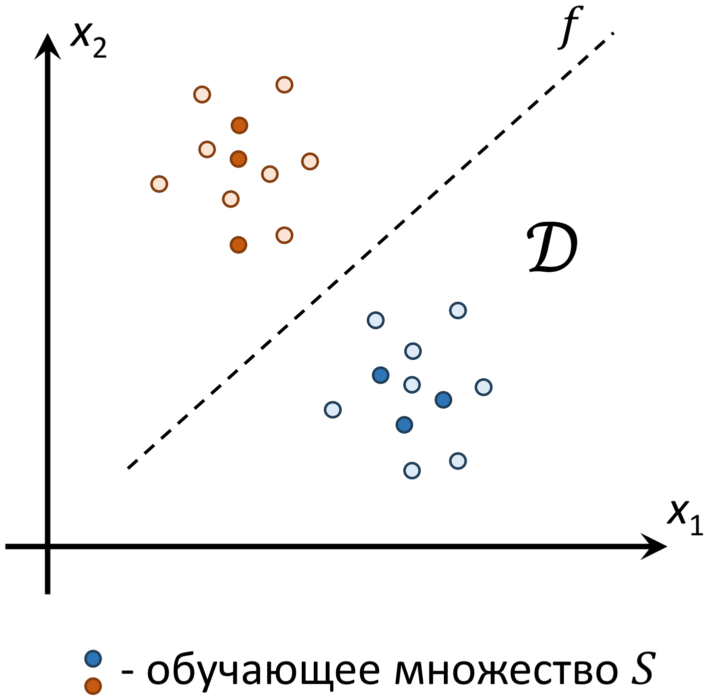
Методы искусственного интеллекта

Лекция 4. Обучение. Нейронные сети

Тема 9. Основы обучения с учителем

Постановка задачи обучения с учителем

- Исходные данные:
  - Множество примеров (образцов) $\mathcal{X}$, представленные как векторы *признаков*.
  - Пространство меток $\mathcal{Y}$:
    - Например, у бинарной классификации $\mathcal{Y} = \{0, 1\}$.
  - Обучающее множество $S = \big((x_1, y_1), \dots, (x_m, y_m)\big)$ - конечное подмножество $\mathcal{X} \times \mathcal{Y}$
  - Образцы распределены в соответствии с некоторым распределением $\mathcal{X} \sim \mathcal{D}$
  - Есть (неизвестная) функция $f: \mathcal{X} \to \mathcal{Y}$, $y_i = f(x_i), \forall i$.

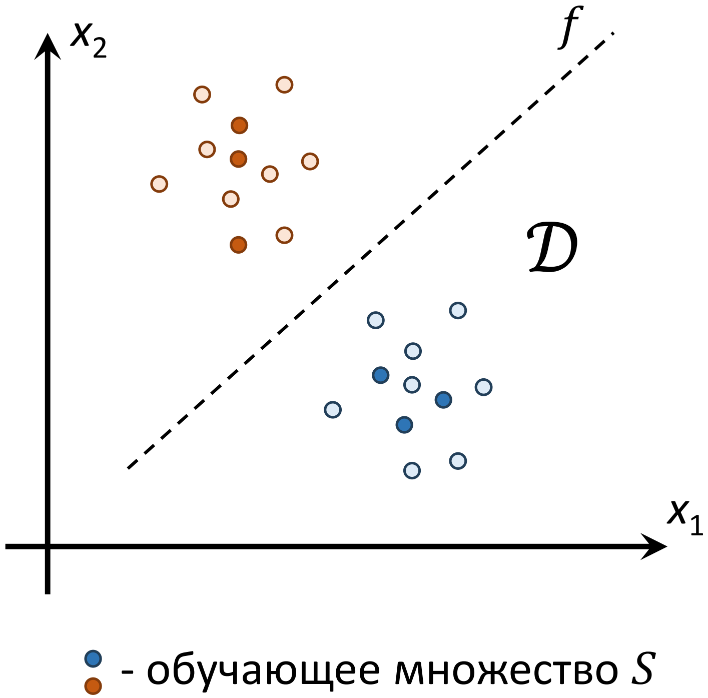


Постановка задачи обучения с учителем

- Выходные данные алгоритма обучения:
  - Модель (гипотеза, решающее правило) $h \in \mathcal{H}$, где $\mathcal{H}$ - пространство гипотез.
- Мера успеха (для классификации):
  - Вероятность того, что объект, извлеченный из распределения $\mathcal{D}$, будет неверно классифицирован моделью h :

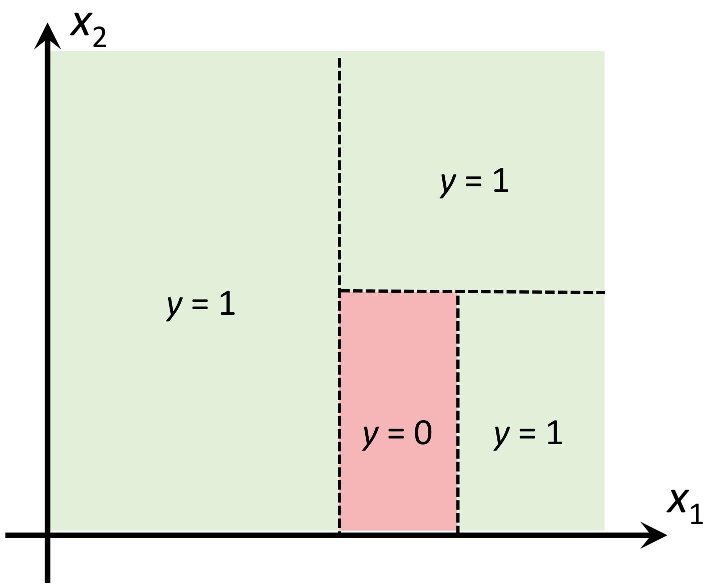
$$L_{\mathcal{D},f}(h) \stackrel{\text{def}}{=} \mathbb{P}_{x\sim\mathcal{D}}[h(x) \neq f(x)]$$

- Обучающий алгоритм <u>не знает</u> $\mathcal{D}$ и f .

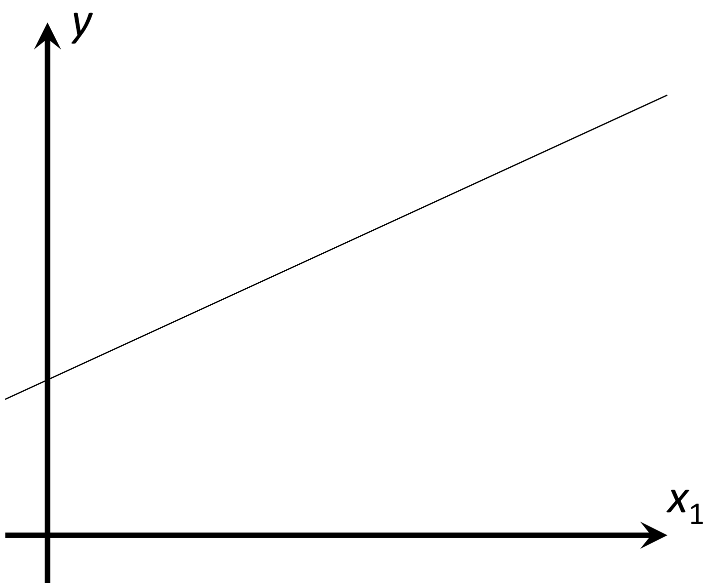


Виды моделей (гипотез)

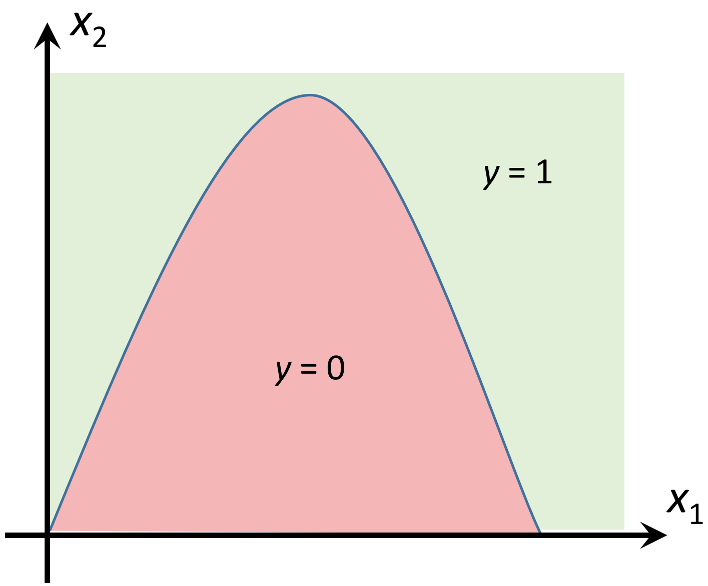
Пространство гипотез, задаваемое тремя делениями пространства признаков (деревья решений)



Пространство гипотез, задаваемое прямой линией

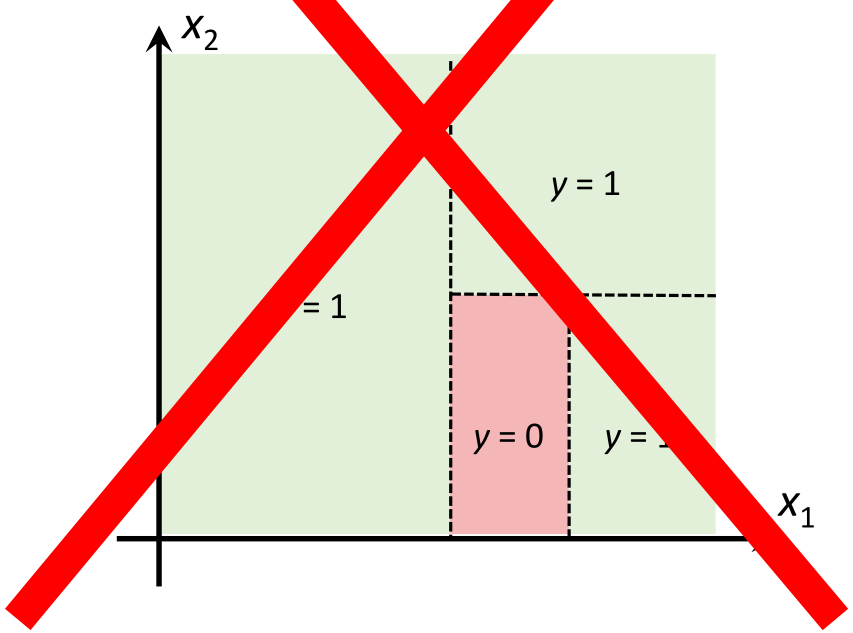


Пространство гипотез, задаваемое параболой, разграничивающей области с разным значением целевой переменной

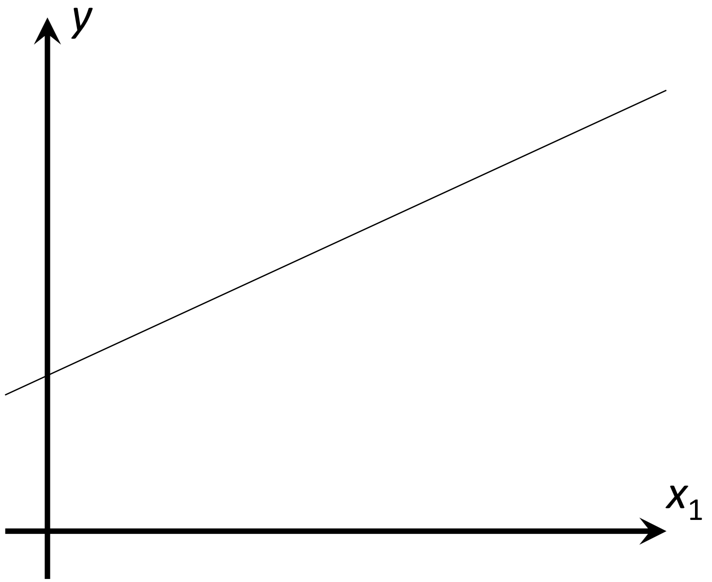


Виды моделей (гипотез). Параметрические модели

Пространство гипотез, задаваемое тремя делениями пространства признаков (деревья решений)

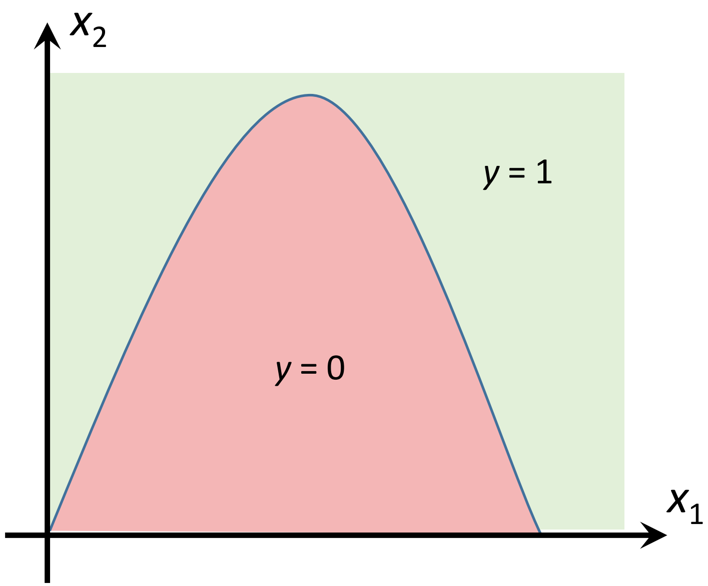


Пространство гипотез, задаваемое прямой линией



$$y = \theta_1 x + \theta_2$$

Пространство гипотез, задаваемое параболой, разграничивающей области с разным значением целевой переменной



$$y = sign[\theta_1 x_1^2 + \theta_2 x_2^2 + \theta_3 x_1 x_2 +$$
$$+ \theta_4 x_1 + \theta_5 x_2 + \theta_6]$$

Виды моделей (гипотез). Параметрические модели

$$y = h(x;\theta)$$

θ – вектор параметров модели.

Примеры конкретных моделей:

- $y = 3x + 5$
  - модель из пространства $\theta_1 x + \theta_2$ при $\theta = (3,\ 5)$
- $y = 3\sin(2x) + 17$
  - модель из пространства $\theta_1 \sin(\theta_2 x + \theta_3) + \theta_4$ при $\theta = (3, 2, 0, 17)$

Минимизация эмпирического риска и функция потерь

- Поскольку $\mathcal{D}$ и f неизвестны, то $L_{\mathcal{D},f}(h)$ оценить невозможно, поэтому истинная функция ошибок заменяется **эмпирической функцией ошибки** (для классификации):

$$L_S(h) \stackrel{\text{def}}{=} \frac{|\{i \in [m] : h(x_i) \neq y_i\}|}{m}, \text{ где } [m] = \{1, \ldots, m\}$$

Функции потерь

- Функция потерь характеризует качество гипотезы (модели) на обучающем множестве.
- Является обобщением понятия эмпирического риска (в зависимости от класса задачи обучения)

Функции потерь

- Задача регрессии - среднеквадратичная ошибка

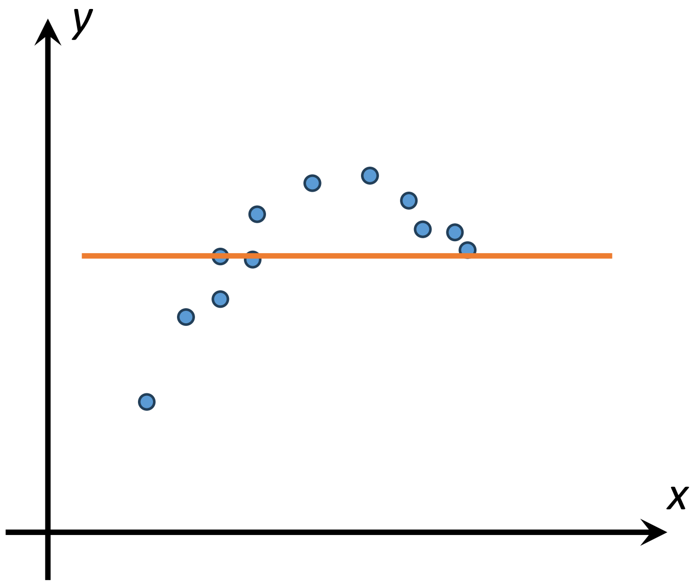
$$L_{MSD}(\hat{y}, y) = \frac{1}{m}\sum_{i=1}^{m}(\hat{y} - y)^2$$

- Задача бинарной классификации - бинарная энтропия:

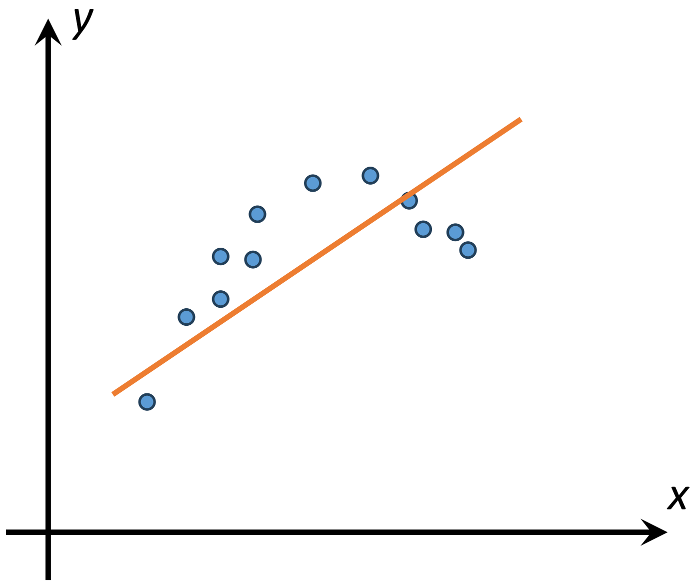
$$L_{BCE}(\hat{y}, y) = -\frac{1}{m}\sum_{i=1}^{m}(y \log \hat{y} + (1 - y) \log (1 - \hat{y}))$$

Мощность (выразительность) гипотез

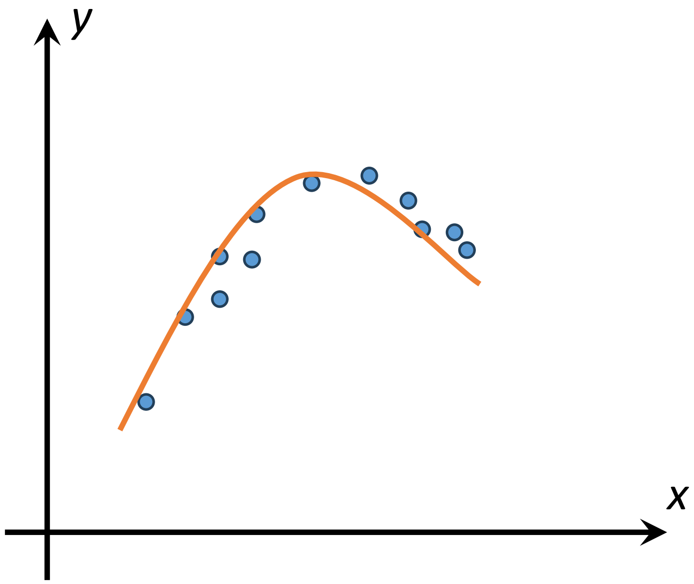
$y = \theta$



$y = \theta_1 x + \theta_2$



$y = \theta_1 x^2 + \theta_2 x + \theta_3$



Решение задачи. Оптимизация в пространстве параметров гипотезы

- За исключением простейших случаев (константа, линейный тренд и ряда других) о поиск глобального минимума функции потерь невозможен
- Решение: итеративные процедуры, основанные на градиентном спуске:

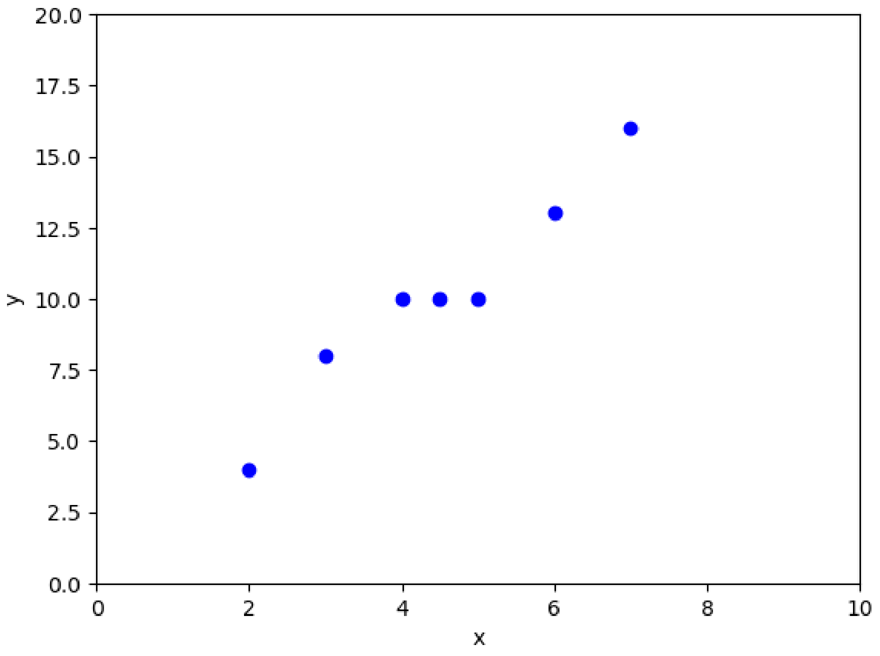
$$\hat{\theta}_i = \hat{\theta}_{i-1} - \alpha \frac{\partial L_S(h)}{\partial \theta}$$

Пример 1. Задача линейной регрессии

S =

x	y
2	4
3	8
4	10
4.5	10
5	10
6	13
7	16

- Дано: набор данных S, включающий значения одной (независимой) переменной *X* и соответствующие им значения другой переменной Y
- Класс гипотез:
  - Линейные функции вида $y = \theta_1 x + \theta_2$
  - Два параметра: θ_1, θ_2
- Функция потерь:
  - MSE: $L = \frac{1}{m}\sum_{i=1}^{m}(\hat{y}_i - y_i)^2$



Пример 1. Задача линейной регрессии

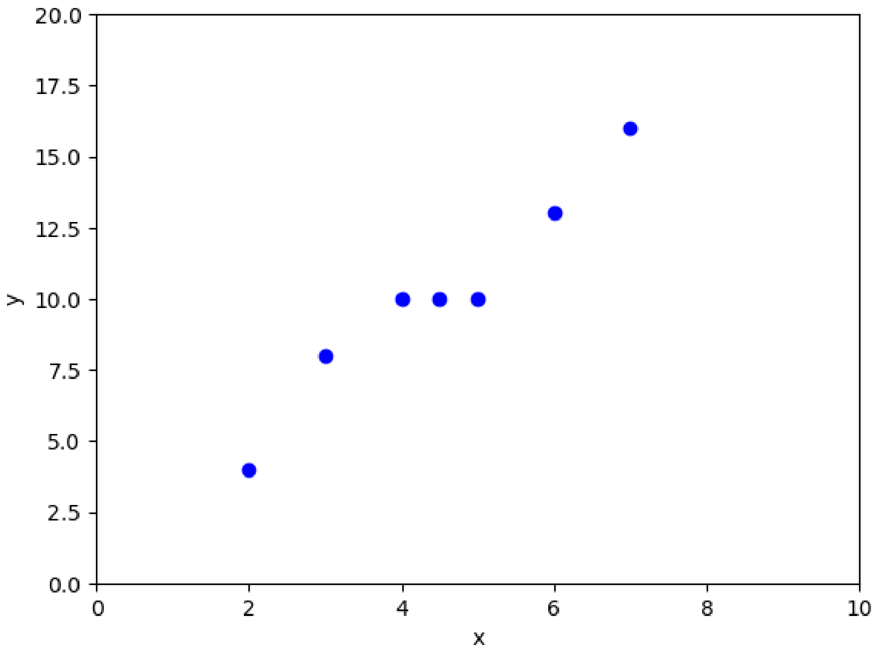
- Градиент функции потерь по параметрам θ_1, θ_2:

$$\frac{\partial L_S(h)}{\partial \theta_1} = \frac{\partial}{\partial \theta_1}\left(\frac{1}{m}\sum_{i=1}^{m}(\hat{y}_i - y_i)^2\right) =$$
$$= \frac{\partial}{\partial \theta_1}\left(\frac{1}{m}\sum_{i=1}^{m}((\theta_1 x + \theta_2) - y_i)^2\right) =$$
$$= \frac{1}{m}\left(\sum_{i=1}^{m} 2x\big((\theta_1 x + \theta_2) - y_i\big)\right) = \frac{2}{m}\left(\sum_{i=1}^{m} x\epsilon_i\right)$$

$$\frac{\partial L_S(h)}{\partial \theta_2} = \frac{\partial}{\partial \theta_2}\left(\frac{1}{m}\sum_{i=1}^{m}(\hat{y}_i - y_i)^2\right) =$$
$$= \frac{\partial}{\partial \theta_2}\left(\frac{1}{m}\sum_{i=1}^{m}\big((\theta_1 x + \theta_2) - y_i\big)^2\right) =$$
$$= \frac{1}{m}\left(\sum_{i=1}^{m} 2\big((\theta_1 x + \theta_2) - y_i\big)\right) = \frac{2}{m}\left(\sum_{i=1}^{m} \epsilon_i\right)$$

S =

x	y
2	4
3	8
4	10
4.5	10
5	10
6	13
7	16



Пример 1. Задача линейной регрессии

Начальные значения:

$$\theta_1 = 1.7$$
$$\theta_2 = 0.0$$

$$\frac{\partial L_S(h)}{\partial \theta} = \begin{pmatrix} \frac{\partial L_S(h)}{\partial \theta_1} \\ \frac{\partial L_S(h)}{\partial \theta_2} \end{pmatrix} = \begin{pmatrix} \frac{2}{m}\left(\sum_{i=1}^{m} x\epsilon_i\right) \\ \frac{2}{m}\left(\sum_{i=1}^{m} \epsilon_i\right) \end{pmatrix}$$

S =

x	y
2	4
3	8
4	10
4.5	10
5	10
6	13
7	16

$\hat{y}_i = \theta_1 x + \theta_2$	$\boldsymbol{\epsilon_i = \hat{y}_i - y_i}$	$x\epsilon_i$
3.4	-0.6	-1.2
5.1	-2.9	-8.7
6.8	-3.2	-12.8
7.65	-2.35	-10.575
8.5	-1.5	-7.5
10.2	-2.8	-16.8
11.9	-4.1	-28.7

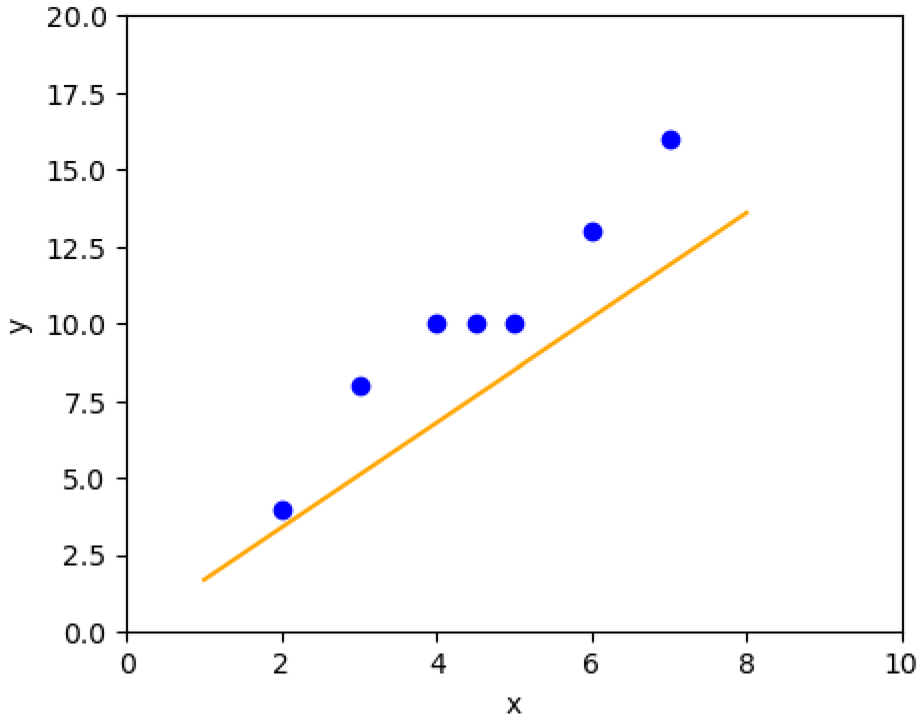
$$\frac{\partial L_S(h)}{\partial \theta} = \begin{pmatrix} -24.65 \\ -4.99 \end{pmatrix}$$

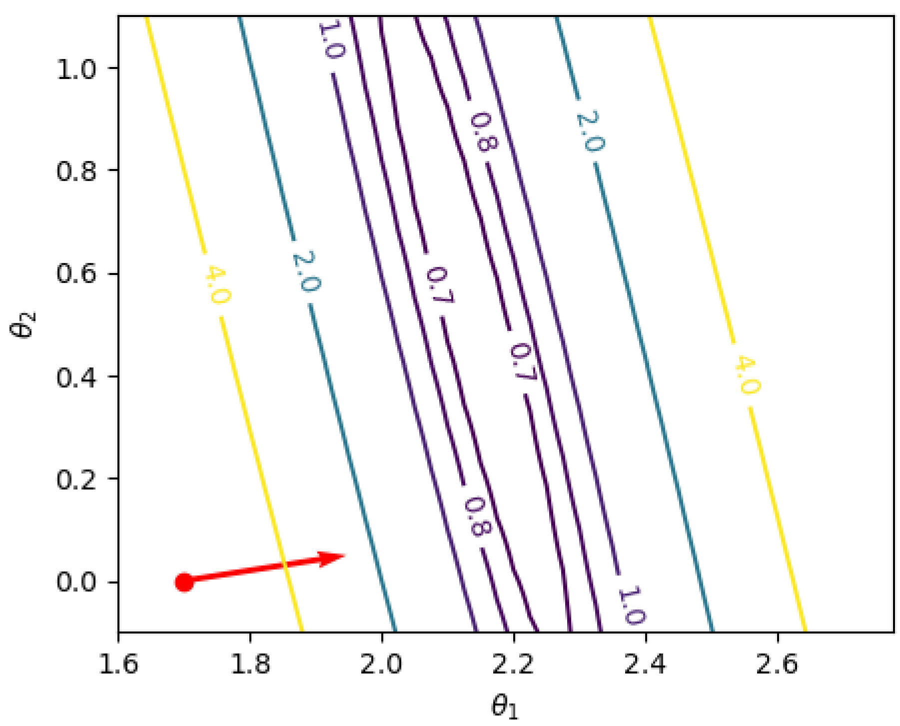
Линейная регрессия

$$y = 1.7x + 0$$

Loss = 7.35

$$\frac{\partial L}{\partial \theta} = (-24.65, -4.99)$$



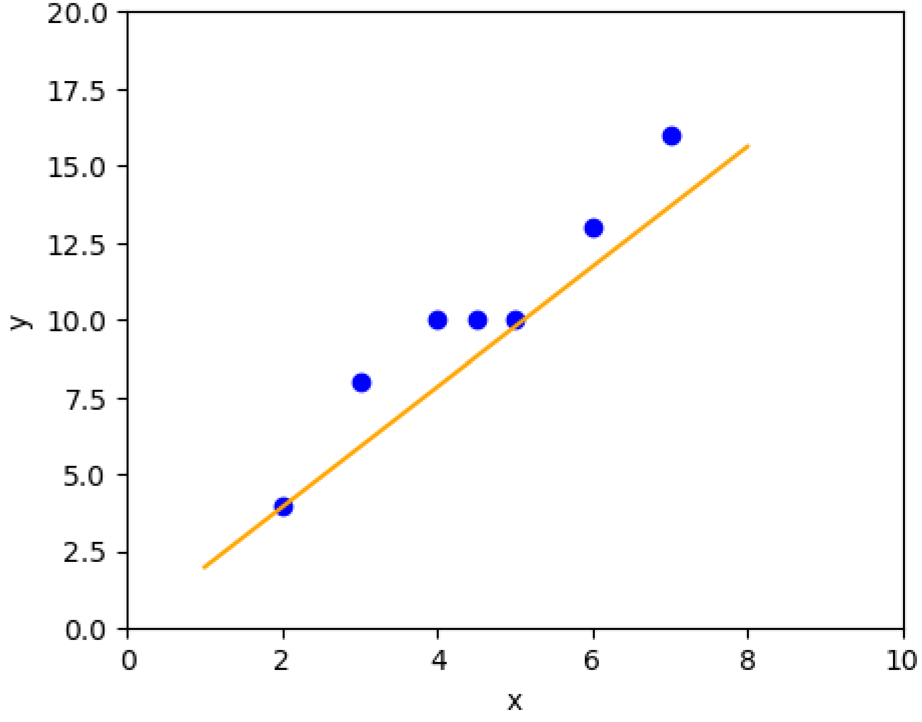


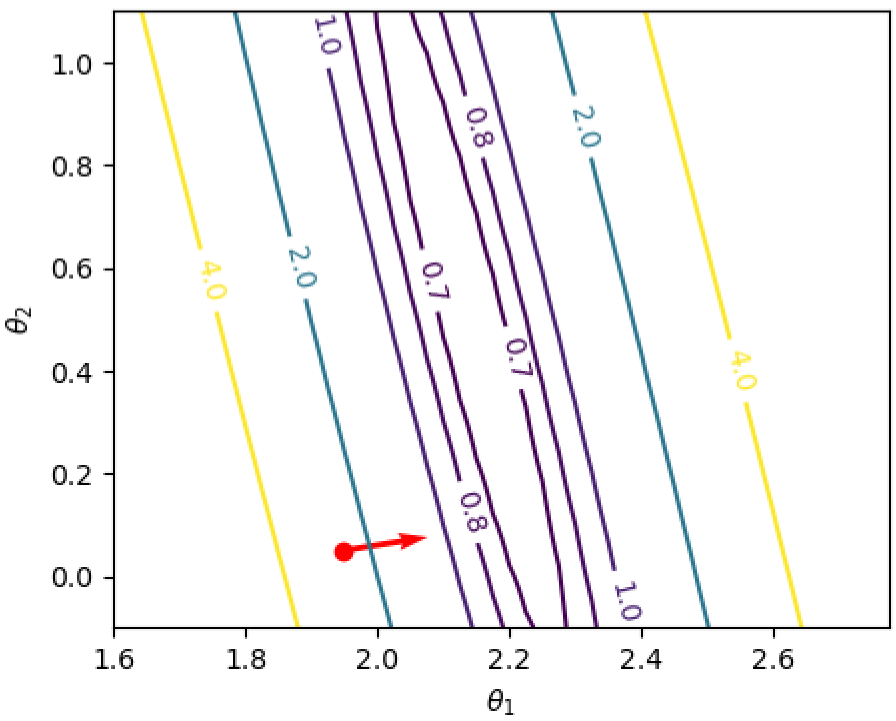
Линейная регрессия

$y = 1.94x + 0.05$

Loss = 2.518

$\frac{\partial L}{\partial \theta} = (-13, -2.67)$

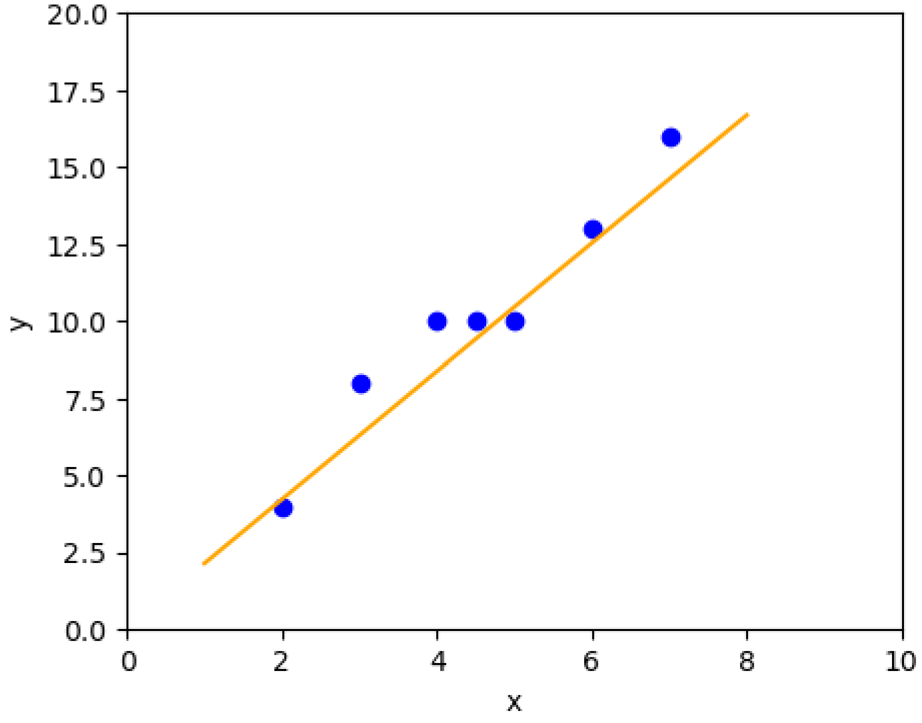




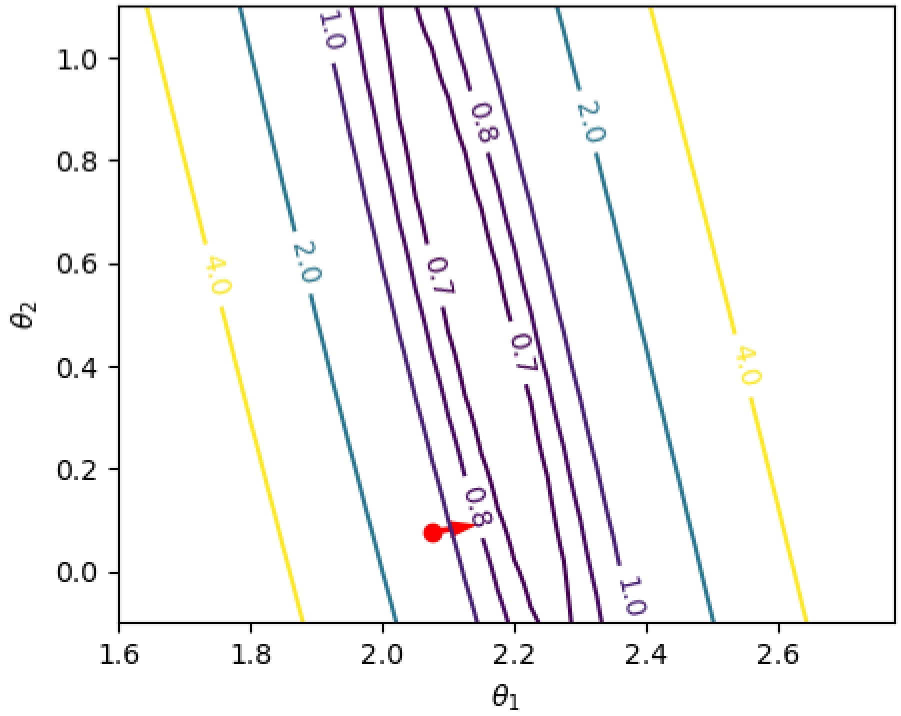
Линейная регрессия

$$y = 2.08x + 0.08$$

Loss = 1.176



$$\frac{\partial L}{\partial \theta} = (-6.84, -1.45)$$

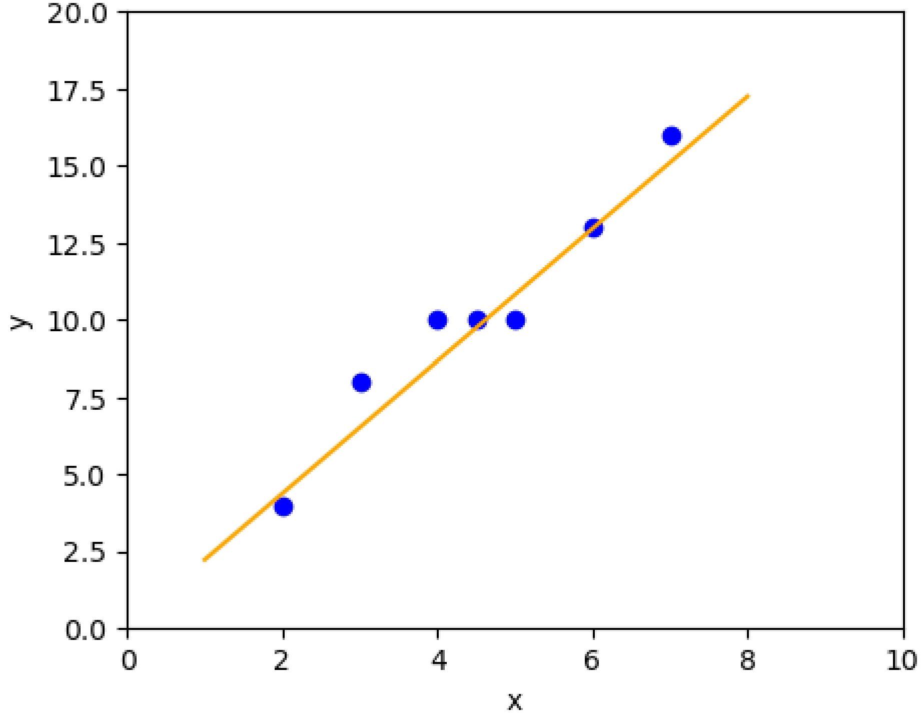


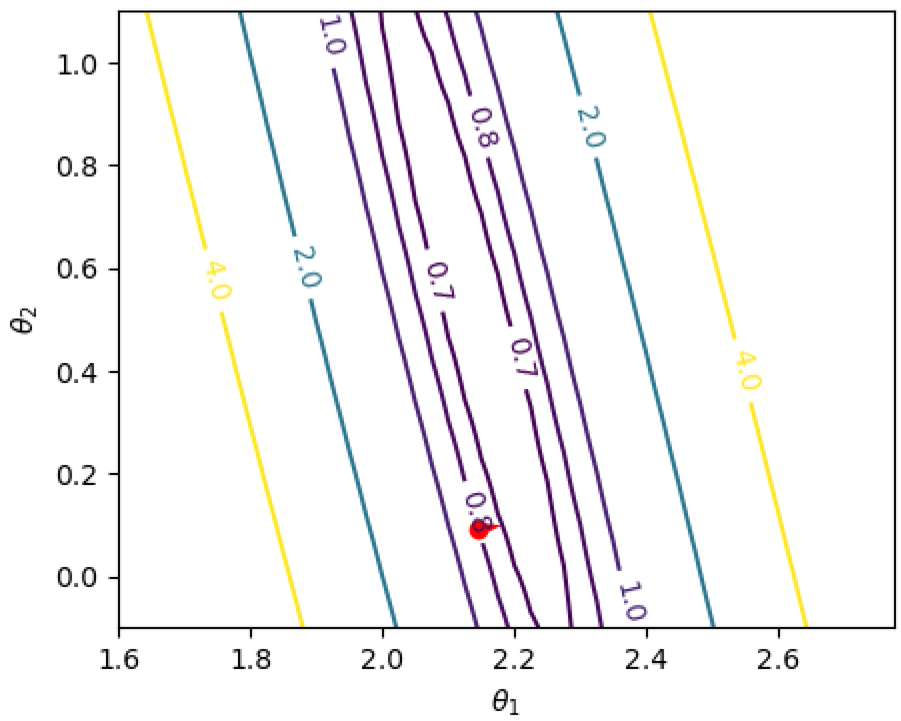
Линейная регрессия

$$y = 2.14x + 0.1$$

Loss = 0.8

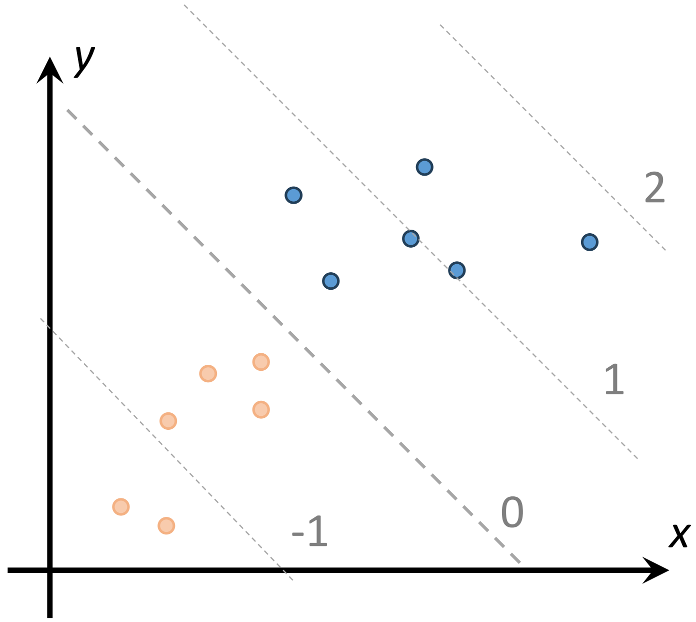
$$\frac{\partial L}{\partial \theta} = (-3.6, -0.8)$$





Пример 2. Задача логистической регрессии

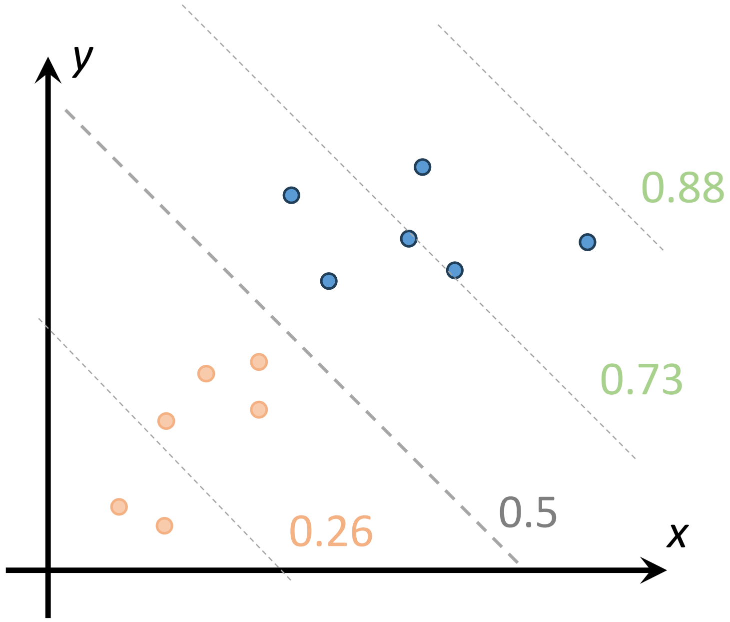
$y = x_1 + x_2 - 2$



Логистическая функция

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

$$(-\infty; +\infty) \rightarrow (0; 1)$$



Пример 2. Задача логистической регрессии

- Вид модели:

$$h(x;\theta) = \sigma(\theta x)$$

- Функция потерь:

$$L(\hat{y}, y) = -\frac{1}{m}\sum_{i=1}^{m}(y\log\hat{y} + (1-y)\log(1-\hat{y})\ )$$

- Градиент:

$$\frac{\partial}{\partial\theta}L(h(x;\theta), y) = -\frac{1}{m}\sum_{i=1}^{m}(h(x;\theta) - y)x$$

Разновидности градиентного спуска

- По количеству данных, на которых вычисляется градиент:
  - Один образец – стохастический градиентный спуск, SGD
  - Мини-пакетный (мини-батч) – подмножество фиксированного размера
  - Пакетный (батч) – всё обучающее множество
- По особенностям процедуры:
  - Моменты
  - Adam
  - …

Метрики

- Отличия функции потерь от метрики
  - Метрика (*более*) *интерпретируема* в терминах конкретной решаемой задачи, в большей степени выражает то, в какой степени *достигается цель* обучения модели
    - Попытка формализации степени достижения *цели*
  - Метрика не обязательно дифференцируема

Метрики. Регрессия

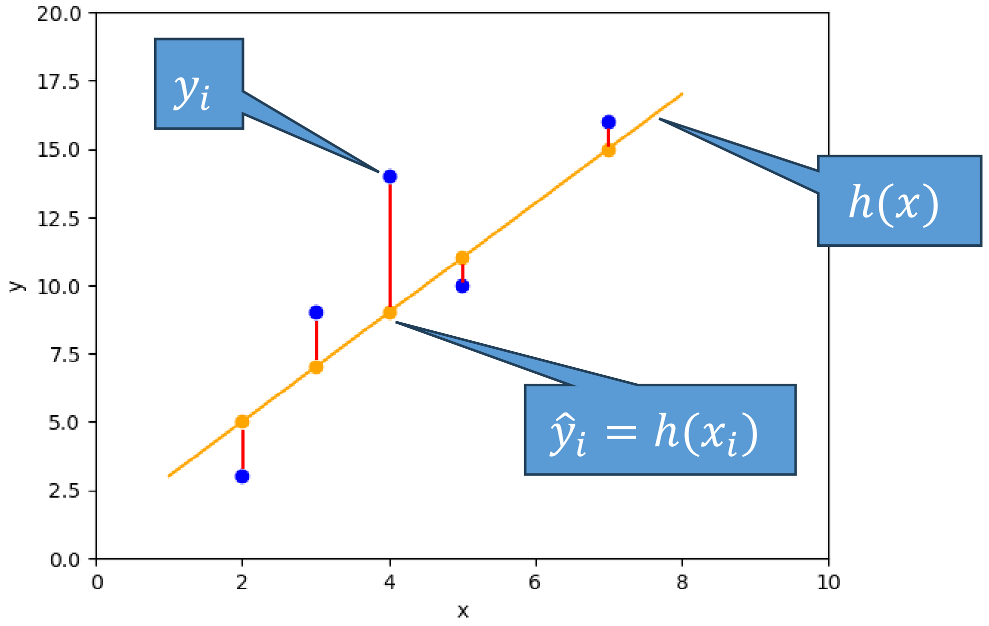
Средняя абсолютная ошибка,
Mean absolute error, MAE

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|h(x_i) - y_i|$$

Среднеквадратичная ошибка,
Root mean squared error, RMSE

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(h(x_i) - y_i)^2}$$

i	*x*	*y*	*h(x)*
1	2	3	5
2	3	9	7
3	4	14	9
4	5	10	11
5	7	16	15



MAE = (2 + 2 + 5 + 1 + 1)/5 = 2.2

$$\mathrm{RMSE} = \sqrt{4 + 4 + 25 + 1 + 1}/5 \approx 2.65$$

Метрики. Классификация

- Доля истинных результатов (Accuracy):

$$Acc = \frac{TP + TN}{TP + TN + FP + FN}$$

- Точность (Precision):

$$P = \frac{TP}{TP + FP}$$

- Полнота (Recall):

$$R = \frac{TP}{TP + FN}$$

- F1:

$$F1 = \frac{2 * P * R}{P + R}$$

Матрица ошибок (матрица неточностей)

Истинный класс

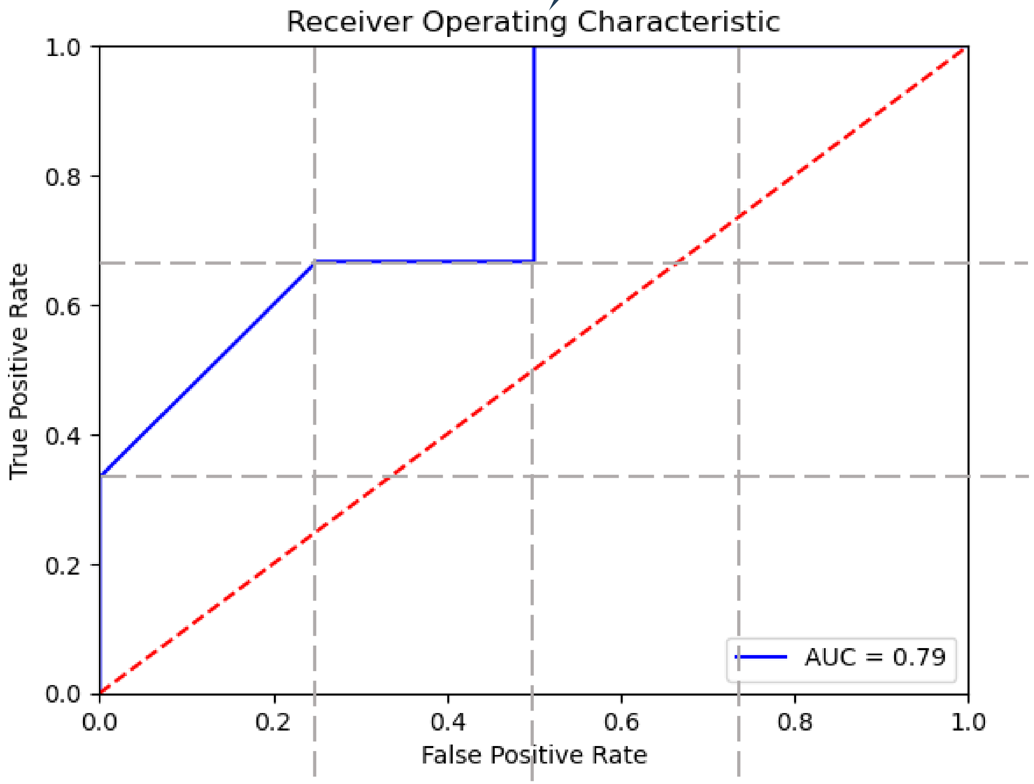
Результат модели

	Положительный, Positive	Отрицательный, Negative
Положительный, Positive	Истинно положительные, **TP**	Ложно положительные, ошибки 1-го рода, **FP**
Отрицательный, Negative	Ложно отрицательные, ошибка 2-го рода, **FN**	Истинно отрицательные, **TN**

Метрики. Классификация. ROC AUC

Какую долю истинно положительных образцов модель может «вернуть», если ей «разрешить» заданную долю ложно положительных

№	Предсказание	Истинный класс
2	0.9	1
4	0.7	1
7	0.7	0
5	0.6	0
6	0.5	1
3	0.1	0
1	0.0	0



Количество положительных в наборе

Количество отрицательных в наборе

Интересные свойства:

- не зависит от порога классификации
- вероятности принадлежности классам не обязаны быть [0; 1], оценивает качество *упорядочивания*

Метрики. Прочие

scikit-learn | Install | User Guide | API | Examples | Community | More | Go
Prev | Up | Next
scikit-learn 1.3.1
Other versions
Please cite us if you use the software.
API Reference
sklearn.base: Base classes and utility functions
sklearn.calibration: Probability Calibration
sklearn.cluster: Clustering
sklearn.compose: Composite Estimators
sklearn.covariance: Covariance Estimators
sklearn.cross_decomposition: Cross decomposition
sklearn.datasets: Datasets
sklearn.decomposition: Matrix Decomposition
sklearn.discriminant_analysis: Discriminant Analysis
sklearn.dummy: Dummy estimators
sklearn.ensemble: Ensemble Methods
sklearn.exceptions: Exceptions and warnings
sklearn.experimental: Experimental
sklearn.feature_extraction: Feature Extraction
sklearn.feature_selection: Feature Selection
sklearn.gaussian_process: Gaussian Processes
sklearn.impute: Impute
sklearn.inspection: Inspection
sklearn.isotonic: Isotonic regression
sklearn.kernel_approximation: Kernel Approximation
sklearn.kernel_ridge: Kernel Ridge Regression
sklearn.linear_model: Linear Models
sklearn.manifold: Manifold Learning
sklearn.metrics: Metrics
sklearn.mixture: Gaussian Mixture Models
sklearn.model_selection: Model Selection
sklearn.multiclass: Multiclass classification
sklearn.multioutput: Multioutput regression and classification
sklearn.naive_bayes: Naive Bayes
sklearn.neighbors: Nearest Neighbors
sklearn.neural_network: Neural network models

Classification metrics

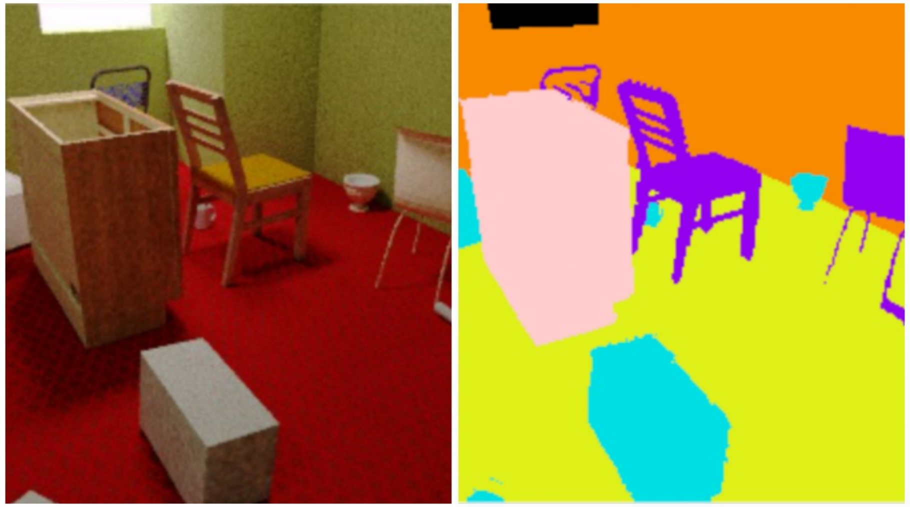
See the Classification metrics section of the user guide for further details.

metrics.accuracy_score(y_true, y_pred, *[, ...])	Accuracy classification score.
metrics.auc(x, y)	Compute Area Under the Curve (AUC) using the trapezoidal rule.
metrics.average_precision_score(y_true, ...)	Compute average precision (AP) from prediction scores.
metrics.balanced_accuracy_score(y_true, ...)	Compute the balanced accuracy.
metrics.brier_score_loss(y_true, y_prob, *)	Compute the Brier score loss.
metrics.class_likelihood_ratios(y_true, ...)	Compute binary classification positive and negative likelihood ratios.
metrics.classification_report(y_true, y_pred, *)	Build a text report showing the main classification metrics.
metrics.cohen_kappa_score(y1, y2, *[, ...])	Compute Cohen's kappa: a statistic that measures inter-annotator agreement.
metrics.confusion_matrix(y_true, y_pred, *)	Compute confusion matrix to evaluate the accuracy of a classification.
metrics.dcg_score(y_true, y_score, *[, k, ...])	Compute Discounted Cumulative Gain.
metrics.det_curve(y_true, y_score[, ...])	Compute error rates for different probability thresholds.
metrics.f1_score(y_true, y_pred, *[, ...])	Compute the F1 score, also known as balanced F-score or F-measure.
metrics.fbeta_score(y_true, y_pred, *, beta)	Compute the F-beta score.
metrics.hamming_loss(y_true, y_pred, *[, ...])	Compute the average Hamming loss.
metrics.hinge_loss(y_true, pred_decision, *)	Average hinge loss (non-regularized).
metrics.jaccard_score(y_true, y_pred, *[, ...])	Jaccard similarity coefficient score.
metrics.log_loss(y_true, y_pred, *[, eps, ...])	Log loss, aka logistic loss or cross-entropy loss.
metrics.matthews_corrcoef(y_true, y_pred, *)	Compute the Matthews correlation coefficient (MCC).
metrics.multilabel_confusion_matrix(y_true, ...)	Compute a confusion matrix for each class or sample.
metrics.ndcg_score(y_true, y_score, *[, k, ...])	Compute Normalized Discounted Cumulative Gain.
metrics.precision_recall_curve(y_true, ...)	Compute precision-recall pairs for different probability thresholds.
metrics.precision_recall_fscore_support(...)	Compute precision, recall, F-measure and support for each class.
metrics.precision_score(y_true, y_pred, *[, ...])	Compute the precision.
metrics.recall_score(y_true, y_pred, *[, ...])	Compute the recall.
metrics.roc_auc_score(y_true, y_score, *[, ...])	Compute Area Under the Receiver Operating Characteristic Curve (ROC AUC) from prediction scores.
metrics.roc_curve(y_true, y_score, *[, ...])	Compute Receiver operating characteristic (ROC).
metrics.top_k_accuracy_score(y_true, y_score, *)	Top-k Accuracy classification score.
metrics.zero_one_loss(y_true, y_pred, *[, ...])	Zero-one classification loss.

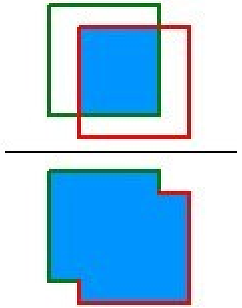
Regression metrics

See the Regression metrics section of the user guide for further details.

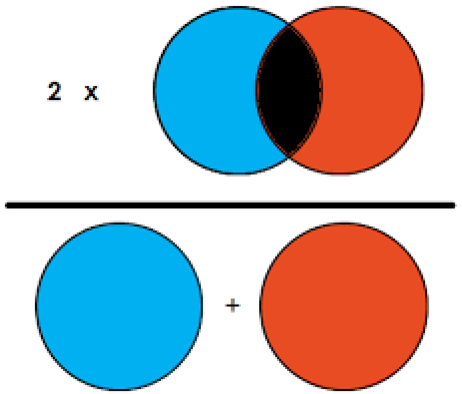
metrics.explained_variance_score(y_true, ...)	Explained variance regression score function.
metrics.max_error(y_true, y_pred)	The max_error metric calculates the maximum residual error.
metrics.mean_absolute_error(y_true, y_pred, *)	Mean absolute error regression loss.
metrics.mean_squared_error(y_true, y_pred, *)	Mean squared error regression loss.
metrics.mean_squared_log_error(y_true, y_pred, *)	Mean squared logarithmic error regression loss.
metrics.median_absolute_error(y_true, y_pred, *)	Median absolute error regression loss.
metrics.mean_absolute_percentage_error(...)	Mean absolute percentage error (MAPE) regression loss.
metrics.r2_score(y_true, y_pred, *[, ...])	R^2 (coefficient of determination) regression score function.
metrics.mean_poisson_deviance(y_true, y_pred, *)	Mean Poisson deviance regression loss.
metrics.mean_gamma_deviance(y_true, y_pred, *)	Mean Gamma deviance regression loss.
metrics.mean_tweedie_deviance(y_true, y_pred, *)	Mean Tweedie deviance regression loss.
metrics.d2_tweedie_score(y_true, y_pred, *)	D^2 regression score function, fraction of Tweedie deviance explained.
metrics.mean_pinball_loss(y_true, y_pred, *)	Pinball loss for quantile regression.
metrics.d2_pinball_score(y_true, y_pred, *)	D^2 regression score function, fraction of pinball loss explained.
metrics.d2_absolute_error_score(y_true, ...)	D^2 regression score function, fraction of absolute error explained.



$$IOU = \frac{\text{area of overlap}}{\text{area of union}} =$$



DICE =

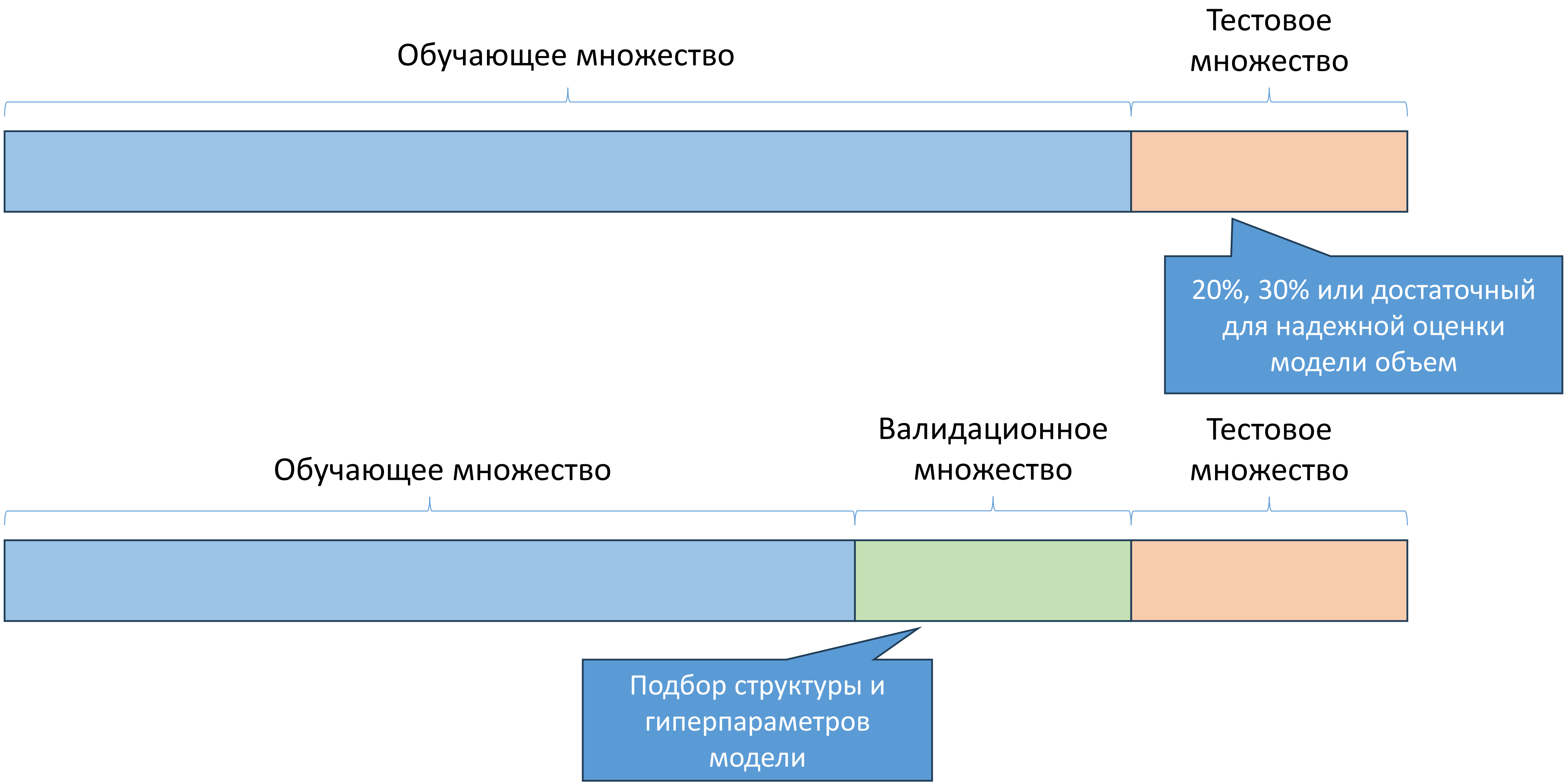


Валидация

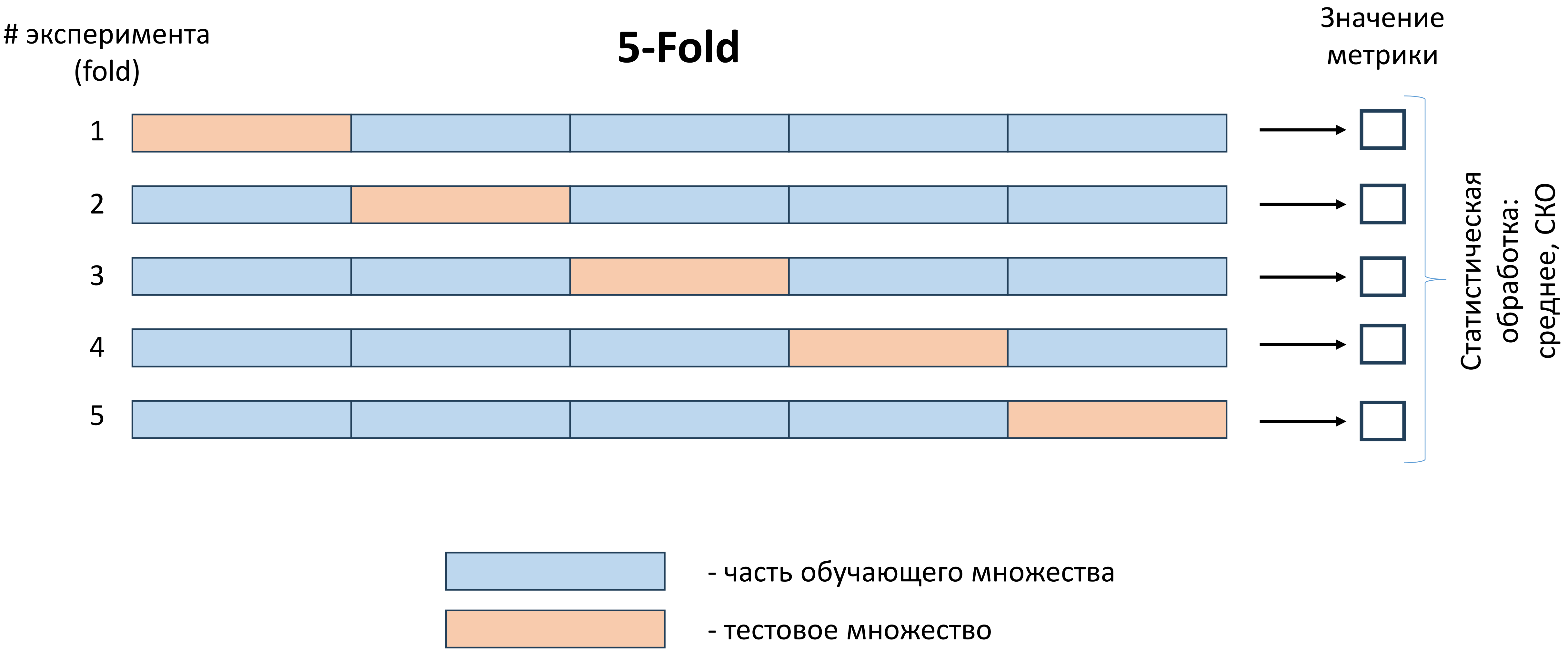
- Валидация – проверка характеристик модели, оценка в какой степени модель удовлетворяет требованиям
  - В частности, на тех данных, которые *не были* использованы при обучении

- Основные схемы валидации:
  - Отложенная выборка
    - Обучающая и тестовая выборки
    - Обучающая, валидационная и тестовая выборки
  - Перекрестная валидация (K-Fold)

Валидация. Отложенные выборки



Валидация. Перекрестная валидация (кросс-валидация)



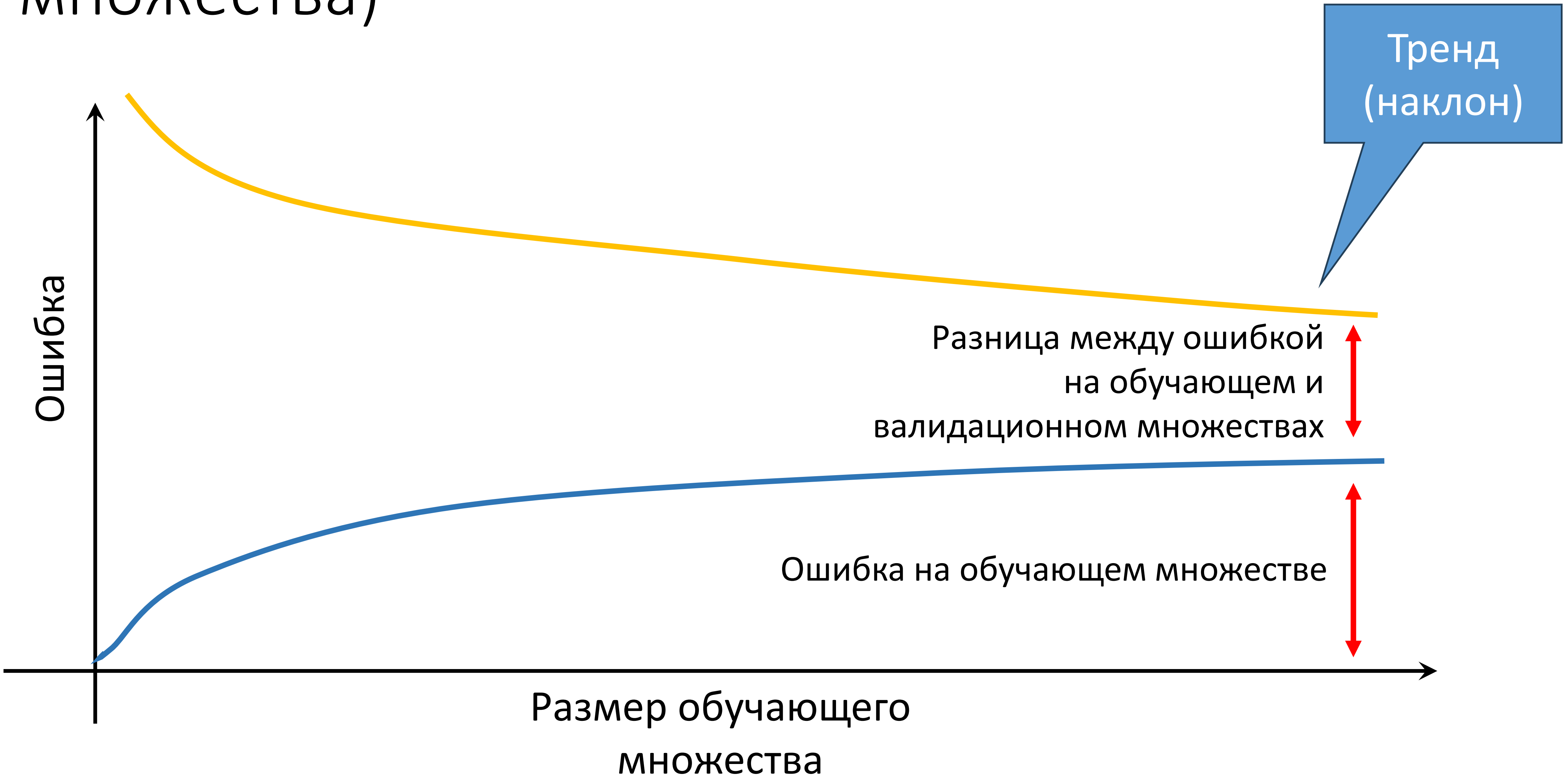
Валидация. Важные аспекты разбиения

- Соответствие тестовой выборки целевому использованию модели
- Достаточный размер
- Отсутствие «утечек» в тестовую выборку
  - Тот же пациент
  - Тот же период (во временн*ы*х моделях)
- Нужна ли группировка?
- Нужна ли стратификация?

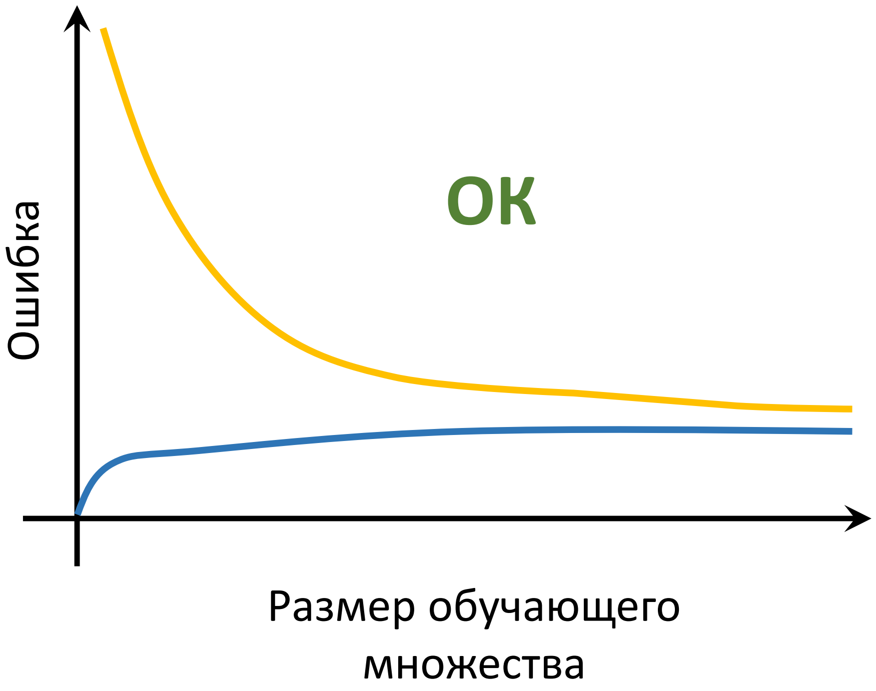
Диагностика моделей. Кривые обучения

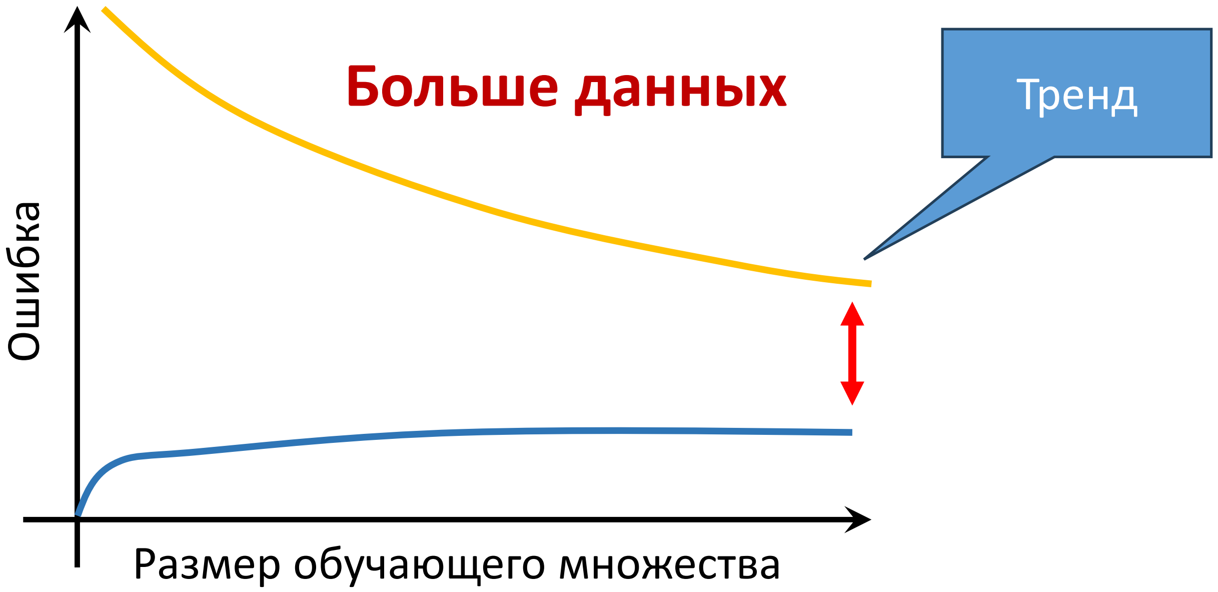
- Два вида:
  - От размера обучающего множества
  - От количества итераций обучения

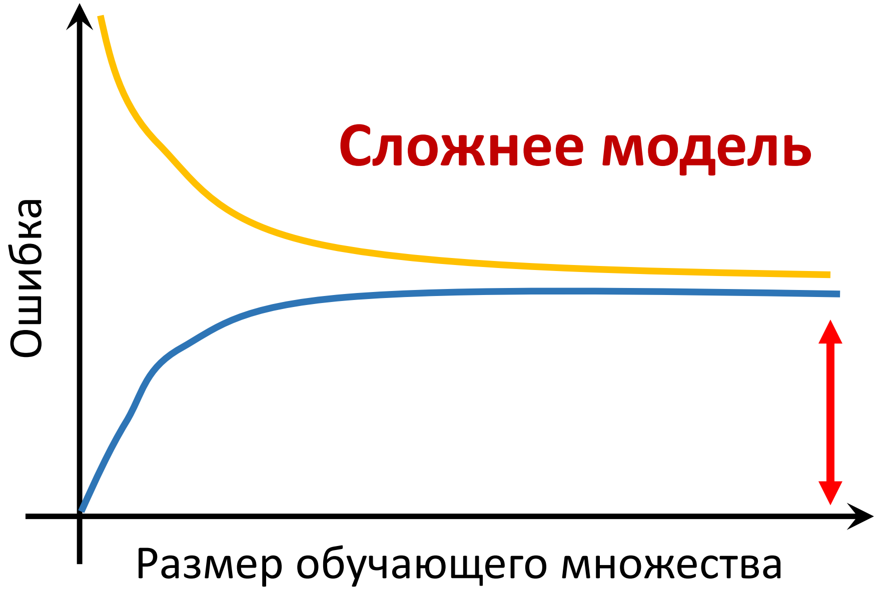
Кривые обучения (от размера обучающего множества)



Кривые обучения (от размера обучающего множества)

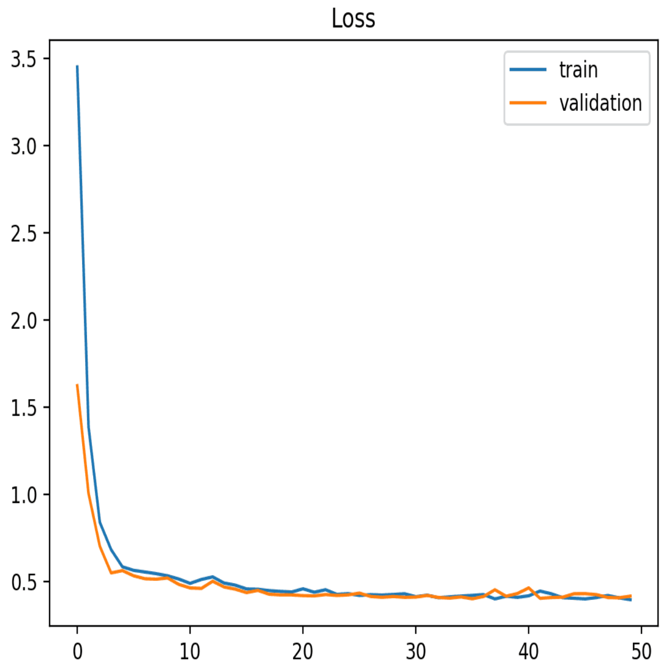




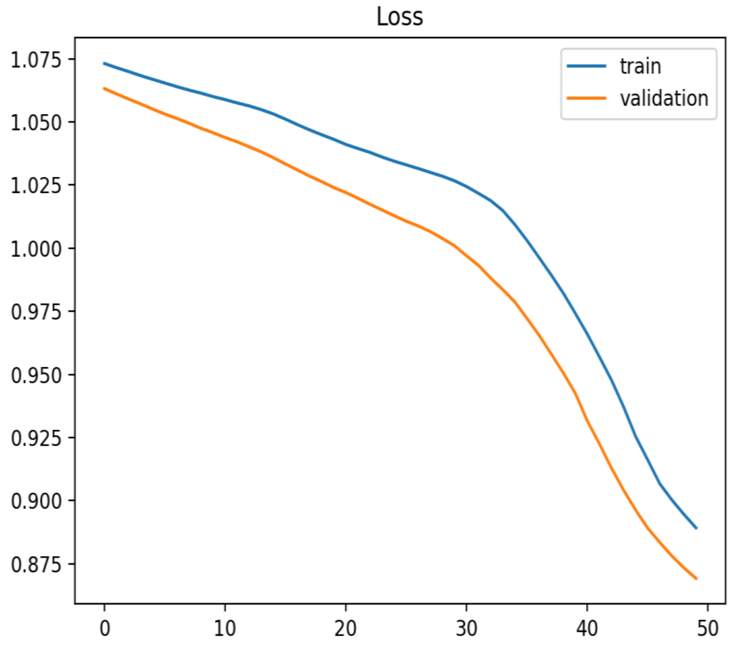


Кривые обучения (от количества итераций)

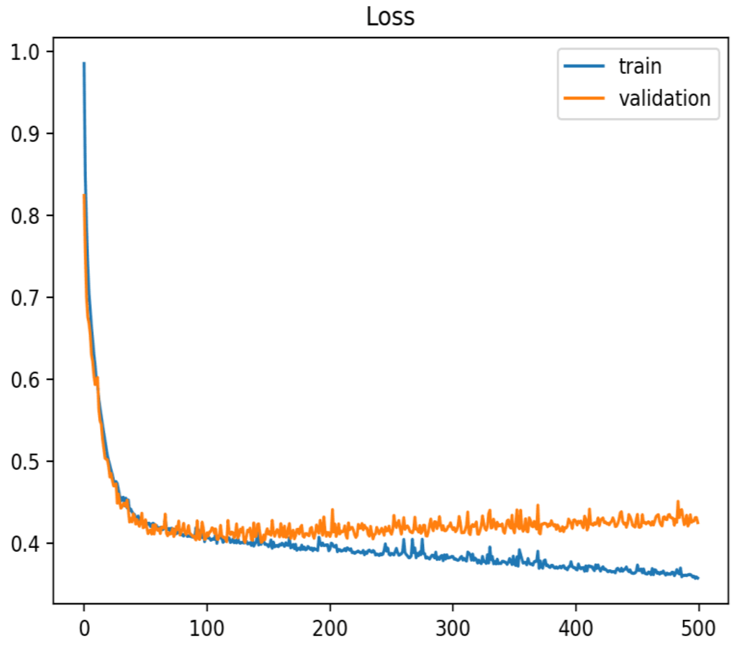
ОК



Недообученная модель
(продолжить обучение)

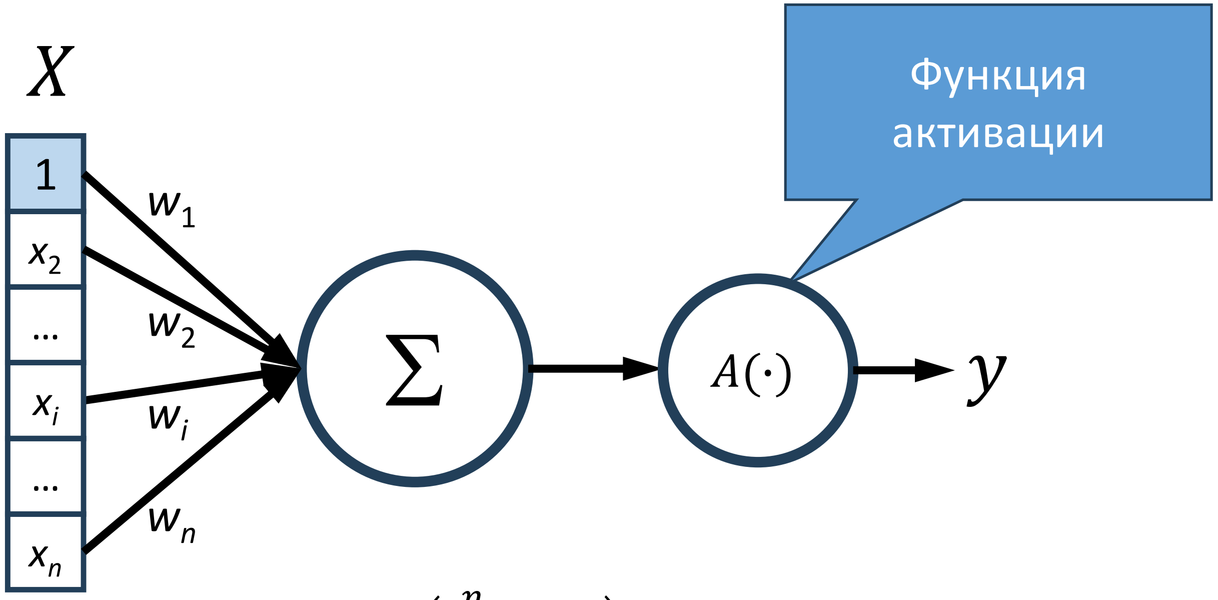


Переобученная модель, оверфиттинг,
(ранний останов, увеличить размер обучающего множества, упростить модель)



Тема 10. Обучение нейронных сетей

Модель одного искусственного нейрона

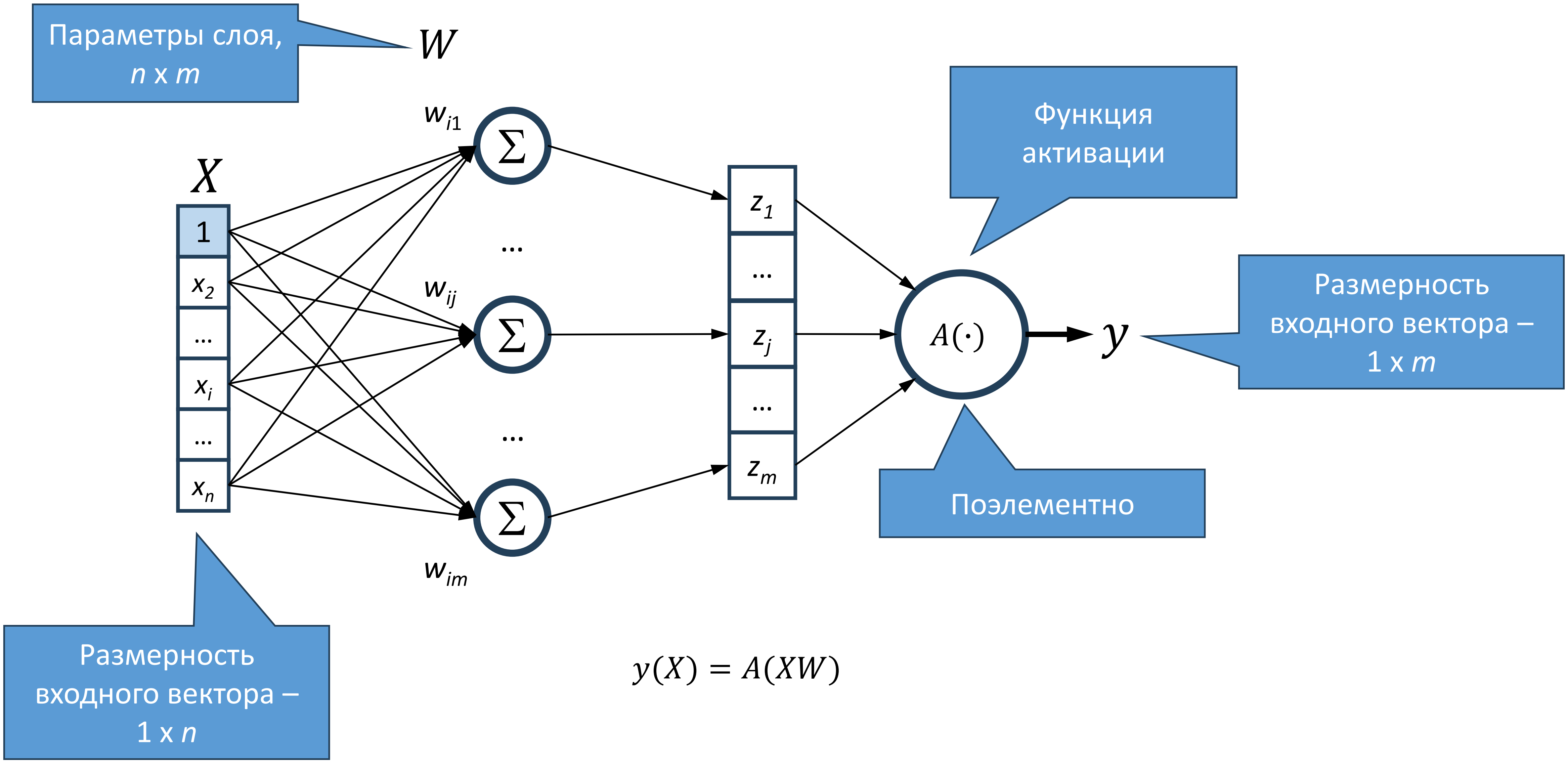


$$y(X) = A\left(\sum_{i=1}^{n} w_i x_i\right) = A(XW^T)$$

Вектор-строка

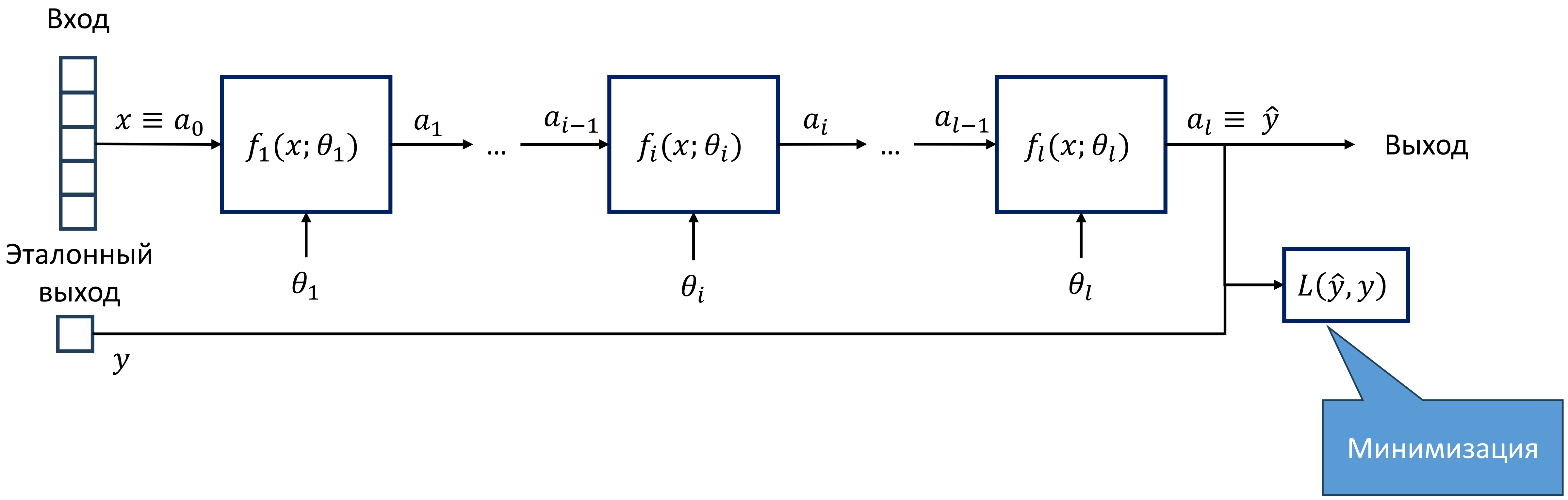
- Функция активации – источник *нелинейности*
- Без $A(.)$ – линейная регрессия
- При $A(x) \equiv \sigma(x) \equiv (1 + e^{-x})^{-1}$ – логистическая регрессия (только линейная разделимость)

Слой искусственных нейронов

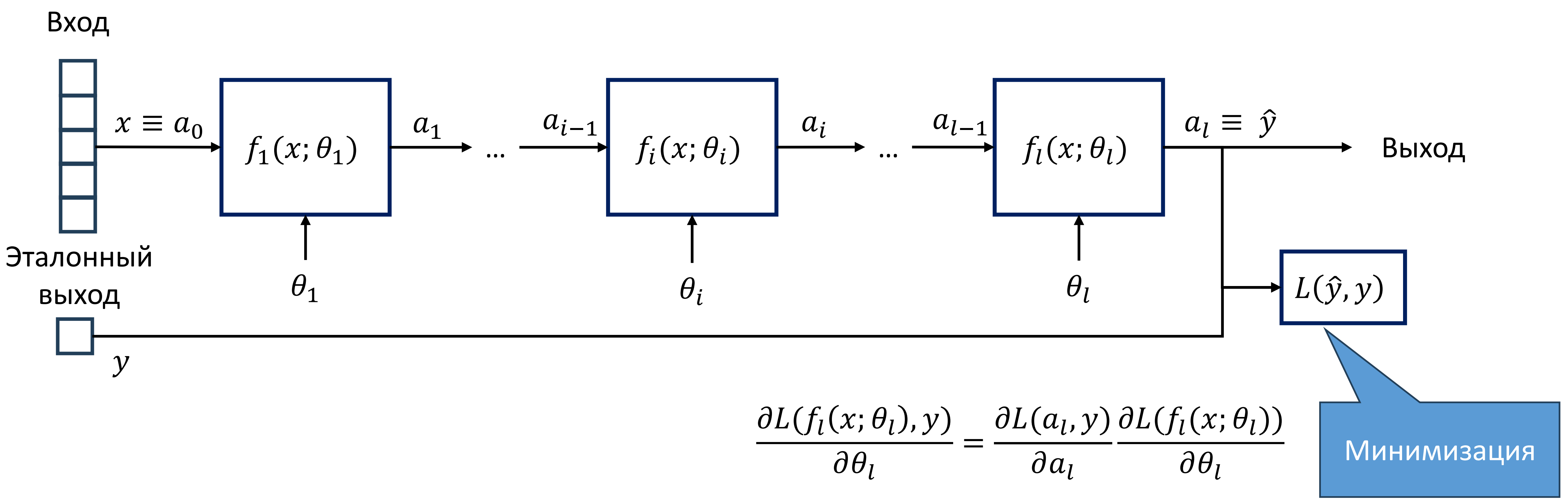


$$y(X) = A(XW)$$

Нейронная сеть



Нейронная сеть



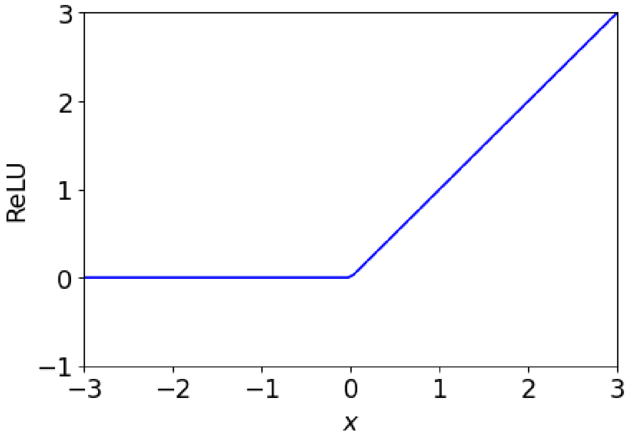
$$\frac{\partial L(f_l(x;\theta_l), y)}{\partial \theta_l} = \frac{\partial L(a_l, y)}{\partial a_l} \frac{\partial L(f_l(x;\theta_l))}{\partial \theta_l}$$

Функции активации

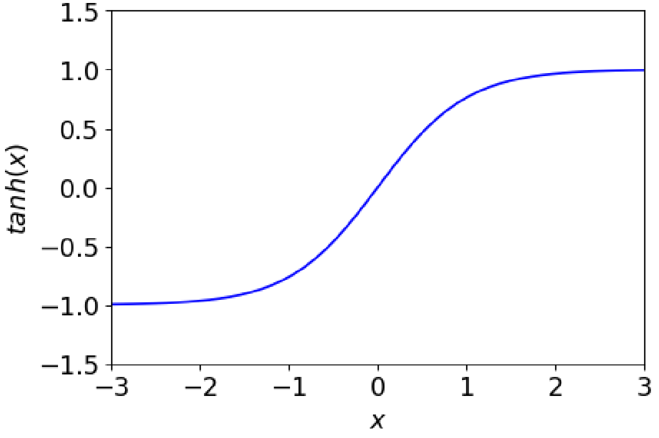
Внутри сети (скрытые слои)

$$a(x) = \max(x, 0)$$

См. также:
Leaky ReLU, ELU и др.



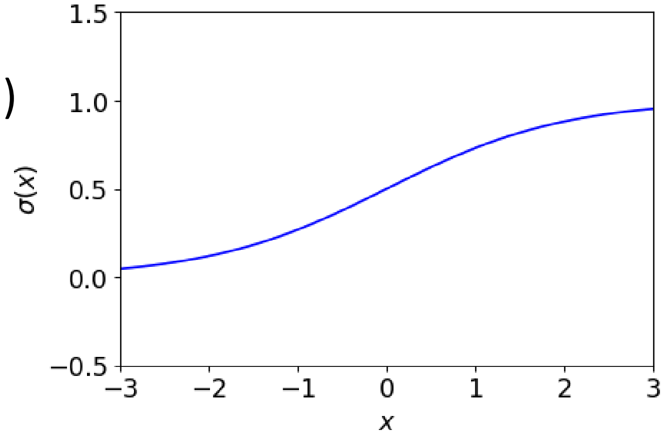
$$a(x) = \tanh(x)$$



Последний слой

Сигмоидальная
(бинарная классификация)

$$a(x) = \frac{1}{1 + e^{-x}}$$



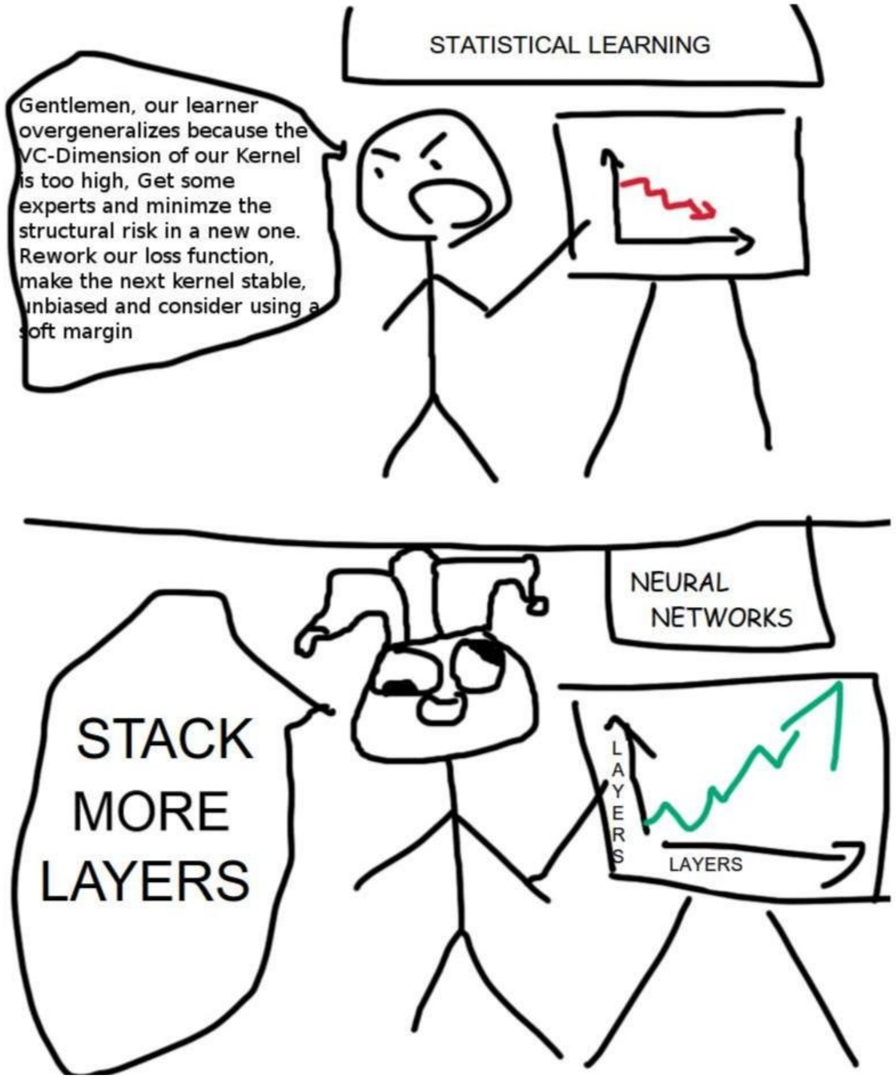
Softmax (многоклассовая классификация)

$$\begin{pmatrix} 2 \\ -1 \\ 3 \end{pmatrix} \rightarrow \frac{e^{x_i}}{\sum_j e^{x_j}} \rightarrow \begin{pmatrix} 0.27 \\ 0.01 \\ 0.72 \end{pmatrix}$$

Сложность и выразительность сети

- «Выразительность» сети:
  - Количество слоев
  - Количество нейронов в слое

- Теоретически:
  - Достаточно одного скрытого слоя
- На практике:
  - Можно обойтись меньшим количеством нейронов, если использовать больше слоёв
  - Но: при очень глубоких сетях (>10) возможны проблемы
    - Возможные решения: residual networks, skip connections



Регуляризация. Исключение/дропаут (Dropout)

- Цель регуляризации:
  - Уменьшение переобучения
- Весовая регуляризация:
  - Наложение штрафа на величину весов сети:

$$L_r(x, y; \theta) = L(x, y; \theta) + \gamma R(\theta)$$

$\|\theta\|_1$ - абсолютные значения

$\|\theta\|_2^2$- сумма квадратов

- Исключение/дропаут:
  - Предотвращение образования сильных связей между нейронами в соседних слоях посредством выбрасывания (зануления) выходов слоя с определенной вероятностью
    - Режим обучения и режим «вывода» (предсказания)

Пакетная нормализация (batch normalization)

- Идея:
  - масштабирование данных, приведение их в «удобный» диапазон

Вход: Пакет $B = \{x_1, \ldots, x_m\}$, параметры γ, β, константа ϵ.

Выход: $\{y_1, \ldots, y_m\}$

$$\mu_B = \frac{1}{m}\sum_{i=1}^{m} x_i$$

$$\sigma_B^2 = \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_B)$$

$$\hat{x}_i = (x_i - \mu_B)/\sqrt{\sigma_B^2 + \epsilon}$$

$$y_i = \gamma\hat{x}_i + \beta$$

Обучаются

Пакетная нормализация (batch normalization)

- Полезные свойства:
  - достигается более быстрая сходимость моделей, несмотря на выполнение дополнительных вычислений;
  - пакетная нормализация позволяет каждому слою сети обучаться более независимо от других слоев;
  - становится возможным использование более высокого темпа обучения, так как пакетная нормализация гарантирует, что выходы узлов нейронной сети не будут иметь слишком больших или малых значений;
  - пакетная нормализация в каком-то смысле также является механизмом регуляризации: данный метод привносит в выходы узлов скрытых слоев некоторый шум, аналогично методу dropout;
  - модели становятся менее чувствительны к начальной инициализации весов.

Литература

- Основное:
  - Рассел С., Норвиг П. Искусственный интеллект. Современный подход. 4-е издание
- Дополнительно:
  - Dive into Deep Learning: http://d2l.ai
  - Coursera/DeepLearning.AI: Deep Learning Specialization (Andrew Ng)
  - Блог А. Дьяконова: https://alexanderdyakonov.wordpress.com
    - ROC AUC и вообще много хороших материалов:
    - https://alexanderdyakonov.wordpress.com/2017/07/28/auc-roc-площадь-под-кривой-ошибок/