

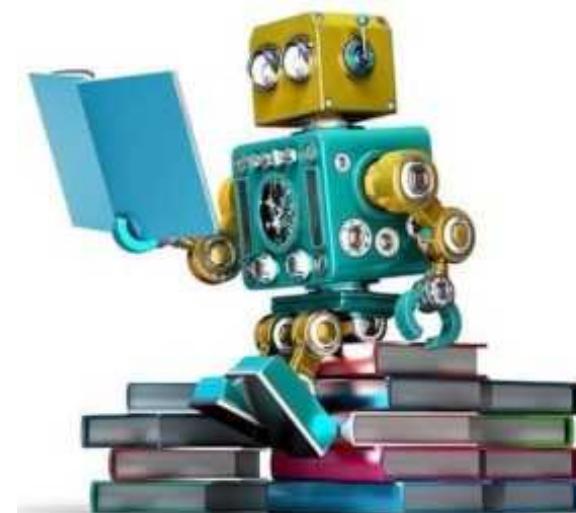
Факультет «Информатика и системы управления»  
Кафедра ИУ7 «Программное обеспечение ЭВМ и информационные технологии»

# Методы машинного обучения

*Солодовников Владимир Игоревич  
к.т.н.*

# Основные понятия, цели, задачи, методы

- Постановка задачи обучения;
- Типы и примеры задач;
- Открытые наборы данных для обучения;
- Предобработка (подготовка) исходных данных;
- Классификация методов машинного обучения;
- Выбор метода и настройка гиперпараметров;
- Алгоритмы обучения моделей;
- Метрики оценки качества полученной модели;
- Интерпретация полученного результата.



# Knowledge Discovery in Databases (KDD)

## Обнаружение знаний в базах данных



Основоположниками концепции KDD считаются Пятецкий-Шапиро (*Gregory I. Piatetsky-Shapiro - слева*) и Усама М. Файад (*Usama M. Fayyad - справа*).

*Data scientists and the co-founders of the KDD conferences and the Association for Computing Machinery SIGKDD group for Knowledge Discovery, Data Mining and Data Science.*



**KDD** представляет собой нетривиальный процесс обнаружения корректных, новых, потенциально полезных и интерпретируемых шаблонов в больших массивах данных.

Данные — множество фактов предметной области, представленных в виде записей базы данных.

Шаблон — это выражение на некотором языке, описывающее подмножество данных или применяемую к нему модель, т.е. поиск шаблонов подразумевает подгонку моделей к данным, обнаружение в них зависимостей, закономерностей и структур.

Процесс — многоэтапная, итеративная процедура, включающая подготовку данных, построение моделей, оценку и уточнение результатов.

KDD не предписывает, какие методы и алгоритмы обработки следует использовать при решении конкретной задачи, а определяет последовательность действий, которую необходимо выполнить для того, чтобы из исходных данных получить знания. Этот подход универсальный и не зависит от предметной области.

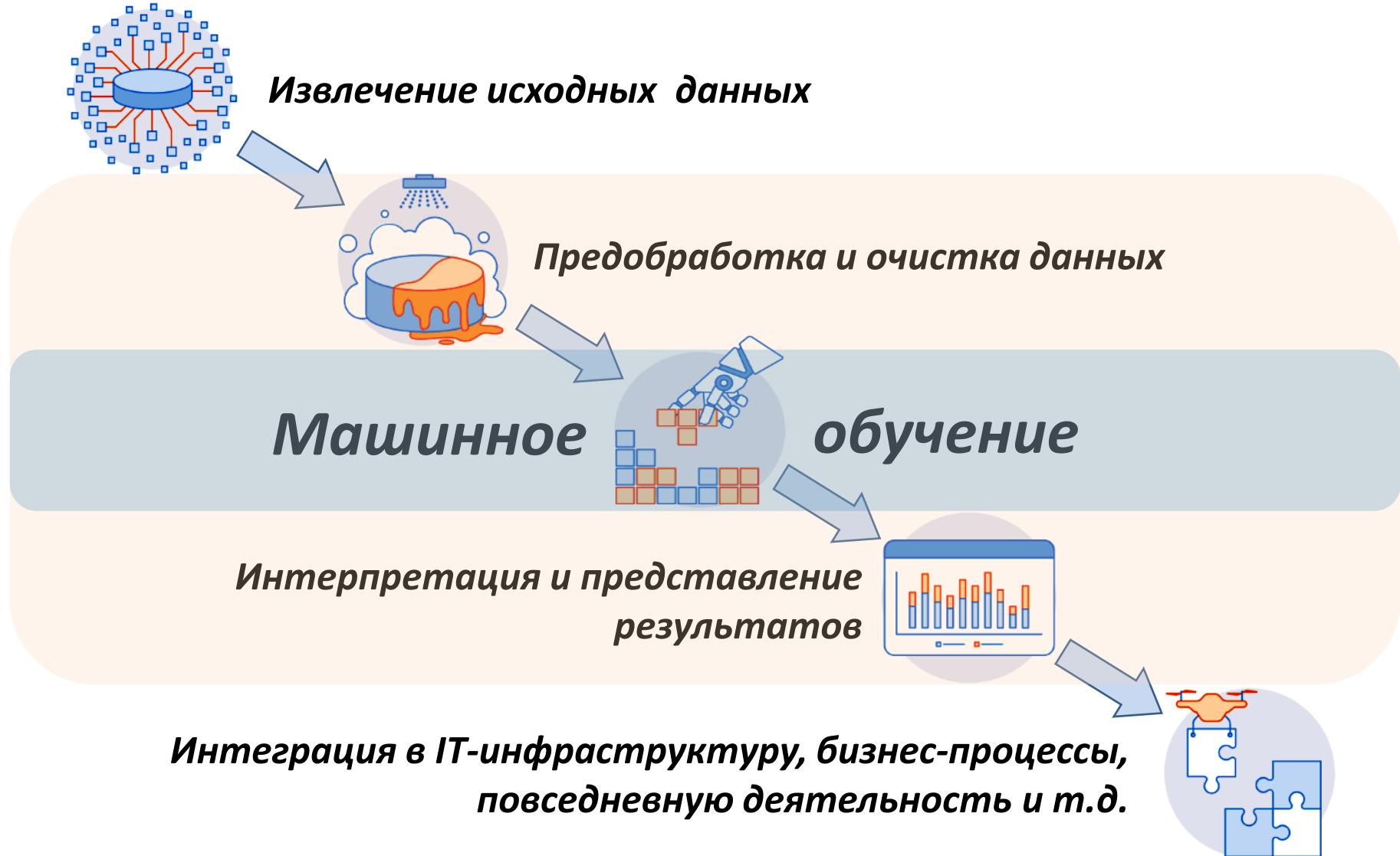
# Интеллектуальный анализ данных (*Data Mining*)

Направлен на обнаружение в исходных данных предварительно неизвестных, практически полезных и доступных интерпретации знаний, шаблонов, закономерностей, отражающих фрагменты многоаспектных взаимоотношений в данных. Особенностью является отсутствие ограничительных рамок априорных предположений о структуре выборки и виде распределений значений анализируемых показателей.



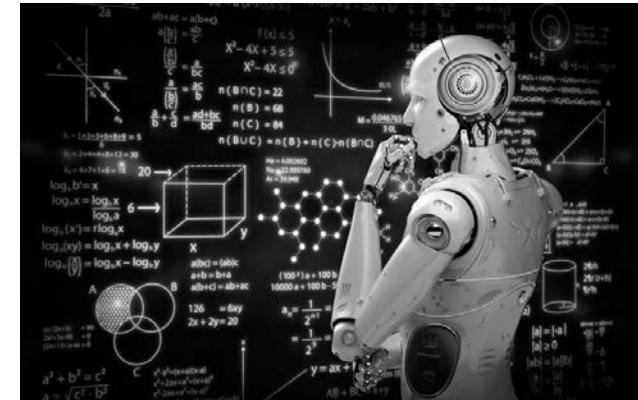
Машинное обучение — это направление искусственного интеллекта, связанное с разработкой и построением аналитических моделей, которые способны автоматически обнаружить в данных скрытые и ранее неизвестные закономерности, а также самостоятельно приобретать свойства, необходимые для распознавания этих закономерностей.

# Процесс интеллектуального анализа данных



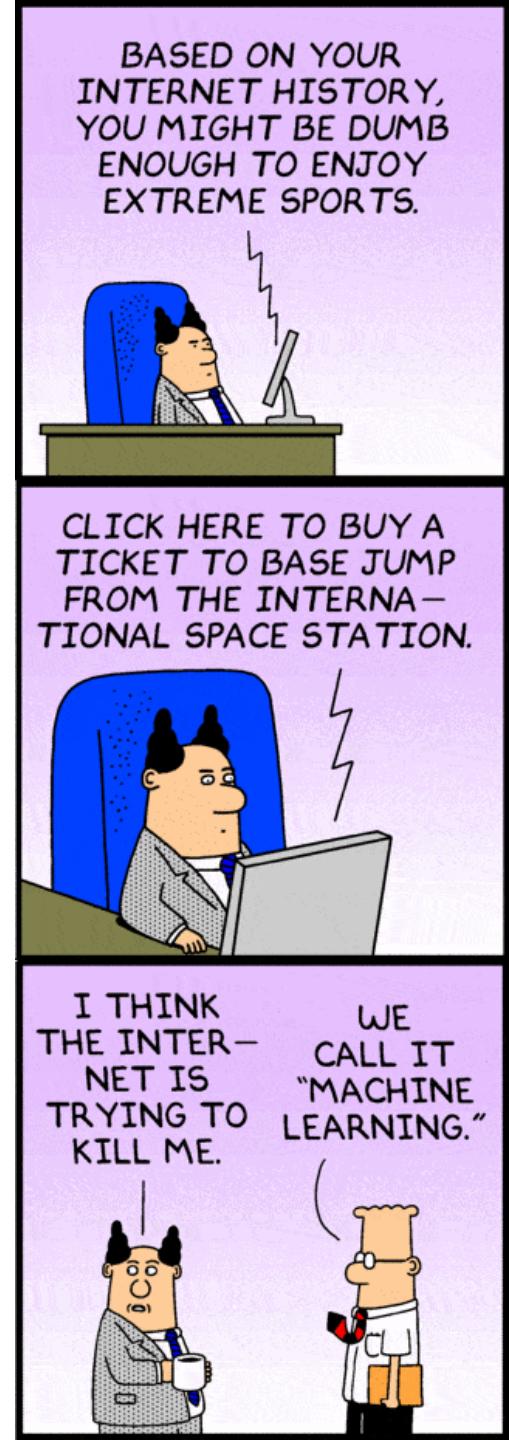
# Классические задачи, решаемые с помощью машинного обучения

- Классификация (*classification*)
- Кластеризация (*clustering/cluster analysis*)
- Регрессия (*regression*)
- Прогнозирование (*forecasting*)
- Поиск субоптимальных решений или стратегий
- Понижение размерности (*dimensionality reduction*)
- Визуализация данных (*data visualization*)
- Восстановление плотности распределения вероятности
- Поиск ассоциативных правил (*association rules learning*)
- Обнаружение аномалий / фильтрация выбросов (*outliers detection*)
- Одноклассовая классификация / Идентификация



# Основные сферы применения

1. Медицинская диагностика.
2. Техника:
  - 2.1. Автоматизация и управление.
  - 2.2. Техническая диагностика.
  - 2.3. Робототехника.
  - 2.4. Компьютерное зрение.
  - 2.5. Распознавание речи.
3. Экономика:
  - 3.1. Кредитный скоринг.
  - 3.2. Предсказание ухода клиентов .
  - 3.3. Обнаружение мошенничества.
  - 3.4. Биржевой технический анализ.
  - 3.5. Биржевой надзор.
4. Офисная автоматизация:
  - 4.1. Распознавание текста.
  - 4.2. Обнаружение спама.
  - 4.3. Категоризация документов.
  - 4.4. Распознавание рукописного ввода.



# Машинное обучение - Machine Learning(ML)

Множество математических, статистических и вычислительных методов для разработки алгоритмов, способных решить задачу не прямым способом, а на основе поиска закономерностей в разнообразных входных данных. Процесс поиска закономерностей называют обучением.

Различают два типа обучения:

- **Обучение по прецедентам**, или **индуктивное обучение** - основано на выявлении эмпирических закономерностей в данных.
- **Дедуктивное обучение** - предполагает формализацию знаний экспертов и их перенос в компьютер в виде базы знаний.

**Особенность:** *Машинное обучение* — не только математическая, но и практическая, инженерная дисциплина, имеющая собственную специфику, связанную с проблемами вычислительной эффективности и переобучения.

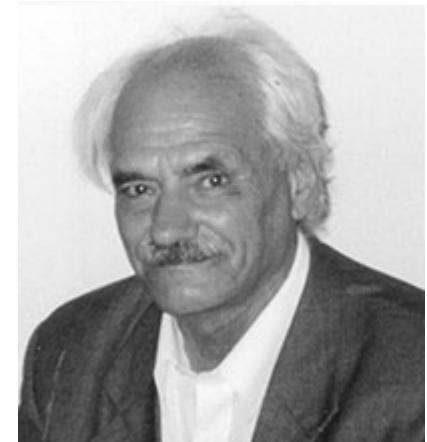


# Теория вычислительного обучения (Computational Learning Theory, COLT)



Вапник  
Владимир Наумович

Работы Вапника и Червоненкиса по статистической теории **восстановления зависимостей по эмпирическим данным** в конце 60-х — начале 70-х послужили основой для создания **теории вычислительного обучения** (*Computational Learning Theory - COLT*), которая изучает методы построения и анализа алгоритмов, обучаемых по прецедентам.



Червоненкис  
Алексей Яковлевич

Основная задача теории вычислительного обучения — дать строгие обоснования алгоритмов обучения по прецедентам. Теория COLT претендует на роль теоретического базиса всего машинного обучения.

**Эмпирические данные** (от др.-греч. *εμπειρία* [*empeiría*] «опыт») — данные, полученные через органы чувств, в частности, путём наблюдения или эксперимента. В философии после Канта полученное таким образом знание принято называть **апостериорным**. Оно противопоставляется **априорному**, доопытному знанию, доступному через чисто умозрительное мышление.

# Обучение по прецедентам (индуктивное обучение)

Дано конечное множество **прецедентов** (объектов)  $\{X_1, \dots, X_m\} \subset X$ , по каждому из которых собраны некоторые данные (описание), а также соответствующие им выходные значения (результаты) из множества допустимые ответов  $Y$ .

Совокупность всех имеющихся описаний прецедентов называется **обучающей выборкой (training sample)**, которая обычно является случайной выборкой объектов из генеральной совокупности  $\Omega$ .

Обучающая выборка имеет вид  $S^m = \{(X_1, Y_1), \dots, (X_m, Y_m)\}$ , где:

- пары «объект–ответ»  $(X_i, Y_i)$  – прецедент  $S_i^m$  из выборки  $S^m$ ;
- $X_i$  – описание  $i$ -ого объекта из  $S^m$ ,  $i = 1, \dots, m$ ;
- $Y_i$  – значение переменной  $Y$  для  $i$ -ого объекта из  $S^m$ ,  $i = 1, \dots, m$ ;
- $m$  – число объектов в  $S^m$ .

Существует неизвестная **целевая функция (Target function)**  $t^*: X \rightarrow Y$ , значения которой известны только для конечного множества объектов обучающей выборки.

Необходимо построить алгоритм, который выдаст достаточно точный результат (выходное значение) для любых возможных входов (наборов значений признаков), в том числе таких, которые ещё не наблюдались.

# Постановка задачи обучения

**Дано:**

- Обучающая выборка  $S^m = \{ (X_1, Y_1), \dots, (X_m, Y_m) \}$ ;
- Предполагается, что существует некоторая неизвестная зависимость  $t^*: X \rightarrow Y$  (*целевая функция*).

**Задача обучения по прецедентам:**

По обучающей выборке  $S^m$  построить некоторую **решающую функцию (decision function)**  $a: X \rightarrow Y$ , которая приближала бы целевую функцию  $t^*$ , причём не только на объектах обучающей выборки, но и на всём множестве возможных объектов генеральной совокупности  $\Omega$ .

# Признаковое описание объектов (*feature vector*)

**Признак (*feature*)** — результат измерения некоторой характеристики объекта, т.е. отображение:  $f: X \rightarrow D_f$ , где  $D_f$  — множество допустимых значений признака.

Для всех прецедентов (объектов) выборки фиксируется совокупность из  $n$  признаков. Если для объекта  $X_i$  заданы признаки  $f_1, \dots, f_n$ , то вектор  $(f_1(X_i), \dots, f_n(X_i)) = (x_i^1, \dots, x_i^n)$  называется признаковым описанием объекта  $X_i \in X$ .

**Признаковое описание объекта (*feature vector*)** — это вектор, который составлен из значений, соответствующих фиксированному набору признаков (характеристик) для данного объекта.

В машинном обучении признаковые описания допустимо отождествлять с самими объектами. При этом все множество  $X$  называют признаковым пространством.

Матрицей объектов-признаков (матрицей/таблицей исходных данных) называется совокупность признаковых описаний объектов обучающей выборки  $S^m$  длины  $m$ , записанной в виде матрицы размера  $m \times n$  ( $m$  строк,  $n$  столбцов). Столбцы соответствуют признакам, строки - признаковым описаниям объектов.

	$f_1$	$\dots$	$f_n$
$X_1$	$x_1^1$	$\dots$	$x_1^n$
$\dots$	$\dots$	$\dots$	$\dots$
$X_m$	$x_m^1$	$\dots$	$x_m^n$

# Типы признаков (Атрибутов)

Значения в исходном признаковом описании прецедентов (объектов) могут подразделяться на **количественные и качественные**.

**Количественным (quantitative)** называется признак, который имеет числовое представление и они могут быть:

- дискретными (discrete data) - выражаемые ограниченным набором значений (обычно целыми числами)
- непрерывными (continuaes data) - принимающие значения на непрерывной шкале значений.

**Качественные (attribute, qualitative)** признаки выражаются нечисловыми значениями и подразделяются на:

- альтернативные (бинарные) – имеют только два варианта значений.
- атрибутивные (неупорядоченные/категориальные) - имеет более двух вариантов, которые при этом выражаются в виде понятий или наименований.
- порядковые (ординальные) - имеют несколько ранжированных, т.е. упорядоченных по возрастанию или убыванию, качественных вариантов.

В результате, после предобработки и кодирования, в зависимости от множества допустимых значений  $D_f$  признаки делятся на следующие типы:

- бинарный признак:  $D_f = \{0, 1\}$ ;
- номинальный признак:  $D_f$  — конечное множество;
- порядковый признак:  $D_f$  — конечное упорядоченное множество;
- количественный признак:  $D_f$  — множество действительных чисел.

# Модель алгоритмов

Любая дисциплина, использующая математический аппарат, так или иначе, занимается математическим моделированием – заменой реального объекта его абстрактным, идеализированным представлением и использованием полученного представления для изучения объекта и добычи знаний.

**Моделью алгоритмов (predictive model)** называется параметрическое семейство функций (отображений), аппроксимирующих связь входных данных  $X$  и выходных данных  $Y$  на обучающей выборке:

$$A = \{g(X, \theta) | \theta \in \Theta\}$$

где  $g: X \times \Theta \rightarrow Y$  – некоторая фиксированная функция,  $\Theta$  – множество допустимых значений набора параметров  $\theta$ , называемое пространством параметров или пространством поиска (search space). При построении моделей элементы множества  $\theta$ , как правило, заранее неизвестны и требуют нахождения.

Типы математических моделей в зависимости от:

- Математического вида функции - линейные / нелинейные.
- Количество переменных - сосредоточенные / распределенные.
- Присутствия случайности - детерминированные / стохастические.
- Изменчивости во времени – статические / динамические.
- Используемых параметров и переменных - дискретные / непрерывные.

# Метод обучения модели (learning algorithm)

Метод обучения модели - это отображение  $\mu: S^m \rightarrow A$ , которое произвольной конечной выборке  $S^m = \{ (X_1, Y_1), \dots, (X_m, Y_m) \}$  ставит в соответствие некоторый алгоритм  $a \in A$ . Говорят также, что метод  $\mu$  строит алгоритм  $a$  по выборке  $S^m$ .

Пусть  $\lambda[Y_i, a(X_i)]$  – величина/функция «потерь» (loss function), произошедших в результате использования  $a(X_i)$  в качестве прогноза значения  $Y$ . Классический метод обучения (empirical risk minimization, ERM), заключается в том, чтобы найти в заданной модели  $A$  алгоритм  $a$ , который позволит **минимизировать функционал эмпирического риска (функционал качества)** на обучающей выборке:

$$Q(S^m, a) = \frac{1}{m} \sum_{i=1}^m \lambda[Y_i, a(X_i)] \rightarrow \min_{a \in A}$$

Примеры функции потерь, при  $Y \subseteq \mathbb{R}$ :

- $\lambda[Y_i, a(X_i)] = [a(X_i) \neq t^*(X_i)]$  - индикатор ошибки;
- $\lambda[Y_i, a(X_i)] = |a(X_i) - t^*(X_i)|$  - модуль величины ошибки;  
функционал  $Q$  называется средней ошибкой;
- $\lambda[Y_i, a(X_i)] = (a(X_i) - t^*(X_i))^2$  - квадратичная функция потерь;  
функционал  $Q$  называется средней квадратичной ошибкой.

# Свойства алгоритма обучения

Алгоритм обучения принимает на входе конечную обучающую выборку прецедентов и настраивает модель. Настроенная (обученная) модель затем используется для предсказания будущих прецедентов. Алгоритм должен обладать свойством **обучаемости** в следующих двух смыслах.

- Во-первых, алгоритм обучения должен обладать способностью к **обобщению** данных. Построенная им модель должна выдавать в среднем достаточно точные предсказания будущих прецедентов, т.е. обобщение определяет адекватный отклик на данные, выходящие за пределы имеющейся обучающей выборки. Оценки обобщающей способности, как правило, основываются на гипотезе, что прошлые и будущее прецеденты поступают случайно и независимо из одного и того же неизвестного вероятностного распределения. Эта гипотеза позволяет применить статистические методы для получения верхних оценок ожидаемой в будущем ошибки.
- Во-вторых, процесс обучения должен завершиться за приемлемое время. Обычно исследуется вопрос, является ли время обучения модели полиномиальным или экспоненциальным по длине выборки. Таким образом, проблематика вычислительного обучения тесно связана также и с вопросами **вычислительной сложности** алгоритмов.

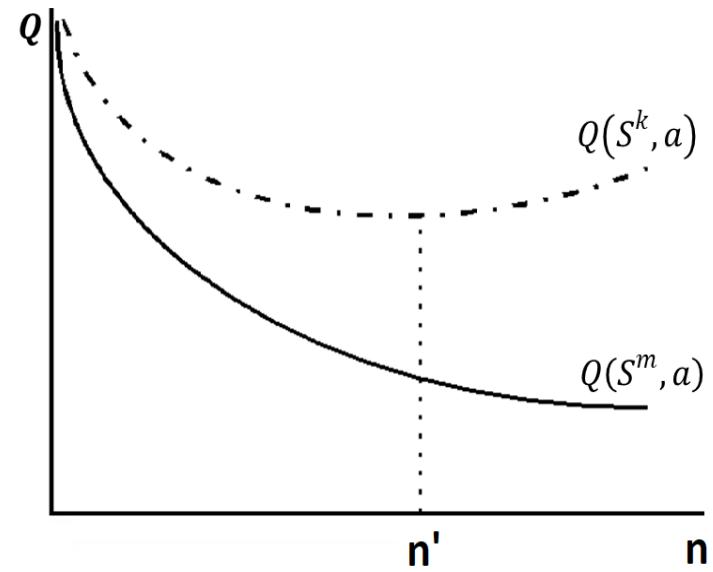
# Обобщающая способность (*generalization ability*)

Обобщающая способность – это свойство модели отражать исходные данные в требуемые результаты ( $X \rightarrow Y$ ) на всем множестве возможных объектов генеральной совокупности (во всех сценариях, а не только на тренировочных примерах).

Если минимум функционала эмпирического риска (функционал качества)  $Q(S^m, a)$  достигается на алгоритме  $a$ , то это не гарантирует, что  $a$  хорошо приближает целевую зависимость на произвольной контрольной выборке  $S^k = (X'_i, Y'_i)_{i=1}^k$ . **Обобщающая способность (*generalization ability*)** метода  $\mu$  характеризуется величиной  $Q(S^k, a) = Q(S^k, \mu(S^m))$  при условии, что выборки  $S^k$  и  $S^m$  являются представительными.

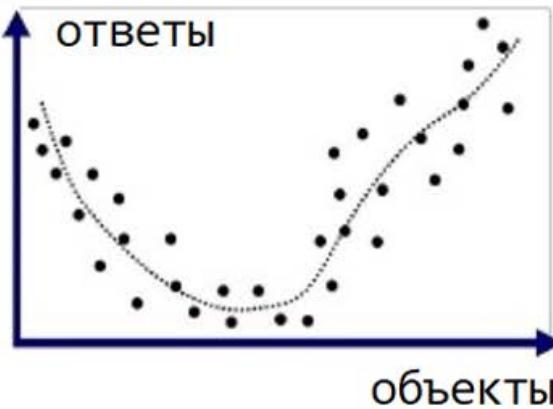
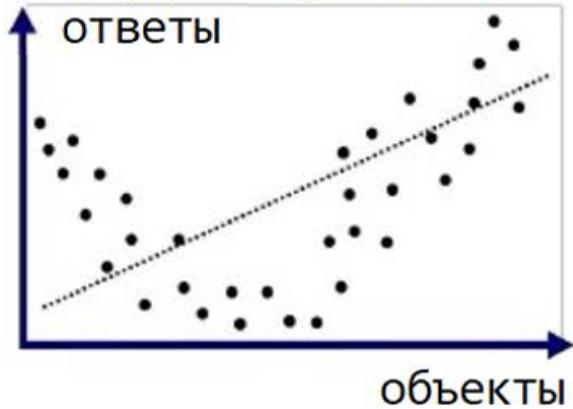
Метод обучения  $\mu$  называется состоятельным, если при заданных достаточно малых значениях  $\varepsilon$  и  $\eta$  для любых простых выборок  $S^k$  и  $S^m$  справедлива оценка  $Q(S^k, \mu(S^m)) \leq \varepsilon$  с вероятностью не менее  $1 - \eta$ .

Когда качество работы алгоритма на новых объектах, не вошедших в состав обучения, оказывается существенно хуже, чем на обучающей выборке, говорят об эффекте **переобучения (overtraining)** или **переподгонки (overfitting)**.

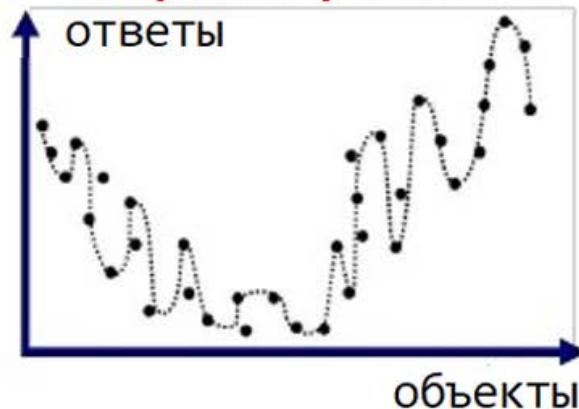


# Проблема недообучения и переобучения

## недообучение



## переобучение



**Недообучение** (underfitting) возникает в том случае, когда модель слишком проста и содержит недостаточное число параметров  $n$ .

**Переобучение** (overfitting) возникает в том случае, когда модель слишком сложная и содержит избыточное число параметров  $n$ .

Как бороться:

- Уменьшить число настраиваемых параметров модели;
- По возможности увеличить число обучающих примеров;
- Уменьшить число итераций алгоритма обучения;
- Использовать эмпирические оценки обобщающей способности.

# Эмпирические оценки обобщающей способности

Пусть дана выборка  $S^m = \{(X_1, Y_1), \dots, (X_m, Y_m)\}$ . Разобьём её  $N$  различными способами на две непересекающиеся подвыборки - обучающую  $S_n^\ell$  длины  $\ell$  и контрольную  $S_n^k$  длины  $k = m - \ell$ . Для каждого разбиения  $n=1, \dots, N$  построим алгоритм  $a_n = \mu(S_n^\ell)$  и вычислим значение функционала эмпирического риска  $Q_n = Q(S_n^k, a_n)$ .

Среднее арифметическое значений  $Q_n$  по всем разбиениям называется оценкой скользящего контроля (**cross-validation, CV**):

$$CV(\mu, S^L) = \frac{1}{N} \sum_{n=1}^N Q(S_n^k, \mu(S_n^\ell))$$

Стандартом «де факто» считается методика  $t \times q$ -кратного скользящего контроля ( **$t \times q$ -fold cross-validation**), когда выборка случайным образом разбивается на  $q$  блоков равной (или почти равной) длины, каждый блок по очереди становится контрольной выборкой, а объединение всех остальных блоков – обучающей выборкой. Выборка  $S^m$  по-разному  $t$  раз разбивается на  $q$  блоков. Итого получается  $N = t \times q$  разбиений.

# Основные проблемные вопросы машинного обучения

Опираясь на понятие обобщающей способности, можно выделить следующие основные проблемные вопросы машинного обучения:

- Достаточно ли данных для нахождения в них полезных знаний?
- Может ли в принципе данная модель обучиться на имеющихся данных?
- Будет ли полученная модель приближать требуемый «закон природы» на всем возможном множестве  $\Omega$ ?
- Насколько хорошо будет работать обученная модель или насколько часто и насколько сильно будет ошибаться модель на реальных (контрольных) данных?

# Общий порядок действий при обучении по прецедентам

- Анализ постановки задачи и исходных данных;
- Формулировка решения на математическом языке (задача формализуема, а результаты работы модели могут быть проверены);
- Предобработка данных и выделение ключевых признаков;
- Выбирается и фиксируется модель восстанавливаемой зависимости;
- Вводится функционал качества, значение которого показывает, насколько хорошо модель описывает наблюдаемые данные;
- Алгоритм обучения (learning algorithm) ищет такой набор параметров модели, при котором функционал качества на заданной обучающей выборке принимает оптимальное значение;
  - Процесс настройки (fitting) модели по выборке данных в большинстве случаев сводится к применению численных методов оптимизации;
- Эксплуатация модели при достижении требуемого качества, либо возврат к одному из предыдущих шагов (перенастройка модели, добыча новых данных и т. п.).

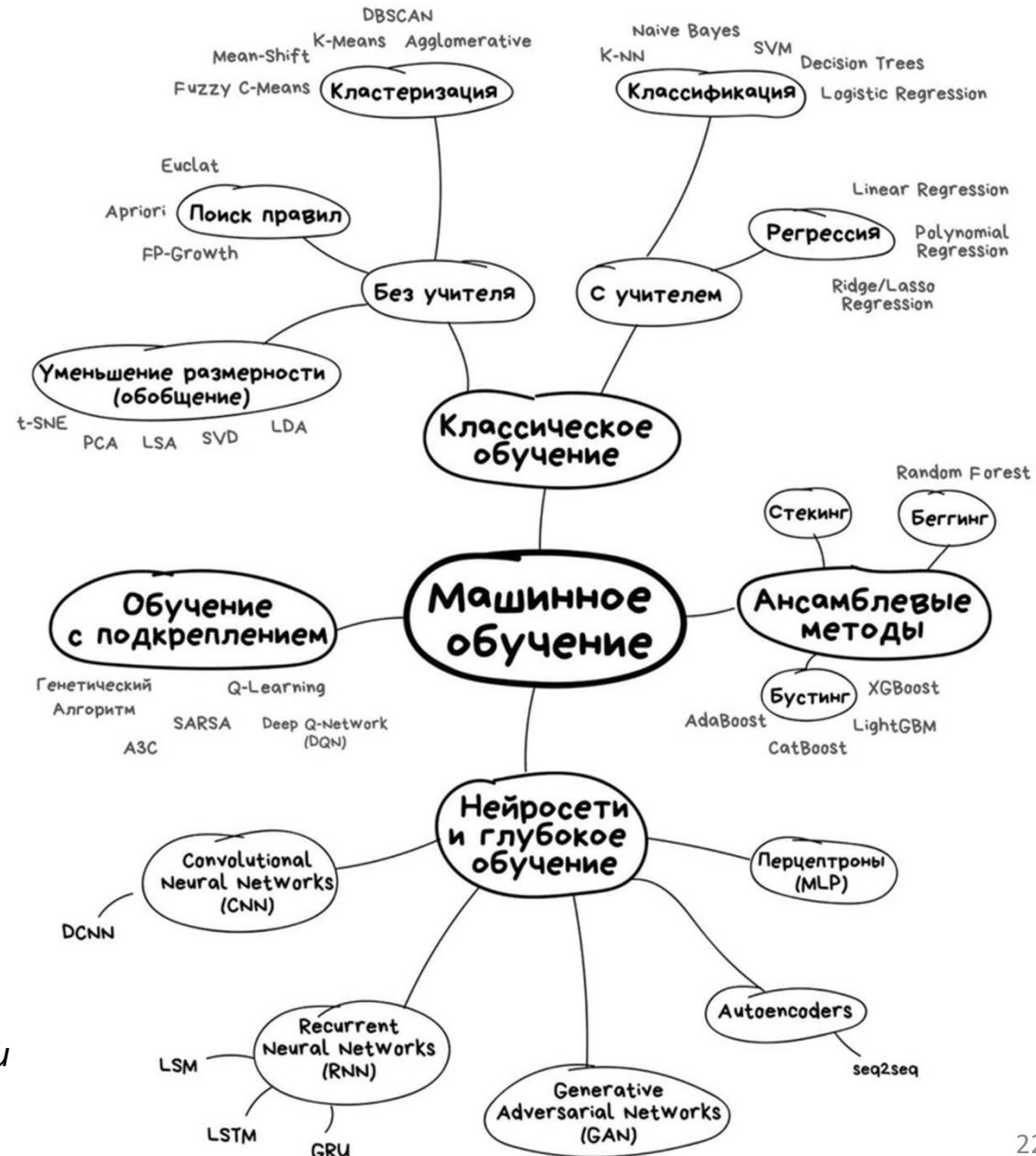
Можно сказать, что машинное обучение реализует подход **Case Based Reasoning (CBR)** — метод решения проблем рассуждением по аналогии, путем предположения на основе подобных случаев (прецедентов).

# Типы машинного обучения

- Классическое обучение:
  - Обучение с учителем (supervised learning)
  - Обучение без учителя (unsupervised learning)
- Обучение с подкреплением (reinforcement learning)
- Ансамблевые методы (Ensemble of models)
- Нейронные сети и глубокое обучение

Дополнительно выделяют:

- Частичное обучение (*semi-supervised learning*)
- Трансдуктивное обучение (*transductive learning*)
- Динамическое обучение (*online learning*)
- Активное обучение (*active learning*)
- Метаобучение (*meta-learning* или *learning-to-learn*)

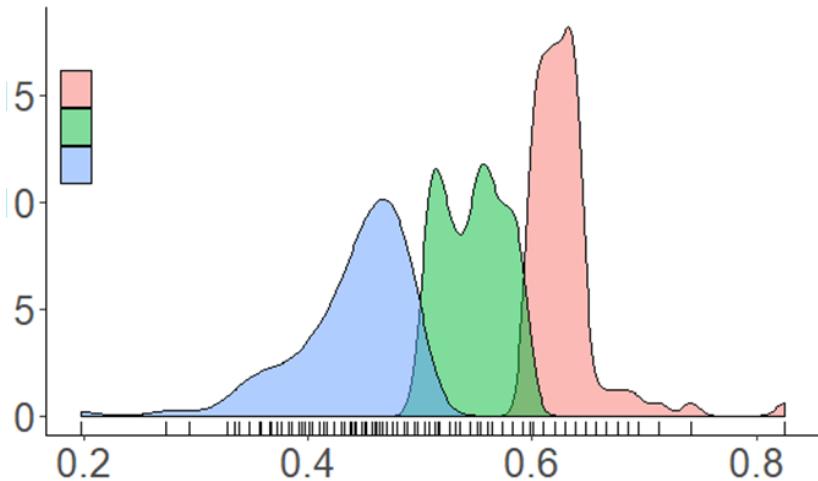


# Тема: «Основные понятия машинного обучения»

- Основные понятия: модель, признаки (непрерывные и дискретные), выборка, меры близости, метрические пространства.
- Неопределенность, виды неопределенности.
- Основы статистического анализа.
- Предобработка данных.

# Основы математической статистики

- Случайная природа входных данных.
- Возможные распределения значений признаков: нормальное, равномерное и др.
- Независимость признаков, матрица корреляции.



	1	2	3	4	5	6	7	8	9	10	11	12
1	0,27	-0,28	0,38	-0,45	0,45	-0,08	-0,02	-0,16	-0,25	0,05	-0,18	
2	0,27		-0,43	0,45	-0,29	0,26	0,41	-0,30	0,32	0,27	-0,25	0,12
3	-0,28	-0,43		-0,91	0,40	-0,37	-0,47	0,49	-0,47	-0,28	0,50	-0,29
4	0,38	0,45	-0,91		-0,48	0,45	0,45	-0,49	0,44	0,11	-0,45	0,13
5	-0,45	-0,29	0,40	-0,48		-0,94	-0,12	0,25	-0,18	0,20	0,15	0,27
6	0,45	0,26	-0,37	0,45	-0,94		0,01	-0,10	0,07	-0,23	0,00	-0,31
7	-0,08	0,41	-0,47	0,45	-0,12	0,01		-0,71	0,82	0,44	-0,65	0,28
8	-0,02	-0,30	0,49	-0,49	0,25	-0,10	-0,71		-0,75	-0,52	0,81	-0,42
9	-0,16	0,32	-0,47	0,44	-0,18	0,07	0,82	-0,75		0,55	-0,58	0,28
10	-0,25	0,27	-0,28	0,11	0,20	-0,23	0,44	-0,52	0,55		-0,47	0,62
11	0,05	-0,25	0,50	-0,45	0,15	0,00	-0,65	0,81	-0,58	-0,47		-0,65
12	-0,18	0,12	-0,29	0,13	0,27	-0,31	0,28	-0,42	0,28	0,62	-0,65	

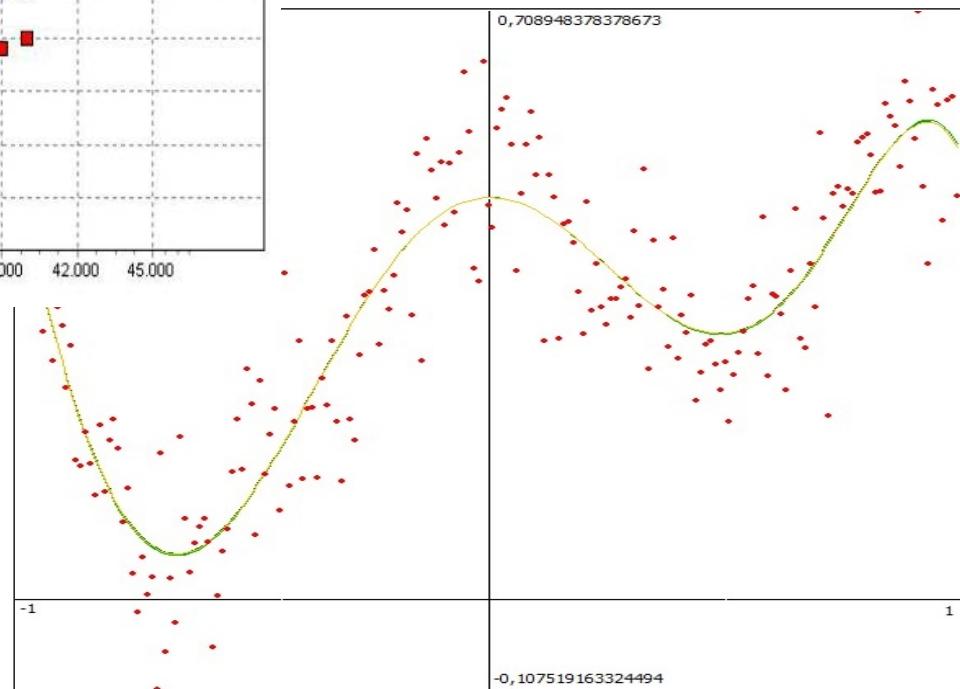
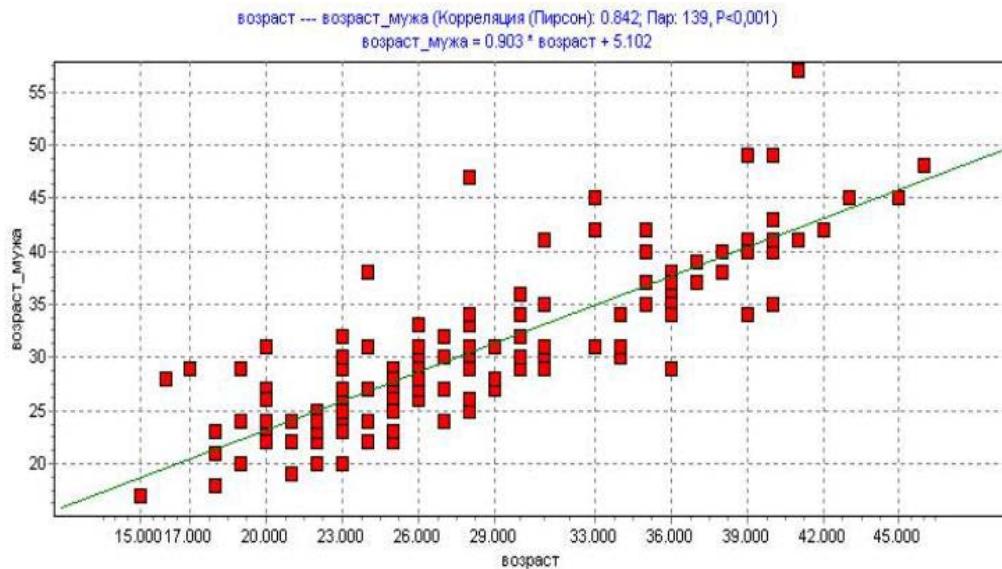
# Тема: «Классическое машинное обучение»

## Обучение с учителем (supervised learning)

- Постановка задачи регрессии, линейная регрессия, логистическая, основные метрики качества регрессии.
- Постановка задачи классификации, методы классификации, основные метрики качества классификации.

# Регрессионные модели

- Постановка задачи;
- Линейная и нелинейная регрессия;
- Основные метрики оценки качества.



# Задачи классификации

- Постановка задачи;
- Методы классификации;
- Основные метрики оценки качества.

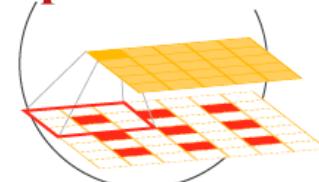


Дерево  
решений

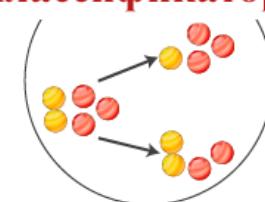


Метод опорных  
векторов

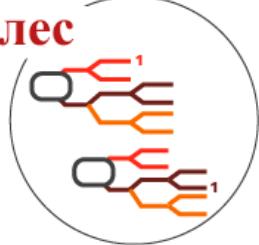
Сверточные  
нейронные сети



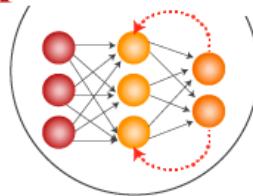
Наивный  
байесовский  
классификатор



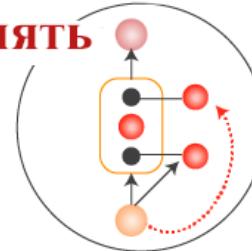
Случайный  
лес



Рекуррентные  
нейронные сети



Долгая  
краткосрочная  
память

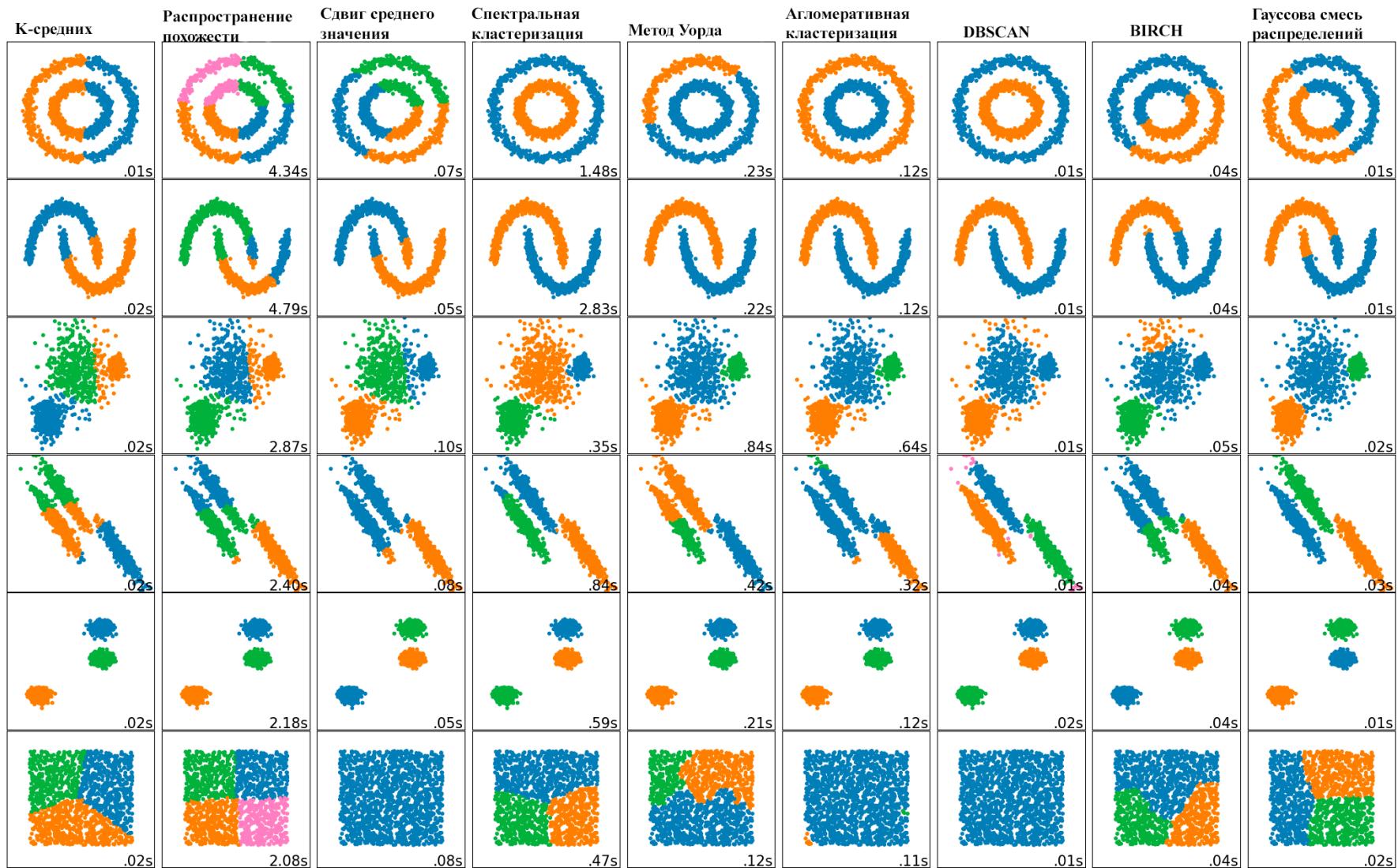


# Тема: «Классическое машинное обучение»

## Обучение без учителя (unsupervised learning)

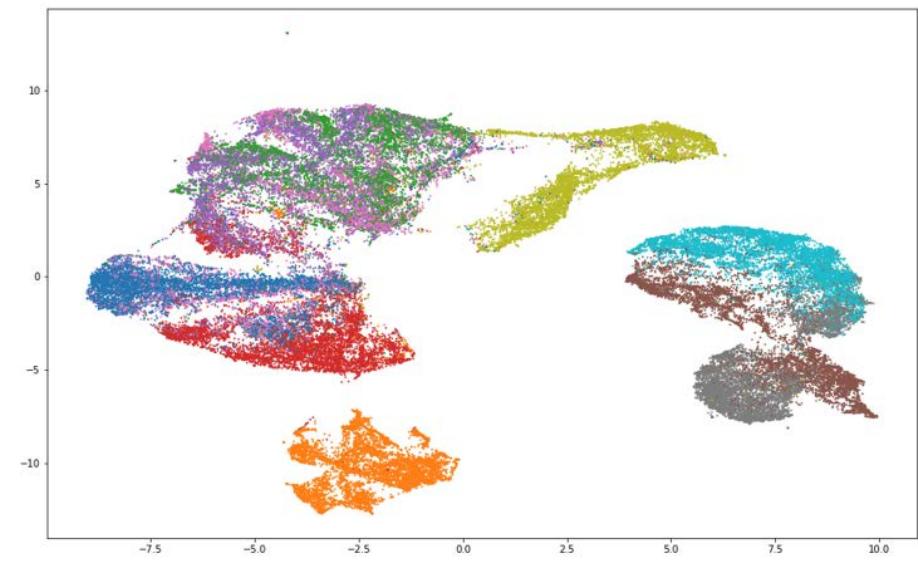
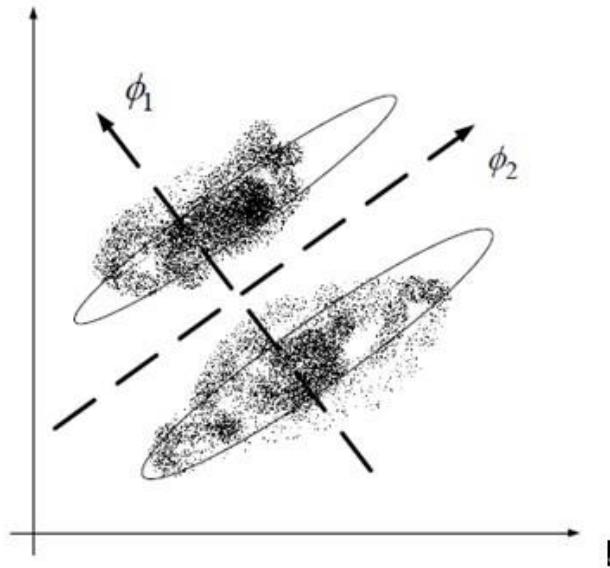
- Методы кластеризации: иерархическая, k-средних, DBSCAN. Применение кластеризации в анализе данных.
- Методы понижения размерности: PCA, T-SNE, UMAP.

# Методы кластеризации



# Методы понижения размерности

- Алгоритм анализа главных компонентов (PCA);
- Линейный дискриминантный анализ (линейный дискриминант Фишера) (LDA);
- Локально линейное вложение (LLE);
- t-distributed stochastic neighbor embedding (T-SNE);
- Uniform Manifold Approximation and Projection (UMAP).

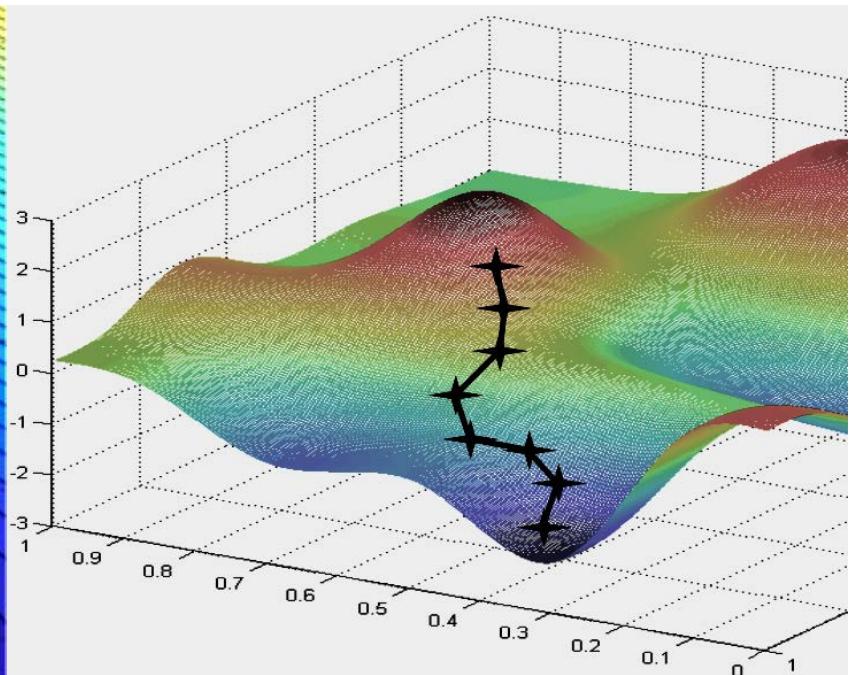
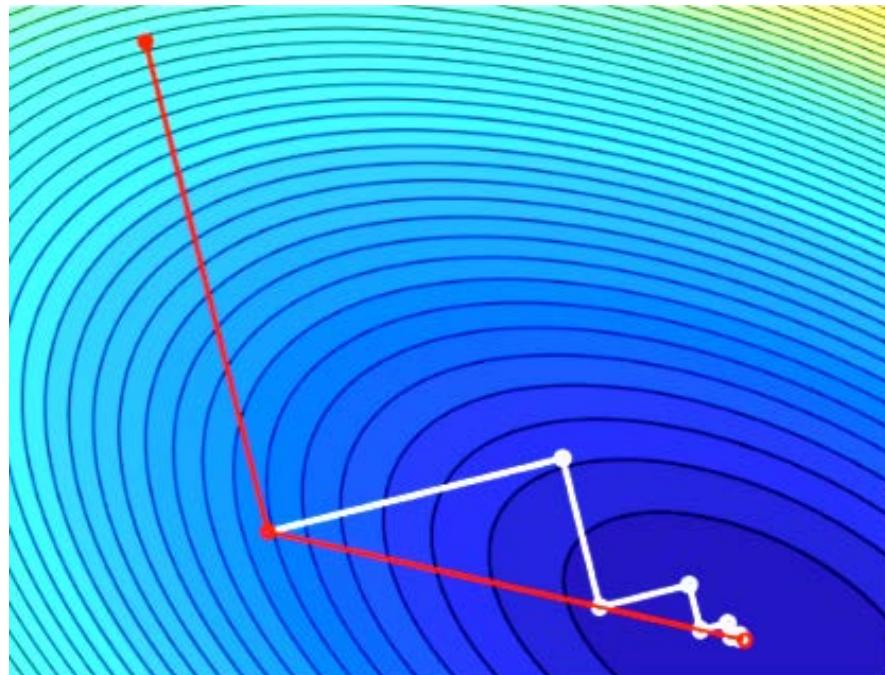


# Тема: «Ансамбли классификаторов»

- Методы оптимизации в приложении к задачам машинного обучения. Градиентный спуск и его адаптации.
- Слабые и сильные классификаторы (байесовский классификатор, метод опорных векторов и др.), их ансамбли. Алгоритмы бустинга (AdaBoost, CatBoost и др.).

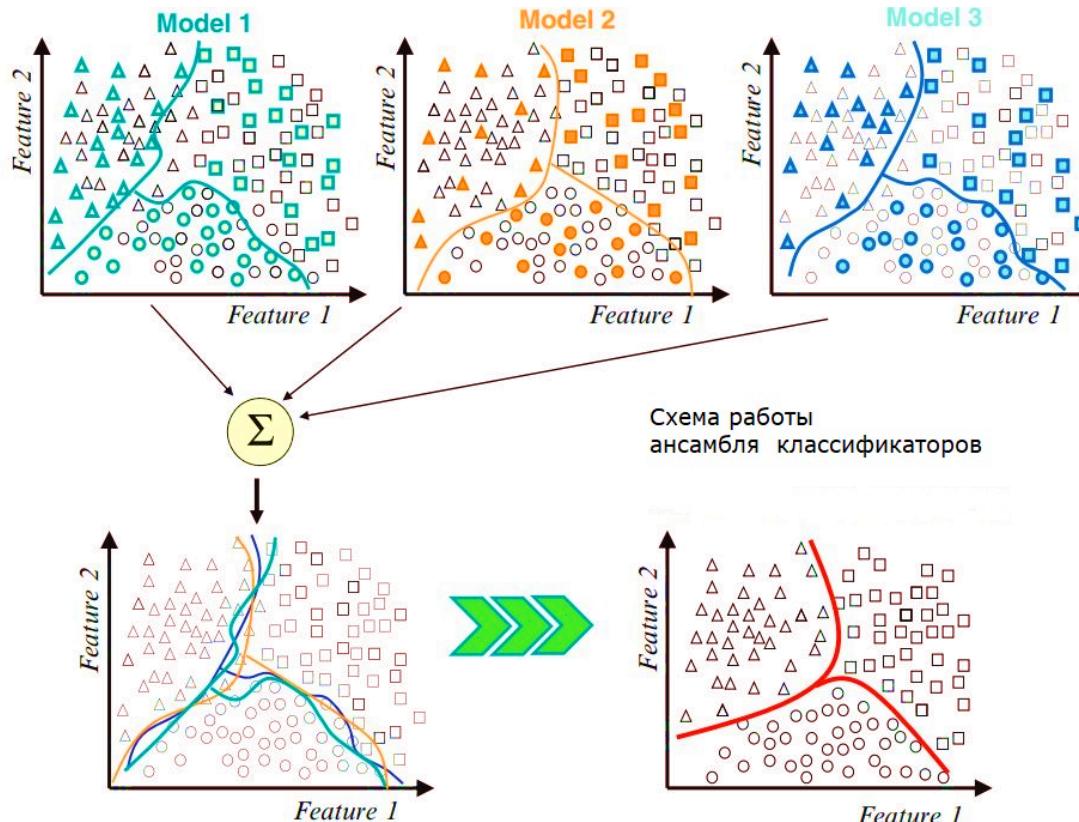
# Методы оптимизации применительно к задачам машинного обучения

- Обзор методов численной оптимизации;
- Скорость сходимости;
- Градиентный спуск и его адаптации;
- Метод сопряженных градиентов.



# Слабые и сильные классификаторы

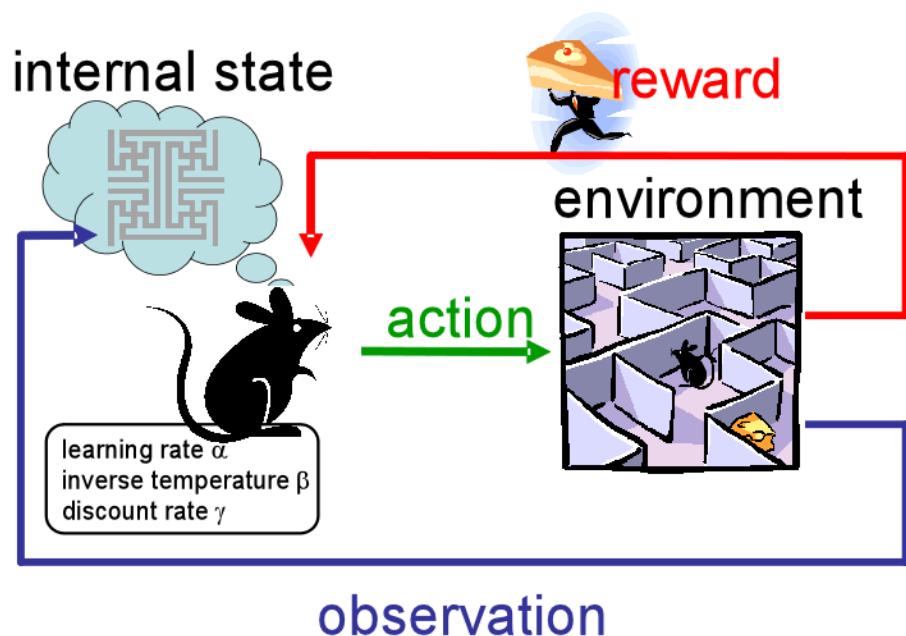
- Байесовский классификатор;
- Метод опорных векторов;
- Ансамблевые классификаторы;
- Алгоритмы бустинга (AdaBoost, CatBoost и др.)



# Тема: «Обучение с подкреплением» (reinforcement learning)

Способ машинного обучения, при котором система обучается, взаимодействуя с некоторой средой.

- Марковские процессы. Марковские модели. Агент (*agent*), среда (*environment*), обратная связь, состояние, функции ценности состояния (Value function), качества действия (Q-function);
- Многорукие бандиты;
- Способы взаимодействия и оптимизационные задачи с помощью агентов;
- Генетические алгоритмы.

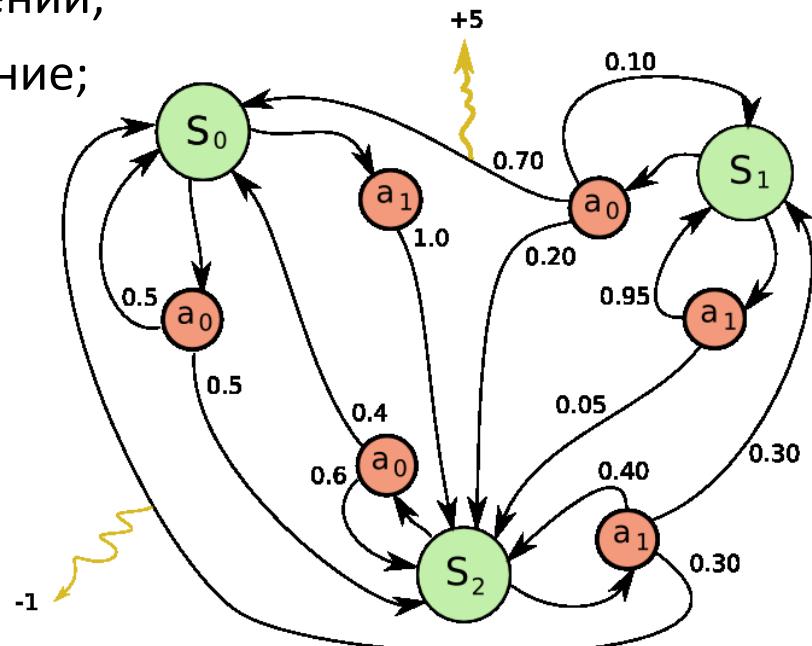


# Марковские процессы

**Марковский процесс** — случайный процесс, эволюция которого после любого заданного значения временного параметра  $t$  не зависит от эволюции, предшествовавшей  $t$ , при условии, что значение процесса в этот момент фиксировано («будущее» процесса зависит от «прошлого» лишь через «настоящее»).

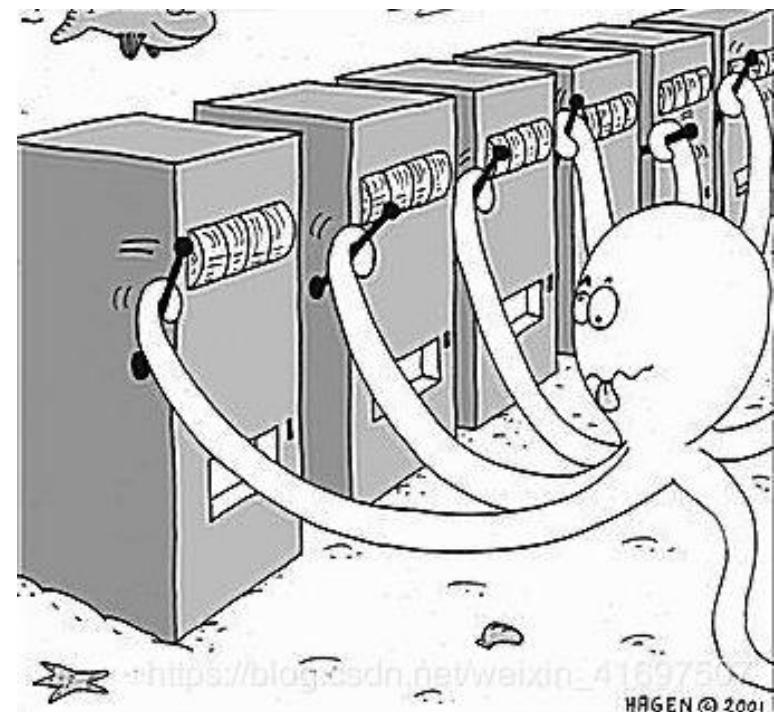
**Марковское свойство** — в теории вероятностей и статистике термин, который относится к памяти случайного процесса.

- Марковский процесс принятия решений;
- Агент, среда, обратная связь, состояние;
- Функция полезности состояния (Value function);
- Функция полезности действия (Q-function);
- Q-обучение (Q-learning).



# Задача о многоруком бандите (*The multi-armed bandit problem*)

- Агенты с одним состоянием, т.е. состояние агента не меняется. У него фиксированный набор действий и возможность выбора из этого набора действий.
- Модель: агент в комнате с несколькими игровыми автоматами. У каждого автомата своё ожидание выигрыша.
- Нужно заработать побольше:  
Exploration vs. Exploitation  
(разведка против эксплуатации).
- Жадные и  $\epsilon$ -жадные стратегии  
(greedy &  $\epsilon$ -greedy)



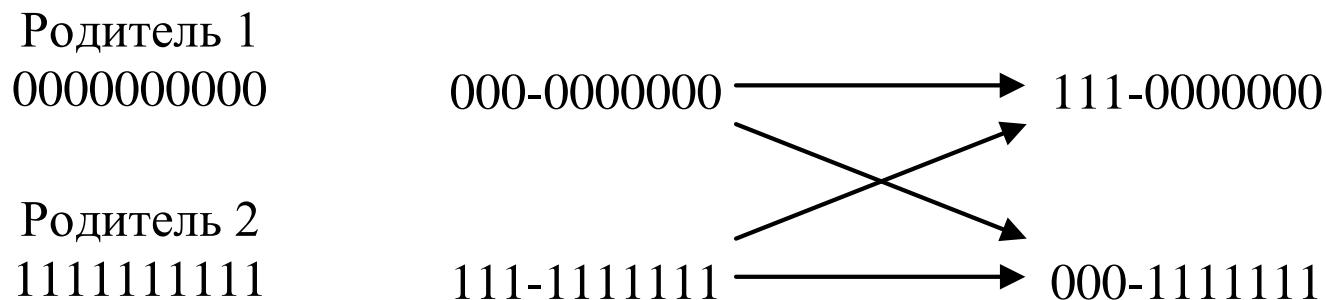
HAGEN © 2001

# Генетический алгоритм (*genetic algorithm*)

Эвристический алгоритм поиска, используемый для решения задач оптимизации и моделирования путём случайного подбора, комбинирования и вариации искомых параметров с использованием механизмов, аналогичных естественному отбору в живой природе, таких как: скрещивание, наследование, мутации и отбор.

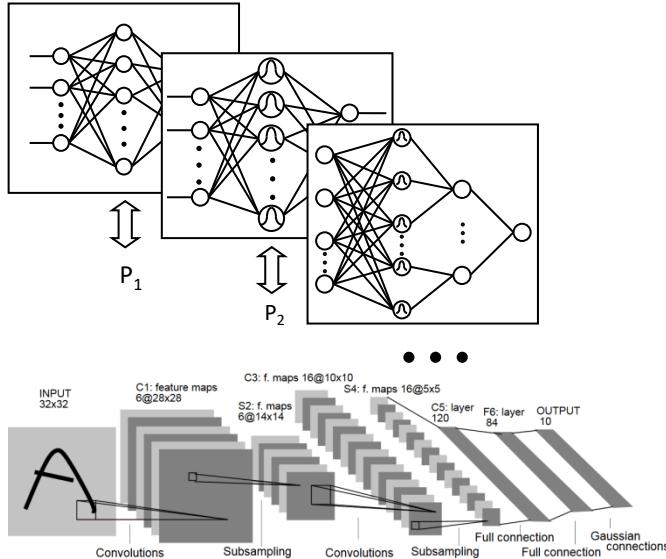
Область применения:

- поиск логических закономерностей - сегментация данных, поиск оптимального набора элементарных событий, выделение наиболее значимых значений признаков и их сочетаний;
- предобработка исходных данных в задаче классификации;
- поиск значений параметров аппроксимирующей функции для числовых последовательностей;
- обучение нейронной сети.



# Тема: «Нейронные сети»

Цель: выбор оптимальной топологии сети, значений параметров и структурных особенностей, которые бы наилучшим образом удовлетворяли решаемой задаче на имеющихся исходных данных.



- Многослойный персепtron
- RBF-сеть
- Гибридные нейросети
- Самоорганизующиеся карты Кохонена
- Сверточные сети
- Глубокое обучение
- И т.д.

Необходимо учитывать:

- Решаемая задача;
- Исходные данные (объем, структура и т.д.);
- Внутреннее представление информации нейросетью;
- Процесс обучения (обобщение vs переобучение);
- Интерпретируемость полученного результата.

# Инструменты Machine Learning

Инструменты машинного обучения используют на следующих этапах:

- сбор и подготовка данных;
- построение модели;
- интерпретация работы модели;
- обучение и развертывание приложений.

Для выполнения каждого из этих этапов могут применяться специализированные платформы. Они различаются по языку программирования (Python, R, Java, C, C++, Scala, CUDA), операционным системам (Linux, Mac OS, Windows) и тому, какие задачи можно решить с их помощью. Поддерживаются многими интегрированными средами разработки, в частности, R-Studio, R-Brain, Visual Studio, Eclipse, PyCharm, Spyder, IntelliJ IDEA, Jupyter Notebooks, Juno и др.

Сегодня на рынке представлено несколько десятков программных инструментов:

- TensorFlow;
- PyTorch ;
- Shogun;
- Keras.io;
- Rapid Miner;
- Google Cloud ML Engine;
- Amazon Machine Learning (AML);
- Accord.NET;
- Apache Mahout;
- Microsoft Azure ML;
- SberCloud ML Space.

# **Перечень учебной литературы и дополнительных материалов, необходимых для освоения дисциплины**

## **Литература по дисциплине**

- В. В. Воронина, А. В. Михеев, Н. Г. Ярушкина, К. В. Святов, Теория и практика машинного обучения : учебное пособие / Ульяновск : УлГТУ, 2017.
- Математические методы обучения по прецедентам (теория обучения машин) Курс лекций К.В.Воронцова, URL: <http://www.machinelearning.ru>.
- Гудфеллоу Я., Бенджио И., Курвилль А. Глубокое обучение / пер. с англ. А. А. Слинкина. – 2-е изд.: ДМК Пресс, 2018. – 652 с.
- СИСТЕМЫ ИСКУССТВЕННОГО ИНТЕЛЛЕКТА 2-е изд., испр. и доп. Учебное пособие для вузов / Бессмертный И. А. - 2022. - URL: <https://urait.ru/book/A7D41B16-CE66-4451-92E9-73529EA6C9F5>.
- Плас Дж. Вандер Python для сложных задач: наука о данных и машинное обучение / Плас Дж. Вандер ; пер. с англ. Пальти И. - СПб. : Питер, 2020. - 572 с. : рис., табл. - (Бестселлеры O'Reilly). - Библиогр. в конце глав. - ISBN 978-5-4461-0914-2.
- Саттон, Р. С. Обучение с подкреплением: введение : руководство / Р. С. Саттон, Э. Д. Барто ; перевод с английского А. А. Слинкина. — Москва : ДМК Пресс, 2020. — 552 с. — ISBN 978-5-97060-097-9. — Текст : электронный // Лань : электронно-библиотечная система. — URL: <https://e.lanbook.com/book/179453>
- Араки, М. Манга: Машинное обучение / М. Араки ; перевод с японского А. С. Слащевой ; Ватари Макана. — Москва : ДМК Пресс, 2020. — 214 с. — ISBN 978-5-97060-830-2. — Текст : электронный // Лань : электронно-библиотечная система. — URL: <https://e.lanbook.com/book/179473>
- Шарден, Б. Крупномасштабное машинное обучение вместе с Python : учебное пособие / Б. Шарден, Л. Массарон, А. Боскетти ; перевод с английского А. В. Логунова. — Москва : ДМК Пресс, 2018. — 358 с. — ISBN 978-5-97060-506-6. — Текст : электронный // Лань : электронно-библиотечная система. — URL: <https://e.lanbook.com/book/105836>

## **Дополнительные материалы**

- Основы искусственного интеллекта : учебное пособие / Е.В.Боровская, Н. А. Давыдова. 4-е изд.,электрон. М. : Лаборатория знаний, 2020. 130 с.
- Искусственный интеллект с примерами на Python. Джоши Пратик. Вильямс. 2019.
- Прикладное машинное обучение с помощью Scikit-Learn, Keras и TensorFlow: концепции, инструменты и техники для создания интеллектуальных систем , 2-е издание. Жерон Орельен.Диалектика-Вильямс. 2020.
- Хенрик Бринк, Джозеф Ричардс, Марк Феверолф «Машинное обучение», Питер 2017.
- Как учится машина: Революция в области нейронных сетей и глубокого обучения. Ян Лекун. Альпина PRO. 2021.
- Грокаем глубокое обучение. Эндрю Траск. Питер. 2019.
- Обучение с подкреплением на PyTorch. Сборник рецептов. Юси Лю. ДМК Пресс. 2020.

# **Методика оценки по рейтингу**

**Промежуточная аттестация** по дисциплине проходит в форме дифференцированного зачета за курсовую работу и экзамена, контролирующего освоение ключевых, базовых положений дисциплины, составляющих основу остаточных знаний по ней.

Студент, выполнивший все предусмотренные учебным планом задания и сдавший все контрольные мероприятия, получает итоговую оценку по дисциплине за семестр в соответствии со шкалой:

Рейтинг	Оценка на экзамене, на дифф. зачете
85 – 100	отлично
71 – 84	хорошо
60 – 70	Удовлетворительно
0 – 59	неудовлетворительно

Оценивание дисциплины ведется в соответствии с Положением о текущем контроле успеваемости и промежуточной аттестации студентов МГТУ им. Н.Э. Баумана.

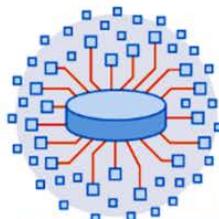
# **Спасибо за внимание**

# **Методы машинного обучения**

*Лекция 2*

*Исходные данные*

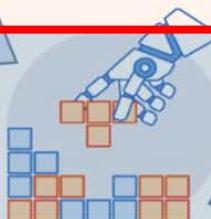
# Предобработка и очистка данных



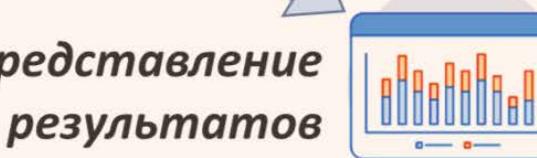
*Извлечение исходных данных*



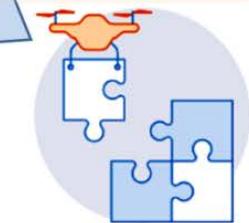
*Предобработка и очистка данных*



*Машинное обучение*



*Интерпретация и представление результатов*



*Интеграция в ИТ-инфраструктуру, бизнес-процессы, повседневную деятельность и т.д.*

# Возможные проблемы исходных данных

- Малый объем обучающей выборки;
- Некорректность входных данных;
  - Неполные
  - Неточные
- Противоречивость данных;
- Разнородность признаков;
- Неструктурированные/отсутствует разметка.

Риски, связанные с постановкой задачи:

- «грязные» данные – заказчик не обеспечивает качество данных;
- Неясные критерии качества модели – заказчик не определился с целями или индикаторами.

# Учет пропусков

- Исключение объектов/признаков, имеющих неполные сведения (удаление строк/столбцов);
- Заполнить при помощи интерполяции;
- Найти в других источниках и тем самым дополнить данные;
- Закодировать пропуски специальным значением;
- Привлечь эксперта в соответствующей предметной области:
  - использование специфичных математических моделей
  - генерация псевдослучайных значений, подчиняющиеся некому распределению с учетом других известных признаков;
  - генерация синтетических данных.

# Увеличение информативности примеров для повышения скорости и эффективности обучения

Еще одной целью предобработки данных, является увеличение информативности примеров для повышения скорости и эффективности обучения. Чем больше бит информации принесет каждый пример, тем лучше используются имеющиеся данные. Среднее количество информации, приносимой каждым примером  $x$ , равно энтропии распределения значений этой компоненты  $H(x)$ . Если эти значения сосредоточены в относительно небольшой области единичного интервала, информационное содержание такой компоненты мало и когда все значения переменной совпадают, эта переменная не несет никакой информации. Напротив, если значения переменной  $x$  равномерно распределены в единичном интервале, информация такой переменной максимальна.

Таким образом, общий принцип предобработки данных состоит в таком кодировании и нормировке непротиворечивых данных, чтобы добиться максимизации энтропии входов и выходов.

# Кодирование входов-выходов

Качественные данные можно разделить на две группы:

- **упорядоченные (ординальные - от англ. order);**
- **неупорядоченные (категориальные).**

В обоих случаях переменная относится к одному из дискретного набора классов  $\{c_1, \dots, c_n\}$ . Но в первом случае эти классы упорядочены, т.е. можно сказать, что  $c_1 < \dots < c_n$ , тогда как во втором такая упорядоченность отсутствует.

# Кодирование - Ординальные переменные

Ординальные переменные близки к числовой форме и для их кодирования достаточно, просто пронумеровать имеющиеся значения переменных числами, таким образом, который бы сохранял существующую упорядоченность. Самым простым способом является установка в соответствие каждому классу своего целого номера, отличающегося на 1 от соседних номеров. Так, например, классу  $c_1$  соответствует номер 1, а классу  $c_n$  - номер  $n$ .

Но может иметься проблема неравномерности выборки, а нам необходимо стремиться к тому, чтобы максимизировать энтропию закодированных данных, что достигается использованием равномерного распределения.

Исходя из этих соображений, единичный отрезок разбивается на  $n$  отрезков, что соответствует числу классов, с длинами пропорциональными числу примеров каждого класса в обучающей выборке:  $\Delta x_k = \frac{P_k}{P}$ , где  $P_k$  - число примеров класса  $k$ , а  $P$  - общее число примеров. Центр каждого такого отрезка будет являться численным значением для соответствующего ординального класса.

# Кодирование – Неупорядоченные (категориальные) переменные

Неупорядоченные (категориальные/номинальные) переменные являются простым обозначением классов.

- Двоичное кодирование (иногда называют one-hot-кодированием). Каждому значению ставится в соответствие собственный атрибут, который может принимать значение 0 или 1, т.е. при наличии значения соответствующий ему атрибут устанавливается равным 1 или в противном случае 0.

Первый класс кодируется как  $(1,0,\dots,0)$ , второй –  $(0,1,0,\dots,0)$  и т.д.

Если кодируемый параметр может принимать значения  $n$  классов, то размерность входного вектора увеличится на  $n-1$  элемент.

- Кодирование  $n$  классов неупорядоченных переменных  $m$ -битным двоичным кодом.

Пример:  $c_1=(0,0)$ ,  $c_2=(0,1)$ ,  $c_3=(1,0)$ ,  $c_4=(1,1)$

Размерность входного вектора увеличится на  $\log_2 n$  элементов.

# Кодирование выходных значений

Кодирование выходных значений направлено на увеличение эффективности обучения и упрощение интерпретации результатов работы модели.

- Метод кодирования выходных значений с помощью двоичного вектора, при использовании которого количество выходных значений равняется количеству классов, где  $i$ -ая компонента вектора соответствует  $i$ -ому классу;
- номер кластера, записанный в двоичной форме, т.е. кодировать  $n$  классов  $m$ -битным двоичным кодом;
- Разбиении задачи с  $n$  классами на  $k$  подзадач с двумя классами каждого.  $k = A_n^2 = n(n - 1)/2$

Выходы	Возможные классы	Класс	Содержится в выходе
1	1-2	1	1,2,3
2	1-3	2	1,4,5
3	1-4	3	2,4,6
4	2-3	4	3,5,6
5	2-4		
6	3-4		

# Нормировка данных

**Стандартизация данных** – это процесс приведения вектора каждого признака к такому виду, что его математическое ожидание станет нулевым, а дисперсия – единичной.

**Нормализация данных** – это процесс масштабирования вектора каждого признака, то есть приведение его к такому виду, что вектор будет иметь единичную норму (при этом есть разные способы оценки\подсчета нормы).

**Линейное преобразование:**

- $\tilde{x}_i = \frac{x_i - x_i^{min}}{x_i^{max} - x_i^{min}}$  в единичный отрезок:  $\tilde{x}_i \in [0,1]$
- $\tilde{x}_i = 2 \frac{x_i - x_i^{min}}{x_i^{max} - x_i^{min}} - 1$  для отображения данных в интервал [-1,1]

**L1 норма:**  $x'_i = \frac{x_i}{\|x\|_1} = \frac{x_i}{\sum_j |x_j|}$ , где  $\|x\|_1$  и есть L1 норма, а вся формула целиком отображает процесс нормализации вектора  $x$ .

**L2 норма:**  $x'_i = \frac{x_i}{\|x\|_2} = \frac{x_i}{\sqrt{\sum_j x_j^2}}$ , где  $\|x\|_2$  и есть L2 норма, а вся формула целиком отображает процесс нормализации вектора  $x$ .

# Нормировка данных на основании статистических характеристик

Другой формой масштабирования является вычисление для каждого признака среднего значения и среднеквадратичного отклонения, т.е. статистических характеристик.

**Выборочное среднее** (несмешённая оценка математического ожидания  $E [ X ]$ ):

$$\bar{X} = \frac{1}{n} \sum_{i=1}^n X_i$$

**Выборочная дисперсия** в математической статистике — это оценка теоретической дисперсии распределения, рассчитанная на основе данных выборки. Пусть  $X = \{x_1 \dots x_n\}$  — выборка из распределения вероятности.

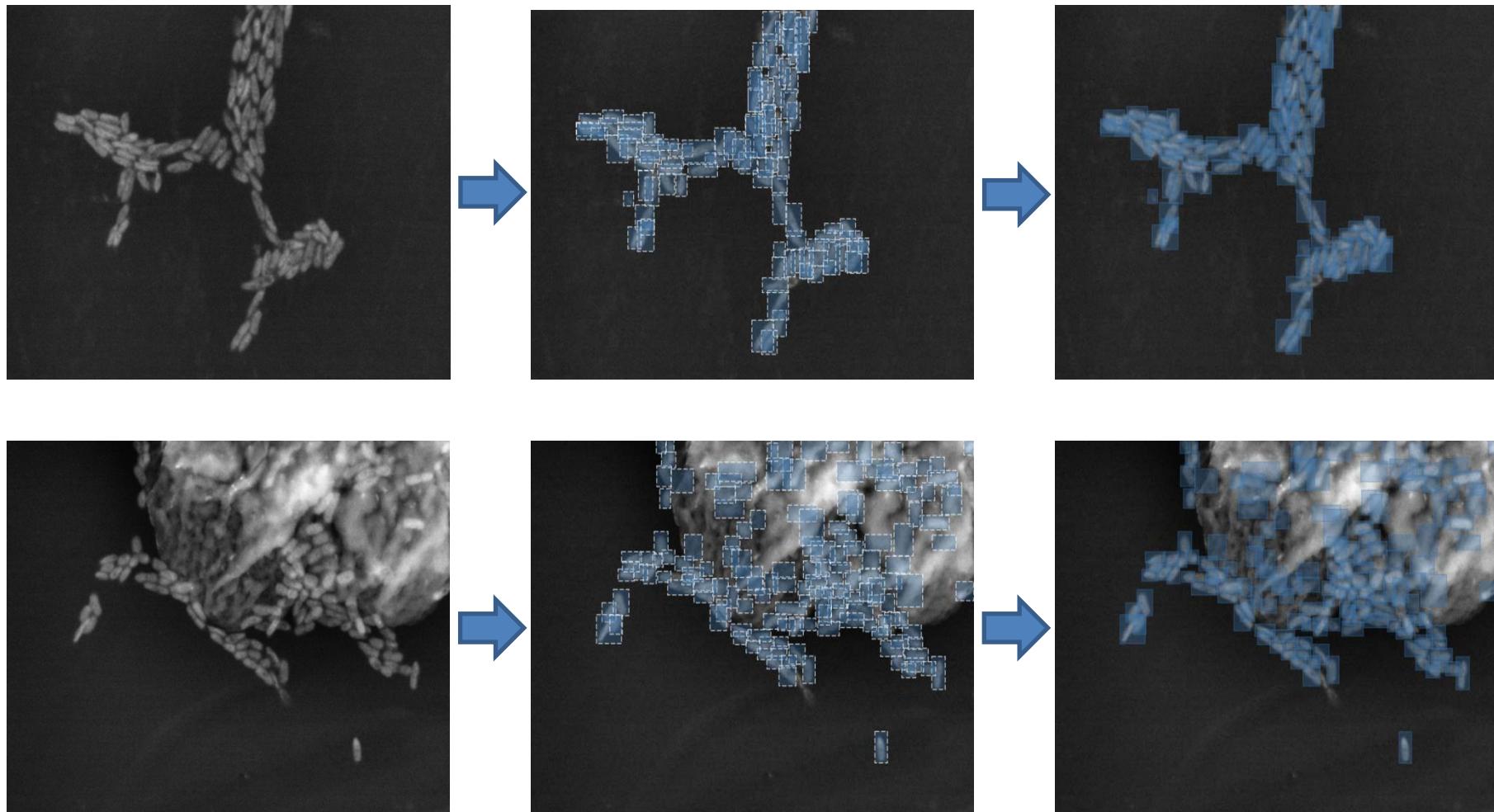
Виды выборочных дисперсий:

- Смешённая:  $S_n^2 = \frac{1}{n} \sum_{i=1}^n (X_i - \bar{X})^2$
- Несмешённая, или исправленная:  $S^2 = \frac{1}{n-1} \sum_{i=1}^n (X_i - \bar{X})^2$

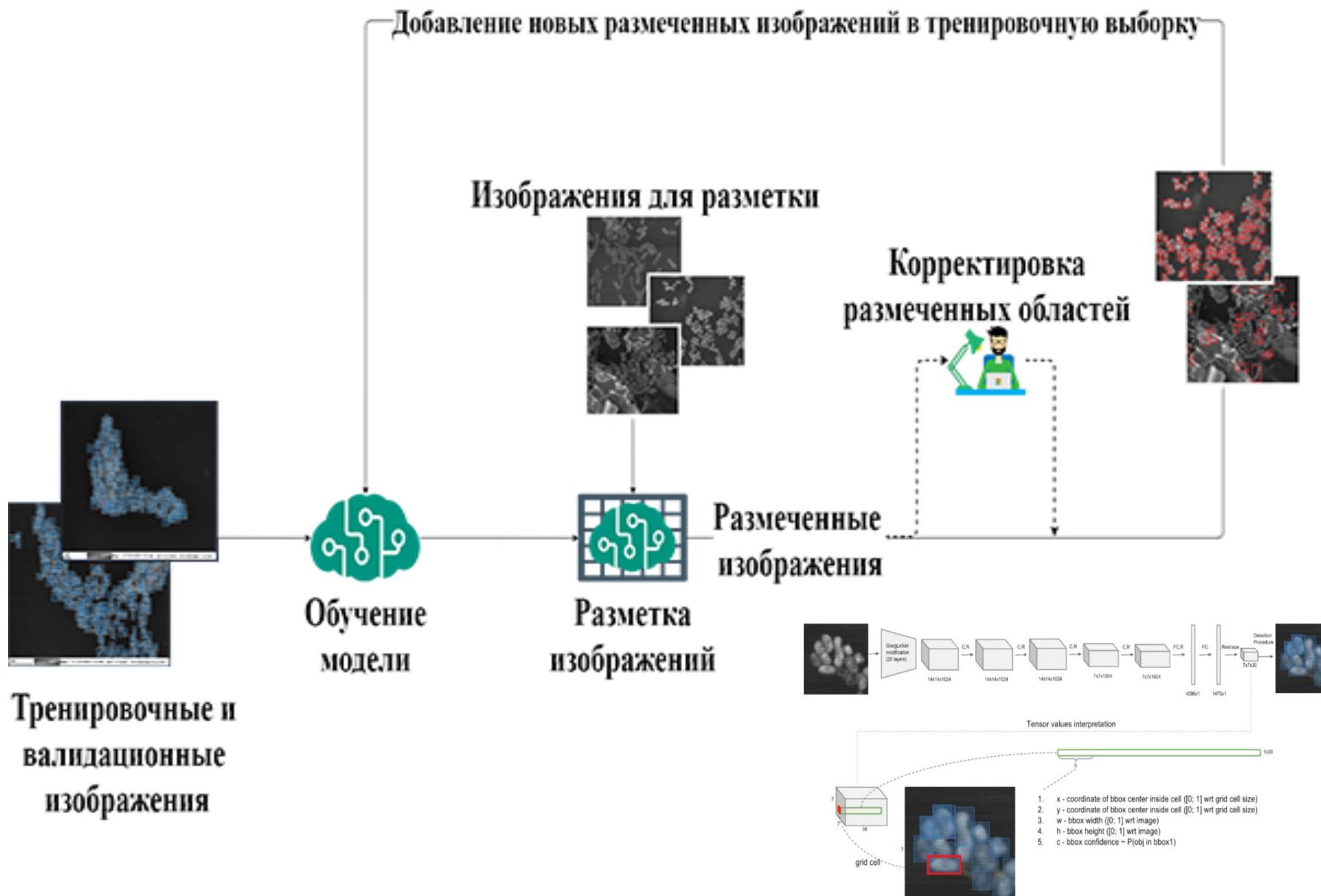
Значения признака масштабируются с помощью вычитания среднего и деления результата на значение среднеквадратичного отклонения для данного признака.

$$\tilde{X}_i = \frac{X_i - \bar{X}}{S^2}$$

# Разметка данных / изображений



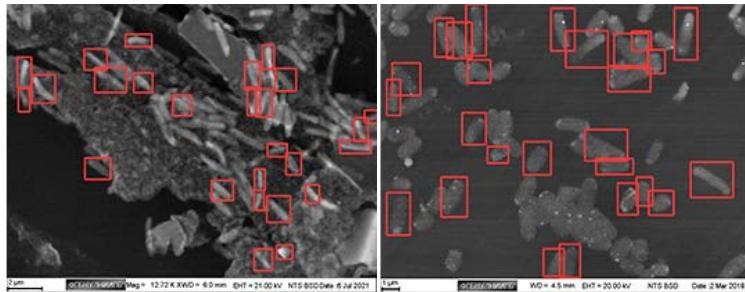
# Автоматизированная разметка изображений



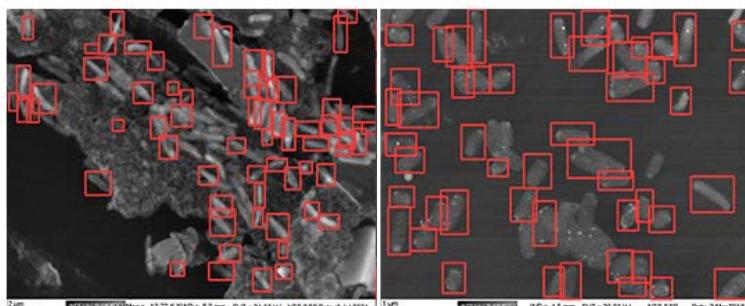
# Обучение моделей для автоматизированной разметки изображений

Результаты разметки бактерий ранга  
*Pseudomonas aeruginosa* и *Escherichia\_coli*

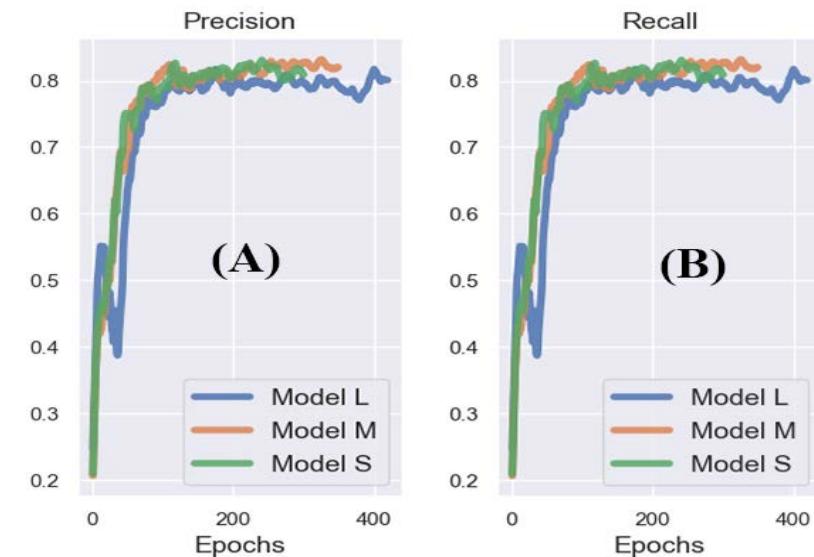
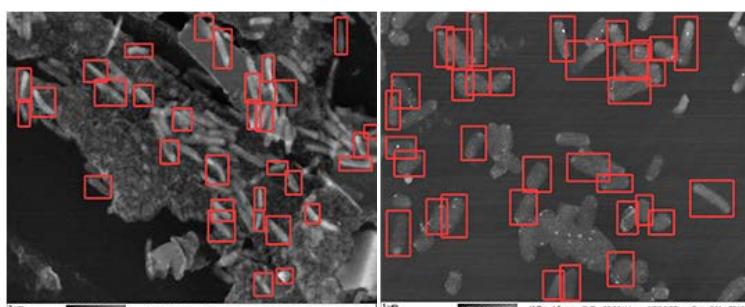
M1



M2



M3



Метрики качества модели	Модели		
	S	M	L
Точность	0.911130	0.920790	0.909750
Полнота	0.836680	0.829020	0.809520
Номер лучшей эпохи	217	257	327
Всего эпох обучения	309	358	428

Метрики качества модели	Модели		
	M1	M2	M3
Точность	0.883990	0.909750	0.911380
Полнота	0.869170	0.809520	0.817360
Номер лучшей эпохи	245	327	9
Всего эпох обучения	328	428	107

# **Теория вероятностей и математическая статистика**

**Теория вероятностей** – раздел математики, в котором изучаются случайные величины и события, их свойства и возможные операции над ними. Таким образом, ключевое понятие, лежащее в основе этой дисциплины, – вероятность события  $P_i$ .

**Математическая статистика (Mathematical statistics)** – это наука о математических методах анализа данных, полученных на основе большого числа наблюдений (измерений, опытов). Большая часть математической статистики основана на **вероятностных моделях**.

Основными задачами математической статистики являются оценивание и проверка гипотез.

# Методы теории вероятностей

Опираясь на понятие вероятности события  $P_i$ , можно дать определение **полной группы событий**, которая определяется как система случайных событий и обладает следующими свойствами:

- В результате случайного эксперимента непременно произойдет одно и только одно из составляющих ее событий;
- Сумма вероятностей всех событий полной группы равна 1.

Среди базовых операций по анализу данных можно выделить подсчеты вероятностных характеристик выборки: медианы, математического ожидания, дисперсии, а также величины среднеквадратического отклонения.

# Вероятностные характеристики выборки

$X = \{x_1 \dots x_n\}$  – изучаемая выборка

- **Медиана** – число, характеризующее выборку по среднему из ее значений. То есть, если все данные выборки различны, и она упорядочена по возрастанию, то ровно половина из элементов выборки будет меньше медианы, и ровно половина – больше.  $x_k$  – медиана, если  $x_j < x_k$ , при  $j \in [1, k)$  и  $x_k < x_i$ , при  $i \in (k, n]$  и  $k=n/2$ .
- **Математическое ожидание** – среднее значение вероятностных элементов выборки. Формально она рассчитывается так:

$$M|X| = \sum x_i p_i$$

- **Выборочное среднее** (несмешённая оценка мат. ожидания  $E[X]$ ).

$$\bar{X} = \frac{1}{n} \sum_{i=1}^n X_i$$

- **Дисперсия** – мера разброса элементов выборки относительно ее математического ожидания. Рассчитывается данная величина по формуле

$$D|X| = \sum (x_i - M|X|)^2 p_i$$

- **Выборочная дисперсия и среднеквадратическое отклонение** – величина, характеризующая рассеивание значений выборки относительно ее математического ожидания. Формула расчета

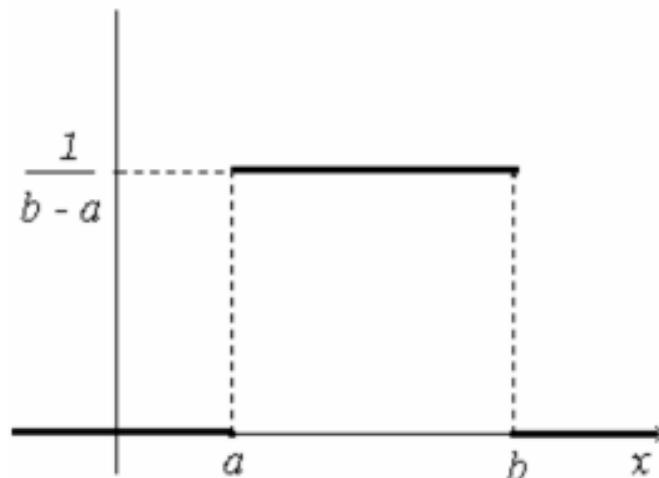
$$\sigma = \sqrt{\frac{1}{n} \sum (x_i - \bar{x})^2}$$

# Непрерывное равномерное распределение

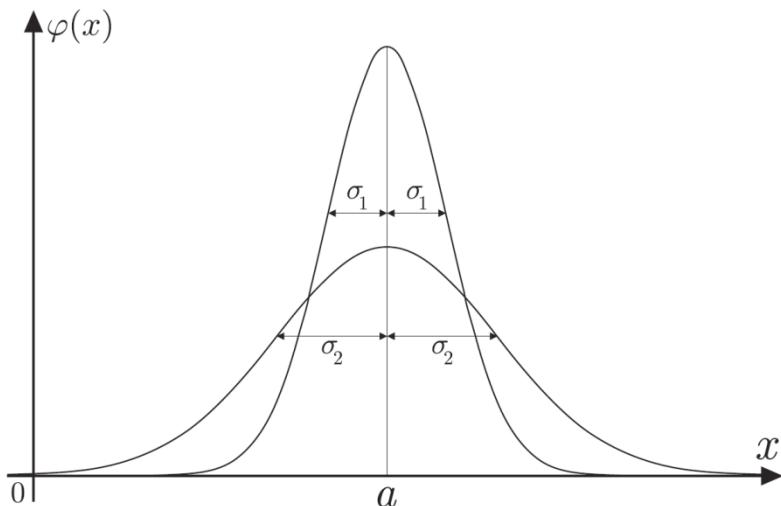
**Непрерывное равномерное распределение** в теории вероятностей — распределение случайной вещественной величины, принимающей значения, принадлежащие некоторому промежутку конечной длины, характеризующееся тем, что плотность вероятности на этом промежутке почти всюду постоянна.

Говорят, что случайная величина имеет непрерывное равномерное распределение на отрезке  $[a,b]$ , где  $a, b \in \mathbb{R}$ , если ее плотность  $f_X(x)$  имеет вид:

$$f_X(x) = \begin{cases} \frac{1}{b-a}, & x \in [a, b] \\ 0, & x \notin [a, b] \end{cases}$$



# Нормальное распределение



**Определение.** Случайная величина  $\xi$  имеет нормальное распределение вероятностей с параметрами  $\mu$  и  $\sigma^2$ , если ее Закон плотности распределения вероятности задается формулой:

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

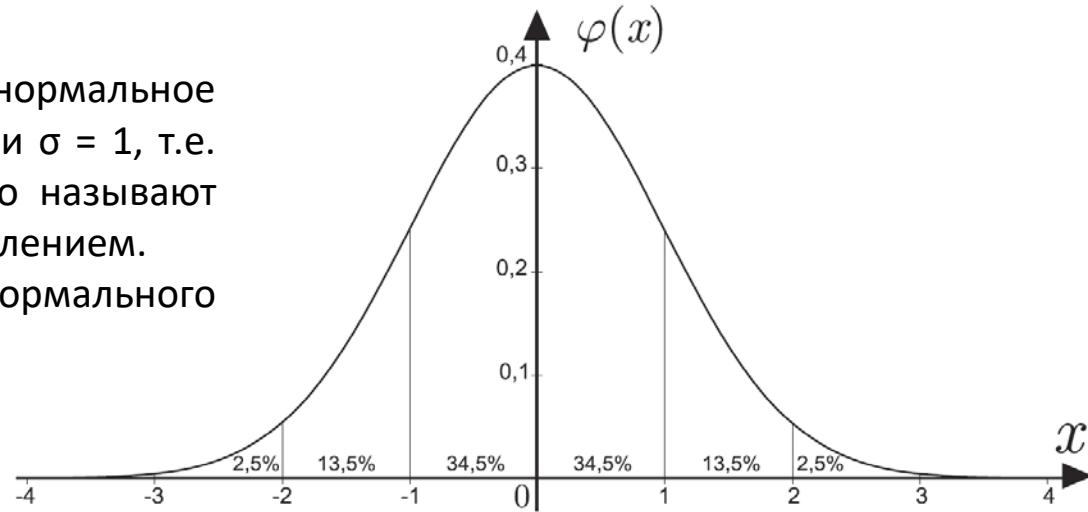
где  $\sigma$  – среднеквадратическое отклонение;  
 $\mu$  – математическое ожидание.

Краткое обозначение:  $\xi \sim N(\mu, \sigma^2)$

Особую роль играет нормальное распределение с параметрами  $\mu = 0$  и  $\sigma = 1$ , т.е. распределение  $N(0, 1)$ , которое часто называют *стандартным* нормальным распределением.

Плотность стандартного нормального распределения:

$$f(x) = \frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}}$$



# Распределения, связанные с нормальным

**Область применения.** При операциях с нормальными случайными величинами, которые приходится проводить при анализе данных, возникает несколько новых видов распределений (и соответствующих им случайных величин). В первую очередь, это распределение **Стьюдента**,  $\chi^2$  и **F-распределения**. Эти распределения играют очень важную роль в прикладном и теоретическом анализе. Так, при выяснении точности и достоверности статистических оценок используются процентные точки распределений Стьюдента и  $\chi^2$ . Распределение статистик многих критериев, использующихся для проверки различных предположений, хорошо приближается этими распределениями.

# Распределение хи-квадрат

**Определение.** Пусть случайные величины  $\xi_1, \xi_2, \dots, \xi_n$  – независимы, и каждое из них имеет стандартное нормальное распределение  $N(0,1)$ . Говорят, что случайная величина  $\chi_n^2$  определенная как:

$$\chi_n^2 = \xi_1^2 + \dots + \xi_n^2$$

имеет распределение хи-квадрат с  $n$  степенями свободы.

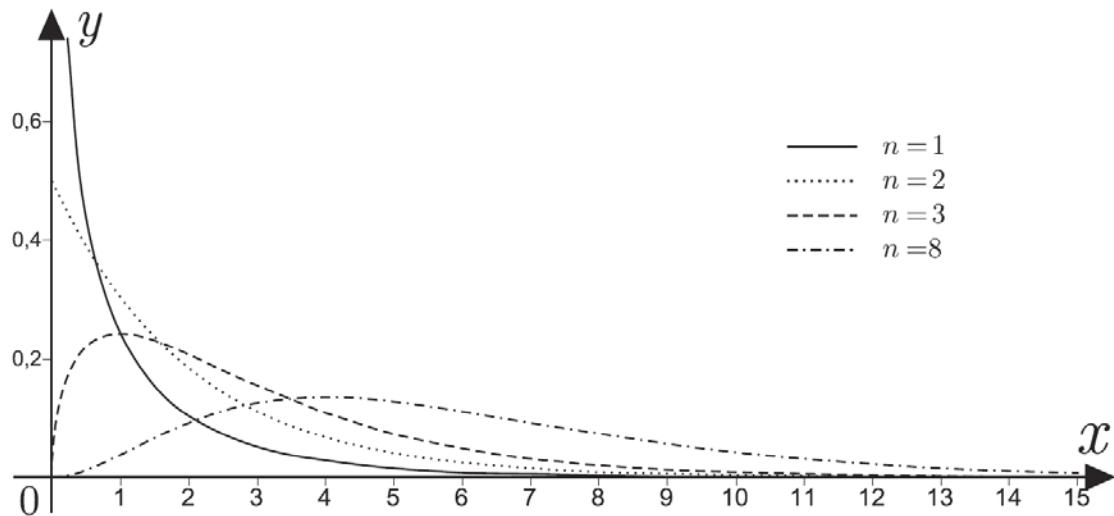
Функция плотности  $\chi_n^2$  в точке  $x$  ( $x > 0$ ) равна:

$$\frac{1}{2^{n/2}} \frac{1}{\Gamma(n/2)} x^{\frac{n}{2}-1} e^{-x/2}$$

где  $\Gamma(\cdot)$  есть гамма функция. На практике эта плотность распределения непосредственно используется редко.

**Свойства.** Математическое ожидание и дисперсия случайной величины  $\chi_n^2$  равны:

$$M\chi_n^2 = n, D\chi_n^2 = 2n.$$



# Распределение Стьюдента

**Определение.** Пусть случайные величины  $\xi_0, \xi_1, \dots, \xi_n$  – независимы, и каждое из них имеет стандартное нормальное распределение  $N(0,1)$ .

Введем случайную величину:  $t_n = \frac{\xi_0}{\sqrt{\frac{1}{n} \sum_{i=1}^n \xi_i^2}}$

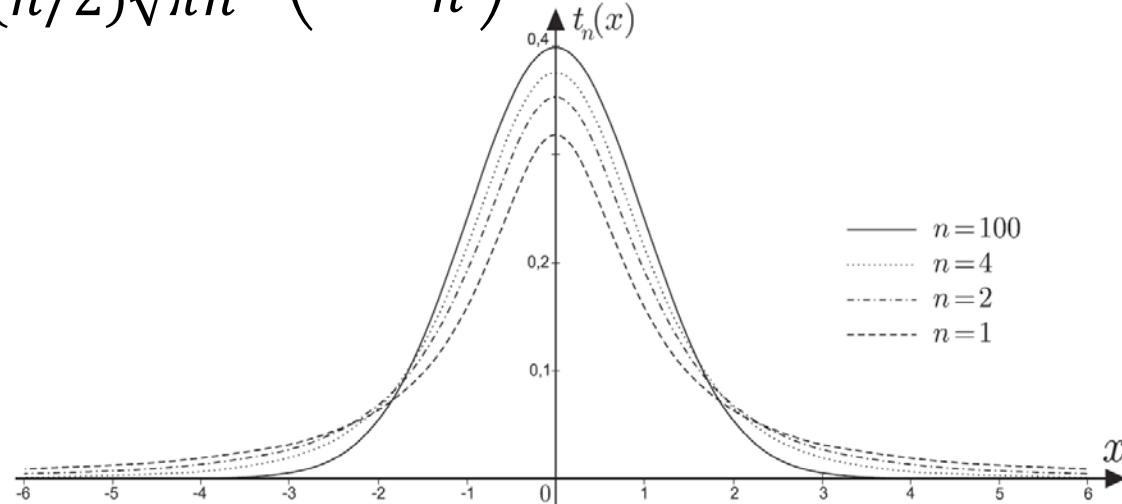
Ее распределение называют распределением Стьюдента. Саму случайную величину часто называют стьюдентовской дробью или стьюдентовым отношением. Число  $n = 1, 2, \dots$  называют числом степеней свободы распределения Стьюдента.

Плотность распределения Стьюдента в точке  $x$  равна:

$$\frac{\Gamma((n+1)/2)}{\Gamma(n/2)\sqrt{\pi n}} \left(1 + \frac{x^2}{n}\right)^{-(n+1)/2}$$

**Свойства:**

$$Mt_n = 0, Dt_n = \frac{n}{n-2}.$$



# F-распределение

**Определение.** Пусть  $\eta_1, \dots, \eta_m; \xi_1, \dots, \xi_n$  (где  $m, n$  – натуральные числа) обозначают независимые случайные величины, каждое из которых распределено по стандартному нормальному закону распределения  $N(0,1)$ . Говорят, что случайная величина  $F_{m,n}$ , определенная как

$$F_{m,n} = \frac{\frac{1}{m}(\eta_1^2 + \dots + \eta_m^2)}{\frac{1}{n}(\xi_1^2 + \dots + \xi_n^2)}$$

имеет F-распределение с параметрами  $m$  и  $n$ . Натуральные числа  $m, n$  называют числами степеней свободы. F-распределение иногда называют еще распределением дисперсионного отношения.

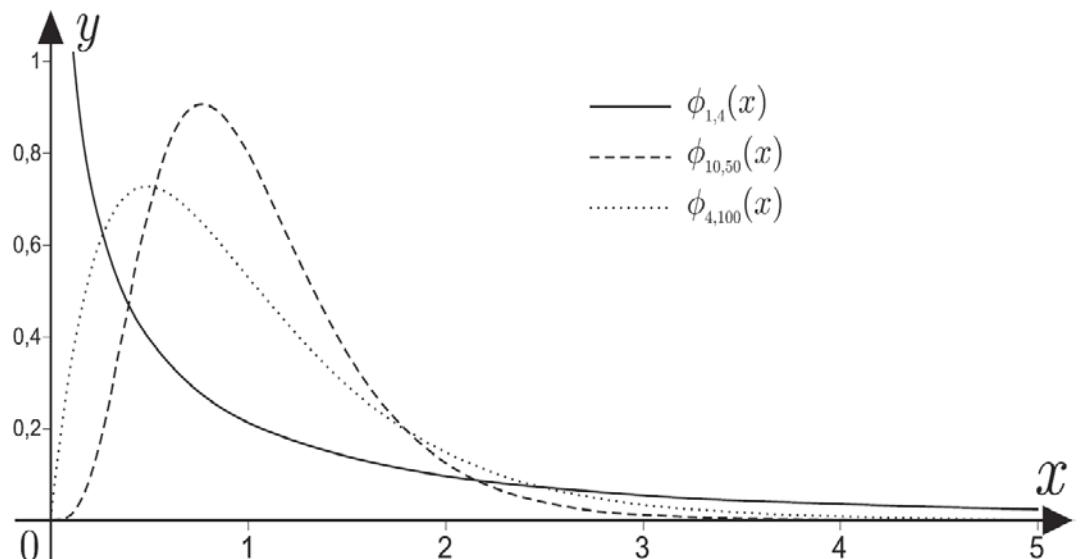
**Свойства.**

Математическое ожидание  
для  $n > 2$  :

$$MF_{m,n} = \frac{n}{n-2}$$

Дисперсия для  $n > 4$ :

$$DF_{m,n} = \frac{2n^2(m+n-2)}{m(n-2)^2(n-4)}$$



**Спасибо за внимание**

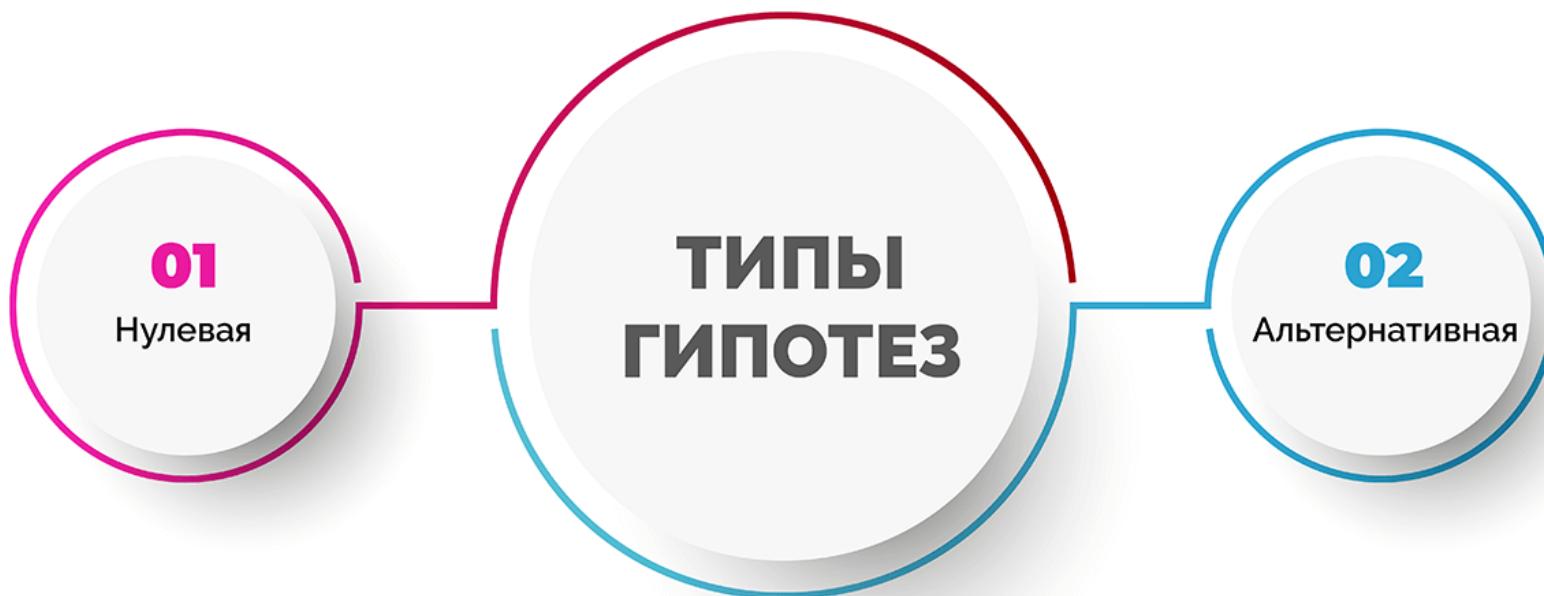
# **Методы машинного обучения**

***Лекция 3***

**Основы статистического  
анализа**

# Статистические гипотезы

Статистическая гипотеза — гипотеза о виде распределения и свойствах случайной величины, которые можно подтвердить или опровергнуть применением статистических методов к данным выборки.



**Нулевая гипотеза** — принимаемое по умолчанию предположение о том, что не существует связи между двумя наблюдаемыми событиями, феноменами. Часто в качестве нулевой гипотезы выступают предположения об отсутствии взаимосвязи или корреляции между исследуемыми переменными, об отсутствии различий (однородности) в распределениях (параметрах распределений) в двух и/или более выборках. Для обозначения нулевой гипотезы часто используют символ  $H_0$ .

# Проверка гипотез

Нулевая гипотеза  $H_0$  считается верной, пока нельзя доказать обратное. Проверка фальсифицируемости нулевой гипотезы — общепринятый способ обеспечения строгости исследования. Если прямого способа проверки у нас нет, приходится прибегать к проверкам косвенным. **Статистика как наука даёт чёткие условия, при наступлении которых нулевая гипотеза может быть отвергнута.**

- Если некоторое явление логически неизбежно следует из гипотезы, но в природе не наблюдается, то это значит, что гипотеза неверна.
- Если происходит то, что при гипотезе происходить не должно, это тоже означает ложность гипотезы.

Заметим, что строго говоря, **косвенным образом доказать гипотезу нельзя, хотя опровергнуть — можно.**

При статистическом выводе исследователь пытается показать несостоятельность нулевой гипотезы, несогласованность её с имеющимися опытными данными, то есть отвергнуть гипотезу. При этом подразумевается, что должна быть принята другая, альтернативная (конкурирующая) гипотеза, исключающая нулевую гипотезу. Если же данные, наоборот, подтверждают нулевую гипотезу, то она не отвергается.

# Проверка статистической значимости

Смысл статистической значимости заключается в том, чтобы определить, имеет ли под собой какое-то основание разница между двумя показателями, или же она случайна.

Выберем уровень вероятности  $\varepsilon > 0$ . Условимся считать событие практически невозможным, если его вероятность меньше  $\varepsilon$ . Когда речь идет о проверке гипотез, число  $\varepsilon$  называют уровнем значимости.

**Уровень значимости** — это вероятность того, что мы сочли различия существенными, в то время как они на самом деле случайны. Уровень значимости показывает степень достоверности выявленных различий между выборками, т.е. показывает, насколько мы можем доверять тому, что различия действительно есть.

**Уровни значимости:**  $p \leq 0,05$ ,  $p \leq 0,01$ ,  $p \leq 0,001$ .

**Определение.** Событие  $A$  называется критическим для гипотезы  $H$ , или критерием для  $H$ . Если  $P(A|H) \leq \varepsilon$ , то  $\varepsilon$  называют гарантированным уровнем значимости критерия  $A$  для  $H$ .

**Критерий для проверки гипотезы** — это решающее правило, отвергающее или принимающее нулевую гипотезу на основе выборочных наблюдений.

# Ошибки первого и второго рода

Возможны ошибки двух родов: первого рода ( $\alpha$ ) и второго рода ( $\beta$ ).

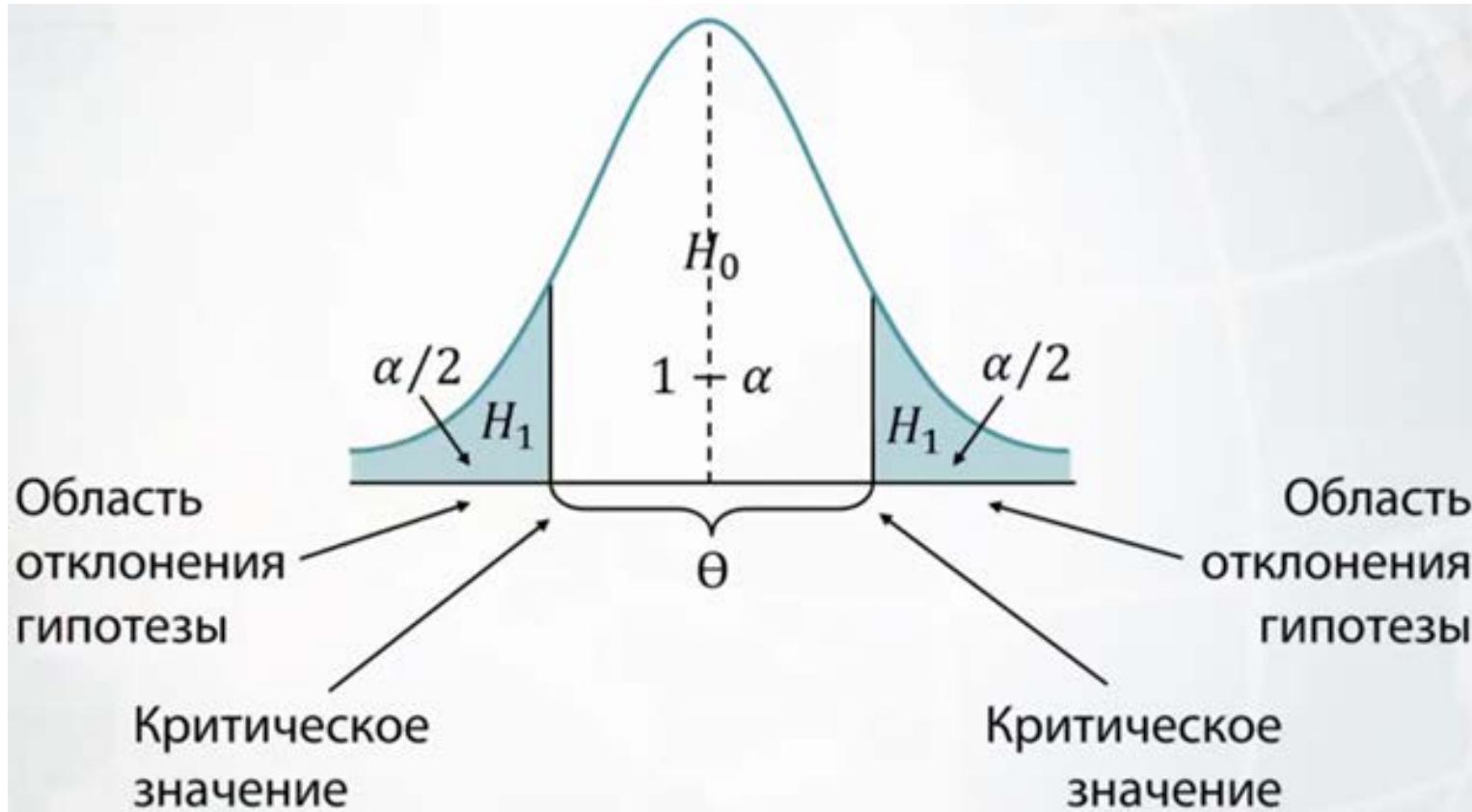
**Ошибка первого рода** состоит в том, что гипотеза  $H_0$  будет отвергнута, хотя на самом деле она правильная. Вероятность допустить такую ошибку называют **уровнем значимости** и обозначают буквой  $\alpha$  («альфа»).

**Ошибка второго рода** состоит в том, что гипотеза  $H_0$  будет принята, но на самом деле она неправильная. Вероятность совершить эту ошибку обозначают буквой  $\beta$  («бета»). Значение  $(1-\beta)$  называют **мощностью критерия** – это вероятность отклонения ложной гипотезы, т.е. способность выявлять даже мелкие различия. Чем мощнее критерий, тем лучше он отвергает нулевую гипотезу.

В практических задачах, как правило, задают **уровень значимости**, и наиболее часто выбирают значения:  $\alpha=0.1$ ,  $\alpha=0.05$ ,  $\alpha=0.01$ .

Нулевая гипотеза $H_0$	Ложная	Истинная
Отклоняется	Ошибки нет ( $1-\beta$ )	Ошибка I рода (ложноположительный вывод), ложная тревога ( $\alpha$ )
Принимается <i>(не отклоняется)</i>	Ошибка II рода (ложно-отрицательный вывод), пропуск цели ( $\beta$ )	Ошибки нет ( $1-\alpha$ )

# Области принятия и отклонения гипотезы для двусторонней альтернативной гипотезы



# Методика проверки статистических гипотез

Пусть задана случайная выборка  $X^m = (x_1, \dots, x_m)$  — последовательность  $m$  объектов из множества  $X$ . Предполагается, что на множестве  $X$  существует некоторая неизвестная вероятностная мера  $P$ .

1. Формулируются нулевая  $H_0$  и альтернативная  $H_1$  гипотезы о распределении вероятностей на множестве  $X$ .
2. Задаётся некоторая статистика (функция выборки — на след. слайде)  $T: X^m \rightarrow \mathbb{R}$ , для которой в условиях справедливости гипотезы  $H_0$  выводится функция распределения  $F(T)$  и/или плотность распределения  $p(T)$ .
3. Фиксируется уровень значимости — допустимая для данной задачи вероятность ошибки первого рода, то есть того, что гипотеза на самом деле верна, но будет отвергнута процедурой проверки. Это должно быть достаточно малое число  $\alpha \in [0,1]$ . На практике часто полагают  $\alpha=0.05$ .
4. На множестве допустимых значений статистики  $T$  выделяется *критическое множество*  $\Omega_\alpha$  наименее вероятных значений статистики  $T$ , такое, что  $P\{T \in \Omega_\alpha | H_0\} = \alpha$ .
5. Собственно *статистический тест* (*статистический критерий*) заключается в проверке условия:
  - Если  $T(X^m) \in \Omega_\alpha$ , то делается вывод «данные противоречат нулевой гипотезе при уровне значимости  $\alpha$ ». Гипотеза отвергается.
  - Если  $T(X^m) \notin \Omega_\alpha$ , то делается вывод «данные не противоречат нулевой гипотезе при уровне значимости  $\alpha$ ». Гипотеза принимается.

# Статистика (функция выборки)

Статистикой называется произвольная измеримая функция выборки  $T: X^m \rightarrow \mathbb{R}$ , которая не зависит от неизвестных параметров распределения.

Условие измеримости статистики означает, что эта функция является случайной величиной. Независимость этой функции от неизвестных параметров заключается в том, что исследователь может по имеющимся в его распоряжении данным найти значение этой функции, а следовательно — основываться на этом значении оценки и прочие статистические выводы.

Пример. Статистики, используемые для оценки моментов (выборочные моменты)

Выборочное среднее:	$\bar{x} = \frac{1}{m} \sum_{i=1}^m x_i$
Выборочная дисперсия:	$s^2 = s_m^2 = \frac{1}{m} \sum_{i=1}^m (x_i - \bar{x})^2$
Несмешённая оценка дисперсии:	$s^2 = s_m^2 = \frac{1}{m-1} \sum_{i=1}^m (x_i - \bar{x})^2$
Выборочный момент $k$ -го порядка (выборочное среднее — момент первого порядка):	$M_k = \frac{1}{m} \sum_{i=1}^m x_i^k$
Выборочный центральный момент $k$ -го порядка (выборочная дисперсия — центральный момент второго порядка):	$\widetilde{M}_k = \frac{1}{m} \sum_{i=1}^m (x_i - \bar{x})^k$

# Замечание о проверке статистических гипотез

**Важно!** Если данные не противоречат нулевой гипотезе, это ещё не значит, что гипотеза верна.

Тому есть две причины:

- По мере увеличения длины выборки нулевая гипотеза может сначала приниматься, но потом выявятся более тонкие несоответствия данных гипотезе, и она будет отвергнута. То есть многое зависит от объёма данных, если данных не хватает, можно принять даже самую неправдоподобную гипотезу.
- Выбранная статистика  $T$  может отражать не всю информацию, содержащуюся в гипотезе  $H_0$ . В таком случае увеличивается вероятность ошибки второго рода — нулевая гипотеза может быть принята, хотя на самом деле она не верна.

# Критерий согласия Пирсона Хи-квадрат (Chi-square test)

Критерий согласия проверяет, согласуется ли заданная выборка с заданным фиксированным распределением (семейством распределений) или с другой выборкой, т.е. критерий согласия для проверки гипотезы о законе распределения.

**Критерий хи-квадрат** — любая статистическая проверка гипотезы, в которой выборочное распределение критерия имеет распределение хи-квадрат при условии верности нулевой гипотезы.

Пусть  $X$  — исследуемая случайная величина. Требуется проверить гипотезу  $H_0$  о том, что данная случайная величина подчиняется закону распределения  $F(x)$ . Данна выборка  $X^n = (x_1, \dots, x_n)$ ,  $x_i \in [a, b]$ ,  $\forall i = 1 \dots n$ . Необходимо построить эмпирический закон распределения  $F'(x)$  случайной величины  $X$ .

- Разделим  $[a, b]$  на  $k$  непересекающихся интервалов  $[a_i, b_i], i=1 \dots k$ .
- Пусть  $n_j$  — количество наблюдений в  $j$ -м интервале;
- $p_j = F(b_j) - F(a_j)$  — вероятность попадания наблюдения в  $j$ -й интервал при выполнении гипотезы  $H'_0$ ;
- $E_j = np_j$  — ожидаемое число попаданий в  $j$ -й интервал;
- тогда распределение Хи-квадрат с числом степеней свободы  $k-1$  будет иметь следующую статистику (критерий Хи-квадрат Пирсона):

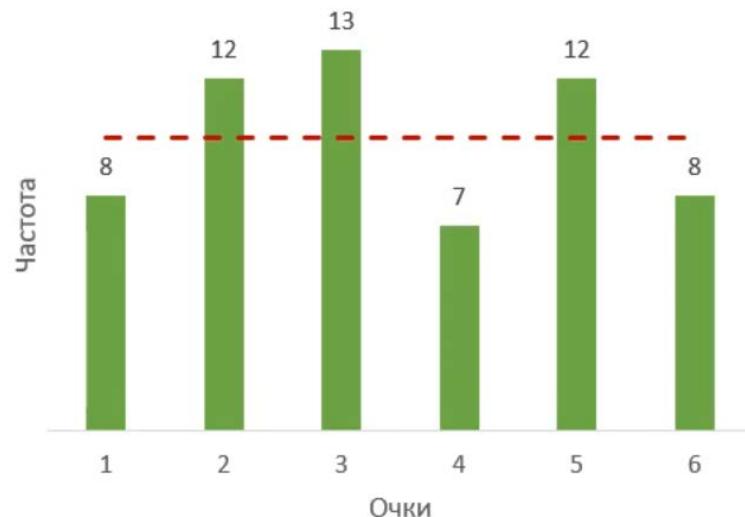
$$\chi^2 = \sum_{j=1}^k \frac{(n_j - E_j)^2}{E_j} \sim \chi^2_{k-1}$$

В зависимости от значения критерия  $\chi^2$  гипотеза  $H_0$  может приниматься либо отвергаться:

$\chi_1^2 < \chi^2 < \chi_2^2$  — гипотеза  $H_0$  выполняется.

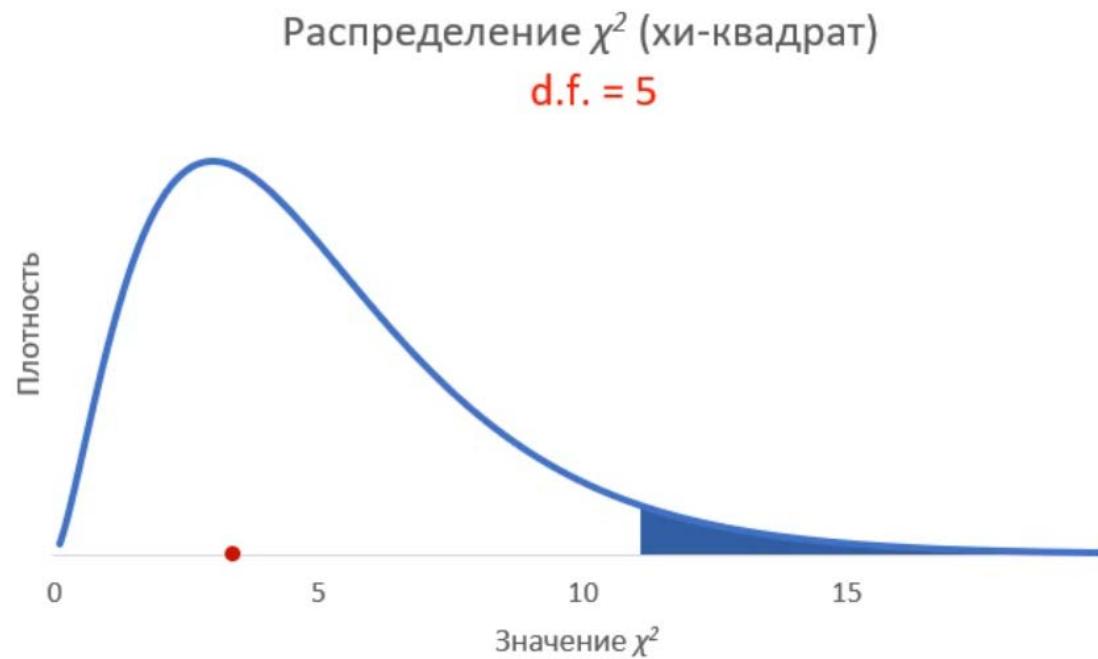
$\chi^2 \geq \chi_2^2$  — попадает в правый «хвост» распределения, гипотеза  $H_0$  отвергается.

# Пример: Критерий согласия Пирсона Хи-квадрат

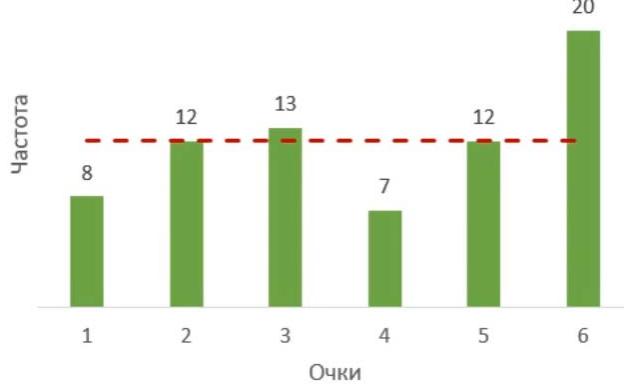


Очки	Частота		
	Наблюдаемая O	Ожидаемая E	
1	8	10	0,4
2	12	10	0,4
3	13	10	0,9
4	7	10	0,9
5	12	10	0,4
6	8	10	0,4
Итого	60		3,4

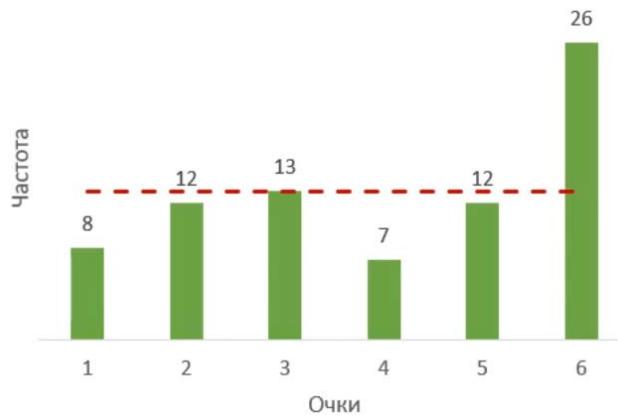
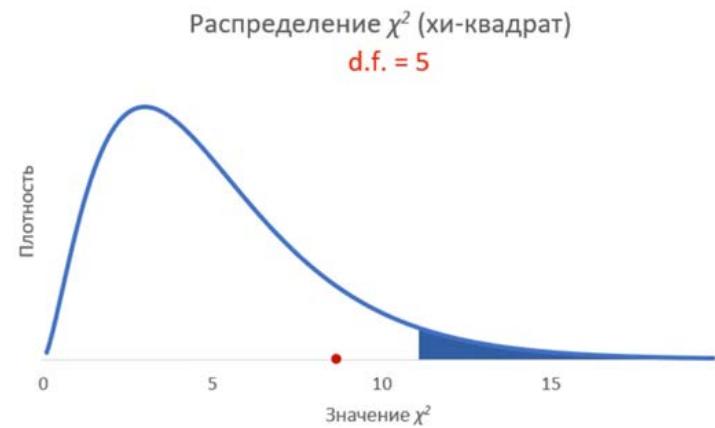
d.f.	5
$\chi^2$	3,4
$\chi^2_{0,05; 5}$	11,0705
p-value	0,63857
Тест $\chi^2$	0,63857



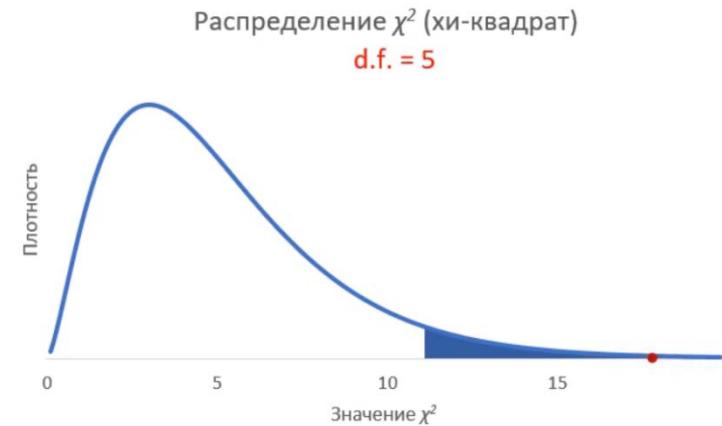
# Пример: Критерий согласия Пирсона Хи-квадрат



d.f.	5
$\chi^2$	8,83333
$\chi^2_{0,05; 5}$	11,0705
p-value	0,1159
Тест $\chi^2$	0,1159



d.f.	5
$\chi^2$	17,8462
$\chi^2_{0,05; 5}$	11,0705
p-value	0,00315
Тест $\chi^2$	0,00315



# Проверка гипотезы о математическом ожидании

Может быть два варианта, когда дисперсия известна и когда неизвестна.  
Рассмотрим случай, когда она неизвестна.

## Сравнение выборочного среднего с заданным значением

Задана выборка  $x^m = (x_1, \dots, x_m)$ ,  $x_i \in \mathbb{R}$ ,  $x_i \sim N(\mu, \sigma^2)$ .

**Дополнительное предположение:** выборка простая и нормальная.

**Нулевая гипотеза**  $H_0: \bar{x} = \mu_0$  (выборочное среднее равно заданному числу  $\mu_0$ ),  
альтернативная  $H_1: \bar{x} \neq \mu_0$

**Статистика критерия:**

$$t = \frac{(\bar{x} - \mu_0)}{\frac{s}{\sqrt{m}}}$$

имеет t-распределение Стьюдента (числитель-нормальное/знаменатель хи-квадрат) с  $m-1$  степенями свободы, где:

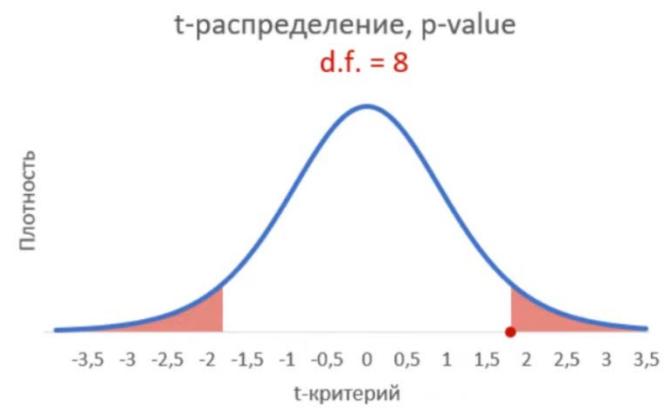
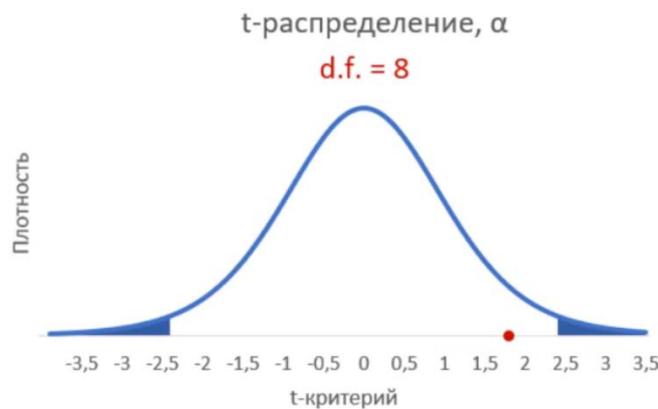
- $\bar{x} = \frac{1}{m} \sum_{i=1}^m x_i$  — выборочное среднее,
- $s^2 = \frac{1}{m-1} \sum_{i=1}^m (x_i - \bar{x})^2$  — выборочная дисперсия.

**Критерий** (при уровне значимости  $\alpha$ ):

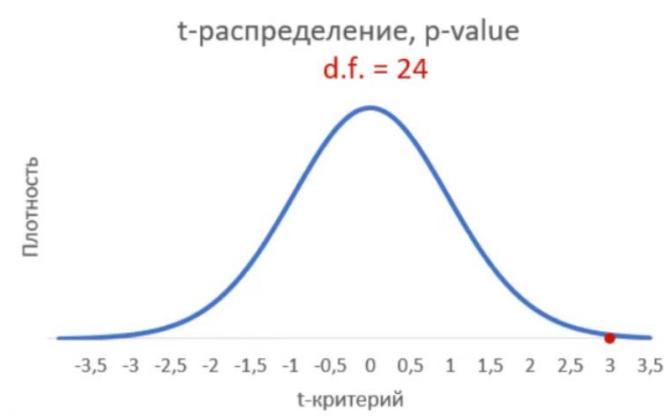
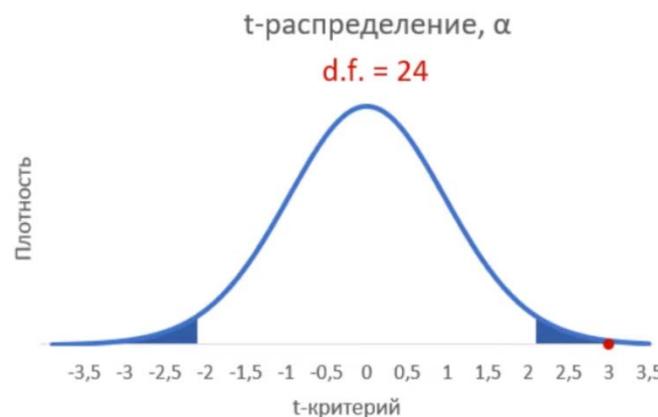
Гипотеза  $H_0$  принимается, если  $|t| < t_{1-\alpha/2}$ , в противном случае отвергается.

# Пример: Проверка гипотезы о математическом ожидании

Показатель	Значение
$\mu$	50
$X_{\text{ср}}$	50,3
n	9
s	0,5
tфакт	1,8
$\alpha$	0,05
d.f.	8
tкритич	2,306
p-value	0,10955



Показатель	Значение
$\mu$	50
$X_{\text{ср}}$	50,3
n	25
s	0,5
tфакт	3
$\alpha$	0,05
d.f.	24
tкритич	2,064
p-value	0,00621



# Проверка гипотезы о математическом ожидании - Две выборки

Специальный случай двухвыборочных критериев согласия. Проверяется гипотеза сдвига, согласно которой распределения двух выборок имеют одинаковую форму и отличаются только сдвигом на константу.

**Критерий Стьюдента.** Рассмотрим теперь задачу сравнения средних значений двух нормальных выборок.

Пусть  $x_1, \dots, x_n; y_1, \dots, y_m$  — нормальные независимые выборки из законов распределения с параметрами  $(a_1, \sigma_1^2)$  и  $(a_2, \sigma_2^2)$  соответственно.

Рассмотрим проверку гипотезы:

$$H_0: a_1 = a_2 \text{ против альтернативы } a_1 \neq a_2$$

Относительно параметров  $\sigma_1^2$  и  $\sigma_2^2$  выделим следующие четыре варианта предположений:

- а) обе дисперсии известны и равны между собой;
- б) обе дисперсии известны, но не равны между собой;
- в) обе дисперсии неизвестны, но предполагается, что они равны между собой;
- г) обе дисперсии неизвестны, их равенство не предполагается.

# Проверка гипотезы о математическом ожидании - Две выборки

Для построения критерия проверки гипотезы  $H_0$  проведем следующие рассуждения.

От выборок  $x_1, \dots, x_n$  и  $y_1, \dots, y_m$  перейдем к выборочным средним  $\bar{x}$  и  $\bar{y}$ . Согласно свойствам нормального распределения и выдвинутой гипотезе, величины  $\bar{x}$  и  $\bar{y}$  имеют нормальные распределения с одними тем же средним и дисперсиями  $\sigma_1^2/n$  и  $\sigma_2^2/m$ .

Далее перейдем к статистике, основанной на выборочных средних  $\bar{x}$  и  $\bar{y}$  и дисперсиях  $\sigma_1^2$  и  $\sigma_2^2$  (если они известны) или их оценках  $s_1^2$  и  $s_2^2$  (если дисперсии неизвестны).

$$\bar{x} = \frac{1}{m} \sum_{i=1}^m x_i \text{ — выборочное среднее,}$$

$$s^2 = \frac{1}{m-1} \sum_{i=1}^m (x_i - \bar{x})^2 \text{ — выборочная дисперсия.}$$

# Проверка гипотезы о математическом ожидании - Дисперсии известны

**а)** Обе дисперсии известны и равны между собой;

$$\frac{(\bar{x} - \bar{y})}{\sigma \sqrt{\frac{1}{n} + \frac{1}{m}}}$$

Статистика имеет стандартное нормальное распределение, так как является линейной комбинацией независимых нормальных величин. Гипотеза  $H_0$  принимается на уровне значимости  $\alpha$ , если

$$\left| \frac{(\bar{x} - \bar{y})}{\sigma \sqrt{\frac{1}{n} + \frac{1}{m}}} \right| < z_{1-\alpha/2}$$

в противном случае гипотеза отвергается в пользу альтернативы  $a_1 \neq a_2$

**б)** Обе дисперсии известны, но не равны между собой;

$$\frac{(\bar{x} - \bar{y})}{\sqrt{\frac{\sigma_1^2}{n} + \frac{\sigma_2^2}{m}}}$$

Статистика имеет также стандартное нормальное распределение. Правило принятия гипотезы аналогично правилу пункта а).

# Проверка гипотезы о математическом ожидании - Дисперсии неизвестны

в) Обе дисперсии неизвестны, но предполагается, что они равны между собой;

В случае, когда обе дисперсии неизвестны, но предполагаются равными между собой, мы имеем две оценки  $s_1^2$  и  $s_2^2$  одной и той же величины дисперсии  $\sigma_1^2 = \sigma_2^2$ . В связи с этим разумно перейти к объединенной оценке:  $S^2 = \frac{s_1^2(n-1)+s_2^2(m-1)}{(n-1)+(m-1)}$

Критерий для проверки гипотезы  $H_0: a_1 = a_2$  опирается на статистику  $\frac{(\bar{x}-\bar{y})}{s \sqrt{\frac{1}{n} + \frac{1}{m}}}$ ,

которая имеет распределение Стьюдента с  $n+m-2$  степенями свободы.

г) Обе дисперсии неизвестны, их равенство не предполагается.

В случае неизвестных дисперсий, равенство которых не предполагается, используется аналог статистики пункта б) с заменой неизвестных дисперсий их оценками:  $\frac{(\bar{x}-\bar{y})}{\sqrt{\frac{s_1^2}{n} + \frac{s_2^2}{m}}}$

В этой ситуации указать точное распределение введенной статистики затруднительно. Известно, однако, что это распределение близко к распределению

Стьюдента с числом степеней свободы, равным:  $\frac{(s_1^2/n+s_2^2/m)^2}{\frac{(s_1^2/n)^2}{n-1} + \frac{(s_2^2/m)^2}{m-1}}$

Критерий проверки гипотезы устроен так же, как и в пункте в).

# P-value или р-значение

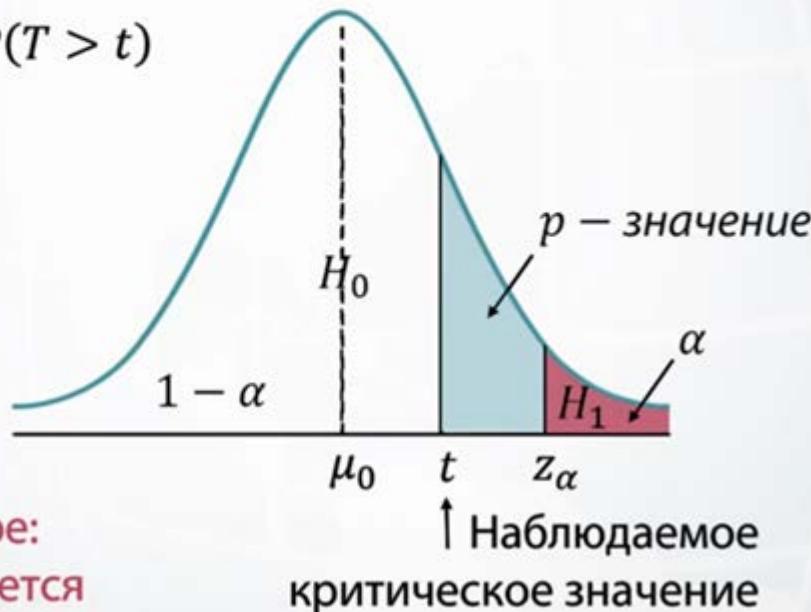
P-value или р-значение – одна из ключевых величин, используемых в статистике при тестировании гипотез. Она показывает вероятность получения наблюдаемых результатов при условии, что нулевая гипотеза верна, или вероятность ошибки в случае отклонения нулевой гипотезы.

Если р-значение  $\geq \alpha$ , то  $H_0$  не отвергается.

Если р-значение  $< \alpha$ , то  $H_0$  отвергается пользу  $H_1$ .

$$H_1 : \mu > \mu_0$$

$$p\text{-значение} = P(T > t)$$



В этом примере:

$H_0$  не отвергается

↑ Наблюданное  
критическое значение

# Доверительный интервал (Confidence interval)

В математической статистике — интервал, в пределах которого с заданной вероятностью лежат выборочные оценки статистических характеристик генеральной совокупности.

Если оценку среднего требуется связать с **определенной вероятностью**, то интересующий параметр генеральной совокупности нужно оценивать не одним числом, а интервалом. Доверительным интервалом называют интервал, в котором с определённой вероятностью  $P$  находится значение оцениваемого показателя генеральной совокупности. Доверительный интервал, в котором с вероятностью  $P = 1 - \alpha$  ( $\alpha$  - квантиль стандартного нормального распределения) находится случайная величина  $\bar{X}$ , рассчитывается следующим образом:

$$\bar{X} - z_{1-\frac{\alpha}{2}} \cdot \frac{\sigma}{\sqrt{n}} \leq \mu \leq \bar{X} + z_{1-\frac{\alpha}{2}} \cdot \frac{\sigma}{\sqrt{n}}$$

где  $z_{1-\frac{\alpha}{2}}$  - критическое значение стандартного нормального распределения для уровня значимости  $\alpha = 1 - P$ , которое можно найти в приложении к практически любой книге по статистике.

# Доверительный интервал (Confidence interval)

На практике среднее значение генеральной совокупности  $\mu$  и дисперсия  $\sigma^2$  не известны, поэтому дисперсия генеральной совокупности заменяется дисперсией выборки  $S^2$ , а среднее генеральной совокупности - средним значением выборки  $\bar{X}$ . Таким образом, доверительный интервал в большинстве случаев рассчитывается так:

$$\bar{X} - t_{1-\frac{\alpha}{2}, n-1} \cdot \frac{S}{\sqrt{n}} \leq \mu \leq \bar{X} + t_{1-\frac{\alpha}{2}, n-1} \cdot \frac{S}{\sqrt{n}}$$

где  $S$  -несмешённое выборочное стандартное отклонение, имеет распределение Стьюдента с  $n-1$  степенями свободы  $t(n-1)$ . Пусть  $t_{\alpha, n-1}$  —  $\alpha$ -квантили распределения Стьюдента.

Формулу доверительного интервала можно использовать для оценки среднего генеральной совокупности, если

- известно стандартное отклонение генеральной совокупности;
- или стандартное отклонение генеральной совокупности не известно, но объём выборки - больше 30.

# Пример: Доверительный интервал

**Пример 1.** Собрана информация из 100 случайно выбранных кафе в некотором городе о том, что среднее число работников в них составляет 10,5 со стандартным отклонением 4,6. Определить доверительный интервал 95% числа работников кафе.

Решение:

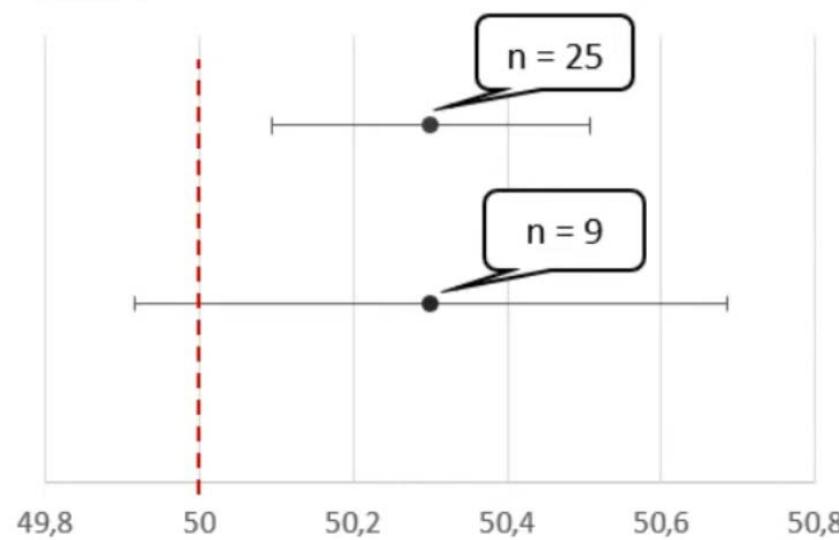
$$10,5 - 1,96 \cdot \frac{4,6}{\sqrt{100}} \leq \mu \leq 10,5 + 1,96 \cdot \frac{4,6}{\sqrt{100}}$$

где  $z_{0,05} = 1,96$  - критическое значение стандартного нормального распределения для уровня значимости  $\alpha = 0,05$ .

Таким образом, доверительный интервал 95% среднего числа работников кафе составил от 9,6 до 11,4.

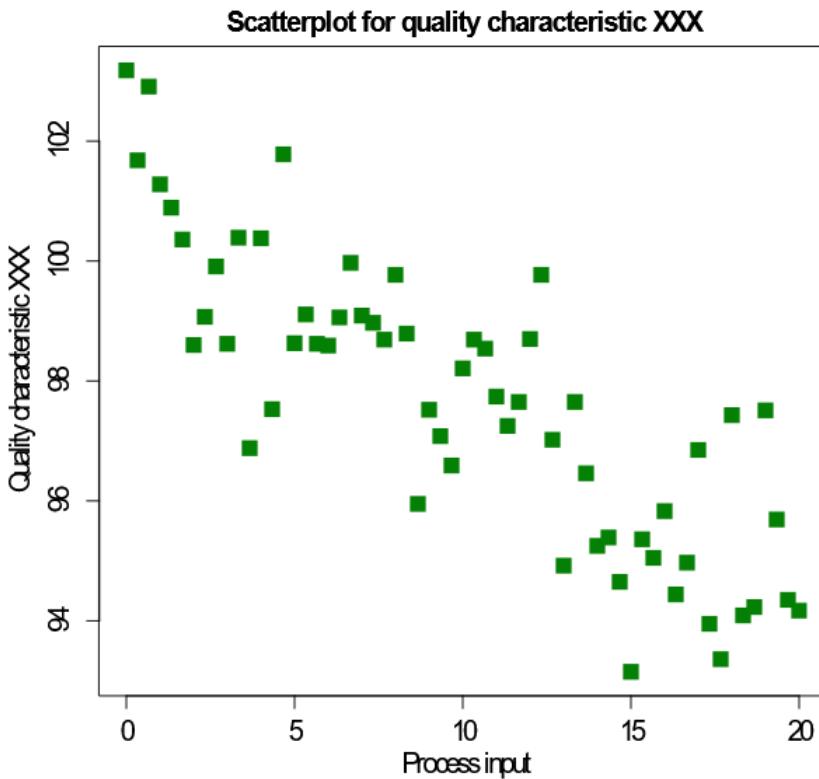
**Пример 2. - со слайда 14.**

Показатель	$n = 9$	$n = 25$
$\mu$	50	50
$X_{ср}$	50,3	50,3
$n$	9	25
$s$	0,5	0,5
$\alpha$	0,05	0,05
$\Delta_x$	0,384	0,206
$\mu_1$	49,916	50,094
$\mu_2$	50,684	50,506



# Корреляция - определение

Корреляция (от лат. *correlatio* «соотношение, взаимосвязь»), или корреляционная зависимость — статистическая взаимосвязь двух или более случайных величин (либо величин, которые можно с некоторой допустимой степенью точности считать таковыми). При этом изменения значений одной или нескольких из этих величин сопутствуют систематическому изменению значений другой или других величин.



	1	2	3	4	5	6	7	8	9	10	11	12
1		0,27	-0,28	0,38	-0,45	0,45	-0,08	-0,02	-0,16	-0,25	0,05	-0,18
2	0,27		-0,43	0,45	-0,29	0,26	0,41	-0,30	0,32	0,27	-0,25	0,12
3	-0,28	-0,43		-0,91	0,40	-0,37	-0,47	0,49	-0,47	-0,28	0,50	-0,29
4	0,38	0,45	-0,91		-0,48	0,45	0,45	-0,49	0,44	0,11	-0,45	0,13
5	-0,45	-0,29	0,40	-0,48		-0,94	-0,12	0,25	-0,18	0,20	0,15	0,27
6	0,45	0,26	-0,37	0,45	-0,94		0,01	-0,10	0,07	-0,23	0,00	-0,31
7	-0,08	0,41	-0,47	0,45	-0,12	0,01		-0,71	0,82	0,44	-0,65	0,28
8	-0,02	-0,30	0,49	-0,49	0,25	-0,10	-0,71		-0,75	-0,52	0,81	-0,42
9	-0,16	0,32	-0,47	0,44	-0,18	0,07	0,82	-0,75		0,55	-0,58	0,28
10	-0,25	0,27	-0,28	0,11	0,20	-0,23	0,44	-0,52	0,55		-0,47	0,62
11	0,05	-0,25	0,50	-0,45	0,15	0,00	-0,65	0,81	-0,58	-0,47		-0,65
12	-0,18	0,12	-0,29	0,13	0,27	-0,31	0,28	-0,42	0,28	0,62	-0,65	

# Корреляция и взаимосвязь величин

Значительная корреляция между двумя случайными величинами всегда является свидетельством существования некоторой статистической связи в данной выборке, но эта связь не обязательно должна наблюдаться для другой выборки и иметь причинно-следственный характер.

- Рассматривая пожары в конкретном городе, можно выявить весьма высокую корреляцию между ущербом, который нанёс пожар, и количеством пожарных, участвовавших в ликвидации пожара, причём эта корреляция будет положительной.
- Из этого не следует вывод «увеличение количества пожарных приводит к увеличению причинённого ущерба», и тем более не будет успешной попытка минимизировать ущерб от пожаров путём ликвидации пожарных бригад.

Отсутствие корреляции между двумя величинами ещё не значит, что между ними нет никакой связи. Например, зависимость может иметь сложный нелинейный характер, который корреляция не выявляет.

# Положительная и отрицательная корреляция

- Некоторые виды коэффициентов корреляции могут быть положительными или отрицательными.
  - В первом случае предполагается, что мы можем определить только наличие или отсутствие связи, а во втором — также и её направление.
- Если предполагается, что на значениях переменных задано отношение строгого порядка, то в этом случае:
  - **Отрицательная** корреляция — корреляция, при которой увеличение одной переменной связано с уменьшением другой.
  - **Положительная** корреляция в таких условиях — это такая связь, при которой увеличение одной переменной связано с увеличением другой переменной.
- Возможна также ситуация отсутствия статистической взаимосвязи — например, для независимых случайных величин.

# Показатели корреляции

Метод вычисления коэффициента корреляции зависит от вида шкалы, к которой относятся переменные. Так, для измерения переменных с интервальной и количественной шкалами необходимо использовать коэффициент **корреляции Пирсона** (корреляция моментов произведений).

Если, по меньшей мере, одна из двух переменных имеет порядковую шкалу, либо не является нормально распределённой, необходимо использовать ранговую корреляцию Спирмена или  $\tau$  (тау) Кендалла.

# Ковариация

Важной характеристикой совместного распределения двух случайных величин является ковариация (или корреляционный момент).

- Ковариация является совместным центральным моментом второго порядка.
- Ковариация определяется как математическое ожидание произведения отклонений случайных величин:

$$\text{cov}_{XY} = M[(X - M(X))(Y - M(Y))] = M(XY) - M(X)M(Y)$$

где  $M$  – математическое ожидание.

## Свойства ковариации:

- Ковариация двух независимых случайных величин  $X$  и  $Y$  равна нулю
- Абсолютная величина ковариации двух случайных величин  $X$  и  $Y$  не превышает среднего геометрического их дисперсий:  $|\text{cov}_{XY}| \leq \sqrt{D_X D_Y}$ .
- Ковариация имеет размерность, равную произведению размерности случайных величин, то есть величина ковариации зависит от единиц измерения независимых величин.
  - Данная особенность ковариации затрудняет её использование в целях корреляционного анализа.

# Линейный коэффициент корреляции

Для устранения недостатка ковариации был введён линейный коэффициент корреляции (или коэффициент корреляции Пирсона).

- Коэффициент корреляции рассчитывается по формуле:

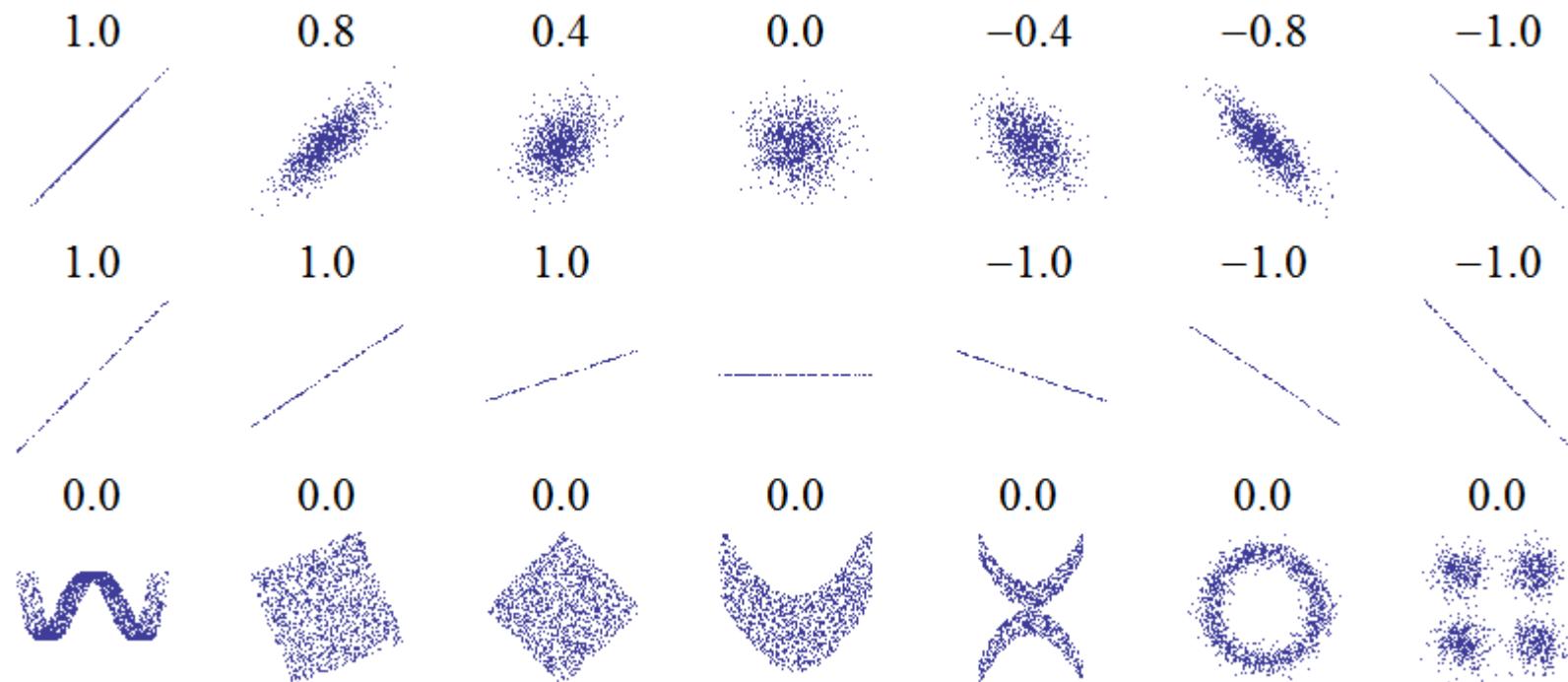
$$r_{XY} = \frac{cov_{XY}}{\sigma_X \sigma_Y} = \frac{\sum (X - \bar{X})(Y - \bar{Y})}{\sqrt{\sum (X - \bar{X})^2 \sum (Y - \bar{Y})^2}}$$

где  $\bar{X} = \frac{1}{n} \sum_{t=1}^n X_t$ ,  $\bar{Y} = \frac{1}{n} \sum_{t=1}^n Y_t$  — среднее значение выборок.

- Коэффициент корреляции изменяется в пределах от -1 до +1.

# Ограничения корреляционного анализа

- Применение возможно при наличии достаточного количества наблюдений для изучения.
- Коэффициент корреляции отражает «зашумлённость» линейной зависимости (верхняя строка)
- Коэффициент корреляции не описывает наклон линейной зависимости (средняя строка)
- Коэффициент корреляции совсем не подходит для описания сложных, нелинейных зависимостей (нижняя строка).
- Для распределения, показанного в центре рисунка, коэффициент корреляции не определен, так как дисперсия у равна нулю.



# Отбор признаков на основе корреляции

Отбор признаков на основе меры корреляции (англ. *Correlation Feature Selection*, CFS) оценивает подмножества признаков на базе следующей гипотезы:

«Хорошие поднаборы признаков содержат признаки, сильно коррелирующие с классификацией, но не коррелирующие друг с другом».

Следующее равенство даёт оценку поднабора признаков  $S$ , состоящего из  $k$  признаков:

$$Merit_{S_k} = \frac{k\bar{r}_{cf}}{\sqrt{k + k(k - 1)\bar{r}_{ff}}}$$

В формулу входят среднее значение всех корреляций признак-классификация  $\bar{r}_{cf}$ , и среднее значение всех корреляций признак-признак  $\bar{r}_{ff}$ .

# Критерий CFS (Correlation Feature Selection)

Критерий CFS определяется следующим образом:

$$CFS = \max_{S_k} \left[ \frac{r_{cf_1} + r_{cf_2} + \dots + r_{cf_k}}{\sqrt{k + 2(r_{f_1f_2} + \dots + r_{f_if_j} + \dots + r_{fkf_1})}} \right]$$

Переменные  $r_{cf_i}$  и  $r_{f_if_j}$  являются корреляциями, но не обязательно коэффициентами корреляции Пирсона.

Пусть  $x_i$  будет индикаторной функцией вхождения в множество для признака  $f_i$ . Тогда формула выше может быть переписана как задача оптимизации:

$$CFS = \max_{x \in \{0,1\}^n} \left[ \frac{(\sum_{i=1}^n a_i x_i)^2}{\sum_{i=1}^n x_i + \sum_{i \neq j} 2b_{ij} x_i x_j} \right]$$

Комбинаторные задачи выше являются, фактически, смешанными 0-1 задачами линейного программирования, которые могут быть решены с помощью алгоритма ветвей и границ.

**Спасибо за внимание**

# Методы машинного обучения

*Лекция 4*

## Регрессионный анализ

# Регрессионный анализ

Набор методов моделирования измеряемых данных и исследования их свойств, относится к разделам математической статистики и машинного обучения. Осуществляет исследование влияния одной или нескольких независимых переменных  $X_1, X_2, \dots, X_p$  на зависимую переменную  $Y$ .

Независимые переменные называют регрессорами, предикторами или объясняющими переменными, а зависимая переменная является результирующей, критериальной или регрессантом. Терминология зависимых и независимых переменных отражает лишь математическую зависимость переменных, а не причинно-следственные отношения.

Наиболее распространенный вид регрессионного анализа — линейная регрессия, когда находят линейную функцию, которая, согласно определённым математическим критериям, наилучшим образом соответствует данным.

Регрессионный анализ очень тесно связан с корреляционным анализом. В корреляционном анализе исследуется направление и теснота связи между количественными переменными. В регрессионном анализе исследуется форма зависимости между количественными переменными.

Регрессионный анализ используется для прогноза, анализа временных рядов, тестирования гипотез и выявления скрытых взаимосвязей в данных.

# Цели и задачи регрессионного анализа

Цель регрессионного анализа – с помощью уравнения регрессии предсказать ожидаемое среднее значение результирующей переменной, т.е. определение степени детерминированности вариации критериальной (зависимой) переменной от независимых переменных (предикторов).

Основные задачи регрессионного анализа следующие:

- определения вида и формы зависимости;
- оценка параметров уравнения регрессии;
- проверка значимости уравнения регрессии;
- проверка значимости отдельных коэффициентов уравнения (вклад отдельных независимых переменных в вариацию зависимой);
- построение интервальных оценок коэффициентов;
- исследование характеристик точности модели;
- построение точечных и интервальных прогнозов результирующей переменной (предсказание значения зависимой переменной с помощью независимых).

Как и корреляционный анализ, регрессионный анализ отражает только количественные зависимости между переменными. Причинно-следственные зависимости регрессионный анализ не отражает. Гипотезы о причинно-следственной связи переменных должны формулироваться и обосновываться исходя из теоретического анализа содержания изучаемого явления.

# Математическое определение регрессии

- Пусть  $Y, X_1, X_2, \dots, X_p$  — случайные величины с заданным совместным распределением вероятностей.
- Если для каждого набора значений  $X_1 = x_1, X_2 = x_2, \dots, X_p = x_p$  определено условное математическое ожидание (**уравнение регрессии в общем виде**):

$$f(x_1, x_2, \dots, x_p) = E(Y|X_1 = x_1, X_2 = x_2, \dots, X_p = x_p),$$

то функция  $f(x_1, x_2, \dots, x_p)$  называется регрессией величины  $Y$  по величинам  $X_1, X_2, \dots, X_p$ , а ее график — **линией регрессии**.

- Зависимость  $Y$  от  $X_1, X_2, \dots, X_p$  проявляется в изменении средних значений  $Y$  при изменении  $X_1, X_2, \dots, X_p$ .
- При каждом фиксированном наборе значений  $X_1 = x_1, X_2 = x_2, \dots, X_p = x_p$  величина  $Y$  остаётся случайной величиной с определённым распределением.
- Для выяснения вопроса, насколько точно регрессионный анализ оценивает изменение  $Y$  при изменении  $X_1, X_2, \dots, X_p$ , используется средняя величина дисперсии  $Y$  при разных наборах значений  $X_1, X_2, \dots, X_p$  (фактически речь идет о мере рассеяния зависимой переменной вокруг линии регрессии).

# Регрессионная модель

Регрессионный анализ осуществляет поиск функции регрессионной зависимости  $f(x) = E(y|x)$ . Регрессия может быть представлена в виде суммы неслучайной и случайной составляющих:

$$y = f(x) + \varepsilon$$

где  $f$  — функция регрессионной зависимости, а  $\varepsilon$  — аддитивная случайная величина с нулевым матожиданием. Обычно предполагается, что величина  $\varepsilon$  имеет гауссово распределение с нулевым средним и дисперсией  $\sigma_\varepsilon^2$ .

**Задача нахождения регрессионной модели нескольких свободных переменных:**

Задана выборка: множество  $\{x_1, x_2, \dots, x_N | x \in \mathbb{R}^M\}$  значений свободных переменных и множество  $\{y_1, y_2, \dots, y_N | y \in \mathbb{R}\}$  соответствующих им значений зависимой переменной. Совместно эти множества обозначаются как  $D$  - множество исходных данных  $\{(x, y)_i\}$ .

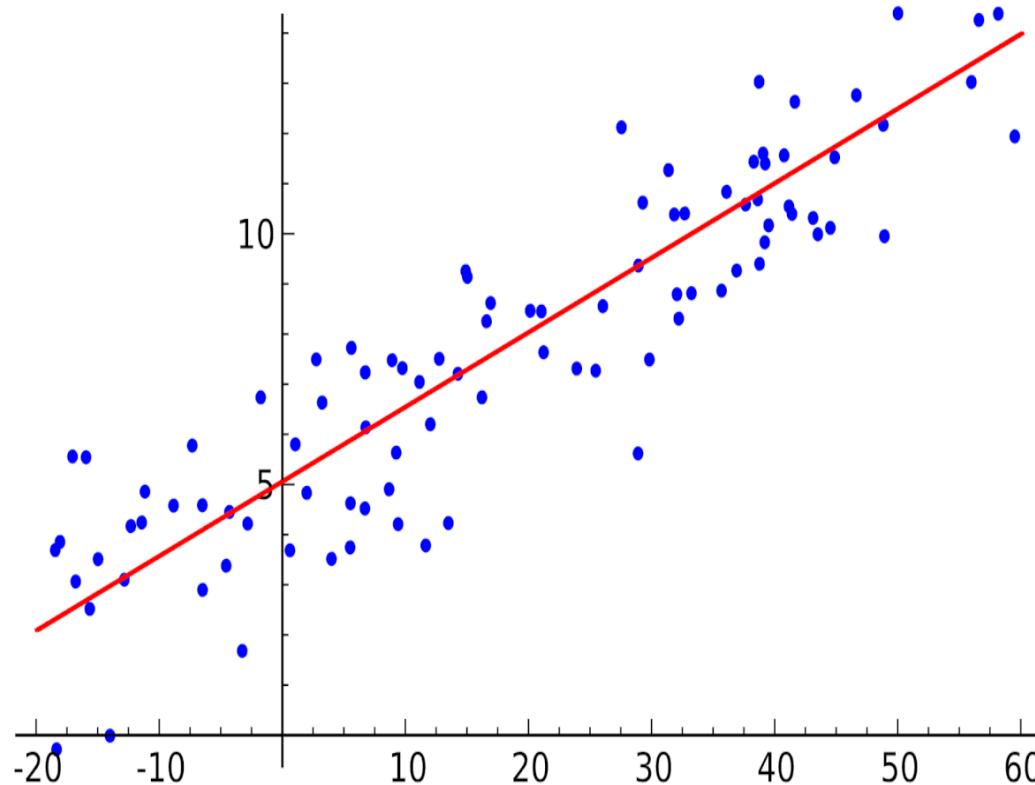
**Регрессионная модель** — параметрическое семейство функций  $f(w, x)$  зависящая от параметров  $w \in \mathbb{R}$  и свободных переменных  $x$ . Требуется найти наиболее вероятные параметры  $\bar{w}$ :

$$\bar{w} = \underset{w \in \mathbb{R}^W}{\operatorname{argmax}} p(y|x, w, f) = p(D|w, f)$$

Функция вероятности  $p$  зависит от гипотезы порождения данных и задается Байесовским выводом или методом наибольшего правдоподобия.

# Линейная регрессия

- Используемая в статистике регрессионная модель зависимости одной (объясняемой, зависимой) переменной  $y$  от другой или нескольких других переменных (факторов, регрессоров, независимых переменных)  $x$  с линейной функцией зависимости.
- Относится к задаче определения «линии наилучшего соответствия» через набор точек данных и стала простым предшественником нелинейных методов, которые используют для обучения нейронных сетей.



# Модель линейной регрессии

Линейная регрессия предполагает, что функция  $f$  зависит от параметров модели  $w$  линейно. При этом линейная зависимость от свободной переменной  $x$  необязательна:

$$y = f(w, x) + \varepsilon = \sum_{j=0}^N w_j \cdot g_j(x) + \varepsilon.$$

В случае, когда функция  $g(x) = x$  линейная регрессия имеет вид:

$$f(w, x) = w_0 + w_1 x_1 + w_2 x_2 + \dots + w_N x_N \text{ и } y = \sum_{j=0}^N w_j \cdot x_j + \varepsilon,$$

где  $x_j$  — компоненты вектора  $x$  (регрессоры или факторы модели),  $N$  — количество факторов модели,  $w_j$  - параметры модели (коэффициенты регрессии),  $\varepsilon$  – случайная ошибка модели.

Коэффициенты линейной регрессии показывают скорость изменения зависимой переменной по данному фактору, при фиксированных остальных факторах (в линейной модели эта скорость постоянна):  $\forall j \quad w_j = \frac{\partial f}{\partial x_j} = const$

Параметр  $w_0$ , при котором нет факторов, называют часто константой. Формально — это значение функции при нулевом значении всех факторов. Для аналитических целей удобно считать, что константа — это параметр при «факторе», равном 1, т.е.  $x_0 = 1$ .

# Парная и множественная регрессии

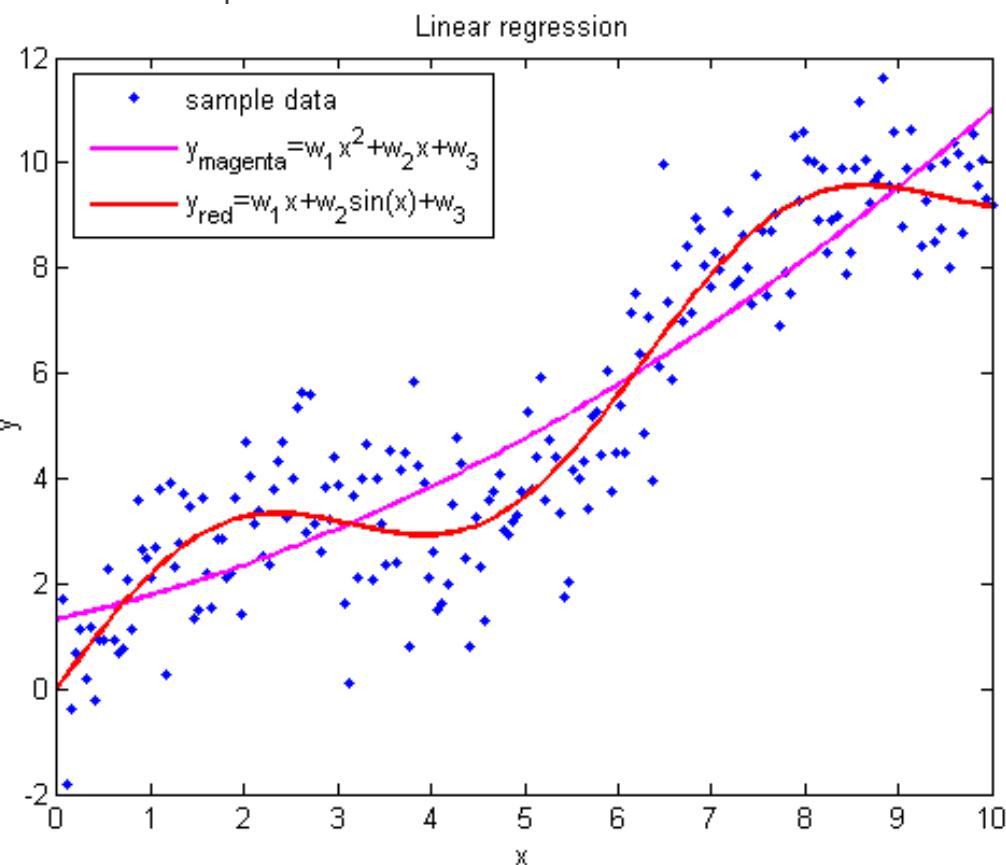
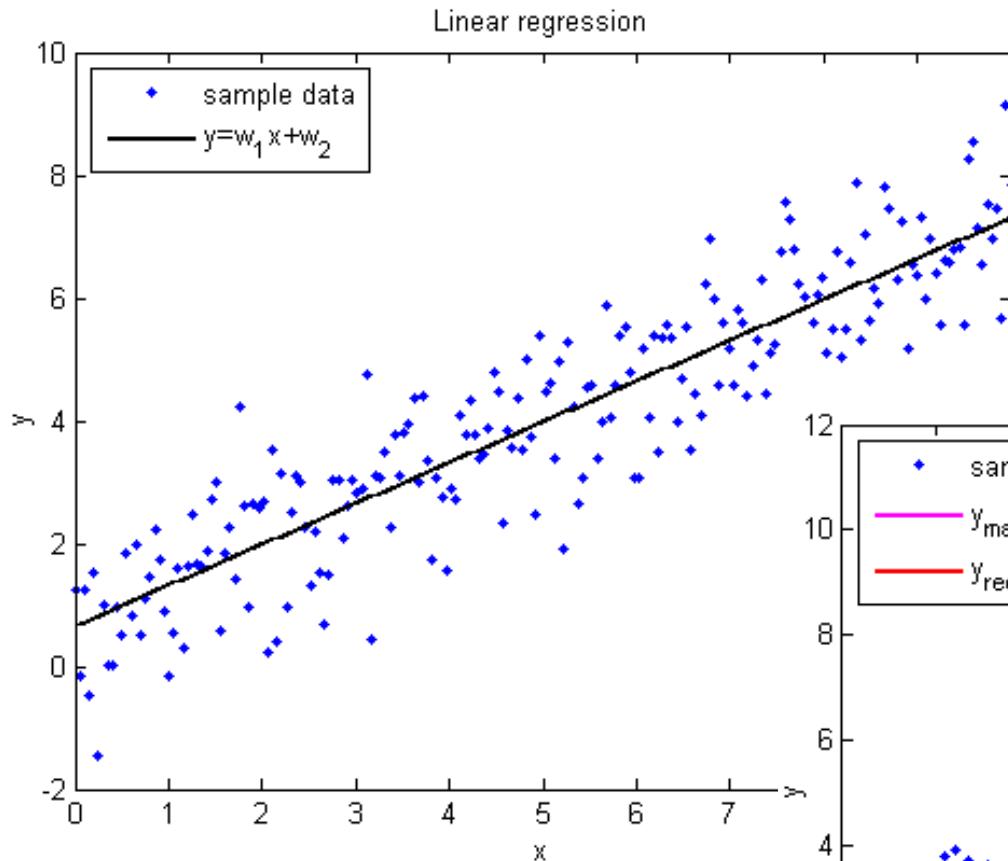
- В частном случае, когда фактор единственный (без учёта константы), говорят о парной или простейшей линейной регрессии:

$$y_t = a + b x_t + \varepsilon_t$$

- Когда количество факторов (без учёта константы) больше одного, то говорят о множественной регрессии:

$$y_t = w_0 + w_1 x_{t1} + w_2 x_{t2} + \cdots + w_N x_{tN} + \varepsilon_t$$

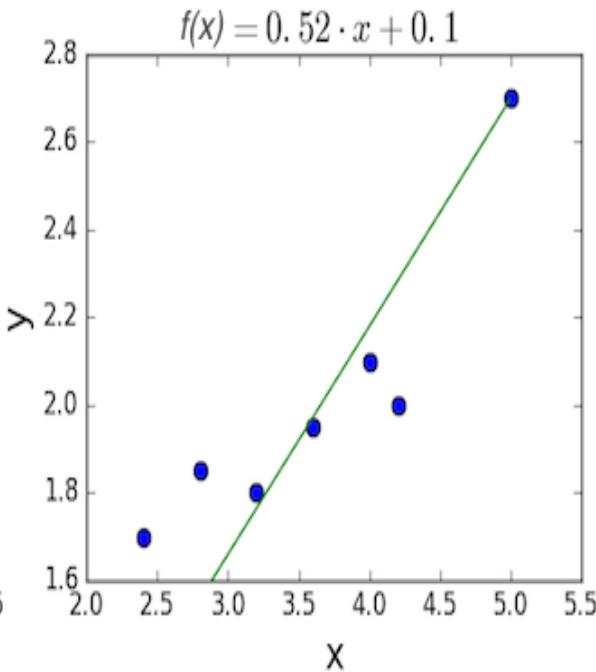
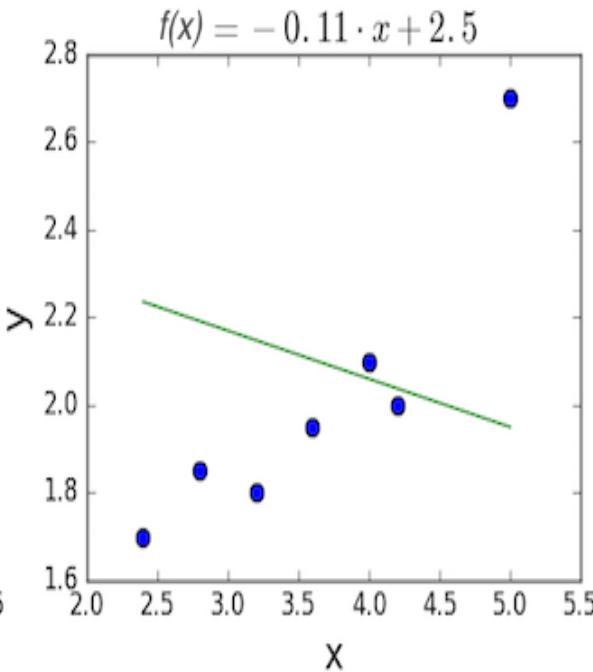
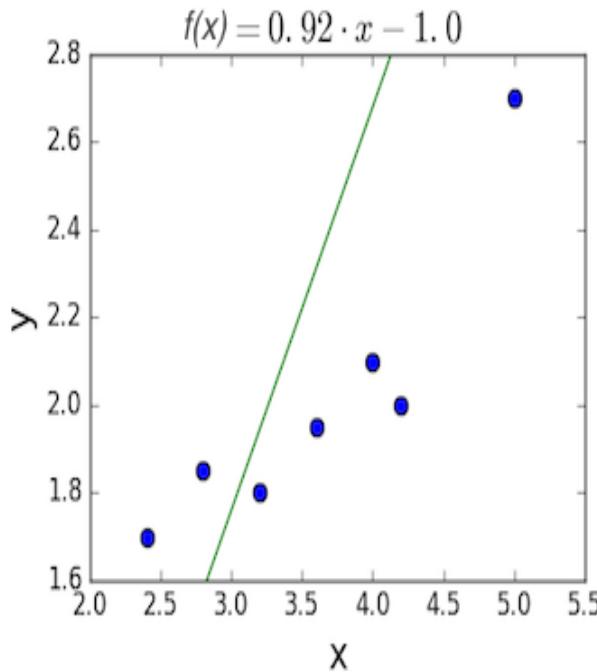
# Примеры моделей линейной регрессии



# Применение линейной регрессии

Предположим, нам задан набор из 7 точек. Цель — поиск линии, которая наилучшим образом соответствует этим точкам. Попробуем несколько случайных кандидатов. Довольно очевидно, что первые две линии не соответствуют нашим данным. Третья, похоже, лучше, чем две другие. Но как мы можем это проверить?

Формально нам нужно выразить, насколько хорошо подходит линия, и мы можем это сделать, определив функцию потерь.



# Сумма квадратов отклонений и среднеквадратичная ошибка

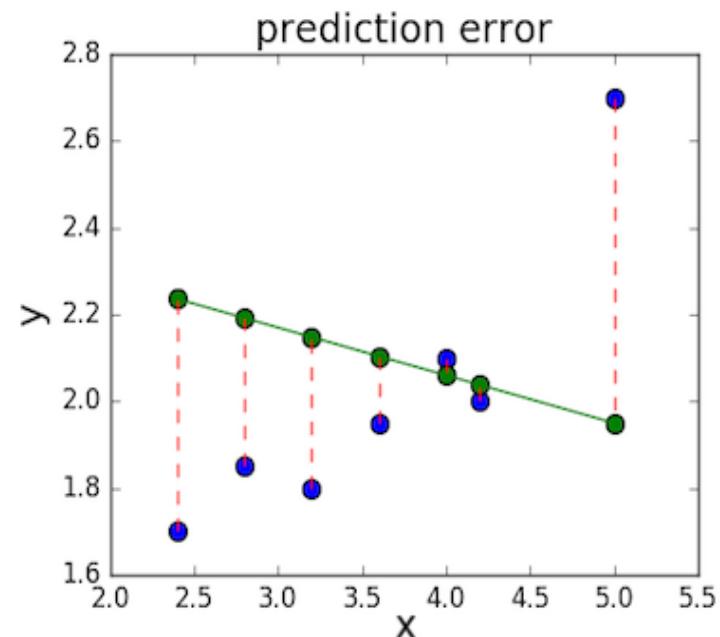
Разности  $y_i - f(x_i)$  между фактическими значениями зависимой переменной и восстановленными называются регрессионными остатками (residuals). В литературе используются также синонимы: *невязки* и *ошибки*. Одной из важных оценок критерия качества полученной зависимости является **сумма квадратов остатков** (Sum of Squared Errors – SSE и соответствует *Residual Sum of Squares - RSS*):

$$SSE = S(w) = \sum_{i=1}^M (y_i - f(w, x_i))^2 = \sum_{i=1}^M (y_i - \hat{y}_i)^2$$

Сумма квадратов отклонений реально наблюдаемых  $y_i$  от их оценок  $\hat{y}_i$ .

Дисперсия остатков или среднеквадратичная ошибка (Mean Square Error – MSE) вычисляется по формуле:

$$\bar{\sigma}_{\varepsilon}^2 = \frac{SSE}{N} = MSE$$



# Метод наименьших квадратов - МНК

Метод нахождения оптимальных параметров линейной регрессии, таких, что сумма квадратов ошибок (регрессионных остатков) минимальна.

Есть система линейных уравнений:  $Aw = y$ , где  $A$  прямоугольная матрица размера  $m \times n$ ,  $m > n$  (то есть число строк матрицы  $A$  больше количества искомых переменных). Такая система уравнений в общем случае не имеет решения. Поэтому «решение» заключается в выборе вектора  $w$ , который минимизирует «расстояние» между векторами  $Aw$  и  $y$ . Для этого можно применить критерий минимизации суммы квадратов разностей левой и правой частей уравнений системы:

$$(Aw - y)^T(Aw - y) \rightarrow \min_w .$$

Решение этой задачи минимизации приводит к решению следующей системы уравнений:  $A^T A w = A^T y$ , которое называется *нормальным уравнением*. Если столбцы матрицы  $A$  линейно независимы, то матрица  $A^T A$  обратима и единственное решение:

$$w = (A^T A)^{-1} A^T y$$

Отыскание решения  $w$  по методу наименьших квадратов эквивалентно задаче отыскания такой точки  $p = Aw$ , которая лежит ближе всего к  $y$  и находится при этом в пространстве столбцов матрицы  $A$ .

$$p = Aw = A(A^T A)^{-1} A^T y = Py$$

Матрица  $P = A(A^T A)^{-1} A^T$  называется матрицей проектирования вектора  $y$  на пространство столбцов матрицы  $A$ .

# Пример построения линейной регрессии

Заданы: выборка (таблица)  $D = \begin{pmatrix} x_1 & y_1 \\ x_2 & y_2 \\ \dots & \dots \\ x_M & y_M \end{pmatrix}$ , регрессионная модель (линейная) — квадратичный полином  $f = w_3x^2 + w_2x + w_1 = \sum_{j=1}^3 w_j x^{j-1}$

Для нахождения оптимального значения вектора параметров  $w = \langle w_1, \dots, w_3 \rangle^T$  выполняется следующая подстановка:  $x_i^0 \rightarrow a_{i1}, x_i^1 \rightarrow a_{i2}, x_i^2 \rightarrow a_{i3}$ .

Тогда матрица  $A$  значений подстановок свободной переменной  $x_i$  будет иметь вид:

$$A = \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ \dots & \dots & \dots \\ a_{M1} & a_{M2} & a_{M3} \end{pmatrix}$$

Задан критерий качества модели (функция ошибки):

$$S(w) = \sum_{i=1}^M (y_i - f(w, x_i))^2 = |Aw - y|^2 \rightarrow \min, \text{ где вектор } y = \langle y_1, \dots, y_M \rangle.$$

Требуется найти такие параметры  $w$ , которые бы доставляли минимум  $S(w)$ :  $w = \underset{w \in \mathbb{R}^3}{\operatorname{argmin}}(S)$

$$\begin{aligned} S(w) &= |Aw - y|^2 = (Aw - y)^T(Aw - y) = y^T y - y^T A w - w^T A^T y + w^T A^T A w \\ &= y^T y - 2y^T A w + w^T A^T A w \end{aligned}$$

Для того, чтобы найти минимум функции невязки, требуется приравнять ее производные к нулю. Производные данной функции по  $w$  составляют:

$$\frac{\partial S}{\partial w} = -2A^T y + 2A^T A w = 0$$

Это выражение совпадает с нормальным уравнением. Решение этой задачи должно удовлетворять системе линейных уравнений:  $A^T A w = A^T y$ , то есть,  $w = (A^T A)^{-1} A^T y$ .

# МНК в регрессионном анализе (аппроксимация данных)

Сущность МНК (обычного, классического) заключается в том, чтобы найти такие параметры  $w$ , при которых сумма квадратов отклонений будет минимальной:

$$S(w) = \sum_{k=1}^M (y_k - f(w, x_k))^2 = \sum_{k=1}^M (y_k - \hat{y}_k)^2 = \text{SSE} \rightarrow \min, w = \underset{w \in \mathbb{R}^3}{\operatorname{argmin}}(S)$$

В общем случае решение этой задачи может осуществляться численными методами оптимизации (минимизации). В этом случае говорят о *нелинейном МНК* (NLS или NLLS — *Non-Linear Least Squares*).

Для аналитического решения задачи минимизации необходимо найти стационарные точки функции  $S(w)$ , продифференцировав её по неизвестным параметрам  $w$ , приравняв производные к нулю и решив полученную систему уравнений.

Для этого определим функцию невязки:  $\sigma(\vec{w}) = \frac{1}{2} \sum_{k=1}^M (y_k - \hat{y}_k)^2$ , где  $y_k$  - наблюдаемое значение,  $\hat{y}_k = w_0 + w_1 x_{k1} + w_2 x_{k2} + \dots + w_N x_{kN}$  - оценка  $y_k$  согласно модели,  $M$ — объём выборки,  $N$ — количество регрессоров (факторов).

Условие минимума функции невязки:

$$\left\{ \begin{array}{l} \frac{\partial \sigma(\vec{w})}{\partial w_i} = 0 \\ i = 0 \dots N \end{array} \right. \Leftrightarrow \left\{ \begin{array}{l} \sum_{i=1}^M y_i = \sum_{i=1}^M \sum_{j=1}^N w_j x_{ij} + w_0 M \\ \sum_{i=1}^M y_i x_{ik} = \sum_{i=1}^M \sum_{j=1}^N w_j x_{ij} x_{ik} + w_0 \sum_{i=1}^M x_{ik} \\ k = 1, \dots, N \end{array} \right.$$

Полученная система является системой  $N + 1$  линейных уравнений с  $N + 1$  неизвестными  $w_0, \dots, w_N$ .

# МНК - нахождение коэффициентов линейной регрессии

Если представить свободные члены левой части уравнений (предыдущий слайд) матрицей  $B$ , а коэффициенты при неизвестных в правой части — матрицей  $A$ , то получаем матричное уравнение:  $B = A \times W$ , для решения которого применимы методы решения СЛАУ, например метод Гаусса.

$$B = \begin{pmatrix} \sum_{i=1}^M y_i \\ \sum_{i=1}^M y_i x_{i,1} \\ \vdots \\ \sum_{i=1}^M y_i x_{i,N} \end{pmatrix}, \quad A = \begin{pmatrix} M & \sum_{i=1}^M x_{i,1} & \sum_{i=1}^M x_{i,2} & \dots & \sum_{i=1}^M x_{i,N} \\ \sum_{i=1}^M x_{i,1} & \sum_{i=1}^M x_{i,1} x_{i,1} & \sum_{i=1}^M x_{i,2} x_{i,1} & \dots & \sum_{i=1}^M x_{i,N} x_{i,1} \\ \sum_{i=1}^M x_{i,2} & \sum_{i=1}^M x_{i,1} x_{i,2} & \sum_{i=1}^M x_{i,2} x_{i,2} & \dots & \sum_{i=1}^M x_{i,N} x_{i,2} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \sum_{i=1}^M x_{i,N} & \sum_{i=1}^M x_{i,1} x_{i,N} & \sum_{i=1}^M x_{i,2} x_{i,N} & \dots & \sum_{i=1}^M x_{i,N} x_{i,N} \end{pmatrix}, \quad W = \begin{pmatrix} w_0 \\ w_1 \\ \vdots \\ w_N \end{pmatrix}$$

# МНК - случай полиномиальной модели

Если данные аппроксимируются полиномиальной функцией регрессии одной переменной  $f(x) = w_0 + \sum_{i=1}^k w_i x^i$ , то, воспринимая степени  $x^i$  как независимые факторы для каждого  $i$  можно оценить параметры модели исходя из общей формулы оценки параметров линейной модели.

Для этого в общей формуле достаточно учесть, что при такой интерпретации  $x_{ti}x_{tj} = x_t^i x_t^j = x_t^{i+j}$  и  $x_{tj}y_t = x_t^j y_t$ . Следовательно, матричные уравнения в данном случае примут вид:

$$\begin{pmatrix} n & \sum_n x_t & \dots & \sum_n x_t^k \\ \sum_n x_t & \sum_n x_t^2 & \dots & \sum_n x_t^{k+1} \\ \vdots & \vdots & \ddots & \vdots \\ \sum_n x_t^k & \sum_n x_t^{k+1} & \dots & \sum_n x_t^{2k} \end{pmatrix} \begin{bmatrix} w_0 \\ w_1 \\ \vdots \\ w_N \end{bmatrix} = \begin{bmatrix} \sum_n y_t \\ \sum_n x_t y_t \\ \vdots \\ \sum_n x_t^k y_t \end{bmatrix}$$

# Интерпретация параметров регрессии

- Параметры  $w_i$  являются частными коэффициентами корреляции  $x_i$  и  $y$ ;
- $(w_i)^2$  - измеряет индивидуальный вклад  $x_i$  в объяснение  $y$ , при закреплении влияния остальных предикторов.
- В случае коррелирующих предикторов возникает проблема неопределённости в оценках, которые становятся зависимыми от порядка включения предикторов в модель.
- В нелинейных моделях регрессионного анализа, важно обращать внимание на тип нелинейности:
  - по независимым переменным (с формальной точки зрения легко сводящейся к линейной регрессии),
  - по оцениваемым параметрам (вызывающей серьёзные вычислительные трудности).

# Нелинейная регрессия

Частный случай регрессионного анализа, в котором рассматриваемая регрессионная модель является функцией, зависящей от параметров и свободных переменных. Главное, что зависимость от параметров предполагается нелинейной.

Задана выборка из  $M$  пар  $(x_i, y_i)$  и регрессионная модель  $f(w, x)$ , которая зависит от параметров  $w = (w_1, \dots, w_W)$  и свободной переменной  $x$ . Требуется найти такие значения параметров, которые доставляли бы минимум сумме квадратов регрессионных остатков SSE (RSS):

$$S(w) = \sum_{i=1}^M (y_i - f(w, x_i))^2 = \sum_{i=1}^M r_i^2,$$

где  $r_i = y_i - f(w, x_i)$  для  $i = 1, \dots, M$ .

Для нахождения минимума функции  $S$ , приравняем к нулю её первые частные производные параметрам  $w$ :

$$\frac{\partial S}{\partial w_j} = 2 \sum_i r_i \frac{\partial r_i}{\partial w_j} = 0, \text{ где } j = 1, \dots, n.$$

Так как функция  $S$  в общем случае может не иметь единственного минимума, то сначала назначается начальное значение вектора параметров  $w_0$  и далее приближаться к оптимальному вектору по шагам:

$$w_j \approx w_j^{k+1} = w_j^k + \Delta w_j$$

где  $k$ - номер итерации,  $\Delta w_j$ - вектор шага.

Для нахождения оптимальных параметров нелинейных регрессионных моделей используются метод сопряжённых градиентов, метод Ньютона-Гаусса или алгоритм Левенберга-Марквардта.

# Оценки качества регрессии (MSE, RMSE, MAE)

## Средняя квадратичная ошибка (Mean Squared Error, MSE)

MSE применяется в ситуациях, когда нам надо подчеркнуть большие ошибки и выбрать модель, которая дает меньше больших ошибок прогноза (ошибки прогноза возводятся в квадрат).

$$MSE = \frac{1}{n} \sum_{i=1}^n (f(x_i) - y_i)^2$$

## Корень из средней квадратичной ошибки (Root Mean Squared Error, RMSE)

RMSE получается из MSE путем извлечения корня.

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

## Средняя абсолютная ошибка (англ. Mean Absolute Error, MAE)

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

Среднеабсолютный функционал слабее штрафует за большие отклонения по сравнению со среднеквадратичным, и поэтому менее чувствителен к выбросам. При использовании любого из этих функционалов может быть полезно проанализировать, какие объекты вносят наибольший вклад в общую ошибку.

# Коэффициент детерминации $R$ -квадрат

Коэффициент детерминации измеряет долю дисперсии, объясненную моделью, в общей дисперсии целевой переменной. Фактически, данная мера качества — это нормированная среднеквадратичная ошибка.

$$R^2 = 1 - \frac{\sum_{i=1}^n (f(x_i) - y_i)^2}{\sum_{i=1}^n (y_i - \bar{y})^2}, \quad \bar{y} = \frac{1}{n} \sum_{i=1}^n y_i$$

Коэффициент детерминации принимает значения от 0 до 1. Чем ближе значение коэффициента к 1, тем сильнее зависимость. При оценке регрессионных моделей это интерпретируется как соответствие модели данным.

Для приемлемых моделей  $R^2 > 50\%$ . Модели с  $R^2 > 80\%$  можно признать достаточно хорошими. Значение  $R^2 = 1$  означает функциональную зависимость между переменными.

# Оценки качества регрессии (MAPE, SMAPE, MASE)

Средняя абсолютная процентная ошибка (Mean Absolute Percentage Error, MAPE)

$$MAPE = 100\% \times \frac{1}{n} \sum_{i=1}^n \frac{|y_i - f(x_i)|}{|y_i|}$$

Это коэффициент, не имеющий размерности, с простой интерпретацией. Его можно измерять в долях или процентах. Если MAPE=11.4%, то это говорит о том, что ошибка составила 11,4% от фактических значений.

Симметричная MAPE (Symmetric MAPE, SMAPE)

$$SMAPE = \frac{1}{n} \sum_{i=1}^n \frac{2|y_i - f(x_i)|}{|y_i| + |f(x_i)|}$$

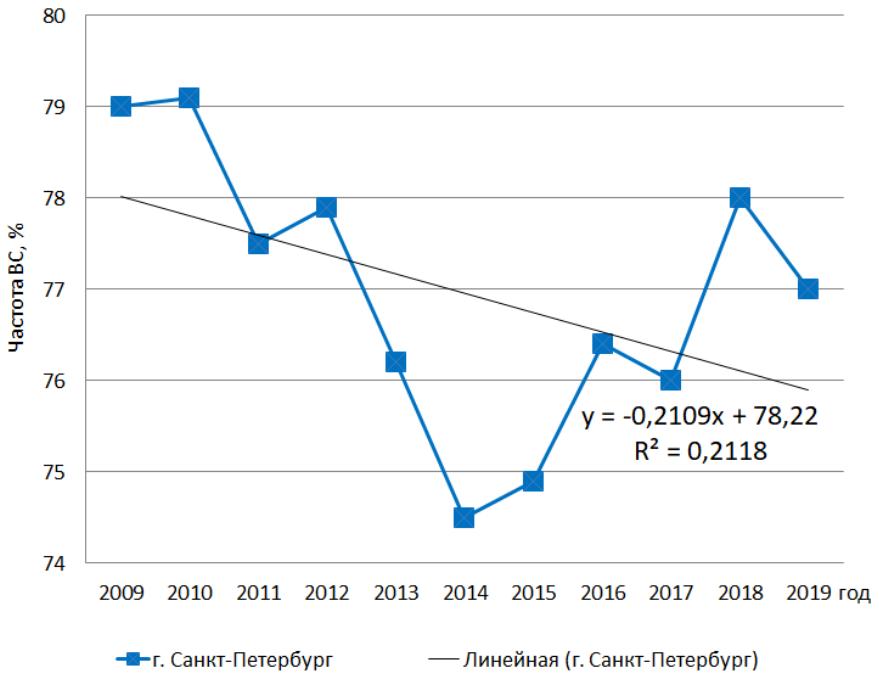
Средняя абсолютная масштабированная ошибка (Mean absolute scaled error, MASE)

$$MASE = \frac{T-1}{h} \frac{\sum_{j=1}^h |e_{T+j}|}{\sum_{t=2}^T |Y_t - Y_{t-1}|}$$

MASE имеет дело с двумя суммами: числитель соответствует тестовой выборке, знаменатель - обучающей. Ошибка не зависит от масштабов данных и является симметричной: то есть положительные и отрицательные отклонения от факта рассматриваются в равной степени.

Проблема MASE в том, что её тяжело интерпретировать. Например, MASE=1.21 ни о чём, по сути, не говорит. Это просто означает, что ошибка прогноза оказалась в 1.21 раза выше среднего абсолютного отклонения ряда в первых разностях, и ничего более.

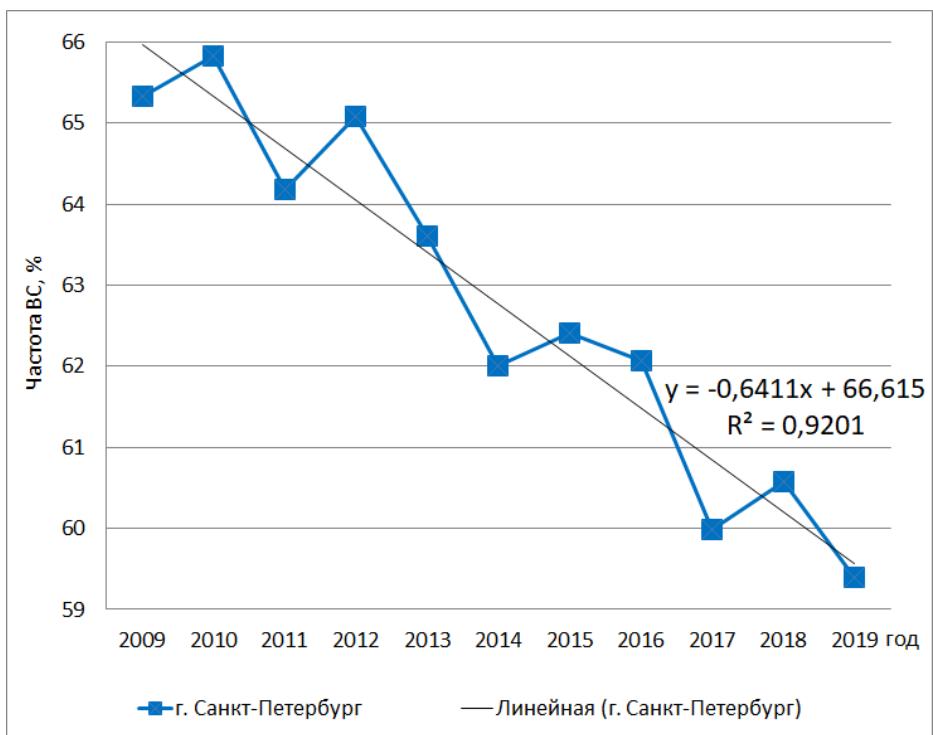
# Пример парной линейной регрессии



$$\begin{cases} \hat{b} = \frac{n \sum_{t=1}^n x_t y_t - (\sum_{t=1}^n x_t)(\sum_{t=1}^n y_t)}{n \sum_{t=1}^n x_t^2 - (\sum_{t=1}^n x_t)^2} \\ \hat{a} = \frac{\sum_{t=1}^n y_t - \hat{b} \sum_{t=1}^n x_t}{n} \end{cases}$$

$$y_t = a + b x_t + \varepsilon_t$$

$$\left( \begin{array}{cc} n & \sum_{t=1}^n x_t \\ \sum_{t=1}^n x_t & \sum_{t=1}^n x_t^2 \end{array} \right) \begin{pmatrix} a \\ b \end{pmatrix} = \left( \begin{array}{c} \sum_{t=1}^n y_t \\ \sum_{t=1}^n x_t y_t \end{array} \right)$$



**Спасибо за внимание**

# Методы машинного обучения

*Лекция 5*

Классификация -  
Байесовский подход

# Байесовский подход

## Основные определения

- Совместная вероятность – вероятность одновременного наступления двух событий:  $P(A, B)$ ;
- Независимость: А и В независимы, если:  $P(A, B) = P(A)P(B)$
- Условная вероятность – вероятность наступления одного события, если известно, что произошло другое,  $P(A|B)$  - вероятность наступления события А при условии, что событие В произошло. Очевидный частный случай:  $P(A|A)=1=100\%$
- Вероятность совместного появления двух зависимых событий равна:

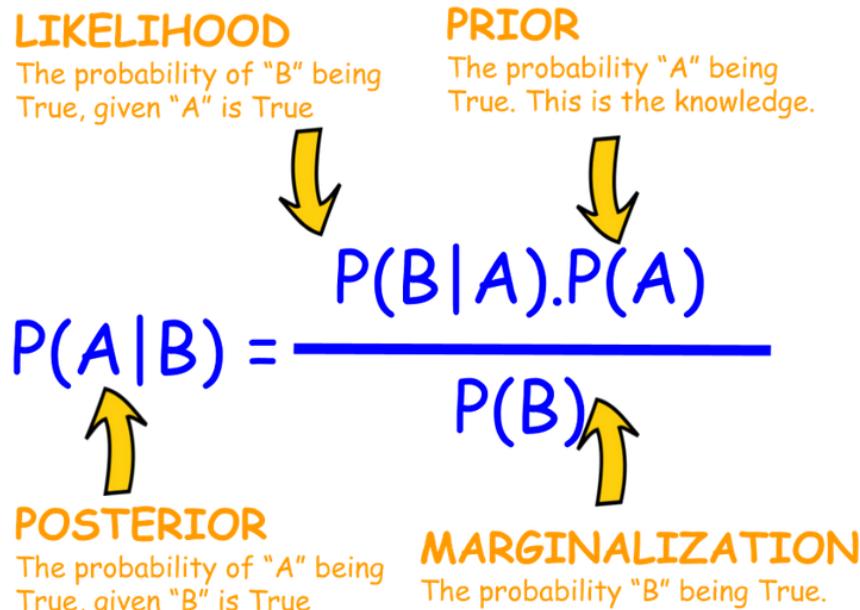
$$P(A, B) = P(A|B)P(B) = P(B|A)P(A)$$

- Теорема Байеса – из предыдущей формулы:

$$P(A|B) = \frac{P(A, B)}{P(B)} = \frac{P(B|A)P(A)}{P(B)}$$

# Теорема Байеса

**Теорема Байеса (или формула Байеса)** — одна из основных теорем элементарной теории вероятностей, которая позволяет определить вероятность какого-либо события при условии, что произошло другое статистически взаимозависимое с ним событие. Другими словами, по формуле Байеса можно более точно пересчитать вероятность, взяв в расчёт как ранее известную информацию, так и данные новых наблюдений.



- $P(A)$  — априорная вероятность гипотезы А (prior probability);
- $P(A|B)$  — вероятность гипотезы А при наступлении события В (апостериорная вероятность - posterior probability);
- $P(B|A)$  — вероятность наступления события В при истинности гипотезы А (правдоподобие - likelihood);
- $P(B)$  — полная вероятность наступления события В (вероятность данных evidence).

# Вычисление $P(B)$ - Маргинализация

В задачах и статистических приложениях  $P(B)$  обычно вычисляется по формуле полной вероятности события, зависящего от нескольких несовместных гипотез, имеющих суммарную вероятность 1.

$$P(B) = \sum_A P(A, B) = \sum_{i=1}^N P(B|A_i)P(A_i)$$

где вероятности под знаком суммы известны или допускают экспериментальную оценку.

Тогда **формула Байеса** примет вид:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)} = \frac{P(B|A)P(A)}{\sum_{i=1}^N P(B|A_i)P(A_i)}$$

# Пример: О болезнях и вероятностях

Пусть некий тест на какую-нибудь болезнь имеет вероятность успеха 95% (т.е. 5% - вероятность как позитивной, так и негативной ошибки). Всего болезнь встречается у 1% респондентов (не учитываем, что они разного возраста и профессий).

Пусть некий человек на диспансеризации неожиданно для него получил позитивный результат теста (тест говорит, что у него есть заболевание). С какой вероятностью он действительно болен?

## Вывод с использованием байесовского подхода

Обозначим через  $t$  результат теста, через  $d$ -наличие болезни.

Тогда:

$$p(t = 1) = p(t = 1|d = 1)p(d = 1) + p(t = 1|d = 0)p(d = 0)$$

Используем теорему Байеса:

$$\begin{aligned} p(d = 1|t = 1) &= \frac{p(t = 1|d = 1)p(d = 1)}{p(t = 1|d = 1)p(d = 1) + p(t = 1|d = 0)p(d = 0)} \\ &= \frac{0,95 \times 0,01}{0,95 \times 0,01 + 0,05 \times 0,99} = 0,16 \end{aligned}$$

## Выводы:

Такие задачи составляют суть вероятностного вывода (probabilistic inference). Поскольку они обычно основаны на теореме Байеса, вывод часто называют байесовским (Bayesian inference).

# Интерпретация вероятности

## Вероятность как частота

- Обычно в классической теории вероятностей, происходящей из физики, вероятность понимается как предел отношения количества определенного результата эксперимента к общему количеству экспериментов.
- Стандартный пример: бросание монетки.

## Вероятность, как степень доверия

Мы можем рассуждать о том, «насколько вероятно» то, что:

- Динозавры вымерли в результате падения метеорита;
- Произойдет столкновение с определенным космическим объектом;
- В соседних звездных системах есть жизнь;
- Сборная России победит на ближайшем чемпионате мира по футболу.

Говорить о «стремящемся к бесконечности количестве экспериментов» совершенно бессмысленно, т.к. эксперимент здесь ровно один. Вероятности выступают как степень доверия (degrees of belief). Это байесовский подход к вероятностям (Томас Байес так их понимал).

# Прямые и обратные задачи

## Прямая задача:

- В корзине лежат 10 шаров, из них 3 черных. Какова вероятность выбрать черный шар?
- В корзине лежат 10 шаров с номерами от 1 до 10. Какова вероятность того, что номера трех последовательно выбранных шаров дадут в сумме 12?

## Обратная задача:

- Перед нами две корзины, в каждой по 10 шаров, но в одной 3 черных, а в другой 6. Некто взял из какой-то корзины шар, и он оказался черным. Насколько вероятно, что он брал шар из первой урны?

В обратной задаче вероятности сразу стали байесовскими, т.е. необходимо определить вероятность какого-либо события при условии, что произошло другое статистически взаимозависимое с ним событие.

Прямые задачи теории вероятностей описывают некий вероятностный процесс или модель и просят подсчитать ту или иную вероятность (т.е. фактически по модели предсказать поведение). Обратные задачи содержат скрытые переменные (в примере – это номер корзины, из которой брали шар) и часто просят по известному поведению построить вероятностную модель.

Теорема Байеса позволяет переставить местами причину и следствие. Зная с какой вероятностью та или иная причина приводит к некоему событию, теорема Байеса позволяет рассчитать вероятность того что именно эта причина привела к наблюдаемому событию.

# ML(maximum likelihood) vs. MAP (maximum a posteriori)

В статистике обычно ищут гипотезу максимального правдоподобия (maximum likelihood):

$$\theta_{ML} = \operatorname{argmax}_{\theta} p(D|\theta)$$

В байесовском подходе ищут апостериорное распределение (posterior)

$$p(\theta|D) \propto p(D|\theta)p(\theta)$$

и, возможно, максимальную апостериорную гипотезу (maximum a posteriori):

$$\theta_{MAP} = \operatorname{argmax}_{\theta} p(\theta|D) = \operatorname{argmax}_{\theta} p(D|\theta)p(\theta)$$

Функция правдоподобия имеет вид:

$$a \mapsto p(y|x=a)$$

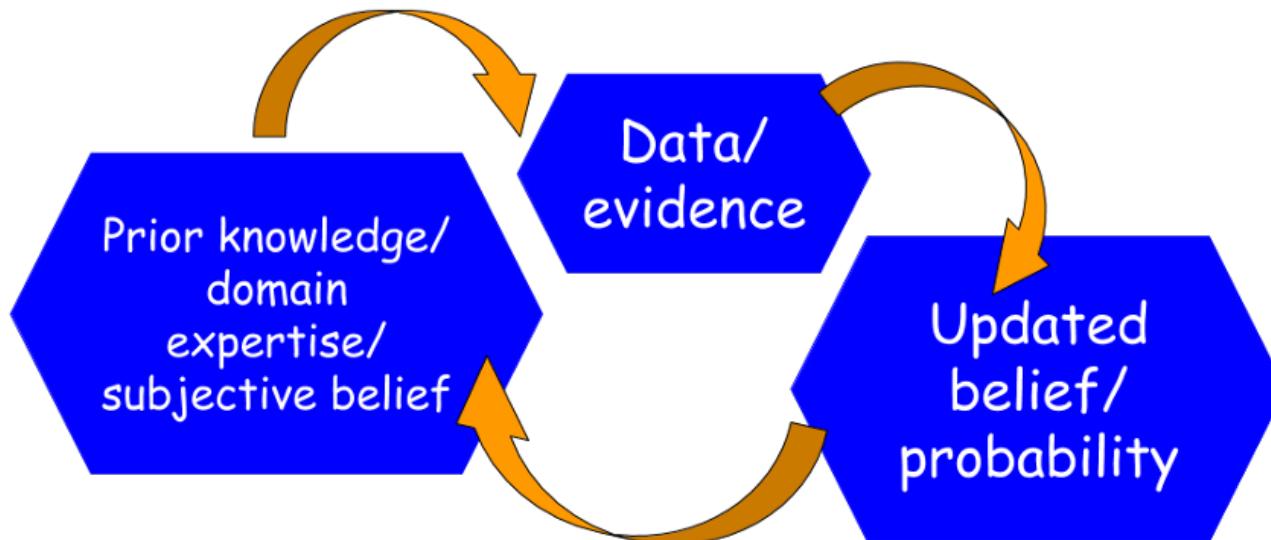
для некоторой случайной величины  $y$ .

# Логический процесс анализа данных

Исторически в большинстве методов статистического обучения (статистических исследований) понятие априорного события не используется или недооценивается.

Теорема Байеса позволяет учитывать субъективную оценку или уровень доверия в строгих статистических расчетах. Это один из методов, который **позволяет постепенно обновлять вероятность события по мере поступления новых наблюдений или сведений**.

- Мы начинаем с гипотезы и уровня доверия к этой гипотезе. Это означает, что на основе знания предметной области или предшествующих других знаний мы приписываем этой гипотезе ненулевую вероятность.
- Затем мы собираем данные и обновляем наши первоначальные убеждения. Если новые данные подтверждают гипотезу, то вероятность возрастает, если не подтверждают - вероятность снижается.



# Пример: Скрининг-тест на употребление наркотиков

Предположим, что тест на применение наркотика имеет **97% чувствительность** (доля истинно положительных результатов) и **95% специфичность** (доля истинно отрицательных результатов). То есть тест даст **97% истинно положительных** результатов для потребителей наркотиков и **95% истинно отрицательных результатов** для лиц, не употребляющих наркотики. Предположим, мы также знаем, что 0,5% населения в целом употребляют наркотики.

*Какова вероятность того, что случайно выбранный человек с положительным результатом анализа является потребителем наркотиков?*

Знание о процентах употребляющих является важнейшей частью **«априорной вероятности»**, которая представляет собой часть обобщенных знаний об общем уровне распространенности. Это наше предварительное суждение о вероятности того, что случайный испытуемый будет употреблять наркотики. Это означает, что **если мы выберем случайного человека из общей популяции без какого-либо тестирования, мы можем только сказать, что вероятность того, что этот человек употребляет наркотики, составляет 0,5%**.

# Расчет по правилу Байеса

Формула для вычисления по правилу Байеса:

- принимает в качестве входных данных чувствительность и специфичность теста, а также предварительные знания о проценте потребителей наркотиков в популяции;
- выдает вероятность того, что тестируемый является потребителем наркотиков, на основе положительного результата теста.

$$P(U^+|T^+) = \frac{P(T^+|U^+)P(U^+)}{P(T^+|U^+)P(U^+) + P(T^+|U^-)P(U^-)}$$

$P(U^+)$  = Уровень распространенности наркомании

$P(U^-)$  = 1 - Уровень распространенности наркомании

$P(T^+|U^+)$  = Чувствительность теста

$P(T^-|U^-)$  = Специфичность теста

$P(T^+|U^-)$  = 1 - Специфичность теста

$$P(U^+|T^+) = \frac{0,97 \cdot 0,005}{0,97 \cdot 0,005 + 0,05 \cdot 0,995} = 0,089$$

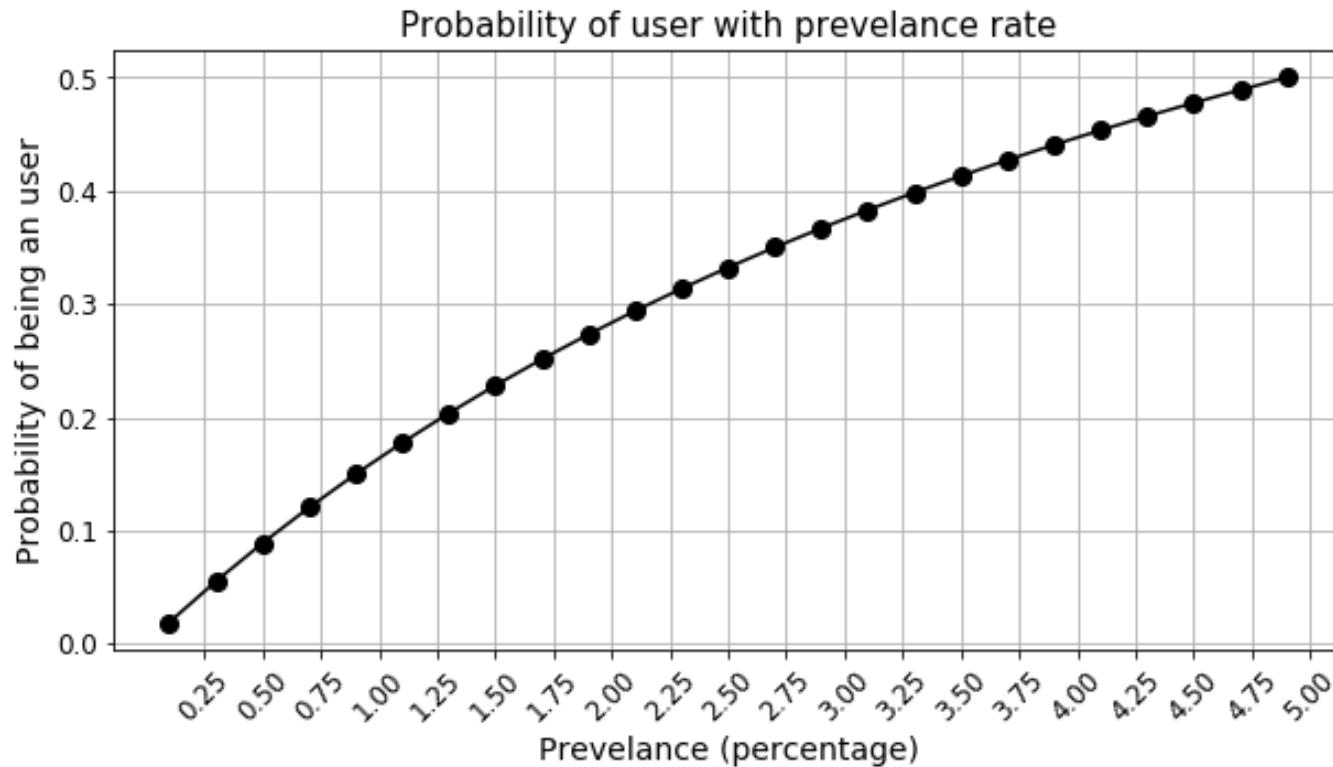
**Что здесь интересного?**

Даже при использовании теста, который в 97% случаях верно выявляет положительные случаи и который в 95% случаях правильно выявляет отрицательные случаи, истинная вероятность выявить человека употребляющего наркотики с положительным результатом теста составляет всего 8,9%!

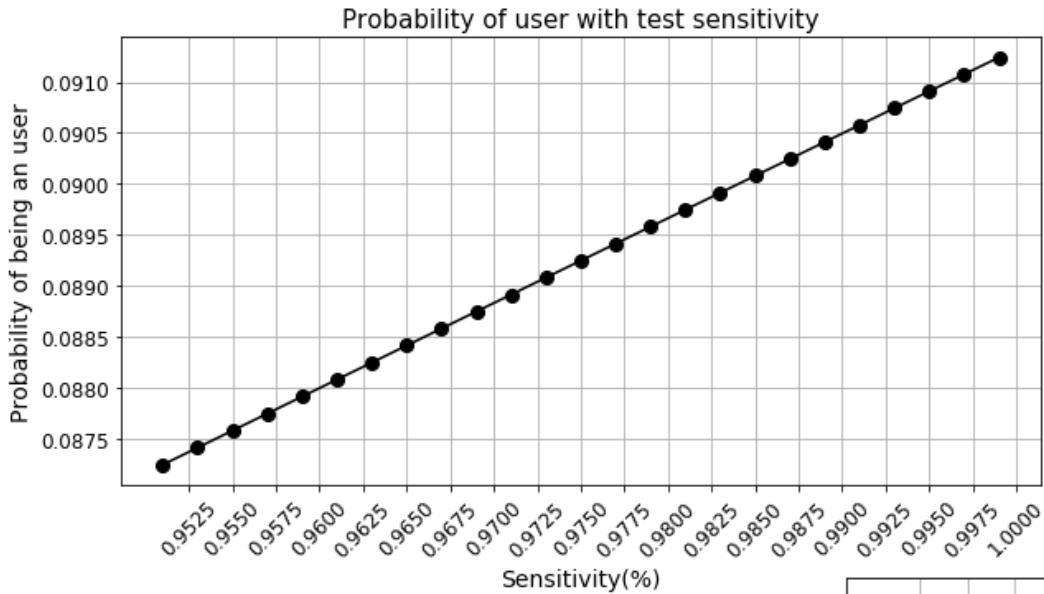
# Всего 8,9%! Что не так?

Это связано с чрезвычайно низким уровнем распространенности. **Количество ложных срабатываний превышает количество истинных срабатываний.**

Если протестировано 1000 человек, ожидается, что будет 995 не наркоманов и 5 наркоманов. Из 995 не наркоманов ожидается  $0,05 \times 995 \approx 50$  ложных срабатываний. Из 5 наркоманов ожидается  $0,97 \times 5 \approx 5$  истинно положительных результатов. Из 55 положительных результатов только 5 являются истинно положительными!

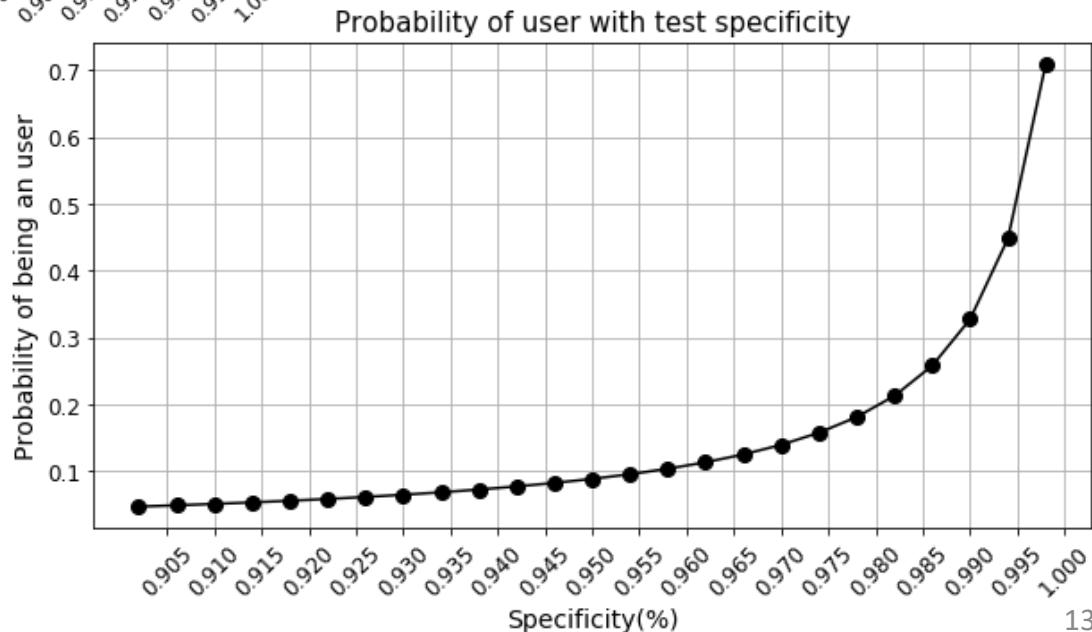


# Какой уровень точности теста необходим для улучшения сценария?



Имеет место нелинейная зависимость вероятности от специфичности теста, и по мере увеличения специфичности, происходит значительное увеличение вероятности.

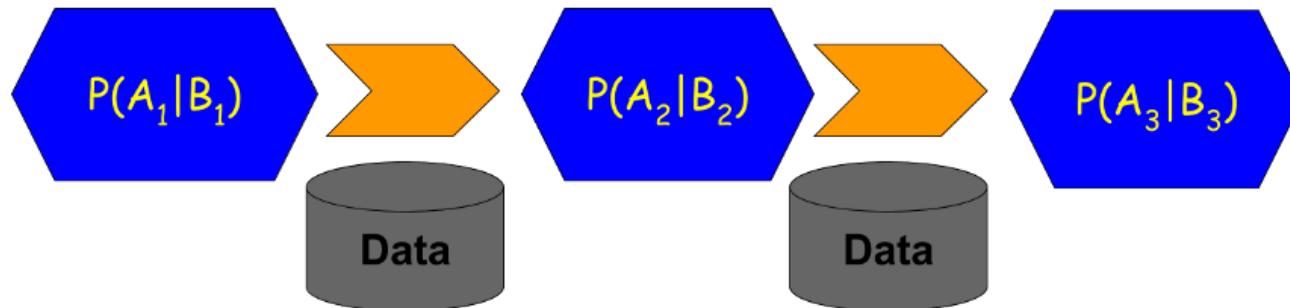
Даже с чувствительностью, близкой к 100%, улучшения практически не происходит.



# Цепочка расчетов и формула Байеса

Лучшее в байесовском выводе - это **возможность использовать предшествующие знания** в форме **априорного** вероятностного члена в числителе теоремы Байеса.

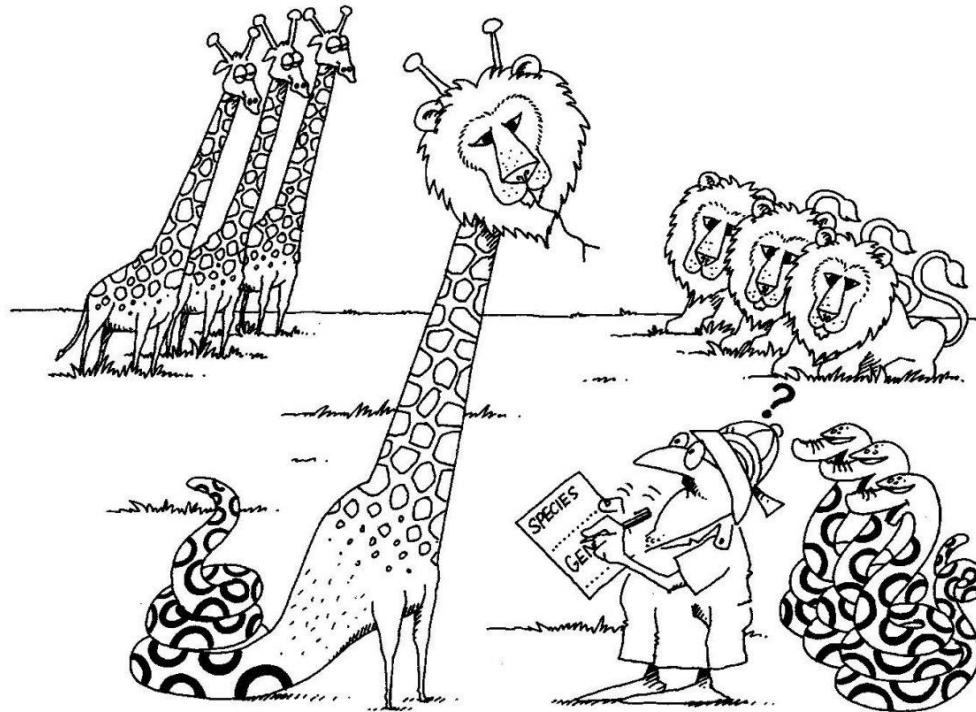
- Предварительные знания - это не что иное, как вычисленная вероятность теста, которая затем возвращается к следующему тесту.
- Для случаев, когда уровень распространенности среди населения в целом чрезвычайно низок, один из способов повысить уверенность в результате теста - назначить последующий тест, если первый результат теста окажется положительным.
- Апостериорная вероятность первого теста становится *априорной вероятностью* для второго теста, т.е.  $P(U^+)$  для второго теста уже не общий показатель распространенности, а вероятность из первого теста.
- Расчетная (апостериорная) вероятность первого теста 8,9%, во втором teste она возрастает до 65,4%, а третий положительный тест дает значение 97,3%.
- Следовательно, неточный тест можно использовать несколько раз, чтобы обновить мнение с помощью последовательного применения правила Байеса.



*Chaining of Bayes' rule for updating probabilities as the data comes in*

# Классификация

**Классификация** (classification) — это задача присвоения меток класса (class label) наблюдениям (Observation) объектам из предметной области. Множество допустимых меток класса конечно. В свою очередь **класс** — это множество всех объектов с данным значением метки. Требуется построить алгоритм, способный классифицировать (присвоить метку) произвольный объект из исходного множества. Классификация, как правило, на этапе настройки использует обучение с учителем.



# Байесовский классификатор

Широкий класс алгоритмов классификации, основанный на принципе максимума апостериорной вероятности. Для классифицируемого объекта вычисляются функции правдоподобия каждого из классов, по ним вычисляются апостериорные вероятности классов. Байесовский классификатор использует оценку апостериорного максимума (*Maximum a posteriori estimation*) для определения наиболее вероятного класса. Объект относится к тому классу, для которого апостериорная вероятность максимальна.

Байесовский подход к классификации основан на теореме, утверждающей, что если плотности распределения каждого из классов известны, то искомый алгоритм можно выписать в явном аналитическом виде. Более того, этот алгоритм оптimalен, то есть обладает минимальной вероятностью ошибок.

На практике плотности распределения классов не известны. Их приходится оценивать (восстановливать) по обучающей выборке. В результате байесовский алгоритм перестаёт быть оптимальным, так как восстановить плотность по выборке можно только с некоторой погрешностью. Чем короче выборка, тем выше шансы подогнать распределение под конкретные данные и столкнуться с эффектом переобучения.

К числу байесовских методов классификации относятся:

Наивный байесовский классификатор	Метод парзеновского окна
Линейный дискриминант Фишера	Метод радиальных базисных функций (RBF)
Квадратичный дискриминант	Логистическая регрессия

# Модель наивного байесовского классификатора

Вероятностная модель для классификатора — это условная модель:

$$p(C|F_1, \dots, F_n)$$

над зависимой переменной класса  $C$  с малым количеством результатов или классов, зависящая от нескольких переменных  $F_1, \dots, F_n$ .

Используя теорему Байеса, запишем:

$$p(C | F_1, \dots, F_n) = \frac{p(C) p(F_1, \dots, F_n | C)}{p(F_1, \dots, F_n)}.$$

На практике интересен лишь числитель этой дроби, так как знаменатель не зависит от  $C$  и значения свойств  $F_i$  даны, так что знаменатель — константа.

Числитель эквивалентен совместной вероятности модели  $p(C, F_1, \dots, F_n)$ , которая может быть переписана, используя повторные приложения определений условной вероятности:

$$\begin{aligned} p(C, F_1, \dots, F_n) &= p(C)p(F_1, \dots, F_n | C) \\ &= p(C)p(F_1 | C)p(F_2, \dots, F_n | C, F_1) \\ &= p(C)p(F_1 | C) p(F_2 | C, F_1)p(F_3, \dots, F_n | C, F_1, F_2) \\ &= p(C)p(F_1 | C) p(F_2 | C, F_1) \dots p(F_n | C, F_1, F_2, F_3, \dots, F_{n-1}) \end{aligned}$$

# «Наивные» предположения условной независимости

Предположим, что каждое свойство  $F_i$  условно независимо от любого другого свойства  $F_j$  при  $j \neq i$ . Это означает, что если вероятность появления события  $F_i|C$ , не зависит от  $F_j$ , значит  $F_j$  можно отбросить

$$p(F_i|C, F_j) = p(F_i|C)$$

таким образом, совместная модель может быть выражена как:

$$\begin{aligned} p(C, F_1, \dots, F_n) &= p(C) \cdot p(F_1|C) \cdot p(F_2|C) \cdot p(F_3|C) \cdots \cdot p(F_n|C) = \\ &= p(C) \prod_{i=1}^n p(F_i|C) \end{aligned}$$

Это означает, что из предположения о независимости, условное распределение по классовой переменной  $C$  может быть выражено так:

$$p(C|F_1, \dots, F_n) = \frac{1}{Z} p(C) \prod_{i=1}^n p(F_i|C)$$

где  $Z = p(F_1, \dots, F_n)$  — это масштабный множитель, зависящий только от  $F_1, \dots, F_n$ , то есть константа, если значения переменных известны.

# Оценка параметров

Все параметры модели могут быть аппроксимированы относительными частотами из набора данных обучения. Это оценки максимального правдоподобия вероятностей. Непрерывные свойства, как правило, оцениваются через нормальное распределение. В качестве математического ожидания и дисперсии вычисляются статистики — среднее арифметическое и среднеквадратическое отклонение соответственно.

Если данный класс и значение свойства никогда не встречаются вместе в наборе обучения, тогда оценка, основанная на вероятностях, будет равна нулю. Это проблема, так как при перемножении нулевая оценка приведет к потере информации о других вероятностях. Поэтому предпочтительно проводить небольшие поправки во все оценки вероятностей так, чтобы никакая вероятность не была строго равна нулю.

# Построение классификатора по вероятностной модели

Наивный байесовский классификатор объединяет модель с правилом решения. Одно общее правило должно выбрать наиболее вероятную гипотезу; оно известно как *апостериорное правило принятия решения* (maximum a posteriori - MAP). Соответствующий классификатор — это функция *classify*, определённая следующим образом:

$$\text{Classify}(f_1, \dots, f_n) = \arg \max_c \left( p(C = c) \cdot \prod_{i=1}^n p(F_i = f_i | C = c) \right)$$

# Байесовская фильтрация спама

Байесовская фильтрация спама — метод для фильтрации спама, основанный на применении наивного байесовского классификатора, опирающегося на прямое использование теоремы Байеса.

## *История*

Первой известной программой, фильтрующей почту с использованием байесовского классификатора, была программа iFile Джейсона Ренни, выпущенная в 1996 году. Программа использовала сортировку почты по папкам.



Первая академическая публикация по наивной байесовской фильтрации спама появилась в 1998 году. Вскоре после этой публикации была развернута работа по созданию коммерческих фильтров спама.

В 2002 г. Пол Грэм смог значительно уменьшить число ложноположительных срабатываний до такой степени, что байесовский фильтр мог использоваться в качестве единственного фильтра спама.

# Описание

При обучении фильтра для каждого встреченного в письмах слова высчитывается и сохраняется его «вес» — оценка вероятности того, что письмо с этим словом — спам. В простейшем случае в качестве оценки используется частота: «появлений в спаме / появлениях всего».

При проверке вновь пришедшего письма вероятность «спамовости» вычисляется по формуле (*Classify*) для множества гипотез.

$$P(B) = \sum_{i=1}^N P(A_i)P(B|A_i)$$

В данном случае «гипотезы» — это слова, и для каждого слова «достоверность гипотезы»  $P(A_i) = N_{wordi}/N_{words\ total}$  — доля этого слова в письме, а «зависимость события от гипотезы»  $P(B|A_i)$  — вычисленный ранее «вес» слова. То есть «вес» письма в данном случае — усреднённый «вес» всех его слов.

Отнесение письма к «спаму» или «не-спаму» производится по тому, превышает ли его «вес» некую планку, заданную пользователем (обычно берут 60-80 %). После принятия решения по письму в базе данных обновляются «веса» для вошедших в него слов.

# Математические основы

Почтовые байесовские фильтры основываются на теореме Байеса. Теорема Байеса используется несколько раз в контексте спама:

- в первый раз, чтобы вычислить вероятность, что сообщение — спам, зная, что данное слово появляется в этом сообщении;
- во второй раз, чтобы вычислить вероятность, что сообщение — спам, учитывая все его слова (или соответствующие им подмножества);
- иногда в третий раз, когда встречаются сообщения с редкими словами.

Вычисление вероятности того, что сообщение, содержащее данное слово, является спамом:

$$P(S|W) = \frac{P(W|S) * P(S)}{P(W)} = \frac{P(W|S) * P(S)}{P(W|S) * P(S) + P(W|H) * P(H)}$$

$P(S|W)$  — условная вероятность того, что сообщение—спам ( $S$ ) , при условии, что слово  $W=«replica»$  находится в нём;

$P(W)$  — полная вероятность того, что слово «Replica» содержится в сообщении

$P(S)$  — полная вероятность того, что произвольное сообщение—спам;

$P(W|S)$  — условная вероятность того, что слово «replica» появляется в сообщениях, если они являются спамом;

$P(H)$  — полная вероятность того, что произвольное сообщение не спам  $H$ ;

$P(W|H)$  — условная вероятность того, что слово «replica» появляется в сообщениях, если они не спам (то есть «ham»).

# Спамовость слова

Большинство байесовских программ обнаружения спама делают предположение об отсутствии априорных предпочтений у сообщения быть «spam» или «ham», и полагают, что у обоих случаев есть равные вероятности 50 %:  $P(S)=0.5$ ,  $P(H)=0.5$ . О фильтрах, которые используют эту гипотезу, говорят как о фильтрах «без предубеждений» и данное предположение позволяет упрощать общую формулу до:

$$P(S|W) = \frac{P(W|S)}{P(W|S) + P(W|H)}$$

Значение  $P(S|W)$  называют «спамовостью» слова  $W$ ;

$P(W|S)$  приближённо равно относительной частоте сообщений, содержащих слово  $W$  и идентифицированных как спам во время фазы обучения:

$$P(W_i|S) = \frac{\text{count}(M: W_i \in M, M \in S)}{\sum_j \text{count}(M: W_j \in M, M \in S)}$$

$P(W|H)$  приближённо равно относительной частоте сообщений, содержащих слово  $W$  и идентифицированных как «ham» во время фазы обучения:

$$P(W_i|H) = \frac{\text{count}(M: W_i \in M, M \in H)}{\sum_j \text{count}(M: W_j \in M, M \in H)}$$

Для того, чтобы эти приближения имели смысл, набор обучающих сообщений должен быть большим и достаточно представительным. Также желательно, чтобы набор обучающих сообщений соответствовал 50 % гипотезе о перераспределении между спамом и «ham», то есть что наборы сообщений «spam» и «ham» имели один и тот же размер.

# Объединение индивидуальных вероятностей

Для решения задачи классификации сообщений лишь на 2 класса:  $S$  (спам) и  $H = \neg S$  («ham» - не спам) из теоремы Байеса можно вывести следующую формулу оценки вероятности «спамовости» всего сообщения, содержащего слова  $W_1, W_2, \dots, W_N$ :

$$\begin{aligned} p(S|W_1, W_2, \dots, W_N) &= [\text{по теореме Байеса}] = \frac{p(W_1, W_2, \dots, W_N|S)*p(S)}{p(W_1, W_2, \dots, W_N)} = \\ &= [\text{так как } W_i \text{ предполагаются независимыми}] = \frac{\prod_i p(W_i|S)*p(S)}{p(W_1, W_2, \dots, W_N)} = \\ &= [\text{по теореме Байеса}] = \frac{\prod_i \frac{p(S|W_i)*p(W_i)}{p(S)} * p(S)}{p(W_1, W_2, \dots, W_N)} = \\ &= [\text{по формуле полной вероятности}] = \frac{\prod_i \frac{p(S|W_i)*p(W_i)}{p(S|W_i)*p(W_i)} * p(S)}{\prod_i (p(W_i|S))*p(S) + \prod_i (p(W_i|\neg S))*p(\neg S)} = \\ &= \frac{\prod_i (p(S|W_i)*p(W_i)) * p(S)^{1-N}}{\prod_i (p(S|W_i)*p(W_i)) * p(S)^{1-N} + \prod_i (p(\neg S|W_i)*p(W_i)) * p(\neg S)^{1-N}} = \\ &= \frac{\prod_i p(S|W_i)}{\prod_i (p(S|W_i)) + \left(\frac{p(\neg S)}{p(S)}\right)^{1-N} * \prod_i p(\neg S|W_i)}. \end{aligned}$$

# В результате

Таким образом, предполагая  $p(S) = p(\neg S)=0.5$ , имеем:

$$p = \frac{p_1 p_2 \dots p_N}{p_1 p_2 \dots p_N + (1 - p_1)(1 - p_2) \dots (1 - p_N)}$$

где:

- $p = p(S | W_1, W_2, \dots, W_N)$  — вероятность, что сообщение, содержащее слова  $W_1, W_2, \dots, W_N$  — спам;
- $p_1$  — условная вероятность  $p(S|W_1)$  того, что сообщение — спам, при условии, что оно содержит первое слово (к примеру, «replica»);
- $p_2$  — условная вероятность  $p(S|W_2)$  того, что сообщение — спам, при условии, что оно содержит второе слово (к примеру, «watches»);
- $p_N$  — условная вероятность  $p(S|W_N)$  того, что сообщение — спам, при условии, что оно содержит N-е слово (к примеру, «home»).

Результат  $p$  обычно сравнивают с некоторым порогом (например, 0.5, чтобы решить, является ли сообщение спамом или нет. Если  $p$  ниже, чем порог, сообщение рассматривают как вероятный «ham», иначе его рассматривают как вероятный спам.

# Проблема редких слов

Она возникает в случае, если слово никогда не встречалось во время фазы обучения: и числитель, и знаменатель равны нулю, и в общей формуле, и в формуле спамовости.

В целом, слова, с которыми программа столкнулась только несколько раз во время фазы обучения, не являются репрезентативными (набор данных в выборке мал для того, чтобы сделать надёжный вывод о свойстве такого слова). Простое решение состоит в том, чтобы игнорировать такие ненадёжные слова.

«Нейтральные» слова — такие, как, «the», «а», «some», или «is» (в английском языке), или их эквиваленты на других языках — могут быть проигнорированы. Вообще говоря, некоторые байесовские фильтры просто игнорируют все слова, у которых спамовость около 0.5, так как в этом случае получается качественно лучшее решение. Учитываются только те слова, спамовость которых около 0.0 (отличительный признак законных сообщений — «ham»), или рядом с 1.0 (отличительный признаки спама).

# Выводы о классификаторе спама

## ПЛЮСЫ

- Прост
- Удобен
- Эффективен

## МИНУСЫ

- Базируется на предположении, что одни слова чаще встречаются в спаме, а другие — в обычных письмах, и неэффективен, если данное предположение неверно
- Работает только с текстом

**Спасибо за внимание**

# **Методы машинного обучения**

*Лекция 6*

**Классификация -  
Продолжение**

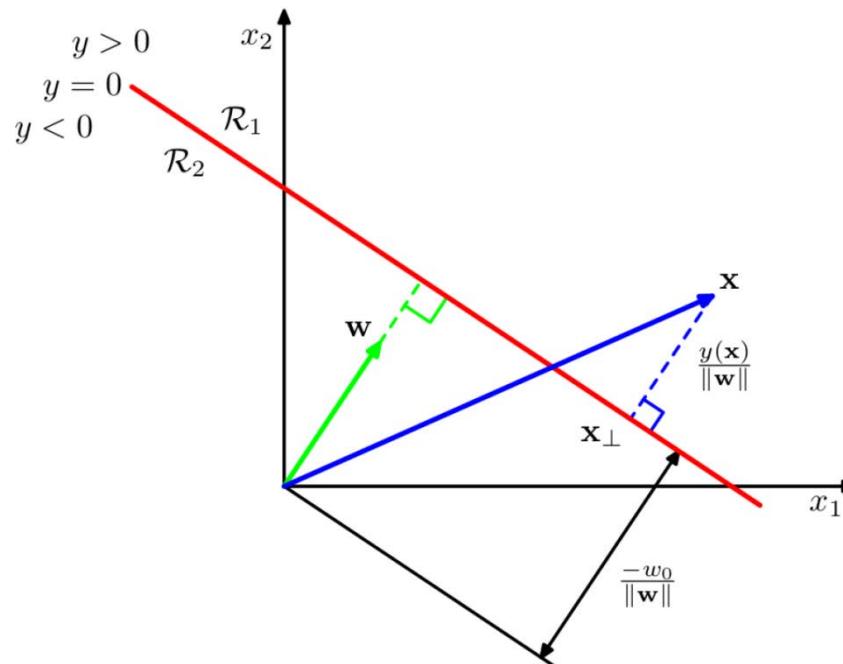
# Задача классификации и разделяющая гиперплоскость

- Рассмотрим линейную дискриминантную функцию:

$$y(x) = w^T x + w_0$$

является гиперплоскостью и  $w$  – нормаль к ней.

- Расстояние от начала координат до гиперплоскости:  $\frac{-w_0}{\|w\|}$
- $y(x)$  связано с расстоянием до гиперплоскости:  $d = \frac{y(x)}{\|w\|}$

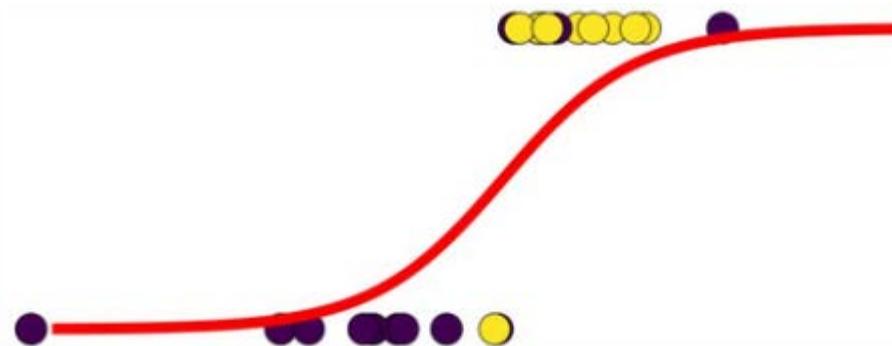


# Логистическая регрессия: определение

Логистическая регрессия или логит-модель (*logit model*) — статистическая модель, используемая для прогнозирования вероятности возникновения некоторого события путём его сравнения с логистической кривой. Эта регрессия выдаёт ответ в виде вероятности бинарного события (1 или 0).

Применяется для прогнозирования вероятности возникновения некоторого события по значениям множества признаков.

- вводится так называемая зависимая переменная  $y \in \{0, 1\}$  (0 - событие не произошло и 1 - событие произошло);
- множество независимых переменных (также называемых признаками, предикторами или регрессорами) —  $x_1, x_2, \dots, x_n \in \mathbb{R}$ , на основе значений которых требуется вычислить вероятность принятия того или иного значения зависимой переменной;
- для простоты записи вводится фиктивный признак  $x_0 = 1$ .



# Логистическая регрессия: описание модели

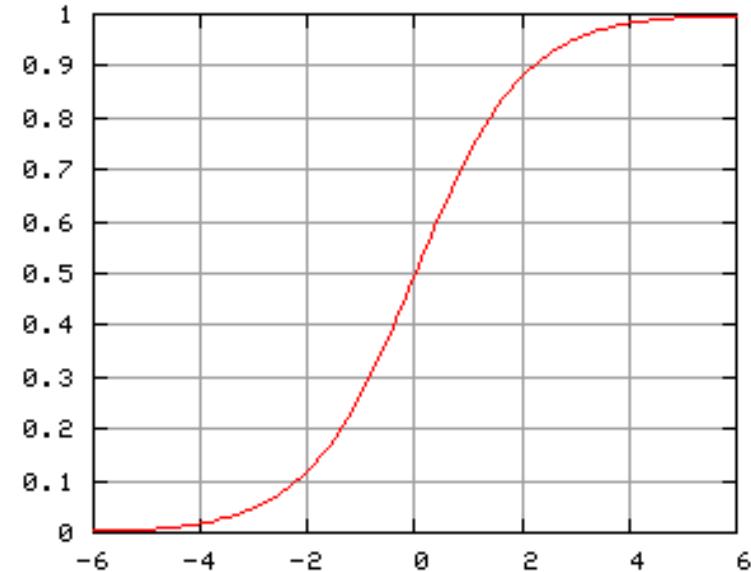
Делается предположение о том, что вероятность наступления события  $y = 1$  равна:

$$P\{y = 1|x\} = f(z), \quad z = \theta^T x = \theta_0 + \theta_1 x_1 + \dots + \theta_n x_n$$

где

- $x$  – вектор-столбец значений независимых переменных 1,  $x_1, x_2, \dots, x_n$
- $\theta$  – вектор-столбец параметров (коэффициентов регрессии) – вещественные числа  $\theta_0, \dots, \theta_n$
- $f(z)$  – логистическая функция (сигмоида или логит-функция):

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



Т.к.  $y$  принимает значения 0 и 1, то вероятность принять значение 0 равна:

$$P\{y = 0|x\} = 1 - f(z) = 1 - f(\theta^T x)$$

Функция распределения  $y$  при заданном  $x$ :

$$P\{y|x\} = f(\theta^T x)^y (1 - f(\theta^T x))^{1-y}, \quad y \in \{0, 1\}$$

# Логистическая регрессия: подбор параметров

Для подбора параметров  $\theta_0, \dots, \theta_n$  необходимо составить обучающую выборку, состоящую из наборов значений независимых переменных и соответствующих им значений зависимой переменной  $y$ .

- Формально, это множество пар  $(x^{(1)}, y^{(1)}), \dots, (x^{(m)}, y^{(m)})$ , где  $x^{(i)} \in \mathbb{R}^n$  и  $y^{(i)} \in \{0, 1\}$ .
- Обычно используется метод максимального правдоподобия, согласно которому подбираются параметры  $\theta$ , максимизирующие значение функции правдоподобия на обучающей выборке:

$$\hat{\theta} = \underset{\theta}{\operatorname{argmax}} L(\theta) = \underset{\theta}{\operatorname{argmax}} \prod_{i=1}^m P\{y = y^{(i)} | x = x^{(i)}\}$$

Определяются оценки максимального правдоподобия (maximum likelihood estimates), для которых значения параметров являются наиболее «правдоподобными» по отношению к наблюдаемым данным.

# Подбор параметров: метод Ньютона

Максимизация функции правдоподобия эквивалентна максимизации её логарифма:

$$\begin{aligned}\ln L(\theta) &= \sum_{i=1}^m \ln P\{y = y^{(i)} | x = x^{(i)}\} = \sum_{i \in I_1} \ln P_i(\theta) + \sum_{i \in I_0} \ln(1 - P_i(\theta)) \\ &= \sum_{i=1}^m [y^{(i)} \ln f(\theta^T x^{(i)}) + (1 - y^{(i)}) \ln(1 - f(\theta^T x^{(i)}))]\end{aligned}$$

где  $\theta^T x^{(i)} = \theta_0 + \theta_1 x_1^{(i)} + \dots + \theta_n x_n^{(i)}$ ,  $f(z) = \frac{1}{1+e^{-z}}$   $I_0, I_1$  - множества наблюдений, для которых  $y^{\{i\}} = 0$  и  $y^{\{i\}} = 1$  соответственно.

Можно показать, что градиент:  $grad = \sum_i (y^{(i)} - f(\theta^T x)) x^{(i)}$

Гессиан  $H$  функции правдоподобия равен:  $H = -\sum_i P_i(1 - P_i) X_i^T X_i \leq 0$

**Гессиан функции** — симметрическая квадратичная форма, описывающая поведение функции во втором порядке. Для функции  $f(\theta)$  дважды дифференцируемой в точке  $\theta$ .

Гессиан всюду отрицательно определенный, поэтому логарифмическая функция правдоподобия всюду вогнута. Для поиска максимума можно использовать метод Ньютона, который в этом случае будет всегда сходиться

$$\theta_{t+1} = \theta_t - (H(\theta_t))^{-1} \cdot grad_t(\theta_t) = \theta_t - \Delta\theta_t$$

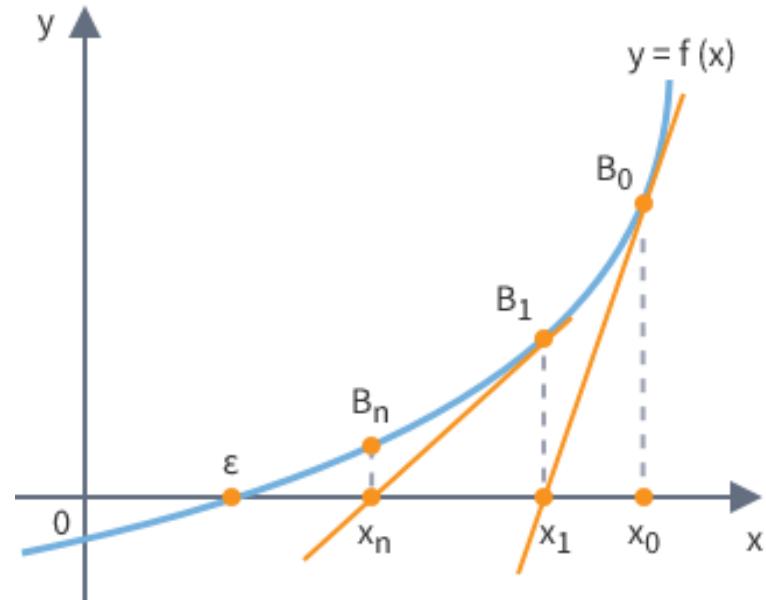
# Метод Ньютона (метод касательных)

Итерационный численный метод нахождения корня (нуля) или поиска экстремума заданной функции многих переменных. Поиск решения осуществляется путём построения последовательных приближений.

**Основная идея:** задаётся начальное приближение вблизи предположительного корня, строится касательная к графику исследуемой функции в точке приближения, для которой находится пересечение с осью абсцисс. Эта точка берётся в качестве следующего приближения и т.д.

Алгоритм нахождения численного решения уравнения  $f(x) = 0$  сводится к итерационной процедуре вычисления:

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$$



# Метод Ньютона применительно к задачам оптимизации

В случае решения задач оптимизации предполагается, что функция  $f(x)$  дважды непрерывно дифференцируема. Поиск минимума функции  $f(x)$  производится при помощи отыскания стационарной точки, т.е. точки удовлетворяющей уравнению:

$$f'(x) = 0.$$

Если  $x_k$  – точка, полученная на  $k$ -м шаге, то функция  $f'(x)$  аппроксимируется своим уравнением касательной:

$$y = f'(x_k) + (x - x_k) \cdot f''(x_k),$$

а точка  $x^{k+1}$  выбирается как пересечение этой прямой с осью  $Ox$ , т.е.

$$x_{k+1} = x_k - \frac{f'(x_k)}{f''(x_k)}.$$

**В общем виде:** Пусть необходимо найти минимум функции многих переменных  $f(\vec{x}): \mathbb{R}^n \rightarrow \mathbb{R}$ . Эта задача равносильна задаче нахождения нуля градиента  $\nabla f(\vec{x})$ .

$$\vec{x}_{k+1} = \vec{x}_k - H^{-1}(\vec{x}_k) \nabla f(\vec{x}_k) \quad H(f) =$$

где  $H(\vec{x})$  – гессиан функции  $f(\vec{x})$ .

**Гессиан функции** – симметрическая квадратичная форма, описывающая поведение функции во втором порядке. Для функции  $f(\vec{x})$  дважды дифференцируемой в точке  $x$ .

$$\begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 \partial x_2} & \dots & \frac{\partial^2 f}{\partial x_1 \partial x_n} \\ \frac{\partial^2 f}{\partial x_2 \partial x_1} & \frac{\partial^2 f}{\partial x_2^2} & \dots & \frac{\partial^2 f}{\partial x_2 \partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_n \partial x_1} & \frac{\partial^2 f}{\partial x_n \partial x_2} & \dots & \frac{\partial^2 f}{\partial x_n^2} \end{bmatrix}$$

# Подбор параметров: градиентный спуск

Максимизация функции правдоподобия эквивалентна максимизации её логарифма:

$$\begin{aligned}\ln L(\theta) &= \sum_{i=1}^m \log P\{y = y^{(i)} | x = x^{(i)}\} \\ &= \sum_{i=1}^m [y^{(i)} \ln f(\theta^T x^{(i)}) + (1 - y^{(i)}) \ln(1 - f(\theta^T x^{(i)}))],\end{aligned}$$

где  $\theta^T x^{(i)} = \theta_0 + \theta_1 x_1^{(i)} + \dots + \theta_n x_n^{(i)}$ ,  $f(z) = \frac{1}{1+e^{-z}}$ ,

а градиент  $\nabla = \sum_i (y^{(i)} - f(\theta^T x)) x^{(i)}$ .

Для максимизации этой функции может быть применён, например, метод градиентного спуска. Он заключается в выполнении следующих итераций, начиная с некоторого начального значения параметров  $\theta$ :

$$\theta_{t+1} = \theta_t + \lambda \nabla \ln L(\theta) = \theta_t + \lambda \sum_{i=1}^m (y^{(i)} - f(\theta^T x^{(i)})) x^{(i)}, \lambda > 0.$$

# Метод градиентного спуска

Численный метод нахождения локального минимума или максимума функции с помощью движения вдоль градиента.

Пусть целевая функция имеет вид:  $F(\vec{x}): \mathbb{X} \rightarrow \mathbb{R}$ .

Задача оптимизации сформулирована следующим образом:  $F(\vec{x}) \rightarrow \min_{\vec{x} \in \mathbb{X}}$  (найти минимум).

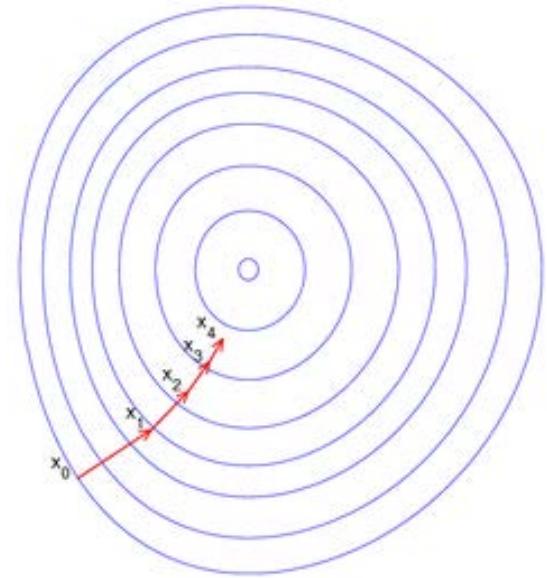
**Основная идея** метода заключается в том, чтобы идти в направлении наискорейшего спуска, а это направление задаётся антиградиентом  $-\nabla F$ :

$$\vec{x}^{[j+1]} = \vec{x}^{[j]} - \lambda^{[j]} \nabla F(\vec{x}^{[j]}),$$

где

- градиентом  $\text{grad } F = \nabla F$  называется  $n$ -мерный вектор  $\left( \frac{\partial F}{\partial x_1}, \frac{\partial F}{\partial x_2}, \dots, \frac{\partial F}{\partial x_n} \right)$ , компоненты которого равны частным производным  $F$  по всем ее аргументам.
- $\lambda^{[j]}$  задает скорость градиентного спуска и может быть выбрана:
  - постоянной (в этом случае метод может не сходиться);
  - убывающей в процессе градиентного спуска;
  - гарантирующей наискорейший спуск, тогда для поиска минимума  $F(\vec{x})$  получаем:

$$\lambda^{[j]} = \underset{\lambda}{\operatorname{argmin}} F(\vec{x}^{[j+1]}) = \underset{\lambda}{\operatorname{argmin}} F(\vec{x}^{[j]} - \lambda^{[j]} \nabla F(\vec{x}^{[j]}))$$



# Метод градиентного спуска: Алгоритм

1. Задать начальное приближение и точность расчета:  $\vec{x}^{[0]}, \varepsilon$ .
2. Рассчитать  $\vec{x}^{[j+1]} = \vec{x}^{[j]} - \lambda^{[j]} \nabla F(\vec{x}^{[j]})$ ,  
где  $\lambda^{[j]} = \underset{\lambda}{\operatorname{argmin}} F(\vec{x}^{[j]} - \lambda^{[j]} \nabla F(\vec{x}^{[j]}))$ .
3. Проверить условие остановки:
  - Если  $|\vec{x}^{[j+1]} - \vec{x}^{[j]}| > \varepsilon$ ,  $|F(\vec{x}^{[j+1]}) - F(\vec{x}^{[j]})| > \varepsilon$  или  $\|\nabla F(\vec{x}^{[j+1]})\| > \varepsilon$  (выбирается одно из условий),  
то  $j = j + 1$  перейти к шагу 2.
  - Иначе  $\vec{x} = \vec{x}^{[j+1]}$  и останов.

# Метод градиентного спуска: Особенности

- Рис.1. Геометрическая интерпретация метода градиентного спуска с постоянным шагом. На каждом шаге мы сдвигаемся по вектору антиградиента, "уменьшенному в  $\lambda$  раз".
- Рис.2. Ситуация, когда метод градиентного спуска сходится медленно.
- Рис.3. Геометрическая интерпретация метода наискорейшего спуска. На каждом шаге  $\lambda^{[j]}$  выбирается так, чтобы следующая итерация была точкой минимума функции на луче  $L$ .
- Рис.4. Схождение к разным локальным минимумам в зависимости от начальной точки старта.

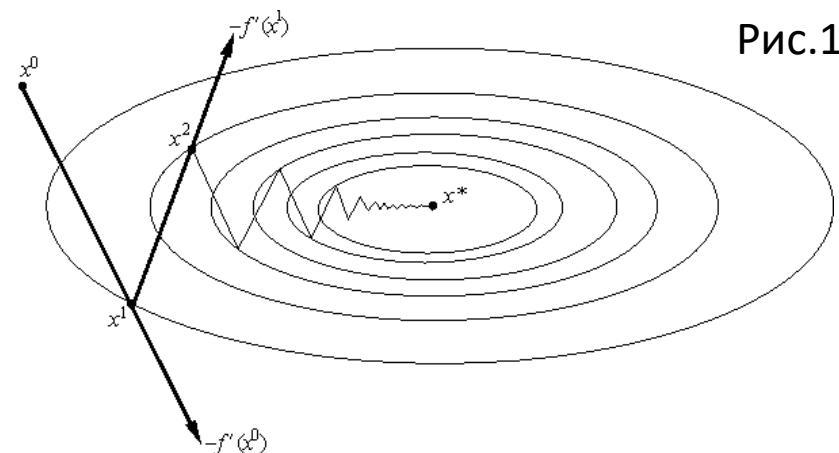


Рис.1.

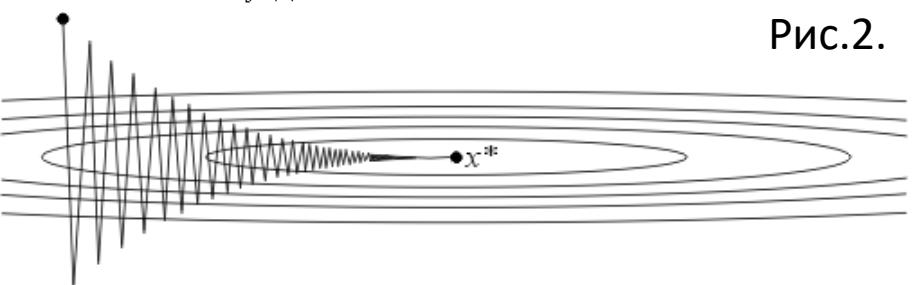


Рис.2.

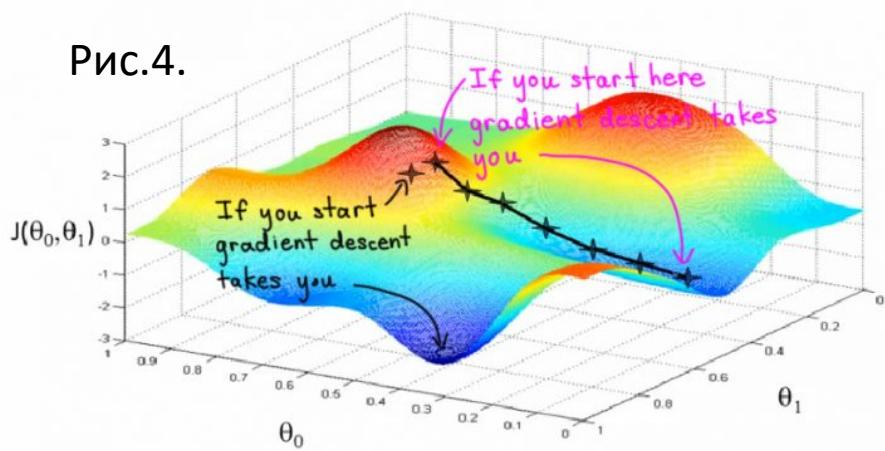


Рис.4.

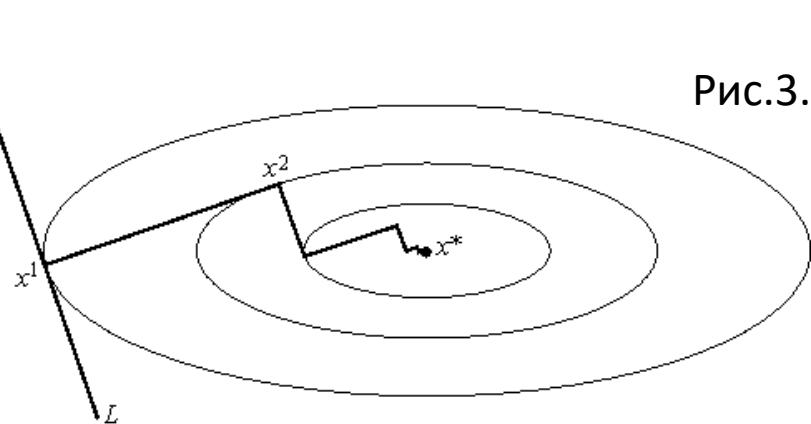


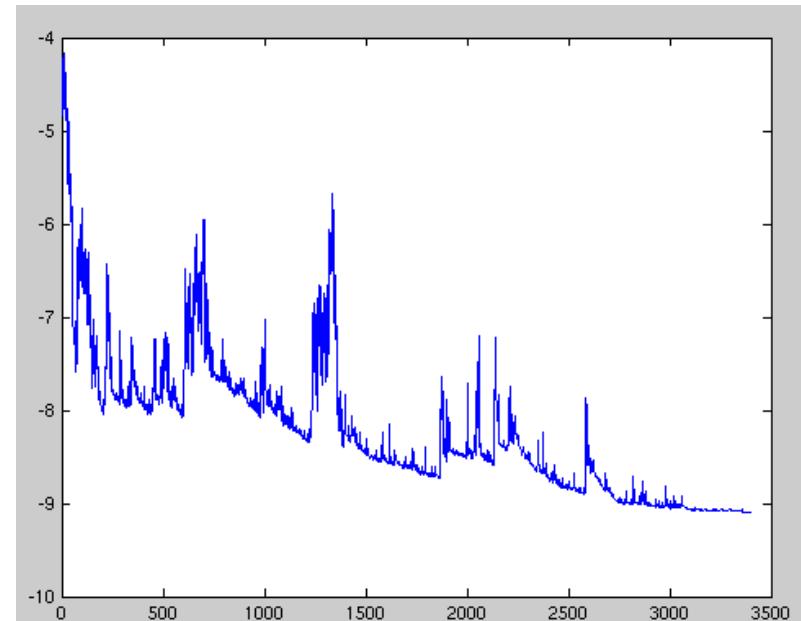
Рис.3.

# Стохастический градиентный спуск (Stochastic gradient descent, SGD)

Итерационный метод для оптимизации целевой функции с подходящими свойствами гладкости (например, дифференцируемость или субдифференцируемость). Его можно расценивать как стохастическую аппроксимацию оптимизации методом градиентного спуска, поскольку он заменяет реальный градиент, вычисленный из полного набора данных, оценкой, вычисленной из случайно выбранного подмножества данных.

Это сокращает задействованные вычислительные ресурсы и помогает достичь более высокой скорости итераций в обмен на более низкую скорость сходимости.

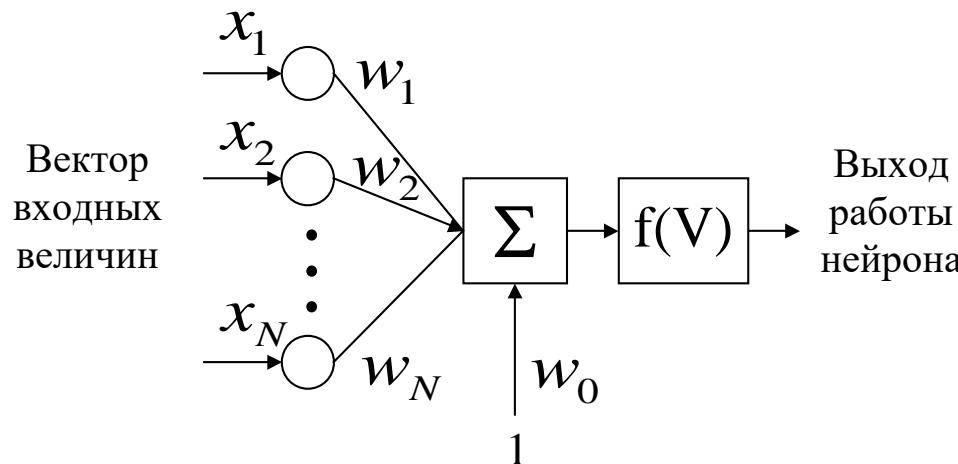
Особенно большой эффект достигается в приложениях, связанных с обработкой больших данных.



# Представление в виде однослойной нейронной сети

Логистическую регрессию можно представить в виде однослойной нейронной сети с сигмоидальной функцией активации, веса которой есть коэффициенты логистической регрессии, а вес поляризации — константа регрессионного уравнения.

Структурная схема нейрона:



$$V = \sum_{i=0}^N w_i \cdot x_i$$

$$f(V) = \frac{1}{1 + \exp(-bV)}$$

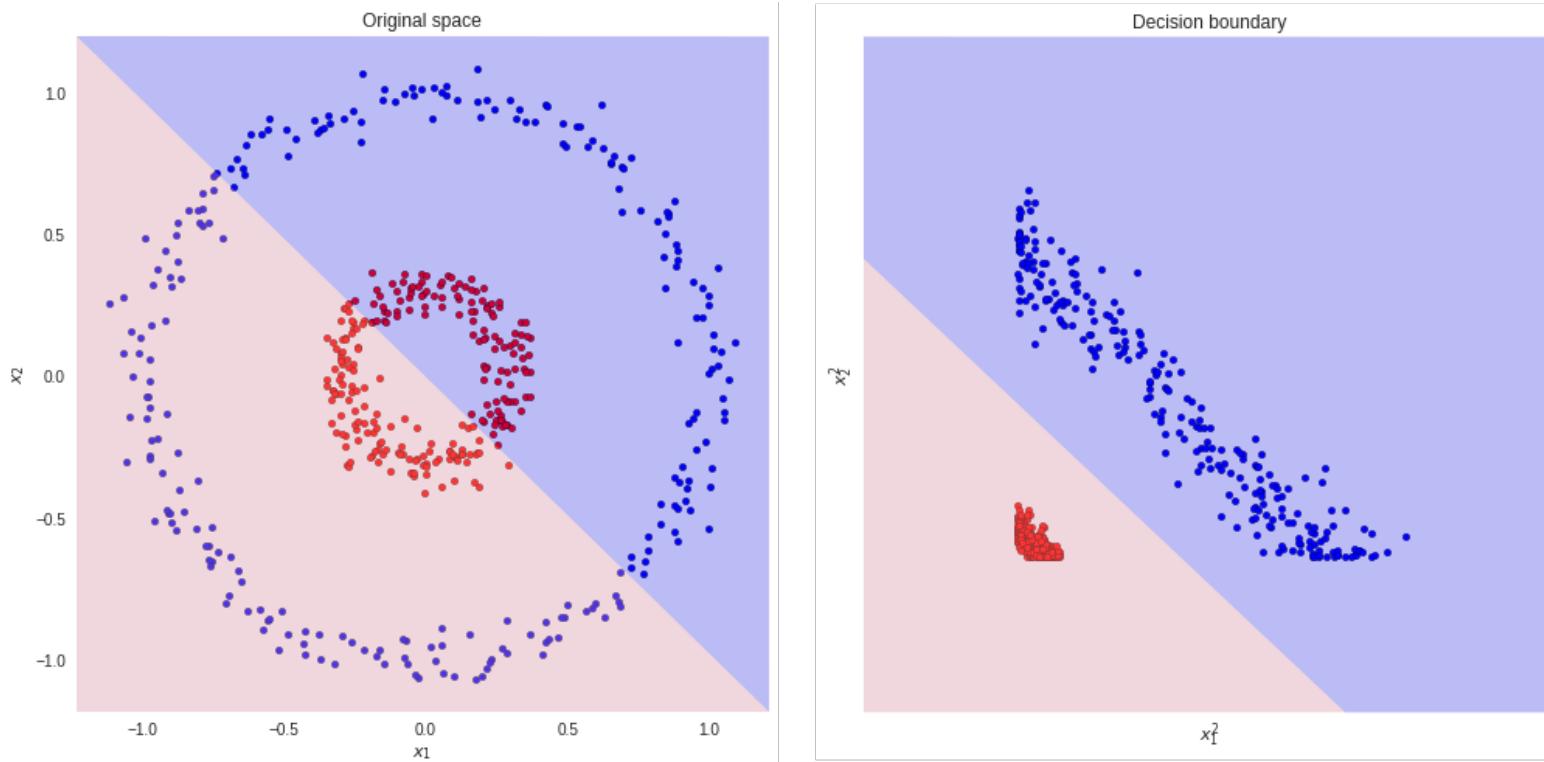
Однослойная нейронная сеть может успешно решить лишь задачу линейной сепарации. Поэтому возможности по моделированию нелинейных зависимостей у логистической регрессии отсутствуют.

# Другой взгляд на классификацию

- В линейном случае мы хотим спроектировать точки в размерность 1 (на нормаль разделяющей гиперплоскости) так, чтобы в этой одномерной размерности они хорошо разделялись
- В этом смысле классификация – это метод радикального сокращения размерности.
- Рассмотрим классификацию с этих позиций и в каком-то смысле попробуем добиться оптимальности.

# Проектирование исходных данных в новое пространство

Для решения задач классификации с двумя или более классами большинство алгоритмов машинного обучения применяют какое-то преобразование к входным данным с эффектом уменьшения исходных входных измерений до меньшего числа. Цель состоит в том, чтобы спроектировать данные в новое пространство. Затем, после проектирования, алгоритм пытается классифицировать точки путем нахождения линейного разделения.



# Линейный дискриминант Фишера

**Линейный дискриминантный анализ** (ЛДА), а также связанный с ним линейный дискриминант Фишера — методы статистики и машинного обучения, применяемые для нахождения линейных комбинаций признаков, наилучшим образом разделяющих два или более класса объектов или событий. Полученная комбинация может быть использована в качестве линейного классификатора или для сокращения размерности пространства признаков перед последующей классификацией.

**Линейный дискриминант Фишера** в первоначальном значении - метод, определяющий расстояние между распределениями двух разных классов объектов или событий. Он может использоваться в задачах машинного обучения при статистическом (байесовском) подходе к решению задач классификации.

Предположим, что обучающая выборка удовлетворяет помимо базовых гипотез байесовского классификатора также следующим гипотезам:

- Классы распределены по нормальному закону
- Матрицы ковариаций классов равны

Тогда статистический подход приводит к линейному дискриминанту, и именно этот алгоритм классификации в настоящее время часто понимается под термином *линейный дискриминант Фишера*

# Разделение на два класса на плоскости

Рассмотрим случай задачи классификации двух классов ( $K = 2$ ). Синие и красные точки в  $\mathbb{R}^2$ . В общем виде, можно взять любой входной D-мерный вектор и спроектировать его на D' измерение. В этом случае  $D$  представляет размерность исходного входного пространства, в то время как  $D'$  - это новая размерность пространства.

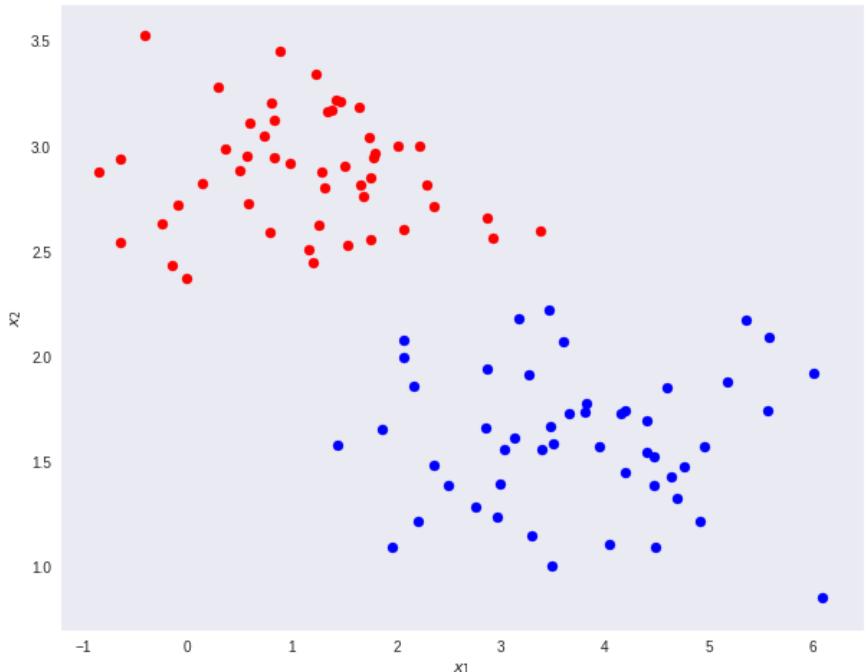
В случае проецирования на одномерное измерение (числовую линию), т.е.  $D'=1$  можно выбрать порог  $T$  для разделения классов в новом пространстве. Учитывая входной вектор  $X$ :

- если прогнозируемое значение  $Y \geq T$  тогда  $X$  принадлежит к классу C1 (класс 1).
- в противном случае он классифицируется как C2 (класс 2).

Вектор  $Y$  (прогнозы) равен линейной комбинации входов  $X$  и весов  $W$ .

$$Y = W^T X$$

Хотим уменьшить исходные размеры данных от  $D=2$  в  $D'=1$ . Другими словами, получить преобразование, которое отображает векторы  $\mathbb{R}^2 \rightarrow \mathbb{R}^1$ .



# Первая идея

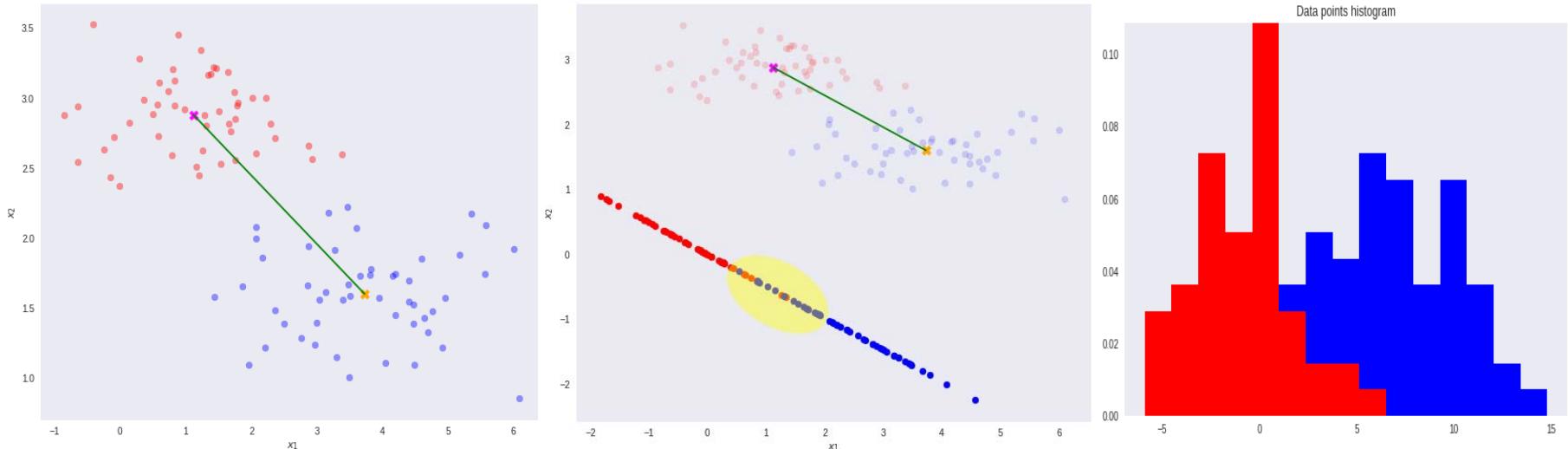
Рассмотрим два класса  $C_1$  и  $C_2$  с  $N_1$  и  $N_2$  точками.

Первая идея – найти серединный перпендикуляр между центрами кластеров. Вычислим средние векторы  $m_1$  и  $m_2$ .

$$m_1 = \frac{1}{N_1} \sum_{n \in C_1} x_n, \quad m_2 = \frac{1}{N_2} \sum_{n \in C_2} x_n$$

Другими словами, хотим проецировать данные на вектор  $W$ , соединяющий центры кластеров.

Важно отметить, что любой вид проекции на меньшее измерение может повлечь некоторую потерю информации. В этом сценарии обратите внимание, что эти два класса четко разделимы (линией) в их исходном пространстве. После проецирования данные демонстрируют некоторое перекрытие классов – это показано желтым эллипсом на графике и гистограммой справа.



# Идея, предложенная Фишером

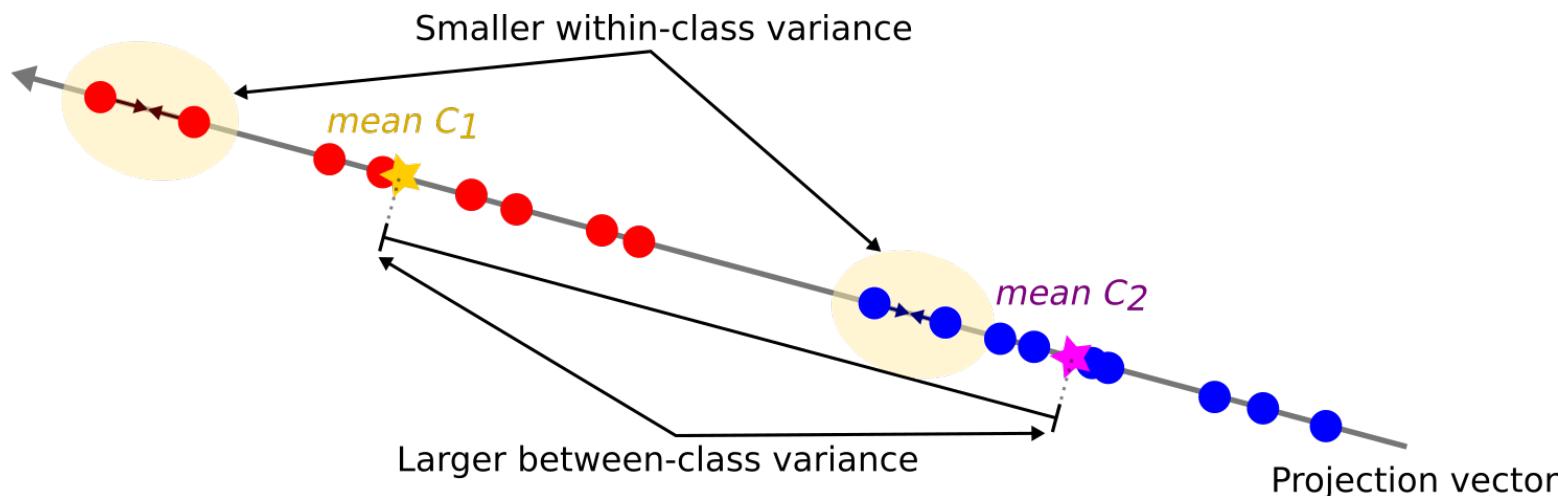
Максимизировать функцию, которая обеспечит наибольшее разделение между спроектированными центрами классов, а также даст наименьшую дисперсию внутри каждого класса, тем самым минимизируя перекрытие классов.

Другими словами, линейный дискриминант Фишера выбирает проекцию, которая максимизирует разделение классов. Для этого он максимизирует соотношение между дисперсией между классами и дисперсией внутри класса.

Для проецирования данных в меньшее измерение и во избежание перекрытия классов метод поддерживает 2 свойства.

- Большая разница между классами наборов данных.
- Небольшая дисперсия в каждом из классов наборов данных.

Обратите внимание, что большая разница между классами означает, что прогнозируемые средние классы должны быть как можно дальше друг от друга. Напротив, небольшая внутриклассовая дисперсия позволяет удерживать проецируемые точки данных ближе друг к другу в рамках одного класса.



# Поиск проекции с указанными свойствами

Выборочная дисперсия в проекции  $y_n = W^T x_n$

$$S_1^2 = \sum_{n \in C_1} (y_n - m_1)^2, \quad S_2^2 = \sum_{n \in C_2} (y_n - m_2)^2$$

Чтобы найти проекцию с указанными свойствами, Линейный дискриминант Фишера вычисляет весовой вектор  $W$  по следующему критерию (критерий Фишера):

$$J(W) = \frac{(m_2 - m_1)^2}{S_1^2 + S_2^2} = \frac{W^T S_B W}{W^T S_W W}$$

Числитель – between-class variance (межклассовая дисперсия).

Знаменатель - within-class variance (внутриклассовая дисперсия).

$$S_B = (m_2 - m_1)(m_2 - m_1)^T$$
$$S_W = \sum_{n \in C_1} (x_n - m_1)(x_n - m_1)^T + \sum_{n \in C_2} (x_n - m_2)(x_n - m_2)^T$$

После дифференцирования по  $W$  получим, что  $J(W)$  максимален при:

$$(W^T S_B W) S_W W = (W^T S_W W) S_B W$$

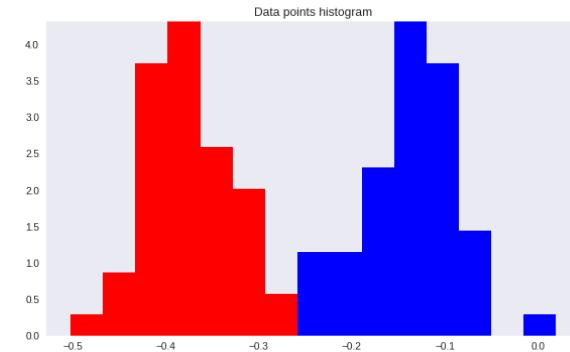
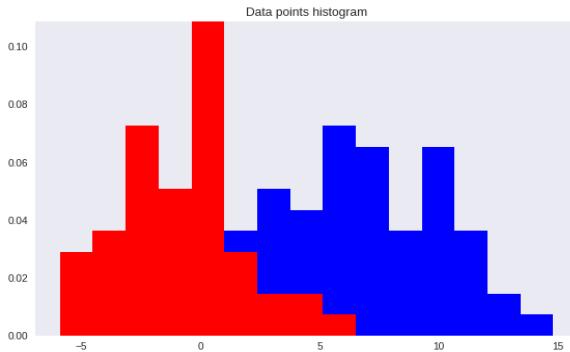
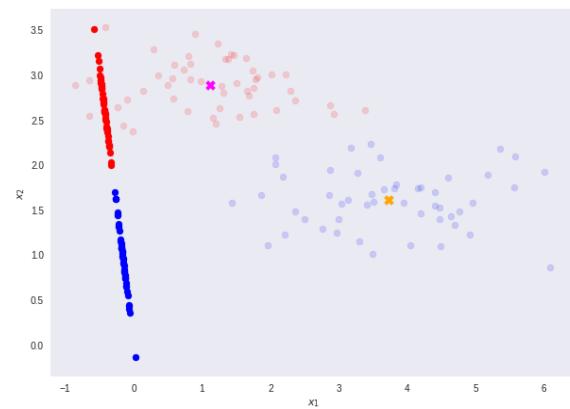
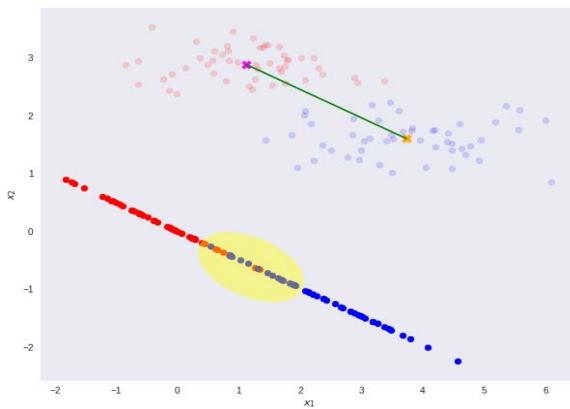
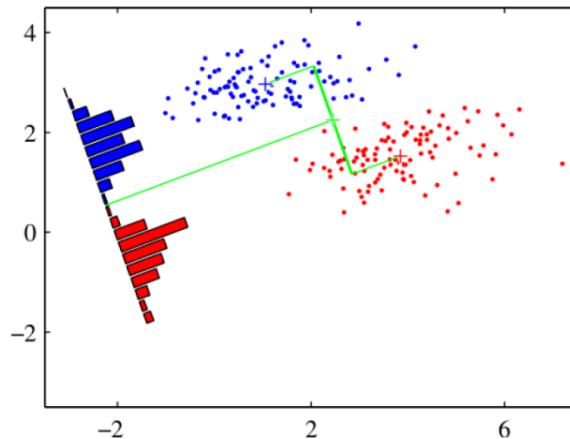
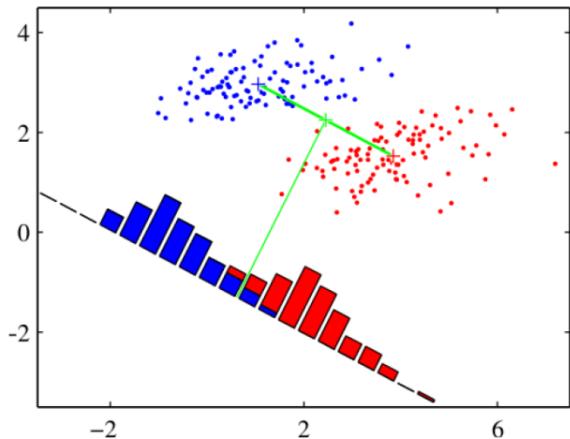
Причем  $S_B W$  будет в направлении  $m_2 - m_1$ , а длина  $W$  нас не интересует.

Получаем:

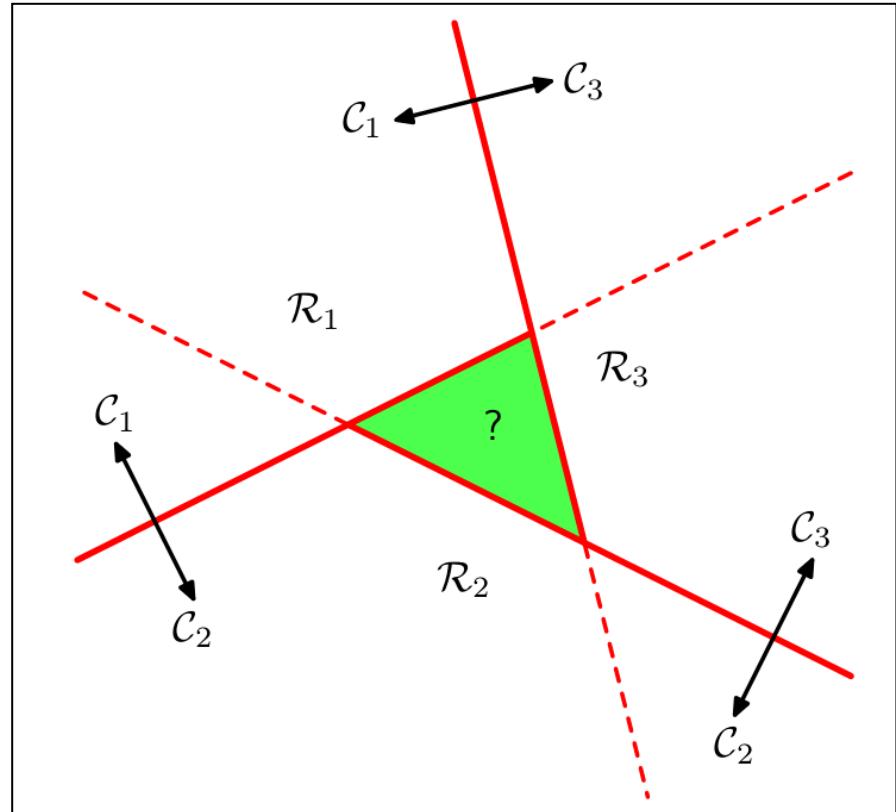
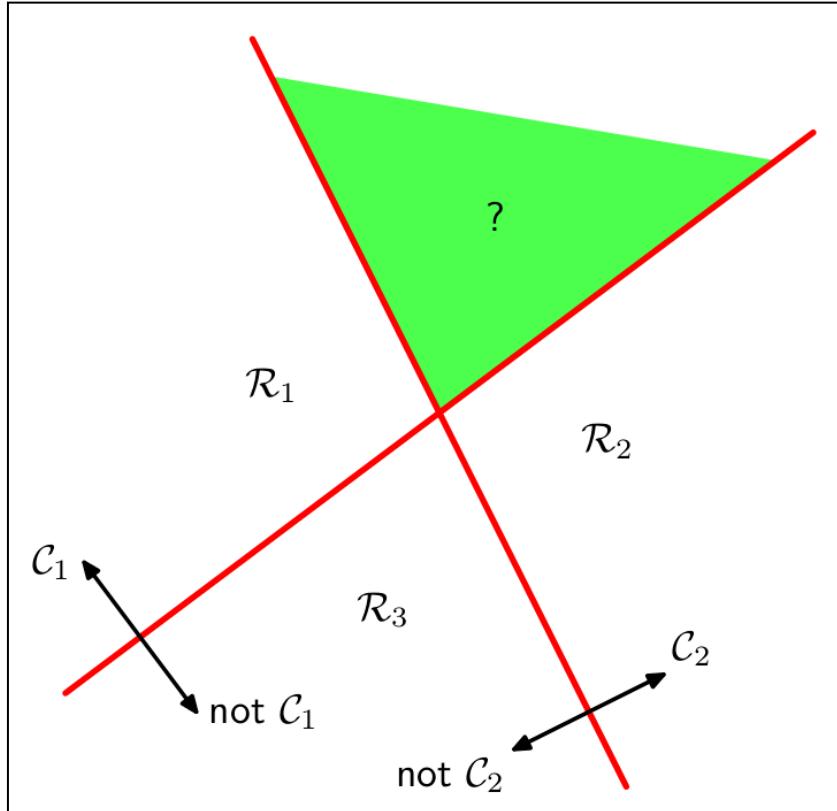
$$W \propto S_W^{-1} (m_2 - m_1)$$

То есть,  $W$  (наше желаемое преобразование) прямо пропорционально обратной внутриклассовой ковариации. Матрица умножает на разность классов.

# Геометрический смысл



# Линейное разделение и несколько классов



# Метрики качества классификации

## Матрица ошибок (Confusion Matrix)

Основной инструмент для сравнения моделей, показывает, сколько объектов каждого класса было верно и неверно классифицировано, позволяет определить, в какой степени алгоритм делает ошибки при классификации каждого класса.

В случае бинарной кластеризации, матрица ошибок состоит из четырех основных элементов:

- True Positives (TP): Количество объектов, которые были правильно классифицированы как положительные (верное предсказание положительного класса).
- False Positives (FP): Количество объектов, которые были неправильно классифицированы как положительные (ложное предсказание положительного класса), а ошибка классификации называется ошибкой I рода (**ложная тревога**).
- True Negatives (TN): Количество объектов, которые были правильно классифицированы как отрицательные (верное предсказание отрицательного класса).
- False Negatives (FN): Количество объектов, которые были неправильно классифицированы как отрицательные (ложное предсказание отрицательного класса), а ошибка классификации называется ошибкой II рода (**пропуск цели**).

	Actual Positive	Actual Negative
Predicted Positive	TP	FP
Predicted Negative	FN	TN

# Метрики качества классификации

## Точность, полнота, F-мера и т.д.

**Точность (Precision)**: показывает, как много из объектов, предсказанных как положительные, действительно принадлежат положительному классу.

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

**Полнота (Recall)**: показывает, как много из всех фактически положительных объектов было правильно обнаружено моделью.

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

**Специфичность (Specificity)**: показывает, как много из объектов отрицательного класса было правильно классифицировано.

$$Specificity = \frac{TN}{TN + FP}$$

**F-мера (F1-score)**: гармоническое среднее точности и полноты. Учитывает оба аспекта, что делает ее полезной в случаях, когда оба показателя важны.

$$F1\text{-score} = 2 * \frac{Precision * Recall}{Precision + Recall}$$

**Среднегармоническая F-мера ( $F_\beta$ )**: Для нивелирования перекоса в балансе классов используется коэффициент  $\beta$ .

$$F_\beta = (1 + \beta) * \frac{Precision * Recall}{\beta^2 * Precision + Recall}$$

**Точность точности (accuracy)**: сравнение по доле правильно классифицированных экземпляров из общего числа экземпляров в выборке.

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

# Метрики качества классификации ROC-AUC

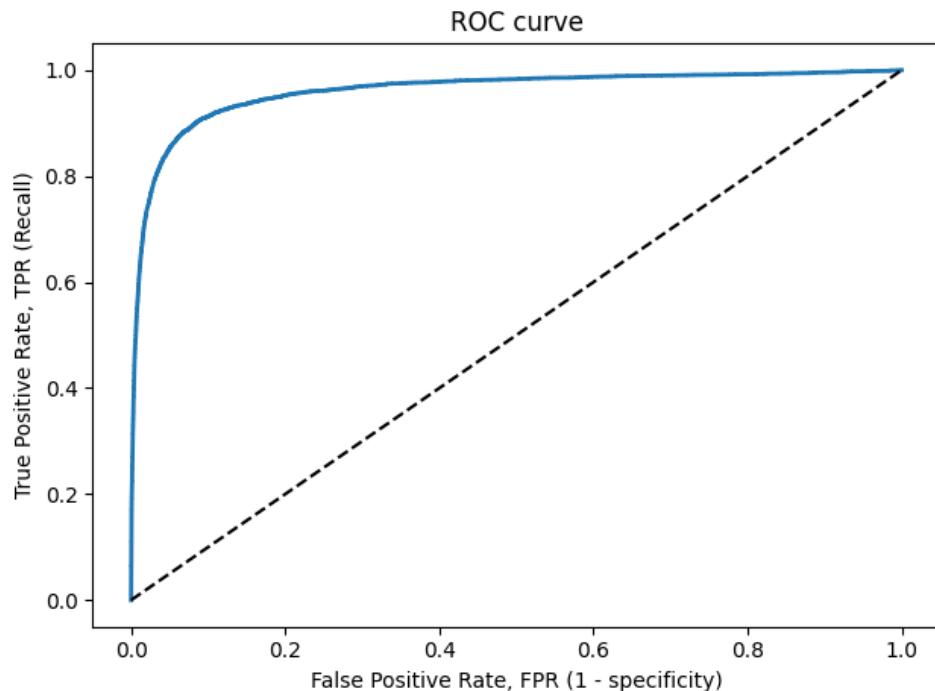
ROC — Receiver Operating Characteristic curve / AUC — Area Under Curve

ROC-кривая показывает зависимость между долей ложноположительных и долей истинно положительных классификаций. Данная оценка работы алгоритма классификации рассчитывается как площадь под ROC кривой, которая строится в координатах TPR (True Positive Rate) и FPR (False Positive Rate).

$$TPR = \frac{TP}{TP + FN}, \quad FPR = \frac{FP}{FP + TN}.$$

Алгоритм построения кривой:

1. Запустить классификатор на тестовой выборке;
2. Отсортировать результаты по уверенности классификатора в принадлежности объекта к классу;
3. Пока не кончились элементы:
  - Взять объект с максимальной уверенностью;
  - Сравнить метку с реальной;
  - Пересчитать TPR и FPR на взятых объектах;
  - Поставить точку;
4. Построить кривую по точкам.



# Метрики качества классификации

## Precision-recall (PR) кривая

Задача: Необходимо выбрать 100 релевантных из 1 миллиона документов.

**Алгоритм 1** возвращает 100 документов, 90 из которых релевантны. Таким образом,

$$TPR = \frac{TP}{TP + FN} = \frac{90}{90 + 10} = 0.9 \quad FPR = \frac{FP}{FP + TN} = \frac{10}{10 + 999890} = 0.00001$$

**Алгоритм 2** возвращает 2000 документов, 90 из которых релевантны.

$$TPR = \frac{TP}{TP + FN} = \frac{90}{90 + 10} = 0.9 \quad FPR = \frac{FP}{FP + TN} = \frac{1910}{1910 + 997990} = 0.00191$$

Если рассматривать полноту и точность :

**Алгоритм 1**

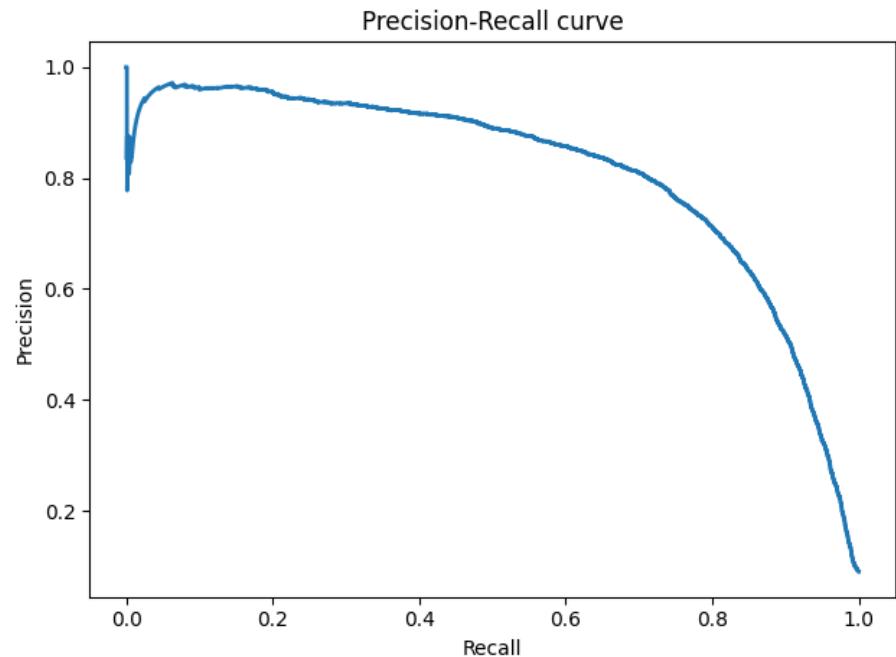
$$precision = \frac{TP}{TP + FP} = 90 / (90 + 10) = 0.9$$

$$recall = \frac{TP}{TP + FN} = 90 / (90 + 10) = 0.9$$

**Алгоритм 2**

$$precision = \frac{TP}{TP + FP} = \frac{90}{90 + 1910} = 0.045$$

$$recall = \frac{TP}{TP + FN} = \frac{90}{90 + 10} = 0.9$$



**Спасибо за внимание**

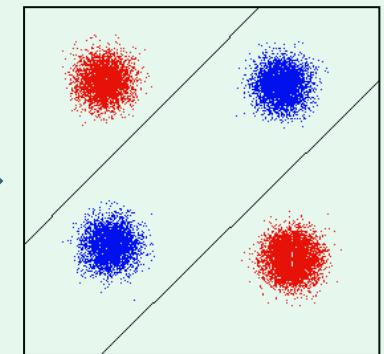
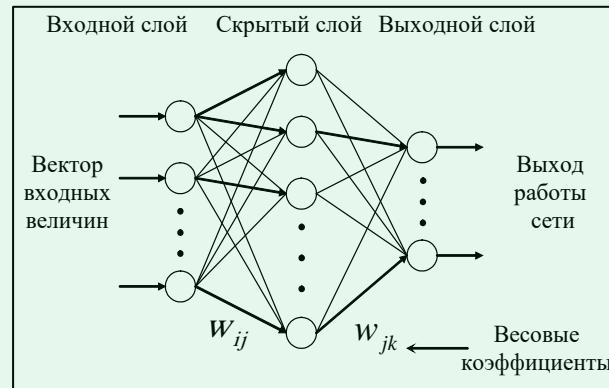
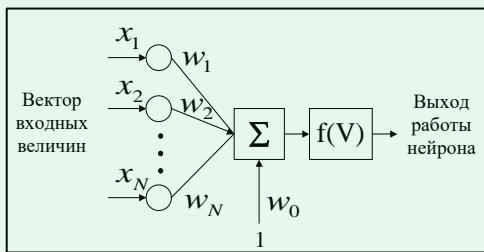
# **Методы машинного обучения**

*Лекция 7*

**Классификация -  
Нейросетевой подход**

# Предпосылки

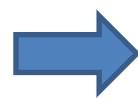
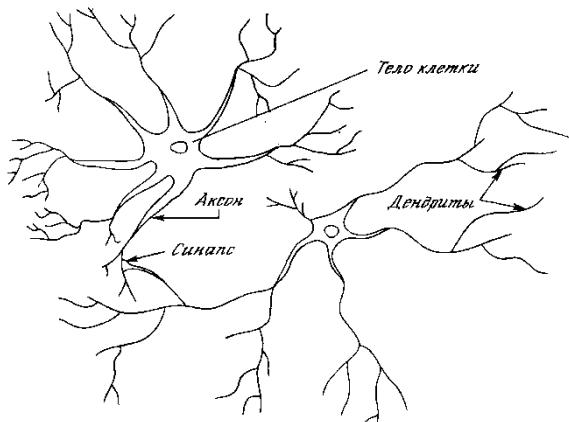
- Логистическая регрессия
- Линейный дискриминантный анализ  
(линейный дискриминант Фишера)
- Метод градиентного спуска



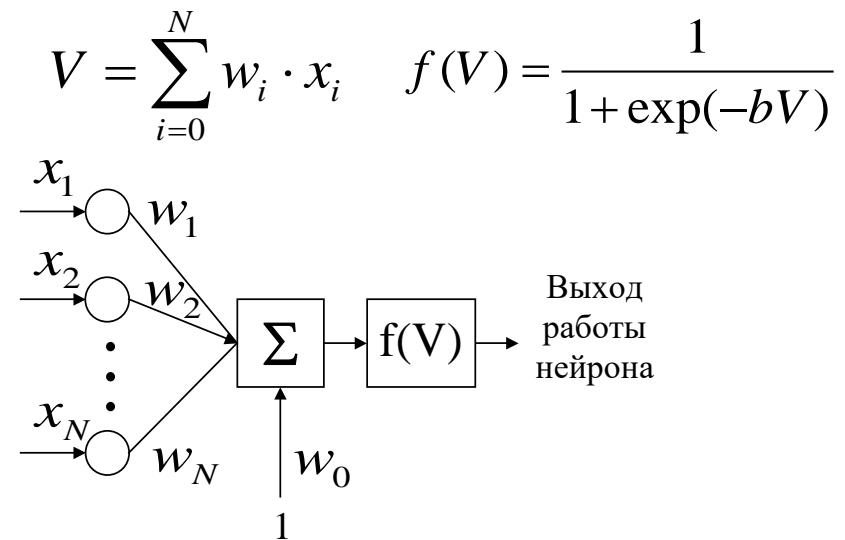
# Логистическая регрессия и математическая модель нейрона

Логистическую регрессию можно представить в виде однослойной нейронной сети с сигмоидальной функцией активации, веса которой есть коэффициенты логистической регрессии.

Структурная схема нейрона:



Вектор входных величин

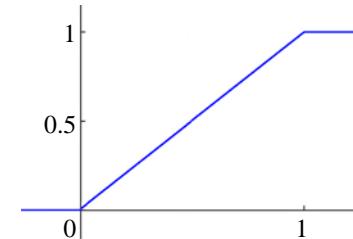


Первой попыткой математически описать процесс функционирования нейрона следует считать работу Мак-Каллока и Питтса, написанную ими еще в 1943 году. Отдельный нейрон состоит из “тела” (по аналогии с биологическим нейроном), суммирующего сигналы, поступающие с синапсов. При этом каждый синапс может передавать либо возбуждающий, либо тормозящий сигнал. В зависимости от соотношения возбуждающих и тормозящих сигналов нейрон либо возбуждается и передает дальше возбуждающий сигнал, либо тормозится и передает тормозящий сигнал.

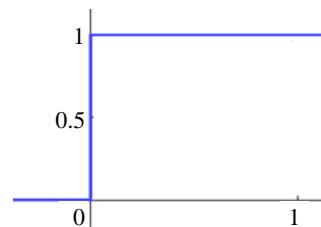
Следующий шаг в описании процесса функционирования нейронных сетей внес Розенблatt, описавший функционирование персептрана (1957). Персептрон Розенблатта представляет полноценную математическую модель отдельного нейрона.

# Функции активации

- Линейная:  $f(V) = V$



- Линейная с насыщением: 
$$f(V) = \begin{cases} 0, & V \leq 0 \\ 1, & V \geq 1 \\ V, & 0 < V < 1 \end{cases}$$



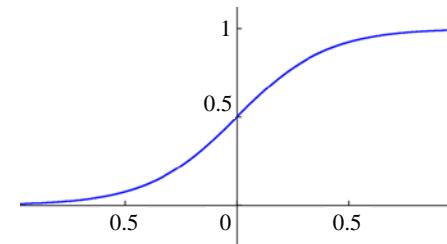
- Ступенчатая (пороговая):

$$f(V) = \begin{cases} 1, & V \geq 0 \\ 0, & V < 0 \end{cases}$$

- Многопороговая

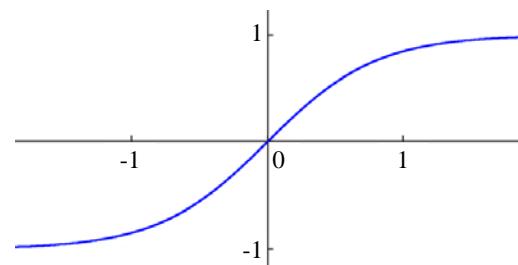
- Сигмоидная (логистическая):

$$f(V) = \frac{1}{1 + \exp(-bV)}$$



- Гиперболический тангенс :

$$f(V) = \tanh(V) \equiv \frac{\exp(bV) - 1}{\exp(bV) + 1}$$



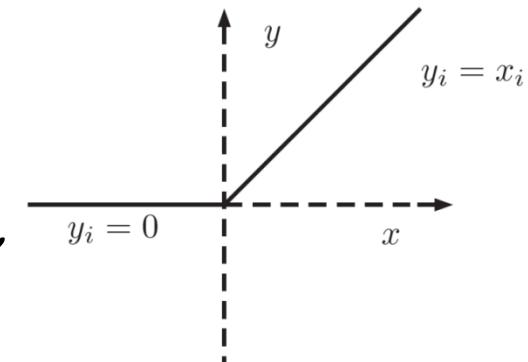
# Семейство функций активации ReLU

- **ReLU (Rectified linear unit)**

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

✓ Отсутствие эффекта насыщения;

✓ «*Dying ReLU Problem*» (проблема «умирающего» ReLU), до 40% ReLU нейронов никогда не активируются.

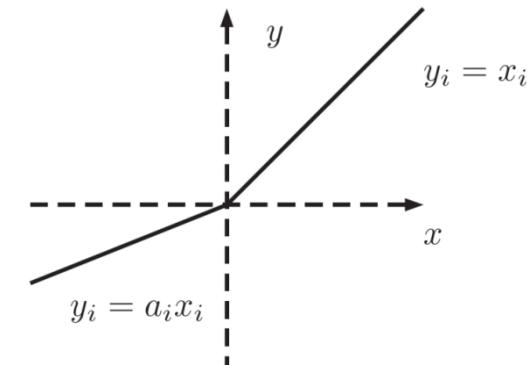


- **Leaky ReLU** - ReLU с «утечкой»

для  $x < 0$  имеет небольшое отрицательное значение, определяемое малым угловым коэффициентом:

$$f(x) = \begin{cases} x & x \geq 0 \\ \alpha x & x < 0 \end{cases}$$

где  $\alpha$  – малая константа порядка 0,01.

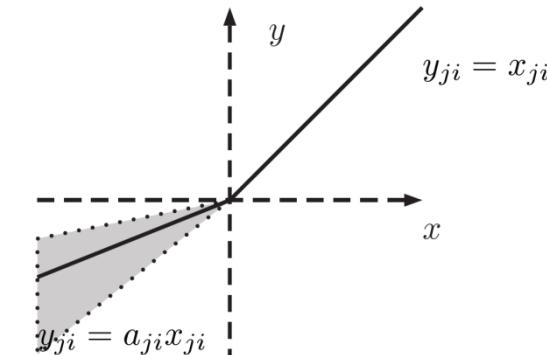


- **Parametric ReLU**

(Parameteric rectified linear unit, PReLU)

- **Randomized ReLU**

(Randomized leaky rectified linear unit)



# Искусственная нейронная сеть (Мак-Каллок и Питтс)

Мак-Каллок и Питтс исходили из предположений о том, что нервная система биологических существ состоит из большого количества нейронов, каждый из которых имеет несколько входных и один выходной сигнал. Описывая процесс функционирования нейронной сети, они приняли следующие допущения (1943 год):

- активность нейрона удовлетворяет принципу “все или ничего”;
- возбуждению нейрона предшествует период накопления возбуждений фиксированного количества его синапсов, причем это число не зависит от предыдущей активности и расположения синапсов в нейроне;
- единственным запаздыванием в нервной системе, имеющим значение, является синаптическая задержка;
- активность какого-либо тормозящего синапса исключает возбуждение нейрона в рассматриваемый момент времени;
- с течением времени структура нейронной сети не меняется.

Поскольку нейрон имеет как входные, так и выходные сигналы, то различные нейроны могут быть соединены между собой, образуя нейронную сеть.

Внутри биологической клетки сигнал распространяется гораздо медленнее, чем в электронных схемах. Однако, биологическая нейросеть в совокупности оказывается высокоэффективной в решении множества сложных задач, таких как распознавание образов (зрение, речь и т.д.). Можно предположить, что причина способностей кроется в высокой организации связей и параллелизме.

# Искусственная нейронная сеть

## Artificial neural network (ANN)

Математическая модель, а также ее программные или аппаратные реализации, построенная в некотором смысле по образу и подобию сетей нервных клеток живого организма.

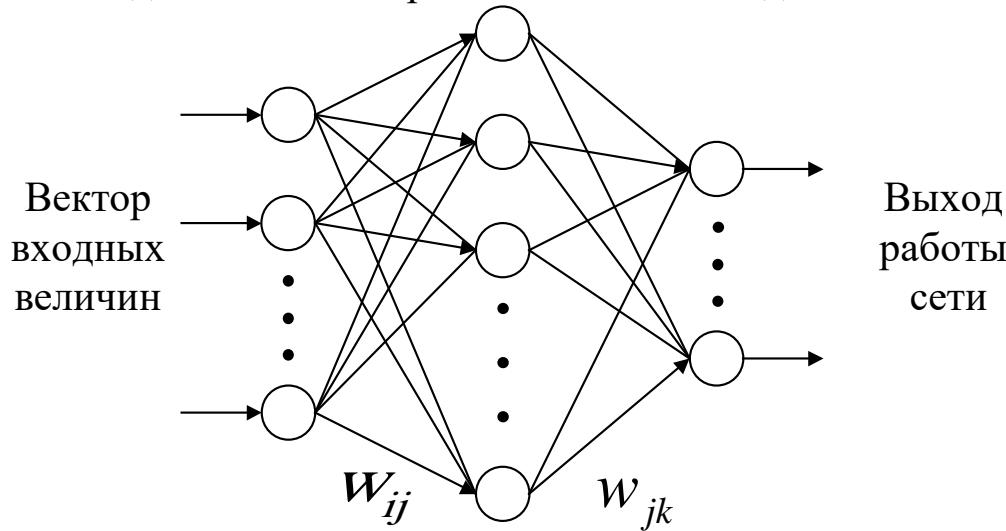
Под искусственной нейронной сетью в дальнейшем будем понимать сеть искусственных нейронов, соединенных между собой. Здесь предполагается, что нейроны могут соединяться между собой произвольным образом и образовывать таким образом разнообразные нейронные структуры.

- С точки зрения машинного обучения, нейронная сеть представляет собой частный случай методов распознавания образов, дискриминантного анализа;
- С точки зрения математики, обучение нейронных сетей — это многопараметрическая задача нелинейной оптимизации;
- С точки зрения кибернетики, нейронная сеть используется в задачах адаптивного управления и как алгоритмы для робототехники;
- С точки зрения развития вычислительной техники и программирования, нейронная сеть — способ решения проблемы эффективного параллелизма;
- С точки зрения искусственного интеллекта, ИНС является основой философского течения коннекционизма и основным направлением в структурном подходе по изучению возможности построения (моделирования) естественного интеллекта с помощью компьютерных алгоритмов.

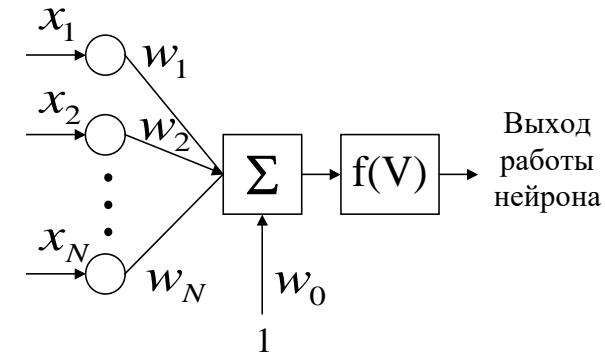
# Искусственная нейронная сеть

## Многослойный персептрон

Входной слой    Скрытый слой    Выходной слой



Структурная схема нейрона:



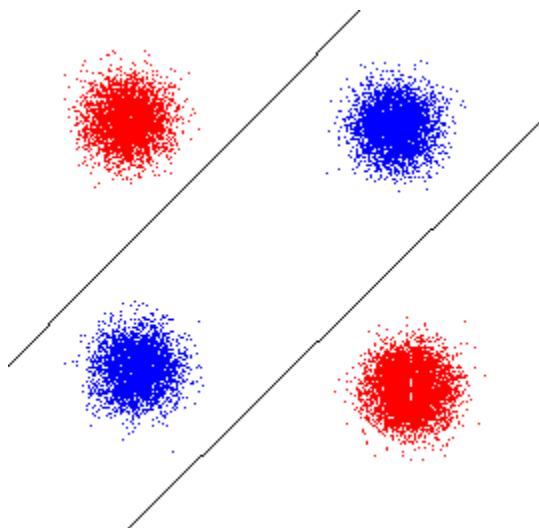
$$V = \sum_{i=0}^N w_i \cdot x_i \quad f(V) = \frac{1}{1 + \exp(-bV)}$$

Функционирование:  $Y_k(x) = Y_k(x_1, \dots, x_l) = f\left(\sum_{j=0}^m w_{jk} f\left(\sum_{i=0}^n w_{ij} x_i\right)\right)$

Сигналы проходят от входного слоя нейронов через скрытые слои к выходным элементам. В процессе функционирования каждый узел многослойной сети проектирует свой входной вектор на вектор весов посредством скалярного произведения (т.е. вычисляет взвешенную сумму входного вектора). После этого результаты проекции подвергаются нелинейным преобразованиям. Их цель - усилить те характеристики, за которые отвечает соответствующий узел.

# Многослойный персепtron – число слоев

Марвин Минский и Сеймур Паперт в своей работе "Персептроны" доказали, что простейшие однослойные нейронные сети, состоящие из входного и выходного слоев (известные, как линейный персептрон), способны решать только линейно разделимые задачи и обеспечивают универсальную линейную аппроксимацию. Это ограничение преодолимо путем добавления скрытых слоев и использования многослойных нейронных сетей.



В общем виде можно сказать, что при решении задач классификации в сети с одним скрытым слоем, входной вектор преобразуется в некоторое новое пространство, возможно с другой размерностью, а затем поверхности, соответствующие нейронам выходного слоя, разделяют его на классы. Таким образом, сеть распознает не только характеристики исходных данных, но и "характеристики характеристик", сформированные скрытым слоем.

Нейронные сети с числом скрытых слоев большим единицы называются глубокими (deep neural network). Они могут содержать меньшее число нейронов в каждом слое, чем сети с одним скрытым слоем, реализующие то же самое отображение. Однако строгой методики сопоставления таких сетей пока не существует.

# Многослойный персепtron

## число нейронов в каждом слое

**Количество нейронов входного слоя** напрямую зависит от размерности исходного пространства входных данных (пространства признаков) и от методов их кодирования.

### Количество нейронов скрытого слоя

Проблема выбора количества скрытых элементов многослойного персептрона заключается в том, что с одной стороны, число скрытых элементов должно быть достаточным для решения поставленной задачи, а с другой не должно быть слишком большим, чтобы обеспечить необходимую обобщающую способность сети и избежать переобучения. То есть, нельзя просто выбрать теоретический максимум числа весовых коэффициентов, так как в этом случае сеть научится иметь дело только с теми данными, которые предъявились в процессе тренировки, и, поэтому обобщающая способность сети будет слабой.

**Количество нейронов выходного слоя** зависит от решаемой задачи и, также как для входного слоя, от способов кодирования

# Многослойный персепtron

## число нейронов скрытого слоя

Из теоремы Колмагорова-Арнольда (1957) о представлении непрерывных функций нескольких переменных в виде суперпозиций непрерывных функций одного переменного и сложения следует теорема Хехт-Нильсена доказывающая представимость функции многих переменных достаточно общего вида с помощью двухслойной нейронной сети с прямыми полными связями с  $i$  нейронами входного слоя и  $(2i+1)$  нейронами скрытого слоя с заранее известными ограниченными функциями активации (например, сигмоидными). Теорема, таким образом, в неконструктивной форме доказывает решаемость задачи представления функции произвольного вида на нейронной сети и указывает минимальные числа нейронов, необходимых для решения.

**Следствие 1.** Неизвестными остаются следующие характеристики функций активации нейронов:

- Ограничение области значений (координаты асимптот) сигмоидных функций активации нейронов скрытого слоя;
- Наклон сигмоид;
- Вид функций активации нейронов выходного слоя.

**Следствие 2.**

Для любого множества пар входных-выходных векторов произвольной размерности  $\{(X_k, Y_k), k = 1\dots N\}$  существует однородная двухслойная нейронная сеть с последовательными связями, с сигмоидными передаточными функциями и с конечным числом нейронов, которая для каждого входного вектора  $X_k$  формирует соответствующий ему выходной вектор  $Y_k$ .

# Многослойный персепtron

## число нейронов скрытого слоя - оценки

- Сеть с одним скрытым слоем, содержащим  $h$  нейронов со ступенчатой функцией активации, способна осуществить произвольную классификацию  $h^*d$  точек  $d$ -мерного пространства (т.е. классифицировать  $h^*d$  примеров). Более того, одного скрытого слоя нейронов с сигмоидной функцией активации достаточно для аппроксимации любой границы между классами или некоторой функции со сколь угодно высокой точностью
- Хайкин, используя результаты из работ Баума и Хесслера, приводит рекомендации относительно размеров набора учебных данных относительно количества весовых коэффициентов с учетом доли ошибок, допустимых в ходе тестирования, которые можно выразить следующим неравенством:

$$p \geq \frac{w}{\varepsilon}$$

где  $\varepsilon$  - доля ошибок, допустимых в ходе тестирования. Так при допустимости 10% ошибок число учебных образцов должно быть в 10 раз больше числа имеющихся в сети весовых коэффициентов.

- Одной из возможных оценок может служить принцип совместной оптимизации эмпирической ошибки и сложности модели, согласно которому ошибка предсказаний сети на новых данных определяется общей длиной описания данных с помощью модели вместе с описанием самой модели. Данный принцип может быть записан в виде:

$$\min \{ \text{описание ошибки} + \text{описание модели} \}.$$

Минимум ошибки (знак равенства) достигается при оптимальном числе весов в сети:  $w \sim \sqrt{p \cdot i}$

Что соответствует числу нейронов в скрытом слое равному по порядку величине:  $h = \frac{w}{i} \sim \sqrt{\frac{p}{i}}$

где  $i, h$  – число нейронов входного и скрытого слоев,  $w$  – количество весов,  $p$  – количество примеров обучающей выборки.

# Многослойный персепtron

## число нейронов скрытого слоя - оценки

Для оценки числа нейронов в скрытом слое однородной нейронной сети можно воспользоваться формулой для оценки необходимого числа синаптических весов  $N_w$  в многослойной сети с сигмоидальными функциями активации:

$$\frac{N_y N_p}{1 + \log_2(N_p)} \leq N_w \leq N_y \left( \frac{N_p}{N_x} + 1 \right) (N_x + N_y + 1) + N_y$$

где  $N_y$  - размерность выходного сигнала,  $N_p$  - число элементов обучающей выборки,  $N_x$  - размерность входного сигнала.

Оценив необходимое число весов, можно рассчитать число нейронов в скрытых слоях. Число нейронов в сети с одним скрытым слоем составит:

$$N = \frac{N_w}{N_x + N_y}$$

Пример: По условию задачи размерность входного вектора равна  $N_x = 21$ ; число нейронов в выходном слое соответствует числу классов  $N_y = 5$ , на которые предполагается разбить выборку данных, состоящую из 6000 объектов. Неизвестным является число нейронов в промежуточном слое  $N$ .

$$\frac{5 \cdot 6000}{1 + 12,55} \leq N_w \leq 5 \cdot \left( \frac{6000}{21} + 1 \right) \cdot (21 + 5 + 1) + 5$$

Округляя до целых, получим  $2222 \leq N_w \leq 7746$ . Тогда число нейронов в скрытом слое составит:  $N_{min} = \frac{2222}{26} \approx 85$  и  $N_{max} = \frac{7746}{26} \approx 297$  т.е.  $85 \leq N \leq 297$ .

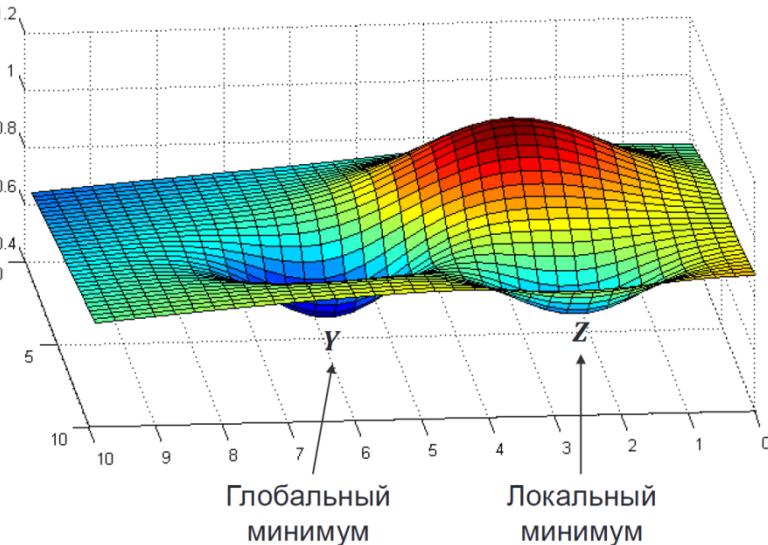
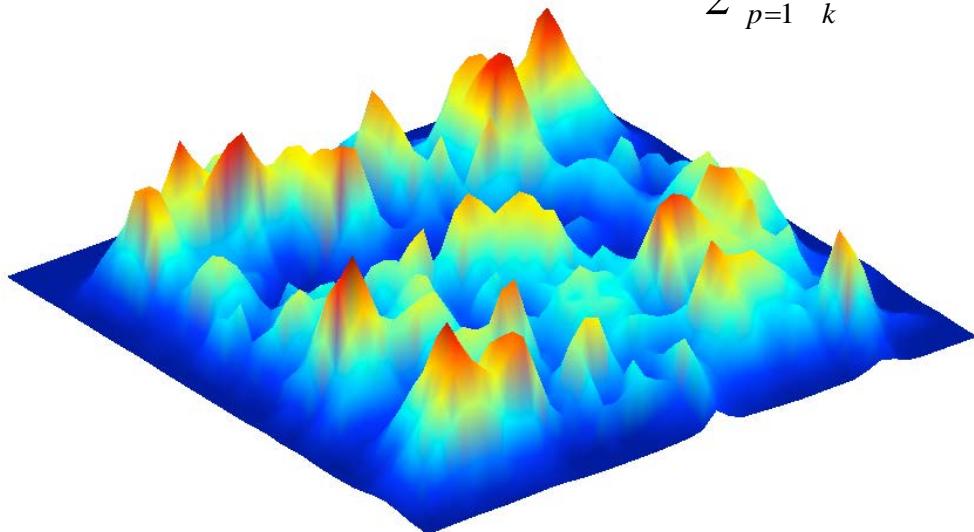
# Многослойный персепtron (методы обучения)

Обучение нейронной сети - интерактивный процесс корректировки синаптических весов и порогов. В процессе обучения нейронная сеть получает и обобщает знания об окружающей среде и тех данных, с которыми ей придется оперировать. Существуют два концептуальных подхода к обучению нейронных сетей: обучение с учителем и обучение без учителя.

Обучение персептрана:

- Обучение Хебба (без учителя);
- Стохастические методы обучения (“имитация отжига”);
- Обратное распространение ошибки (Backpropagation).

Ошибка сети при обучении:  $E = \frac{1}{2} \sum_{p=1}^P \sum_k (u_p^k - y_p^k)^2$



# Обучение Хебба

Канадский нейропсихолог Дональд Хебб сформулировал свой «нейрофизиологический постулат» в 1949 в самом значимом своем труде – «*The Organization of Behavior: A NEUROPSYCHOLOGICAL THEORY*».

Звучит следующим образом:

«Если аксон клетки А находится достаточно близко, чтобы активировать клетку В, и неоднократно или постоянно принимает участие в ее активации, то наблюдается некоторый процесс роста связи или метаболических изменений в одной или обеих клетках, ведущий к увеличению эффективности А, как одной из клеток активирующих В».

Предложена модель обучения без учителя, в которой синаптическая сила (вес) возрастает, если активированы оба нейрона, источник и приемник. Таким образом, часто используемые пути в сети усиливаются и феномен привычки и обучения через повторение получает объяснение.

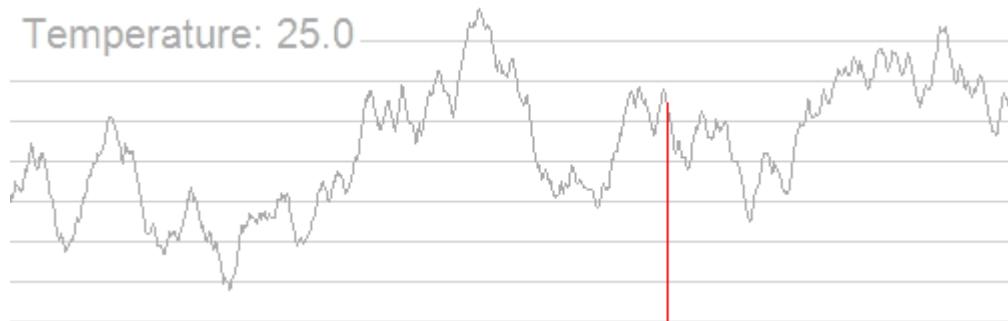
$$w_{ij}(n+1) = w_{ij}(n) + \eta * OUT_i * OUT_j$$

$w_{ij}(n)$  – значение веса от нейрона  $i$  к нейрону  $j$  до подстройки,  $w_{ij}(n+1)$  – значение веса от нейрона  $i$  к нейрону  $j$  после подстройки,  $\eta$  – коэффициент скорости обучения,  $OUT_i$  – выход нейрона  $i$  и вход нейрона  $j$ ,  $OUT_j$  – выход нейрона  $j$ .

# Стохастические методы обучения

## «имитация отжига»

Выполняются псевдослучайные изменения величин весов, сохраняя те изменения, которые ведут к улучшениям.



Для обучения сети может быть использована следующая процедура:

1. Выбрать значения весов случайным образом. Предъявить множество входов и вычислить получающиеся выходы.
2. Сравнить полученные значения с желаемыми выходами и вычислить величину разности между ними (сумма квадратов разностей). Целью обучения является минимизация этой разности (часто называют **целевой функцией**).
3. Выбрать случайный весовой коэффициент и подкорректировать его на небольшое случайное значение. Если коррекция помогает (уменьшает целевую функцию), то сохранить ее, в противном случае вернуться к первоначальному значению веса. Случайное значение корректировки постепенно уменьшается со временем.
4. Повторять шаг 3 до тех пор, пока сеть не будет обучена в достаточной степени.

Скорость уменьшения температуры должна быть обратно пропорциональна логарифму времени, чтобы была достигнута сходимость к глобальному минимуму.

$$T(t) = \frac{T_0}{\log(1 + t)}$$

$T(t)$  – искусственная температура как функция времени;  
 $T_0$  – начальная искусственная температура;  
 $t$  – искусственное время.

# Обратное распространение ошибки (Backpropagation)

Алгоритм обучения с учителем. Фактически является алгоритмом градиентного спуска, минимизирующим суммарную квадратичную ошибку:

$$E = \frac{1}{2} \sum_{k=1}^P \sum_i (d_k^i - y_k^i)^2$$
 где  $d_k^i$  - желаемый выход нейрона  $i$ ,  
 $y_k^i$  - текущий выход нейрона  $i$  последнего слоя.

Синаптические веса настраиваются в соответствии с формулой:

$$w_{ij}(t+1) = w_{ij}(t) + \Delta w_{ij}(t) \quad \text{изменение веса: } \Delta w_{ij} = -\varepsilon \frac{\partial E}{\partial w_{ij}} = -\varepsilon \delta_k^j x_k^i$$

где  $\varepsilon$  - длина шага в направлении, обратном к градиенту;

$\delta_k^j$  - вычисляется через аналогичные множители из последующего слоя, и ошибка передается в обратном направлении.

Для выходных элементов:  $\delta_k^j = -(d_k^j - y_k^j) f'(V_k^j) = -(d_k^j - y_k^j) \cdot y_k^j \cdot (1 - y_k^j)$

Для скрытых элементов:  $\delta_k^j = -f'(V_k^j) \sum_h w_{jh} \delta_h^h = out_k^j \cdot (1 - out_k^j) \cdot \sum_h w_{jh} \delta_h^h$

где индекс  $h$  пробегает номера всех нейронов, на которые воздействует  $j$ -й нейрон,  
 $out_k^j$  - выход нейрона  $j$ ,  $k$  - номер обучающего примера.

# Этапы Backpropagation

Весь алгоритм разбивается на следующие этапы:

1. подать на вход сети один из требуемых образов и определить значения выходов нейронов сети .
2. рассчитать  $\delta_k^j$  для выходного слоя сети по формуле

$$\delta_k^j = -(d_k^j - y_k^j) f'(V_k^j) = -(d_k^j - y_k^j) \cdot y_k^j \cdot (1 - y_k^j)$$

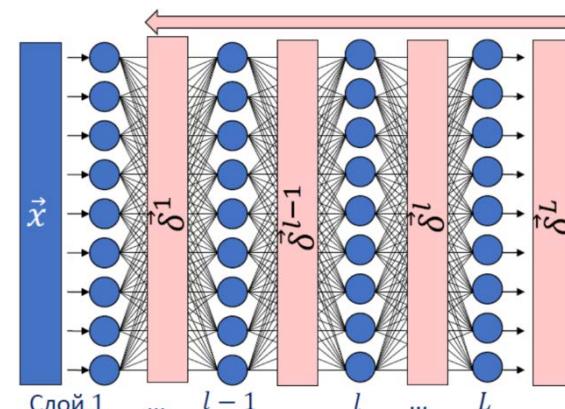
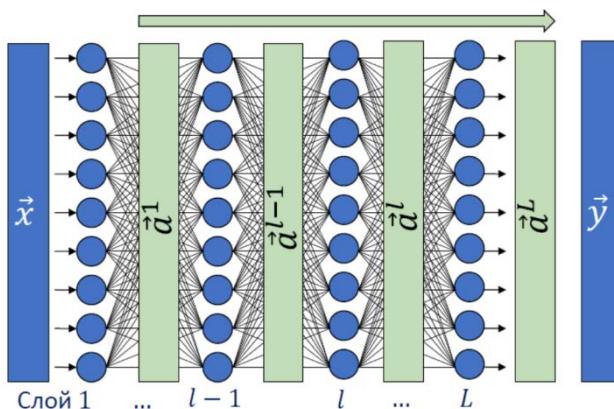
и рассчитать изменения весов  $\Delta w_{ij}$  выходного слоя.

3. Рассчитать по формуле

$$\delta_k^j = -f'(V_k^j) \sum_h w_{jh} \delta_k^h = out_k^j \cdot (1 - out_k^j) \cdot \sum_h w_{jh} \delta_k^h$$

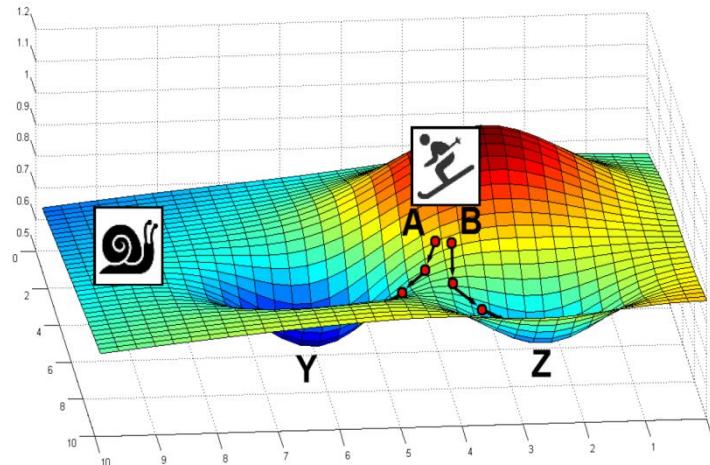
значения  $\delta_k^j$  и  $\Delta w_{ij}$  для остальных слоев нейросети, двигаясь от выходного слоя к входному.

4. Скорректировать все веса сети. Если ошибка существенна, то перейти на шаг 1.



# Модификации алгоритма Backpropagation

Существуют различные модификации алгоритма Backpropagation призванные ускорить процесс обучения, особенно в случаях, когда рельеф функции ошибки напоминает не яму, а длинный овраг, обеспечить способность преодолевать мелкие локальные минимумы ошибки, застревая лишь в относительно глубоких, значимых минимумах, а также избежать и некоторые другие его недостатки.



К таким модификациям можно отнести:

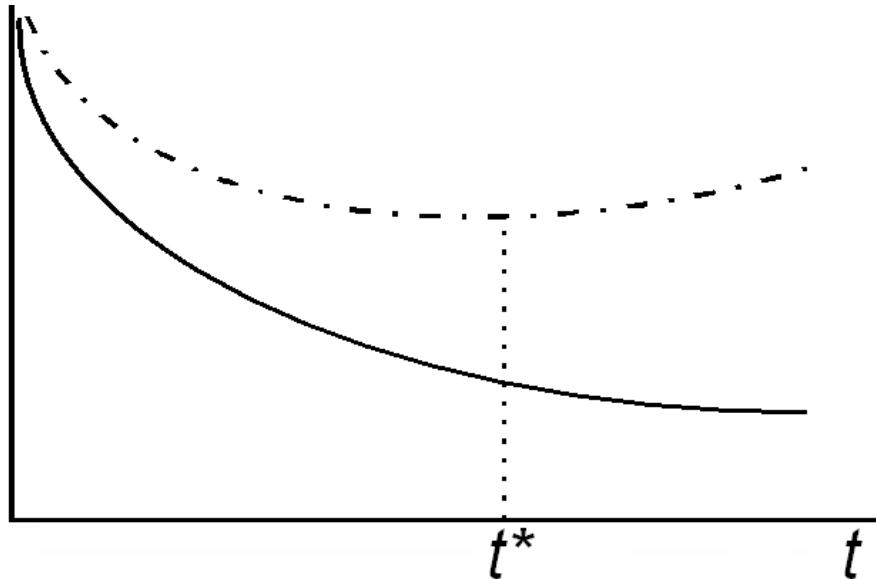
- использование момента  $\mu$ , учитывающего изменение весов на предыдущем шаге обучения:

$$\Delta w_{ij}(t) = -\varepsilon \delta_j x_i + \mu \Delta w_{ij}(t-1)$$

- Алгоритм Resilient Propagation (RPROP) (resilient - эластичный), предложенный М. Ридмиллером (M.Riedmiller) и Г. Брауном (H.Braun), стремится избежать замедления темпа обучения на плоских "равнинах" ландшафта функции ошибки, характерного для схем, где изменения весов пропорциональны величине градиента.

Существуют и другие алгоритмы, ускоряющие процесс обучения, и важно иметь возможность выбрать тот алгоритм, который бы показывал наилучшую сходимость для имеющейся выборки.

## Оценка полученной архитектуры



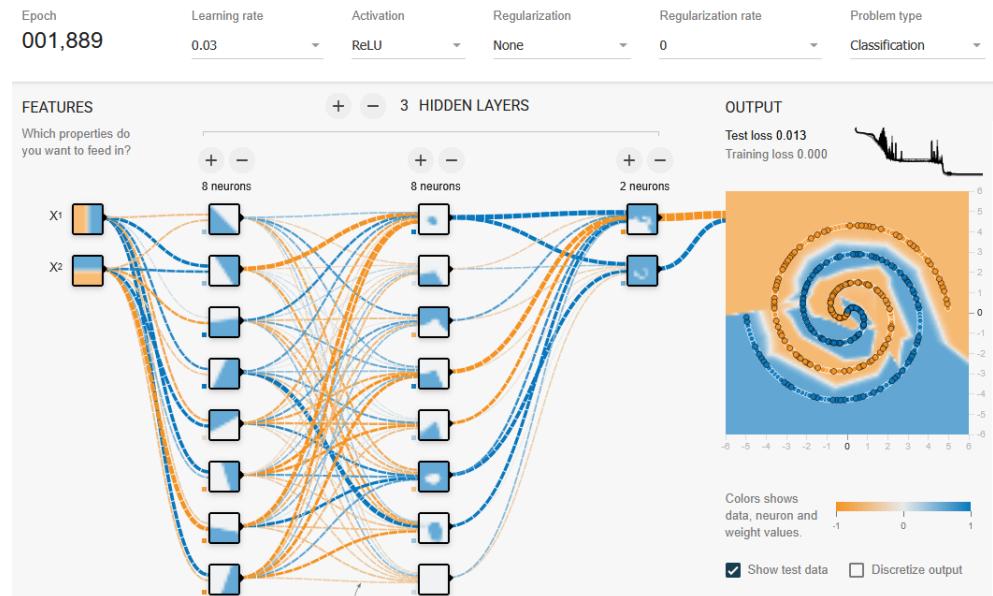
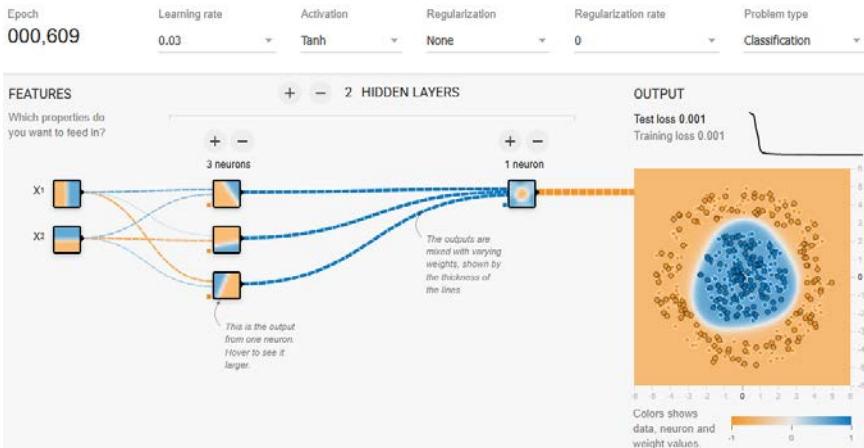
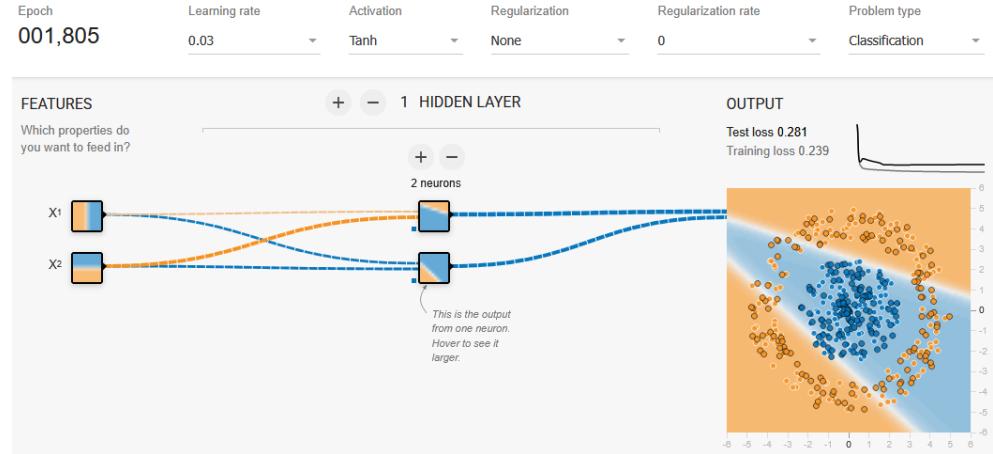
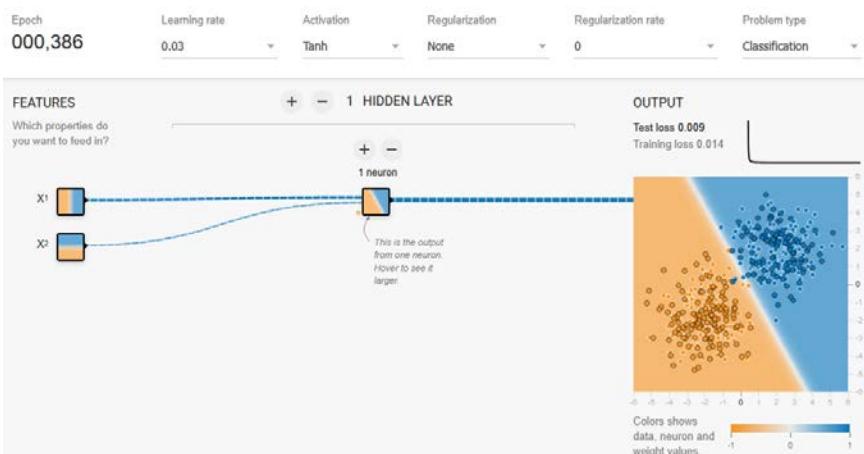
Показан момент остановки, соответствующий минимуму ошибки валидации (штрих-пунктирная кривая), при этом ошибка обучения (сплошная кривая) продолжает понижаться.

Критерий остановки обучения в момент времени  $t^*$ .

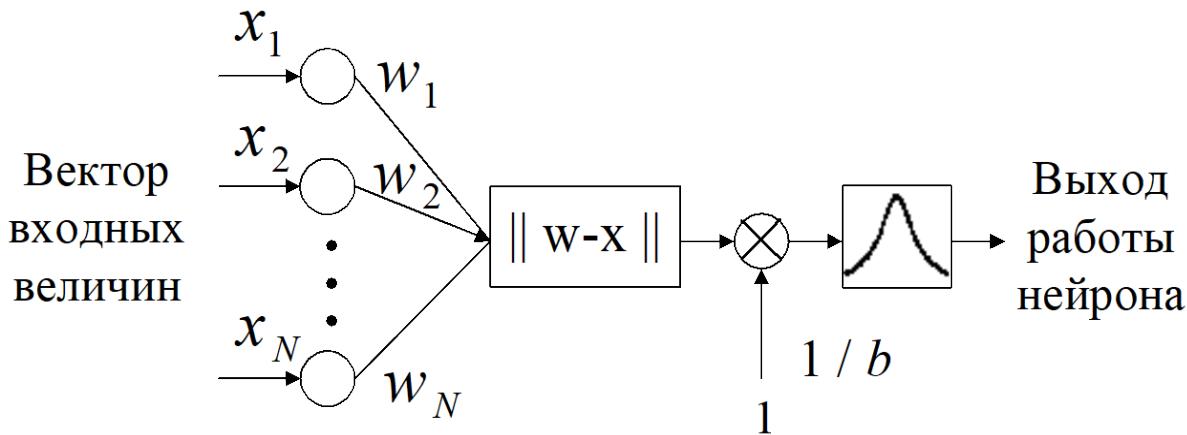
В упрощенном виде для оценки уже построенной нейросетевой модели могут быть сформулированы следующие критерии:

- если ошибка тренировки мала, а ошибка тестирования велика, значит, сеть содержит слишком много весовых коэффициентов;
- если и ошибка тренировки, и ошибка тестирования велики, значит весовых коэффициентов слишком мало;
- если все весовые коэффициенты очень большие, значит весовых коэффициентов слишком мало.

# <https://playground.tensorflow.org>



# RBF-нейрон

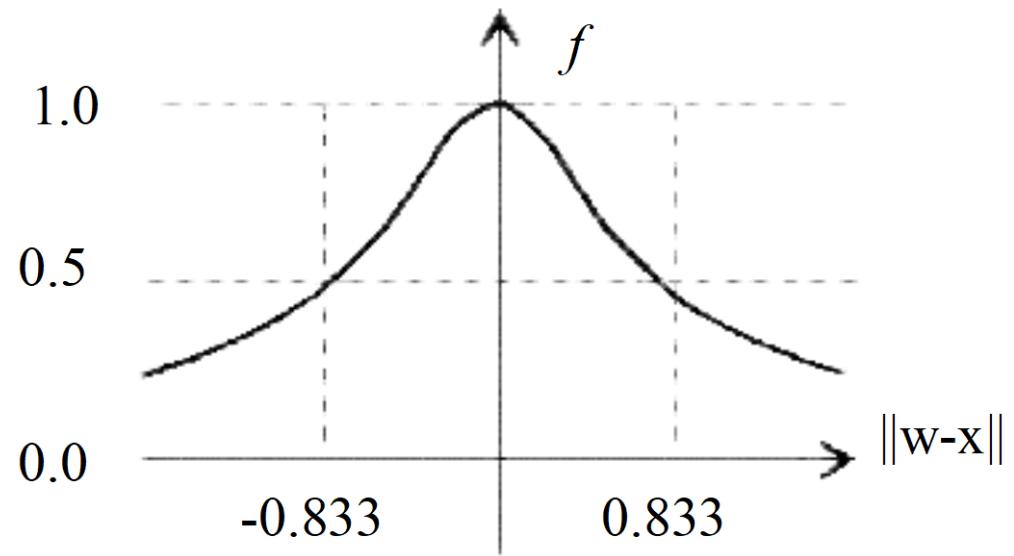


Потенциал нейрона - расстояние между векторами весовых коэффициентов и входных величин (евклидово расстояние)

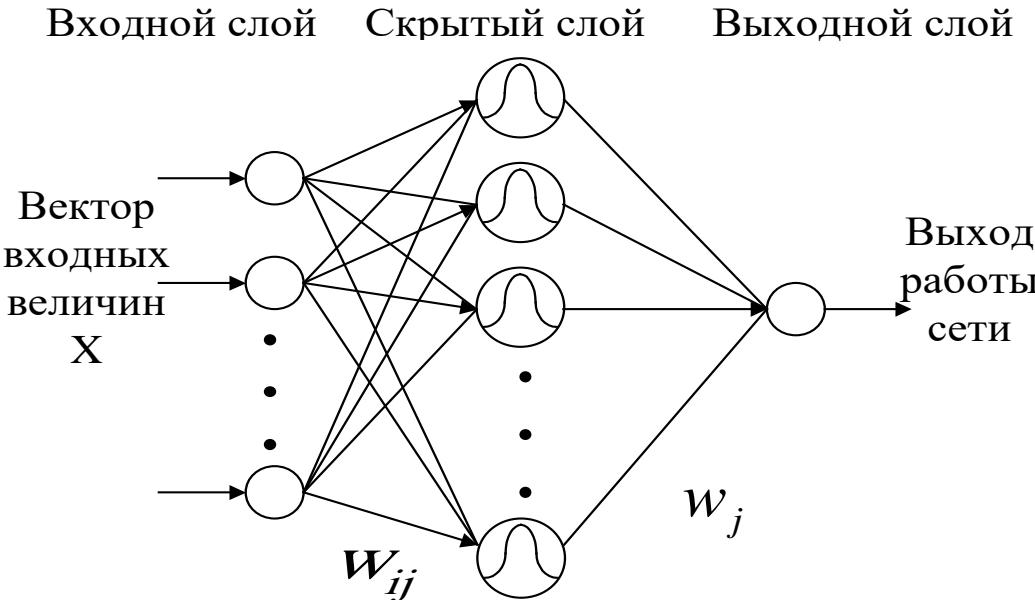
$$\|w - x\| = \sqrt{\sum_{i=1}^N (w_i - x_i)^2}$$

Функция активации:

$$f = e^{-\left(\frac{\|w-x\|}{b}\right)^2}$$



# RBF-сеть



Функционирование:  $u_{net} = \sum_{j=1}^n w_j \cdot a_j$

Обучение RBF-сети:

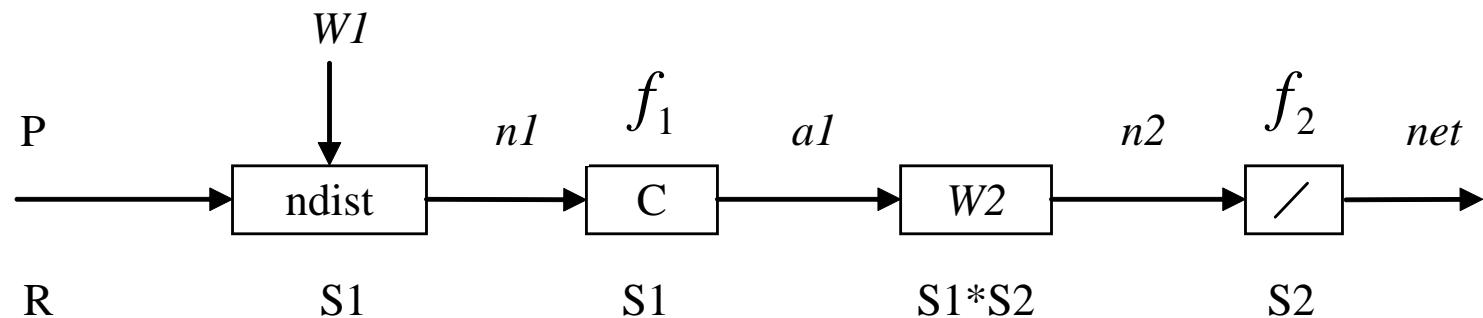
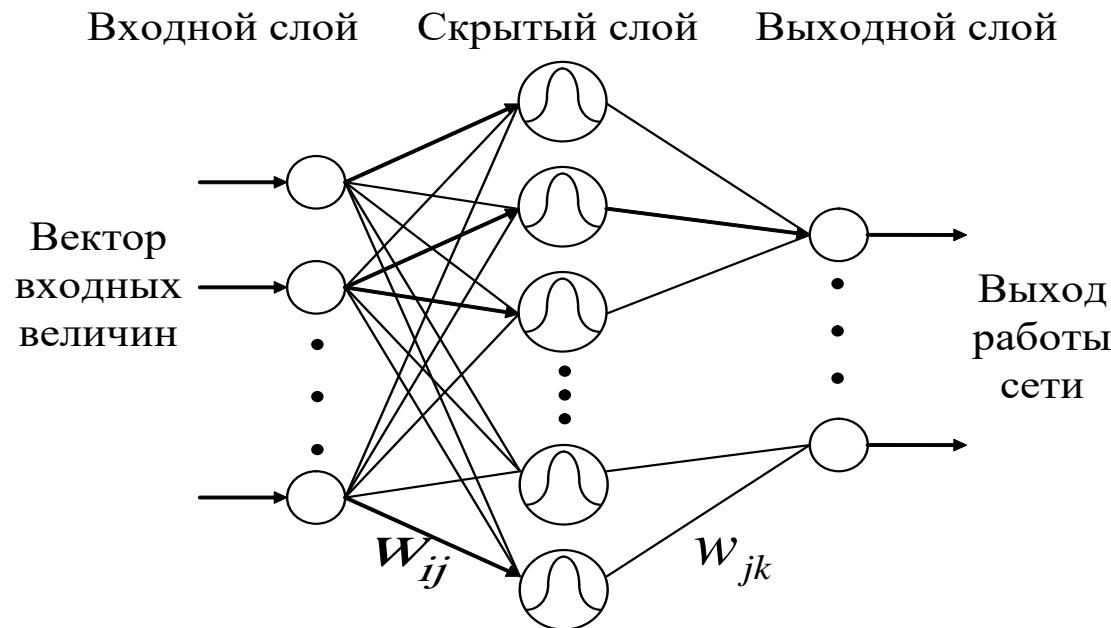
1. Формирование скрытого слоя

- Образцами обучающей выборки;
- С учетом имеющихся кластеров в данных;

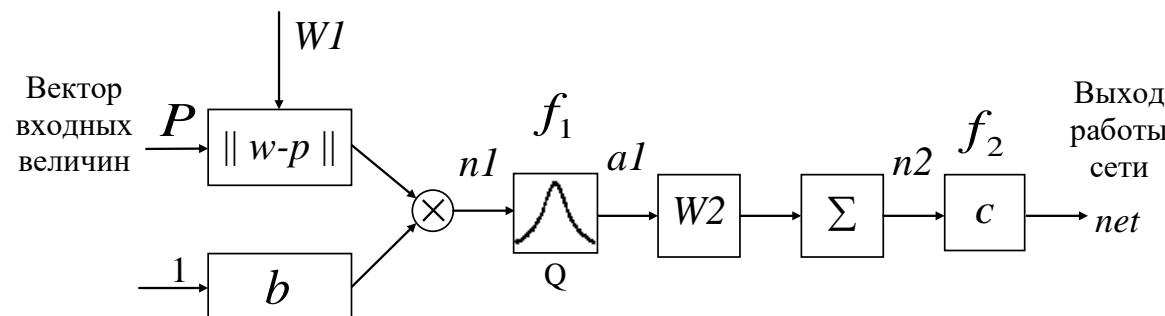
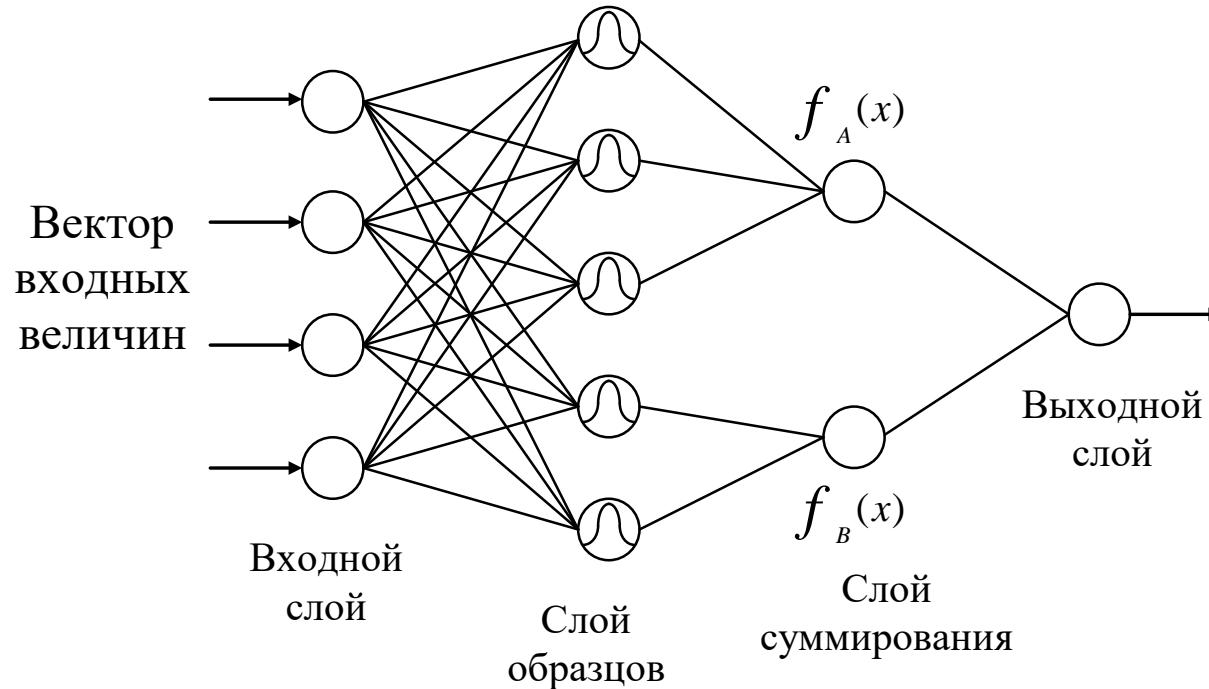
2. Обучение выходного линейного слоя

- Методом псевдообратных матриц:  $\bar{w} = (A^T A)^{-1} A^T \bar{u}$
- NLMS – Normalized Least Mean Squares:  $w_j(t) = w_j(t-1) + \Delta(t) \cdot e_{net}(t) \cdot \frac{a_j(t)}{a^T(t)a(t)}$

# Сеть LVQ (Learning Vector Quantization)



# Вероятностная сеть (Probabilistic Neural Network - PNN)



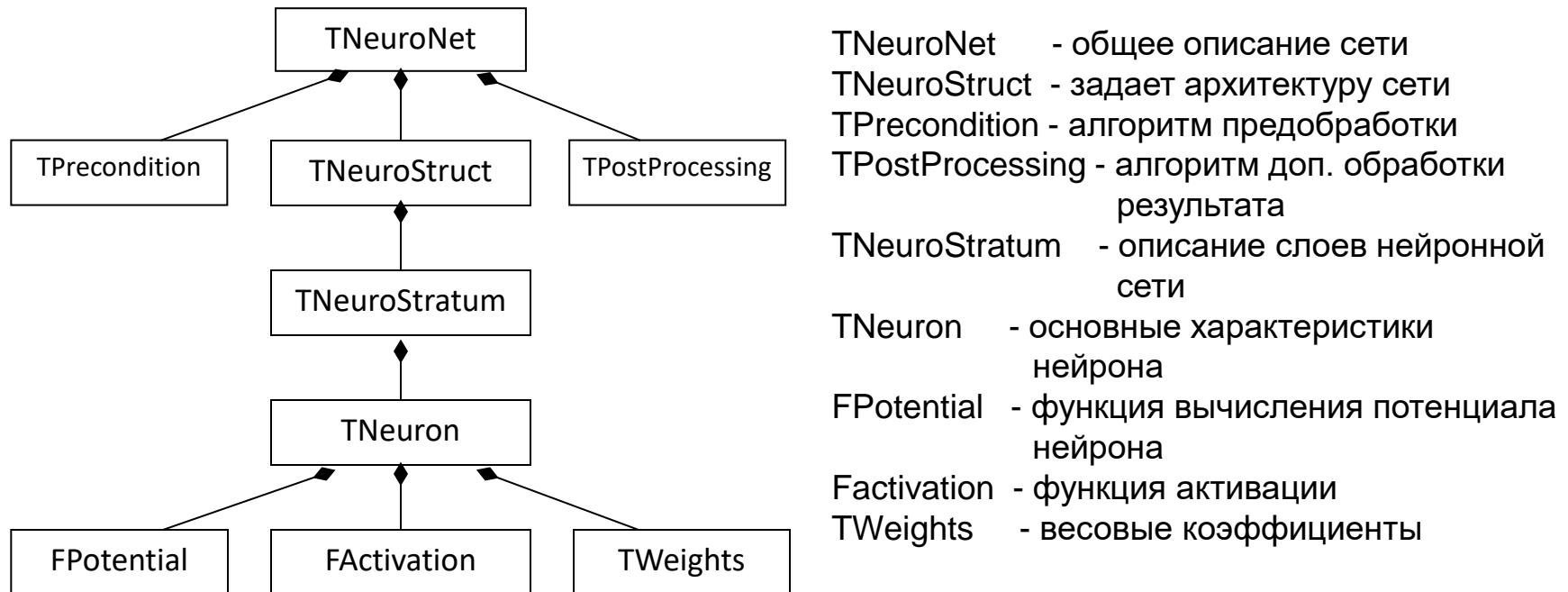
Для двух классов  $A$  и  $B$  в соответствии с данными правилами выбирается класс  $A$ , если  

$$h_A c_A f_A(x) > h_B c_B f_B(x)$$

где  $h$  обозначает априорную вероятность,  $c$  – цену ошибки классификации, а  $f(x)$  – функцию плотности вероятности.

# Иерархия классов описания нейронной сети

С точки зрения объектно-ориентированного подхода нейросеть представляет собой иерархию классов: сеть, слой, нейрон и т.д.



В рамках данной модели формирование нейросети заключается в поэтапном движении сверху вниз по иерархии классов. Таким образом, сначала создаются объекты классов определяющих наиболее общие характеристики сети. Затем определяются все более мелкие, характерные фрагменты строящейся модели, т.е. построение происходит от общего к частному.

**Спасибо за внимание**

# **Методы машинного обучения**

*Лекция 8*

**Деревья решений**

# Деревья решений

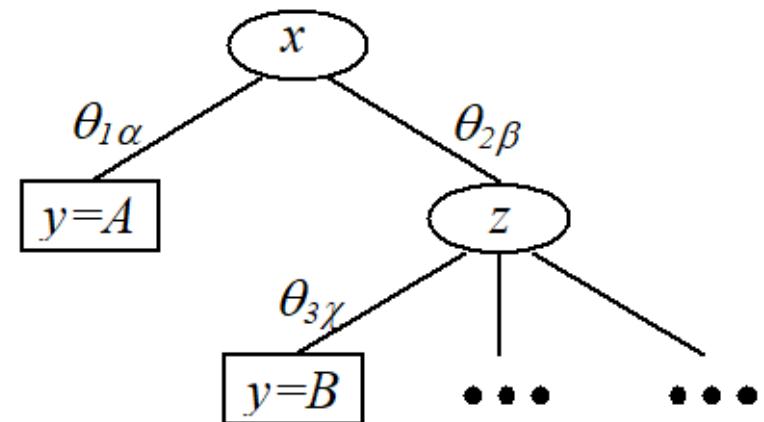
Деревья решений относятся к методам поиска логических закономерностей в данных, а также являются основным подходом, применимым в теории принятия решений.

Они позволяют осуществлять решение целого класса задач классификации и регрессии в виде многошагового процесса принятия решений и используют особенности древовидных классификаторов, связанных с учетом локальных свойств классифицируемых объектов на каждом уровне и в каждом узле дерева, что позволяет реализовать как прямую, так и обратную цепочку рассуждений.

Основными достоинствами деревьев решений является:

- простота и наглядность описания процесса поиска решения;
- представление правил в виде продукции «если... то...».

*Если (условие 1)  $\wedge$  (условие 2)  $\wedge \dots \wedge$  (условие N) то (значение вершины вывода)*



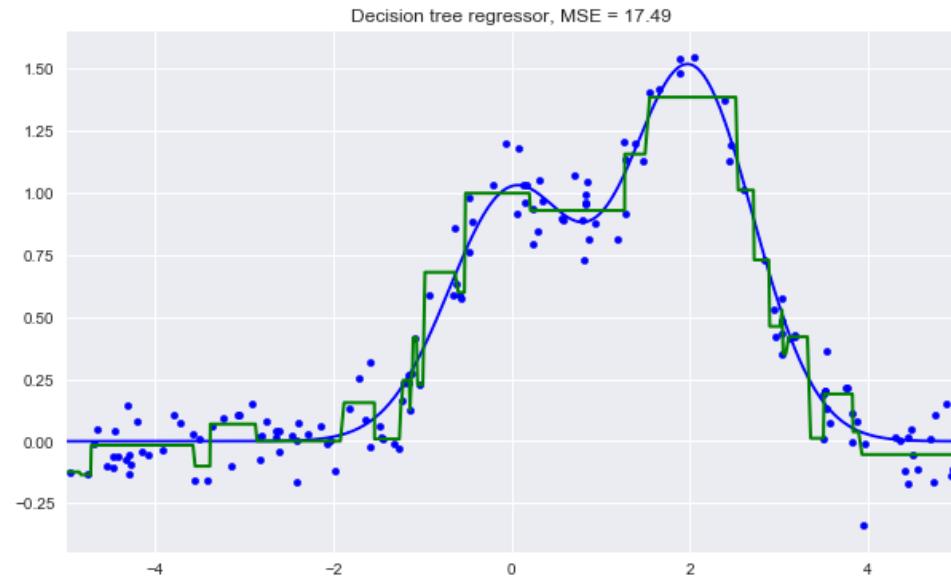
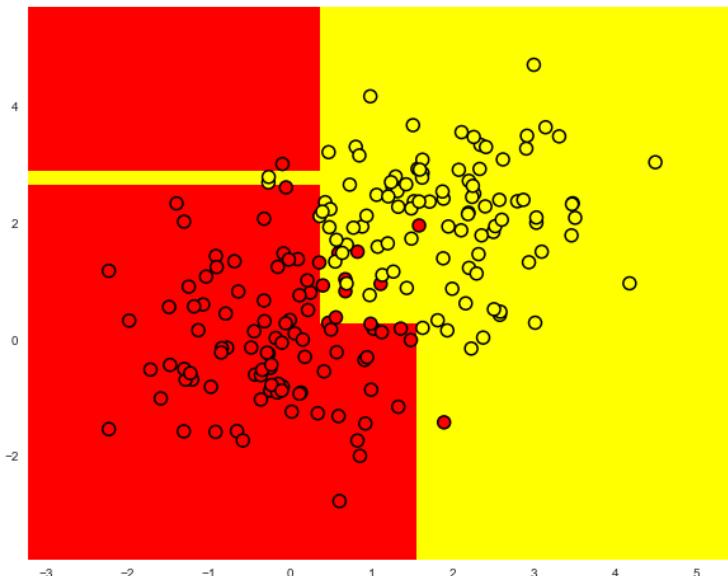
$$\theta_1(x, \alpha) \rightarrow (y = A)$$

$$\theta_2(x, \beta) \cap \theta_3(z, \chi) \rightarrow (y = B)$$

# Задачи

Сфера применения — поддержка процессов принятия управленческих решений, используемая в статистике, анализе данных и машинном обучении. Задачами, решаемыми с помощью данного аппарата, являются:

- **Классификация** — отнесение объектов к одному из заранее известных классов. Целевая переменная должна иметь дискретные значения.
- **Регрессия (численное предсказание)** — предсказание числового значения независимой переменной для заданного входного вектора.
- **Описание объектов** — набор правил в дереве решений позволяет компактно описывать объекты. Поэтому вместо сложных структур, описывающих объекты, можно хранить деревья решений.



# Основные понятия

Название	Описание
<b>Объект</b>	Пример, шаблон, наблюдение
<b>Атрибут</b>	Признак, независимая переменная, свойство
<b>Целевая переменная</b>	Зависимая переменная, метка класса
<b>Узел</b>	Внутренний узел дерева, узел проверки. В узлах находятся решающие правила и производится проверка соответствия примеров этому правилу по какому-либо атрибуту обучающего множества.
<b>Корневой узел</b>	Начальный узел дерева решений
<b>Лист</b>	Конечный узел дерева, узел решения, терминальный узел - определяет решение для каждого попавшего в него примера.
<b>Решающее правило</b>	Условие в узле, проверка

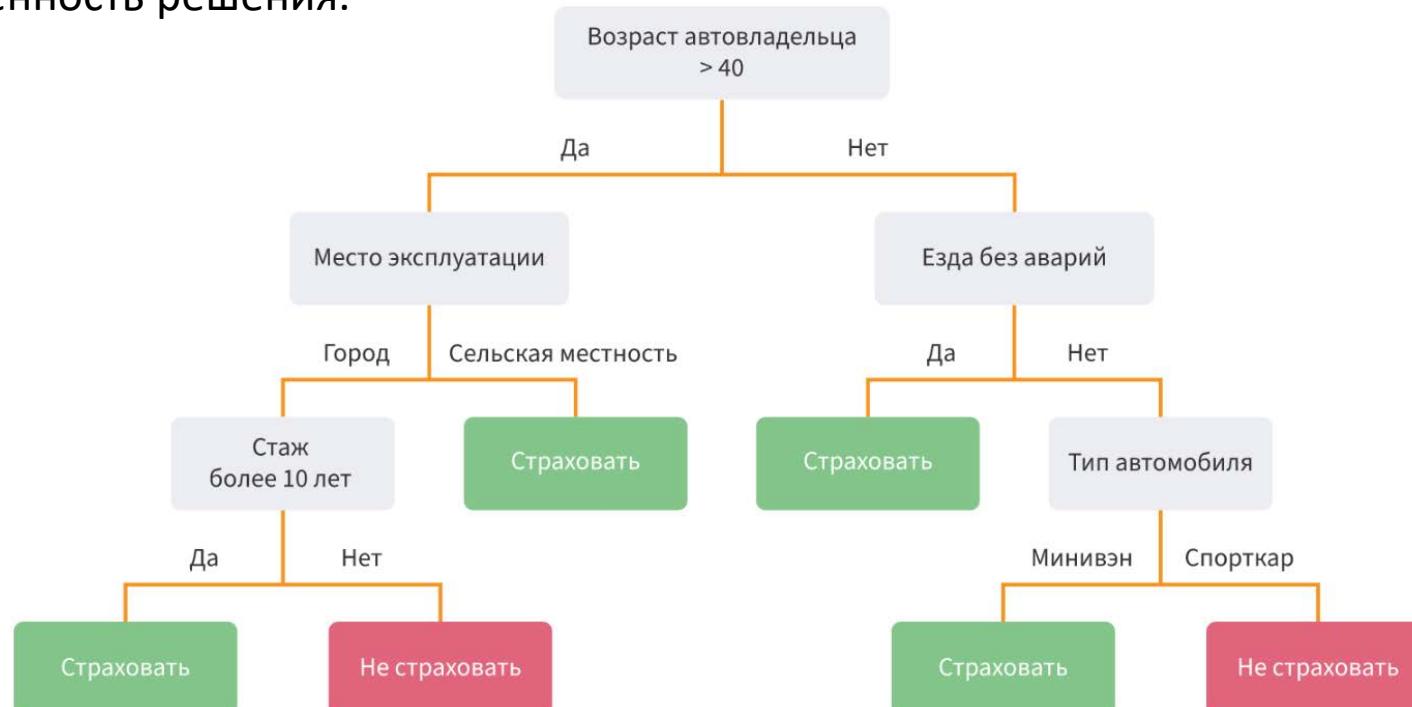
# Структура и функционирование

Дерево решений — метод представления решающих правил в иерархической структуре, состоящей из элементов двух типов — узлов (node) и листьев (leaf).

В узлах находятся решающие правила и производится проверка соответствия примеров этому правилу по какому-либо атрибуту обучающего множества.

Лист определяет решение для каждого попавшего в него примера. Для дерева классификации — класс, для дерева регрессии — модальный интервал целевой переменной.

Чтобы попасть в лист, пример должен удовлетворять всем правилам, лежащим на пути к этому листу. Поскольку путь в дереве к каждому листу единственный, то каждый пример может попасть только в один лист, что обеспечивает единственность решения.



# Алгоритмы обучения дерева решений

В настоящее время разработано значительное число алгоритмов обучения дерева решений: CLS, ID3, C4.5, CART, NewId, IRule, CHAID, CN2 и т.д. Наибольшее распространение и популярность получили следующие:

- **ID3 (Iterative Dichotomizer 3)** — позволяет работать только с дискретной целевой переменной, т.е. строит только классифицирующие деревья решений.
- **C4.5** — усовершенствованная версия алгоритма ID3, в которую добавлена возможность работы с пропущенными значениями атрибутов (по версии издания Springer Science в 2008 году алгоритм занял 1-е место в топ-10 наиболее популярных алгоритмов Data Mining).
- **CART (Classification and Regression Tree)** — алгоритм обучения деревьев решений способный использовать как дискретную, так и непрерывную целевую переменную, т.е. решать как задачи классификации, так и регрессии. Алгоритм строит деревья, которые в каждом узле имеют только два потомка.

# Процесс построения

Деревья решений являются моделями, строящимися на основе обучения с учителем, следовательно, в обучающем множестве для примеров должно быть задано целевое значение.

- Целевая переменная дискретная -> дерево классификации.
- Целевая переменная непрерывная -> дерево регрессии.

Процесс построения деревьев решений заключается в рекурсивном разбиении обучающего множества на подмножества с применением решающих правил в узлах. Разбиения продолжается до тех пор, пока все узлы в конце всех ветвей не будут объявлены листьями.

Объявление узла листом может произойти:

- содержит единственный объект, или объекты только одного класса;
- достижение некоторого условия остановки (минимально допустимое число примеров в узле или максимальная глубина дерева).

Алгоритмы построения деревьев решений относят к категории **«жадных алгоритмов»**. Жадными называются алгоритмы, которые допускают, что локально-оптимальные решения на каждом шаге (разбиения в узлах), приводят к оптимальному итоговому решению. Поэтому на этапе построения нельзя сказать обеспечит ли выбранный атрибут, в конечном итоге, оптимальное разбиение.

# Принцип «разделяй и властвуй»

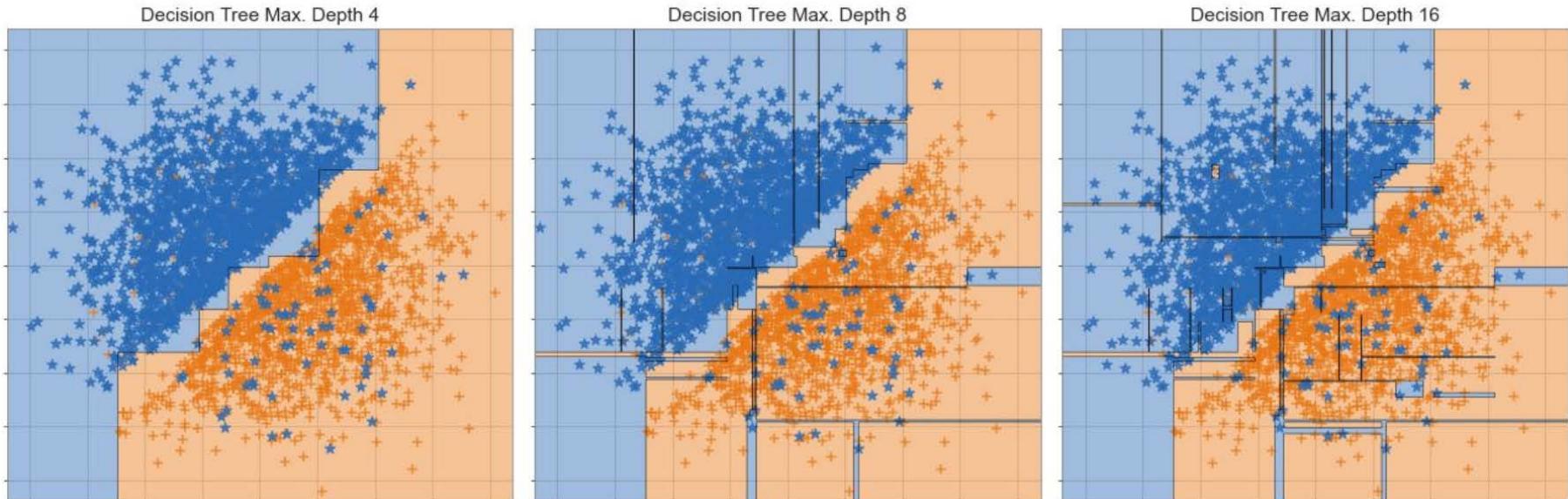
Пусть задано обучающее множество  $S$ , содержащее  $n$  примеров, для каждого из которых задана метка класса  $C_i (i = 1..k)$ , и  $m$  атрибутов  $A_j (j = 1..m)$ , которые, как предполагается, определяют принадлежность объекта к тому или иному классу. Тогда возможны три случая:

- Все примеры множества  $S$  имеют одинаковую метку класса  $C_i$  (т.е. все обучающие примеры относятся только к одному классу). Дерево решений в этом случае будет представлять собой лист, ассоциированный с классом  $C_i$ .
- Множество  $S$  не содержит примеров, т.е. является пустым. Для него тоже будет создан лист. Применять правило, чтобы создать узел, к пустому множеству бессмысленно.
- Множество  $S$  содержит обучающие примеры всех классов  $C_k$ . Требуется разбить множество  $S$  на подмножества, ассоциированные с классами. Для этого:
  - выбирается один из атрибутов  $A_j$  множества  $S$ , который содержит два и более уникальных значения  $(a_1, a_2, \dots, a_p)$ , где  $p$  — число уникальных значений признака.
  - множество  $S$  разбивается на  $p$  подмножеств  $(S_1, S_2, \dots, S_p)$ , каждое из которых включает примеры, содержащие соответствующее значение атрибута.
  - выбирается следующий атрибут и разбиение повторяется.
  - процедура рекурсивно повторяется до тех пор, пока все примеры в результирующих подмножествах не окажутся одного класса.

При использовании данной методики, построение дерева решений будет происходить сверху вниз (от корневого узла к листьям).

# Проблемы основных этапов построения

- Выбор атрибута, по которому будет производиться разбиение в данном узле (атрибут разбиения).
- Выбор критерия остановки обучения.
- Выбор метода отсечения ветвей (упрощения).
- Оценка точности построенного дерева.



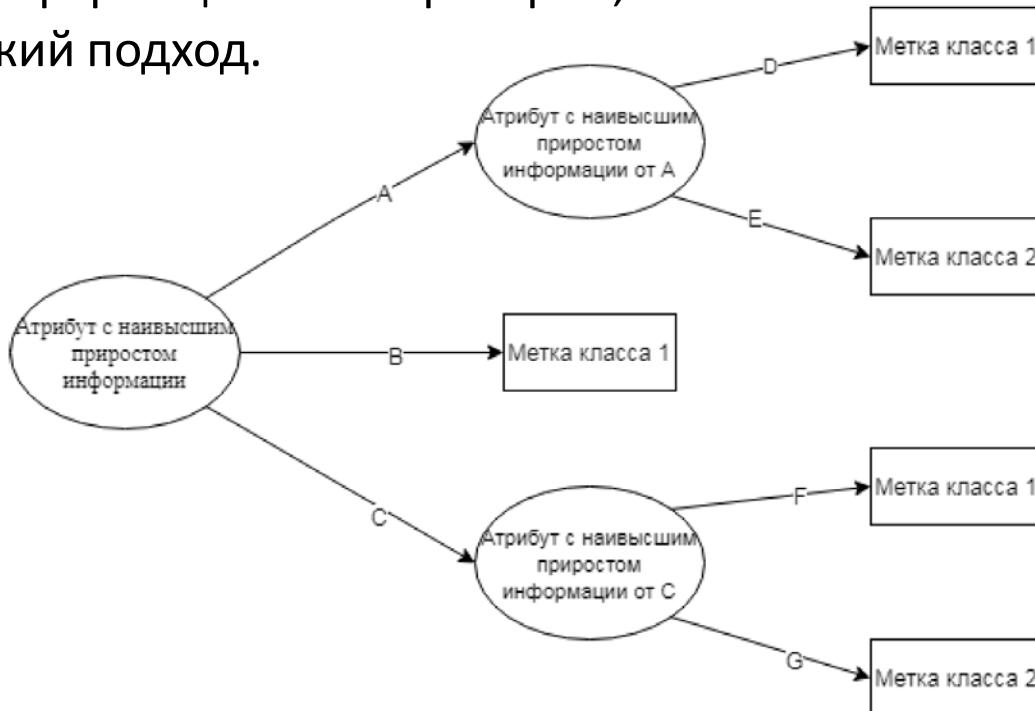
# Выбор атрибута разбиения

При формировании правила для разбиения в очередном узле дерева необходимо выбрать атрибут.

**Общее правило:** выбранный атрибут должен разбить множество наблюдений в узле так, чтобы результирующие подмножества содержали примеры с одинаковыми метками класса, или были максимально приближены к этому, т.е. количество объектов из других классов («примесей») в каждом из этих множеств было как можно меньше.

Наиболее популярные критерии:

- Теоретико-информационный критерий;
- Статистический подход.



# Теоретико-информационный критерий

Критерий основан на понятиях теории информации (информационной энтропии):

$$H(S, C) = - \sum_{i=1}^n \frac{N_i}{N} \log \left( \frac{N_i}{N} \right)$$

где  $n$  — число классов в исходном подмножестве,  $N_i$  — число примеров  $i$ -го класса,  $N$  — общее число примеров в подмножестве.

Энтропия может рассматриваться как мера неоднородности подмножества по представленным в нём классам. Если классы представлены в равных долях, то неопределённость классификации и энтропия максимальны. Если все примеры в узле относятся к одному классу, т.е.  $N = N_i$ , логарифм от единицы обращает энтропию в ноль.

Лучшим атрибутом разбиения  $A_j$  будет тот, который обеспечит максимальное снижение энтропии результирующего подмножества относительно родительского. На практике говорят не об энтропии, а об обратной величине, которая называется информацией. Тогда лучшим атрибутом разбиения будет тот, который обеспечит максимальный прирост информации результирующего узла относительно исходного:

$$Gain(S, A) = Info(S) - Info(S_A)$$

где  $Info(S)$  — информация, связанная с подмножеством  $S$  до разбиения,  $Info(S_A)$  — информация, связанная с подмножеством, полученным при разбиении по атрибуту  $A$ .

$$Gain(S, A) = H(S, C) - \sum_{j=1}^p \frac{|S_j|}{|S|} H(S_j, C)$$

где  $S_j$ -множество элементов  $S$ , на которых атрибут  $A$  имеет значение  $a_j$ .

Таким образом, задача выбора атрибута разбиения в узле заключается в максимизации величины  $Gain(S, A)$ , называемой приростом информации (gain — прирост, увеличение). Этот критерий впервые был применён в алгоритме ID3, а затем в C4.5 и других алгоритмах.

# Непрерывный атрибут

Если атрибут, по которому производится разбиение, непрерывный, то его сначала преобразуют в дискретный вид, например, с помощью операции квантования. Затем значения ранжируются и ищется среднее, которое используется для выбора порога. Все примеры, имеющие значения атрибута выше порогового, помещаются в один узел-потомок, а те, которые ниже — в другой.

**Выбор порога.** Пусть числовой атрибут  $X$  принимает конечное множество значений  $(x_1, x_2, \dots, x_p)$ . Упорядочив примеры по возрастанию значений атрибута, получим, что любое значение, лежащее между  $x_i$  и  $x_{i+1}$  делит все примеры на два подмножества. Первое подмножество будет содержать значения атрибута  $(x_1, x_2, \dots, x_i)$ , а второе —  $(x_{i+1}, x_{i+2}, \dots, x_p)$ . Тогда в качестве порога можно выбрать среднее:  $T_i = \frac{x_i + x_{i+1}}{2}$

Задача нахождения порога сводится к рассмотрению  $n - 1$  потенциальных пороговых значений  $T_1, T_2, \dots, T_{n-1}$ .

Последовательно применяя формулы для расчета теоретико-информационного критерия (информационной энтропии) ко всем потенциальным порогам, выбираем то, которое даёт максимальное значение по критерию:

$$Gain(S, A) = Info(S) - Info(S_A).$$

Затем, полученное значение сравнивается со значениями, рассчитанными для других атрибутов. Если это значение окажется наибольшим из всех атрибутов, то оно выбирается в качестве порога для проверки.

Следует отметить, что все числовые тесты являются бинарными, т.е. делят узел дерева на две ветви (два потомка).

# Статистический подход

В основе статистического подхода лежит использование индекса Джини (назван в честь итальянского статистика и экономиста Коррадо Джини). Статистический смысл данного показателя в том, что он показывает — насколько часто случайно выбранный пример обучающего множества будет распознан неправильно, при условии, что целевые значения в этом множестве были взяты из определённого статистического распределения.

Таким образом индекс Джини фактически показывает расстояние между двумя распределениями — распределением целевых значений, и распределением предсказаний модели. Очевидно, что чем меньше данное расстояние, тем лучше работает модель.

Индекс Джини может быть рассчитан по формуле:

$$Gini(S) = 1 - \sum_{i=1}^k p_i^2$$

где  $S$  — результирующее множество,  $k$  — число классов в нём,  $p_i^2$  — вероятность  $i$ -го класса (относительная частота примеров соответствующего класса). Данный показатель меняется от 0 до 1 (равен 0, если все примеры  $S$  относятся к одному классу). Лучшим будет то разбиение, для которого значение индекса Джини будет минимальным.

Соответственно, для выборки  $S$ , атрибута  $A$  ( $m$  значений) и целевого свойства  $C$  ( $k$  значений), индекс для выбора разделяющего атрибута вычисляется следующим образом:

$$Gini(S, A, C) = Gini(S, A) - \sum_{j=1}^p \frac{|S_j|}{|S|} Gini(S_j, A) \rightarrow \max$$

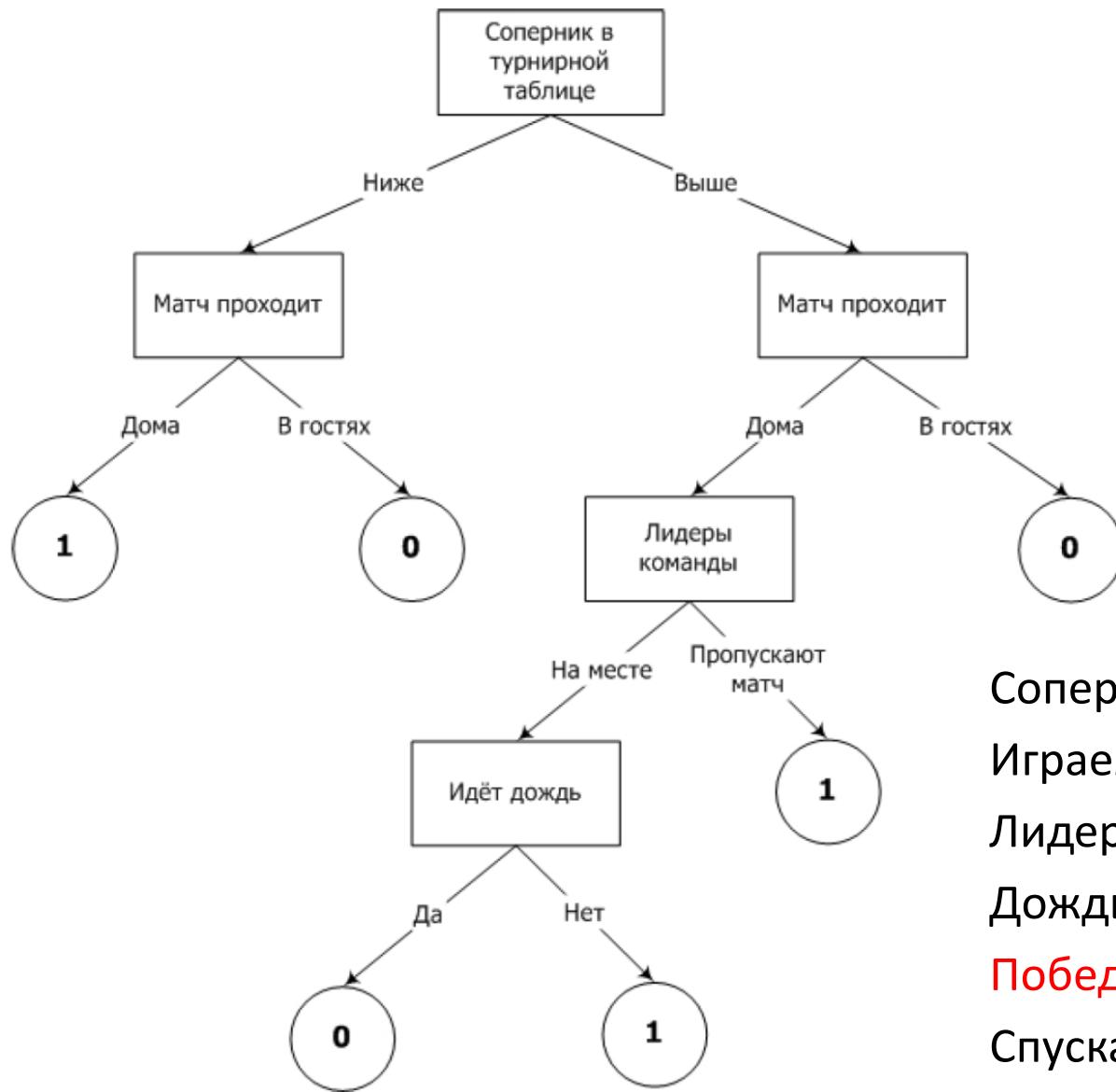
# Пример: Футбольная команда

Соперник	Играем	Лидеры	Дождь	Победа
Выше	Дома	На месте	Да	Нет
Выше	Дома	На месте	Нет	Да
Выше	Дома	Пропускают	Нет	Да
Ниже	Дома	Пропускают	Нет	Да
Ниже	В гостях	Пропускают	Нет	Нет
Ниже	Дома	Пропускают	Да	Да
Выше	В гостях	На месте	Да	Нет
Ниже	В гостях	На месте	Нет	???

Атрибуты: Соперник (выше/ниже по турнирной таблице), Играем (домашняя или выездная игра), Лидеры (Лидеры команды: играют/не играют), Дождь (наличие дождя: Да/Нет).

Целевой показатель: Победа (Да/Нет).

# Пример: построенное дерево



Соперник = Ниже

Играем = В гостях

Лидеры = На месте

Дождь = Нет

**Победа = ?**

Спускаясь по дереву получаем,  
что команда должна проиграть.

# Пример: Строим дерево

Воспользуемся Теоретико-информационным критерием (энтропией и приростом информации).

В нашем примере из 7 матчей команда три проиграла и четыре выиграла. Поэтому исходная энтропия:

$$H(S, \text{Победа}) = -\frac{4}{7} \log_2 \frac{4}{7} - \frac{3}{7} \log_2 \frac{3}{7} \approx 0.9852.$$

Прирост информации:

$$\begin{aligned} \text{Gain}(S, \text{Соперник}) &= \\ &= H(S, \text{Победа}) - \frac{4}{7} H(S_{\text{выше}}, \text{Победа}) - \frac{3}{7} H(S_{\text{ниже}}, \text{Победа}) \approx \\ &\approx 0.9852 - \frac{4}{7} \left( -\frac{1}{2} \log_2 \frac{1}{2} - \frac{1}{2} \log_2 \frac{1}{2} \right) - \frac{3}{7} \left( -\frac{2}{3} \log_2 \frac{2}{3} - \frac{1}{3} \log_2 \frac{1}{3} \right) \approx 0.0202. \end{aligned}$$

Явно не самый удачный критерий для корня дерева.

$$\text{Gain}(S, \text{Играем}) \approx 0.4696.$$

$$\text{Gain}(S, \text{Лидеры}) \approx 0.1281.$$

$$\text{Gain}(S, \text{Дождь}) \approx 0.1281.$$

# Проблема критерия прироста информации

Можно заметить, что критерий разбиения на основе прироста информации отдаёт предпочтение атрибутам, которые содержат большее число уникальных значений. Это связано с тем, что при этом создается большое количество узлов-потомков, дающее в итоге более равномерное распределение классов и, соответственно, меньшую энтропию разбиения. В предельном случае, если все значения атрибута будут уникальными, то для каждого примера будет создано отдельное правило и отдельный лист с единственным классом.

Как следствие, разбиение даст нулевую энтропию:

$$Info(S, A) = 0$$

и максимальный прирост информации.

Это оптимально с «точки зрения» алгоритма, но абсолютно бессмысленно с практический точки зрения, поскольку:

- значимость правил будет чрезвычайно низкая, т.к. правило действует только для конкретного объекта;
- дерево решений окажется очень сложным для понимания;
- дерево решений получится переобученным, т.е. не будет обладать обобщающей способностью.

# Пример: Проблема критерия прироста информации

Пусть в таблице игр были дополнительно записаны даты матчей. Тогда прирост информации:

$$\begin{aligned} \text{Gain}(S, \text{Дата}) &= H(S, \text{Победа}) - \\ &- \sum_{i=1}^n \frac{1}{n} H(S_{\text{Дата}=i}, \text{Победа}) = H(S, \text{Победа}) \end{aligned}$$

И равен энтропии победы в исходном множестве. Это происходит потому, что в каждой из веток только один случай, а энтропия в каждой ветке равна нулю.

Прирост информации - максимальный из возможных но полученное дерево абсолютно бесполезно.

Решение данной проблемы достаточно очевидно — чтобы алгоритм при выборе разбиения не отдавал предпочтение атрибутам с большим числом уникальных значений, следует ввести некоторый штраф, накладываемый на атрибут за количество созданных с его помощью узлов-потомков.

# Split-Info и Gain-Ratio

Штраф может быть представлен в виде некоторого коэффициента для значения прироста информации.

Введём в рассмотрение показатель:

$$SplitInfo(S, A) = - \sum_{j=1}^p \frac{|S_j|}{|S|} \log_2 \frac{|S_j|}{|S|}$$

который представляет собой оценку потенциальной информации, созданной при разбиении множества  $S$  на  $p$  подмножеств  $S_j$ . Тогда критерий прироста информации с учётом показателя  $SplitInfo(S, A)$  будет иметь вид:

$$GainRatio(S, A) = \frac{Gain(S, A)}{SplitInfo(S, A)}$$

Новый критерий позволяет оценить долю информации, полученной при разбиении, которая является «полезной», то есть способствует улучшению классификации. Применение данного показателя, как правило, приводит к выбору более удачного атрибута, чем использование обычного критерия прироста информации.

Смысл этой модификации достаточно прост.  $|S_j|/|S|$  — отношение числа примеров в подмножестве, полученном в результате разбиения, к числу примеров в родительском множестве  $|S|$ . Если в результате разбиения получается большое число подмножеств с небольшим числом примеров, что характерно для переобучения, то показатель  $SplitInfo$  растет. Поскольку он стоит в знаменателе выражения, то  $GainRatio$  есть обычный прирост информации, «оштрафованный» с помощью  $SplitInfo$ .

# Пример: Split-Info и Gain-Ratio

У атрибута «Дата»:

$$\text{SplitInfo}(S, \text{Дата}) = - \sum_{i=1}^7 \frac{1}{7} \log_2 \frac{1}{7} \approx 2.80735\dots$$

$$\text{GainRatio}(S, \text{Дата}) = \frac{\text{Gain}(S, \text{Дата})}{\text{SplitInfo}(S, \text{Дата})} \approx 0.350935\dots$$

А для атрибута «Играем», показывающего, где проходит матч:

$$\text{SplitInfo}(S, \text{Играем}) = -\frac{5}{7} \log_2 \frac{5}{7} - \frac{2}{7} \log_2 \frac{2}{7} \approx 0.86312\dots$$

$$\text{GainRatio}(S, \text{Играем}) = \frac{\text{Gain}(S, \text{Играем})}{\text{SplitInfo}(S, \text{Играем})} \approx 0.5452\dots$$

# Обработка пропусков в данных - Критерий

Пусть  $S$  — множество обучающих примеров. Обозначим  $U$  — количество примеров, у которых не определено значение атрибута  $A$ . Изменим слагаемые из критерия  $Gain(S, A) = Info(S) - Info(S_A)$  таким образом, чтобы учитывать только те примеры, для которых значения атрибута существуют.

$$Info(S) = H(S, C) = - \sum_{i=1}^k \frac{N(C_i, S)}{N(S) - U} \log \left( \frac{N(C_i, S)}{N(S) - U} \right)$$
$$Info(S_A) = \sum_{j=1}^p \frac{N(S_j)}{N(S) - U} H(S_j, C)$$

где  $k$  — число классов в исходном подмножестве,  $N(C_i, S)$  — число примеров  $i$ -го класса в множестве  $S$  (учитываются примеры, не содержащие пропусков в значениях атрибута  $A$ ),  $N(S)$  — общее число примеров в множестве  $S$ ,  $p$  — число уникальных значений признака атрибута  $A$ .

Критерий  $Gain(S, A)$  может быть преобразован к виду:

$$Gain(S, A) = \frac{N(S) - U}{N(S)} (Info(S) - Info(S_A)).$$

# Обработка пропусков в данных - Разбиение

Пусть разбиение, формирующее  $r$  выходов  $(O_1, O_2, \dots, O_p)$ , использует атрибут  $A$ , выбранный по критерию  $Gain(S, A)$ . Каждый пример из множества  $S$ , порождающий выход  $O_j$ , может быть ассоциирован с подмножеством  $S_j$  с некоторой вероятностью, выраженной через веса:

- Если пример содержит значение атрибута  $A$ , то вес примера равен 1.
- В противном случае пример ассоциируется со всеми подмножествами  $S_1, S_2, \dots, S_p$  с соответствующими весами:

$$\frac{N(S_1)}{N(S) - U}, \frac{N(S_2)}{N(S) - U}, \dots, \frac{N(S_p)}{N(S) - U}$$

Несложно убедиться, что

$$\sum_{j=1}^p \frac{N(S_j)}{N(S) - U} = 1$$

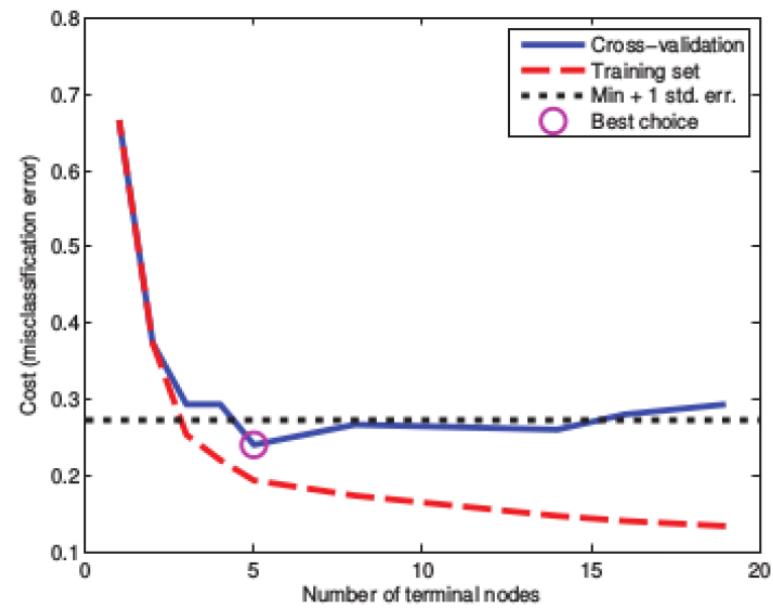
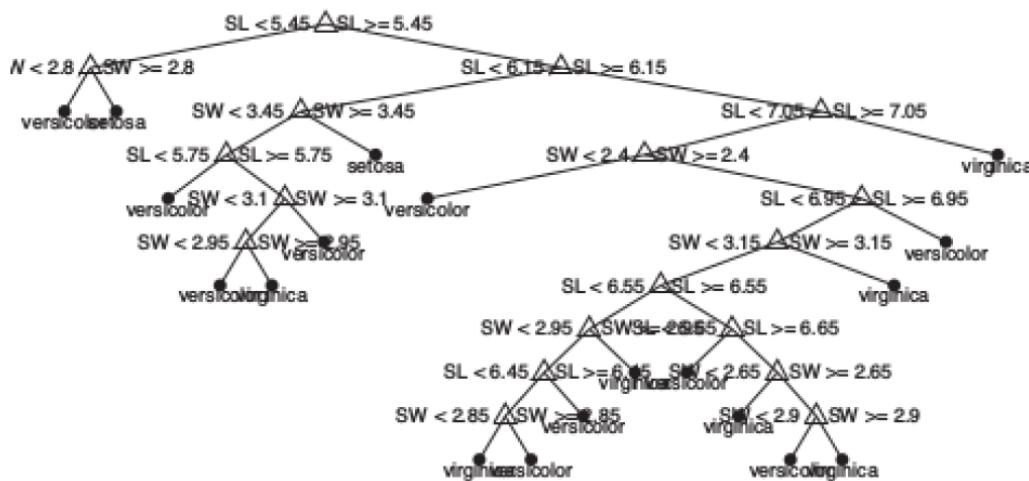
Таким образом, этот подход можно сформулировать так: предполагается, что пропущенные значения по атрибуту распределены пропорционально частоте появления существующих значений.

Если при классификации новых объектов на каком-то узле выясняется, что значение соответствующего атрибута пропущено, то алгоритм исследует все возможные пути вниз по дереву и определяет с какой вероятностью пример относится к различным классам.

# Эффект переобучения

Теоретически, алгоритм обучения дерева решений будет работать до тех пор, пока в результате не будут получены абсолютно «чистые» подмножества, в каждом из которых будут примеры одного класса. Возможна ситуация, что для каждого примера будет создан отдельный лист. Такое дерево окажется переобученным, т.к. каждому примеру будет соответствовать свой уникальный путь в дереве, а следовательно, и набор правил.

Переобучение — точное распознавание примеров, участвующих в обучении и полная несостоительность на новых данных. Переобученные деревья имеют очень сложную структуру, и их сложно интерпретировать.



# Критерий остановки алгоритма

Подходы для борьбы с переобучением:

- **Ранняя остановка** — алгоритм будет остановлен, как только будет достигнуто заданное значение некоторого критерия (например, процентной доли правильно распознанных примеров). Преимущество - снижение времени обучения. Главный недостаток – снижение точности.
- **Ограничение глубины дерева** — задание максимального числа разбиений в ветвях. Данный метод также ведёт к снижению точности дерева.
- **Задание минимально допустимого** числа примеров в узле — запрет алгоритму создавать узлы с числом примеров меньше заданного значения (например, 5).

Все перечисленные подходы являются эвристическими, т.е. не гарантируют лучшего результата или вообще работают только в каких-то частных случаях.

# Борьба с переобучением

## Отсечение ветвей - pruning

Представляет интерес подход, альтернативный ранней остановке — построить все возможные деревья и выбрать то из них, которое при разумной глубине обеспечивает приемлемый уровень ошибки распознавания, т.е. найти наиболее выгодный баланс между сложностью и точностью дерева.

К сожалению, это задача относится к классу NP-полных задач, что было показано Л. Хайфилем (L. Hyafil) и Р. Ривестом (R. Rivest), и, как известно, этот класс задач не имеет эффективных методов решения.

Альтернативным подходом является так называемое **отсечение ветвей** (**pruning**). Он содержит следующие шаги:

- Построить полное дерево (чтобы все листья содержали примеры одного класса).
- Определить два показателя: относительную точность модели — отношение числа правильно распознанных примеров к общему числу примеров, и абсолютную ошибку — число неправильно классифицированных примеров.
- Удалить из дерева листья и узлы, отсечение которых не приведёт к значимому уменьшению точности модели или увеличению ошибки.

Отсечение ветвей, очевидно, производится в направлении, противоположном направлению роста дерева, т.е. снизу вверх, путём последовательного преобразования узлов в листья. Преимуществом отсечения ветвей по сравнению с ранней остановкой является возможность поиска оптимального соотношения между точностью и понятностью дерева. Недостатком является большее время обучения из-за необходимости сначала построить полное дерево.

# Извлечение правил

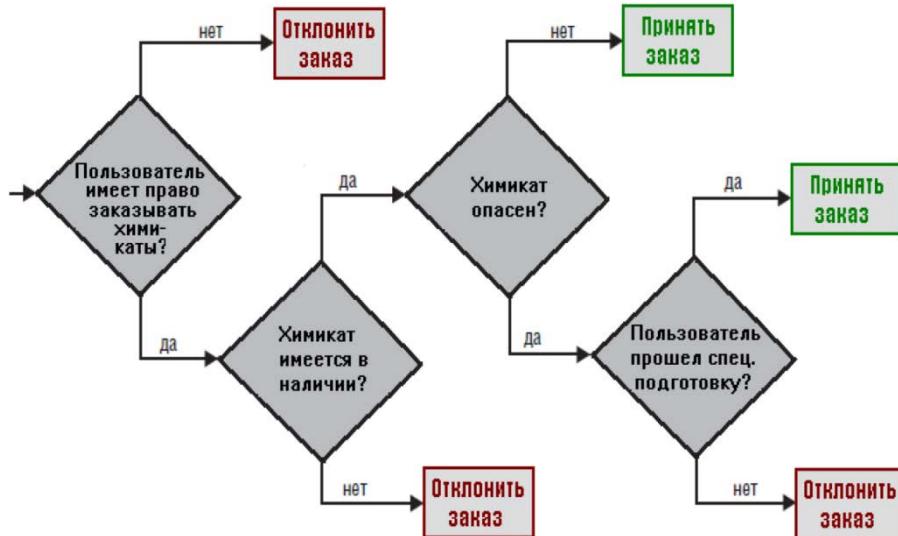
Для упрощения интерпретации результата может оказаться полезным извлечь из дерева **решающие правила** и организовать их в наборы, описывающие классов.

Для извлечения правил нужно отследить все пути от корневого узла к листьям дерева. Каждый такой путь даст правило, состоящее из множества условий, представляющих собой проверку в каждом узле пути.

Визуализация сложных деревьев решений в виде решающих правил вместо иерархической структуры из узлов и листьев может оказаться более удобной для визуального восприятия.

**Если (условие 1)  $\wedge$  (условие 2)  $\wedge \dots \wedge$  (условие N)**  
**то (значение вершины вывода)**

Представление решающих правил в **табличной форме** с возможностью фильтрации по значениям атрибутов и целевой переменной.



Номер требования		1	2	3	4	5
Условие		нет	да	да	да	да
Пользователь имеет право заказывать химикаты?		нет	да	да	да	да
Химикат в наличии?		—	нет	да	да	да
Химикат опасен?		—	—	нет	да	да
Пользователь прошел спец.подготовку?		—	—	—	нет	да
Действие						
Принять заказ				X		X
Отклонить заказ		X	X		X	

# Алгоритм CART (CART algorithm)

Может работать как с дискретной, так и с непрерывной выходной переменной, т.е. решать задачи и классификации, и регрессии. Строит бинарные деревья решений, которые содержат только два потомка в каждом узле.

Алгоритм реализует обучение с учителем и использует в качестве критерия для выбора разбиений в узлах индекс Джини (*Gini impurity index*). В процессе роста дерева алгоритм CART проводит для каждого узла полный перебор всех атрибутов, на основе которых может быть построено разбиение, и выбирает тот, который максимизирует значение индекса Джини. Полученные дочерние узлы должны быть максимально однородными.

Если атрибут  $X$ , по которому производится разбиение, является номинальным с  $i$  категориями, то для него существует  $2^{(i-1)} - 1$  возможных разбиения, а если порядковым или непрерывным с  $K$  различными значениями, существует  $K-1$  различных разбиений по  $X$ .

Дерево строится, начиная с корневого узла, путем итеративного использования следующих шагов в каждом узле:

- Для каждого атрибута ищется лучшее разбиение (в смысле однородности результирующих подмножеств).
- Среди всех разбиений, найденных на предыдущем шаге, выбирается то, для которого критерий разбиения наибольший.
- Узел разбивается с использованием лучшего разбиения, найденного на шаге 2, если не выполнено условие остановки.

# Основные достоинства деревьев решений

- быстрый процесс обучения;
- генерация правил в областях, где эксперту трудно формализовать свои знания;
- извлечение правил на естественном языке;
- интуитивно понятная классификационная модель;
- высокая точность предсказания, сопоставимая с другими методами анализа данных (статистика, нейронные сети);
- построение непараметрических моделей.

# Основные недостатки деревьев решений

- Деревья очень чувствительны к шумам во входных данных, вся модель может кардинально измениться, если немного изменится обучающая выборка (например, если убрать один из признаков или добавить несколько объектов), поэтому и правила классификации могут сильно изменяться;
- Нестабильность. Небольшие изменения в данных могут существенно изменять построенное дерево решений. С этой проблемой борются с помощью ансамблей деревьев решений (рассмотрим далее);
- Разделяющая граница, построенная деревом решений, имеет свои ограничения (состоит из гиперплоскостей, перпендикулярных какой-то из координатной оси), и на практике дерево решений по качеству классификации уступает некоторым другим методам;
- Необходимость отсекать ветви дерева (pruning) или устанавливать минимальное число элементов в листьях дерева или максимальную глубину дерева для борьбы с переобучением;
- Проблема поиска оптимального дерева решений (минимального по размеру и способного без ошибок классифицировать выборку) NP-полна, поэтому на практике используются эвристики типа жадного поиска признака с максимальным приростом информации, которые не гарантируют нахождения глобально оптимального дерева;
- Сложно поддерживать пропуски в данных. Friedman оценил, что на поддержку пропусков в данных ушло около 50% кода CART (Classification And Regression Trees);
- Модель умеет только интерполировать, но не экстраполировать (это же верно и для леса и бустинга на деревьях).

**Спасибо за внимание**

# Методы машинного обучения

*Лекция 9*

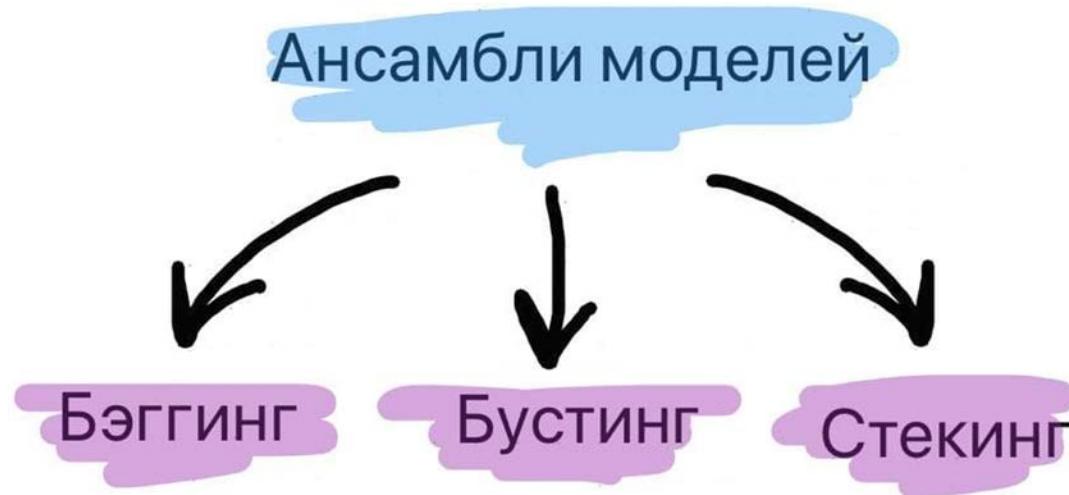
## Ансамбли классификаторов

# Ансамбли (Ensemble)

Ансамбли — это контролируемые алгоритмы обучения, которые комбинируют прогнозы из двух и более алгоритмов машинного обучения для построения более точных результатов. Результаты можно комбинировать с помощью голосования или усреднения. Первое зачастую применяется в классификации, а второе — в регрессии.

Существует 3 основных типа ансамблевых алгоритмов:

- Бэггинг.** Алгоритмы обучаются и работают параллельно на разных тренировочных наборах одного размера. Затем все они тестируются на одном наборе данных, а конечный результат определяется с помощью голосования.
- Бустинг.** В этом типе алгоритмы обучаются последовательно, а конечный результат отбирается с помощью голосования с весами.
- Стекинг (наложение).** Исходя из названия, этот подход состоит из двух уровней, расположенных друг на друге. Базовый представляет собой комбинацию алгоритмов, а верхний — мета-алгоритмы, основанные на базовом уровне.



# Оценка качества работы алгоритма

Предположим, что мы решаем задачу регрессии с квадратичной функцией потерь. При использовании квадратичной функции потерь для оценки качества работы алгоритма  $a$  можно использовать следующий функционал:

$$Q(a) = \mathbb{E}_x \mathbb{E}_{X,\epsilon} [y(x, \epsilon) - a(x, X)]^2$$

где

- $X$  – обучающая выборка
- $x$  – точка из тестового множества
- $y(x, \epsilon) = f(x) + \epsilon$  – целевая зависимость, которую мы можем измерить с точностью до случайного шума  $\epsilon$
- $a(x, X)$  – значение алгоритма, обученного на выборке  $X$ , в точке  $x$
- $\mathbb{E}_x$  – среднее по всем тестовым точкам
- $\mathbb{E}_{X,\epsilon}$  – среднее по всем обучающим выборкам  $X$  и случайному шуму  $\epsilon$ .

Для  $Q(a)$  существует разложение на три компонента:

**шум, смещение (отклонение), разброс (дисперсия).**

Это разложение называется **bias-variance decomposition**, и оно является одним из мощных средств для анализа работы ансамблей.

# Компромисс отклонение-дисперсия

**Компромисс отклонение-дисперсия** в статистике и машинном обучении — это свойство набора моделей предсказания, когда модели с меньшим отклонением от имеющихся данных имеют более высокую дисперсию на новых данных (то есть подвержены переобучению), и наоборот. **Компромисс отклонение-дисперсия** — конфликт при попытке одновременно минимизировать эти два источника ошибки, которые мешают алгоритмам обучения с учителем делать обобщение за пределами тренировочного набора.

- **Смещение** — это погрешность оценки, возникающая в результате ошибочного предположения в алгоритме обучения на имеющихся (тренировочных) данных. В результате большого смещения алгоритм может пропустить связь между признаками и выводом (недообучение).
- **Дисперсия** — это ошибочная чувствительность к малым отклонениям в тренировочном наборе. Высокая дисперсия говорит о том, что алгоритм пытается как-то трактовать случайный шум в тренировочном наборе, а не желаемый результат (переобучение).

**Разложение смещения-дисперсии** — это способ анализа ожидаемой ошибки обобщения алгоритма обучения для частной задачи сведением к сумме трёх членов — смещения, дисперсии и величины, называемой **неустранимой погрешностью**, которая является результатом шума в самой задаче.

# Bias-variance decomposition (смещение, дисперсия, шум)

Существует представление  $Q(a)$  в виде трёх компонент:

$$Q(a) = \mathbb{E}_x bias_X^2 a(x, X) + \mathbb{E}_x \mathbb{V}_X[a(x, X)] + \sigma^2$$

где

- $bias_X a(x, X) = f(x) - \mathbb{E}_X[a(x, X)]$  - **смещение** предсказания алгоритма в точке  $x$ , усреднённого по всем возможным обучающим выборкам, относительно истинной зависимости  $f(x)$ ;
- $\mathbb{V}_X[a(x, X)] = \mathbb{E}_X[a(x, X) - \mathbb{E}_X[a(x, X)]]^2$  - **дисперсия (разброс)** предсказаний алгоритма в зависимости от обучающей выборки;
- $\sigma^2 = \mathbb{E}_x \mathbb{E}_\epsilon [y(x, \epsilon) - f(x)]^2$  **неустранимый шум** в данных.

Понятно, что с шумом ничего сделать нельзя, а вот смещение и дисперсия нас будут интересовать с точки зрения из минимизации.

# Теорема Кондорсе о присяжных

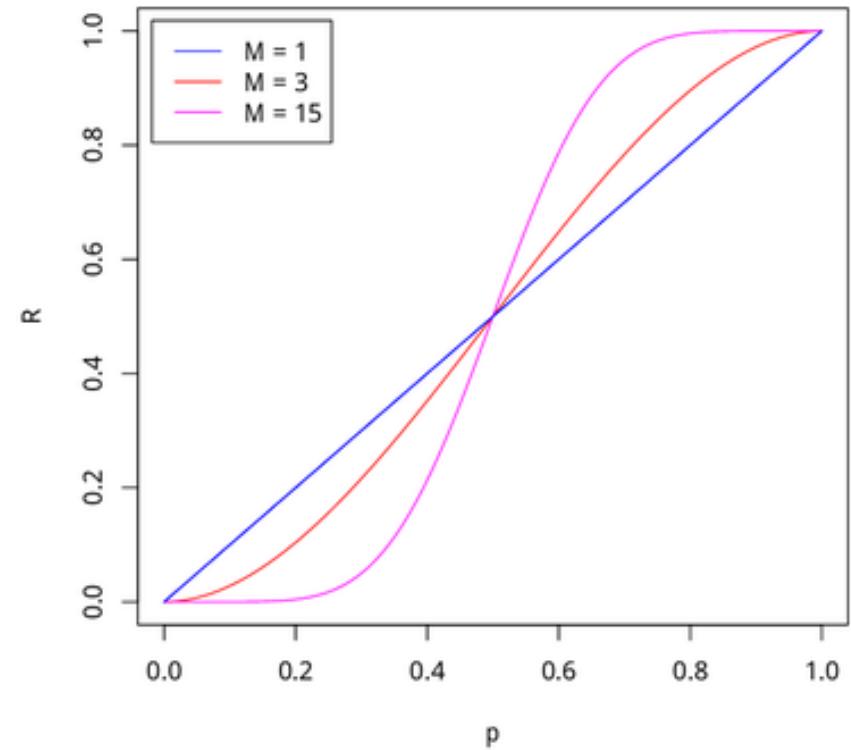
Если каждый член жюри присяжных имеет независимое мнение, и если вероятность правильного решения члена жюри больше 0.5, то тогда вероятность правильного решения присяжных в целом возрастает с увеличением количества членов жюри, и стремится к единице. Если же вероятность быть правым у каждого из членов жюри меньше 0.5, то вероятность принятия правильного решения присяжными в целом монотонно уменьшается и стремится к нулю с увеличением количества присяжных.

Пусть  $M$  – количество присяжных,  $p$  – вероятность правильного решения одного эксперта,  $R$  – вероятность правильного решения всего жюри,  $m$  – минимальное большинство членов жюри:

$$m = \left\lfloor \frac{M}{2} \right\rfloor + 1$$

Тогда

$$R = \sum_{i=m}^M C_M^i p^i (1-p)^{M-i}$$



# Бэггинг - Идея

Пусть обучающая выборка состоит из  $n$  объектов. Выберем из неё  $n$  примеров равновероятно, с возвращением. Получим новую выборку  $X^1$ , в которой некоторых элементов исходной выборки не будет, а какие-то могут войти несколько раз. С помощью некоторого алгоритма  $b$  обучим на этой выборке модель  $b_1(x) = b(x, X^1)$ . Повторим процедуру: сформируем вторую выборку  $X^2$  из  $n$  элементов с возвращением и с помощью того же алгоритма обучим на ней модель  $b_2(x) = b(x, X^2)$ . Повторив процедуру  $k$  раз, получим  $k$  моделей, обученных на  $k$  выборках. Чтобы получить одно предсказание, усредним предсказания всех моделей:

$$a(x) = \frac{1}{k} (b_1(x) + \dots + b_k(x))$$

Процесс генерации подвыборок с помощью семплирования с возвращением называется **бутстрепом** (**bootstrap**), а модели  $b_1(x), \dots, b_k(x)$  часто называют **базовыми алгоритмами** (моделями). Модель  $a(x)$  называется **ансамблем** этих моделей.

## Бэггинг - Смещение

Будем считать, что когда мы берём матожидание по всем обучающим выборкам  $X$ , то в эти выборки включены также все подвыборки, полученные бутстрепом.

$$\begin{aligned} bias_X a(x, X) &= f(x) - \mathbb{E}_X[a(x, X)] = f(x) - \mathbb{E}_X \left[ \frac{1}{k} \sum_{i=1}^k b(x, X^i) \right] = \\ &= f(x) - \frac{1}{k} \sum_{i=1}^k \mathbb{E}_X[b(x, X^i)] = f(x) - \frac{1}{k} \sum_{i=1}^k \mathbb{E}_X[b(x, X)] = \\ &= f(x) - \mathbb{E}_X[b(x, X)] = bias_X b(x, X) \end{aligned}$$

Получаем, что смещение ансамбля не изменилось по сравнению со средним смещением отдельных моделей.

# Бэггинг – Разброс (Дисперсия)

$$\begin{aligned}\mathbb{V}_X[a(x, X)] &= \mathbb{E}_X[a(x, X) - \mathbb{E}_X[a(x, X)]]^2 = \\&= \mathbb{E}_X \left[ \frac{1}{k} \sum_{i=1}^k b(x, X^i) - \mathbb{E}_X \left[ \frac{1}{k} \sum_{i=1}^k b(x, X^i) \right] \right]^2 = \frac{1}{k^2} \mathbb{E}_X \left[ \sum_{i=1}^k (b(x, X^i) - \mathbb{E}_X[b(x, X^i)]) \right]^2 = \\&= \frac{1}{k^2} \sum_{i=1}^k \mathbb{E}_X (b(x, X^i) - \mathbb{E}_X[b(x, X^i)])^2 + \\&\quad + \frac{1}{k^2} \sum_{k_1 \neq k_2} \mathbb{E}_X [(b(x, X^{k_1}) - \mathbb{E}_X[b(x, X^{k_1})])(b(x, X^{k_2}) - \mathbb{E}_X[b(x, X^{k_2})])] = \\&= \frac{1}{k^2} \sum_{i=1}^k \mathbb{V}_X b(x, X^i) + \frac{1}{k^2} \sum_{k_1 \neq k_2} cov(b(x, X^{k_1}), b(x, X^{k_2}))\end{aligned}$$

Если предположить, что базовые алгоритмы некоррелированы, то:

$$\mathbb{V}_X[a(x, X)] = \frac{1}{k^2} \sum_{i=1}^k \mathbb{V}_X b(x, X^i) = \frac{1}{k^2} \sum_{i=1}^k \mathbb{V}_X b(x, X) = \frac{1}{k} \mathbb{V}_X b(x, X)$$

**Дисперсия композиции в  $k$  раз меньше дисперсии отдельного алгоритма!**

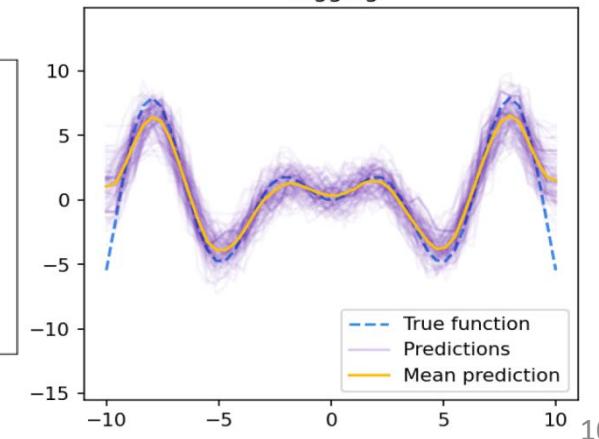
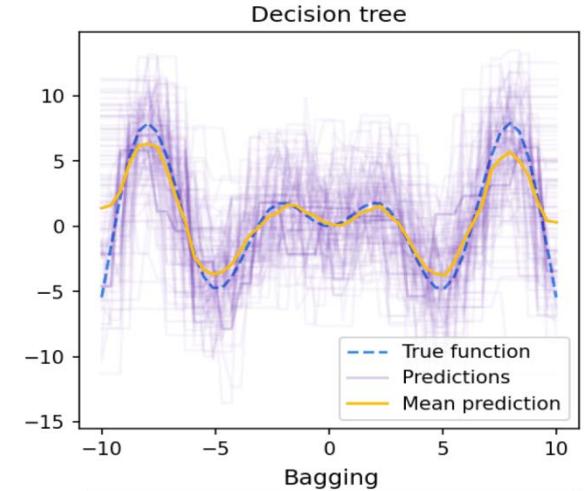
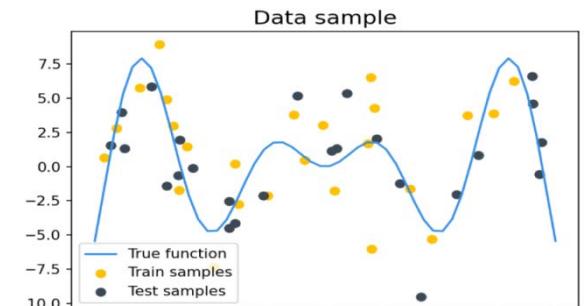
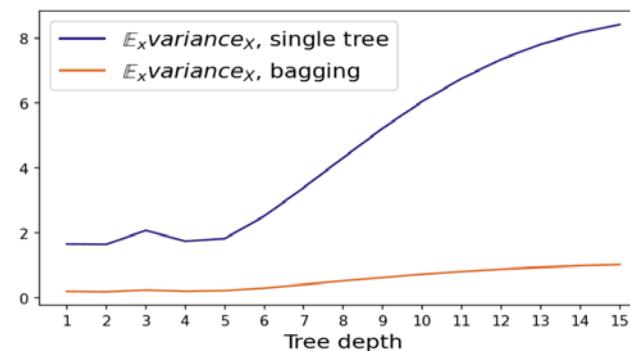
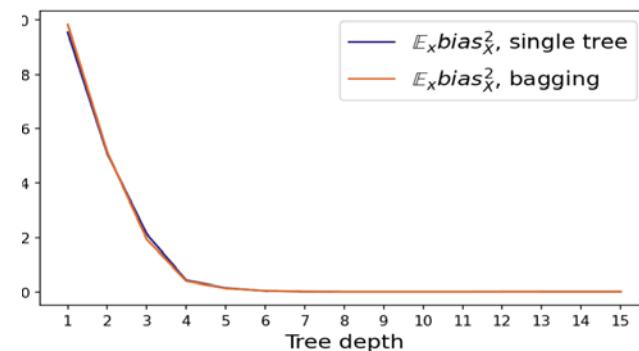
# Пример: бэггинг над решающими деревьями

Пусть наша целевая зависимость  $f(x)$  задаётся как  $f(x) = x \cdot \sin(x)$  и к ней добавляется нормальный шум  $\epsilon \sim N(0, 0.9)$ .

Обучим 100 решающих деревьев глубины 7 на различных случайных выборках размера 20. Возьмём также бэггинг над 10 решающими деревьями глубины 7 в качестве базовых классификаторов и тоже 100 раз обучим его на случайных выборках размера 20.

Общая дисперсия предсказаний в зависимости от обучающего множества у бэггинга значительно ниже, чем у отдельных деревьев, а в среднем предсказания деревьев и бэггинга не отличаются.

Дисперсия уменьшилась практически в  $k$  раз, что равняется числу базовых алгоритмов, которые бэггинг использовал для предсказания.



# Random Forest (Случайный лес)

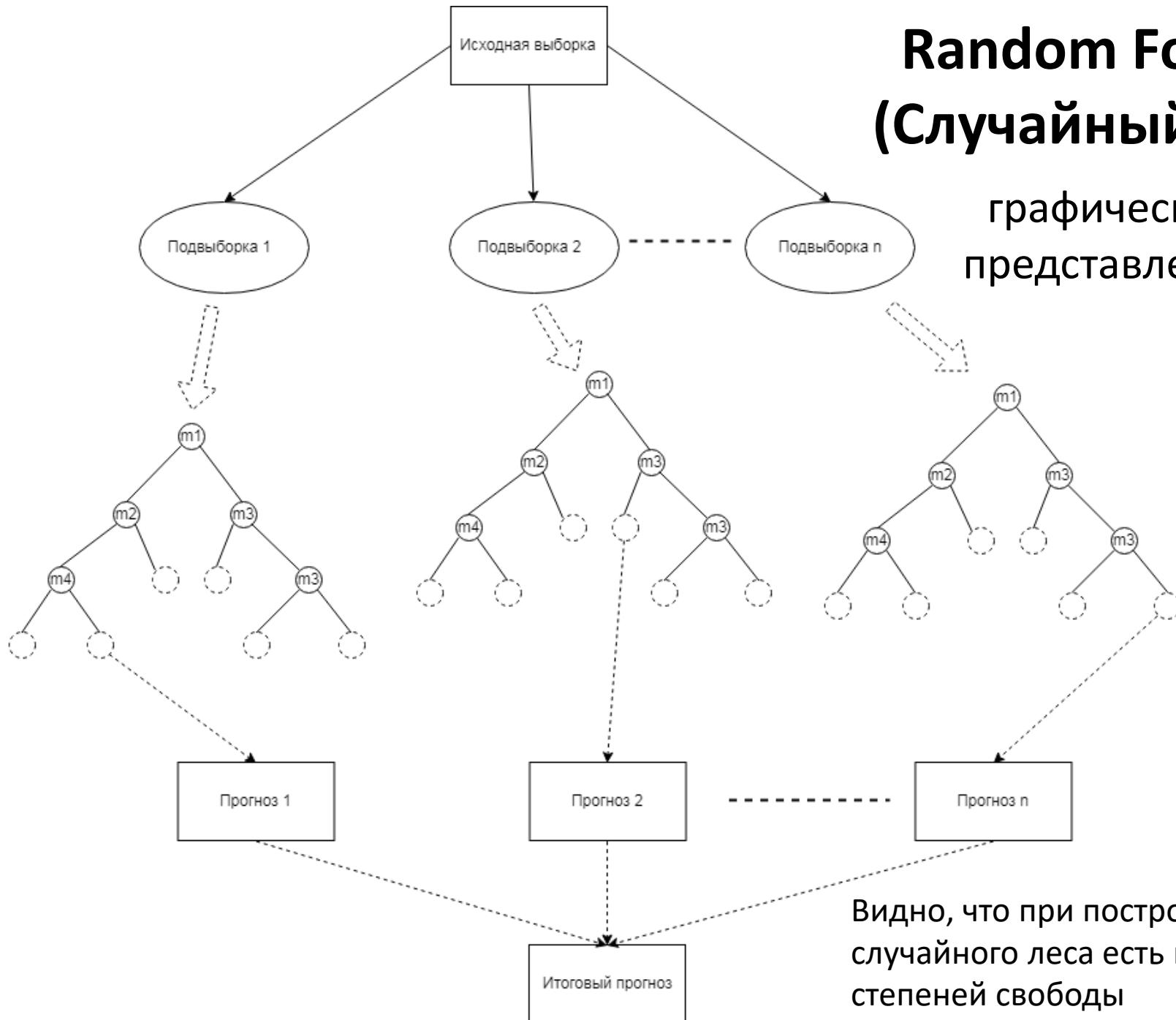
Мы делали предположение, что базовые алгоритмы некоррелированы. Однако в реальной жизни добиться этого проблематично: ведь базовые алгоритмы учат одну и ту же зависимость на пересекающихся выборках. В реальности достаточно, чтобы алгоритмы были в некоторой степени не похожи друг на друга. Развитие идеи бэггинга для решающих деревьев — **случайный лес**. Строим ансамбль алгоритмов по следующей схеме:

1. Для построения  $i$ -го дерева:
  - Сначала, как в обычном бэггинге, из обучающей выборки  $X$  выбирается с возвращением случайная подвыборка  $X^i$  того же размера, что и  $X$ .
  - В процессе обучения каждого дерева **в каждой вершине** случайно выбираются  $n < N$  признаков, где  $N$  — полное число признаков (метод случайных подпространств), и среди них ищется оптимальное разбиение. Такой приём позволяет управлять степенью коррелированности базовых алгоритмов.
2. Чтобы получить предсказание ансамбля на тестовом объекте, усредняем отдельные ответы деревьев (для регрессии) или берём самый популярный класс (для классификации).

**Random Forest (случайный лес)** — комбинация бэггинга и метода случайных подпространств над решающими деревьями.

# Random Forest (Случайный лес)

графическое  
представление



# Основные параметры при построении случайного леса

- **Глубина деревьев** в случайном лесу

Ошибка модели (на которую мы можем повлиять) состоит из смещения и разброса. Разброс уменьшаем с помощью процедуры бэггинга. У неглубоких деревьев малое число параметров, то есть дерево способно запомнить только верхнеуровневые статистики обучающей подвыборки (низкая дисперсия, высокое смещение). Глубокие деревья запоминают подвыборку подробно (высокая дисперсия, низкое смещение).

**Вывод: используем глубокие деревья.**

- **Число признаков** при разбиении вершины в процессе обучения

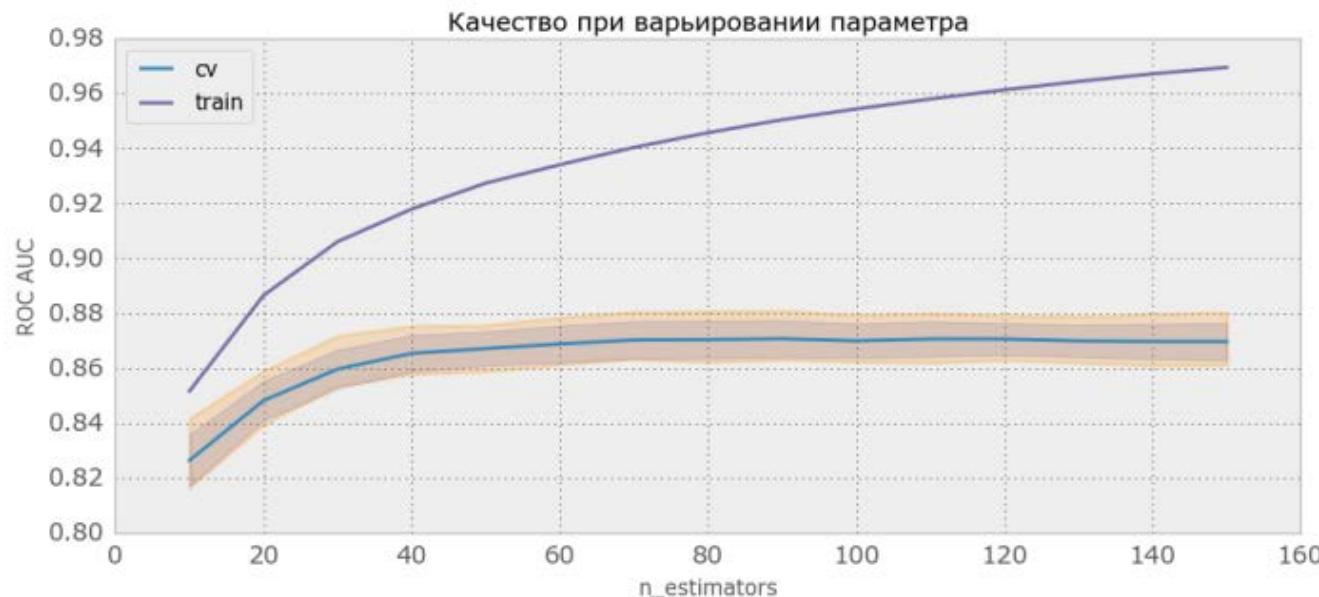
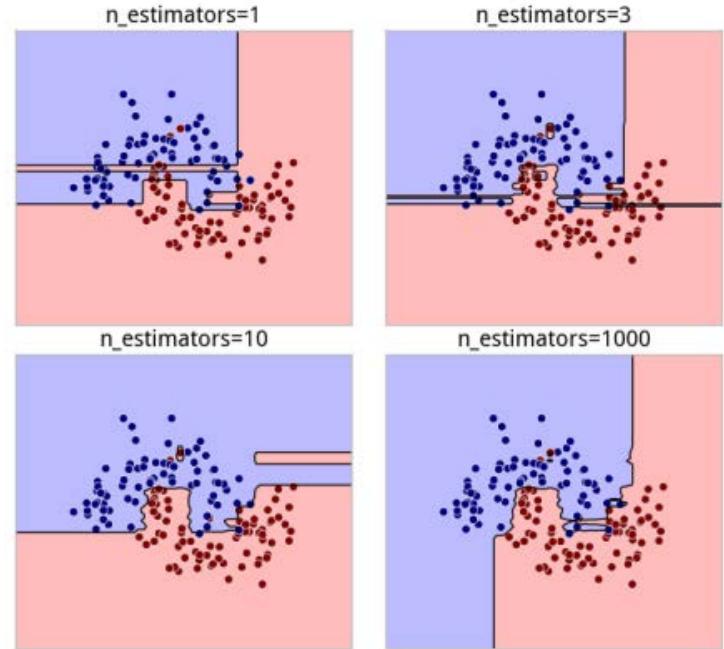
Управляет качеством случайного леса. Чем больше признаков, тем больше корреляция между деревьями и тем меньше эффект от ансамблирования. Чем меньше признаков, тем слабее деревья. Практическая рекомендация — брать корень из числа всех признаков для классификации и треть признаков для регрессии.

- **Число деревьев** в случайном лесу

Можно построить график ошибки от числа деревьев и ограничить размер леса в тот момент, когда ошибка перестанет значимо уменьшаться.

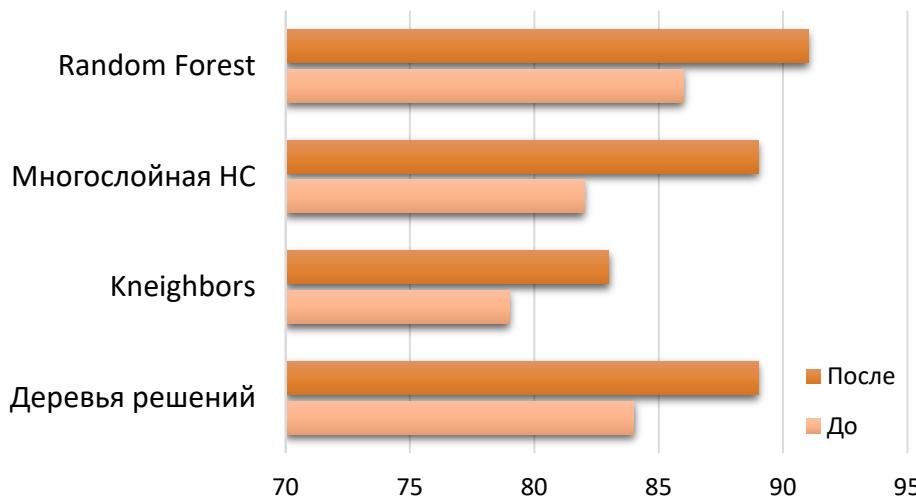
# Число деревьев в случайном лесу

Чем больше деревьев, тем лучше качество, но время настройки и работы также пропорционально увеличиваются. Также можно обратить внимание, что часто при увеличении числа деревьев качество на обучающей выборке повышается (может даже доходить до 100%), а качество на teste выходит на асимптоту.



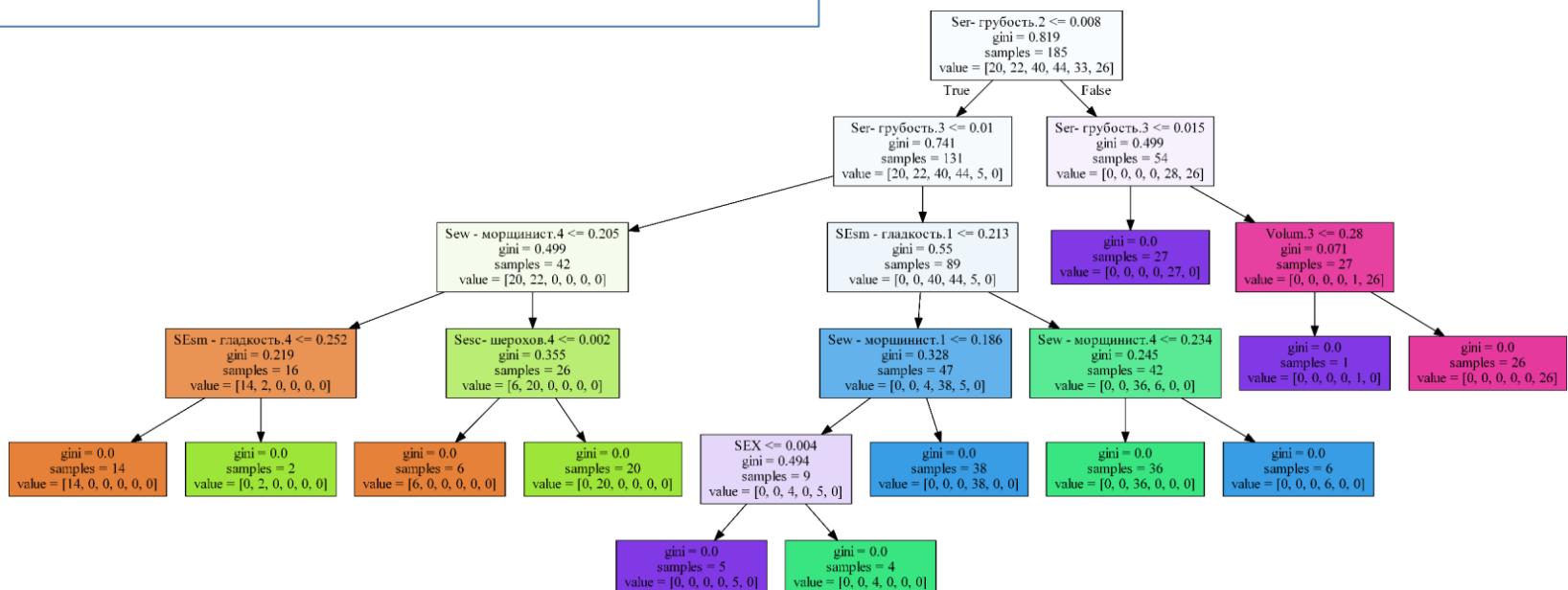
# Пример: Оценка биологического возраста на основе комплексного анализа морфометрических данных в задачах судебной медицины

Точность (accuracy) определения возрастной группы до и после определения пола



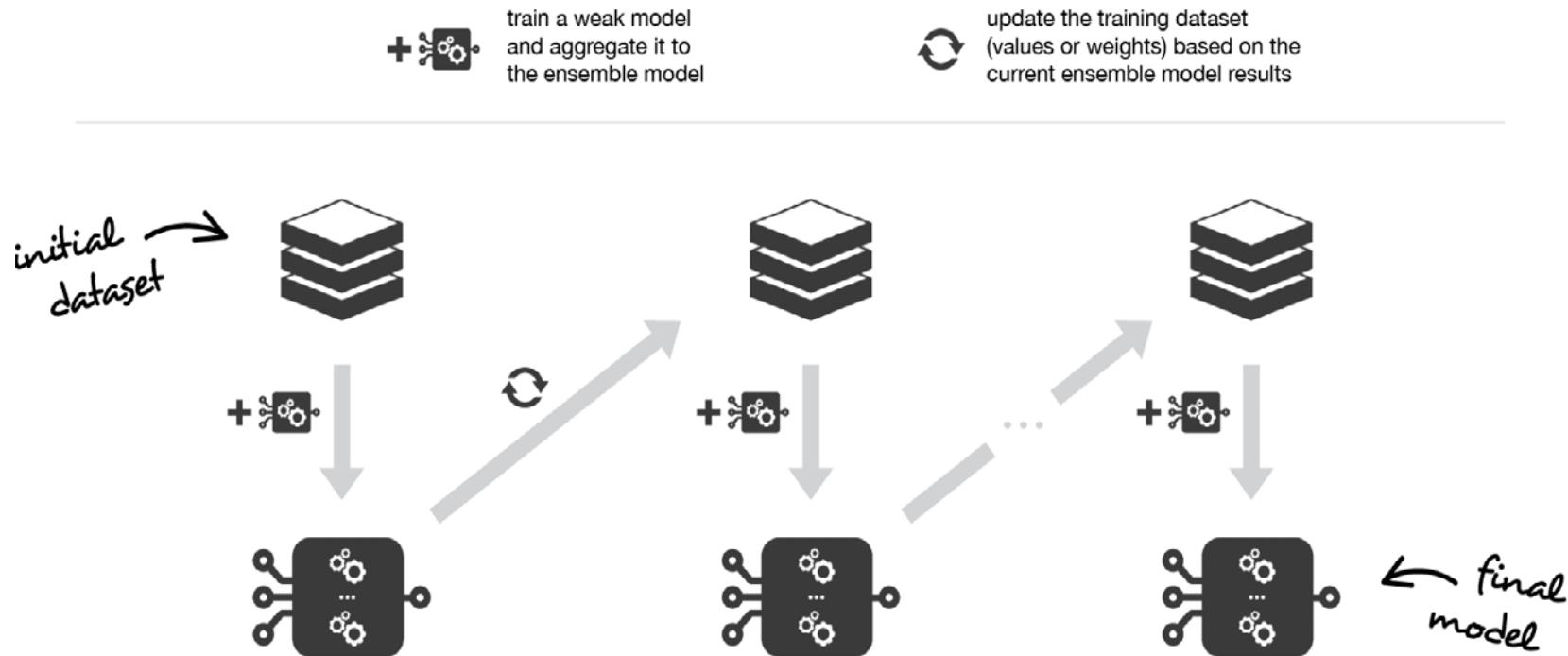
1. Построена модель для определения возрастной группы, использующая данным по коже  
Использован алгоритм Random Forest

2. Построена модель для определения пола, использующая данные по минеральной плотности костей  
Проведен анализ алгоритмов.  
Использована логистическая регрессия.



# Бустинг (boosting)

**Бустинг (boosting)** — это ансамблевый метод, в котором строится множество базовых алгоритмов из одного семейства слабых предсказывающих моделей, объединяющихся затем в более сильную модель. Отличие состоит в том, что в бэггинге и случайному лесу базовые алгоритмы учатся независимо и параллельно, а в бустинге — последовательно. Каждый следующий базовый алгоритм в бустинге обучается так, чтобы уменьшить общую ошибку всех своих предшественников, т.е. ошибку текущего ансамбля.



# AdaBoost (Adaptive Boosting)

Алгоритм, предложенный Йоавом Фройндом и Робертом Шапире, может использоваться в сочетании с несколькими алгоритмами классификации для улучшения их эффективности за счет объединения их в ансамбль. Является адаптивным в том смысле, что каждый следующий ансамбль классификаторов строится по объектам, неверно классифицированным предыдущими комитетами.

AdaBoost — алгоритм построения «сильного» классификатора

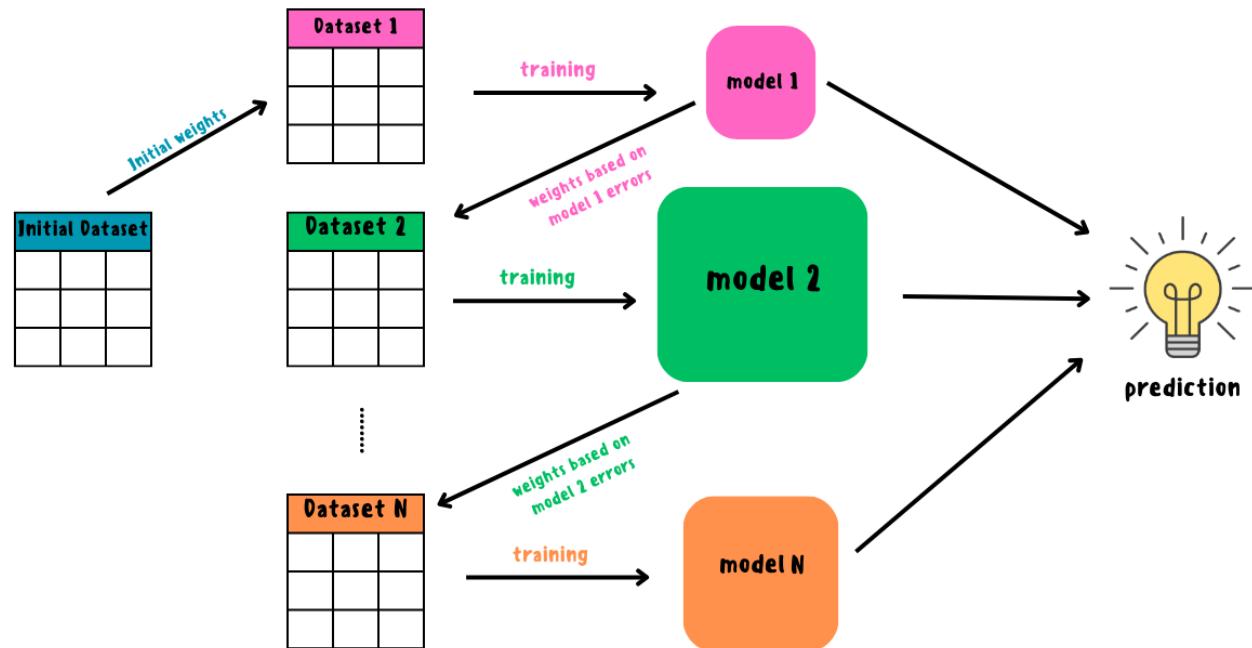
$$H(x) = \text{sign}(f(x)),$$

где

$$f(x) = \sum_{t=1}^T \alpha_t h_t(x)$$

является линейной комбинацией «слабых» классификаторов

$$h_t(x): \chi \rightarrow \{-1, +1\}.$$



## Основная идея

AdaBoost вызывает слабые классификаторы в цикле  $t = 1, \dots, T$ . После каждого вызова обновляется распределение весов  $D_t$ , которые отвечают за важность объектов обучающего множества для классификации. На каждой итерации веса каждого неверно классифицированного объекта возрастают, таким образом, новый комитет классификаторов «фокусирует своё внимание» на этих объектах.

# Алгоритм AdaBoost

Дано:  $(x_1, y_1), \dots, (x_m, y_m)$  где  $x_i \in X, y_i \in Y = \{-1, +1\}$ .

1. Инициализировать веса  $D_1(i) = \frac{1}{m}, i = 1, \dots, m$ .

2. Для каждого  $t = 1, \dots, T$ :

- Используя выборку и распределение весов обучить слабый классификатор  $h_t(x): \chi \rightarrow \{-1, +1\}$  который минимизирует взвешенную ошибку классификации:

$$h_t = \arg \min_{h_j \in H} \epsilon_j, \text{ где } \epsilon_j = \sum_{i=1}^m D_t(i) [y_i \neq h_j(x_i)]$$

- Выбрать  $\alpha_t \in \mathbb{R}, \alpha_t = \frac{1}{2} \ln \left( \frac{1-\epsilon_t}{\epsilon_t} \right)$ , где  $\epsilon_t$  взвешенная ошибка классификатора  $h_t$ .

- Обновить веса примеров:  $D_{t+1}(i) = \frac{D_t(i) \cdot \exp(-\alpha_t y_i h_t(x_i))}{Z_t}$ ,

где  $Z_t$  является нормализующим параметром, чтобы  $\sum_{i=1}^m D_{t+1}(i) = 1$ .

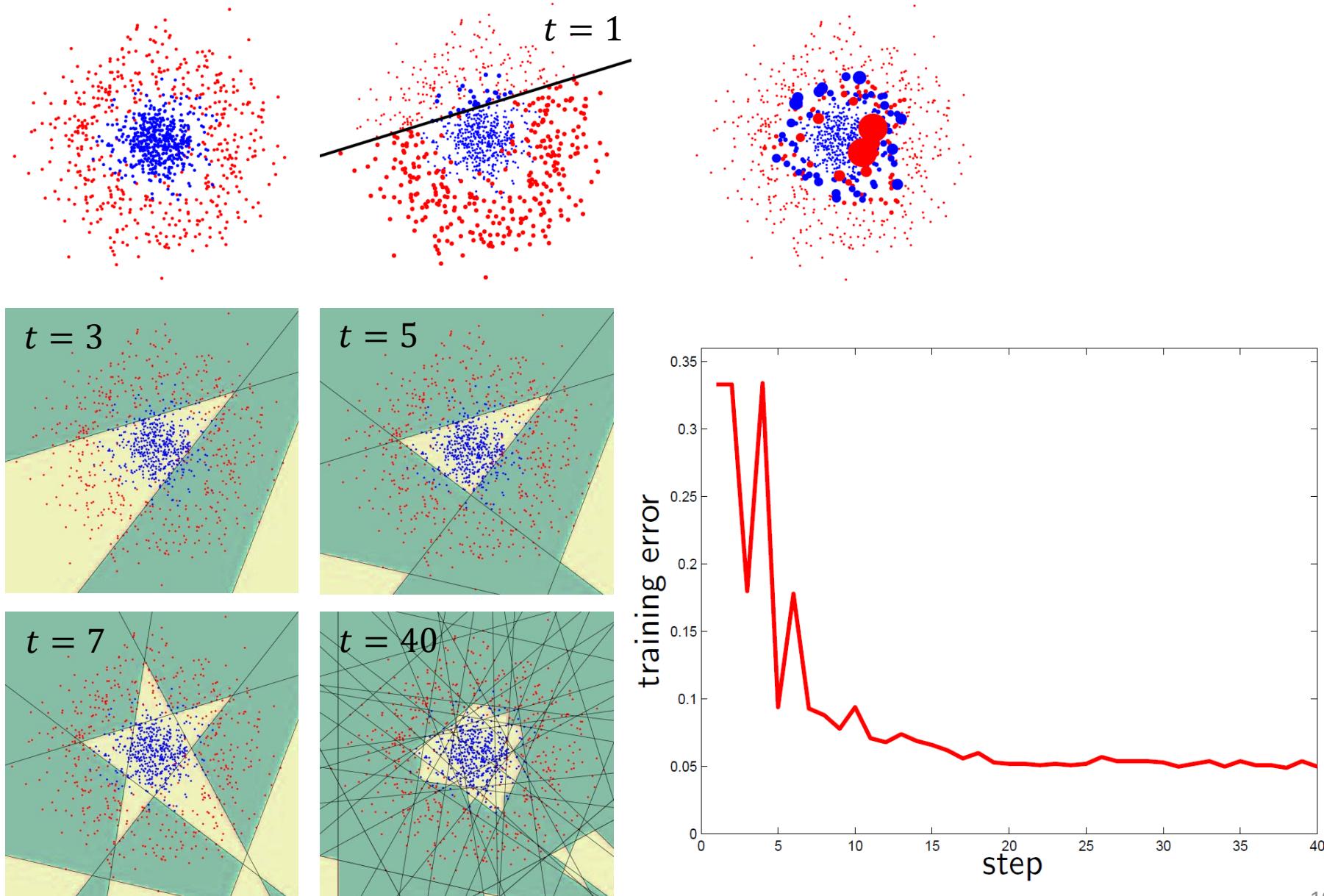
Выражение для обновления  $D_t$  должно быть выбрано таким образом, чтобы выполнялось условие:

$$\exp(-\alpha_t y_i h_t(x_i)) \begin{cases} < 1, & y(i) = h_t(x) \\ > 1, & y(i) \neq h_t(x) \end{cases}$$

3. Рассчитать значение результирующего классификатора:  $H(x) = \operatorname{sign}(\sum_{t=1}^T \alpha_t h_t(x))$ .

Таким образом, после выбора оптимального классификатора  $h_t$  для распределения  $D_t$ , объекты  $x_i$ , которые классификатор  $h_t$  идентифицирует корректно, имеют веса меньшие, чем те, которые идентифицируются некорректно. Следовательно, когда алгоритм тестирует классификаторы на распределении  $D_{t+1}$ , он будет выбирать классификатор, который лучше идентифицирует объекты неверно распознаваемые предыдущим классификатором.

# Визуализация AdaBoost

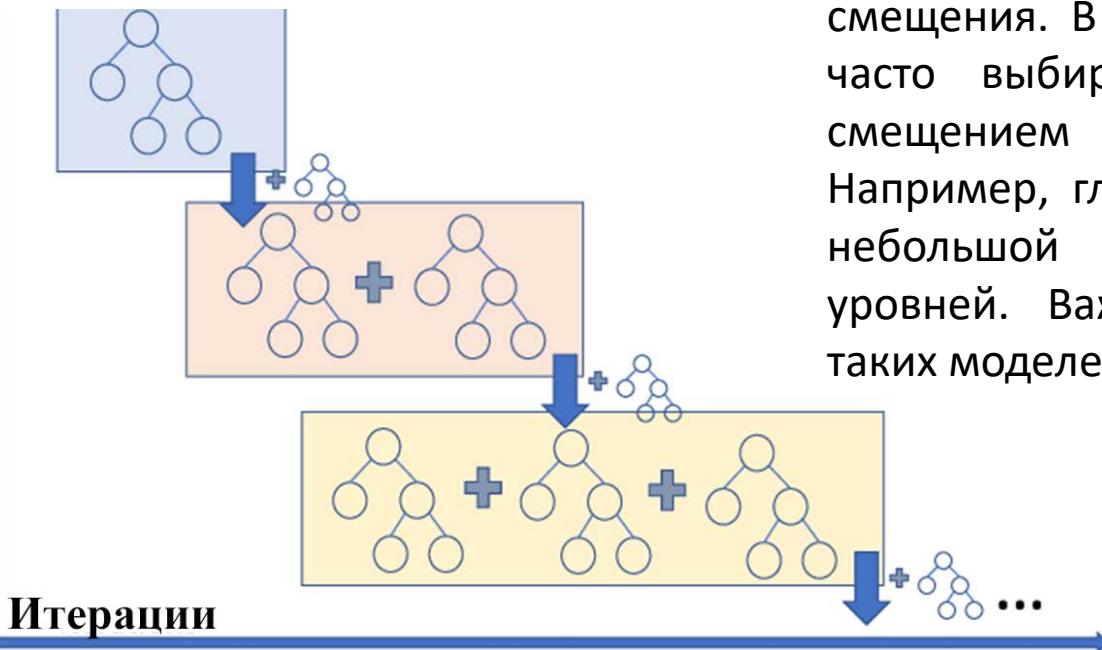


# Градиентный бустинг над решающими деревьями

Бустинг, использующий деревья решений в качестве базовых алгоритмов - **Gradient Boosting on Decision Trees (GBDT)**. Отлично работает на выборках с «табличными», неоднородными данными. Такой бустинг способен эффективно находить нелинейные зависимости в данных различной природы. Широко применяется во многих конкурсах по машинному обучению и задачах из индустрии (поисковом ранжировании, рекомендательных системах, таргетировании рекламы, предсказании погоды, пункта назначения такси и многих других).

Основная цель бустинга — уменьшение смещения. В качестве базовых алгоритмов часто выбирают алгоритмы с высоким смещением и небольшим разбросом. Например, глубина деревьев должна быть небольшой — обычно не больше 2-3 уровней. Важной причиной для выбора таких моделей — скорость обучения.

Ошибка



# Бустинг - Идея

Рассмотрим задачу регрессии с квадратичной функцией потерь:

$$\mathcal{L}(y, x) = \frac{1}{2} \sum_{i=1}^N (y_i - a(x_i))^2 \rightarrow \min$$

Для решения будем строить композицию из  $K$  базовых алгоритмов

$$a(x) = a_K(x) = b_1(x) + \dots + b_K(x)$$

Если мы обучим единственное решающее дерево, то качество такой модели, скорее всего, будет низким. Однако о построенном дереве мы знаем, на каких объектах оно давало точные предсказания, а на каких ошибалось.

Попробуем использовать эту информацию и обучим еще одну модель. Допустим, что предсказание первой модели на объекте  $x_l$  на 10 больше, чем необходимо (т.е.  $b_1(x_l) = y_l + 10$ ). Если бы мы могли обучить новую модель, которая на  $x_l$  будет выдавать ответ -10, то сумма ответов этих двух моделей на объекте  $x_l$  в точности совпала бы с истинным значением:

$$b_1(x_l) + b_2(x_l) = (y_l + 10) + (-10)$$

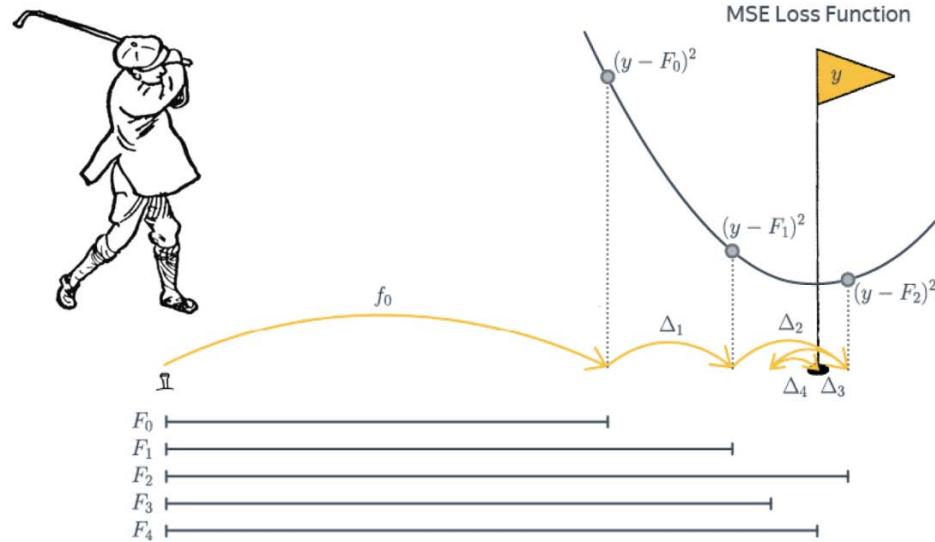
Другими словами, если вторая модель научится предсказывать разницу между реальным значением и ответом первой, то это позволит уменьшить ошибку композиции.

В реальности вторая модель тоже не сможет обучиться идеально, поэтому обучим третью, которая будет «компенсировать» неточности первых двух. Будем продолжать так, пока не построим композицию из  $K$  алгоритмов.

# Бустинг – Аналогии

- **Гольфист.**

Цель – загнать мяч в лунку с координатой  $y_{ball}$ . Положение мяча здесь – ответ композиции  $a(x_{ball})$ . Каждым ударом гольфиста переводит мяч из текущего положения  $a_k(x_{ball})$  в положение  $a_{k+1}(x_{ball})$  и это та поправка, которую вносит очередной базовый алгоритм в композицию.



Если гольфист все делает правильно, то функция потерь будет уменьшаться:

$$\mathcal{L}(y, a_{k+1}(x)) < \mathcal{L}(y, a_k(x))$$

то есть мяч постепенно будет приближаться к лунке.

- **Ряд Тейлора**

Бесконечно дифференцируемую функцию  $f(x)$  на интервале  $x \in (a - R, a + R)$  можно представить в виде бесконечной суммы степенных функций:

$$f(x) = \sum_{n=0}^{\infty} \frac{f^{(n)}(a)}{n!} (X - A)^n$$

Одна, самая первая степенная функция в разложении, очень грубо приближает  $f(x)$ . Прибавляя к ней следующую, мы получим более точное приближение. Каждая следующая элементарная функция увеличивает точность приближения, но менее заметна в общей сумме.

# Задача регрессии: Формальное описание – Шаг 1

Рассмотрим тот же пример с задачей регрессии и квадратичной функцией потерь:

$$\mathcal{L}(y, x) = \frac{1}{2} \sum_{i=1}^N (y_i - a(x_i))^2 \rightarrow \min$$

Для решения также будем строить композицию из  $K$  базовых алгоритмов семейства  $B$ :

$$a(x) = a_K(x) = b_1(x) + \dots + b_K(x)$$

В качестве базовых алгоритмов выберем семейство  $B$  решающих деревьев некоторой фиксированной глубины.

Используя известные нам методы построения решающих деревьев, обучим алгоритм  $b_1(x) \in B$ , который наилучшим образом приближает целевую переменную:

$$b_1(x) = \underset{b \in B}{\operatorname{argmin}} \mathcal{L}(y, b(x))$$

Построенный алгоритм  $b_1(x)$ , скорее всего, работает не идеально. Более того, если базовый алгоритм работает слишком хорошо на обучающей выборке, то высока вероятность переобучения (низкий уровень смещения, но высокий уровень разброса). Далее вычислим, насколько сильно отличаются предсказания этого дерева от истинных значений:

$$s_i^1 = y_i - b_1(x_i)$$

# Задача регрессии: Формальное описание – Шаг 2

Теперь мы хотим скорректировать  $b_1(x)$  с помощью  $b_2(x)$ ; в идеале так, чтобы  $b_2(x)$  идеально предсказывал величины  $s_i^1$ , ведь в этом случае

$$a_2(x_i) = b_1(x_i) + b_2(x_i) = b_1(x_i) + s_i^1 = b_1(x_i) + (y_i - b_1(x_i)) = y_i$$

Найти совершенный алгоритм, скорее всего, не получится, но по крайней мере мы можем выбрать из семейства наилучшего представителя для такой задачи. Итак, второе решающее дерево будет обучаться предсказывать разности  $s_i^1$ :

$$b_2(x) = \underset{b \in B}{\operatorname{argmin}} \mathcal{L}(s^1, b(x))$$

Ожидается, что композиция из двух таких моделей  $a_2(x_i) = b_1(x_i) + b_2(x_i)$  станет более качественно предсказывать целевую переменную  $y$ .

Далее рассуждения повторяются до построения всей композиции. На  $k$ -ом шаге вычисляется разность между правильным ответом и текущим предсказанием композиции из  $k-1$  алгоритмов:

$$s_i^{k-1} = y_i - \sum_{i=1}^{k-1} b_{k-1}(x_i) = y_i - a_{k-1}(x_i)$$

Затем  $k$ -ый алгоритм учится предсказывать эту разность:

$$b_k(x) = \underset{b \in B}{\operatorname{argmin}} \mathcal{L}(s^{k-1}, b(x))$$

а композиция в целом обновляется по формуле:

$$a_k(x) = a_{k-1}(x) + b_k(x)$$

Обучение  $K$  базовых алгоритмов завершает построение композиции.

# Обобщение на другие функции потерь

Отметим важное свойство функции потерь в рассмотренном выше примере с регрессией. Для этого посчитаем производную функции потерь по предсказанию  $z = a_k(x_i)$  модели для  $i$ -го объекта:

$$\frac{\partial \mathcal{L}(y_i, z)}{\partial z} \Big|_{z=a_k(x_i)} = \frac{\partial}{\partial z} \frac{1}{2} (y_i - z)^2 \Big|_{z=a_k(x_i)} = a_k(x_i) - y_i$$

Видим, что разность, на которую обучается  $k$ -й алгоритм, выражается через производную:

$$s_i^k = y_i - a_k(x_i) = -\frac{\partial \mathcal{L}(y_i, z)}{\partial z} \Big|_{z=a_k(x_i)}$$

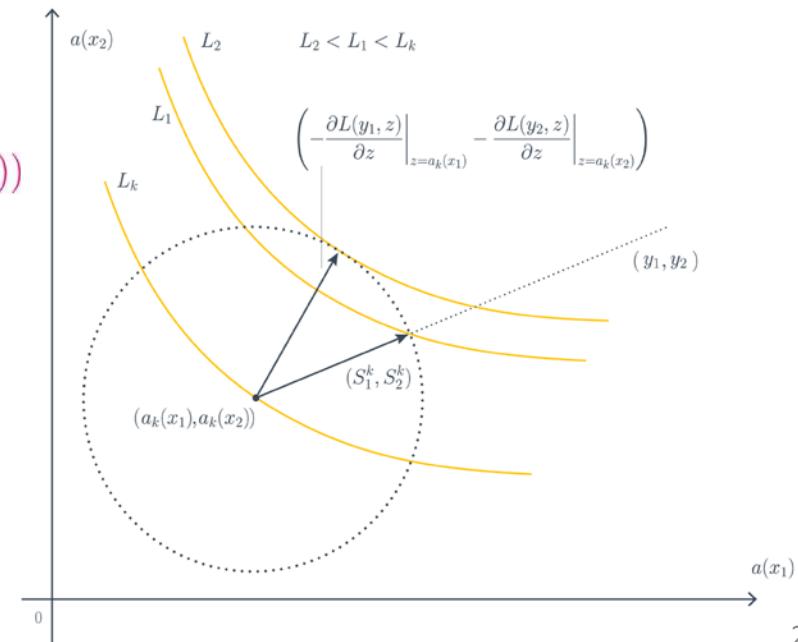
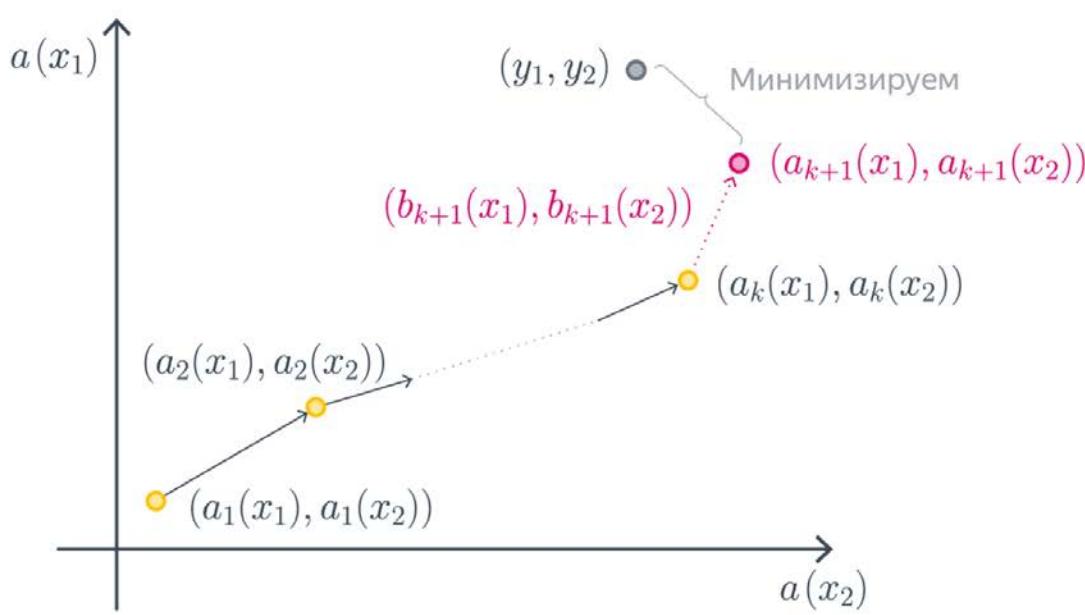
Таким образом, для каждого объекта  $x_i$  очередной алгоритм в бустинге обучается предсказывать антиградиент функции потерь по предсказанию модели  $-\frac{\partial \mathcal{L}(y_i, z)}{\partial z}$  в точке  $a_k(x_i)$  предсказания текущей части композиции на объекте  $x_i$ .

# Почему же это важно?

Это наблюдение позволяет обобщить подход построения бустинга на произвольную дифференцируемую функцию потерь. Для этого мы заменяем обучение на разность  $s_i^k$  обучением на антиградиент функции потерь  $(-\partial L(y_i, z))$ , где

$$g_i^k = \left. \frac{\partial \mathcal{L}(y_i, z)}{\partial z} \right|_{z=a_k(x_i)}$$

Обучение композиции можно представить (вспомните аналогию с гольфистом) как перемещение предсказания из точки  $(a_k(x_1), a_k(x_2), \dots, a_k(x_N))$  в точку  $(a_{k+1}(x_1), a_{k+1}(x_2), \dots, a_{k+1}(x_N))$ . В конечном итоге мы ожидаем, что точка  $(a_K(x_1), a_K(x_2), \dots, a_K(x_N))$  будет располагаться как можно ближе к точке с истинными значениями  $(y_1, y_2, \dots, y_N)$ .



# Обучение базового алгоритма

При построении очередного базового алгоритма  $b_{k+1}$  мы решаем задачу регрессии с таргетом, равным антиградиенту функции потерь исходной задачи на предсказании  $a_k = b_1 + \dots + b_k$ .

Теоретически можно воспользоваться любым методом построения регрессионного дерева. Важно выбрать оценочную функцию  $S$ , которая будет показывать, насколько текущая структура дерева хорошо приближает антиградиент. Её нужно будет использовать для построения критерия ветвления:

$$|R| \cdot S(R) - |R_{right}| \cdot S(R_{right}) - |R_{left}| \cdot S(R_{left}) \rightarrow \max$$

где  $S(R)$  – значение функции  $S$  в вершине  $R$ ,

$S(R_{right}), S(R_{left})$  – значения в левой и правой дочерних вершинах  $R$  после добавления предиката,

$|\cdot|$  – количество элементов, пришедших в вершину.

В итоге обучение базового алгоритма проходит в два шага:

1. по **функции потерь** вычисляется целевая переменная для обучения следующего базового алгоритма:

$$g_i^k = \left. \frac{\partial \mathcal{L}(y_i, z)}{\partial z} \right|_{z=a_k(x_i)}$$

2. строится регрессионное дерево на обучающей выборке  $(x_i, -g_i^k)$ , минимизирующее выбранную **оценочную функцию**.

# Темп обучения (learning rate)

Обучение ансамбля с помощью градиентного бустинга может привести к переобучению, если базовые алгоритмы слишком сложные. Например, если сделать решающие деревья слишком глубокими (более 10 уровней), то при обучении бустинга ошибка на обучающей выборке даже при малом числе базовых моделей  $K$  может приблизиться к нулю, но на тестовой выборке ошибка будет большой.

Существует два решения этой проблемы:

- Во-первых, необходимо упростить базовую модель, уменьшив глубину дерева (либо примерив какие-либо ещё техники регуляризации).
- Во-вторых, мы можем ввести параметр, называемый **темпом обучения (learning rate)**  $\eta \in (0,1]$ :

$$a_{k+1}(x) = a_k(x) + \eta b_{k+1}(x)$$

Присутствие этого параметра означает, что каждый базовый алгоритм вносит относительно небольшой вклад во всю композицию: если расписать сумму целиком, она будет иметь вид

$$a_{k+1}(x) = b_1(x) + \eta b_2(x) + \dots + \eta b_{k+1}(x)$$

Значение параметра обычно определяется эмпирически по входным данным. Темп обучения связан с количеством итераций градиентного бустинга. Чем меньше learning rate, тем больше итераций потребуется сделать для достижения того же качества на обучающей выборке.

# Значимость признаков (Feature importance)

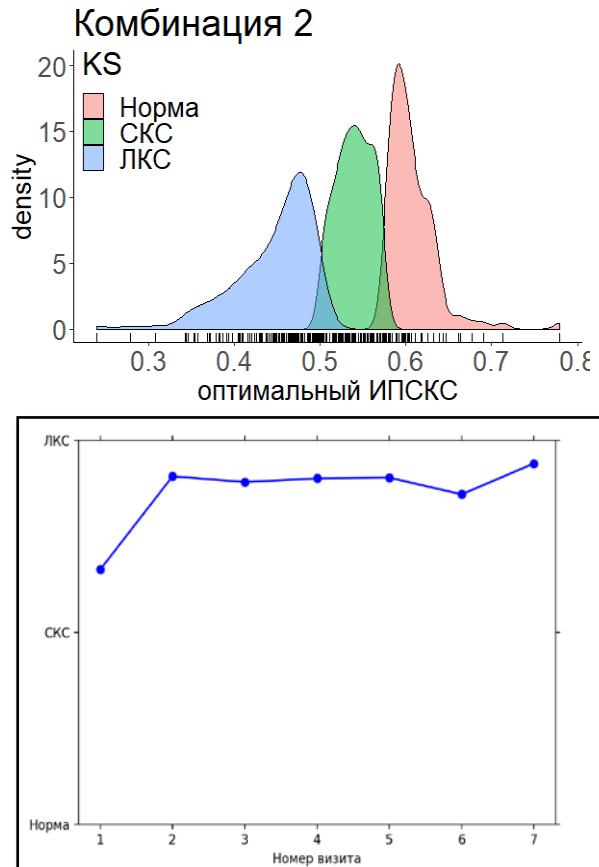
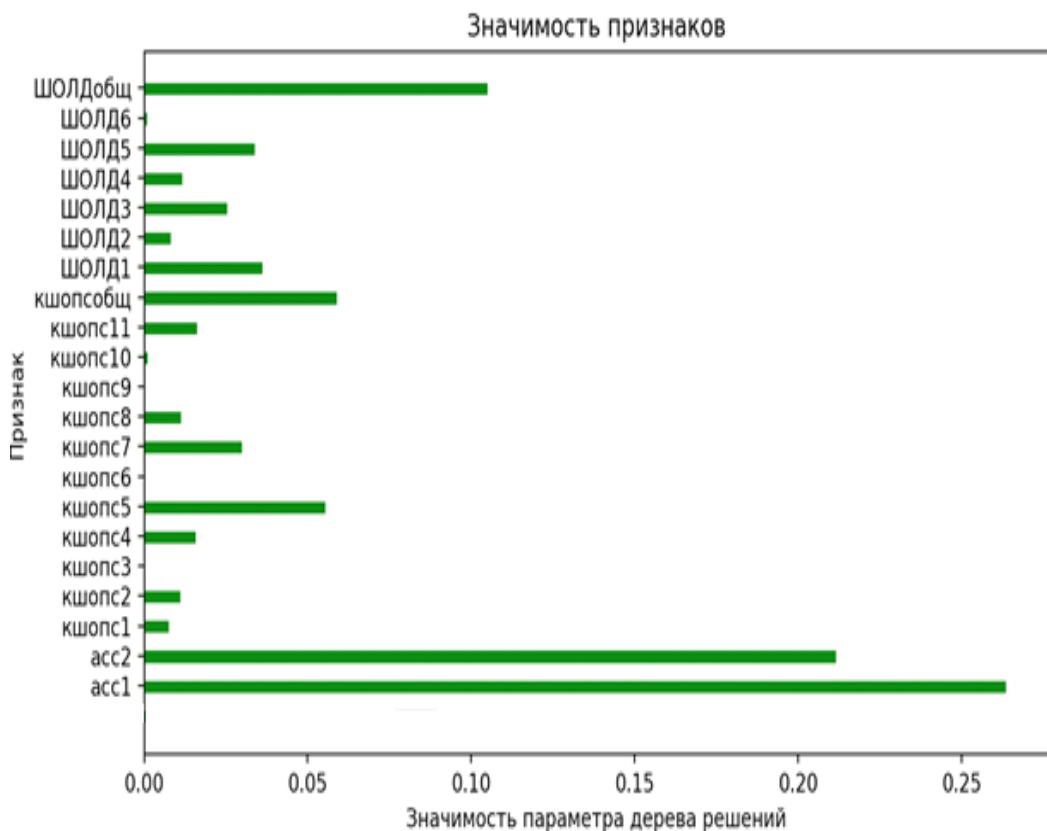
Признаки, используемые в верхней части дерева, влияют на окончательное предсказание для большей доли обучающих объектов, чем признаки, попавшие на более глубокие уровни.

Таким образом, ожидаемая доля обучающих объектов, для которых происходило ветвление по данному признаку, может быть использована в качестве оценки его относительной важности для итогового предсказания. Усредняя полученные оценки важности признаков по всем решающим деревьям из ансамбля, можно уменьшить дисперсию такой оценки и использовать ее для отбора признаков.

Этот метод известен как **MDI** (**mean decrease in impurity**).

# Исследование роли возрастных, сердечно-сосудистых и нейродегенеративных факторов в развитии субъективного и легкого когнитивного снижения в среднем и пожилом возрасте

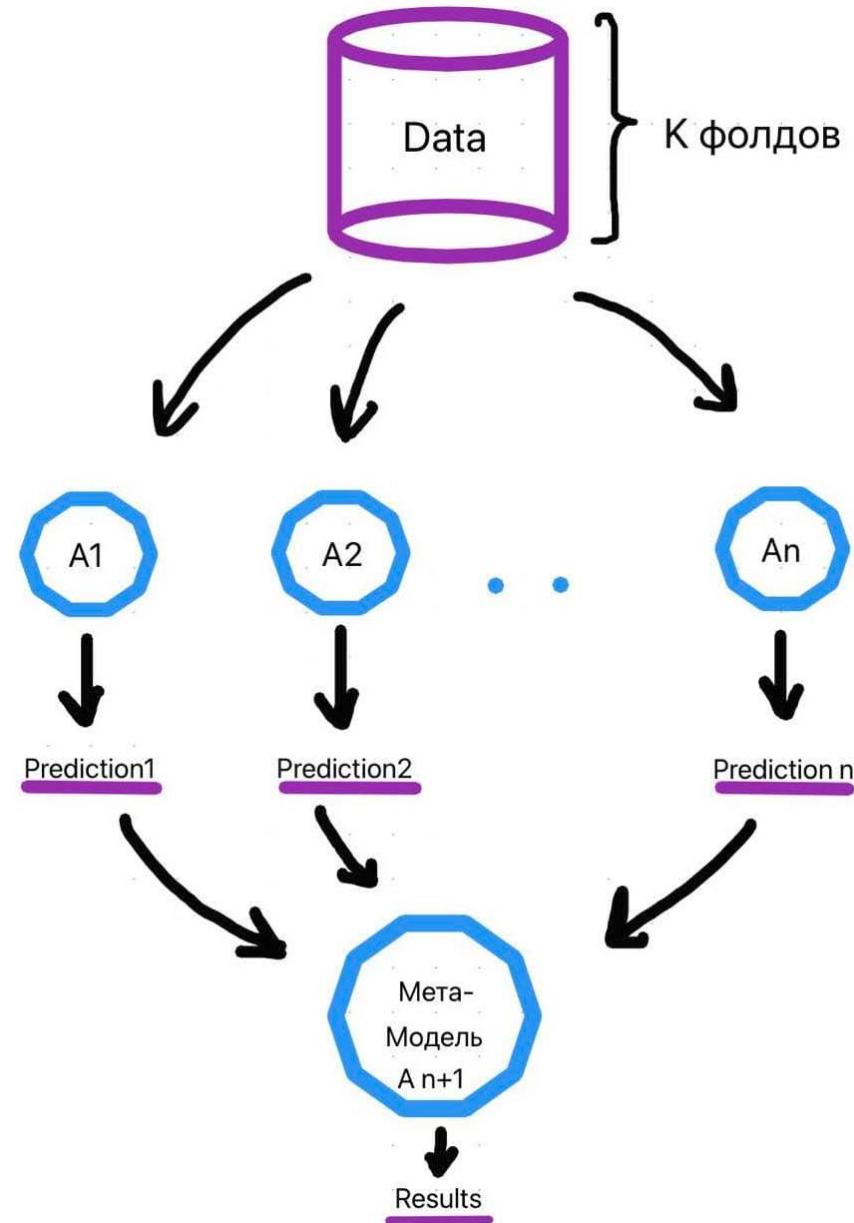
Разработаны алгоритмы расчета интегрального показателя степени когнитивного снижения (ИПСКС) и классификаторов на их основе, (оптимизированный статистический классификатор и классификатор на основе методов машинного обучения).



# Стекинг (stacking)

Алгоритм ансамблирования, основные отличия которого от предыдущих состоят в следующем:

- может использовать алгоритмы разного типа, а не только из какого-то фиксированного семейства;
- результаты базовых алгоритмов объединяются в один с помощью обучаемой мета-модели, а не с помощью какого-либо обычного способа агрегации (суммирования или усреднения).



# Стекинг (stacking) - Обучение

- Общая выборка разделяется на тренировочную и тестовую.
- Тренировочная выборка делится на n фолдов. Затем эти фолды перебираются тем же способом, что используется при кросс-валидации: на каждом шаге фиксируются (n-1) фолдов для обучения базовых алгоритмов и один — для их предсказаний (вычисления мета-факторов). Такой подход нужен для того, чтобы можно было использовать всё тренировочное множество, и при этом базовые алгоритмы не переобучались.
- На полученных мета-факторах обучается мета-модель. Кроме мета-факторов, она может принимать на вход и параметры из исходного датасета. Выбор зависит от решаемой задачи.



**Спасибо за внимание**

# **Методы машинного обучения**

*Лекция 10*

**Обучение с подкреплением**

# Обучение с подкреплением

**Обучение с подкреплением (*reinforcement learning*)** - способ машинного обучения, при котором испытуемая система (агент) обучается, взаимодействуя с некоторой средой. Роль объектов играют пары «ситуация, принятное решение», ответами являются значения функционала качества, характеризующего правильность принятых решений (реакцию среды). При обучении с подкреплением, в отличии от обучения с учителем, не предоставляются верные пары "входные данные - ответ", а принятие субоптимальных решений (дающих локальный экстремум) не ограничивается явно.

Обучение с подкреплением пытается найти компромисс между исследованием неизученных областей и применением имеющихся знаний (***exploration vs exploitation***).

Примеры прикладных задач: формирование инвестиционных стратегий, автоматическое управление технологическими процессами, самообучение роботов, и т.д.

Осуществляется поиск субоптимальных решений или стратегий того, как агент должен действовать в окружении, чтобы максимизировать некоторый **долговременный** выигрыш.

Маленький мальчик заходит в парикмахерскую. Парикмахер сразу же его узнаёт и говорит своим клиентам: «Смотрите, это самый глупый мальчик среди всех на свете! Сейчас я вам докажу!». В одной руке парикмахер держит доллар, в другой 25 центов. Зовёт мальчика, тот подходит и выбирает 25 центов. Все смеются, мальчик уходит. По дороге обратно, мальчика догоняет один из смеявшихся и спрашивает:

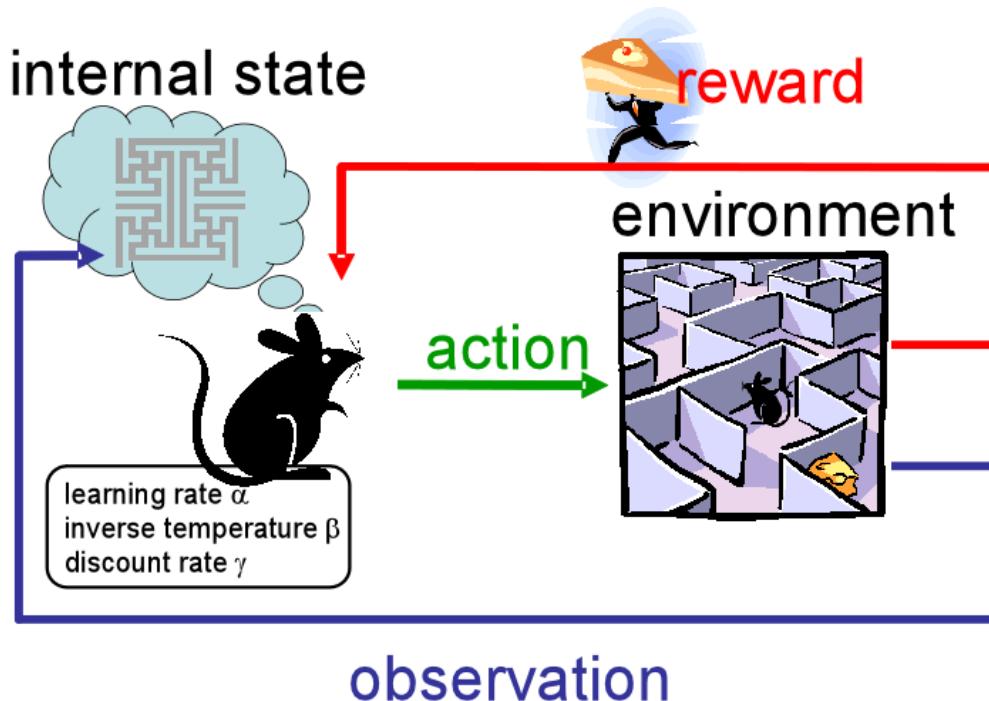
— А почему всё-таки ты выбрал 25 центов, а не 1 доллар?

— Потому что в тот день, когда я выберу 1 доллар, игра будет окончена.

# Среда и агент

В обучении с подкреплением существует агент (*agent*), который взаимодействует с окружающей средой (*environment*), предпринимая действия (*actions*). Окружающая среда дает награду (*reward*) за эти действия, а агент продолжает их предпринимать (обучение методом проб и ошибок).

В искусственном интеллекте под термином **интеллектуальный агент** понимаются сущности, получающие информацию через систему сенсоров о состоянии управляемых ими процессов и осуществляющие влияние на них через систему актуаторов, при этом их реакция рациональна в том смысле, что процессы выполняемые ими способствуют достижению определённых параметров.

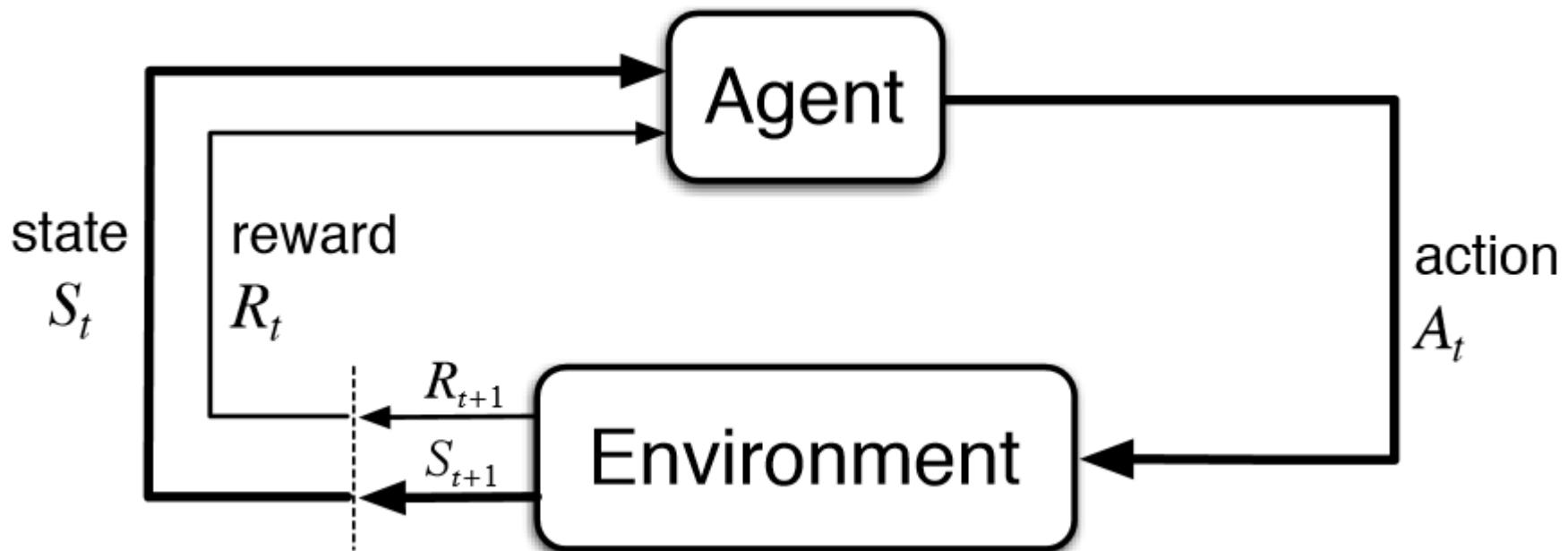


# Необходимые термины в Reinforcement Learning

- **Агент (agent):** Наша система, которая выполняет действия в среде, чтобы получить некоторую награду.
- **Среда (environment, e):** сценарий/окружение, с которым должен взаимодействовать агент.
- **Награда (reward, R):** немедленный возврат, который предоставляется агенту, после выполнения определенного действия или задачи. Является положительной или отрицательной величиной.
- **Состояние (state, s):** Состояние относится к текущему положению, возвращаемому средой.
- **Политика (policy, π):** стратегия, которая применяется агентом для принятия решения о следующем действии на основе текущего состояния.
- **Стоймость (value, V):** награда, которая ожидается в долгосрочной перспективе. По сравнению с краткосрочным вознаграждением, принимаем во внимание скидку (discount).
- **Функция полезности состояния (value function):** определяет размер переменной, которой является общая сумма награды.
- **Модель среды (Model of the environment):** имитатор поведения окружающей среды (демо-версия модели). Помогает определить, как будет вести себя среда.
- **Значение Q или значение полезности действия (Q):** значение Q очень похоже на value (V). Главное различие в том, что Q принимает текущее действие в качестве дополнительного параметра.

# Простейшая постановка задачи

- На каждом шаге агент может находиться в состоянии  $s \in S$ .
- На каждом шаге агент выбирает из имеющегося набора действий некоторое действие  $a \in A$ .
- Окружающая среда сообщает агенту, какую награду  $r$  он за это получил и в каком состоянии  $s'$  после этого оказался.



# Пример взаимодействия среды и агента

- Взаимодействие:
  - Среда: Агент, ты в состоянии №1. Есть 5 возможных действий.
  - Агент: Выбираю действие 2.
  - Среда: Вознаграждение 2 единицы. Новое состояние № 5. Есть 2 возможных действия.
  - Агент: Выбираю действие 1.
  - Среда: Вознаграждение -5 единиц. Новое состояние № 1. Есть 5 возможных действий.
  - Агент: Выбираю действие 4.
  - Среда: Вознаграждение 14 единиц. Новое состояние № 3. Есть 3 возможных действия.
- Результат:

Агент успел вернуться в состояние 1 и исследовать ранее незнакомую опцию 4, получив за это существенную награду.

# Exploration vs Exploitation

- Каждый алгоритм должен и изучать окружающую среду, и пользоваться своими знаниями, чтобы максимизировать прибыль.
- Та или иная стратегия может быть хороша, но вдруг она не оптимальная? Как достичь оптимального соотношения между исследованием нового и использованием имеющихся знаний?
- Обучение с подкреплением пытается найти компромисс между исследованием неизученных областей и применением имеющихся знаний, т.е. *exploration vs exploitation*. Эта проблематика всегда присутствует в обучении с подкреплением.

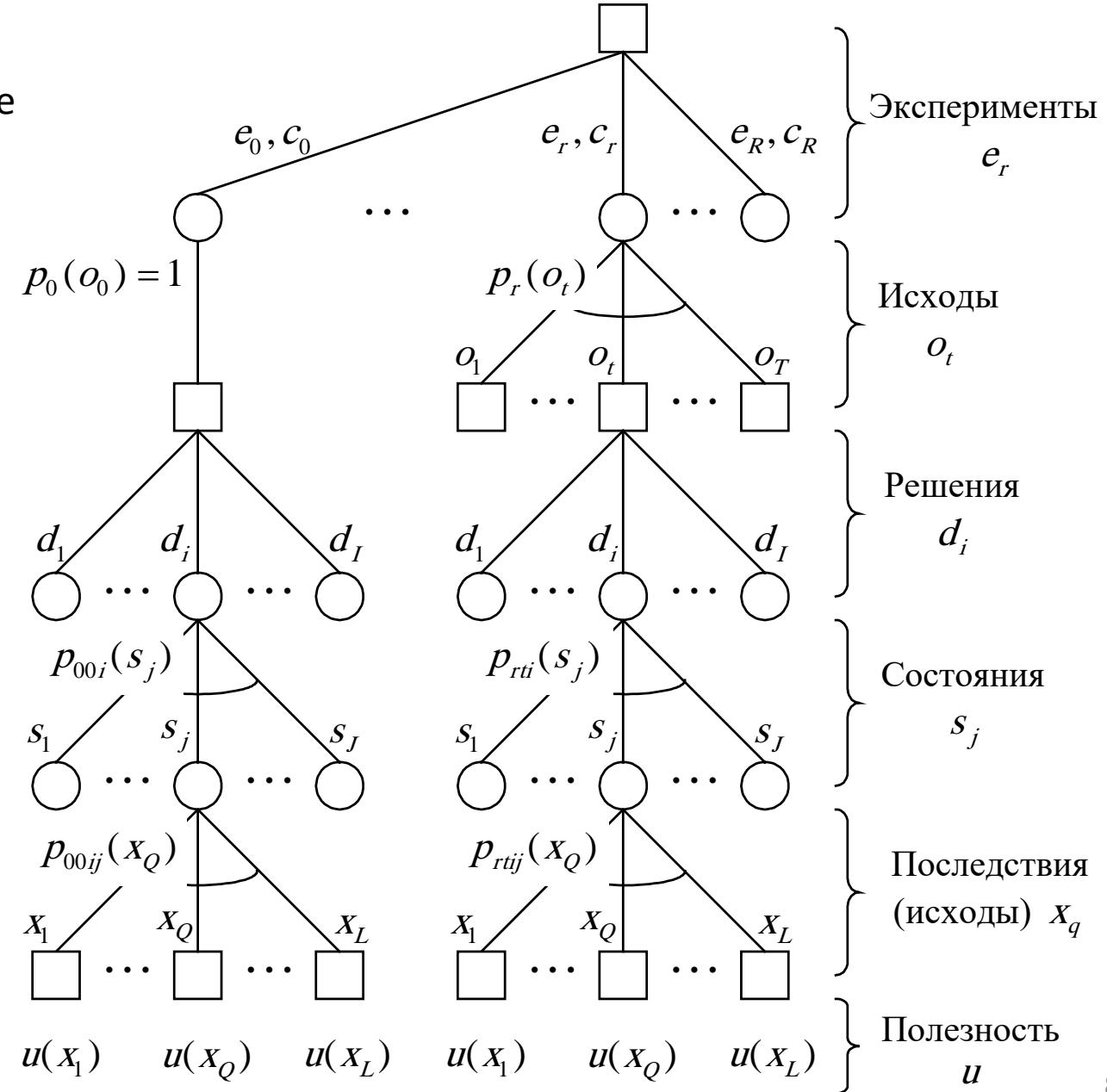
# Дерево принятия решений

Представление задачи принятия решений в виде дерева решений.

Два типа узлов:

- узлы решений, обозначенные квадратами;
- узлы возможностей, обозначенные окружностями.

Анализ дерева решений осуществляется снизу вверх, используя принцип максимизации ожидаемой полезности.



# Анализ дерева решений

В узлах возможностей с помощью полученного для данного узла распределения вероятностей вычисляется ожидаемая полезность. Для любого узла решений лицо, принимающее решение (ЛПР), выбирает альтернативу, которая приводит к наибольшей ожидаемой полезности, и приписывает полученную полезность узлу решений.

Так обозначим через  $\bar{u}_{rtij}$  ожидаемую полезность проведенного эксперимента  $e_r$  при наблюдаемом исходе  $o_t$ , выбранном решении  $d_i$  и внешних условиях  $s_j$ , а  $\bar{u}_{rtijl}$  является функцией полезности последствий  $u(x_l)$ , где индекс  $l$  обозначает соответствующее последствие.

Для дискретных задач:

$$\bar{u}_{rtij} = \sum_x u(x_l) p_{rtij}(x_l).$$

Аналогично ожидаемая полезность выбранного эксперимента  $e_r$ , наблюдаемого исхода  $o_t$  и выбранного решения  $d_i$  равна:

$$\bar{u}_{rti} = \sum_s \bar{u}_{rtij} p_{rti}(s_j).$$

В узле решений выбирается решение  $d_i$ , приводящее к максимальной ожидаемой полезности. Следовательно:

$$\bar{u}_{rt} = \max_{d_i} (\bar{u}_{rti}).$$

Выражение для ожидаемой полезности эксперимента  $e_r$ :

$$\bar{u}_r = \sum_o \bar{u}_{rt} p_r(o_t).$$

Наилучшим является эксперимент  $e_{r*}$ , который позволяет получить максимальное значение ожидаемой полезности из соотношения:

$$\bar{u}_{r*} = \max_{e_r} (\bar{u}_r).$$

Пусть выбран эксперимент  $r*$  и реализовался исход  $o_t$ ; тогда оптимальное решение  $d_{i*}$  определяется с помощью выражения:

$$\bar{u}_{r*t i*} = \max_{d_i} (\bar{u}_{r*t i}).$$

Любую задачу принятия решений можно представить последовательностью узлов решения и узлов возможностей.

# Марковские процессы

**Марковский процесс** — случайный процесс, эволюция которого после любого заданного значения временного параметра  $t$  не зависит от эволюции, предшествовавшей  $t$ , при условии, что значение процесса в этот момент фиксировано («будущее» процесса зависит от «прошлого» лишь через «настоящее»).

**Марковское свойство** — в теории вероятностей и статистике термин, который относится к памяти случайного процесса.

Стochasticкий процесс обладает марковским свойством, если условное распределение вероятностей будущих состояний процесса зависит только от нынешнего состояния, а не от последовательности событий, которые предшествовали этому. Процесс, обладающий этим свойством, называется марковским процессом.

**Для марковских цепей с дискретным временем.** В случае, если  $S$  является дискретным множеством состояний и время дискретно, то марковское свойство может быть сформулировано следующим образом:

$$\begin{aligned}\mathbb{P}(X_n = x_n | X_{n-1} = x_{n-1}, X_{n-2} = x_{n-2}, \dots, X_0 = x_0) = \\ \mathbb{P}(X_n = x_n | X_{n-1} = x_{n-1})\end{aligned}$$

# Марковский процесс принятия решений (МПР)

## *Markov decision process (MDP)*

Спецификация задачи последовательного принятия решений для полностью наблюдаемой среды с марковской моделью перехода и дополнительными вознаграждениями. Слово марковский в названии отражает выполнение марковского свойства для таких процессов. Такой процесс служит математической основой для моделирования последовательного принятия решений в ситуациях, где результаты частично случайны и частично под контролем лица, принимающего решения.

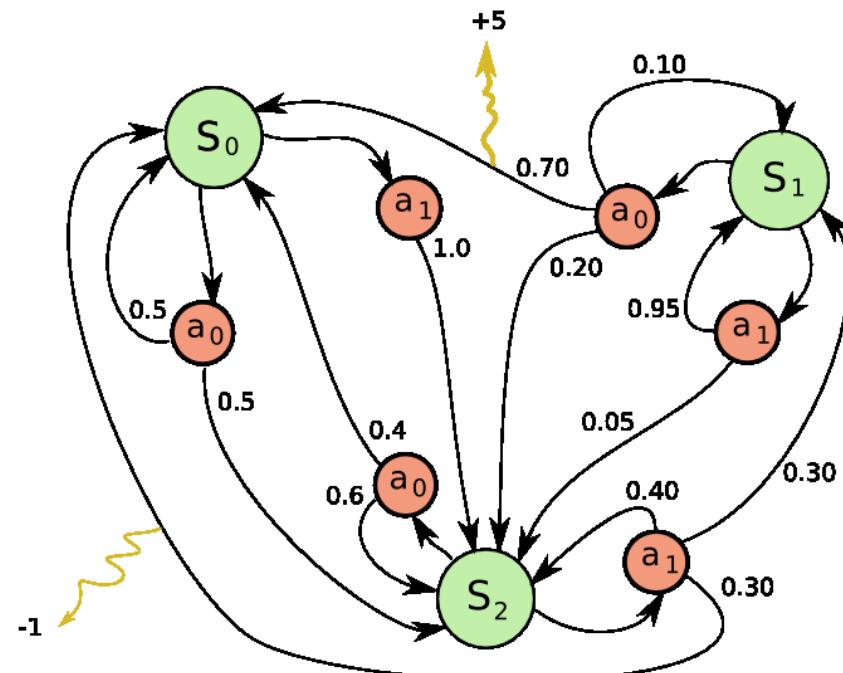
Эта спецификация используется во множестве областей, включая робототехнику, автоматизированное управление, экономику, производство, а также в качестве основы обучения с подкреплениями.

# МППР - Определение

Чтобы определить марковский процесс принятия решений, нужно задать 4-кортеж  $(S, A, P(\cdot, \cdot), R(\cdot, \cdot))$ , где:

- $S$  - конечное множество состояний;
- $A$  - конечное множество действий (часто представляется в виде множеств  $A_s$ , действий доступных из состояния  $s$ );
- $P_a(s, s') = \Pr(s_{t+1} = s' | s_t = s, a_t = a)$  - вероятность, что действие  $a$  в состоянии  $s$  во время  $t$  приведет в состояние  $s'$  ко времени  $t + 1$ ;
- $R_a(s, s')$  - вознаграждение, получаемое после перехода в состояние  $s'$  из состояния  $s$ , при совершении действия  $a$ .

**Стратегия  $\pi$**  — функция (в общем случае распределение вероятностей), сопоставляющая состоянию действие. Такой Марковский процесс принятия решений можно рассматривать, как Марковскую цепь.



# МППР - Цель оптимизации

Решить марковский процесс принятия решений означает найти стратегию, максимизирующую "вознаграждение" (функцию ценности) - оптимальную стратегию. Самая простая функция ценности это математическое ожидание формального ряда:

$$E \left[ \sum_{t=0}^{\infty} R_{a_t}(s_t, s_{t+1}) \right]$$

где  $a_t = \pi(s_t)$ , а математическое ожидание берётся в соответствии с  $s_{t+1} \sim P_{a_t}(s_t, \cdot)$ . Такую функцию можно использовать если гарантируется, что ряд сходится, а значит наличие терминального состояния, где  $P_a(s, s) = 1$  и  $R_a(s, s) = 0$ .

Если же сходимость ряда не гарантируется, то обычно делают одно из двух:

- Рассматривают только конечное число слагаемых:

$$E \left[ \sum_{t=0}^{N} R_{a_t}(s_t, s_{t+1}) \right]$$

- Вводят коэффициент обесценивания (дисконтирования)  $\gamma \in [0,1]$ :

$$E \left[ \sum_{t=0}^{\infty} \gamma^t R_{a_t}(s_t, s_{t+1}) \right]$$

На практике второй вариант более гибкий, так как учитывает более долгосрочную перспективу и чаще используется именно он.

# МППР - Функции полезности

Для максимизации ряда  $E\left[\sum_{t=0}^{\infty} R_{a_t}(s_t, s_{t+1})\right]$  вводят две функции полезности:

- Функция полезности состояния:

$$V_{\pi}(s) = E \left[ \sum_{t=0}^{\infty} \gamma^t R_{a_t}(s_t, s_{t+1}) | s_0 = s, a_t = \pi(s_t) \right]$$

- Функция полезности действия:

$$Q_{\pi}(s, a) = E \left[ \sum_{t=0}^{\infty} \gamma^t R_{a_t}(s_t, s_{t+1}) | s_0 = s, a_0 = a, a_t = \pi(s_t) \forall t \geq 1 \right]$$

где математическое ожидание берётся в соответствии с  $s_{t+1} \sim P_{a_t}(s_t, \cdot)$ .

А также их максимумы по всем стратегиям:

$$V_*(s) = \max_{\pi} V_{\pi}(s) \text{ и } Q_*(s, a) = \max_{\pi} Q_{\pi}(s, a)$$

Можно доказать, что эти функции также являются функциями полезности состояния и полезности действия соответственно, а также, что они достигаются на детерминированной стратегии. Заметим, что по функции  $Q_*$  можно восстановить её стратегию, которая будет оптимальной.

Для определения оптимальной стратегии используется отношение порядка на множестве стратегий.

$$\pi_1 \leq \pi_2 \Leftrightarrow \forall V_{\pi_1}(s) \leq V_{\pi_2}(s), s \in S$$

Наибольшая стратегия называется оптимальной.

# Постановка задачи обучения с подкреплением

## Составные части:

- множество состояний среды (*states*)  $S$ ;
- множество действий (*actions*)  $A$ ;
- множество вещественнонозначных скалярных "выигрышей" (*rewards*)  $R$ .

## Игра агента со средой:

- инициализация стратегии  $\pi_1(a|s)$  и состояния среды  $s_1$
- для всех  $t = 1 \dots T$ :
  - агент выбирает действие  $a_t \sim \pi_1(a|s_t)$
  - среда генерирует награду  $r_{t+1} \sim p(r|a_t, s_t)$  и новое состояние  $s_{t+1} \sim p(s|a_t, s_t)$
  - агент корректирует стратегию  $\pi_{t+1}(a|s)$

Таким образом в произвольный момент времени  $t$  агент характеризуется состоянием  $s_t \in S$  и множеством возможных действий  $A(s_t)$ . Выбирая действие  $a \in A(s_t)$ , он переходит в состояние  $s_{t+1}$  и получает выигрыш  $r_{t+1}$ . Основываясь на таком взаимодействии с окружающей средой, агент, обучающийся с подкреплением, должен выработать стратегию  $\pi: S \rightarrow A$ , которая максимизирует величину:

- $R = r_0 + r_1 + \dots + r_n$  в случае марковского процесса принятия решений (МППР), имеющего терминальное состояние;
- $R = \sum_t \gamma^t r_t$  для МППР без терминальных состояний (где  $0 \leq \gamma \leq 1$  – дисконтирующий множитель для "предстоящего выигрыша").

## Марковское свойство (МППР):

$$P(s_{t+1} = s', r_{t+1} = r | s_t, a_t, r_t, s_{t-1}, a_{t-1}, r_{t-1}, \dots, s_1, a_1) = P(s_{t+1} = s', r_{t+1} = r | s_t, a_t)$$

МППР называется финитным, если  $|A| < \infty$ ,  $|S| < \infty$ .

Таким образом, обучение с подкреплением особенно хорошо подходит для решения задач, связанных с выбором между долгосрочной и краткосрочной выгодой.

# Подход к решению

Наивный подход к решению этой задачи подразумевает следующие шаги:

- опробовать все возможные стратегии;
- выбрать стратегию с наибольшим ожидаемым выигрышем.

Проблемы:

- количество доступных стратегий может быть велико или бесконечно;
- выигрыши стохастические — чтобы точно оценить выигрыш от каждой стратегии потребуется многократно применить каждую из них.

Решения:

- оценка функций полезности;
- прямая оптимизация стратегий.

Подход с использованием функции полезности использует множество оценок ожидаемого выигрыша только для одной стратегии  $\pi$  (либо текущей, либо оптимальной). При этом пытаются оценить либо ожидаемый выигрыш, начиная с состояния  $s$ , при дальнейшем следовании стратегии  $\pi$ ,

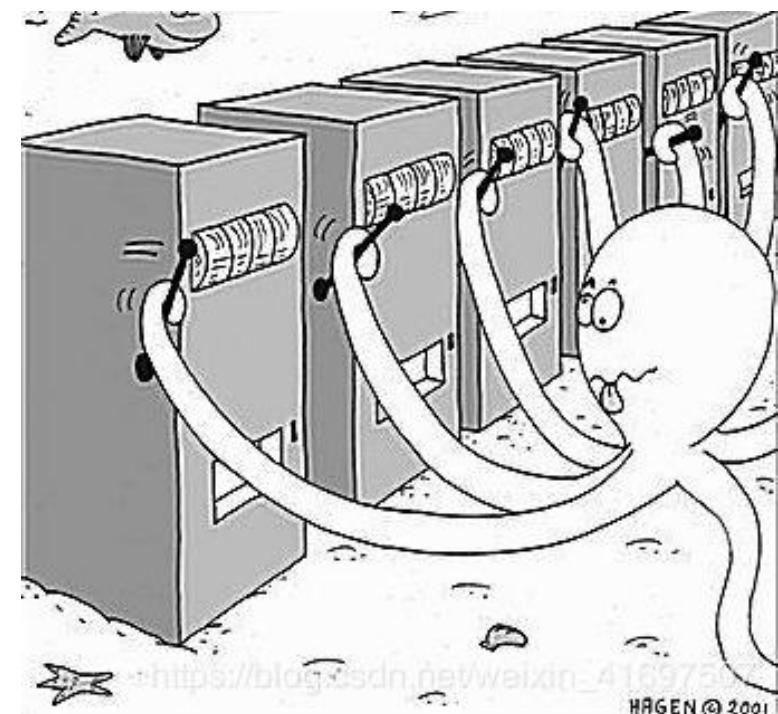
$$V(s) = E[R|s, \pi],$$

либо ожидаемый выигрыш, при принятии решения  $a$  в состоянии  $s$  и дальнейшем соблюдении  $\pi$ ,

$$Q(s, a) = E[R|s, \pi, a].$$

# Задача о многоруком бандите (*The multi-armed bandit problem*)

- Агенты с одним состоянием, т.е. состояние агента не меняется. У него фиксированный набор действий и возможность выбора из этого набора действий.
- Модель: агент в комнате с несколькими игровыми автоматами. У каждого автомата своё ожидание выигрыша.
- Нужно заработать побольше:  
Exploration vs. Exploitation  
(разведка против эксплуатации).
- Жадные и  $\epsilon$ -жадные стратегии  
(greedy &  $\epsilon$ -greedy)



HAGEN © 2001

# Задача о многоруком бандите - Формулировка

$A$  — множество возможных действий (ручек автомата),

$p_a(r)$  — неизвестное распределение награды  $r \in R \quad \forall a \in A$ ,

$\pi_t(a)$  — стратегия агента в момент  $t \quad \forall a \in A$ .

**Игра агента со средой:**

- инициализация стратегии  $\pi_1(a)$ ;
- для всех  $t = 1 \dots T$ :
  - агент выбирает действие (ручку)  $a_t \sim \pi_t(a)$ ;
  - среда генерирует награду  $r_t \sim p_{a_t}(r)$ ;
  - агент корректирует стратегию  $\pi_{t+1}(a)$ .

Средняя награда в  $t$  играх:  $Q_t(a) = \frac{\sum_{i=1}^t r_i[a_i=a]}{\sum_{i=1}^t [a_i=a]} \rightarrow \max$ ,

Ценность действия  $a$ :  $Q^*(a) = \lim_{t \rightarrow \infty} Q_t(a) \rightarrow \max$

$N$ -рукой бандит - на каждом шаге выбираем за какую из  $N$  ручек автомата дернуть. Каждому действию соответствует некоторое распределение (не меняется со временем). Если распределения известны, то стратегия заключается в том, чтобы подсчитать математическое ожидание для каждого из распределений, выбрать действие с максимальным математическим ожиданием и совершать это действие на каждом шаге. Проблема, что распределения неизвестны, однако можно оценить математическое ожидание случайной величины  $\xi$  с неизвестным распределением.

$$E(\xi) = \frac{1}{K} \sum_{k=1}^K \xi_k$$

# Жадная (greedy) стратегия

**Начальные значения:**

- $P_a = 0$  для  $\forall a \in \{1 \dots N\}$  — сколько раз было выбрано действие  $a$ ,
- $Q_a = 0$  для  $\forall a \in \{1 \dots N\}$  — текущая оценка математического ожидания награды для действия  $a$

**На каждом шаге  $t$ :**

- Выбираем действие с максимальной оценкой математического ожидания:

$$a_t = \operatorname{argmax}_{a \in A} Q_a$$

- Выполняем действие  $a_t$  и получаем награду  $R(a_t)$ ;
- Обновляем оценку математического ожидания для действия  $a_t$ :

$$\begin{aligned} P_{a_t} &= P_{a_t} + 1 \\ Q_{a_t} &= Q_{a_t} + \frac{1}{P_{a_t}}(R(a_t) - Q_{a_t}) \end{aligned}$$

**В чём проблема?**

Пусть у нас есть "двурукий" бандит. Первая ручка всегда выдаёт награду равную 1, вторая всегда выдаёт 2. Действуя согласно жадной стратегии мы дёрнем в начале первую ручку, так как в начале оценки математических ожиданий равны нулю, увеличим её оценку до  $Q_1 = 1$ . В дальнейшем всегда будем выбирать первую ручку, а значит на каждом шаге будем получать на 1 меньше, чем могли бы.

# $\epsilon$ -жадная ( $\epsilon$ -greedy) стратегия

Введем параметр  $\epsilon \in (0,1)$ .

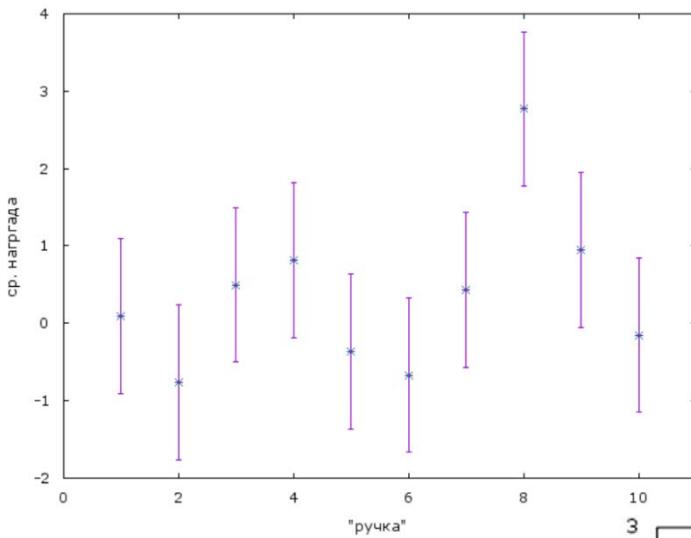
На каждом шаге  $t$ :

- Получим значение  $a$  - случайной величины равномерно распределенной на отрезке  $(0,1)$ ;
- Если  $a \in (0, \epsilon)$ , то выберем действие  $a_t \in A$  случайно и равновероятно, иначе как в жадной стратегии выберем действие с максимальной оценкой математического ожидания;
- Обновляем оценки так же как в жадной стратегии.

Если  $\epsilon = 0$ , то это обычная жадная стратегия. Однако если  $\epsilon > 0$ , то в отличии от жадной стратегии на каждом шаге с вероятностью  $\epsilon$  происходит "исследование" случайных действий.

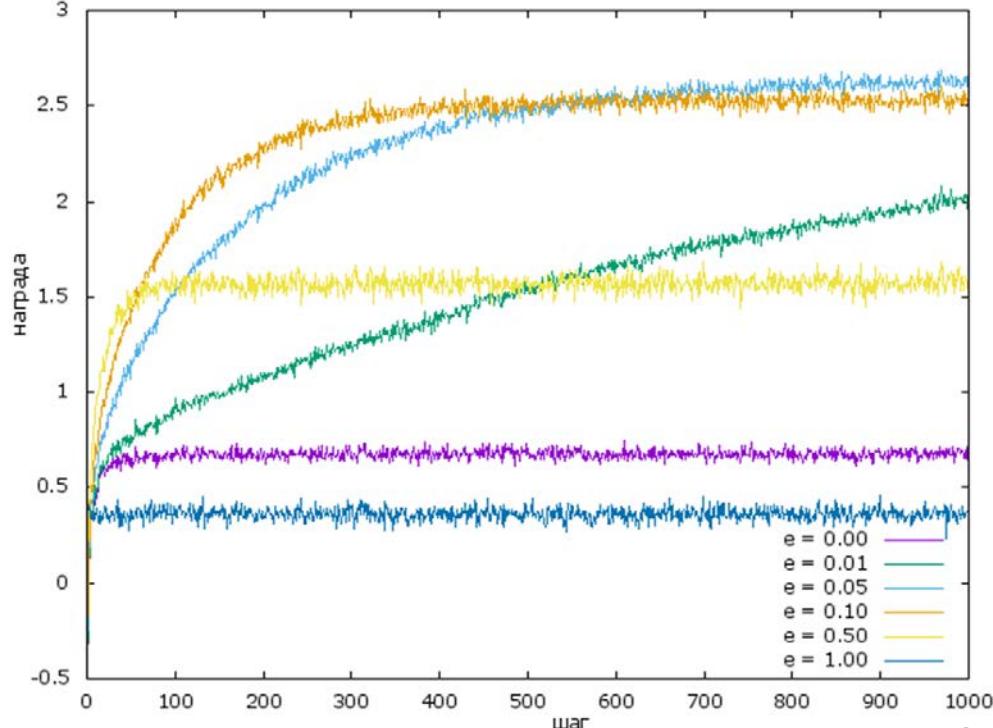
# Пример: $\epsilon$ -жадная стратегия

Ручка	Ср. награда
1	0.089668
2	-0.757752
3	0.497168
4	0.811979
5	-0.367975
6	-0.666402
7	0.432601
8	2.769230
9	0.943522
10	-0.151123



Рассмотрим 10-рукого бандита. Выберем 10 случайных нормально распределенных чисел с центром в нуле и единичной дисперсией:  $E = \{E_a | a = 1, \dots, 10\}$ . Каждой ручке поставим соответствие нормальное распределение с математическим ожиданием из  $E$  и дисперсией 1.

- $\epsilon = 1$  - худший результат, т.е. ручка выбирается случайно и равновероятно.
- $\epsilon = 0$  - “жадная” стратегия находится на предпоследнем месте.
- $\epsilon = 0.5$  - лучше предыдущих, но тратить половину ходов, выбирая ручку случайно; начиная с некоторого момента полученная награда стабилизируется не в максимальном значении.
- $\epsilon = 0.01$  - растет слишком медленно на начальном этапе (слишком мало исследований), вероятно догонит варианты с  $\epsilon=0.05$  и  $\epsilon=0.1$ , но для этого ей надо больше времени.
- $\epsilon = 0.05$  и  $\epsilon = 0.1$  - работают вполне не плохо. Правда достигнуть среднего 2.76923 (т.е. собственно того, которое будет, если дергать только 8-ю ручку) ни та, ни другая не смогли.



# Стратегия Softmax

Основная идея алгоритма *softmax* — уменьшение потерь при исследовании за счёт более редкого выбора действий, которые получали небольшую награду в прошлом. Чтобы этого добиться для каждого действия вычисляется весовой коэффициент на базе которого происходит выбор действия. Чем больше  $Q_t(a)$ , тем больше вероятность выбора  $a$ :

$$\pi_{t+1}(a) = \frac{\exp(Q_t(a)/\tau)}{\sum_{b \in A} \exp(Q_t(b)/\tau)}$$

$\tau \in (0, \infty)$  — параметр, с помощью которого можно настраивать поведение алгоритма.

При  $\tau \rightarrow \infty$  стратегия стремится к равномерной, то есть softmax будет меньше зависеть от значения выигрыша и выбирать действия более равномерно (exploration).

При  $\tau \rightarrow 0$  стратегия стремится к жадной, то есть алгоритм будет больше ориентироваться на известный средний выигрыш действий (exploitation).

Экспонента используется для того, чтобы данный вес был ненулевым даже у действий, награда от которых пока нулевая. Параметр  $\tau$  имеет смысл уменьшать со временем.

# Алгоритм верхнего доверительного интервала (*upper confidence bound* или UCB)

UCB — семейство алгоритмов, которые при выборе действия используют данные не только о среднем выигрыше, но и о том, насколько можно доверять значениям выигрыша.

Также как *softmax* в UCB при выборе действия используется весовой коэффициент, который представляет собой верхнюю границу доверительного интервала (*upper confidence bound*) значения выигрыша:

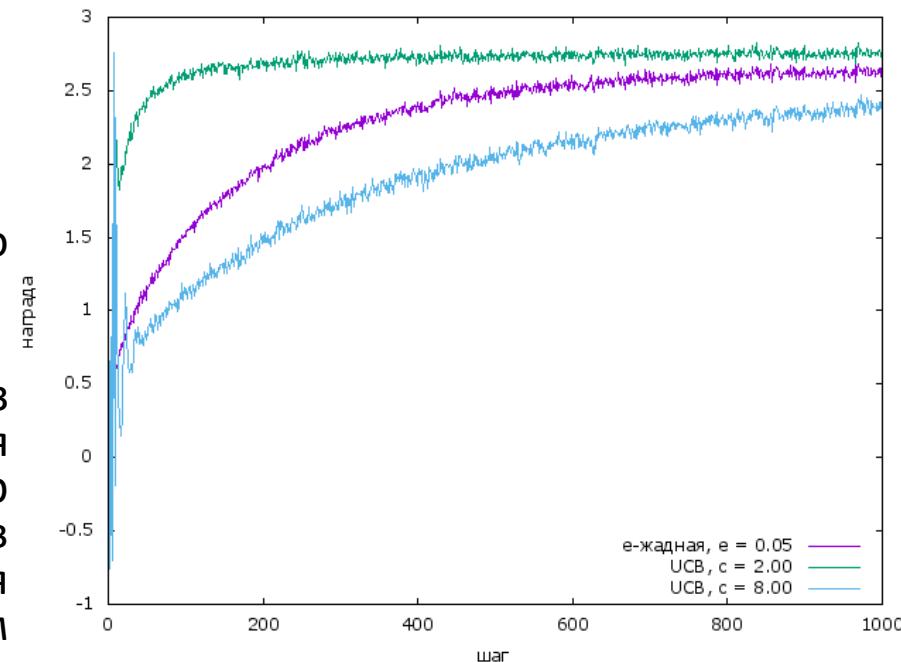
$$a_{t+1} = \operatorname{argmax}_{a=1,\dots,N} \{ Q_t(a) + b_a \}$$

где  $b_a$  — бонусное значение, которое показывает, насколько недоисследовано действие по сравнению с остальными:

$$b_a = c \cdot \sqrt{\frac{\ln(t)}{P_a}} \text{ или } b_a = \sqrt{\frac{2 \cdot \ln \sum_a P_a}{P_a}}$$

$P_a$  — сколько раз было выбрано действие  $a$ ;  
 $Q_t(a)$  — текущая оценка математического ожидания награды для действия  $a$ ;  
 $c$  — коэффициент настройки.

В начале работы алгоритма каждое из действий выбирается по одному разу (для того чтобы можно было вычислить размер бонуса для всех действий). После этого в каждый момент времени выбирается действие с максимальным значением весового коэффициента.



# Q-обучение (Q-learning)

На основе получаемого от среды вознаграждения агент формирует функцию полезности  $Q$ , что впоследствии дает ему возможность уже не случайно выбирать стратегию поведения, а учитывать опыт предыдущего взаимодействия со средой. Преимущество Q-learning — способен сравнивать ожидаемую полезность доступных действий, не формируя модели окружающей среды. Применяется для ситуаций, которые можно представить в виде МППР.

Таким образом, алгоритм это функция качества от состояния и действия:

$$Q: S \times A \rightarrow \mathbb{R}$$

Перед обучением  $Q$  инициализируется случайными значениями. После этого в каждый момент времени  $t$  агент выбирает действие  $a_t$ , получает награду  $r_t$ , переходит в новое состояние  $s_{t+1}$ , которое может зависеть от предыдущего состояния  $s_t$  и выбранного действия, и обновляет функцию  $Q$ . Обновление функции использует взвешенное среднее между старым и новым значениями:

$$Q^{new}(s_t, a_t) \leftarrow \underbrace{(1 - \alpha) \cdot Q(s_t, a_t)}_{\text{old value}} + \underbrace{\alpha}_{\text{learning rate}} \cdot \overbrace{\left( \underbrace{r_t}_{\text{reward}} + \underbrace{\gamma}_{\text{discount factor}} \cdot \underbrace{\max_a Q(s_{t+1}, a)}_{\substack{\text{learned value} \\ \text{estimate of optimal future value}}} \right)}^{\text{learned value}}$$

где  $r_t$ - это награда, полученная при переходе из состояния  $s_t$  в состояние  $s_{t+1}$ , и  $\alpha$  - скорость обучения ( $0 < \alpha \leq 1$ ).

Алгоритм заканчивается, когда агент переходит в терминальное состояние  $s_{t+1}$ .

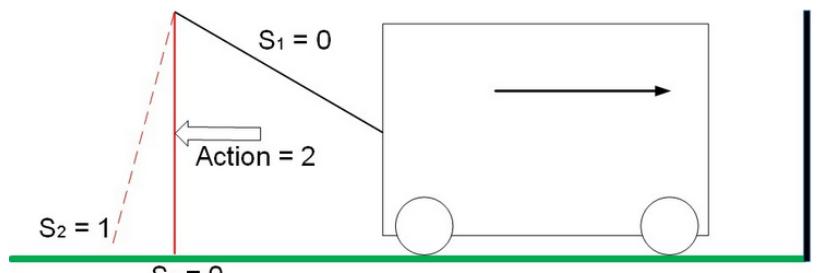
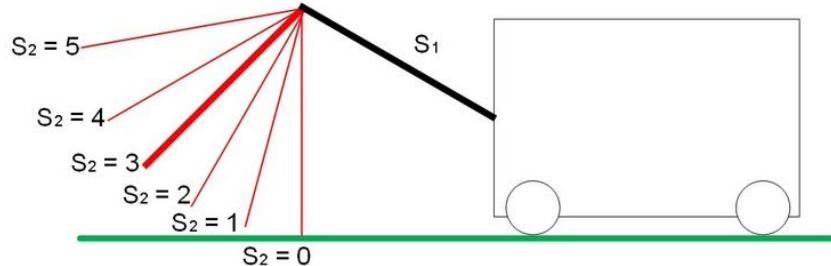
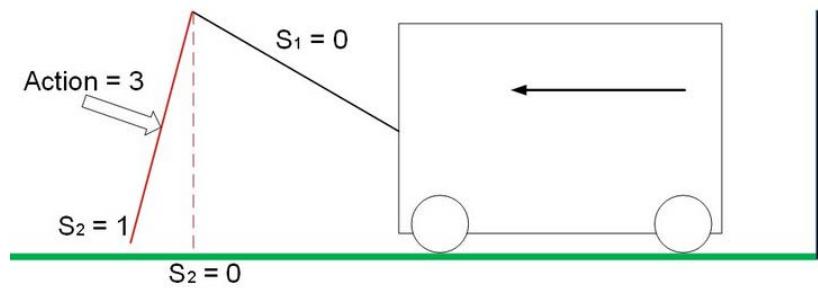
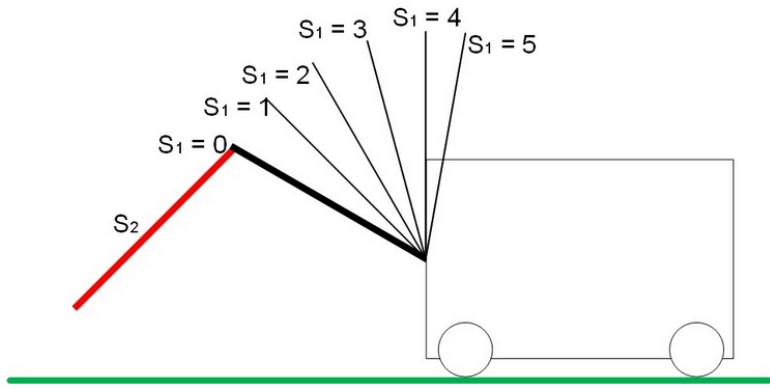
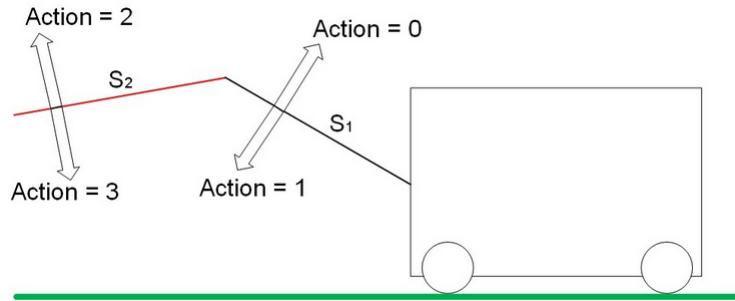
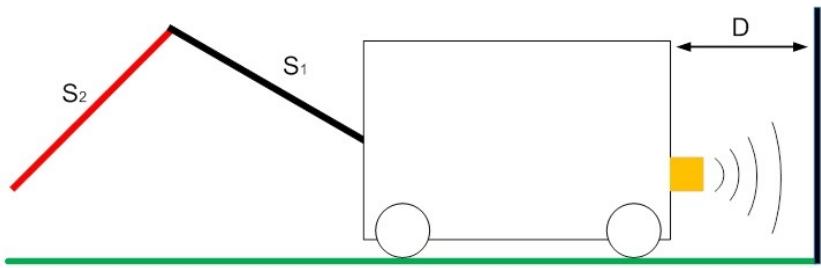
# Алгоритм Q-learning

Обозначения:

- $S$  — множество состояний;  $A$  — множество действий;
- $R = S \times A \rightarrow \mathbb{R}$  — функция награды;  $T = S \times A \rightarrow S$  — функция перехода;
- $\alpha \in [0,1]$  — learning rate (обычно 0.1), чем он выше, тем сильнее агент доверяет новой информации;
- $\gamma \in [0,1]$  — discounting factor, чем он меньше, тем меньше агент задумывается о выгоде от будущих своих действий.

```
fun Q-learning(S, A, R, T, α, γ):
    for s ∈ S:
        for a ∈ A:
            Q(s, a) = rand()
    while Q is not converged:
        s = ∀s ∈ S
        while s is not terminal:
            π(s) = argmaxaQ(s, a)
            a = π(s)
            r = R(s, a)
            s' = T(s, a)
            Q(s', a) = (1 - α)Q(s', a) + α(r + γ maxa'Q(s', a'))
            s = s'
    return Q
```

# Пример Q-learning – Тележка 1



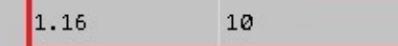
# Пример Q-learning – Тележка 2

Index	S1	S2	A0	A1	A2	A3
0	0	0	10	10	10	10
1	0	1	10	10	10	10
2	0	2	10	10	10	10
3	0	3	10	10	10	10
4	0	4	10	10	10	10
5	0	5	10	10	10	10
6	1	0	10	10	10	10
7	1	1	10	10	10	10
8	1	2	10	10	10	10
9	1	3	10	10	10	10
10	1	4	10	10	10	10
11	1	5	10	10	10	10

Index	S1	S2	A0	A1	A2	A3
0	0	0	0.617	10	0.332	10
1	0	1	1.16	10	1.06	1.46
2	0	2	1.13	10	1.13	1.14
3	0	3	1.09	10	1.08	1.09
4	0	4	1.06	10	1.05	1.06
5	0	5	1.03	10	1.0	1.04
6	1	0	0.796	0.704	0.823	10
7	1	1	0.971	1.1	1.05	1.09
8	1	2	1.07	1.07	1.07	1.07
9	1	3	1.08	1.07	1.06	1.06
10	1	4	1.04	1.05	1.05	1.06
11	1	5	1.05	1.03	1.0	1.04

Index	S1	S2	A0	A1	A2	A3
0	0	0	0.617	10	0.332	10
6	1	0	0.796	0.704	0.823	10
7	1	1	0.971	1.1	1.05	1.09
1	0	1	1.16	10	1.06	1.46

Index	S1	S2	A0	A1	A2	A3
0	0	0	0.617	10	0.332	10
6	1	0	0.796	0.704	0.823	10
7	1	1	0.971	1.1	1.05	1.09
1	0	1	1.16	10	1.06	1.46



**Спасибо за внимание**

# **Методы машинного обучения**

## ***Лекция 11***

### **Эволюционные алгоритмы**

# Эволюционный алгоритм

## Evaluationary algorithm



В искусственном интеллекте и машинном обучении эволюционные алгоритмы — это раздел эволюционных вычислений (моделирования), в которых используются модели процессов естественного отбора (размножение, рекомбинация, мутация и отбор) и принципы природной эволюции для решения задач **оптимизации**.

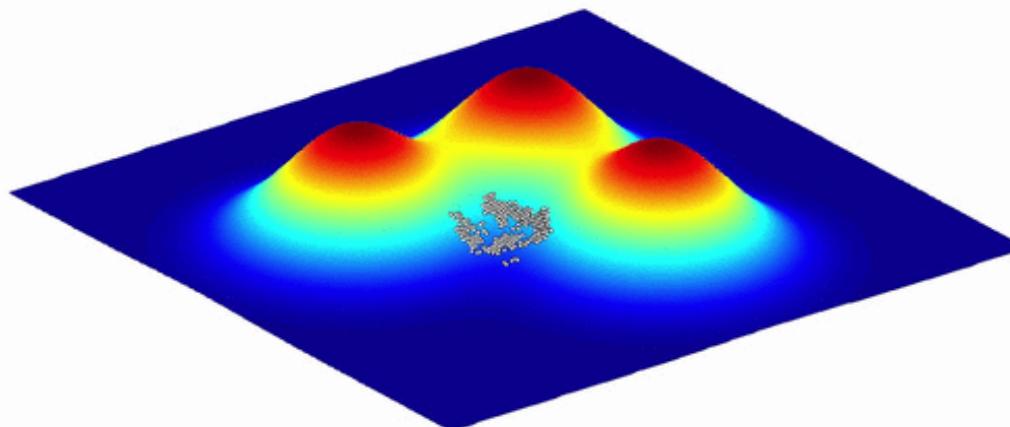
Решения задачи рассматриваются как особи популяции (агенты или индивидуумы). Множество особей принято называть популяцией. Качество каждого решения оценивается с помощью специальной целевой функции приспособленности (fitness function — фитнес-функция), которая задается окружающей средой. Такие алгоритмы относятся к адаптивным поисковым механизмам, включают эволюцию популяции и содержат следующие шаги:

1. Формируется начальная популяция методом случайного отбора (первое поколение).
2. Оценивается пригодность каждого члена популяции с помощью фитнес-функции.
3. Повторяются следующие действия (эволюция):
  - **отбор** — выбор наиболее приспособленных особей для размножения (родители);
  - **размножение** — формирование новых особей путем скрещивания и мутации, а затем оценка их пригодности;
  - **рекомбинация** — наименее приспособленные особи предыдущего поколения заменяются наиболее приспособленными особями нового поколения.

# Эволюционный алгоритм

## Преимущества и недостатки

Преимущества:	Недостатки:
<ul style="list-style-type: none"><li>• применимы к широкому классу задач оптимизации;</li><li>• легко комбинируются с другими методами;</li><li>• позволяют получать хорошо интерпретируемые результаты;</li><li>• обладают интерактивностью — можно включить в популяцию предложенные пользователем решения;</li><li>• рассматривают большое количество альтернативных решений.</li></ul>	<ul style="list-style-type: none"><li>• отсутствие гарантии нахождения оптимального решения за конечное время — алгоритм реализует локальную оптимизацию;</li><li>• слабая теоретическая база — алгоритм является эвристическим, т.е. точность и строгость постановки приносятся в жертву реализуемости;</li><li>• высокие вычислительные затраты.</li></ul>



# Генетический алгоритм (ГА)

Эвристический алгоритм поиска, предложенный в 1975 году Джоном Холландом (John Holland) из Мичиганского университета, используется для решения задач оптимизации и моделирования путём случайного подбора, комбинирования и вариации искомых параметров с использованием механизмов, аналогичных естественному отбору в природе. Является разновидностью эволюционных вычислений

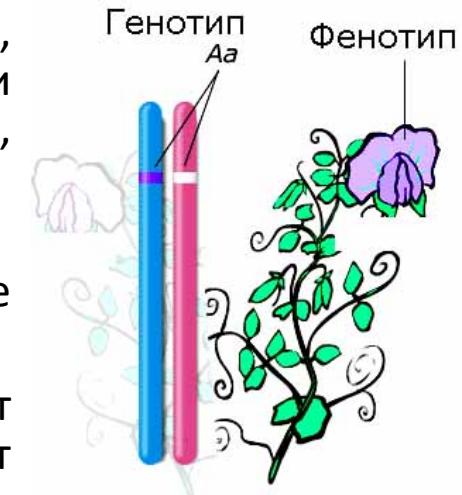
Работает с кодовыми последовательностями (КП), безотносительно к их смысловой интерпретации, поэтому КП и ее структура задается понятием **генотипа**, а его интерпретация, с точки зрения решаемой задачи, понятием **фенотипа**.

- **Фенотип** определяет, чем является объект в реальном мире.
- **Генотип** содержит всю информацию об объекте на уровне хромосомного набора.

Каждый ген, то есть элемент информации генотипа, имеет свое отражение в фенотипе, а совокупность генов называют хромосомой.

Каждая КП представляет, по сути, точку пространства поиска и называется особью или индивидуумом. Набор КП (особей) образует исходное множество решений  $K$  (популяцию). Количество особей в популяции характеризуется ее размером.

Особи в популяции оцениваются с использованием «функции приспособленности» (fitness function), в результате чего с каждым генотипом ассоциируется значение, которое определяет, насколько хорошо им описываемый фенотип решает поставленную задачу.



# Кодирование признаков

Для представления генотипа объекта применяются битовые строки, т.е. ген — битовая строка, чаще всего фиксированной длины, которая представляет собой значение признака (атрибута) объекта.

## 1. Целые числа

- битовое значение признака.
- использование кода Грея.

## 2. Действительные числа

- битовое представление.
- интервальное с кодом Грея:
  - Разбивают весь интервал допустимых значений признака на участки с требуемой точностью.
  - Принимают значение гена как целочисленное число, определяющее номер интервала (используя код Грея).
  - В качестве значения параметра принимают число, являющееся серединой этого интервала.

## 3. Нечисловые признаки

- Кодирование - преобразование в числовые

Число	Бинарный код	Код Грея
0	0000	0000
1	0001	0001
2	0010	0011
3	0011	0010
4	0100	0110
5	0101	0111
6	0110	0101
7	0111	0100
8	1000	1100

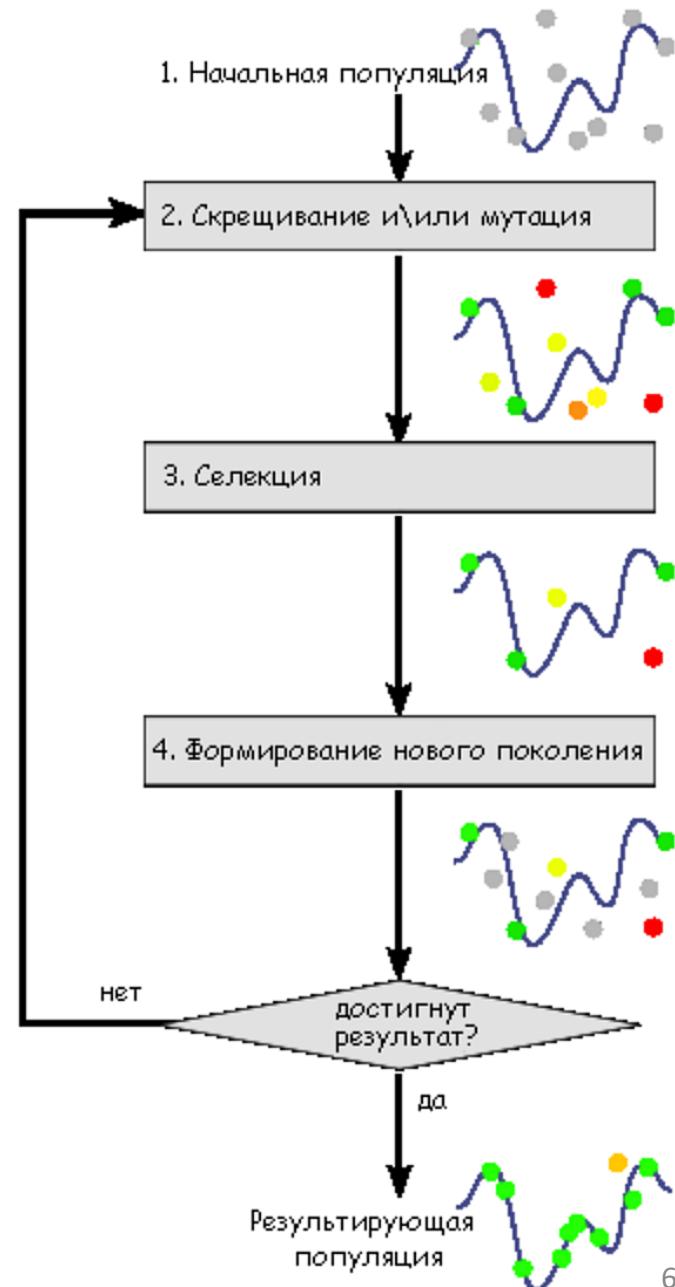
Код Грея — двоичный код, в котором две «соседние» кодовые комбинации различаются только цифрой в одном двоичном разряде. Иными словами, расстояние Хэмминга между соседними кодовыми комбинациями равно 1.

# Основные этапы ГА

1. Задать целевую функцию приспособленности (fitness function) для особей популяции
2. Создать начальную популяцию
3. Начало цикла эволюции
  - Выбор родительских особей
  - «Скрещивание» (Crossover)
  - «Мутация» (Mutation) / Инверсия
  - Вычисление значений fitness function
  - Формирование нового поколения (селекция)
  - Если выполняются условия остановки, то «конец эволюционного процесса», иначе переход в начало цикла эволюции.

Этот набор действий повторяется итеративно, так моделируется «эволюционный процесс», продолжающийся несколько жизненных циклов (поколений), пока не будет выполнен критерий остановки алгоритма. Критерием может быть:

- нахождение глобального, либо субоптимального решения;
- исчерпание числа поколений, отпущеных на эволюцию;
- исчерпание времени, отпущеного на эволюцию.



# Целевая функция приспособленности (fitness function)

Вещественная или целочисленная функция одной или нескольких переменных, подлежащая оптимизации в результате работы генетического алгоритма, направляет эволюцию в сторону оптимального решения. Позволяет оценить степень приспособленности конкретных особей в популяции.

На выбор (построение) фитнесс-функции оказывают влияние следующие факторы:

- тип задачи – максимизация или минимизация;
- содержание шумов окружающей среды в фитнесс-функции;
- возможность динамического изменения фитнесс-функции в процессе решения задачи;
- объем допустимых вычислительных ресурсов;
- насколько различные значения для близких особей должна давать фитнесс-функция для упрощения отбора родительских особей;
- должна ли она содержать ограничения решаемой задачи;
- может ли она совмещать различные подцели (например, для многокритериальных задач).

В ГА фитнесс-функция может использоваться в виде черного ящика, т.е. для данной хромосомы она вычисляет значение, определяющее качество данной особи, и при этом внутри может быть реализована: в виде математической функции, программы моделирования (в том числе имитационного), нейронной сети, экспертной оценки и др.

# Создание начальной популяции

Множество генотипов начальной популяции обычно создаётся случайным образом, тем самым выбирается нужное количество точек поискового пространства. Важно чтобы они соответствовали формату особей популяции, и на них можно было подсчитать функцию приспособленности.

Шаги:

- Инициализировать начальный момент времени  $t = 0$ . Случайным образом сформировать начальную популяцию, состоящую из  $k$  особей  $B_0 = A_1, A_2, \dots, A_k$ .
- Вычислить приспособленность каждой особи:

$$F_{A_i} = fit(A_i), \quad i = 1 \dots k,$$

и популяции в целом:

$$F_t = fit(B_t) = \sum_{i=1}^k fit(A_i)$$

Итогом первого шага является популяция  $B_0$ , состоящая из  $k$  особей.

# Выбор родительских особей

- Панмиксия — оба родителя выбираются случайно, каждая особь популяции имеет равные шансы быть выбранной
- Селективный выбор - родителями могут быть только те особи, значение приспособленности которых не меньше среднего значения по популяции.
- Рулетка — вероятность выбора хромосомы определяется ее приспособленностью, то есть:

$$P_{Get\ A_i} = \frac{Fit(A_i)}{Fit(B_t)}$$

- Турнирный отбор — случайно выбираются несколько особей из популяции и победителем выбирается особь с наибольшей приспособленностью.
- Инбридинг — первый родитель выбирается случайно, а вторым выбирается такой, который наиболее похож на первого родителя
- Аутбридинг — первый родитель выбирается случайно, а вторым выбирается такой, который наименее похож на первого родителя

Инбридинг и аутбридинг построены на основе близкого и дальнего родства и бывают в двух формах: фенотипной и генотипной. В случае фенотипной формы похожесть измеряется в зависимости от геометрического расстояния между особями популяции или значениями функции приспособленности. В случае генотипной формы похожесть измеряется в смысле хемминговского расстояния между хромосомными наборами особей (чем меньше отличий между генотипами особей, тем особи похожее).

# Скрещивание (Crossover)

Различают: одноточечные, двухточечные и равномерные операторы скрещивания.

- Одноточечный оператор случайным образом выбирает одну точку разрыва. Обе родительские строки разрываются на два сегмента по этой точке. Затем соответствующие сегменты различных родителей склеиваются и получаются два генотипа потомков (Рисунок внизу слайда).
- В двухточечном операторе выбирается две точки разрыва, и родительские хромосомы обмениваются сегментом, который находится между этими точками.
- В равномерном операторе скрещивания каждый бит первого родителя наследуется первым потомком с заданной вероятностью, в противном случае этот бит передается второму потомку.

Потомки добавляются в популяцию.

Родитель 1

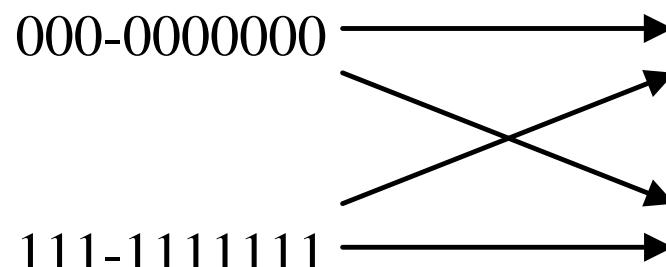
0000000000

000-0000000 → 111-0000000

Родитель 2

1111111111

111-1111111 → 000-1111111



# Мутация (mutation) / Инверсия

**Оператор мутации** предназначен для поддержания разнообразия особей в популяции и представляет собой случайное изменение КП. Применение оператора позволяет выскочить из локального экстремума и продолжить движение в поисках лучших решений. Выполнение задается некоторой вероятностью. При использовании каждый бит в хромосоме с определенной вероятностью инвертируется.

**Важно!**

- Если оператор мутации имеет слишком большую вероятность применения, это приводит к разрушению генетического материала популяции и переключению на обычный поиск в произвольном порядке (метод перебора).
- Если вероятность слишком мала, то может возникнуть проблема с недостаточным разнообразием в популяции, алгоритм застрянет в локальном минимуме или поиск займет слишком много времени.

**Оператор инверсии** заключается в том, что хромосома делится на две части, и затем они меняются местами. Схематически это можно представить следующим образом:

000 1111111	⇒	1111111 000
-------------	---	-------------

# Формирование нового поколения (Селекция)

На этапе отбора из всей популяции выбирают определённую долю особей, которая останется «в живых» и перейдет на следующий этап эволюции. Вероятность выживания особи  $A_i$  зависит от функции приспособленности  $Fit(A_i)$ . Из  $N$  особей популяции  $B_t$  должны остаться  $S$  особей, которые войдут в популяцию  $B_{t+1}$ , остальные отбрасываются.

- Турнирная селекция — случайно выбирается установленное количество особей (обычно две), затем из них выбирается особь с лучшим значением функции приспособленности.
- Метод рулетки — вероятность выбора особи тем вероятнее, чем лучше её значение функции приспособленности  $P_{A_i} = \frac{Fit(A_i)}{\sum_{j=1}^N Fit(A_j)}$ , где  $P_{A_i}$  — вероятность выбора  $i$  особи.
- Метод ранжирования — вероятность выбора зависит от места в списке особей отсортированных по значению функции приспособленности  $P_{A_i} = \frac{1}{N} \left( a - (a - b) \frac{i-1}{N-1} \right)$ , где  $a \in [1,2]$ ,  $b = 2 - a$ ,  $i$ - порядковый номер особи в отсортированном списке (т.е.  $\forall i \forall j$  если  $i < j$ , то  $Fit(A_i) \leq Fit(A_j)$  для поиска минимума фитнес-функции).
- Равномерное ранжирование — вероятность выбора особи определяется выражением:

$$P_{A_i} = \begin{cases} \frac{1}{\mu}, & \text{if } 1 \leq i \leq \mu \\ 0, & \text{if } \mu \leq i \leq N \end{cases}, \text{ где } \mu \leq N \text{ параметр метода.}$$

- Сигма-отсечение — для предотвращения преждевременной сходимости используются методы, масштабирующие значение фитнес-функции. Вероятность выбора особи тем больше, чем оптимальнее значение масштабируемой функции приспособленности  $P_{A_i} = \frac{F_i}{\sum_{j=1}^N F_j}$ , где  $F_i = 1 + \frac{Fit(A_i) - Fit_{Avg}}{2\sigma}$ ,  $Fit_{Avg}$  - среднее значение фитнес-функции для всей популяции,  $\sigma$  — среднеквадратичное отклонение значения фитнес-функции.
- Элитный отбор, который заключается в переносе некоторого количества (например, 10%) самых здоровых хромосом в следующее поколение.

# Настройка параметров и процессов ГА

Не существует оптимального набора методов и параметров, которые сумели бы решить любую задачу. В связи с этим важную роль приобретает настройка параметров и процессов ГА. В частности, можно менять не только множество параметров (например, метод отбора и рекомбинирования), но и управлять коэффициентами применения генетических операторов.

- В зависимости от того, какой метод кодирования выбран для задачи, некоторые генетические операторы могут стать деструктивными. Важное значение приобретает правильный выбор кодировки и понимания эффектов использования операторов.
- Большое значение имеет параметр, отвечающий за размер популяции ГА. Если популяция мала, генетического материала может не хватить для решения задачи. Размер популяции также влияет на коэффициент применения операторов мутации и скрещивания.
- Особое внимание стоит уделять выбору типа и вероятностям применения генетических операторов.

Основная сложность, которая может возникать при использовании генетических алгоритмов и которая может быть преодолена путем настройки параметров и процессов.

- Преждевременное схождение, связанное с недостаточным разнообразием хромосом в популяции. Если большинство хромосом в популяции схожи между собой, то для селекции доступно мало рабочего материала.
- Обнаружение эффекта преждевременного схождения возможно сравнением среднего здоровья популяции с максимальным здоровьем. Если они схожи, это означает, что произошло преждевременное схождение популяции.
- Причиной может быть применяемый алгоритм отбора. Например, при использовании метода элиты выбираются хромосомы с самым высоким здоровьем, что приводит к сильному уменьшению размера популяции по сравнению с начальным.
- Если популяция слишком рано сошлась и решение не было найдено, следует перезапустить алгоритм с целью добавления нового генетического материала.

# Непрерывный генетический алгоритм

В стандартном ГА используется двоичное кодирование признаков характеризующих объект, что не всегда удобно при работе с наборами вещественных чисел. Использование непрерывного генетического алгоритма может быть продиктовано необходимостью осуществлять оптимизацию в непрерывном пространстве значений вещественных чисел, тогда как двоичное представление может резко снизить точность найденного решения, в совокупности с резким увеличением длины хромосомы.

Основными преимуществами данного типа алгоритмов являются:

- возможность поиск в больших пространствах, что затруднительно в случае двоичных генов, когда увеличение пространства поиска сокращает точность решения при неизменной длине хромосомы;
- удобство представления решений, что обусловлено отсутствием операций кодирования/декодирования.

# Непрерывный ГА - Основные операторы

Отбор родительских особей для создания потомков осуществляется любым стандартным способом: рулетка, турнирный, случайный.

**Оператор скрещивания** - SBC (Simulated Binary Crossover) – кроссовер, который имитирует работу двоичного оператора скрещивания. Из двух векторов вещественных чисел осуществляется формирование двух новых векторов.

- Выбрать две родительские особи  $\{a_0^1, a_1^1, a_2^1 \dots a_N^1\}$  и  $\{a_0^2, a_1^2, a_2^2 \dots a_N^2\}$ ;
- Сгенерировать случайное число  $u \in [0,1)$  ;
- Вычислить значение переменной  $\beta$ , характеризующей распределение вероятностей случайной величины;
- где  $\eta$  - параметр, влияющий на появления потомков вдали от родительских особей.

$$\beta = \begin{cases} (2 \cdot u)^{1/(\eta+1)}, & u \leq 0.5 \\ \left( \frac{1}{2 \cdot (1-u)} \right)^{1/(\eta+1)}, & u > 0.5 \end{cases}$$

- Вычислить значения компонент векторов новых хромосом по формулам:

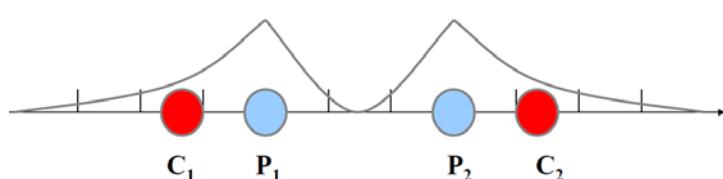
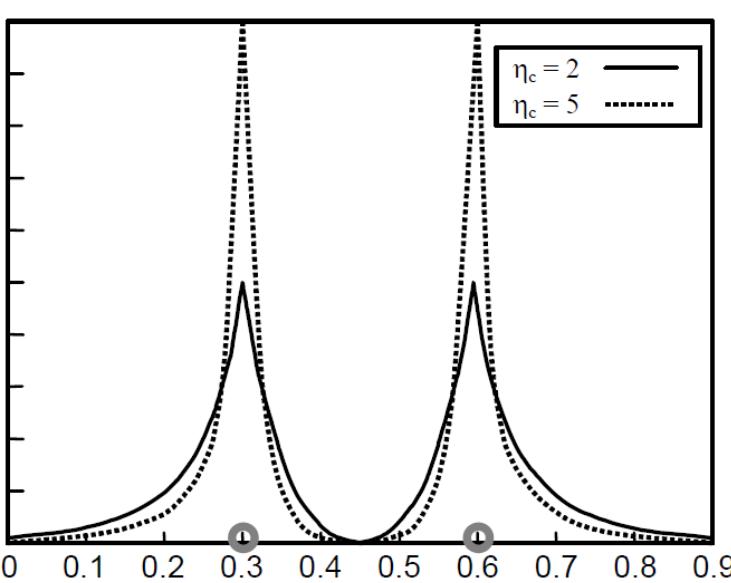
$$a_i^{1(new)} = 0.5[(1 + \beta) \cdot a_i^1 + (1 - \beta) \cdot a_i^2]$$

$$a_i^{2(new)} = 0.5[(1 - \beta) \cdot a_i^1 + (1 + \beta) \cdot a_i^2]$$

Среднее значение функции приспособленности остается неизменным у родителей и их потомков.

$$\frac{a_i^{1(new)} + a_i^{2(new)}}{2} = \frac{a_i^1 + a_i^2}{2}$$

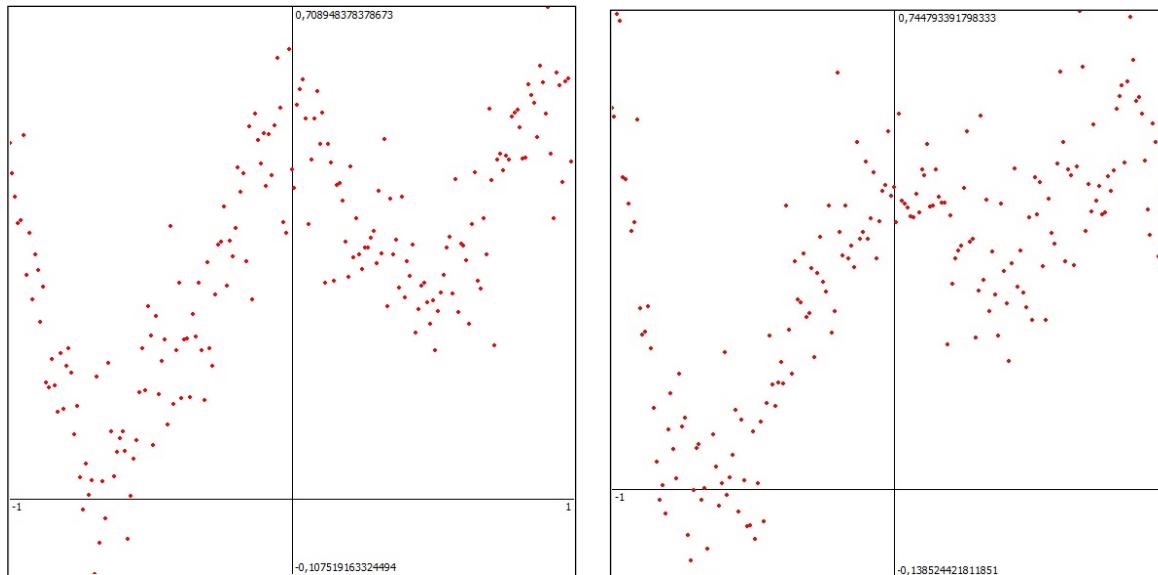
**Оператор мутации** выполняется с малой вероятностью, не превышающей 10%, для случайно выбранной компоненты вектора хромосомы на случайную величину в соответствии с нормальным законом распределения.



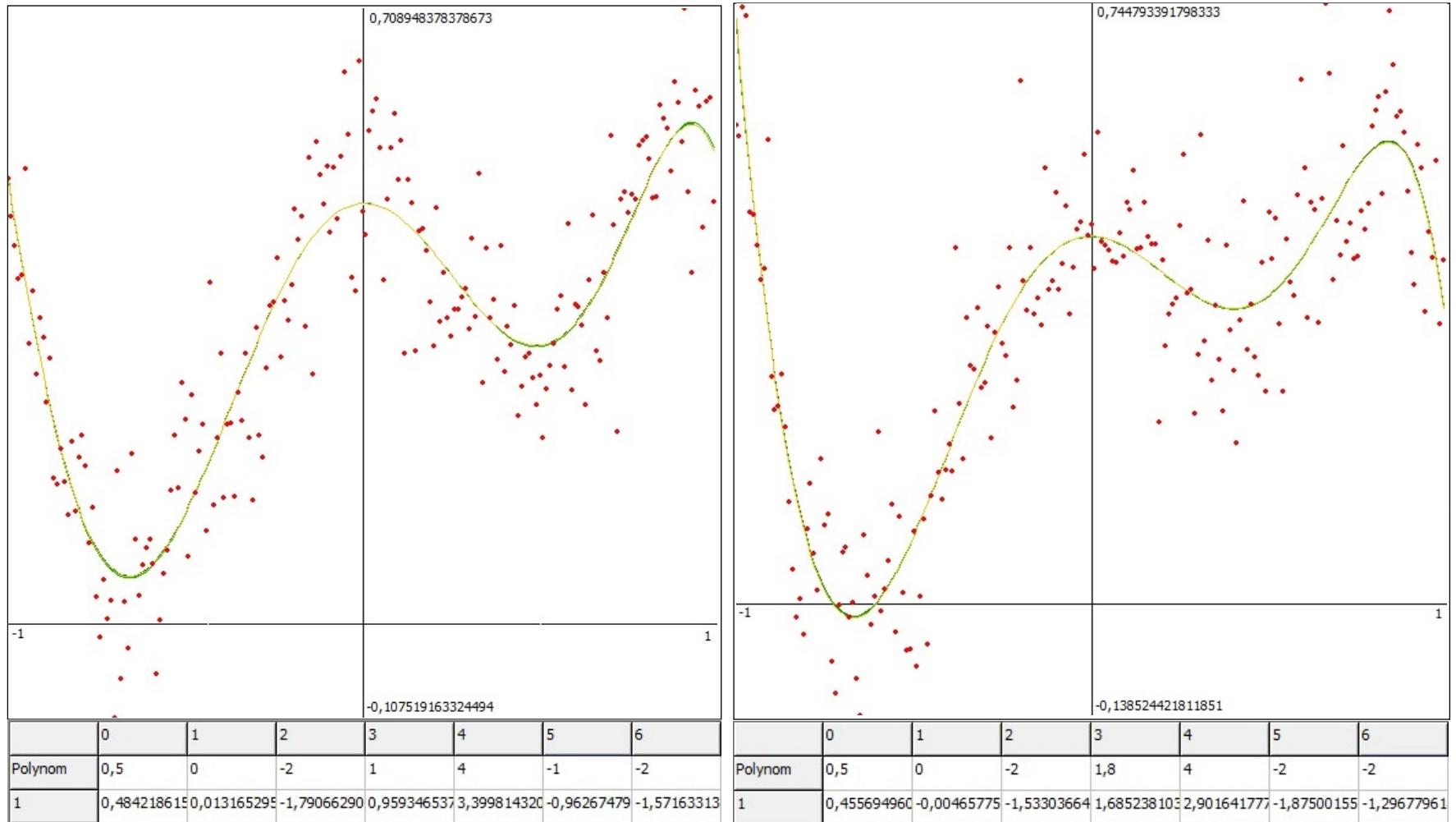
# Пример: Построение аппроксимирующего полинома N-ой степени

Задача заключается в поиске таких значений коэффициентов  $\{a_0, a_1, a_2 \dots a_N\}$  аппроксимирующего полинома N-ой степени вида:  $a_0 + a_1 \cdot x + a_2 \cdot x^2 + \dots + a_N \cdot x^N$ , которые бы наилучшим образом соответствовали исходному набору данных.

**Целевая функция** характеризует величину ошибки аппроксимации числовой последовательности с помощью полинома N-ой степени:  $F_k = \frac{\sum_{i=1}^m (y_i - \hat{y}_i)^2}{m}$  где  $y_i$  и  $\hat{y}_i$  - вычисленное и измеренное значения соответственно,  $m$  – количество значений в числовой последовательности,  $k$  – особь популяции. Цель процесса оптимизации заключается в поиске таких значений  $\{a_0, a_1, a_2 \dots a_N\}$  для которых:  $F_k \rightarrow 0$ ,  $k = 1 \div K$ , где  $K$  – количество особей в популяции.



# Результат моделирования работы непрерывного генетического алгоритма



# Использование генетического алгоритма для обучения нейронной сети

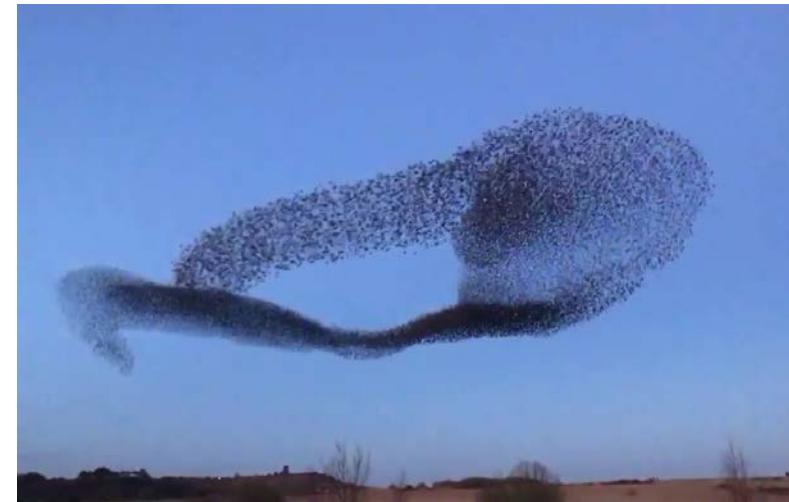
Генетические алгоритмы применимы для отыскания глобального экстремума многоэкстремальной функции и могут быть использованы для минимизации суммарной квадратической ошибки нейронной сети типа многослойный персепtron путем подстройки весовых коэффициентов.

Соотнесение основных структурных характеристик нейронной сети и параметров ГА осуществляется следующим образом:

- ген – весовой коэффициент нейронной сети;
- хромосома – набор генов или упорядоченный набор весовых коэффициентов нейронной сети, при этом каждая хромосома является возможным решением (т.е. таким набором весовых коэффициентов, который лучше подходят для решения имеющейся задачи);
- популяция – множество хромосом, вариантов наборов весовых коэффициентов (множество вариантов нейросети);
- эпоха – итерация, соответствующая созданию нового поколения хромосом.
- Над хромосомами осуществляются операции скрещивания, мутации, отбора и редукции, таким образом, происходит параллельная обработка множества альтернативных решений.

# Алгоритм роя частиц - предпосылки

Пример коллективного поведения животных - стая птиц. Стая двигается плавно и скоординировано, словно ей кто-то управляет, представляя собой роевой интеллект. Отдельные птицы в ней действуют согласно определенным – довольно простым – правилам. Кружка в небе, каждая из птиц следит за своими сородичами и координирует свое движение согласно их положению, а найдя источник пищи, она оповестит их об этом.

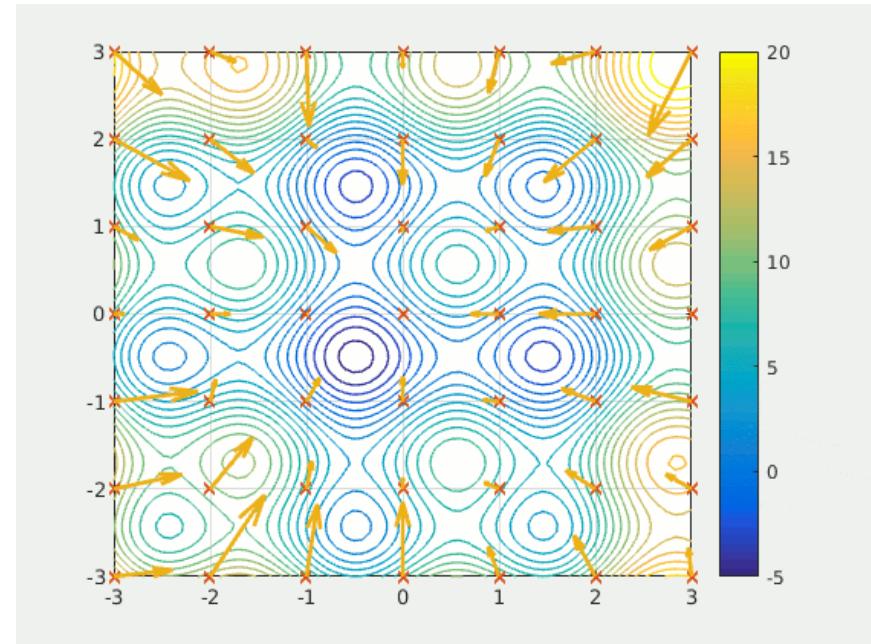


Наблюдение за птицами вдохновило Крейга Рейнольдса (Craig Reynolds) на создание в 1986 году компьютерной модели, которую он назвал Boids. Для имитации поведения стаи птиц, Рейнольдс запрограммировал поведение каждой из них в отдельности, а также их взаимодействие. При этом он использовал три простых принципа:

1. каждая птица в его модели стремилась избежать столкновений с другими птицами;
2. каждая птица двигалась в том же направлении, что и находящиеся неподалеку птицы;
3. птицы стремились двигаться на одинаковом расстоянии друг от друга.

# Классический алгоритм роя частиц

Метод роя частиц (МРЧ) был предложен Кеннеди, Эберхартом и Ши (1995 год) и изначально предназначался для имитации социального поведения. Алгоритм моделирует многоагентную систему, где агенты-частицы (популяция возможных решений) двигаются к оптимальным решениям, обмениваясь при этом информацией с соседями.



Текущее состояние частицы характеризуется координатами в пространстве решений (то есть, собственно, связанным с ними решением), а также вектором скорости перемещения. Оба этих параметра выбираются случайным образом на этапе инициализации. Кроме того, каждая частица хранит координаты лучшего из найденных ей решений, а также лучшее из пройденных всеми частицами решений – этим имитируется мгновенный обмен информацией между птицами.

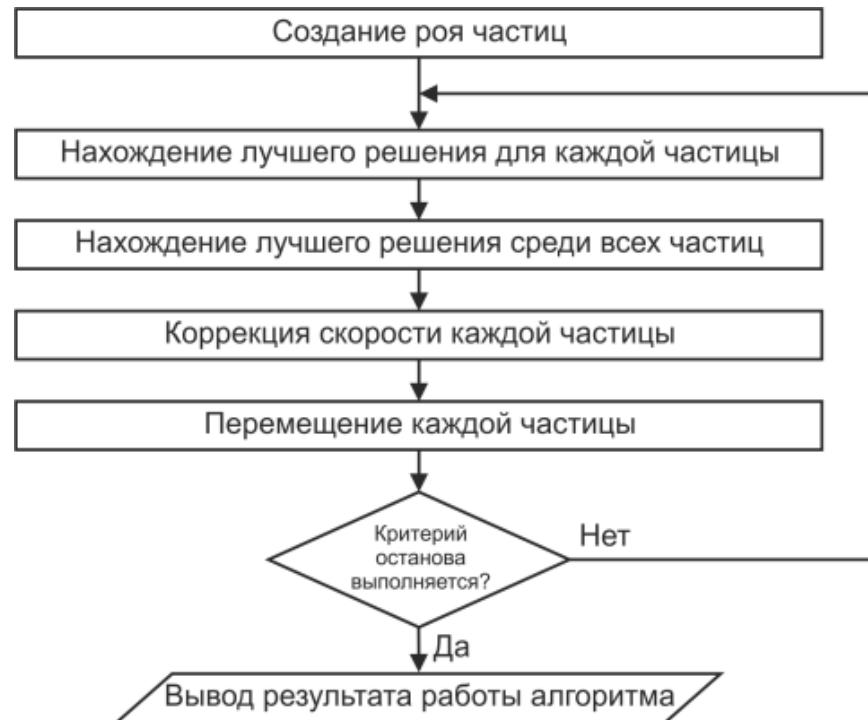
# Алгоритм роя частиц - инициализация

Пусть  $f: \mathbb{R}^n \rightarrow \mathbb{R}$  — целевая функция, которую требуется минимизировать,  $S$  — количество частиц в рое, каждой из которых сопоставлена координата  $x_i \in \mathbb{R}^n$  в пространстве решений и скорость  $v_i \in \mathbb{R}^n$ . Пусть также  $p_i$  — лучшее из известных положений частицы с индексом  $i$ , а  $g$  — наилучшее известное состояние роя в целом.

Для каждой частицы  $i = 1, \dots, S$ :

- Сгенерировать начальное положение частицы с помощью случайного вектора  $x_i \sim U(b_{lo}, b_{up})$ , имеющего многомерное равномерное распределение, где  $b_{lo}, b_{up}$  — нижняя и верхняя границы пространства решений соответственно.
- Присвоить лучшему известному положению частицы его начальное значение:  $p_i = x_i$ .
- Если  $f(p_i) < f(g)$ , то обновить наилучшее известное состояние роя:  $g = p_i$ .
- Присвоить значение скорости частицы:

$$v_i \sim U(-(b_{up} - b_{lo}), (b_{up} - b_{lo})).$$



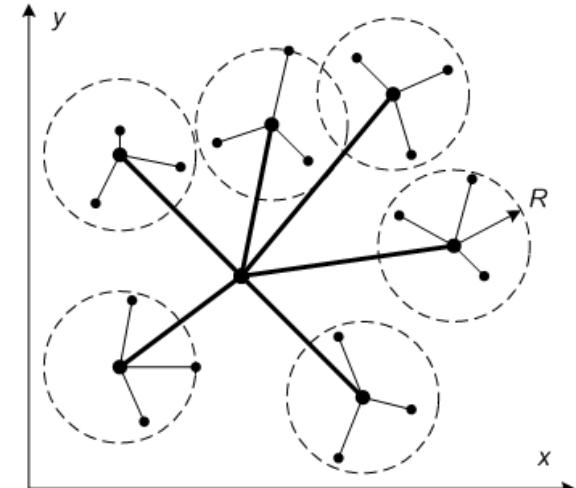
# Алгоритм роя частиц - выполнение

- Пока не выполнен критерий остановки (например, достижение заданного числа итераций или необходимого значения целевой функции), повторять для каждой частицы  $i = 1, \dots, S$  следующие действия:
  - Сгенерировать случайные векторы  $r_p, r_g \sim U(0,1)$ .
  - На каждой итерации направление и длина вектора скорости каждой из частиц изменяются в соответствие со сведениями о найденных оптимумах:
    - Обновить скорость частицы:  $v_{i,t+1} = \omega v_{i,t} + \varphi_p r_p \odot (p_i - x_{i,t}) + \varphi_g r_g \odot (g - x_{i,t})$ , где операция  $\odot$  - покомпонентное умножение,  $\varphi_p, \varphi_g$ -постоянные ускорения,  $\omega$  – коэффициент инерции.
    - Обновить положение частицы  $x_i$  на вектор скорости:  $x_{i,t+1} = x_{i,t} + v_{i,t+1}$ . Этот шаг выполняется вне зависимости от улучшения значения целевой функции.
  - Если  $f(x_i) < f(p_i)$ , то:
    - Обновить лучшее известное положение частицы:  $p_i = x_i$ .
    - Если  $f(p_i) < f(g)$ , то лучшее известное состояние роя:  $g = p_i$
- $g$  содержит лучшее из найденных решений.

Параметры  $\omega, \varphi_p, \varphi_g$  выбираются вычислителем и определяют поведение и эффективность метода в целом. Эти параметры составляют предмет многих исследований.

# Алгоритм пчелиной колонии

Алгоритм оптимизации подражанием пчелиной колонии (*artificial bee colony optimization, ABC*) — один из полиномиальных эвристических алгоритмов для решения дискретных (комбинаторных) и непрерывных задач глобальной оптимизации в области информатики и исследования операций. Относится к категории стохастических бионических алгоритмов, основан на имитации поведения колонии медоносных пчел при сборе нектара в природе. Предложен Д. Карабога в 2005 г.



Алгоритм включает два основных этапа:

- При инициализации (начальной разведке) производится выполнение разведки пространства признаков с целью определения  $K$  его наиболее перспективных точек с наилучшими значениями целевой функции  $f(X) = f(x_1, x_2, \dots, x_n)$  (в простейшем случае с использованием метода случайного перебора), которые запоминаются в улье.
- Последующая работа пчел улья. В окрестностях выбранных точек производится локальная разведка в пределах заданного радиуса разведки  $R$  с целью попытки уточнения решения, при этом при достижении улучшения в улье сохраняется обновленное значение  $f^*$  и соответствующий ему вектор параметров целевой функции  $X^*$ . Комбинируя работу пчел-разведчиков и рабочих пчел на протяжении заданного числа итераций  $C$  алгоритм обеспечивает постепенное улучшение запоминаемой выборки  $\mathcal{R} = [X_1, X_2, \dots, X_K]$  из  $K$  решений.

По завершении его работы из указанного множества решений выбирается наилучшее, что является результатом работы алгоритма.

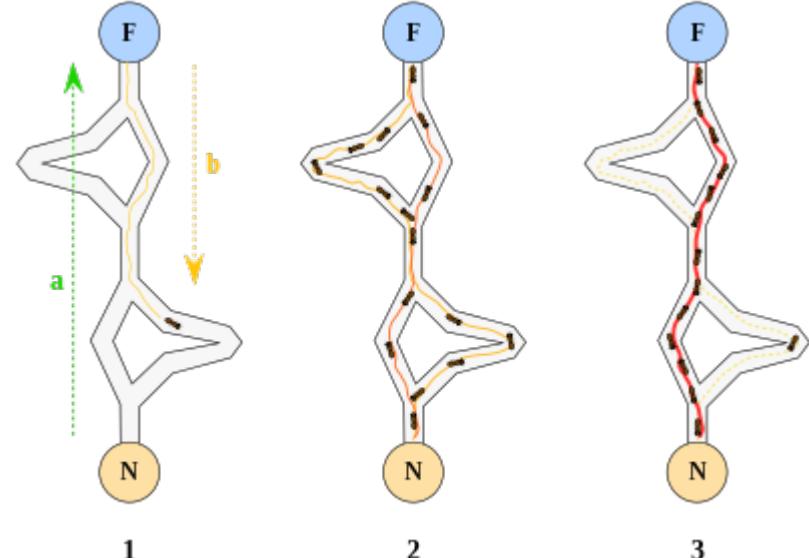
# Муравьиный алгоритм

Муравьиный алгоритм (алгоритм оптимизации подражанием муравьиной колонии - *ant colony optimization*, ACO) — алгоритм для нахождения приближенных решений задач оптимизации на графах, таких как задача коммивояжера, транспортная задача и аналогичных задач поиска маршрутов на графах. Вычислительные затраты алгоритма полиномиально зависят от объема исходных данных.

В основе метода лежит поведение муравьев некоторых видов, которые изначально перемещаются в поисках пищи случайным образом, но, найдя ее, возвращаются в свою колонию, помечая путь феромонами. Это избавляет других муравьев от необходимости случайного поиска пищи и делает его более целенаправленным: найдя путь, помеченный феромонами, муравьи движутся по нему, усиливая плотность феромонов.

Однако со временем плотность феромонов уменьшается. И происходит это тем быстрее, чем длиннее путь, поскольку интервал перемещения по нему муравьев велик. Напротив, чем путь короче и чем интенсивнее он используется, тем выше плотность феромона на нем. Таким образом, выбирая пути с наибольшей плотностью феромона, муравьи сокращают расстояние, проходимое в поисках пищи, что эквивалентно поиску кратчайшего пути на графике.

На коротком пути, для сравнения, прохождение будет более быстрым, и, как следствие, плотность феромонов остаётся высокой. Если бы феромоны не испарялись, то путь, выбранный первым, был бы самым привлекательным.



# Муравьиный алгоритм

Работа начинается с размещения муравьёв в вершинах графа (городах), затем начинается движение муравьёв — направление определяется вероятностным методом, на основании формулы вида:

**Рёбра:** Муравей будет двигаться от узла  $i$  к узлу  $j$  с вероятностью:

$$p_{i,j} = \frac{(\tau_{i,j}^\alpha)(\eta_{i,j}^\beta)}{\sum(\tau_{i,j}^\alpha)(\eta_{i,j}^\beta)}$$

$\tau_{i,j}^\alpha$  - количество феромонов на ребре  $i, j$ ;

$\alpha$  - параметр, контролирующий влияние  $\tau_{i,j}^\alpha$  (определяет «стадность» алгоритма);

$\eta_{i,j}^\beta$  - привлекательность ребра  $i, j$  (начальное значение  $1/d_{i,j}$ , где  $d$  расстояние);

$\beta$  - параметр, контролирующий влияние  $\eta_{i,j}^\beta$  (определяет «жадность» алгоритма);

$$\alpha + \beta = 1.$$

## Обновление феромонов

$$\tau_{i,j} = (1 - \rho)\tau_{i,j} + \Delta\tau_{i,j}$$

$\tau_{i,j}$  - количество феромонов на ребре  $i, j$

$\rho$  - скорость испарения феромона,

$\Delta\tau_{i,j}$  - количество отложенного феромона, обычно определяется как:

$$\Delta\tau_{i,j}^k = \begin{cases} 1/L_k & \text{if Ant } k \text{ travels on edge } i, j \\ 0 & \text{otherwise} \end{cases}$$

где  $L_k$ -стоимость  $k$ -ого пути муравья (обычно длина).

**Спасибо за внимание**

# **Методы машинного обучения**

*Лекция 12*

## **Кластеризация**

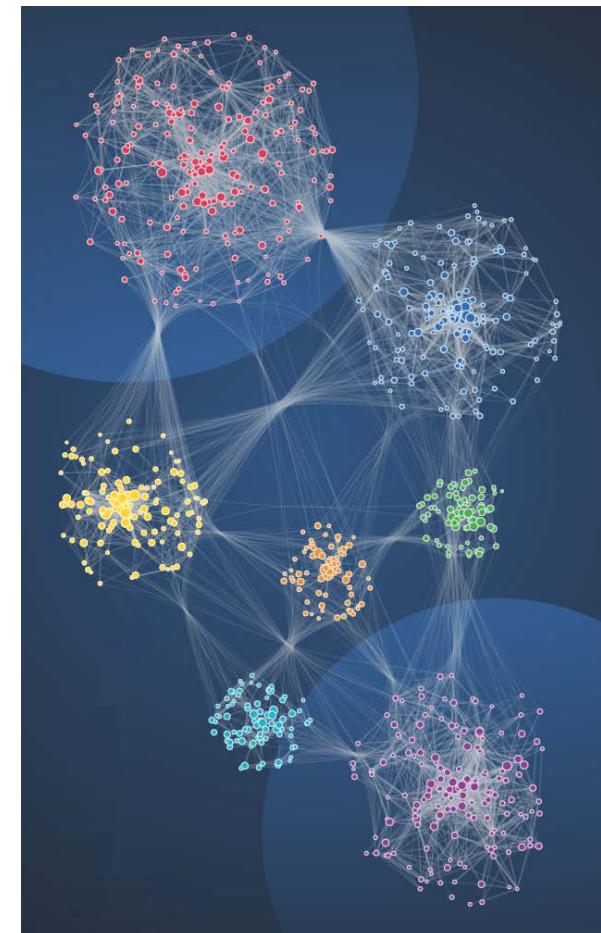
# Кластеризация

**Кластеризация** (*clustering/cluster analysis*) — задача группировки множества объектов на подмножества (**кластеры**) таким образом, чтобы объекты из одного кластера были более похожи друг на друга, чем на объекты из других кластеров по какому-либо критерию. Это может быть информация о попарном сходстве объектов. Функционалы качества могут определяться по-разному, например, как отношение средних межкластерных и внутрикластерных расстояний.

Исторически возникла из задачи группировки схожих объектов в единую структуру (кластер) с последующим выявлением общих черт.

Примеры областей применения:

- Экономическая география: по физико-географическим и экономическим показателям разбить страны мира на группы схожих по экономическому положению государств;
- Финансовая сфера: по сводкам банковских операций выявить группы «подозрительных», нетипичных банков, сгруппировать остальные по степени близости проводимой стратегии;
- Маркетинг: по результатам маркетинговых исследований среди множества потребителей выделить характерные группы по степени интереса к продвигаемому продукту;
- Социология: по результатам социологических опросов выявить группы общественных проблем, вызывающих схожую реакцию у общества, а также характерные фокус-группы населения.



# Формальная постановка задачи

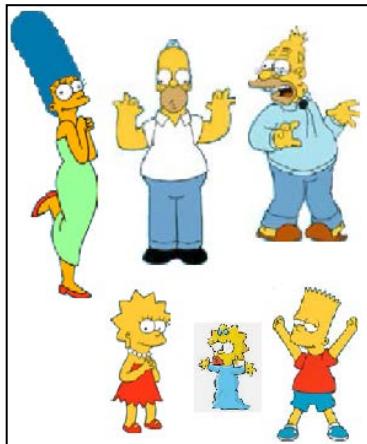
Пусть  $X$  — множество объектов,  $Y$  — множество идентификаторов (номеров, имён, меток) кластеров. Задана функция расстояний между объектами  $\rho(x, x')$ . Имеется конечная обучающая выборка объектов  $X^m = \{x_1, \dots, x_m\} \subset X$ . Требуется разбить выборку на непересекающиеся подмножества, называемые кластерами, так, чтобы каждый кластер состоял из объектов, близких по метрике  $\rho$ , а объекты разных кластеров существенно отличались. При этом каждому объекту  $x_i \in X^m$  приписывается номер кластера  $y_i$ .

Алгоритм кластеризации — это функция  $\alpha: X \rightarrow Y$ , которая любому объекту  $x \in X$  ставит в соответствие номер кластера  $y \in Y$ . Множество  $Y$  в некоторых случаях известно заранее, однако чаще ставится задача определить оптимальное число кластеров, с точки зрения того или иного критерия качества кластеризации.

# Принципиальная неоднозначность

Решение задачи кластеризации принципиально неоднозначно:

- Не существует однозначно наилучшего критерия качества кластеризации.
- Число кластеров, как правило, неизвестно заранее и устанавливается в соответствии с некоторым субъективным критерием.
- Результат кластеризации существенно зависит от метрики, выбор которой, как правило, также субъективен и определяется экспертом.



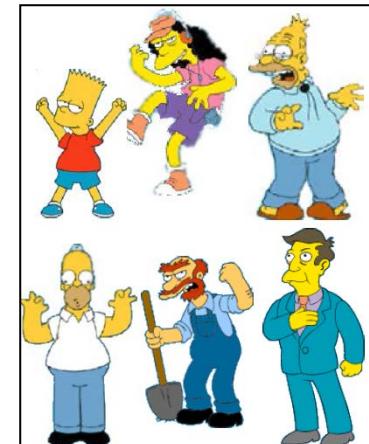
Семья



Сотрудники



Женщины



Мужчины

# Цели кластеризации

- **Улучшение понимания данных** за счет выявления структурных групп;
- **Классификация объектов.** Попытка понять зависимости между объектами путем выявления их кластерной структуры. Разбиение выборки на группы схожих объектов упрощает дальнейшую обработку данных, позволяет применить к каждому кластеру свой метод анализа (стратегия «разделяй и властвуй»). В данном случае стремятся уменьшить число кластеров для выявления наиболее общих закономерностей;
- **Сжатие данных.** Можно сократить размер исходной выборки, взяв один или несколько наиболее типичных представителей каждого кластера. Здесь важно наиболее точно очертить границы каждого кластера, их количество не является важным критерием;
- **Обнаружение новизны (обнаружение шума).** Выделение объектов, которые не подходят по критериям ни в один кластер. Обнаруженные объекты в дальнейшем обрабатывают отдельно.



# Основные этапы

- Отбор выборки объектов для кластерного анализа.
- Определение множества переменных, по которым будут оцениваться объекты в выборке, то есть признакового пространства.
- Выбор и вычисление значений меры сходства (или различия) между объектами.
- Применение метода кластерного анализа для создания групп сходных объектов.
- Проверка достоверности и представление результатов анализа.

# Типы входных данных

- **Признаковое описание объектов.** Каждый объект описывается набором своих характеристик/признаков (*features*). Признаки могут быть как числовыми, так и нечисловыми (категориальными);
- **Матрица расстояний между объектами.** Каждый объект описывается расстоянием до всех объектов из обучающей выборки.

Фундаментальные требования, предъявляемые к данным:

- **Однородность (uniformity)** - требует, чтобы все кластеризуемые сущности были одной природы, описывались сходным набором характеристик, т.е. значения всех атрибутов должны быть сравнимыми для всех данных.
- **Полнота (comprehensiveness)** — набор данных должен содержать достаточное количество параметров или признаков, чтобы не осталось неохваченных пограничных случаев.

# Меры расстояний

- Евклидово расстояние:

$$\rho(x, x') = \sqrt{\sum_i^n (x_i - x'_i)^2}$$

- Квадрат евклидова расстояния:

$$\rho(x, x') = \sum_i^n (x_i - x'_i)^2$$

- Расстояние городских кварталов L1 (манхэттенское расстояние):

$$\rho(x, x') = \sum_i^n |x_i - x'_i|$$

- Расстояние Чебышева:

$$\rho(x, x') = \max(|x_i - x'_i|)$$

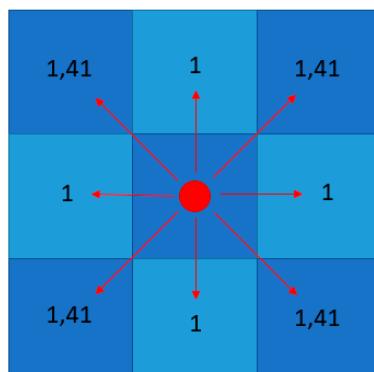
- Степенное расстояние:

$$\rho(x, x') = \sqrt[r]{\sum_i^n (x_i - x'_i)^p}$$

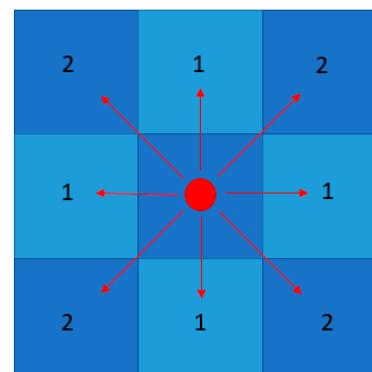
где r и p – параметры,

определяемые пользователем.

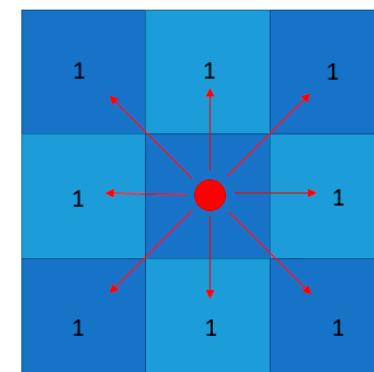
Евклидово расстояние



Расстояние L1



Расстояние Чебышёва



# Кластерный центроид

**Центром кластера  $C_k$  (центроидом)** называется геометрический центр точек  $k$ -ого класса в евклидовом пространстве:

$$c_k = \frac{1}{|C_k|} \sum_{x_i \in C_k} x_i$$

где  $|C_k|$  – число точек в  $k$ -ом кластере,  $k = \overline{1, K}$ ,  $K$  – число кластеров.

**Дисперсия кластера  $C_k$**  – мера рассеяния точек в пространстве относительно центра кластера:

$$D_k = \frac{1}{|C_k|} \sum_{x_i \in C_k} \rho^2(x_i, c_k)$$

**Радиус кластера  $C_k$**  – мера рассеяния точек относительно центра кластера – максимальное расстояние до центра кластера:

$$R_k = \max_{x_i \in C_k} \rho(x_i, c_k)$$

# Меры качества кластеризации

Задачу кластеризации можно ставить как задачу дискретной оптимизации, т.е. необходимо так присвоить номера кластеров  $y_i$  объектам  $x_i$ , чтобы значение выбранного функционала качества приняло наилучшее значение.

Функционал качества:

- Среднее внутрикластерное расстояние должно быть как можно меньше:
- Среднее межкластерное расстояние должно быть как можно больше:

$$F_0 = \frac{\sum_{i < j} [y_i = y_j] \rho(x_i, x_j)}{\sum_{i < j} [y_i = y_j]} \rightarrow \min$$

$$F_1 = \frac{\sum_{i < j} [y_i \neq y_j] \rho(x_i, x_j)}{\sum_{i < j} [y_i \neq y_j]} \rightarrow \max$$

Если алгоритм кластеризации вычисляет центры кластеров  $\mu_y$ ,  $y \in Y$ , то можно определить функционалы:

- Сумма средних внутрикластерных расстояний  $\Phi_0 = \sum_{y \in Y} \frac{1}{|K_y|} \sum_{i: y_i = y} \rho^2(x_i, \mu_y) \rightarrow \min$  должна быть как можно меньше:  
где  $K_y = \{x_i \in X^l | y_i = y\}$  кластер с номером  $y$ , состоящий из  $|K_y|$  точек.
- Сумма межкластерных расстояний  $\Phi_1 = \sum_{y \in Y} \rho^2(\mu_y, \mu) \rightarrow \max$  должна быть как можно больше:  
где  $\mu$  - центр масс всей выборки.

На практике вычисляют отношение пары функционалов, чтобы сразу учесть как межкластерные, так и внутрикластерные расстояния:

$$\frac{F_0}{F_1} \rightarrow \min, \text{ или } \frac{\Phi_0}{\Phi_1} \rightarrow \min$$

# Методы оценки качества кластеризации

Принято выделять две группы методов оценки качества кластеризации:

- **Внешние** (англ. *External*) меры основаны на сравнении результата кластеризации с априори известным разделением на классы. Данные меры используют дополнительные знания о кластеризуемом множестве: распределение по кластерам, количество кластеров и т.д.
- **Внутренние** (англ. *Internal*) меры отображают качество кластеризации только по информации в данных и не используя внешней информации.

# Внешние меры оценки качества – Таблица сопряженности

Дано множество  $S$  из  $n$  элементов, разделение на классы  $X = \{X_1, X_2, \dots, X_r\}$ , и полученное разделение на кластеры  $Y = \{Y_1, Y_2, \dots, Y_s\}$ , совпадения между  $X$  и  $Y$  могут быть отражены в таблице сопряженности  $[n_{ij}]$ , где каждое  $n_{ij}$  обозначает число объектов, входящих как в  $X_i$ , так и в  $Y_j$ :  $n_{ij} = |X_i \cap Y_j|$ .

$X \setminus Y$	$Y_1$	$Y_2$	$\dots$	$Y_s$	Sums
$X_1$	$n_{11}$	$n_{12}$	$\dots$	$n_{1s}$	$a_1$
$X_2$	$n_{21}$	$n_{22}$	$\dots$	$n_{2s}$	$a_2$
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$
$X_r$	$n_{r1}$	$n_{r2}$	$\dots$	$n_{rs}$	$a_r$
Sums	$b_1$	$b_2$	$\dots$	$b_s$	$n$

Пусть  $p_{ij} = \frac{n_{ij}}{n}$ ,  $p_i = \frac{a_i}{n}$ ,  $p_j = \frac{b_j}{n}$ .

Также рассмотрим пары  $(x_i, x_j)$  из элементов кластеризуемого множества  $X$ . и подсчитаем количество пар, в которых:

- Элементы принадлежат одному кластеру и одному классу —  $TP$
- Элементы принадлежат одному кластеру, но разным классам —  $FP$
- Элементы принадлежат разным кластерам, но одному классу —  $FN$
- Элементы принадлежат разным кластерам и разным классам —  $TN$

# Внешние меры оценки качества - Индексы

- Индекс Rand

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

- Индекс Жаккара (Jaccard Index)

$$Jaccard = \frac{TP}{TP + FN + FP}$$

- Индекс Фоулкса – Мэллова (Fowlkes-Mallows Index)

$$FM = \sqrt{\frac{TP}{TP + FP} \cdot \frac{TP}{TP + FN}}$$

- Индекс Phi

$$\Phi = \frac{TP \times TN - FN \times FP}{(TP + FN)(TP + FP)(FN + TN)(FP + TN)}$$

- Entropy (Энтропия)

$$E = - \sum_i p_i \left( \sum_j \frac{p_{ij}}{p_i} \log \left( \frac{p_{ij}}{p_i} \right) \right)$$

- Purity (чистота)

$$P = \sum_i \max_j p_{ij}$$

- F-мера

$$F = \sum_j p_j \max_i \left[ 2 \frac{p_{ij}}{p_i} \frac{p_{ij}}{p_j} / \left( \frac{p_{ij}}{p_i} + \frac{p_{ij}}{p_j} \right) \right]$$

- Variation of Information (изменение информации)

$$VI = - \sum_i p_i \log p_i - \sum_i p_j \log p_j - 2 \sum_i \sum_j p_{ij} \log \frac{p_{ij}}{p_i p_j}$$

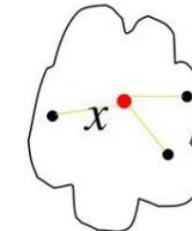
# Внутренние меры оценки качества

Данные меры оценивают качество структуры кластеров опираясь только непосредственно на нее, не используя внешней информации.

- **Компактность кластеров (Cluster Cohesion)**

*Within cluster Sum of Squares (WSS) → min*

$$WSS = \sum_{j=1}^M \sum_{i=1}^{|C_j|} (x_{ij} - \bar{x}_j)^2, \text{ где } M \text{ — количество кластеров.}$$

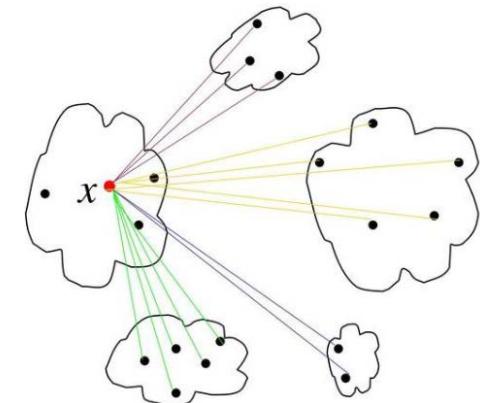


- **Отделимость кластеров (Cluster Separation)**

*Between cluster Sum of Squares (BSS) → max*

$$BSS = \sum_i |C_i| (m - m_i)^2$$

где  $C_i$  —  $i$ -ый кластер,  $m_i$ -центроид  $C_i$ ,  $m$  — общий центр.



- **Силуэт (Silhouette)**

$$Sil(C) = \frac{1}{N} \sum_{c_k \in C} \sum_{x_i \in c_k} \frac{b(x_i, c_k) - a(x_i, c_k)}{\max\{a(x_i, c_k), b(x_i, c_k)\}},$$

$$a(x_i, c_k) = \frac{1}{|c_k|} \sum_{x_j \in c_k} \|x_i - x_j\| \text{ - компактность.}$$

$$b(x_i, c_k) = \min_{c_l \in C \setminus c_k} \left\{ \frac{1}{|c_l|} \sum_{x_j \in c_l} \|x_i - x_j\| \right\} \text{ - отделимость.}$$

Значение:  $-1 \leq Sil(C) \leq 1$ . Чем ближе к 1, тем лучше.

# Внутренние меры оценки качества

- Индекс Дэвиса–Болдуина (Davies–Bouldin Index)

$$DB(C) = \frac{1}{K} \sum_{c_k \in C} \max_{c_l \in C \setminus c_k} \left\{ \frac{S(c_k) + S(c_l)}{\|\bar{c}_k - \bar{c}_l\|} \right\}, \quad S(c_k) = \frac{1}{|c_k|} \sum_{x_i \in c_k} \|x_i - \bar{c}_k\|$$

- Индекс Calinski–Harabasz

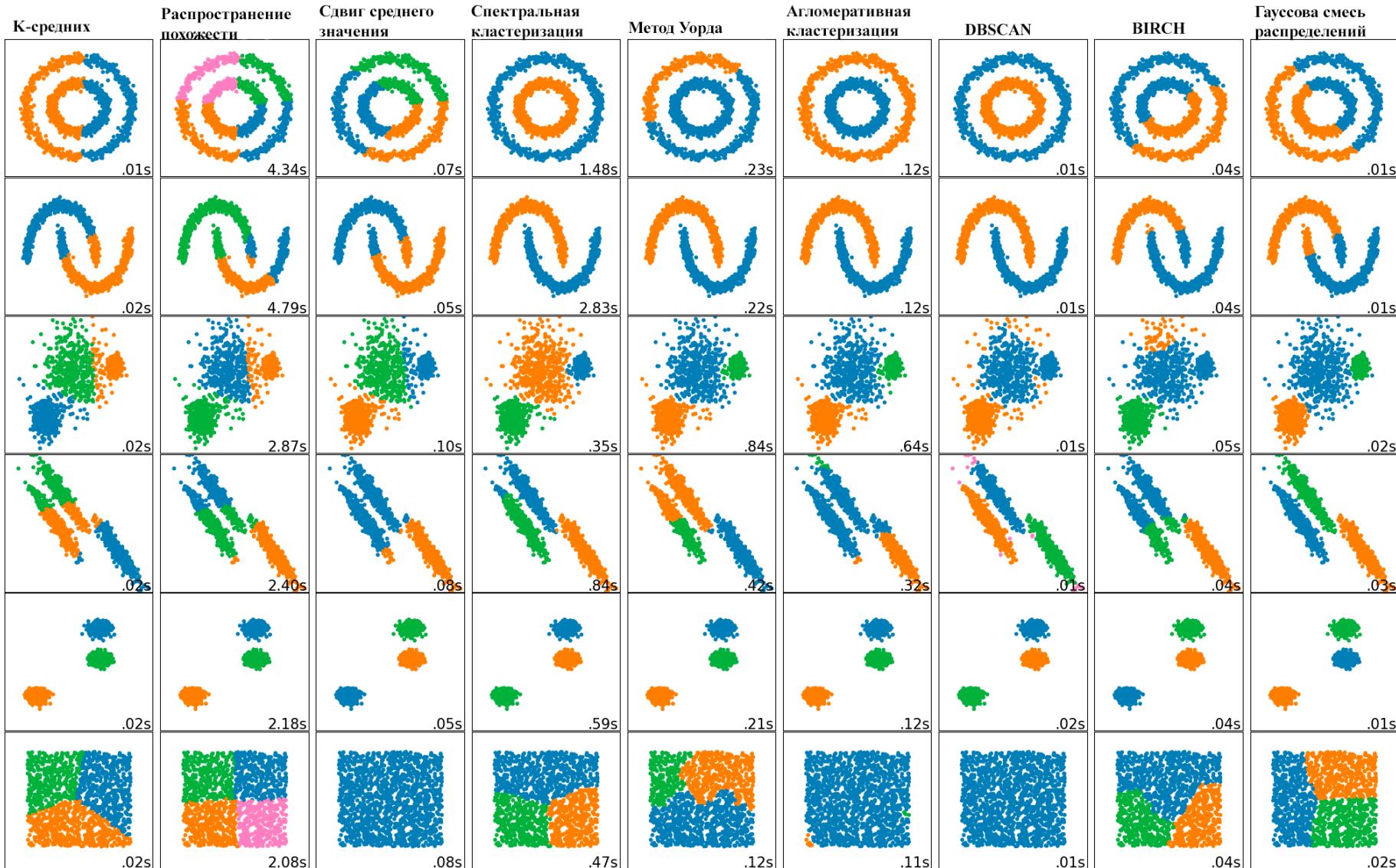
$$CH(C) = \frac{N - K}{K - 1} \cdot \frac{\sum_{c_k \in C} |c_k| \cdot \|\bar{c}_k - \bar{X}\|}{\sum_{c_k \in C} \sum_{x_i \in c_k} \|x_i - \bar{c}_k\|}$$

- Score function

$$SF(C) = 1 - \frac{1}{e^{bcd(C) - wcd(C)}},$$

$$bcd(C) = \frac{\sum_{c_k \in C} |c_k| \cdot \|\bar{c}_k - \bar{X}\|}{N \times K}, \quad wcd(C) = \sum_{c_k \in C} \frac{1}{|c_k|} \sum_{x_i \in c_k} \|x_i - \bar{c}_k\|$$

# Алгоритмы кластеризации



# Типы алгоритмов кластеризации

Алгоритмы кластеризации можно разделить на 4 типа:

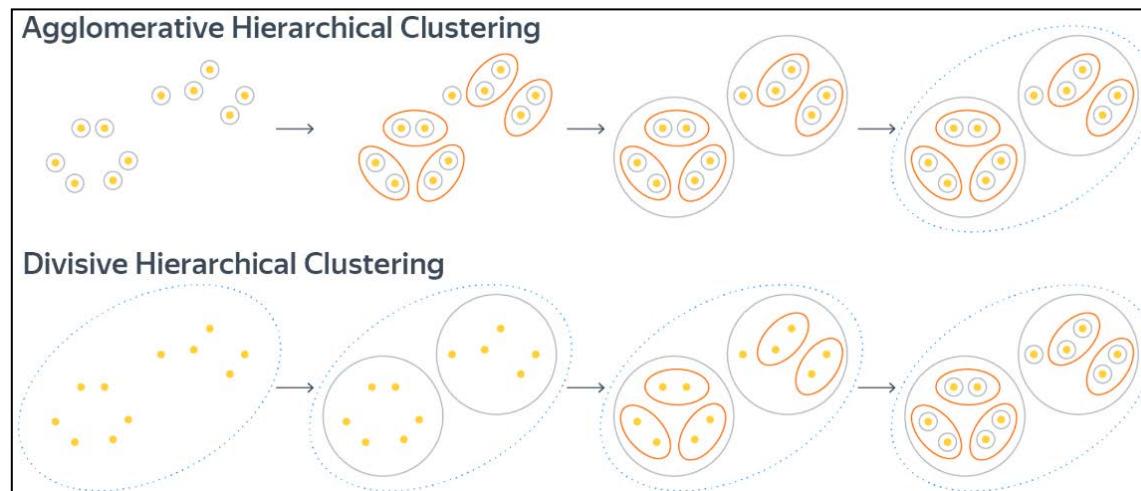
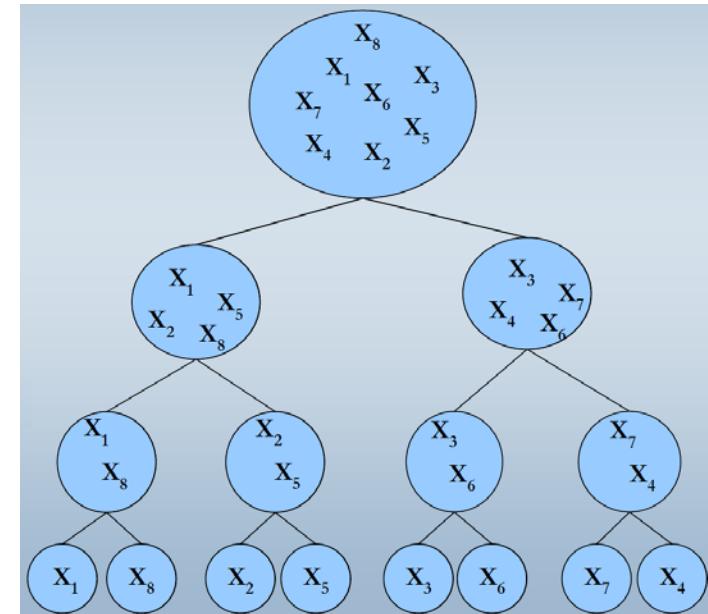
- 1. Кластерный центроид.** Он объединяет данные в кластеры, основываясь на заранее заданных условиях и характеристиках. k-средних — наиболее популярный алгоритм из этой категории.
- 2. Кластеризация на основе плотности.** В этом типе используется алгоритм, который соединяет области с высокой плотностью в кластеры, создавая распределения произвольной формы.
- 3. Кластеризация на основе распределений.** Алгоритм этого типа предполагает гауссовские распределения данных, которые далее объединяются в различные варианты того же распределения.
- 4. Иерархические алгоритмы.** В этом алгоритме строится иерархическая древо кластеров. Количество кластеров можно менять, объединяя их на определённом уровне дерева.

# Иерархическая кластеризация

Среди алгоритмов иерархической кластеризации выделяются два основных типа:

**1. Нисходящая кластеризация (divisive):** Работает по принципу «сверху-вниз»: в начале, все объекты принадлежат одному кластеру. В ходе итеративного процесса крупные кластеры разделяются на более мелкие. Такие задачи называются задачами таксономии. При этом, получается дерево кластеров (дендrogramма).

**2. Восходящая кластеризация (agglomerative):** Изначально каждый элемент множества является отдельным кластером. Процесс образования новых кластеров заключается в объединение некоторых кластеров в один на основе заданного расстояния. В итоге итеративного объединения получаем дерево, которое сходится к одному кластеру.



# Алгоритмы иерархической кластеризации - расстояние между кластерами

- Одиночная связь (расстояния ближайшего соседа) - расстояние между двумя кластерами определяется расстоянием между двумя наиболее близкими объектами (ближайшими соседями) в различных кластерах.
- Полная связь (расстояние наиболее удаленных соседей) - расстояния между кластерами определяются наибольшим расстоянием между любыми двумя объектами в различных кластерах (т.е. наиболее удаленными соседями).
- Невзвешенное попарное среднее - расстояние между двумя различными кластерами вычисляется как среднее расстояние между всеми парами объектов в них.
- Взвешенное попарное среднее - метод идентичен методу невзвешенного попарного среднего, за исключением того, что при вычислениях размер соответствующих кластеров (т.е. число объектов, содержащихся в них) используется в качестве весового коэффициента.
- Невзвешенный центроидный метод - расстояние между двумя кластерами определяется как расстояние между их центрами тяжести.
- Взвешенный центроидный метод (медиана) - идентичен предыдущему, за исключением того, что при вычислениях используются веса для учета разницы между размерами кластеров.

# Метод k-средних (k-means)

Алгоритм относится к классу эвристических EM-алгоритмов (Expectation–maximization), используемых в математической статистике для нахождения оценок максимального правдоподобия параметров вероятностных моделей.

Основная идея метода — итеративное повторение двух шагов:

- распределение объектов выборки по кластерам;
- пересчёт центров кластеров.

Алгоритм заключается в том, что он стремится минимизировать суммарное квадратичное отклонение точек кластеров от центров этих кластеров:

$$V = \sum_{k=1}^K \sum_{x_i \in C_k} \rho^2(x_i, c_k) \rightarrow \min_c$$

где  $K$  — число кластеров,  $C_k$  — полученные кластеры,  $c_k$  — центр кластера  $C_k$ ,  $x_i$  вектор из кластера  $C_k$ .

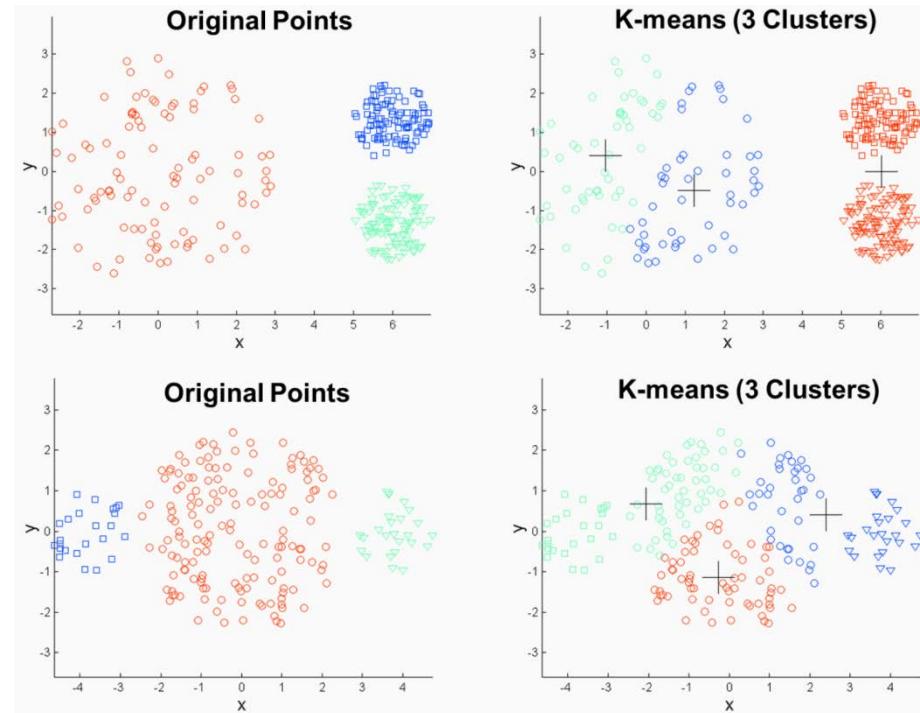
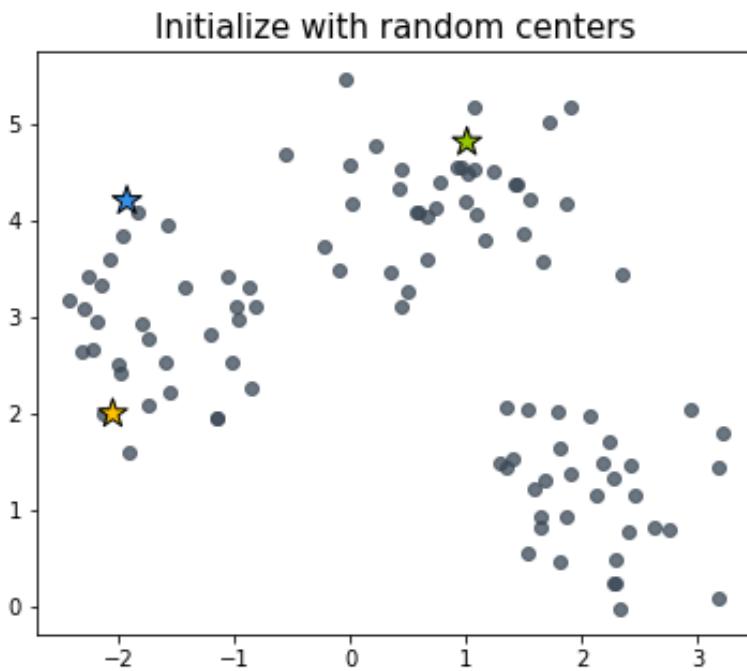
В начале работы алгоритма выбираются  $K$  случайных центров в пространстве признаков. Каждый объект выборки относят к тому кластеру, к центру которого объект оказался ближе по выбранной метрике.

$$x_i \in C_k, \text{ если } \rho(x_i, c_k) = \min_{k=1, K}$$

Далее центры кластеров пересчитывают как среднее арифметическое векторов признаков всех вошедших в этот кластер объектов (то есть центр масс кластера). Как только мы обновили центры кластеров, объекты заново перераспределяются по ним, а затем можно снова уточнить положение центров. Процесс продолжается до тех пор, пока центры кластеров не перестанут меняться.

# Недостатки метода k-средних

- Не гарантирует достижения глобального минимума суммарного квадратичного отклонения  $V$ , а только одного из локальных минимумов.
- Результат зависит от начального выбора центров кластеров  $\{c_k^{(0)}\}$ , а их оптимальный выбор неизвестен.
- Число кластеров  $K$  надо знать заранее.



# Метод k-средних

## Инициализация центров кластеров

- Метод Forgy. Из имеющегося набора данных случайным образом выбираются  $K$  наблюдений.
- Случайное разбиение. Каждому наблюдению на начальном этапе случайным образом присваивается номер кластера.
- Алгоритм k-means++. Из имеющегося набора данных случайным образом выбирается одна точка (первый центроид). Затем следующая точка выбирается из оставшихся с вероятностью, пропорционально зависящей от квадрата расстояния от точки до ближайшего центроида. Итерации повторяются до тех пор, пока не будут выбраны  $K$  центроидов.

# Алгоритмы семейства FOREL

**FOREL (Формальный Элемент)** — алгоритм кластеризации, основанный на идее объединения в один кластер объектов в областях их наибольшего сгущения.

**Цель:** Разбить выборку на такое (заранее неизвестное) число таксонов (групп/кластеров), чтобы сумма расстояний от объектов кластеров до центров кластеров была минимальной по всем кластерам. То есть наша задача — выделить группы максимально близких друг к другу объектов, которые в силу гипотезы схожести и будут образовывать наши кластеры.

## Входные данные

- Кластеризуемая выборка - может быть задана признаковыми описаниями объектов либо матрицей попарных расстояний между объектами.
- Параметр  $R$  — радиус поиска локальных сгущений (можно задавать как из априорных соображений, так и настраивать скользящим контролем).
- В модификациях возможно введение параметра  $k$  — количества кластеров.

## Выходные данные

- Кластеризация на заранее неизвестное число таксонов.

## Минимизируемый алгоритмом функционал качества

$$F = \sum_{j=1}^k \sum_{x \in K_j} \rho(x, W_j),$$

где первое суммирование ведется по всем кластерам выборки, второе суммирование — по всем объектам  $x$ , принадлежащим текущему кластеру  $K_j$ , а  $W_j$  — центр текущего кластера,  $\rho(x, y)$  — расстояние между объектами.

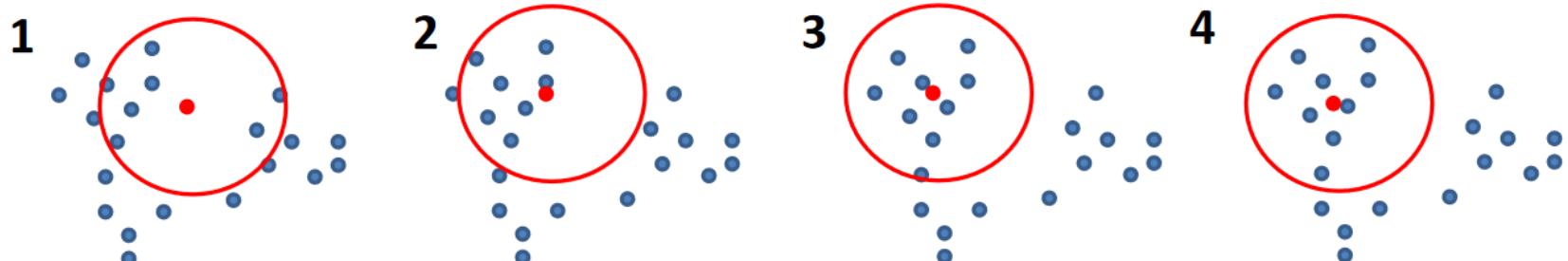
# FOREL- Алгоритм

## Алгоритм

1. Случайно выбираем текущий объект из выборки;
2. Помечаем объекты выборки, находящиеся на расстоянии менее, чем  $R$  от текущего;
3. Вычисляем их центр тяжести, помечаем этот центр как новый текущий объект;
4. Повторяем шаги 2-3, пока новый текущий объект не совпадет с прежним;
5. Помечаем объекты внутри сферы радиуса  $R$  вокруг текущего объекта как кластеризованные, выкидываем их из выборки;
6. Повторяем шаги 1-5, пока не будет кластеризована вся выборка.

## Выбор центра тяжести:

- В линейном пространстве — центр масс;
- В метрическом пространстве — объект, сумма расстояний до которого минимальна, среди всех внутри сферы;
- Объект, который внутри сферы радиуса  $R$  содержит максимальное количество других объектов из всей выборки (медленно);
- Объект, который внутри сферы маленького радиуса содержит максимальное количество объектов (из сферы радиуса  $R$ ).



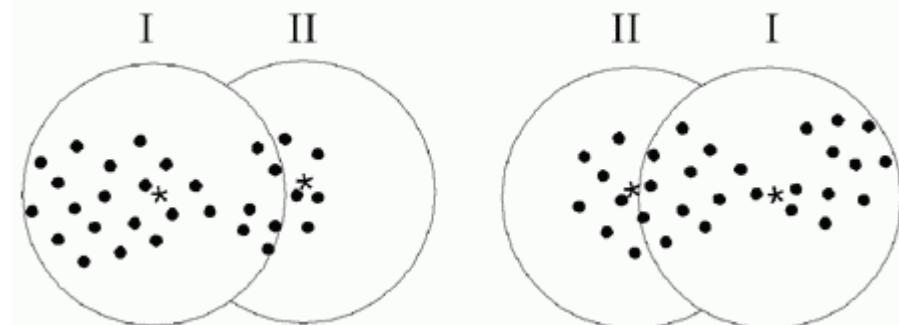
# FOREL- Преимущества и Недостатки

## Преимущества

- Точность минимизации функционала качества (при удачном подборе параметра R);
- Наглядность визуализации кластеризации;
- Сходимость алгоритма;
- Возможность операций над центрами кластеров — они известны в процессе работы алгоритма;
- Возможность подсчета промежуточных функционалов качества, например, длины цепочки локальных сгущений;
- Возможность проверки гипотез схожести и компактности в процессе работы алгоритма.

## Недостатки

- Относительно низкая производительность (решается введением функции пересчета поиска центра при добавлении 1 объекта внутрь сферы);
- Плохая применимость алгоритма при плохой разделимости выборки на кластеры;
- Неустойчивость алгоритма (зависимость от выбора начального объекта);
- Произвольное по количеству разбиение на кластеры;
- Необходимость априорных знаний о ширине (диаметре) кластеров.



# DBSCAN

*Density-based spatial clustering of applications with noise (DBSCAN)* - Основанная на плотности пространственная кластеризация для приложений с шумами.

Если дан набор точек в некотором пространстве, алгоритм группирует вместе точки, которые тесно расположены (точки со многими близкими соседями), помечая как выбросы точки, которые находятся одиноко в областях с малой плотностью (ближайшие соседи которых лежат далеко). **DBSCAN** развивает идею кластеризации с помощью выделения связных компонент.

Плотность в DBSCAN определяется в окрестности каждого объекта выборки  $x_i$  как количество других точек выборки в шаре  $B(\varepsilon, x_i)$ . Кроме радиуса  $\varepsilon$  окрестности в качестве гиперпараметра алгоритма задается порог **MinPts** по количеству точек в окрестности.

Все объекты выборки делятся на три типа:

- **внутренние / основные точки (core points),**
- **границные/ достижимые по плотности (border points)**
- **шумовые/ выпадающие точки (noise points).**

К основным относятся точки, в окрестности которых больше *MinPts* объектов выборки. К границным — точки, в окрестности которых есть основные, но общее количество точек в окрестности меньше *MinPts*. Шумовыми называют точки, в окрестности которых нет основных точек и в целом содержится менее *MinPts* объектов выборки.

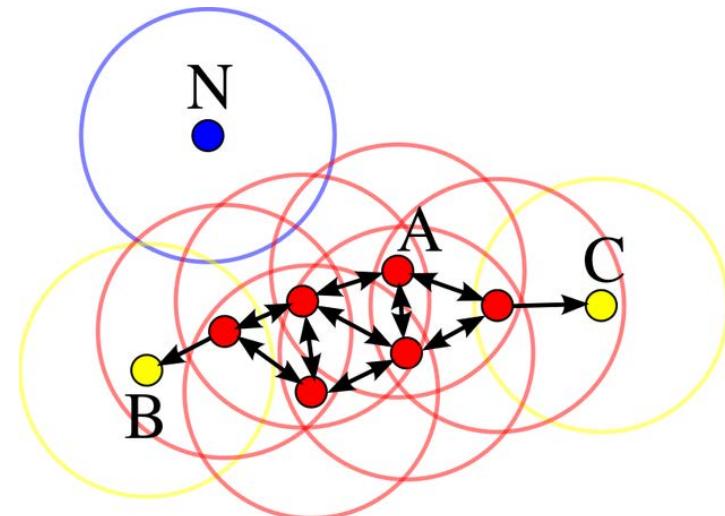
# DBSCAN - Алгоритм

1. Выделяются основные точки.
2. Основные точки, у которых есть общая окрестность, соединяются ребром. Говорят, что эти точки *прямо достижимы*.
3. В полученном графе выделяются компоненты связности.
4. Каждая граничная точка (*достижимая* точка) относится к тому кластеру, в который попала ближайшая к ней основная точка.
5. Шумовые точки (не достижимые из основных точек) убираются из рассмотрения и не приписываются ни к какому кластеру.

Если  $A$  является основной точкой, то она формирует *кластер* вместе со всеми точками (основными или неосновными), достижимые из этой точки. Каждый кластер содержит по меньшей мере одну основную точку. Неосновные точки могут быть частью кластера, но они формируют его «край», поскольку не могут быть использованы для достижения других точек.

Две точки  $P$  и  $Q$  связаны по плотности, если имеется точка  $O$ , такая что и  $P$ , и  $Q$  достижимы из  $O$ . Кластер удовлетворяет двум свойствам:

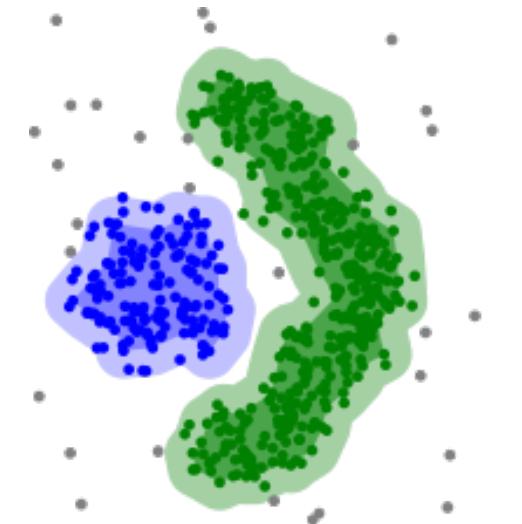
- Все точки в кластере попарно связны по плотности.
- Если точка достижима по плотности из какой-то точки кластера, она также принадлежит кластеру.



# DBSCAN - Преимущества и Недостатки

## Преимущества

- Не требует спецификации числа кластеров в данных.
- Может найти кластеры произвольной формы.
- Имеет понятие шума и устойчив к выбросам.
- Требует лишь двух параметров и большей частью нечувствителен к порядку выбора точек.
- Параметры  $\text{minPts}$  и  $\varepsilon$  могут быть установлены экспертами в рассматриваемой области, если данные хорошо понимаются.



## Недостатки

- Не полностью однозначен — краевые точки, которые могут быть достигнуты из более чем одного кластера, могут принадлежать любому из этих кластеров, что зависит от порядка просмотра точек.
- Качество DBSCAN зависит от измерения расстояния, используемого для подсчета количества других точек выборки в сфере  $B(\varepsilon, x_i)$ .
- Не может хорошо кластеризовать наборы данных с большой разницей в плотности, поскольку не удается выбрать приемлемую для всех кластеров комбинацию  $\text{minPts}$  и  $\varepsilon$ .
- Если есть сложности с пониманием особенностей данных и масштаба, выбор порога расстояния  $\varepsilon$  может оказаться проблематичным.

# OPTICS

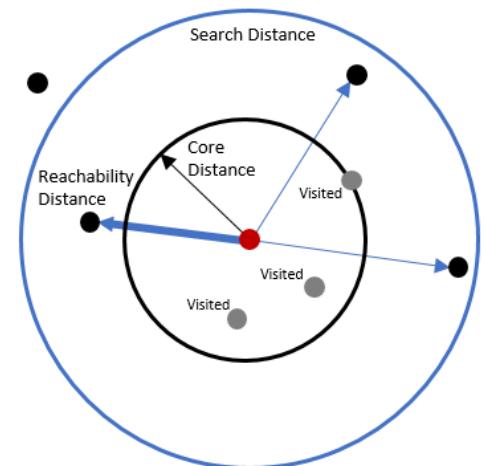
*Ordering points to identify the clustering structure (OPTICS)* - Упорядочение точек для обнаружения кластерной структуры — это алгоритм нахождения кластеров в пространственных данных на основе плотности. Основная идея алгоритма похожа на DBSCAN, но алгоритм предназначен для избавления от одной из главных слабостей алгоритма DBSCAN — проблемы обнаружения содержательных кластеров в данных, имеющих различные плотности. Чтобы это сделать, точки имеющихся данных (линейно) упорядочиваются так, что пространственно близкие точки становятся соседними в упорядочении. Кроме того, для каждой точки запоминается специальное расстояние, представляющее плотность, которую следует принять для кластера, чтобы точки принадлежали одному кластеру.

Основное расстояние, которое описывает расстояние до  $MinPts$ -ой ближайшей точки:

$$\text{core-dist}_{\varepsilon, MinPts} = \begin{cases} \text{UNDEFINED} & |N_\varepsilon(p)| < MinPts \\ MinPts-\text{th}N_\varepsilon(p) & |N_\varepsilon(p)| \geq MinPts \end{cases}$$

Достигимое расстояние точки  $o$  от точки  $p$ :

$$\text{reachability-dist}_{\varepsilon, MinPts}(o, p) = \begin{cases} \text{UNDEFINED} & |N_\varepsilon(p)| < MinPts \\ \max(\text{core-dist}_{\varepsilon, MinPts}(p), \text{dist}(p, o)) & |N_\varepsilon(p)| \geq MinPts \end{cases}$$



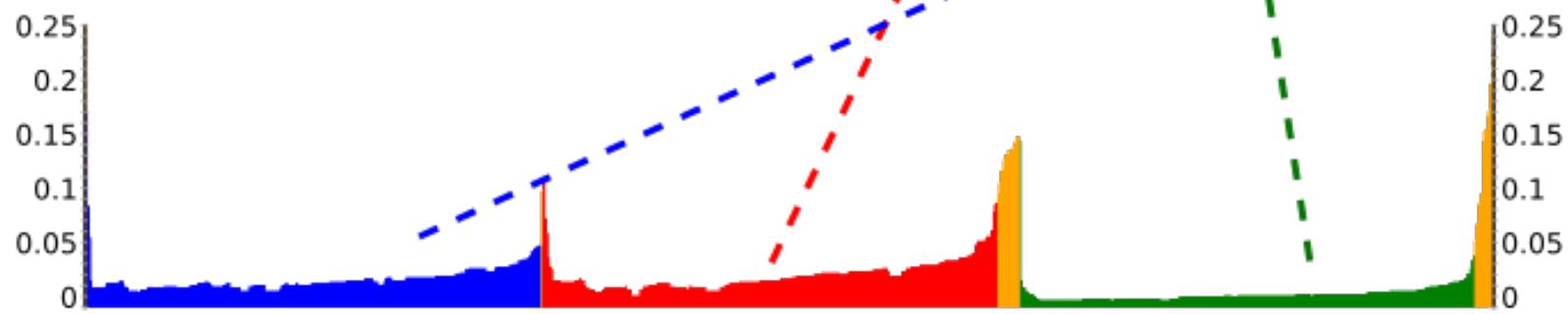
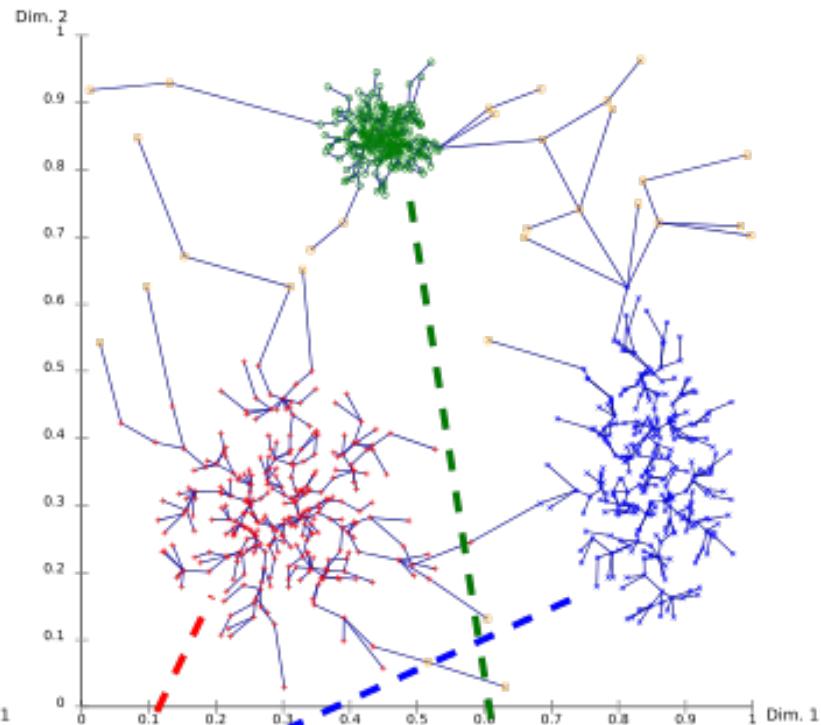
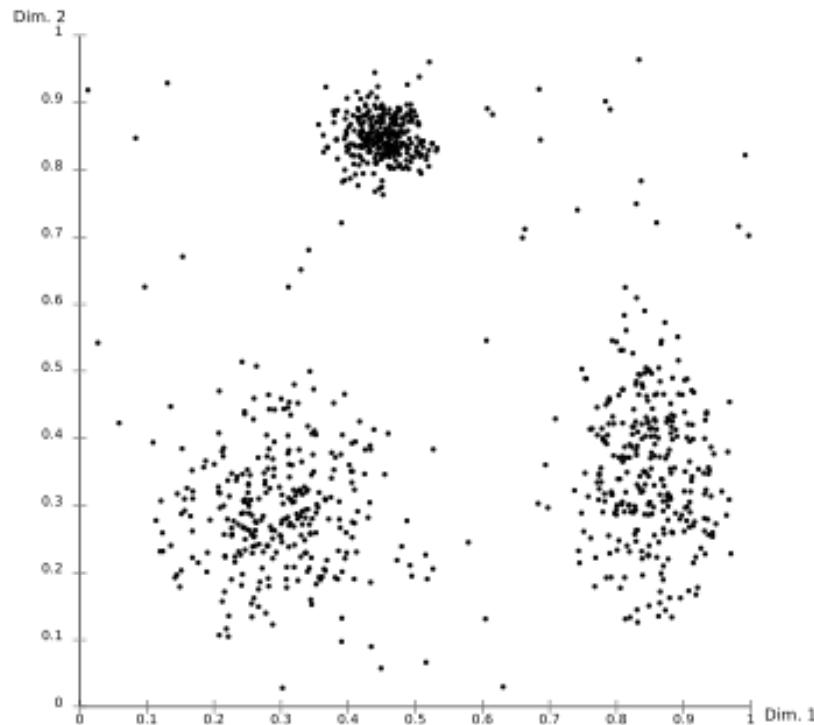
# OPTICS - Алгоритм

```
OPTICS(DB, eps, MinPts)
для каждой точки p из DB
    p.достижимое_расстояние=не_определен
для каждой необработанной точки p из DB
    N=получитьСоседей (p, eps)
    пометить p как обработанную
    поместить p в упорядоченный список
    если (основное_расстояние (p, eps, Minpts) != не_определен)
        Seeds=пустая приоритетная очередь
        обновить(N, p, Seeds, eps, Minpts)
        для каждой следующей q из Seeds
            N'=получитьСоседей(q, eps)
            пометить q как обработанную
            поместить q в упорядоченный список
            если (основное_расстояние(q, eps, Minpts) != не_определен)
                обновить(N', q, Seeds, eps, Minpts)
```

---

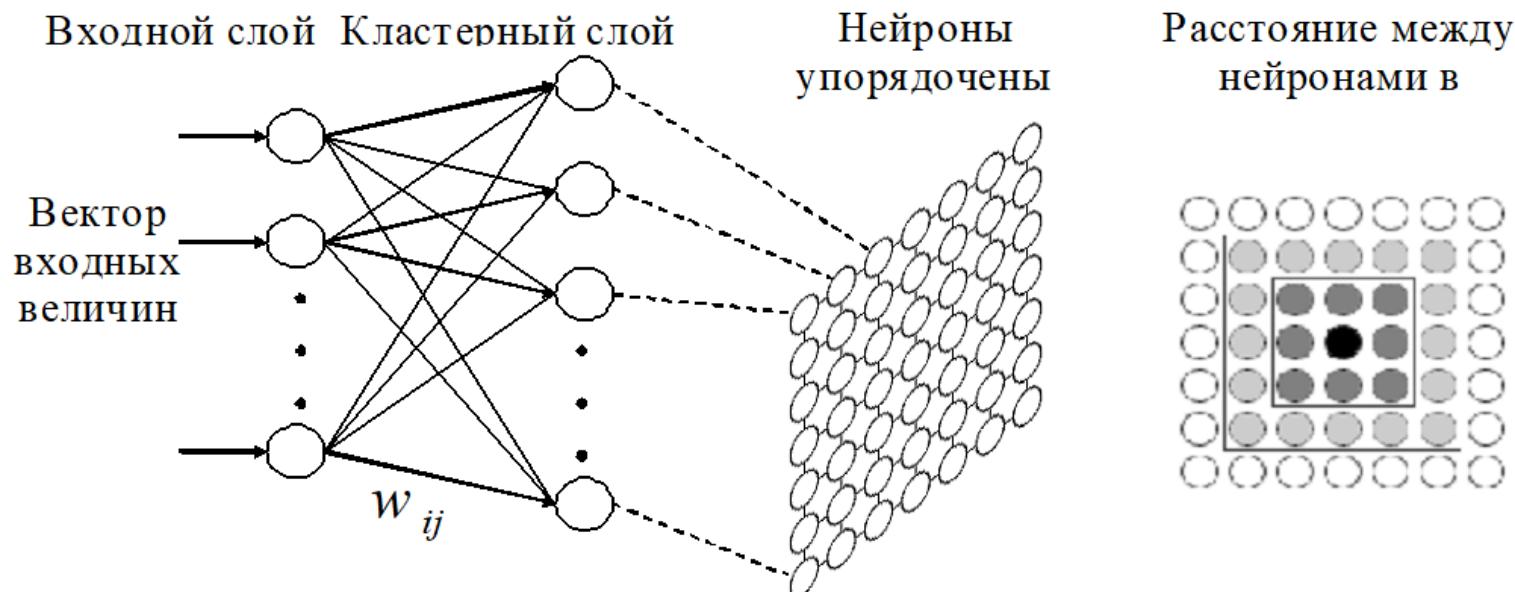
```
обновить (N, p, Seeds, eps, Minpts)
coredist=основное_расстояние(p, eps, MinPts)
для каждого o в N
    если (o не обработана)
        новое_дост_расст=max(coredist, dist(p,o))
        если (o.достижимое_расстояние == не_определен) // точка o не в Seeds
            o.достижимое_расстояние=новое_дост_расст
            Seeds.вставить(o, новое_дост_расст)
        иначе // точка o в Seeds, проверить на улучшение
            если (новое_дост_раст < o.достижимое_расстояние)
                o.достижимое_расстояние=новое_дост_раст
                Seeds.передвинуть_вверх(o, новое_дост_раст)
```

# OPTICS – результат разбиения



# Самоорганизующиеся карты Кохонена (SOM)

Состоит из входного слоя и одного конкурирующего кластерного слоя, нейроны которого вычисляют расстояние между входным вектором и вектором весовых коэффициентов. Победителем является тот нейрон, расстояние до которого минимально. Важная особенность алгоритма SOM (Self Organizing Maps) заключается в том, что все нейроны кластерного слоя (ядра классов) упорядочены в некоторую структуру, что позволяет ввести меру взаимодействия между нейронами кластерного слоя не в пространстве входных признаков, а на используемой карте. Величина этого взаимодействия определяется расстоянием между нейронами на карте.



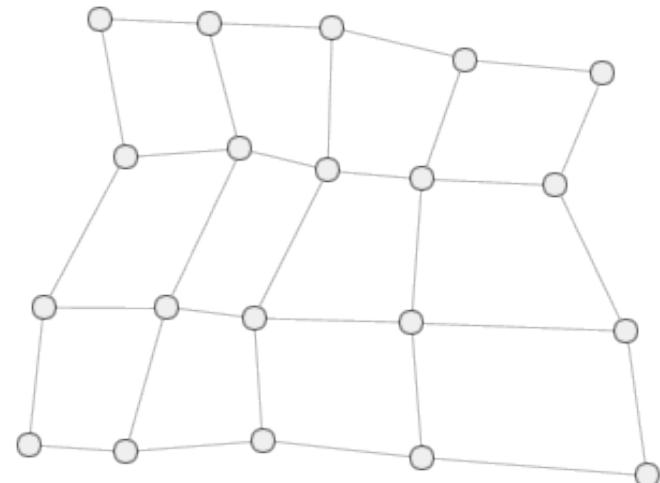
# SOM - Обучение сети

- инициализацию сети (количество нейронов в сети и условие взаимодействия между нейронами кластерного слоя)
- инициализация весовых коэффициентов
  - Инициализация случайными значениями
  - Инициализация примерами
  - Линейная инициализация.
- Обучение (корректировка весов). Подстройке подлежат веса, как нейрона-победителя, так и его соседей по сетке (одно или двумерной), но в меньшей степени.

$$w_i(t+1) = w_i(t) + h_{ci}(t) * [x(t) - w(t)]$$

$$h(t) = h(\|a_c - a_i\|, t) * f(t)$$

$$h(d, t) = e^{-\left(\frac{d}{b(t)}\right)^2}$$



# SOM - Раскраска обученной карты

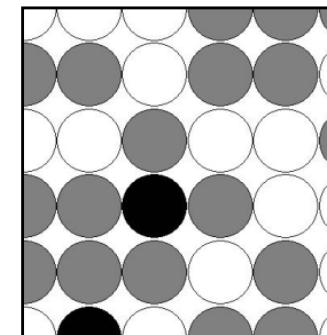
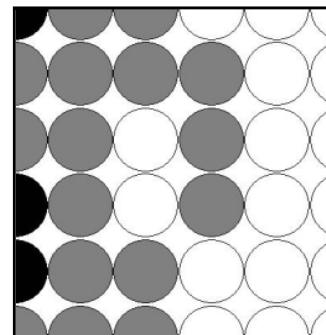
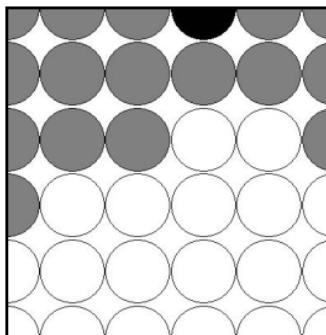
Добавление дополнительных связей в пространстве отображения результата позволяют использовать сеть Кохонена, как для решения задач распознавания и определения близости кластеров в данных, так и для визуализации многомерного пространства

- в соответствии с расстояниями между векторами весовых коэффициентов нейронов кластерного слоя;
- в зависимости от значений некоторой компоненты вектора обучающей выборки;
- в зависимости от числа примеров отнесенных к соответствующему нейрону карты.

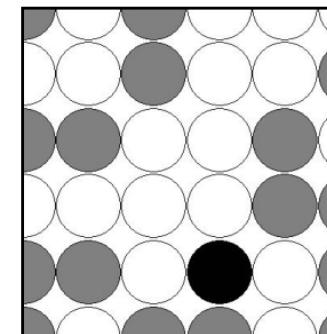
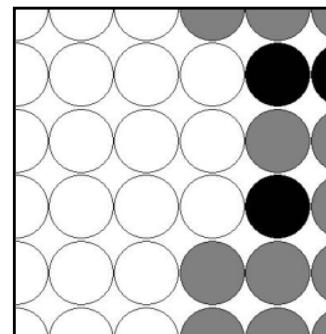
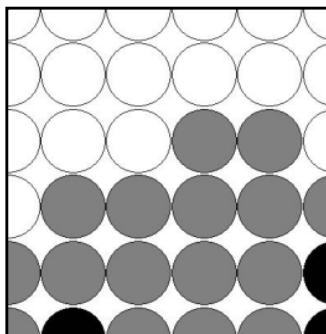
В совокупности полученные раскраски образуют атлас, отображающий присутствующие в данных кластеры близких объектов, зависимости от значений отдельных или наборов компонент, относительное расположение их различных значений, что в значительной степени может помочь пользователю разобраться в структуре обрабатываемых данных.

# Пример раскраски обученной SOM

Изолированная  
БА



БА + ЦМА



(A)

(B)

(C)

На рисунке представлены результаты обучения и раскраски SOM-сетей, как для всей совокупности измеренных ДТ-МРТ показателей всех областей мозга (A), так и для отдельных анатомических структур на примере наиболее характерных случаев, когда классы разделимы (B) и не разделимы (C). Белый цвет нейрона говорит о том, что он не был активирован ни одним из примеров выборки указанного класса. Градации серого характеризуют число примеров выборки, принадлежащих соответствующему нейрону.

**Спасибо за внимание**

# Методы машинного обучения

*Лекция 14*

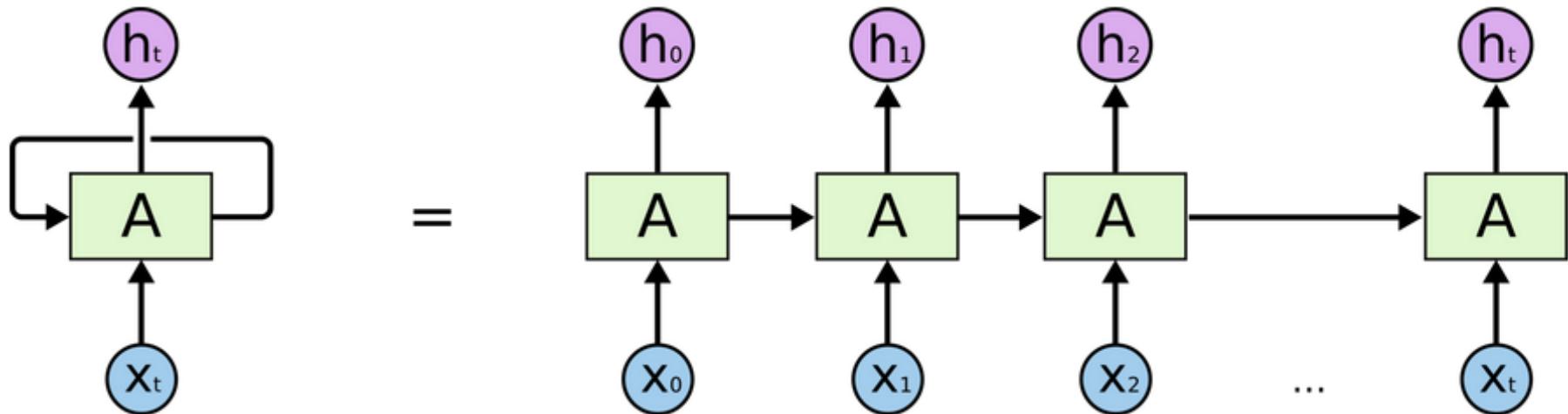
Рекуррентные нейронные сети

*или как правильно кусать себя за хвост*

# Рекуррентные нейронные сети

Рекуррентные нейронные сети (Recurrent Neural Networks, RNN) - это сети, содержащие обратные связи, позволяющие сохранять информацию (скрытое состояние) от предыдущих входных данных. Хорошо подходят для обработки последовательностей, например, временные ряды (изменения цен акций, показания датчиков), последовательности с зависимыми элементами (предложения естественного языка), т.е. любые данные, где соседние экземпляры (точки выборки) зависят друг от друга и эту зависимость нельзя игнорировать.

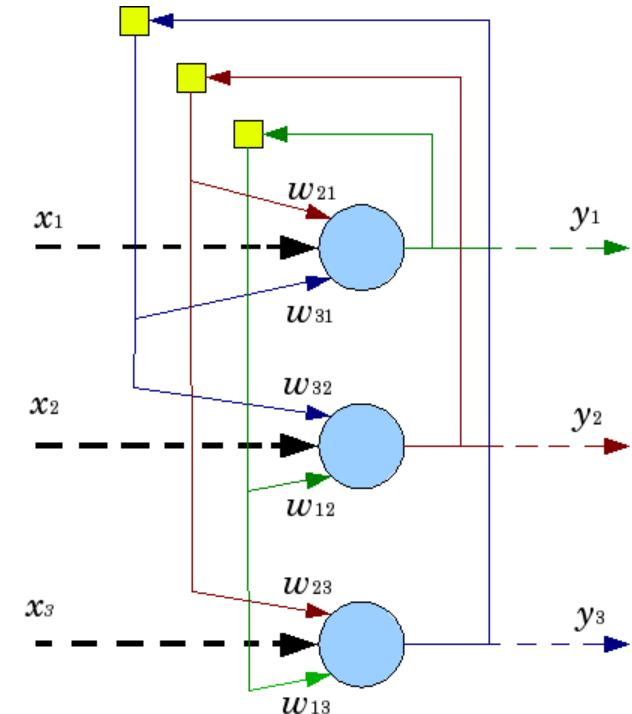
Фрагмент нейронной сети A принимает входное значение  $x_t$  и возвращает значение  $h_t$ . Наличие обратной связи позволяет передавать информацию от одного шага сети к другому. Рекуррентную сеть можно рассматривать, как несколько копий одной и той же сети, каждая из которых передает информацию последующей копии.



# Нейронная сеть Хопфилда

**Нейронная сеть Хопфилда** - полносвязная однослойная нейронная сеть с симметричной матрицей связей. Функционирование до достижения равновесия, т.е. когда следующее состояние сети в точности равно предыдущему: начальное состояние является входным образом, а при равновесии получают выходной образ. N искусственных нейронов. Каждый нейрон системы может принимать на входе и на выходе одно из двух состояний (что аналогично выходу нейрона с пороговой функцией активации):

$$y_i(t) \in \{-1; +1\}$$



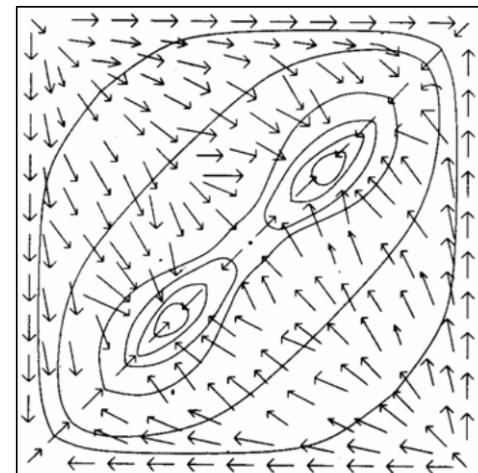
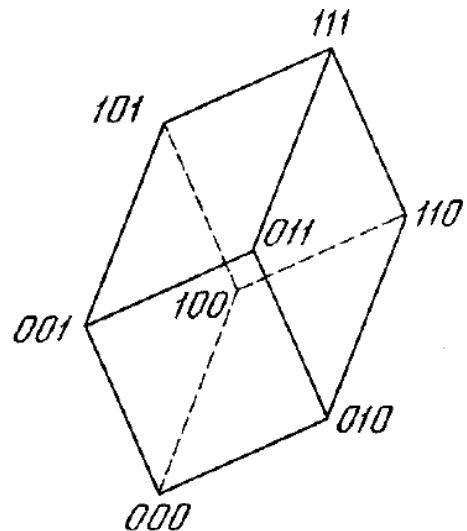
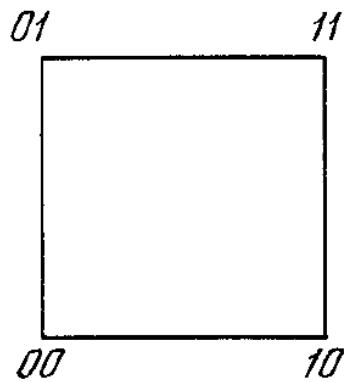
Каждый нейрон связан со всеми остальными нейронами. Взаимодействие нейронов сети описывается выражением:

$$E = \frac{1}{2} \sum_{i,j=1}^N w_{ij} x_i x_j$$

где  $w_{ij}$  — элемент матрицы взаимодействий  $W$ , которая состоит из весовых коэффициентов связей между нейронами.

# Нейронная сеть Хопфилда - Состояние сети

*Состояние сети* – это просто множество текущих значений сигналов  $y_i$  от всех нейронов. В первоначальной сети Хопфилда состояние каждого нейрона менялось в дискретные случайные моменты времени, в последующей работе состояния нейронов могли меняться одновременно. Так как выходом бинарного нейрона может быть только два значения (промежуточных уровней нет), то текущее состояние сети является двоичным числом, каждый бит которого является сигналом  $y_i$  некоторого нейрона.



# Нейронная сеть Хопфилда - Обучение

В процессе обучения формируется выходная матрица  $W$ , которая запоминает  $m$  эталонных «образов» —  $N$ -мерных бинарных векторов:  $X_m = (x_{m1}, x_{m2}, \dots, x_{mN})$ , эти образы во время эксплуатации сети будут выражать отклик системы на входные сигналы, или иначе - окончательные значения выходов  $y_i$  после серии итераций.

Вычисление коэффициентов основано на следующем правиле: для всех запомненных образов  $X_i$  матрица связи должна удовлетворять уравнению :

$$X_i = WX_i$$

Расчёт весовых коэффициентов проводится по следующей формуле:

$$w_{ij} = \frac{1}{N} \sum_{d=1..m} X_{id} X_{jd}$$

где  $N$  — размерность векторов,  $m$  — число запоминаемых выходных векторов,  $d$  — номер запоминаемого выходного вектора,  $X_{ij}$  —  $i$ -я компонента запоминаемого выходного  $j$ -го вектора.

Может быть записано в векторном виде :

$$W = \frac{1}{N} \sum_i X_i X_i^T$$

где  $X_i$  —  $i$ -й запоминаемый вектор-столбец.

В работе Кохонена и Гроссберга (доказана теорема) показано, что сеть с обратными связями является устойчивой, если её матрица симметрична и имеет нули на главной диагонали.

# Нейронная сеть Хопфилда – Применение обученной сети

Обученная сеть способна распознавать входные сигналы - то есть, определять, к какому из запомненных образцов они относятся. Сеть последовательно меняет свои состояния согласно формуле:

$$X(t + 1) = F(W \cdot X(t))$$

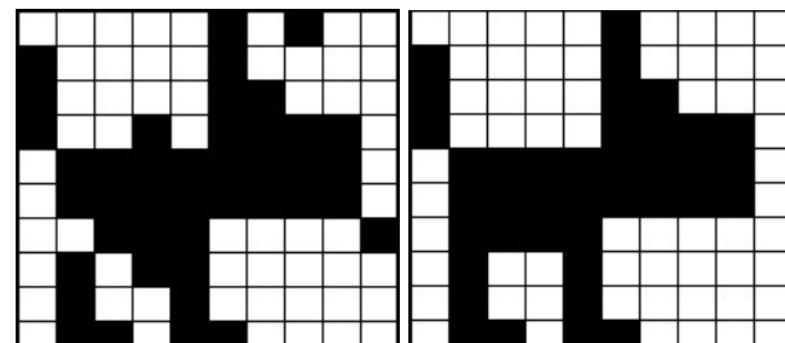
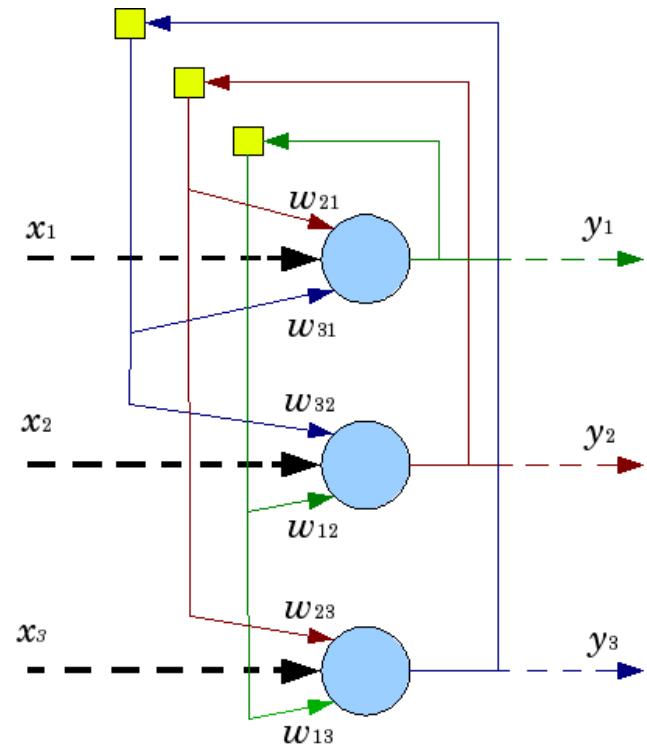
где  $F$  – активационная функция,  $X(t)$  и  $X(t + 1)$  – текущее и следующее состояния сети.

Локальным полем  $a_i$ , действующим на нейрон  $x_i$  со стороны всех остальных нейронов сети, является значение:

$$a_i(t) = \sum_{j=1, j \neq i}^N w_{ji} x_j(t - 1)$$

Значение выхода нейрона  $i$  в текущий момент времени  $x_i(t)$  рассчитывается по формуле:

$$x_i(t) = sign \left( \sum_{j=1, j \neq i}^N w_{ji} x_j(t - 1) \right)$$



Искажённый образ

Эталон

# **Нейронная сеть Хопфилда – Режимы работы**

Две модификации (режима работы), отличающиеся по времени передачи сигнала: **Синхронный** и **Асинхронный**.

**Синхронный режим работы сети** – последовательно просматриваются нейроны, их состояния запоминаются отдельно и не меняются до тех пор, пока не будут пройдены все нейроны сети. Когда все нейроны просмотрены, их состояния одновременно (то есть синхронно) меняются на новые.

**Асинхронный режим работы сети** – состояния нейронов в следующий момент времени меняются последовательно: вычисляется локальное поле для первого нейрона в момент  $t$ , определяется его реакция, и нейрон устанавливается в новое состояние (которое соответствует его выходу в момент  $t + 1$ ), и так далее — состояние каждого следующего нейрона вычисляется с учетом всех изменений состояний рассмотренных ранее нейронов.

В асинхронном режиме невозможен динамический аттрактор: вне зависимости от количества запомненных образов и начального состояния сеть непременно придёт к устойчивому состоянию (статическому аттрактору).

# Нейронная сеть Хопфилда – Ограничения сети

1. Относительно небольшой объём памяти, величину которого можно оценить выражением:

$$M \approx \frac{N}{2 \cdot \log_2 N}$$

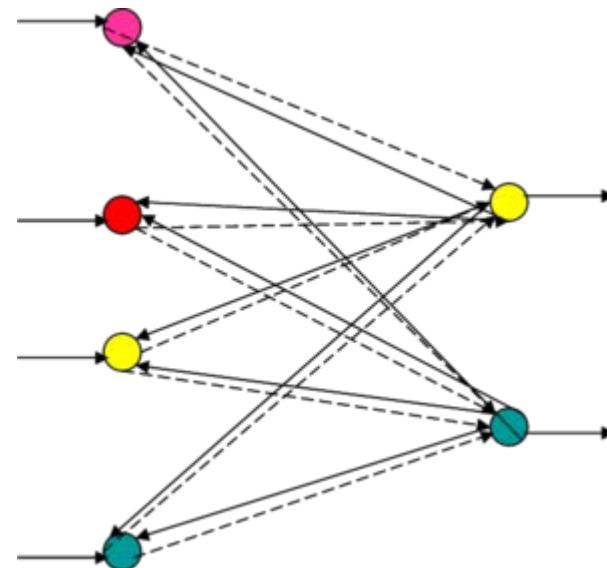
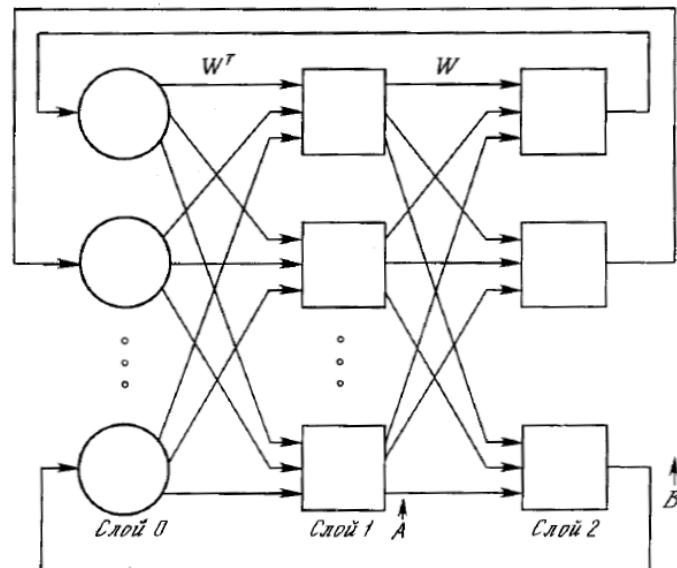
Попытка записи большего числа образов приводит к тому, что нейронная сеть перестаёт их распознавать.

2. В синхронном режиме сеть может прийти к динамическому аттрактору.
3. Достижение устойчивого состояния не гарантирует правильный ответ сети. Это происходит из-за того, что сеть может сойтись к так называемым ложным аттракторам, иногда называемым «химерами» (как правило, химеры склеены из фрагментов различных образов возникают при слишком большом количестве запомненных образов).

# Двунаправленная ассоциативная память

Двунаправленная ассоциативная память (ДАП) является гетероассоциативной; входной вектор поступает на один набор нейронов, а соответствующий выходной вектор вырабатывается на другом наборе нейронов.

На рисунке приведены две базовые конфигурации ДАП. Одна из них выбрана таким образом, чтобы подчеркнуть сходство с сетями Хопфилда и предусмотреть увеличения количества слоев.



# ДАП – работа сети

Входной вектор  $A$  обрабатывается матрицей весов  $W$  сети, в результате чего вырабатывается вектор выходных сигналов нейронов  $B$ . Вектор  $B$  затем обрабатывается транспонированной матрицей  $W^T$  весов сети, которая вырабатывает новые выходные сигналы, представляющие собой новый входной вектор  $A$ . Этот процесс повторяется до тех пор, пока сеть не достигнет стабильного состояния, в котором ни вектор  $A$ , ни вектор  $B$  не изменяются.

В векторной форме:

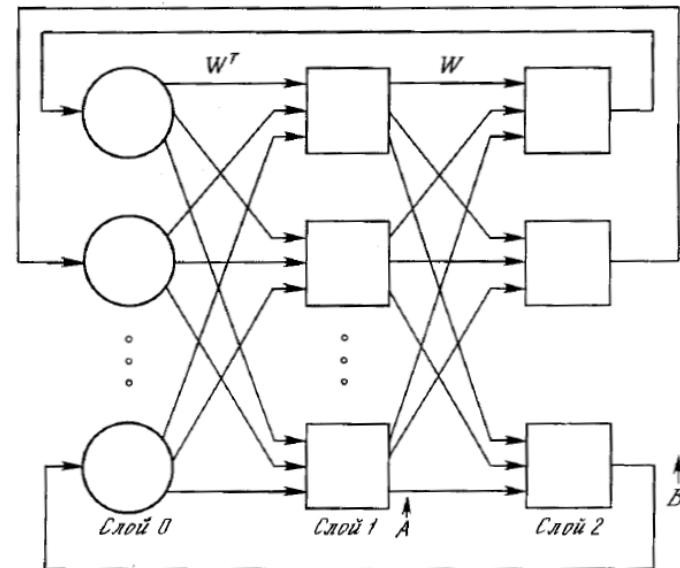
$$B = F(AW)$$

где  $B$  – вектор выходного сигнала нейронов слоя 2,  $A$  – вектор выходных сигналов нейрона слоя 1,  $W$  – матрица весов связей между слоями 1 и 2,  $F$ - функция активации.

$$A = F(BW^T)$$

где  $W^T$  – является транспонированной матрицей  $W$ .

Сеть функционирует в направлении минимизации функции энергии Ляпунова в основном таким же образом, как и сети Хопфилда в процессе сходимости. Таким образом, каждый цикл модифицирует систему в направлении энергетического минимума, расположение которого определяется значениями весов.



# ДАП – Кодирование ассоциаций и Емкость памяти

Обучение производится с использованием обучающего набора, состоящего из пар векторов А и В. Процесс обучения реализуется в форме вычислений; это означает, что весовая матрица вычисляется как сумма произведении всех векторных пар обучающего набора. В символьной форме:

$$W = \sum_i A_i^T B_i$$

Существует взаимосвязь между ДАП и рассмотренными сетями Хопфилда. Если весовая матрица  $W$  является квадратной и симметричной, то  $W = W^T$ . В этом случае, если слои 1 и 2 являются одним и тем же набором нейронов, ДАП превращается в автоассоциативную сеть Хопфилда.

**Емкость сети.** Если  $M$  векторов выбраны случайно и представлены в указанной выше форме, и если  $M$  меньше чем:

$$M_1 \approx \frac{N}{2 \cdot \log_2 N}$$

где  $N$  – количество нейронов в наименьшем слое, тогда все запомненные образы, за исключением «малой части», могут быть восстановлены.

Если все образы должны восстанавливаться,  $M$  должно быть меньше

$$M_2 \approx \frac{N}{4 \cdot \log_2 N}$$

Например, если  $N = 1024$ , тогда  $M_1$  должно быть меньше 51, а  $M_2$  меньше 25.

# Обработка последовательностей

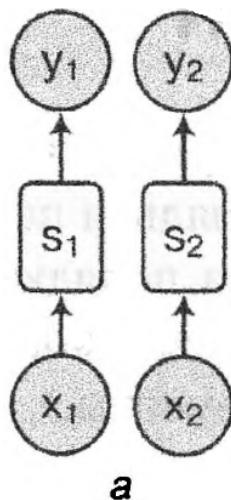
а) Один вход, один выход  
**(one-to-one);**

б) Один вход,  
последовательность выходов  
**(one-to-many);**

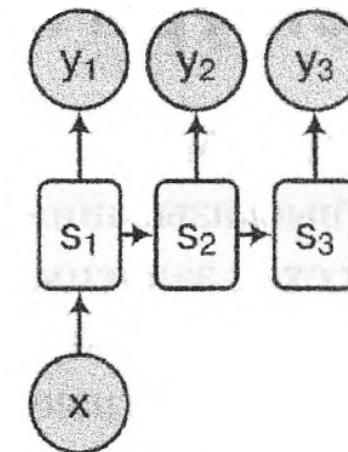
в) Последовательность  
входов, один выход  
**(many-to-one);**

г) Последовательность  
входов, затем  
последовательность выходов  
**(many-to-many);**

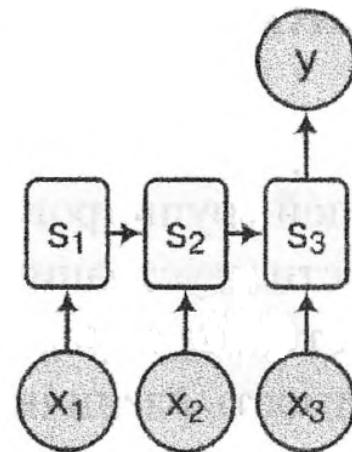
д) Синхронизированные  
последовательности  
входов и выходов  
**(synchronized many-to-many).**



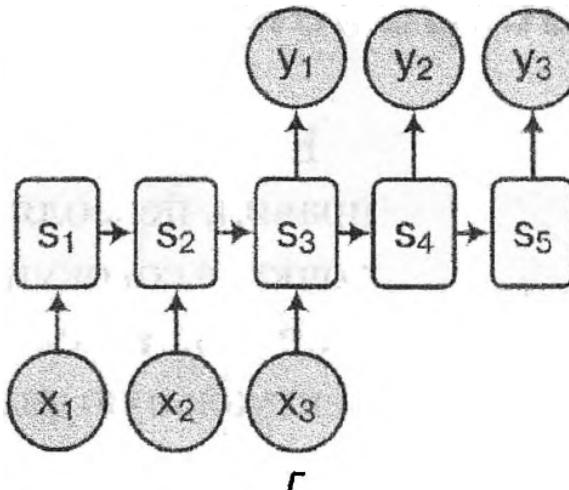
а



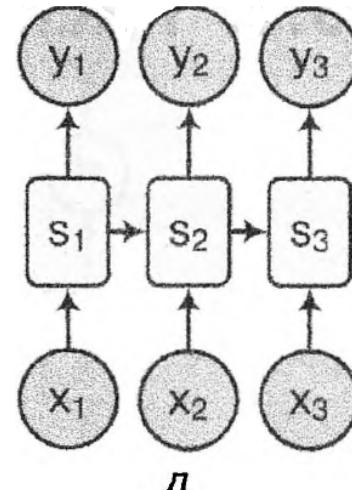
б



в



г



д

# Архитектура «простой» рекуррентной нейронной сети

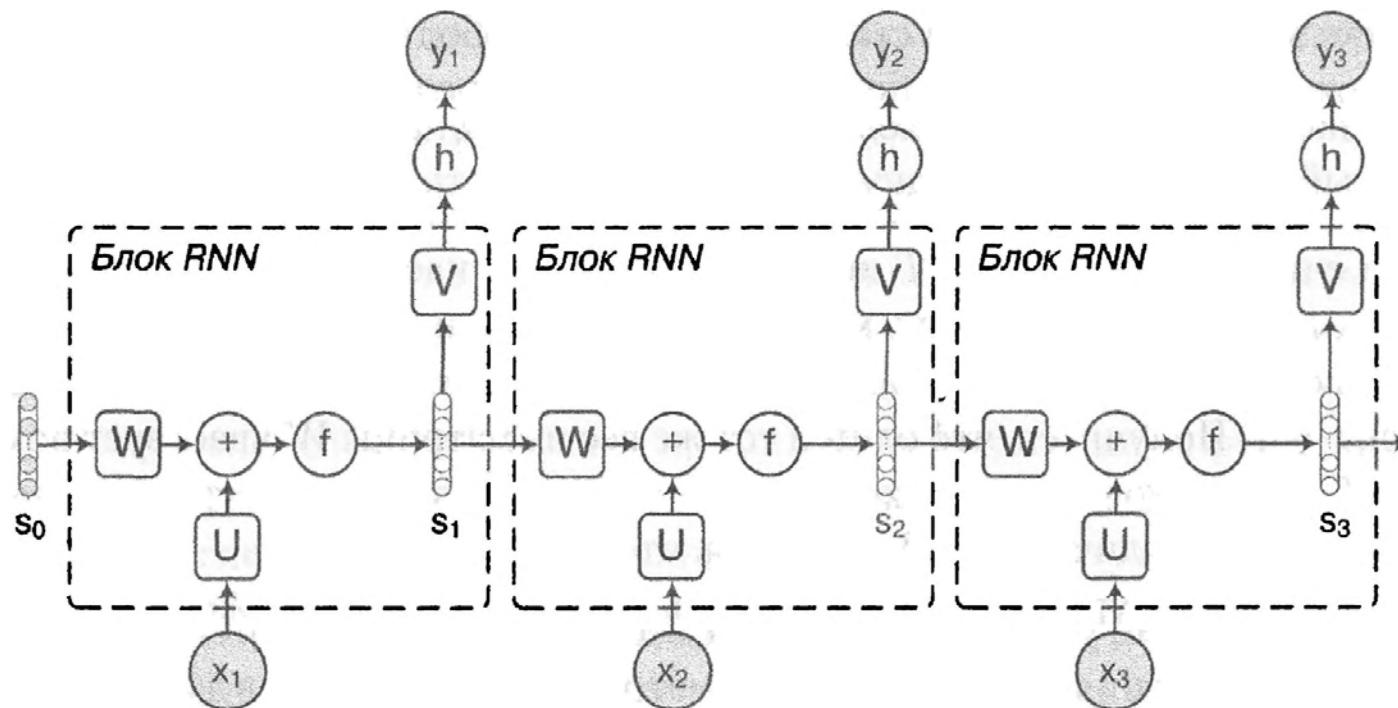
На каждом шаге сеть создает копию самой себя. Каждая из этих копий в определенный момент времени принимает на вход часть последовательности (текущее окно) и значение, полученное из предыдущей копии, затем их комбинирует и передает получившийся результат в следующий элемент.

В итоге в момент времени  $t$  получаем следующее описание сети:

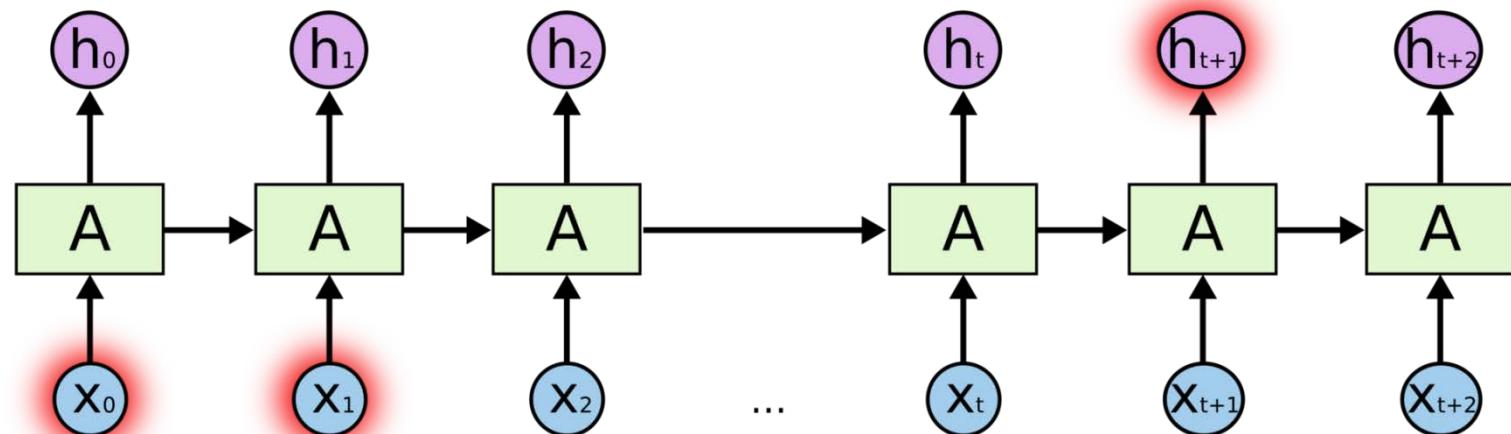
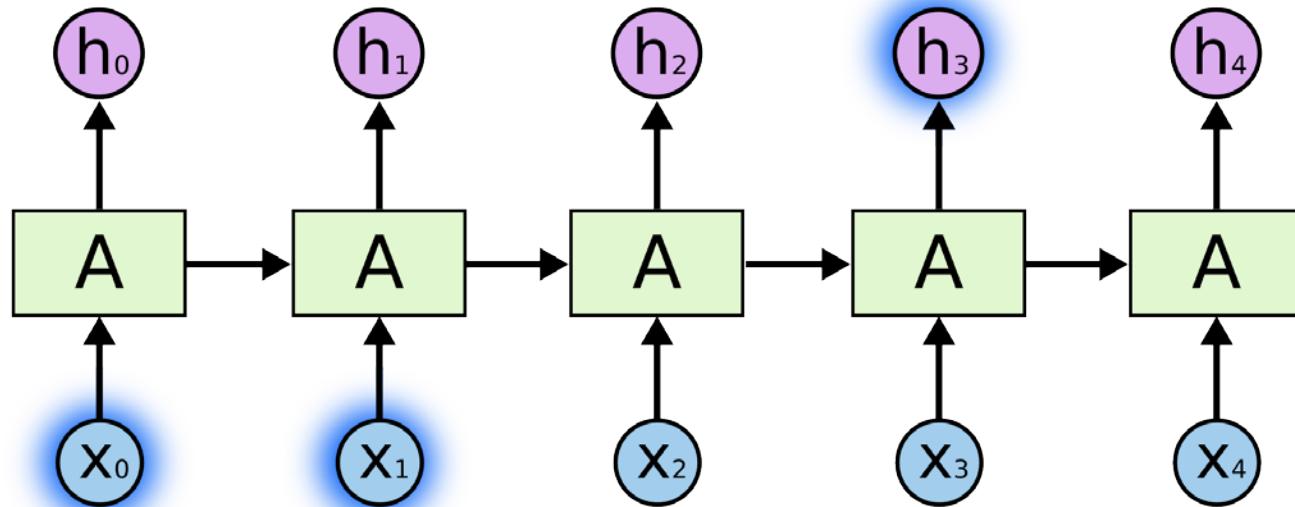
$$a_t = b + Ws_{t-1} + Ux_t \quad s_t = f(a_t)$$

$$o_t = c + Vs_t \quad y_t = h(o_t)$$

где  $f$  — это нелинейность собственно рекуррентной сети (обычно  $\sigma$  - сигмоида,  $\tanh$  или  $\text{ReLU}$ ), а  $h$  — функция, с помощью которой получается ответ (например, softmax).



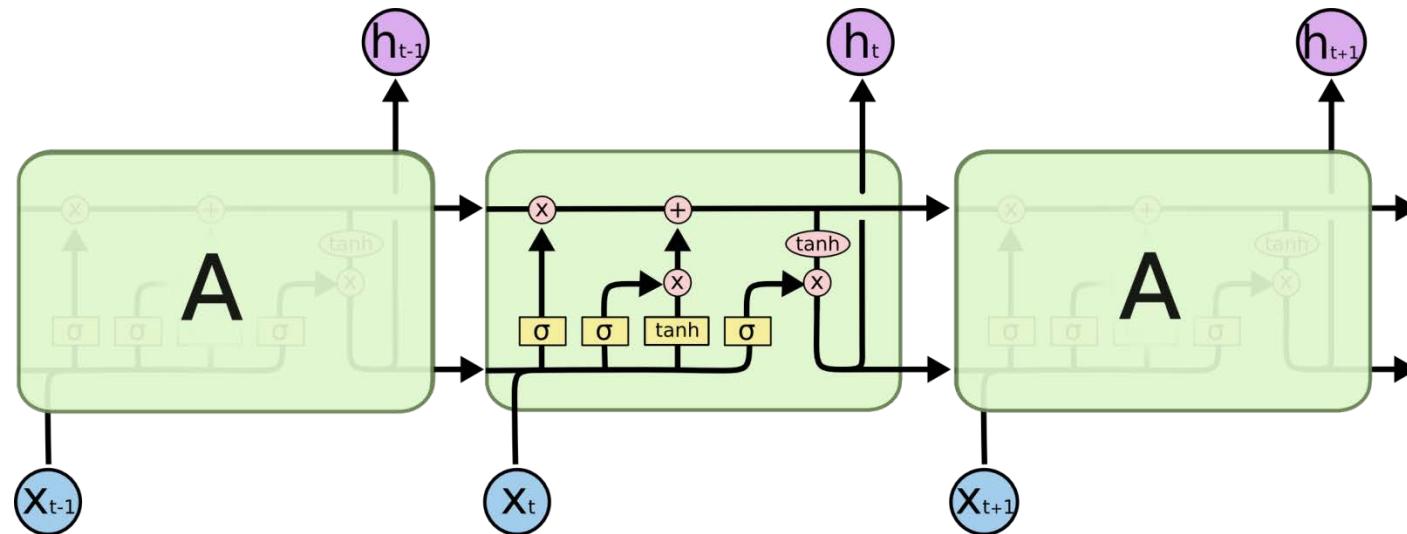
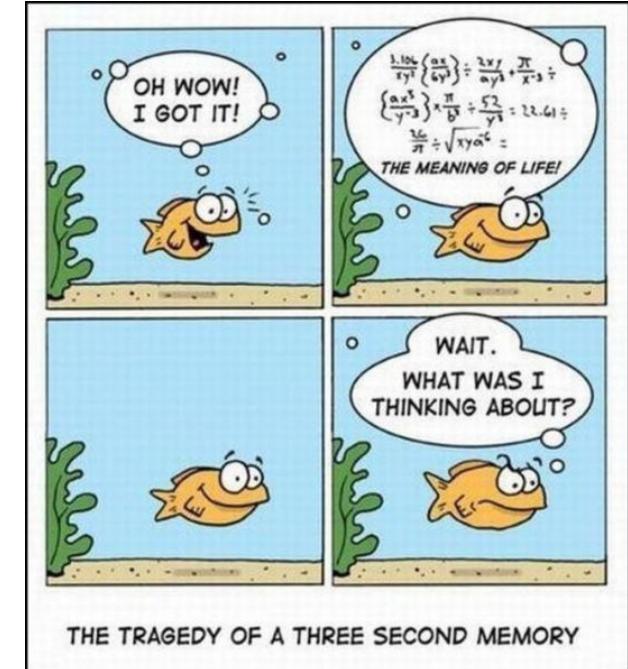
# Проблема долговременных зависимостей



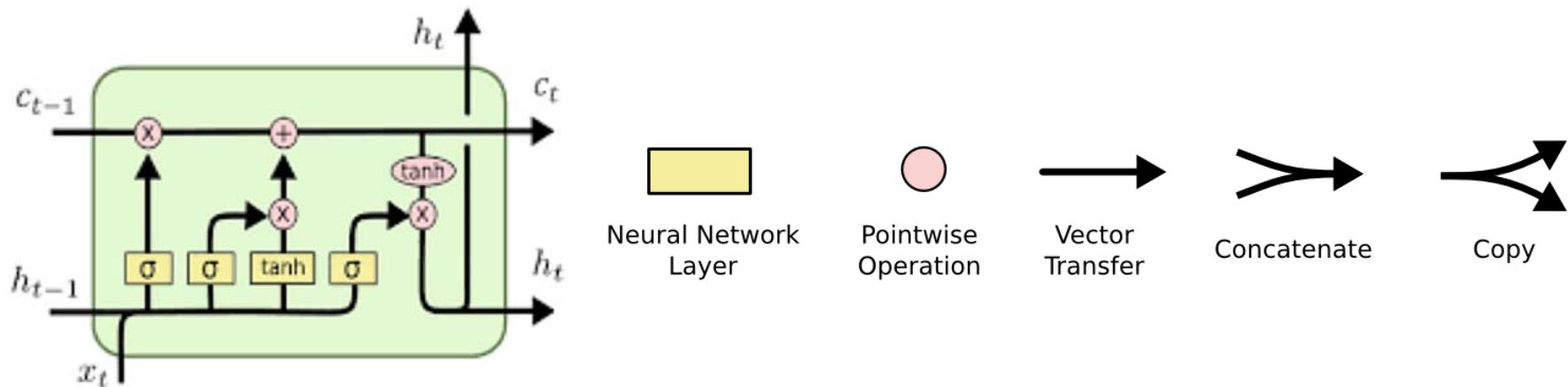
# Долгая краткосрочная память – Long short-term memory (LSTM)

Особая разновидность архитектуры рекуррентных нейронных сетей, способная к обучению долговременным зависимостям, предложенная в 1997 году Сеппом Хохрайтером и Юргеном Шмидхубером.

LSTM-модули разработаны специально, чтобы избежать проблемы долговременной зависимости, запоминая значения как на короткие, так и на длинные промежутки времени. Хранимое значение не размывается во времени и градиент не исчезает при обучении с использованием метода обратного распространения ошибки во времени (Backpropagation Through Time - BPTT).



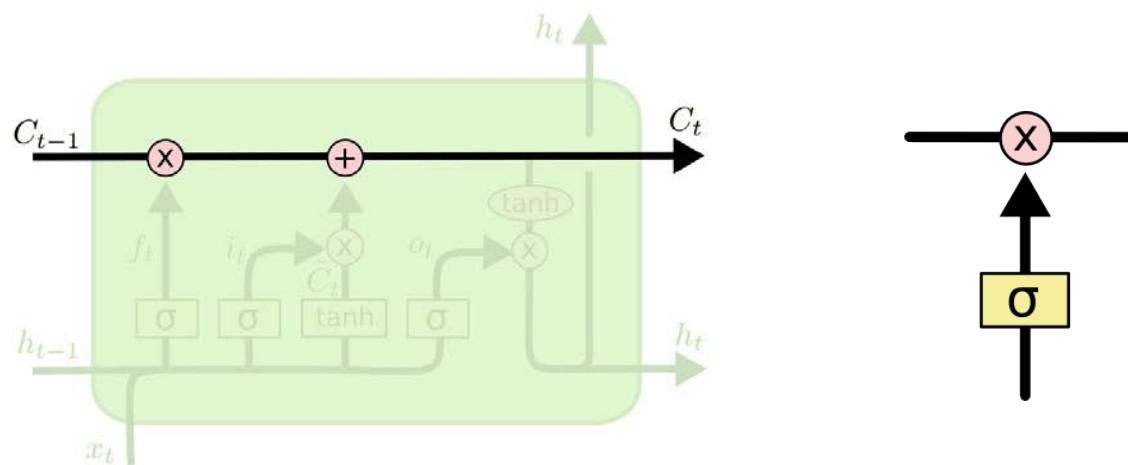
# LSTM - специальные обозначения



- Слой нейронной сети (Желтый прямоугольник);
- Поточечная операция (Розовая окружность) - например, сложение векторов;
- Векторный перенос - каждая линия переносит целый вектор от выхода одного узла ко входу другого;
- Объединение (Сливающиеся линии);
- Копирование (разветвляющиеся стрелки) - данные копируются и копии уходят в разные компоненты сети.

# Ключевые компоненты LSTM-модуля

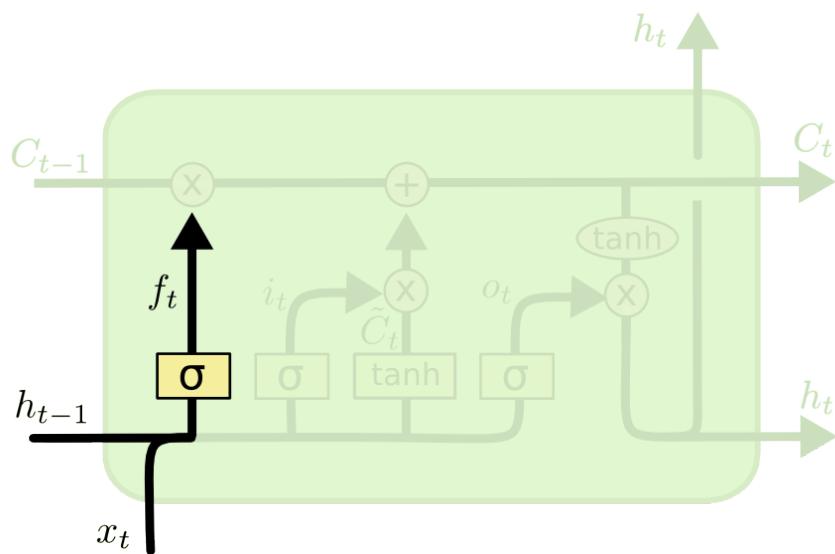
- Состояние ячейки (cell state) - память сети, которая передает соответствующую информацию по всей цепочке модулей.
- **Фильтры**, контролирующие состояние ячейки - контролируют поток информации на входах и на выходах модуля на основании некоторых условий. Состоят из слоя сигмоидальной нейронной сети и операции поточечного умножения.
  - Забывания - контролирует меру сохранения значения в памяти;
  - Входной - контролирует меру вхождения нового значения в память.
  - Выходной - контролирует меру того, в какой степени значение, находящееся в памяти, используется при расчёте выходной функции активации.



# Принцип работы LSTM-модуля – Шаг 1

Определить, какую информацию можно выбросить из состояния ячейки. Для этого используется «слой фильтра забывания» (англ. *forget gate layer*).

Значения предыдущего выхода  $h_{t-1}$  и текущего входа  $x_t$  пропускаются через сигмоидальный слой. Полученные значения находятся в диапазоне [0; 1]. Значения, которые ближе к 0 будут забыты, а к 1 оставлены.

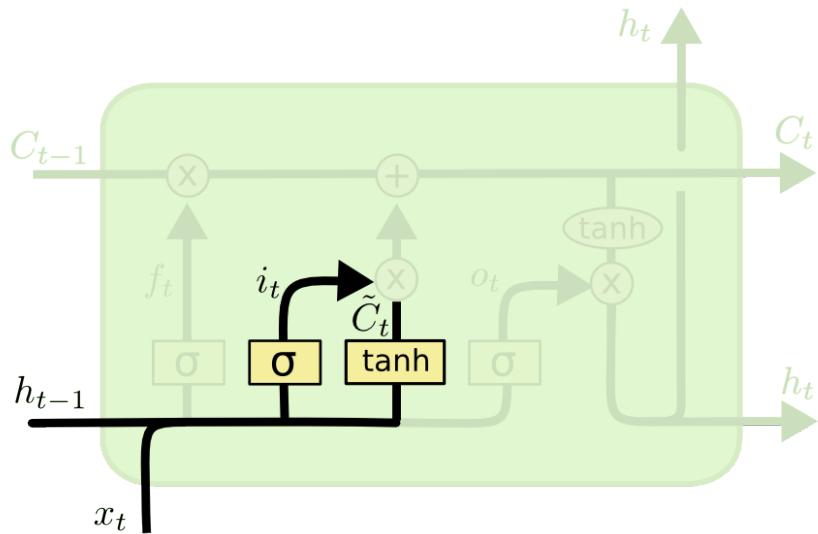


$$f_t = \sigma (W_f \cdot [h_{t-1}, x_t] + b_f)$$

# Принцип работы LSTM-модуля – Шаг 2

Далее решается, какая новая информация будет храниться в состоянии ячейки. Этот этап состоит из двух частей.

- Сначала сигмоидальный слой под названием “**слой входного фильтра**” (англ. *input layer gate*) определяет, какие значения следует обновить.
- Затем  $\tanh$ -слой строит вектор новых значений-кандидатов  $\tilde{C}_t$ , которые можно добавить в состояние ячейки.

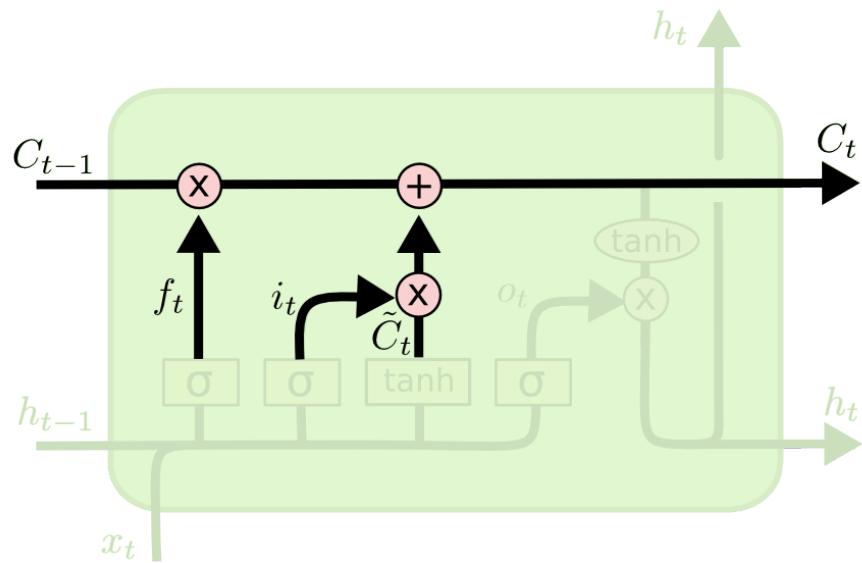


$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$$

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C)$$

# Принцип работы LSTM-модуля – Шаг 3

Для замены старого состояния ячейки  $C_{t-1}$  на новое состояние  $C_t$ . Необходимо умножить старое состояние на  $f_t$ , забывая то, что решили забыть ранее. Затем прибавляем  $i_t * \tilde{C}_t$ . Это новые значения-кандидаты, умноженные на  $i_t$  – на сколько обновить каждое из значений состояния.

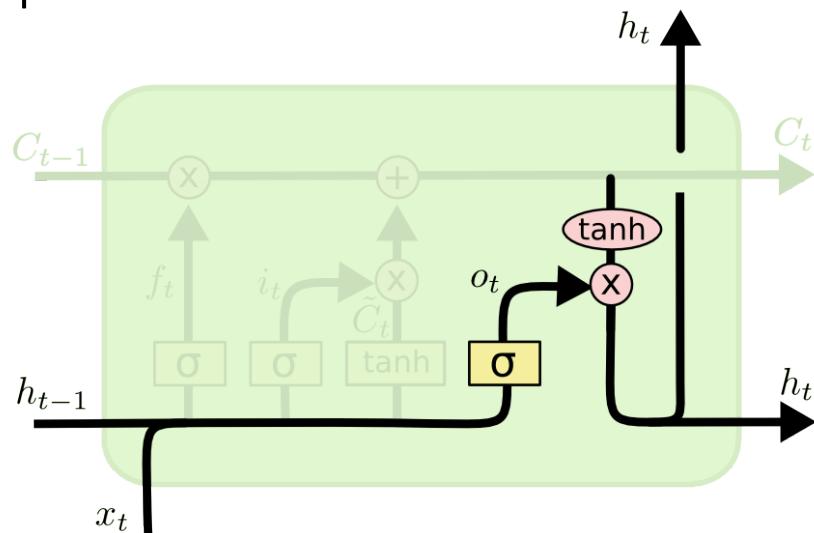


$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

# Принцип работы LSTM-модуля – Шаг 4

На последнем этапе определяется то, какая информация будет получена на выходе. Выходные данные будут основаны на нашем состоянии ячейки, к ним будут применены некоторые фильтры. Сначала значения предыдущего выхода  $h_{t-1}$  и текущего входа  $x_t$  пропускаются через сигмоидальный слой, который решает, какая информация из состояния ячейки будет выведена. Затем значения состояния ячейки проходят через  $\tanh$ -слой, чтобы получить на выходе значения из диапазона от -1 до 1, и перемножаются с выходными значениями сигмоидального слоя, что позволяет выводить только требуемую информацию.

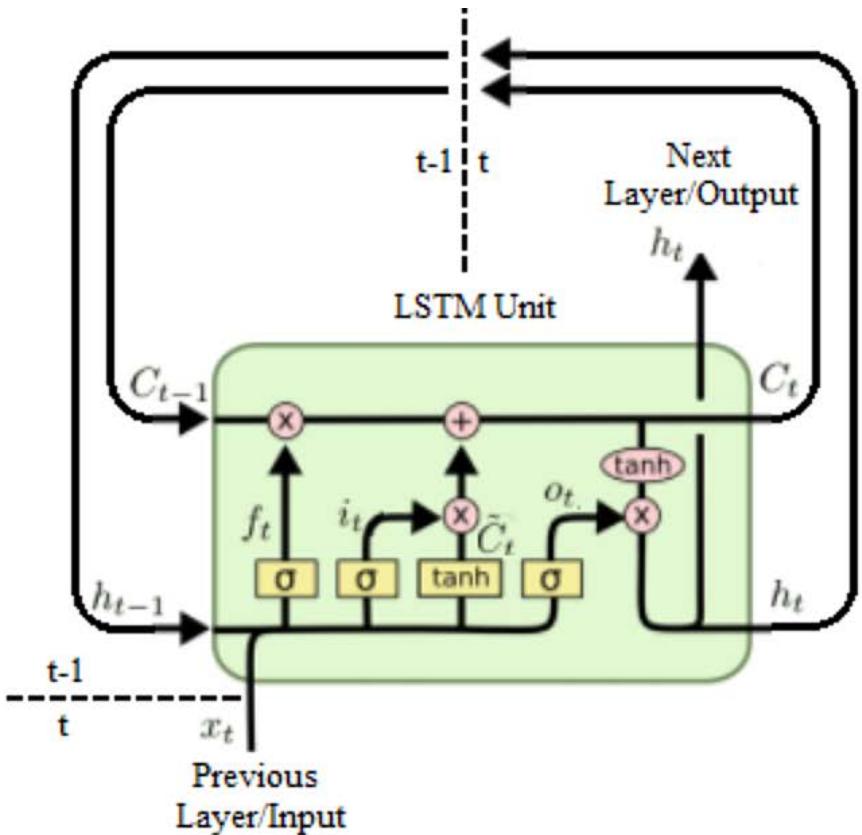
Полученные таким образом  $h_t$  и  $C_t$  передаются далее по цепочке.



$$o_t = \sigma (W_o [ h_{t-1}, x_t ] + b_o)$$

$$h_t = o_t * \tanh (C_t)$$

# Работа LSTM-модуля



$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f)$$

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$$

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C)$$

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o)$$

$$h_t = o_t * \tanh(C_t)$$

# Области и примеры применения

Используются, когда важно соблюдать последовательность, когда важен порядок поступающих объектов.

- Обработка текста на естественном языке:
  - Анализ текста;
  - Автоматический перевод;
- Обработка аудио:
  - Автоматическое распознавание речи;
- Обработка видео:
  - Прогнозирование следующего кадра на основе предыдущих;
  - Распознавание эмоций;
- Обработка изображений:
  - Прогнозирование следующего пикселя на основе окружения;
  - Генерация описания изображений.

**Спасибо за внимание**

# **Методы машинного обучения**

*Лекция 15*

**Глубокие НС**

**Интерпретация работы НС**

# Глубокое обучение (Deep learning)

**Глубокое обучение** — это совокупность методов машинного обучения, основанных на обучении представлениям (*feature/representation learning*), а не специализированных алгоритмах под конкретные задачи, и которые:

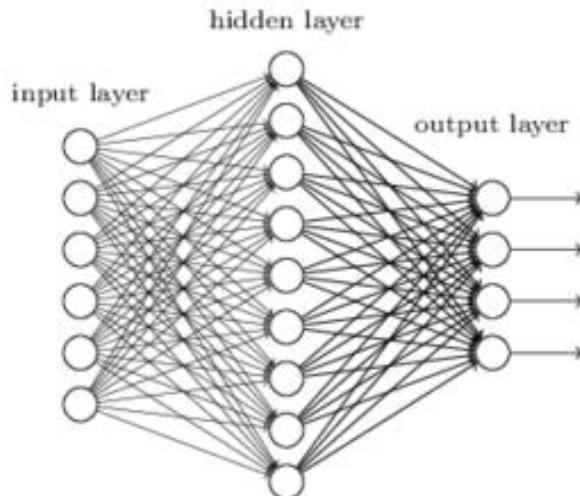
- Используют многослойную систему нелинейных фильтров для извлечения признаков с преобразованиями. Каждый последующий слой получает на входе выходные данные предыдущего слоя;
- Могут сочетать алгоритмы обучения с учителем, с частичным привлечением учителя, без учителя, с подкреплением;
- Формируют в процессе обучения слои выявления признаков (*features*) на нескольких уровнях представлений, которые соответствуют различным уровням абстракции; при этом признаки организованы иерархически — признаки более высокого уровня являются производными от признаков более низкого уровня.

Как правило, глубокое обучение предназначено для работы с большими объемами данных и использует сложные алгоритмы для обучения модели.

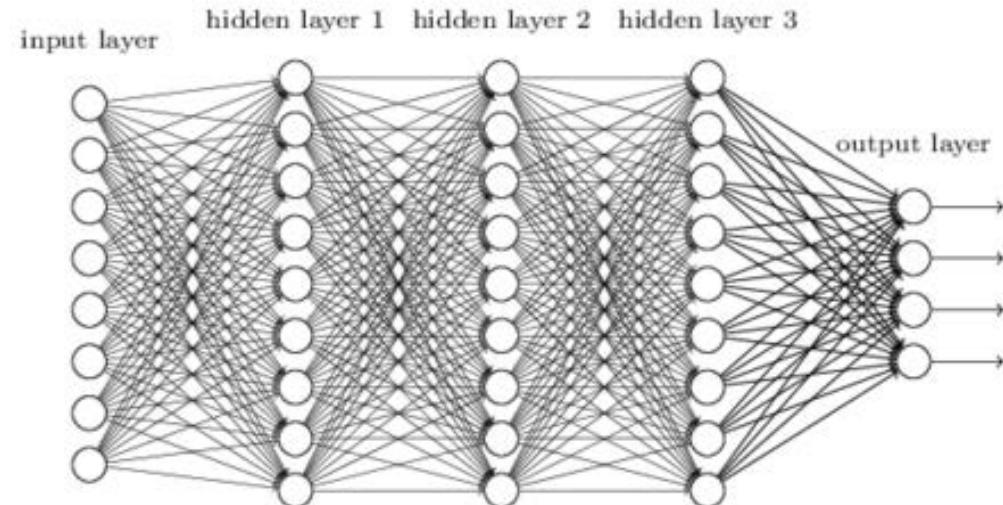
# Глубокие нейронные сети Deep Neural Network

Нейронные сети с числом скрытых слоев большим единицы называются глубокими. Они могут содержать меньшее число нейронов в каждом слое, чем сети с одним скрытым слоем, реализующие то же самое отображение, однако строгой методики сопоставления таких сетей не существует.

"Non-deep" feedforward neural network

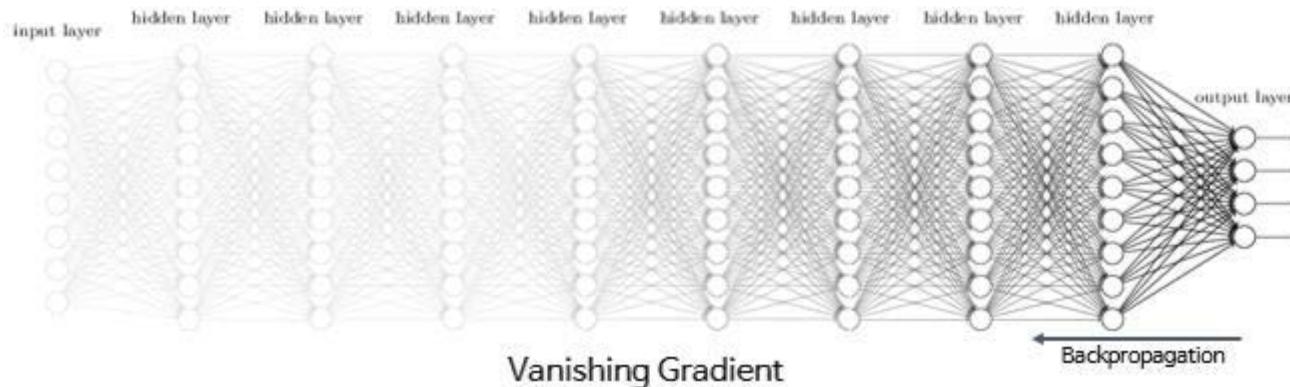
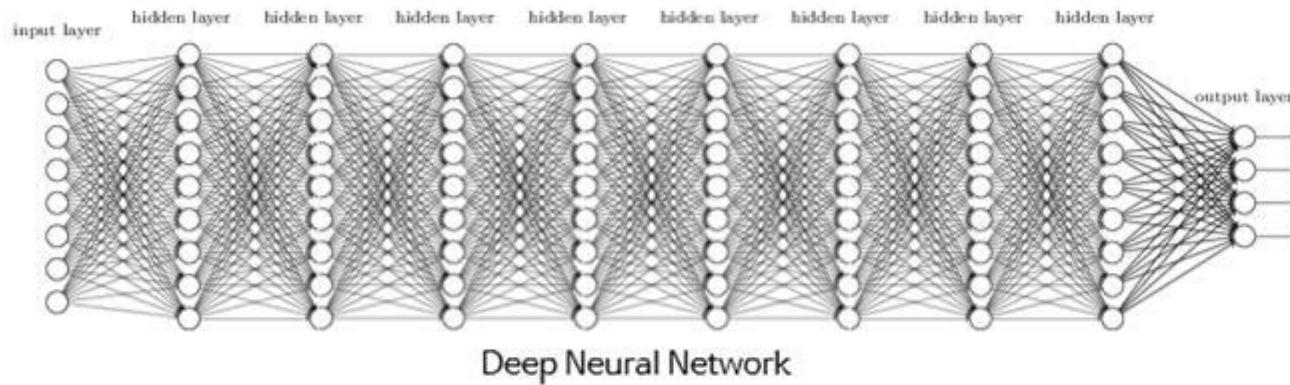


Deep neural network



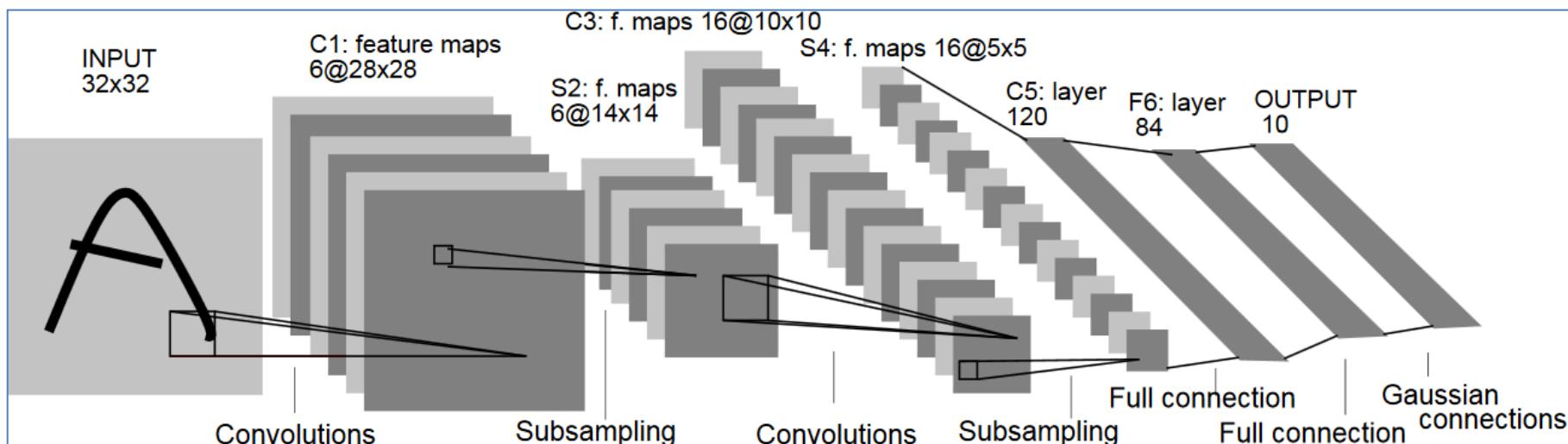
# Основные проблемы глубоких полносвязных нейронных сетей

- большое число настраиваемых параметров;
- затухающими градиентами при обучении с помощью алгоритма обратного распространения ошибки (Backpropagation);
- полностью игнорируется топология локального взаиморасположения входных величин.



# Свёрточная нейронная сеть

- Неокогнитрон (*Neocognitron*) — это иерархическая многослойная нейронная сеть сверточного типа, предложена Кунихику Фукусимой в 1980 году, способная к робастному распознаванию образов, обучаемая по принципу «обучение без учителя».
- В 1998 году Яну Лекуну удалось использовать алгоритм обратного распространения ошибки для обучения глубокой сверточной нейронной сети для решения задачи распознавания рукописных ZIP-кодов.

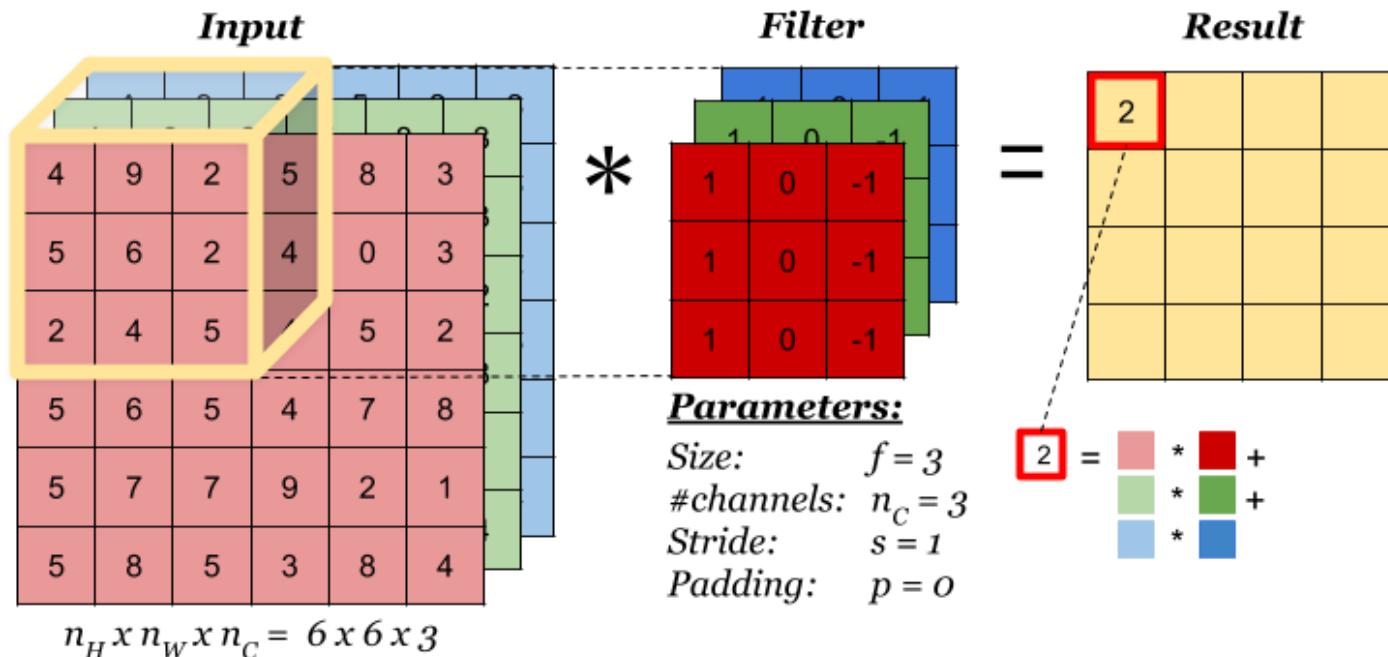


Архитектура сверточной нейронной сети LeNet-5, использованной для распознавания цифр.

# Операция свертки (Convolution)

Операция свертки (Convolution) аналогична использованию малого фильтра, например, размером  $3 \times 3$  с шагом 1 ко всему изображению. Применение одного такого фильтра фактически является построением некой карты нахождения определенного признака на изображении. Каждый фильтр соответствует одному нейрону.

**Свертка** - операция над парой матриц  $A$  (размера  $n_x \times n_y$ ) и  $B$  (размера  $m_x \times m_y$ ), результатом которой является матрица  $C = A * B$  размера  $(n_x - m_x + 1) \times (n_y - m_y + 1)$ . Каждый элемент результата вычисляется как скалярное произведение матрицы  $B$  и некоторой подматрицы  $A$  такого же размера (подматрица определяется положением элемента в результате). То есть,  $C_{i,j} = \sum_{u=0}^{m_x-1} \sum_{v=0}^{m_y-1} A_{i+u,j+v} B_{u,v}$ .

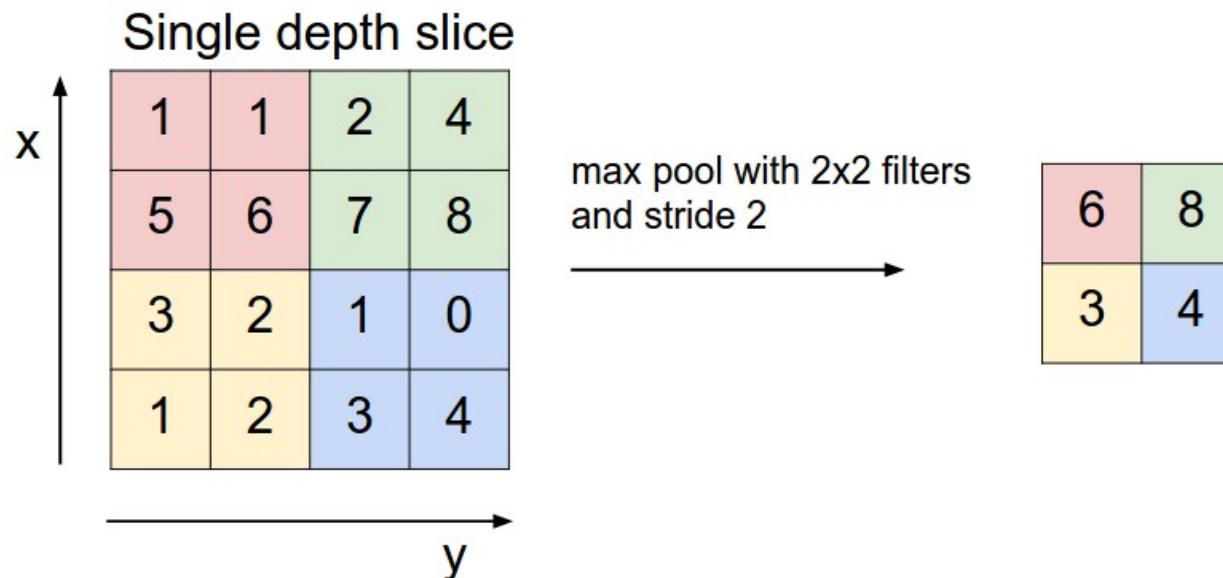


# Пулинг / сабсемплинг (Subsampling)

Пулинг (или сабсемплинг) используется для уменьшения размерности. Он основан на том факте, что изображения обладают свойством локальной скоррелированности пикселей, т.е. соседние пиксели, как правило, не сильно отличаются друг от друга. Таким образом, если из нескольких соседних пикселей получить какой-либо агрегат, то потери информации будут незначительными. Рекомендуемый размер объединения составляет  $2 \times 2$ .

Основные цели пулингового слоя:

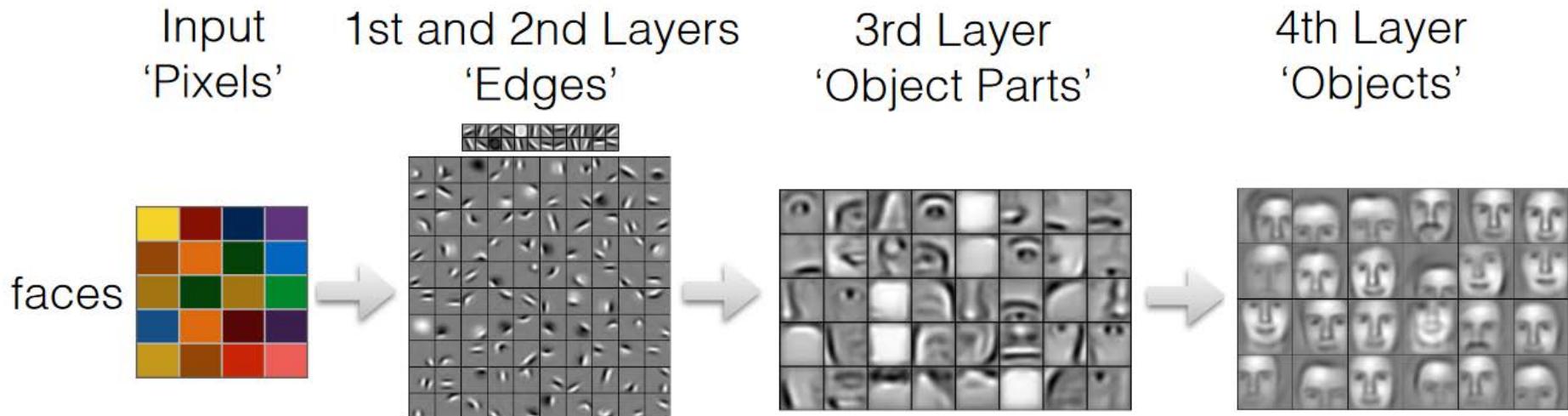
- уменьшение изображения, чтобы последующие свертки оперировали над большей областью исходного изображения;
- увеличение инвариантности выхода сети по отношению к малому переносу входа;
- ускорение вычислений.



# Общий принцип функционирования

На вход сети подается изображение и к нему последовательно применяются операции свертки (Convolution), которые чередуются с пулингом (Subsampling) несколько раз, а затем полученные данные проходят через набор полносвязных слоев.

Тем самым сверточные нейронные сети, позволяют создавать модели, состоящие из множества слоев, которые способны обучаться представлениям данных с различными уровнями абстракции.



# Ключевые этапы реализации глубоких сетей

Можно выделить восемь ключевых этапов для реализации глубоких сетей:

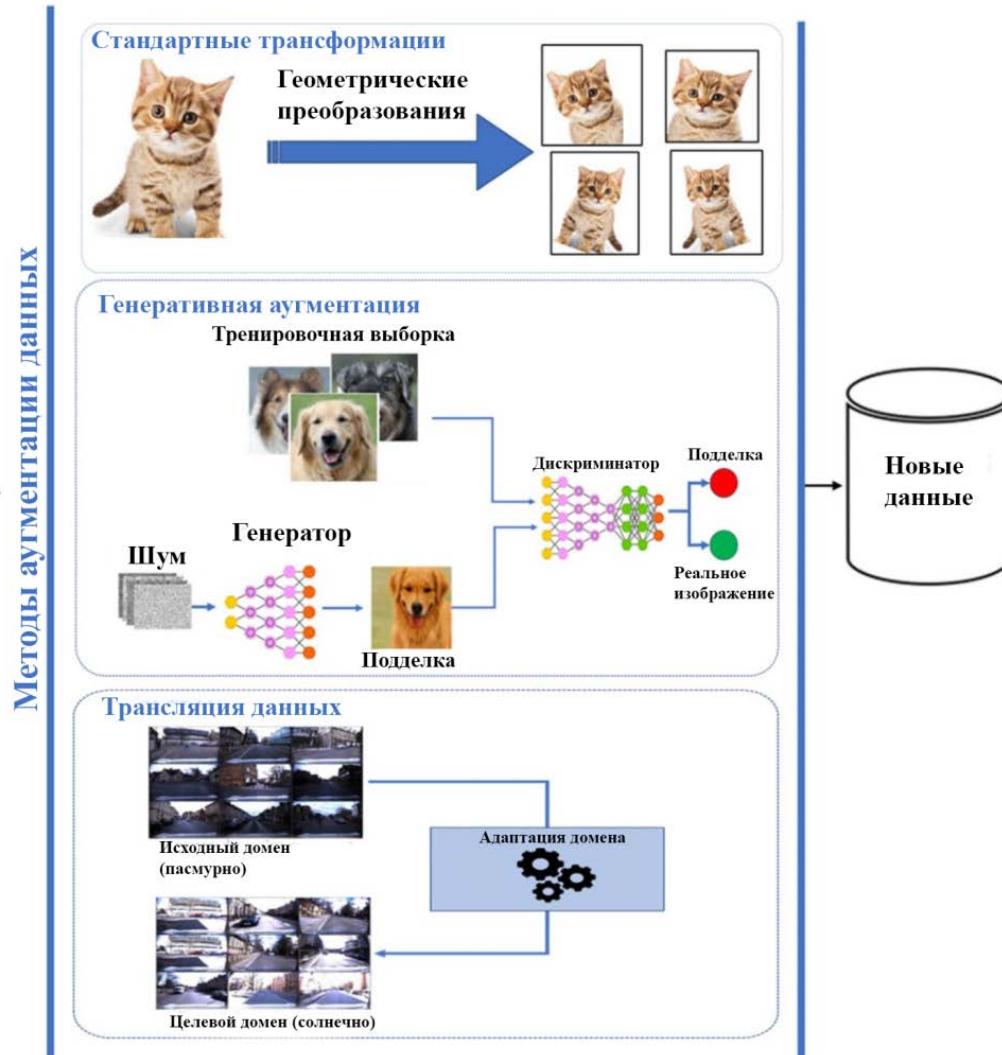
- Аугментация данных.
- Предобработка данных.
- Инициализация.
- Выбор функций активации.
- Процесс обучения.
- Регуляризация.
- Визуализация.
- Ансамбли глубоких сетей.

# Аугментация данных

Аугментация данных – это методика создания дополнительных обучающих данных из уже имеющихся.

Самыми популярными (классическими) методами аугментации являются:

- отражение по горизонтали (horizontal flip);
- случайное кадрирование (random crop);
- изменение цвета (color jitter).



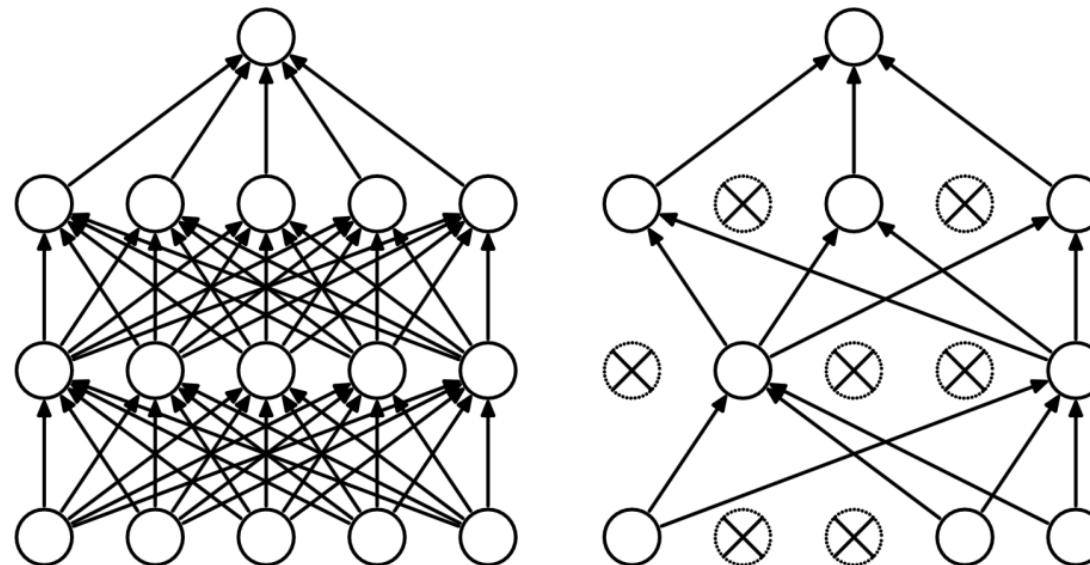
Возможно применение различных комбинаций, например, одновременно выполнять поворот и случайное масштабирование.

# Регуляризация

Регуляризация позволяет осуществлять контроль емкости нейронной сети, что способствует предотвращению переобучения. Основная идея заключается в учете дополнительной информации, которая имеет вид штрафа за сложность модели. Также регуляризация позволяет ограничивать значения весовых коэффициентов и осуществлять отбор наиболее важных факторов, которые сильнее всего влияют на результат.

Основные методы регуляризации применительно к нейронным сетям: L1 и L2-регуляризация, ограничения нормы вектора весов, дропаут.

Dropout регуляризация нейронной сети:



# Извлечение правил из полносвязной нейронной сети в задачах классификации

Под извлекаемой логической закономерностью будем понимать легко интерпретируемое правило, выделяющее из обучающей выборки достаточно много объектов какого-то одного класса и практически не выделяющее объекты остальных классов. Правила, выражающие закономерности, формулируются на языке логических предикатов первого порядка вида:

**ЕСЛИ** (условие $1$ ) И (условие $2$ ) И ... И (условие $N$ ) **ТО** (вывод).

Можно выделить следующие основные подходы:

- извлечение локальных правил из совокупности простейших однослойных сетей, на которые разделяется построенная многослойная модель;
- построение глобальных правил, которые характеризуют классы на выходе непосредственно через значения входных параметров.

# Извлечение локальных правил из обученных нейронных сетей

Основные этапы метода NeuroRule:

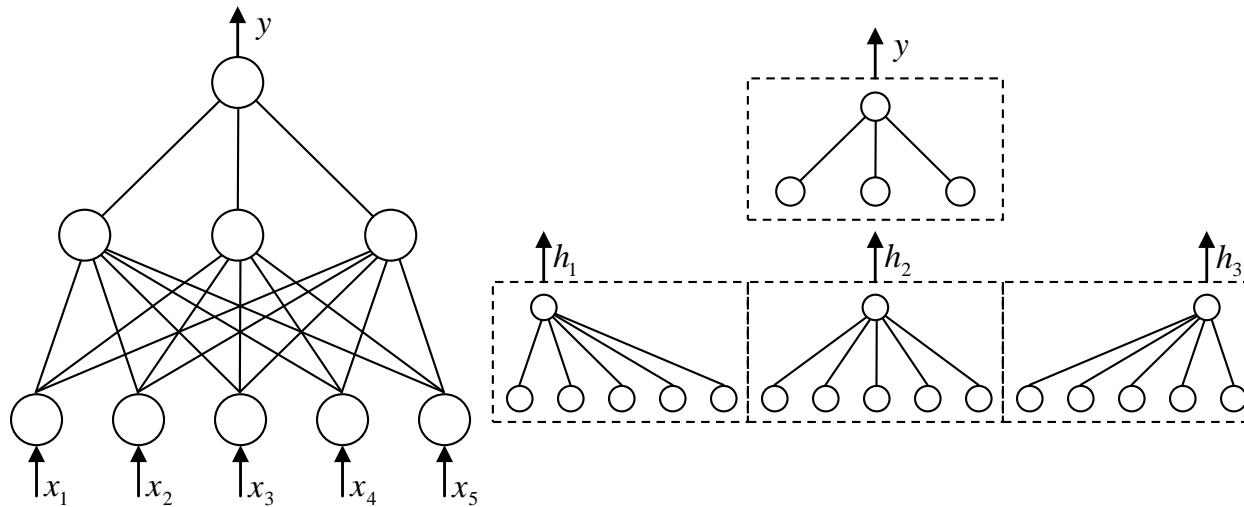
Этап 1. Обучение нейронной сети.

Этап 2. Прореживание нейронной сети.

Этап 3. Подготовка к извлечению правил,

кодирование признаков классифицируемых объектов.

Этап 4. Извлечение правил.

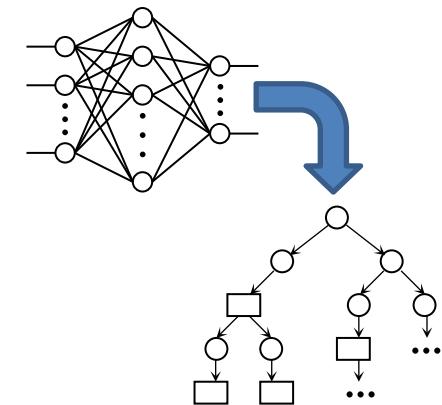


Основным недостатком является наличие жестких ограничений на архитектуру нейросети, число элементов, связей и вид функций активации, т.е. отсутствие универсальности и масштабируемости.

# Извлечение глобальных правил из обученных нейронных сетей

Построение глобальных правил, которые характеризуют классы на выходе непосредственно через значения входных параметров.

Данный подход осуществляет построение дерева решений на основе знаний, заложенных в обученную нейросеть, причем достаточно того, что сеть является неким «черным ящиком» или «Оракулом/Экспертом», которому можно задавать вопросы и получать от него ответы.

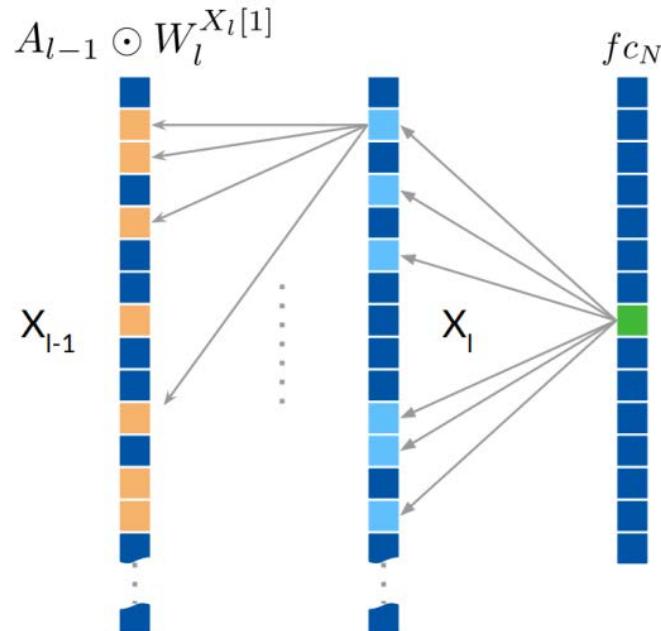


- Достоинство заключается в обобщающей способности искусственных нейронных сетей, что позволяет получать более простые деревья решений. Использование такого «Эксперта» позволяет компенсировать недостаток данных, наблюдающийся при построении деревьев решений на нижних уровнях.
- Недостаток заключается в отсутствии прозрачности внутреннего функционирования сети.

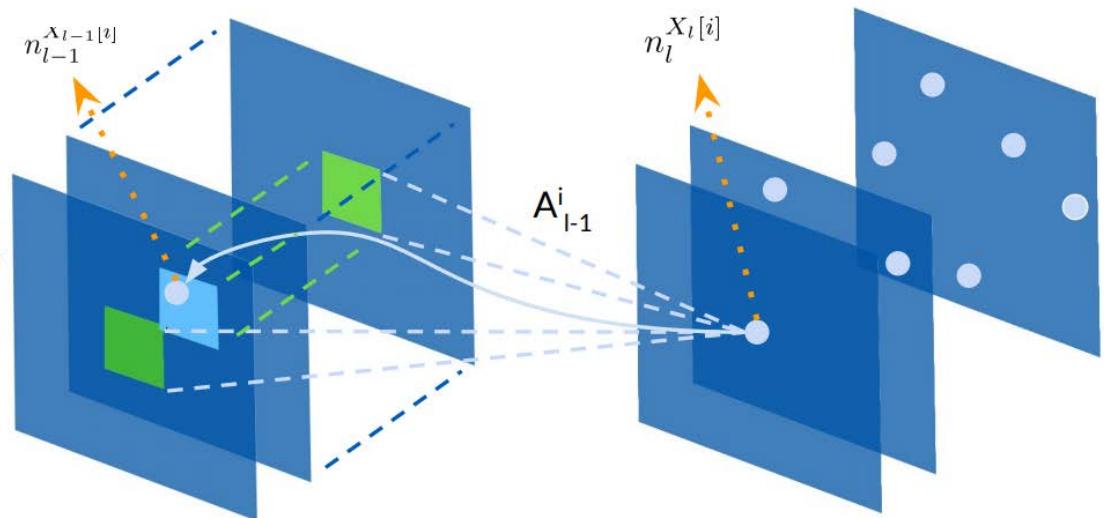
Дополнительным важным направлением является оценка чувствительности влияния значений входных параметров на выход сети. Для этого значения выбранного входа варьируются в области его определения, в то время как остальные параметры остаются фиксированными и отслеживаются изменения в выходе сети. Знания, полученные из этой формы анализа, могут быть представлены в виде таких правил, как:

«ЕСЛИ X уменьшается на 5%, ТО Y увеличивается на 8%».

# Интерпретация работы нейронных сетей - Обратное прохождение сигнала

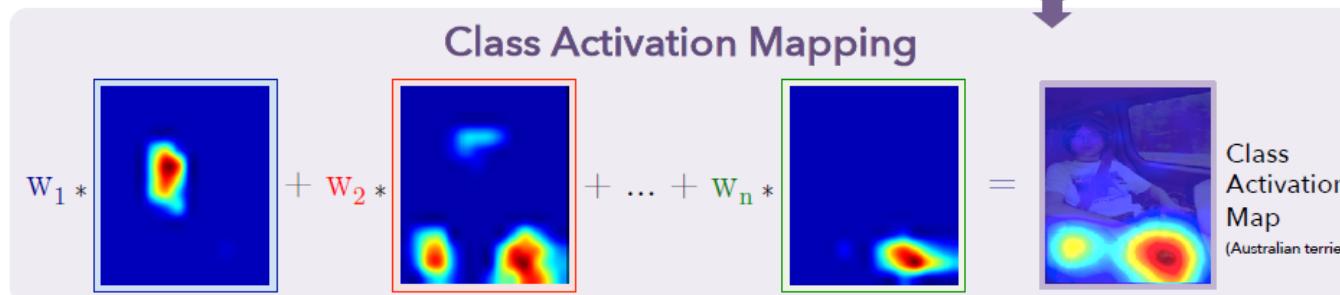
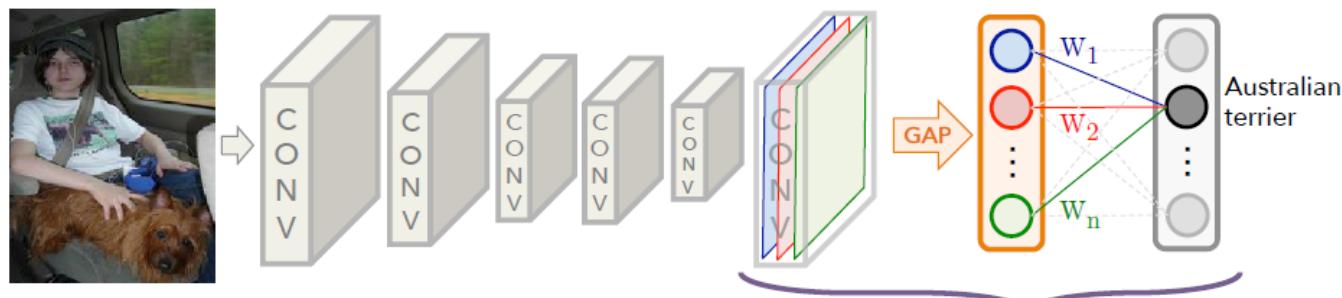
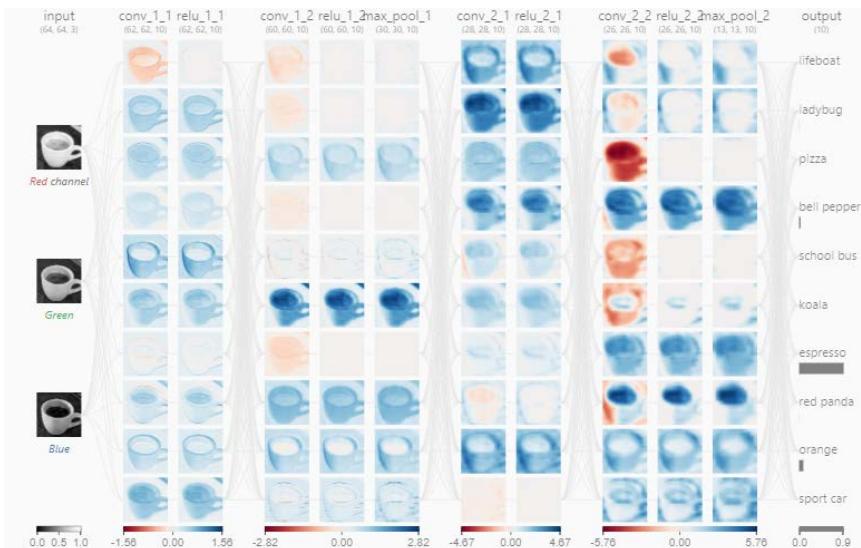


- Обратное прохождение  
полносвязанных слоев.
- Обратное прохождение  
сверточных слоев.



# Обобщение и интерпретация функционирования сверточных слоев

Одним из подходов, для обобщения и интерпретации результатов функционирования сверточных слоев является метод Class Activation Mapping (CAM), который представляет собой взвешенную карту активации, созданную для каждого изображения, что помогает определить область, на которую акцентирует внимание НС, при классификации изображения.

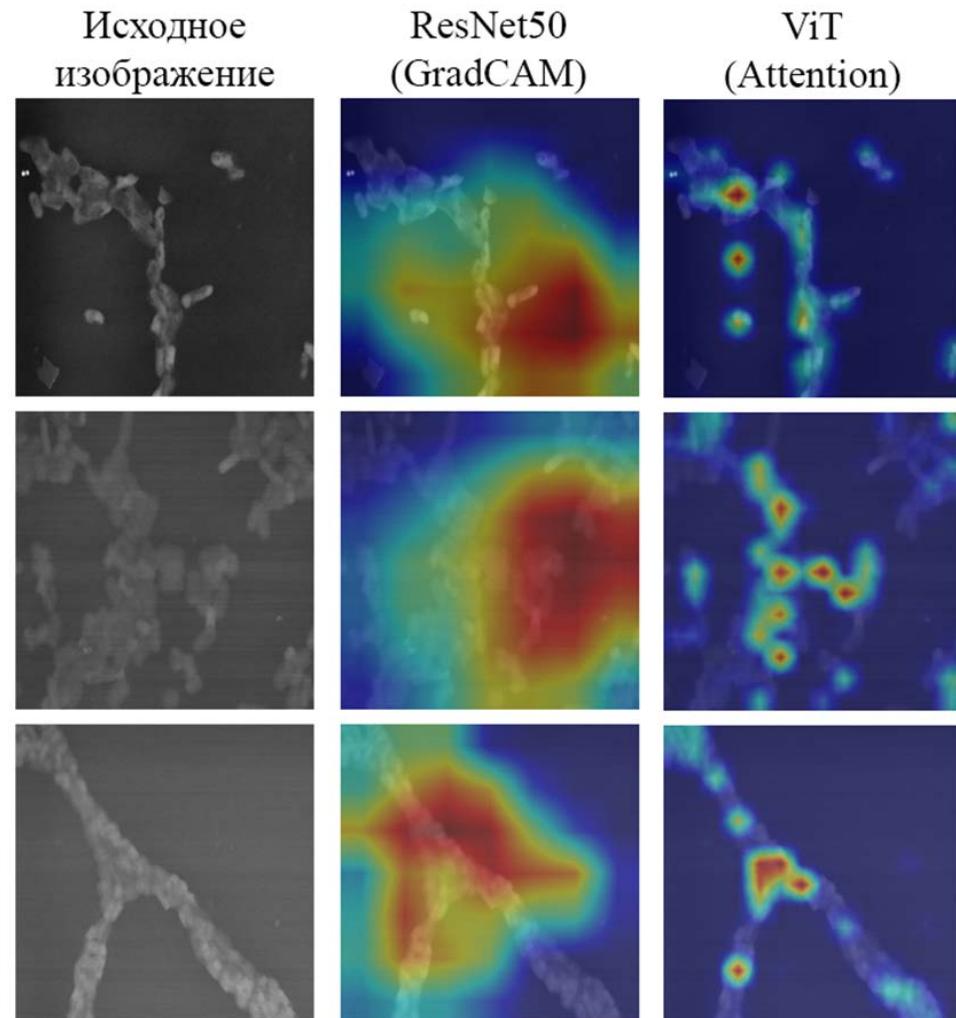


# Методы интерпретации

## GradCAM и Vision Transformer

Интерпретируемость двух популярных архитектур глубокого обучения — ResNet50 и Vision Transformer (ViT-224) — при задаче классификации микроскопических изображений бактерий. Представлена работа встроенных карт-внимания Vision Transformer и пост-интерпретация с помощью Grad-CAM для ResNet50. Показаны тепловые карты с выделенными зонами с наибольшим влиянием на прогноз модели.

Механизм внимания демонстрирует более сконцентрированные участки, однако в обоих способах интерпретации работы моделей наблюдаются фоновые зоны с высокой значимостью на прогноз модели. Это свидетельствует, что фоновый шум может иметь влияние на формирование прогноза, что в целом негативно влияет на способность генерализации моделей.



# «Доверенный ИИ»

Объединяет программные инструменты и методики для противодействия принципиально новым угрозам, возникающим на всех этапах жизненного цикла соответствующих технологий:

- Доверенные фреймворки и библиотеки машинного обучения.
- Инструменты проверки наличия аномалий в наборах данных.
- Инструменты оценки устойчивости обученных моделей к атакам.
- Инструменты для повышения доверия к предобученным моделям.
- Методы защиты моделей от атак на этапе эксплуатации.
- Методы объяснения моделей.
- Методы обнаружения дрейфа данных.
- Методы выявления предвзятости моделей.

ИСП РАН (<https://www.ispras.ru/ai-center/>)

# Угрозы ИИ

## Страхи перед ИИ

- ИИ принимает решения, которые невозможно объяснить или предсказать
- ИИ может ошибаться, а ответственность за эти ошибки размыта
- ИИ может быть предвзятым и манипулировать мнением
- ИИ собирает и использует персональные данные без явного согласия
- ИИ внедряется везде и его использование может выйти из-под контроля
- ИИ эволюционирует и изменяет алгоритмы без внешнего контроля

## Конкретные риски

- Потеря контроля над критическими инфраструктурами
- Финансовые потери из-за ошибок ИИ
- Утечки и неправомерное использование данных
- Репутационные риски из-за ошибок ИИ
- Юридическая ответственность за решения ИИ на пользователе не способном управлять ИИ
- Нарушение конфиденциальности и сбор данных без согласия
- Потеря контроля над личной информацией
- Манипуляция сознанием и поведенческая реклама
- Опасность автоматизированных решений без возможности апелляции

# Атаки на ИИ

С 2019 по 2024 год было опубликовано около 17000 научных статей, которые описывают состязательные атаки на ИИ.

По данным ресурса

<https://owasp.org/www-project-machine-learning-security-top-10/>

Top 10 Machine Learning Security Risks:

- ML01:2023 Input Manipulation Attack
- ML02:2023 Data Poisoning Attack
- ML03:2023 Model Inversion Attack
- ML04:2023 Membership Inference Attack
- ML05:2023 Model Theft
- ML06:2023 AI Supply Chain Attacks
- ML07:2023 Transfer Learning Attack
- ML08:2023 Model Skewing
- ML09:2023 Output Integrity Attack
- ML10:2023 Model Poisoning

- Инъекция (Prompt Injection)

Атака по смыслу схожа с sql-инъекцией. Но на текущий момент, это уже что-то вроде социальной инженерии для ИИ. В контексте атак на ИИ её относят к jailbreak.

- Инфекция (Infection)

Заражение вредоносным ПО. В OWASP один из примеров – это атака на цепочку поставок.

- Уклонение (Evasion)

Данные на вход для ИИ модифицируют незначительно. Нарушитель стремится заставить ИИ ошибиться.

- Отравление ИИ (Poisoning)

Отравление данных, которые поступают в ИИ. Доказано, что достаточно внести 0,001% ошибок в данные, чтобы результаты оказались аномальными и неверными.

- Извлечение (Extraction)

Допустим, кто-то иногда делает запросы в модель и медленно собирает из нее данные. В конце концов атакующий соберет достаточно информации, чтобы восстановить данные на своей стороне.

**Спасибо за внимание**