

Лекция 3

# Линейные модели

Машинное обучение  
Сергей Муравьёв / Андрей Фильченков

18.09.2020

# План лекции

- Линейная классификация
  - Градиентный спуск
  - Линейная регрессия и матричное разложение
  - Регуляризация
- 
- В презентации используются материалы курса «Машинное обучение» К.В. Воронцова
  - Слайды доступны: [shorturl.at/ltVZ3](https://shorturl.at/ltVZ3)  
Видео доступны: [shorturl.at/hjyAX](https://shorturl.at/hjyAX)

# План лекции

- Рекламная интеграция
- Линейная классификация
- Градиентный спуск
- Линейная регрессия и матричное разложение
- Регуляризация

# Партнер курса



# Стипендии от Huawei

- Для третьекурсников:
  - взять курсовую по ML и защитить ее в июне
  - хорошо сдать курс
  - стипендия в 7-м семестре
- Для четверокурсников:
  - взять диплом по ML и защитить промежуточные результаты в январе
  - хорошо сдать курс
  - стипендия в 8-м семестре



# План лекции

- Линейная классификация
- Градиентный спуск
- Линейная регрессия и матричное разложение
- Регуляризация

# Формулировка задачи

**Условия:**  $Y = \{-1, +1\}$

Дано  $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^{|\mathcal{D}|}$

Требуется найти  $a_w(x, \mathcal{D})$  в виде  $\text{sign}(f(x, w))$ ,  
где  $f(x, w)$  — функция распознавания,  
 $w$  — вектор параметров

**Ключевая гипотеза:** объекты (хорошо) разделимы.

**Основная идея:** поиск среди разделяющих  
поверхностей, описываемых уравнением  $f(x, w) = 0$ .

# Понятие отступа

**Функция отступа (margin)** для объекта  $x_i$ :

$$M_i(w) = y_i f(x_i, w),$$

$M_i(w) < 0$  — свидетельство того, что объект классифицирован некорректно.



# Сглаживание функции ошибки

Эмпирический риск:

$$\mathcal{L}(a_w, \mathcal{D}) = \mathcal{L}(w) = \sum_i^{|D|} [M_i(w) < 0]$$

отражает количество объектов, на которых классификатор  $a_w$  допускает ошибки.

Функция не является гладкой  $\rightarrow$  невозможен поиск экстремумов

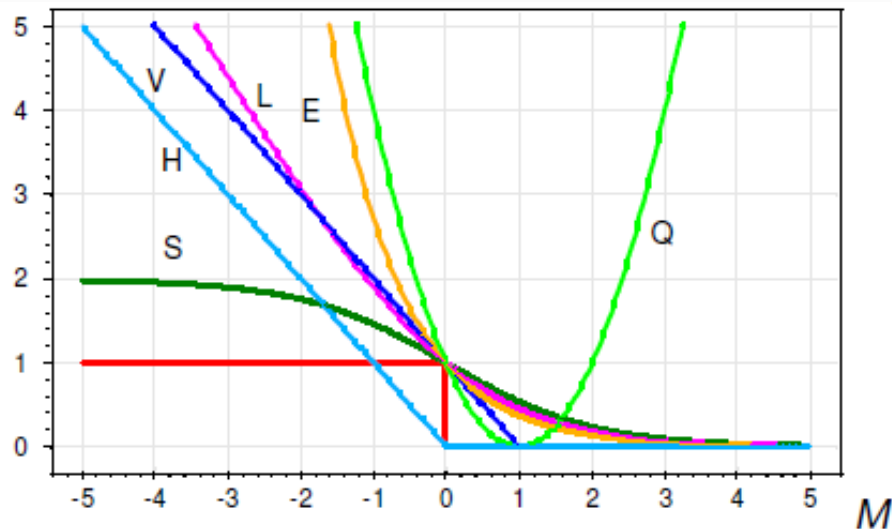
Аппроксимация:

$$\tilde{\mathcal{L}}(w) = \sum_i^{|D|} \mathcal{L}(M_i(w)),$$

где  $\mathcal{L}(M_i(w)) = \mathcal{L}(a_w(x, \mathcal{D}), x_i)$  – функция потерь.

# Гладкие функции ошибки

Требуется, чтобы  $\mathcal{L}$  была неотрицательной, невозрастающей и гладкой:



$H(M) = (-M)_+$	— кусочно-линейная (правило Хебба)
$V(M) = (1 - M)_+$	— кусочно-линейная (SVM)
$L(M) = \log_2(1 + e^{-M})$	— логарифмическая
$Q(M) = (1 - M)^2$	— квадратичная
$S(M) = 2(1 + e^M)^{-1}$	— сигмоидная
$E(M) = e^{-M}$	— экспоненциальная

# Линейный классификатор

$f_j: X \rightarrow \mathbb{R}, j = 1, \dots, n$  — численные признаки

**Линейный классификатор:**

$$a_w(x, \mathcal{D}) = \text{sign} \left( \sum_{i=1}^n w_i f_i(x) - w_0 \right).$$

$w_1, \dots, w_n \in \mathbb{R}$  — **веса** признаков.

Эквивалентная запись:

$$a_w(x, \mathcal{D}) = \text{sign}(\langle w, x \rangle),$$

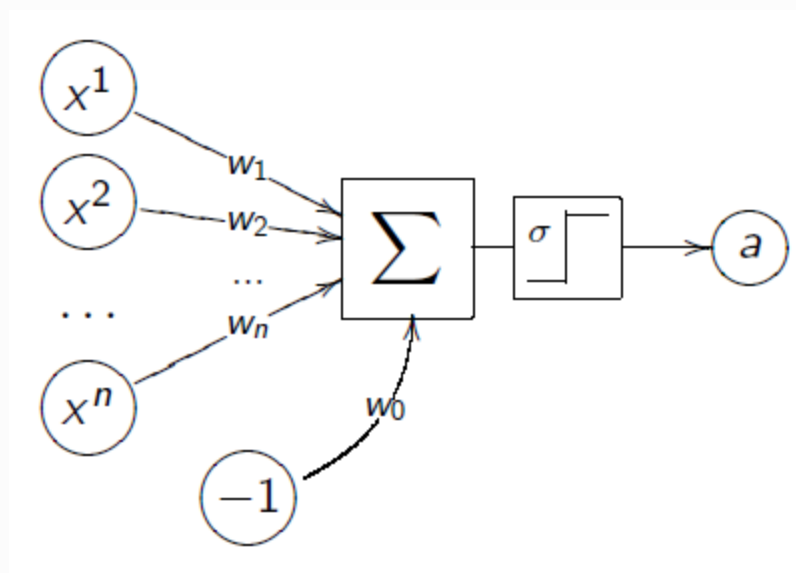
если добавить признак  $f_0(x) = -1$ .

# Нейрон

Нейрон МакКаллока-Питтса:

$$a_w(x, \mathcal{D}) = \sigma \left( \sum_{i=1}^n w_i f_i(x) - w_0 \right),$$

где  $\sigma$  — функция активации



# Семейство алгоритмов

Предположение о том, как должен выглядеть классификатор, задаёт семейство алгоритмов  $A_{\text{linear}}$ , из которого требуется выбрать конкретный алгоритм.

**Как выглядит такое семейство алгоритмов?**

# Семейство алгоритмов

Предположение о том, как должен выглядеть классификатор, задаёт семейство алгоритмов  $A_{\text{linear}}$ , из которого требуется выбрать конкретный алгоритм.

Как выглядит такое семейство алгоритмов?

$$A_{\text{linear}} = \{a_w(x) = \text{sign}(\langle w, x \rangle) | w \in \mathbb{R}^n\}$$

# Как обучать классификатор?

Требуется найти вектор параметров  $w$ .

Мы можем использовать практически любой алгоритм оптимизации, способный оптимизировать эмпирический риск в соответствующем пространстве.

Эмпирический риск не является функцией «чёрного ящика»

Более того, известно, что эта функция гладкая.

# План лекции

- Линейная классификация
- Градиентный спуск
- Линейная регрессия и матричное разложение
- Регуляризация



# Задача оптимизации

Дано:  $X \subset \mathbb{R}^n$  — допустимое множество

Целевая функция  $f: X \rightarrow R$

Критерий поиска: **минимум** или **максимум**

**Классификация методов оптимизации (методы поиска):**

- детерминированные;
- случайные (стохастические);
- комбинированные

**Классификация методов по критерию размерности  $X$ :**

- многомерные методы
- одномерные методы

# Классификация с точки зрения $X$

- задачи дискретного программирования —  $X$  конечно или счётно;
- задачи целочисленного программирования —  $X \subset \mathbb{Z}$ ;
- задачи нелинейного программирования, если  $f$  — нелинейная функция,  $X \subset K^n$ ,  $|K^n| < \infty$ .
- задачи линейного программирования, если  $f$  — линейная функция.

# Классификация с точки зрения гладкости $f$

- **Прямые** методы — вычисления целевой функции в точках приближений  
Метод перебора, метод золотого сечения.
- Методы **первого порядка** — вычисления первых частных производных  
Градиентный спуск, метод Коши
- Методы **второго порядка** — вычисления вторых частных производных, то есть *гессиана* целевой функции  
Метод Ньютона

# Градиентный спуск

Задача минимизации эмпирического риска

$$\tilde{\mathcal{L}}(w) = \sum_i^{|D|} \mathcal{L}(M_i(w)) = \sum_i^{|D|} \mathcal{L}(\langle w, x_i \rangle y_i) \rightarrow \min_w.$$

Градиентный спуск (классический):

$w_{(0)}$  — **некоторое начальное значение**;

$$w_{(k+1)} = w_{(k)} - \mu \nabla \mathcal{L}(w_{(k)}),$$

где  $\mu$  — **шаг градиента** или **скорость сходимости**.

$$w_{(k+1)} = w_{(k)} - \mu \sum_i^{|D|} \mathcal{L}'(\langle w, x_i \rangle y_i) x_i y_i.$$

# Стохастический градиентный спуск

**Проблема:** существует слишком много объектов, функции которых необходимо переоценивать на каждом шаге.

**Стохастический градиентный спуск:**

$w_{(0)}$  — некоторое начальное значение;

$x_{(1)}, \dots, x_{(|\mathcal{D}|)}$  — некоторый порядок объектов;

$$w_{(k+1)} = w_{(k)} - \mu \mathcal{L}'(\langle w_{(k)}, x_{(k)} \rangle y_{(k)}) x_{(k)} y_{(k)},$$

$$\mathcal{L}_{(k+1)} = (1 - \alpha) \mathcal{L}_{(k)} + \alpha \mathcal{L}(\langle w_{(k)}, x_{(k)} \rangle y_{(k)}).$$

Критерий останова: когда значения  $\mathcal{L}$  и/или  $w$  почти не меняются

# Пакетный градиентный спуск

**Проблема:** стохастический градиентный спуск слишком случайный, поскольку зависит только от одного объекта.

**Пакетный градиентный спуск (mini-batch):**

$w_{(0)}$  — некоторое начальное значение;  $b$  — размер пакета (батча)

$x_{(1)}, \dots, x_{(|\mathcal{D}|)}$  — некоторый порядок объектов;

$$w_{(K+1)} = w_{(K)} - \mu \sum_{k=Kb}^{(K+1)b} \mathcal{L}'(\langle w_{(K)}, x_{(k)} \rangle y_{(k)}) x_{(k)} y_{(k)},$$

$$\mathcal{L}_{(K+1)} = (1 - \alpha) \mathcal{L}_{(K)} + \alpha \sum_{k=Kb}^{(K+1)b} \mathcal{L}(\langle w_{(K)}, x_{(k)} \rangle y_{(k)}).$$

Критерий останова: когда значения  $\mathcal{L}$  и/или  $w$  почти не меняются

# Сравнение подходов



- Классический градиентный спуск
- Пакетный градиентный спуск
- Стохастический градиентный спуск

# Правило Розенблатта и правило Хебба

**Правило Розенблатта** для классификации на множестве  $\{1; 0\}$ , служащее для настройки весов: для каждого  $x_{(k)}$  изменяем вектор весов:

$$w_{(k+1)} := w_{(k)} - \eta(a_w(x_{(k)}) - y_{(k)})x_{(k)}.$$

**Правило Хебба** для классификации на множестве  $\{1; -1\}$ , служащее для настройки весов: для каждого  $x_{(k)}$  изменяем вектор весов: если  $\langle w_{(k)} x_{(k)} \rangle y_{(k)} < 0$  то  $w_{(k+1)} := w_{(k)} + \eta x_{(k)} y_{(k)}$ .



# Теорема Новикова

## Теорема (Алекс Новиков)

Пусть выборка  $\mathcal{D}$  линейно разделима:  $\exists \tilde{w}, \exists \delta > 0$ :

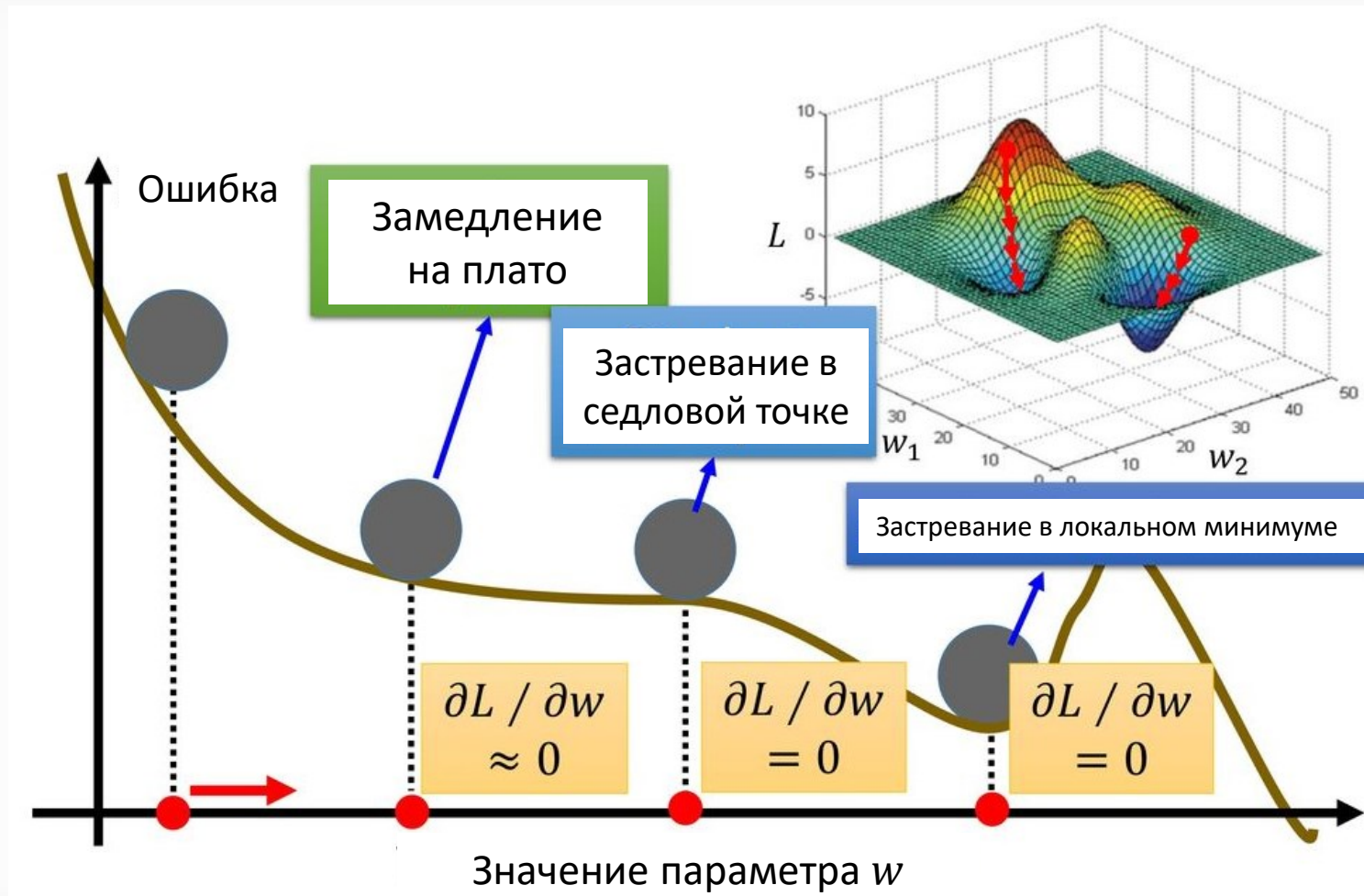
$\langle \tilde{w}, x_i \rangle y_i > \delta$  для всех  $i = 1, \dots, |\mathcal{D}|$ .

Тогда стохастический градиентный спуск с правилом Хебба находит вектор весов  $w$ , который:

- разделяет выборку безошибочно;
- при любом начальном значении  $w_{(0)}$ ;
- при любом  $\mu > 0$ ;
- независимо от порядка объектов  $x_{(i)}$ ;
- за конечное число изменений вектора  $w$ ;
- если  $w_{(0)} = 0$ , тогда количество изменений вектора  $w$ :

$$t_{\max} \leq \frac{1}{\delta^2} \max ||x_j||.$$

# Проблемы сходимости



# Эвристики для начальных значений

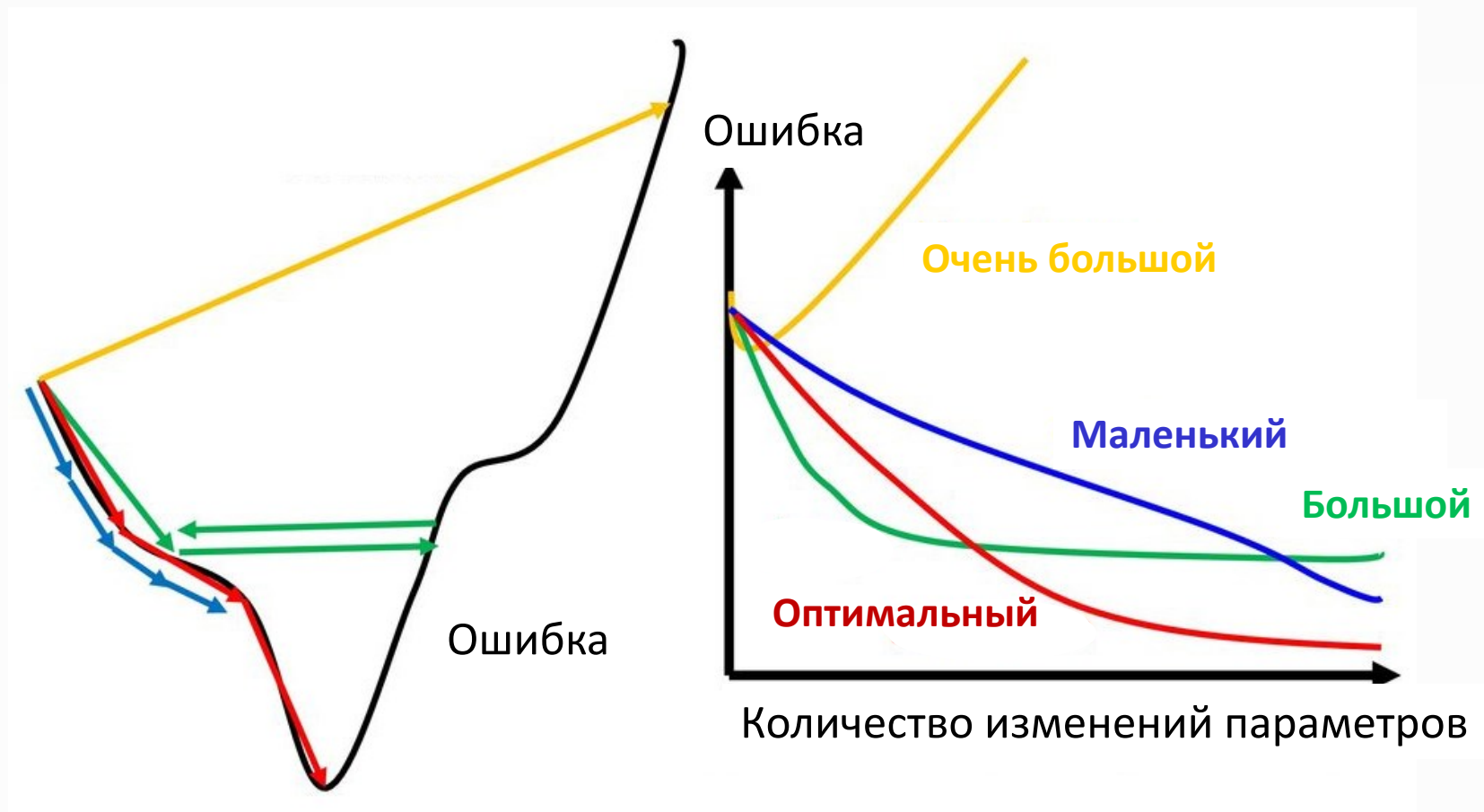
Важно для невыпуклых функций:

- $w_j = 0$  для всех  $j = 0, \dots, n$ ;
- маленькие случайные значения:

$$w_j \in \left[ -\frac{1}{2n}, \frac{1}{2n} \right];$$

- $w_j = \frac{\langle y, f_j \rangle}{\langle f_j, f_j \rangle}$ ;
- обучение по небольшой случайной подвыборке объектов;
- многократные запуски из разных начальных приближений и выбор лучшего решения.

# Сравнение скоростей сходимости



# Эвристики на скорость сходимости

- Сходимость достигается для выпуклых функций, когда

$$\mu_{(k)} \rightarrow 0, \sum \mu_{(k)} = \infty, \sum (\mu_{(k)})^2 < \infty$$

- **Наискорейший спуск:**

$$\mathcal{L} \left( w_{(k)} - \mu_{(k)} \nabla \mathcal{L}(w_{(k)}) \right) \rightarrow \min_{\mu_{(k)}}$$

- Шаги для «выпрыгивания» из локальных минимумов
- Методы второго порядка
- Использование среднего вектора недавних шагов

# Эвристики на порядок предъявления объектов

- на каждом шаге брать предметы из разных классов;
- чаще брать неверно классифицированные объекты (маленькие  $M_i$ );
- чаще не брать «хорошие» объекты, у которых  $M_i > \kappa_+$ ;
- чаще не брать объекты-«шумы», у которых  $M_i < \kappa_-$ ;

# Анализ алгоритма градиентного спуска

Преимущества:

- простота реализации;
- легко обобщается для любых  $f$  и  $\mathcal{L}$ ;
- возможность динамического обучения;
- поддерживает сверхмалые выборки.

Недостатки:

- медленная сходимость, возможна расходимость;
- застревание в локальных минимумах и седловых точках;
- очень важен правильный подбор эвристик;
- переобучение.

# План лекции

- Линейная классификация
- Градиентный спуск
- Линейная регрессия и матричное разложение
- Регуляризация



# Линейная регрессия

Модель многомерной линейной регрессии:

$$f(x, \theta) = \sum_{j=1}^n \theta_j f_j(x), \quad \theta \in \mathbb{R}^n.$$

Матричные обозначения:

$$F = \begin{pmatrix} f_1(x_1) & \dots & f_n(x_1) \\ \dots & \dots & \dots \\ f_1(x_{|\mathcal{D}|}) & \dots & f_n(x_{|\mathcal{D}|}) \end{pmatrix}, y = \begin{pmatrix} y_1 \\ \dots \\ y_{|\mathcal{D}|} \end{pmatrix}, \theta = \begin{pmatrix} \theta_1 \\ \dots \\ \theta_n \end{pmatrix}.$$

Эмпирический риск в матричной записи:

$$\mathcal{L}(\theta, \mathcal{D}) = \sum_{i=1}^{|\mathcal{D}|} (f(x_i, \theta) - y_i)^2 = \|F\theta - y\|^2 \rightarrow \min_{\theta \in \mathbb{R}^n}.$$

# Матричное разложение

Существует много способов решения такой задачи.

Один из наиболее популярных — применение метода **матричного разложения** (или **матричной факторизации**).

# Система нормальных уравнений

Условие минимума:

$$\frac{\partial \mathcal{L}(\theta)}{\partial \theta} = 2F^\top (F\theta - y) = 0.$$

$$\theta^* = (F^\top F)^{-1} F^\top y$$

$F^+ = (F^\top F)^{-1} F^\top$  — псевдообратная матрица (обратное преобразование Мура-Пенроуза)

$P_F = FF^+$  — проекционная матрица

Решение:

$$\theta^* = F^+ y.$$

Минимальное приближение:

$$\mathcal{L}(\theta^*) = \|P_F y - y\|^2.$$

# Сингулярное разложение

**Теорема:** любая матрица  $F$  размера  $|\mathcal{D}| \times n$  может быть представлена в виде сингулярного разложения

$$F = VDU^T.$$

- $V = (v_1, \dots, v_n)$  размера  $|\mathcal{D}| \times n$ , являющаяся ортогональной:  $V^T V = I_n$ , столбцы  $v_j$  — собственные вектора матрицы  $FF^T$ ;
- $U = (u_1, \dots, u_n)$  размера  $n \times n$ , являющаяся ортогональной:  $U^T U = I_n$ , столбцы  $u_j$  — собственные вектора матрицы  $F^T F$ ;
- $D = \text{diag}(\sqrt{\lambda_1}, \dots, \sqrt{\lambda_n})$  размера  $n \times n$ ,  $\sqrt{\lambda_j}$  — **сингулярные числа**, квадратные корни собственных значений  $F^T F$ .

# Интерпретация метода

Представим какое-то скрытое пространство, в которое мы хотим проецировать данные.

$D$  представляет важность каждого базисного вектора

$V$  представляет, как объекты соответствуют базисным векторам

$U$  показывает, как признаки соответствуют базисным векторам

# Метод наименьших квадратов при помощи сингулярного разложения

$$F^+ = (UDV^T VDU^T)^{-1}UDV^T = UD^{-1}V^T = \sum_{j=1}^n \frac{1}{\sqrt{\lambda_j}} u_j v_j^T;$$

$$\theta^* = F^+ y = UD^{-1}V^T y = \sum_{j=1}^n \frac{1}{\sqrt{\lambda_j}} u_j (v_j^T y);$$

$$F\theta^* = P_F y = (VDU^T)UD^{-1}V^T y = VV^T y = \sum_{j=1}^n v_j (v_j^T y);$$

$$\|\theta^*\|^2 = \|D^{-1}V^T y\|^2 = \sum_{j=1}^n \frac{1}{\lambda_j} (v_j^T y)^2.$$

# Анализ

- Когда мы можем вычислить сингулярное разложение, мы можем легко найти решение для МНК.
- Сингулярное разложение вычисляется за  $O(|\mathcal{D}|n^2 + n^3)$
- Сингулярное разложение — важный инструмент во многих других задачах машинного обучения, в первую очередь, в снижении размерности.

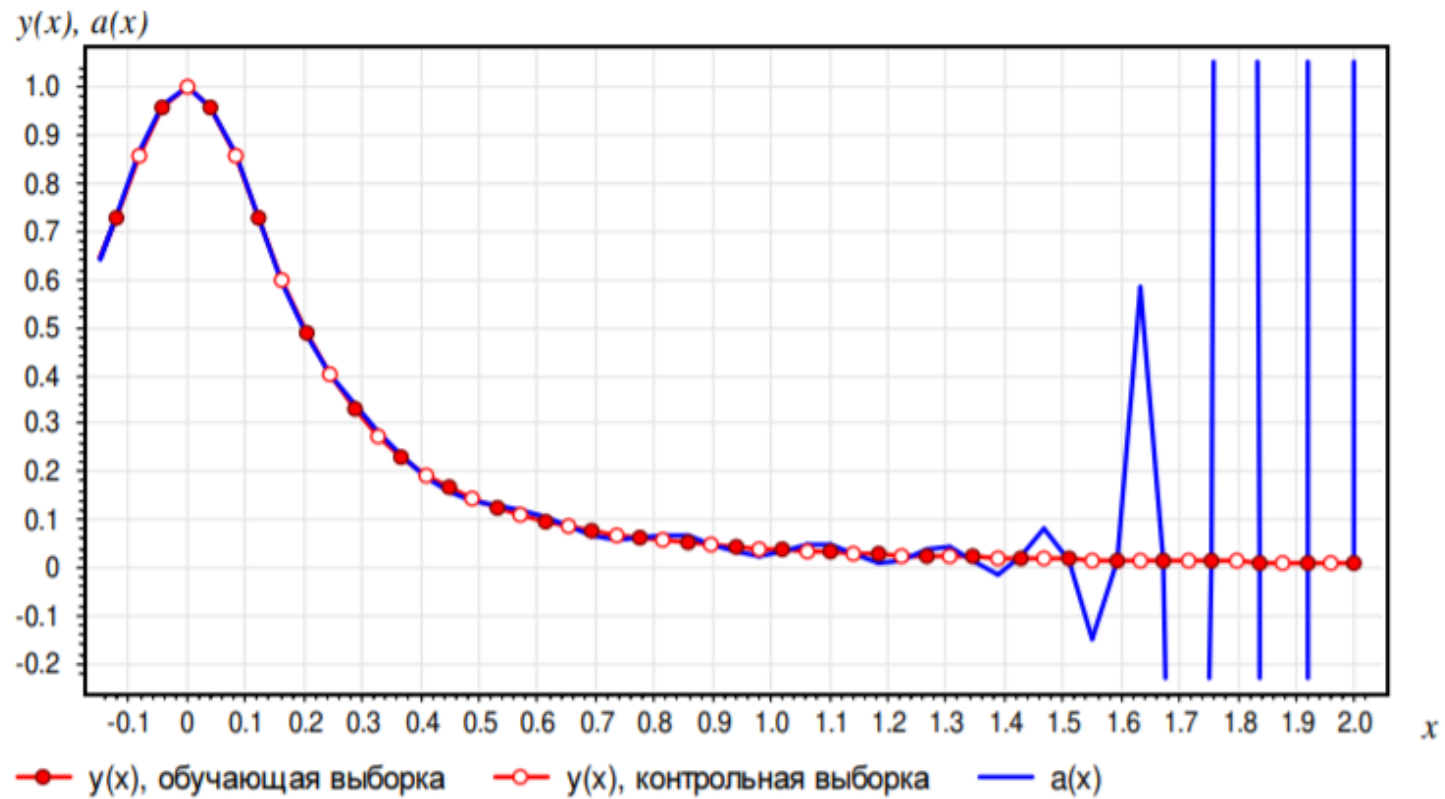
# План лекции

- Линейная классификация
- Градиентный спуск
- Линейная регрессия и матричное разложение
- Регуляризация



# Переобученный алгоритм (напоминание)

$$y(x) = \frac{1}{1 + 25x^2}; \quad a(x) \text{ — многочлен степени } n = 38$$



# Регуляризация

**Ключевая гипотеза:**  $w$  «скачет», что и вызывает переобучение

**Основная идея:** ограничим норму  $w$ .

Добавим штраф регуляризации для нормы весов:

$$\mathcal{L}_\tau(a_w, \mathcal{D}) = \mathcal{L}(a_w, \mathcal{D}) + \frac{\tau}{2} \|w\|^2 \rightarrow \min_w.$$

$\tau$  — коэффициент регуляризации, отражает баланс между качеством и обобщаемостью.

# Гребневая регуляризация

**Предположение:** значения вектора  $\theta$  имеют распределение Гаусса с ковариационной матрицей  $\sigma I_n$ :

$$\mathcal{L}_\tau(\theta) = \|F\theta - y\|^2 + \frac{1}{2\sigma} \|\theta\|^2 \rightarrow \min_{\theta},$$

где  $\tau = 1/\sigma$  — коэффициент регуляризации.

Внедрим такую регуляризацию в решение задачи МНК:

$$\theta_\tau^* = (F^\top F + \tau I_n)^{-1} F^\top y.$$

# МНК

$$\theta_{\tau}^* = U(D^2 + \tau I_n)^{-1} D V^T y = \sum_{j=1}^n \frac{\sqrt{\lambda_j}}{\lambda_j + \tau} u_j (v_j^T y);$$

$$\begin{aligned} F \theta_{\tau}^* &= (V D U^T) \theta_{\tau}^* = V \text{diag} \left( \frac{\lambda_j}{\lambda_j + \tau} \right) V^T y = \\ &= \sum_{j=1}^n \frac{\lambda_j}{\lambda_j + \tau} v_j (v_j^T y); \end{aligned}$$

$$\|\theta^*\|^2 = \|D^2 (D^2 + \tau I_n)^{-1} D^{-1} V^T y\|^2 = \sum_{j=1}^n \frac{1}{\lambda_j + \tau} (v_j^T y)^2.$$

# Лассо Тибширани

**Предположение:** значения вектора  $\theta$  имеют распределение Лапласа:

$$\begin{cases} \mathcal{L}_\tau(\theta) = ||F\theta - y||^2 \rightarrow \min_{\theta}; \\ \sum_{i=1}^n |\theta_i| \leq \kappa. \end{cases}$$

**LASSO** (least absolute shrinkage and selection operator).

# МНК с регуляризацией LASSO

Результирующая задача оптимизации

$$\mathcal{L}_\tau(\theta) = \|F\theta - y\|^2 + \tau\|\theta\|_1 \rightarrow \min_{\theta},$$

где  $\|\theta\|_1$  —  $l_1$ -норма:  $\|\theta\|_1 = \sum |\theta_i|$ .

Хорошего аналитического решения не существует.

Однако хорошее вычислительное решение существует.

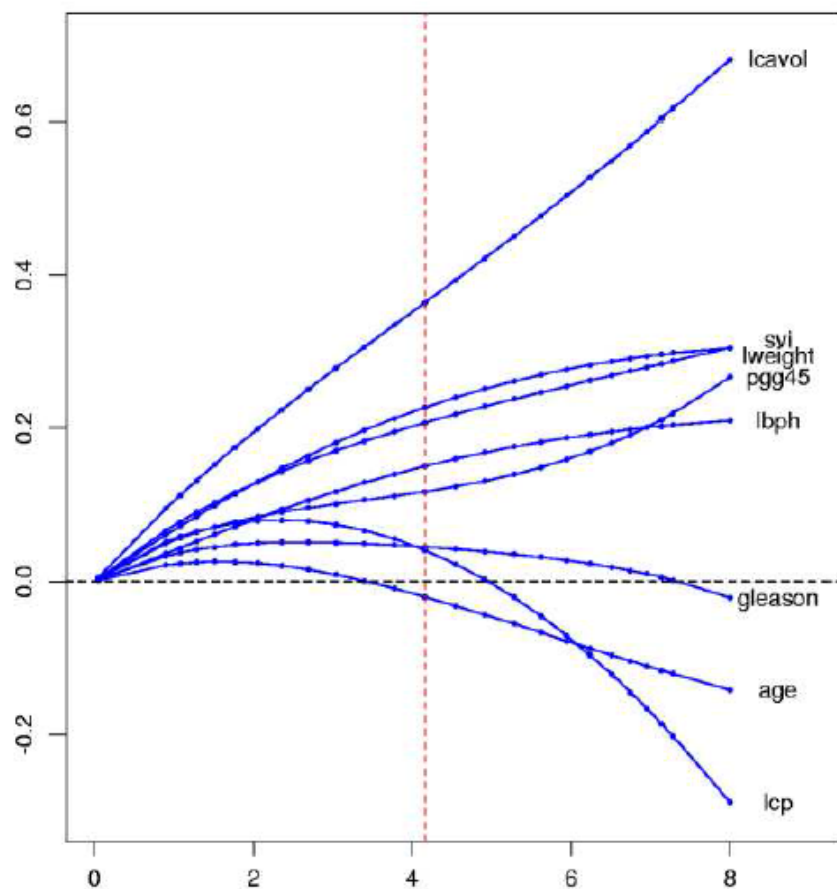
# Регуляризация для градиентного спуска

Регуляризация может быть легко использована для градиентного спуска:

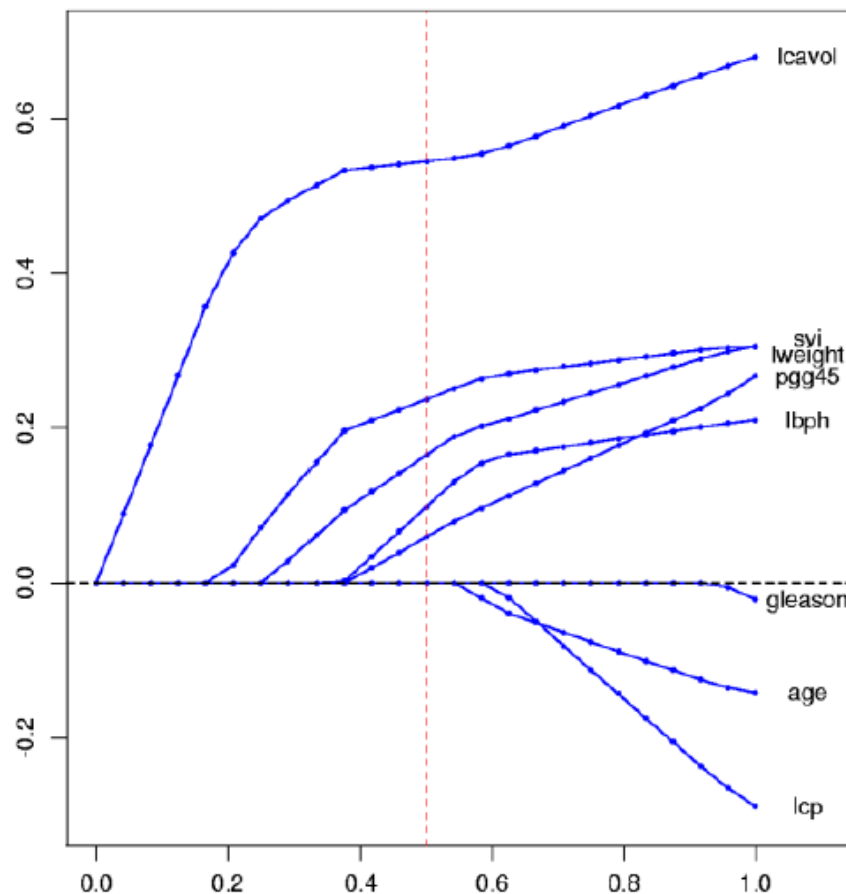
$$\begin{aligned}\nabla \mathcal{L}_\tau(w) &= \nabla \mathcal{L}(w) + \tau w, \\ w_{k+1} &= w_k(1 - \mu\tau) - \mu \nabla \mathcal{L}(w).\end{aligned}$$

# Сравнение методов регуляризации

## Гребневая



## LASSO





# Регуляризация в целом

Предположим, мы решаем задачу

$$\theta = \arg \min_{\theta \in \Theta} \mathcal{L}(\theta)$$

**Регуляризация** — это некоторые ограничения, которые мы накладываем на  $\theta$ , которые представляют сложность или вероятность модели и могут быть формализованы с помощью некоторой функции  $c(\theta)$  :

$$\theta_{\text{reg}} = \arg \min_{\theta \in \Theta} \mathcal{L}(\theta) + \alpha c(\theta)$$

# Анализ понятия регуляризации (1/2)

- Регуляризаторы, использующие  $l_1$ -норму или  $l_2$ -норму, наиболее популярные
- **ElasticNet**, являющийся суммой двух предыдущих также популярен.
- Многие другие могут быть использованы в отношении исходных предположений.
- Некоторые техники де-факто являются регуляризацией или могут быть интерпретированы подобным образом.

# Анализ понятия регуляризации (2/2)

- Один из двух общих подходов к борьбе с переобучением
- Требуется для решения плохо поставленных задач
- Регуляризаторы могут быть добавлены для представления определённых специфических свойств модели
- Коэффициент регуляризации необходимо настраивать
- Не всегда понятно, какой регуляризатор выбирать