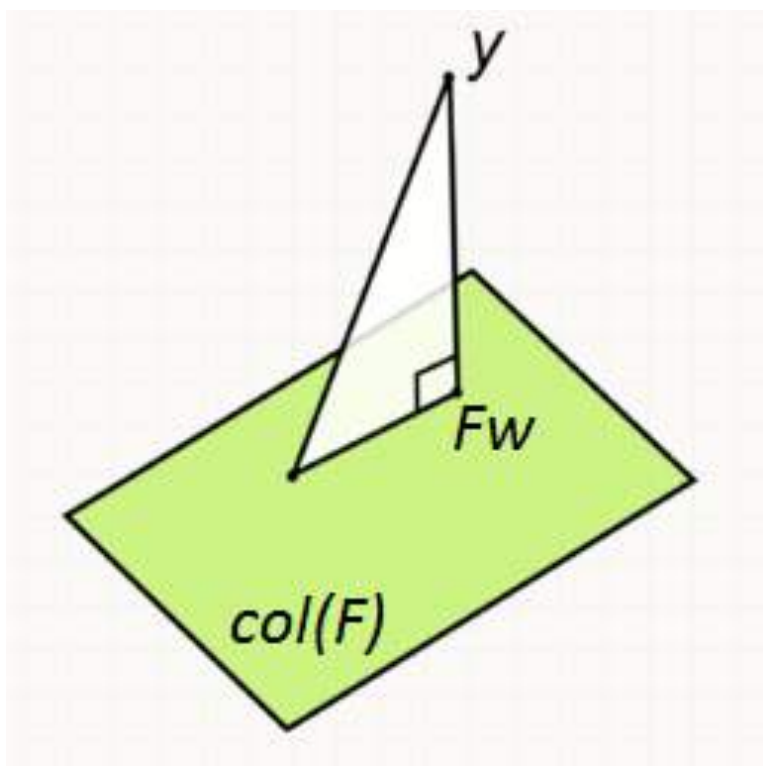
# Геометрическая интерпретация

$$(Fw - y) \perp F_{(k)} \quad \forall k = 1, \dots, m$$



# Геометрическая интерпретация

$$(Fw - y) \perp F_{(k)} \quad \forall k = 1, \dots, m$$

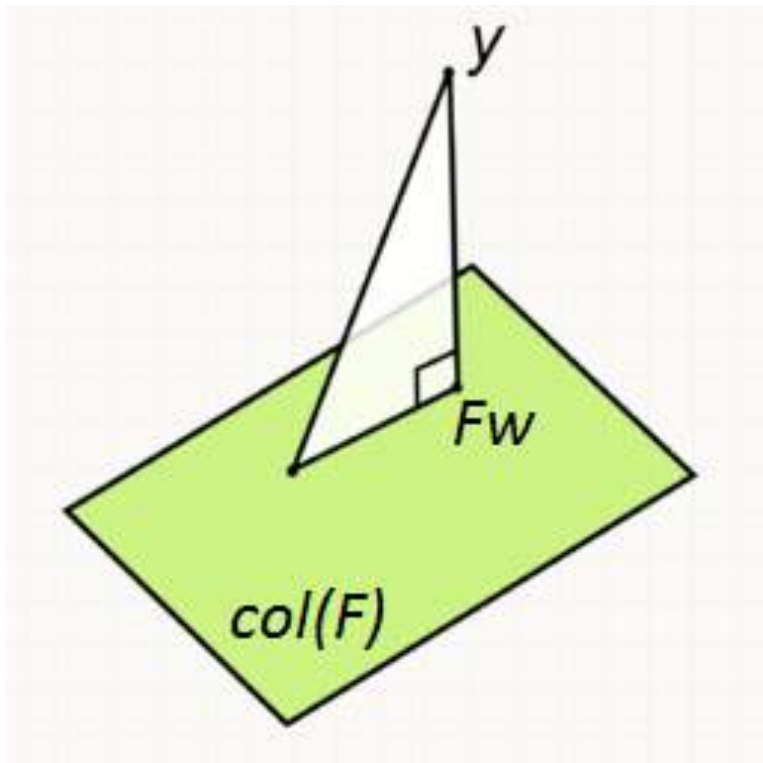


$$F_{(k)}^T (Fw - y) = 0 \quad \forall k$$



# Геометрическая интерпретация

$$(Fw - y) \perp F_{(k)} \quad \forall k = 1, \dots, m$$

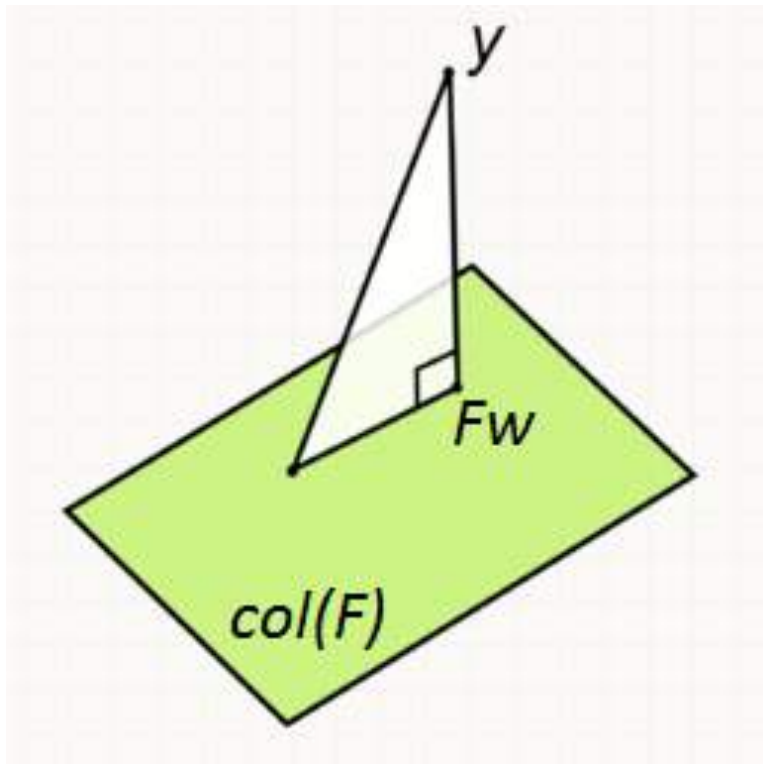


$$F_{(k)}^T (Fw - y) = 0 \quad \forall k$$

$$F^T (Fw - y) = 0$$

# Геометрическая интерпретация

$$(Fw - y) \perp F_{(k)} \quad \forall k = 1, \dots, m$$



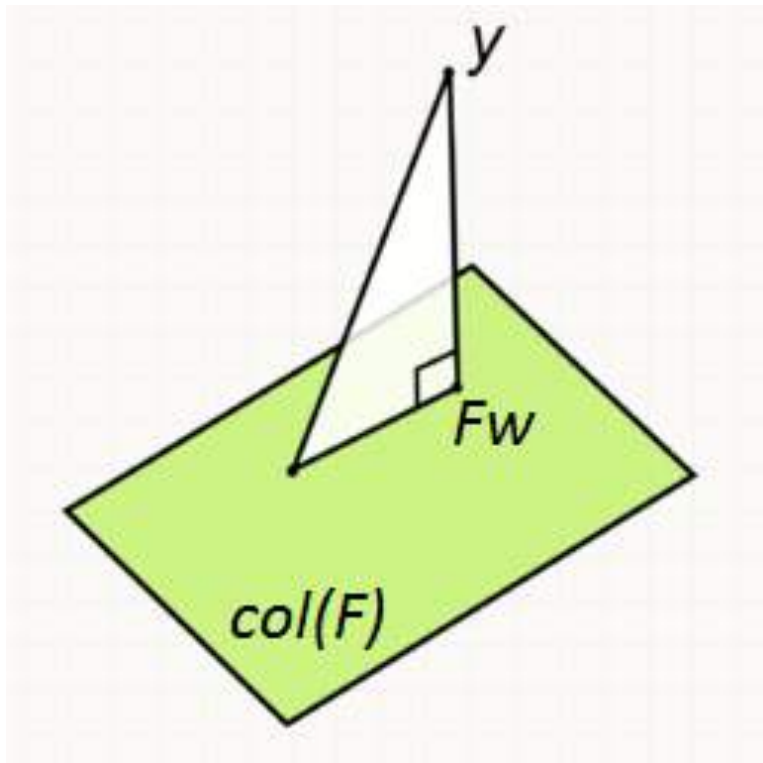
$$F_{(k)}^T (Fw - y) = 0 \quad \forall k$$

$$F^T (Fw - y) = 0$$

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

# Геометрическая интерпретация

$$(Fw - y) \perp F_{(k)} \quad \forall k = 1, \dots, m$$



$$F_{(k)}^T (Fw - y) = 0 \quad \forall k$$

$$F^T (Fw - y) = 0$$

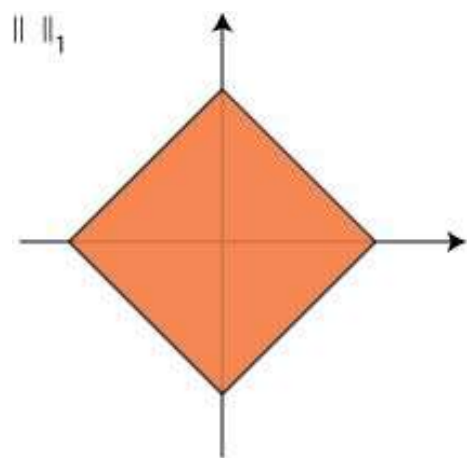
$$F^T Fw = F^T y$$

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

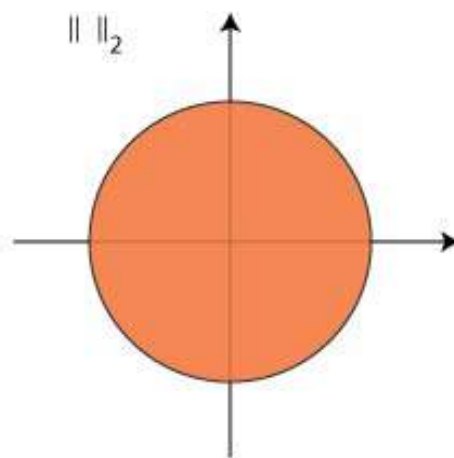
# LASSO и гребневая регрессия

$l_1$  — регуляризация

$l_2$  — регуляризация



LASSO



Ridge

# Итог

1. Линейная классификация и оптимизационная задача в ней
2. Как настраиваются коэффициенты: SGD
3. Как бороться с переобучением: регуляризация
4. Стандартные линейные классификаторы
5. О линейных моделях в регрессии



# 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)

# Упражнение

В реализации линейного классификатора на Python, приведенной в этих слайдах, намеренно допущены небольшие ошибки

Попробуйте их найти (обсудим в начале следующей лекции)

**Подсказка:**

подумайте о том, как мы обозначаем классы, а также об отличии правила обновления  $w_0$  и  $w$  в SGD

# Конкурс

Найти ошибку на этой картинке и написать комментарий в соответствующей записи в группе DMIA в ВК

Победитель (первый, кто ответит верно) получит приз

## Machine Learning



what society thinks I do



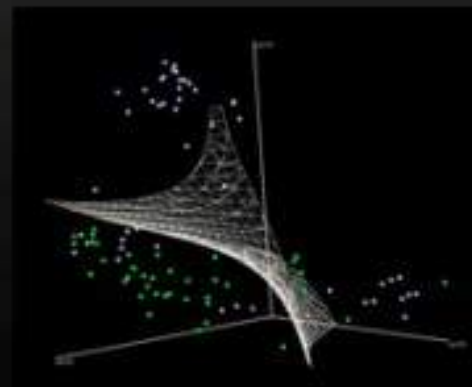
what my friends think I do



what my parents think I do

$$\begin{aligned} L_p &= ||\mathbf{w}'||^2 - \sum_i \alpha_i y_i (\mathbf{x}_i \cdot \mathbf{w} + b) + \sum_i \alpha_i \\ \alpha_i &\geq 0, \forall i \\ \mathbf{w} &= \sum_i \alpha_i y_i \mathbf{x}_i, \sum_i \alpha_i y_i = 0 \\ \nabla \hat{g}(\theta_t) &= \frac{1}{n} \sum_{i=1}^n \nabla \ell(x_i, y_i; \theta_t) + \nabla r(\theta_t) \\ \theta_{t+1} &= \theta_t - \eta_t \nabla \ell(x_{(t)}, y_{(t)}; \theta_t) - \eta_t \cdot \nabla r(\theta_t) \\ \mathbb{E}_{(t)}[\ell(x_{(t)}, y_{(t)}; \theta_t)] &= \frac{1}{n} \sum_i \ell(x_i, y_i; \theta_t) \end{aligned}$$

what other programmers think I do



what I think I do

```
>>> from sklearn import svm
```

what I really do

# Конкурс

В самой первой версии мема было:

```
from scipy import SVM
```

В опубликованном:

```
from sklearn import svm
```

Мы этого не заметили, а когда заметили, решили, что так не получится

Скрыть комментарии



Дмитрий Волынкин

```
import xgboost
```

10 фев в 16:19 Ответить



Александр Погачёв

```
»» from sklearn.svm import SVC
```

10 фев в 17:40 Ответить



1



Александр Ничипоренко

Про "from sklearn import svm" - это корректно, из документации:



10 фев в 17:45 Ответить



3



Светлана Погребняк

На нижней левой картинке плохо все видно, но это не svm. На других нижних действительно он :)

10 фев в 18:24 Ответить



Александр Кузнецов

Кто-нибудь встречал использование svm в prod? По ощущениям, его изучение это дань уважения поколению ученых, посвятивших себя его исследованиям.



# Конкурс

В опубликованном:

```
from sklearn import svm
```

Мы этого не заметили, а когда заметили, решили, что так не получится

Но так можно: мы же можем импортировать весь модуль `sklearn.svm`

Скрыть комментарии



Дмитрий Волынкин

```
import xgboost
```

10 фев в 16:19 Ответить



Александр Рогачёв

```
>> from sklearn.svm import SVC
```

10 фев в 17:40 Ответить



1



Александр Ничипоренко

Про "from sklearn import svm" - это корректно, из документации:



10 фев в 17:45 Ответить



3



Светлана Погребняк

На нижней левой картинке плохо все видно, но это не svm. На других нижних действительно он :)

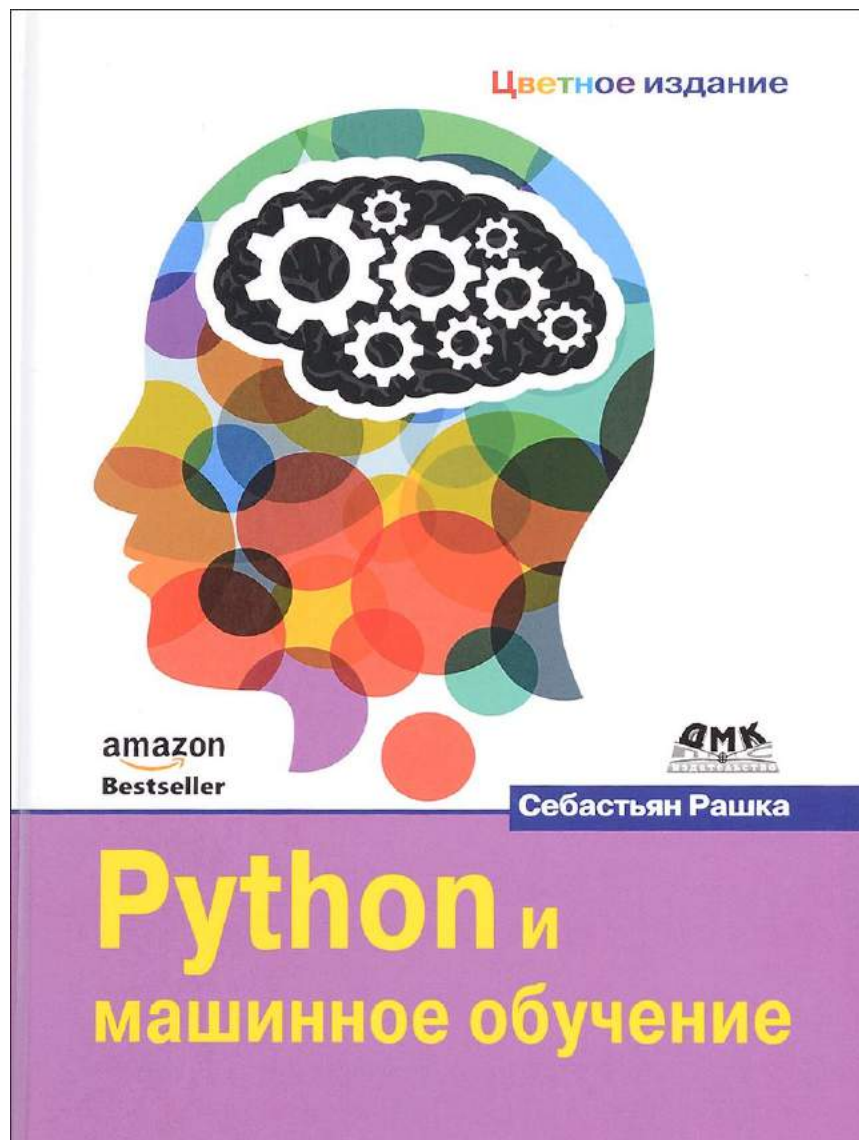
10 фев в 18:24 Ответить



Александр Кузнецов

Кто-нибудь встречал использование svm в prod? По ощущениям, его изучение это дань уважения поколению ученых, посвятивших себя его исследованиям.

# Приз



Скрыть комментарии



Дмитрий Волынкин

`import xgboost`

10 фев в 16:19 Ответить



Александр Рогачёв

`>> from sklearn.svm import SVC`

10 фев в 17:40 Ответить



1



Александр Ничипоренко

Про "from sklearn import svm" - это корректно, из документации:



10 фев в 17:45 Ответить



3



Светлана Погребняк

На нижней левой картинке плохо все видно, но это не svm. На других нижних действительно он :)

10 фев в 18:24 Ответить



Александр Кузнецов

Кто-нибудь встречал использование svm в prod? По ощущениям, его изучение это дань уважения поколению ученых, посвятивших себя его исследованиям.

# Контакты



[dmia@applieddatascience.ru](mailto:dmia@applieddatascience.ru)



<https://t.me/joinchat/B1OITk74nRV56Dp1TDJGNA>



<https://goo.gl/forms/1k17ALSW2urgM91m2>