# Машинное обучение и анализ данных

## Линейные методы

Дьяконов А.Г.

Московский государственный университет
имени М.В. Ломоносова (Москва, Россия)

# Линейные методы

## Линейная регрессия

**Гипотеза о линейной зависимости целевой переменной**

**Ищем решение в виде**

$$a(X_1,\ldots,X_n) = w_0 + w_1 X_1 + \ldots + w_n X_n$$

### Практика:

- **часто неплохо работает и при монотонных зависимостях**
- **хорошо работает, когда есть много «однородных» зависимостей:**

  **цель – число продаж**

  **признак 1 – число заходов на страницу продукта**

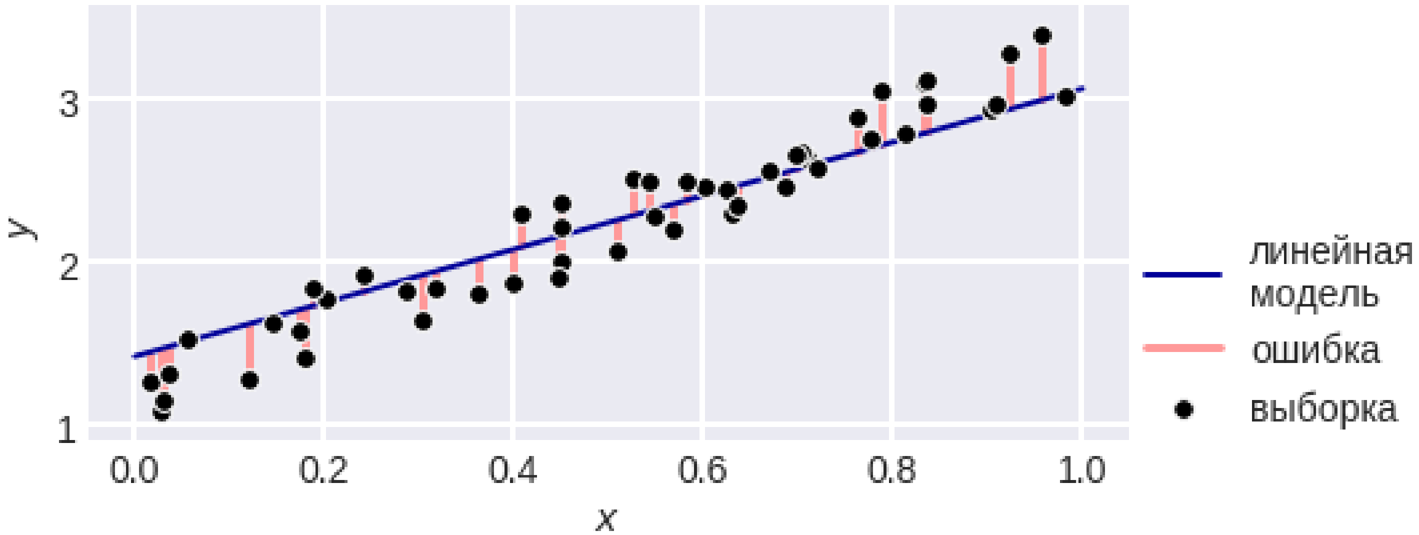
  **признак 2 – число добавлений в корзину**

  **признак 3 – число появлений продукта в поисковой выдачи**

  **...**

## Линейная регрессия от одной переменной

$$a(X_1,\ldots,X_n) = w_0 + w_1 X_1$$




**обучение:** $\{(x_1, y_1),\ldots,(x_m, y_m)\}$**,**

$x_i \in \mathbb{R}$**,**

$$\begin{cases} w_0 + w_1 x_1 = y_1 \\ \ldots \\ w_0 + w_1 x_m = y_m \end{cases}$$

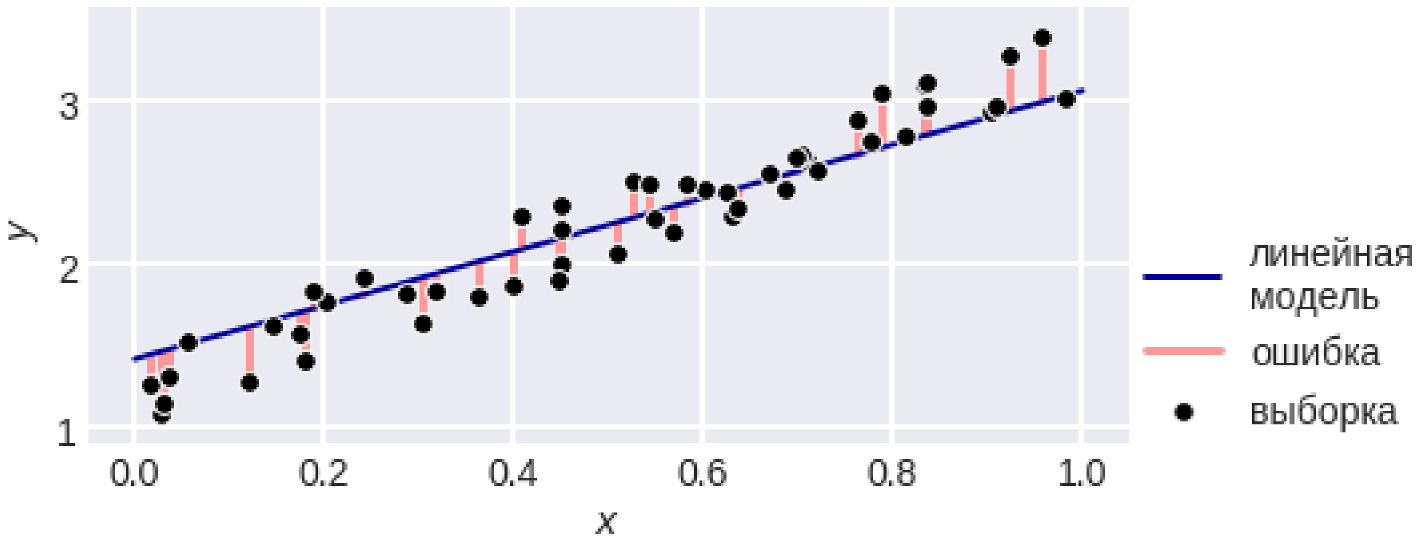
**невязки/отклонения (residuals):**

$$\begin{cases} e_1 = y_1 - w_0 + w_1 x_1 \\ \ldots \\ e_m = y_m - w_0 + w_1 x_m \end{cases}$$

# Линейная регрессия от одной переменной

**Задача минимизации суммы квадратов отклонений (residual sum of squares)**

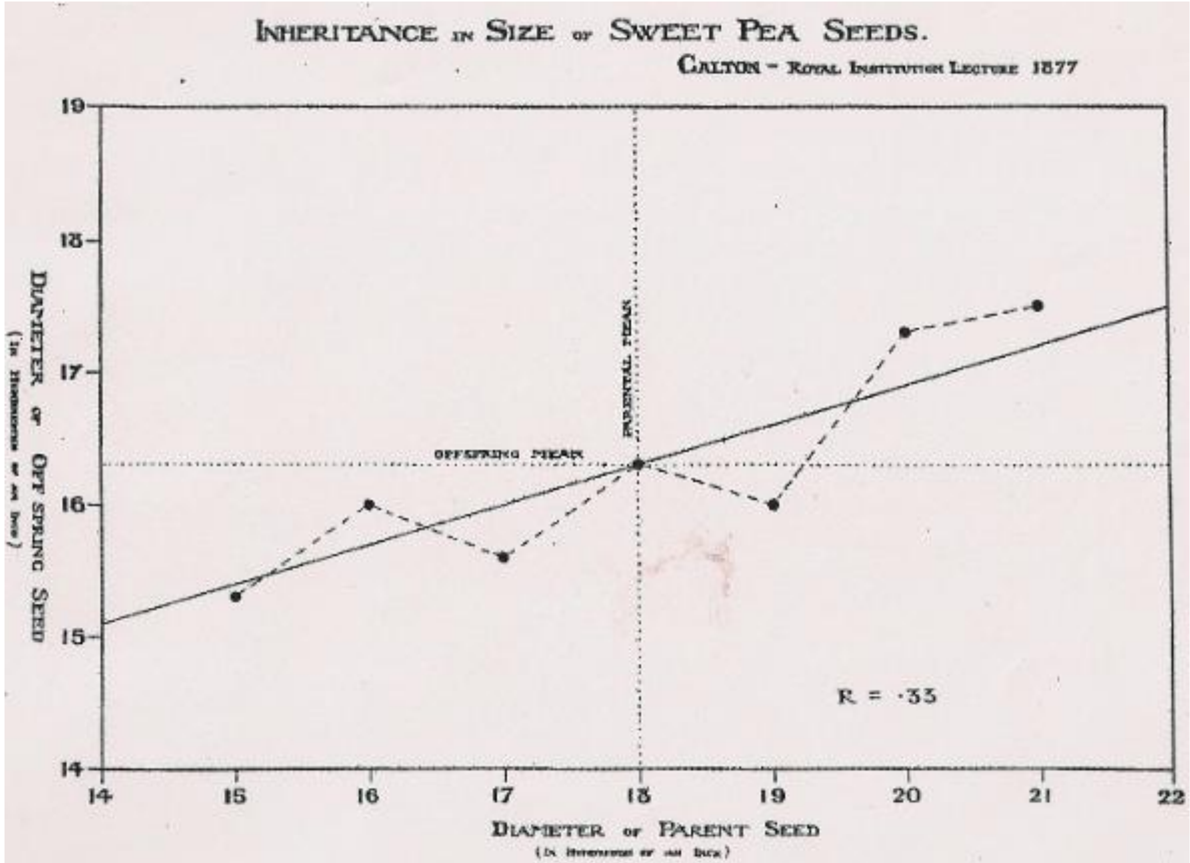
$$\mathrm{RSS} = e_1^2 + \ldots + e_m^2 \to \min$$




**~ задача описания данных гиперплоскостью (но ф-л качества!)**

**Есть вероятностное обоснование, но пока… логично**

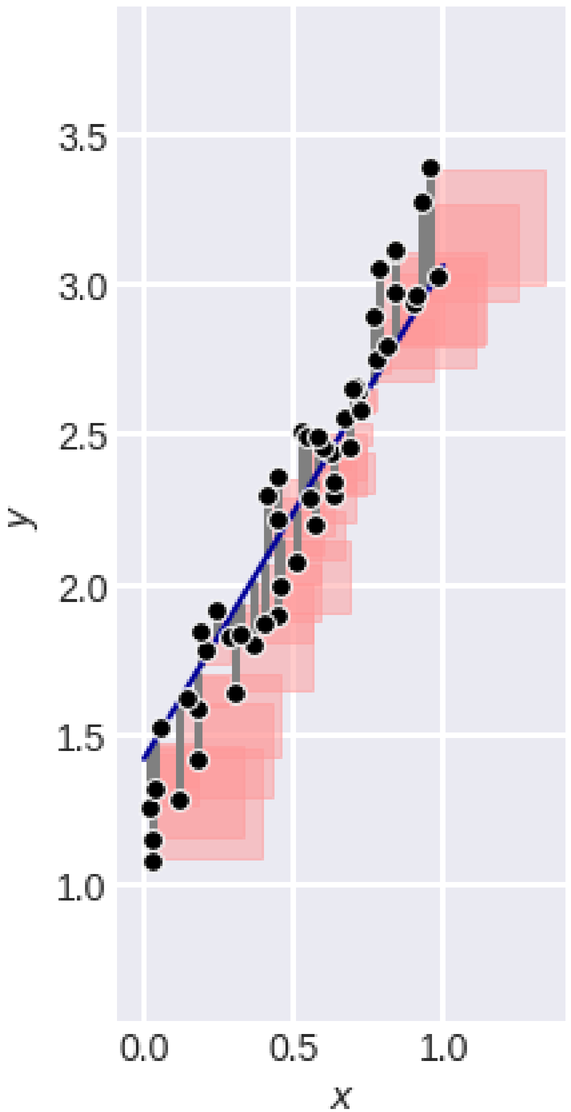
# Линейная регрессия от одной переменной

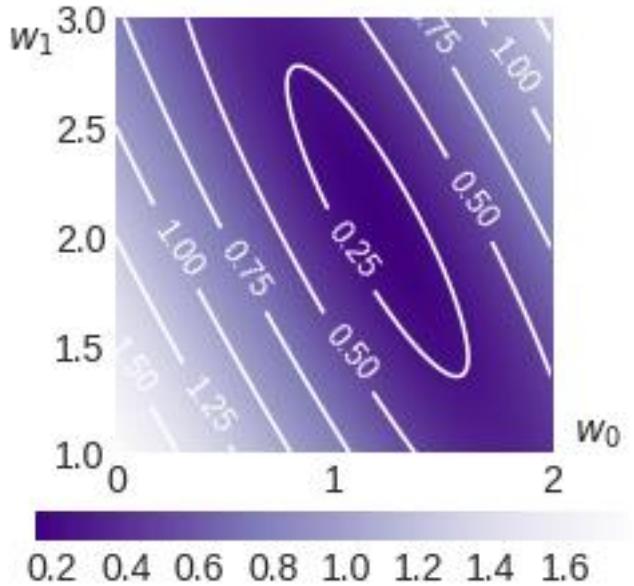



**Francis Galton, 1877**

## Линейная регрессия от одной переменной

### Геометрический смысл ошибки




**Отличается от суммы расстояний до поверхности!**

## Линейная регрессия от одной переменной

**Нетрудно показать (ДЗ):**

$$w_1 = \frac{\sum_{i=1}^{m}(x_i - \overline{x})(y_i - \overline{y})}{\sum_{i=1}^{m}(x_i - \overline{x})^2} = \frac{\operatorname{cov}(\{x_i\},\{y_i\})}{\operatorname{var}(\{x_i\})},$$

$$w_0 = \overline{y} - w_1\overline{x}.$$

**где** $\overline{x} = \sum_{i=1}^{m} x_i$, $\overline{y} = \sum_{i=1}^{m} y_i$.

## Общий случай (многих переменных)

$$a(X_1,\ldots,X_n) = w_0 + w_1X_1 + \cdots + w_nX_n = x^{\mathrm{T}}w$$

$$w = (w_0, w_1, \ldots, w_n)^{\mathrm{T}}$$

$$x = (X_0, X_1, \ldots, X_n)^{\mathrm{T}}$$

**для удобства записи вводим фиктивный признак** $x_0 \equiv 1$

**обучение:** $\{(x_1, y_1), \ldots, (x_m, y_m)\}$**,** $x_i \in \mathbf{R}^{n+1}$**,**

$$\begin{cases} x_1^{\mathrm{T}}w = y_1 \\ \ldots \\ x_m^{\mathrm{T}}w = y_m \end{cases}$$

$Xw = y$ **– как решать?**

## Общий случай (многих переменных)

**или в матричной форме**

$$Xw = y$$

**в матрице $X$ по строкам записаны описания объектов,**
**в векторе $y$ значения их целевого признака**

**(здесь есть коллизия в обозначении y)**

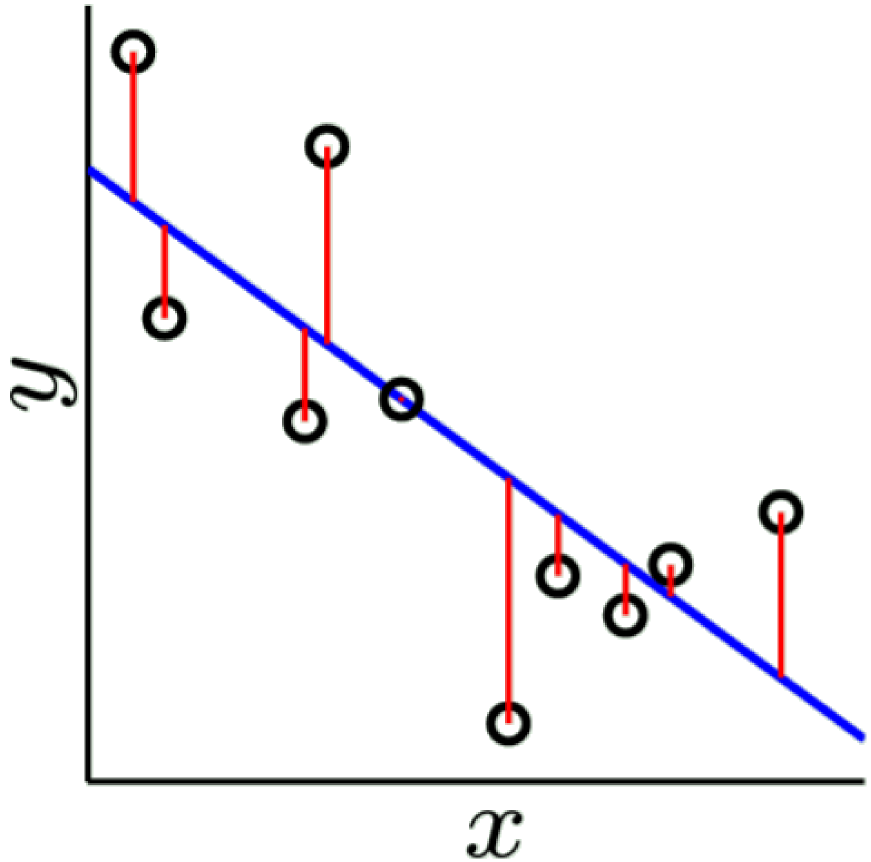
**будем решать так:**

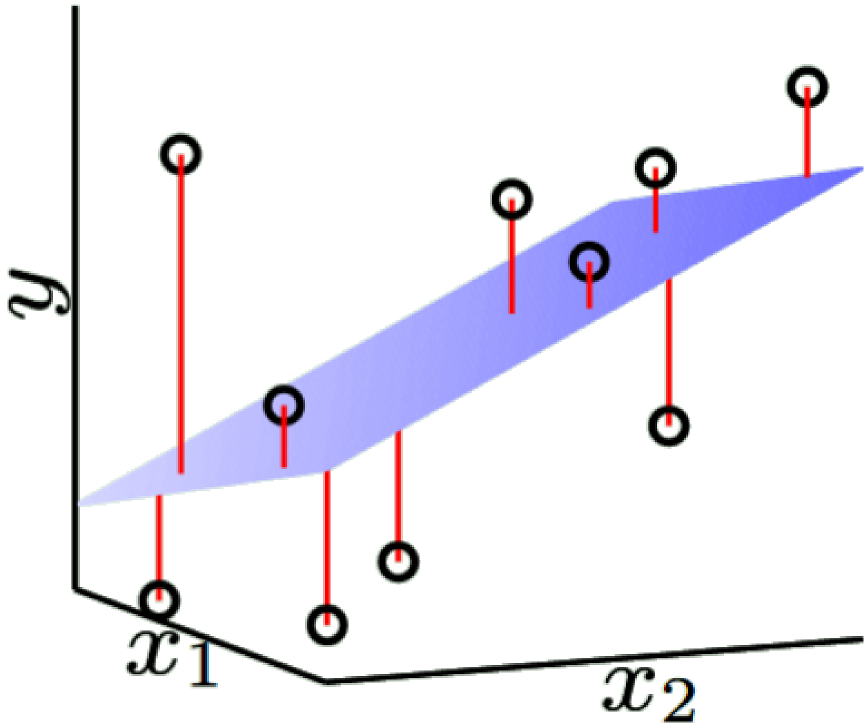
$$\| Xw - y \|_2^2 \to \min_w$$

**почему?**

## Общий случай (многих переменных)

**геометрический смысл**



## Решение задачи минимизации

$$\| Xw - y \|_2^2 \to \min_w$$

$$\| Xw - y \|_2^2 = (Xw - y)^{\mathrm{T}} (Xw - y) =$$
$$w^{\mathrm{T}} X^{\mathrm{T}} Xw - w^{\mathrm{T}} X^{\mathrm{T}} y - y^{\mathrm{T}} Xw + y^{\mathrm{T}} y$$

$$\nabla \| Xw - y \|_2^2 = 2X^{\mathrm{T}} Xw - 2X^{\mathrm{T}} y = 0$$
$$X^{\mathrm{T}} Xw = X^{\mathrm{T}} y$$
$$w = (X^{\mathrm{T}} X)^{-1} X^{\mathrm{T}} y$$

**решение существует, если столбцы линейно независимые**

$(X^{\mathrm{T}} X)^{-1} X^{\mathrm{T}}$ **– псевдообратная матрица Мура-Пенроуза**

**обобщение обратной на неквадратные матрицы**

# Обобщённая линейная регрессия

**вместо X – что угодно**

$$a(X_1,\ldots,X_n) = w_0 + w_1\varphi_1(X_1,\ldots,X_n) + \cdots + w_k\varphi_k(X_1,\ldots,X_n) = x^{\mathrm{T}}w$$

$$w = (w_0, w_1,\ldots,w_k)^{\mathrm{T}}$$

$$x = (X_0, X_1,\ldots,X_n)^{\mathrm{T}}$$

$$\varphi(x) = (\underbrace{\varphi_0(x)}_{\equiv 1}, \varphi_1(x),\ldots,\varphi_k(x))^{\mathrm{T}}$$

$$a(x) = \sum_{i=1}^{k} w_i\varphi_i(x) = \varphi(x)^{\mathrm{T}}w$$

**базисные функции (basis functions)**

**они фиксированы**

**Подробности в нелинейных методах…**

## Проблема вырожденности матрицы

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

**Решения:**

1. **Регуляризация – здесь и в «сложности»**
2. **Селекция (отбор) признаков – «селекция» / «PZAD»**
3. **Уменьшение размерности (в том числе, PCA) – USL**
4. **Увеличение выборки**

**если объектов много – то работать с гигантской матрицей невозможно… но выдели как это делается в оптимизации онлайн-методами**

# Регуляризация

**Упрощённое объяснение смысла регуляризации**

$$a(X_1,\ldots,X_n)=w_0+w_1X_1+\ldots+w_nX_n$$

**если есть два похожих объекта, то должны быть похожи метки**

**пусть отличаются в j-м признаке, тогда ответы модели отличаются на**

$$\varepsilon_j w_j$$

**Поэтому не должно быть больших весов**

**(у признаков, по которым могут отличаться похожие объекты)!**

**П.С. Плохо, когда модель заточена на один признак!**

**Поэтому вместе с** $\| Xw-y\|_2^2 \to \min$

**Хотим** $\| w\|_2^2 \to \min$

## Регуляризация

**Иванова**

$$\begin{cases} \| Xw - y \|_2^2 \to \min \\ \| w \|_2^2 \le \lambda \end{cases}$$

**Тихонова**

$$\| Xw - y \|_2^2 + \lambda \| w \|_2^2 \to \min$$

**Удобнее: безусловная оптимизация**

**Всё это справедливо и для общих задач минимизации!**

$$\begin{cases} L(a) \to \min \\ \mathrm{complexity}(a) \le \lambda \end{cases}$$

$$L(a) + \lambda\,\mathrm{complexity}(a) \to \min$$

**Часто эти две формы эквивалентны: решение одного можно получить как решение другого.**

**Есть ещё регуляризация Морозова…**

## Регуляризация

$$\arg\min \| Xw - y \|_2^2 + \lambda \| w \|_2^2 = (X^{\mathrm{T}} X + \lambda I)^{-1} X^{\mathrm{T}} y$$

**ДЗ Доказать!**

**– гребневая регрессия (Ridge Regression)**

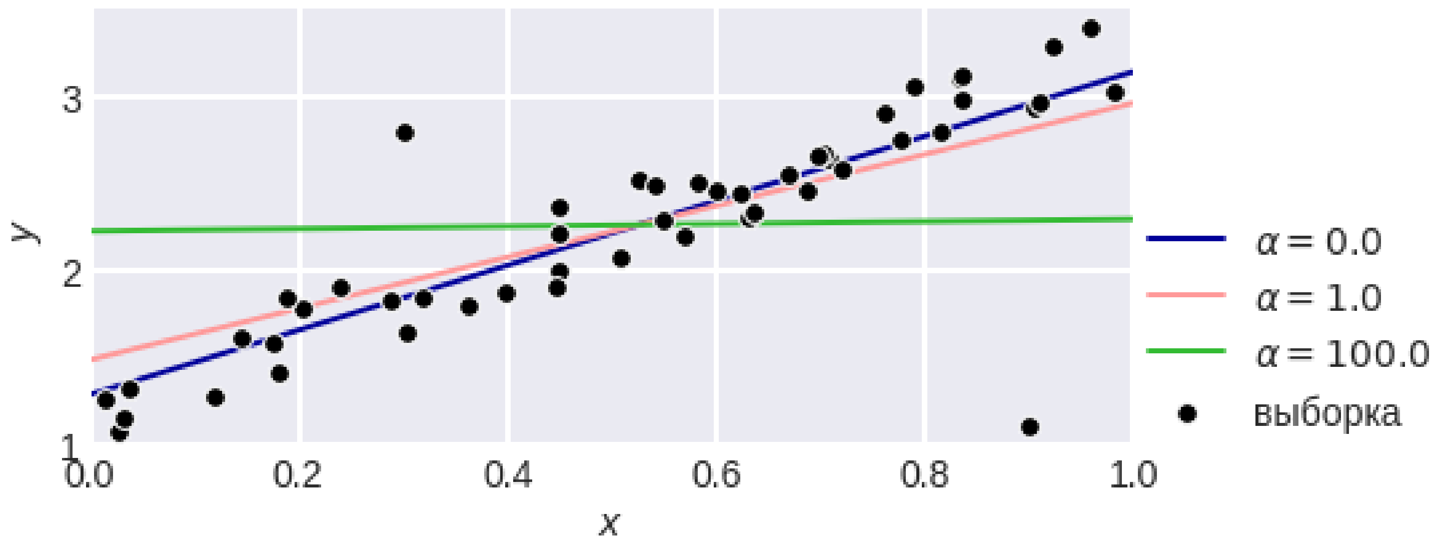
**Другой смысл – боремся с вырожденностью матрицы!**

$\lambda = 0$ **– получаем классическое решение**

$\lambda \to +\infty$ **– меньше «затачиваемся на данные» и больше регуляризуем**

**Матрица очевидно становится обратимой!**

## Регуляризация – минутка кода




```
from sklearn.linear_model import Ridge

model = Ridge(alpha=0.0) # ридж-регрессия
# обучение
model.fit(x_train[:, np.newaxis], y_train)
# обратите внимание: np.newaxis
# контроль
a_train = model.predict(x_train[:, np.newaxis])
a_test = model.predict(x_test[:, np.newaxis])
```

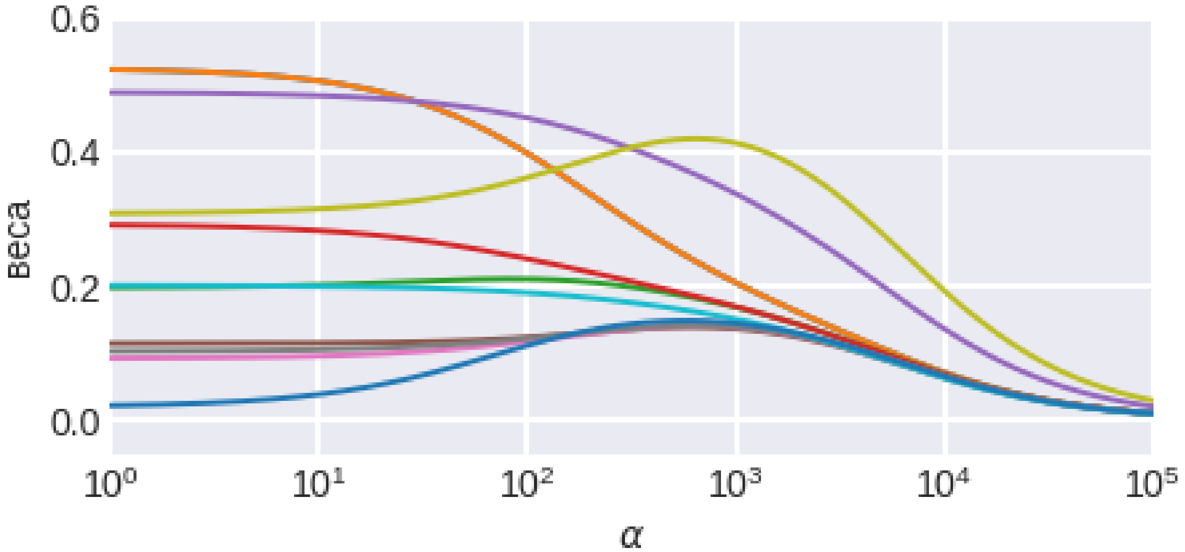
**Интересно, что рисунок неудачный – получилась антиреклама регуляризации… почему?**

## Ridge-регрессия

$$\sum_{i=1}^{m}(y_i - a(x_i))^2 + \lambda\sum_{j=1}^{n} w_j^2 \to \min$$

$$\lambda \geq 0$$

**добавление shrinkage penalty (регуляризатора)**




**параметр регуляризации может подбираться с помощью скользящего контроля**

## Ridge-регрессия

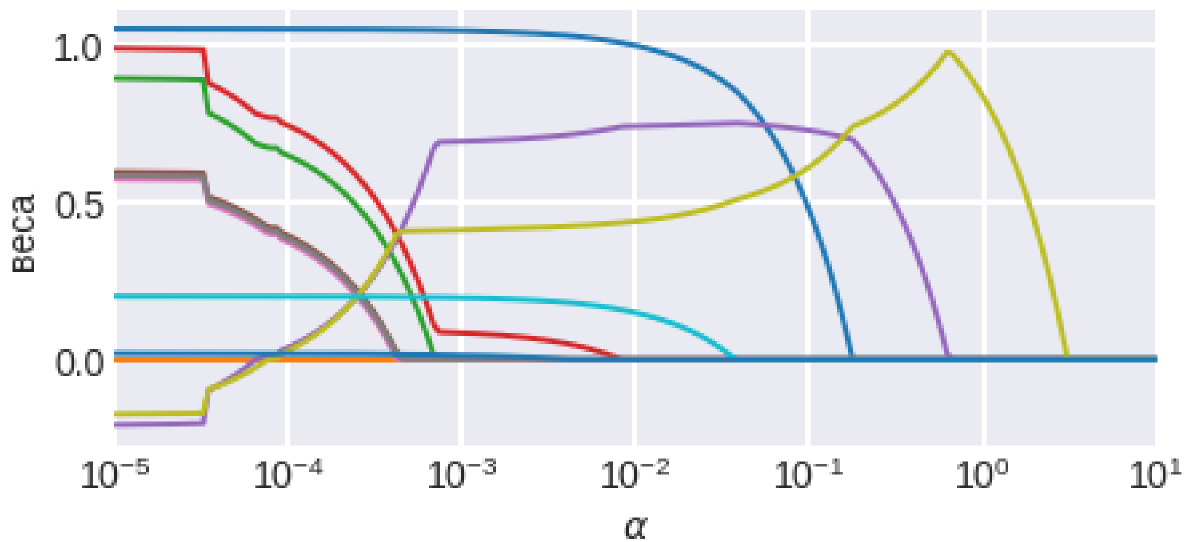
**Для ridge-регрессии нужна правильная нормировка признаков!**
**Нет инвариантности (в отличие от линейной) от умножения признаков на скаляры**

**Перед регуляризацией – стандартизация!!!**

# LASSO

$$\sum_{i=1}^{m}(y_i - a(x_i))^2 + \lambda \sum_{j=1}^{n} | w_j | \to \min$$

$$\lambda \geq 0$$




**Здесь коэффициенты интенсивнее зануляются при увеличении** $\lambda \geq 0$**.**

# Эксперименты с одинаковыми и зависимыми признаками

здесь была задача

```
X = np.random.rand(1000, 11)
X[:,1] = X[:,0]
X[:,4] = X[:,2] + X[:,3]
X[:,8] = X[:,5] + X[:,6] + X[:,7]
y = X[:,0] + 0.8*X[:,4] + 0.4*X[:,8] + 0.2*X[:,9] +
             0.5*np.random.randn(1000)
```

зависит от масштаба признаков, но из-за предварительной нормировки этот эффект не наблюдается

**Не на все коэффициенты нужна регуляризация!**

**Почему?**

**Масштаб очень важен! см. дальше**

## Семейство регуляризированных линейных методов

**Ridge**

$$\| y - Xw \|_2^2 + \lambda \| w \|_2^2 \to \min_w$$

**LASSO (Least Absolute Selection and Shrinkage Operator)**

$$\| y - Xw \|_2^2 + \lambda \| w \|_1^2 \to \min_w$$

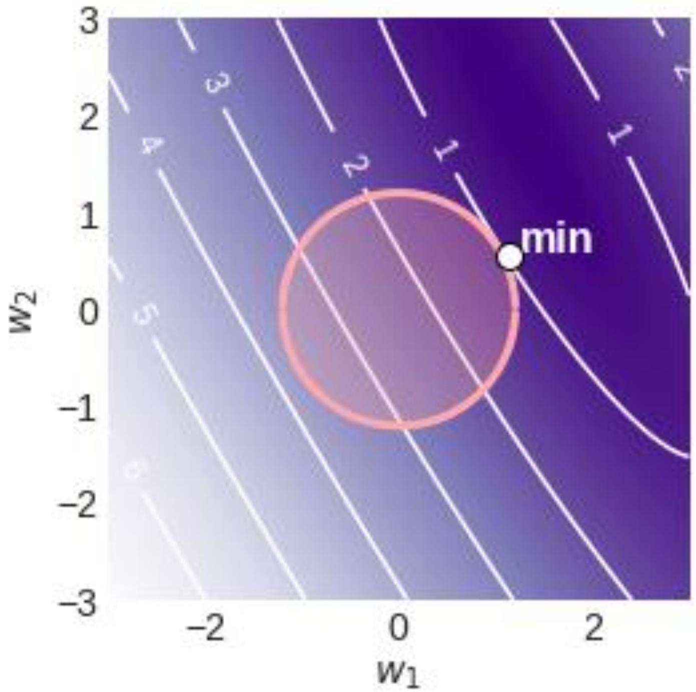
**Elastic Net = LASSO + Ridge**

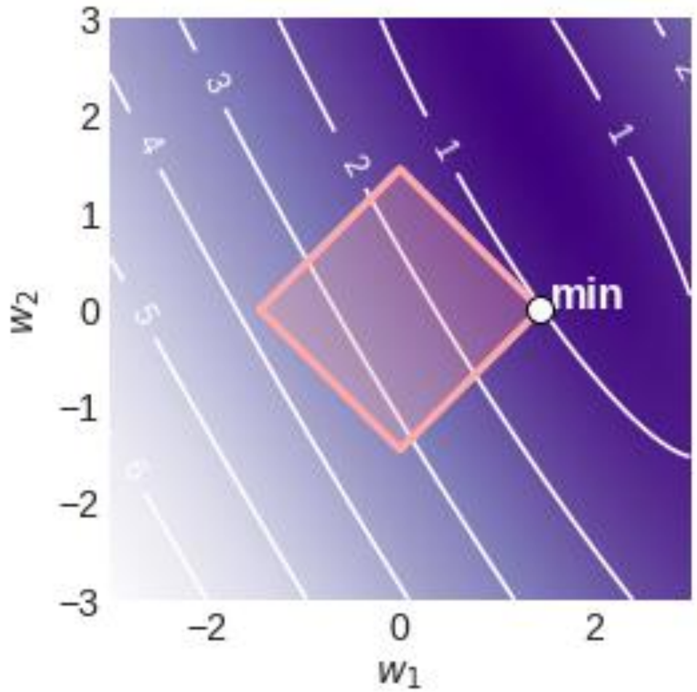
$$\| y - Xw \|_2^2 + \lambda_1 \| w \|_1^2 + \lambda_2 \| w \|_2^2 \to \min_w$$

## Геометрический смысл Ridge, LASSO и Elastic Net

$$\sum_{i=1}^{m}\left(y_i - w_0 - \sum_{j=1}^{n} w_j x_{ij}\right)^2 \to \min_w, \quad \sum_{j=1}^{n} w_j^2 \le s$$

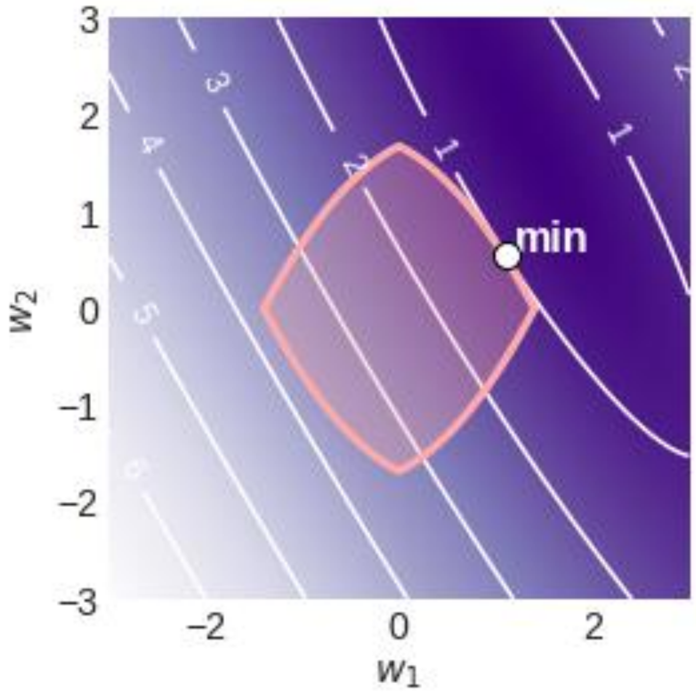
$$\sum_{i=1}^{m}\left(y_i - w_0 - \sum_{j=1}^{n} w_j x_{ij}\right)^2 \to \min_w, \quad \sum_{j=1}^{n} | w_j | \le s$$



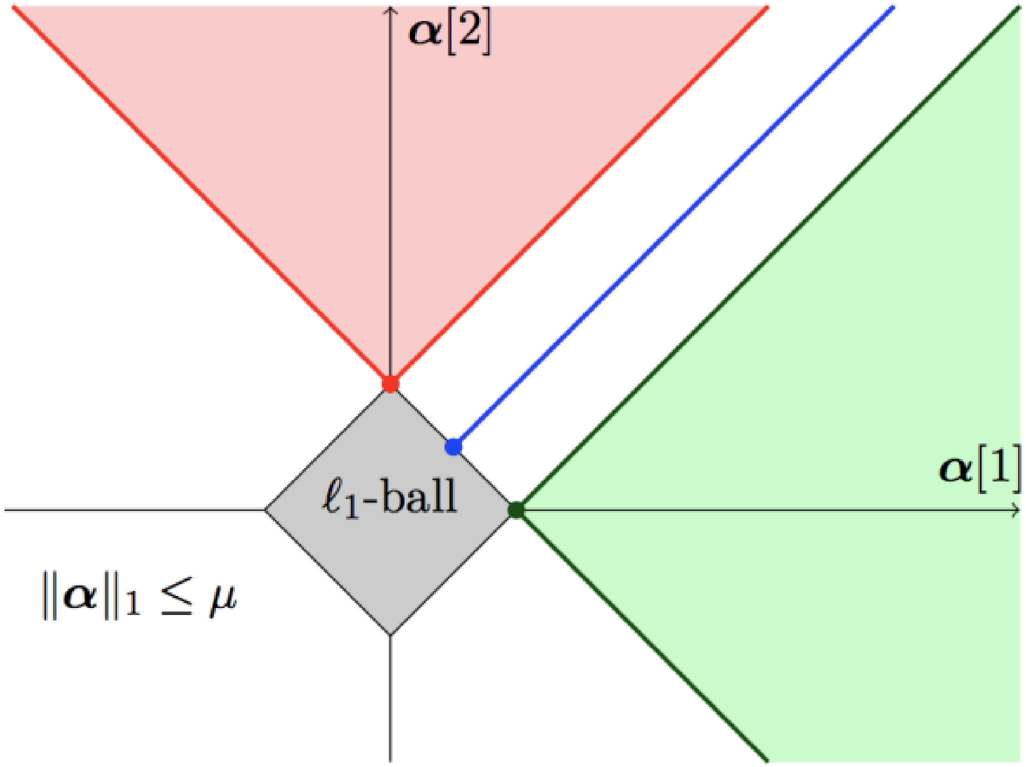
## Геометрический смысл Ridge, LASSO и Elastic Net

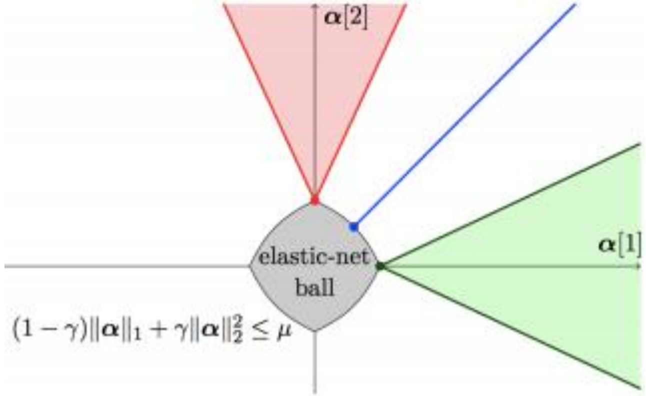
**Эти методы несравнимы… на практике часто модель и не может зависеть от небольшого числа переменных.**

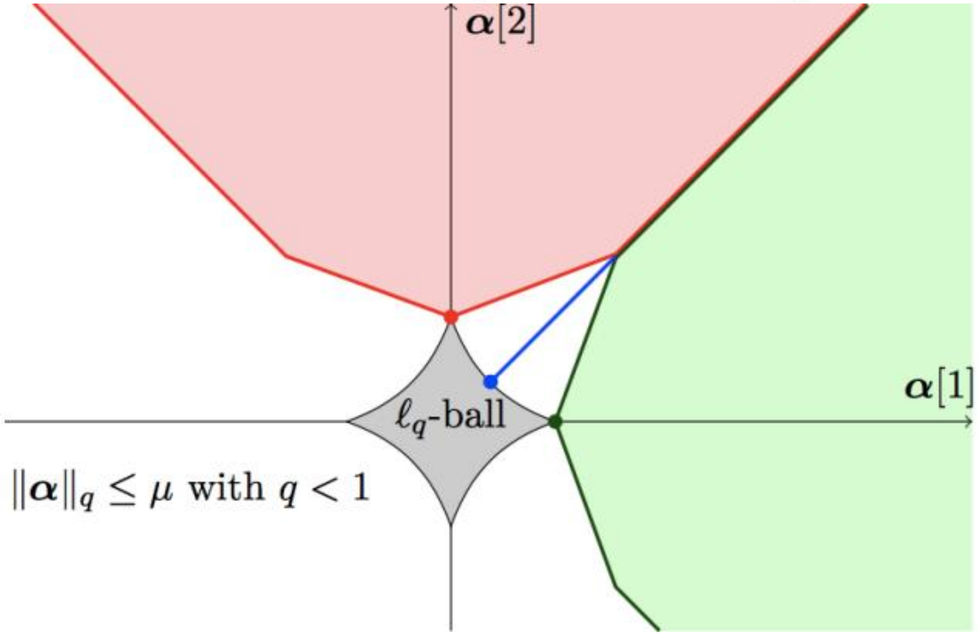


# Эффект разреженности

**если линии уровня оптимизируемой функции – концентрические окружности…**



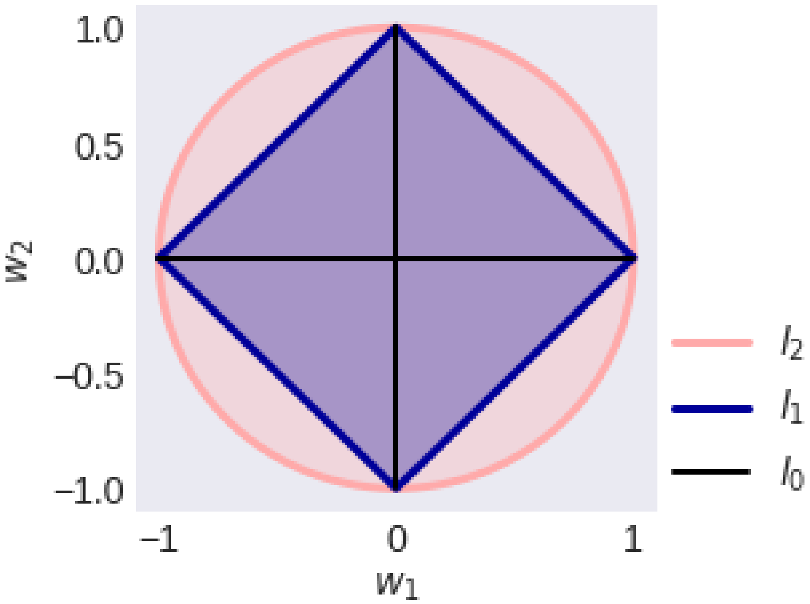




**David S. Rosenberg «Foundations of Machine Learning» https://bloomberg.github.io/foml/**

## Почему L1-норма ⇒ разреженность

**1. См. рис. больше вероятность, что линии уровней функции ошибки касаются области ограничений в точках с нулевыми координатами**

**2. L1-норма больше похожа на L0, чем L2**




**При увеличении коэффициента регуляризации веса стремятся к нулю**

**Обеспечивается автоматическая селекция признаков!**

## Регуляризация ⇒ упрощение

**Соблюдение принципа Оккама**

**регуляризация ⇒ зануление коэффициентов ⇒ упрощение модели**

**В целом, неверно, что чем меньше коэффициентов, тем проще модель, но у нас линейная модель..**

# Проблема вырожденности матрицы

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

**Решения:**

1. **Регуляризация**
2. **Селекция (отбор) признаков**
3. **Уменьшение размерности (в том числе, PCA)**
4. **Увеличение выборки**

## Селекция признаков в линейной регрессии

**~ отдельная тема**

**Какие признаки включить в модель:**

$$a(X_1,\ldots,X_n)=w_0+w_1X_1+\ldots+w_nX_n$$

**пока маленький обзор стратегий:**

**1 стратегия – умный перебор подмножества признаков**

**2 стратегий – оценка качества признаков (фильтры)**

**3 стратегия – встроенные методы (ex: LASSO)**

## Обоснование необходимости селекции

**1. Проблема вырожденности в линейной регрессии**

**2. Проблема «почти дубликатов»**

**3. Уменьшение модели и интерпретация**

**4. Уменьшение стоимости данных**

# Проблема вырожденности матрицы

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

**Решения:**

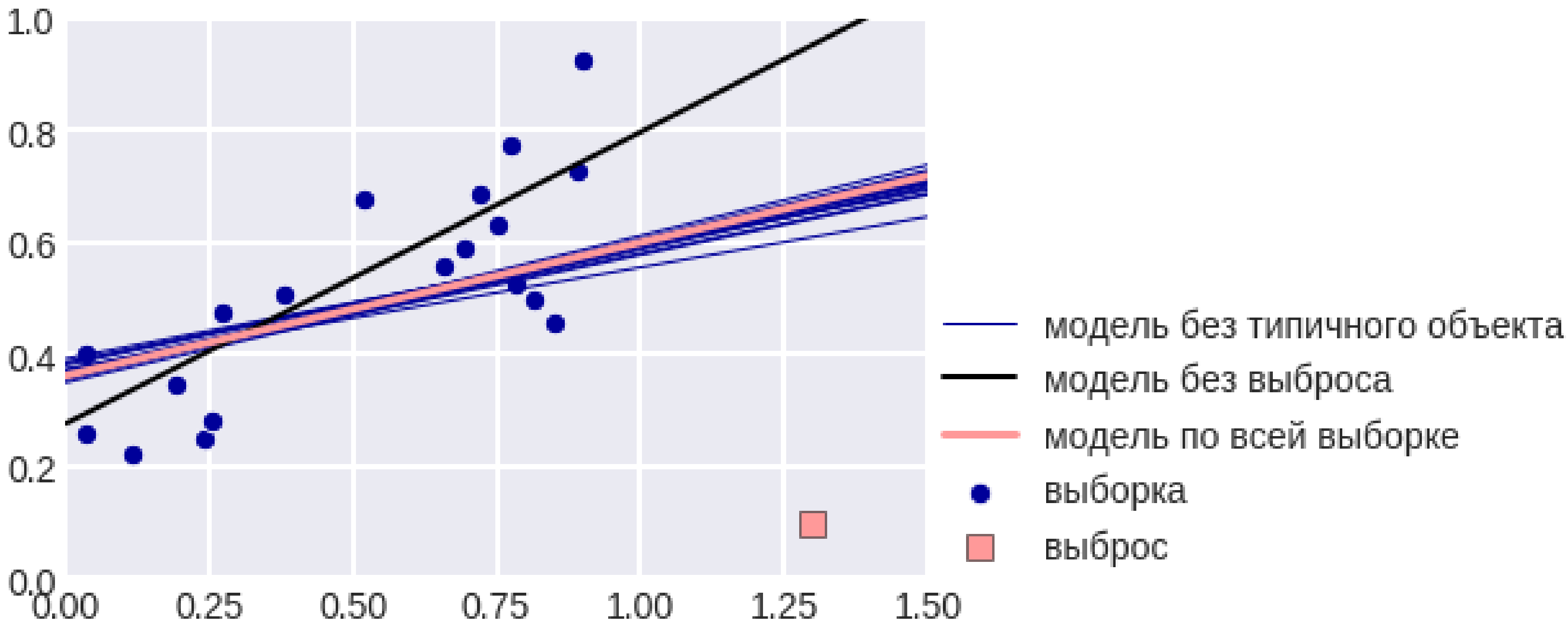
1. Регуляризация
2. Селекция (отбор) признаков
3. Уменьшение размерности (в том числе, PCA)
4. Увеличение выборки

| | x1 | x2 | x3 | y |
|---|---|---|---|---|
| **0** | 0.44 | 0.62 | 0.51 | -0.25 |
| **1** | 0.03 | 0.53 | 0.07 | -0.51 |
| **2** | 0.55 | 0.13 | 0.43 | 0.41 |
| **3** | 0.44 | 0.51 | 0.10 | 0.04 |
| **4** | 0.42 | 0.18 | 0.13 | 0.12 |
| **5** | 0.33 | 0.79 | 0.60 | -0.45 |

| | x1-x2 | y |
|---|---|---|
| **0** | -0.18 | -0.25 |
| **1** | -0.50 | -0.51 |
| **2** | 0.42 | 0.41 |
| **3** | -0.07 | 0.04 |
| **4** | 0.24 | 0.12 |
| **5** | -0.46 | -0.45 |

обоснование необходимости аналогично селекции

# Линейная регрессия – неустойчивость к выбросам



## Ошибка с весами

**Если у каждого объекта есть цена ошибки...**

$$\sum_{i=1}^{m} v_i \left(y_i - w^{\mathrm{T}} x_i\right)^2 + \ldots = \sum_{i=1}^{m} \left(\sqrt{v_i}\, y_i - w^{\mathrm{T}} (\sqrt{v_i}\, x_i)\right)^2 + \ldots \to \min$$

**небольшая переформулировка задачи:**

$$\{(x_1, y_1), \ldots, (x_m, y_m)\} \to \{(\sqrt{v_1}\, x_1, \sqrt{v_1}\, y_1), \ldots, (\sqrt{v_m}\, x_m, \sqrt{v_m}\, y_m)\}$$

**если веса целые числа – можно продублировать объекты**

**если веса из отрезка [0, 1] – при численном градиентном решении можно выбирать следующий объект с соответствующей вероятностью**

# Устойчивая регрессия (Robust Regression)

**0. Инициализация весов объектов**

$$v = (v_1, \ldots, v_m) = (1/m, \ldots, 1/m)$$

**1. Цикл**

**1.1. Настроить алгоритм, учитывая веса объектов**

$$a = \mathrm{fit}(\{x_i, y_i, v_i\})$$

можно использовать любую регрессионную модель

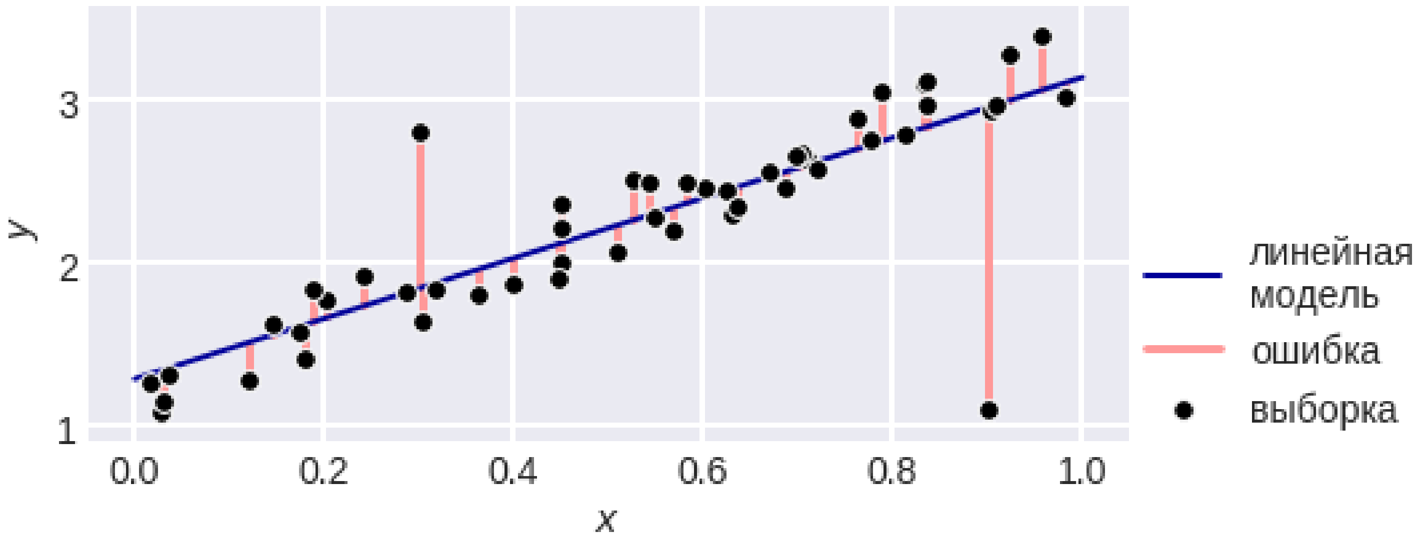
**1.2. Вычислить ошибки на обучении**

$$\varepsilon_i = a(x_i) - y_i$$

**1.3. Пересчитать веса объектов**

$$v_i = \exp(\varepsilon_i)$$

можно использовать другую невозрастающую функцию; можно (иногда нужно) нормировать



## Линейные скоринговые модели в задаче бинарной классификации

**Пусть** $X = \mathbb{R}^n, Y = \{0, 1\}$

**Как решать задачи классификации с помощью линейной модели: будем получать вероятность принадлежности к классу 1**

$$a(x) \in [0, 1]$$

**Любая линейная функция на $\mathbb{R}^n$ будет получать значения в $\mathbb{R}$, поэтому нужна деформация (transfer function):**

$$\sigma : \mathbb{R} \to [0, 1]$$

**В логистической регрессии**

$$\sigma(z) = \frac{1}{1 + e^{-z}}$$

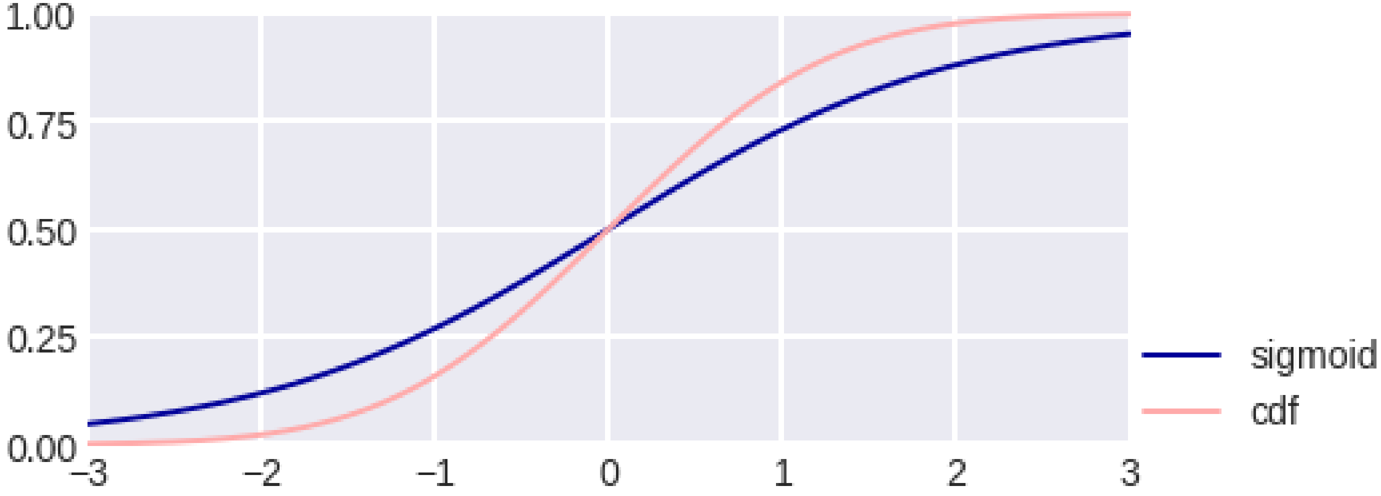
**Логистическая функция (сигмоида)**

**В Probit-регрессии**

$$\Phi(z) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{z} \exp(-t^2 / 2) \partial t$$

**Normal Cumulative distribution function**

# Функции деформации



## Логистическая регрессия

$$p(x) \equiv P(Y=1\,|\,x) = \sigma(z) = \frac{1}{1+e^{-z}} \in (0,1),$$

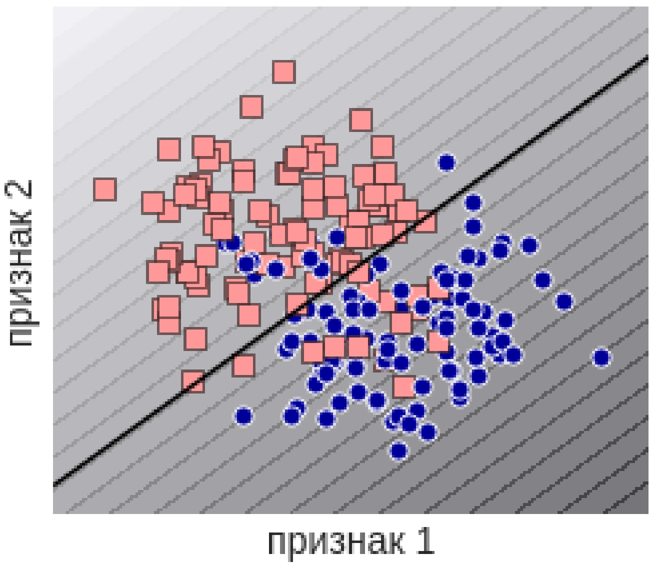
$$z = w_0 + w_1 X_1 + \ldots + w_n X_n,$$

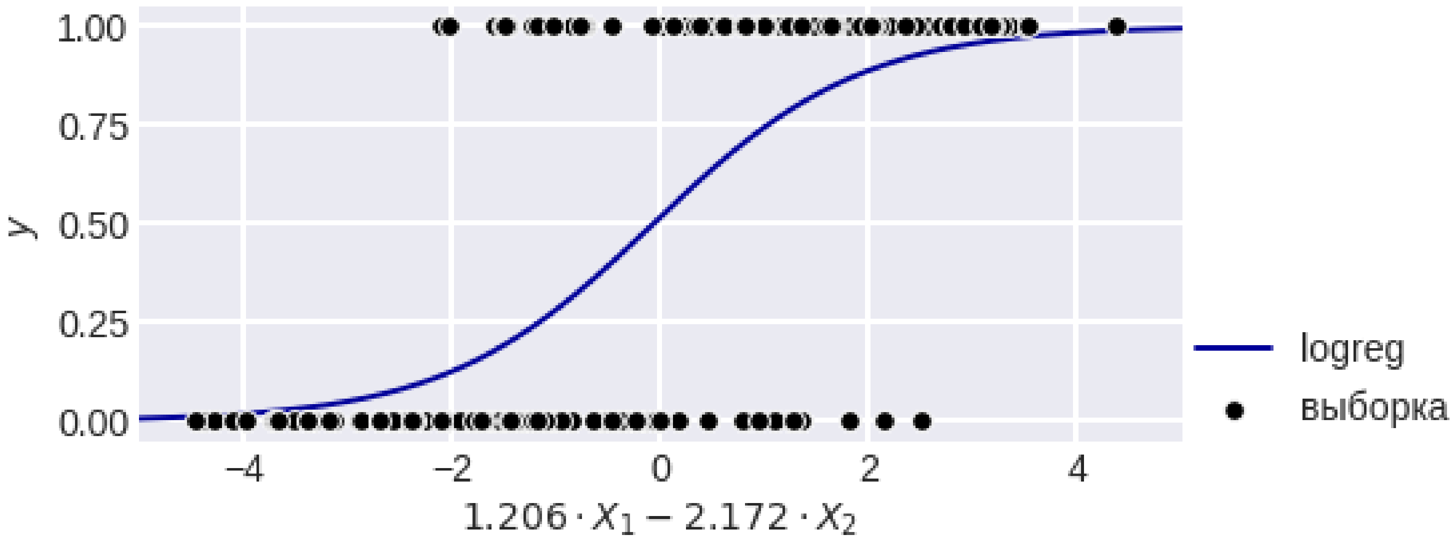
$$\log\left(\frac{p(x)}{1-p(x)}\right) = z$$

**– монотонное преобразование, которое называют logit-transformation**

# Геометрический смысл логистической регрессии

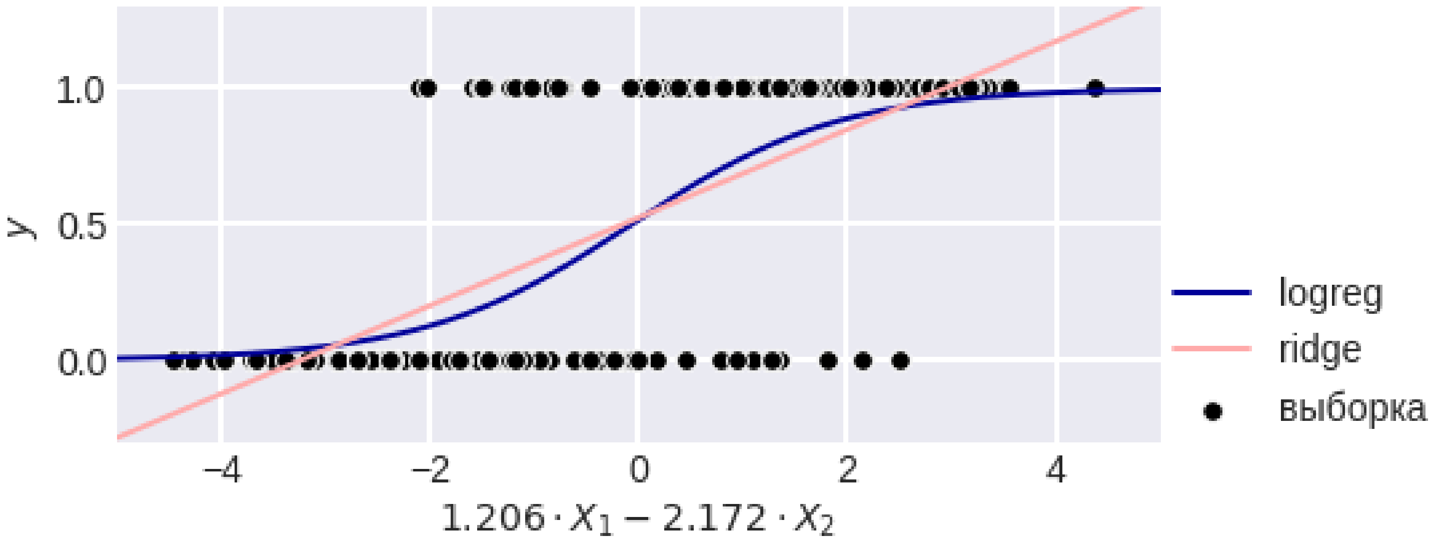
```
from sklearn.linear_model import LogisticRegression
model = LogisticRegression()
model.fit(X, y)
a = model.predict_proba(X_test)[:,1]
```




**Можно и одномерную картинку**

## Чем логистическая регрессия лучше регрессии



## Обучение логистической регрессии

**Метод максимального правдоподобия**

$$L(w_0,\ldots,w_n) = \prod_{i:y_i=1} p(x_i) \prod_{i:y_i=0} (1 - p(x_i)) \to \max$$

$$\log L = -\sum_{i:y_i=1} \log(1+e^{-z_i}) - \sum_{i:y_i=0} \log(1+e^{+z_i}) \equiv -\sum_{i} \log(1+e^{-y_i' z_i})$$

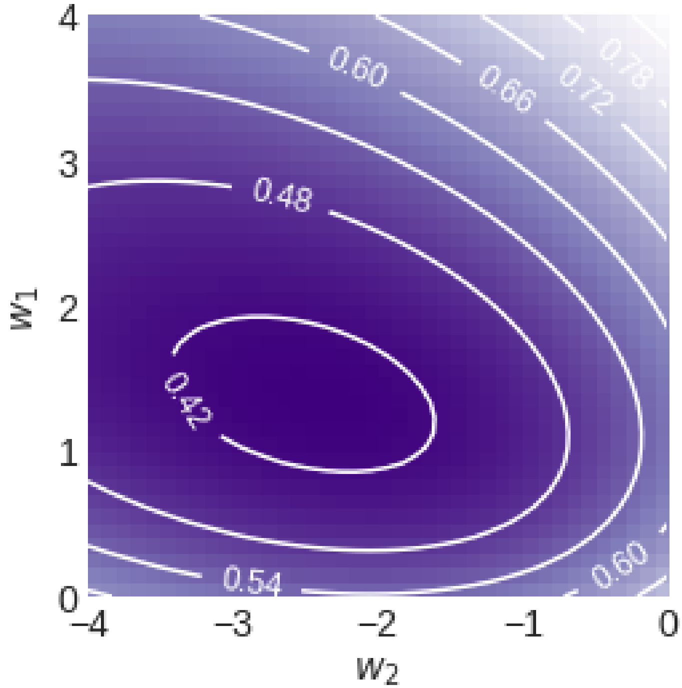
$$\nabla_w \log L = \sum_{i:y_i=1} \frac{1}{1+e^{-w^{\mathrm{T}} x_i}} e^{-w^{\mathrm{T}} x_i} x_i - \sum_{i:y_i=0} \frac{1}{1+e^{+w^{\mathrm{T}} x_i}} e^{+w^{\mathrm{T}} x_i} x_i =$$

$$= \sum_i \frac{y_i' x_i}{1+e^{+y_i' w^{\mathrm{T}} x_i}} = \sum_i y_i' x_i \sigma(-y_i' w^{\mathrm{T}} x_i)$$

**где (для удобства записи)**

$$y_i' = 2y_i - 1$$

## Ошибка логистической регрессии




**метод SGD**

$$w \leftarrow w + \eta\sigma(-y_i' w^{\mathrm{T}} x_i) y_i' x_i$$

**Запомним!**

## Многоклассовая логистическая регрессия
## Multiclass logistic regression (multinomial regression)

**в glmnet такой «симметричный вариант»**

$$P(Y=k\mid x)=\frac{e^{w_{0k}+w_{1k}X_1+\ldots+w_{nk}X_n}}{\sum_{j=1}^{l}e^{w_{0j}+w_{1j}X_1+\ldots+w_{nj}X_n}}$$

**Если**

$$\mathrm{softmax}(a_1,\ldots,a_l)=\frac{1}{Z}[e^{a_1},\ldots,e^{a_l}],$$

**где** $Z=e^{a_1}+\ldots+e^{a_l}$

**тогда**

$$P(Y=k\mid x)=\mathrm{softmax}(w(1)^{\mathrm{T}}x,\ldots,w(l)^{\mathrm{T}}x)$$

## Линейные решающие модели в задаче бинарной классификации

**Пусть** $X = \mathbb{R}^n, Y = \{\pm 1\}$

**обучающая выборка:** $\{(x_i, y_i')\}_{i=1}^m$

**хотим линейную модель:**

$$a(x) = \operatorname{sgn}(w^{\mathrm{T}}x + b) = \begin{cases} +1, & w^{\mathrm{T}}x + b > 0 \\ -1, & w^{\mathrm{T}}x + b < 0 \end{cases}$$

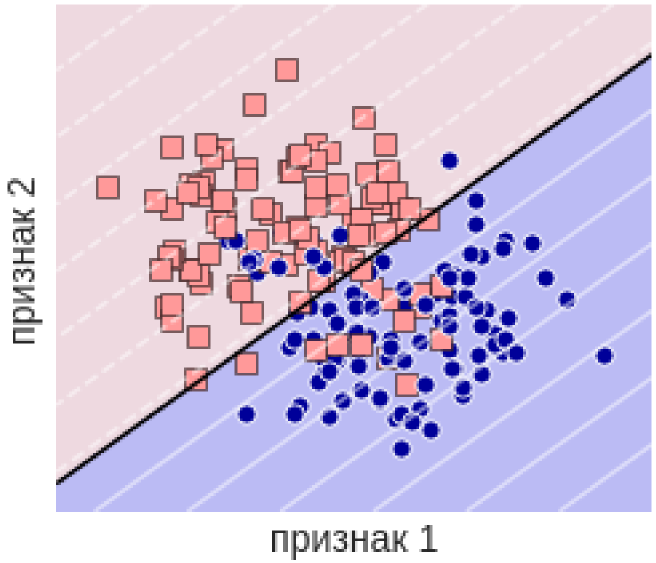
**случай** $w^{\mathrm{T}}x + b = 0$ **нам тут не особо важен**

**Пополняя признаковое пространство фиктивным признаком,**

$$a(x) = \operatorname{sgn}(w^{\mathrm{T}}x)$$

**– линейный классификатор**

# Геометрический смысл линейного классификатора




**Делим пространство гиперплоскостью на две части**

# Линейный классификатор

**Общая идея:**

$$L(y_t, a(x_t)) = \theta(-y_t' w^{\mathrm{T}} x_t) = \begin{cases} 1, & \operatorname{sgn} w^{\mathrm{T}} x_t = y_t' \\ 0, & \operatorname{sgn} w^{\mathrm{T}} x_t \neq y_t', \end{cases}$$

$$L(X_{\text{train}}, a) = \sum_{t=1}^{m} L(y_t, a(x_t)) \to \min$$

**естественно минимизировать число ошибок,**

**но**

- **ф-ия не дифференцируема**
- **выдаёт мало информации**
- **оптимизация здесь – NP-полная задача**

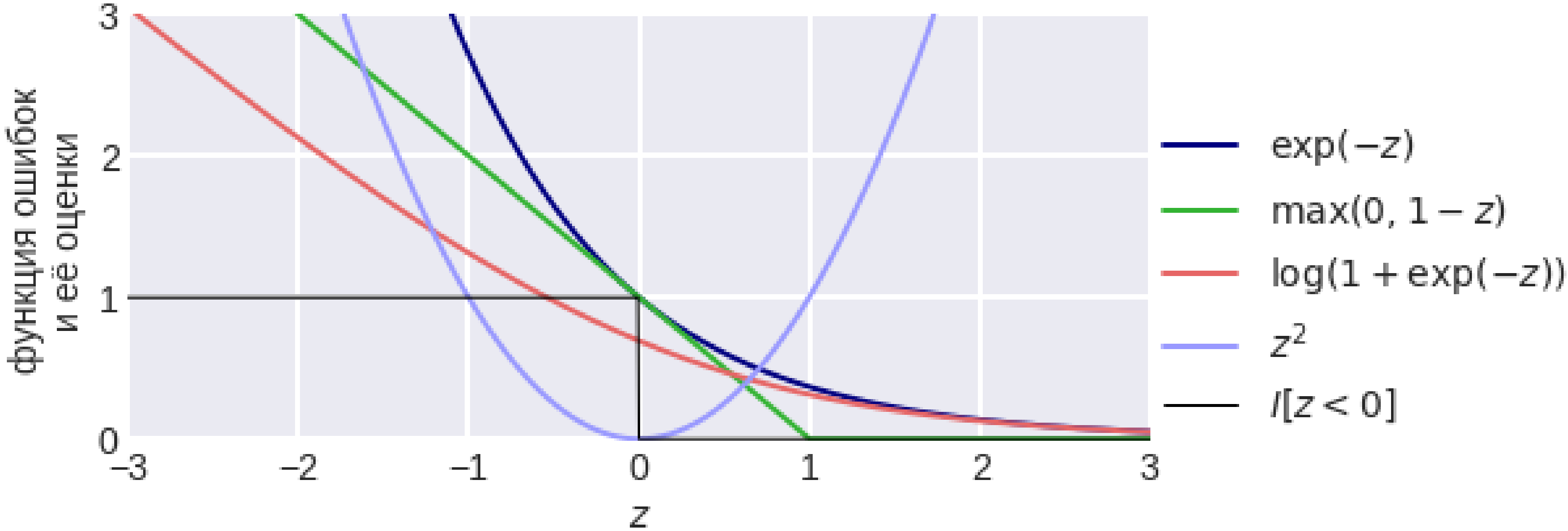
**только число ошибок, а не их «фатальность»**

$y_t' w^{\mathrm{T}} x_t$ **~ чем меньше, тем хуже (зазор)**

## Оценка функции ошибок через гладкую функцию

$$\sum_{t=1}^{m} L(y_t, a(x_t)) \leq \sum_{t=1}^{m} L'(y_t, a(x_t)) \to \min$$

$$\sum_{t=1}^{m} \theta(-\xi_t) \leq \sum_{t=1}^{m} f(-\xi_t) \to \min$$



## Оценка функции ошибок через гладкую функцию

**Примеры замен:**

$$f(-\xi) = \exp(-\xi)$$
$$f(-\xi) = \max(0, 1-\xi)$$
$$f(-\xi) = \log(1+\exp(-\xi))$$

**Обучение – минимизация оценки на обучающей выборке (+ регуляризация)**

**Персептрон, SVM, логистическая регрессия минимизируют выпуклые аппроксимации 0-1-loss, сводя NP-трудную задачу к задаче выпуклой оптимизации**

## Пример персептронного алгоритма

$$f(-\xi) = \max(0, -\xi)$$

**Персептрон:**

$$\sum_{i=1}^{m} \max[0, -y_i'(w^{\mathrm{T}} x_i)] \to \min$$

**SGD**

$$\frac{\partial \max[0, -y_i'(w^{\mathrm{T}} x_i)]}{\partial w} = \begin{cases} 0, & -y_i'(w^{\mathrm{T}} x_i) < 0, \\ -y_i' x_i & -y_i'(w^{\mathrm{T}} x_i) > 0, \end{cases} = \begin{cases} 0, & \mathrm{sgn}(w^{\mathrm{T}} x_i) = y_i', \\ -y_i' x_i & \mathrm{sgn}(w^{\mathrm{T}} x_i) \neq y_i', \end{cases}$$

$$w \leftarrow w + \eta \begin{cases} 0, & \mathrm{sgn}(w^{\mathrm{T}} x_i) = y_i', \\ +x_i & w^{\mathrm{T}} x_i \le 0, y_i' = +1, \\ -x_i, & w^{\mathrm{T}} x_i \ge 0, y_i' = -1, \end{cases}$$

## Пример персептронного алгоритма

**Есть теорема Новикова**

**Если две выборки линейно разделимы, то разделяющая поверхность находится персептронным алгоритмом за конечное число шагов**

# Пример персептронного алгоритма

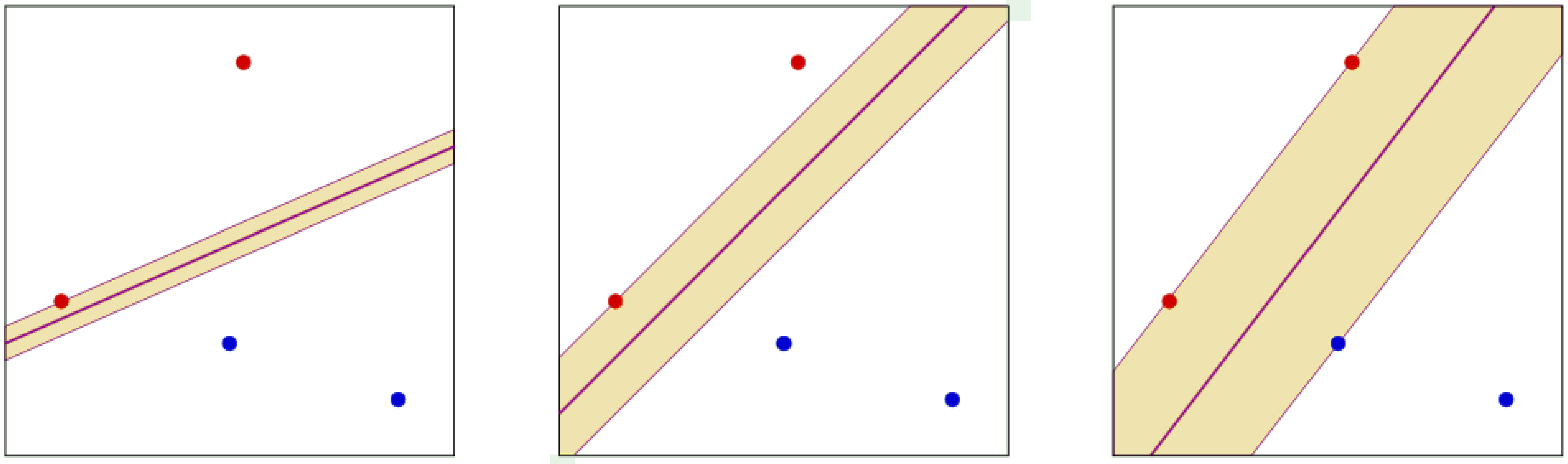
$$\begin{cases} 2w_1 + w_2 > 0 \\ -w_1 > 0 \\ w_1 - w_2 < 0 \\ -2w_1 - 2w_2 < 0 \end{cases}$$

$$\begin{bmatrix} 2 & 1 \\ -1 & 0 \\ -1 & 1 \\ 2 & 2 \end{bmatrix} \begin{pmatrix} w_1 \\ w_2 \end{pmatrix} > 0$$

```
X = list(np.array([[2,1], [-1, 0],
                   [-1, 1], [2, 2]]))
w = np.array([0, 0])
print ('w=', w)
change = True
while (change):
    change = False
    for x in X:
        if np.dot(x, w) <= 0:
            print ('w=', w, 'x=', x, '-')
            w += x
            change = True
print ('w=', w)
```

```
w=[ 0 0]
w=[ 0 0] x= [ 2 1] -
w=[ 2 1] x= [-1 0] -
w=[ 1 1] x= [-1 1] -
w=[ 0 2] x= [-1 0] -
w=[-1 2] x= [ 2 1] -
w=[ 1 3] x= [-1 0] -
w=[ 0 3] x= [-1 0] -
w=[-1 3]
```

## SVM: идея максимального зазора

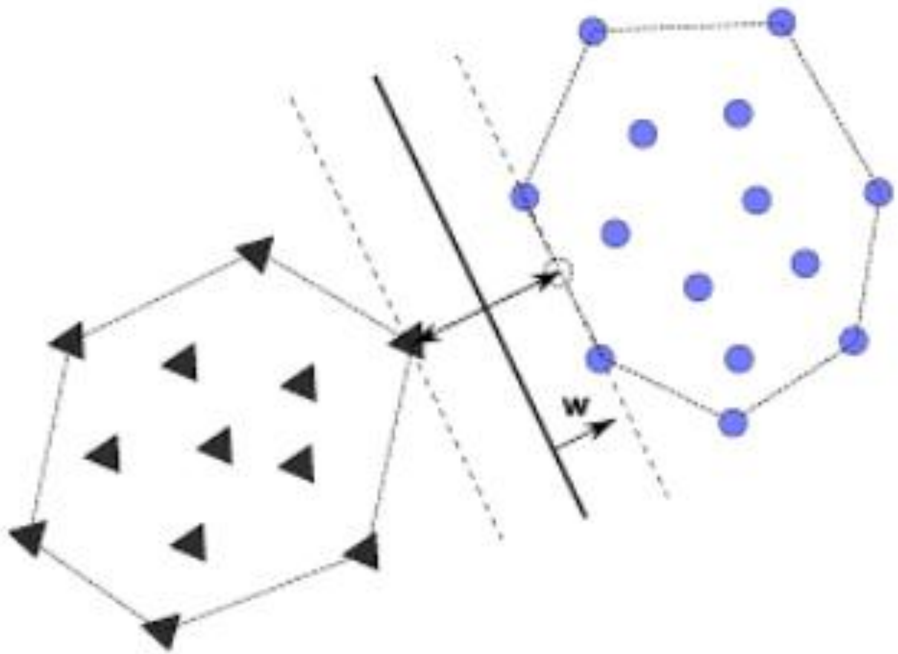


**до сих пор хотели разделить точки гиперплоскостью…**

**а как лучше?**

## SVM: идея максимального зазора

**Построение SVM эквивалентно нахождению кратчайшего отрезка, соединяющего выпуклые оболочки двух классов**



## SVM: постановка задачи

**Хотим разделить точки двух разных классов гиперплоскостью**

$$a(x) = \mathrm{sgn}(w^{\mathrm{T}} x + b)$$

**Обучающая выборка:**

$$\{(x_i, y_i)\}_{i=1}^{m}$$

**должно быть** **(здесь пускай нет штрихов)**

$$w^{\mathrm{T}} x_i + b \geq 1 \text{ **если** } y_i = +1$$

$$w^{\mathrm{T}} x_i + b \leq 1 \text{ **если** } y_i = -1$$

**Другая форма записи:**

$$y_i (w^{\mathrm{T}} x_i + b) \geq 1$$

**можно считать (из-за нормировки), что**

$$\min_i | w^{\mathrm{T}} x_i + b | = 1$$

## SVM: постановка задачи

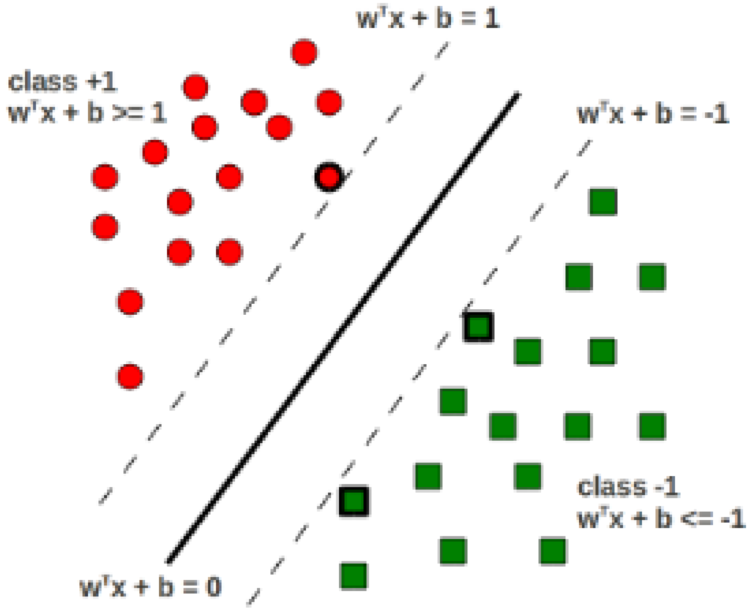
**Расстояние от точки до гиперплоскости:**

$$\rho(x_i, w^{\mathrm{T}}x + b) = \frac{|\, w^{\mathrm{T}} x_i + b \,|}{\|\, w \,\|}$$

**хотим, чтобы минимум из этих расстояний был максимален**

$$\min_i \frac{|\, w^{\mathrm{T}} x_i + b \,|}{\|\, w \,\|} = \frac{1}{\|\, w \,\|} \to \max$$

**– зазор (margin)**



## Зазор (margin)

**В общем случае, когда**

**Пусть** $X = \mathbb{R}^n, Y = \{\pm 1\}$

**обучающая выборка:** $\{(x_i, y_i)\}_{i=1}^m$

**Алгоритм со скоринговой функцией (score function)**

$$a(x) = \mathrm{sgn}(b(x)), b(x) \in \mathbb{R}$$

**Зазор –** $y_i b(x_i)$

**~ уверенность в ответе**

## SVM: постановка задачи

$$\frac{\|w\|^2}{2} \to \min$$

$$y_i(w^{\mathrm{T}}x_i + b) \geq 1,\ i \in \{1, 2, \ldots, m\}$$

**– задача квадратичного программирования**
**(QP = Quadratic Program)**
**с m ограничениями (constraints)**

**Заметим, что здесь тоже, как и в регуляризации линейной регрессии, хотим квадрат нормы весов сделать меньше**

**«квадрат» – для удобства оптимизации**

## SVM: приближённое решение «в лоб»

$$\frac{\|w\|^2}{2} \to \min$$

$$1 - y_i(w^{\mathrm{T}} x_i + b) \le 0,\ i \in \{1, 2, \ldots, m\}$$

**не решаем задачу точно, но стремимся выполнить условия:**

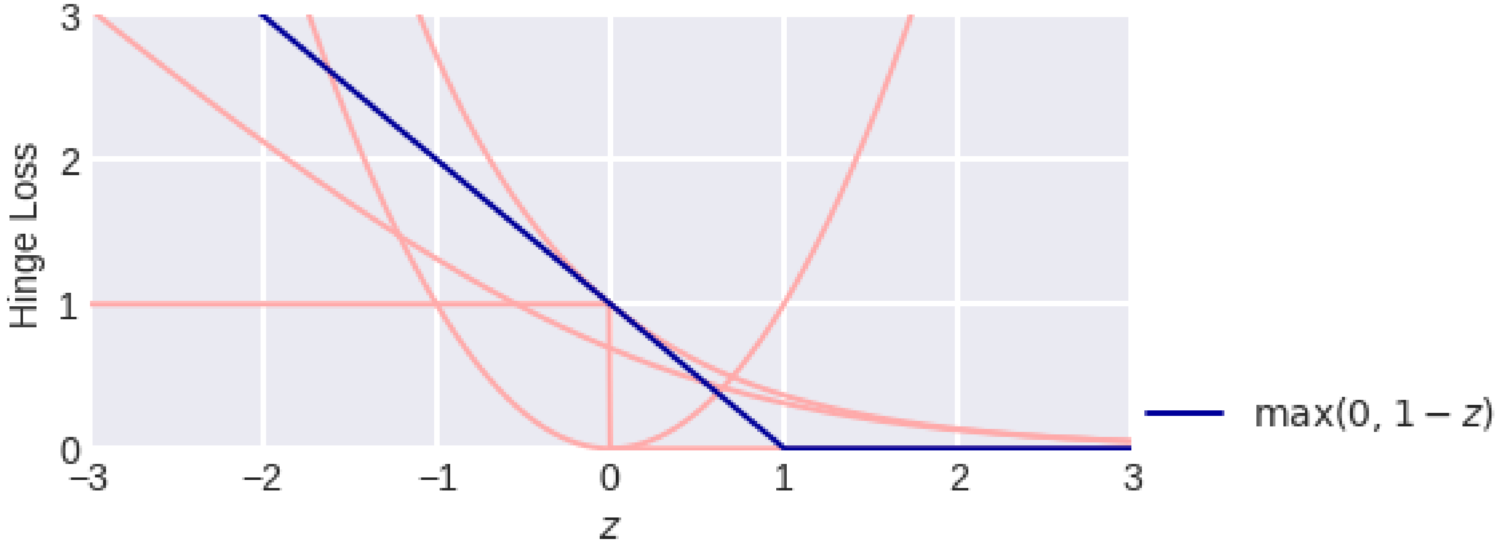
$$\frac{1}{m}\sum_{i=1}^{m} \max[0, 1 - y_i(w^{\mathrm{T}} x_i + b)] + \lambda \|w\|^2 \to \min$$

**Удивительно:**

**ошибка** $L(y, a) = \max[0, 1 - ya]$ **+ регуляризатор**

**но тут нет дифференцируемости из-за max**

## Hinge loss




**А в логистической регрессии:**
**логистическая функция ошибки + регуляризатор**

## SVM: решение строгое

$$\frac{\|w\|^2}{2} \to \min$$

$$1 - y_i(w^{\mathrm{T}} x_i + b) \leq 0,\ i \in \{1, 2, \ldots, m\}$$

**Вспоминаем оптимизацию с ограничениями:**

$$\min_{w,b} \max_{\alpha \geq 0} L(w, b, \alpha) = \frac{w^{\mathrm{T}} w}{2} + \sum_{i=1}^{m} \alpha_i (1 - y_i (w^{\mathrm{T}} x_i + b))$$

**тут будет дифференцируемость, но ограничения**

## SVM: решение строгое

$$\min_{w,b}\max_{\alpha\geq 0} L(w,b,\alpha)=\frac{w^{\mathrm{T}}w}{2}+\sum_{i=1}^{m}\alpha_i(1-y_i(w^{\mathrm{T}}x_i+b))$$

**возьмём производные, приравняем к нулю**

$$\frac{\partial L}{\partial w}=0\Rightarrow w=\sum_{i=1}^{m}\alpha_i y_i x_i$$

$$\frac{\partial L}{\partial b}=0\Rightarrow \sum_{i=1}^{m}\alpha_i y_i=0$$

**Таким образом, оптимальный вектор весов – взвешенная сумма признаковых описаний объектов из обучения**

**если подумать – аналогично происходит и при настройке персептрона и логистической регресии…**

## Переход к двойственной задаче

### Задача квадратичного программирования

$$\max_{\alpha\geq 0}\min_{w,b} L(w,b,\alpha)=\sum_{i=1}^{m}\alpha_i-\frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m}\alpha_i\alpha_j y_i y_j x_i^{\mathrm{T}} x_j$$

**при условиях** $\sum_{i=1}^{m}\alpha_i y_i=0$

**Информацию об описаниях объектов мы используем лишь в виде их попарных скалярных произведений!**

**когда решим задачу…**

$$w=\sum_{i=1}^{m}\alpha_i y_i x_i$$

$$b=-\frac{1}{2}\left(\min_{i:\,y_i=+1} w^{\mathrm{T}}x_i+\max_{i:\,y_i=-1} w^{\mathrm{T}}x_i\right)$$

**большинство** $\alpha_i$ **обратиться в ноль (из-за условий Кунна-Таккера)**

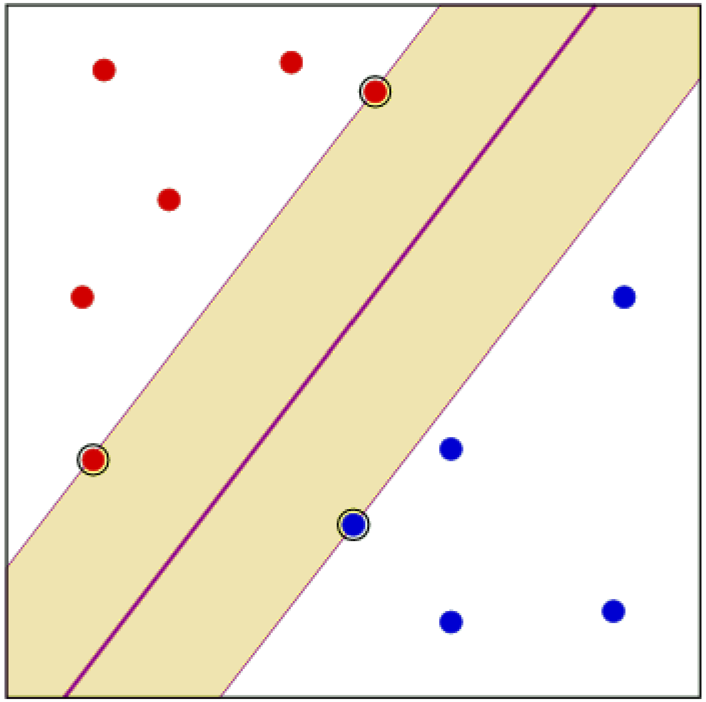
# Условия Кунна-Таккера

**для решения:**

$$\alpha_i(1 - y_i(w^{\mathrm{T}} x_i + b)) = 0$$

**если** $\alpha_i > 0$**, то** $x_i$ **– опорный вектор (support vector)**

**– лежит на границе** $y_i(w^{\mathrm{T}} x_i + b) = 1$



## Зачем переходить к двойственной задаче

- **размерности**

м.б. удобно решать

**выгодно, если признаковое пространство большое**

- **известная задача**

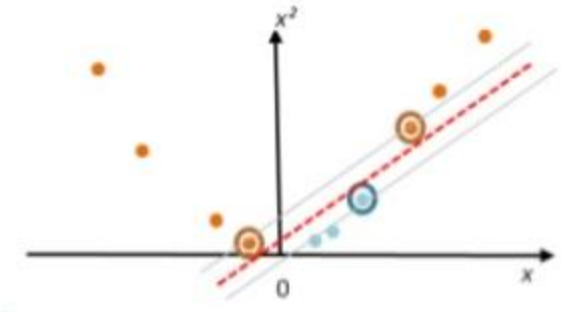
можно использовать солверы из готовых библиотек

- **возникли попарные произведения**

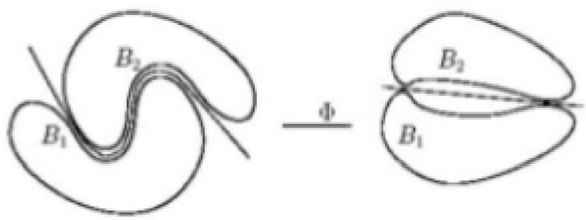
потом используем для kernel tricks

## Если нет линейной разделимости

**Два подхода (часто используются вместе)**




**1) разделять так, чтобы ошибок было мало**

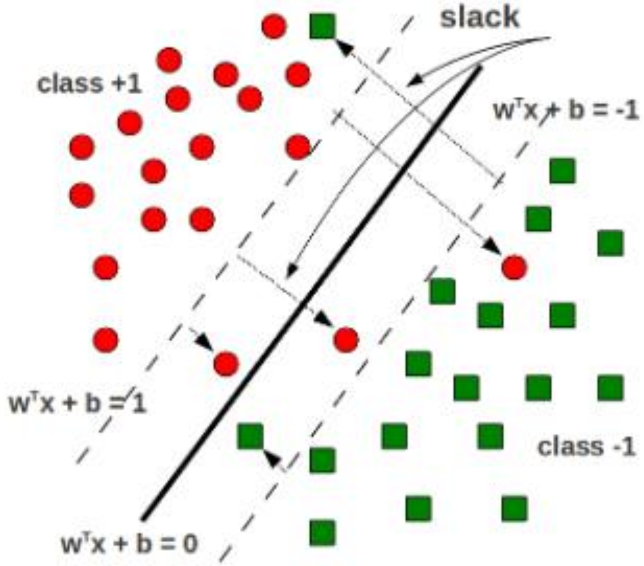



**2) использование нелинейных разделяющих поверхностей**

**переход в другое признаковое пространство**

**потом подробно разберём!**

## Soft-Margin SVM: разделение допуская ошибки




**позволить объектам «залезать» на половину другого класса**

$$y_i(w^{\mathrm{T}}x_i + b) \geq 1 - \xi_i$$

$$\xi_i \geq 0$$

**но не хотим, чтобы было много больших залезаний**

## Soft-Margin SVM: разделение допуская ошибки

**Прямая задача:**

$$\frac{\|w\|^2}{2} + C\sum_{i=1}^{m}\xi_i \to \min$$

$$y_i(w^{\mathrm{T}}x_i + b) \ge 1 - \xi_i,\ \xi_i \ge 0,\ i \in \{1,2,\ldots,m\}$$

**тоже задача квадратичного программирования,**

**но в два раза больше ограничений**

**C – баланс между оптимизацией зазора и ошибки на обучении**

**Soft-Margin SVM решается аналогично… ДЗ вывести!**

**Двойственная задача:**

$$\sum_{i=1}^{m}\alpha_i - \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m}\alpha_i\alpha_j y_i y_j x_i^{\mathrm{T}}x_j \to \max_{0\le\alpha\le C}$$

**появляется лишь ограничение** $\alpha \le C$

**одно нетривиальное ограничение** $\sum_{i=1}^{m}\alpha_i y_i = 0$

# SVM Regression

**хотим решить с $\varepsilon$-точностью**

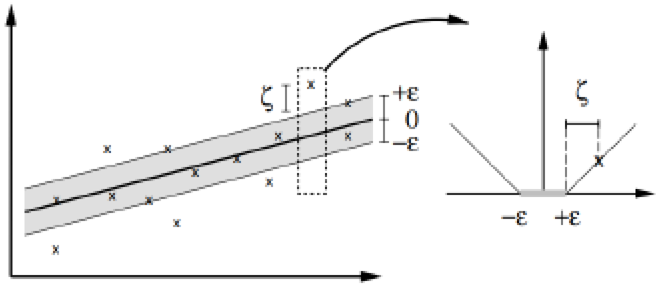
$$\frac{\|w\|^2}{2} \to \min$$

$$|w^{\mathrm{T}} x_i + b - y_i| \le \varepsilon,\ i \in \{1, 2, \ldots, m\}$$

Equivalent unconstrained formulation:

$$\frac{1}{2}\|w\|^2 + C\sum_{n=1}^{N} \mathcal{L}(\langle w, x_n\rangle + w_0 - y_n) \to \min_{w}$$

with $\varepsilon$ insensitive loss $\mathcal{L}(u) = \begin{cases} 0, & \text{if } |u| \le \varepsilon \\ |u| - \varepsilon & \text{otherwise} \end{cases}$




Solution will depend only on objects with $|\text{error}| \ge \varepsilon$, called *support vectors.*

# SVM

• должно быть хорошее пространство
(однородные признаки в одной шкале)

• тогда работают линейные SVM
(нелинейные – с ядрами – успешно заменяются другими алгоритмами)

• не подходят для больших данных
(особенно нелинейные)

• требуют хранения опорных векторов

## Проблемы с линейными алгоритмами

+ простой, надёжный, быстрый, популярный метод
+ интерпретируемость (⇒ нахождение закономерностей)
+ интерполяция и экстраполяция
+ может быть добавлена нелинейность, с помощью генерации новых признаков (дальше – это можно автоматизировать)

– линейная гипотеза вряд ли верна
– в теоретическом обосновании ещё предполагается нормальность ошибок
– «страдает» из-за выбросов
– признаки в одной шкале и однородные
– статистический вывод регрессии – много предположений
– проблема коррелированных признаков
⇒ необходимость регуляризации, селекции, PCA, data↑

## Проблемы мультиколлинеарности

- большие коэффициенты
- большие изменения коэффициентов при добавлении/удалении признаков
- нелогичности
  (чем больше доход, меньше вероятность дать кредит)
- большое число статистически незначимых оценок коэффициентов

## зависимость от масштабирования

| простая модель | с регуляризацией | пайплайн: нормировка + линейная |
| --- | --- | --- |
| нет | есть | нет |

## Зачем нужен дискриминантный анализ?

**Когда классы хорошо разделимы оцениваемые параметры, скажем, для логистической регрессии могу быть нестабильны.**

**Линейный дискриминантный анализ меньше подвержен этой проблеме.**

**Если размерность малая, распределения нормальные – линейная дискриминантная модель опять лучше.**

**Также LDA можно приспособить для представления данных в маломерных пространствах.**

**LDA – для малых размерностей и нормально распределенных данных**
**Наивный Байес – для больших размерностей**

## Линейный дискриминант Фишера

**рассмотрим случай двух классов:**

$$X = \mathbb{R}^n,\ Y = \{\pm 1\}$$

$$X_\alpha = \{x_i \mid y_i = \alpha\}$$

$$m_\alpha = | X_\alpha |$$

$$m_{+1} + m_{-1} = m$$

$$\mu_\alpha = \frac{1}{m_\alpha} \sum_{x_i \in X_\alpha} x_i$$

$$\sigma_\alpha^2 = \sum_{x_i \in X_\alpha} (x_i - m_\alpha)^2$$

**хотим, чтобы проекции на некоторую прямую**

$$\frac{(\mu_{+1} - \mu_{-1}) |_w^2}{\sigma_{+1}^2 |_w + \sigma_{-1}^2 |_w} \to \max$$

## Линейный дискриминант Фишера

$$(\mu_{+1} - \mu_{-1})\,|_w^{\,2} = (w^{\mathrm{T}}\mu_{+1} - w^{\mathrm{T}}\mu_{-1})^2 = (w^{\mathrm{T}}(\mu_{+1} - \mu_{-1}))^2 =$$
$$= w^{\mathrm{T}}(\mu_{+1} - \mu_{-1})(\mu_{+1} - \mu_{-1})^{\mathrm{T}} w = w^{\mathrm{T}} S w$$

$$\sigma_\alpha^2\,|_w = \sum_{x_i \in X_\alpha} (w^{\mathrm{T}} x_i - w^{\mathrm{T}}\mu_\alpha)^2 = w^{\mathrm{T}} \sum_{x_i \in X_\alpha} (x_i - \mu_\alpha)(x_i - \mu_\alpha)^{\mathrm{T}} w =$$
$$= w^{\mathrm{T}} S_\alpha w$$

$$\frac{w^{\mathrm{T}} S w}{w^{\mathrm{T}}(S_{+1} + S_{-1}) w} \to \max$$

междуклассовый разброс
внутриклассовый разбор

## Линейный дискриминант Фишера

**решение** $w \propto (S_{+1} + S_{-1})^{-1}(\mu_2 - \mu_1)$

**обосновать!**

**интересный факт:**
**можно получить из МНК, выбрав целевые значения**

$$\alpha \to \frac{m_{+1} + m_{-1}}{m_{\alpha}}$$

**обосновать!**