

# Машинное обучение

Лекция 6. Линейные модели, метод стохастического градиента

МФТИ 2018

Материалы: В. Кантор, К. Воронцов

# План

1. Знакомство с линейными моделями
2. Метод Стохастического градиента
3. Регуляризация

# 1. Знакомство с линейными моделями

Пример: выходить из дома или нет

Признаки (1/0):

1. Вы свободны в данный момент
2. Вам хочется где-то поесть
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 +1
2. Вам хочется где-то поесть
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 +1
2. Вам хочется где-то поесть +2
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 **+1**
2. Вам хочется где-то поесть **+2**
3. Вам хочется спать **-3**
4. Вам хочется увидеться с друзьями

Пример: выходить из дома или нет

Признаки (1/0):

1. Вы свободны в данный момент +1
2. Вам хочется где-то поесть +2
3. Вам хочется спать -3
4. Вам хочется увидеться с друзьями +4

# Пример: выходить из дома или нет

Признаки (1/0):

1. Вы свободны в данный момент **+1**
2. Вам хочется где-то поесть **+2**
3. Вам хочется спать **-3**
4. Вам хочется увидеться с друзьями **+4**

**Порог для решающего правила: +1**

**Если сумма больше – выходим :)**

Более серьезный пример:  
дать кредит или нет

Признаки (1/0):

- Работоспособный возраст
- Имеет счет в вашем банке
- Много просрочек по платежам за другие кредиты
- Нет просрочек по платежам за другие кредиты, причем другие 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

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

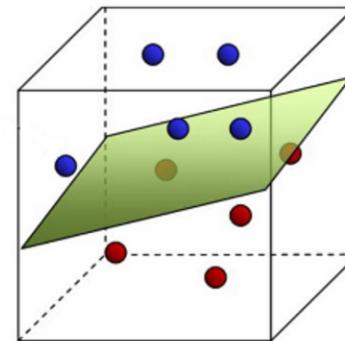
$$f(x) = w_0 + w_1 x_1 + \cdots + w_n x_n$$

Формализуем линейный классификатор

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

$$f(x) = w_0 + w_1x_1 + \cdots + w_nx_n = w_0 + \langle w, x \rangle$$

Геометрическая интерпретация:  
разделяем классы плоскостью

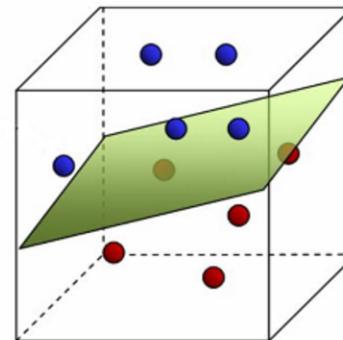


Формализуем линейный классификатор

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

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

Геометрическая интерпретация:  
разделяем классы плоскостью



## 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  $x_i$ )

$$\begin{aligned}M_i \leq 0 &\Leftrightarrow y_i \neq a(x_i) \\M_i > 0 &\Leftrightarrow y_i = a(x_i)\end{aligned}$$

## Функция потерь

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

## Функция потерь

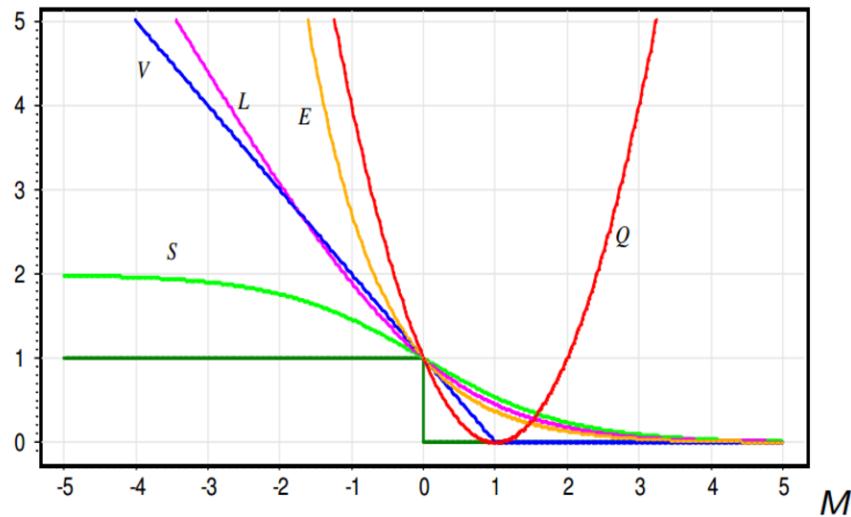
$$Q(w) = \sum_{i=1}^{\ell} [M_i(w) < 0] \leq \tilde{Q}(w) = \sum_{i=1}^{\ell} \mathcal{L}(M_i(w)) \rightarrow \min_w;$$

Функция эмпирического риска

Функция потерь

# ФУНКЦИЯ ПОТЕРЬ

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

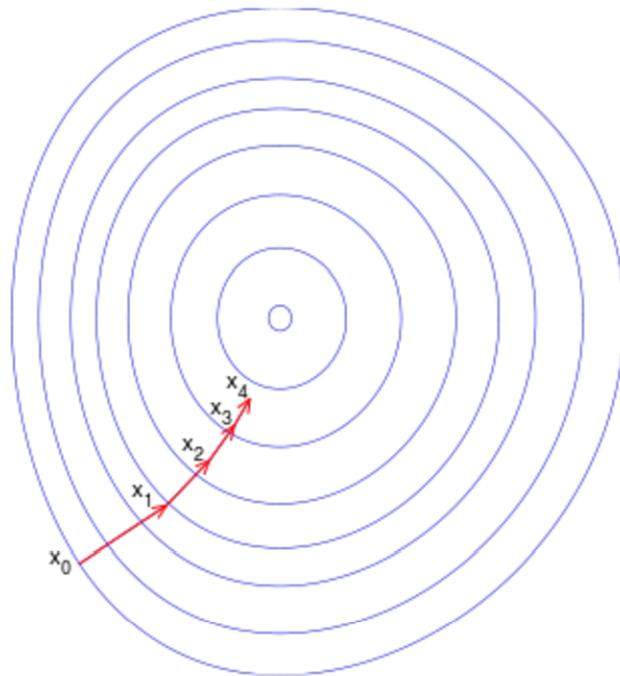


$$\begin{aligned} Q(M) &= (1 - M)^2 \\ V(M) &= (1 - M)_+ \\ S(M) &= 2(1 + e^M)^{-1} \\ L(M) &= \log_2(1 + e^{-M}) \\ E(M) &= e^{-M} \end{aligned}$$

## 2. Метод Стохастического градиента

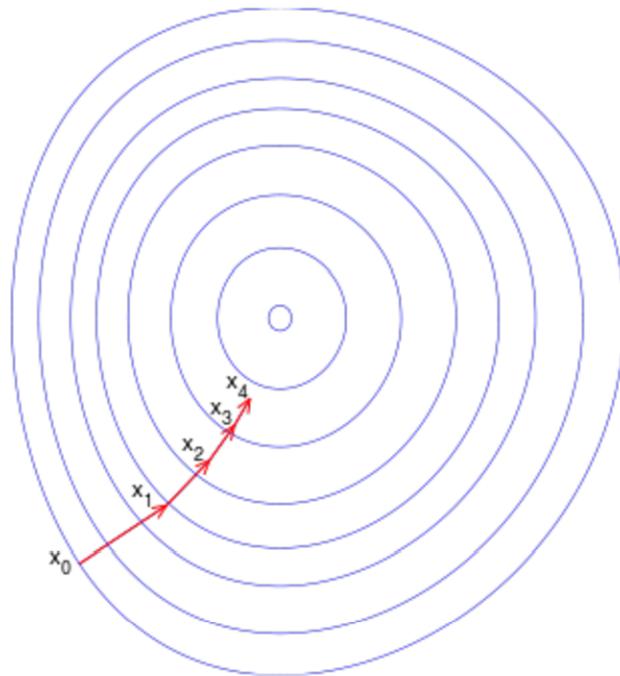
# Градиентный спуск

$$\mathbf{x}_{n+1} = \mathbf{x}_n - \gamma_n \nabla F(\mathbf{x}_n), \quad n \geq 0.$$



# Градиентный спуск

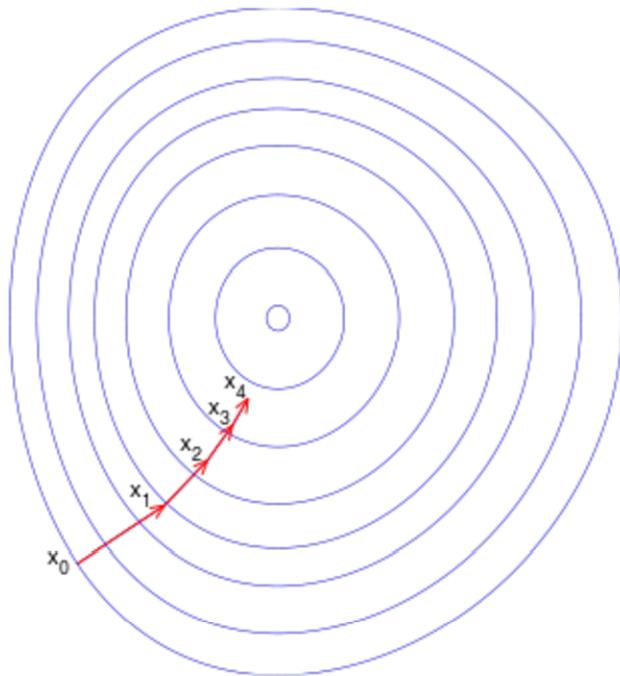
$$\mathbf{x}_{n+1} = \mathbf{x}_n - \gamma_n \nabla F(\mathbf{x}_n), \quad n \geq 0.$$



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

# Градиентный спуск

$$\mathbf{x}_{n+1} = \mathbf{x}_n - \gamma_n \nabla F(\mathbf{x}_n), \quad n \geq 0.$$

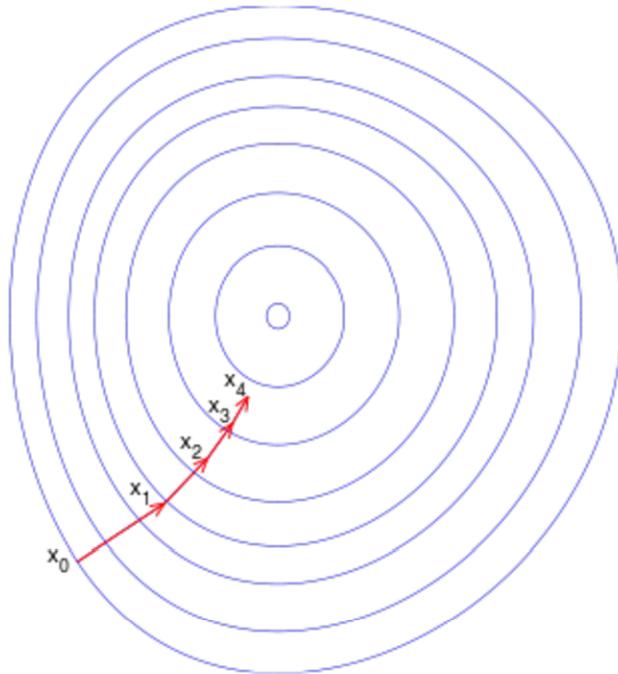


$$\nabla_w \tilde{Q} = \sum_{i=1}^l \nabla L(M_i)$$

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

# Градиентный спуск

$$\mathbf{x}_{n+1} = \mathbf{x}_n - \gamma_n \nabla F(\mathbf{x}_n), \quad n \geq 0.$$



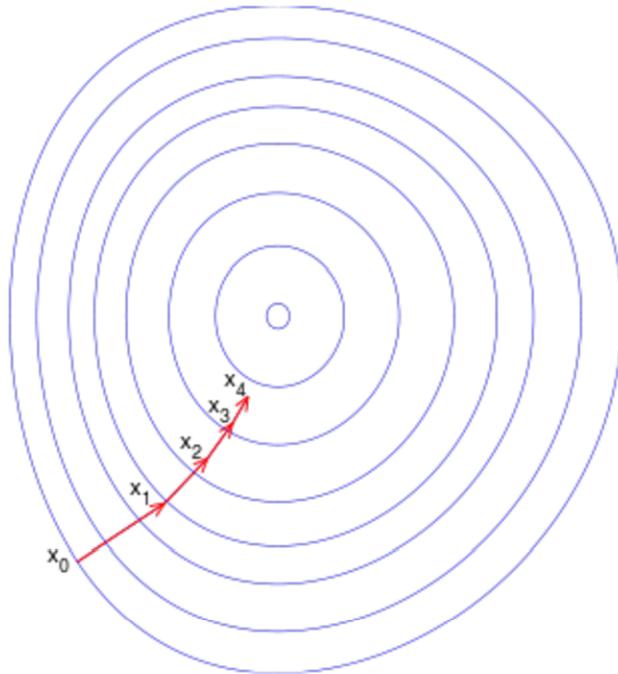
$$\nabla_w \tilde{Q} = \sum_{i=1}^l \nabla L(M_i)$$

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

$$\frac{\partial M_i}{\partial w} = y_i x_i$$

# Градиентный спуск

$$\mathbf{x}_{n+1} = \mathbf{x}_n - \gamma_n \nabla F(\mathbf{x}_n), \quad n \geq 0.$$



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

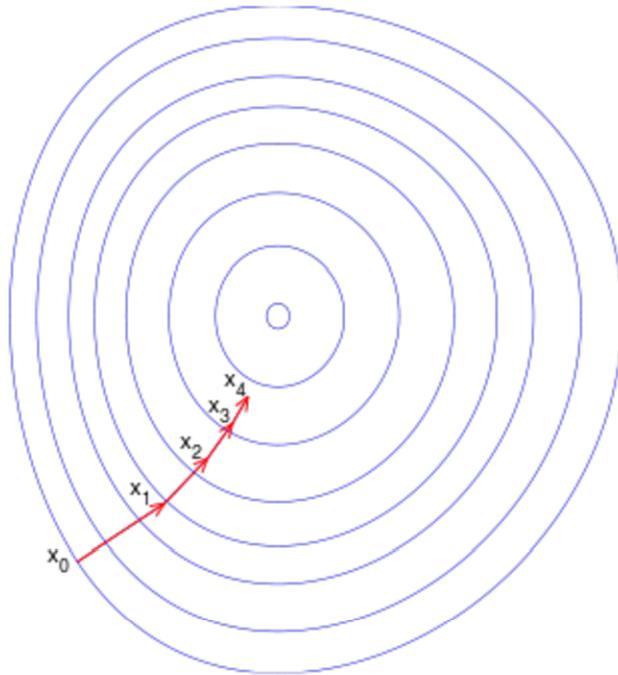
$$\nabla \tilde{Q} = \sum_{i=1}^l L'(M_i) \frac{\partial M_i}{\partial w}$$

$$\frac{\partial M_i}{\partial w} = y_i x_i$$

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

# Градиентный спуск

$$\mathbf{x}_{n+1} = \mathbf{x}_n - \gamma_n \nabla F(\mathbf{x}_n), \quad n \geq 0.$$



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

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

$$\frac{\partial M_i}{\partial w} = y_i x_i$$

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

$$w_{n+1} = w_n - \gamma_n \sum_{i=1}^l y_i x_i L'(M_i)$$

# Стохастический градиент

$$w_{n+1} = w_n - \gamma_n \sum_{i=1}^l y_i x_i L'(M_i)$$

$$w_{n+1} = w_n - \gamma_n y_i x_i L'(M_i)$$

$x_i$  — 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  $\gamma_n = \frac{1}{\alpha + n}$ 

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)
```

$$\gamma_n = \frac{1}{\alpha + n}$$
$$\gamma_n = \frac{1}{\sqrt{\alpha + n}}$$

# 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)
```

$$\gamma_n = \frac{1}{\alpha + n}$$

$$\gamma_n = \frac{1}{\sqrt{\alpha + n}}$$

$$\gamma_n = (\alpha + n)^{-\beta}$$

# 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)
```

$$\gamma_n = \frac{1}{\alpha + n}$$
$$\gamma_n = \frac{1}{\sqrt{\alpha + n}}$$
$$\gamma_n = (\alpha + n)^{-\beta}$$
$$\gamma_n = \tau \beta_n$$

# 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
         $w_{n+1} = w_n - \gamma_n y_i x_i L'(M_i)$ 
        w -= x * step * y * der_loss(x, y)
```

# SGD: нюансы реализации

Инициализация весов

Правило останова

Выбор темпа обучения

# SGD: нюансы реализации

Инициализация весов

$$1) \quad w = \vec{0}$$

$$2) \quad w = \text{random} \left( -\frac{1}{\ell}, \frac{1}{\ell} \right)$$

Правило останова

Выбор темпа обучения

# SGD: нюансы реализации

Инициализация весов

$$1) \quad w = \vec{0}$$

$$2) \quad w = \text{random} \left( -\frac{1}{\ell}, \frac{1}{\ell} \right)$$

Правило останова

1) Пока веса не сойдутся

и

1) Пока сглаженная ошибка  $\bar{Q}$  не сойдётся

Выбор темпа обучения

# SGD: нюансы реализации

Инициализация весов

1)  $w = \vec{0}$

или

1)  $w = \text{random}\left(-\frac{1}{\ell}, \frac{1}{\ell}\right)$

Правило останова:

1) Пока веса не сойдутся

и

2) Пока сглаженная ошибка  $\bar{Q}$  не сойдётся

Выбор темпа обучения

1) Подбираем оптимальный шаг

$$\mathcal{L}_i(w - h \nabla \mathcal{L}_i(w)) \rightarrow \min_h,$$

2) Методы второго порядка

# Стохастический градиент: алгоритм целиком

**Вход:** выборка  $X^\ell$ , темп обучения  $h$ , темп забывания  $\lambda$ ;

**Выход:** вектор весов  $w$ ;

1 инициализировать веса  $w_j$ ,  $j = 0, \dots, n$ ;

2 инициализировать оценку функционала:

$$\bar{Q} := \frac{1}{\ell} \sum_{i=1}^{\ell} \mathcal{L}_i(w);$$

3 **повторять**

4   | выбрать объект  $x_i$  из  $X^\ell$  случайным образом;

5   | вычислить потерю:  $\varepsilon_i := \mathcal{L}_i(w)$ ;

6   | сделать градиентный шаг:  $w := w - h \nabla \mathcal{L}_i(w)$ ;

7   | оценить функционал:  $\bar{Q} := (1 - \lambda) \bar{Q} + \lambda \varepsilon_i$ ;

8 **пока** значение  $\bar{Q}$  и/или веса  $w$  не сойдутся;

---

*Robbins, H., Monro S. A stochastic approximation method // Annals of Mathematical Statistics, 1951, 22 (3), p. 400–407.*

# Стохастический градиент: SAG

**Вход:** выборка  $X^\ell$ , темп обучения  $h$ , темп забывания  $\lambda$ ;

**Выход:** вектор весов  $w$ ;

- 1 инициализировать веса  $w_j$ ,  $j = 0, \dots, n$ ;
- 2 инициализировать градиенты:  $G_i := \nabla \mathcal{L}_i(w)$ ,  $i = 1, \dots, \ell$ ;
- 3 инициализировать оценку функционала:

$$\bar{Q} := \frac{1}{\ell} \sum_{i=1}^{\ell} \mathcal{L}_i(w);$$

## 4 ПОВТОРЯТЬ

- 5    выбрать объект  $x_i$  из  $X^\ell$  случайным образом;
- 6    вычислить потерю:  $\varepsilon_i := \mathcal{L}_i(w)$ ;
- 7    вычислить градиент:  $G_i := \nabla \mathcal{L}_i(w)$ ;
- 8    сделать градиентный шаг:  $w := w - h \sum_{i=1}^{\ell} G_i$ ;
- 9    оценить функционал:  $\bar{Q} := (1 - \lambda) \bar{Q} + \lambda \ell \varepsilon_i$ ;
- 10 **ПОКА** значение  $\bar{Q}$  и/или веса  $w$  не сойдутся;

# SGD: правило останова

**Проблема:** после каждого шага  $w$  по одному объекту  $x_i$ ,  
не хотелось бы оценивать  $Q$  по всей выборке  $x_1, \dots, x_\ell$ .

**Решение:** использовать рекуррентную формулу.

Среднее арифметическое  $\bar{Q}_m = \frac{1}{m} \sum_{i=1}^m \varepsilon_i$ :

$$\bar{Q}_m = \left(1 - \frac{1}{m}\right) \bar{Q}_{m-1} + \frac{1}{m} \varepsilon_m.$$

*Экспоненциальное скользящее среднее*

$$\bar{Q}_m = \lambda \varepsilon_m + \lambda(1 - \lambda) \varepsilon_{m-1} + \lambda(1 - \lambda)^2 \varepsilon_{m-2} + \lambda(1 - \lambda)^3 \varepsilon_{m-3} + \dots$$

$$\bar{Q}_m := (1 - \lambda) \bar{Q}_{m-1} + \lambda \varepsilon_m.$$

Чем больше  $\lambda$ , тем быстрее забывается предыстория ряда.  
Параметр  $\lambda$  называется *темпом забывания*.

# SGD: выбор темпа обучения $h_t$

- ➊ сходимость гарантируется (для выпуклых функций) при

$$h_t \rightarrow 0, \quad \sum_{t=1}^{\infty} h_t = \infty, \quad \sum_{t=1}^{\infty} h_t^2 < \infty,$$

в частности можно положить  $h_t = 1/t$ ;

- ➋ метод скорейшего градиентного спуска:

$$\mathcal{L}_i(w - h \nabla \mathcal{L}_i(w)) \rightarrow \min_h$$

позволяет найти адаптивный шаг  $h^*$ ;

**Упражнение:** доказать, что при квадратичной функции потерь  $h^* = \|x_i\|^{-2}$ .

- ➌ пробные случайные шаги

— для «выбивания» из локальных минимумов;

- ➍ метод Левенберга-Марквардта (второго порядка)

# SGD: метод Левенберга-Марквардта

Метод Ньютона-Рафсона,  $\mathcal{L}_i(w) \equiv \mathcal{L}(\langle w, x_i \rangle y_i)$ :

$$w := w - h(\mathcal{L}_i''(w))^{-1} \nabla \mathcal{L}_i(w),$$

где  $\mathcal{L}_i''(w) = \left( \frac{\partial^2 \mathcal{L}_i(w)}{\partial w_j \partial w_{j'}} \right)$  — гессиан,  $n \times n$ -матрица

**Эвристика.** Считаем, что гессиан диагонален:

$$w_j := w_j - h \left( \frac{\partial^2 \mathcal{L}_i(w)}{\partial w_j^2} + \mu \right)^{-1} \frac{\partial \mathcal{L}_i(w)}{\partial w_j},$$

$h$  — темп обучения, можно полагать  $h = 1$

$\mu$  — параметр, предотвращающий обнуление знаменателя.

Отношение  $h/\mu$  есть темп обучения на ровных участках функционала  $\mathcal{L}_i(w)$ , где вторая производная обнуляется.

# Метод стохастического градиента: резюме

Достоинства:

- легко реализуется;
- легко обобщается на любые функции потерь;
- возможно динамическое (потоковое) обучение;
- на сверхбольших выборках можно получить неплохое решение, даже не обработав все ( $x_i, y_i$ );
- всё чаще применяется для Big Data

Недостатки:

- возможна расходимость или медленная сходимость;
- застревание в локальных минимумах;
- подбор комплекса эвристик является искусством;
- проблема переобучения

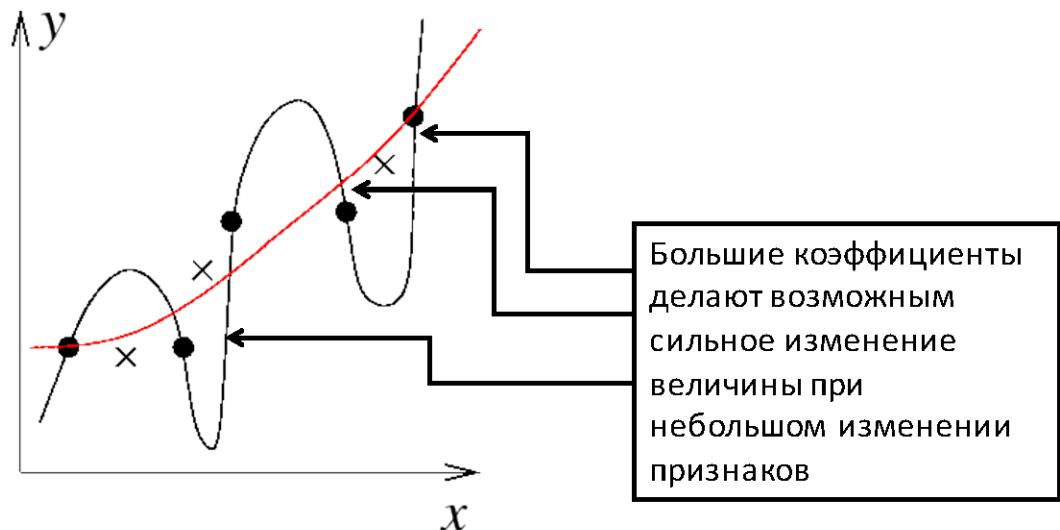
# Метод стохастического градиента: резюме

- Возможные причины переобучения:
  - слишком мало объектов; слишком много признаков;
  - линейная зависимость (мультиколлинеарность) признаков
- Симптоматика:
  - слишком большие веса  $|w_j|$  разных знаков;
  - неустойчивость  $Q(x, w)$ ;
  - высокая ошибка на тестовой выборке;
- Терапия:
  - регуляризация (сокращение весов, weight decay);
  - ранний останов (early stopping);

### 3. Регуляризация

# Регуляризация

- Переобучение в задаче обучения с учителем как правило означает большие коэффициенты:



- Идея: добавить ограничение на коэффициенты

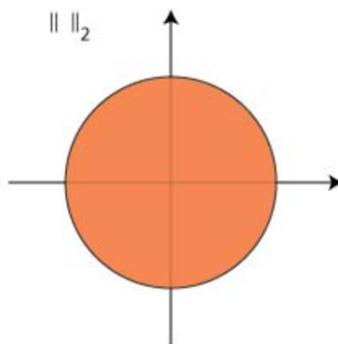
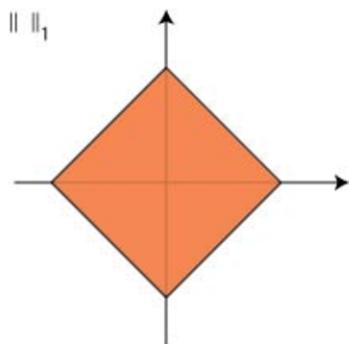
# Регуляризация

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

*l1 – регуляризация*

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

*l2 – регуляризация*

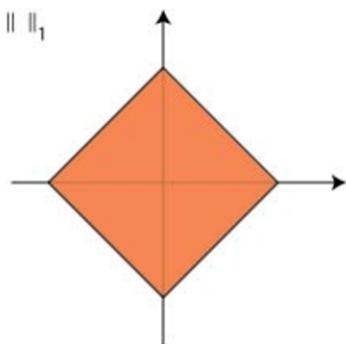


# Регуляризация

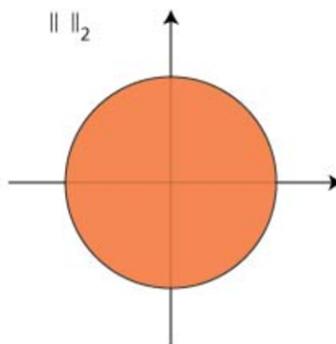
$$\sum_{i=1}^l L(M_i) + \gamma \sum_{k=1}^m |w_k| \rightarrow \min$$

$$\sum_{i=1}^l L(M_i) + \gamma \sum_{k=1}^m {w_k}^2 \rightarrow \min$$

*l1 – регуляризация*



*l2 – регуляризация*

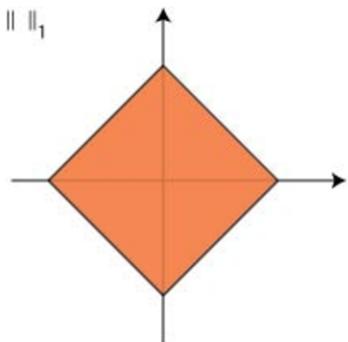


# Регуляризация

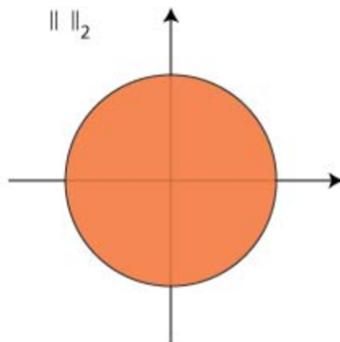
$$\sum_{i=1}^l L(M_i) + \gamma \sum_{k=1}^m |w_k| \rightarrow \min$$

$$\sum_{i=1}^l L(M_i) + \gamma \sum_{k=1}^m {w_k}^2 \rightarrow \min$$

*l1 – регуляризация*



*l2 – регуляризация*



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

## Общий случай

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

Функция потерь

Коэффициент регуляризации

Регуляризатор

## Различия между $\ell_1$ и $\ell_2$

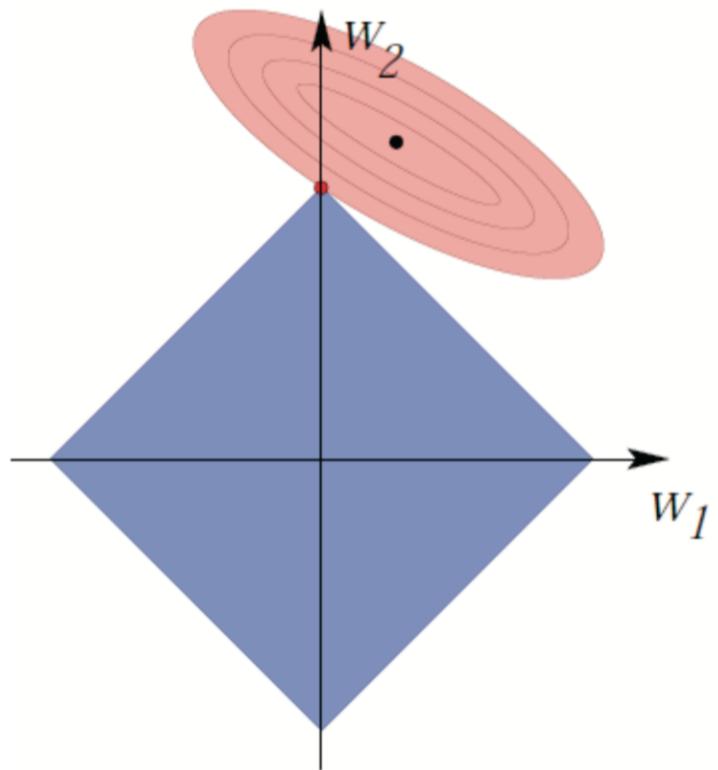
- **Разреженность** –  $\ell_1$ -регуляризация делает вектор весов более разреженным (содержащим больше нулей)
- В случае линейной классификации это означает **отбор признаков**: признаки с нулевыми весами не используются в классификации

Почему  $\ell_1$  разреживает: простое и неубедительно объяснение

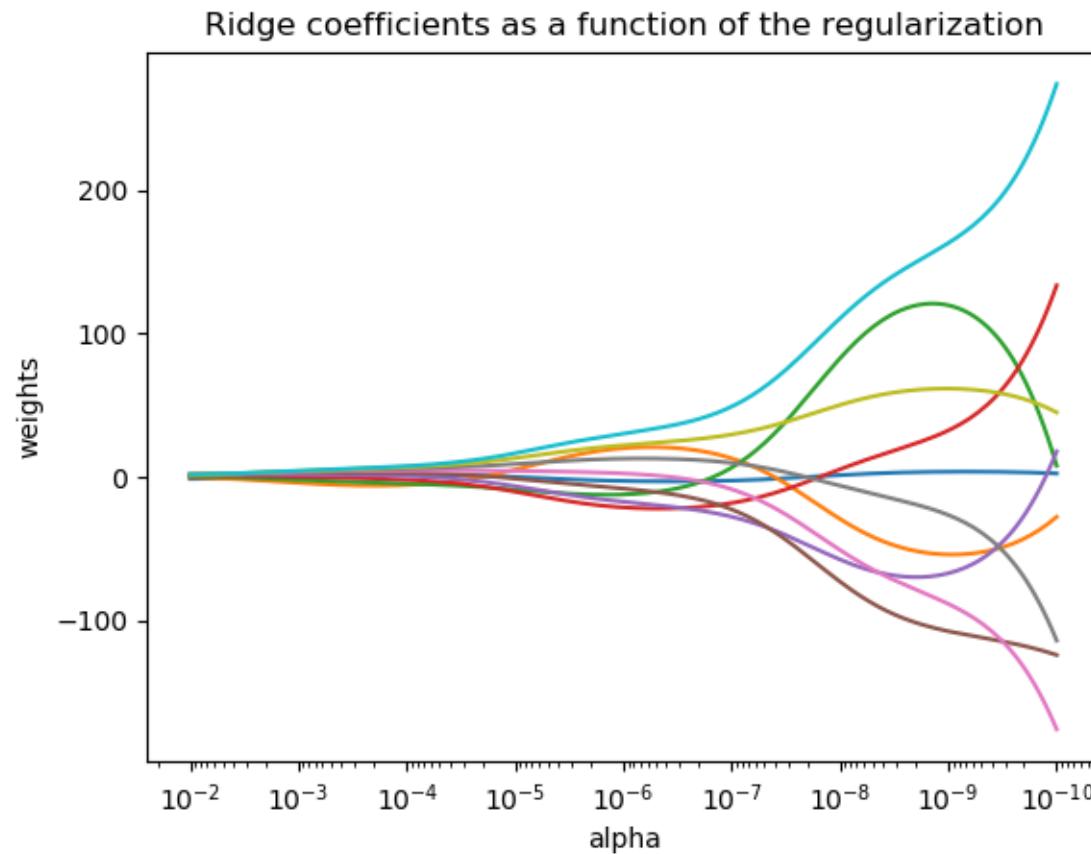
$$\left\{ \begin{array}{l} \tilde{Q} = \sum_{i=1}^l L(M_i) \rightarrow \min \\ \sum_{k=1}^m |w_k| \leq \tau \end{array} \right.$$

Почему  $\ell_1$  разреживает: простое и неубедительно объяснение

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

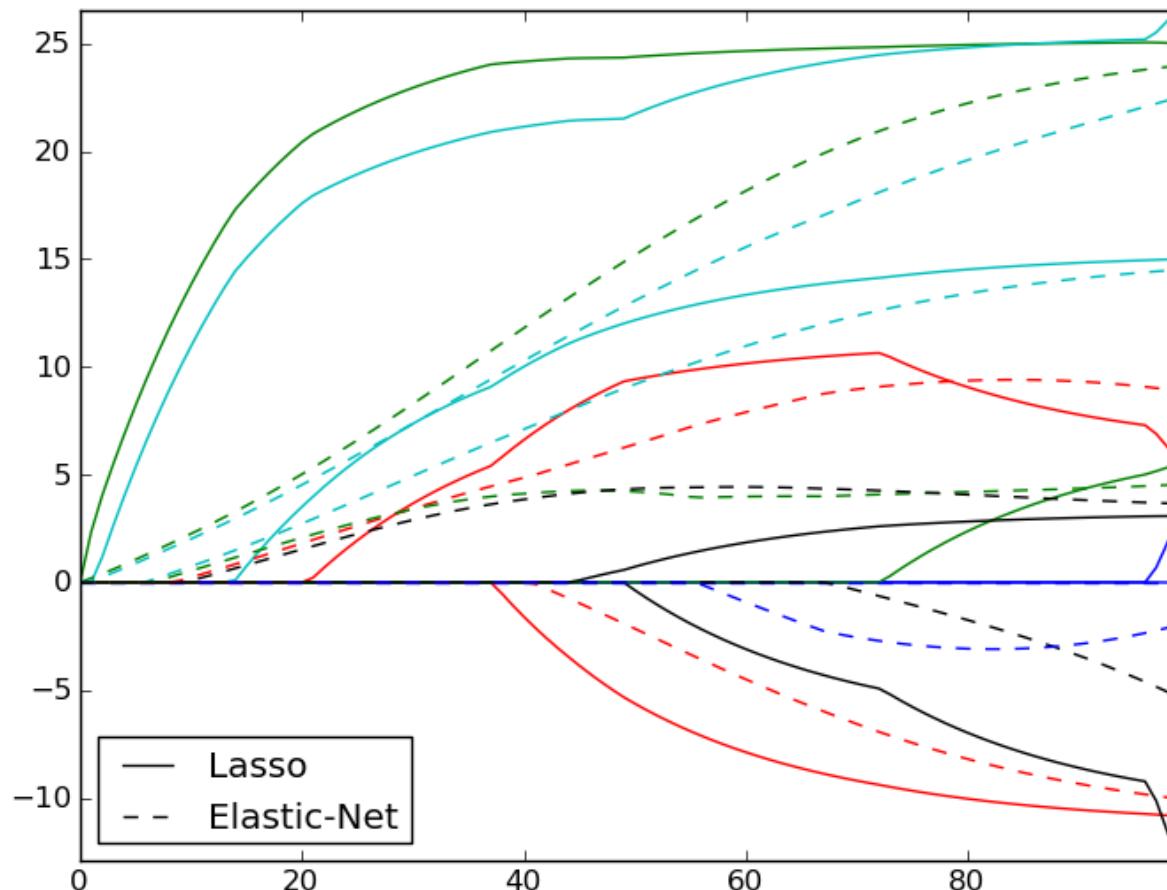


Компоненты веса в зависимости от значения коэффициента регуляризации  $\ell_2$



Lasso:  $\|w\|_2$

## Компоненты веса в зависимости от значения коэффициента регуляризации



$$\text{Lasso: } \|w\|_1$$

$$\text{Elastic-Net: } \alpha \|w\|_1 + (1 - \alpha) \|w\|_2$$

Вероятностный смысл регуляризаторов

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

Вероятностный смысл регуляризаторов

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

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

## Вероятностный смысл регуляризаторов

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

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

$$\sum_{i=1}^{\ell} \ln e^{-L(y_i, f(x_i))} + \ln e^{-\gamma V(w)} \rightarrow \max_w$$

Вероятностный смысл регуляризаторов

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

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

$$\sum_{i=1}^{\ell} \ln e^{-L(y_i, f(x_i))} + \ln e^{-\gamma V(w)} \rightarrow \max_w$$

$$e^{-\gamma V(w)} \prod_{i=1}^{\ell} e^{-L(y_i, f(x_i))} \rightarrow \max_w$$

Вероятностный смысл регуляризаторов

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

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

$$\sum_{i=1}^{\ell} \ln e^{-L(y_i, f(x_i))} + \ln e^{-\gamma V(w)} \rightarrow \max_w$$

$$P(w) \sim \boxed{e^{-\gamma V(w)}} \prod_{i=1}^{\ell} e^{-L(y_i, f(x_i))} \rightarrow \max_w$$

## Вероятностный смысл регуляризаторов

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

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

$$\sum_{i=1}^{\ell} \ln e^{-L(y_i, f(x_i))} + \ln e^{-\gamma V(w)} \rightarrow \max_w$$

$$P(w) \sim e^{-\gamma V(w)} \prod_{i=1}^{\ell} e^{-L(y_i, f(x_i))} \sim P(x_i, y_i | w)$$

# Вероятностный смысл регуляризаторов

- $\ell_2$  – соответствует гауссовскому распределению (гауссовский регуляризатор)

$$V(w) \sim \exp\left(-\frac{(w - m)^2}{2\sigma^2}\right)$$

- $\ell_1$  – соответствует распределению Лапласа (лапласовский регуляризатор)

$$V(w) \sim \exp(-\alpha|w - m|)$$

Упражнение: покажите связь этих регуляризаторов с заявленным распределением

## Упражнение

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

# Резюме

1. Линейные модели классификации
  1. Простое решающее правило
  2. При обучении используется аппроксимация пороговой функции потерь
2. Метод стохастического градиента
  1. Проблемы со сходимостью
  2. Плохая обобщающая способность: требуется регуляризация
3. Регуляризация
  1. Эффективный способ избежать переобучения линейной модели
  2.  $\ell_1$  приводит также к отбору признаков

Отзывы о лекции: <https://goo.gl/forms/zeZiu1fSgrpPGp6T2>