«Машинное обучение»

Линейные методы (часть 1): линейная регрессии

Александр Дьяконов

28 сентября 2021 года

План

Линейная регрессия

Решение проблемы вырожденности

Регуляризация, гребневая регрессия, LASSO, Elastic Net

Устойчивая регрессия

Градиентный метод обучения

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:

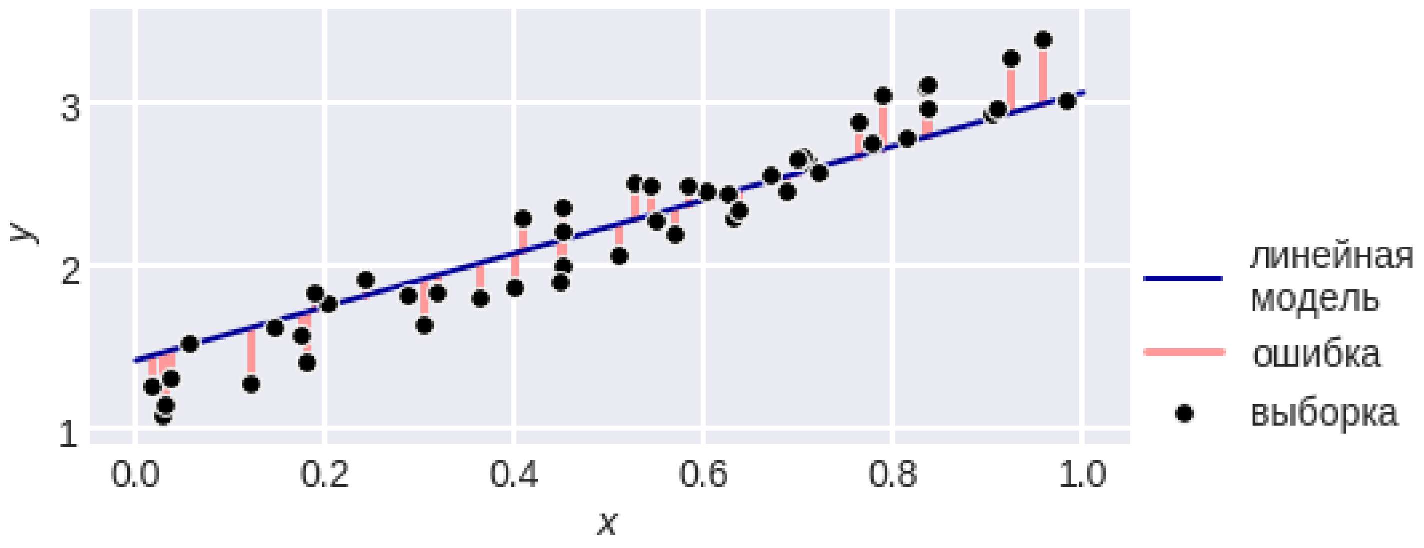
- **часто неплохо работает и при монотонных зависимостях**
- **хорошо работает, когда есть много «однородных» зависимостей:**

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

$$a(X_1) = w_0 + w_1 X_1$$

обучение: $\{(x_1, y_1), \ldots, (x_m, y_m)\},\ x_i \in \mathbb{R}$



хотели бы...

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

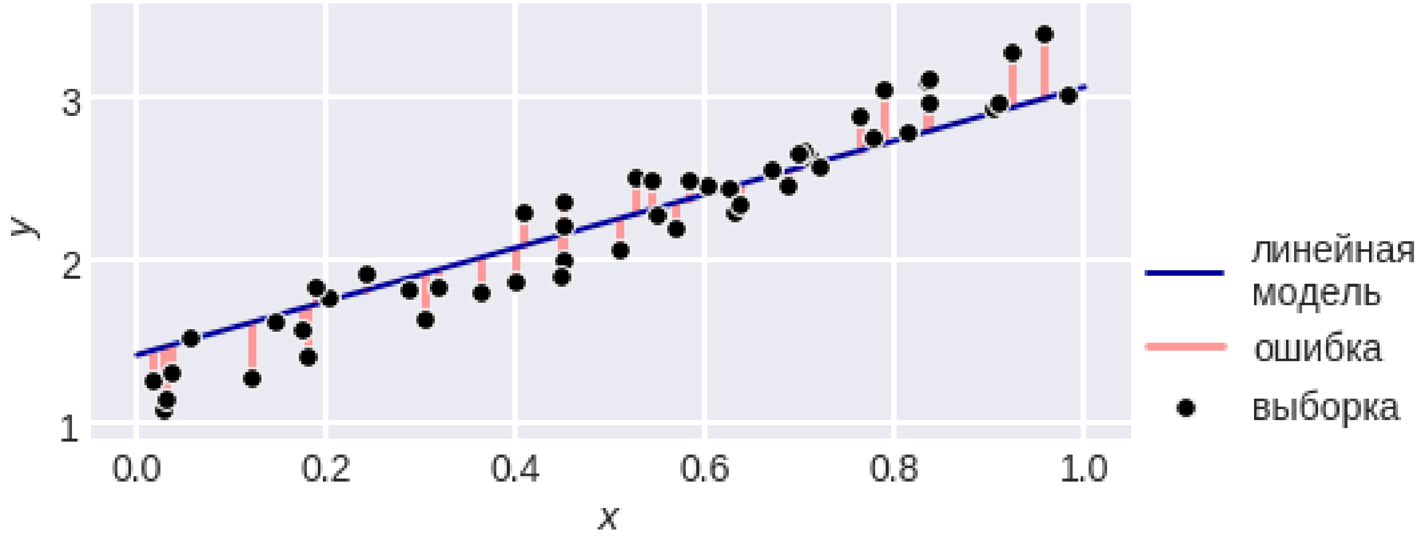
невязки / отклонения (residuals):

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

Линейная регрессия от одной переменной

Задача минимизации суммы квадратов отклонений (residual sum of squares)

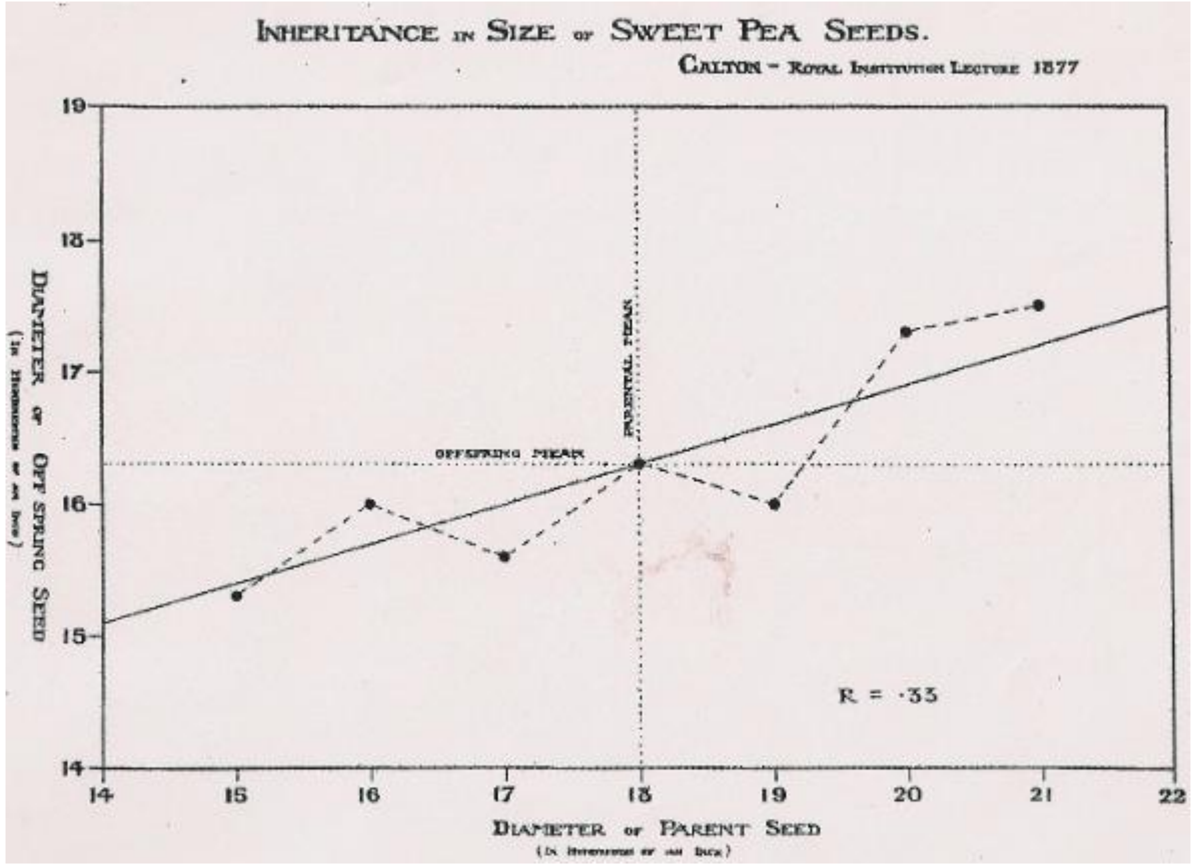
$$\mathrm{RSS} = e_1^2 + \ldots + e_m^2 \to \min$$



~ задача описания данных гиперплоскостью (но тут конкретная ф-я ошибки)

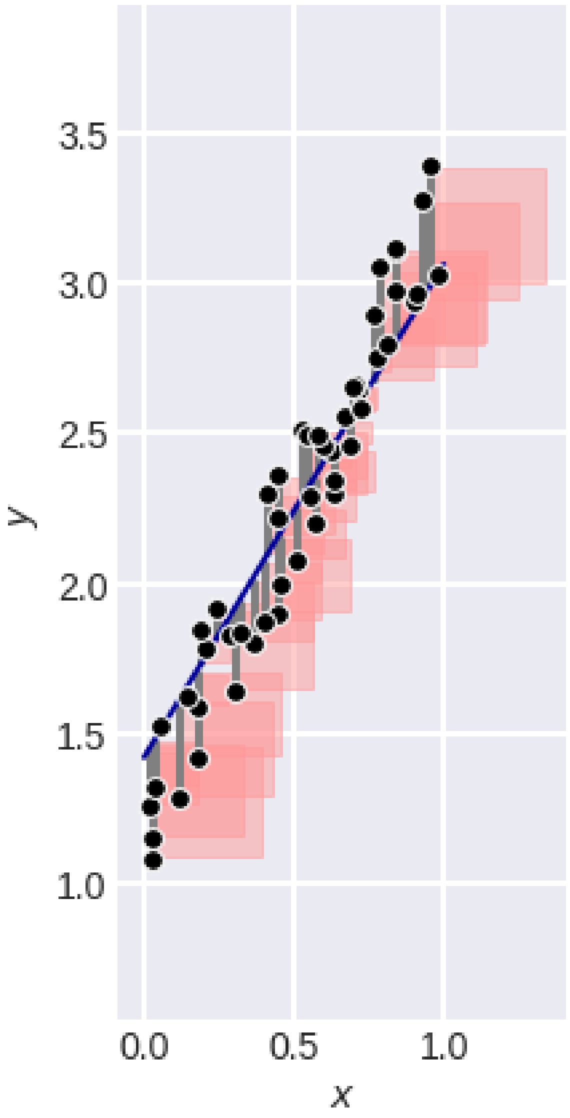
потом вероятностное обоснование, пока… довольно логично

Линейная регрессия от одной переменной

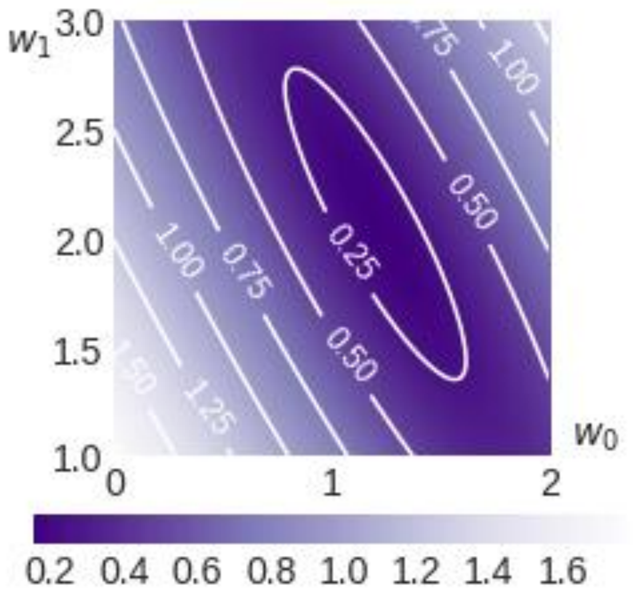


Francis Galton, 1877

Линейная регрессия от одной переменной: геометрический смысл ошибки



$$a(X_1) = w_0 + w_1 X_1$$



$$\sum_{i=1}^{m} (y_i - w_0 + w_1 x_i)^2$$

Отличается от суммы расстояний до поверхности!

Линейная регрессия от одной переменной

Нетрудно показать (ДЗ):

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

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

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

Общий случай (многих переменных)

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

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

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

для удобства записи вводим фиктивный признак $X_0 \equiv 1$

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

опять хотим решить $Xw = y$**:**

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

как решать?

Общий случай (многих переменных): в матричной форме

$$Xw = y$$

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

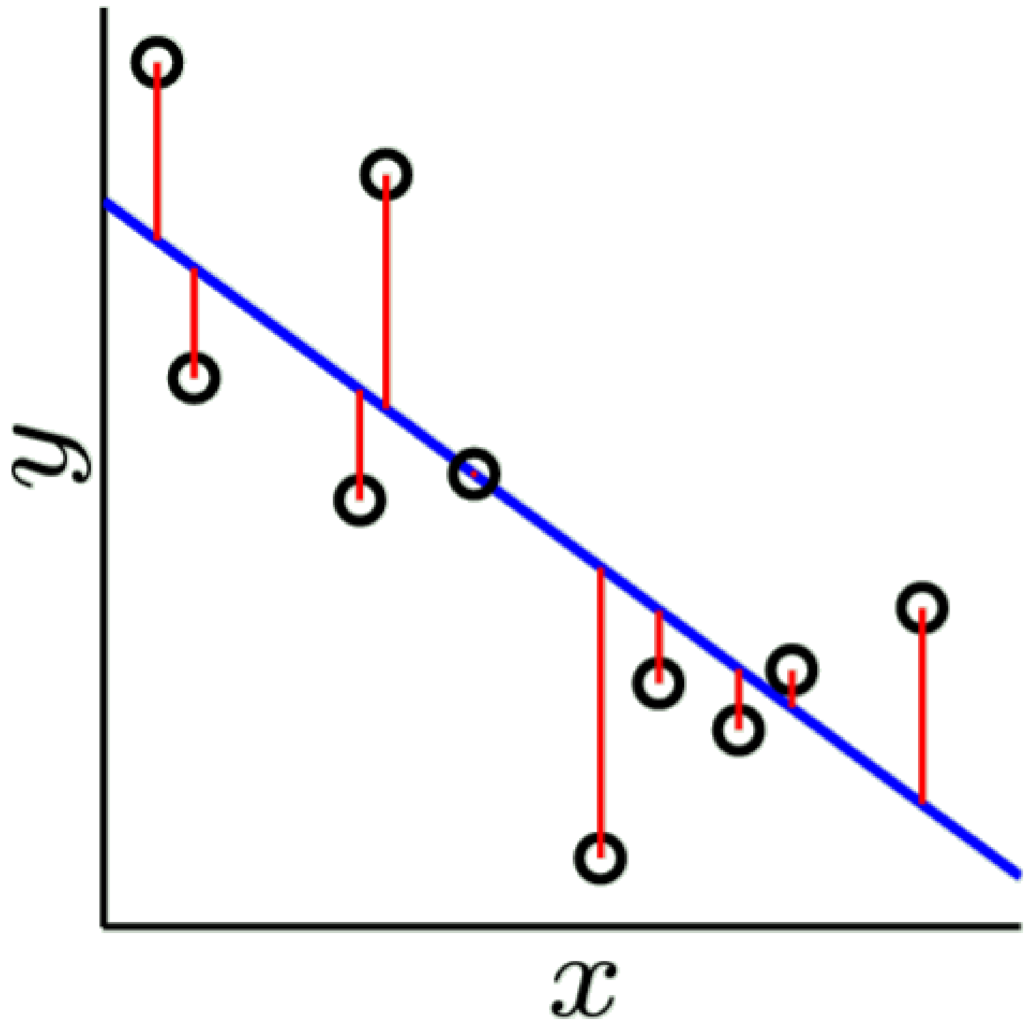
(здесь есть коллизия в обозначении y)

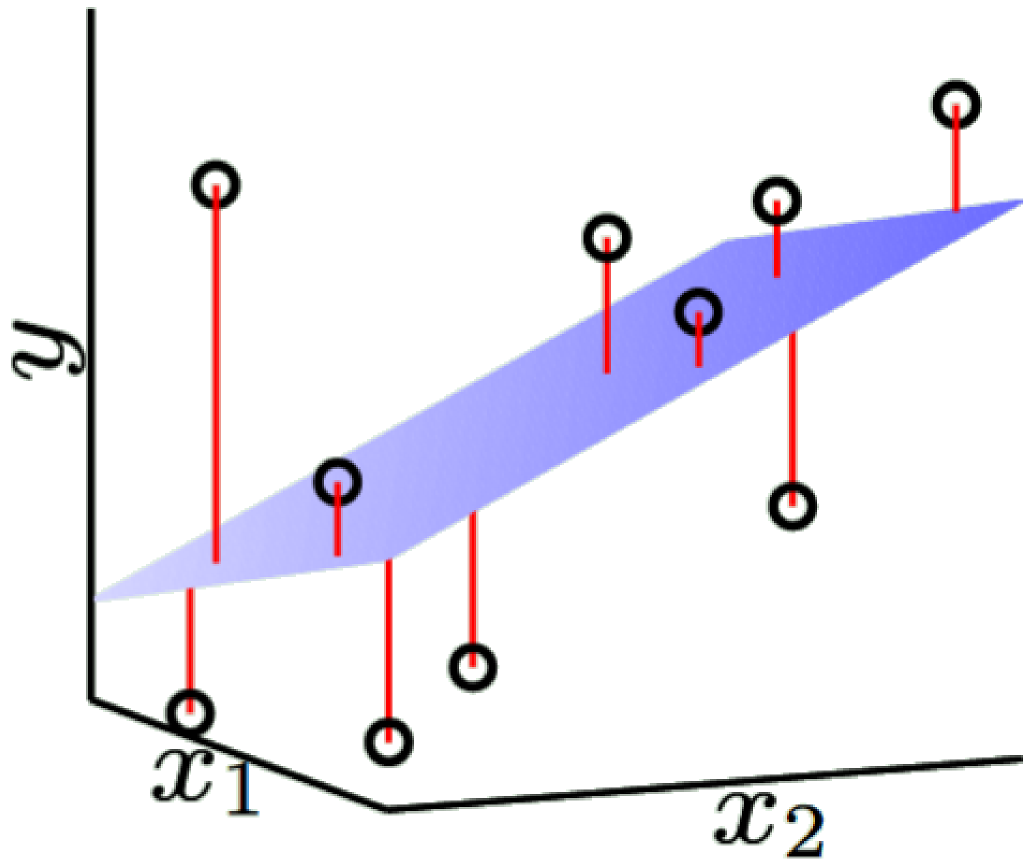
будем решать так:

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

почему?

Общий случай (многих переменных): геометрический смысл





Кстати, полученная задача оптимизации выпукла, единственный глобальный минимум

(кроме вырожденного случая)

Решение задачи минимизации: прямой метод

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

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

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

$$X^{\mathrm{T}} Xw = X^{\mathrm{T}} y$$

решение существует, если столбцы л/н

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

помним, что $\mathrm{rg}(X^{\mathrm{T}} X) = \mathrm{rg}(X)$

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

обобщение обратной на неквадратные матрицы

Обобщённая линейная регрессия: вместо X – что угодно

выражаем целевое значение через л/к базисных функций
(они фиксированы)

$$a(X_1,\ldots,X_n) = w_0 + w_1\varphi_1(X_1,\ldots,X_n) + \cdots + w_k\varphi_k(X_1,\ldots,X_n)$$

$$w = (w_0, w_1, \ldots, w_k)^{\mathrm{T}}$$
$$x = (X_0, X_1, \ldots, X_n)^{\mathrm{T}}$$

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

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

$$\varphi(X) = \begin{bmatrix} \varphi_0(x_1) & \cdots & \varphi_k(x_1) \\ \cdots & \cdots & \cdots \\ \varphi_0(x_m) & \cdots & \varphi_k(x_m) \end{bmatrix}$$

$$\| \varphi(X)w - y \|_2^2 \to \min_w$$

Подробности в нелинейных методах…

Проблема вырожденности матрицы

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

Только ли вырожденность плоха?

Что делать?

Проблема вырожденности матрицы

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

Проблемы, когда матрица X плохо обусловлена…

Решения:

1. **Регуляризация – здесь и в «сложности»**
2. **Селекция (отбор) признаков – «селекция»**
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

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

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 <u>очень больших весов</u>
(у признаков, по которым могут отличаться похожие объекты)

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

Не на все коэффициенты нужна регуляризация! Почему?

Регуляризация: упрощённое объяснение смысла

Пусть есть какая-то зависимость и лишние признаки, например

$$y = X_1 = X_1 + w'X_2 - w'X_3 \text{ при } X_2 = X_3$$

Если теперь $X_2 \approx X_3$**, тогда** $\varepsilon = X_2 - X_3$

$$a = X_1 + w'\varepsilon$$

– может быть сколь угодно большим при больших w'

аналогично при линейных зависимостях!
автоматически, когда объектов мало (сколько?)

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

$$\| w \|_2^2 = w_1^2 + w_2^2 + \ldots + w_n^2$$

эти две формы эквивалентны: решение одного можно получить как решение другого

Всё это справедливо и для общих задач минимизации!

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

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

Есть ещё регуляризация Морозова…

Регуляризация и гребневая регрессия

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

ДЗ Доказать!

$$\lambda \geq 0$$

Такая регрессия называется гребневой регрессией (Ridge Regression)

Виден другой смысл регрессии: складываем две матрицы Грама, неотрицательно определённая + положительно определённая

– боремся с вырожденностью матрицы

потом вернёмся к этому

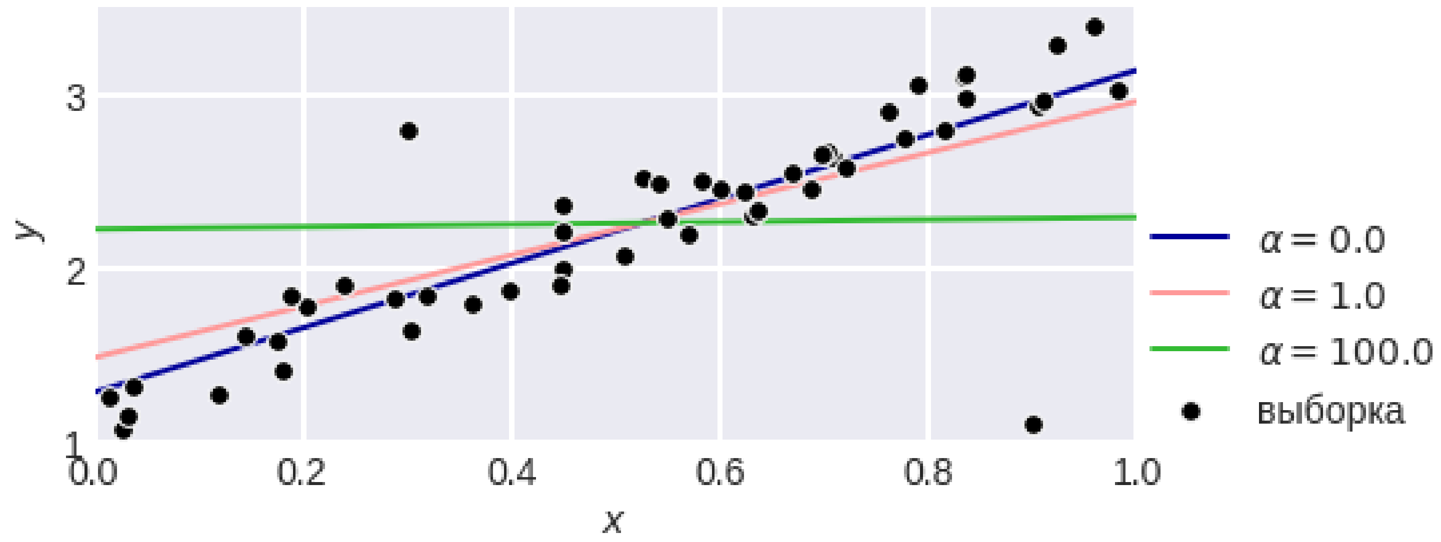
$\lambda = 0$ **– получаем классическое решение**

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

Матрица очевидно становится обратимой!

значение параметра регуляризации можно выбрать на скользящем контроле

Минутка кода: регуляризация и гребневая регрессия




```
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])
```

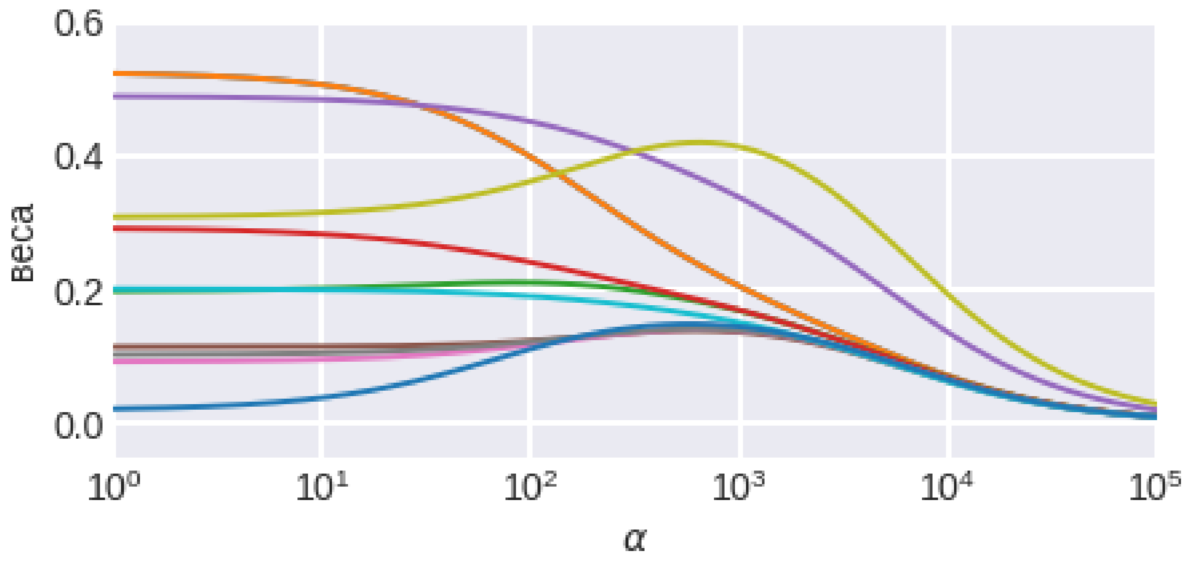
Кажется, что при регуляризации отклоняемся к выбросам, но дело не в этом

Регуляризация и гребневая регрессия

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

$$\lambda \geq 0$$

добавление shrinkage penalty (регуляризатора)



параметр регуляризации может подбираться с помощью скользящего контроля

Регуляризация и гребневая регрессия

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

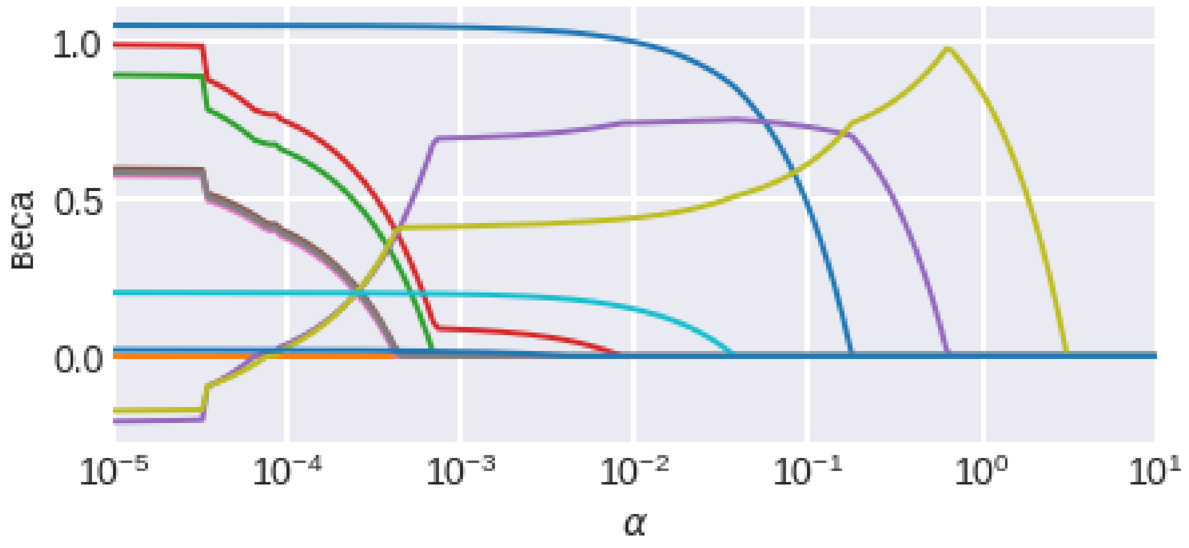
Перед регуляризацией – стандартизация!!!

LASSO (Least Absolute Selection and Shrinkage Operator)

Попробуем другой «штраф за сложность» (сейчас поймём название)

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

$$\lambda \geq 0$$



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

Эксперименты с одинаковыми и зависимыми признаками

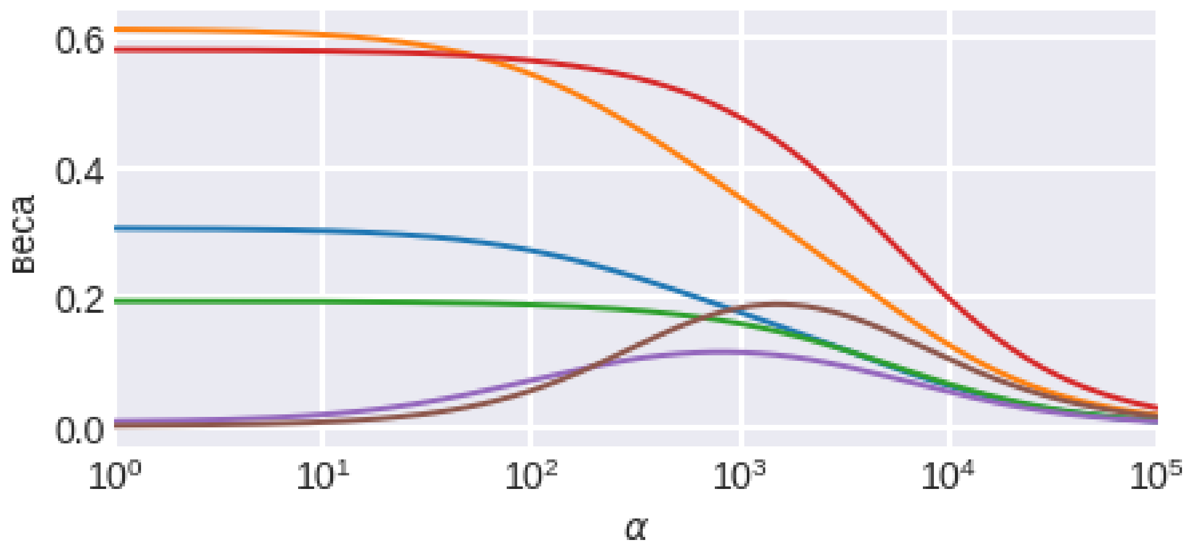
здесь была задача

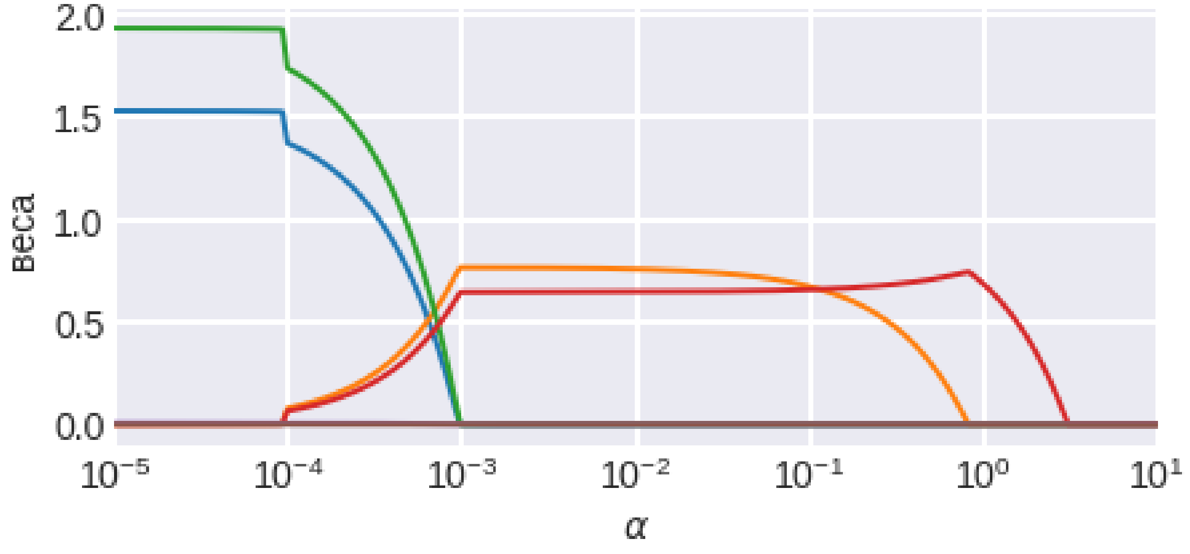
```
X = np.random.rand(1000, 11)
X[:,1] = X[:,0]
X[:,4] = X[:,2] + X[:,3]
X[:,8] = X[:,5] + X[:,6] + X[:,7]
y = X[:,0] + 0.8*X[:,4] + 0.4*X[:,8] + 0.2*X[:,9] +
             0.5*np.random.randn(1000)
```

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

Масштаб очень важен! см. дальше

Эксперименты с одинаковыми и зависимыми признаками





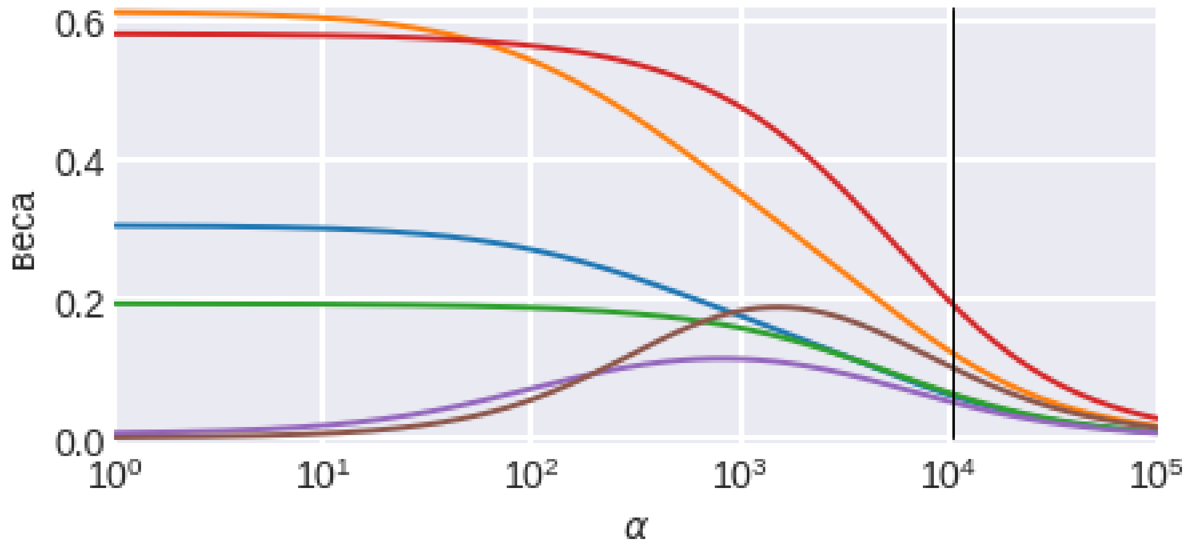

```
np.random.seed(10)
X = np.random.rand(1000, 6)
X[:,1] = X[:,0]
X[:,2] = X[:,3]

X[:,0] = 1 * X[:,0]
X[:,1] = 2 * X[:,1]
X[:,2] = 1 * X[:,2]
X[:,3] = 3 * X[:,3]
X[:,4] = 1 * X[:,4]
X[:,5] = 2 * X[:,5]

y = 1.5 * X[:,0] + 2*X[:,2] +
0.5*np.random.randn(1000)
```

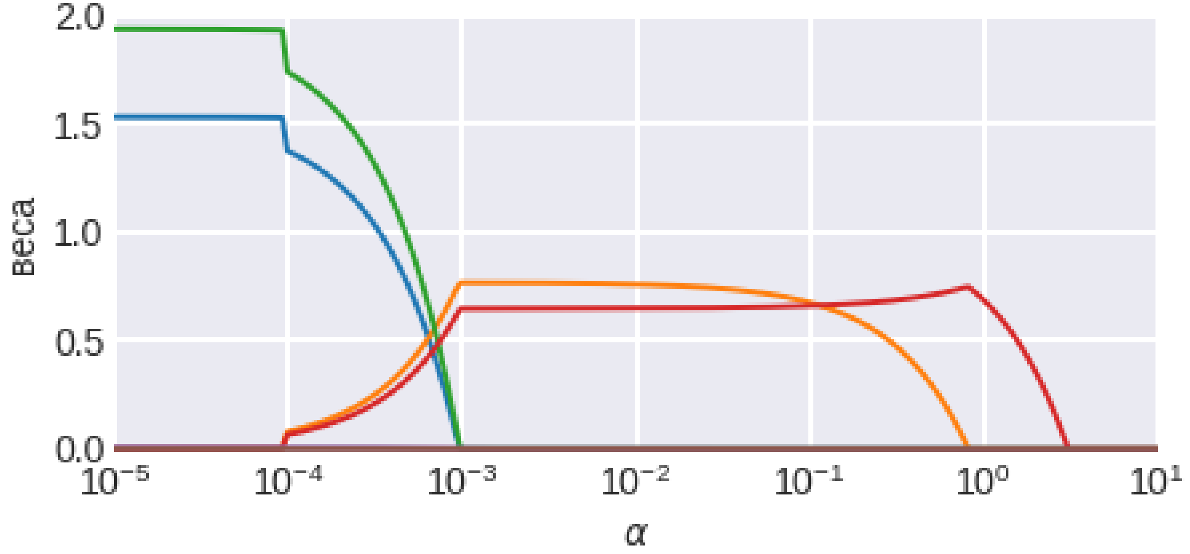
Эксперименты с одинаковыми и зависимыми признаками

$$Y = 1.5X_1 + 2X_3 = 0.75X_2 + 0.66X_4$$



$$\lambda = 1$$
$$Y = 0.31X_1 + 0.61X_2 + 0.19X_3 + \\ + 0.58X_4 + 0.01X_5 + 0.0X_6$$

$$\lambda \sim 10500$$
$$Y = 0.06X_1 + 0.12X_2 + 0.06X_3 + \\ + 0.19X_4 + 0.05X_5 + 0.1X_6$$



$$\lambda = 10^{-5}$$
$$Y = 1.53X_1 + 1.94X_3$$

$$\lambda \sim 0.01$$
$$Y = 0.76X_2 + 0.65X_4$$

Эксперименты с одинаковыми и зависимыми признаками

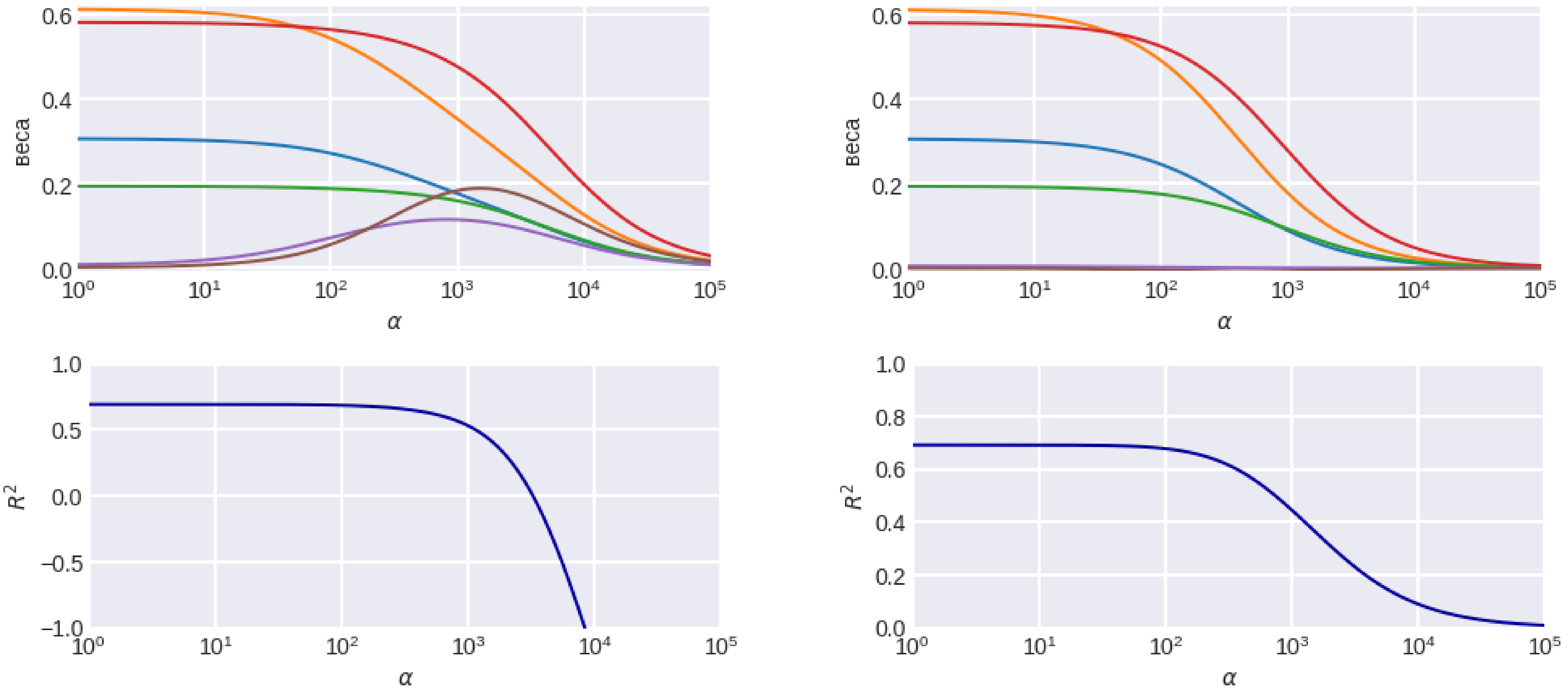
веса зависят от масштаба признаков

при сильной регуляризации меняется распределение весов зависимых признаков

Пусть $Y = 4X_1$, $X_1 = X_2$

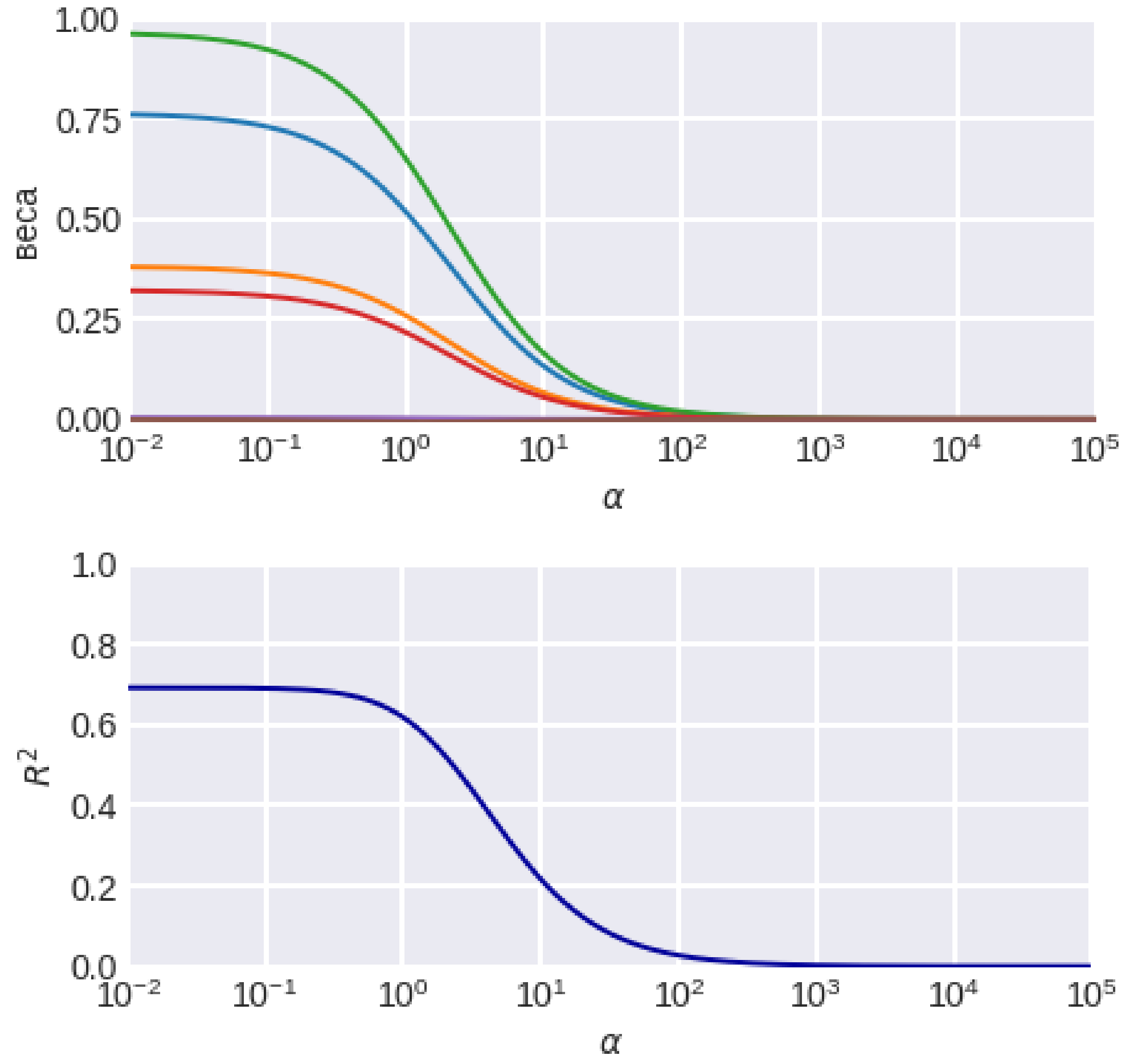
w_1	w_2	$\|\|w\|\|_1$	$\|\|w\|\|_2^2$
5	– 1	6	26
4	0	4	16
3	1	4	10
2	2	4	8

Эксперименты с одинаковыми и зависимыми признаками: L_2-регуляризация

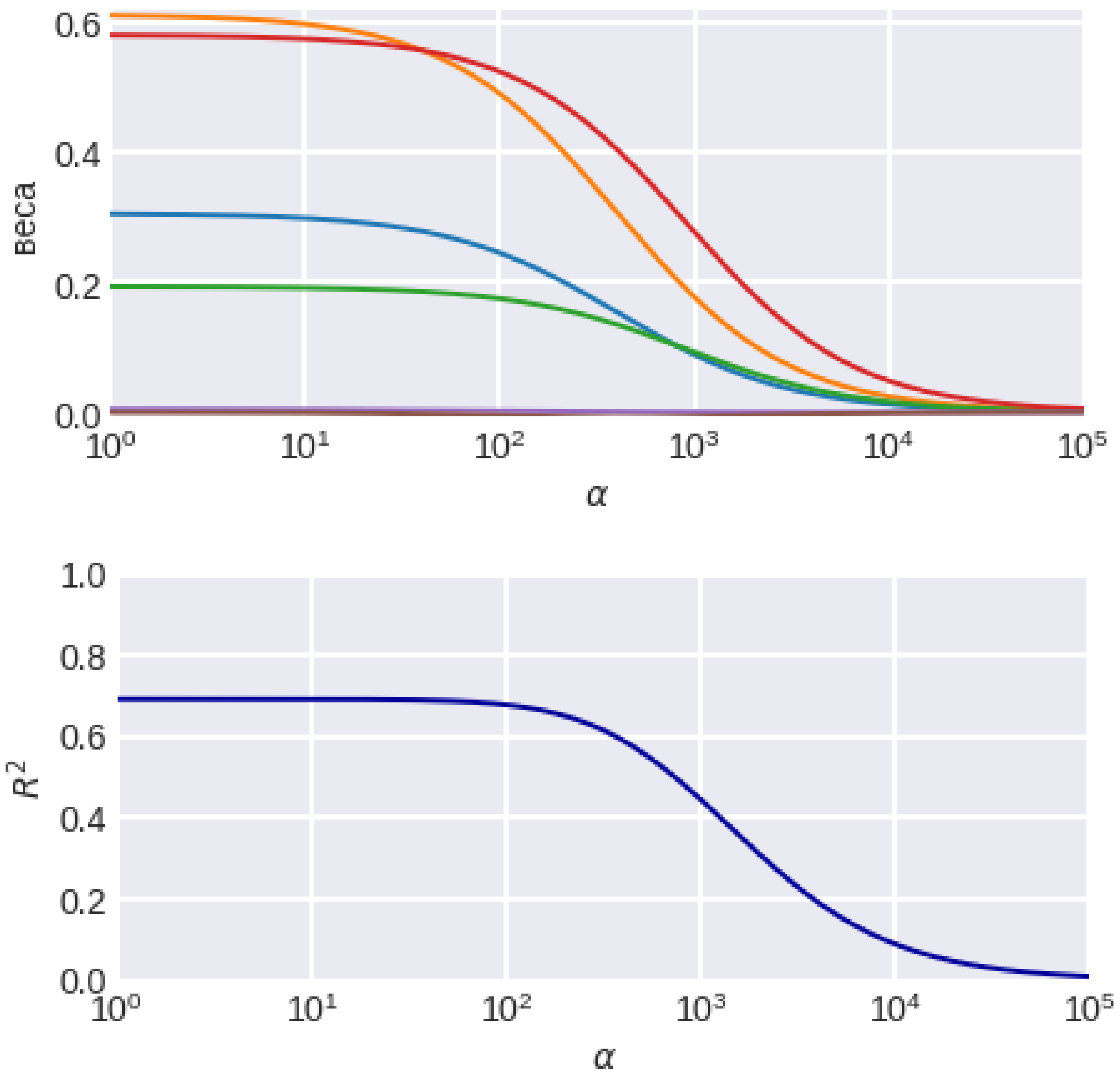


`fit_intercept=True`

Эксперименты с одинаковыми и зависимыми признаками: L_2-регуляризация

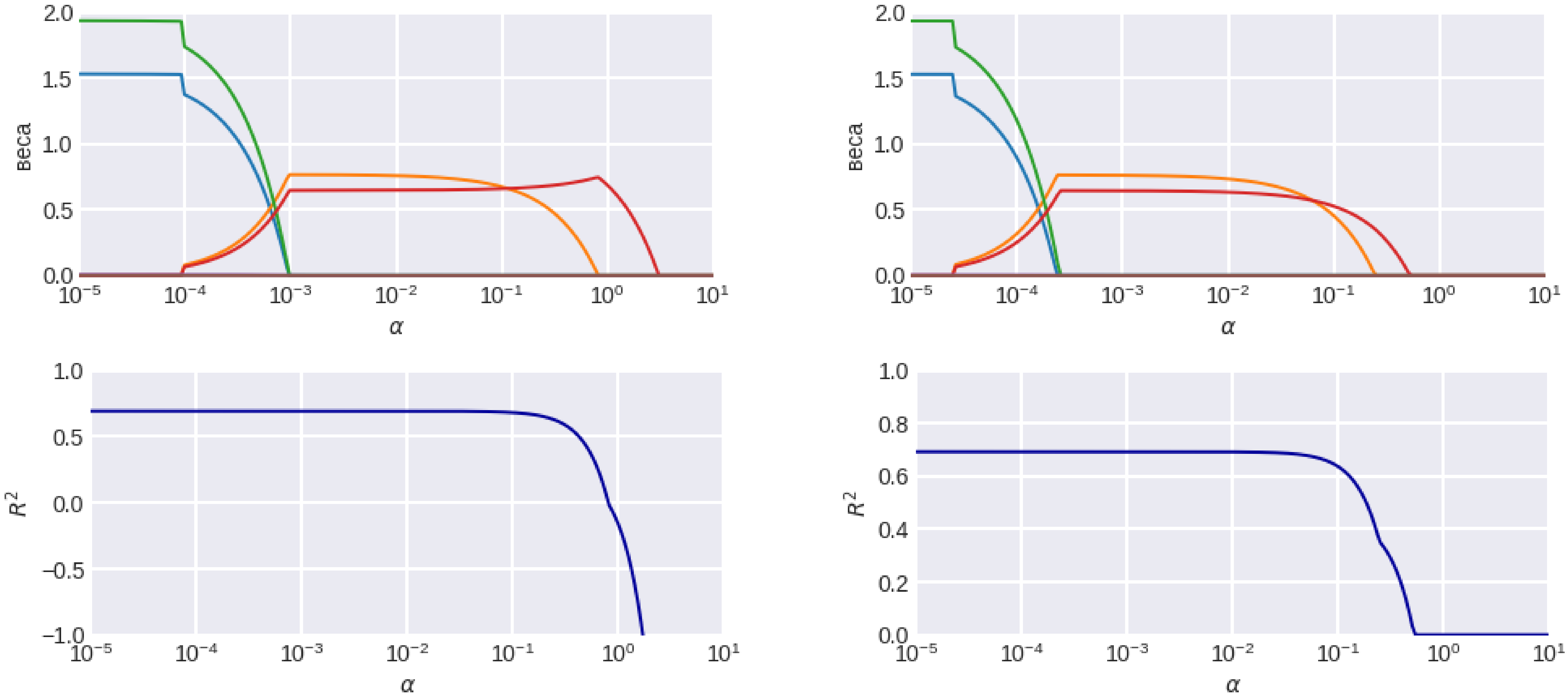


`fit_intercept=True, normalize=True`



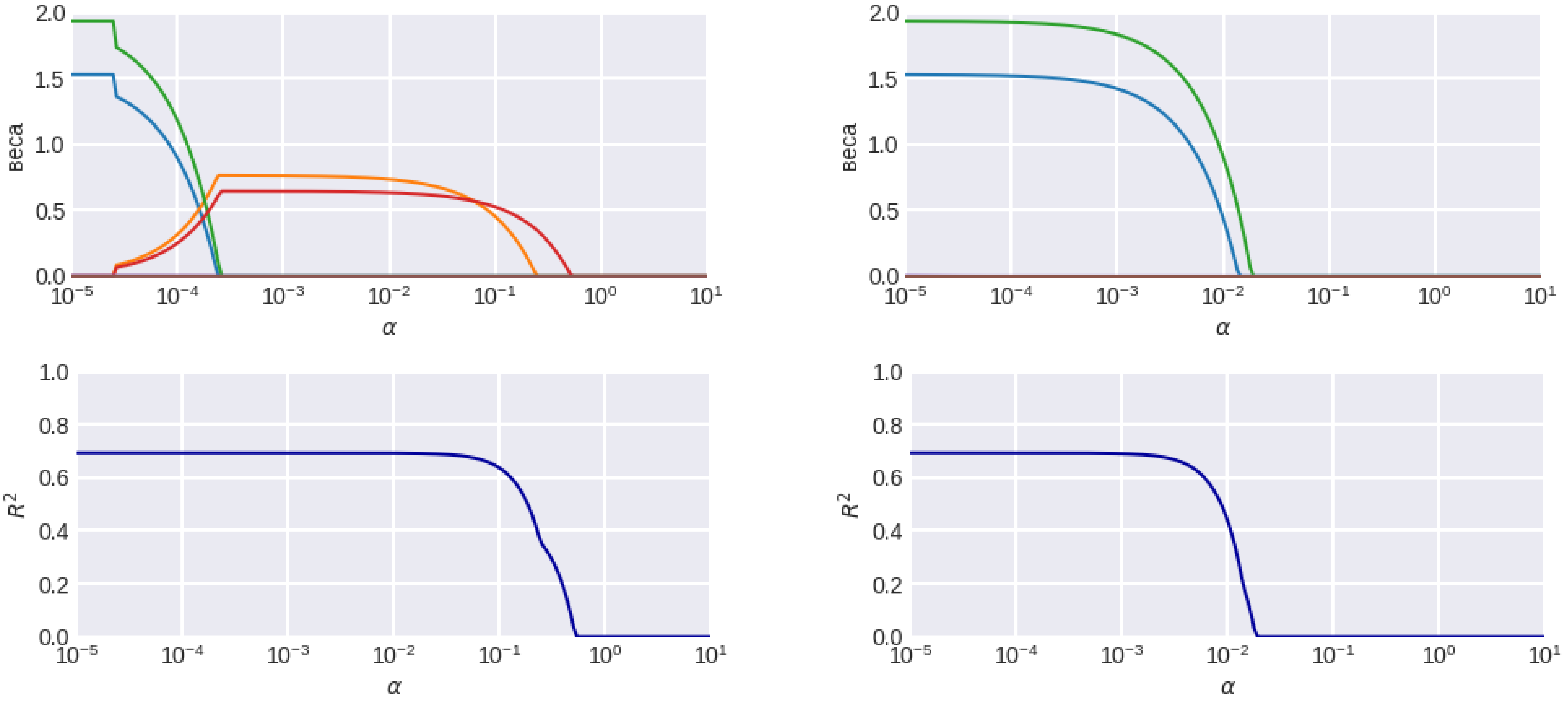
`fit_intercept=True`

Эксперименты с одинаковыми и зависимыми признаками: L_1-регуляризация



`fit_intercept=True`

Эксперименты с одинаковыми и зависимыми признаками: L_1-регуляризация



`fit_intercept=True` `fit_intercept=True, normalize=True`

Эксперименты с одинаковыми и зависимыми признаками

Часто важно

- **использовать свободный член**
- **предварительно нормировать данные**

Семейство регуляризированных линейных методов

Ridge

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

LASSO (Least Absolute Selection and Shrinkage Operator)

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

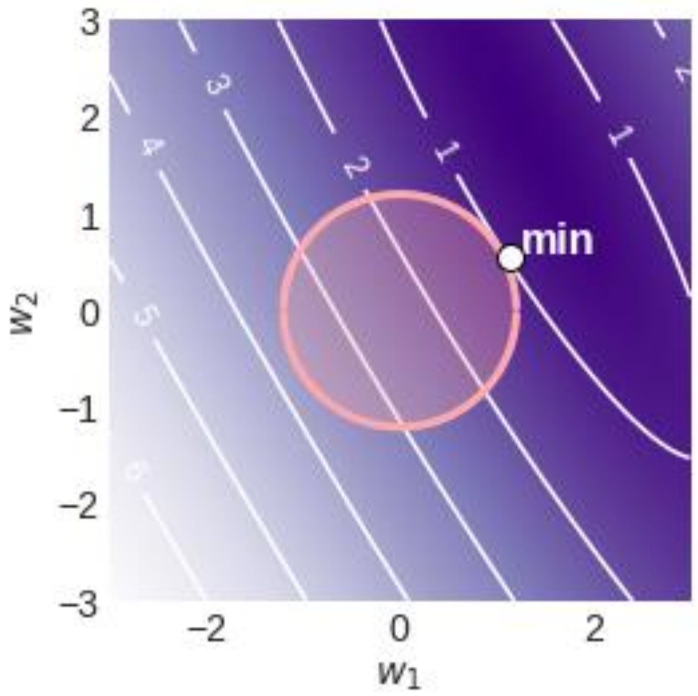
Elastic Net = LASSO + Ridge

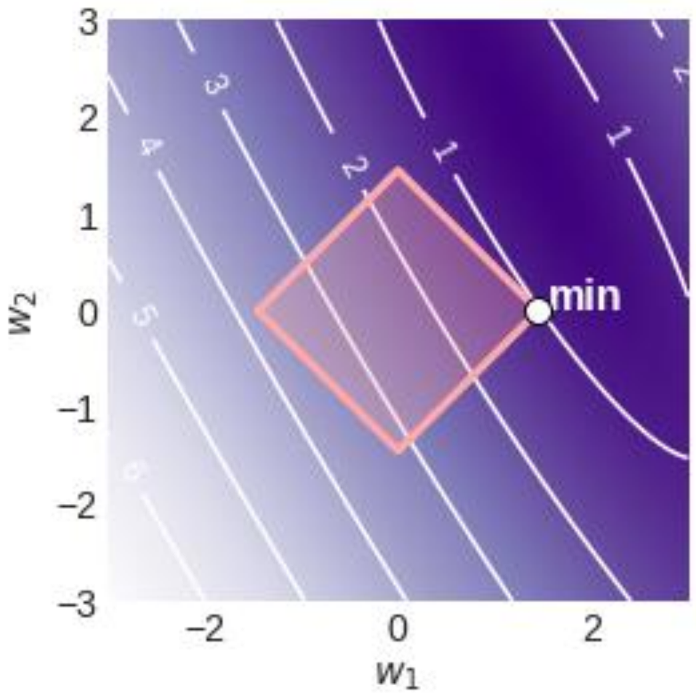
$$\| y - Xw \|_2^2 + \lambda_1 \| w \|_1 + \lambda_2 \| w \|_2^2 \to \min_w$$

Геометрический смысл Ridge, LASSO и Elastic Net

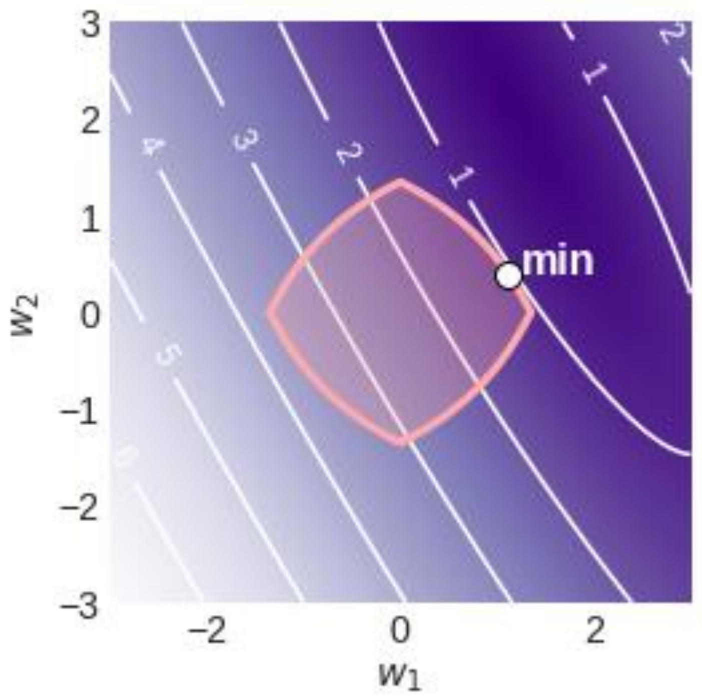
$$\sum_{i=1}^{m}\left(y_i - w_0 - \sum_{j=1}^{n} w_j x_{ij}\right)^2 \to \min_w, \quad \sum_{j=1}^{n} w_j^2 \le s$$

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





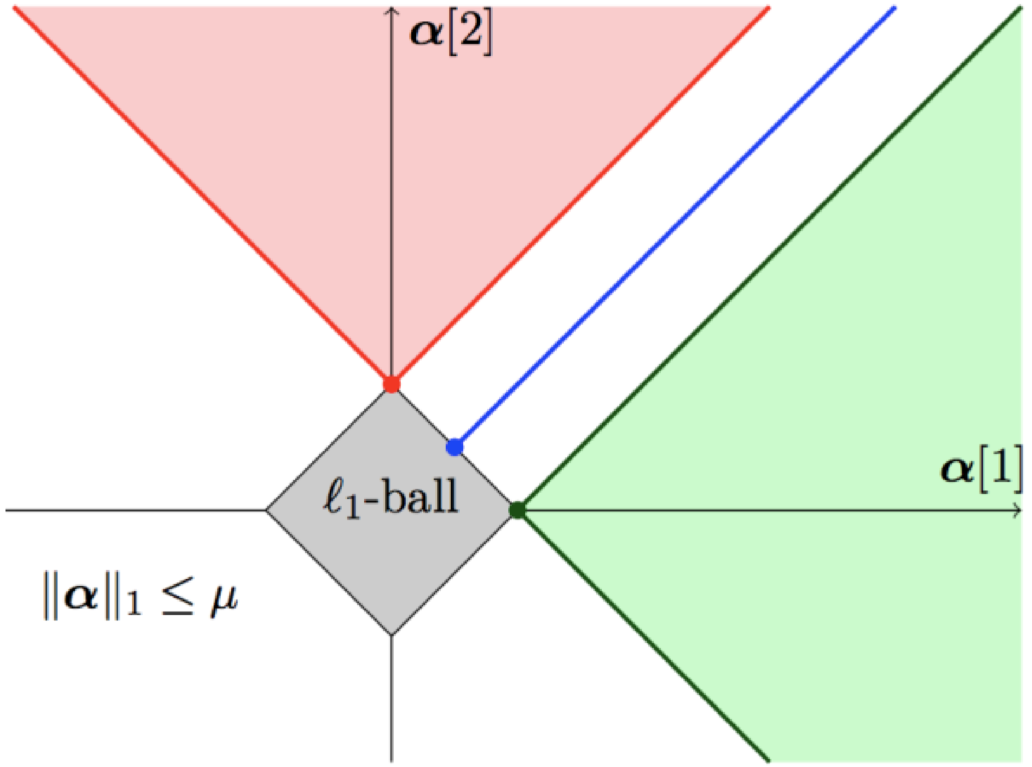
Геометрический смысл Ridge, LASSO и Elastic Net

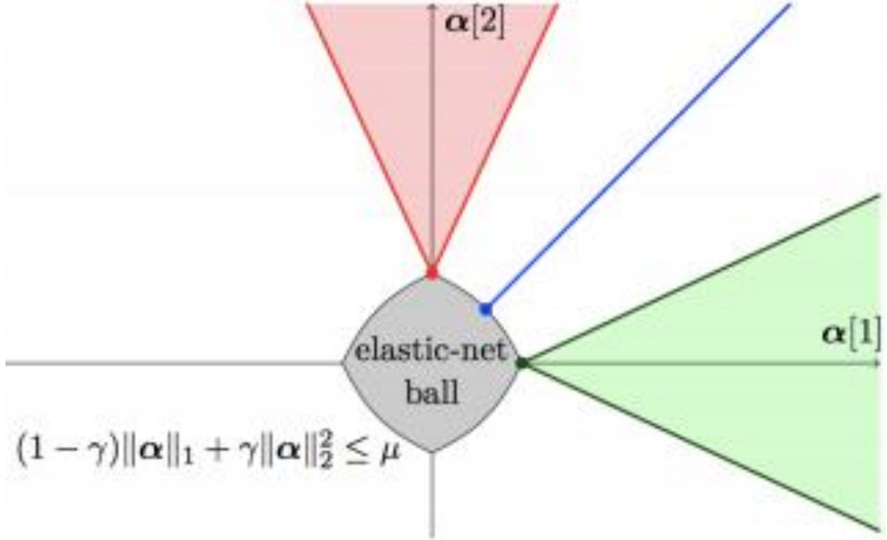


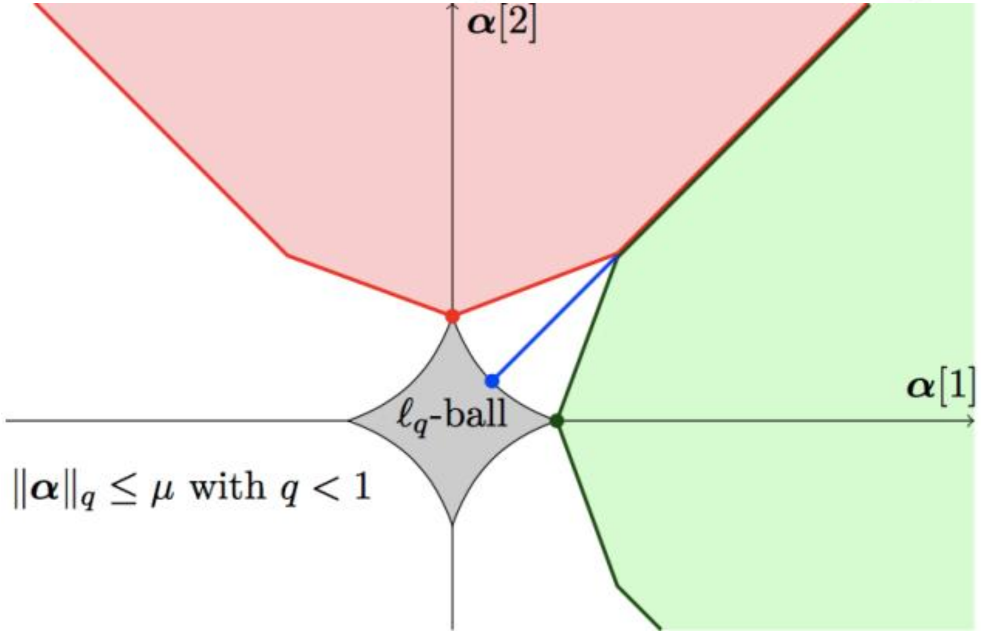
на практике часто модель и не может зависеть от небольшого числа переменных

Эффект разреженности

если линии уровня оптимизируемой функции – концентрические окружности…





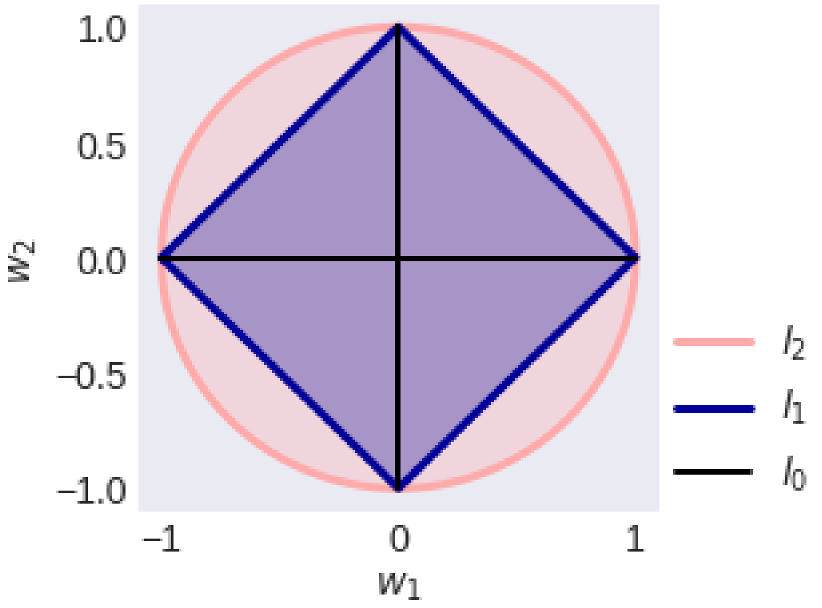


David S. Rosenberg «Foundations of Machine Learning»
https://bloomberg.github.io/foml/

Почему L1-норма ⇒ разреженность

1. Больше вероятность, что линии уровней функции ошибки касаются области ограничений в точках с нулевыми координатами, см. рис.

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



$$\|w\|_0 = |\{t \mid w_t \neq 0\}|$$

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..

потом будет обоснование регуляризации с помощью вероятностных предположений

Проблема вырожденности / плохой обусловленности 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

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

Маленький обзор стратегий:

1 стратегия – **перебор** – умный перебор подмножества признаков

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

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

Обоснование необходимости селекции

- **Проблема вырожденности в линейной регрессии**
- **Проблема «почти дубликатов»**
- **Уменьшение модели и интерпретация**
- **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

$\Rightarrow$

	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: градиентный метод обучения

недостатки прямого…

работа с большими матрицами (тем более обращение)

В лекции «оптимизация»…

$$\frac{1}{2}\sum_{i=1}^{m}(a(x_i \mid w) - y_i)^2 \to \min$$

$$w^{(t+1)} = w^{(t)} - \eta\sum_{i=1}^{m}(a(x_i \mid w^{(t)}) - y_i)\frac{\partial a(x_i \mid w^{(t)})}{\partial w}$$

Gradient Descent

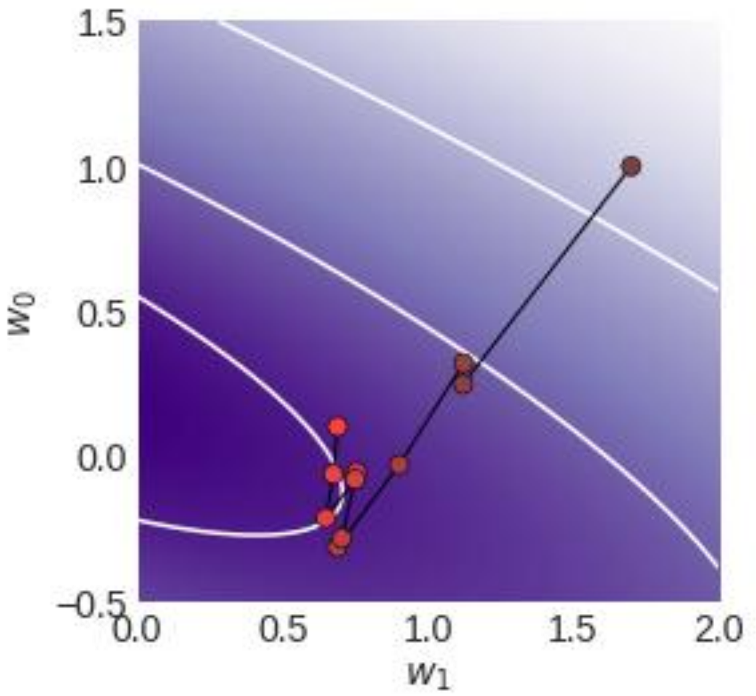
$$a(x \mid w) = w^{\mathrm{T}}x$$

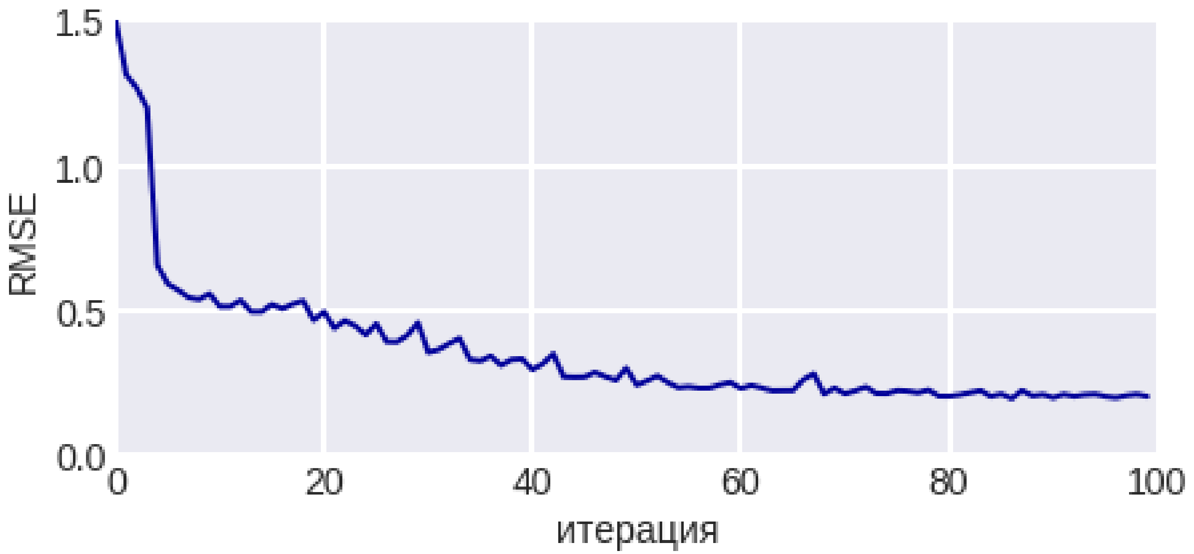
$$w^{(t+1)} = w^{(t)} - \eta\sum_{i=1}^{m}(a(x_i \mid w^{(t)}) - y_i)x_i$$

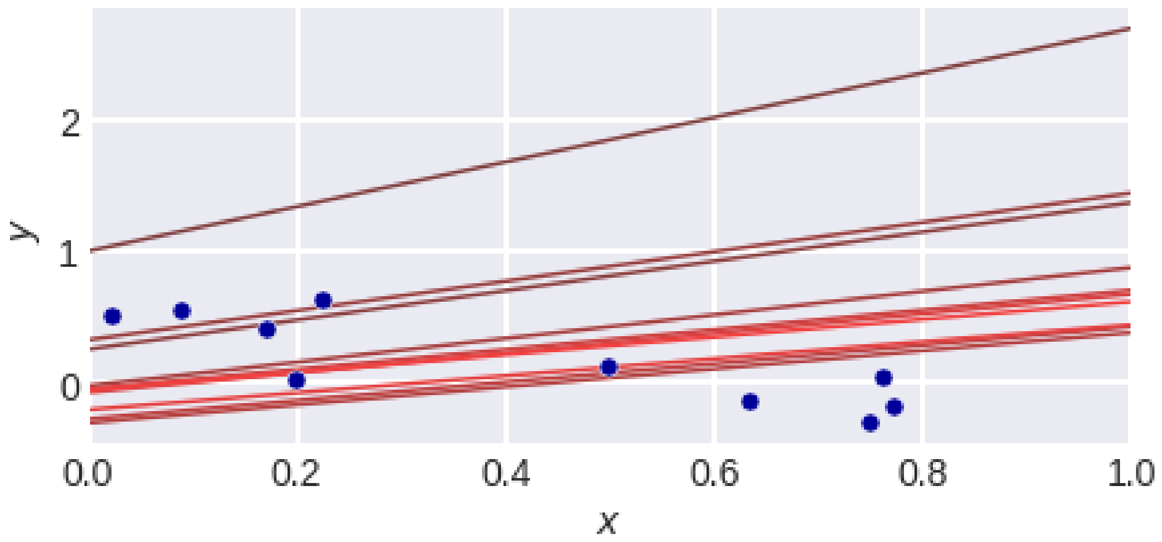
Stochastic Gradient Descent

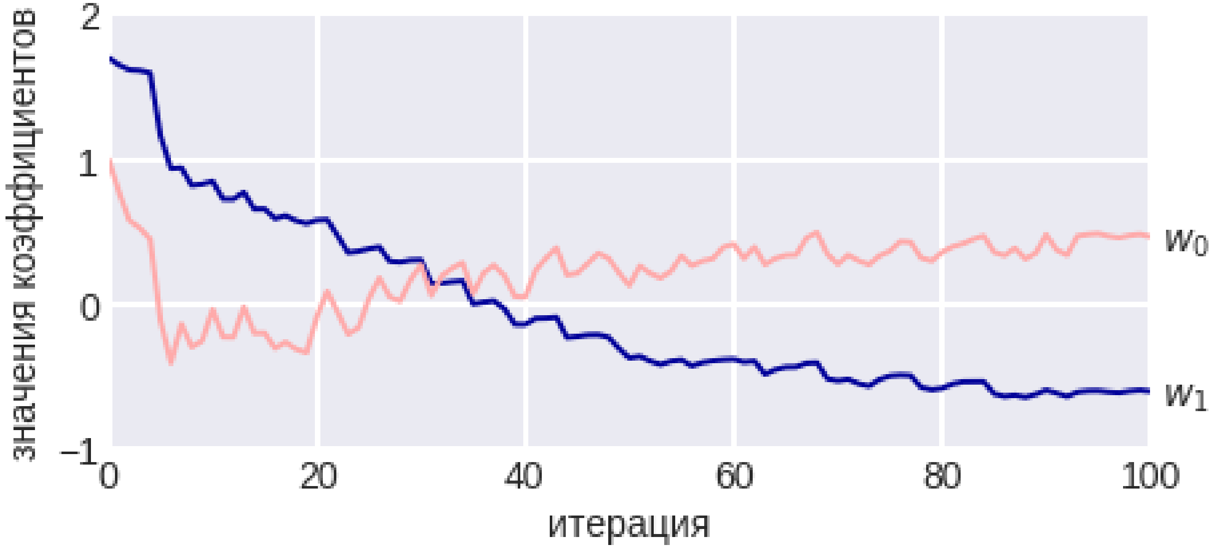
$$w^{(t+1)} = w^{(t)} - \eta_t(a(x_i \mid w^{(t)}) - y_i)x_i$$

Линейная регрессия: градиентный метод обучения









Реализация в scikit-learn

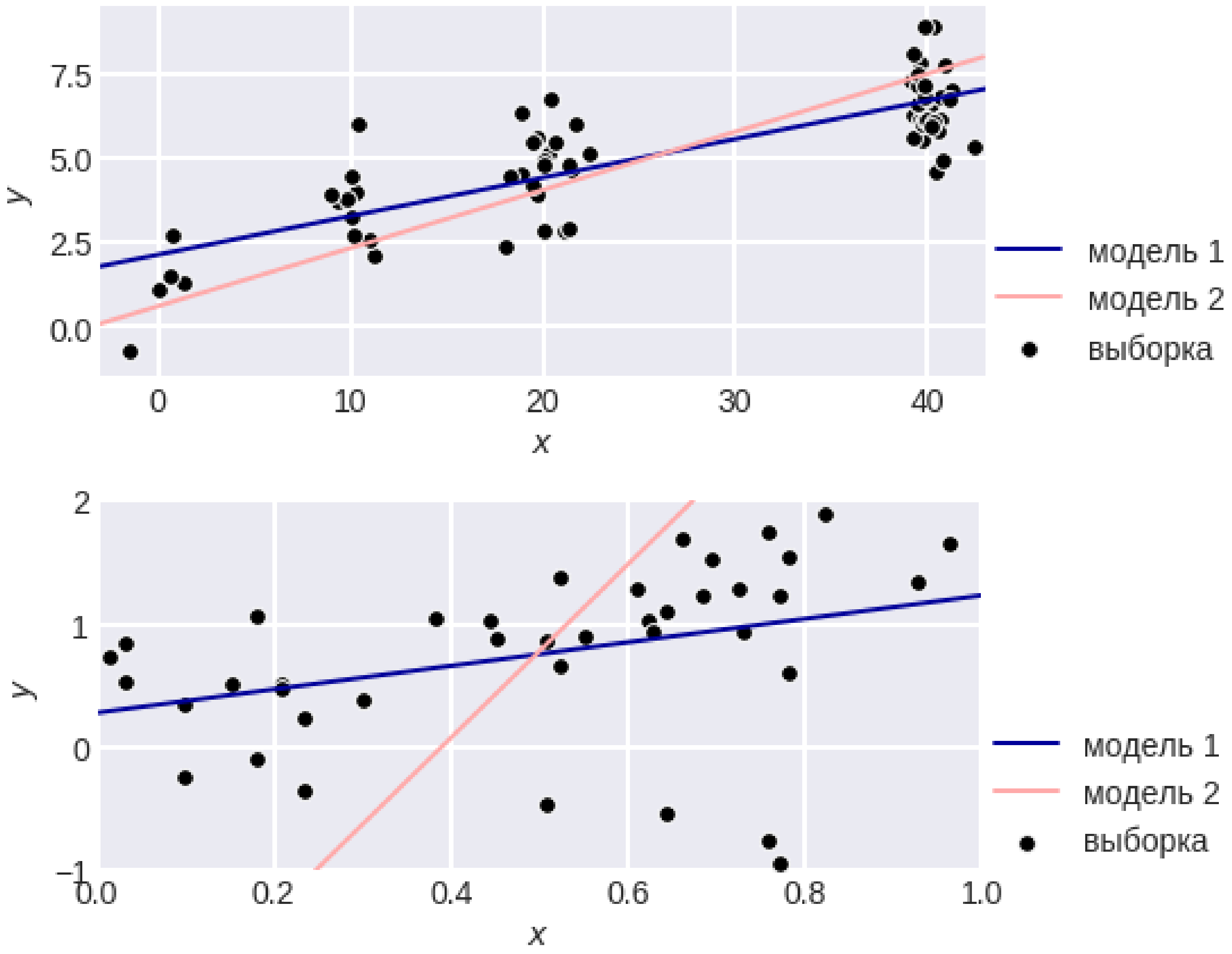
`sklearn.linear_model.Ridge`

alpha=1.0	**Коэффициент регуляризации, больше – сильнее** (в отличие от других функций)
fit_intercept=True	**Использовать ли свободный член**
normalize=False	**Нормализация данных** Игнорируется без свободного члена
solver="auto"	**Метод оптимизации** "auto", "svd", "cholesky", "lsqr", "sparse_cg", "sag", "saga"
copy_X=True, max_iter=None, tol=0.001, random_state=None	

```
sklearn.linear_model.Lasso(alpha=1.0, fit_intercept=True, normalize=False,
precompute=False, copy_X=True, max_iter=1000, tol=0.0001, warm_start=False,
positive=False, random_state=None, selection="cyclic")
```

```
sklearn.linear_model.ElasticNet(alpha=1.0, l1_ratio=0.5, fit_intercept=True,
normalize=False, precompute=False, max_iter=1000, copy_X=True, tol=0.0001,
warm_start=False, positive=False, random_state=None, selection="cyclic")
```

Две регрессии



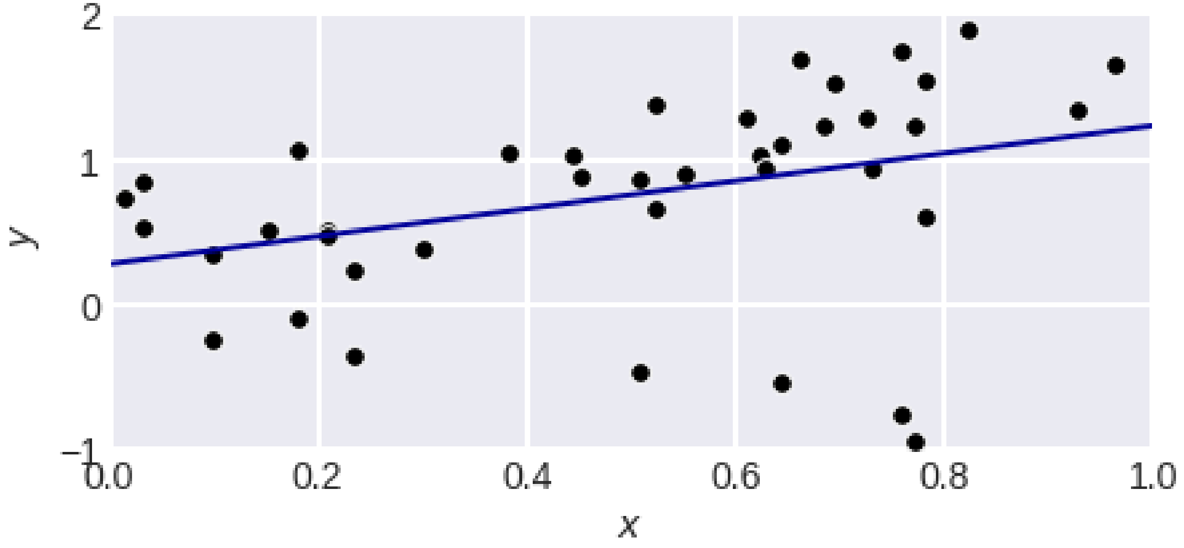
Чем отличаются модели 1 и 2?

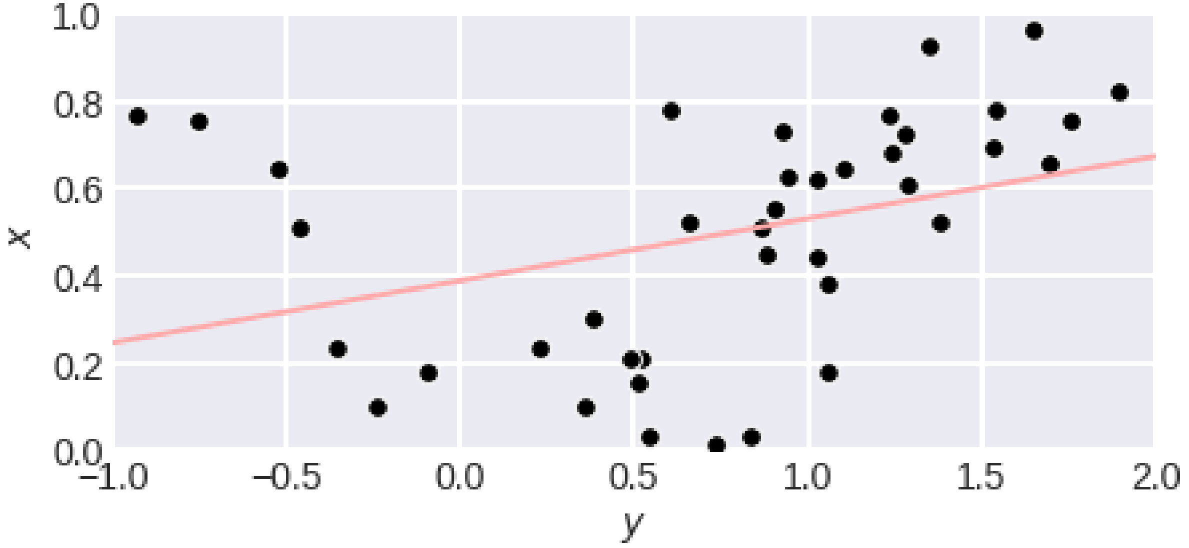
Две регрессии: y(x) vs x(y)

разные задачи $y(x)$ и $x(y)$, хотя зависимость линейная

$$Y = w_0 + w_1 X_1 \qquad X_1 = w_0 + w_1 Y$$

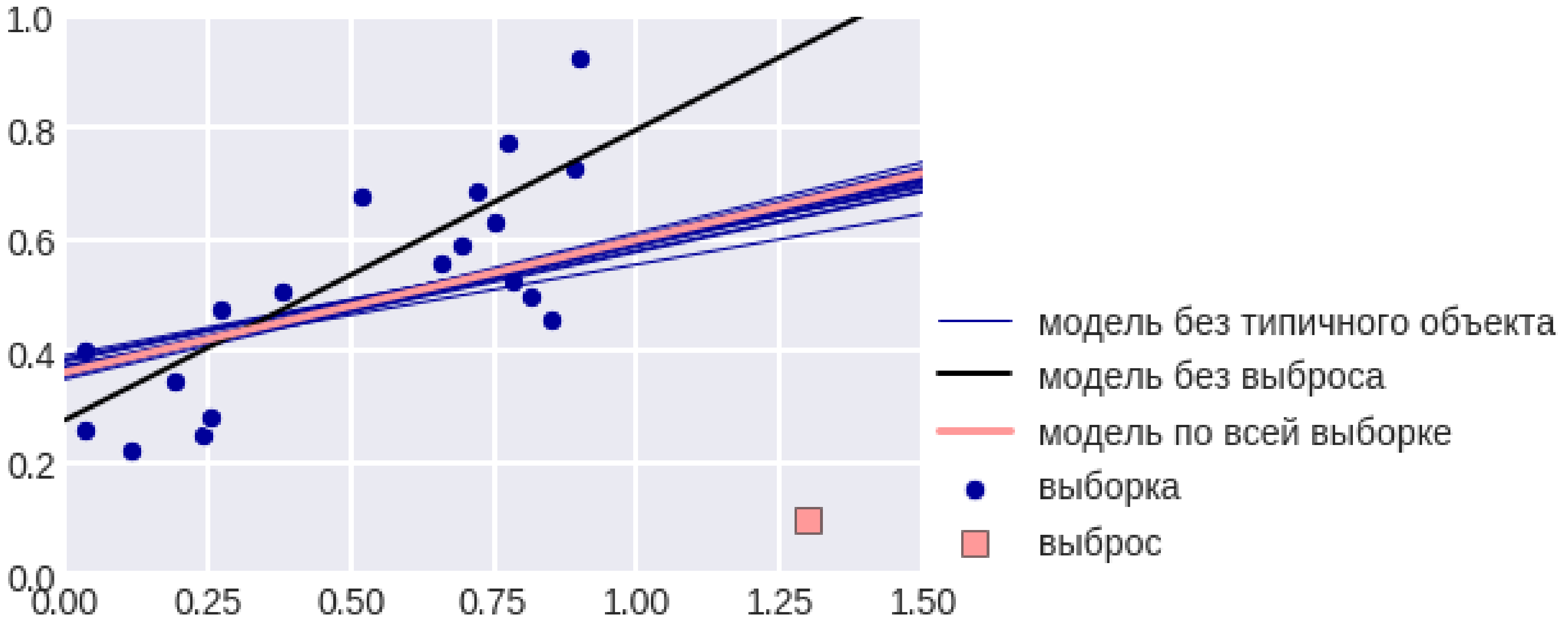
$$\left\| \begin{bmatrix} x_1 & 1 \\ \cdots & \cdots \\ x_m & 1 \end{bmatrix} \begin{pmatrix} w_1 \\ w_0 \end{pmatrix} - \begin{pmatrix} y_1 \\ \cdots \\ y_m \end{pmatrix} \right\|_2^2 \to \min \qquad \left\| \begin{bmatrix} y_1 & 1 \\ \cdots & \cdots \\ y_m & 1 \end{bmatrix} \begin{pmatrix} w_1 \\ w_0 \end{pmatrix} - \begin{pmatrix} x_1 \\ \cdots \\ x_m \end{pmatrix} \right\|_2^2 \to \min$$





есть и «промежуточная стратегия» – дальше PCA

Линейная регрессия – неустойчивость к выбросам



Ошибка с весами

Если у каждого объекта есть цена ошибки…

$$\sum_{i=1}^{m} \underbrace{v_i}_{>0} \left(y_i - w^{\mathrm{T}} x_i\right)^2 + \ldots = \sum_{i=1}^{m} \left(\sqrt{v_i} y_i - w^{\mathrm{T}} (\sqrt{v_i} x_i)\right)^2 + \ldots \to \min$$

небольшая переформулировка задачи:

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

$$(y - Xw)^{\mathrm{T}} V (y - Xw) \sim \| V^{1/2} y - V^{1/2} X w \|_2^2 \to \min_w$$

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

Ошибка с весами

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

Как реализовать:

1) перейти к новым данным («испорченными весам»)

2) если веса целые числа – можно продублировать объекты

3) 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

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

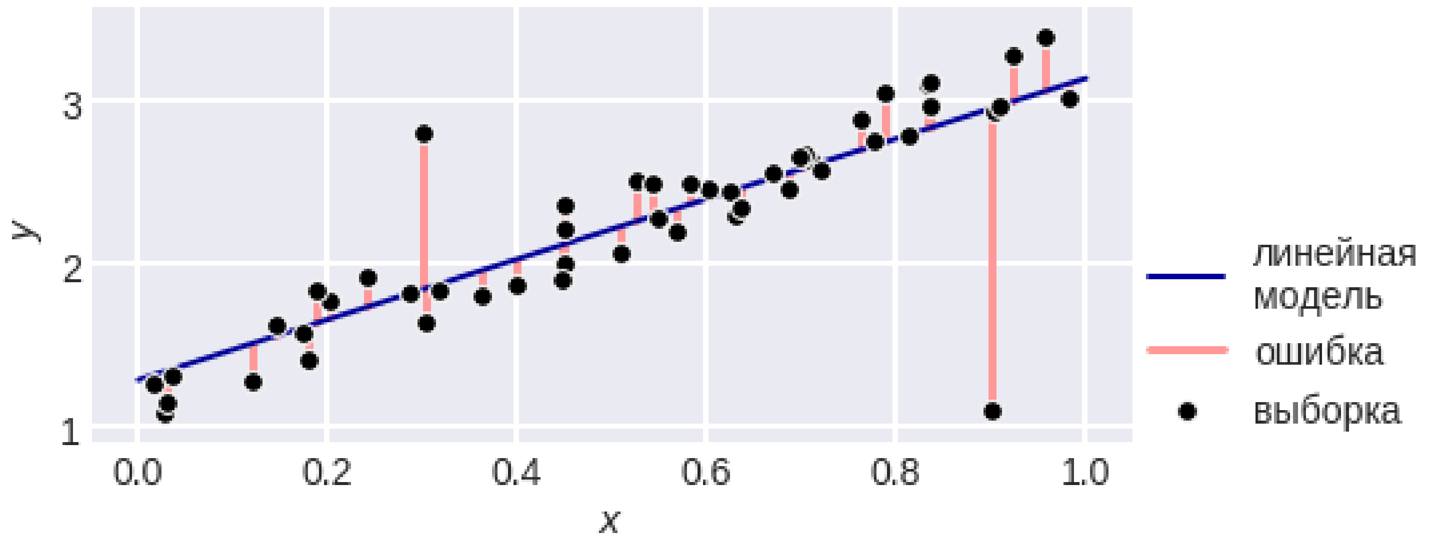
1.2. Вычислить ошибки на обучении

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

1.3. Пересчитать веса объектов

$$v_i = \exp(-\varepsilon_i^2)$$

нормировать на сумму

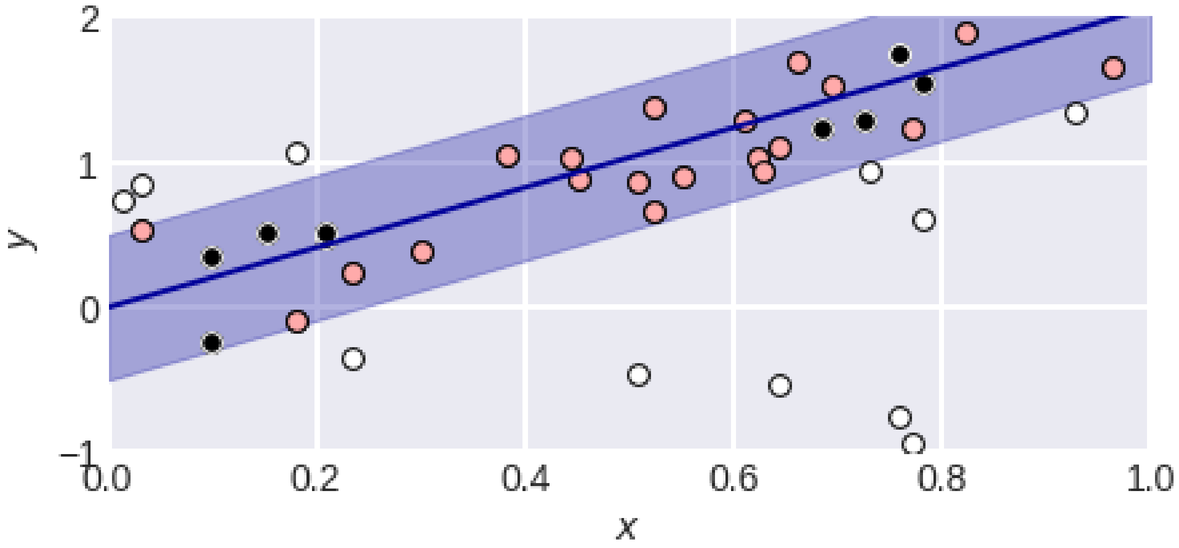


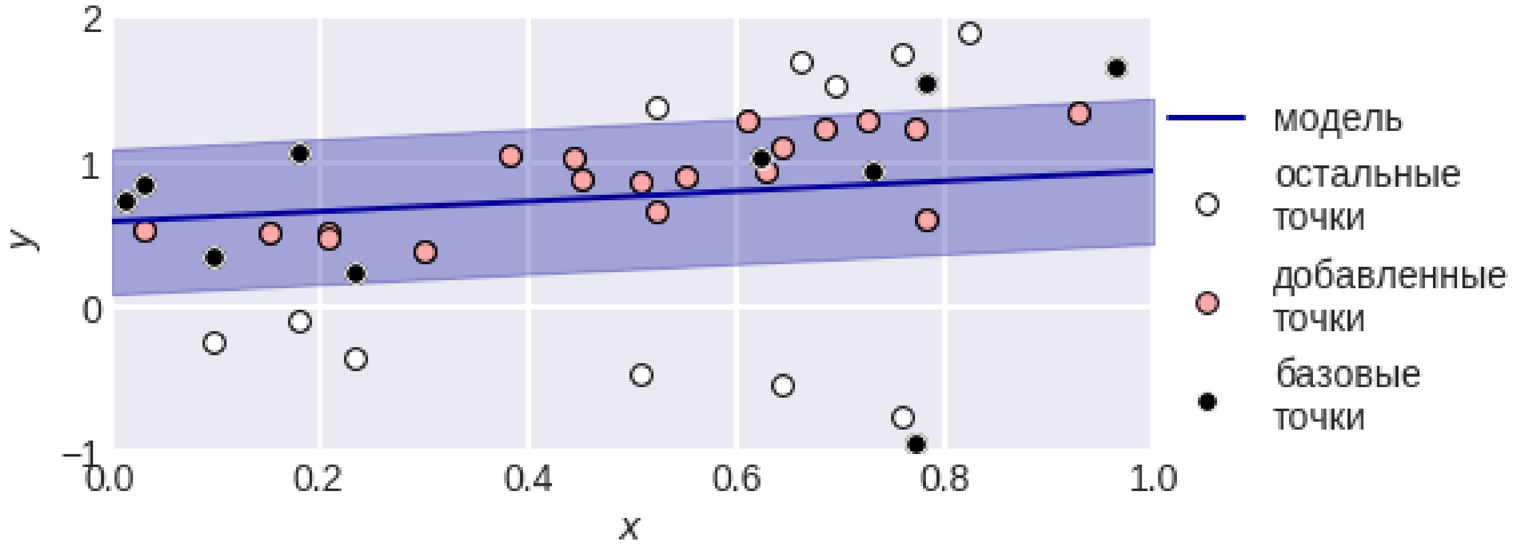
можно использовать любую регрессионную модель

при пересчёте весов можно использовать другую невозрастающую функцию; можно (иногда нужно) нормировать

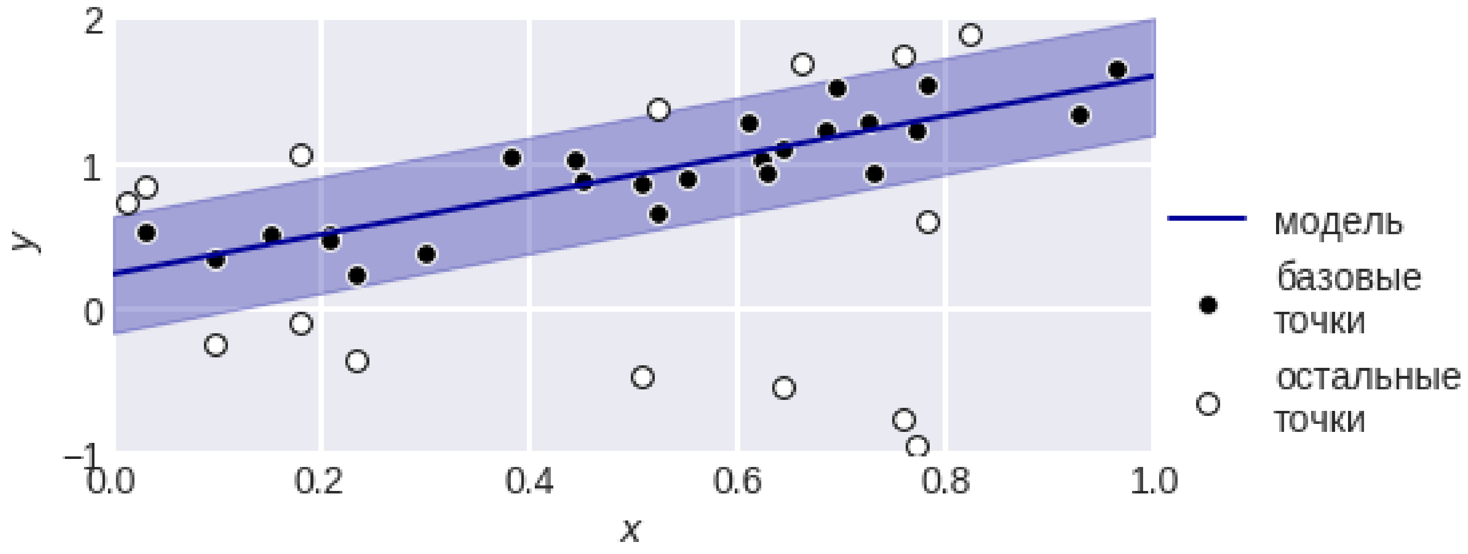
RANdom SAmple Consensus (RANSAC)

- **несколько раз**
  - **выбрать случайное подмножество точек – базовое (inliers)**
  - **обучить модель на базовом подмножестве**
  - **найти все точки, которые хорошо предсказываются моделью**
    например, ошибка не больше ε
  - **пополнить ими базовое множество**
  - (если добавили много) **переобучить модель на новом множестве**
- **выбрать модель с наименьшей ошибкой**





Минутка кода: RANSAC в scikit-learn




```
from sklearn.linear_model import RANSACRegressor
# Robustly fit linear model with RANSAC algorithm
ransac = RANSACRegressor()
ransac.fit(x[:, np.newaxis], y)
inlier_mask = ransac.inlier_mask_
outlier_mask = np.logical_not(inlier_mask)
```

Минутка кода: RANSAC в scikit-learn

```
sklearn.linear_model import RANSACRegressor
```

`base_estimator=None`	**Базовый алгоритм (по умолчанию – линейная регрессия)**
`min_samples=None`	**Число / доля базовых объектов (*n*+1)**
`residual_threshold=None`	**Порог для пополнения базового множества (MAD(y))**
`max_trials=100`	**Число итераций**
`stop_n_inliers`	**Остановить вычисления, если найдено столько базовых точек**
`loss="absolute_loss"`	**Как оценивать ошибку**

```
is_data_valid=None, is_model_valid=None, max_skips=inf,
stop_score=inf, stop_probability=0.99, lossrandom_state=None
```

Реализация в scikit-learn немного отличается от некоторых описаний:

Качество = число базовых (inliers) объектов;
лучшая модель выбирается по числу базовых,
если у нескольких моделей число совпадает,
тогда сравнивается ошибка на всей выборке.

Плюсы и минусы линейных алгоритмов

+ простой, надёжный, быстрый, популярный метод
+ интерпретируемость (⇒ нахождение закономерностей)
+ интерполяция и экстраполяция
+ может быть добавлена нелинейность, с помощью генерации новых признаков
(дальше – это можно автоматизировать)
+ хороши для теоретических исследований (в RIdge есть явная формула)
+ коэффициенты асимптотически нормальны
(можно тестировать гипотезы о влиянии признаков)
+ глобальный минимум в оптимизируемом функционале
– линейная гипотеза вряд ли верна
– в теоретическом обосновании ещё предполагается нормальность ошибок
(зависит от функции ошибок)
– «страдает» из-за выбросов
– признаки в одной шкале и однородные (см. успешные примеры)
– проблема коррелированных признаков
⇒ необходимость регуляризации, селекции, PCA, data↑