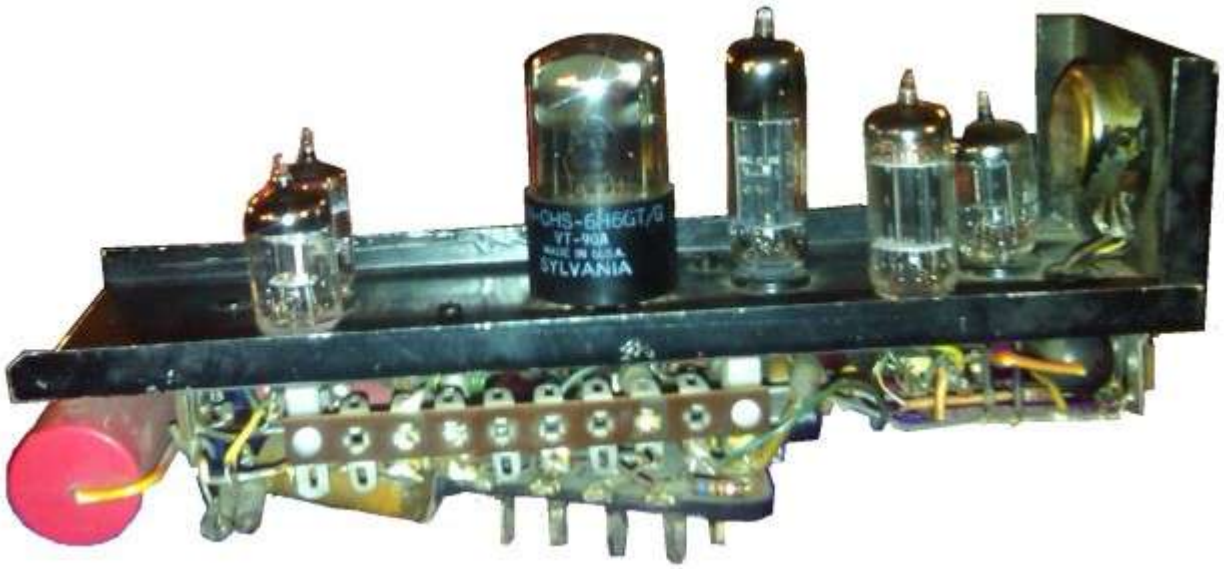
Содержание

- История
  - Кибернетика и первые нейронные сети
  - Первые цифровые компьютеры
  - Появление экспертных систем
  - Вероятностный подход
  - Случайный лес
- Последние достижения

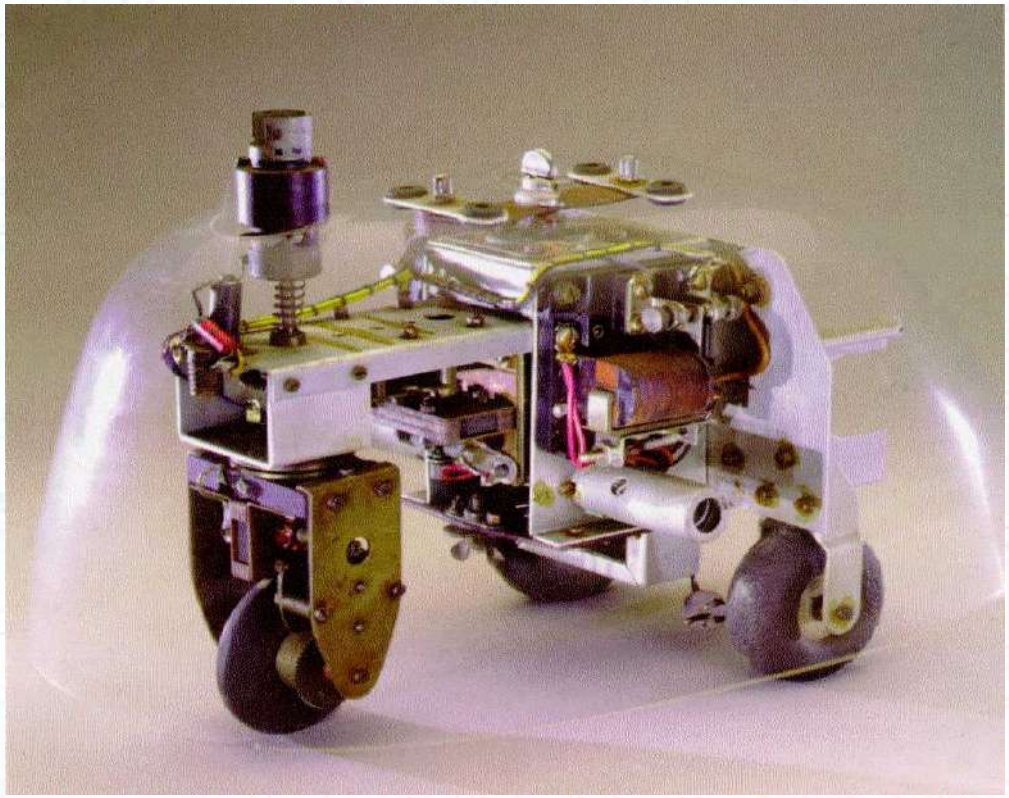
Кибернетика
и первые нейронные сети



SNARC - Marvin Lee Minsky

Кибернетика и первые нейронные сети

- Черепаха (конструктор Grey Walter, 1949)



Кибернетика и первые нейронные сети

- Цифровая электронная вычислительная машина Б.И. Рамеева и И.С. Брука
- Решала дифференциальные уравнения до шестого порядка
- Авторское свидетельство выдано 4 декабря 1948

ГОСУДАРСТВЕННЫЙ КОМИТЕТ СОВЕТА МИНИСТРОВ СССР
ПО ВНЕДРЕНИЮ ПЕРЕДОВОЙ ТЕХНИКИ В НАРОДНОЕ ХОЗЯЙСТВО
ПАТЕНТНЫЙ ОТДЕЛ

АВТОРСКОЕ СВИДЕТЕЛЬСТВО

№ 10475

Настоящее авторское свидетельство выдано
гр. РАМЕЕВУ Бакиру Искандеровичу
и БРУК И.С.

на изобретение, по заявке № 365968
с приоритетом 4 декабря 1948 года

Материалы к авторскому свидетельству хранятся в Гостехнике СССР.

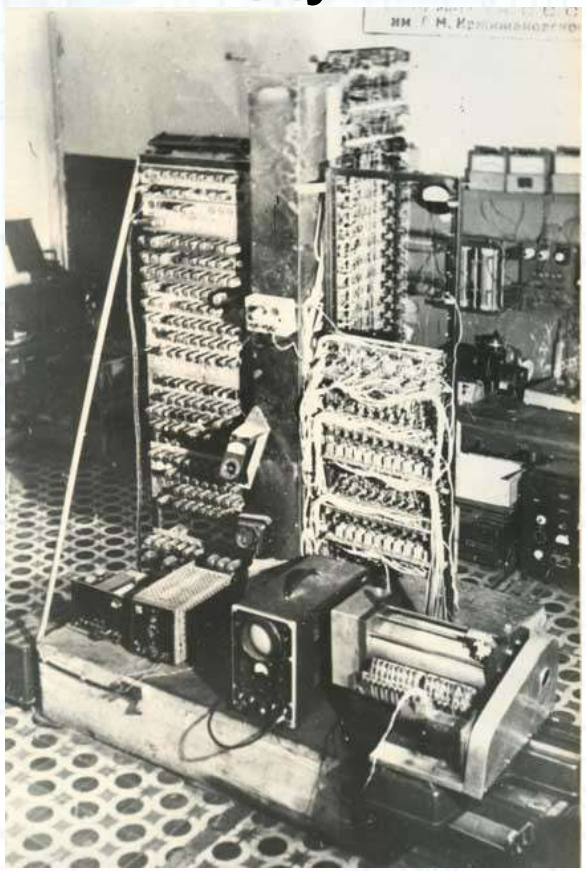
Действие авторского свидетельства распространяется на всю территорию Союза ССР.

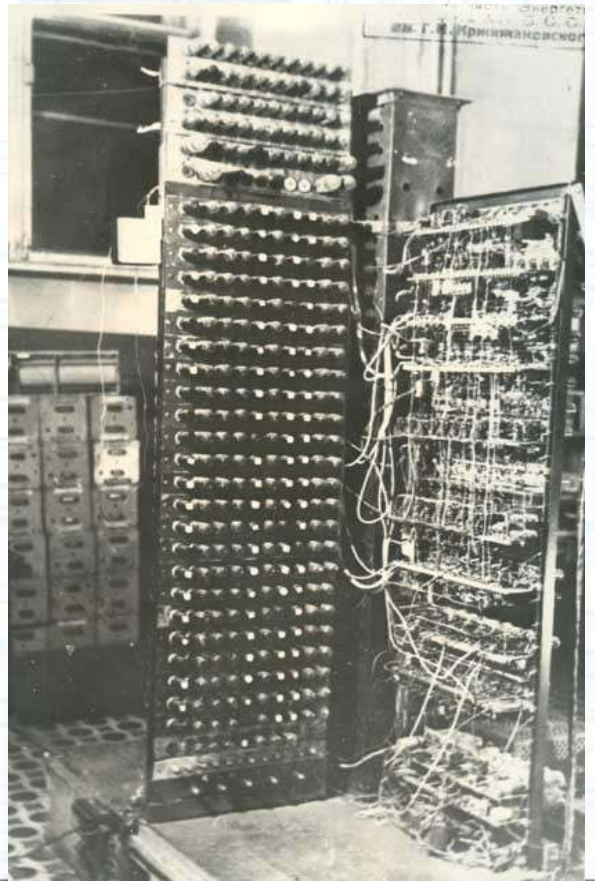
Внесено в Государственный реестр изобретений Союза ССР 16 февраля 1950 г.

Москва, 21 февраля 1950 г.

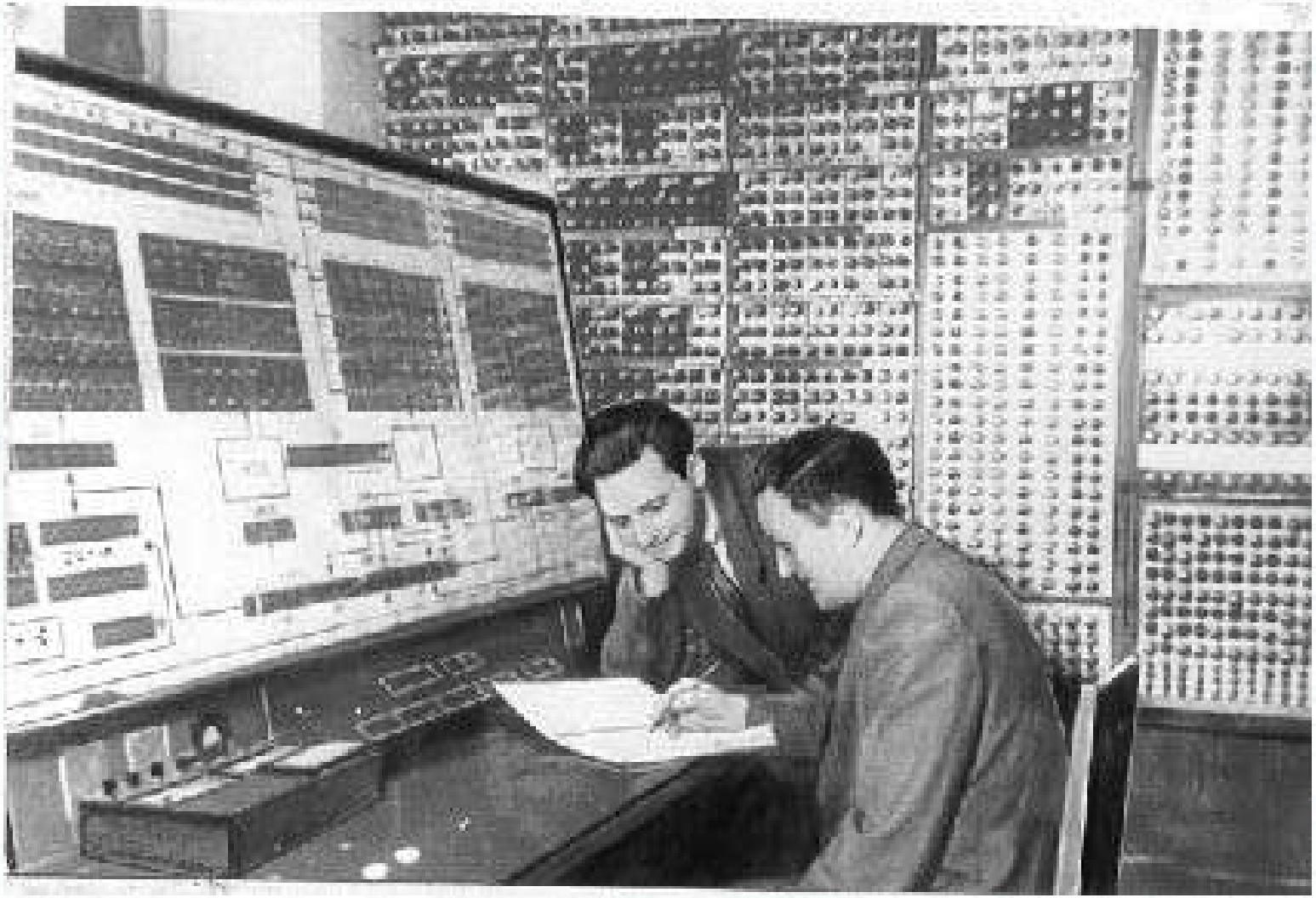
Первые цифровые компьютеры

- ЭВМ М1 - первая в мире машина на полупроводниковых приборах
- Построена под руководством И.С.Брука в 1951 году

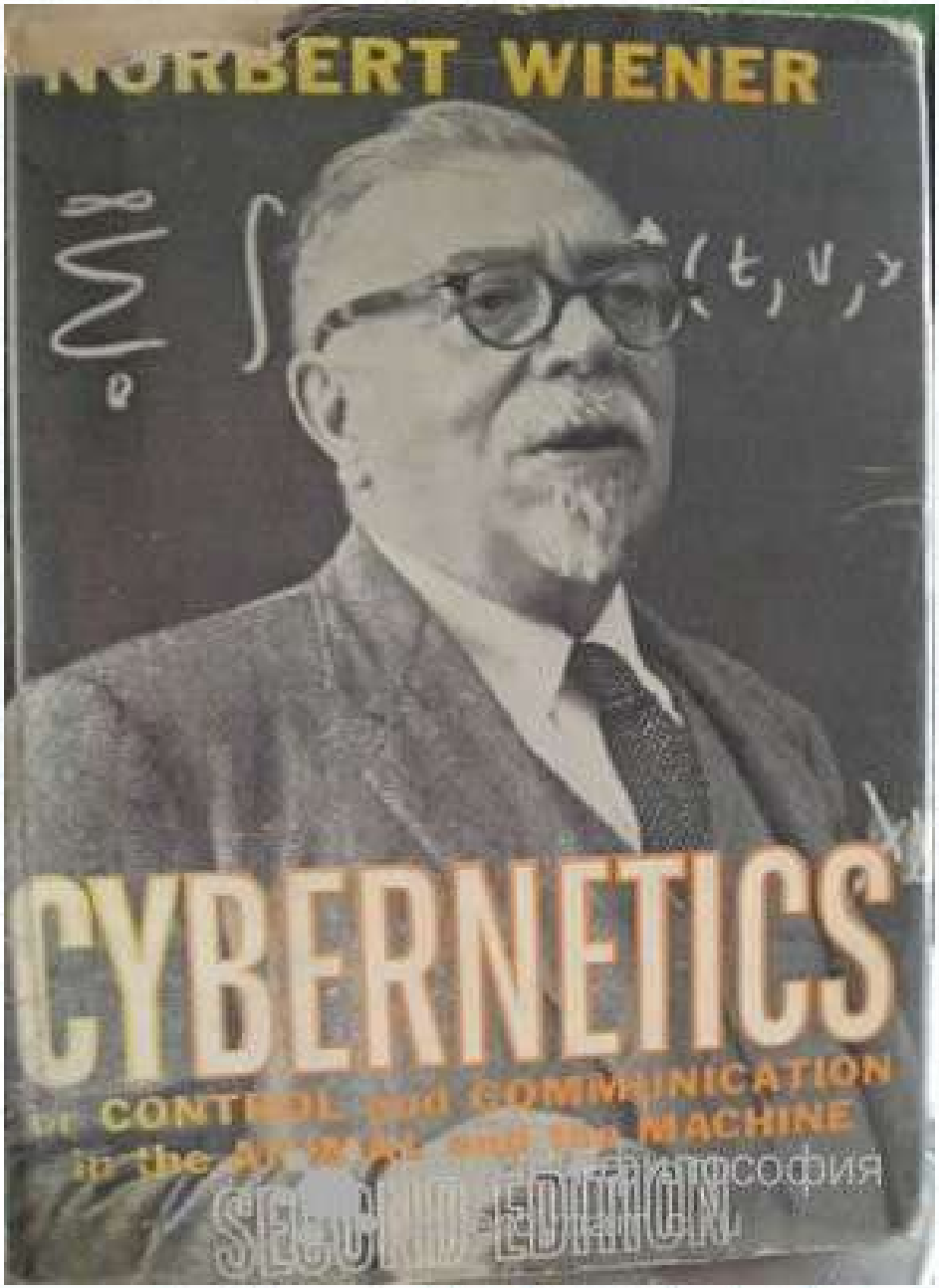




Первые цифровые компьютеры



1948г. “Кибернетика” Н.Винера

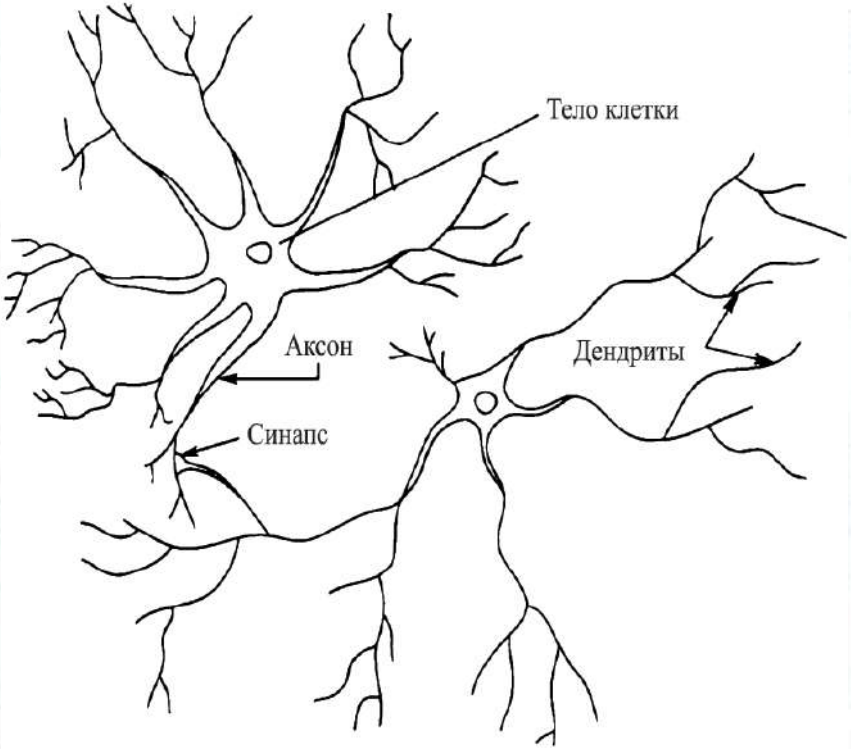


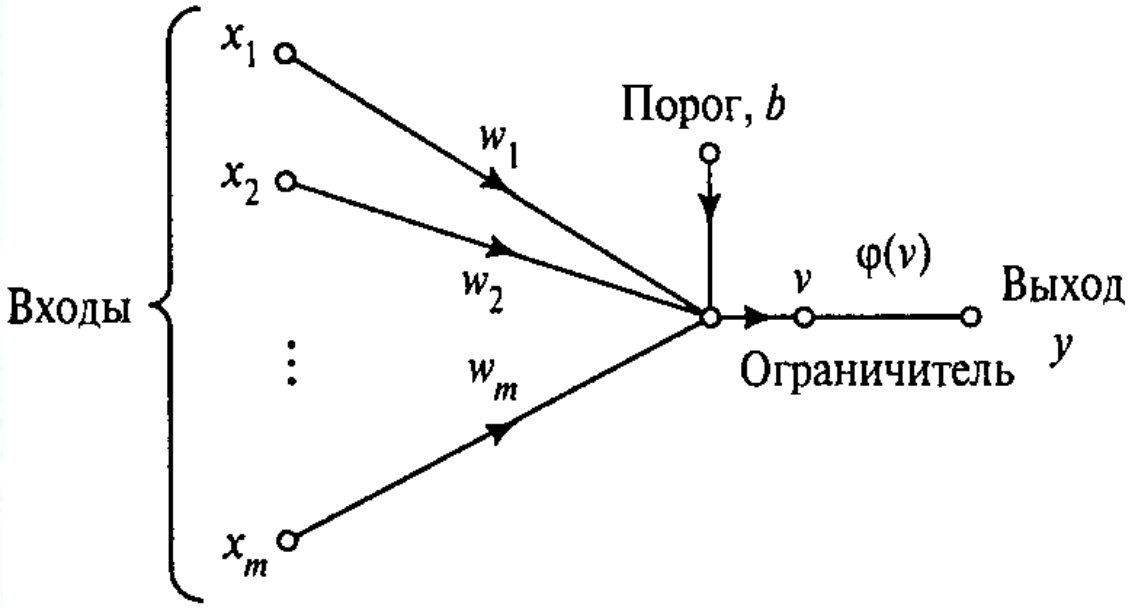
Алан Тьюринг

Может ли машина мыслить как человек?

Тест Тьюринга

Перцептрон Розенблатта





“Шашки” Артура Самуэля

1956г. Дартмутский семинар

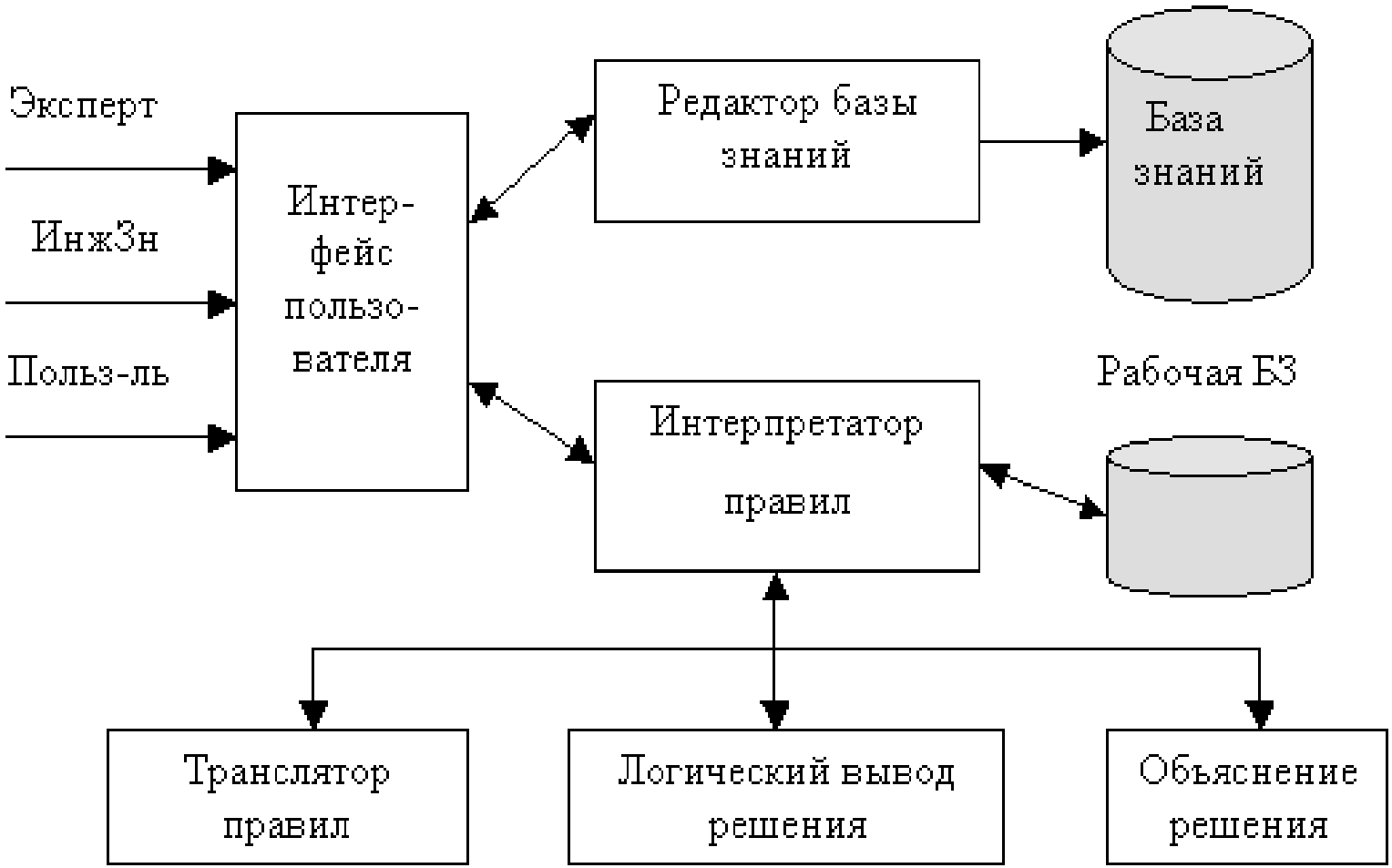
- Целью конференции было рассмотрение вопроса: можно ли моделировать рассуждения, интеллект и творческие процессы с помощью вычислительных машин
- Утвердила появление новой области науки и дала ей название — «Artificial Intelligence»

1974–1980 – зима для Искусственного интеллекта

- Наивный оптимизм:
  - 1954 через 3 — 5 лет задача машинного перевода будет полностью решена
  - 1958 через 10 лет компьютер станет чемпионом мира по шахматам; докажет важную математическую теорему
  - 1970 через 3 — 8 лет будет создан искусственный интеллект общего назначения, сравнимый с интеллектом среднестатистического человека
- парадокс Моравека

Первые экспертные системы: Dendral и MYCIN

Эдвард Фейгенбаум

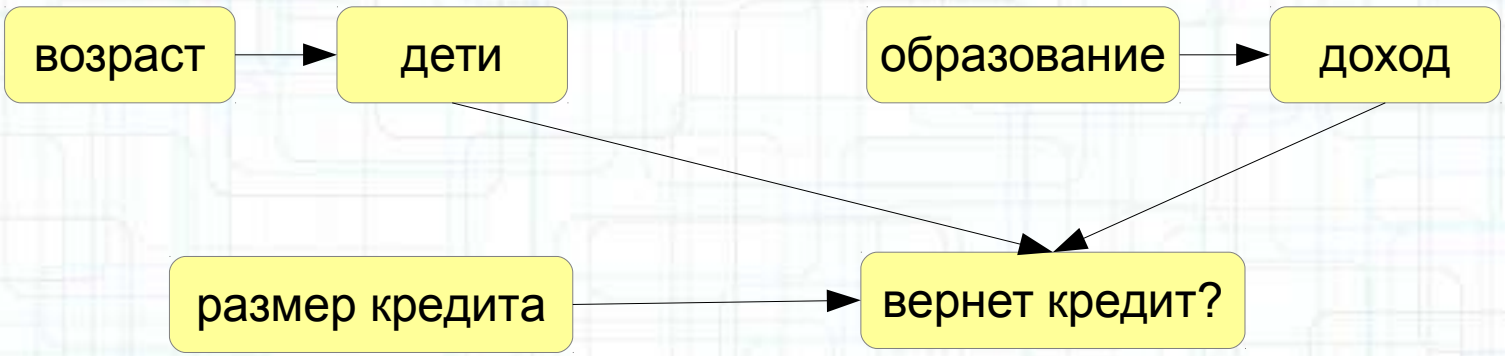


1980–1987 внедрение экспертных систем

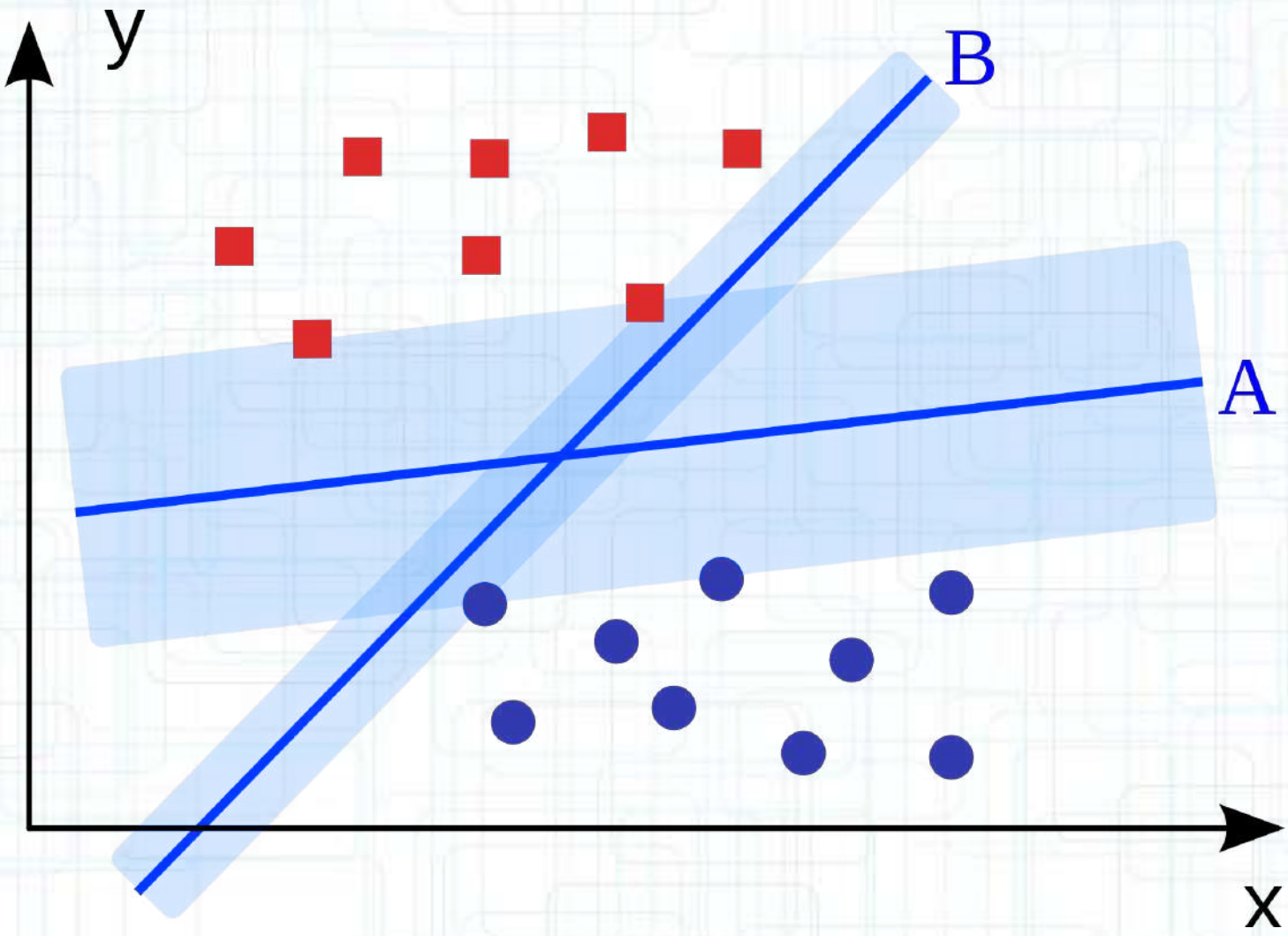
- XCON экономила фирме DEC 25 млн $ в год
- PROSPECTOR для геологической разведки месторождений полезных ископаемых
- Количество экспертных систем в СССР исчисляется сотнями (обзор в справочнике под. ред. Д.А.Поспелова)

Байесовские сети доверия

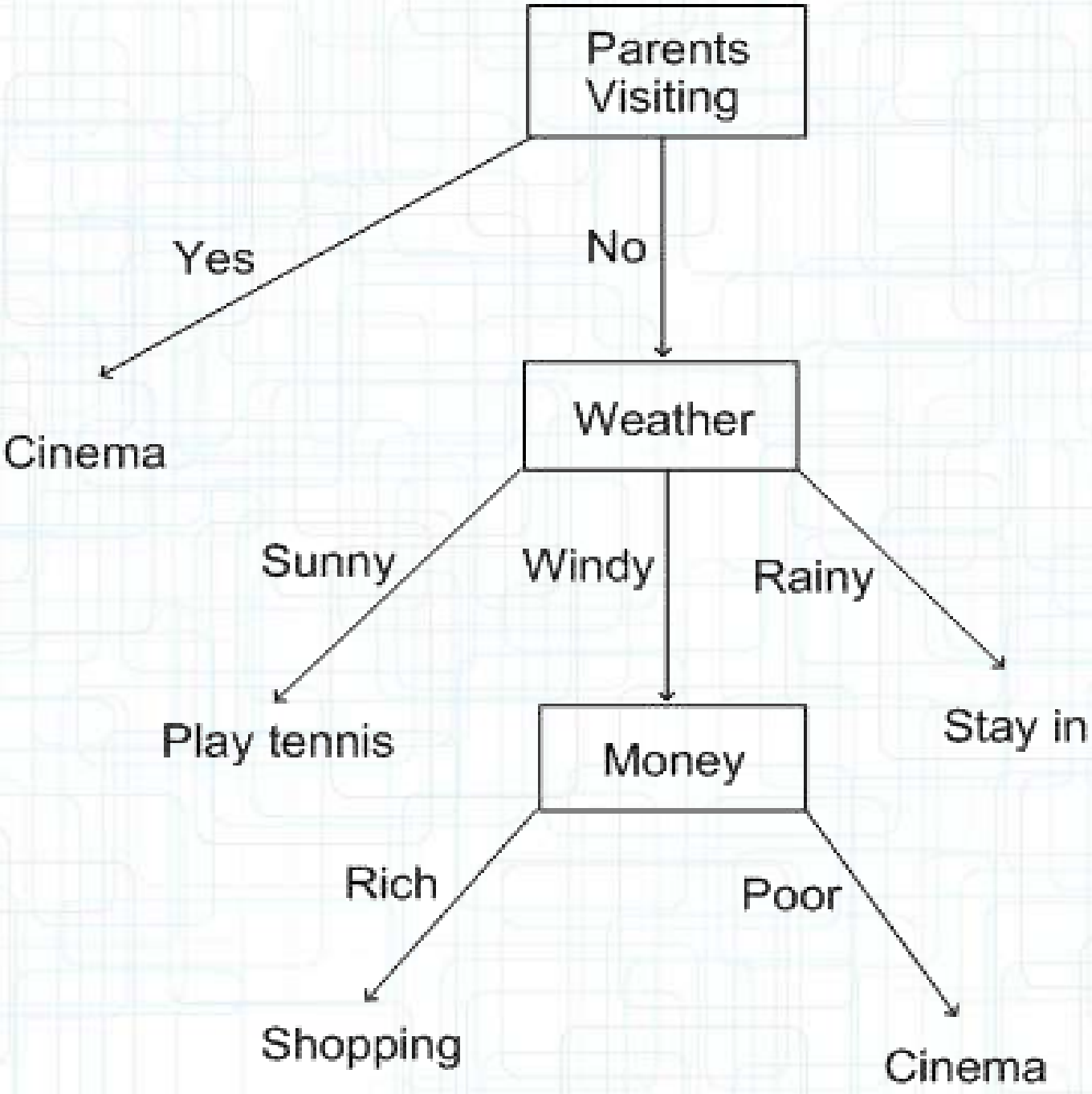
- Решение задач ИИ основывается не на знаниях экспертов, а на статистических данных
- Джуда Перл создает аппарат байесовских сетей доверия для задач классификации (1988 г.)



1995 К.Кортес и В.Вапник изобретают SVM



Деревья решений



Деревья решений

- Поиск наилучших закономерностей
- Группа 100 пациентов с подозрением на менингит (вероятность диагноза: 0.3)

	признаки			
	головная боль	температура	тошнота	симптом Кернига
% пациентов с данным признаком, болеющих менингитом	30	15	5	2
% пациентов с данным признаком, но без менингита	70	35	5	0

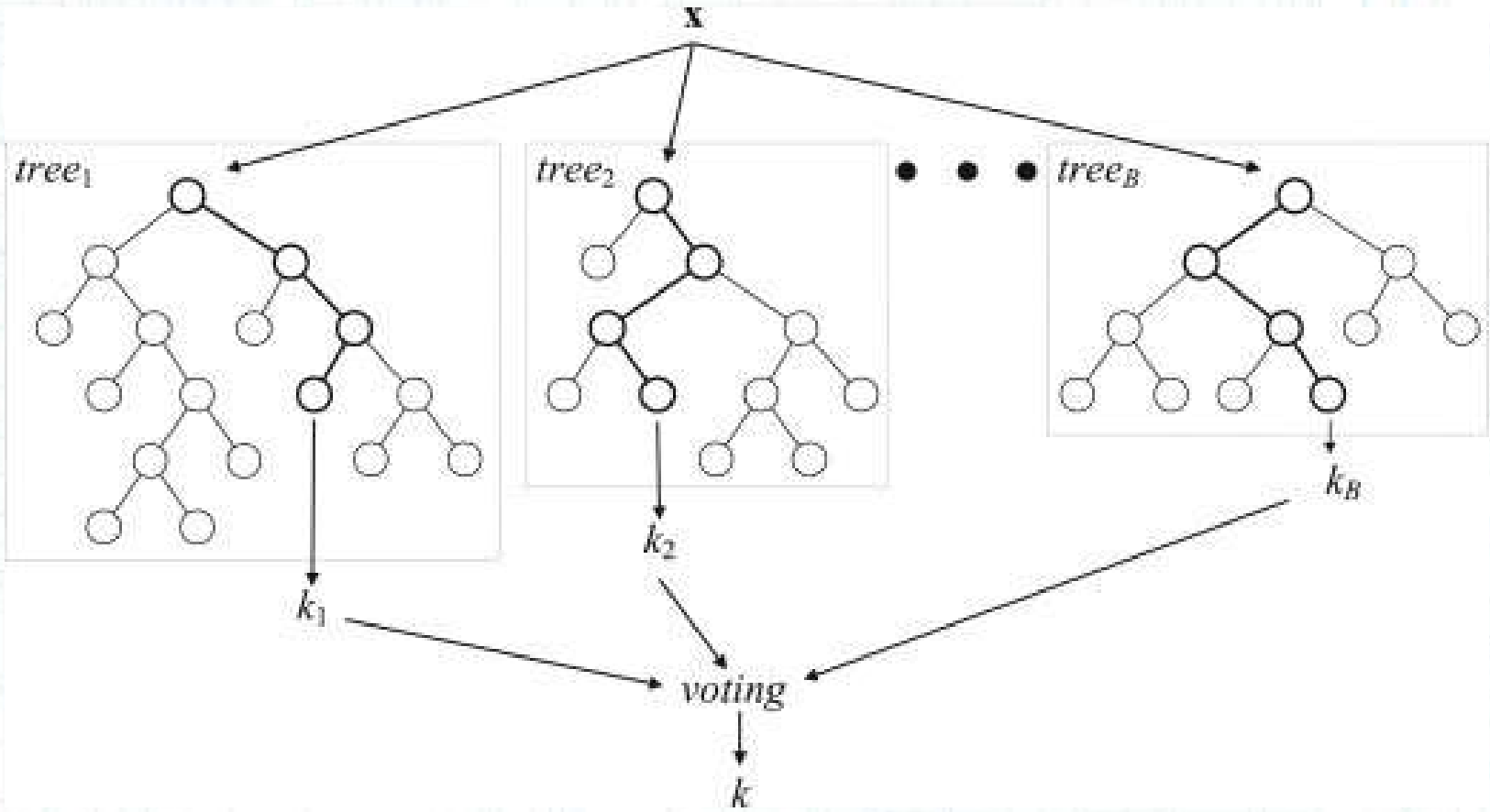
Деревья решений

- Поиск наилучших закономерностей
- Группа 100 пациентов с подозрением на менингит (вероятность диагноза: 0.3)

	признаки			
	головная боль	температура	тошнота	симптом Кернига
% пациентов с данным признаком, болеющих менингитом	30	15	5	2
% пациентов с данным признаком, но без менингита	70	35	5	0
вероятность такого распределения пациентов при условии независимости признака и диагноза	1	0.1725	0.0996	0.4879

Случайный лес

- Лео Брейман и Адель Катлер публикуют описание алгоритма, заключающегося в использовании ансамбля решающих деревьев (2002)



Развитие нейронных сетей

- 1957 – Розенблат создает простую модель перцептрона
- 1986 – метод обратного распространения ошибки
- 1998 – метод стохастического градиента
- 2000 – Игорь Айзенберг придумал термин «deep learning»
- 2009 – запуск БД изображений ImageNet
- 2011 – сверточная сеть AlexNet от Alex Krizhevsky побеждает во всех конкурсах
- 2014 – изобретение генеративных сетей

NLP – обработка естественного языка

- синтез речи
- распознавание речи
- распознавание текста (печатного и рукописного)
- машинный перевод
- информационный поиск
- ответы на вопросы

NLP

- 1988 — латентный семантический анализ (LSA)
- 2000 — латентное размещение Дирихле (LDA)
- 2000 - вероятностный латентно-семантический анализ (PLSA)
- 2013 — изобретение word2vec и начало широкого применения нейронных сетей в NLP
- 2014 — применение сетей-трансформеров для перевода
- 2016 — использование сетей на основе attention для перевода (частный случай сетей с памятью)
- 2018 — использование предобученных нейронных сетей

Компьютерные шахматы

- Deep Blue
- Deep Fritz
- Рейтинг CCRL
  - Stockfish
  - Komodo
  - Houdini
  - Fire
  - Rybka
  - Strelka

Watson


PIENSE
THINK
ΣΚΕΨΟΥ
DENKE
PENSER
$24,000
Who is Stoker?
(I FOR ONE WELCOME OUR NEW COMPUTER OVERLORDS)
$1,000
$77,147
Who is Bram Stoker?
$ 17,973
$21,600
WHO IS BRAM STOKER?
$5600

Информационный поиск

- поиск документов по запросу (разрешение лексической многозначности, морфологический анализ)
- классификация документов
- фильтрация документов
- аннотирование и реферирование документов

Яndex Google

Рекомендательные системы

- Netflix prise
- Коллаборативная фильтрация
- FastXML

Анализ тональности текста

- Тональность – позиция автора, относительно темы сообщения
- Сопряженная задача: определение темы
- Приложения:
  - политология
  - маркетинг (мнения людей о товарах, контекстная реклама)

Соционика

- построение модели личности
- методы использования показателей
- применения:
  - таргетинг рекламы
  - найм сотрудников
  - предвыборные кампании

Анализ временных рядов

За последние 10 дней

Дата	Курс	Изменение
19.10.18	**65,5825**	**-0,2375** ↓
18.10.18	65,8200	0,3225 ↑
17.10.18	65,4975	0,1900 ↑
16.10.18	65,3075	-0,3425 ↓
15.10.18	65,6500	-0,4675 ↓
12.10.18	66,1175	-0,1525 ↓
11.10.18	66,2700	-0,6300 ↓
10.10.18	66,9000	0,6500 ↑
09.10.18	66,2500	-0,3075 ↓
08.10.18	66,5575	-0,0400 ↓

Соревнования

- kaggle.com

	**Algorithmic Trading Challenge** Develop new models to accurately predict the market response to large trades.	$10,000	113	20 months ago
	**Don't Get Kicked!** Predict if a car purchased at auction is a lemon	$10,000	571	20 months ago
	**Give Me Some Credit** Improve on the state of the art in credit scoring by predicting the probability that somebody will experience financial distress in the next two years.	$5,000	926	21 months ago
	**Photo Quality Prediction** Given anonymized information on thousands of photo albums, predict whether a human evaluator would mark them as 'good'.	$5,000	206	22 months ago
	**Semi-Supervised Feature Learning** There's been a lot of recent work done in unsupervised feature learning for classification and there are a ton of older methods that also work well. The purpose of this competition is to find out which of these methods work best on relatively large-scale high dimensional learning tasks.	$500	28	23 months ago
	**Claim Prediction Challenge (Allstate)** A key part of insurance is charging each customer the appropriate price for the risk they represent. Risk varies widely from customer to customer, and a deep understanding of different risk factors helps predict the likelihood and cost of insurance claims. The goal of this competition is to better predict Bodily Injury Liability insurance claim payments based on the characteristics of the insured customer's vehicle.	$10,000	105	23 months ago
	**dunnhumby's Shopper Challenge** Going grocery shopping, we all have to do it, some even enjoy it, but can you predict it? dunnhumby is looking to build a model to better predict when supermarket shoppers will next visit the store and how much they will spend.	$10,000	279	23 months ago
	**Wikipedia's Participation Challenge** This competition challenges data-mining experts to build a predictive model that predicts the number of edits an editor will make five months from the end date of the training dataset.	$10,000	94	24 months ago
	**Mapping Dark Matter** Supported by NASA and the Royal Astronomical Society. A cosmological image analysis competition to measure the small distortion in galaxy images caused by dark matter. The prize is an expenses paid visit to the NASA Jet Propulsion Laboratory (JPL).	$3,000	72	2 years ago

Репозиторий UCI

- Ссылка

UCI

Machine Learning Repository

Center for Machine Learning and Intelligent Systems

About Citation Policy Donate a Data Set Contact

Search

Repository Web

Google™

View ALL Data Sets

Welcome to the UC Irvine Machine Learning Repository!

We currently maintain 481 data sets as a service to the machine learning community. You may **view all data sets** through our searchable interface. For a general overview of the Repository, please visit our About page. For information about citing data sets in publications, please read our citation policy. If you wish to donate a data set, please consult our donation policy. For any other questions, feel free to contact the Repository librarians.

Supported By: In Collaboration With: Rexa.info

Latest News:

09-24-2018: Welcome to the new Repository admins Dheeru Dua and Efi Karra Taniskidou!
04-04-2013: Welcome to the new Repository admins Kevin Bache and Moshe Lichman!
03-01-2010: Note from donor regarding Netflix data
10-16-2009: Two new data sets have been added.
09-14-2009: Several data sets have been added.
03-24-2008: New data sets have been added!
06-25-2007: Two new data sets have been added: UJI Pen Characters, MAGIC Gamma Telescope

Featured Data Set: Abalone

Task: Classification
Data Type: Multivariate
 # Attributes: 8
 # Instances: 4177

Newest Data Sets:

07-30-2019: PPG-DaLiA
07-24-2019: Divorce Predictors data set
07-22-2019: Alcohol QCM Sensor Dataset
07-14-2019: Incident management process enriched event log
06-30-2019: Wave Energy Converters
06-22-2019: Query Analytics Workloads Dataset

Most Popular Data Sets (hits since 2007):

2808763: Iris
1568695: Adult
1217307: Wine
1028814: Car Evaluation
1009132: Wine Quality
998034: Heart Disease

Соревнование от Сбербанка

Агрегатор соревнований

Тренировки ML

Тренировки и разбор соревнований по анализу данных

СООБЩИТЬ О СОРЕВНОВАНИИ

ХОЧУ ВЫСТУПИТЬ

Активны | Завершены | Все | Поиск

Zindi: Farm Pin Crop Detection Challenge

4 марта 2019 — 15 сентября 2019
Осталось 3 дня, 15 часов

The objective of this competition is to create a machine learning model to classify fields by crop type using Sentinel-2 satellite imagery. The fields in this training set are along the Orange River, a major agricultural region in South Africa that has been stricken by drought in recent years.

fastMRI Challenge

5 сентября 2019 — 19 сентября 2019
Осталось 1 неделя

The winners of the challenges will be granted reserved registration slots to the NeurIPS 2019 conference. To participate in the fastMRI challenge, you must run your model on the challenge dataset and submit the reconstructions along with a 1-page abstract describing your model to the fastMRI website. 1 page abstract you submit needs to follow the requirements of the Medical Imaging Workshop meets NeurIPS workshop guidelines.

Topcoder: PINS Master: Extracting OI Ionogram Parameters

24 июня 2019 — 20 сентября 2019
Осталось 1 неделя, 1 день

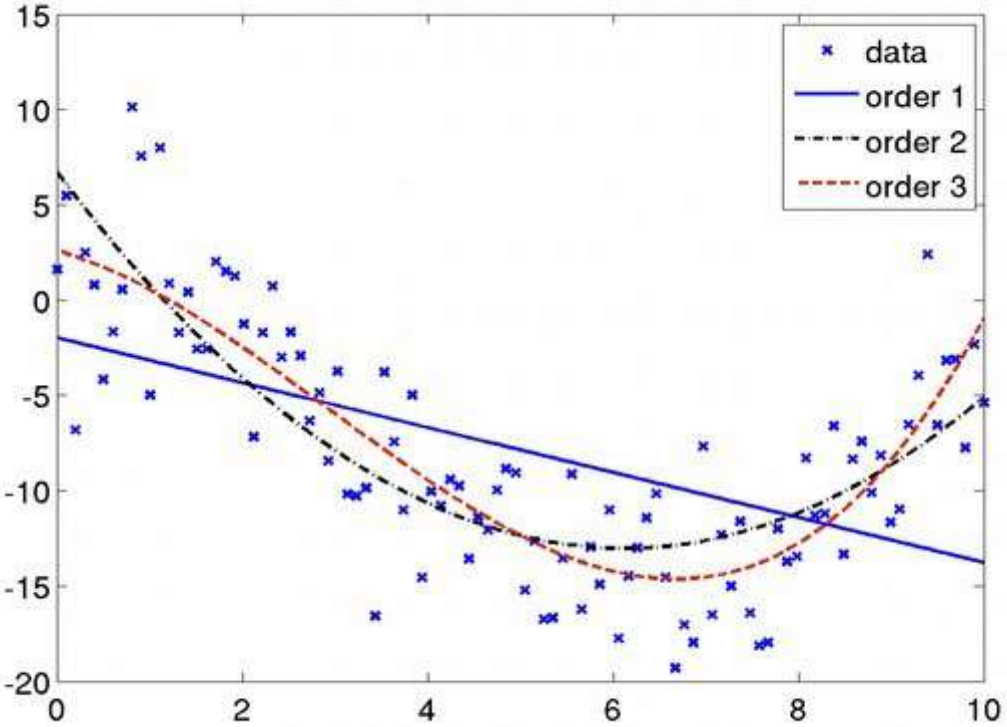
The goal of the Master Challenge is to derive specification of the HF sky-wave environment (the bottom-side ionosphere) across a longer circuit. This will be accomplished through passive reception of active sounders from an Oblique Incidence (OI). This process will require modification of the solver algorithms developed in the Explorer Challenge to determine ionospheric characteristics derived from more difficult OI datasets.

CrowdANALYTIX: Quality Forecasting in Cement Manufacturing

15 августа 2019 — 22 сентября 2019
Осталось 1 неделя, 3 дня

The objective of this contest is as follows: Task 1/ Task 2: Develop a robust model to predict Free Lime. Task 3: Identify significant parameters which influence the target variable. Variable to be predicted (Free Lime): Output Parameter Outcome: The successful solvers will have created the most accurate and robust model to predict Free Lime on the data provided.

Машинное обучение
Лекция 2. Основные понятия



Содержание лекции

- Задача обучения
- Матрица объектов-признаков
- Модель алгоритмов и метод обучения
- Функционал качества
- Вероятностная постановка задачи обучения
- Проблема переобучения

Литература

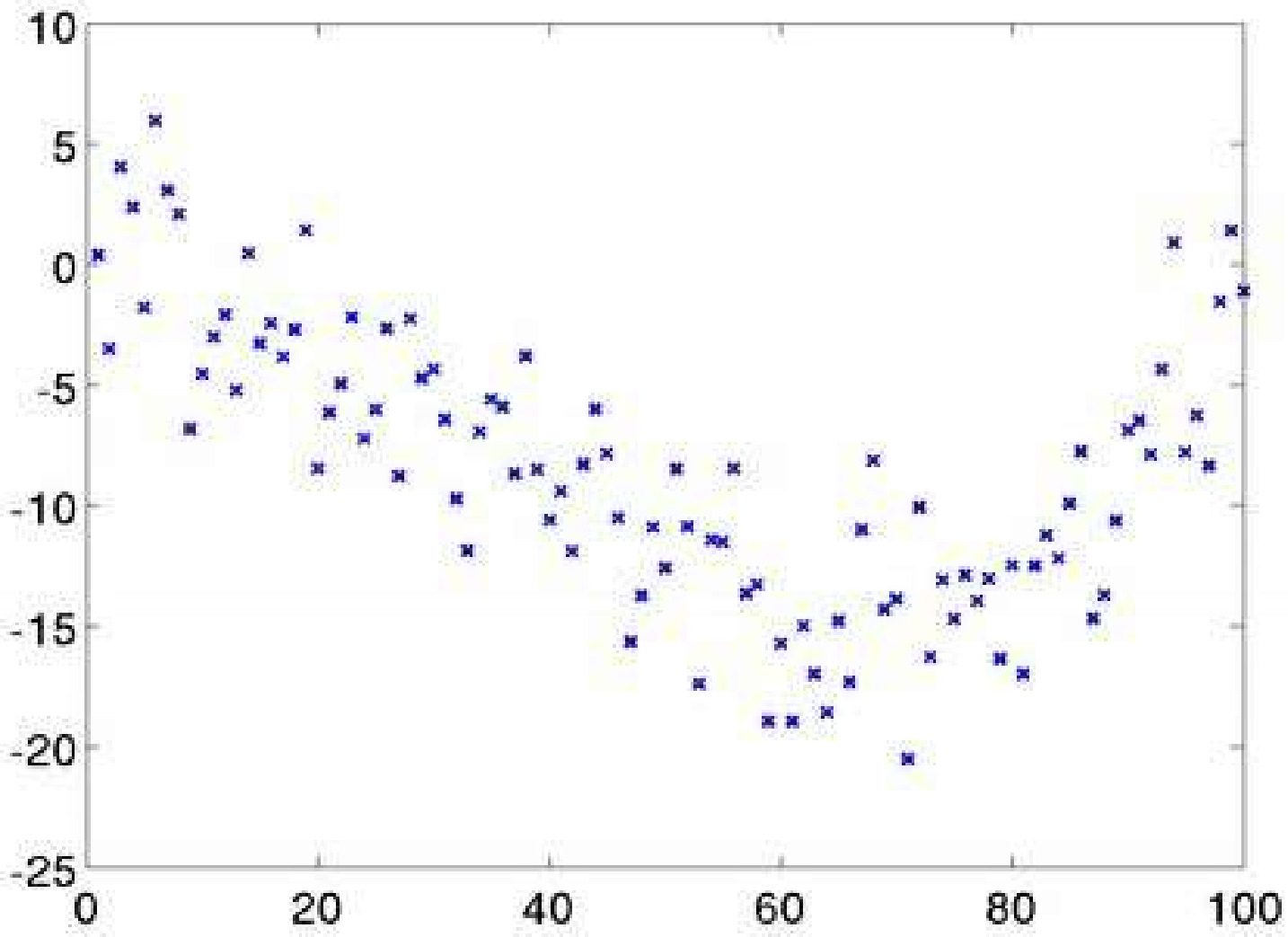
- http://www.machinelearning.ru
- Курс К.В.Воронцова
- https://www.kaggle.com/

Задача обучения

X — множество объектов
Y — множество ответов
$y : X \to Y$ — неизвестная зависимость (target function)

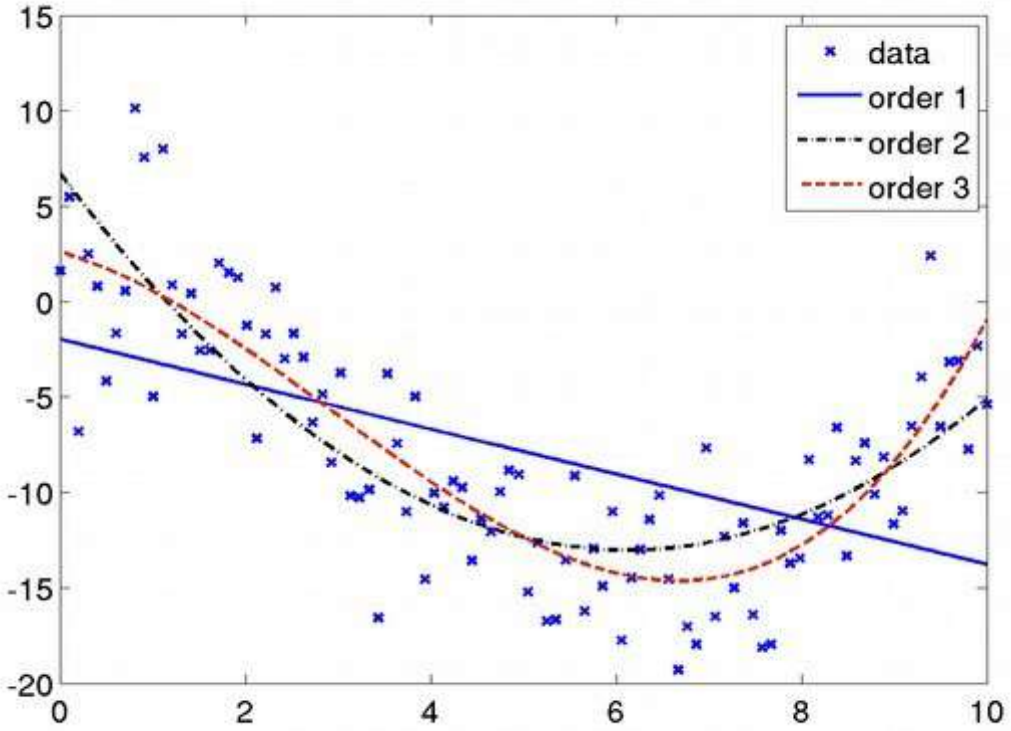
Дано:
$\{x_1, \ldots, x_\ell\} \subset X$ — обучающая выборка (training sample)
$y_i = y(x_i)$, $i = 1, \ldots, \ell$ — известные ответы

Задача обучения



Задача обучения

Найти:
$a : X \to Y$ — алгоритм, решающую функцию (decision function), приближающую y на всём множестве X



Типы задач

Задачи классификации (classification):

$Y = \{-1, +1\}$ — классификация на 2 класса

$Y = \{1, \ldots, M\}$ — на M непересекающихся классов (multi-class classification)

$Y = \{0, 1\}^M$ — на M классов, которые могут пересекаться (multi-label classification).

Задачи восстановления регрессии (regression):

$Y = R$ или $Y = R^m$

Задачи ранжирования (ranking):

Y — конечное упорядоченное множество

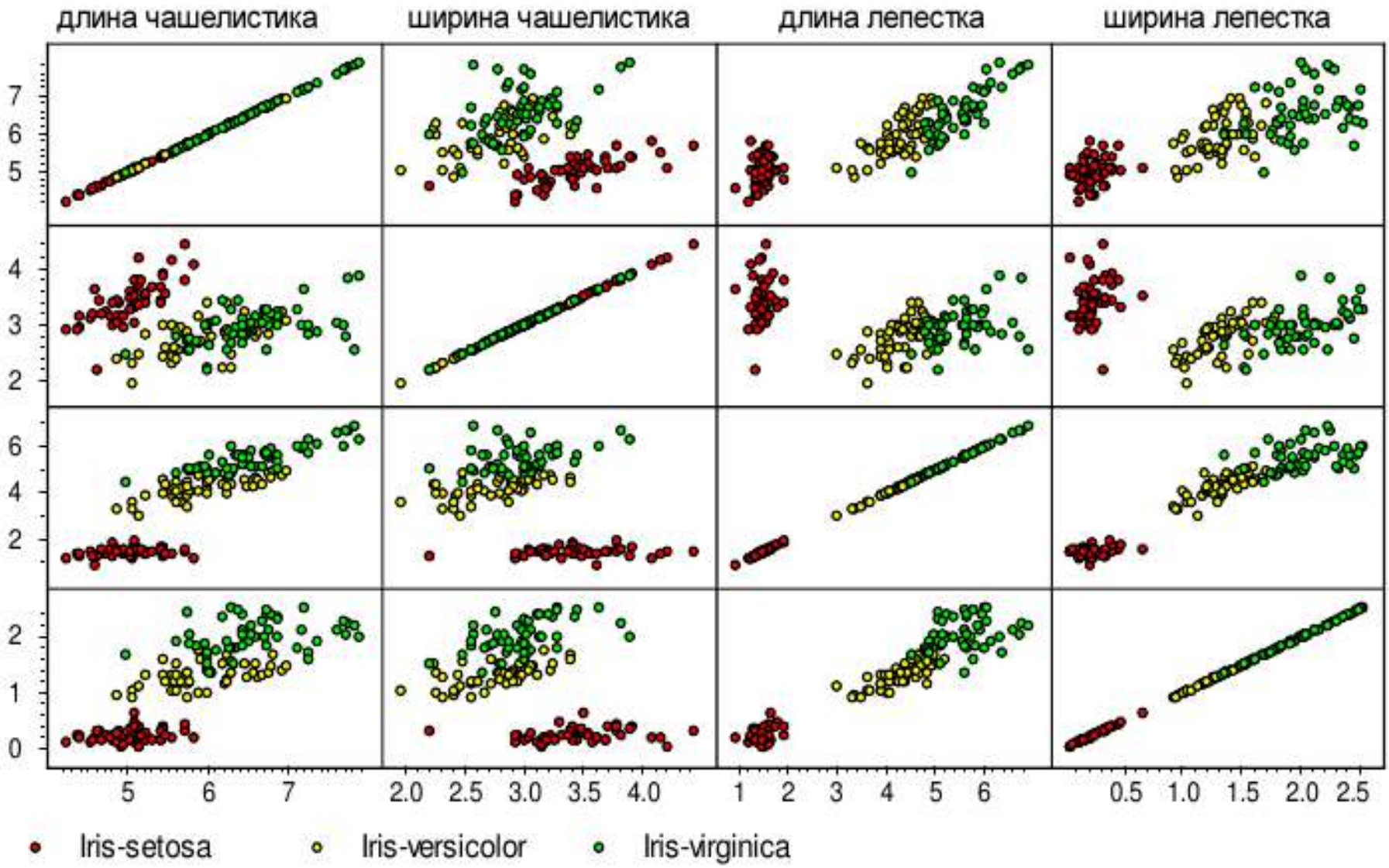
Признаки

- Компьютер всегда имеет дело с признаковым описанием объектов. Например: пациента можно описать признаками: имя, возраст, номер полиса, жалобы, давление, температура, результаты анализов
- $f : X \to D_f$
- Типы признаков:
  - бинарный
  - номинальный
  - порядковый
  - количественный

Матрица объектов-признаков:

$$\begin{pmatrix} f_1(x_1) & \dots & f_n(x_1) \\ \dots & \dots & \dots \\ f_1(x_\ell) & \dots & f_n(x_\ell) \end{pmatrix}$$

Пример. Задача классификации видов ириса (Фишер 1936)



Модель и алгоритм обучения

- **Модель** – это семейство “гипотез”

$$A = \{g(x, \theta) \mid \theta \in \Theta\}$$

одна из которых (как мы надеемся) хорошо приближает целевую функцию

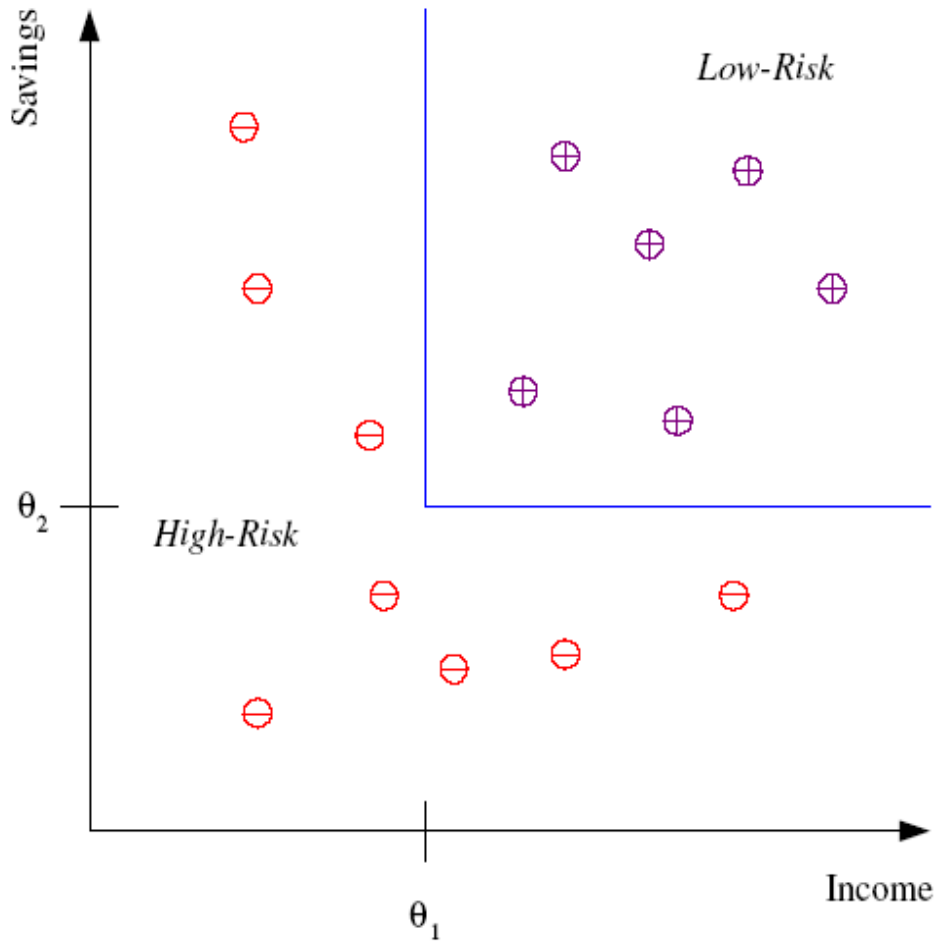
- **Алгоритм обучения**

$$\mu \colon (X \times Y)^{\ell} \to A$$

находит гипотезу в модели, которая наилучшим образом приближает целевую функцию, используя известные значения (обучающую выборку)

Пример - классификация

- Кредитный скоринг
- Разделение клиентов на low-risk и high-risk по их зарплате и сбережениям



IF *income* > θ_1 AND *savings* > θ_2

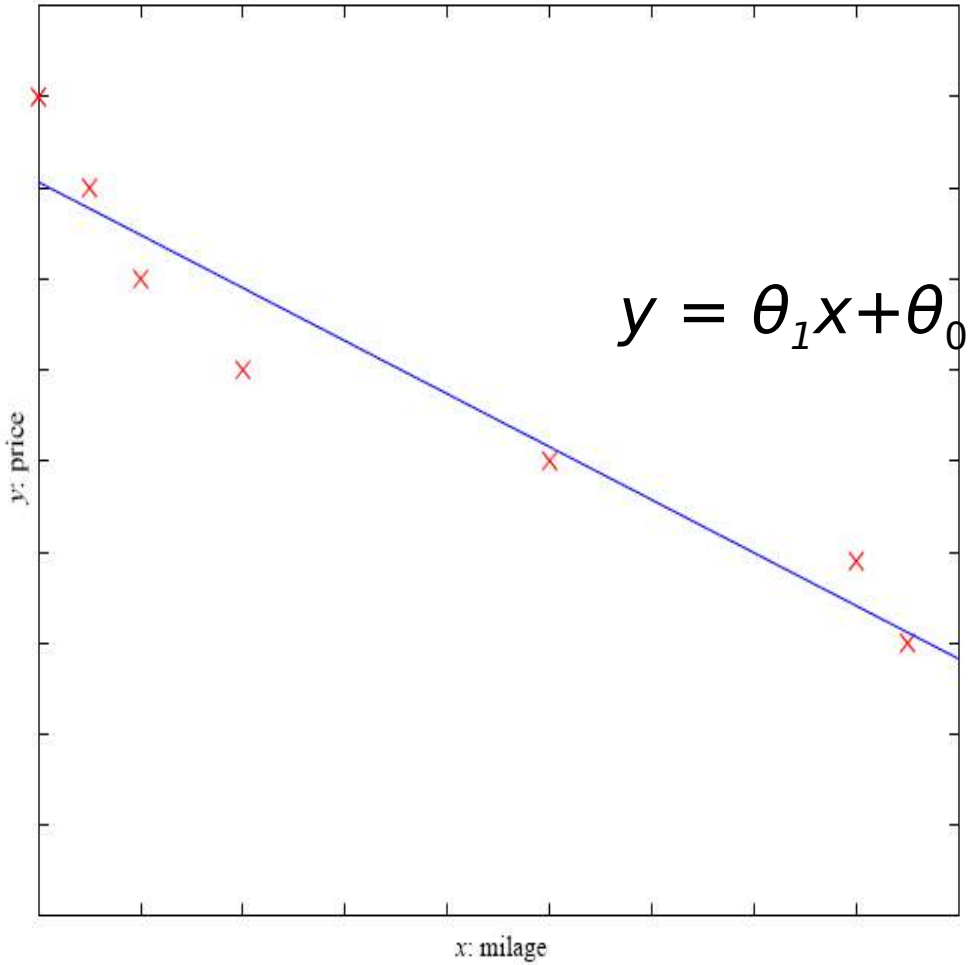
THEN low-risk ELSE high-risk

Модель

11

Пример - регрессия

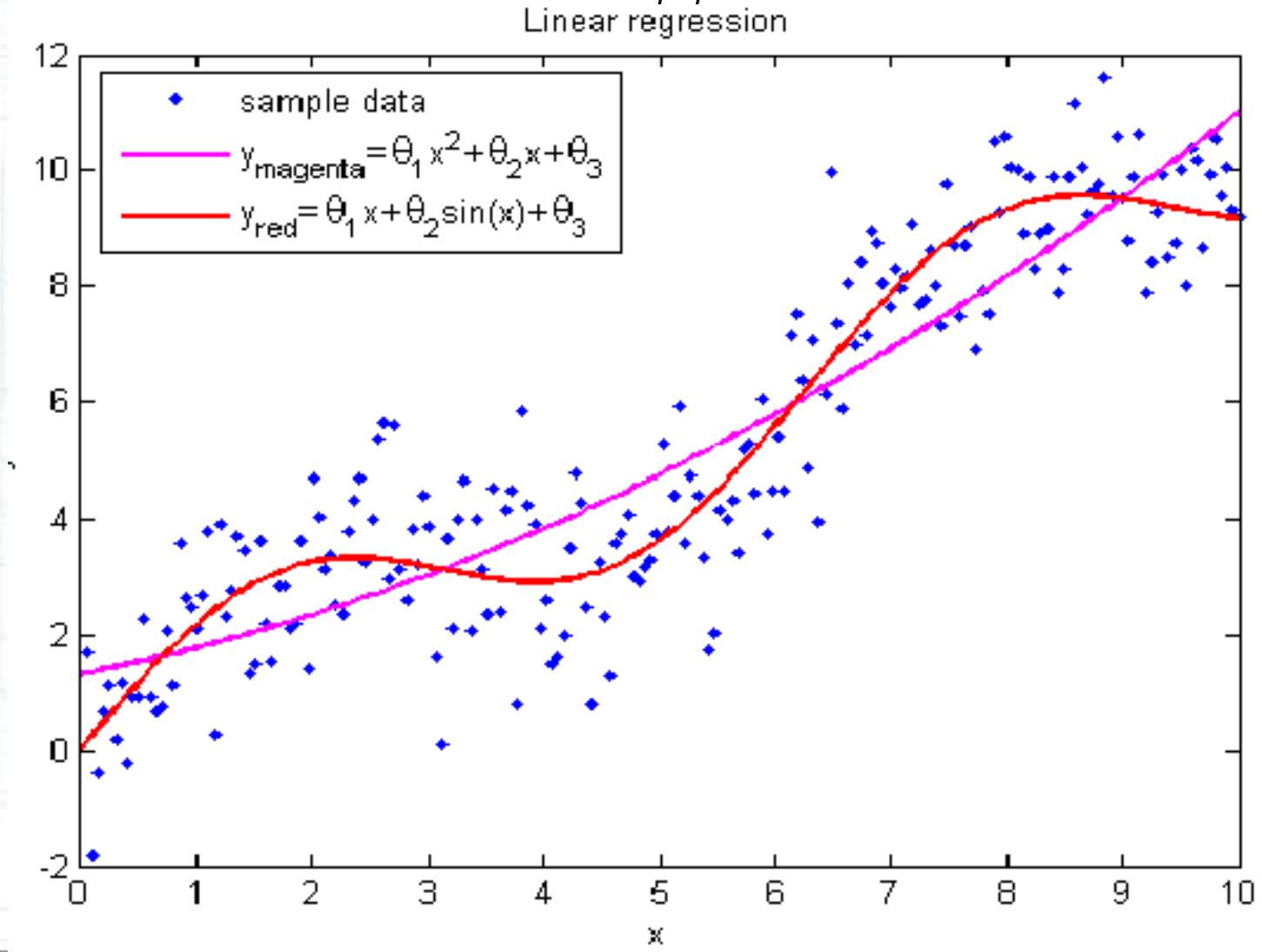
- y - цена автомобиля
- x – пробег
- $y = \theta_1 x + \theta_0$ – модель
- θ_0, θ_1 - *параметры*



12

Пример – две точки зрения

1. x – один признак, $\theta_1 x^2+\theta_2 x+\theta_3$ и $\theta_1 x+\theta_2 sin(x)+\theta_3$ – две модели
2. $\{x^2,x\}$, $\{x,sin(x)\}$ – два набора разных признаков, модель – одна (линейная $\theta_1 f_1+\theta_2 f_2+\theta_3$)



Обучение на основе минимизации эмпирического риска

- Функция потерь $\mathscr{L}(a, x)$ - величина ошибки гипотезы a на объекте x. Примеры:
  - бинарная (где используется?)
  - $\mathscr{L}(a, x) = |a(x) - y^*(x)|$
  - $\mathscr{L}(a, x) = (a(x) - y^*(x))^2$
- Эмпирический риск: $Q(a, X^\ell) = \frac{1}{\ell} \sum_{i=1}^{\ell} \mathscr{L}(a, x_i)$
- Самый популярный алгоритм обучения – минимизация эмпирического риска:

$$\mu(X^\ell) = \arg\min_{a \in A} Q(a, X^\ell)$$

Проблемы реальных задач

- Одинаковые признаковые описания могут соответствовать разным объектам
- Объекты с похожими (даже одинаковыми) значениями признаков могут иметь различные значения целевой функции

Вероятностная постановка задачи

- p(x,y) – неизвестная точная плотность распределения на X×Y
- X^ℓ - выборка из случайных, независимых и одинаково распределенных прецендентов
- $p(X^\ell) = p\big((x_1, y_1), \ldots, (x_\ell, y_\ell)\big) = p(x_1, y_1) \times \cdots \times p(x_\ell, y_\ell)$
- $\varphi(x, y, \theta)$ - модель
- Принцип максимума правдоподобия:

$$L(\theta, X^\ell) = \prod_{i=1}^{\ell} \varphi(x_i, y_i, \theta) \to \max_{\theta}$$

Decision function

- Предположим, что мы нашли вероятность p(y|x)=p(x,y)/p(x). Какое значение y нужно предсказать для заданного x ?
- Минимизация среднего риска:

$$a(x) = \arg\min_{s} E_y \mathscr{L}(s, y)$$

- Упражнение:

y	2	3	4	5
p(y|x)	0.1	0.2	0.3	0.4

примите правильные решения a(x) для каждой функции потерь со слайда 14

Несимметричные потери

- Пример: нужно предсказать, сколько фирма потратит на рекламу
(**m** - мало 50 т.р, **B** - много 200 т.р.)
- $\mathcal{L}$(a=m, y=B) = ???
- $\mathcal{L}$(a=B, y=m) = ???

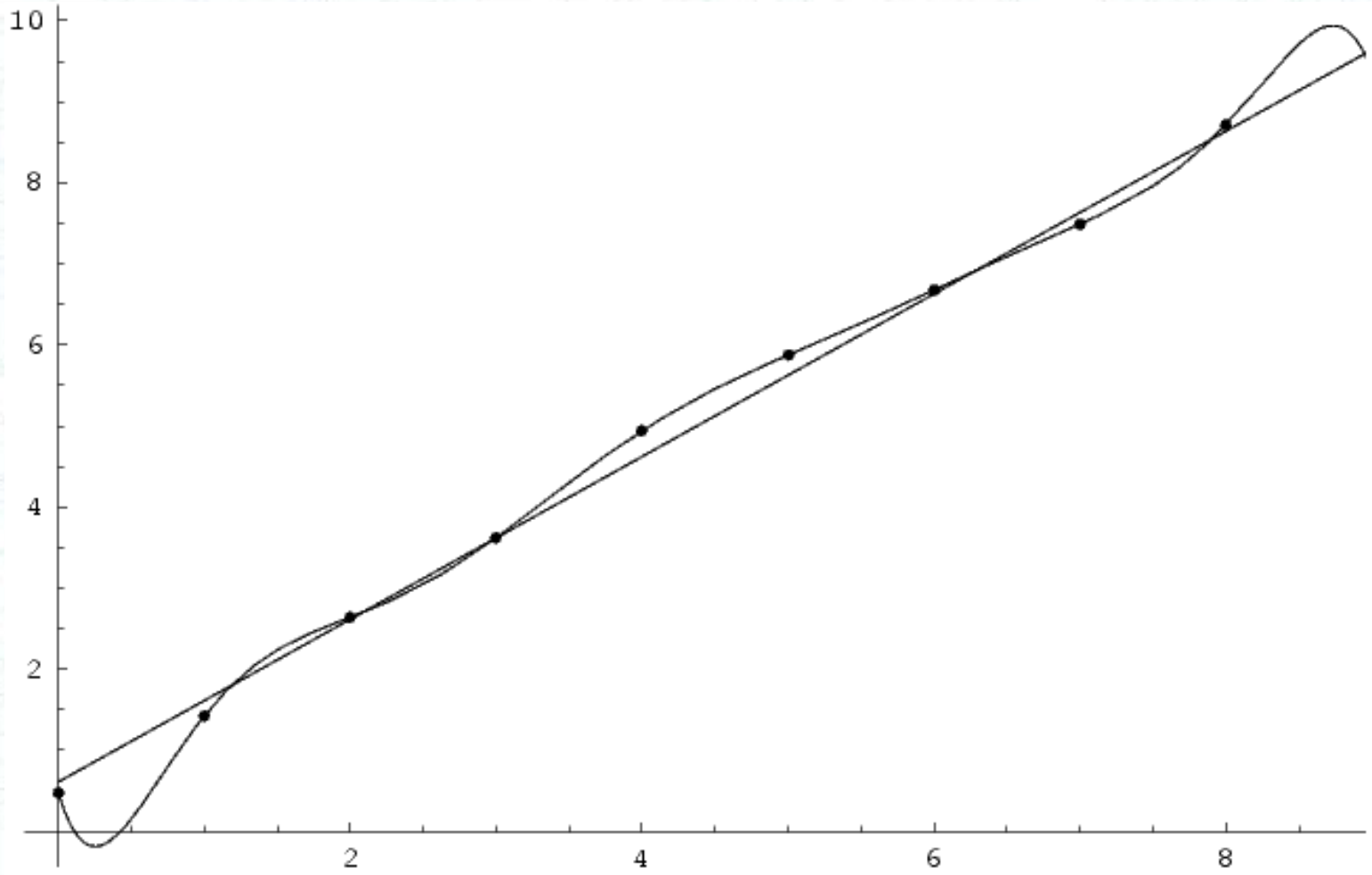
Несимметричные потери

- Пример: нужно предсказать, сколько фирма потратит на рекламу
  (**m** - мало 50 т.р, **B** - много 200 т.р.)
- $\mathcal{L}$(a=m, y=B) — не сделаем подарочную скидку и от нас могут отказаться и не заказать много рекламы
  $\mathcal{L}$(m, B) = 200т.р.* 0.8 (вер-ть отказа)
- $\mathcal{L}$(a=B, y=m) — сделаем большую скидку, а закажут мало рекламы по низкой цене
  $\mathcal{L}$(B, m) = 50т.р.*0.3 (скидка)
- Допустим, что метод МО предсказал
  p(m|x) = 0.9 p(B|x) = 0.1
  Скидку делать?

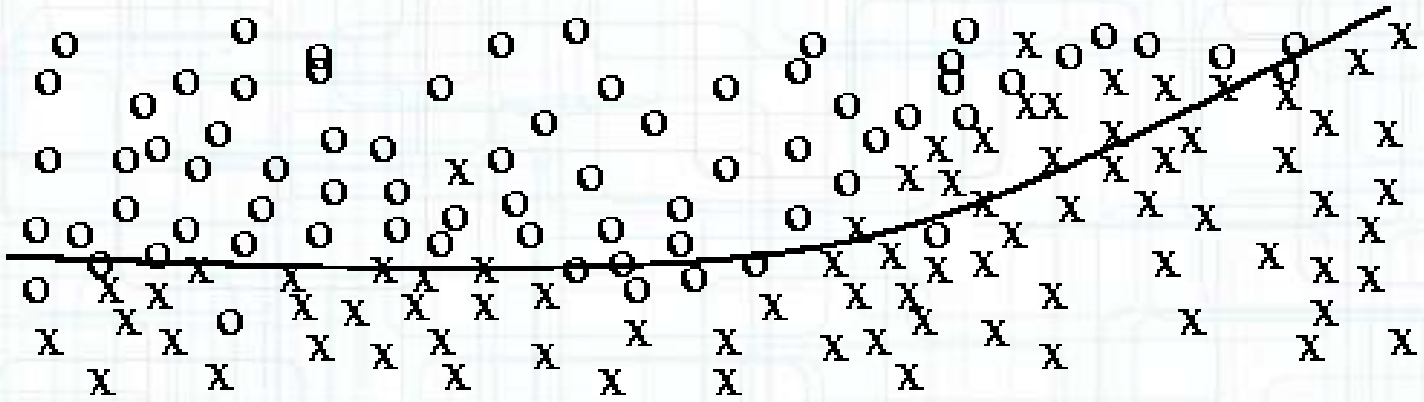
Степени обученности модели

- Недообученная модель
  - Модель, слишком сильно упрощающая закономерность $X \rightarrow Y$.
- Переобученная модель
  - Модель, слишком сильно настроенная на особенности обучающей выборки (на шум в наблюдениях), а не на реальную закономерность $X \rightarrow Y$.

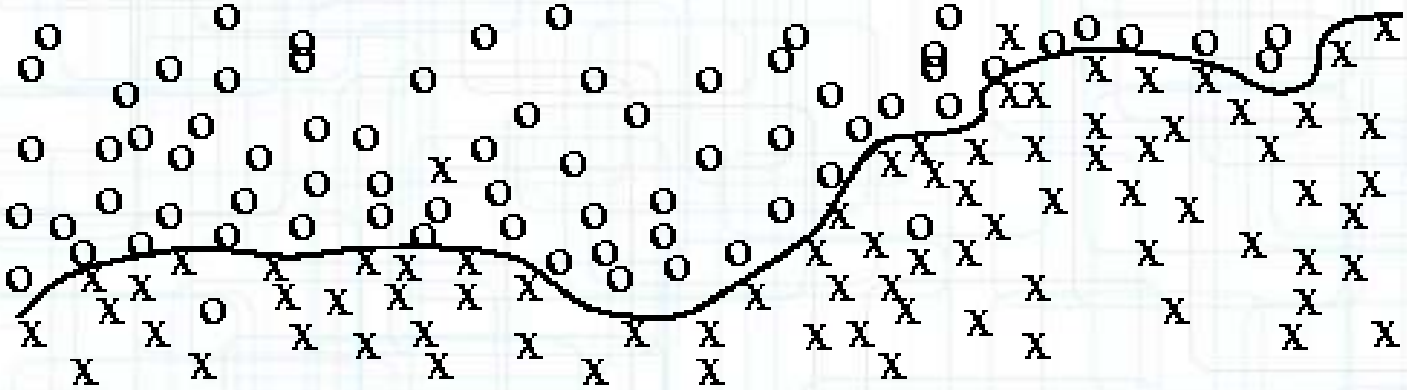
Переобучение



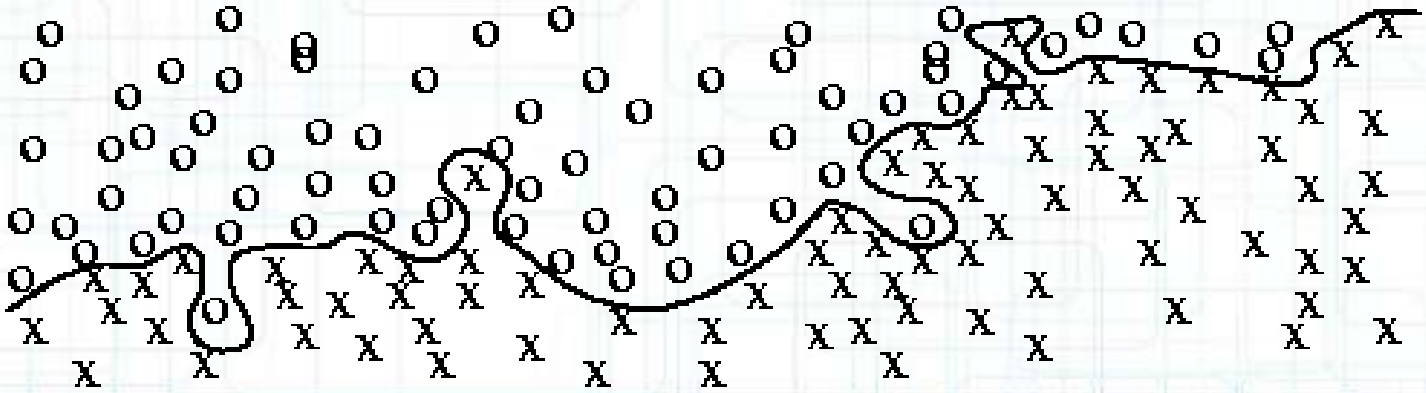
Переобучение



Under-Trained

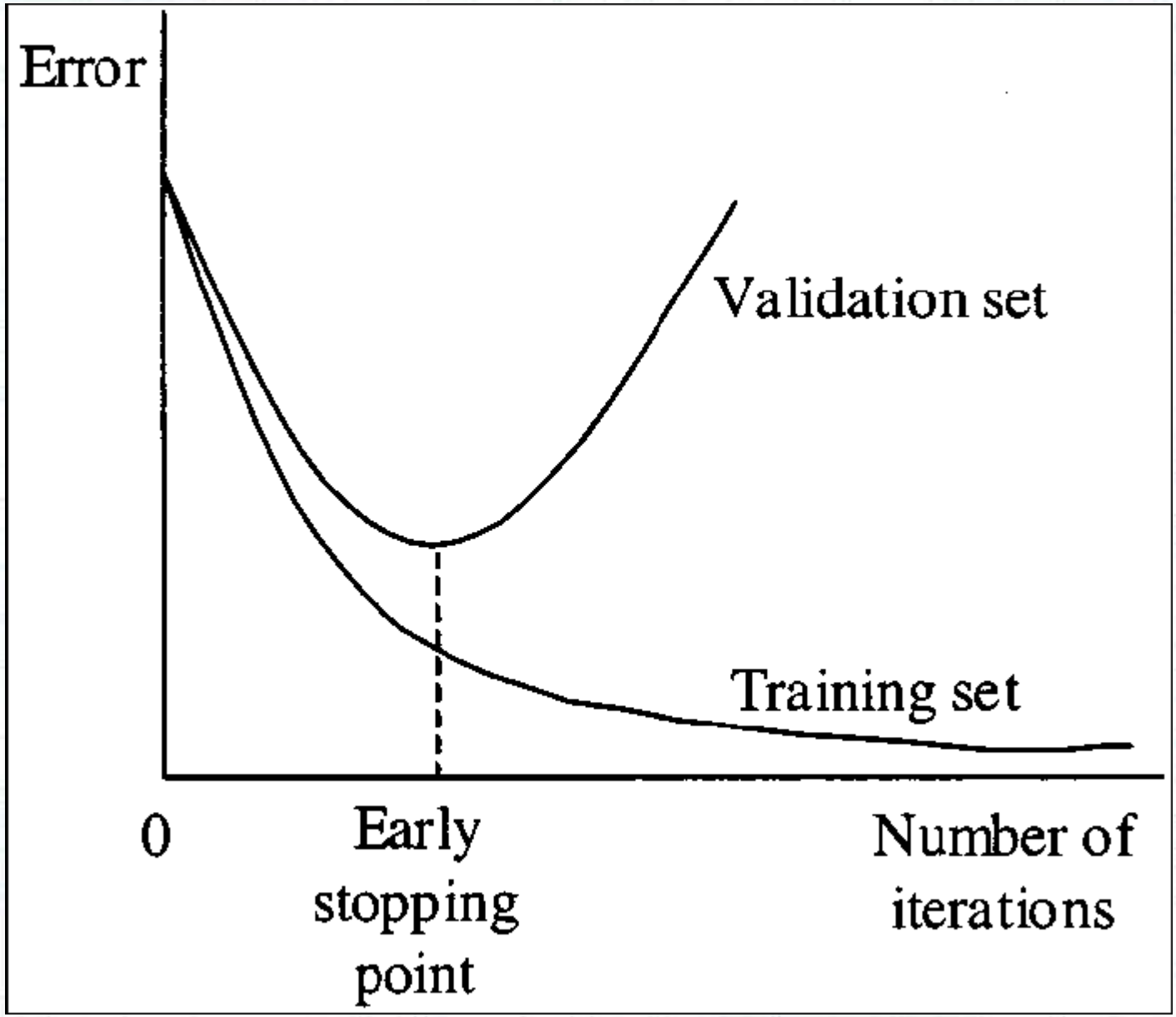


Well-Trained



Overfitted

Когда нужно заканчивать обучаться?



Контроль переобучения

- Для оценки обобщающей способности алгоритма обучения μ используют:
  - Эмпирический риск на тестовых данных (hold-out):

$$\mathrm{HO}(\mu, X^\ell, X^k) = Q(\mu(X^\ell), X^k) \to \min$$

  - Скользящий контроль (leave-one-out), L=l+1:

$$\mathrm{LOO}(\mu, X^L) = \frac{1}{L}\sum_{i=1}^{L} \mathscr{L}(\mu(X^L \setminus \{x_i\}), x_i) \to \min$$

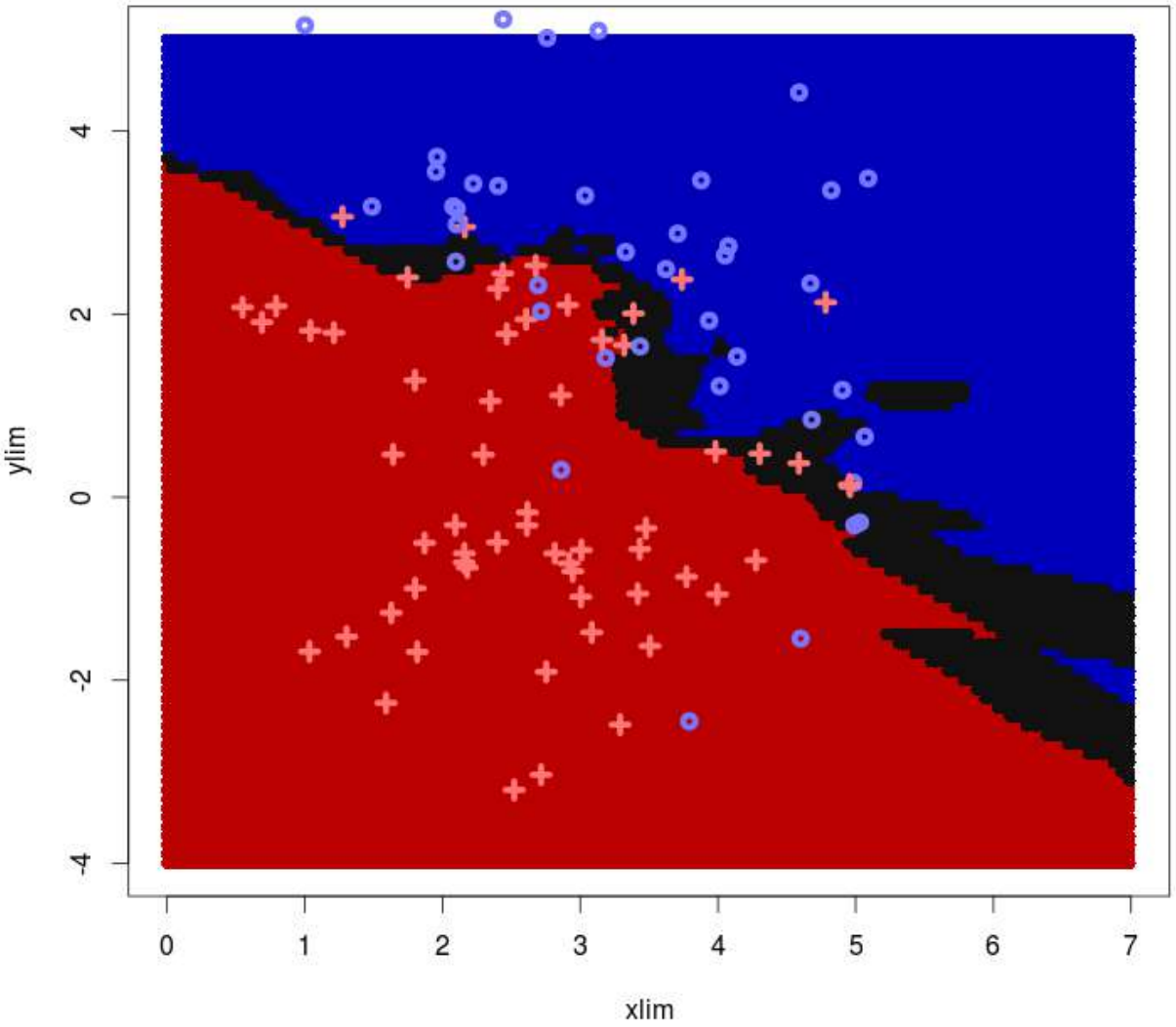
  - Кросс-проверка (cross-validation):

$$\mathrm{CV}(\mu, X^L) = \frac{1}{|N|}\sum_{n\in N} Q(\mu(X_n^\ell), X_n^k) \to \min$$

  - Оценка вероятности переобучения:

$$Q_\varepsilon(\mu, X^L) = \frac{1}{|N|}\sum_{n\in N}\Big[Q(\mu(X_n^\ell), X_n^k) - Q(\mu(X_n^\ell), X_n^\ell) \geqslant \varepsilon\Big] \to \min$$

Машинное обучение
Лекция 3. Метрические алгоритмы



https://yandexdataschool.ru/edu-process/courses/machine-learning

Содержание лекции

- Обобщенный алгоритм
- Примеры частных алгоритмов:
  - метод ближайших соседей
  - метод окна Парзена
- Понятие выступа объекта
- Алгоритм отбора эталонов
- Проклятие размерности
- Выбор метрики

Гипотезы

- Задачи классификации и регрессии:
  - X — объекты, Y — ответы;
  - $X_\ell = (x_i, y_i)$ — обучающая выборка;
- Гипотеза компактности (для классификации):
  - Близкие объекты, лежат в одном классе.
- Гипотеза непрерывности (для регрессии):
  - Близким объектам соответствуют близкие ответы.
- Формализация понятия «близости»:
  - Задана функция расстояния $\rho : X \times X \to [0, \infty)$.
- Пример. Евклидово расстояние и его обобщение:

$$\rho(x, x_i) = \left(\sum_{j=1}^{n} |x^j - x_i^j|^2 \right)^{1/2} \qquad \rho(x, x_i) = \left(\sum_{j=1}^{n} w_j |x^j - x_i^j|^p \right)^{1/p}$$

Обобщенный алгоритм

- Для заданного $x \in X$ отсортируем объекты $x_1, \ldots, x_\ell$:

$$\rho(x, x^{(1)}) \leqslant \rho(x, x^{(2)}) \leqslant \cdots \leqslant \rho(x, x^{(\ell)}),$$

  - $x^{(i)}$ — i-тый ближайший сосед объекта x
- Метрический алгоритм классификации:

$$a(x) = \arg\max_{y \in Y} \Gamma_y(x) = \arg\max_{y \in Y} \sum_{\substack{i=1 \\ y_i = y}}^{\ell} \mathrm{w}(i, x)$$

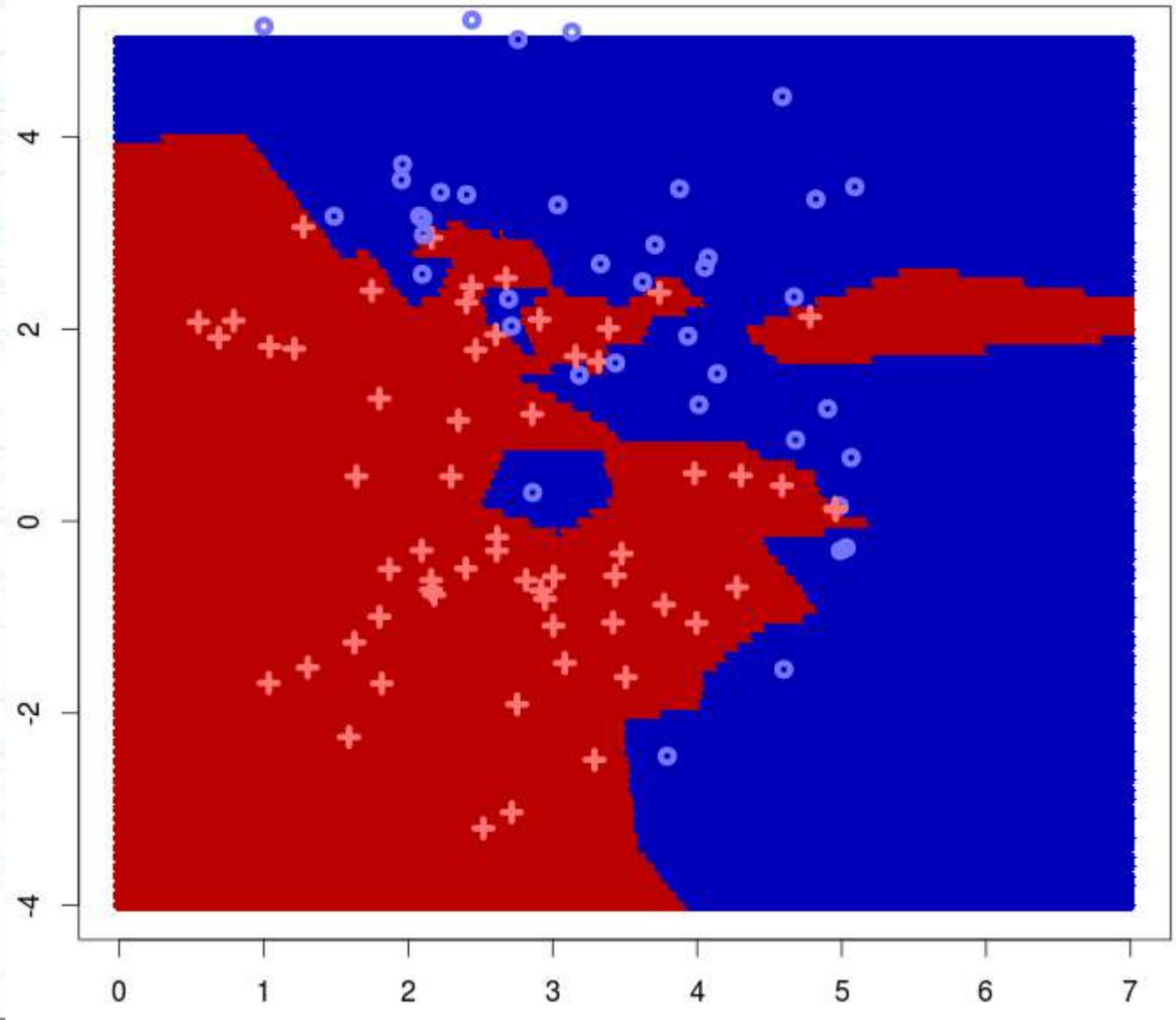
- w (i , x) — вес (степень важности) i -го соседа объекта x, ≥ 0, $\searrow$ по i
- Γ_y (x) — близость объекта x к классу y

Lazy learning

- Это так называемое ленивое обучение, в котором нет этапа тренировки параметров модели. Сразу происходит этап предсказания.
- Подходит для задач, в которых сложно сформулировать набор признаков, но легко сравнивать объекты (пример: сравнительная геномика)
- Недостаток: медленный процесс предсказания

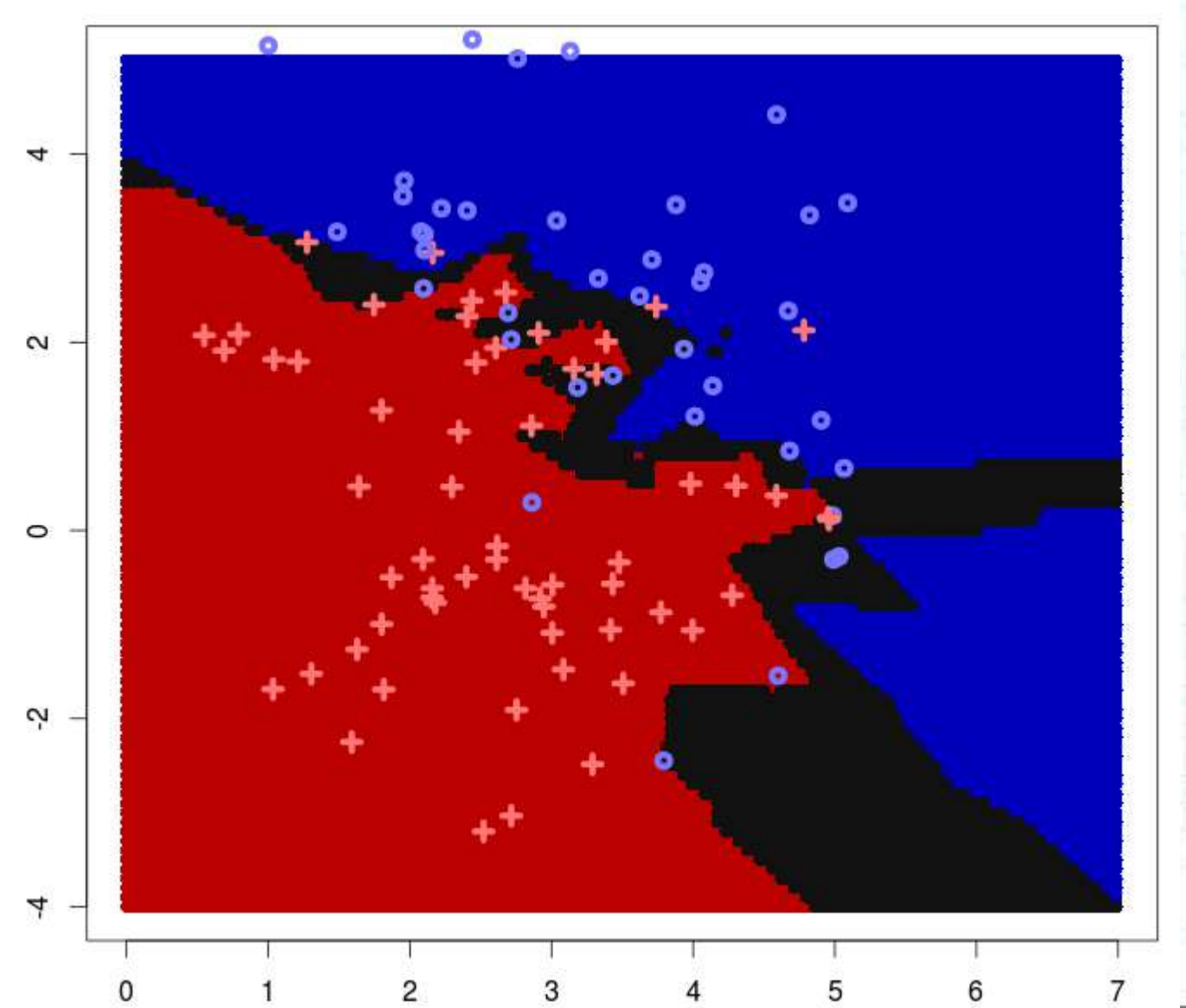
Метод ближайшего соседа

- w(i,x) = 1, если i = 1
- w(i,x) = 0, в противном случае



Метод k ближайших соседей

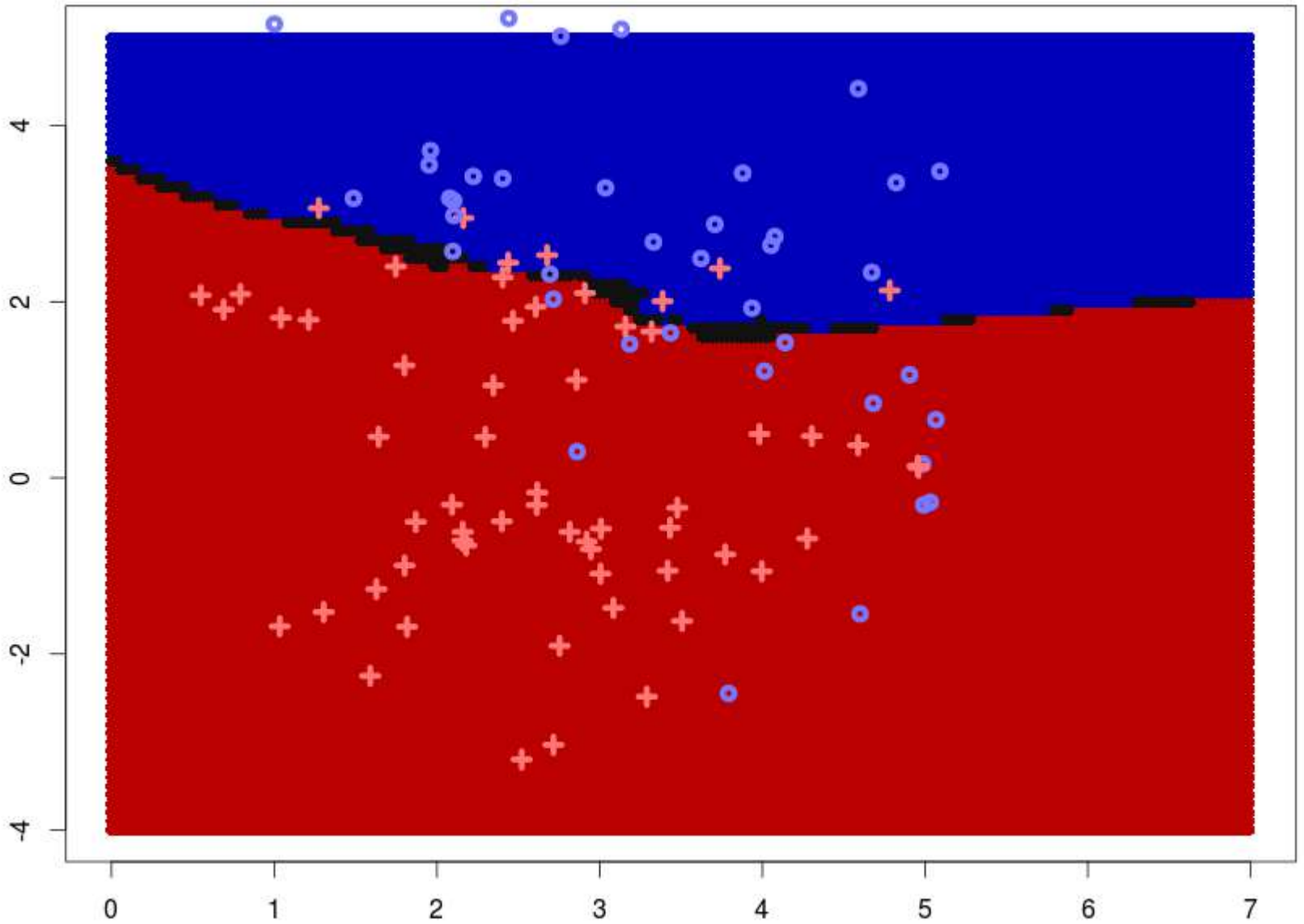
- w(i,x) = 1, если i <= k
- w(i,x) = 0, в противном случае



k = 4

Метод k ближайших соседей

- k = 60



Оптимизация k

- Функционал скользящего контроля (leave-one-out):
- Для каждого x_i
  - сортируем всех его соседей (за исключением его самого) по возрастанию расстояния
  - делаем предсказание для x_i
- Цель: минимизировать число ошибок

$$\mathrm{LOO}(k, X^\ell) = \sum_{i=1}^{\ell} \left[a\left(x_i; X^\ell \setminus \{x_i\}, k\right) \neq y_i \right] \to \min_k$$

Метод k взвешенных ближайших соседей

Проблема метода ближайших соседей – близкие и далекие учитываются с одним весом

$w(i, x) = [i \leqslant k] w_i,$

где w_i — вес, зависящий только от номера соседа;

Возможные эвристики:

$w_i = \frac{k+1-i}{k}$ — линейное убывающие веса;

$w_i = q^i$ — экспоненциально убывающие веса, $0 < q < 1$;

Метод k взвешенных ближайших соседей

Проблема метода ближайших соседей – близкие и далекие учитываются с одним весом

$w(i, x) = [i \leqslant k] w_i$,

где w_i — вес, зависящий только от номера соседа;

Возможные эвристики:

$w_i = \frac{k+1-i}{k}$ — линейное убывающие веса;

$w_i = q^i$ — экспоненциально убывающие веса, $0 < q < 1$;

Проблема. Две ситуации:

1) $\rho(x,x_i) = 1;\ 1.1;\ 1.2;\ 1.3$

2) $\rho(x,x_i) = 1;\ 1.1;\ 50;\ 51$

приведут к одним и тем же весам

Метод окна Парзена

$w(i,x) = K\left(\frac{\rho(x,x^{(i)})}{h}\right)$, где h — ширина окна,

$K(r)$ — ядро, не возрастает и положительно на $[0,1]$

Метод окна Парзена

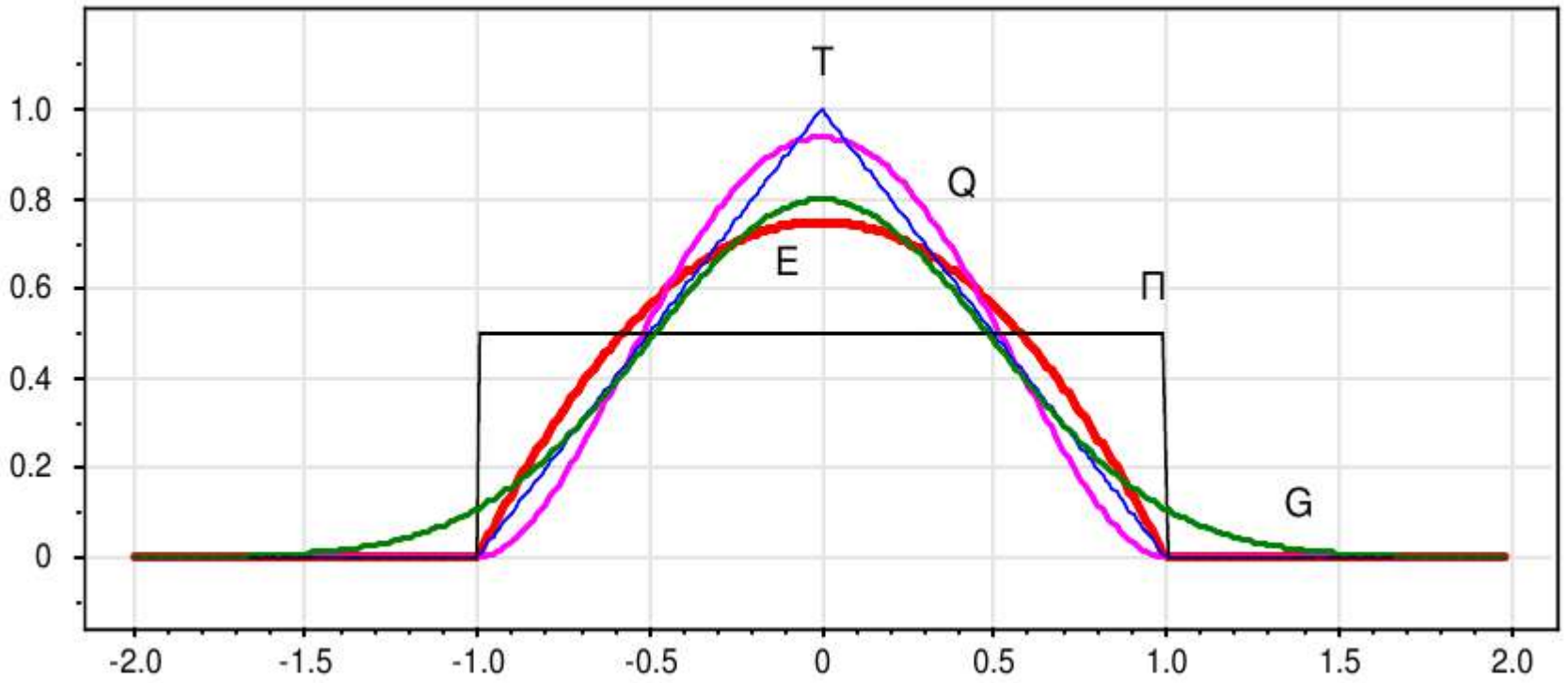
$w(i,x) = K\left(\frac{\rho(x,x^{(i)})}{h}\right)$, где h — ширина окна,

$K(r)$ — ядро, не возрастает и положительно на $[0,1]$

При фиксированной ширине окна качество классификатора сильно зависит от плотности точек.

Выход: положить ширину h равной расстоянию до k-того соседа

Часто используемые ядра



$П(r) = \big[|r| \leqslant 1\big]$ — прямоугольное

$T(r) = \big(1 - |r|\big)\big[|r| \leqslant 1\big]$ — треугольное

$E(r) = (1 - r^2)\big[|r| \leqslant 1\big]$ — квадратичное (Епанечникова)

$Q(r) = (1 - r^2)^2\big[|r| \leqslant 1\big]$ — квартическое

$G(r) = \exp\big(-2r^2\big)$ — гауссовское

Отступ (выступ) объекта

- Пусть классификатор a(x) работает по правилу:

$$a(x) = \arg\max_{y \in Y} \Gamma_y(x)$$

- Отступом (margin) объекта x_i обучающей выборки называется величина

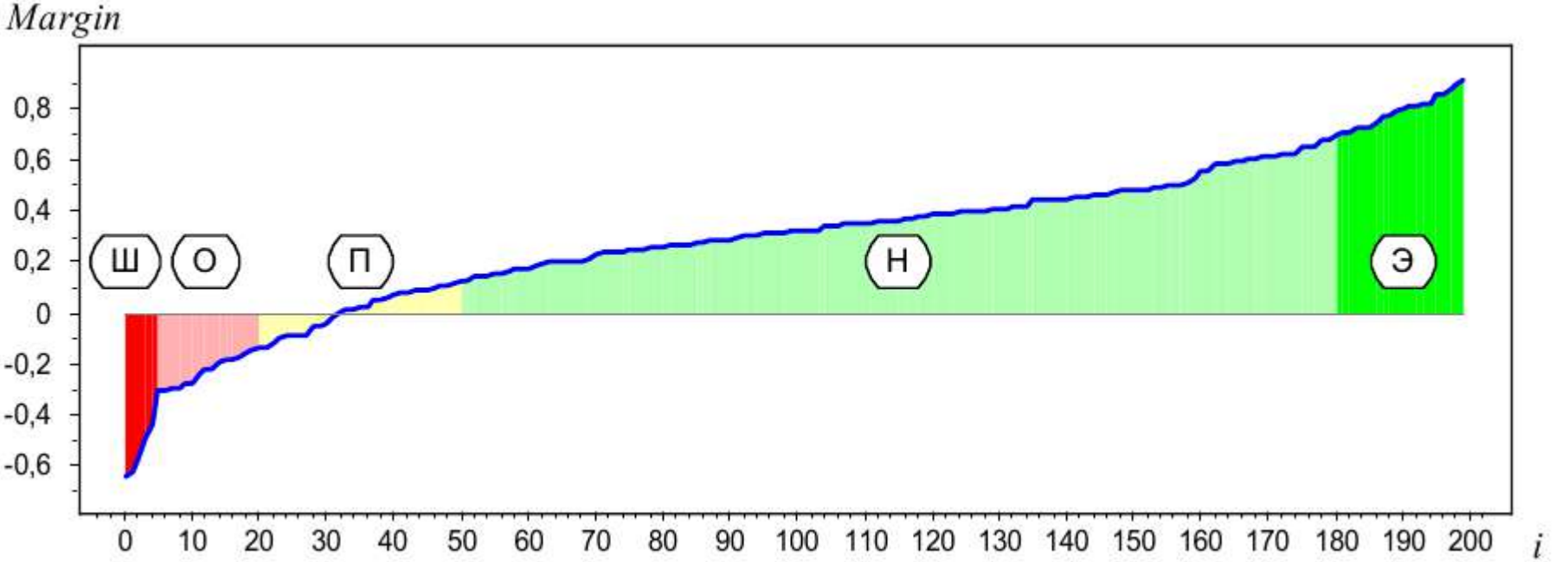
$$M(x_i) = \Gamma_{y_i}(x_i) - \max_{y \in Y \setminus y_i} \Gamma_y(x_i)$$

- Отступ показывает степень типичности объекта: чем больше $M(x_i)$, тем «глубже» x_i в своём классе;

$$M(x_i) < 0 \Leftrightarrow a(x_i) \neq y_i;$$

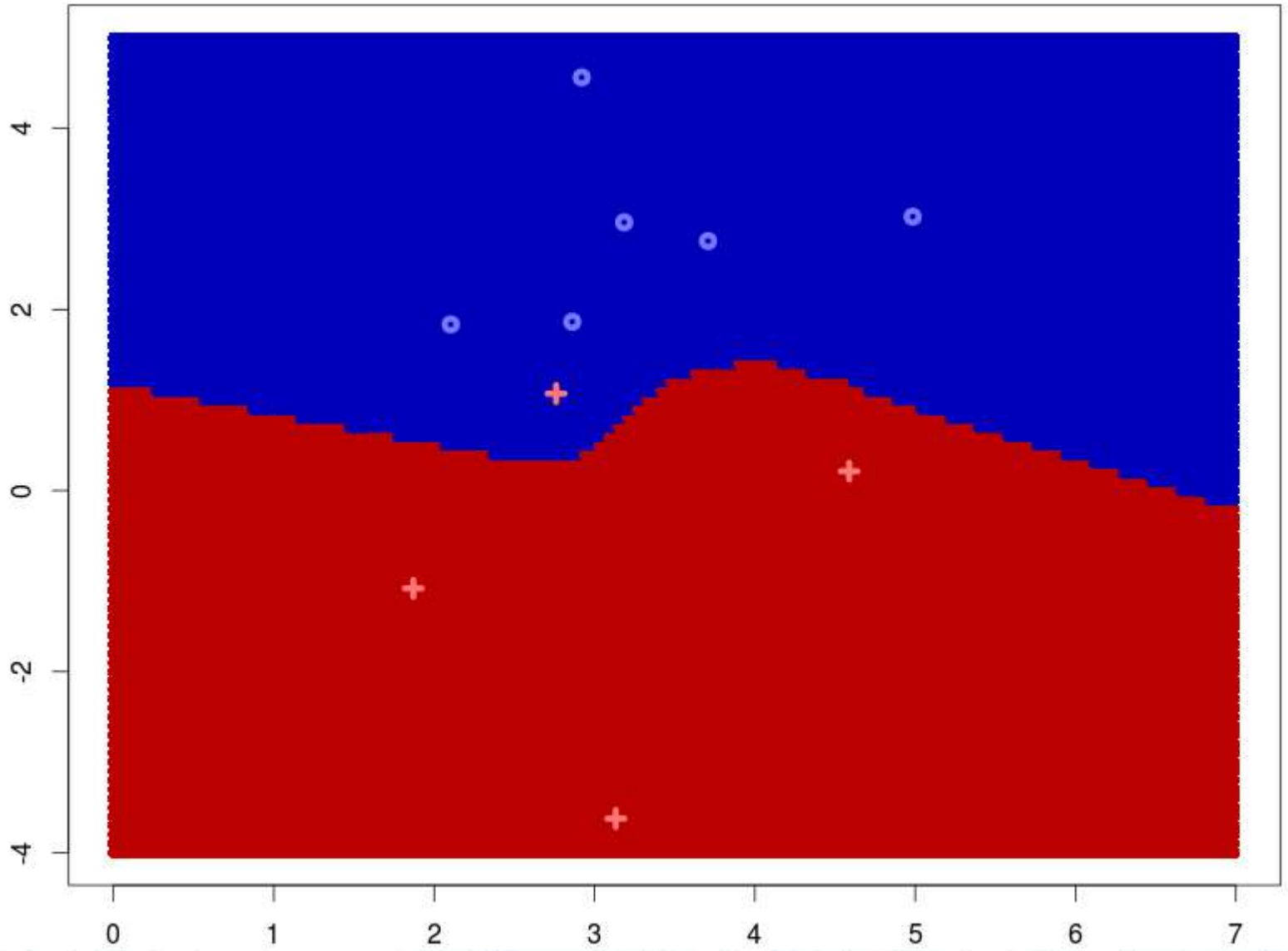
Типы объектов в зависимости от выступа

- Э — эталонные (можно оставить только их);
- Н — неинформативные (можно удалить из выборки);
- П — пограничные (их классификация неустойчива);
- О — ошибочные (причина ошибки — плохая модель);
- Ш — шумовые (причина ошибки — плохие данные).



Отступ (выступ) объекта

Вычислите отступы для всех объектов для метода 3NN



Отбор эталонов

- Задача: выбрать оптимальное подмножество эталонов Ω из обучающей выборки. Классификатор будет иметь вид:

$$a(x,\Omega) = \arg\max_{y \in Y} \sum_{\substack{x^{(i)} \in \Omega \\ y_i = y}} \mathrm{w}(i,x)$$

- Алгоритм STOLP. Три основных этапа:
  - исключить выбросы и, возможно, пограничные объекты;
  - найти по одному эталону в каждом классе;
  - добавлять эталоны, пока есть отрицательные отступы;

Алгоритм STOLP

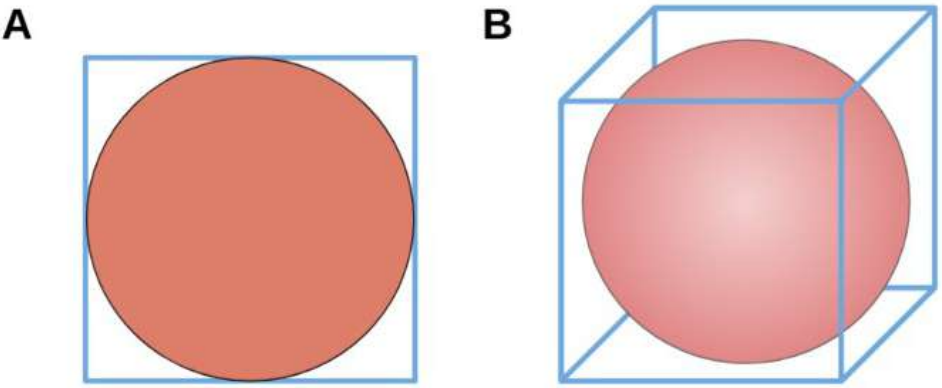
- Исключаем из X_ℓ выбросы x_i : $M(x_i)<\delta$
- Инициализируем множество эталонов Ω, выбирая по одному элементу из каждого класса с максимальным выступом
- Цикл: пока процент ошибок классификации велик и эталонов Ω не слишком много
  - добавляем в Ω объект с наименьшим выступом

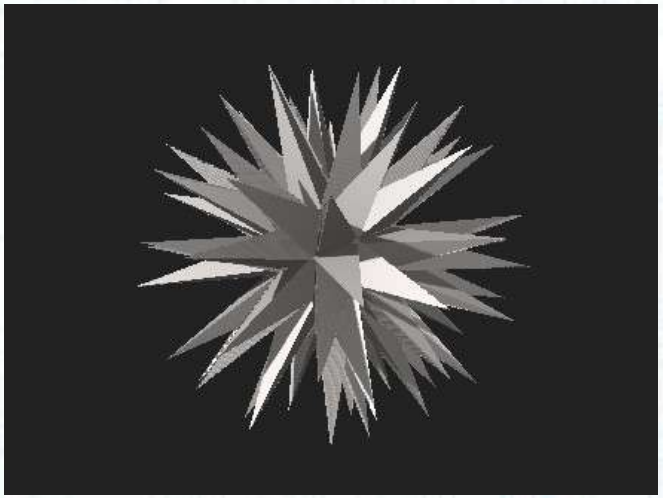
Проклятие размерности

- Проклятие размерности - усреднение значений метрики при большом количестве признаков. Почти до всех ближайших соседей расстояние одинаково
- Почему это происходит:
  - Шар радиуса R имеет объем $V(R) \sim R^D$
  - Объем шара радиуса 0.9 в 20-мерном пространстве составляет всего 12% от объема шара радиуса 1.
    Т.е. 88% точек лежит на сфере: 0.9<R<1

$$\frac{V(R-\varepsilon)}{V(R)} = \left(\frac{R-\varepsilon}{R}\right)^D \xrightarrow{D\to\infty} 0$$

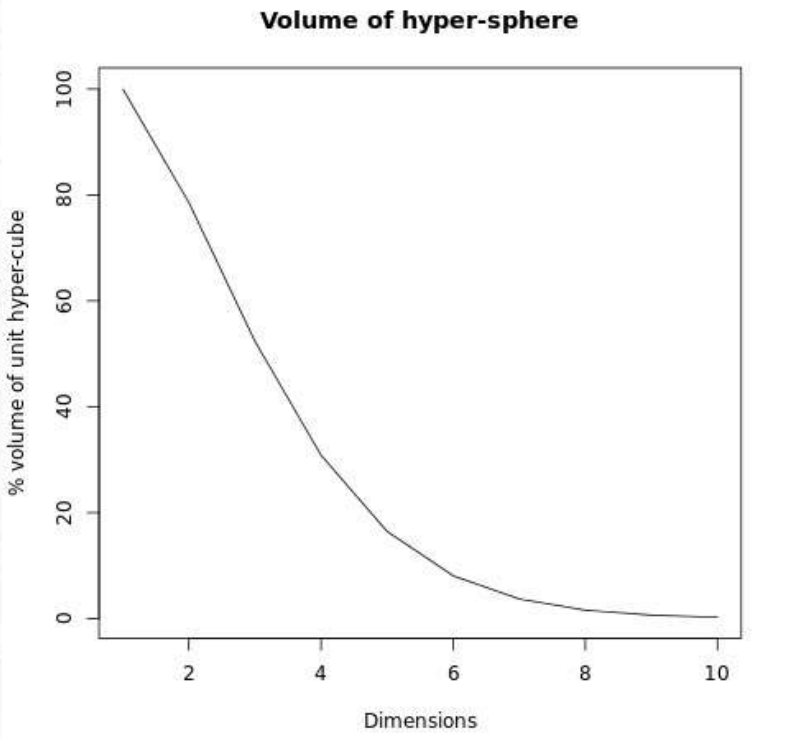
Проклятие размерности





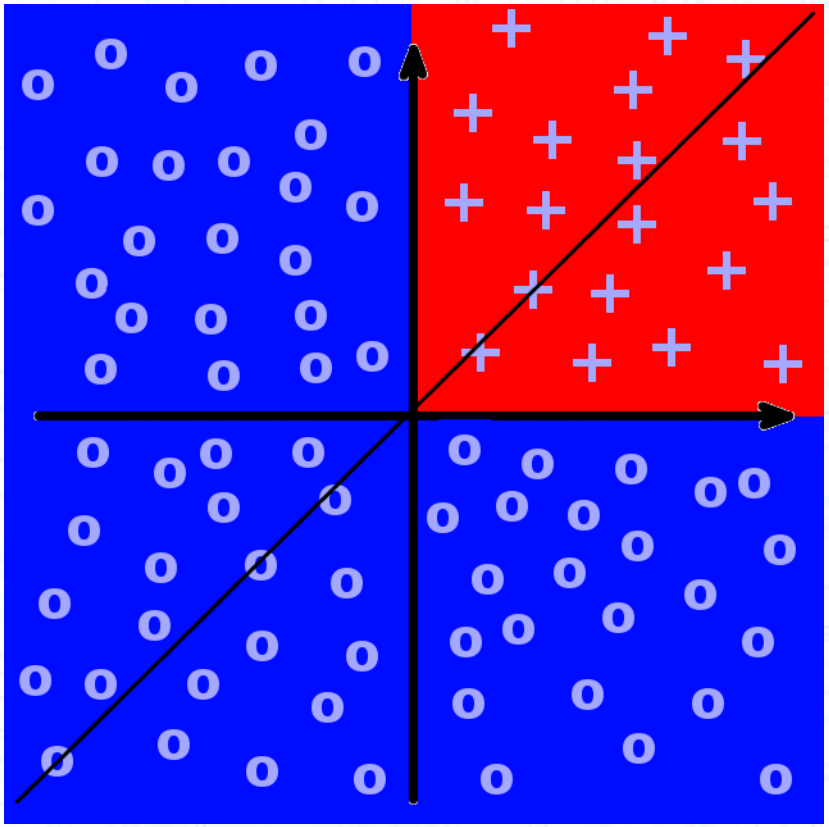
Объем вписанной в куб сферы в многомерном пространстве во много раз меньше объема куба!

Расстояние до вершины куба: $\sqrt{n}$. Количество вершин: 2^n



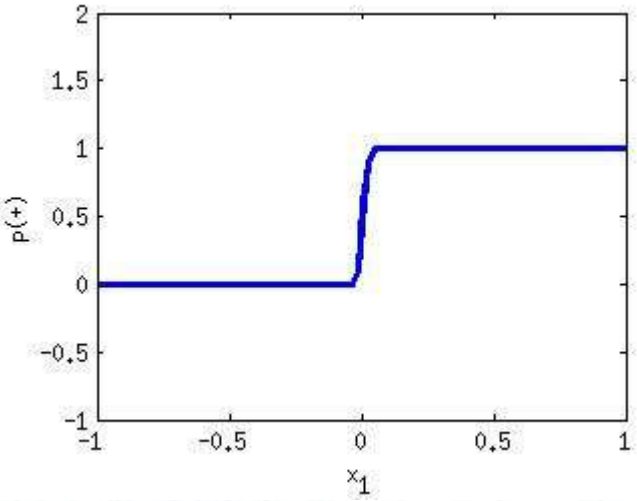
Проклятие размерности Пример

- Пространство признаков: R^n.
- Класс +: область $x_{1,2}>0$ (остальные координаты произвольны)
- X_ℓ - равномерно распределена

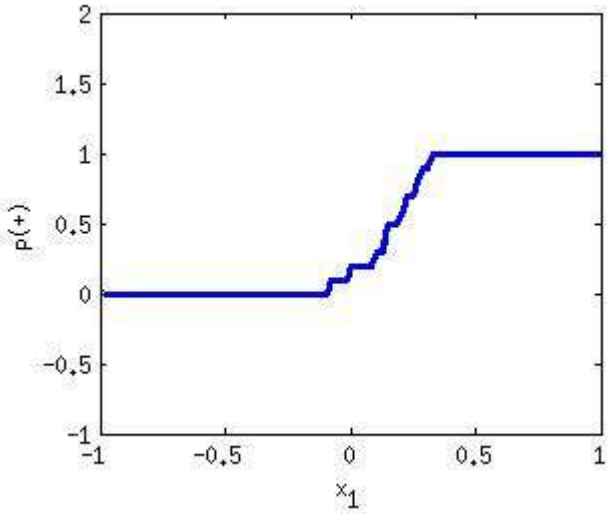


Проклятие размерности
Пример

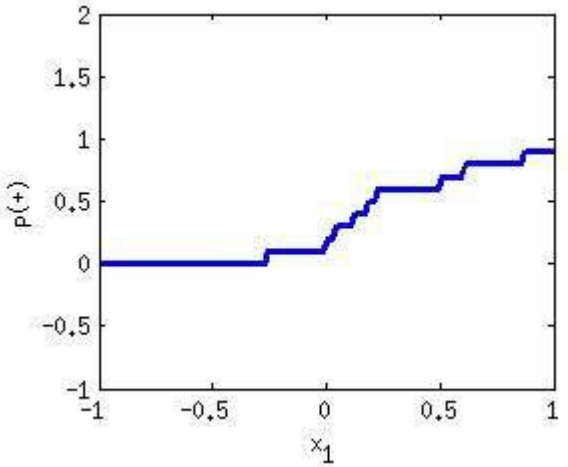
- Метод 10 ближайших соседей. ℓ = 10000
- Относительная частота класса “+” на прямой: $x_1=x_2$, $x_3=0$, $x_4=0$,...



n=2



n=5



n=20

Вывод: для больших размерностей
метрические алгоритмы сглаживают границы областей классов

Выбор метрики

Взвешенная метрика Минковского:

$$\rho(x, x_i) = \left(\sum_{j=1}^{n} w_j \left| f_j(x) - f_j(x_i) \right|^p \right)^{\frac{1}{p}},$$

где w_j — неотрицательные веса признаков, $p > 0$.

В частности, если $w_j \equiv 1$ и $p = 2$, то имеем евклидову метрику.

Роль весов w_j:

1) нормировка признаков;
2) степень важности признаков;
3) отбор признаков (какие $w_j = 0$?);

Жадное добавление признаков

1. А вдруг одного признака уже достаточно?
Расстояние по j-му признаку: $\rho_j(x, x_i) = \left|x^j - x_i^j\right|$.
Выберем наилучшее расстояние: $\mathrm{LOO}(j) \to \min$.

2. Добавим к расстоянию ρ ещё один признак j:

$$\rho^p(x, x_i) := \rho^p(x, x_i) + w_j \rho_j^p(x, x_i), \quad w_j \geqslant 0.$$

Найдём признак j и вес w_j, при которых $\mathrm{LOO}(j, w_j) \to \min$
(два вложенных цикла перебора).

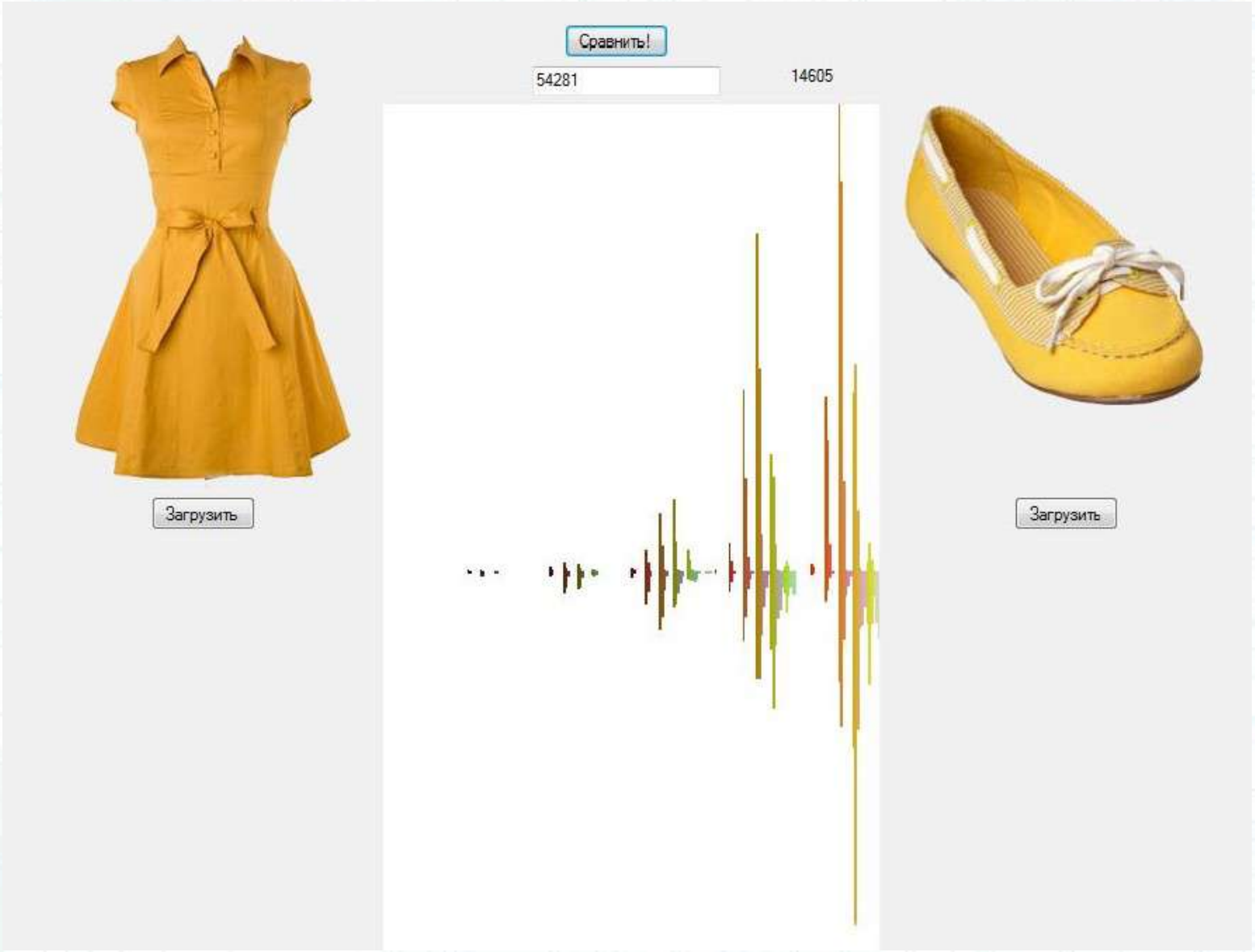
3. Можно корректировать вес признака k, уже вошедшего в ρ:

$$\rho^p(x, x_i) := \rho^p(x, x_i) + w'_k \rho_k^p(x, x_i), \quad w'_k \geqslant -w_k.$$

4. Будем добавлять признаки, пока LOO уменьшается.

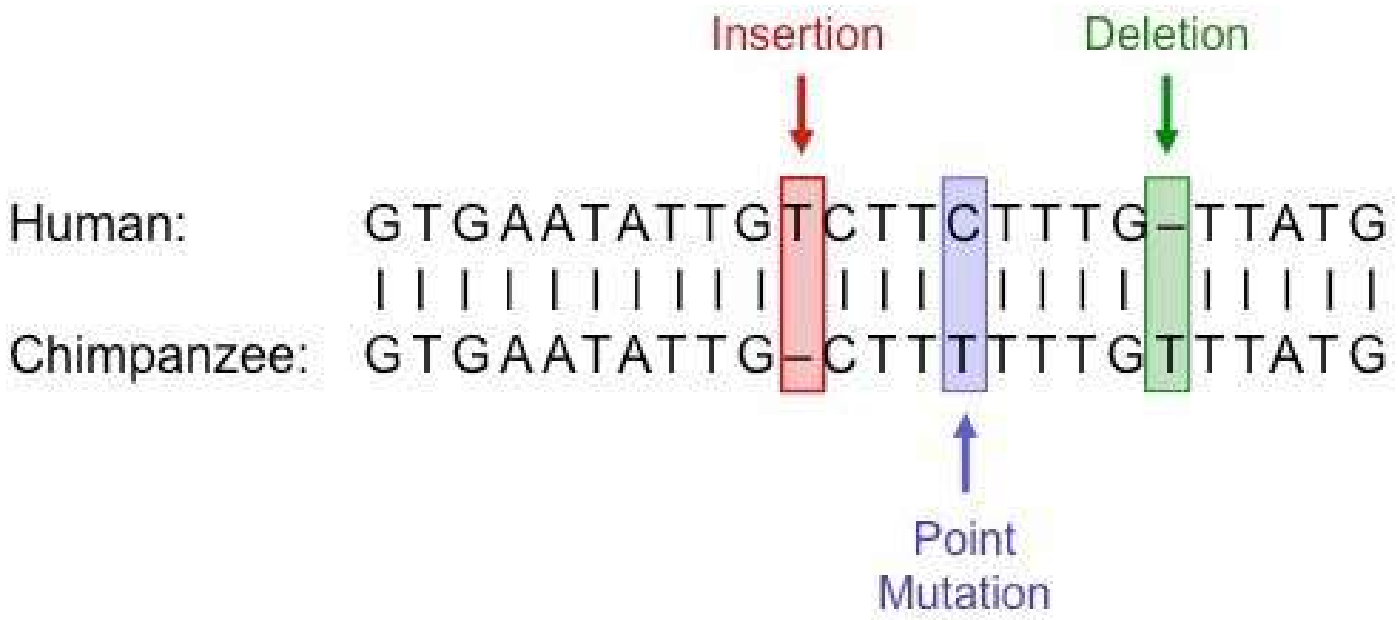
Пример необычной метрики

Задача поиска подходящих по цвету вещей

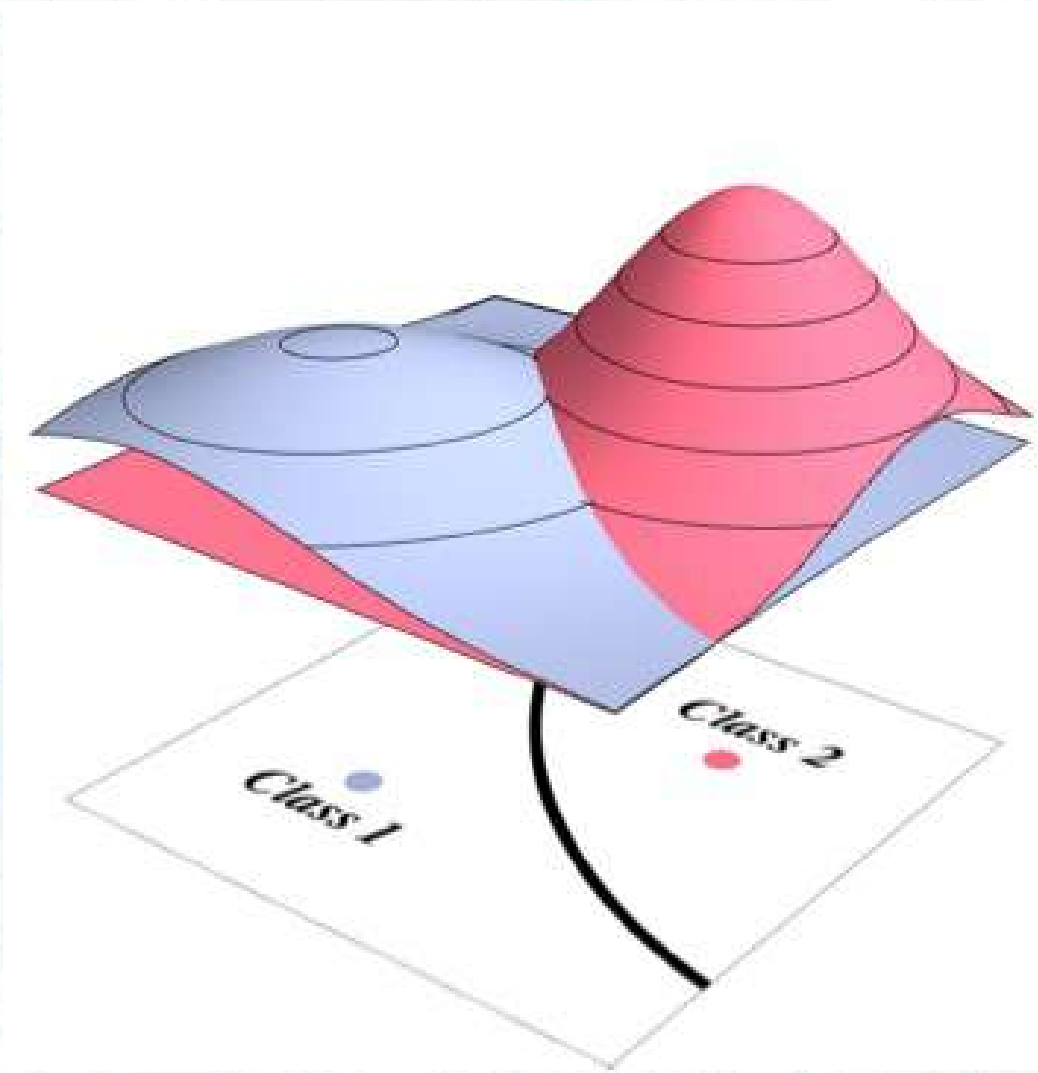


Пример необычной метрики 2

Задача сравнения генных последовательностей



Машинное обучение
Лекция 4. Байесовский подход



https://yandexdataschool.ru/edu-process/courses/machine-learning

Содержание лекции

- Байесовский классификатор
- Восстановление плотности распределения
  - непараметрическое
  - параметрическое

Вероятностная постановка задачи

- P(x,y) – неизвестная точная плотность распределения на X×Y
- X_ℓ - выборка из случайных, независимых и одинаково распределенных прецедентов
- Найти: эмпирическую оценку плотности
- Классификатор с минимальной вероятностью ошибки:

$$a(x) = \arg\max_{y \in Y} P(y|x) = \arg\max_{y \in Y} P(y)p(x|y)$$

- Классификатор с минимальным средним риском: $a(x) = \arg\min_y E_s \, \mathscr{L}(s, y)$

Пример

y	+	o
p(y|x)	0.3	0.7
- Классификатор с минимальной вероятностью ошибки: a(x) = o
- Классификатор с минимальным средним риском (пусть $\mathcal{L}$(+,o)=3, $\mathcal{L}$(o,+)=1):
  a(x) = ???

Пример

y	+	o
p(y|x)	0.3	0.7
- Классификатор с минимальной вероятностью ошибки: a(x) = o
- Классификатор с минимальным средним риском (пусть $\mathcal{L}$(+,o)=3, $\mathcal{L}$(o,+)=1):

  a(x) = +

Подходы к восстановлению плотности распределения

- Непараметрическое оценивание плотности:

$$\hat{p}(x) = \sum_{i=1}^{\ell} w_i K\left(\frac{\rho(x, x_i)}{h}\right)$$

- Параметрическое оценивание плотности:

$$\hat{p}(x) = \varphi(x, \theta)$$

Наивный байесовский классификатор

- Восстановление n одномерных плотностей — намного более простая задача, чем одной n-мерной.
- Допущение (наивное): признаки являются независимыми случайными величинами
- Тогда совместная плотность распределения представима в виде произведения частных плотностей:

$$p(x|y) = p_1(\xi_1|y) \cdots p_n(\xi_n|y), \quad x = (\xi_1, \ldots, \xi_n), \quad y \in Y.$$

Непараметрическое оценивание

- Определение плотности вероятности (одномерный случай):

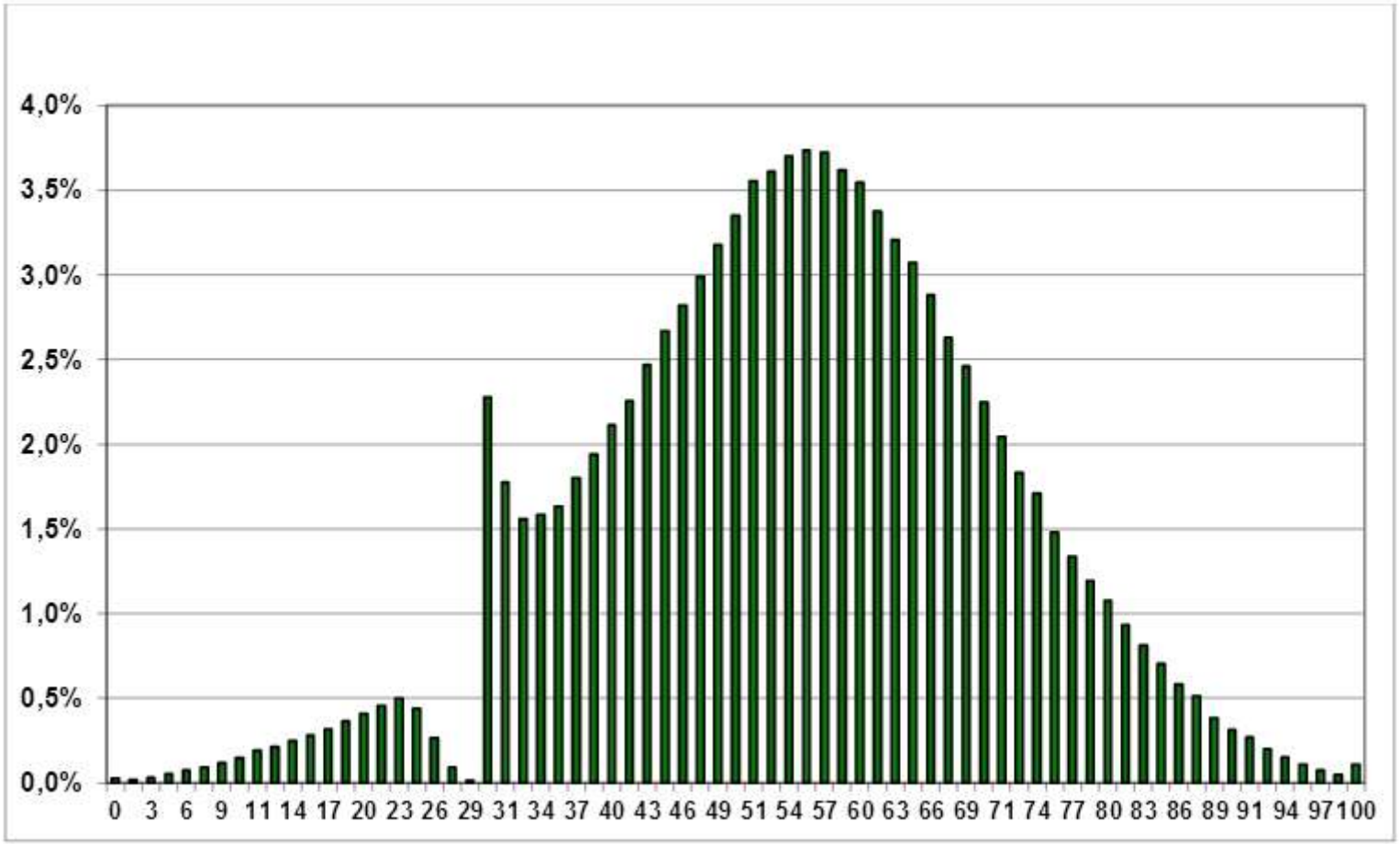
$$p(x) = \lim_{h \to 0} \frac{1}{2h} P[x - h, x + h]$$

- Эмпирическая оценка:

$$\hat{p}_h(x) = \frac{1}{2h} \frac{1}{\ell} \sum_{i=1}^{\ell} [|x - x_i| < h]$$

Пример – гистограмма оценок

2.1. Poziom podstawowy



Пример – гистограмма возрастов (Россия 2012г)

Локальная непараметрическая оценка Парзена-Розенблатта

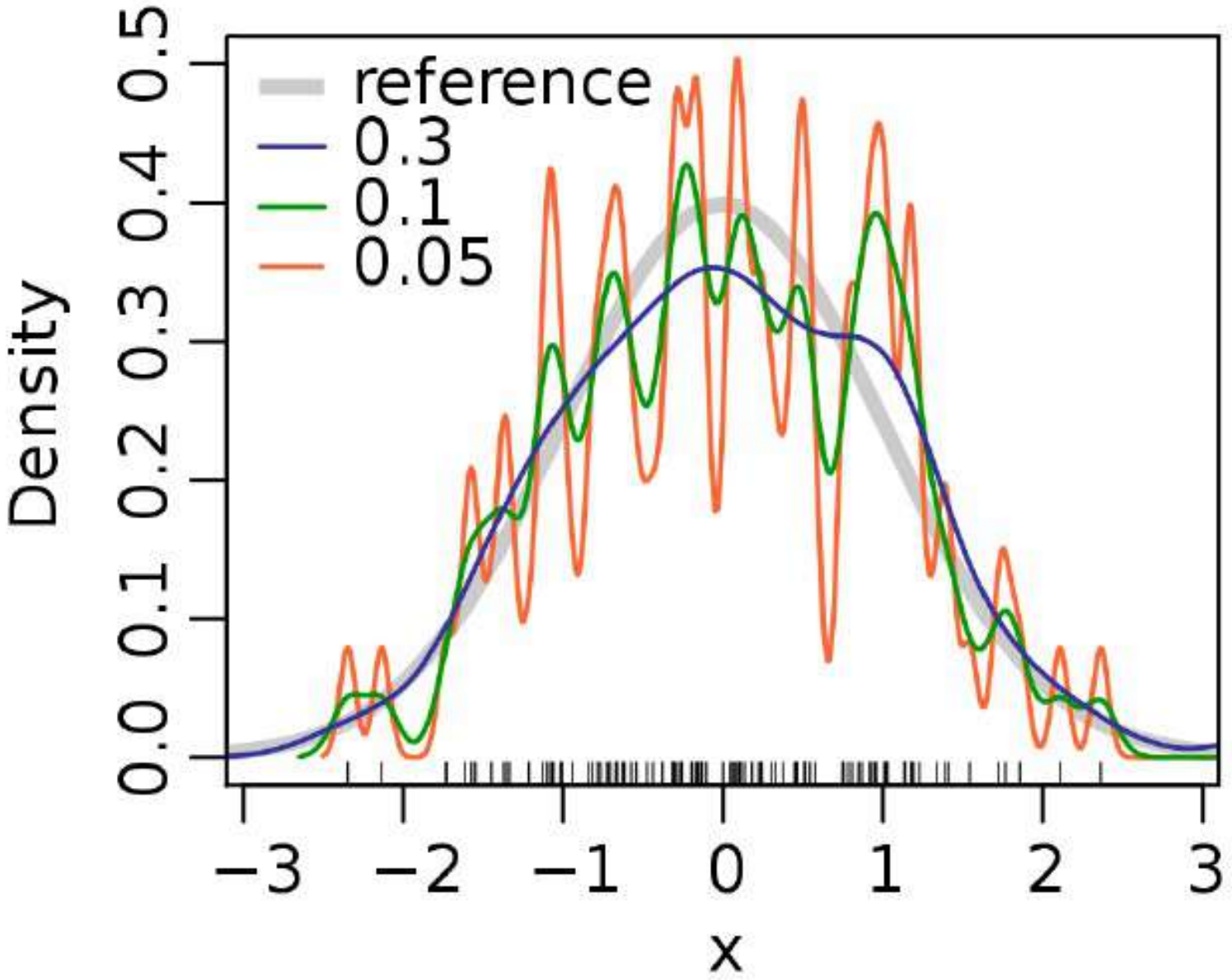
$$\hat{p}_h(x) = \frac{1}{\ell\, h} \sum_{i=1}^{\ell} K\left(\frac{x - x_i}{h}\right)$$

K(z) — функция, называемая ядром, чётная и нормированная:

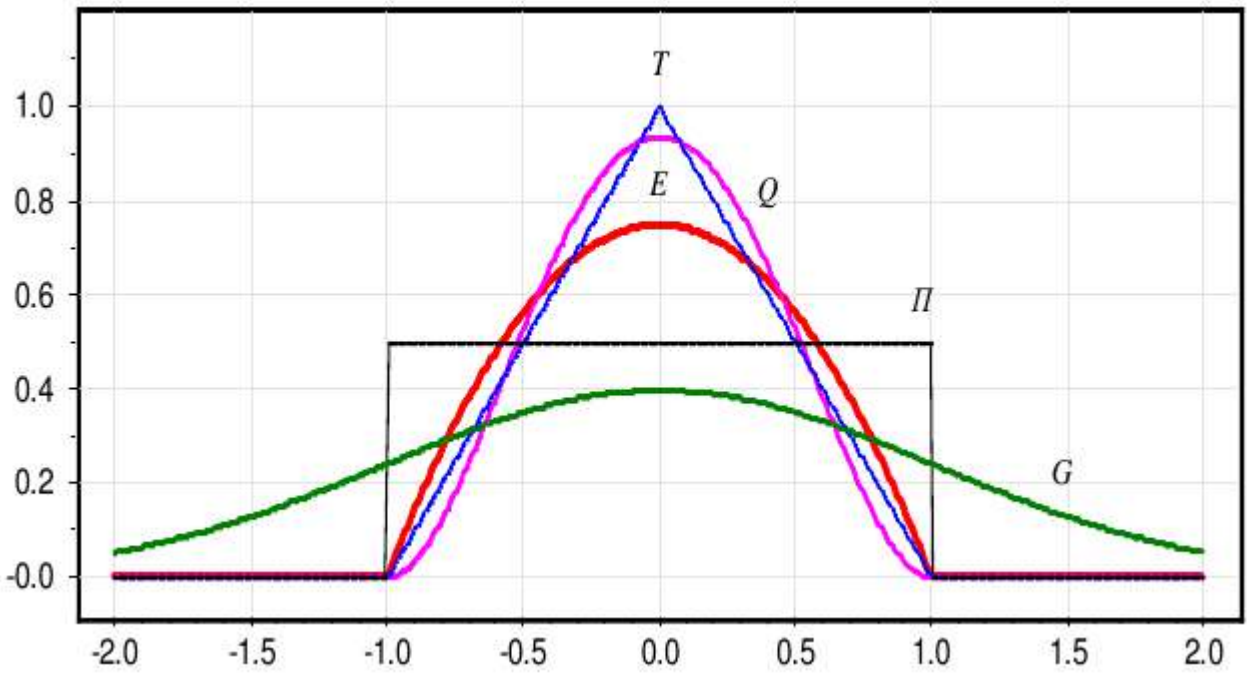
$$\int K(z)\, dz = 1$$

$\hat{p}_h$ сходится к p при $h \to 0$, $\ell \to \infty$, $h\ell \to \infty$

Зависимость от h



Выбор ядра



$E(r) = \frac{3}{4}(1 - r^2)\big[|r| \leqslant 1\big]$ — оптимальное (Епанечникова);

$Q(r) = \frac{15}{16}(1 - r^2)^2\big[|r| \leqslant 1\big]$ — квартическое;

$T(r) = \big(1 - |r|\big)\big[|r| \leqslant 1\big]$ — треугольное;

$G(r) = (2\pi)^{-1/2}\exp(-\frac{1}{2}r^2)$ — гауссовское;

$\Pi(r) = \frac{1}{2}\big[|r| \leqslant 1\big]$ — прямоугольное.

Выбор ядра

Функционал качества восстановления плотности:

$$J(K) = \int_{-\infty}^{+\infty} \mathsf{E}\big(\hat{p}_h(x) - p(x)\big)^2\, dx$$

ядро $K(r)$	степень гладкости	$J(K^*)/J(K)$
Епанечникова $K^*(r)$	$\hat{p}'_h$ разрывна	1.000
Квартическое	$\hat{p}''_h$ разрывна	0.995
Треугольное	$\hat{p}'_h$ разрывна	0.989
Гауссовское	∞ дифференцируема	0.961
Прямоугольное	$\hat{p}_h$ разрывна	0.943

В таблице представлены асимптотические значения отношения J(K*)/J(K) при m → ∞, причём это отношение не зависит от p(x).

Параметрическое оценивание плотности

$$p(x) = \varphi(x;\theta)$$

- Принцип максимума правдоподобия:

$$L(\theta; X^\ell) = \sum_{i=1}^{\ell} \ln \varphi(x_i;\theta) \to \max_\theta$$

- Необходимое условие оптимума:

$$\frac{\partial}{\partial\theta} L(\theta; X^\ell) = \sum_{i=1}^{\ell} \frac{\partial}{\partial\theta} \ln \varphi(x_i;\theta) = 0$$

Многомерное нормальное распределение

$$a(x) = \arg\max_{y \in Y} \mathrm{P}(y|x) = \arg\max_{y \in Y} P(y)p(x|y)$$

$$p(x|y) = \mathcal{N}(x; \mu_y, \Sigma_y) = \frac{e^{-\frac{1}{2}(x-\mu_y)^\intercal \Sigma_y^{-1}(x-\mu_y)}}{\sqrt{(2\pi)^n \det \Sigma_y}}$$

где $\mu_y \in \mathbb{R}^n$ — вектор матожидания (центр) класса $y \in Y$
$\Sigma_y \in \mathbb{R}^{n \times n}$ — ковариационная матрица класса $y \in Y$

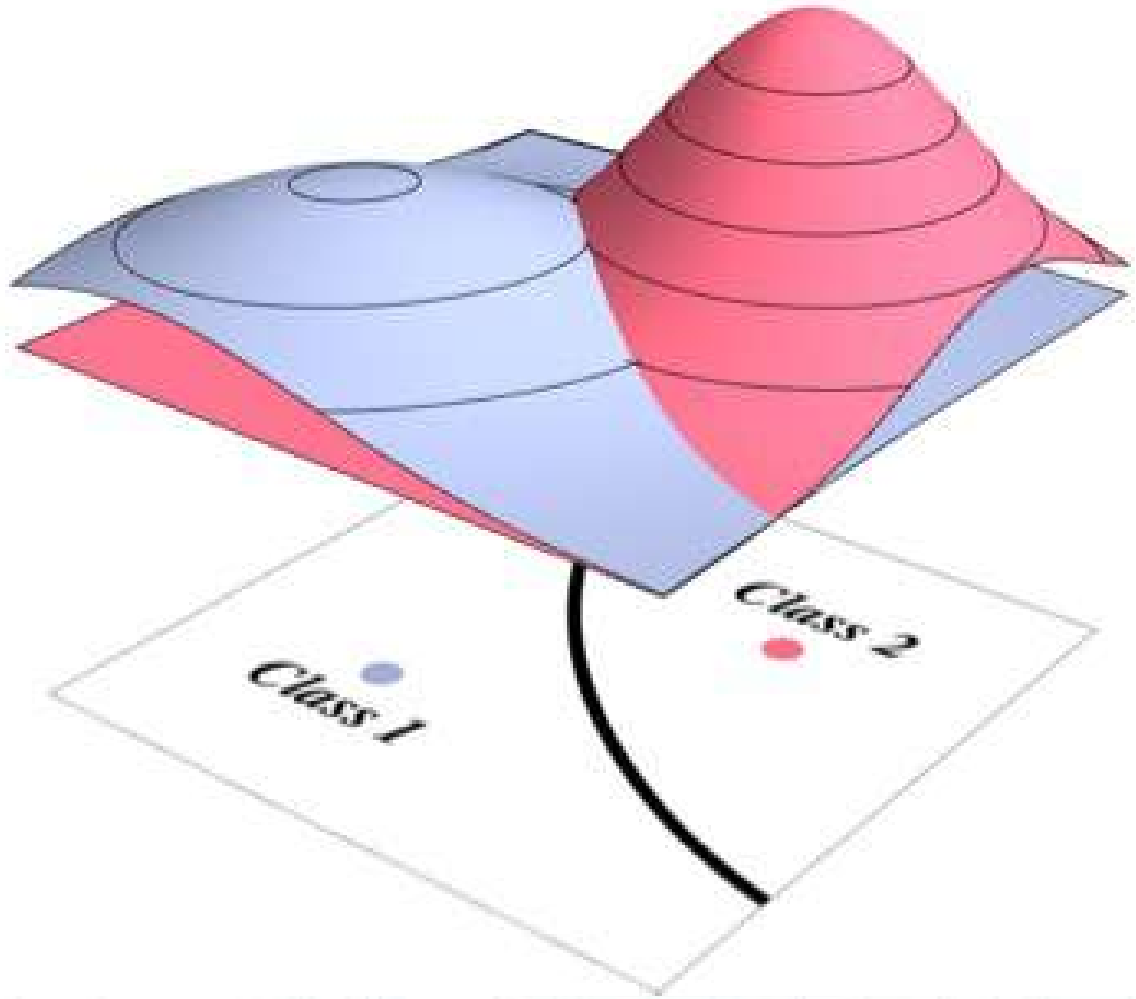
Принцип максимума правдоподобия:

$$L(\theta, X^\ell) = \prod_{i=1}^{\ell} \varphi(x_i, y_i, \theta) \to \max_{\theta}$$

Решение – подстановочный алгоритм:

$$\hat{\mu} = \frac{1}{\ell} \sum_{i=1}^{\ell} x_i; \qquad \hat{\Sigma} = \frac{1}{\ell} \sum_{i=1}^{\ell} (x_i - \hat{\mu})(x_i - \hat{\mu})^\intercal$$

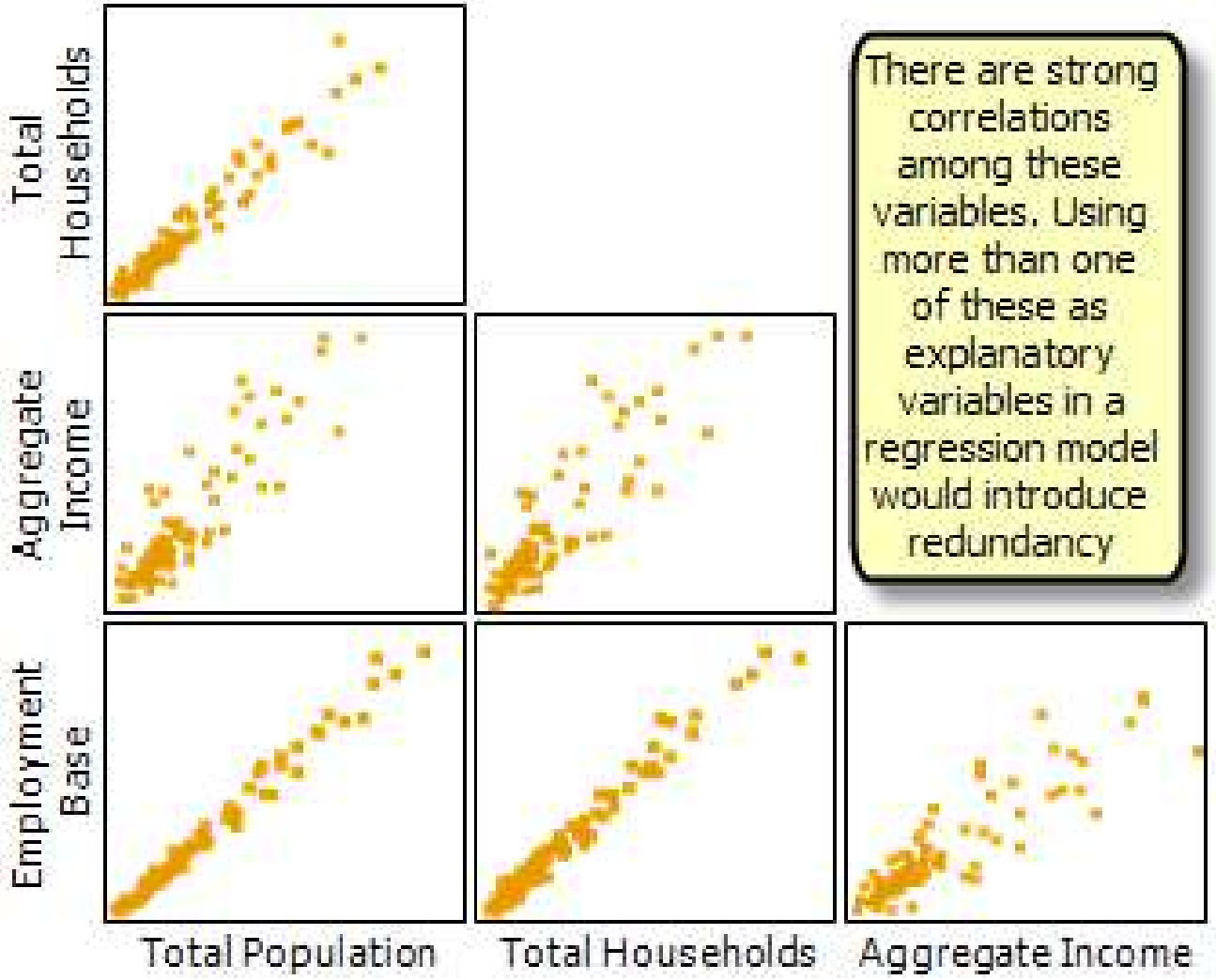
Многомерное нормальное распределение



Недостатки подстановочного алгоритма

- Функции правдоподобия классов могут существенно отличаться от гауссовских.
- Проблема мультиколлинеарности: на практике встречаются задачи, в которых признаки «почти линейно зависимы». Тогда матрица Σ_y^{-1} является плохо обусловленной. Она может непредсказуемо и сильно изменяться при незначительных вариациях исходных данных.
- Выборочные оценки чувствительны к нарушениям нормальности распределений, в частности, к редким большим выбросам.

Мультиколлинеарность признаков



Методы устранения мультиколлинеарности

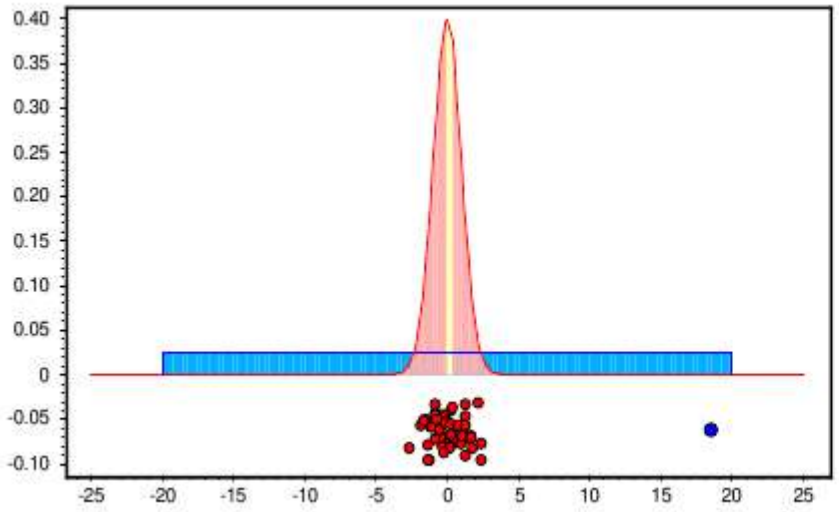
- Регуляризация ковариационной матрицы: обращение $\Sigma+\tau$ вместо Σ
- Диагонализация ковариационной матрицы - нормальный наивный байесовский классификатор:

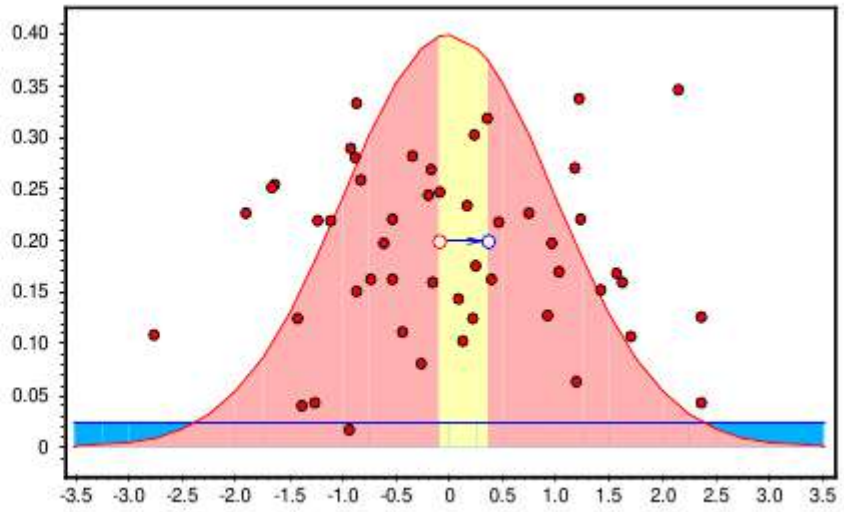
$$p_j(\xi|y) = \frac{1}{\sqrt{2\pi}\sigma_{yj}} \exp\left(-\frac{(\xi-\mu_{yj})^2}{2\sigma_{yj}^2}\right)$$

$$p_y(x) = p_{y1}(\xi_1)\cdots p_{yn}(\xi_n)$$

Проблема выбросов

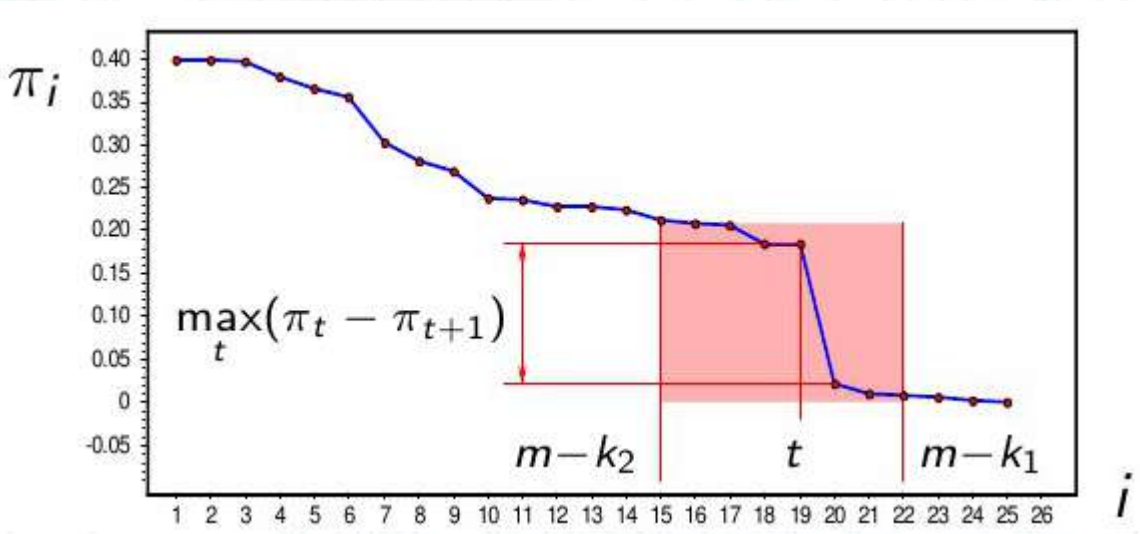
- Эмпирическое среднее является оценкой матожидания, неустойчивой к редким большим выбросам.



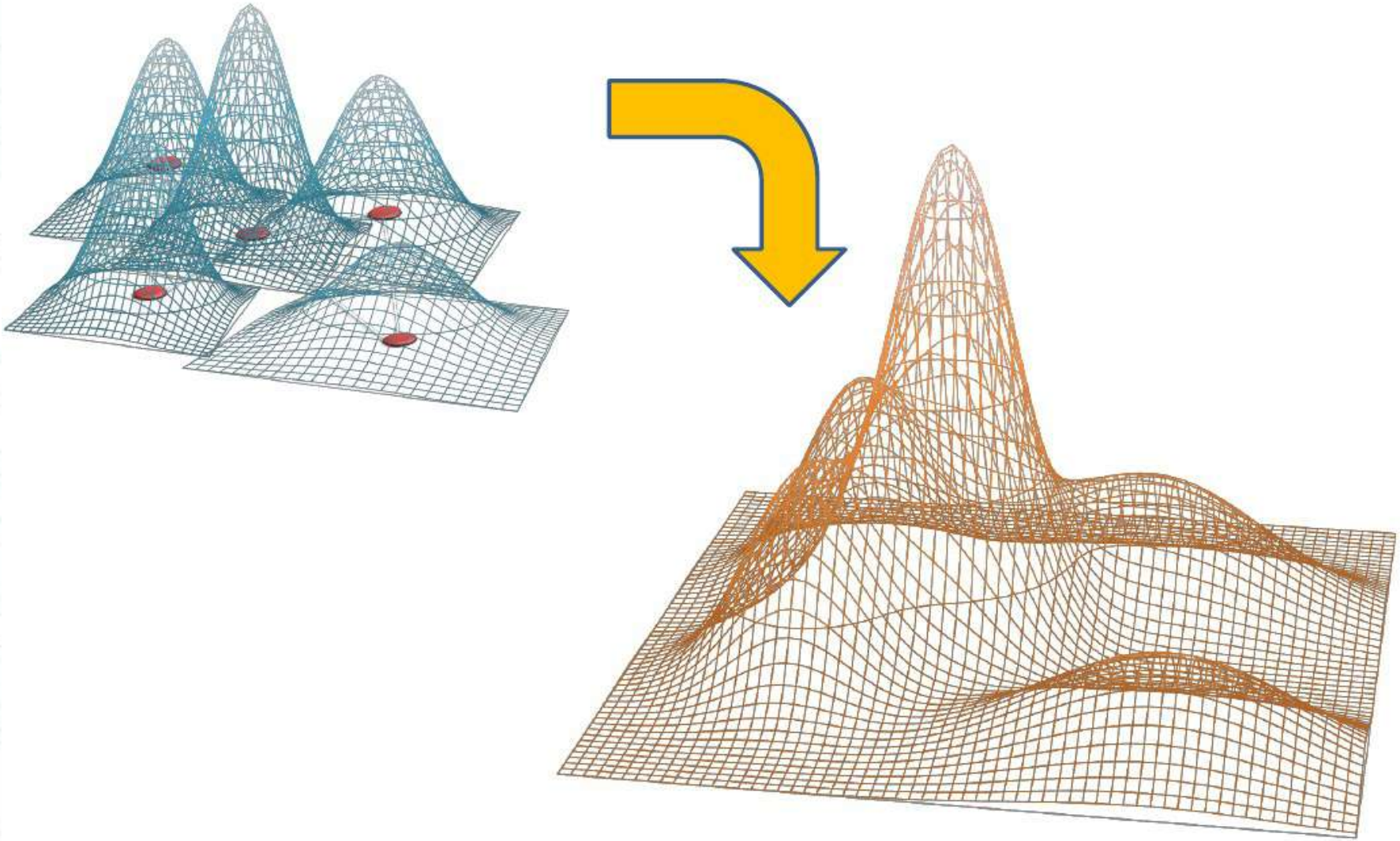


Отсев выбросов

- Идея: решать задачу два раза
  - В первый – найти и исключить выбросы
  - Во второй – построить более точное решение по выборке без выбросов
- Критерий крутого склона:



Машинное обучение
Лекция 5. Логистическая регрессия. Смеси распределений



https://yandexdataschool.ru/edu-process/courses/machine-learning

Содержание лекции

- Логистическая регрессия
- Бинаризация признаков
- Скоринг
- Смеси распределений
- EM-алгоритм восстановления смеси

Предположение 1

- $X=R^n$, $Y=\{+1,-1\}$, X^ℓ
- Распределение p(x|y) из экспоненциального семейства:

$$p(x|y)=\exp\left(c_y(\delta)\langle\theta_y,x\rangle+b_y(\delta,\theta_y)+d(x,\delta) \right)$$

$\theta_y \in R^n$ – параметр сдвига

δ – параметр разброса

b_y, c_y, d – произвольные числовые функции

- Экспоненциальное семейство распределений широко: равномерное, нормальное, Лапласа, Пуассона, Парето, Дирихле, биномиальное, Г-распределение, χ^2-распределение, и др.

Предположение 2

- Плотности p(x|y) имеют равные значения параметров c, d и δ, но отличаются значениями параметра сдвига θ_y.

Теорема

Если выполняются предположения 1 и 2 и среди признаков есть константа, то

- оптимальный байесовский классификатор для заданных штрафов λ_+ и λ_- является линейным:

$$a(x) = \text{sign}\big(\langle w, x\rangle - w_0\big)$$

- апостериорные вероятности классов вычисляются по формуле: $\mathsf{P}(y|x) = \sigma\big(\langle w, x\rangle\, y\big)$

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

Доказательство

$$a(x) = 1 \quad \Leftrightarrow \quad \lambda_+ P(+1|x) > \lambda_- P(-1|x) \quad \Leftrightarrow \quad \frac{P(+1|x)}{P(-1|x)} > \frac{\lambda_-}{\lambda_+}$$

$$\Leftrightarrow \quad \frac{P(x|+1)P(+1)}{P(x|-1)P(-1)} > \frac{\lambda_-}{\lambda_+} \quad \Leftrightarrow \quad \ln \frac{P(x|+1)P(+1)}{P(x|-1)P(-1)} > \ln \frac{\lambda_-}{\lambda_+}$$

подставим сюда $p(x|\pm 1) = \exp\left(c_\pm(\delta)\langle\theta_\pm, x\rangle + b_\pm(\delta, \theta_\pm) + d(x, \delta)\right)$

$$\ln \frac{P(+1|x)}{P(-1|x)} = \underbrace{\langle c(\delta)(\theta_+ - \theta_-), x\rangle}_{w=\text{const}(x)} + \underbrace{b_+(\delta, \theta_+) - b_-(\delta, \theta_-) + \ln \frac{P_+}{P_-}}_{\beta=\text{const}(x)}$$

Добавим β к коэффициенту w_j при константном признаке $f_j = 1$

Доказательство

Получим:

$$\frac{\mathrm{P}(+1|x)}{\mathrm{P}(-1|x)} = e^{\langle w,x\rangle}$$

По формуле полной вероятности $\mathrm{P}(-1|x) + \mathrm{P}(+1|x) = 1$,

следовательно

$$\mathrm{P}(+1|x) = \frac{1}{1+e^{-\langle w,x\rangle}}; \qquad \mathrm{P}(-1|x) = \frac{1}{1+e^{\langle w,x\rangle}}$$

$$\mathrm{P}(y|x) = \frac{1}{1+e^{-\langle w,x\rangle y}} = \sigma\big(\langle w,x\rangle y\big)$$

Разделяющая классы поверхность – линейна:

$$\lambda_- \mathrm{P}(-1|x) = \lambda_+ \mathrm{P}(+1|x),$$

$$\langle w,x\rangle - \ln\frac{\lambda_-}{\lambda_+} = 0.$$

Поиск w

- Максимизация логарифма правдоподобия обучающей выборки:

$$\ln \prod_{i=1}^{\ell} p(x_i, y_i) = \sum_{i=1}^{\ell} \ln p(x_i, y_i) \to \max_{\mathrm{w}}$$

- Для логистического распределения:

$$p(x, y) = p(y|x)p(x) = \sigma(\langle \mathrm{w}, x \rangle y)p(x)$$ отсюда

$$\sum_{i=1}^{\ell} \ln \left(1 + e^{-\langle \mathrm{w}, x_i \rangle y_i}\right) + \mathrm{Const}(\mathrm{w}) \to \min_{\mathrm{w}}$$

- Напоминает минимизацию функционала эмпирического риска

$$Q(a, X^{\ell}) = \frac{1}{\ell} \sum_{i=1}^{\ell} \mathscr{L}(a, x_i)$$

Сравнение с другими видами функционала эмпирического риска

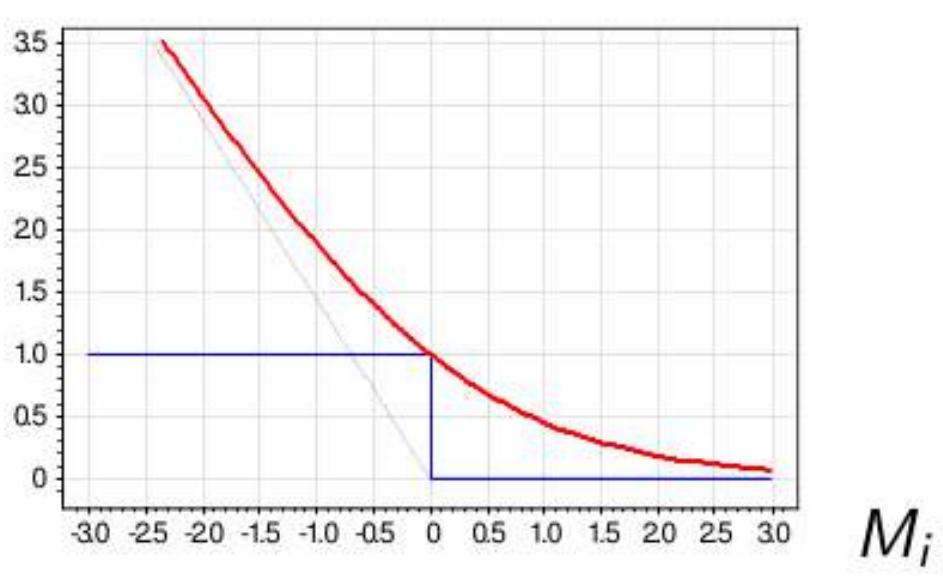
- Определим выступ объекта как:

$$M_i = \langle \mathrm{w}, x_i \rangle y_i$$

- В случае логистической регрессии:

$$\mathscr{L}(M) = \ln\left(1 + e^{-M}\right)$$

- Сравним:



Поиск w

Метод первого порядка — стохастический градиент:

$$w^{(t+1)} := w^{(t)} + \eta_t\, y_i x_i\,(1 - \sigma_i),$$

η_t — градиентный шаг,
$\sigma_i = \sigma(y_i w^{\mathsf{T}} x_i) = \mathsf{P}(y_i|x_i)$ — вероятность правильной классификации x_i.

Метод второго порядка (Ньютона-Рафсона) приводит к IRLS, Iteratively Reweighted Least Squares:

$$w^{(t+1)} := w^{(t)} + \eta_t (F^{\mathsf{T}} \Lambda F)^{-1} F^{\mathsf{T}} \tilde{y},$$

F — матрица объекты–признаки $\ell \times n$,
$\tilde{y} = \big(y_i(1 - \sigma_i)\big)$,
$\Lambda = \operatorname{diag}\big((1 - \sigma_i)/\sigma_i\big)$,

Бинаризация признаков (One Hot encoding)

- Пусть x - единственный признак (номинальный, закодированный: 0,1,2,...k)
- Классификатор: $a(x) = \text{sign}(wx+w_0)$
- Проблема: вес w нельзя подобрать так, чтобы классификатор был не монотонным.
- Для любых w и w0 значения $a(x) > 0$ когда $x > w_0/w$ и $a(x) \leq 0$ в противном случае

Бинаризация признаков (One Hot encoding)

- Вместо одного номинального признака вводим k бинарных признаков. Пример (k=5):

	x1	x2	x3	x4	x5
Азов	0	0	0	0	1
Аксай	0	0	0	1	0
Ростов	0	0	1	0	0
Новочеркасск	0	1	0	0	0
Таганрог	1	0	0	0	0

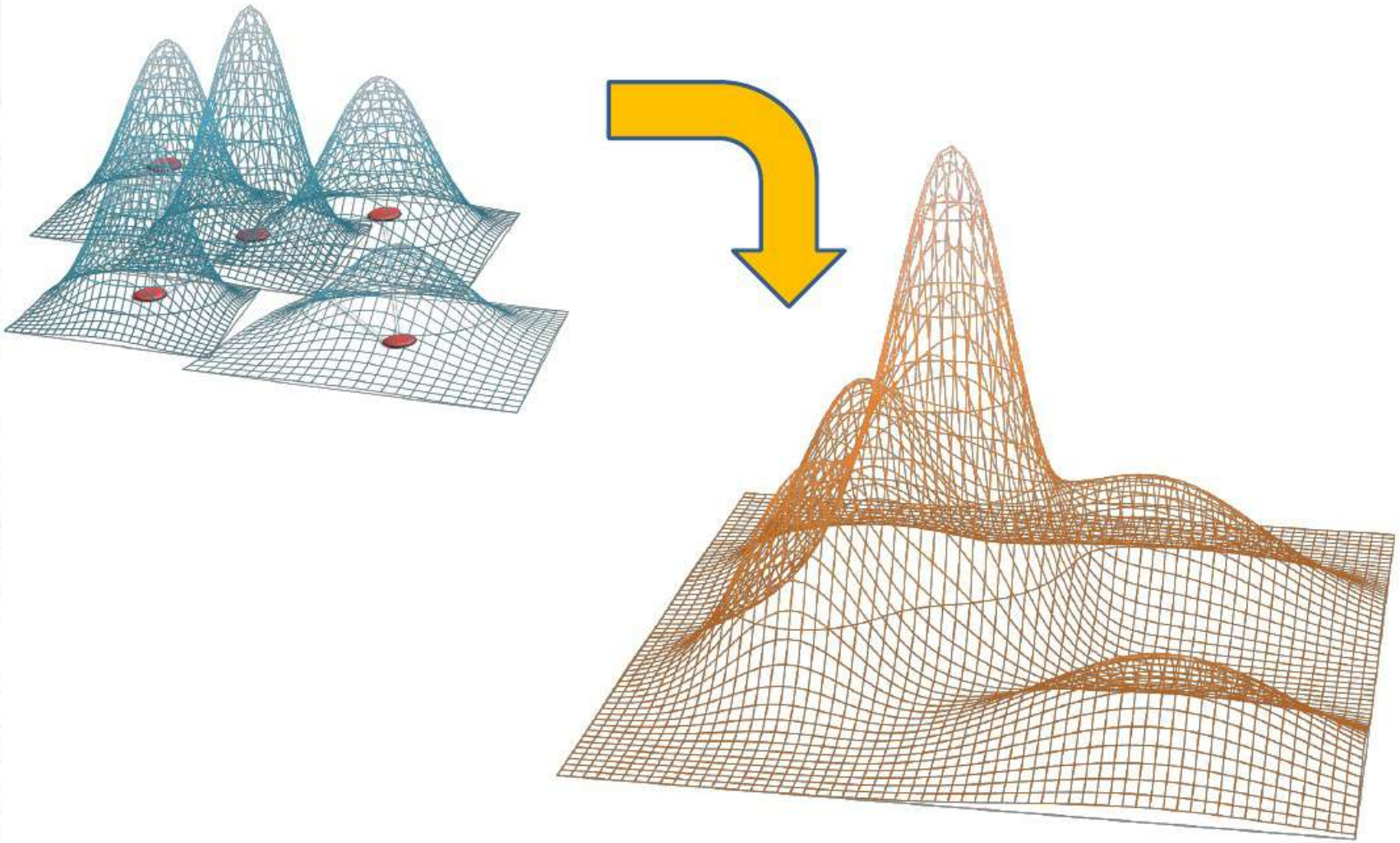
- Возможна бинаризация и количественных признаков путем предварительной дискретизации

Скоринг

- Если все признаки – бинарные, то линейный классификатор удобно рассматривать как суммирование баллов (score): Sum += w_j, если x_j=1
- Рисунок – фрагмент скоринговой карты для вопроса о выдаче кредита

Возраст	до 25	5
	25 - 40	10
	40 - 50	15
	50 и больше	10
Собственность	владелец	20
	совладелец	15
	съемщик	10
	другое	5
Работа	руководитель	15
	менеджер среднего звена	10
	служащий	5
	другое	0
Стаж	1/безработный	0
	1..3	5
	3..10	10
	10 и больше	15
Работа_мужа /жены	нет/домохозяйка	0
	руководитель	10
	менеджер среднего звена	5
	служащий	1

Смеси распределений



$$p(x) = \sum_{j=1}^{k} w_j p_j(x;\theta_j), \qquad \sum_{j=1}^{k} w_j = 1, \qquad w_j \geqslant 0$$

Смеси распределений

Задача 1: имея простую выборку $X^m \sim p(x)$ и зная k, оценить вектор параметров $\Theta = (w_1, \ldots, w_k, \theta_1, \ldots, \theta_k)$.

Задача 2: оценить ещё и k.

Задача максимизации логарифма правдоподобия

$$L(\Theta) = \ln \prod_{i=1}^{m} p(x_i) = \sum_{i=1}^{m} \ln \sum_{j=1}^{k} w_j p_j(x_i; \theta_j) \to \max_{\Theta}$$

при ограничениях $\sum_{j=1}^{k} w_j = 1; \;\; w_j \geqslant 0.$

Решение оптимизационной задачи

$$Q(\Theta) = \ln \prod_{i=1}^{m} p(x_i) = \sum_{i=1}^{m} \ln \sum_{j=1}^{k} w_j p_j(x_i) \to \max_{\Theta} \qquad \sum_{j=1}^{k} w_j = 1$$

$$L(\Theta; X^m) = \sum_{i=1}^{m} \ln \left(\sum_{j=1}^{k} w_j p_j(x_i) \right) - \lambda \left(\sum_{j=1}^{k} w_j - 1 \right)$$

$$\frac{\partial L}{\partial w_j} = \sum_{i=1}^{m} \frac{p_j(x_i)}{\sum_{s=1}^{k} w_s p_s(x_i)} - \lambda = 0, \quad j = 1, \ldots, k$$

Умножим левую и правую части на w_j, просуммируем все k этих равенств, и поменяем местами знаки суммирования по j и по i:

$$\sum_{i=1}^{m} \underbrace{\sum_{j=1}^{k} \frac{w_j p_j(x_i)}{\sum_{s=1}^{k} w_s p_s(x_i)}}_{=1} = \lambda \underbrace{\sum_{j=1}^{k} w_j}_{=1}, \quad \Rightarrow \quad \lambda = m$$

Решение оптимизационной задачи

$$w_j = \frac{1}{m}\sum_{i=1}^{m}\frac{w_j p_j(x_i)}{\sum_{s=1}^{k} w_s p_s(x_i)} = \frac{1}{m}\sum_{i=1}^{m} g_{ij}, \quad j = 1, \ldots, k$$

где $g_{ij} = \dfrac{w_j p_j(x_i)}{\sum_{s=1}^{k} w_s p_s(x_i)}$

“похожи” на вероятности того, что x_i попал в j-тое скопление смеси:

$$g_{ij} = \mathsf{P}(j|x_i) = \frac{P(j)p(x_i|j)}{p(x_i)} = \frac{w_j p_j(x_i;\theta_j)}{p(x_i)} = \frac{w_j p_j(x_i;\theta_j)}{\sum_{s=1}^{k} w_s p_s(x_i;\theta_s)}$$

$$\sum_{j=1}^{k} g_{ij} = 1$$

Решение оптимизационной задачи

Приравняем к нулю производную лагранжиана по θ_j, помня, что $p_j(x) = \varphi(x; \theta_j)$:

$$\frac{\partial L}{\partial \theta_j} = \sum_{i=1}^{m} \frac{w_j}{\sum_{s=1}^{k} w_s p_s(x_i)} \frac{\partial}{\partial \theta_j} p_j(x_i) = \sum_{i=1}^{m} \frac{w_j p_j(x_i)}{\sum_{s=1}^{k} w_s p_s(x_i)} \frac{\partial}{\partial \theta_j} \ln p_j(x_i) =$$

$$= \sum_{i=1}^{m} g_{ij} \frac{\partial}{\partial \theta_j} \ln p_j(x_i) = \frac{\partial}{\partial \theta_j} \sum_{i=1}^{m} g_{ij} \ln p_j(x_i) = 0, \quad j = 1, \ldots, k.$$

Полученное условие совпадает с необходимым условием максимума в задаче максимизации взвешенного правдоподобия:

$$\theta_j := \arg\max_{\theta} \sum_{i=1}^{m} g_{ij} \ln \varphi(x_i; \theta), \quad j = 1, \ldots, k.$$

при условии, что g_{ij} не зависят от θ. Что, конечно же, не так.

EM-алгоритм

Итерационный алгоритм Expectation–Maximization:

1: начальное приближение вектора параметров Θ;
2: **повторять**
3: $G :=$ E-шаг (Θ); // оцениваются *скрытые переменные* G
4: $\Theta :=$ M-шаг (Θ, G);
5: **пока** Θ и G не стабилизируются.

EM-алгоритм

Вход: $X^m = \{x_1, \ldots, x_m\}$, k, δ, начальное $\Theta = (w_j, \theta_j)_{j=1}^k$;

Выход: $\Theta = (w_j, \theta_j)_{j=1}^k$ — параметры смеси распределений

1: **повторять**

2: E-шаг (expectation):

для всех $i = 1, \ldots, m$, $j = 1, \ldots, k$

$$g_{ij}^0 := g_{ij}; \quad g_{ij} := \frac{w_j p_j(x_i; \theta_j)}{\sum_{s=1}^k w_s p_s(x_i; \theta_s)};$$

3: M-шаг (maximization):

для всех $j = 1, \ldots, k$

$$\theta_j := \arg\max_\theta \sum_{i=1}^m g_{ij} \ln p_j(x_i; \theta); \quad w_j := \frac{1}{m} \sum_{i=1}^m g_{ij};$$

4: **пока** $\max_{i,j} |g_{ij} - g_{ij}^0| > \delta$;

5: **вернуть** $(w_j, \theta_j)_{j=1}^k$;

Смеси гауссовских распределений

$$p(x|y) = \sum_{j=1}^{k_y} w_{yj} p_{yj}(x), \quad p_{yj}(x) = \mathcal{N}(x; \mu_{yj}, \Sigma_{yj})$$

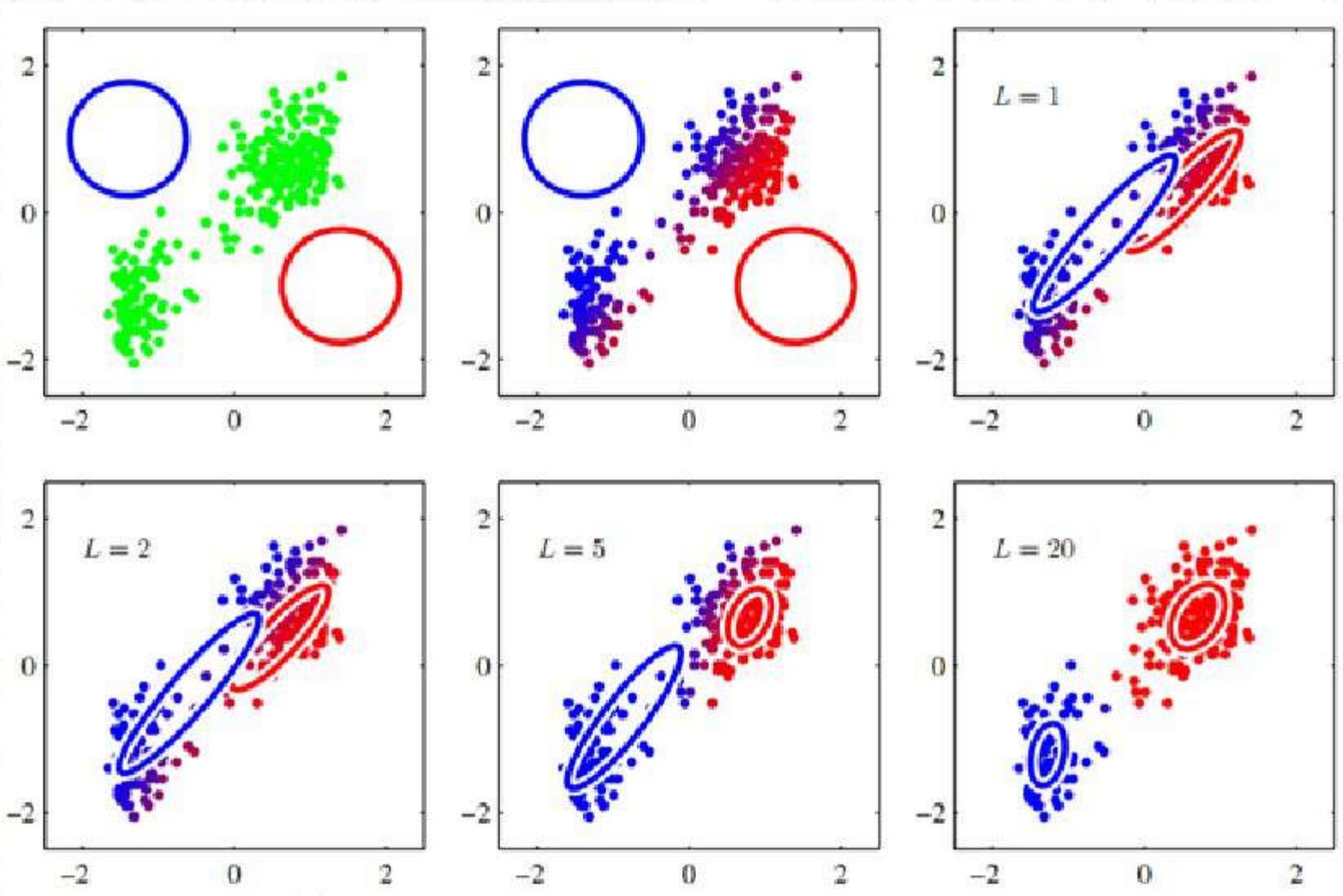
Решение М-шага:

$$\hat{\mu}_j = \frac{1}{m w_j} \sum_{i=1}^{m} g_{ij} x_i, \quad j = 1, \ldots, k;$$

$$\hat{\Sigma}_j = \frac{1}{m w_j} \sum_{i=1}^{m} g_{ij} (x_i - \hat{\mu}_j)(x_i - \hat{\mu}_j)^\intercal, \quad j = 1, \ldots, k.$$

Пример работы алгоритма

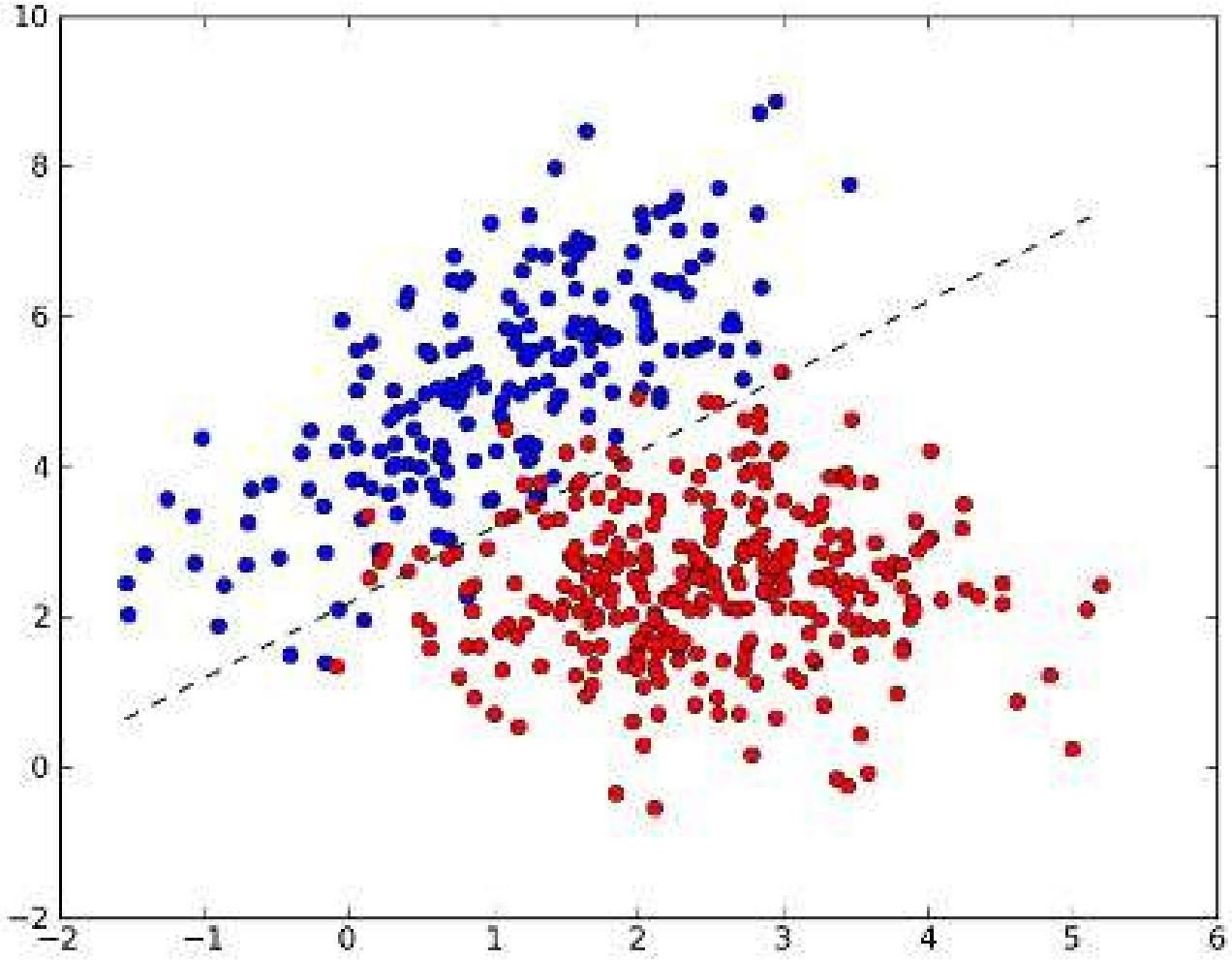
- Две гауссовские компоненты k = 2 в R^2.
- Расположение компонент в зависимости от номера итерации:



EM-алгоритм с добавлением и удалением компонент

- Проблемы базового варианта EM-алгоритма:
  - Как выбирать начальное приближение?
  - Как определять число компонент?
  - Как ускорить сходимость?
- Добавление и удаление компонент в EM-алгоритме:
  - Если слишком много объектов x_i имеют слишком низкие правдоподобия $p(x_i)$, то создаём новую k+1-ю компоненту, по этим объектам строим её начальное приближение.
  - Если у j-й компоненты слишком низкий w_j, удаляем её.

Машинное обучение
Лекция 6. Линейные алгоритмы классификации



https://yandexdataschool.ru/edu-process/courses/machine-learning

Содержание лекции

- Общая формула линейного классификатора
- Метод стохастического градиента
- Частные случаи
- Обоснование метода СГ
- Выступ объекта для лин. классификатора
- ROC и AUC

Классификация линейной функцией

Обучающая выборка: $X^\ell = (x_i, y_i)_{i=1}^\ell, \quad x_i \in \mathbb{R}^n, \quad y_i \in \{-1, +1\}$

1. Модель классификации — *линейная*:

$$a(x, w) = \text{sign}\langle x, w\rangle$$

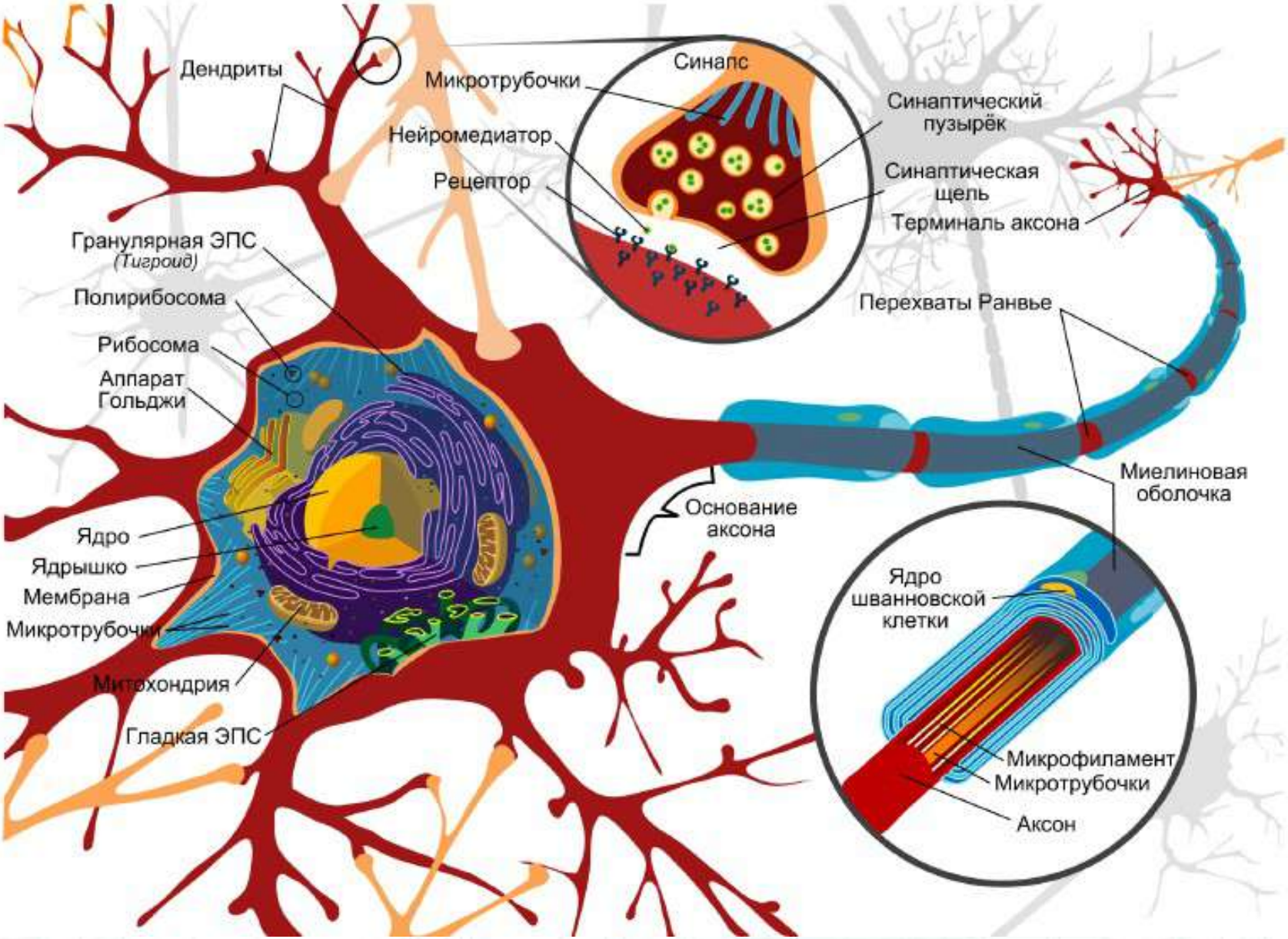
2. Функция потерь — бинарная или её аппроксимация:

$$\mathscr{L}(a, y) = \big[\langle x_i, w\rangle y_i < 0\big] \leqslant \mathscr{L}\big(\langle x_i, w\rangle y_i\big)$$

3. Метод обучения — *минимизация эмпирического риска*:

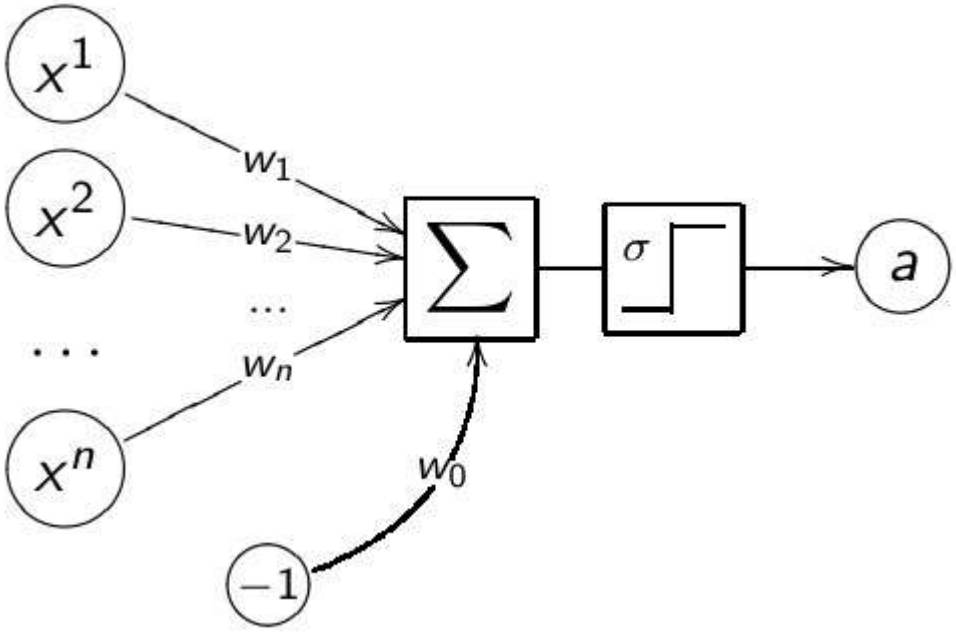
$$Q(w) = \sum_{i=1}^{\ell} \big[a(x_i, w) y_i < 0\big] \leqslant \sum_{i=1}^{\ell} \mathscr{L}\big(\langle x_i, w\rangle y_i\big) \to \min_w$$

Линейный классификатор – математическая модель нейрона



Линейная модель нейрона МакКаллока-Питтса (1943)

$$a(x, w) = \sigma(\langle w, x \rangle) = \sigma\left(\sum_{j=1}^{n} w_j f_j(x) - w_0 \right)$$



Градиентный метод численной минимизации

Минимизация эмпирического риска:

$$Q(w) = \sum_{i=1}^{\ell} \mathscr{L}\big(g(w, x_i), y_i\big) = \sum_{i=1}^{\ell} \mathscr{L}_i(w) \to \min_w .$$

Численная минимизация методом *градиентного спуска*:

$$w^{(0)} := \text{начальное приближение;}$$

$$w^{(t+1)} := w^{(t)} - h \cdot \nabla Q(w^{(t)}), \qquad \nabla Q(w) = \left(\frac{\partial Q(w)}{\partial w_j}\right)_{j=0}^{n},$$

где h — *градиентный шаг*, называемый также *темпом обучения*.

$$w^{(t+1)} := w^{(t)} - h \sum_{i=1}^{\ell} \nabla \mathscr{L}_i\big(w^{(t)}\big).$$

Идея ускорения сходимости:

брать (x_i, y_i) по одному и сразу обновлять вектор весов.

Метод стохастического градиента

Вход: выборка X^ℓ, темп обучения h, темп забывания λ

Выход: вектор весов w

1: инициализировать веса w_j, $j = 0, \ldots, n$;

2: инициализировать оценку функционала: $\bar{Q} := \frac{1}{\ell} \sum_{i=1}^{\ell} \mathscr{L}_i(w)$;

3: **повторять**

4: выбрать объект x_i из X^ℓ случайным образом;

5: вычислить потерю: $\varepsilon_i := \mathscr{L}_i(w)$;

6: сделать градиентный шаг: $w := w - h\nabla \mathscr{L}_i(w)$;

7: оценить функционал: $\bar{Q} := (1 - \lambda)\bar{Q} + \lambda\varepsilon_i$;

8: **пока** значение $\bar{Q}$ и/или веса w не сойдутся;

Пересчет функционала

Проблема: после каждого шаг w по одному объекту x_i, не хотелось бы оценивать Q по всей выборке $x_1, \ldots, x_\ell$.

Решение: использовать рекуррентную формулу.

Среднее арифметическое $\bar{Q}_m = \frac{1}{m}\sum_{i=1}^{m} \varepsilon_i$:

$$\bar{Q}_m = (1 - \tfrac{1}{m})\bar{Q}_{m-1} + \tfrac{1}{m}\varepsilon_m.$$

Экспоненциальное скользящее среднее

$$\bar{Q}_m = \lambda\varepsilon_m + \lambda(1-\lambda)\varepsilon_{m-1} + \lambda(1-\lambda)^2\varepsilon_{m-2} + \lambda(1-\lambda)^3\varepsilon_{m-3} + \ldots$$

$$\bar{Q}_m := (1 - \lambda)\bar{Q}_{m-1} + \lambda\varepsilon_m.$$

Чем больше λ, тем быстрее забывается предыстория ряда.
Параметр λ называется *темпом забывания*.

Персептрон Розенблатта (1957)

- При синхронном возбуждении двух связанных нервных клеток синаптическая связь между ними усиливается.
- Математически (x_j – бинарные коорд. x_i):
  - $a(x_i, w) = y_i \implies w$ менять не нужно;
  - $a(x_i, w) = 0,\ y_i = 1 \implies w_j := w_j + h \cdot x_j$
  - $a(x_i, w) = 1,\ y_i = 0 \implies w_j := w_j - h \cdot x_j$
- Объединяем:

$$w := w - h\big(a(x_i, w) - y_i\big)x_i$$

Дельта-правило ADALINE

Задача регрессии: $x_i \in \mathbb{R}^{n+1}$, $y_i \in \mathbb{R}$

Адаптивный линейный элемент ADALINE [Видроу, Хофф 1960]:

$$a(x, w) = \langle w, x \rangle, \qquad \mathscr{L}_i(w) = (\langle w, x_i \rangle - y_i)^2.$$

Градиентный шаг SG — дельта-правило (delta-rule):

$$w := w - h\underbrace{(\langle w, x_i \rangle - y_i)}_{\Delta_i} x_i,$$

Δ_i — ошибка алгоритма $a(x, w)$ на объекте x_i.

Формально совпадает с правилом персептрона Розенблатта!

Правило Хэбба

Задача классификации: $x_i \in \mathbb{R}^{n+1}$, $y_i \in \{-1, +1\}$,

$$a(x, w) = \text{sign}\langle w, x\rangle, \qquad \mathscr{L}_i(w) = \big(-\langle w, x_i\rangle y_i\big)_+.$$

Градиентный шаг SG — правило Хэбба [1949]:

$$\textbf{если } \langle w, x_i\rangle y_i < 0 \textbf{ то } w := w + h x_i y_i,$$

То же самое для случая $y_i \in \{0, 1\}$,

$$a(x, w) = \big[\langle w, x\rangle > 0\big], \qquad \mathscr{L}_i(w) = \big(a(x_i, w) - y_i\big)\langle w, x_i\rangle,$$

Градиентный шаг SG — персептрон Розенблатта [1957]:

$$w := w - h\big(a(x_i, w) - y_i\big)x_i.$$

Правило Хэбба

Теорема (Новиков, 1962)

Пусть выборка X^ℓ линейно разделима:

$$\exists \tilde{w},\ \exists \delta > 0\colon \langle \tilde{w}, x_i \rangle y_i > \delta \quad \text{для всех} \quad i = 1, \ldots, \ell.$$

Тогда Алгоритм SG с правилом Хэбба находит вектор весов w,

- разделяющий обучающую выборку без ошибок;
- при любом начальном положении $w^{(0)}$;
- при любом темпе обучения $h > 0$;
- независимо от порядка предъявления объектов x_i;
- за конечное число исправлений вектора w;
- если $w^{(0)} = 0$, то число исправлений $t_{\max} \leqslant \frac{1}{\delta^2} \max \|x_i\|^2$.

Доказательство

Рассмотрим $\cos(\widehat{\tilde{w}, w^t}) = \dfrac{\langle \tilde{w}, w^t \rangle}{\|w^t\|}$ после t-го исправления w^t, при $\|\tilde{w}\| = 1$.

При t-м исправлении $\langle x_i, w^{t-1} \rangle y_i < 0$. В силу линейной разделимости

$$\langle \tilde{w}, w^t \rangle = \langle \tilde{w}, w^{t-1} \rangle + h\langle \tilde{w}, x_i \rangle y_i > \langle \tilde{w}, w^{t-1} \rangle + h\delta > \langle \tilde{w}, w^0 \rangle + th\delta.$$

В силу ограниченности выборки, $\|x_i\| < D$:

$$\|w^t\|^2 = \|w^{t-1}\|^2 + h^2\|x_i\|^2 + 2h\langle w^{t-1}, x_i \rangle y_i < \|w^{t-1}\|^2 + h^2 D^2 < \|w^0\|^2 + th^2 D^2.$$

Подставим эти соотношения в выражение для косинуса:

$$\cos(\widehat{\tilde{w}, w^t}) > \frac{\langle \tilde{w}, w^0 \rangle + th\delta}{\sqrt{\|w^0\|^2 + th^2 D^2}} \to \infty \text{ при } t \to \infty.$$

$\cos \leqslant 1$, значит при некотором t не найдётся ни одного $x_i \in X^\ell$ такого, что $\langle w^t, x_i \rangle y_i < 0$, то есть выборка окажется поделенной безошибочно.

Если $w^0 = 0$, то из условия $\cos = \dfrac{\sqrt{t}\delta}{D} \leqslant 1$ находим $t_{\max} = \left(\dfrac{D}{\delta}\right)^2$.

Понятие выступа для линейного классификатора

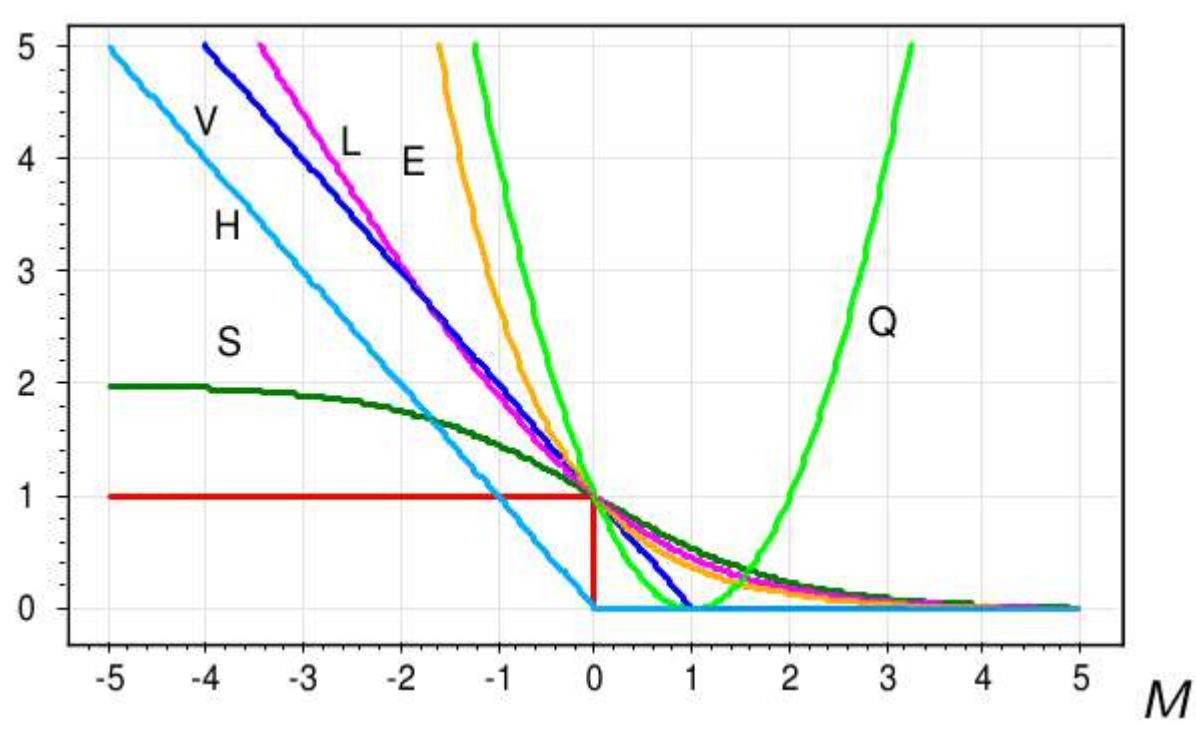
- Линейный классификатор:

$$a(x, w) = \text{sign}(x, w)$$

- $(x, w) = 0$ — разделяющая гиперплоскость,

- $M_i(w) = (w, x_i)\, y_i$ — отступ объекта x_i

Часто используемые непрерывные функции потерь



$V(M) = (1 - M)_+$ — кусочно-линейная (SVM);

$H(M) = (-M)_+$ — кусочно-линейная (Hebb's rule);

$L(M) = \log_2(1 + e^{-M})$ — логарифмическая (LR);

$Q(M) = (1 - M)^2$ — квадратичная (FLD);

$S(M) = 2(1 + e^{M})^{-1}$ — сигмоидная (ANN);

$E(M) = e^{-M}$ — экспоненциальная (AdaBoost);

$[M < 0]$ — пороговая функция потерь.

Начальное значение w

1. $w_j := 0$ для всех $j = 0, \ldots, n$;
2. небольшие случайные значения:
   $w_j := \text{random}\left(-\frac{1}{2n}, \frac{1}{2n}\right)$;
3. $w_j := \frac{\langle y, f_j \rangle}{\langle f_j, f_j \rangle}$, $f_j = \left(f_j(x_i)\right)_{i=1}^{\ell}$ — вектор значений признака.

   Упражнение: доказать, что оценка w оптимальна, если
   1) функция потерь квадратична и
   2) признаки некоррелированы, $\langle f_j, f_k \rangle = 0$, $j \neq k$.
4. $w_j := \ln \frac{\sum_i [y_i = +1] f_j(x_i)}{\sum_i [y_i = -1] f_j(x_i)}$ — для классификации, $Y = \{-1, +1\}$
5. обучение по небольшой случайной подвыборке объектов;
6. мультистарт: многократные запуски из разных случайных начальных приближений и выбор лучшего решения.

Порядок предъявления x_i

Возможны варианты:

1. *перетасовка объектов* (shuffling):
   попеременно брать объекты из разных классов;
2. чаще брать те объекты, на которых была допущена бóльшая ошибка
   (чем меньше M_i, тем больше вероятность взять объект)
   (чем меньше $|M_i|$, тем больше вероятность взять объект);
3. вообще не брать «хорошие» объекты, у которых $M_i > \mu_+$
   (при этом немного ускоряется сходимость);
4. вообще не брать объекты-«выбросы», у которых $M_i < \mu_-$
   (при этом может улучшиться качество классификации);

Параметры μ_+, μ_- придётся подбирать.

Выбор шага h

1. сходимость гарантируется (для выпуклых функций) при
$$h_t \to 0, \quad \sum_{t=1}^{\infty} h_t = \infty, \quad \sum_{t=1}^{\infty} h_t^2 < \infty,$$
в частности можно положить $h_t = 1/t$;
2. *метод скорейшего градиентного спуска*:
$$\mathscr{L}_i\big(w - h\nabla\mathscr{L}_i(w)\big) \to \min_h,$$
позволяет найти *адаптивный шаг* h^*;

   Упражнение: доказать, что при квадратичной функции потерь $h^* = \|x_i\|^{-2}$.
3. пробные случайные шаги
   — для «выбивания» из локальных минимумов;

Достоинства и недостатки

Достоинства:

1. легко реализуется;
2. легко обобщается на любые $g(x, w)$, $\mathscr{L}(a, y)$;
3. возможно динамическое (потоковое) обучение;
4. на сверхбольших выборках можно получить неплохое решение, даже не обработав все (x_i, y_i);
5. всё чаще применяется для Big Data

Недостатки:

1. возможна расходимость или медленная сходимость;
2. застревание в локальных минимумах;
3. подбор комплекса эвристик является искусством;
4. проблема переобучения;

Проблема переобучения

Возможные причины переобучения:

1. слишком мало объектов; слишком много признаков;
2. линейная зависимость (мультиколлинеарность) признаков:
   пусть построен классификатор: $a(x, w) = \text{sign}\langle w, x\rangle$;
   мультиколлинеарность: $\exists u \in \mathbb{R}^{n+1}$: $\forall x \;\; \langle u, x\rangle \equiv 0$;
   тогда $\forall \gamma \in \mathbb{R} \quad a(x, w) = \text{sign}\langle w + \gamma u, x\rangle$

Симптоматика:

1. слишком большие веса $|w_j|$ разных знаков;
2. неустойчивость $a(x, w)$;
3. $Q(X^\ell) \ll Q(X^k)$;

Терапия:

1. регуляризация (сокращение весов, weight decay);
2. ранний останов (early stopping);

Регуляризация

Штраф за увеличение нормы вектора весов:

$$\widetilde{\mathscr{L}}_i(w) = \mathscr{L}_i(w) + \frac{\tau}{2}\|w\|^2 = \mathscr{L}_i(w) + \frac{\tau}{2}\sum_{j=1}^{n} w_j^2 \to \min_w .$$

Градиент:

$$\nabla\widetilde{\mathscr{L}}_i(w) = \nabla\mathscr{L}_i(w) + \tau w.$$

Модификация градиентного шага:

$$w := w(1 - h\tau) - h\nabla\mathscr{L}_i(w).$$

Разные штрафы за ошибки

Задача классификации на два класса, $y_i \in \{-1, +1\}$.
Модель классификации: $a(x; w, w_0) = \text{sign}\big(g(x, w) - w_0\big)$.
Чем меньше w_0, тем больше x_i: $a(x_i) = +1$.

Пусть λ_y — штраф за ошибку на объекте класса y.
Функция потерь теперь зависит от штрафов:

$$\mathscr{L}(a, y) = \lambda_{y_i}\big[a(x_i; w, w_0) \neq y_i\big] = \lambda_{y_i}\big[(g(x_i, w) - w_0)y_i < 0\big].$$

Проблема

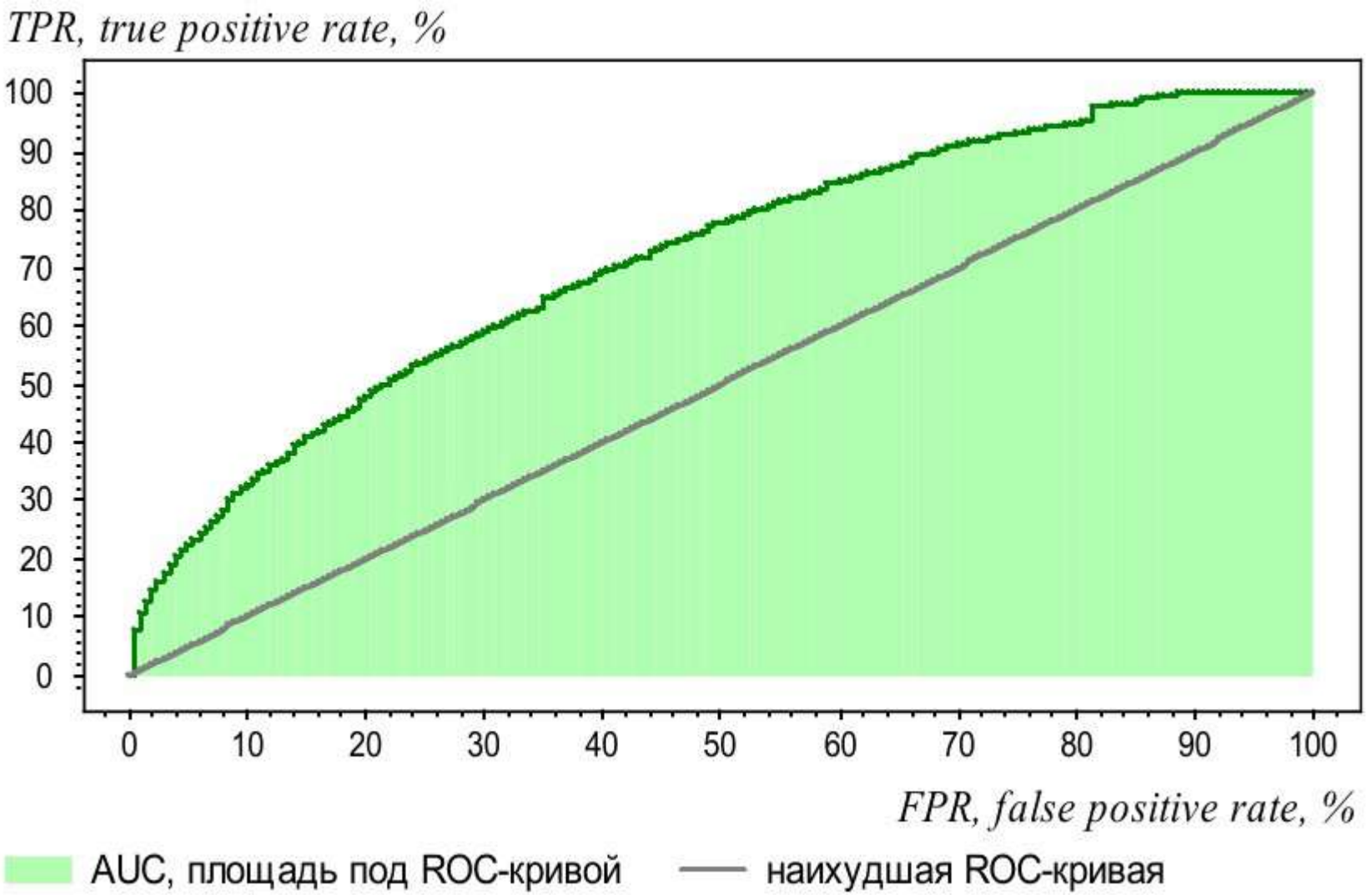
На практике штрафы $\{\lambda_y\}$ могут пересматриваться

- Нужен удобный способ выбора w_0 в зависимости от $\{\lambda_y\}$, не требующий построения w заново.
- Нужна характеристика качества модели $g(x, w)$, не зависящая от штрафов $\{\lambda_y\}$ и численности классов.

ROC-кривая

- ROC – receiver operating characteristic
- Каждая точка кривой соответствует одному значению порога (цен за ошибки, w_0)
- По оси X:
  FPR (false positive rate) – процент объектов с y=-1 и a(x)=+1 среди всех y=-1
- По оси Y:
  TPR (true positive rate) – процент объектов с y=+1 и a(x)=+1 среди всех y=+1

Пример


TPR, true positive rate, %
100
90
80
70
60
50
40
30
20
10
0
0
10
20
30
40
50
60
70
80
90
100
FPR, false positive rate, %
AUC, площадь под ROC-кривой
наихудшая ROC-кривая

Алгоритм построения ROC-кривой

Вход: выборка X^ℓ; дискриминантная функция $g(x, w)$;

Выход: $\{(\mathrm{FPR}_i, \mathrm{TPR}_i)\}_{i=0}^{\ell}$, AUC — площадь под ROC-кривой.

$\ell_y := \sum_{i=1}^{\ell} [y_i = y]$, для всех $y \in Y$;

упорядочить выборку X^ℓ по убыванию значений $g(x_i, w)$;

поставить первую точку в начало координат:
$(\mathrm{FPR}_0, \mathrm{TPR}_0) := (0, 0)$; $\mathrm{AUC} := 0$;

для $i := 1, \ldots, \ell$

  если $y_i = -1$ **то** сместиться на один шаг вправо:

    $\mathrm{FPR}_i := \mathrm{FPR}_{i-1} + \frac{1}{\ell_-}$; $\mathrm{TPR}_i := \mathrm{TPR}_{i-1}$;

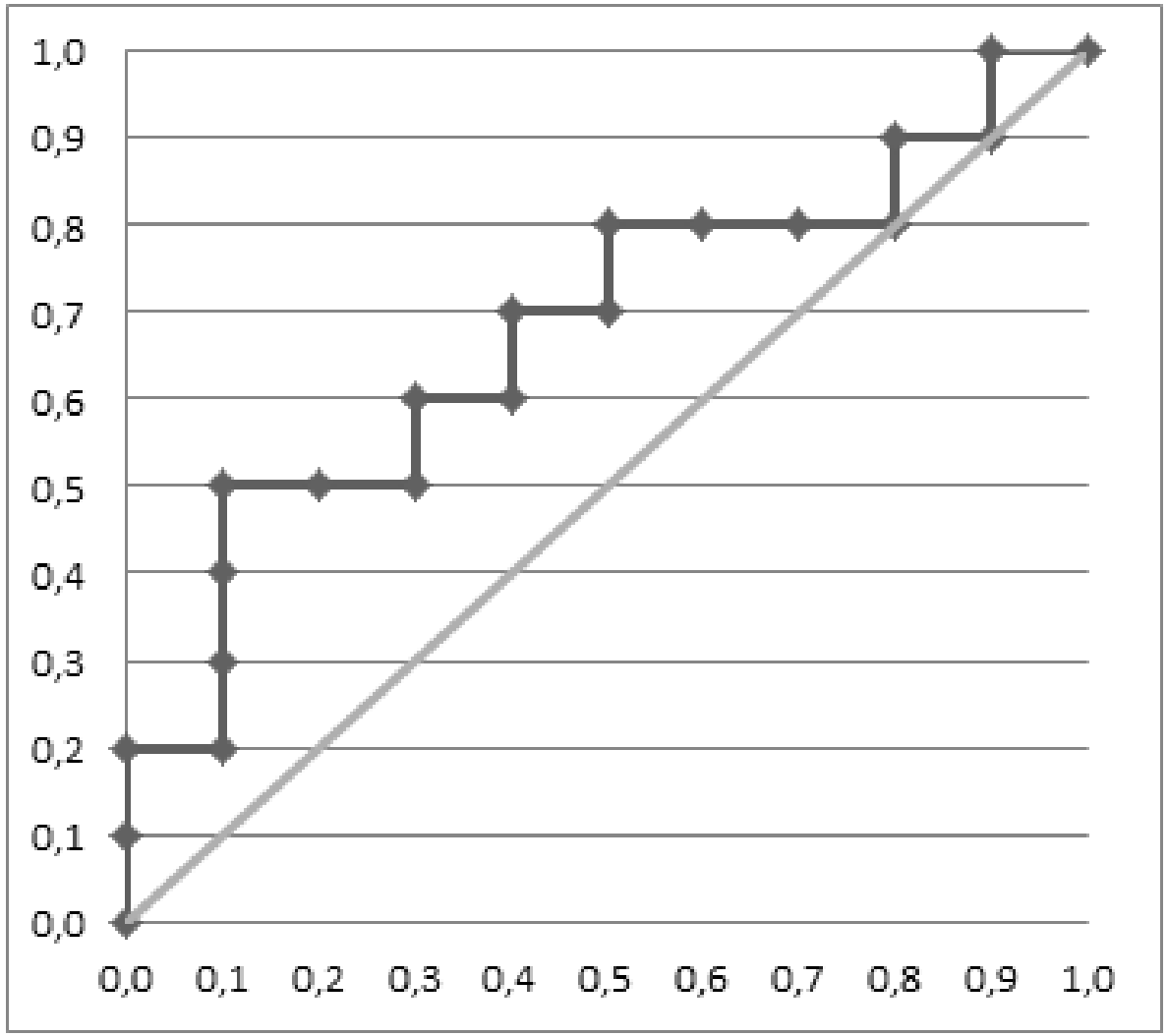
    $\mathrm{AUC} := \mathrm{AUC} + \frac{1}{\ell_-}\mathrm{TPR}_i$;

  иначе сместиться на один шаг вверх:

    $\mathrm{FPR}_i := \mathrm{FPR}_{i-1}$; $\mathrm{TPR}_i := \mathrm{TPR}_{i-1} + \frac{1}{\ell_+}$;

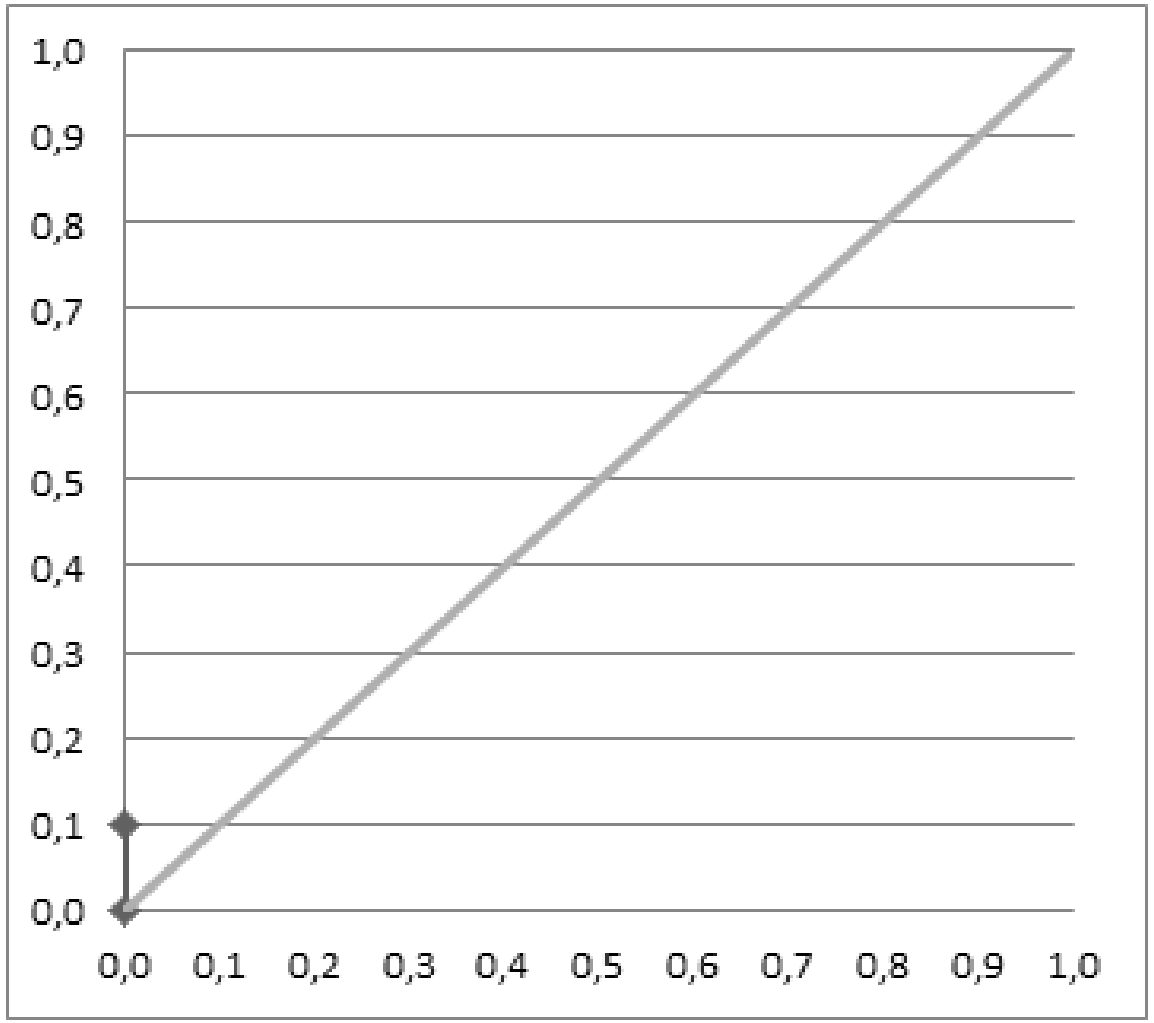
Пример

#	C	Score
1	P	0,9
2	P	0,8
3	N	0,7
4	P	0,6
5	P	0,55
6	P	0,54
7	N	0,53
8	N	0,52
9	P	0,51
10	N	0,505
11	P	0,4
12	N	0,39
13	P	0,38
14	N	0,37
15	N	0,36
16	N	0,35
17	P	0,34
18	N	0,33
19	P	0,3
20	N	0,1

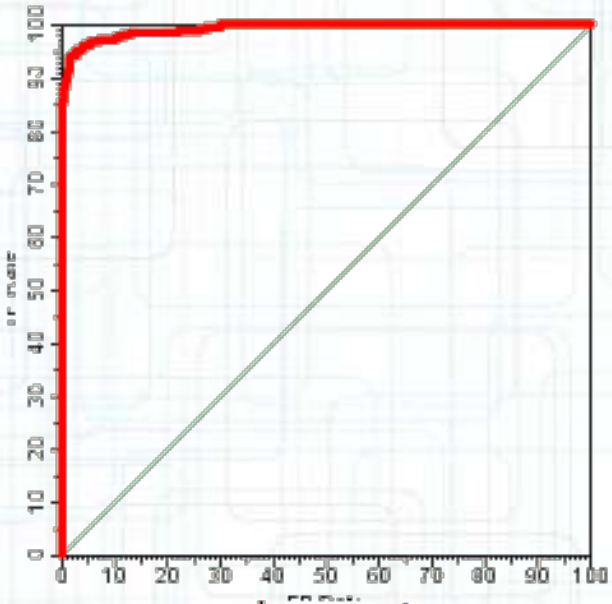


Пример

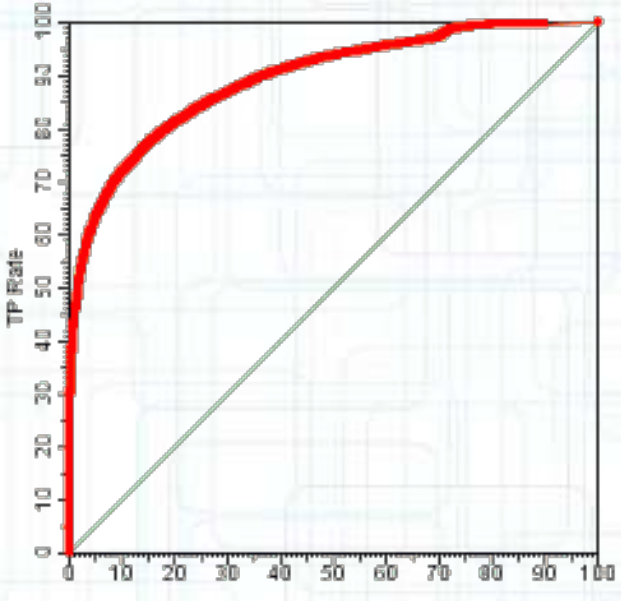
#	C	Score
1	P	0,9
2	P	0,8
3	N	0,7
4	P	0,6
5	P	0,55
6	P	0,54
7	N	0,53
8	N	0,52
9	P	0,51
10	N	0,505
11	P	0,4
12	N	0,39
13	P	0,38
14	N	0,37
15	N	0,36
16	N	0,35
17	P	0,34
18	N	0,33
19	P	0,3
20	N	0,1



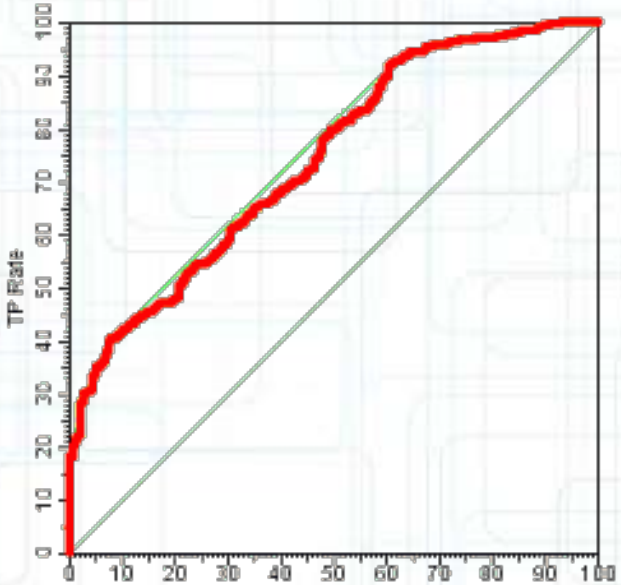
Еще примеры



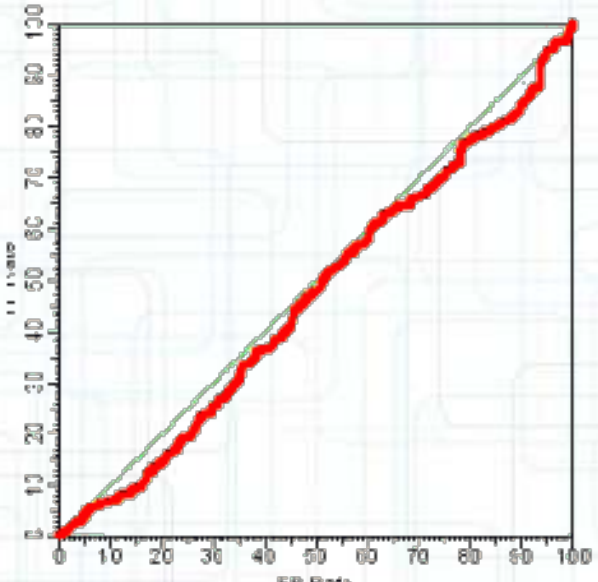
good separation



reasonable



poor separation



random separation

Градиентная максимизация AUC

Модель: $a(x_i, w, w_0) = \text{sign}\big(g(x_i, w) - w_0\big)$.

AUC — это доля правильно упорядоченных пар (x_i, x_j):

$$\text{AUC}(w) = \frac{1}{\ell_-}\sum_{i=1}^{\ell}\big[y_i = -1\big]\text{TPR}_i =$$

$$= \frac{1}{\ell_-\ell_+}\sum_{i=1}^{\ell}\sum_{j=1}^{\ell}\big[y_i < y_j\big]\big[g(x_i, w) < g(x_j, w)\big] \to \max_w$$

Явная максимизация аппроксимированного AUC:

$$\text{AUC}(w) \leqslant Q(w) = \sum_{i,j:\ y_i<y_j} \mathscr{L}\big(\underbrace{g(x_j, w) - g(x_i, w)}_{M_{ij}(w)}\big) \to \min_w,$$

где $\mathscr{L}(M)$ — гладкая убывающая функция отступа,
$M_{ij}(w)$ — новое понятие отступа для пар объектов.

Алгоритм стохастического градиента для AUC

Возьмём для простоты линейный классификатор:

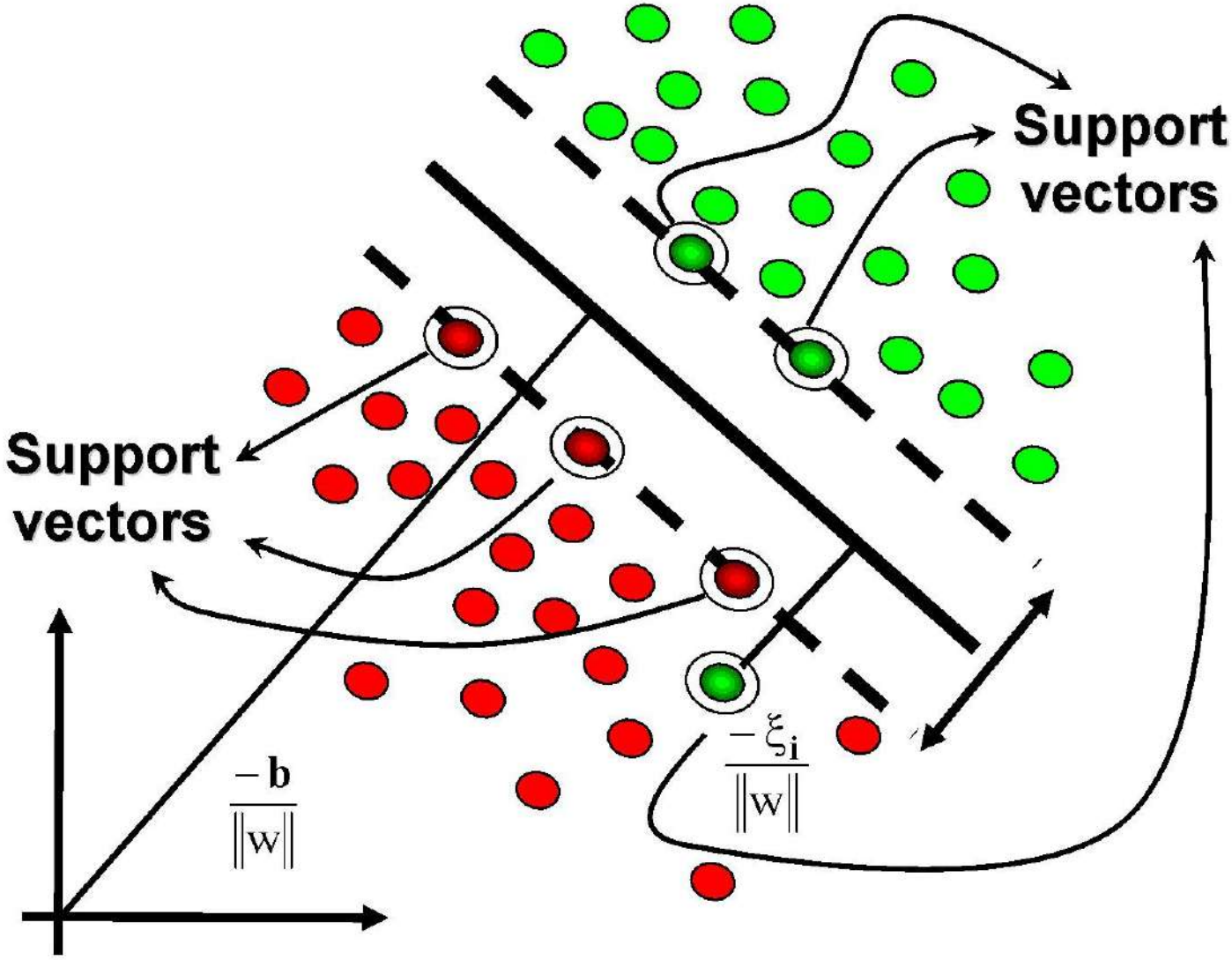
$$g(x, w) = \langle x, w \rangle, \qquad M_{ij}(w) = \langle x_j - x_i, w \rangle.$$

Вход: выборка X^ℓ, темп обучения h, темп забывания λ

Выход: вектор весов w

1: инициализировать веса w_j, $j = 0, \ldots, n$;
2: инициализировать оценку: $\bar{Q} := \frac{1}{\ell_+ \ell_-} \sum_{i,j:\, y_i < y_j} \mathscr{L}(M_{ij}(w))$;
3: **повторять**
4: выбрать пару объектов (i, j): $y_i < y_j$, случайным образом;
5: вычислить потерю: $\varepsilon_{ij} := \mathscr{L}(M_{ij}(w))$;
6: сделать градиентный шаг: $w := w - h\mathscr{L}'(M_{ij}(w))(x_j - x_i)$;
7: оценить функционал: $\bar{Q} := (1 - \lambda)\bar{Q} + \lambda\varepsilon_{ij}$;
8: **пока** значение $\bar{Q}$ и/или веса w не сойдутся;

Машинное обучение
Метод опорных векторов (SVM)



https://yandexdataschool.ru/edu-process/courses/machine-learning

Содержание лекции

- Случаи линейно разделимой и неразделимой выборок
- Двойственная задача
- Типы объектов
- Нелинейное обобщение SVM
- SVM-регрессия
- L_1 регуляризация

Самая широкая разделяющая полоса

- Рассмотрим линейный классификатор:

$$a(x, w) = \operatorname{sign}\big(\langle w, x\rangle - w_0\big)$$

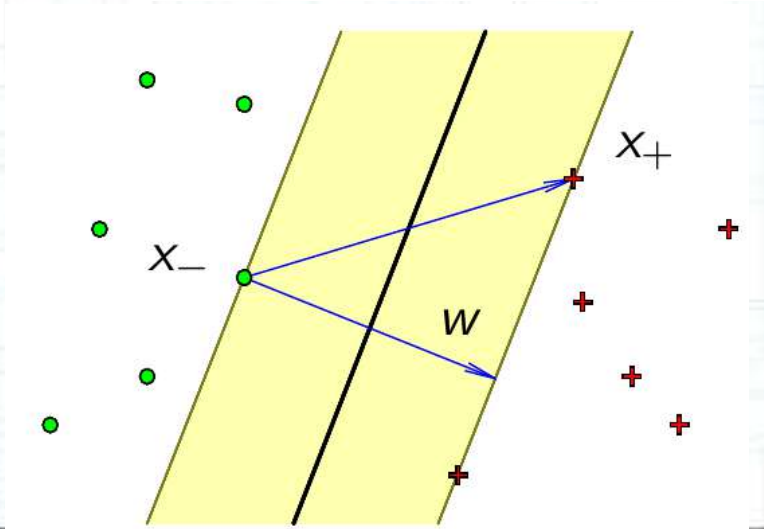
- Допустим, что обучающая выборка линейно разделима:

$$\exists w, w_0 : \quad M_i(w, w_0) = y_i\big(\langle w, x_i\rangle - w_0\big) > 0, \quad i = 1, \ldots, \ell$$

- w и w0 определены с точностью до множителя $\Rightarrow$ нормируем $\min\limits_{i=1,\ldots,\ell} M_i(w, w_0) = 1$

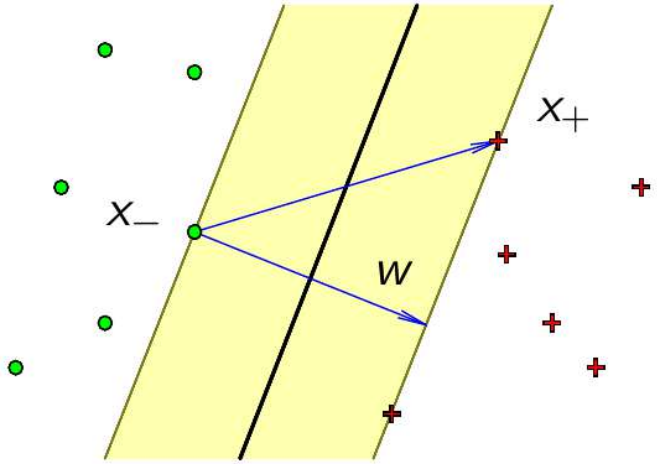
- Ширина полосы:

$$\frac{\langle x_+ - x_-, w\rangle}{\|w\|} = \frac{2}{\|w\|} \to \max$$



Метод опорных векторов для линейно разделимой выборки

$$\begin{cases} \frac{1}{2}\|w\|^2 \to \min\limits_{w,w_0}; \\ M_i(w,w_0) \geqslant 1, \quad i = 1,\ldots,\ell \end{cases}$$



Что делать, если выборка
не разделима гиперплоскостью?

Случай линейно неразделимой выборки

$$\begin{cases} \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{\ell}\xi_i \to \min\limits_{w,w_0,\xi}; \\ M_i(w,w_0) \geqslant 1 - \xi_i, \quad i = 1,\ldots,\ell; \\ \xi_i \geqslant 0, \quad i = 1,\ldots,\ell. \end{cases}$$

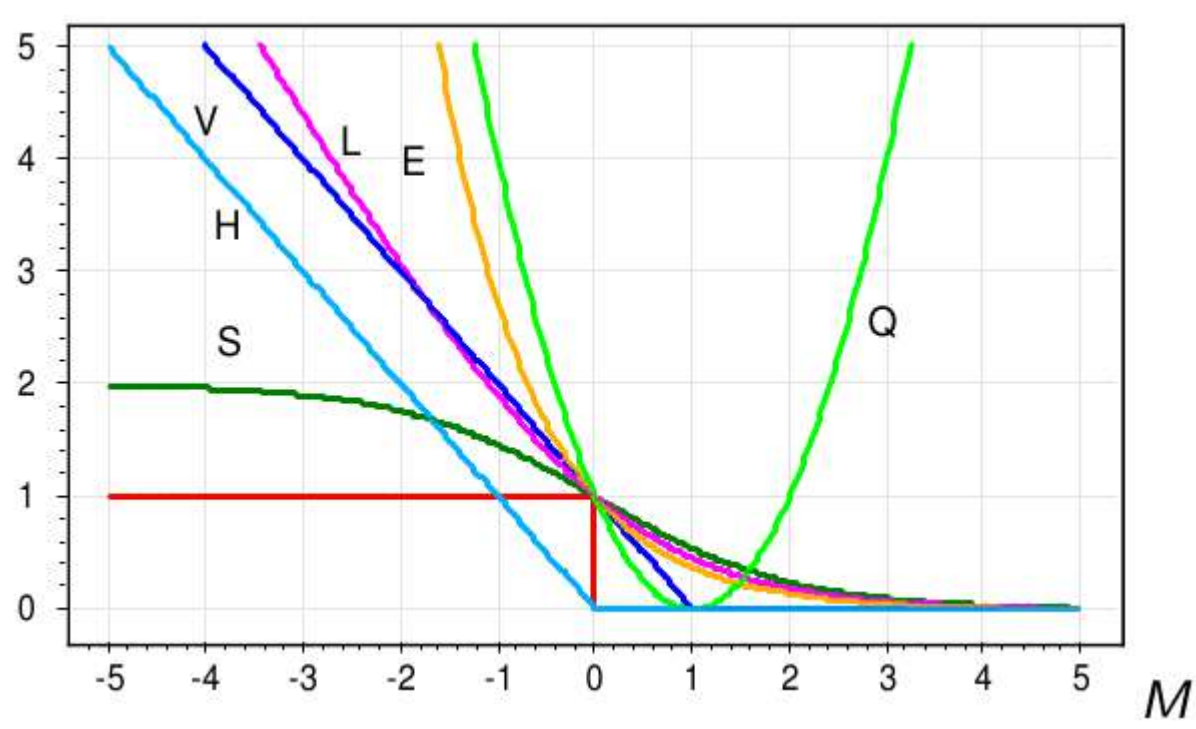
Так как $\xi_i \geqslant 0$ и $\xi_i \geqslant 1 - M_i$ **то в силу минимизации суммы** ξ_i

$$\xi_i = (1 - M_i)_+$$

Следовательно, наша задача эквивалентна минимизации функционала

$$Q(w,w_0) = \sum_{i=1}^{\ell}\big(1 - M_i(w,w_0)\big)_+ + \frac{1}{2C}\|w\|^2 \to \min_{w,w_0}$$

Часто используемые функции потерь



$V(M) = (1 - M)_+$ — кусочно-линейная (SVM);

$H(M) = (-M)_+$ — кусочно-линейная (Hebb's rule);

$L(M) = \log_2(1 + e^{-M})$ — логарифмическая (LR);

$Q(M) = (1 - M)^2$ — квадратичная (FLD);

$S(M) = 2(1 + e^{M})^{-1}$ — сигмоидная (ANN);

$E(M) = e^{-M}$ — экспоненциальная (AdaBoost);

$[M < 0]$ — пороговая функция потерь.

Условия Каруша–Куна–Таккера

Задача математического программирования:

$$\begin{cases} f(x) \to \min\limits_{x}; \\ g_i(x) \leqslant 0, \quad i = 1, \ldots, m; \\ h_j(x) = 0, \quad j = 1, \ldots, k. \end{cases}$$

Необходимые условия. Если x — точка локального минимума, то существуют множители μ_i, $i = 1, \ldots, m$, λ_j, $j = 1, \ldots, k$:

$$\begin{cases} \dfrac{\partial \mathscr{L}}{\partial x} = 0, \quad \mathscr{L}(x; \mu, \lambda) = f(x) + \sum\limits_{i=1}^{m} \mu_i g_i(x) + \sum\limits_{j=1}^{k} \lambda_j h_j(x); \\ g_i(x) \leqslant 0; \ h_j(x) = 0; \ \text{(исходные ограничения)} \\ \mu_i \geqslant 0; \ \text{(двойственные ограничения)} \\ \mu_i g_i(x) = 0; \ \text{(условие дополняющей нежёсткости)} \end{cases}$$

Применение условий ККТ к задаче SVM

Функция Лагранжа: $\mathscr{L}(w, w_0, \xi; \lambda, \eta) =$

$$= \frac{1}{2}\|w\|^2 - \sum_{i=1}^{\ell} \lambda_i \big(M_i(w, w_0) - 1\big) - \sum_{i=1}^{\ell} \xi_i (\lambda_i + \eta_i - C),$$

λ_i — переменные, двойственные к ограничениям $M_i \geqslant 1 - \xi_i$;
η_i — переменные, двойственные к ограничениям $\xi_i \geqslant 0$.

$$\begin{cases} \dfrac{\partial \mathscr{L}}{\partial w} = 0, \quad \dfrac{\partial \mathscr{L}}{\partial w_0} = 0, \quad \dfrac{\partial \mathscr{L}}{\partial \xi} = 0; \\ \xi_i \geqslant 0, \quad \lambda_i \geqslant 0, \quad \eta_i \geqslant 0, \quad i = 1, \ldots, \ell; \\ \lambda_i = 0 \;\text{ либо }\; M_i(w, w_0) = 1 - \xi_i, \quad i = 1, \ldots, \ell; \\ \eta_i = 0 \;\text{ либо }\; \xi_i = 0, \quad i = 1, \ldots, \ell; \end{cases}$$

Необходимые условия седловой точки функции Лагранжа

Функция Лагранжа: $\mathscr{L}(w, w_0, \xi; \lambda, \eta) =$

$$= \frac{1}{2}\|w\|^2 - \sum_{i=1}^{\ell} \lambda_i \big(M_i(w, w_0) - 1\big) - \sum_{i=1}^{\ell} \xi_i \big(\lambda_i + \eta_i - C\big),$$

Необходимые условия седловой точки функции Лагранжа:

$$\frac{\partial \mathscr{L}}{\partial w} = w - \sum_{i=1}^{\ell} \lambda_i y_i x_i = 0 \quad \Longrightarrow \quad w = \sum_{i=1}^{\ell} \lambda_i y_i x_i;$$

$$\frac{\partial \mathscr{L}}{\partial w_0} = -\sum_{i=1}^{\ell} \lambda_i y_i = 0 \quad \Longrightarrow \quad \sum_{i=1}^{\ell} \lambda_i y_i = 0;$$

$$\frac{\partial \mathscr{L}}{\partial \xi_i} = -\lambda_i - \eta_i + C = 0 \quad \Longrightarrow \quad \eta_i + \lambda_i = C, \quad i = 1, \ldots, \ell.$$

Типы объектов

Типизация объектов:

1. $\lambda_i = 0;\ \ \eta_i = C;\ \ \xi_i = 0;\ \ M_i \geqslant 1.$
   — периферийные (неинформативные) объекты.

2. $0 < \lambda_i < C;\ \ 0 < \eta_i < C;\ \ \xi_i = 0;\ \ M_i = 1.$
   — опорные граничные объекты.

3. $\lambda_i = C;\ \ \eta_i = 0;\ \ \xi_i > 0;\ \ M_i < 1.$
   — опорные-нарушители.

- Объект x_i называется опорным, если $\lambda_i \neq 0$.

Двойственная задача

$$\begin{cases} -\mathscr{L}(\lambda) = -\sum\limits_{i=1}^{\ell} \lambda_i + \frac{1}{2} \sum\limits_{i=1}^{\ell} \sum\limits_{j=1}^{\ell} \lambda_i \lambda_j y_i y_j \langle x_i, x_j \rangle \;\to\; \min\limits_{\lambda}; \\ 0 \leqslant \lambda_i \leqslant C, \quad i = 1, \ldots, \ell; \\ \sum\limits_{i=1}^{\ell} \lambda_i y_i = 0. \end{cases}$$

Решение прямой задачи выражается через решение двойственной:

$$\begin{cases} w = \sum\limits_{i=1}^{\ell} \lambda_i y_i x_i; \\ w_0 = \langle w, x_i \rangle - y_i, \qquad \text{для любого } i\text{: } \lambda_i > 0,\ M_i = 1. \end{cases}$$

Линейный классификатор:

$$a(x) = \operatorname{sign}\Big(\sum_{i=1}^{\ell} \lambda_i y_i \langle x_i, x \rangle - w_0 \Big).$$

Обучение SVM

1. Find $\boldsymbol{\alpha}^1$ as the initial feasible solution. Set $k = 1$.

2. If $\boldsymbol{\alpha}^k$ is an optimal solution of (1), stop. Otherwise, find a *two-element* working set $B = \{i, j\} \subset \{1, \ldots, l\}$. Define $N \equiv \{1, \ldots, l\} \backslash B$ and $\boldsymbol{\alpha}_B^k$ and $\boldsymbol{\alpha}_N^k$ to be sub-vectors of $\boldsymbol{\alpha}^k$ corresponding to B and N, respectively.

3. Solve the following sub-problem with the variable $\boldsymbol{\alpha}_B$:

$$\begin{aligned}
\min_{\boldsymbol{\alpha}_B} \quad & \frac{1}{2}\begin{bmatrix}\boldsymbol{\alpha}_B^T & (\boldsymbol{\alpha}_N^k)^T\end{bmatrix}\begin{bmatrix}Q_{BB} & Q_{BN} \\ Q_{NB} & Q_{NN}\end{bmatrix}\begin{bmatrix}\boldsymbol{\alpha}_B \\ \boldsymbol{\alpha}_N^k\end{bmatrix} - \begin{bmatrix}\mathbf{e}_B^T & \mathbf{e}_N^T\end{bmatrix}\begin{bmatrix}\boldsymbol{\alpha}_B \\ \boldsymbol{\alpha}_N^k\end{bmatrix} \\
& = \frac{1}{2}\boldsymbol{\alpha}_B^T Q_{BB}\boldsymbol{\alpha}_B + (-\mathbf{e}_B + Q_{BN}\boldsymbol{\alpha}_N^k)^T\boldsymbol{\alpha}_B + \text{ constant} \\
& = \frac{1}{2}\begin{bmatrix}\alpha_i & \alpha_j\end{bmatrix}\begin{bmatrix}Q_{ii} & Q_{ij} \\ Q_{ij} & Q_{jj}\end{bmatrix}\begin{bmatrix}\alpha_i \\ \alpha_j\end{bmatrix} + (-\mathbf{e}_B + Q_{BN}\boldsymbol{\alpha}_N^k)^T\begin{bmatrix}\alpha_i \\ \alpha_j\end{bmatrix} + \text{ constant} \\
\text{subject to} \quad & 0 \le \alpha_i, \alpha_j \le C, \qquad (2) \\
& y_i\alpha_i + y_j\alpha_j = -\mathbf{y}_N^T\boldsymbol{\alpha}_N^k,
\end{aligned}$$

where $\begin{bmatrix}Q_{BB} & Q_{BN} \\ Q_{NB} & Q_{NN}\end{bmatrix}$ is a permutation of the matrix Q.

4. Set $\boldsymbol{\alpha}_B^{k+1}$ to be the optimal solution of (2) and $\boldsymbol{\alpha}_N^{k+1} \equiv \boldsymbol{\alpha}_N^k$. Set $k \leftarrow k + 1$ and goto Step 2.

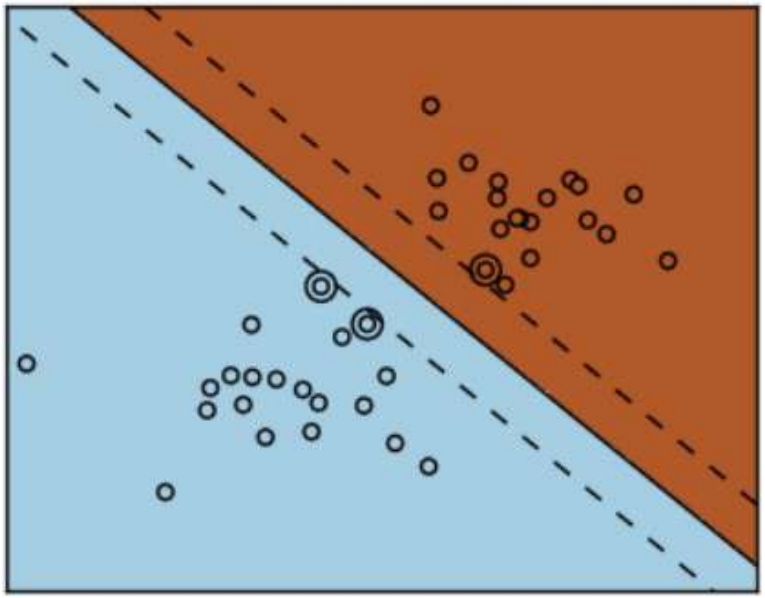
R.-E. Fan, P.-H. Chen, and C.-J. Lin. Working set selection using second order information for training SVM. Journal of Machine Learning Research 6, 1889-1918, 2005

Влияние константы C на решение SVM

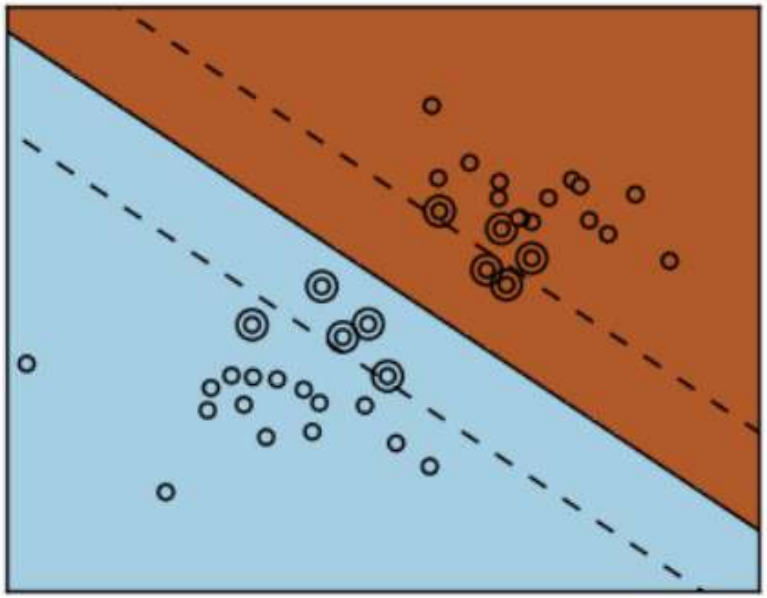
SVM — аппроксимация и регуляризация эмпирического риска:

$$\sum_{i=1}^{\ell}\big(1 - M_i(w, w_0)\big)_+ + \frac{1}{2C}\|w\|^2 \;\to\; \min_{w, w_0}.$$

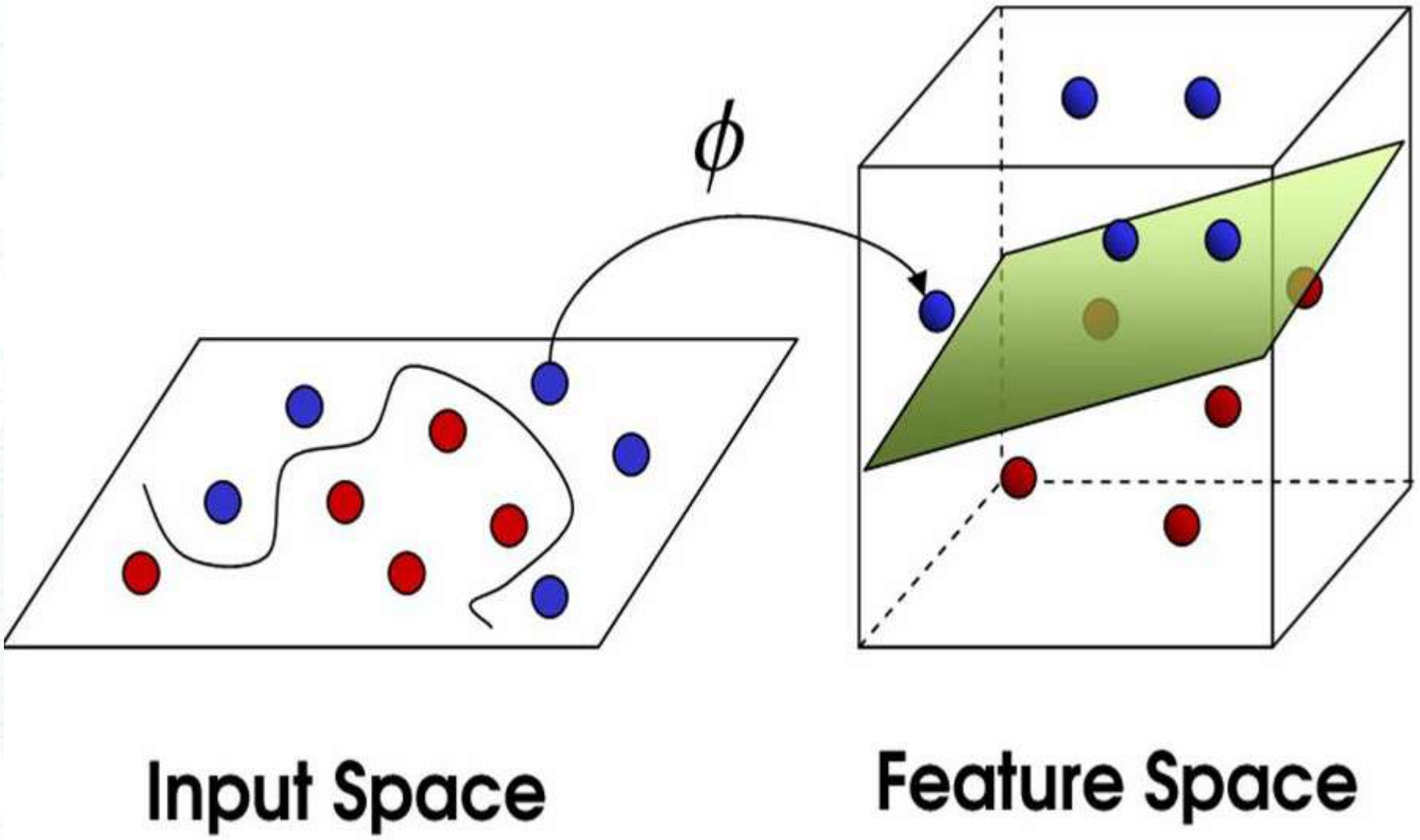
большой C
слабая регуляризация



малый C
сильная регуляризация



Нелинейное обобщение SVM
Расширение пространства



Видео-демострация

Полиномиальные ядра

$$\psi\colon (u_1, u_2) \mapsto \left(u_1^2, u_2^2, \sqrt{2}u_1 u_2\right)$$

$$K(x, x') = \langle \psi(x), \psi(x') \rangle_H$$

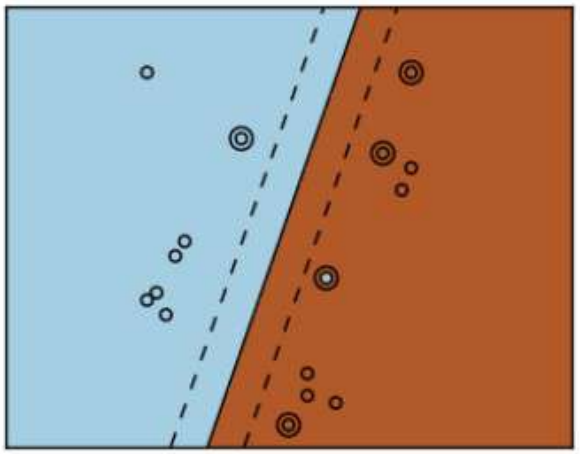
$$\begin{aligned} K(u, v) = \langle u, v \rangle^2 &= \langle (u_1, u_2), (v_1, v_2) \rangle^2 = \\ &= (u_1 v_1 + u_2 v_2)^2 = u_1^2 v_1^2 + u_2^2 v_2^2 + 2u_1 v_1 u_2 v_2 = \\ &= \langle (u_1^2, u_2^2, \sqrt{2}u_1 u_2), (v_1^2, v_2^2, \sqrt{2}v_1 v_2) \rangle. \end{aligned}$$

В общем случае: $K(x, x') = \left(\langle x, x' \rangle + 1\right)^d$

Примеры классификаций с различными ядрами

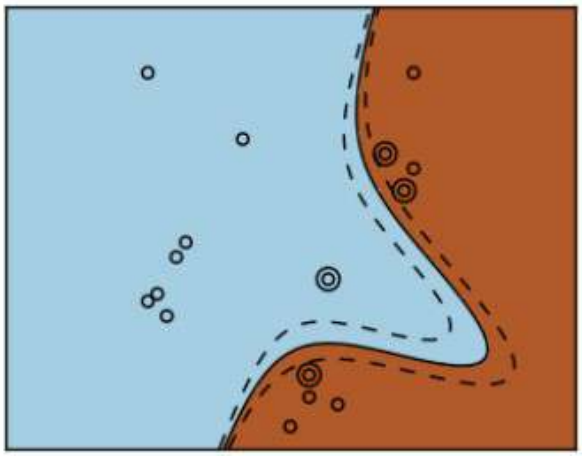
линейное

$\langle x, x' \rangle$



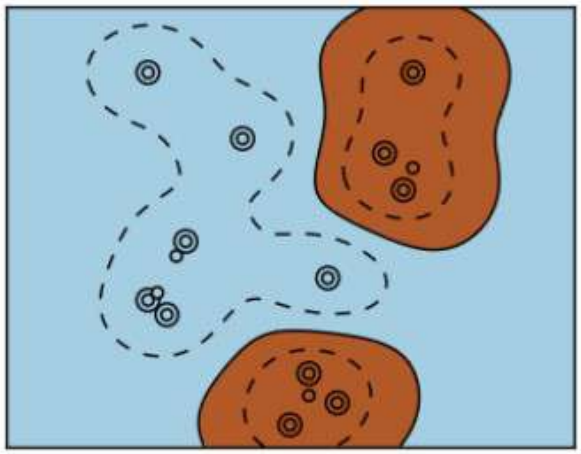
полиномиальное

$(\langle x, x' \rangle + 1)^d, \quad d=3$

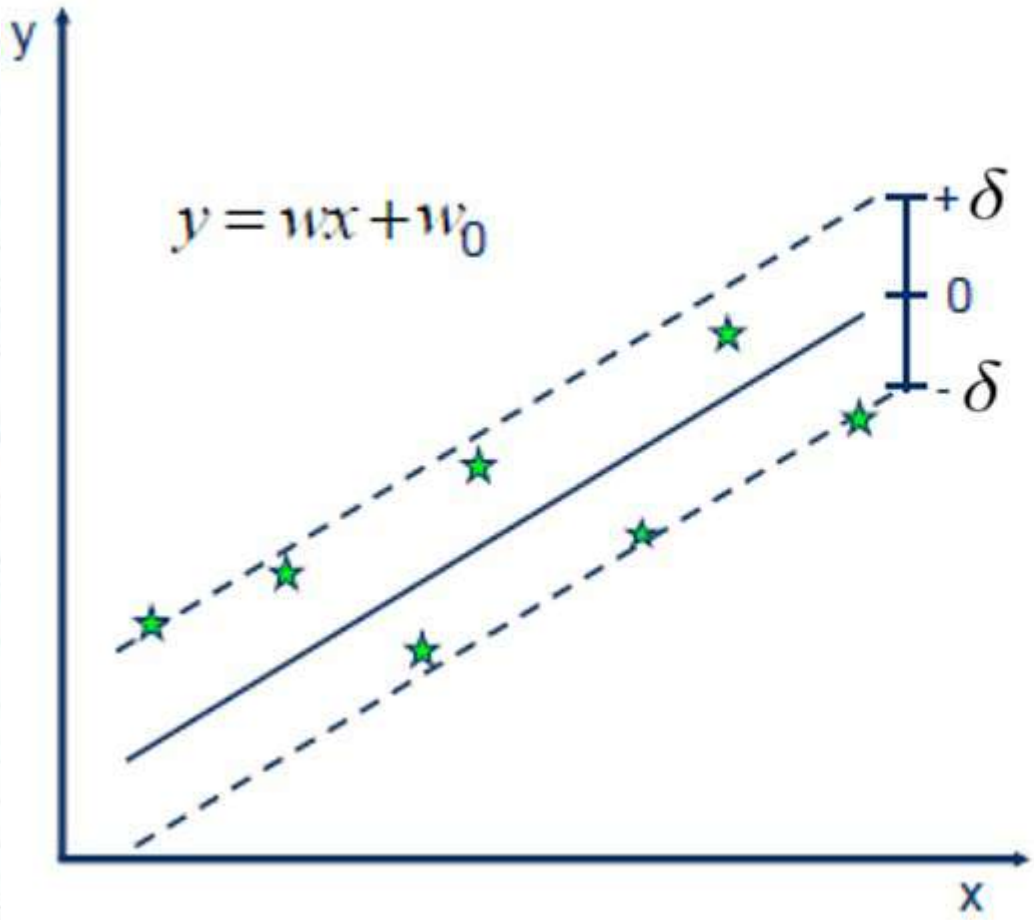


гауссовское (RBF)

$\exp(-\beta \|x - x'\|^2)$



SVM-регрессия



- Задача:

$$\min \frac{1}{2}\|w\|^2$$

- Ограничения

$$y_i - wx_i - w_0 \le \delta$$

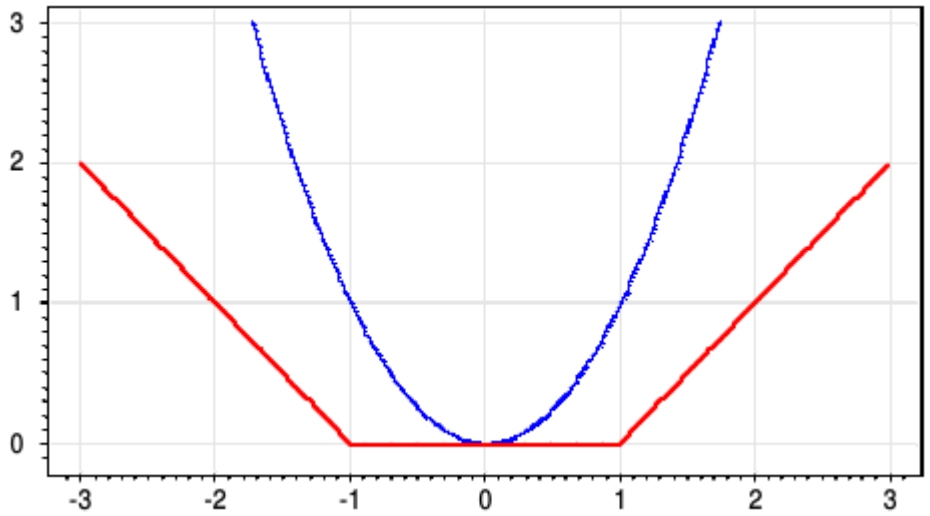
$$wx_i + w_0 - y_i \le \delta$$

Эквивалентная постановка задачи

$$\sum_{i=1}^{\ell}\left(|\langle w, x_i\rangle - w_0 - y_i| - \delta\right)_+ + \frac{1}{2C}\|w\|^2 \to \min_{w, w_0}$$

Сравнение с обычной регрессией (МНК)

Функция потерь: $\mathscr{L}(\varepsilon) = \left(|\varepsilon| - \delta\right)_+$ в сравнении с $\mathscr{L}(\varepsilon) = \varepsilon^2$



Задача решается путём замены переменных и сведения к задаче квадратичного программирования

Решение задачи оптимизации

Замена переменных:

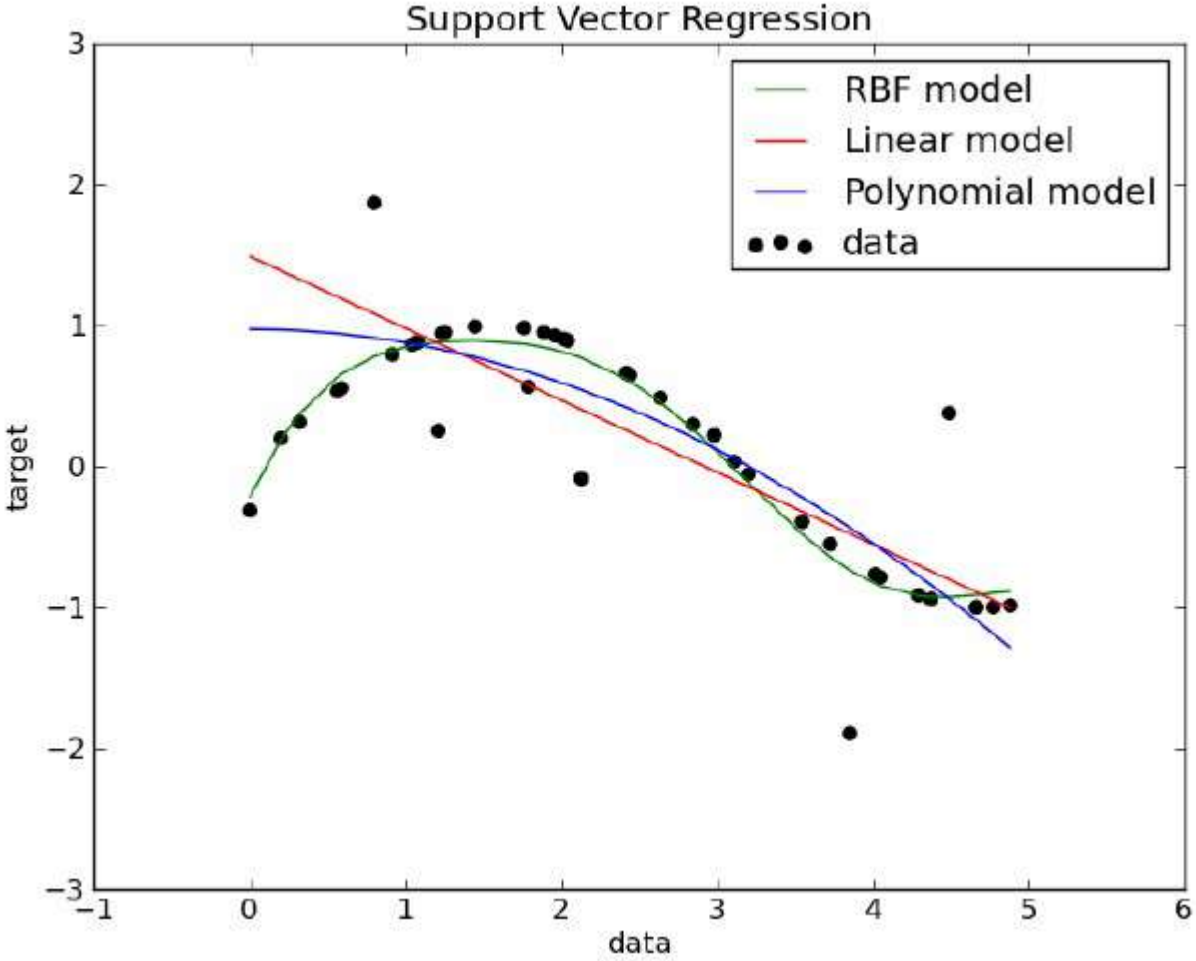
$$\xi_i^+ = (\langle w, x_i \rangle - w_0 - y_i - \delta)_+;$$
$$\xi_i^- = (-\langle w, x_i \rangle + w_0 + y_i - \delta)_+;$$

$$\begin{cases} \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{\ell}(\xi_i^+ + \xi_i^-) \to \min\limits_{w, w_0, \xi^+, \xi^-}; \\ y_i - \delta - \xi_i^- \leqslant \langle w, x_i \rangle - w_0 \leqslant y_i + \delta + \xi_i^+, \quad i = 1, \ldots, \ell; \\ \xi_i^- \geqslant 0, \quad \xi_i^+ \geqslant 0, \quad i = 1, \ldots, \ell. \end{cases}$$

Это задача квадратичного программирования с линейными ограничениями-неравенствами, решается также сведением к двойственной задаче.

Сравнение

- Сравнение SVM-регрессии с гауссовским (RBF) ядром, линейной и полиномиальной регрессией:



1-norm SVM (LASSO SVM)

Аппроксимация эмпирического риска с L_1 -регуляризацией:

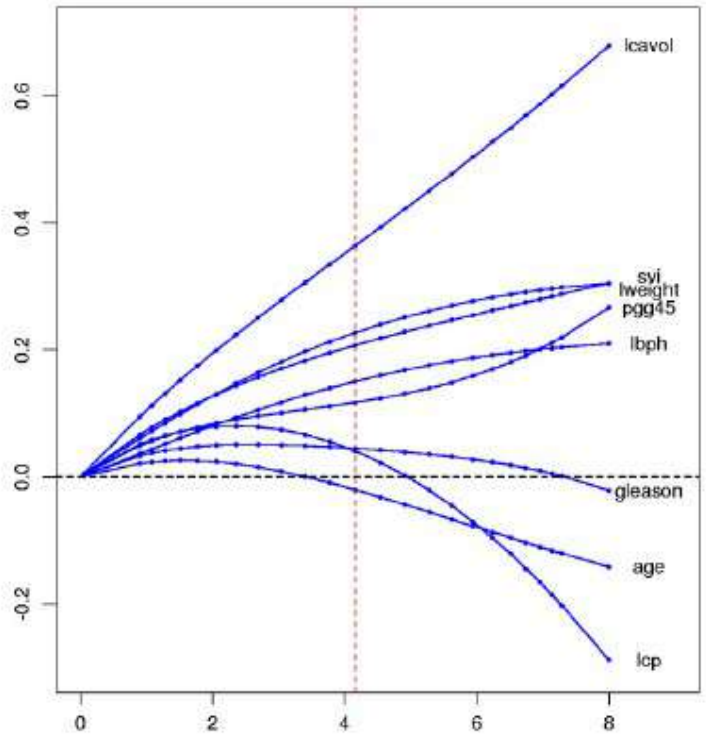
$$\sum_{i=1}^{\ell}\left(1 - M_i(w, w_0)\right)_+ + \mu \sum_{j=1}^{n} |w_j| \;\to\; \min_{w, w_0}$$

Отбор признаков с параметром селективности μ: чем больше μ, тем меньше признаков останется

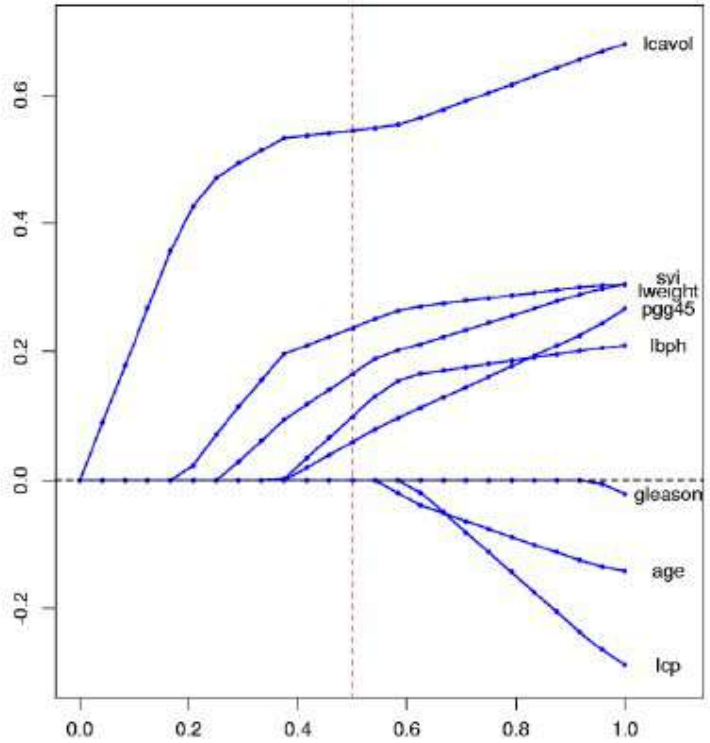
Сравнение L_2 и L_1 регуляризации

Зависимость весов w_j от коэффициента $\frac{1}{\mu}$

L_2 регуляризатор: $\mu \sum_j w_j^2$



L_1 регуляризатор: $\mu \sum_j |w_j|$

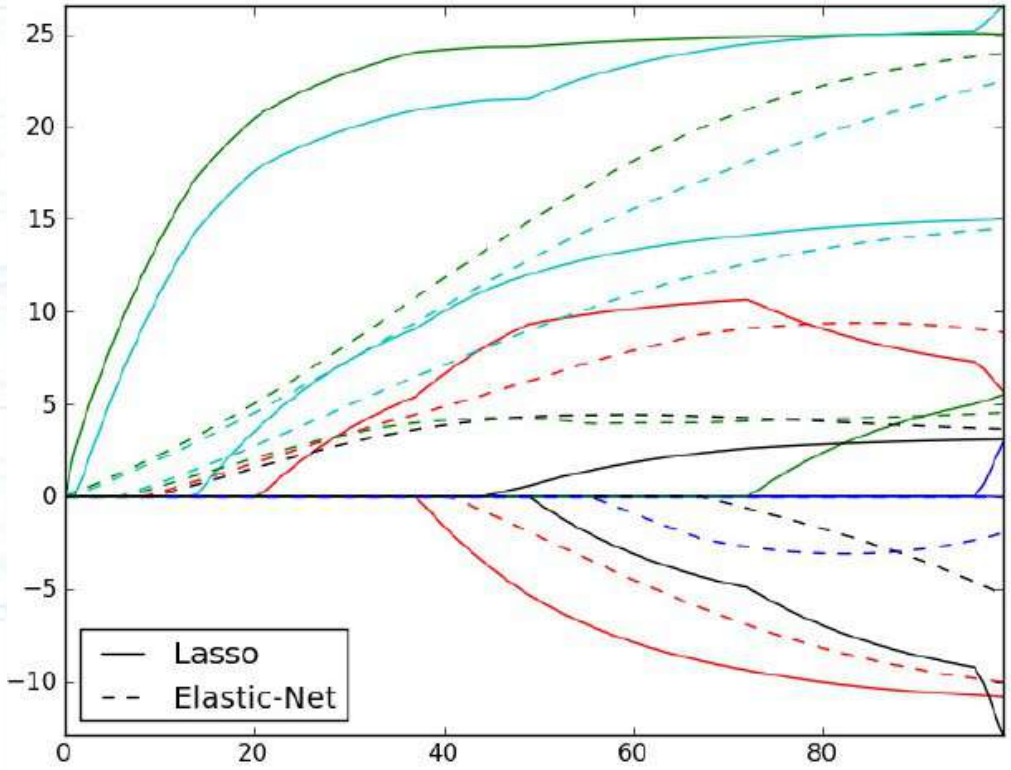


Задача из UCI: prostate cancer (диагностика рака)

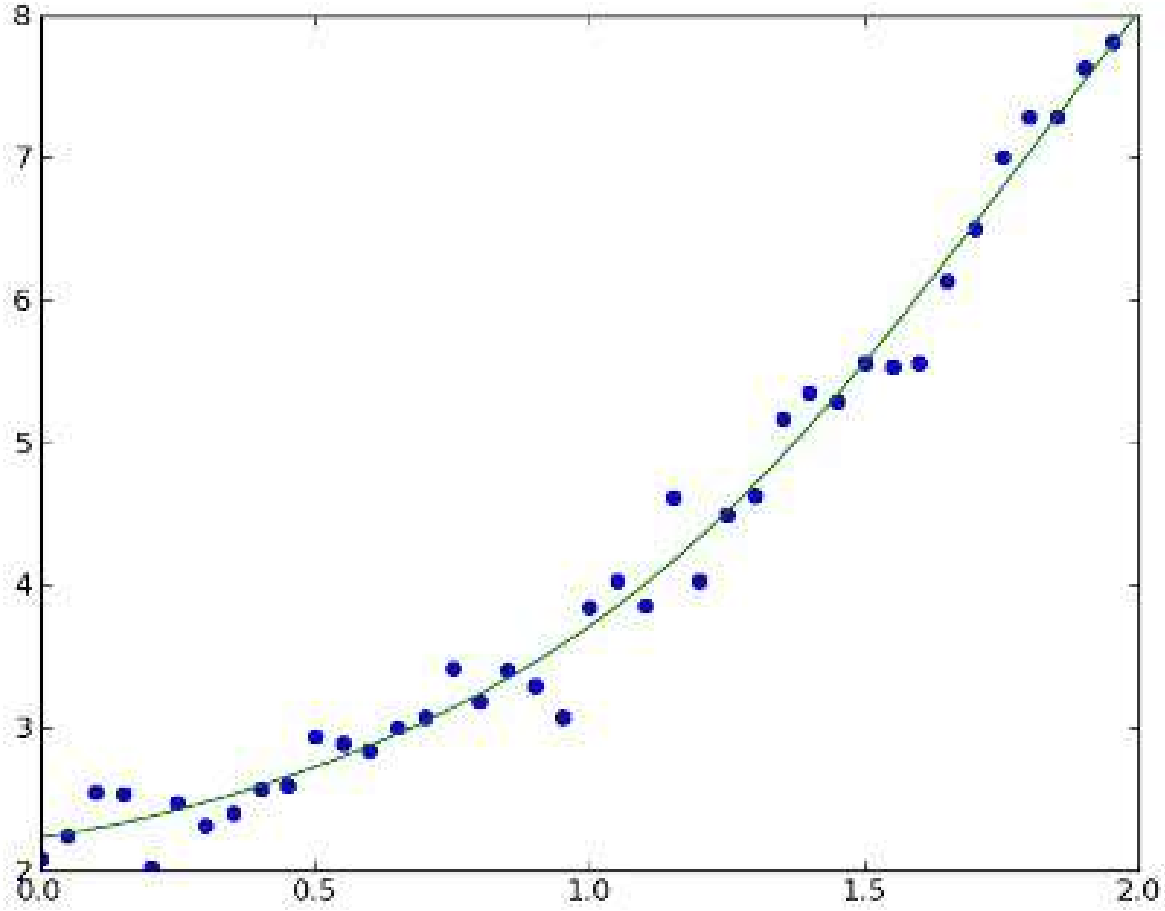
Doubly Regularized SVM (Elastic Net SVM)

$$C\sum_{i=1}^{\ell}\left(1 - M_i(w, w_0)\right)_+ + \mu\sum_{j=1}^{n}|w_j| + \frac{1}{2}\sum_{j=1}^{n}w_j^2 \;\to\; \min_{w,w_0}$$

Elastic Net менее жёстко отбирает признаки.
Зависимости весов w_j от коэффициента $\log\frac{1}{\mu}$:



Машинное обучение
Методы восстановления регрессии



https://yandexdataschool.ru/edu-process/courses/machine-learning

Содержание лекции

- Метод наименьших квадратов
- Геометрический смысл
- Регуляризация
- Сингулярное разложение
- Непараметрическая регрессия

Метод наименьших квадратов

- $X = R^n$, $Y = R$
- Модель: $a(x) = f(x, \alpha)$
- Метод наименьших квадратов (МНК):

$$Q(\alpha, X^\ell) = \sum_{i=1}^{\ell} w_i \left(f(x_i, \alpha) - y_i \right)^2 \to \min_{\alpha}$$

- w_i — вес, степень важности i-го объекта

Многомерная линейная регрессия

- $f_1(x), \ldots, f_n(x)$ — числовые признаки;
- Модель:

$$f(x,\alpha) = \sum_{j=1}^{n} \alpha_j f_j(x), \qquad \alpha \in \mathbb{R}^n$$

- Матричная форма:

$$\underset{\ell\times n}{F} = \begin{pmatrix} f_1(x_1) & \ldots & f_n(x_1) \\ \ldots & \ldots & \ldots \\ f_1(x_\ell) & \ldots & f_n(x_\ell) \end{pmatrix}, \quad \underset{\ell\times 1}{y} = \begin{pmatrix} y_1 \\ \ldots \\ y_\ell \end{pmatrix}, \quad \underset{n\times 1}{\alpha} = \begin{pmatrix} \alpha_1 \\ \ldots \\ \alpha_n \end{pmatrix}$$

$$Q(\alpha, X^\ell) = \sum_{i=1}^{\ell} \big(f(x_i,\alpha) - y_i\big)^2 = \|F\alpha - y\|^2 \to \min_\alpha$$

Нормальная система уравнений

- Необходимое условие минимума

$$\frac{\partial Q}{\partial \alpha}(\alpha) = 2F^{\intercal}(F\alpha - y) = 0$$

$$F^{\intercal}F\alpha = F^{\intercal}y$$

- где $F^{\intercal}F$ — ковариационная матрица n×n набора признаков $f_1, \ldots, f_n$
- Решение системы: $\alpha^* = (F^{\intercal}F)^{-1}F^{\intercal}y = F^{+}y$
- Значение функционала: $Q(\alpha^*) = \|P_F y - y\|^2$
  где P_F - проекционная матрица

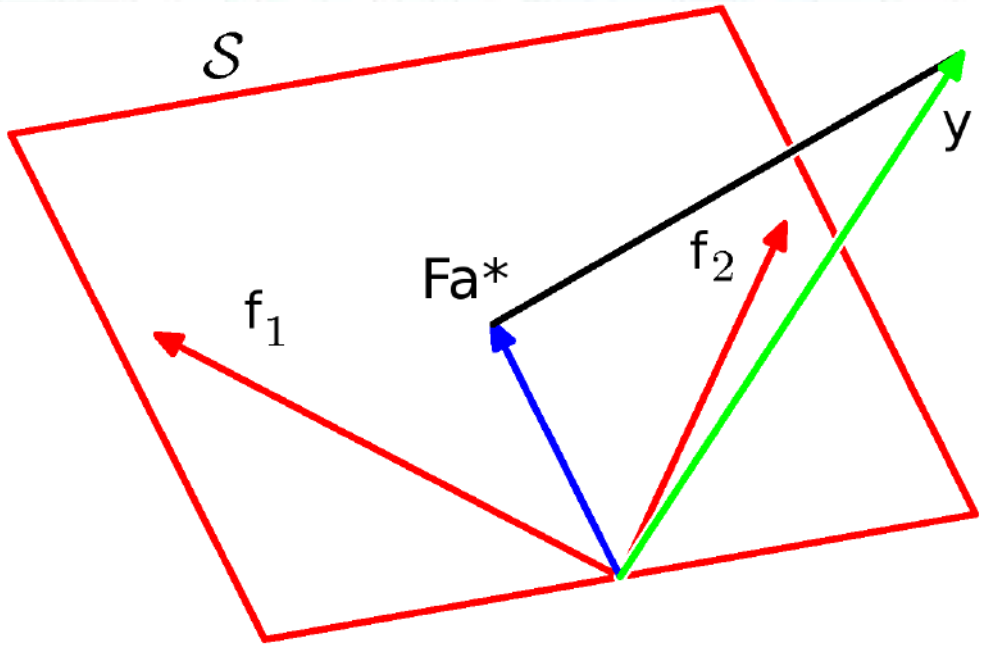
$$P_F = FF^{+} = F(F^{\intercal}F)^{-1}F^{\intercal}$$

Геометрический смысл

- Любой вектор вида $y = F\alpha$ – линейная комбинация признаков

$$\|F\alpha - y\|^2 \to \min_{\alpha}$$

- $F\alpha^*$ – аппроксимация вектора y с наименьшим квадратом тогда и только тогда, когда $F\alpha^*$ - проекция y на подпространство признаков



Вероятностный подход

- Модель данных с некоррелированным гауссовским шумом:

$$y(x_i) = f(x_i, \alpha) + \varepsilon_i, \quad \varepsilon_i \sim \mathcal{N}(0, \sigma_i^2), \quad i = 1, \ldots, \ell$$

- Принцип максимума правдоподобия:

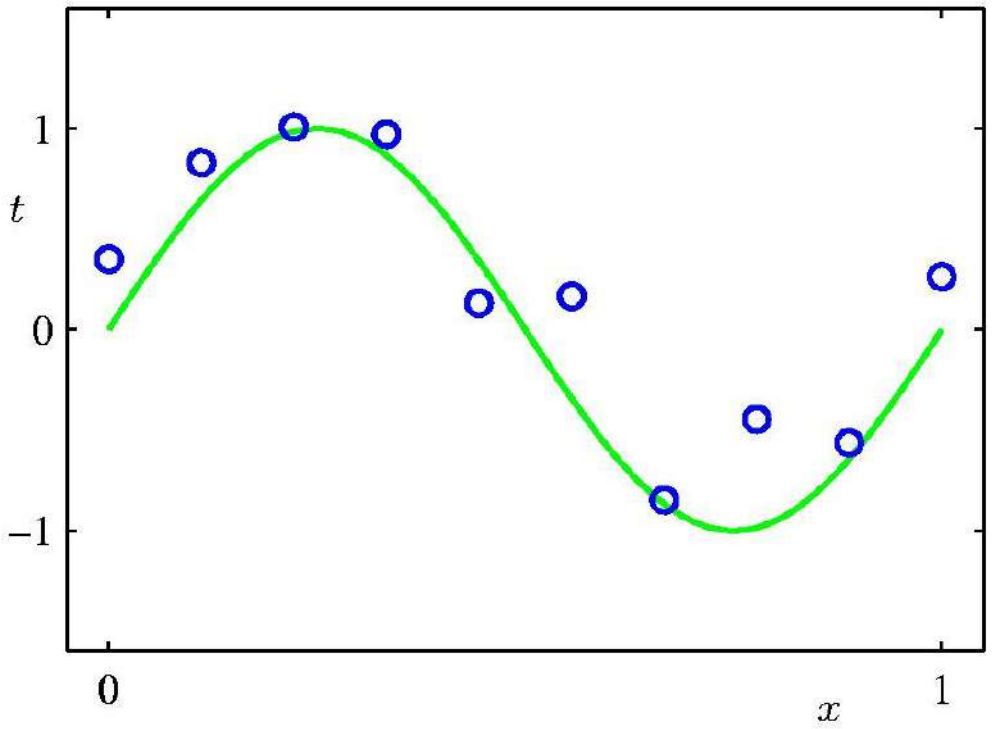
$$L(\varepsilon_1, \ldots, \varepsilon_\ell | \alpha) = \prod_{i=1}^{\ell} \frac{1}{\sigma_i \sqrt{2\pi}} \exp\left(-\frac{1}{2\sigma_i^2}\varepsilon_i^2\right) \to \max_\alpha$$

$$-\ln L(\varepsilon_1, \ldots, \varepsilon_\ell | \alpha) = \text{const}(\alpha) + \frac{1}{2}\sum_{i=1}^{\ell} \frac{1}{\sigma_i^2}\big(f(x_i, \alpha) - y_i\big)^2 \to \min_\alpha$$

- В итоге пришли к МНК

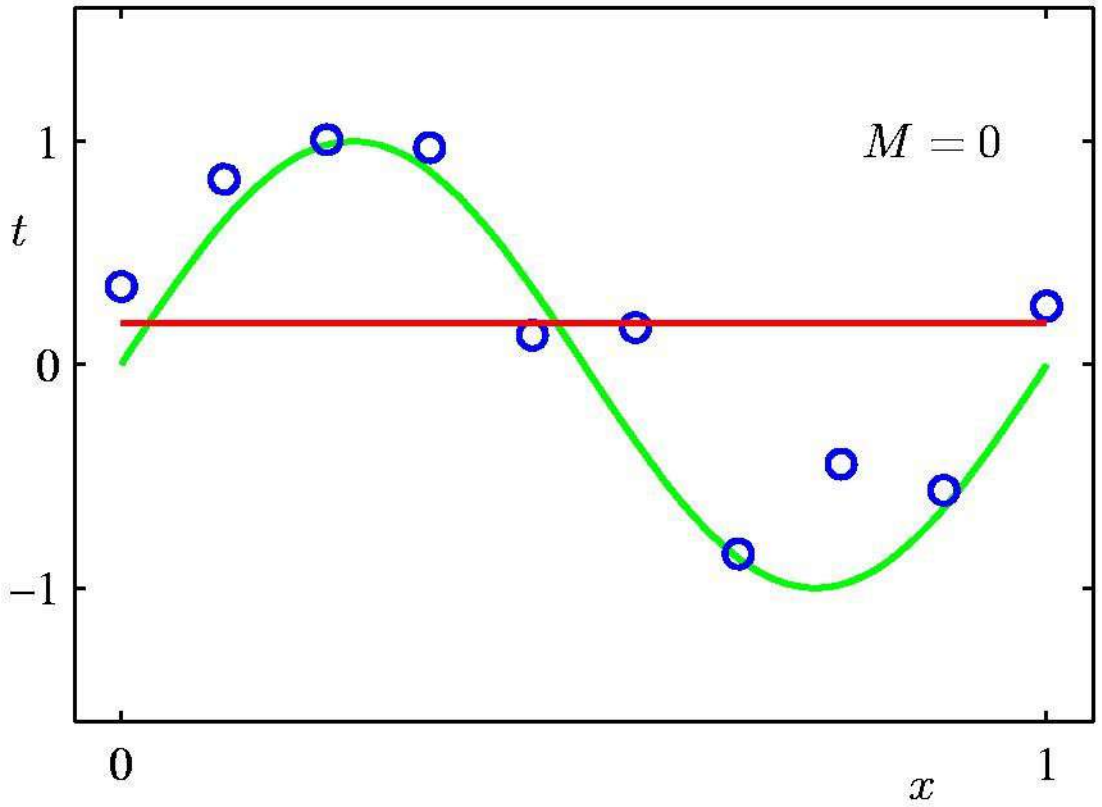
Пример – приближение многочленами

Данные: sin(x) + случайный шум

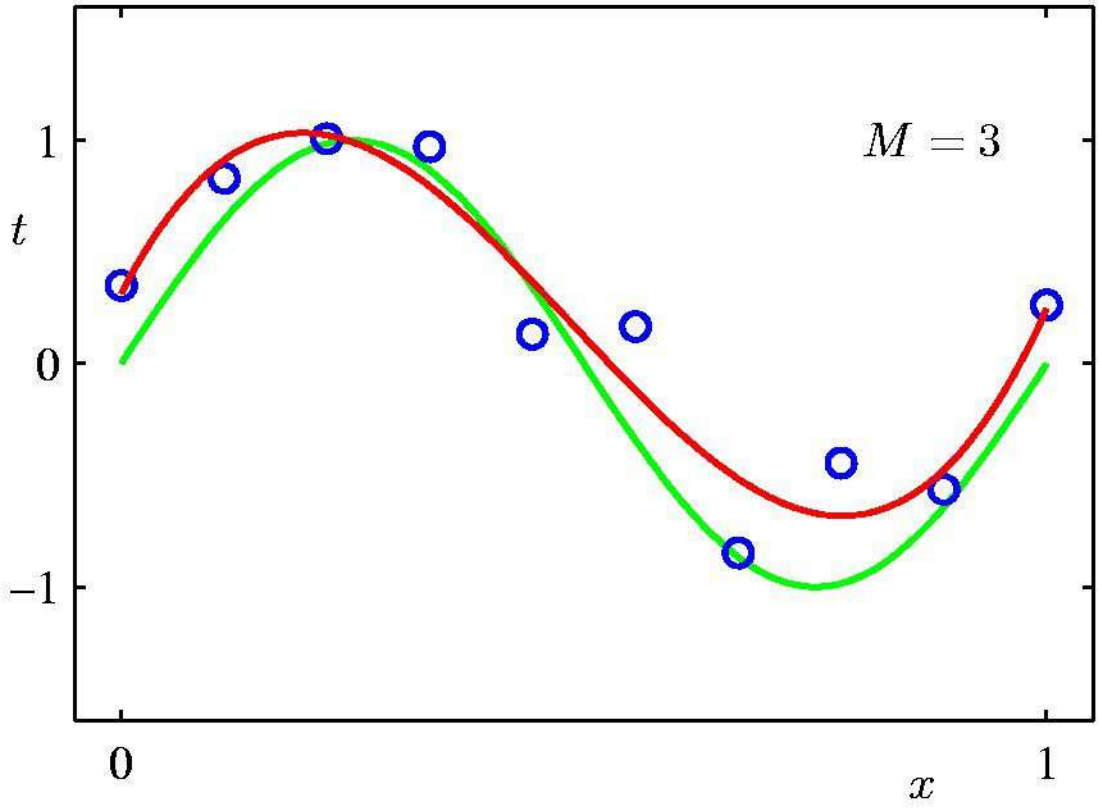


$$y(x, \mathbf{w}) = w_0 + w_1 x + w_2 x^2 + \ldots + w_M x^M = \sum_{j=0}^{M} w_j x^j$$

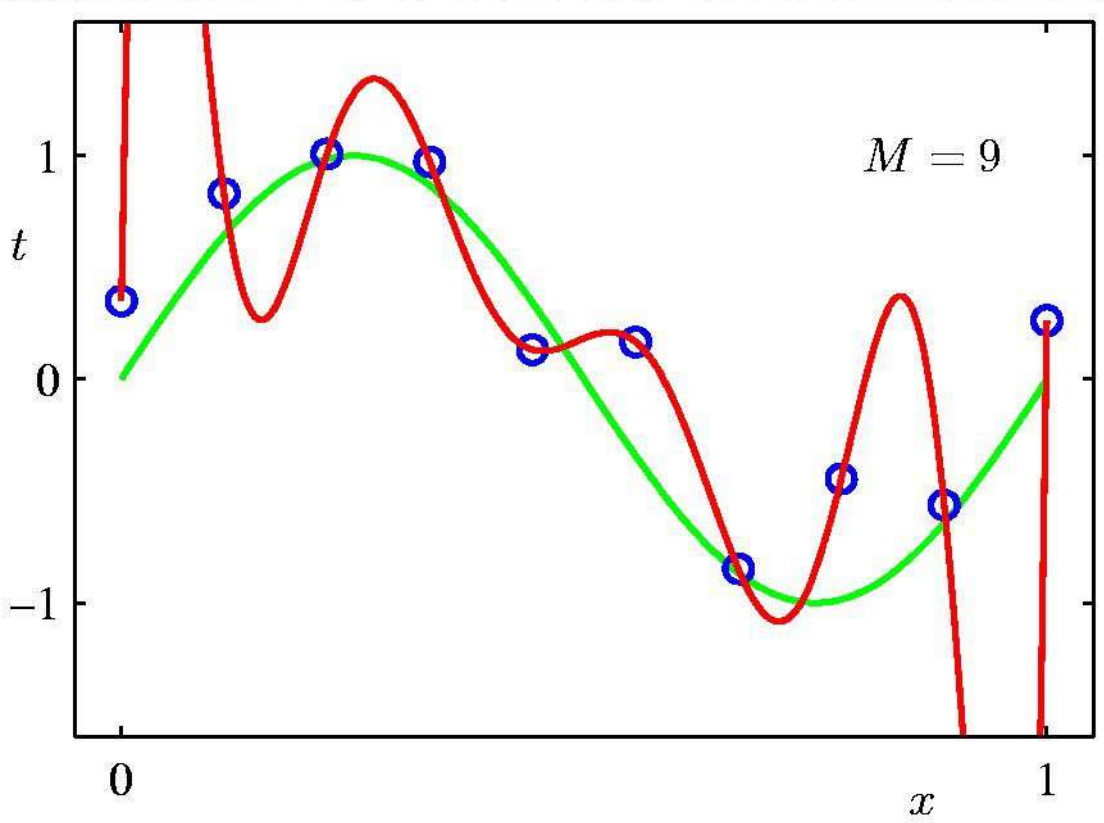
Многочлен степени 0



Многочлен степени 3



Многочлен степени 9



Коэффициенты многочленов

	$M=0$	$M=1$	$M=3$	$M=9$
$w_0^\star$	0.19	0.82	0.31	0.35
$w_1^\star$		-1.27	7.99	232.37
$w_2^\star$			-25.43	-5321.83
$w_3^\star$			17.37	48568.31
$w_4^\star$				-231639.30
$w_5^\star$				640042.26
$w_6^\star$				-1061800.52
$w_7^\star$				1042400.18
$w_8^\star$				-557682.99
$w_9^\star$				125201.43

В переобученном случае наблюдаются аномально большие коэффициенты многочлена. Выход - регуляризация

Гребневая регрессия

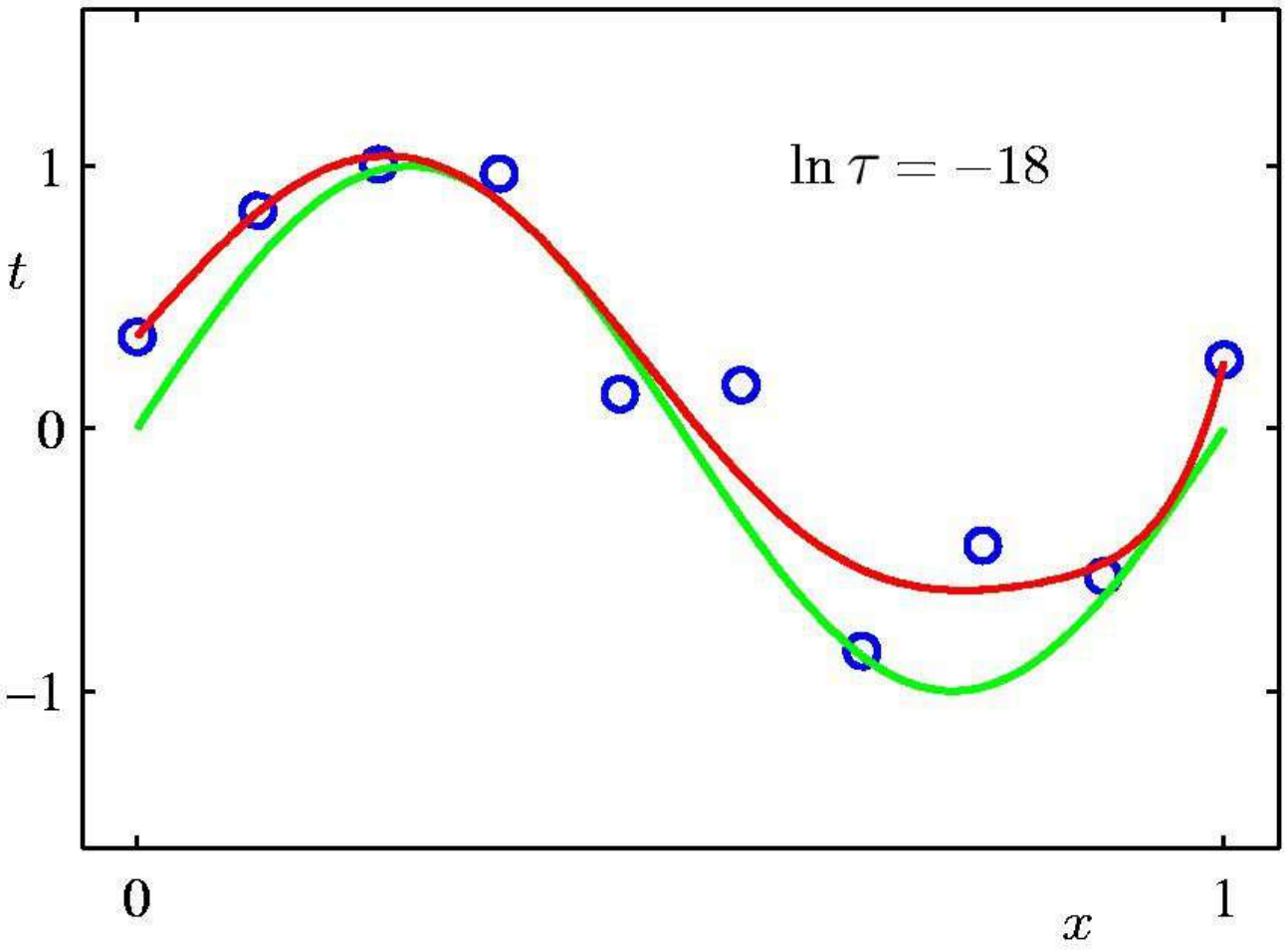
- Штраф за увеличение нормы вектора весов ||α|| :

$$Q_\tau(\alpha) = \|F\alpha - y\|^2 + \frac{1}{\sigma}\|\alpha\|^2 \qquad \tau = \frac{1}{\sigma}$$

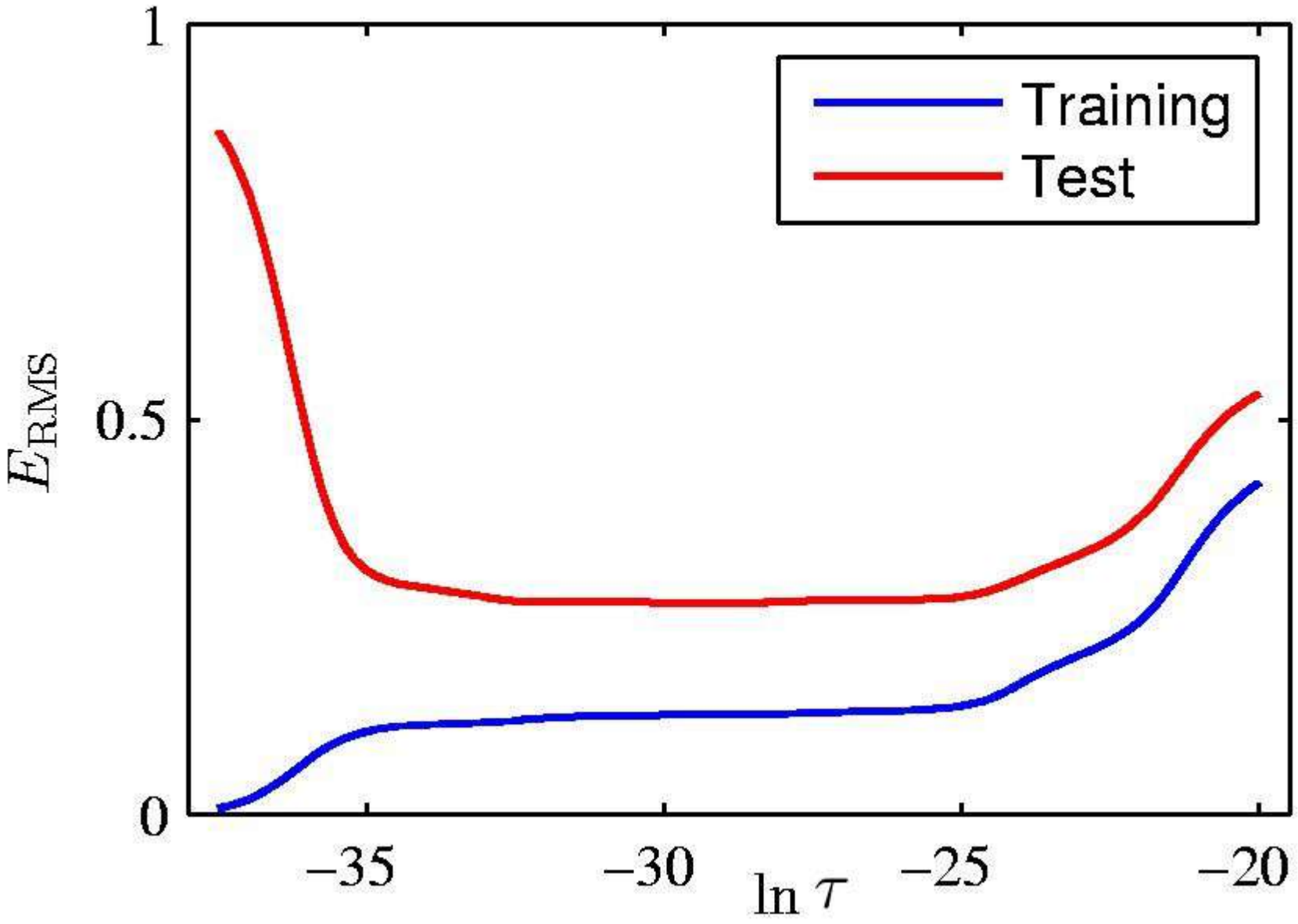
- Модифицированное МНК-решение (τI_n — «гребень»):

$$\alpha_\tau^* = (F^\intercal F + \tau I_n)^{-1} F^\intercal y$$

Многочлен степени 9 с регуляризацией



Гребневая регрессия



Сингулярное разложение

Произвольная $\ell \times n$-матрица представима в виде *сингулярного разложения* (singular value decomposition, SVD):

$$F = VDU^{\mathsf{T}}.$$

Основные свойства сингулярного разложения:

1. $\ell \times n$-матрица $V = (v_1, \ldots, v_n)$ ортогональна, $V^{\mathsf{T}}V = I_n$, столбцы v_j — собственные векторы матрицы FF^{T};
2. $n \times n$-матрица $U = (u_1, \ldots, u_n)$ ортогональна, $U^{\mathsf{T}}U = I_n$, столбцы u_j — собственные векторы матрицы $F^{\mathsf{T}}F$;
3. $n \times n$-матрица D диагональна, $D = \operatorname{diag}\bigl(\sqrt{\lambda_1}, \ldots, \sqrt{\lambda_n}\bigr)$, $\lambda_j \geqslant 0$ — собственные значения матриц $F^{\mathsf{T}}F$ и FF^{T}.

Решение МНК через сингулярное разложение

$$\alpha^* = (F^\intercal F)^{-1} F^\intercal y = F^+ y$$

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

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

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

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

Проблема мультиколлинеарности

- Число обусловленности n×n-матрицы Σ:

$$\mu(\Sigma) = \|\Sigma\|\|\Sigma^{-1}\| = \frac{\max\limits_{u:\ \|u\|=1} \|\Sigma u\|}{\min\limits_{u:\ \|u\|=1} \|\Sigma u\|} = \frac{\lambda_{\max}}{\lambda_{\min}}$$

- При умножении обратной матрицы на вектор, $z = \Sigma^{-1} u$, относительная погрешность усиливается в μ(Σ) раз:

$$\frac{\|\delta z\|}{\|z\|} \leqslant \mu(\Sigma) \frac{\|\delta u\|}{\|u\|}$$

Проблема мультиколлинеарности

- Если матрица $\Sigma = F^T F$ плохо обусловлена, то:
  - решение становится неустойчивым и неинтерпретируемым, $\|\alpha^{\in}\|$ велико;
  - возникает переобучение:
    на обучении $Q(\alpha^*, X^\ell) = \|F\alpha^* - y\|^2$ мало
    на контроле $Q(\alpha^*, X^k) = \|F'\alpha^* - y'\|^2$ велико
- Стратегии устранения мультиколлинеарности и переобучения:
  - регуляризация
  - отбор признаков
  - преобразование признаков

Регуляризация с точки зрения SVD-разложения

$$\alpha_\tau^* = (F^\intercal F + \tau I_n)^{-1} F^\intercal y$$

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

Без регуляризации:

$$\alpha^* = F^+ y = U D^{-1} V^\intercal y = \sum_{j=1}^{n} \frac{1}{\sqrt{\lambda_j}} u_j (v_j^\intercal y)$$

Отбор признаков

- LASSO — Least Absolute Shrinkage and Selection Operator

$$\begin{cases} Q(\alpha) = \|F\alpha - y\|^2 \to \min\limits_{\alpha}; \\ \sum\limits_{j=1}^{n} |\alpha_j| \leqslant \varkappa; \end{cases}$$

- Чем меньше $\varkappa$, тем больше нулевых α_j

Метод главных компонент (PCA)

- $f_1(x), \ldots, f_n(x)$ — исходные числовые признаки;
- $g_1(x), \ldots, g_m(x)$ — новые числовые признаки, m<n;
- Требование: старые признаки должны линейно восстанавливаться по новым:

$$\hat{f}_j(x) = \sum_{s=1}^{m} g_s(x) u_{js}, \quad j = 1, \ldots, n, \quad \forall x \in X$$

как можно точнее на обучающей выборке:

$$\sum_{i=1}^{\ell} \sum_{j=1}^{n} \left(\hat{f}_j(x_i) - f_j(x_i)\right)^2 \to \min_{\{g_s(x_i)\},\{u_{js}\}}$$

Постановка задачи PCA в матричной форме

$$\hat{F} = GU^{\mathsf{T}} \overset{\text{хотим}}{\approx} F$$

Найти: и новые признаки G , и преобразование U:

$$\sum_{i=1}^{\ell}\sum_{j=1}^{n}\left(\hat{f}_j(x_i) - f_j(x_i)\right)^2 = \left\|GU^{\mathsf{T}} - F\right\|^2 \to \min_{G,U}$$

Теорема

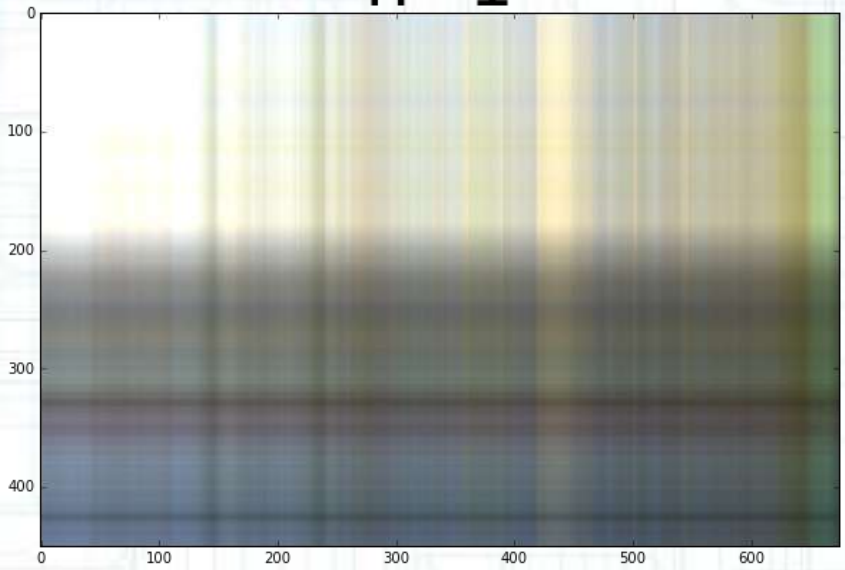
Если $m \leqslant \operatorname{rk} F$, *то минимум* $\left\| GU^{\mathrm{T}} - F \right\|^2$ *достигается, когда столбцы* U *— это с.в. матрицы* $F^{\mathrm{T}}F$, *соответствующие* m *максимальным с.з.* $\lambda_1, \ldots, \lambda_m$, *а матрица* $G = FU$.

При этом:

1. *матрица* U *ортонормирована:* $U^{\mathrm{T}}U = I_m$;
2. *матрица* G *ортогональна:* $G^{\mathrm{T}}G = \Lambda = \operatorname{diag}(\lambda_1, \ldots, \lambda_m)$;
3. $U\Lambda = F^{\mathrm{T}}FU$; $G\Lambda = FF^{\mathrm{T}}G$;
4. $\left\| GU^{\mathrm{T}} - F \right\|^2 = \|F\|^2 - \operatorname{tr}\Lambda = \sum\limits_{j=m+1}^{n} \lambda_j$.

Применение SVD к сжатию изображений

n=1



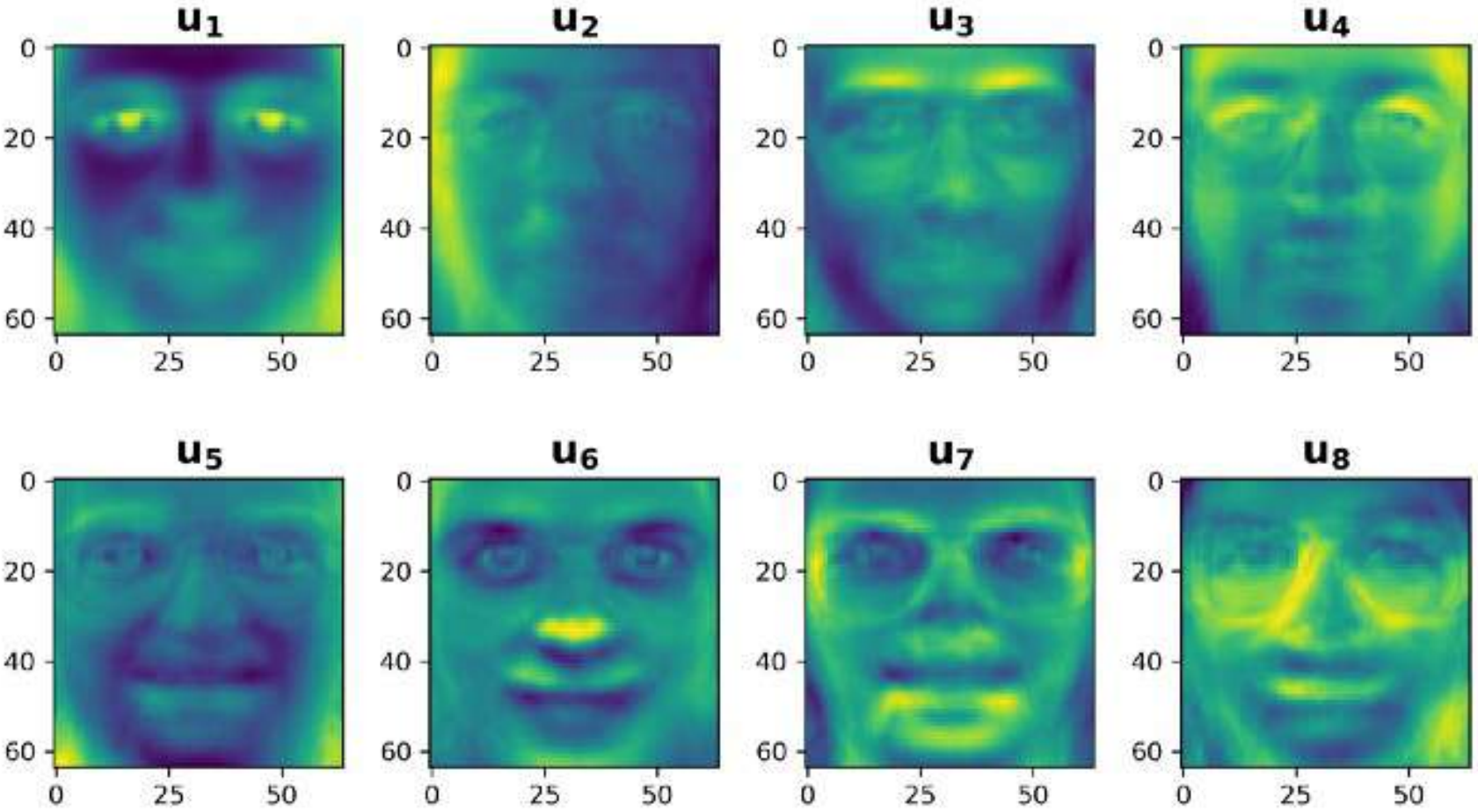
n=10

n=30

n=100

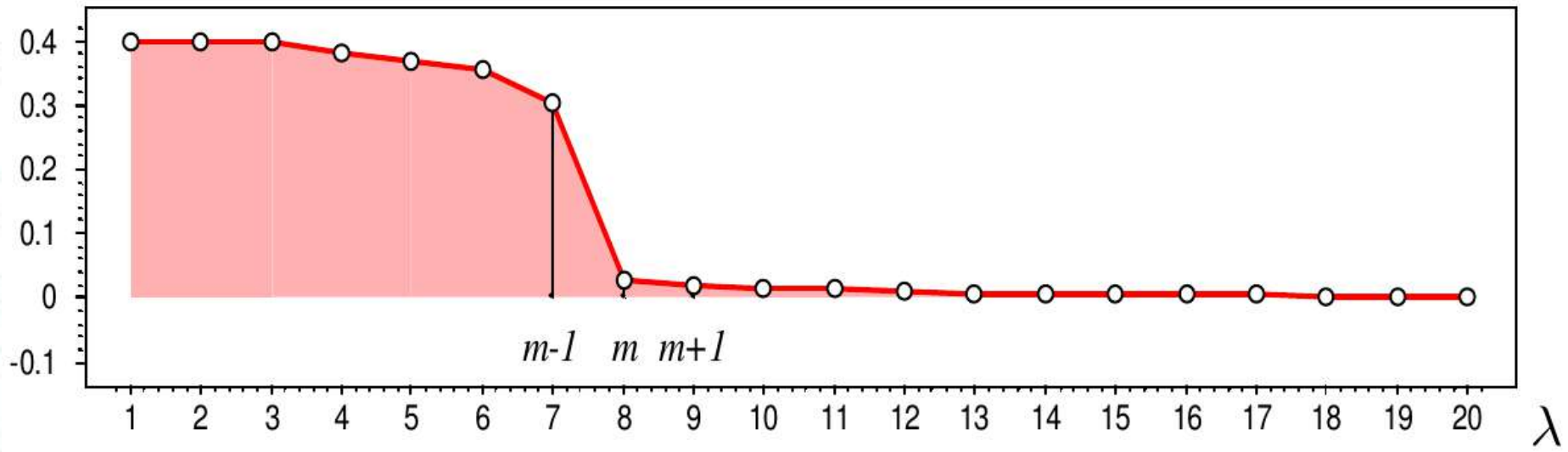
Главные компоненты датасета Olivetti faces

Показывают ортогональные направления, вдоль которых лица датасета меняются сильнее всего



Сколько главных компонент брать?

- Критерий “крутого склона”:



Непараметрическая регрессия

- Обычная задача МНК:

$$Q(\alpha, X^\ell) = \sum_{i=1}^{\ell} w_i \left(f(x_i, \alpha) - y_i \right)^2 \to \min_\alpha$$

- Приближение константой f (x, α) = α в окрестности x ∈ X

$$Q(\alpha; X^\ell) = \sum_{i=1}^{\ell} w_i(x) \left(\alpha - y_i \right)^2 \to \min_{\alpha \in \mathbb{R}}$$

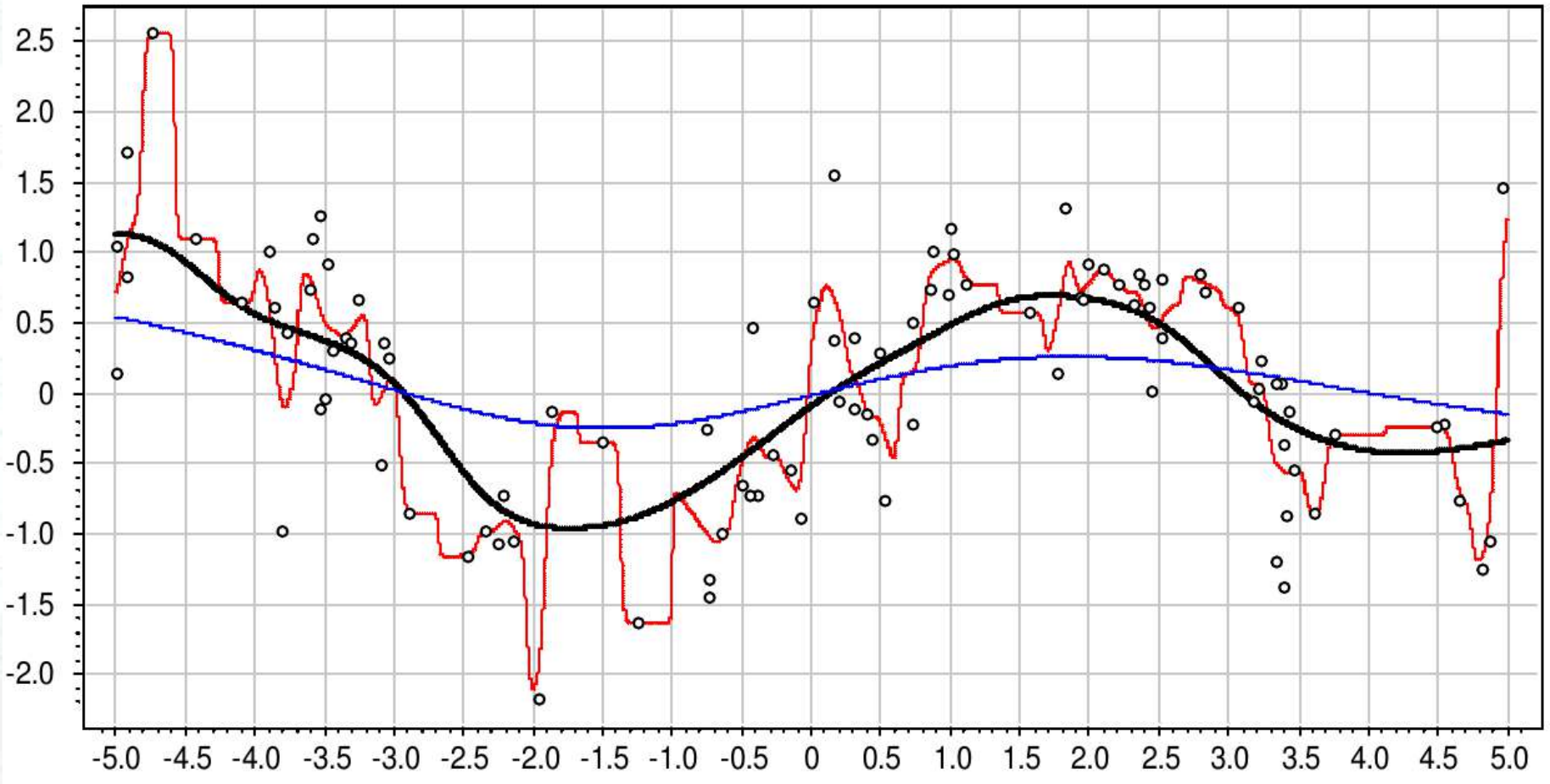
$w_i(x) = K\left(\frac{\rho(x, x_i)}{h}\right)$ - веса объектов x_i относительно x;

Формула ядерного сглаживания Надарая–Ватсона

$$a_h(x; X^\ell) = \frac{\sum_{i=1}^{\ell} y_i w_i(x)}{\sum_{i=1}^{\ell} w_i(x)} = \frac{\sum_{i=1}^{\ell} y_i K\left(\frac{\rho(x,x_i)}{h}\right)}{\sum_{i=1}^{\ell} K\left(\frac{\rho(x,x_i)}{h}\right)}$$

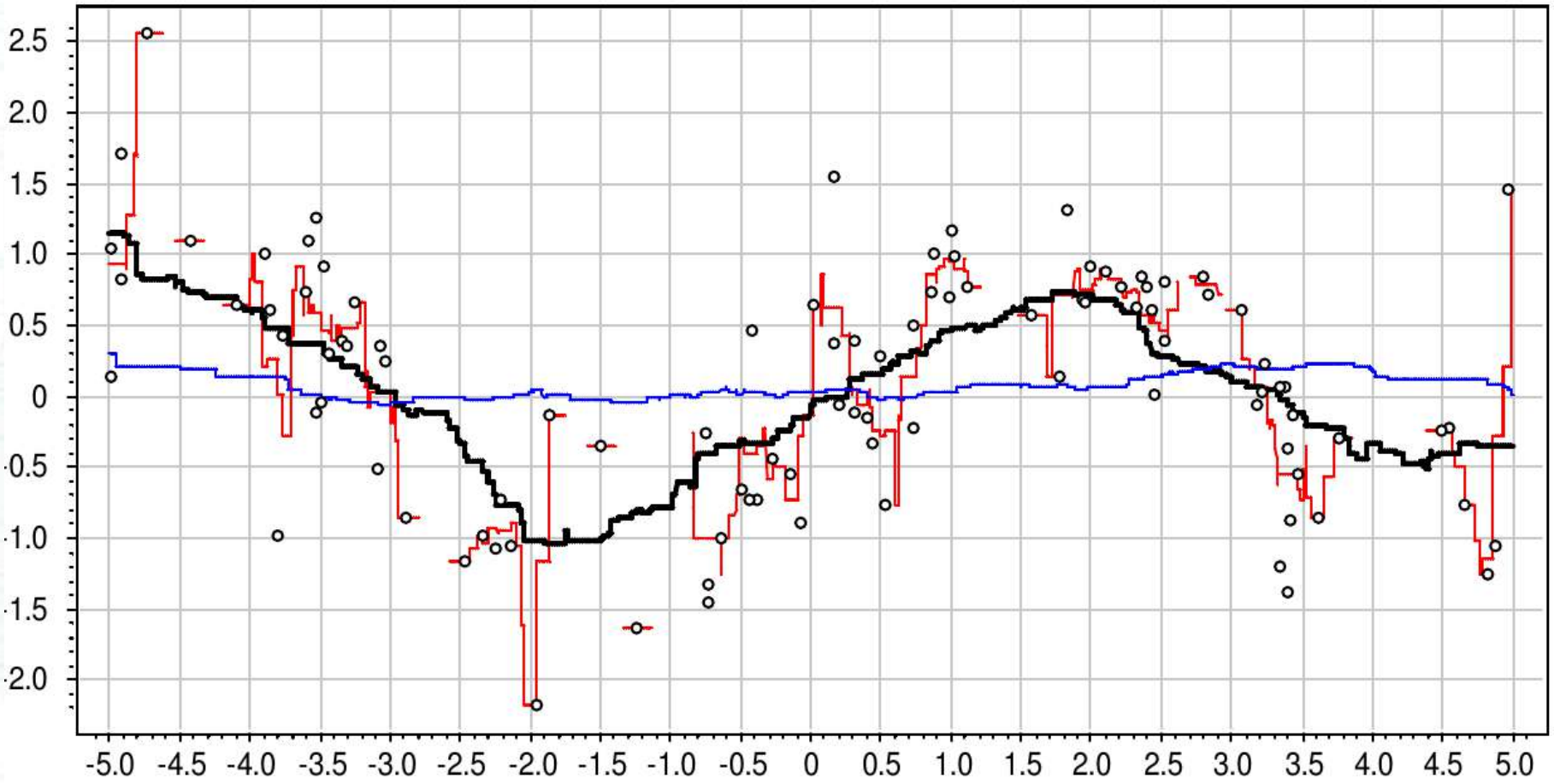
Выбор ширины окна и ядра

- $h \in \{0.1, 1.0, 3.0\}$, гауссовское ядро



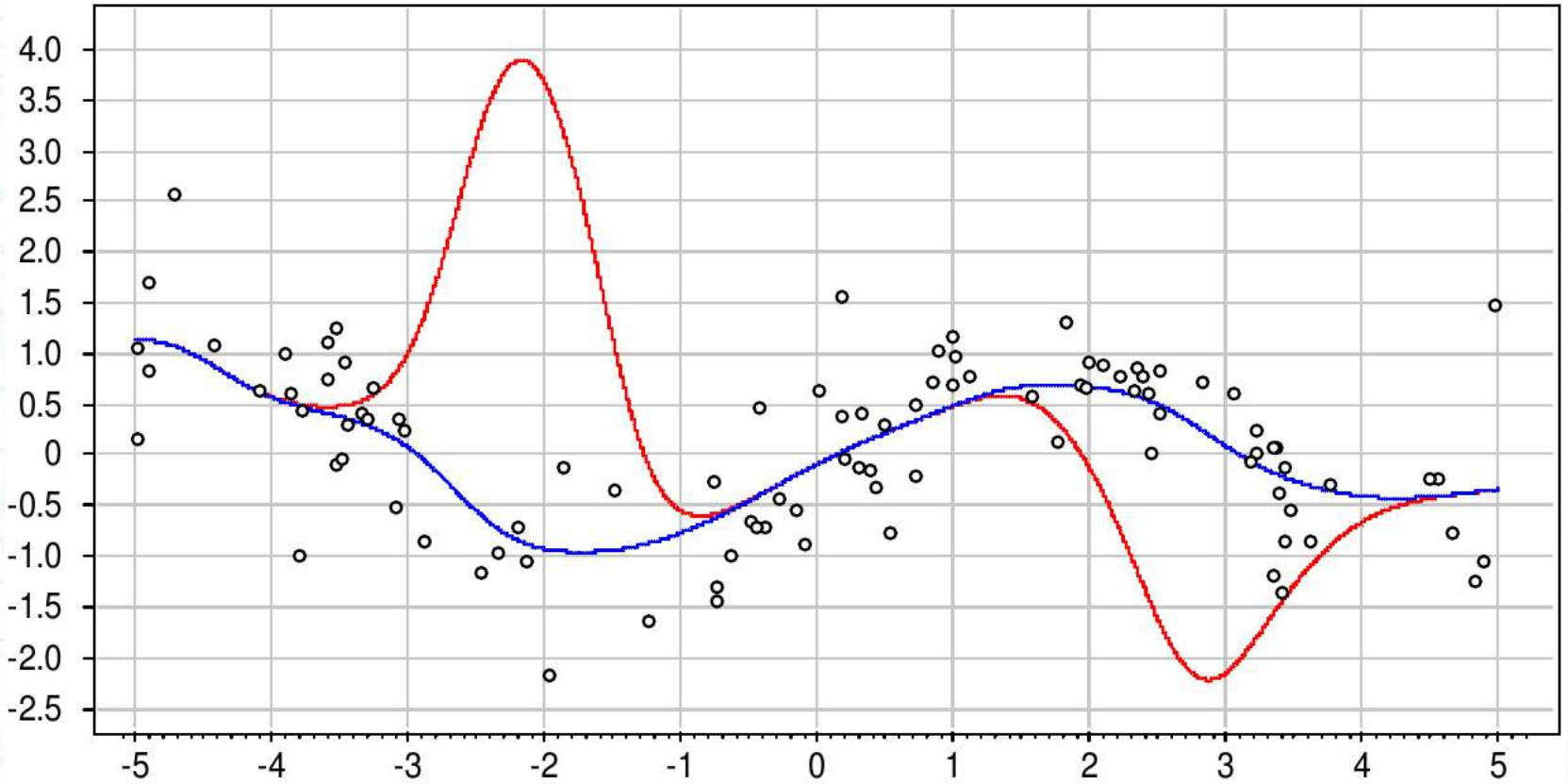
Выбор ширины окна и ядра

- $h \in \{0.1, 1.0, 3.0\}$, прямоугольное ядро



Проблема выбросов

- **l** = 100, h = 1.0, гауссовское ядро $K(r) = \exp(-2r^2)$
- Две точки - выбросы с ординатами 40 и −40
- Синяя кривая — выбросов нет



Локально взвешенное сглаживание

- Основная идея: чем больше величина ошибки $\varepsilon_i = \left|a_h\left(x_i; X^\ell \setminus \{x_i\}\right) - y_i\right|$ тем больше прецедент (x_i, y_i) похож на выброс, тем меньше должен быть его вес $w_i(x)$.
- Эвристика: домножить веса $w_i(x)$ на коэффициенты $\gamma_i = \tilde{K}(\varepsilon_i)$ где $\tilde{K}$ — ещё одно ядро
- Рекомендация: квартическое ядро

$$\tilde{K}(\varepsilon) = K_Q\left(\frac{\varepsilon}{6\,\mathrm{med}\{\varepsilon_i\}}\right)$$

где $\mathrm{med}\{\varepsilon_i\}$ — медиана вариационного ряда ошибок.

Алгоритм LOWESS (LOcally WEighted Scatter plot Smoothing)

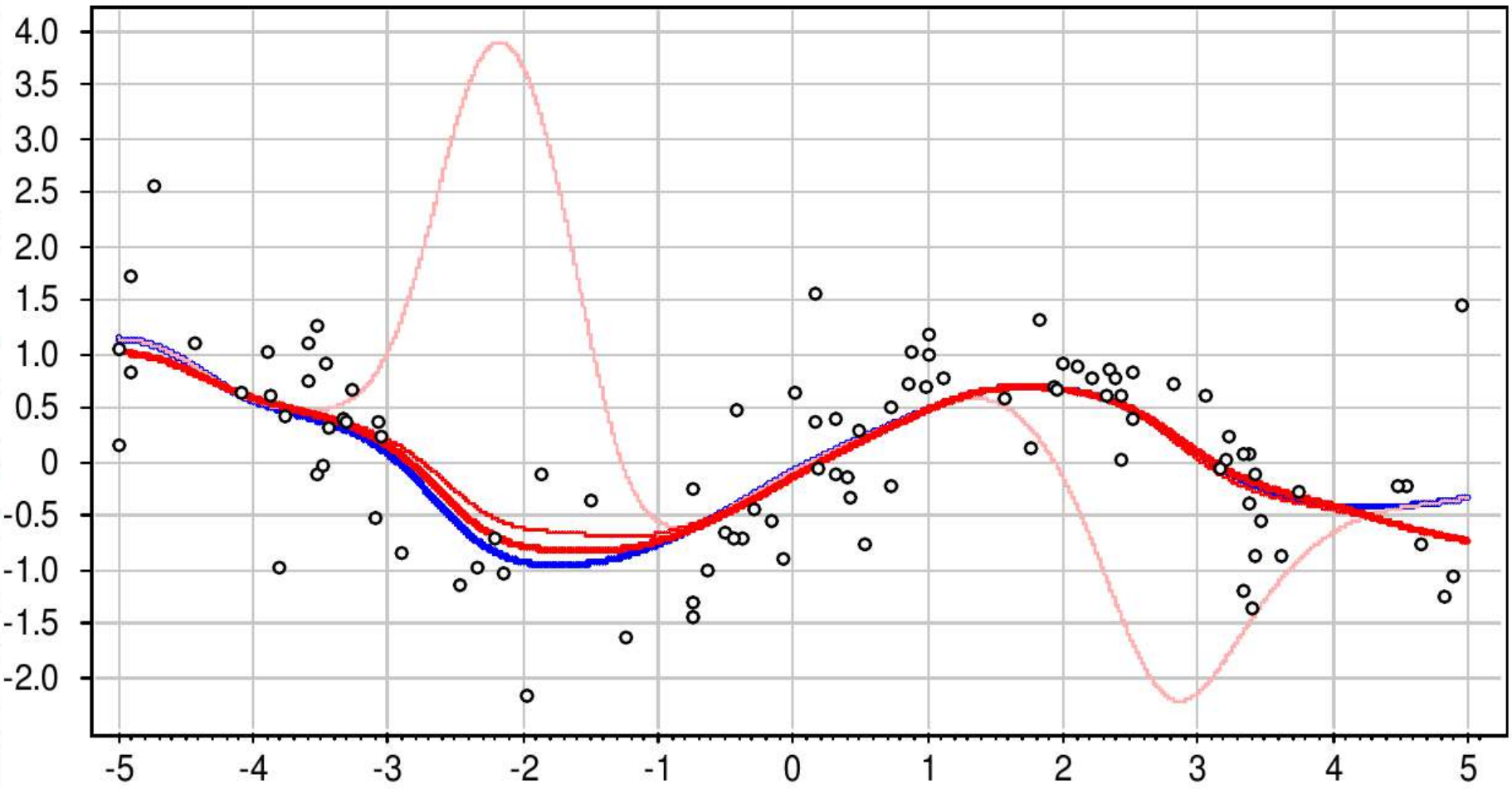
Вход: X^ℓ — обучающая выборка;
Выход: коэффициенты γ_i, $i = 1, \ldots, \ell$;

1: инициализация: $\gamma_i := 1$, $i = 1, \ldots, \ell$;
2: **повторять**
3: **для всех** объектов $i = 1, \ldots, \ell$
4: вычислить оценки скользящего контроля:

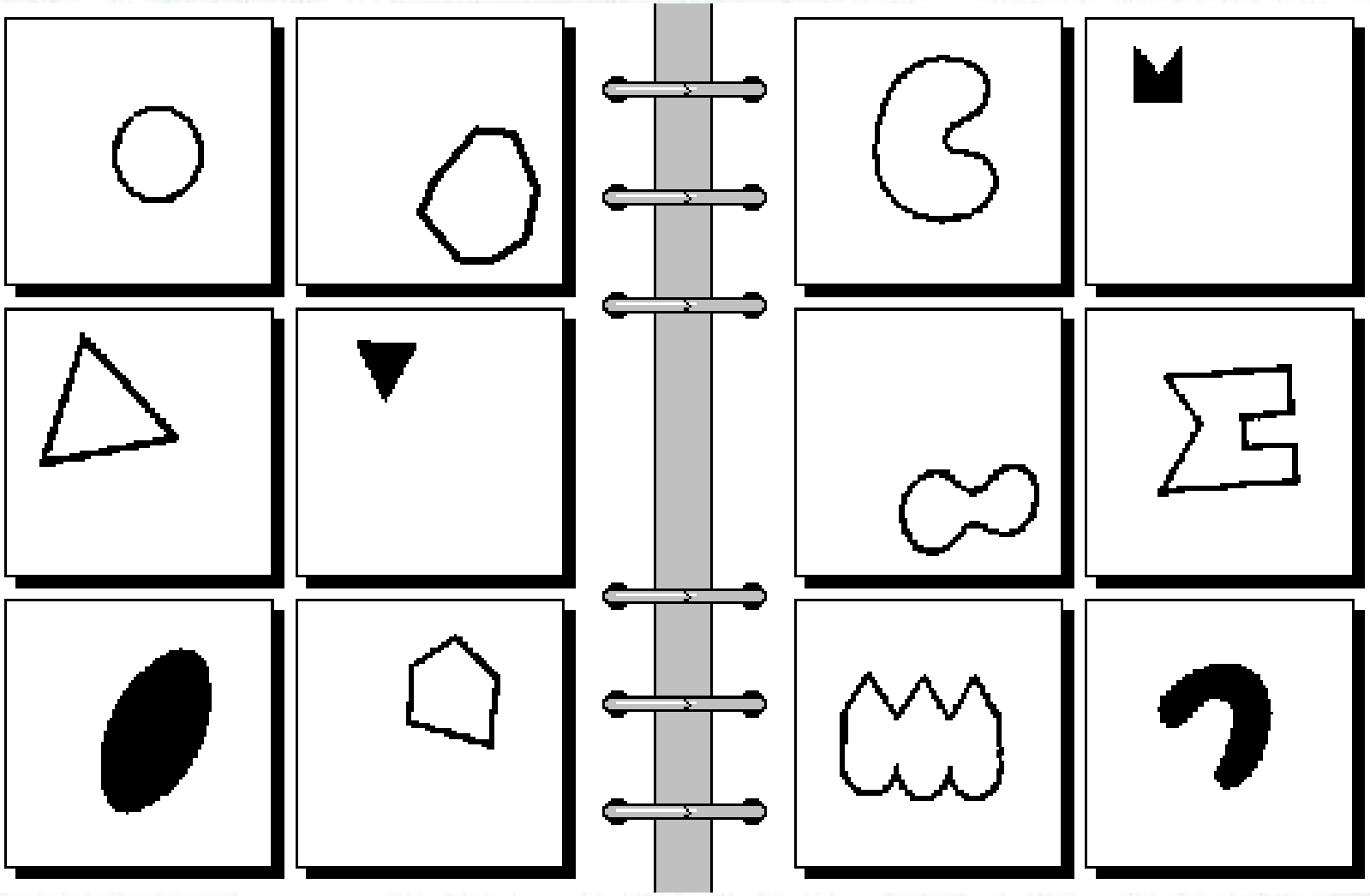
$$a_i := a_h\left(x_i; X^\ell \setminus \{x_i\}\right) = \frac{\sum\limits_{j=1,\, j\neq i}^{\ell} y_j \gamma_j K\left(\frac{\rho(x_i,x_j)}{h(x_i)}\right)}{\sum\limits_{j=1,\, j\neq i}^{\ell} \gamma_j K\left(\frac{\rho(x_i,x_j)}{h(x_i)}\right)};$$

5: **для всех** объектов $i = 1, \ldots, \ell$
6: $\gamma_i := \tilde{K}\left(|a_i - y_i|\right)$;
7: **пока** коэффициенты γ_i не стабилизируются;

Пример работы LOWESS



Машинное обучение
Логические методы классификации



https://yandexdataschool.ru/edu-process/courses/machine-learning

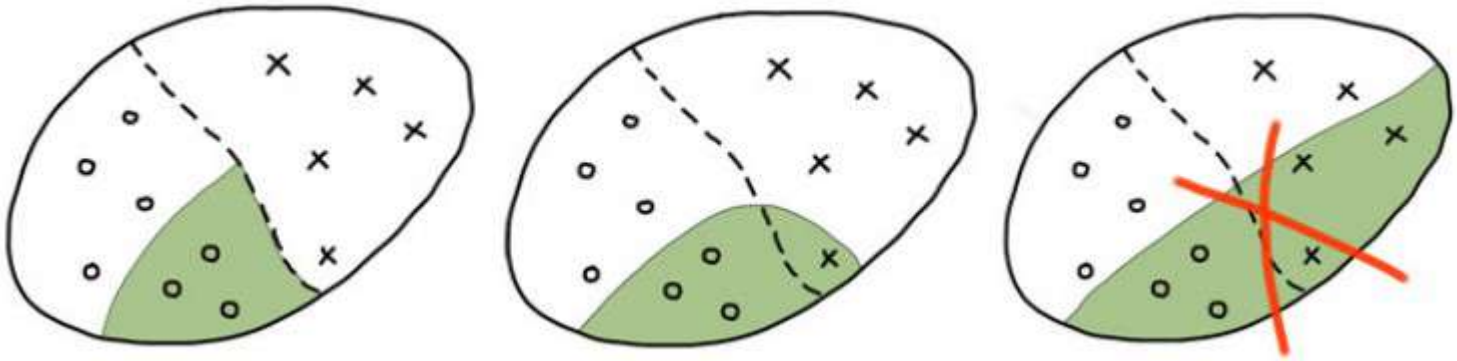
Содержание лекции

- Понятие закономерности
- Критерий качества закономерностей
- Поиск закономерностей
- Алгоритмы классификации на основе логических закономерностей

Понятие закономерности

- Предикат R: X → {0,1} – закономерность, если он выделяет (R(x)=1) достаточно много объектов одного класса C и практически не выделяет объектов других классов

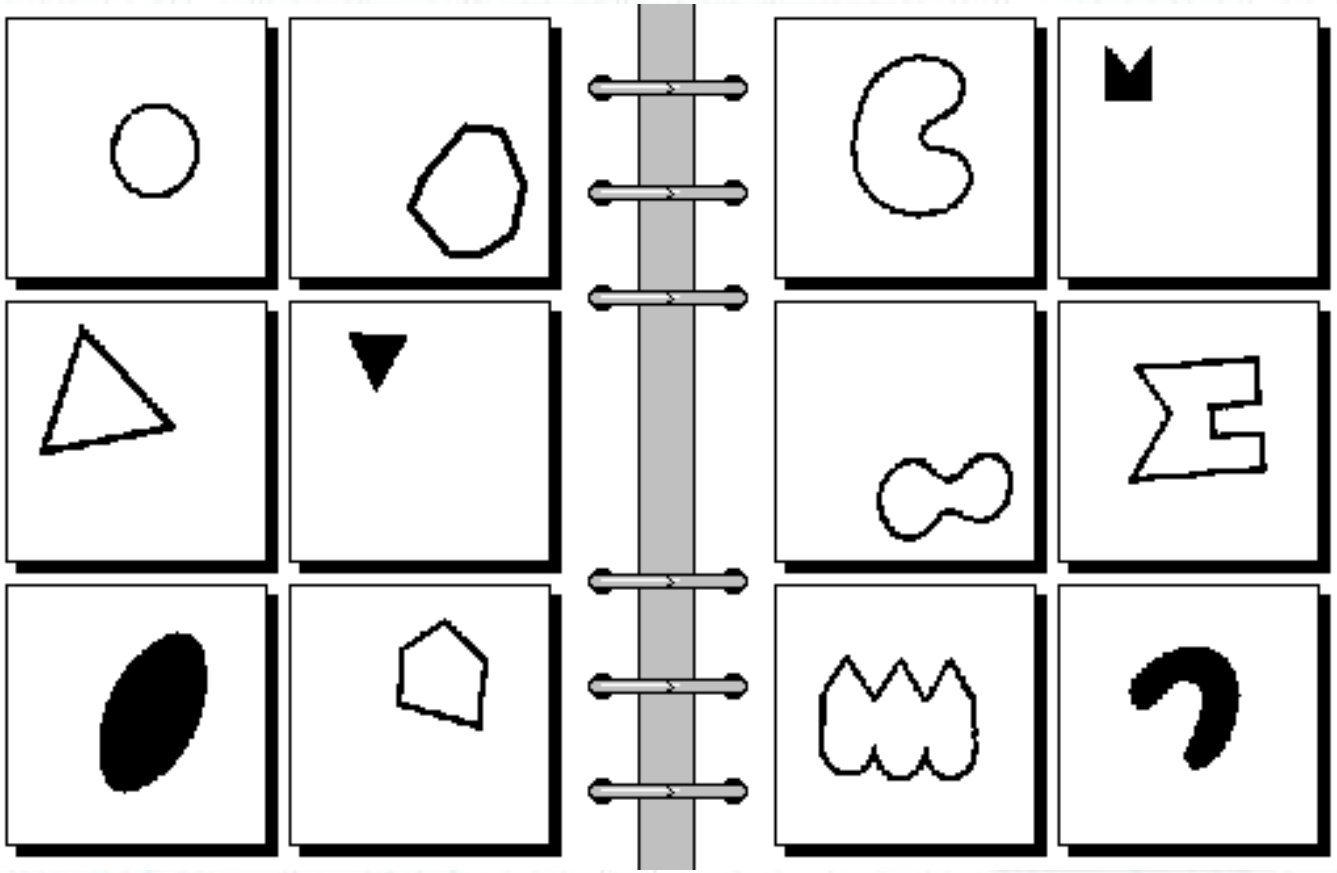
$$p_c(R) = \#\{x_i : R(x_i)=1 \text{ и } y_i=c\} \to \max;$$
$$n_c(R) = \#\{x_i : R(x_i)=1 \text{ и } y_i\neq c\} \to \min;$$



Примеры

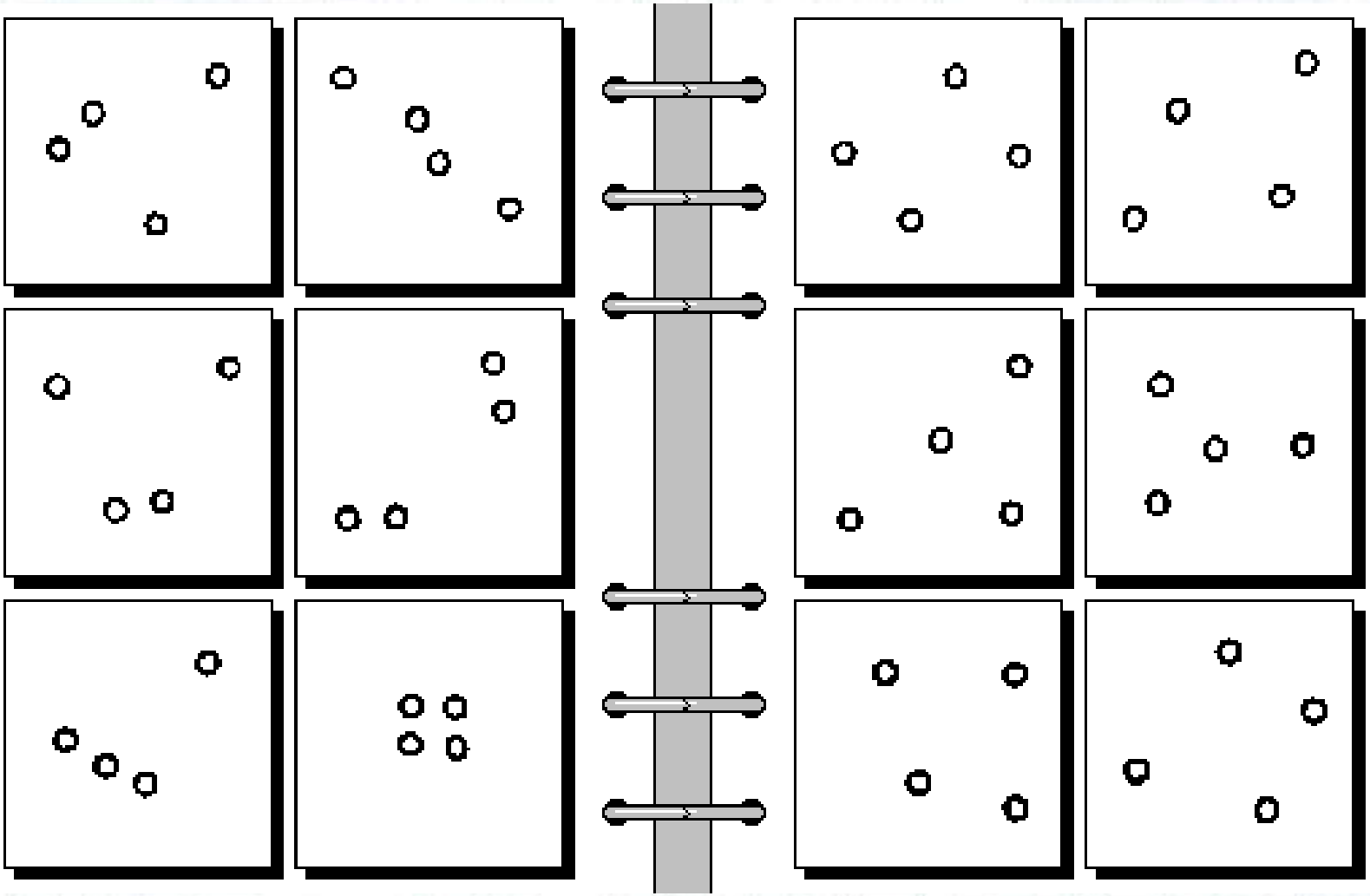
- Если «возраст > 60» и «пациент ранее перенёс инфаркт», то операцию не делать, риск отрицательного исхода 60%.
- Если «в анкете указан домашний телефон» и «зарплата > $2000» и «сумма кредита < $5000» то кредит можно выдать, риск дефолта 5%.

Тесты М.М.Бонгарда

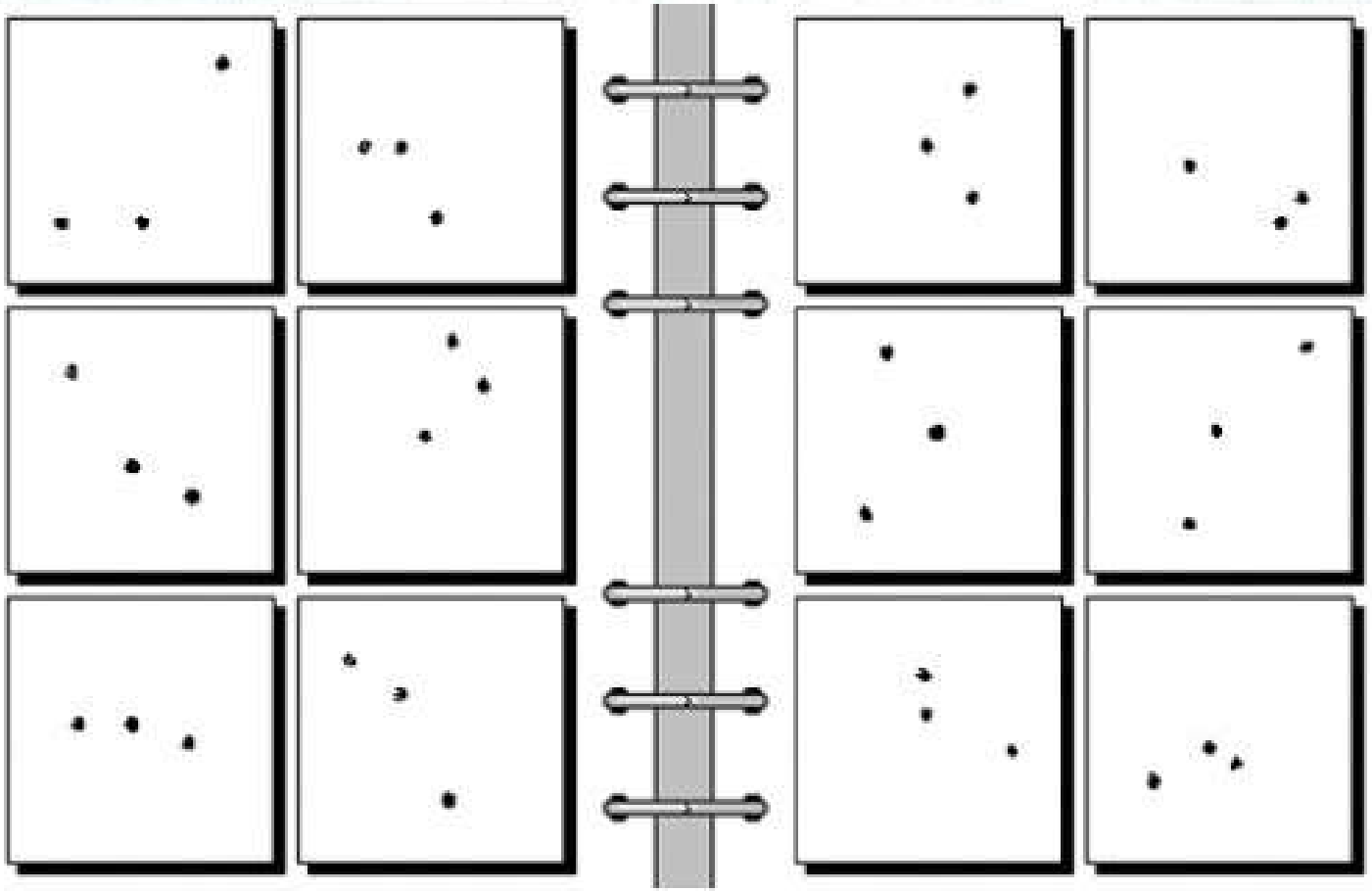


Этот тип головоломки изобрёл выдающийся русский кибернетик, основоположник теории распознавания образов Михаил Моисеевич Бонгард: в 1967-м году он впервые опубликовал одну из них в своей книге "Проблема узнавания"

Тесты М.М.Бонгарда



Тесты М.М.Бонгарда



Как сравнивать закономерности?

$$\begin{cases} p(R) \to \max \\ n(R) \to \min \end{cases} \stackrel{?}{\Longrightarrow} \quad I(p,n) \to \max$$

$I(p,n) = \dfrac{p}{p+n} \to \max$ (precision);

$I(p,n) = p - n \to \max$ (accuracy);

$I(p,n) = p - Cn \to \max$ (linear cost accuracy);

$I(p,n) = \dfrac{p}{P} - \dfrac{n}{N} \to \max$ (relative accuracy);

$P_c = \#\{x_i : y_i = c\}$ — число «своих» во всей выборке;

$N_c = \#\{x_i : y_i \neq c\}$ — число «чужих» во всей выборке.

Как сравнивать закономерности?

при P = 200, N = 100 и различных p и n

p	n	p/(p+n)	p-n	p-5n	p/P-n/N
10	0	1	10	10	0,05
200	10	0,95	190	150	0,9
10	0	1	10	10	0,05
60	50	0,55	10	-190	-0,2
200	40	0,83	160	0	0,6
5	1	0,83	4	0	0,02
10	0	1	10	10	0,05
200	95	0,68	105	-275	0,05

Вероятностный подход

- Рассмотрим опыт – отбор предикатом объектов обучающей выборки
- Предикат – закономерность ттт,к события: “объект отобран предикатом” и “объект имеет класс **c**” зависимы
- Качество закономерности = мера зависимости случайных событий

Точный тест Фишера

- Предположим, что события “объект отобран предикатом” и “объект имеет класс **c**” независимы
- Тогда вероятность отобрать p объектов класса **c** и n – других классов:

Точный тест Фишера

- Предположим, что события “объект отобран предикатом” и “объект имеет класс **c**” независимы
- Тогда вероятность отобрать p объектов класса **c** и n – других классов: $\frac{C_P^p C_N^n}{C_{P+N}^{p+n}}$
- Это правдоподобие гипотезы независимости событий. Чем меньше данная вероятность, тем более зависимы события

$$\mathrm{IStat}(p,n) = -\tfrac{1}{\ell}\log_2 \frac{C_P^p C_N^n}{C_{P+N}^{p+n}} \to \max$$

Точный тест Фишера

p	n	p/(p+n)	p-n	p-5n	p/P-n/N	Вероятность
10	0	1	10	10	0,05	0,016
200	10	0,95	190	150	0,9	8,80E-66
10	0	1	10	10	0,05	0,016
60	50	0,55	10	-190	-0,2	3,50E-04
200	40	0,83	160	0	0,6	1,50E-36
5	1	0,83	4	0	0,02	0,26
10	0	1	10	10	0,05	0,016
200	95	0,68	105	-275	0,05	0,0038

Точный тест Фишера

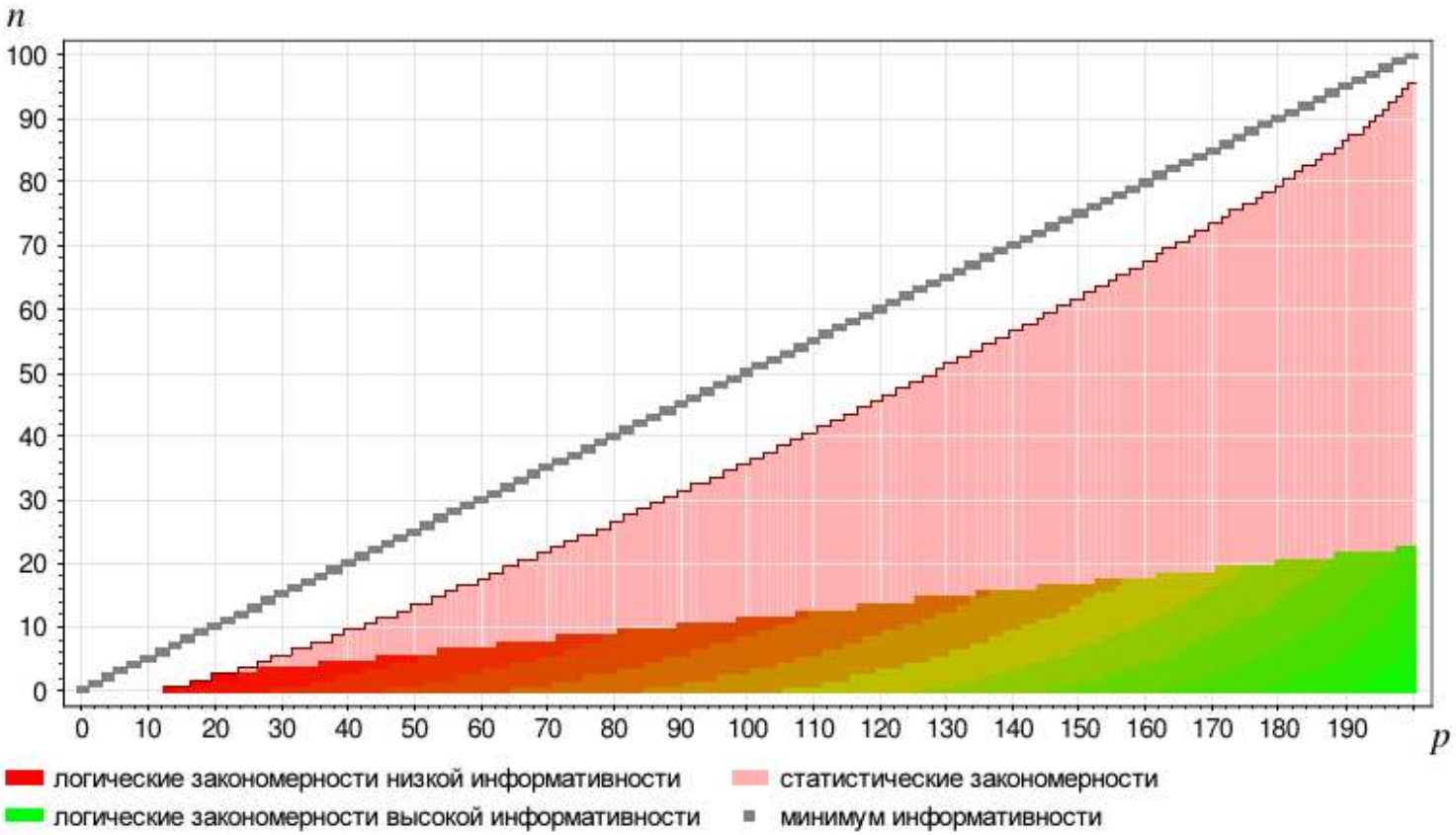
p	n	p/(p+n)	p-n	p-5n	p/P-n/N	Вероятность	Плотность
10	0	1	10	10	0,05	0,016	0,16
200	10	0,95	190	150	0,9	8,80E-66	1,848E-063
10	0	1	10	10	0,05	0,016	0,16
60	50	0,55	10	-190	-0,2	3,50E-04	0,0385
200	40	0,83	160	0	0,6	1,50E-36	3,6E-034
5	1	0,83	4	0	0,02	0,26	1,56
10	0	1	10	10	0,05	0,016	0,16
200	95	0,68	105	-275	0,05	0,0038	1,121

Для дискретных распределений модели с меньшим числом значений случайной величины более правдоподобны, чем с б**о**льшим. Это – переобучение. Лучше переходить к приближению дискретной плотности непрерывной функцией, например, кусочно-постоянной т.е. умножать вероятность теста Фишера на (p+n) – количество различных значений случайной величины.

Логические и статистические закономерности

Логические закономерности: $\frac{n}{p+n} \leqslant 0.1, \quad \frac{p}{P+N} \geqslant 0.05.$

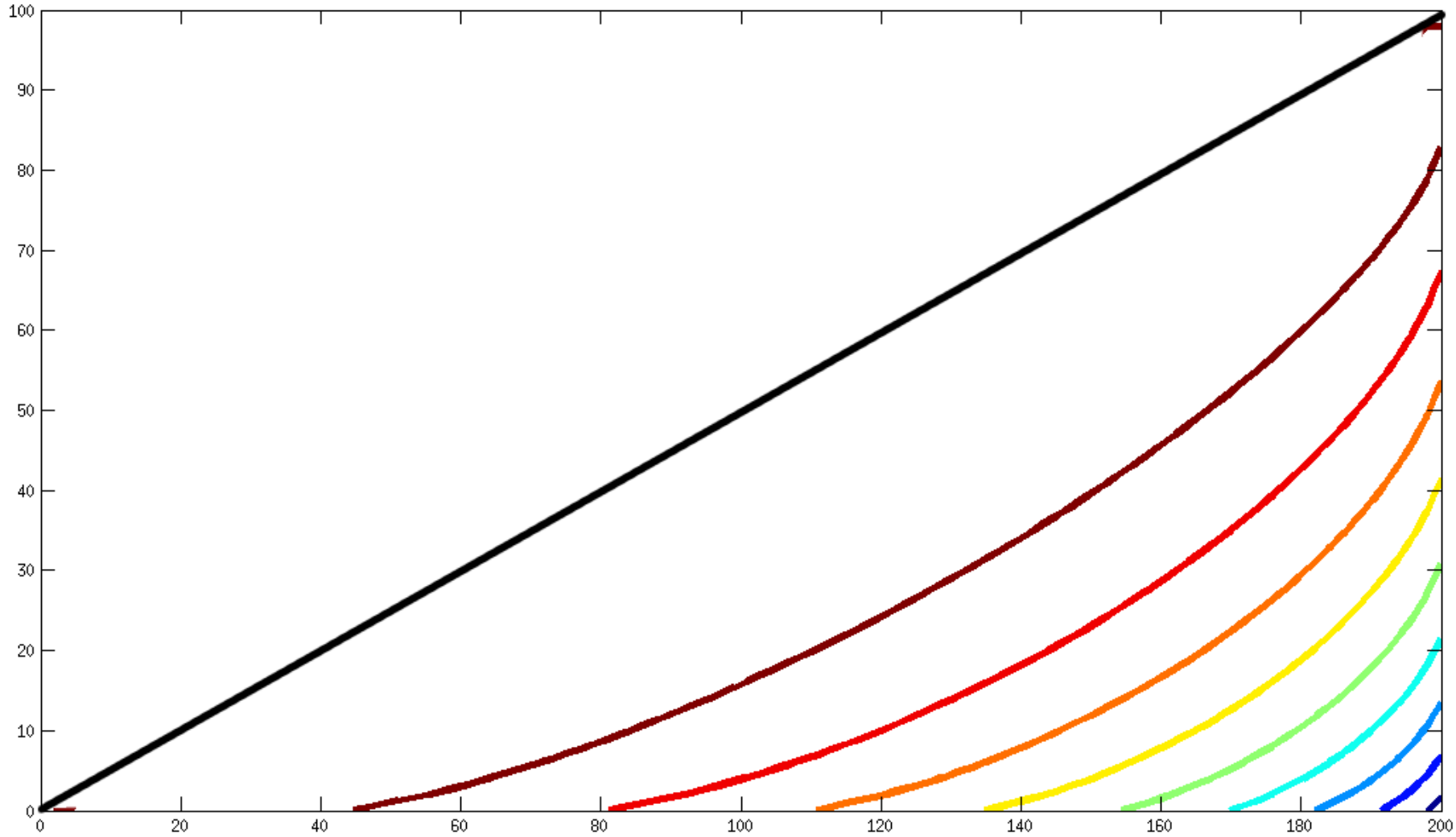
Статистические закономерности: $\mathrm{IStat}(p, n) \geqslant 3.$



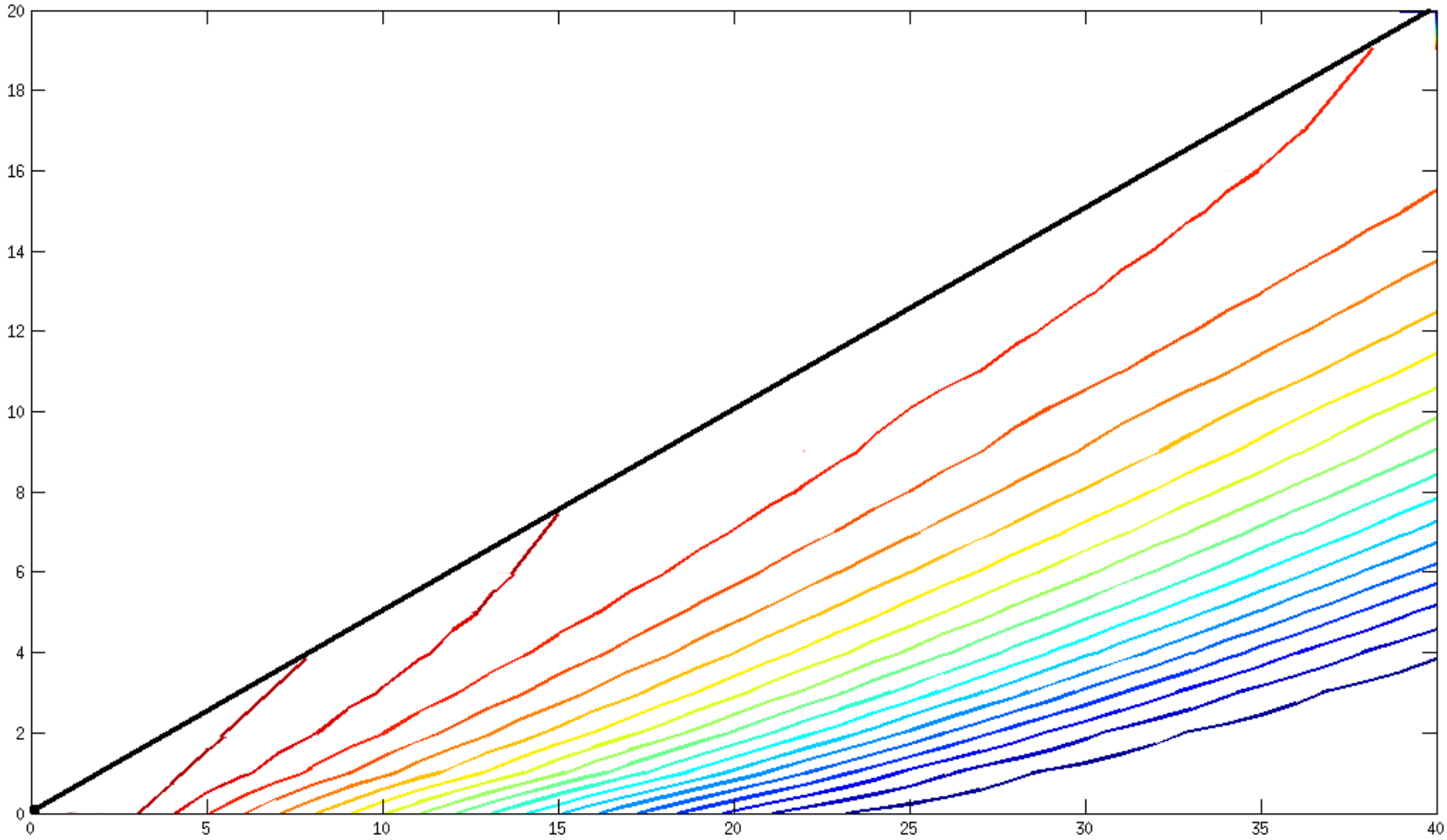
$P = 200$

$N = 100$

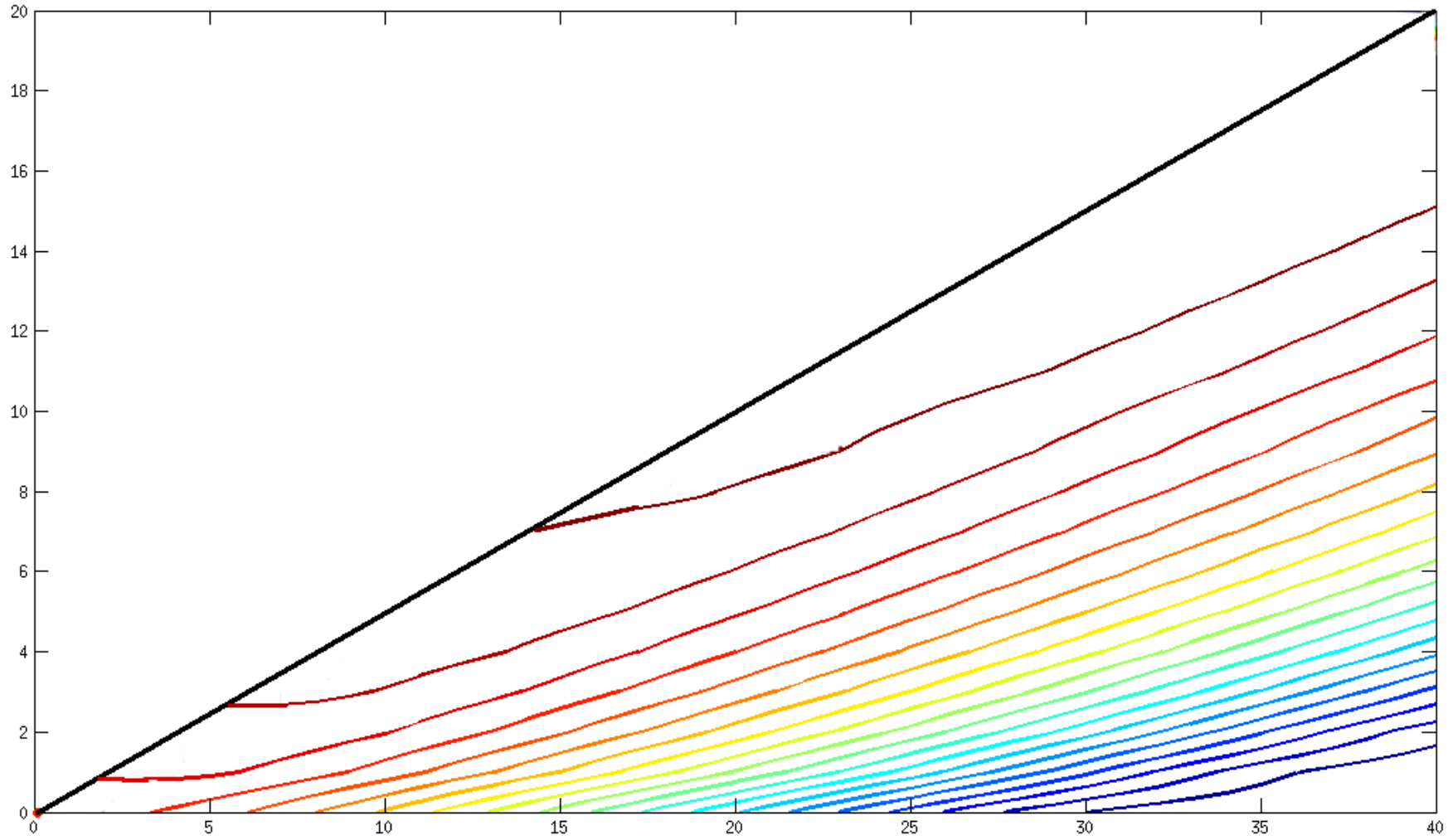
Линии уровня теста Фишера



Линии уровня теста Фишера
Малые p и n



Линии уровня плотности вероятности. Малые p и n



Энтропийный критерий информативности

Пусть ω_0, ω_1 — два исхода с вероятностями q и $1-q$.

Количество информации: $I_0 = -\log_2 q$, $I_1 = -\log_2(1-q)$.

Энтропия — математическое ожидание количества информации:

$$h(q) = -q\log_2 q - (1-q)\log_2(1-q).$$

Энтропия выборки X^ℓ, если исходы — это классы $y{=}c$, $y{\neq}c$:

$$H(y) = h\left(\frac{P}{\ell}\right).$$

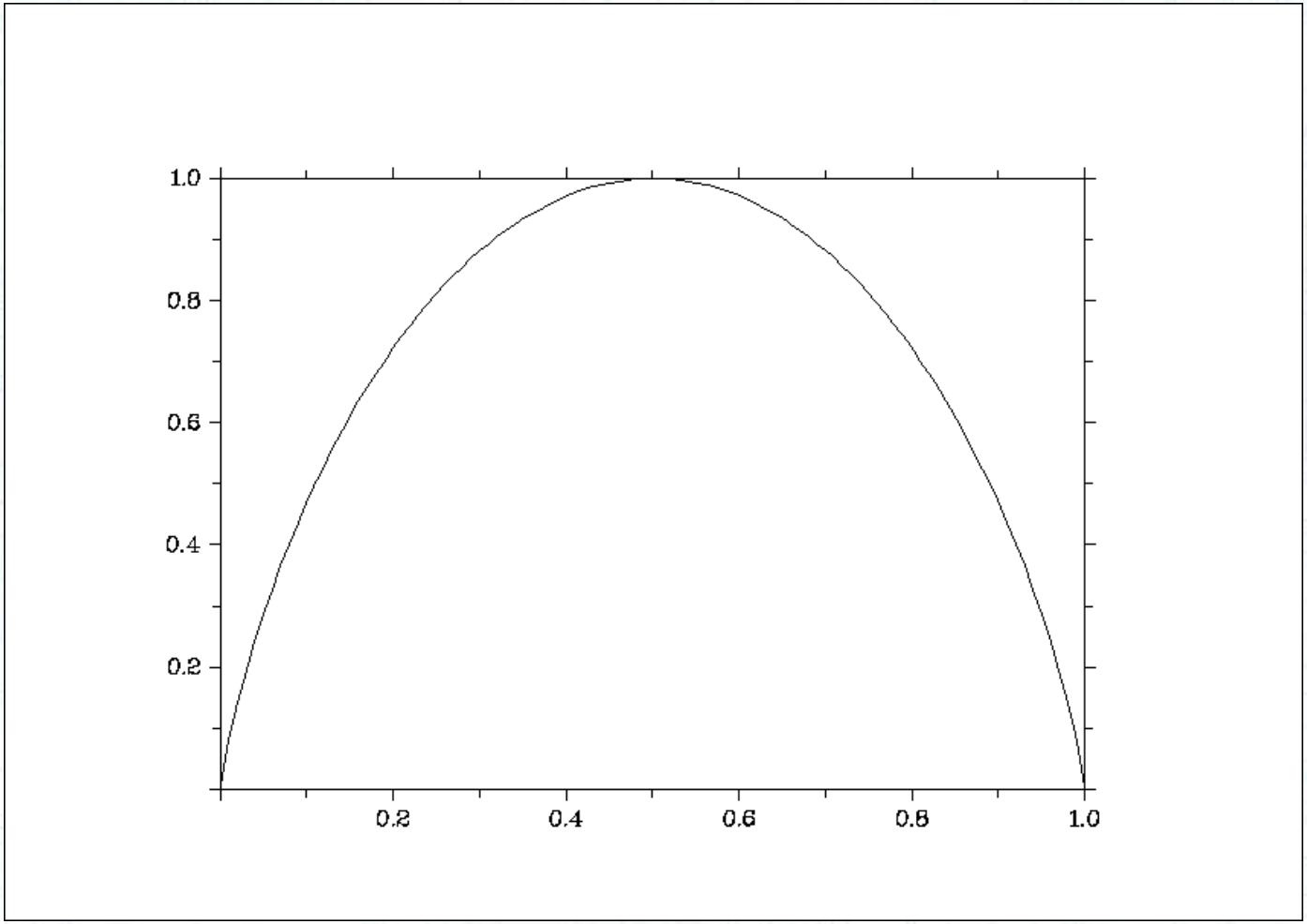
Энтропия выборки X^ℓ после получения информации $R(x_i)_{i=1}^{\ell}$:

$$H(y|R) = \frac{p+n}{\ell} h\left(\frac{p}{p+n}\right) + \frac{\ell-p-n}{\ell} h\left(\frac{P-p}{\ell-p-n}\right).$$

Прирост информации (Information gain, IGain):

$$\mathrm{IGain}(p,n) = H(y) - H(y|R).$$

Энтропия для различных q



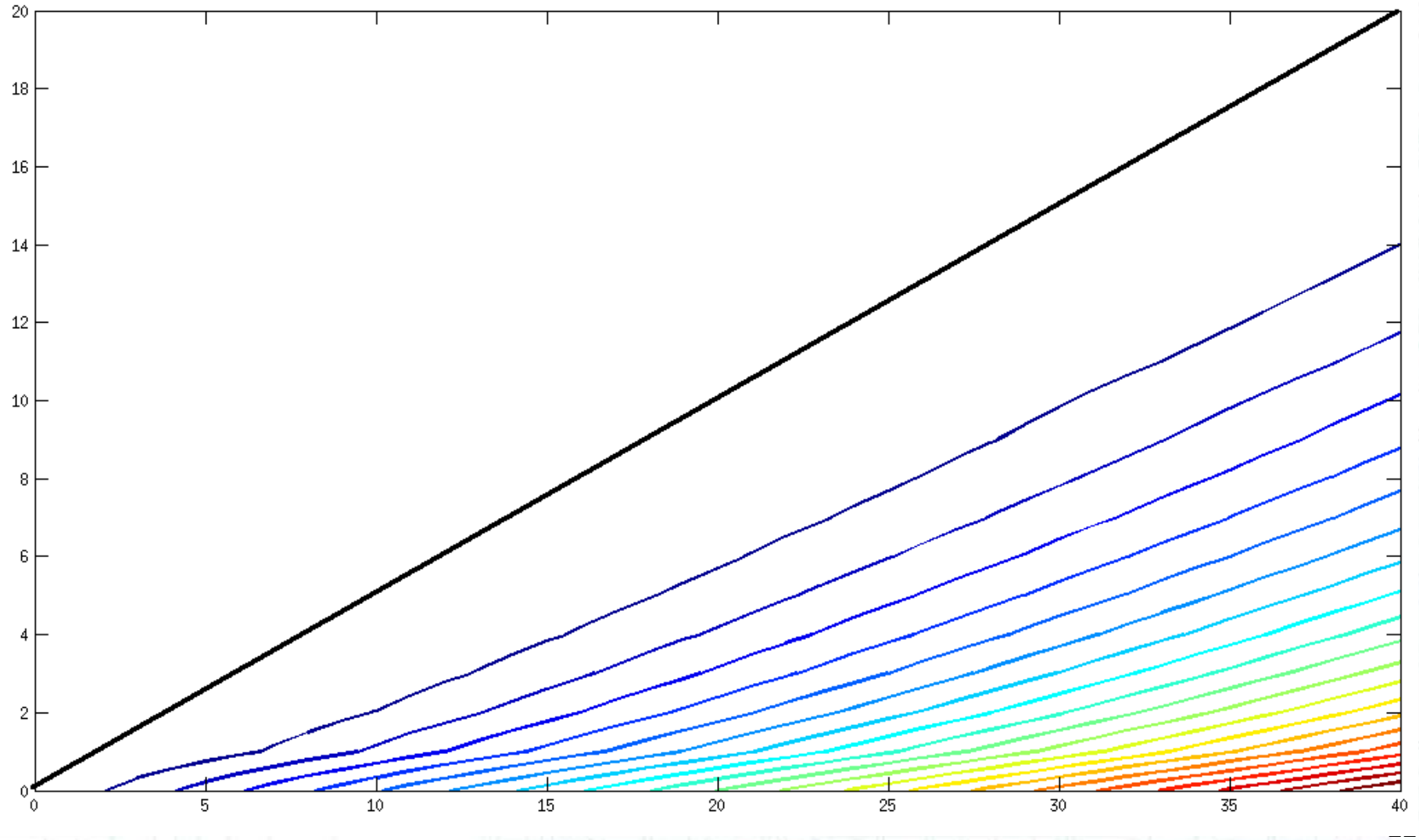
Соотношение статистического и энтропийного критериев

Энтропийный критерий IGain асимптотически эквивалентен статистическому IStat:

$$\mathsf{IStat}(p,n) \to \mathsf{IGain}(p,n) \quad \textit{при } \ell \to \infty$$

Доказательство: применить формулу Стирлинга к критерию IStat.

Линии уровня энтропийного критерия. Малые p и n



Построение закономерностей (rule induction)

1. Пороговое условие (решающий пень, decision stump):

$$R(x) = \big[f_j(x) \leqslant a_j\big] \text{ или } \big[a_j \leqslant f_j(x) \leqslant b_j\big].$$

2. *Конъюнкция* пороговых условий:

$$R(x) = \bigwedge_{j \in J} \big[a_j \leqslant f_j(x) \leqslant b_j\big].$$

3. *Синдром* — выполнение не менее d условий из J, (при $d = |J|$ это конъюнкция, при $d = 1$ — дизъюнкция):

$$R(x) = \left[\sum_{j \in J} \big[a_j \leqslant f_j(x) \leqslant b_j\big] \geqslant d\right],$$

Параметры J, a_j, b_j, d настраиваются по обучающей выборке путём оптимизации *критерия информативности*.

Построение закономерностей (rule induction)

4. *Полуплоскость* — линейная пороговая функция:

$$R(x) = \Big[\sum_{j \in J} w_j f_j(x) \geqslant w_0 \Big].$$

5. *Шар* — пороговая функция близости:

$$R(x) = \big[r(x, x_0) \leqslant w_0 \big],$$

ABO — алгоритмы вычисления оценок [Ю. И. Журавлёв, 1971]:

$$r(x, x_0) = \max_{j \in J} w_j \big| f_j(x) - f_j(x_0) \big|.$$

SCM — машины покрывающих множеств [M. Marchand, 2001]:

$$r(x, x_0) = \sum_{j \in J} w_j \big| f_j(x) - f_j(x_0) \big|^{\gamma}.$$

Параметры J, w_j, w_0, x_0 настраиваются по обучающей выборке путём оптимизации *критерия информативности*.

Поиск информативных закономерностей

Вход: выборка X^ℓ;

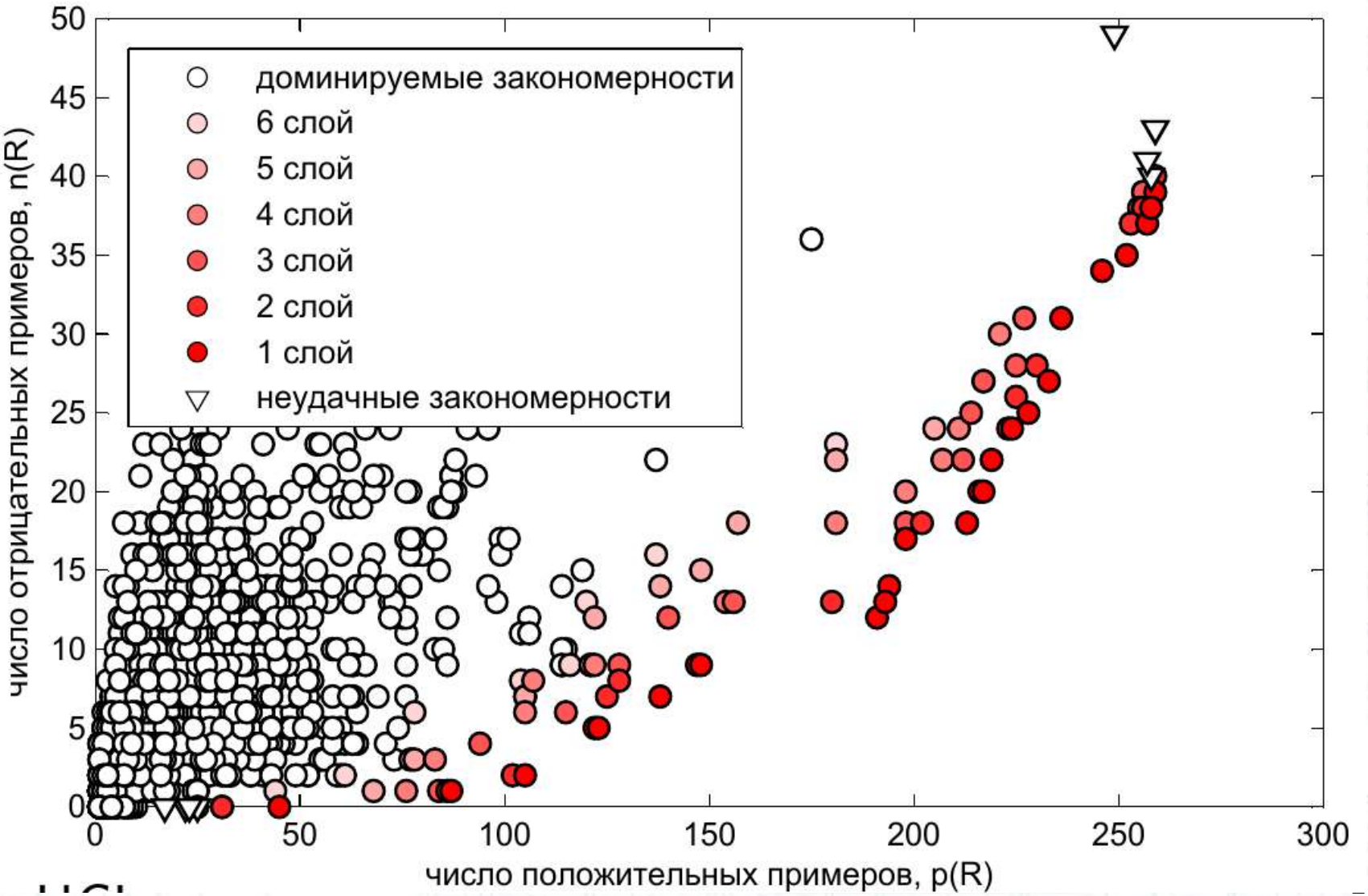
Выход: множество закономерностей Z;

1: начальное множество правил Z;
2: **повторять**
3: $\quad Z' :=$ множество модификаций правил $R \in Z$;
4: $\quad$ удалить слишком похожие правила из $Z \cup Z'$;
5: $\quad$ оценить информативность всех правил $R \in Z'$;
6: $\quad Z :=$ наиболее информативные правила из $Z \cup Z'$;
7: **пока** правила продолжают улучшаться
8: **вернуть** Z.

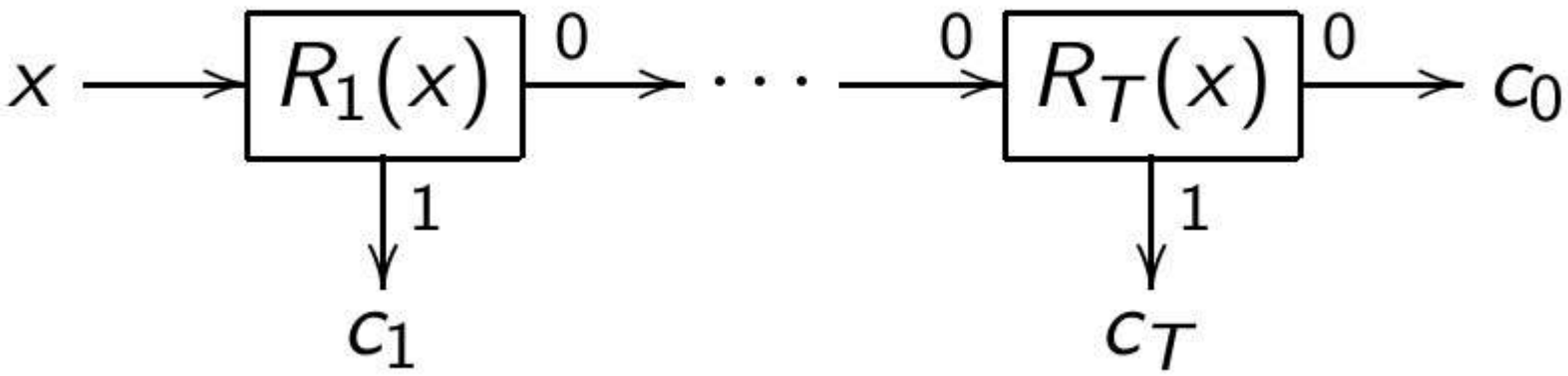
Частные случаи:

— стохастический локальный поиск (SLS)
— генетические (эволюционные) алгоритмы
— метод ветвей и границ

Отбор закономерностей по Парето



Алгоритмы классификации. Решающий список



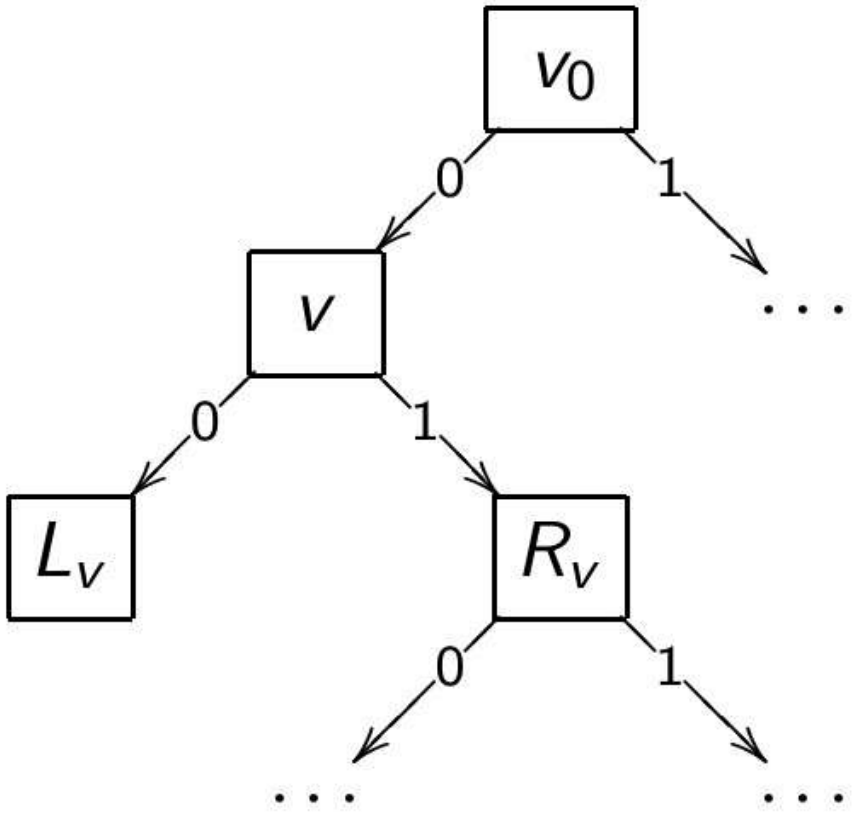
1: **для всех** $t = 1, \ldots, T$
2: **если** $R_t(x) = 1$ **то**
3: **вернуть** c_t;
4: **вернуть** c_0 — отказ от классификации объекта x

Жадный алгоритм построения решающего списка

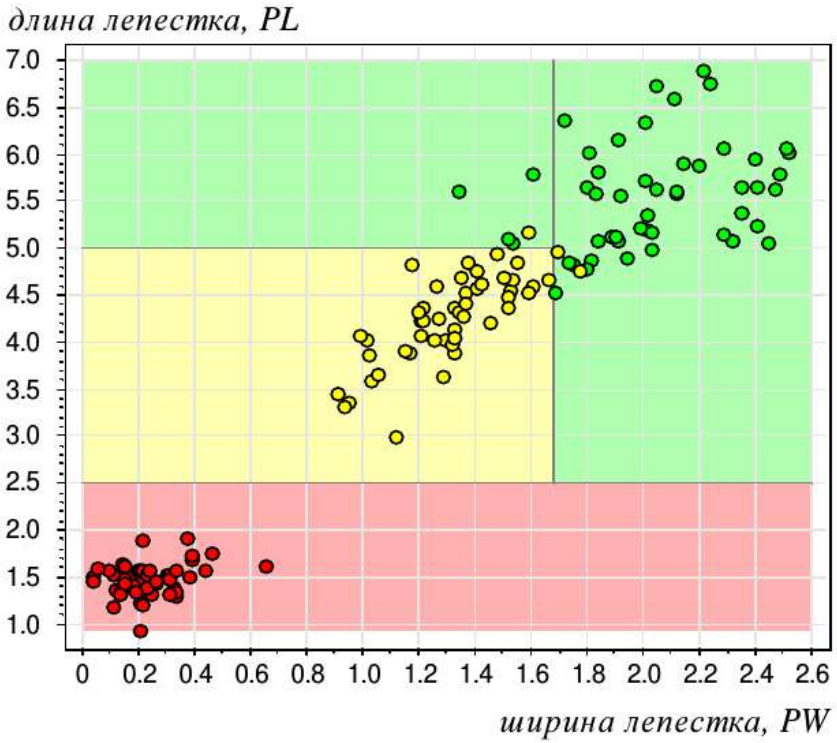
- $U := X_\ell$; $t := 1$
- Повторять:
  - Выбор класса =: c_t
  - Выбор предиката R_t: $I(R_t,U) \to \max$
  - $U := \{ x \in U \mid R_t(x) = 0 \}$
- Пока: $I > I_{min}$; $|U| > \ell_0$

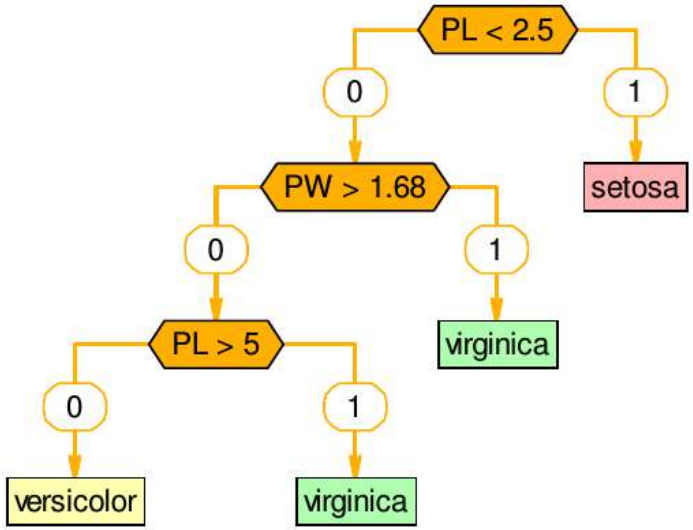
Решающее дерево

1: $v := v_0$;
2: **пока** $v \in V_{\text{внутр}}$
3: **если** $\beta_v(x) = 1$ **то**
4: переход вправо:
$v := R_v$;
5: **иначе**
6: переход влево:
$v := L_v$;
7: **вернуть** c_v.



Решающее дерево → покрывающий набор конъюнкций





setosa	$r_1(x) = [PL \leqslant 2.5]$
virginica	$r_2(x) = [PL > 2.5] \wedge [PW > 1.68]$
virginica	$r_3(x) = [PL > 5] \wedge [PW \leqslant 1.68]$
versicolor	$r_4(x) = [PL > 2.5] \wedge [PL \leqslant 5] \wedge [PW < 1.68]$

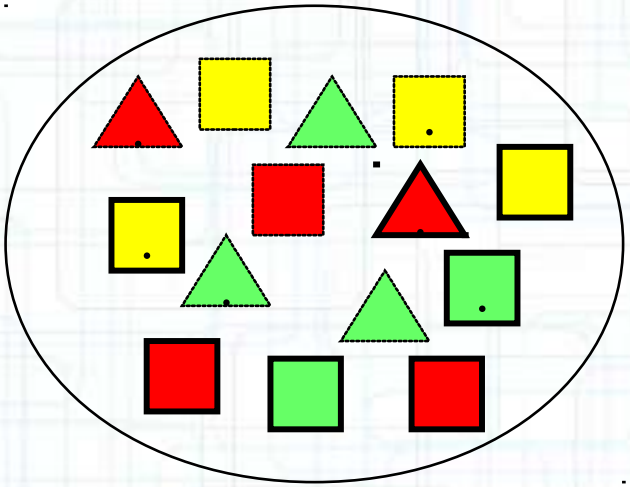
Жадный алгоритм построения решающего дерева

- Функция:
- Tree buildTree(U) {
  - Выбор предиката β_v: $I(\beta_v, U) \to \max$
  - $U_0 := \{ x \in U \mid \beta_v(x) = 0 \}$
  - $U_1 := \{ x \in U \mid \beta_v(x) = 1 \}$
  - Если $|U_0| < \ell_0$ или $|U_1| < \ell_0$ вернуть лист
  - Иначе:
    - L_v := buildTree(U_0)
    - R_v := buildTree(U_1)
- }

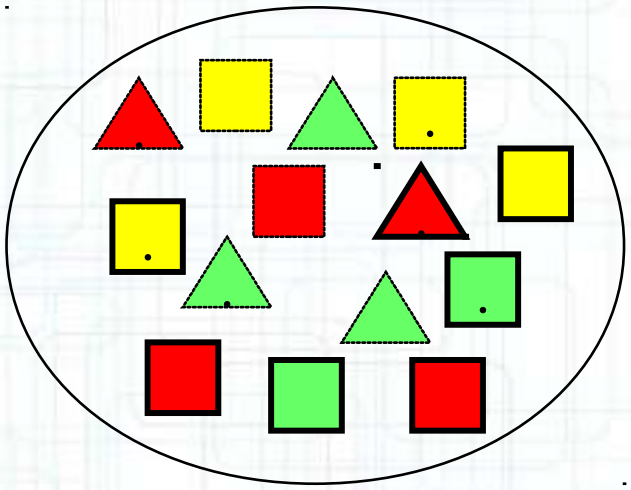
Пример: треугольники и квадраты

#	Attribute			Shape
	Color	Outline	Dot	
1	green	dashed	no	triange
2	green	dashed	yes	triange
3	yellow	dashed	no	square
4	red	dashed	no	square
5	red	solid	no	square
6	red	solid	yes	triange
7	green	solid	no	square
8	green	dashed	no	triange
9	yellow	solid	yes	square
10	red	solid	no	square
11	green	solid	yes	square
12	yellow	dashed	yes	square
13	yellow	solid	no	square
14	red	dashed	yes	triange

Обучающая выборка



Энтропия



- 5 треугольников
- 9 квадратов
- Вероятности классов

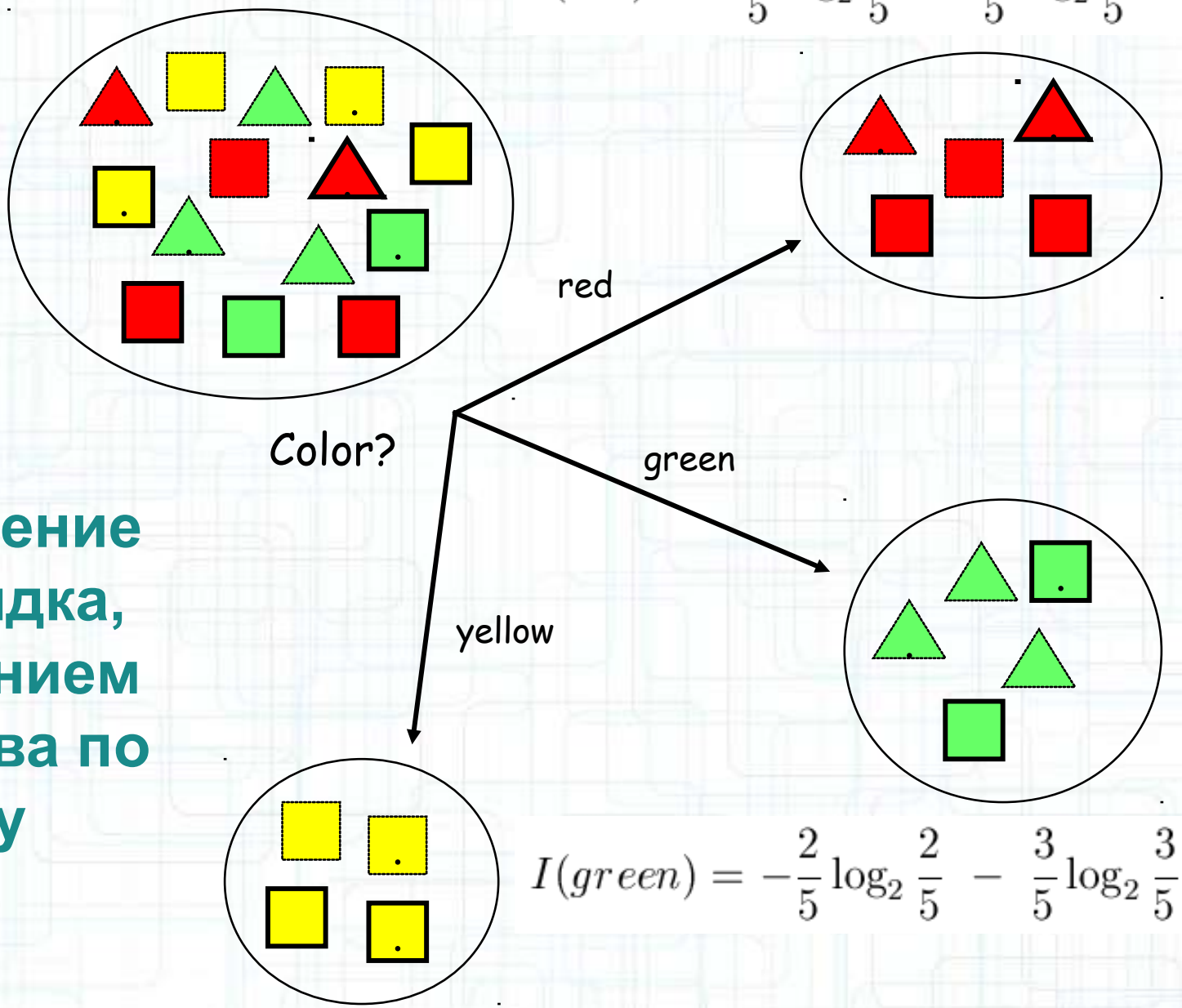
$$p(\square) = \frac{9}{14}$$

$$p(\triangle) = \frac{5}{14}$$

энтропия

$$I = -\frac{9}{14}\log_2\frac{9}{14} - \frac{5}{14}\log_2\frac{5}{14} = 0.940 \text{ bits}$$

$$I(red) = -\frac{3}{5}\log_2\frac{3}{5} - \frac{2}{5}\log_2\frac{2}{5} = 0.971 \text{ bits}$$

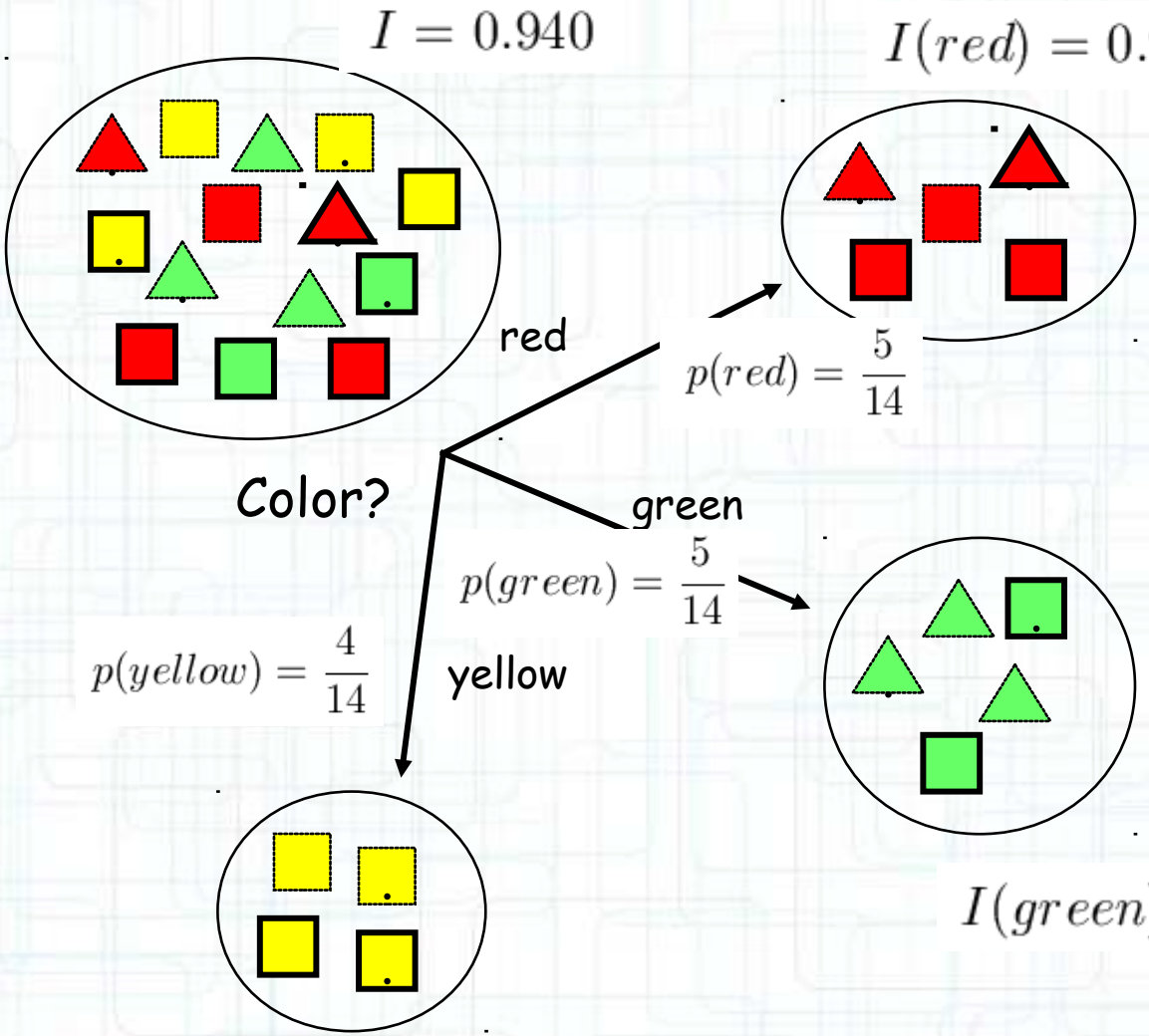


Уменьшение беспорядка, разделением множества по цвету

$$I(green) = -\frac{2}{5}\log_2\frac{2}{5} - \frac{3}{5}\log_2\frac{3}{5} = 0.971 \text{ bits}$$

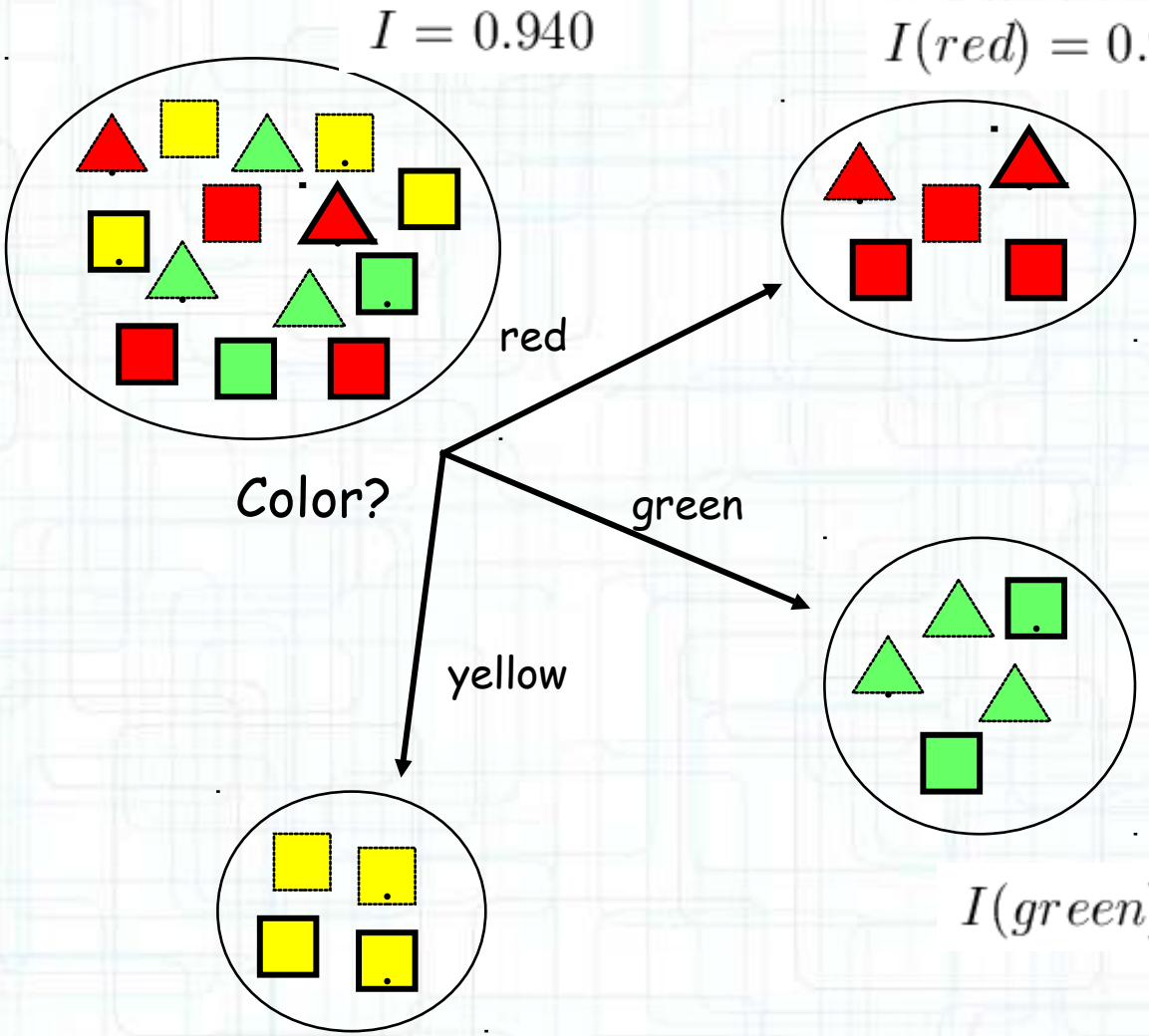
$$I(yellow) = -\frac{4}{4}\log_2\frac{4}{4} - \frac{0}{4}\log_2\frac{0}{4} = 0.0 \text{ bits}$$

Энтропия после разделения



$$I_{res}(\text{Color}) = \sum p(v)I(v) = \frac{5}{14}0.971 + \frac{5}{14}0.971 + \frac{4}{14}0.0 = 0.694 \; bits$$

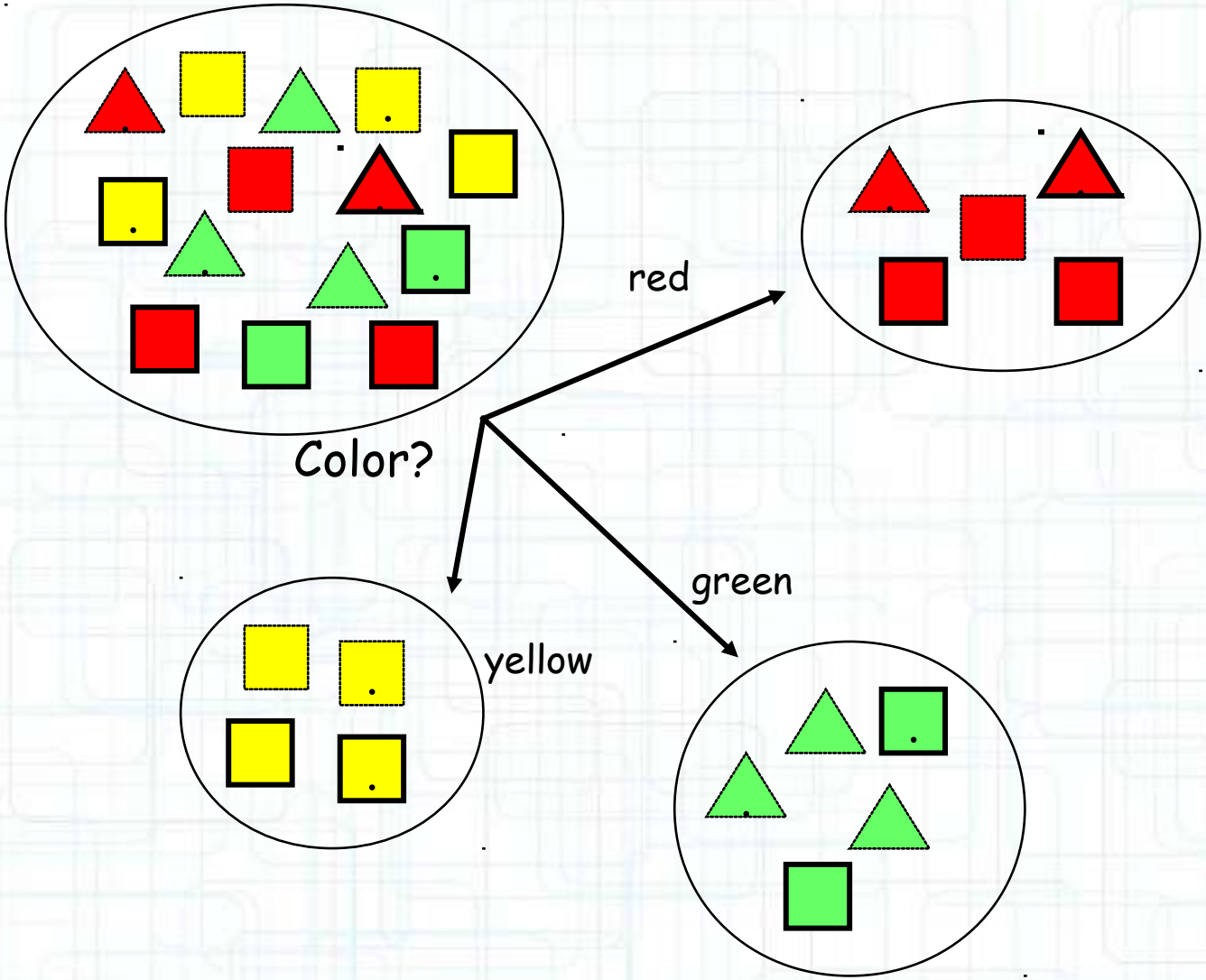
Прирост информации



$$Gain(\text{Color}) = I - I_{res}(\text{Color}) = 0.940 - 0.694 = 0.246\ bits$$

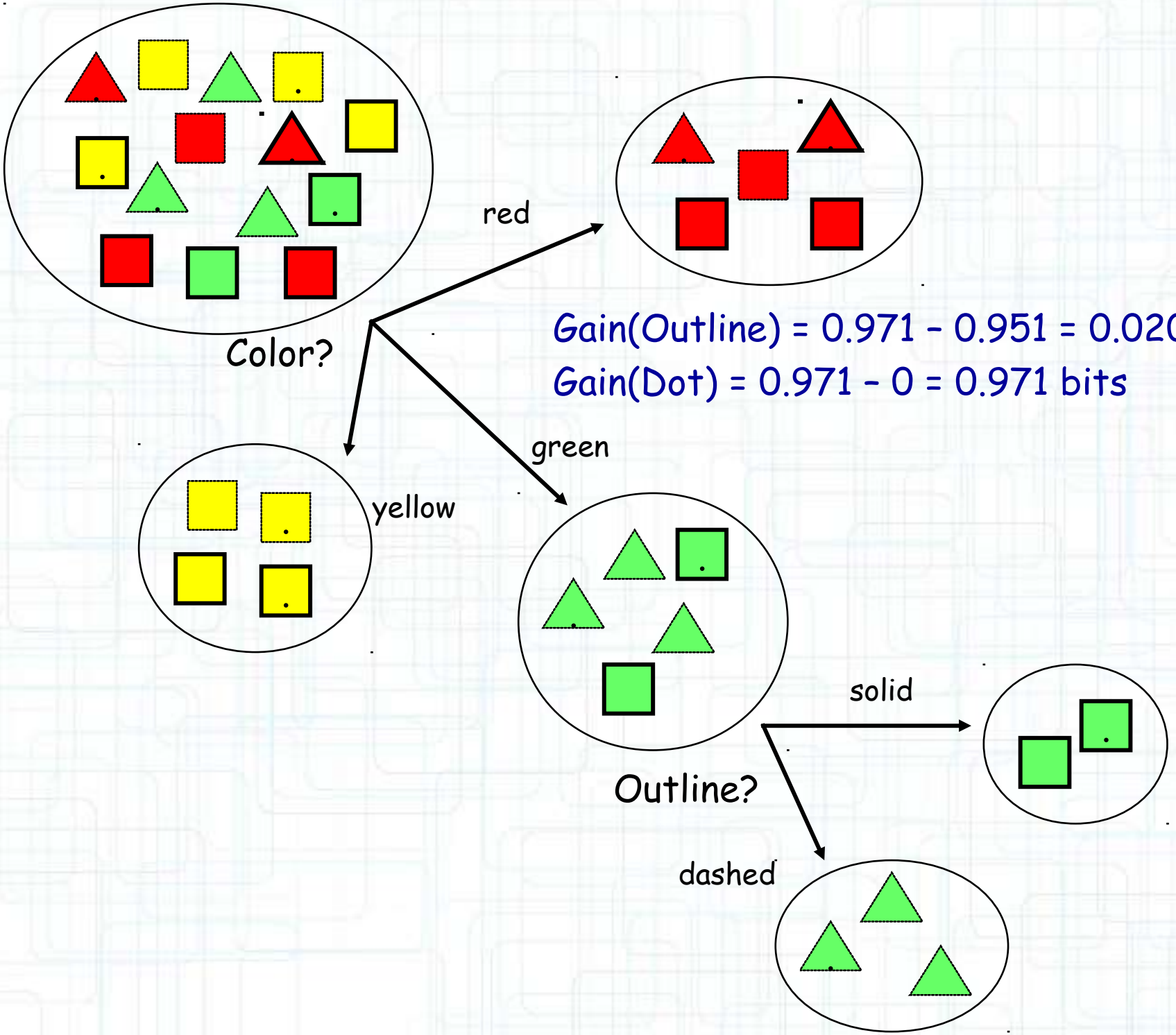
Прирост информации для каждого признака

- Признаки
  - Gain(Color) = 0.246
  - Gain(Outline) = 0.151
  - Gain(Dot) = 0.048
- Лучше всего разбивать множество по признаку, вносящему наибольший порядок



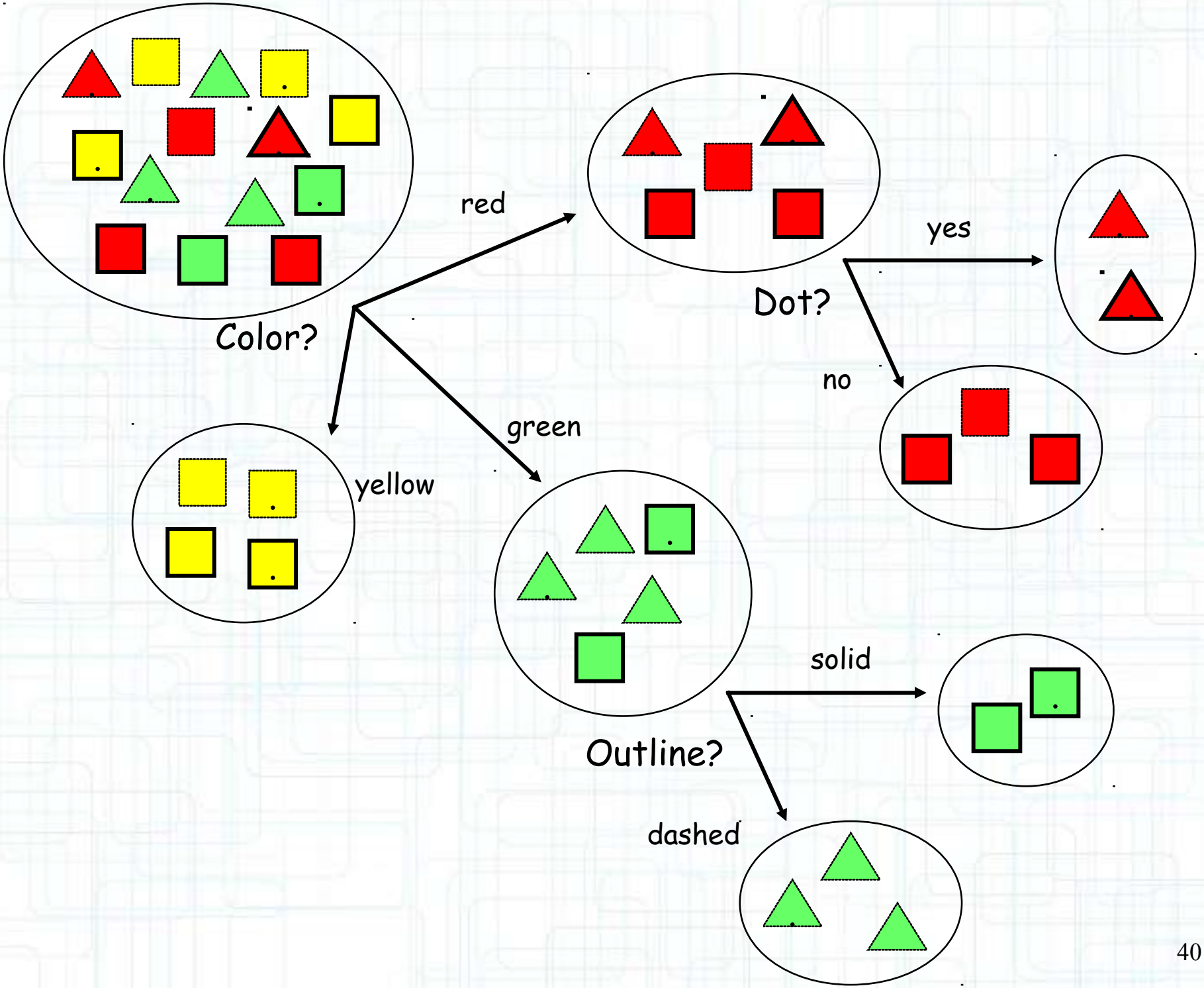
Gain(Outline) = 0.971 – 0 = 0.971 bits

Gain(Dot) = 0.971 – 0.951 = 0.020 bits

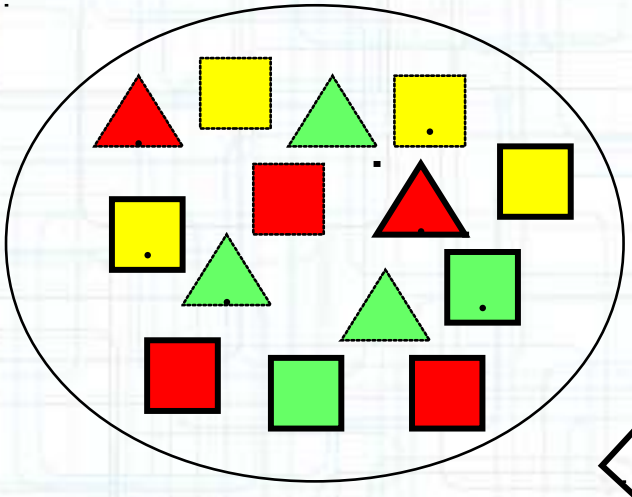


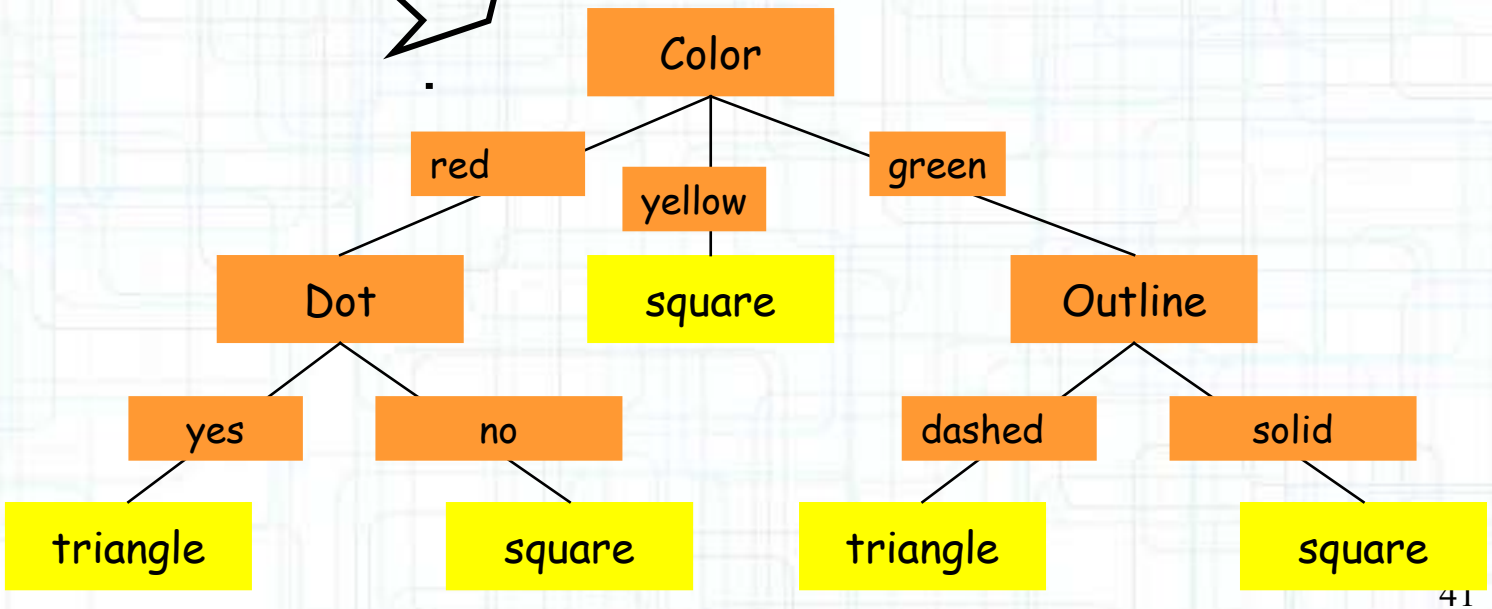
Gain(Outline) = 0.971 – 0.951 = 0.020 bits

Gain(Dot) = 0.971 – 0 = 0.971 bits


Color?
red
yellow
green
Dot?
yes
no
Outline?
solid
dashed

Итоговое дерево





Редукция дерева (pruning)

- Pre-pruning – критерий раннего останова. Досрочное прекращение ветвления, если информативность < порога или глубина велика.
- Post-pruning – пост-редукция. Простматриваем все внутренние вершины дерева и проверяем их качество на тестовой выборке (OutOfBag error). Заменяем листом, где качество после разделения ухудшается

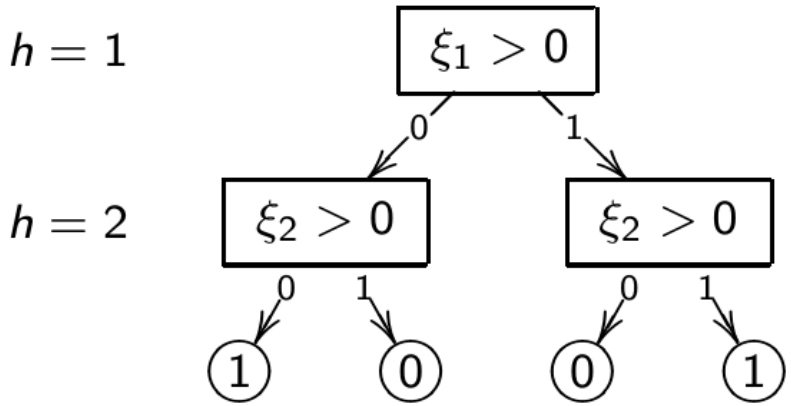
Обобщение на случай задачи регрессии

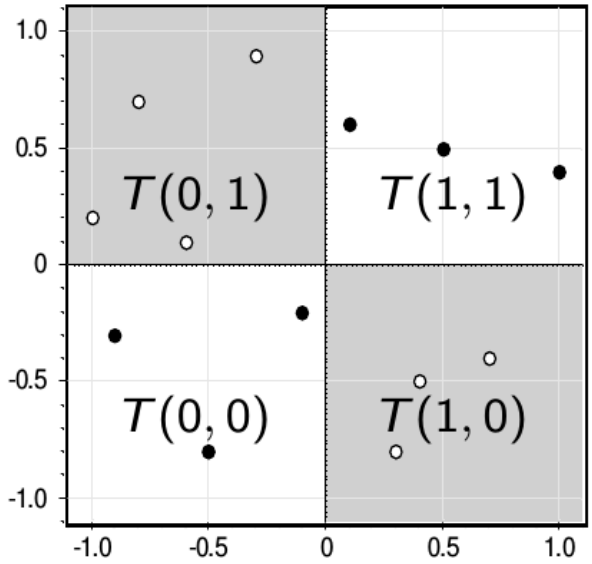
- В каждом листе целевое значение определяется по методу наименьших квадратов
- Критирий информативности – среднеквадратическая ошибка

Небрежные решающие деревья (Oblivious Decision Tree)

- Для всех узлов на глубине h условие ветвления одинаково
- Дерево получается сбалансированным, на глубине h ровно 2^{h-1} вершин

Пример: задача XOR, $H = 2$.





FastXML: A Fast, Accurate and Stable Tree-classifier for eXtreme Multi-label Learning

Yashoteja Prabhu
Indian Institute of Technology - Delhi
yashoteja.prabhu@gmail.com

Manik Varma
Microsoft Research
manik@microsoft.com

ABSTRACT

The objective in extreme multi-label classification is to learn a classifier that can automatically tag a data point with the most relevant *subset* of labels from a large label set. Extreme multi-label classification is an important research problem since not only does it enable the tackling of applications with many labels but it also allows the reformulation of ranking problems with certain advantages over existing formulations.

Our objective, in this paper, is to develop an extreme multi-label classifier that is faster to train and more accurate at prediction than the state-of-the-art Multi-label Random Forest (MLRF) algorithm [2] and the Label Partitioning for Sub-linear Ranking (LPSR) algorithm [35]. MLRF and LPSR learn a hierarchy to deal with the large number of labels but optimize task independent measures, such as the Gini index or clustering error, in order to learn the hierarchy. Our proposed FastXML algorithm achieves significantly higher accuracies by directly optimizing an nDCG based ranking loss function. We also develop an alternating minimization algorithm for efficiently optimizing the proposed formulation. Experiments reveal that FastXML can be trained on problems with more than a million labels on a standard desktop in eight hours using a single core and in an hour using multiple cores.

Categories and Subject Descriptors

I.2.6 [**Artificial Intelligence**]: Learning

General Terms

Algorithms, Performance, Machine Learning, Optimization

Keywords

Multi-label Learning; Ranking; Extreme Classification

1. INTRODUCTION

The objective in extreme multi-label classification is to learn a classifier that can automatically tag a data point with the most relevant *subset* of labels from a large label set. For instance, there are more than a million categories on Wikipedia and one might wish to build a classifier that annotates a novel web page with the subset of most relevant Wikipedia categories. It should be emphasized that multi-label learning is distinct from multi-class classification which aims to predict a single mutually exclusive label.

Extreme classification is an important research problem not just because modern day applications have many categories but also because it allows the reformulation of core learning problems such as recommendation and ranking. For instance, [2] treated search engine queries as labels and built an extreme classifier which, given a novel web page, returned a ranking of queries in decreasing order of relevance. Similarly, [35] treated YouTube videos as distinct labels in an extreme classifier so as to recommend a ranked list of videos to users. Both methods provided a fresh way of thinking about ranking and recommendation problems that led to significant improvements over the state-of-the-art.

Extreme classification is also a challenging research problem as one needs to simultaneously deal with a large number of labels, dimensions and training points. An obvious baseline is provided by the 1-vs-All technique where an independent classifier is learnt per label. Such a baseline has at least two major limitations. First, training millions of high dimensional classifiers might be computationally expensive. Second, the cost of prediction might be high since all the classifiers would need to be evaluated every time a prediction needed to be made. These problems could be ameliorated if a label hierarchy was provided. Unfortunately, such a hierarchy is unavailable in many applications [2, 35].

State-of-the-art methods therefore learn a hierarchy from training data as follows: The root node is initialized to contain the entire label set. A node partitioning formulation is then optimized to determine which labels should be assigned to the left child and which to the right. Nodes are recursively partitioned till each leaf contains only a small number of labels. During prediction, a novel data point is passed down the tree until it reaches a leaf node. A base multi-label classifier of choice can then be applied to the data point focusing exclusively on the leaf node label distribution. This leads to prediction costs that are sub-linear or even logarithmic if the tree is balanced.

Tree based methods can often outperform the 1-vs-All baseline in terms of prediction accuracy at a fraction of the prediction cost. However, such methods can also be expensive to train. In particular, the Label Partitioning by Sublinear Ranking (LPSR) algorithm of [35] can have

KDD'14, August 24–27, 2014, New York, NY, USA.

ACM 978-1-4503-2956-9/14/08 ...$15.00.
http://dx.doi.org/10.1145/2623330.2623651.

even higher training costs than the 1-vs-All baseline since it needs to learn the hierarchy in addition to the base classifier. Similarly, the Multi-label Random Forest (MLRF) approach of [2] required a cluster of a thousand nodes as random forest training was found to be expensive in high dimensional spaces. Such expensive approaches not only increase the cost of deploying extreme classification models and the cost of daily experimentation but also put such models beyond the reach of most practitioners.

Our objective, in this paper, is to tackle this problem and build a tree based extreme multi-label classifier, referred to as FastXML, that is faster to train as well as more accurate than the state-of-the-art MLRF and LPSR. Our key technical contributions are a novel node partitioning formulation and an algorithm for its efficient optimization. In terms of training time, FastXML can be significantly faster than MLRF since the proposed node partitioning formulation can be optimized more efficiently than the entropy or Gini index based formulations in random forests. At the same time, FastXML can be faster to train than LPSR which has to first learn computationally expensive base classifiers for accurate prediction. In terms of prediction accuracy, almost all extreme classification applications deal with scenarios where the number of relevant positive labels for a data point is orders of magnitude smaller than the number of irrelevant negative ones. Prediction accuracy is therefore not measured using traditional multi-label metrics, such as the Hamming loss, which give equal weightage to all labels whether positive or negative. Instead, most applications prefer employing ranking based measures such as precision at k which focuses exclusively on the positive labels by counting the number of correct predictions in the top k positive predictions. FastXML's proposed node partitioning formulation therefore directly optimizes a rank sensitive loss function which can lead to more accurate predictions over MLRF's Gini index or LPSR's clustering error.

Experiments indicated that FastXML could be significantly more accurate at prediction (by as much as 5% in some cases) on benchmark data sets with thousands of labels where accurate models for MLRF, LPSR and the 1-vs-All baseline could be learnt. Furthermore, FastXML could efficiently scale to problems involving a million labels where accurate training of MLRF, LPSR and 1-vs-ALL models would require a very large cluster. For instance, using a single core on a standard desktop, a single FastXML tree could be trained in under 10 minutes on a data set with about 4 M training points, 160 K features and 1 M labels. The entire ensemble could be trained in 8 hours using a single core and in 1 hour using multiple cores. MLRF and 1-vs-All could not be trained on such extreme multi-label data sets on a standard desktop. LPSR training could be made tractable by replacing the computationally expensive 1-vs-All base classifier by the cheaper Naïve Bayes but then its classification accuracy was found to lag behind by more than 20% on data sets such as Wikipedia.

Our contributions are: (a) We formulate a novel node partitioning objective which directly optimizes an nDCG based ranking loss and which implicitly learns balanced partitions; (b) We propose an efficient optimization algorithm for the novel formulation; and (c) we combine these in a tree algorithm which can train on problems with a million labels on a standard desktop while increasing prediction accuracy. Code for FastXML can be downloaded by clicking here.

2. RELATED WORK

There has been much recent progress in extreme multi-label classification [2,4,8,10,12,15,19,20,22,29,34–36,38] and most approaches are either based on trees or on embeddings.

To clarify the discussion, it is assumed that the training set can be represented as $\{(\mathbf{x}_i, \mathbf{y}_i)_{i=1}^N\}$ with D dimensional real-valued feature vectors $\mathbf{x}_i \in \mathcal{R}^D$ with sparsity $O(\hat{D})$, and L dimensional binary label vectors $\mathbf{y}_i \in \{0,1\}^L$ with $\mathbf{y}_{il} = 1$ if label l is relevant for point i and 0 otherwise. If one further assumes that the cost of training a linear binary classifier is $O(N\hat{D})$, then the training cost of the linear 1-vs-All baseline is $O(LN\hat{D})$ and the prediction cost is $O(L\hat{D})$. This is infeasible when L and N are in the range $10^5 - 10^6$. One can mitigate the training cost by sub-sampling the negative class for each binary classifier but the prediction cost would still be high.

Embedding methods [4, 8, 10, 12, 15, 19, 20, 22, 29, 34, 36, 38] exploit label correlations and sparsity to compress the number of labels from L to $\hat{L}$. A low-dimensional embedding of the label space is found typically through a linear projection $\hat{\mathbf{y}} = \mathbf{P}\mathbf{y}$ where $\mathbf{P}$ is a $\hat{L} \times L$ projection matrix. The 1-vs-All strategy can now be applied and the cost of training and prediction in the compressed space reduces to $O(\hat{L}N\hat{D})$ and $O(\hat{L}\hat{D})$ respectively. The compressed label predictions also need to be uncompressed at an additional cost of $O(\hat{L}L)$ or higher. Methods mainly differ in the choice of compression and decompression techniques such as compressed sensing [19, 22], Bloom filters [12], SVD [29], landmark labels [4, 8], output codes [38], *etc.* An interesting variant can be obtained by compressing the features $\hat{\mathbf{x}} = \mathbf{R}\mathbf{x}$ to the same $\hat{L}$ dimensional space as the labels [34]. Predictions can then be efficiently made in the embedding space via nearest neighbour techniques and no decompression is required.

Embedding methods have many advantages including simplicity, ease of implementation, strong theoretical foundations, the ability to handle label correlations, the ability to adapt to online and incremental scenarios, *etc.* Unfortunately, embedding methods can also pay a heavy price in terms of prediction accuracy due to the loss of information during the compression phase. For instance, none of the embedding methods developed so far have been able to consistently outperform the 1-vs-All baseline when $\hat{L} \approx \log(L)$.

Tree methods that learn a hierarchy enjoy many of the same advantages as the embedding methods. In addition, they can outperform the 1-vs-All baseline even when the prediction costs are $O(\hat{D}\log L)$. Our primary comparison is therefore with tree-based methods.

The Label Partitioning by Sub-linear Ranking (LPSR) method of [35] focussed on reducing the prediction time by learning a hierarchy over a base classifier or ranker. First, a base multi-label classifier was learnt for the entire label set. This step governs the overall training complexity and prediction accuracy. Generative classifiers such as Naïve Bayes are quick to train but have low accuracy whereas discriminative classifiers such as 1-vs-All with linear SVMs [18], deep nets or Wsabie [34] are expensive to train but have higher accuracy. Next, a hierarchy was learnt in terms of a single binary tree. Nodes in the tree were grown by partitioning the node's data points into 2 clusters, corresponding to the left and the right child, using a variant of k-means over the feature vectors. The tree was grown until each leaf node had a small number of data points and corresponding labels. Fi-

nally, a relaxed integer program was optimized at each leaf node via gradient descent to activate a subset of the labels present at the node. During prediction, a novel point was passed down the tree until it reached a leaf node. Predictions were then made using the base classifier restricted to the set of active leaf node labels.

The Multi-label Random Forest (MLRF) approach of [2] did not need to learn a base classifier. Instead, an ensemble of randomized trees was learnt. Nodes were partitioned into a left and a right child by brute force optimization of a multi-label variant of the Gini index defined over the set of positive labels in the node. Trees were grown until each leaf node had only a few labels present. During testing, a novel point was passed down each tree and predictions were made by aggregating all the leaf node label distributions.

Both LPSR and MLRF have high training costs. LPSR needs to train an accurate base multi-label classifier, perform hierarchical k-means clustering and solve a label assignment problem at each leaf node. MLRF needs to learn an ensemble of trees where the cost of growing each node is high. In particular, while training in high dimensional spaces, random forests need to sample a large number of features at each node in order to achieve a good quality, balanced split (extremely random trees [17] and other variants do not work in extreme classification scenarios as they learn imbalanced splits). Furthermore, brute force optimization of the Gini index or entropy over each feature is expensive when there are a large number of training points and labels. All in all, accurate LPSR and MLRF training can require large clusters – with up to a thousand nodes in the case of MLRF.

Finally, care should be taken to note that our objective of learning a multi-label hierarchy is very different from the objective of learning a multi-class hierarchy [5, 11, 13, 16] or exploiting a pre-existing multi-label hierarchy [7, 9, 27].

3. FASTXML

Our primary objective, in developing FastXML, is to enable training on a single desktop or a small cluster. At the same time, we aim to achieve greater prediction accuracy by optimizing a more suitable rank sensitive loss function as compared to MLRF's Gini index and LPSR's clustering error. We start this Section by presenting an overview of the FastXML algorithm, then go into the details of optimizing a loss function to learn a node partition and finally analyze FastXML's cost of prediction.

3.1 FastXML overview

FastXML learns a hierarchy, not over the label space as is traditionally done in the multi-class setting [5, 13, 16], but rather over the feature space. The intuition is similar to LPSR and MLRF's and comes from the observation that only a small number of labels are present, or active, in each region of feature space. Efficient prediction can therefore be carried out by determining the region in which a novel point lies by traversing the learnt feature space hierarchy and then focusing exclusively on the set of labels active in the region. Like MLRF, and unlike LPSR, FastXML defines the set of labels active in a region to be the union of the labels of all training points present in that region. This speeds up training as FastXML does not need to solve the label assignment integer program in each region. Furthermore, like MLRF, and unlike LPSR, FastXML learns an ensemble of trees and does not need to rely on base classifiers.

Algorithm 1 FastXML($\{\mathbf{x}_i, \mathbf{y}_i\}_{i=1}^N, T$)

parallel-for $i = 1, .., T$ **do**
  $n^{root} \leftarrow$ new node
  $n^{root}.Id \leftarrow \{1, .., N\}$ # Root contains all instances
  GROW-NODE-RECURSIVE(n^{root})
  $\mathcal{T}_i \leftarrow$ new tree
  $\mathcal{T}_i$.root $\leftarrow n^{root}$
end parallel-for
return $\mathcal{T}_1, .., \mathcal{T}_T$

procedure GROW-NODE-RECURSIVE(n)
  if $|n.Id| \leq$ MaxLeaf **then** # Make n a leaf
    $n.\mathbf{P} \leftarrow$PROCESS-LEAF($\{\mathbf{x}_i, \mathbf{y}_i\}_{i=1}^N, n$)
  else # Split node and grow child nodes recursively
    $\{n.\mathbf{w}, n.\text{left\_child}, n.\text{right\_child}\}$
      $\leftarrow$ SPLIT-NODE($\{\mathbf{x}_i, \mathbf{y}_i\}_{i=1}^N, n$)
    GROW-NODE-RECURSIVE(n.left_child)
    GROW-NODE-RECURSIVE(n.right_child)
  end if
end procedure

procedure PROCESS-LEAF($\{\mathbf{x}_i, \mathbf{y}_i\}_{i=1}^N, n$)
  $\mathbf{P} \leftarrow$ top-k $\left(\frac{\sum_{i \in n.Id} \mathbf{y}_i}{|n.Id|}\right)$
return P # Return scores for top k labels
end procedure

Predictions are made by returning the ranked list of most frequently occurring active labels in all the leaf nodes in the ensemble containing the novel point. Algorithms 1 and 3 present pseudo-code for FastXML training and prediction respectively.

3.2 Learning to partition a node

Training FastXML consists of recursively partitioning a parent's feature space between its children. Such a node partition should ideally be learnt by optimizing a global measure of performance such as the ranking predictions induced by the leaf nodes, Unfortunately, optimizing global measures can be expensive as all nodes in the tree would need to be learnt jointly [21]. Existing approaches therefore optimize local measures of performance which depend solely on predictions made by the current node being partitioned. This allows the hierarchy to be learnt node by node starting from the root and going down to the leaves and is more efficient than learning all the nodes jointly.

MLRF and LPSR optimize the Gini index and clustering error as their local measure of performance. Unfortunately, neither measure is particularly well suited for ranking or extreme multi-label applications where correctly predicting the few positive relevant labels is much more important than predicting the vast number of irrelevant ones.

FastXML therefore proposes to learn the hierarchy by directly optimizing a ranking loss function. In particular, it optimizes the normalized Discounted Cumulative Gain (nDCG) [33]. This results in learning superior partitions due to two main reasons. First, nDCG is a measure which is sensitive to both ranking and relevance and therefore ensures that the relevant positive labels are predicted with ranks that are as high as possible. This cannot be guaranteed by rank insensitive measures such as the Gini index or the clustering error. Second, by being rank sensitive, nDCG

Algorithm 2 SPLIT-NODE($\{\mathbf{x}_i, \mathbf{y}_i\}_{i=1}^N, n$)

$Id \leftarrow n.Id$
$\delta_i[0] \sim \{-1, 1\}, \forall i \in Id$ # Random coin tosses
$\mathbf{w}[0] \leftarrow \mathbf{0}, t \leftarrow 0, t_{\mathbf{w}} \leftarrow 0, \mathcal{W}_0 \leftarrow 0$ # Various counters
repeat
  $\mathbf{r}^{\pm}[t+1] \leftarrow \text{rank}_L\left(\sum_{i \in Id} \frac{1}{2}(1 \pm \delta_i[t]) I_L(\mathbf{y}_i)\mathbf{y}_i\right)$
  for $i \in Id$ **do**
    $v^{\pm} \leftarrow C_\delta(\pm 1)\log(1 + e^{\mp \mathbf{w}[t]^\top \mathbf{x}_i}) - C_r I_L(\mathbf{y}_i) \sum_{l=1}^{L}\left(\frac{y_{ir_l^{\pm}[t+1]}}{\log(1+l)}\right)$ # Refer to (5)
    if $v^+ = v^-$ **then**
      $\delta_i[t+1] = \delta_i[t]$
    else
      $\delta_i[t+1] = \text{sign}(v^- - v^+)$
    end if
  end for
  if $\boldsymbol{\delta}[t+1] = \boldsymbol{\delta}[t]$ **then** # Update $\mathbf{w}$
    $\mathbf{w}[t+1] \leftarrow \underset{\mathbf{w}}{\text{argmin}} \|\mathbf{w}\|_1 + C_\delta(\delta_i[t]) \sum_{i \in Id} \log(1 + e^{-\delta_i[t]\mathbf{w}^\top \mathbf{x}_i})$
    $\mathcal{W}_{t_{\mathbf{w}}+1} \leftarrow t+1$ # Store time step of the update
    $t_{\mathbf{w}} \leftarrow t_{\mathbf{w}} + 1$ # We made one more update to $\mathbf{w}$
  else
    $\mathbf{w}[t+1] \leftarrow \mathbf{w}[t]$
  end if
  $t \leftarrow t+1$
until $\boldsymbol{\delta}[\mathcal{W}_{t_{\mathbf{w}}}] = \boldsymbol{\delta}[\mathcal{W}_{t_{\mathbf{w}}-1}]$ # Convergence
$n^+ \leftarrow$ new node, $n^- \leftarrow$ new node
$n^+.Id \leftarrow \{i \in Id : \mathbf{w}[t]^\top \mathbf{x}_i > 0\}$
$n^-.Id \leftarrow \{i \in Id : \mathbf{w}[t]^\top \mathbf{x}_i \leq 0\}$
return $\mathbf{w}[t], n^+, n^-$

can be optimized across all L labels at the current node thereby ensuring that the local optimization is not myopic.

We stick to the notation introduced in the previous Section. it is assumed that the training set can be represented as $\{(\mathbf{x}_i, \mathbf{y}_i)_{i=1}^N\}$ with D dimensional real-valued feature vectors $\mathbf{x}_i \in \mathcal{R}^D$ and L dimensional binary label vectors $\mathbf{y}_i \in \{0,1\}^L$ with $\mathbf{y}_{il} = 1$ if label l is relevant for point i and 0 otherwise. Permutation indices $i_1^{\text{desc}}, \ldots, i_L^{\text{desc}}$ that sort a real-valued vector $\mathbf{y} \in \mathcal{R}^L$ in descending order are defined such that if $j > k$ then $y_{i_j^{\text{desc}}} \geq y_{i_k^{\text{desc}}}$. The $\text{rank}_k(\mathbf{y})$ operator, which returns the indices of the k largest elements of $\mathbf{y}$ ranked in descending order (with ties broken randomly), can then be defined as

$$\text{rank}_k(\mathbf{y}) = [i_1^{\text{desc}}, \ldots, i_k^{\text{desc}}]^\top \tag{1}$$

Let $\Pi(1, L)$ denote the set of all permutations of $\{1, 2, \ldots, L\}$. The Discounted Cumulative Gain (DCG) at k of a ranking $\mathbf{r} \in \Pi(1, L)$ given a ground truth label vector $\mathbf{y}$ with binary levels of relevance is

$$\mathcal{L}_{\text{DCG@}k}(\mathbf{r}, \mathbf{y}) = \sum_{l=1}^{k} \frac{y_{r_l}}{\log(1+l)} \tag{2}$$

Note that the $\log(1+l)$ term ensures that it is beneficial to predict the positive labels with high ranks. Thus, unlike precision (which would have been obtained had the $\log(1+l)$ term been absent), DCG is sensitive to both the ranking and the relevance of predictions. The normalized DCG, or

Algorithm 3 PREDICT($\{\mathcal{T}_1, .. \mathcal{T}_T\}, \mathbf{x}$)

for $i = 1, .., T$ **do**
  $n \leftarrow \mathcal{T}_i$.root
  while n is not a leaf **do**
    $\mathbf{w} \leftarrow n.\mathbf{w}$
    if $\mathbf{w}^\top \mathbf{x} > 0$ **then**
      $n \leftarrow n$.left_child
    else
      $n \leftarrow n$.right_child
    end if
  end while
  $\mathbf{P}_i^{\text{leaf}}(\mathbf{x}) \leftarrow n.\mathbf{P}$
end for
$\mathbf{r}(\mathbf{x}) = \text{rank}_k\left(\frac{1}{T}\sum_{i=1}^{T} \mathbf{P}_i^{\text{leaf}}(\mathbf{x})\right)$
return $\mathbf{r}(\mathbf{x})$

nDCG, is then defined as

$$\mathcal{L}_{\text{nDCG@}k}(\mathbf{r}, \mathbf{y}) = I_k(\mathbf{y}) \sum_{l=1}^{k} \frac{y_{r_l}}{\log(1+l)} \tag{3}$$

$$\text{where } I_k(\mathbf{y}) = \frac{1}{\sum_{l=1}^{\min(k, \mathbf{1}^\top \mathbf{y})} \frac{1}{\log(1+l)}} \tag{4}$$

where $I_k(\mathbf{y})$ is the inverse of the DCG@k of the ideal ranking for $\mathbf{y}$ obtained by predicting the ranks of all of $\mathbf{y}$'s positive labels to be higher than any of its negative ones. This normalizes $\mathcal{L}_{\text{nDCG@}k}$ to lie between 0 and 1 with larger values being better and ensures that nDCG can be used to compare rankings across label vectors with different numbers of positive labels.

FastXML partitions the current node's feature space by learning a linear separator $\mathbf{w}$ such that

$$\begin{aligned} \min \quad & \|\mathbf{w}\|_1 + \sum_i C_\delta(\delta_i)\log(1 + e^{-\delta_i \mathbf{w}^\top \mathbf{x}_i}) \\ & - C_r \sum_i \tfrac{1}{2}(1+\delta_i)\mathcal{L}_{\text{nDCG@}L}(\mathbf{r}^+, \mathbf{y}_i) \\ & - C_r \sum_i \tfrac{1}{2}(1-\delta_i)\mathcal{L}_{\text{nDCG@}L}(\mathbf{r}^-, \mathbf{y}_i) \\ \text{w.r.t.} \quad & \mathbf{w} \in \mathcal{R}^D, \boldsymbol{\delta} \in \{-1, +1\}^L, \mathbf{r}^+, \mathbf{r}^- \in \Pi(1, L) \end{aligned} \tag{5}$$

where i indexes all the training points present at the node being partitioned, $\delta_i \in \{-1, +1\}$ indicates whether point i was assigned to the negative or positive partition and $\mathbf{r}^+$ and $\mathbf{r}^-$ represent the predicted label rankings for the positive and negative partition respectively. C_δ and C_r are user defined parameters which determine the relative importance of the three terms.

The first term in (5) is an ℓ_1 regularizer on $\mathbf{w}$ which ensures that a sparse linear separator is used to define the partition. Given a novel point $\mathbf{x}$, the FastXML trees can therefore be traversed efficiently during prediction depending on the sign of $\mathbf{w}^\top \mathbf{x}$ at each node. The second term is the log loss of $\delta_i \mathbf{w}^\top \mathbf{x}_i$. This term couples $\boldsymbol{\delta}$ and $\mathbf{w}$ as the optimal solution of this term alone is $\delta_i^* = \text{sign}(\mathbf{w}^{*\top} \mathbf{x}_i)$. This makes it likely that points assigned to the positive (negative) partition, i.e. points for which $\delta_i = +1$ ($\delta_i = -1$), will have positive (negative) values of $\mathbf{w}^\top \mathbf{x}_i$. C_δ is relaxed to be a function of δ_i so as to allow different misclassification penalties for the positive and negative points. The third and fourth term in (5) maximize the nDCG@L of the rankings

predicted for the positive and negative partitions, $\mathbf{r}^+$ and $\mathbf{r}^-$ respectively, given the ground truth label vectors $\mathbf{y}_i$ assigned to these partitions. These terms couple $\mathbf{r}^\pm$ to $\boldsymbol{\delta}$ and thus to $\mathbf{w}$. As discussed, maximizing nDCG makes it likely that the relevant positive labels for each point are predicted with ranks as high as possible. As a result, points within a partition are likely to have similar labels whereas points across partitions are likely to have different labels.

Furthermore, it is beneficial to maximize nDCG@L at each node even though the ultimate leaf node rankings will be evaluated at $k \ll L$. This leads to non-myopic decisions at the root and internal nodes. For example, optimizing nDCG at $k = 5$ at the root node of the Wikipedia data set would be equivalent to finding a separator such that all the hundreds of thousands of Wikipedia articles assigned to the positive partition could be accurately labeled with just the five most frequently occurring Wikipedia categories in the positive partition and similarly for the negative partition. Clearly this will not lead to good results at the root node and superior partitions can be learnt by considering all the Wikipedia categories rather than just the top five. Of course, as already pointed out, rank insensitive measures such as precision cannot be optimized at $k = L$ as they become more and more uninformative with increasing k and $\mathcal{L}_{\text{precision@}L}(\mathbf{r}^\pm, \mathbf{y}_i) = 0$ for all points irrespective of the partitioning.

It is also worth noting that (5) allows a label to be assigned to both partitions if some of the points containing the label are assigned to the positive partition and some to the negative. This makes the FastXML trees somewhat robust as the child nodes can potentially recover from mistakes made by the parents [2, 13, 16, 35].

Finally, note that $\boldsymbol{\delta}$ and $\mathbf{r}^\pm$ were deliberately chosen to be independent variables for efficient optimization rather than functions dependent on $\mathbf{w}$. In particular, (5) could have been formulated as an optimization problem in just $\mathbf{w}$ by discarding the log loss term and defining $\delta_i(\mathbf{w}) = \text{sign}(\mathbf{w}^\top \mathbf{x}_i)$ and $\mathbf{r}^\pm(\mathbf{w}) = \text{rank}_L\left(\sum_i \frac{1}{2}(1 \pm \delta_i(\mathbf{w}))I_L(\mathbf{y}_i)\mathbf{y}_i\right)$. Such a formulation would also have been natural but intractable at scale. Direct optimization via efficient techniques such as stochastic sub-gradient descent would not be possible due to the sharp discontinuities in $\boldsymbol{\delta}(\mathbf{w})$ and $\mathbf{r}^\pm(\mathbf{w})$. Furthermore, updates to $\mathbf{w}$ would necessitate expensive updates to $\boldsymbol{\delta}$ and $\mathbf{r}^\pm$. We therefore decouple $\boldsymbol{\delta}$ and $\mathbf{r}$ from $\mathbf{w}$ by treating them as variables for efficient optimization but then couple their optimal values through the objective function. We develop the optimization algorithm for (5) in Section 4.

3.3 Prediction

The objective function defined in (5) can be optimized efficiently and can lead to accurate predictions. A good objective function should, in addition, also lead to balanced partitions in order to ensure efficient prediction.

Given a novel point $\mathbf{x} \in \mathcal{R}^D$, FastXML's top ranked k predictions are given by

$$\mathbf{r}(\mathbf{x}) = \text{rank}_k\left(\frac{1}{T}\sum_{t=1}^{T}\mathbf{P}_t^{\text{leaf}}(\mathbf{x})\right) \tag{6}$$

where T is the number of trees in the FastXML ensemble and $\mathbf{P}_t^{\text{leaf}}(\mathbf{x}) \propto \sum_{i \in S_t^{\text{leaf}}(\mathbf{x})}\mathbf{y}_i$ and $S_t^{\text{leaf}}(\mathbf{x})$ are the label distribution and set of points respectively of the leaf node containing $\mathbf{x}$ in tree t. The average cost of prediction is upper bounded by $O(TDH + T\hat{L} + \hat{L}\log\hat{L})$ where H is the average length of the path traversed by $\mathbf{x}$ in order to reach the leaf nodes in the T trees and $\hat{L}$ is the number of non-zero elements in the vector $\sum_{t=1}^{T}\mathbf{P}_t^{\text{leaf}}(\mathbf{x})$. The cost is dominated by $O(TDH)$ as $\hat{L} \ll DH$. If the FastXML trees are balanced then $H = \log N \approx \log L$ and the overall cost of prediction becomes $O(TD\log L)$ which is logarithmic in the total number of labels.

One might therefore be tempted to add a balancing term to (5) so as to get H as close to $\log N$ as possible. However, this comes at the cost of reduced prediction accuracy as the objective function trades-off accuracy for balance. As it empirically turns out, our proposed nDCG based objective function learns highly balanced trees and a balancing term does not need to be added to (5). Thus, FastXML's predictions can be made accurately in logarithmic time.

4. OPTIMIZING FASTXML

It is well recognized in the learning to rank literature that nDCG is a difficult function to optimize [25,26,31,32] since it is sharply discontinuous with respect to $\mathbf{w}$ and hence standard stochastic sub-gradient descent techniques cannot be applied. FastXML therefore employs an alternate strategy and optimizes (5) using an iterative alternating minimization algorithm. The algorithm is initialized by setting $\mathbf{w} = \mathbf{0}$ and δ_i to be -1 or $+1$ uniformly at random. Each iteration, then, consists of taking three steps. First, $\mathbf{r}^+$ and $\mathbf{r}^-$ are optimized while keeping $\mathbf{w}$ and $\boldsymbol{\delta}$ fixed. This step determines the ranked list of labels that will be predicted by the positive and negative partitions respectively. Second, $\boldsymbol{\delta}$ is optimized while keeping $\mathbf{w}$ and $\mathbf{r}^\pm$ fixed. This step assigns training points in the node to the positive or negative partition. The third step of optimizing $\mathbf{w}$ while keeping $\boldsymbol{\delta}$ and $\mathbf{r}^\pm$ fixed is taken only if the first two steps did not lead to a decrease in the objective function. This is done to speed up training since optimizing with respect to $\boldsymbol{\delta}$ and $\mathbf{r}^\pm$ takes only seconds while optimizing with respect to $\mathbf{w}$ can take minutes. This is the primary reason why (5) was formulated as a function of $\mathbf{w}$, $\boldsymbol{\delta}$ and $\mathbf{r}^\pm$ rather than just $\mathbf{w}$. The algorithm terminates when $\mathbf{r}^\pm$, $\boldsymbol{\delta}$ and $\mathbf{w}$ do not change from one iteration to the next. We prove that the objective decreases strictly in each iteration and that the proposed algorithm terminates in a finite number of iterations. In practice, it was observed on all data sets that the algorithm made rapid progress and yielded state-of-the-art results as soon as a single update to $\mathbf{w}$ had been made. We now detail the steps in each iteration.

4.1 Optimizing with respect to $\mathbf{r}^\pm$

Given $\mathbf{w}$ and $\boldsymbol{\delta}$, the first step in each iteration is to find the optimal rankings $\mathbf{r}^+$ and $\mathbf{r}^-$ that will be predicted by the positive and negative partition respectively. Fixing $\mathbf{w}$ and $\boldsymbol{\delta}$ simplifies (5) to

$$\max_{\mathbf{r}^\pm \in \Pi(1,L)} C_r \sum_i \tfrac{1}{2}(1+\delta_i)\mathcal{L}_{\text{nDCG@}L}(\mathbf{r}^+, \mathbf{y}_i) + C_r \sum_i \tfrac{1}{2}(1-\delta_i)\mathcal{L}_{\text{nDCG@}L}(\mathbf{r}^-, \mathbf{y}_i) \tag{7}$$

which can be compactly expressed as two independent optimization problems

$$\max_{\mathbf{r}^\pm \in \Pi(1,L)} \sum_{i:\delta_i=\pm 1} I_L(\mathbf{y}_i)\sum_{l=1}^{L}\frac{y_{ir_l^\pm}}{\log(1+l)} \tag{8}$$

$$\equiv \max_{\mathbf{r}^{\pm}\in\Pi(1,L)} \sum_{l=1}^{L} \sum_{i:\delta_i=\pm 1} \frac{I_L(\mathbf{y}_i)y_{il}}{\log(1+r_l^{\pm})} \tag{9}$$

$$\equiv \max_{\mathbf{r}^{\pm}\in\Pi(1,L)} \left(\sum_{i:\delta_i=\pm 1} I_L(\mathbf{y}_i)\mathbf{y}_i \right)^{\top} \mathbf{d}^{\pm} \tag{10}$$

where $\mathbf{d}^{\pm}$ is an L-vector such that $d_l^{\pm} = 1/\log(1+r_l^{\pm})$. Since $\mathbf{r}^{+}$ and $\mathbf{r}^{-}$ are permutations of $1, 2, \ldots, L$ it is clear that (10) will be maximized if $r_l^{\pm}$ is chosen as the index of the l^{th} largest value in the vector $\sum_{i:\delta_i=\pm 1} I_L(\mathbf{y}_i)\mathbf{y}_i$. Thus

$$\mathbf{r}^{\pm^*} = \text{rank}_L \left(\sum_{i:\delta_i=\pm 1} I_L(\mathbf{y}_i)\mathbf{y}_i \right). \tag{11}$$

Note that the optimal values of $\mathbf{r}^{\pm}$ can be computed efficiently in time $O(n \log L + \hat{L} \log \hat{L})$ where n is the number of training points present in the node being partitioned, $\hat{L}$ is the sparsity of the vector $\sum_i I_L(\mathbf{y}_i)\mathbf{y}_i$ and it is assumed that $\mathbf{y}_i$ is $\log L$-sparse.

4.2 Optimizing with respect to $\boldsymbol{\delta}$

The next step in an iteration is to optimize (5) with respect to $\boldsymbol{\delta}$ while keeping $\mathbf{w}$ and $\mathbf{r}^{\pm}$ fixed. This reduces to

$$\begin{aligned} \min_{\boldsymbol{\delta}\in\{-1,+1\}^L} \quad & \sum_i C_\delta(\delta_i) \log(1 + e^{-\delta_i \mathbf{w}^\top \mathbf{x}_i}) \\ & - C_r \sum_i \tfrac{1}{2}(1+\delta_i)\mathcal{L}_{\text{nDCG}@L}(\mathbf{r}^{+}, \mathbf{y}_i) \\ & - C_r \sum_i \tfrac{1}{2}(1-\delta_i)\mathcal{L}_{\text{nDCG}@L}(\mathbf{r}^{-}, \mathbf{y}_i) \end{aligned} \tag{12}$$

which decomposes over i. Thus, each δ_i can be optimized independently by seeing whether (12) is optimized by $\delta_i^* = +1$ or -1. This yields

$$\delta_i^* = \text{sign}(v_i^- - v_i^+) \quad \text{where} \tag{13}$$

$$v_i^{\pm} = C_\delta(\pm 1) \log(1 + e^{\mp \mathbf{w}^\top \mathbf{x}_i}) - C_r I_L(\mathbf{y}_i) \sum_{l=1}^{L} \frac{y_{ir_l^{\pm}}}{\log(1+l)}$$

The time complexity of obtaining $\boldsymbol{\delta}^*$ is $O(n\hat{D} + n \log L)$ assuming that the feature vectors are $\hat{D}$-sparse and the label vectors are $\log L$-sparse. This reduces to $O(n \log L)$ since $\mathbf{w} = \mathbf{0}$ in the first iteration and $\mathbf{w}^\top \mathbf{x}_i$ can be cached for all points in subsequent updates of $\mathbf{w}$.

4.3 Optimizing with respect to w

The final step of optimizing (5) with respect to $\mathbf{w}$ while keeping $\boldsymbol{\delta}$ and $\mathbf{r}^{\pm}$ fixed is carried out only if the first two steps did not make any progress in decreasing the objective function. This can be efficiently determined in time $O(n)$ by checking that $\boldsymbol{\delta}$ has remained unchanged. Optimizing (5) with respect to $\mathbf{w}$ while keeping $\boldsymbol{\delta}$ and $\mathbf{r}^{\pm}$ fixed is equivalent to solving the standard ℓ_1 regularized logistic regression problem with the labels given by $\boldsymbol{\delta}$

$$\min_{\mathbf{w}\in\mathcal{R}^D} \quad \|\mathbf{w}\|_1 + \sum_i C_\delta(\delta_i) \log(1 + e^{-\delta_i \mathbf{w}^\top \mathbf{x}_i}) \tag{14}$$

This problem has been extensively studied in the literature [3,24,37]. FastXML optimizes (14) using the newGLMNET algorithm [37] as implemented in the Liblinear package [14]. No tuning of the learning rate parameter is required since (14) is optimized in the dual. The algorithm is terminated after 10 passes over the training set in case it hasn't converged already. The overall time complexity of the method is $O(n\hat{D})$ where n is the number of training points in the node being partitioned and $\hat{D}$ is the average number of non-zero entries in a feature vector. This is the most time consuming of the three steps.

4.4 Finite termination

The formulation in (5) is non-convex, non-smooth and has a mix of discrete and continuous variables. Furthermore, even the sub-problems obtained by optimizing with respect to only one block of variables might not be convex or smooth. It is well recognized that alternating minimization based techniques can fail to converge for such hard problems in general [6]. However, in our case, it is straightforward to show that FastXML's alternating minimization algorithm for optimizing (5) will not oscillate and will converge in a finite number of iterations.

THEOREM 1. *Suppose Algorithm 2 has not yet terminated and let $\mathcal{W} = \langle \mathcal{W}_1, \mathcal{W}_2, \ldots, \mathcal{W}_i, \ldots \rangle$ be the sequence of iterations at which $\mathbf{w}$ is updated. Let $\overline{\mathcal{W}_i} = \mathcal{W} - \mathcal{W}_i$ be the sequence obtained by removing $\mathcal{W}_i$ from $\mathcal{W}$. Furthermore, let $\mathcal{W}_i \leq t < \mathcal{W}_{i+1}$. Then (a) $\boldsymbol{\delta}[\mathcal{W}_i] \notin \{\boldsymbol{\delta}[\mathcal{W}_j] | \mathcal{W}_j \in \overline{\mathcal{W}_i}\}$; and (b) $\boldsymbol{\delta}[t] \notin \{\boldsymbol{\delta}[j] | \mathcal{W}_i \leq j \neq t < \mathcal{W}_{i+1}\}$.*

PROOF. Let the objective value in (5), after iteration i, be $O[i]$. The individual sub-problems (10,12,14), that comprise a single iteration of 2, are all minimization problems, which minimize (5) w.r.t a single block of variables, and hence can never increase the objective value. In addition, algorithm 2 also ensures that any change in $\boldsymbol{\delta}$ is accompanied by a non-zero decrease in the objective.

(a) Let $\mathcal{W}_i$ be the iteration at which $\mathbf{w}$ is updated for the i^{th} time. The algorithm ensures that, when $\boldsymbol{\delta} = \boldsymbol{\delta}[\mathcal{W}_i]$, $\mathbf{w}[\mathcal{W}_i]$ is the minimizer of (14), and $\mathbf{r}^{\pm}[\mathcal{W}_i]$ is the minimizer of (10), which together imply that $O[\mathcal{W}_i]$ is the unique minimum value of (5) when optimized over $\mathbf{w}, \mathbf{r}^{\pm}$, while fixing $\boldsymbol{\delta} = \boldsymbol{\delta}[\mathcal{W}_i]$.
Hence, for a given $\mathcal{W}_i$ and a $\mathcal{W}_j \in \overline{\mathcal{W}_i}$

$$(\boldsymbol{\delta}[\mathcal{W}_i] = \boldsymbol{\delta}[\mathcal{W}_j]) \implies (O[\mathcal{W}_i] = O[\mathcal{W}_j]) \tag{15}$$

Without loss of generality, let $\mathcal{W}_i < \mathcal{W}_j$. Since by assumption, the algorithm has not yet terminated, $\boldsymbol{\delta}[\mathcal{W}_i] \neq \boldsymbol{\delta}[\mathcal{W}_{(i+1)}]$. But, since there has been an update to $\boldsymbol{\delta}$, $O[\mathcal{W}_i] > O[\mathcal{W}_{(i+1)}] \geq O[\mathcal{W}_j]$. This, combined with (15) gives us $\boldsymbol{\delta}[\mathcal{W}_i] \neq \boldsymbol{\delta}[\mathcal{W}_j]$.

(b) Let $u \in \{j : \mathcal{W}_i \leq j \neq t \leq \mathcal{W}_{i+1}\}$. Without loss of generality, assume $t < u$. Since we are between two $\mathbf{w}$ updates, $\mathbf{w}[t] = \mathbf{w}[u]$. If $\boldsymbol{\delta}[t] = \boldsymbol{\delta}[u]$, then using $\mathbf{w}[t] = \mathbf{w}[u]$, we essentially solve the same optimization w.r.t $\mathbf{r}^{\pm}$ and $\boldsymbol{\delta}$ in steps $t+1$ and $u+1$. Hence, $O[t+1] = O[u+1]$. But, absence of $\mathbf{w}$ updates imply that $\boldsymbol{\delta}[t+1] \neq \boldsymbol{\delta}[t+2]$, which further implies $O[t+1] > O[t+2] \geq O[u+1]$, contradicting the earlier equality. Hence, $\boldsymbol{\delta}[t] \neq \boldsymbol{\delta}[u]$. □

Theorem 1 (b) states that $\boldsymbol{\delta}$ cannot repeat between two consecutive updates to $\mathbf{w}$ (in iterations $\mathcal{W}_i$ and $\mathcal{W}_{i+1}$). Since $\boldsymbol{\delta}$ can only take a finite number of values, this implies the number of iterations between $\mathcal{W}_i$ and $\mathcal{W}_{i+1}$ is bounded. Similarly, Theorem 1(a) states that $\boldsymbol{\delta}[\mathcal{W}_i]$ can never repeat for values $\mathcal{W}_j \in \mathcal{W}$. By a similar argument as above and Theorem 1(b), we conclude that $\mathcal{W}_i$ is bounded for all i. Thus, the proposed alternating minimization algorithm cannot oscillate and terminates in a finite number of iterations.

Table 1: Data set statistics

Data set	Train N	Features D	Labels L	Test M	Avg. labels per pt.	Avg. pts per label
BibTeX	4880	1836	159	2515	2.40	111.71
Delicious	12920	500	983	3185	19.02	311.61
MediaMill	30993	120	101	12914	4.38	1902.16
RCV1-X	781265	47236	2456	23149	4.61	1510.13
WikiLSHTC	1892600	1617899	325056	472835	3.26	23.74
Ads-430K	1118084	87890	434594	502926	2.10	7.86
Ads-1M	3917928	164592	1082898	1563137	1.96	7.07

Table 2: Results on small and medium data sets. FastXML was run with default hyper-parameter settings on all data sets. FastXML-T presents results when the parameters were tuned.

(a) BibTeX $N = 4.8K, D = 1.8K, L = 159$

Algorithm	P1 (%)	P3 (%)	P5 (%)
FastXML-T	$\mathbf{64.53 \pm 0.72}$	$\mathbf{40.17 \pm 0.63}$	$\mathbf{29.27 \pm 0.53}$
FastXML	63.26 ± 0.84	39.19 ± 0.66	28.72 ± 0.48
MLRF	62.81 ± 0.84	38.74 ± 0.69	28.45 ± 0.43
LPSR	62.95 ± 0.70	39.16 ± 0.64	28.75 ± 0.45
1-vs-All	63.39 ± 0.64	39.55 ± 0.65	29.13 ± 0.45

(b) Delicious $N = 13K, D = 500K, L = 983$

Algorithm	P1 (%)	P3 (%)	P5 (%)
FastXML	$\mathbf{69.65 \pm 0.82}$	$\mathbf{63.93 \pm 0.50}$	$\mathbf{59.36 \pm 0.57}$
MLRF	67.86 ± 0.70	62.02 ± 0.55	57.59 ± 0.43
LPSR	65.55 ± 0.99	59.39 ± 0.48	53.99 ± 0.31
1-vs-All	65.42 ± 1.05	59.34 ± 0.56	53.72 ± 0.50

(c) MediaMill $N = 30K, D = 120, L = 101$

Algorithm	P1 (%)	P3 (%)	P5 (%)
FastXML	$\mathbf{87.35 \pm 0.27}$	$\mathbf{72.14 \pm 0.20}$	$\mathbf{58.15 \pm 0.15}$
MLRF	86.83 ± 0.18	71.18 ± 0.19	57.09 ± 0.16
LPSR	82.33 ± 2.15	66.37 ± 0.35	50.00 ± 0.20
1-vs-All	82.31 ± 2.19	66.17 ± 0.43	50.32 ± 0.56

(d) RCV1-X $N = 781K, D = 47K, L = 2.5K$

Algorithm	P1 (%)	P3 (%)	P5 (%)
FastXML	$\mathbf{91.23 \pm 0.22}$	$\mathbf{73.51 \pm 0.25}$	$\mathbf{53.31 \pm 0.65}$
MLRF	87.66 ± 0.46	69.89 ± 0.43	50.36 ± 0.74
LPSR	90.04 ± 0.19	72.27 ± 0.20	52.34 ± 0.61
1-vs-All	90.18 ± 0.18	72.55 ± 0.16	52.68 ± 0.57

Table 3: Results on large data sets comparing the performance of FastXML to LPSR trained with Naïve Bayes as the base classifier.

(a) WikiLSHTC $N = 1.78M, D = 1.62M, L = 325K$

Algorithm	P1 (%)	P3 (%)	P5 (%)	Train Time (hr)	Test Time (min)
FastXML	**49.78**	**33.06**	**24.40**	9.14	5.10
LPSR-NB	27.91	16.04	11.57	**1.59**	**3.52**

(b) Ads-430K $N = 1.12M, D = 88K, L = 0.43M$

Algorithm	P1 (%)	P3 (%)	P5 (%)	Train Time (hr)	Test Time (min)
FastXML	**27.24**	**16.28**	**11.91**	1.81	**1.68**
LPSR-NB	19.69	12.71	9.70	**0.84**	3.95

(c) Ads-1M $N = 3.91M, D = 165K, L = 1.08M$

Algorithm	P1 (%)	P3 (%)	P5 (%)	Train Time (hr)	Test Time (min)
FastXML	**23.45**	**14.21**	**10.41**	8.09	**6.26**
LPSR-NB	17.08	11.38	8.83	**3.78**	19.32

Table 4: FastXML's wall clock training time (in hours) vs the number of cores used on a single machine.

Cores	WikiLSHTC (hr)	Ads-430K (hr)	Ads-1M (hr)
1	16.80 (1.00×)	2.46 (1.00×)	12.34 (1.00×)
2	8.62 (1.94×)	1.24 (1.98×)	7.09 (1.74×)
4	4.53 (3.71×)	0.62 (3.97×)	3.93 (3.13×)
8	2.21 (7.60×)	0.33 (7.45×)	2.09 (5.90×)
16	1.27 (13.22×)	0.19 (12.95×)	1.02 (12.10×)

Table 5: The variation in FastXML's performance with the number of training iterations. $\mathcal{W}_i$ denotes the iteration at which $\mathbf{w}$ is updated for the i^{th} time at the root node on the Ads-430K data set. Precision values and training times are reported for the full ensemble.

i	$\mathcal{W}_i$	Objective value	Train time (hr)	P1 (%)	P3 (%)	P5 (%)
1	15	403419.87	1.88	27.21	16.28	11.89
2	21	237151.20	3.79	27.01	16.39	12.09
3	24	183542.04	5.60	26.95	16.37	12.10
4	26	163416.65	7.33	26.98	16.38	12.11
5	29	153592.21	8.99	26.93	16.38	12.11

5. EXPERIMENTS

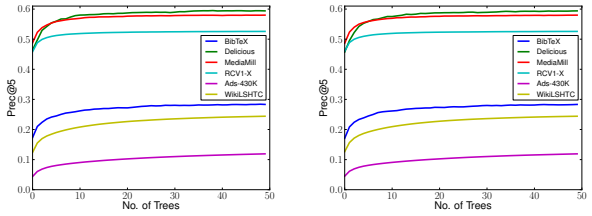
This Section compares the performance of FastXML to MLRF, LPSR and the 1-vs-All baseline on some of the largest multi-label classification data sets.

Data sets: Experiments were carried out on data sets with label set sizes ranging from a hundred to a million to benchmark the performance of FastXML in various regimes. The data sets include two small scale data sets with hundreds of labels, BibTeX [23] and MediaMill [28], two medium scale data sets with thousands of labels, Delicious [30] and RCV1-X, and three large scale data sets with up to a million labels, WikiLSHTC [1], Ads430K and Ads1M. Table 1 lists the statistics of these data sets.

Features and labels are publically available for all the data sets, apart from the two proprietary Ads data sets, and were used in the experiments. WikiLSHTC is a challenge data set for which the test set has not been released and we therefore partitioned the data provided into 75% for training and 25% for testing. The RCV1-X data set has the same features as the original RCV1 data set but its label set has been expanded by forming new labels from pairs of original labels. The two proprietary Ads data sets comprise of bag of words TF-IDF features extracted from expansions of queries from the Bing query logs. Other queries similar to a given query are used as labels for that query.

Table 6: FastXML learns more stable and balanced trees than MLRF and LPSR leading to both faster training as well as faster prediction. Tree balance is measured as $H/\log(N/\text{MaxLeaf})$, where H is the average length of the path traversed by a point in that tree and $\log(N/\text{MaxLeaf})$ is the average length of a path traversed in a perfectly balanced tree with at most MaxLeaf points at each leaf node. Smaller values of tree balance are better with a balance of 1 indicating a perfectly balanced tree.

Data set	Tree Balance: $\frac{H}{log(N/MaxLeaf)}$			Avg. labels per
	FastXML	MLRF	LPSR	leaf for FastXML
BibTeX	$\mathbf{1.02 \pm 0.01}$	3.45 ± 0.01	1.56 ± 0.11	6.15
Delicious	$\mathbf{1.03 \pm 0.01}$	4.95 ± 0.01	3.14 ± 0.71	69.12
MediaMill	$\mathbf{1.01 \pm 0.01}$	1.59 ± 0.01	1.06 ± 0.01	10.26
RCV1-X	$\mathbf{1.02 \pm 0.01}$	5.62 ± 0.15	1.32 ± 0.02	18.73
WikiLSHTC	$\mathbf{1.01 \pm 0.01}$	-	3.69 ± 0.01	13.36
Ads-430K	$\mathbf{1.00 \pm 0.01}$	-	94.71 ± 0.01	10.97
Ads-1M	$\mathbf{1.00 \pm 0.01}$	-	105.83 ± 0.01	11.03



(a) Random order (b) Forward sequential

Figure 1: The variation in FastXML's precision at 5 with the number of trees selected according to (1a) random order; and (1b) highest individual prediction accuracy on the training set. The training time can be halved on most data sets with a minimal decrease in prediction accuracy by training only 25 trees in random order.

Results are reported by averaging over ten random train and test splits for the small and medium data sets apart from RCV1-X for which only 3 splits were used. A single split was used for the large data sets.

Baseline algorithms: FastXML was compared to state-of-the-art tree based extreme multi-label methods such as MLRF and LPSR (see Section 2 for details) as well as the 1-vs-All baseline as implemented in M3L [18]. The 1-vs-All strategy was also used to learn the base classifier for LPSR on the small and medium data sets as it offered better performance as compared to other base classifiers such as Wsabie [34]. Unfortunately, M3L and Wsabie cannot be trained on the large data sets on a single desktop in a day. Naïve Bayes was therefore used as a base classifier for LPSR on these data sets.

Finally, note that we do not compare explicitly to embedding methods [4, 8, 10, 12, 15, 19, 20, 22, 29, 34, 36, 38] since none of these have been shown to consistently outperform the 1-vs-All base classifier.

Parameters: The following hyper-parameters settings were used for FastXML across all data sets: (a) Co-efficient of logistic-loss: $C_\delta = 1.0$; (b) Co-efficient of negative-nDCG loss: $C_r = 1.0$; (c) Number of trees: $T = 50$; (d) Maximum number of instances allowed in a leaf node: MaxLeaf = 10; (e) Number of labels in a leaf node whose probability scores are retained: $k = 20$; (f) Bias multiplier for Liblinear: 1.0; (g) Number of training iterations in Algorithm 2: 1; and (h) Maximum number of (outer,inner) iterations of Liblinear before termination: (10, 10). Using these default settings, it was observed that FastXML could outperform LPSR, MLRF and the 1-vs-All M3L. Tuning the hyper-parameters for each data set would lead to even superior prediction accuracies. The hyper-parameters for MLRF, LPSR and M3L were set

Table 7: Results obtained by replacing the nDCG@L loss function in FastXML with others such as nDCG@5 (FastXML-nDCG5) or precision at 5 (FastXML-P5) and by replacing the Gini index in MLRF with the proposed nDCG@L loss function.

(a) BibTeX $N = 4.8K, D = 1.8K, L = 159$

Algorithm	P1 (%)	P3 (%)	P5 (%)
FastXML	$\mathbf{63.26 \pm 0.84}$	$\mathbf{39.19 \pm 0.66}$	$\mathbf{28.72 \pm 0.48}$
FastXML-P5	39.95 ± 1.09	23.01 ± 0.46	17.42 ± 0.36
FastXML-nDCG5	51.75 ± 1.28	30.87 ± 0.63	22.70 ± 0.42
MLRF-nDCG	58.41 ± 1.20	36.45 ± 0.65	26.95 ± 0.49

(b) Delicious $N = 13K, D = 500K, L = 983$

Algorithm	P1 (%)	P3 (%)	P5 (%)
FastXML	$\mathbf{69.65 \pm 0.82}$	$\mathbf{63.93 \pm 0.50}$	$\mathbf{59.36 \pm 0.57}$
FastXML-P5	60.11 ± 0.91	53.97 ± 0.53	49.81 ± 0.58
FastXML-nDCG5	64.96 ± 0.83	59.28 ± 0.69	54.70 ± 0.56
MLRF-nDCG	66.70 ± 0.75	61.08 ± 0.44	56.72 ± 0.44

(c) MediaMill $N = 30K, D = 120, L = 101$

Algorithm	P1 (%)	P3 (%)	P5 (%)
FastXML	$\mathbf{87.35 \pm 0.27}$	$\mathbf{72.14 \pm 0.20}$	$\mathbf{58.15 \pm 0.15}$
FastXML-P5	80.73 ± 0.29	67.48 ± 0.22	54.11 ± 0.20
FastXML-nDCG5	86.66 ± 0.27	70.81 ± 0.21	56.51 ± 0.20
MLRF-nDCG	86.67 ± 0.26	71.13 ± 0.22	57.24 ± 0.21

(d) RCV1-X $N = 781K, D = 47K, L = 2.5K$

Algorithm	P1 (%)	P3 (%)	P5 (%)
FastXML	$\mathbf{91.23 \pm 0.22}$	$\mathbf{73.51 \pm 0.25}$	$\mathbf{53.31 \pm 0.65}$
FastXML-P5	66.62 ± 0.23	56.41 ± 0.40	40.83 ± 0.14
FastXML-nDCG5	75.60 ± 0.39	60.85 ± 0.43	43.98 ± 0.16
MLRF-nDCG	87.19 ± 0.41	69.69 ± 0.46	50.25 ± 0.56

(e) WikiLSHTC $N = 1.78M, D = 1.62M, L = 325K$

Algorithm	P1 (%)	P3 (%)	P5 (%)
FastXML	**49.78**	**33.06**	**24.40**
FastXML-P5	18.74	12.33	8.71
FastXML-nDCG5	20.50	12.51	8.80

(f) Ads-430K $N = 1.12M, D = 88K, L = 0.43M$

Algorithm	P1 (%)	P3 (%)	P5 (%)
FastXML	**27.24**	**16.28**	**11.91**
FastXML-P5	25.06	14.36	10.27
FastXML-nDCG5	24.93	14.34	10.29

(g) Ads-1M $N = 3.91M, D = 165K, L = 1.08M$

Algorithm	P1 (%)	P3 (%)	P5 (%)
FastXML	**23.45**	**14.21**	**10.41**
FastXML-P5	21.09	12.32	8.87
FastXML-nDCG5	21.25	12.47	9.01

using fine grained validation on each data set so as to achieve the highest possible prediction accuracy for each method.

Evaluation Metrics: Extreme multi-label classification data sets exhibit positive label sparsity in that each data point has only a few positive labels associated with it. It therefore becomes important to focus on the accurate prediction of the few positive labels per data point than on the vast number of negative ones. As such, most papers [2, 19, 22, 34–36] evaluate prediction accuracy using precision at k which counts the number of correct predictions in the top k positive predictions. More formally, the precision at k for a prediction $\hat{\mathbf{y}} \in \mathcal{R}^L$ given the ground truth label vector $\mathbf{y} \in \{0,1\}^L$ is defined as

$$\mathrm{P}k(\hat{\mathbf{y}}, \mathbf{y}) = \frac{1}{k} \sum_{i \in \mathrm{rank}_k(\hat{\mathbf{y}})} \mathbf{y}_i. \tag{16}$$

All experiments were carried out on a single core of an Intel Core 2 Duo machine running at 3.3 GHz with 8 Gb of RAM. Training times were measured using the `clock` function available in C++. Parallelization experiments in Table 4 were carried out on multiple cores of Intel Xeon processors running at 2.1 GHz.

Results on small and medium data sets: Table 2 benchmarks the performance of FastXML on the small and medium data sets where accurate MLRF, LPSR and 1-vs-All models could be learnt. The focus is on prediction accuracy as neither training nor prediction present any challenge on these data sets – even the expensive MLRF, LPSR and 1-vs-All can be trained in seconds on BibTeX and Delicious, in minutes on MediaMill and in two hours on RCV1-X.

As compared to LPSR and 1-vs-All, FastXML could be up to 5% more accurate in terms of precision at 1 and up to 8% in terms of precision at 5. Large improvements were obtained on both Delicious and MediaMill. The smallest improvements were obtained on BibTeX which had less than five thousand training points. FastXML with default parameter settings gave more or less the same P1 as LPSR and 1-vs-All on BibTeX. A marginal improvement of 1% over the other methods could be obtained by tuning FastXML's hyper-parameters (referred to as FastXML-T in Table 2). FastXML's gains over MLRF were smaller as compared to the gains over LPSR and 1-vs-All but could still go up to 3% in terms of both precision at 1 and at 5 (on RCV1-X). All in all, these results indicate that FastXML could be significantly more accurate at prediction than highly tuned MLRF, LPSR and 1-vs-All classifiers.

Results on large data sets: Extreme classification is most concerned with large scale data sets having hundreds of thousands or even millions of labels. Training MLRF and 1-vs-All on such data sets was found to be infeasible without using a large cluster. LPSR training could be made tractable on a single core by replacing the 1-vs-All base classifier with Naïve Bayes. Table 3 compares the performance of FastXML to LPSR-NB. FastXML's improvements over LPSR-NB in terms of P1 ranged from 5% on Ads-1M to almost 22% on WikiLSHTC and in terms of P5 ranged from approximately 1.5% on Ads-1M to almost 13% on WikiLSHTC. FastXML could train in 1.81 hours on Ads-430K using a single core and in 8 to 9 hours on Ads-1M and WikiLSHTC with individual trees being grown in about 10 minutes. This opens up the possibility of practitioners training accurate extreme classification models on commodity hardware. Finally, FastXML could be almost 1.5 to 3 times faster at prediction than LPSR-NB which could be a critical factor in certain applications.

Validating FastXML's hyper-parameter settings and design choices: Table 4 lists the reduction in wall clock training time obtained by growing FastXML's trees in parallel across multiple cores on a single machine. Training could be speeded up 12 to 13 times by utilizing 16 cores demonstrating that FastXML is trivially parallelizable. The entire ensemble could be trained in approximately an hour on Ads-1M and WikiLSHTC and in 12 minutes on Ads-430K.

All the FastXML results presented so far were obtained by terminating the proposed optimization algorithm after a single update to $\mathbf{w}$. Table 5 shows the effects of allowing multiple updates to $\mathbf{w}$ while training on the Ads-430K data set. High precisions were reached after the first update to $\mathbf{w}$ which occurred after 15 updates to $\boldsymbol{\delta}$ and $\mathbf{r}$. Subsequent updates to $\mathbf{w}$, $\boldsymbol{\delta}$ and $\mathbf{r}$ yielded a significant drop in the value of the objective function but little change in prediction accuracy. As such, it would appear that training time could be significantly reduced without much loss in precision by early termination.

Table 7 reports the effects of replacing the proposed nDCG@L loss function in FastXML with others such as precision at 5 (FastXML-P5) and nDCG@5 (FastXML-nDCG5). As is evident, both these loss functions were inferior to nDCG@L even when performance was measured using precision at 5. This demonstrates that rank sensitive loss functions such as nDCG are better suited to extreme multi-label classification as compared to rank insensitive ones such as precision. Furthermore, nDCG should be computed non-myopically over all labels rather than just the top few.

Finally, the key difference in FastXML's formulation as compared to MLRF's was the use of the nDCG based loss function and the use of a linear separator to partition each node. Table 7 demonstrates that both these ingredients were necessary as simply replacing the Gini index or entropy in MLRF with the nDCG based loss function (MLRF-nDCG) yielded significantly poorer results as compared to FastXML.

6. CONCLUSIONS

This paper developed the FastXML algorithm for multi-label learning with a large number of labels. FastXML learnt an ensemble of trees with prediction costs that were logarithmic in the number of labels. The key technical contribution in FastXML was a novel node partitioning formulation which optimized an nDCG based ranking loss over all the labels. Such a loss was found to be more suitable for extreme multi-label learning than the Gini index optimized by MLRF or the clustering error optimized by LPSR. nDCG is known to be a hard loss to optimize using gradient descent based techniques. FastXML therefore developed an efficient alternating minimization algorithm for its optimization. It was proved that the proposed alternating minimization algorithm would not oscillate and would converge in a finite number of iterations. Experiments revealed that FastXML could be significantly more accurate than MLRF and LPSR while efficiently scaling to problems with more than a million labels. The FastXML code is publically available and should enable practitioners to train accurate extreme multi-label models without needing large clusters.

Acknowledgments

We are very grateful to Purushottam Kar and Prateek Jain for helpful discussions. Yashoteja Prabhu is supported by a TCS PhD Fellowship at IIT Delhi.

7. REFERENCES

[1] Wikipedia dataset for the 4th large scale hierarchical text classification challenge. http://lshtc.iit.demokritos.gr/.

[2] R. Agrawal, A. Gupta, Y. Prabhu, and M. Varma. Multi-label learning with millions of labels: Recommending advertiser bid phrases for web pages. In *WWW*, pages 13–24, 2013.

[3] G. Andrew and J. Gao. Scalable training of L_1-regularized log-linear models. In *ICML*, pages 33–40, 2007.

[4] K. Balasubramanian and G. Lebanon. The landmark selection method for multiple output prediction. In *ICML*, 2012.

[5] S. Bengio, J. Weston, and D. Grangier. Label embedding trees for large multi-class tasks. In *NIPS*, 2010.

[6] D. Bertsekas. *Nonlinear Programming*. Athena Scientific, 1999.

[7] W. Bi and J. T.-Y. Kwok. Multilabel classification on tree- and dag-structured hierarchies. In *ICML*, 2011.

[8] W. Bi and J. T.-Y. Kwok. Efficient multi-label classification with many labels. In *ICML*, pages 405–413, 2013.

[9] N. Cesa-Bianchi, C. Gentile, and L. Zaniboni. Incremental algorithms for hierarchical classification. *JMLR*, 7, 2006.

[10] Y.-N. Chen and H.-T. Lin. Feature-aware label space dimension reduction for multi-label classification. In *NIPS*, pages 1538–1546, 2012.

[11] A. Choromanska and J. Langford. Logarithmic time online multiclass prediction. http://arxiv.org/abs/1406.1822, 2014.

[12] M. Cissé, N. Usunier, T. Artières, and P. Gallinari. Robust bloom filters for large multilabel classification tasks. In *NIPS*, pages 1851–1859, 2013.

[13] J. Deng, S. Satheesh, A. C. Berg, and F. Li. Fast and balanced: Efficient label tree learning for large scale object recognition. In *NIPS*, 2011.

[14] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin. LIBLINEAR: A library for large linear classification. *JMLR*, 9:1871–1874, 2008.

[15] C.-S. Feng and H.-T. Lin. Multi-label classification with error-correcting codes. *JMLR*, pages 289–295, 2011.

[16] T. Gao and D. Koller. Discriminative learning of relaxed hierarchy for large-scale visual recognition. In *ICCV*, pages 2072–2079, 2011.

[17] P. Geurts, D. Ernst, and L. Wehenkel. Extremely randomized trees. *ML*, pages 3–42, 2006.

[18] B. Hariharan, S. V. N. Vishwanathan, and M. Varma. Efficient max-margin multi-label classification with applications to zero-shot learning. *ML*, 2012.

[19] D. Hsu, S. Kakade, J. Langford, and T. Zhang. Multi-label prediction via compressed sensing. In *NIPS*, 2009.

[20] S. Ji, L. Tang, S. Yu, and J. Ye. Extracting shared subspace for multi-label classification. In *KDD*, pages 381–389, 2008.

[21] C. Jose, P. Goyal, P. Aggrwal, and M. Varma. Local deep kernel learning for efficient non-linear svm prediction. In *ICML*, June 2013.

[22] A. Kapoor, R. Viswanathan, and P. Jain. Multilabel classification using bayesian compressed sensing. In *NIPS*, 2012.

[23] I. Katakis, G. Tsoumakas, and I. Vlahavas. Multilabel text classification for automated tag suggestion. In *ECML/PKDD Discovery Challenge*, 2008.

[24] K. Koh, S.-J. Kim, and S. Boyd. An interior-point method for large-scale l1-regularized logistic regression. *JMLR*, 8:1519–1555, 2007.

[25] A. Kustarev, Y. Ustinovsky, Y. Logachev, E. Grechnikov, I. Segalovich, and P. Serdyukov. Smoothing ndcg metrics using tied scores. In *CIKM*, pages 2053–2056, 2011.

[26] P. D. Ravikumar, A. Tewari, and E. Yang. On ndcg consistency of listwise ranking methods. In *AISTATS*, pages 618–626, 2011.

[27] J. Rousu, C. Saunders, S. Szedmak, and J. Shawe-Taylor. Kernel-based learning of hierarchical multilabel classification models. *JMLR*, 7, 2006.

[28] C. Snoek, M. Worring, J. van Gemert, J.-M. Geusebroek, and A. Smeulders. The challenge problem for automated detection of 101 semantic concepts in multimedia. In *ACM Multimedia*, pages 421–430, 2006.

[29] F. Tai and H.-T. Lin. Multi-label classification with principal label space transformation. In *Workshop proceedings of learning from multi-label data*, 2010.

[30] G. Tsoumakas, I. Katakis, and I. Vlahavas. Effective and efficient multilabel classification in domains with large number of labels. In *ECML/PKDD 2008 Workshop on Mining Multidimensional Data*, 2008.

[31] H. Valizadegan, R. Jin, R. Zhang, and J. Mao. Learning to rank by optimizing ndcg measure. In *SIGIR*, pages 41–48, 2000.

[32] M. N. Volkovs and R. S. Zemel. Boltzrank: Learning to maximize expected ranking gain. In *ICML*, pages 1089–1096, 2009.

[33] Y. Wang, L. Wang, Y. Li, D. He, and T.-Y. Liu. A theoretical analysis of nDCG type ranking measures. In *COLT*, pages 25–54, 2013.

[34] J. Weston, S. Bengio, and N. Usunier. Wsabie: Scaling up to large vocabulary image annotation. In *IJCAI*, 2011.

[35] J. Weston, A. Makadia, and H. Yee. Label partitioning for sublinear ranking. In *ICML*, volume 28, pages 181–189, 2013.

[36] H.-F. Yu, P. Jain, P. Kar, and I. S. Dhillon. Large-scale multi-label learning with missing labels. *ICML*, 2014.

[37] G.-X. Yuan, C.-H. Ho, and C.-J. Lin. An improved glmnet for l1-regularized logistic regression. *JMLR*, 13:1999–2030, 2012.

[38] Y. Zhang and J. G. Schneider. Multi-label output codes using canonical correlation analysis. In *AISTATS*, pages 873–882, 2011.

Машинное обучение
Ранжирование

https://yandexdataschool.ru/edu-process/courses/machine-learning

Содержание лекции

- Постановка задачи
- Примеры применения
- Оценки качества
- Подходы к решению задачи
  - поточечный
  - попарный
  - списочный
  - многозначная классификация

Постановка задачи

X — множество объектов

$X^\ell = \{x_1, \ldots, x_\ell\}$ — обучающая выборка

$i \prec j$ — правильный порядок на парах $(i, j) \in \{1, \ldots, \ell\}^2$

Задача:

построить ранжирующую функцию $a\colon X \to \mathbb{R}$ такую, что

$$i \prec j \quad \Rightarrow \quad a(x_i) < a(x_j)$$

Линейная модель ранжирования:

$$a(x; w) = \langle x, w \rangle$$

где $x \mapsto \big(f_1(x), \ldots, f_n(x)\big) \in \mathbb{R}^n$ — вектор признаков объекта x

Часто на практике объекты разделяются на группы (list, списки), и их нужно ранжировать в пределах одной группы. При этом число групп велико.

Пример 1. Ранжирование результатов поисковой выдачи

D — коллекция текстовых документов (documents)
Q — множество запросов (queries)
$D_q \subseteq D$ — множество документов, найденных по запросу q
$X = Q \times D$ — объектами являются пары «запрос, документ»:

$$x \equiv (q, d),\ q \in Q,\ d \in D_q$$

Y — упорядоченное множество рейтингов
$y : X \to Y$ — оценки релевантности, поставленные асессорами:
чем выше оценка $y(q, d)$, тем релевантнее документ d запросу q

Правильный порядок определён только между документами, найденными по одному и тому же запросу q:

$$(q, d) \prec (q, d') \quad \Leftrightarrow \quad y(q, d) < y(q, d')$$

Пример 2.
Рекомендательные системы

U — пользователи, users
I — предметы, items (фильмы, книги, и т.п.)
$X = U \times I$ — объектами являются пары «user, item»

Правильный порядок определён между предметами, которые выбирал или рейтинговал один и тот же пользователь:

$$(u, i) \prec (u, i') \quad \Leftrightarrow \quad y(u, i) < y(u, i')$$

Рекомендация пользователю u — это список предметов i, упорядоченный с помощью функции ранжирования $a(u, i)$

В роли признаков объекта $x = (u, i)$ могут выступать
$y(u', i)$ — рейтинги, поставленные другими пользователями u'

Оценки качества

- AUC
- Точность (precision)

$$\text{precision} = \frac{|\{\text{relevant documents}\} \cap \{\text{retrieved documents}\}|}{|\{\text{retrieved documents}\}|}$$

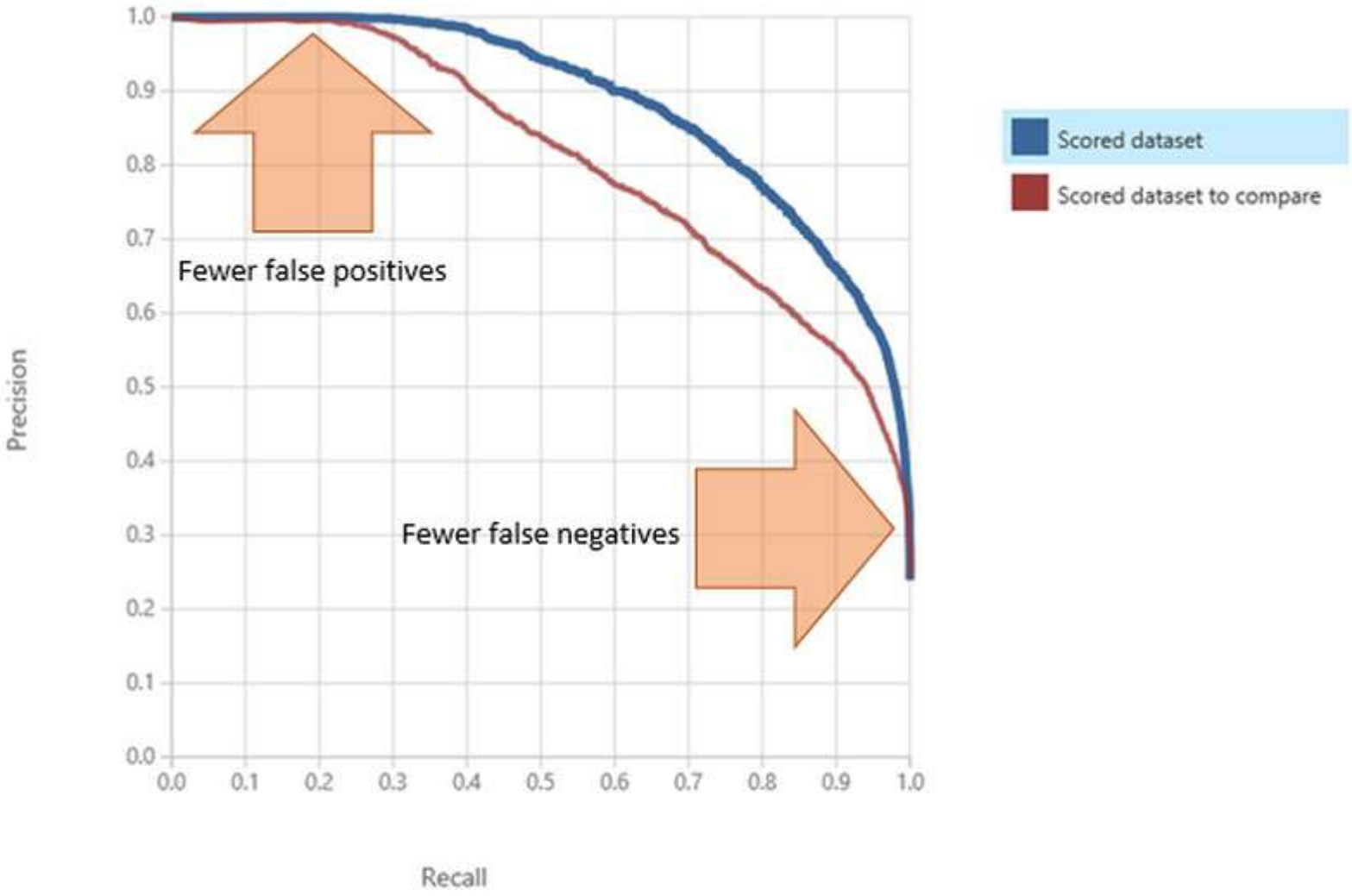
- Полнота (recall, TPR)

$$\text{recall} = \frac{|\{\text{relevant documents}\} \cap \{\text{retrieved documents}\}|}{|\{\text{relevant documents}\}|}$$

- Выпадение (fall-out, FPR) - вероятность нахождения нерелевантного ресурса

$$\text{fall-out} = \frac{|\{\text{non-relevant documents}\} \cap \{\text{retrieved documents}\}|}{|\{\text{non-relevant documents}\}|}$$

PR-кривая



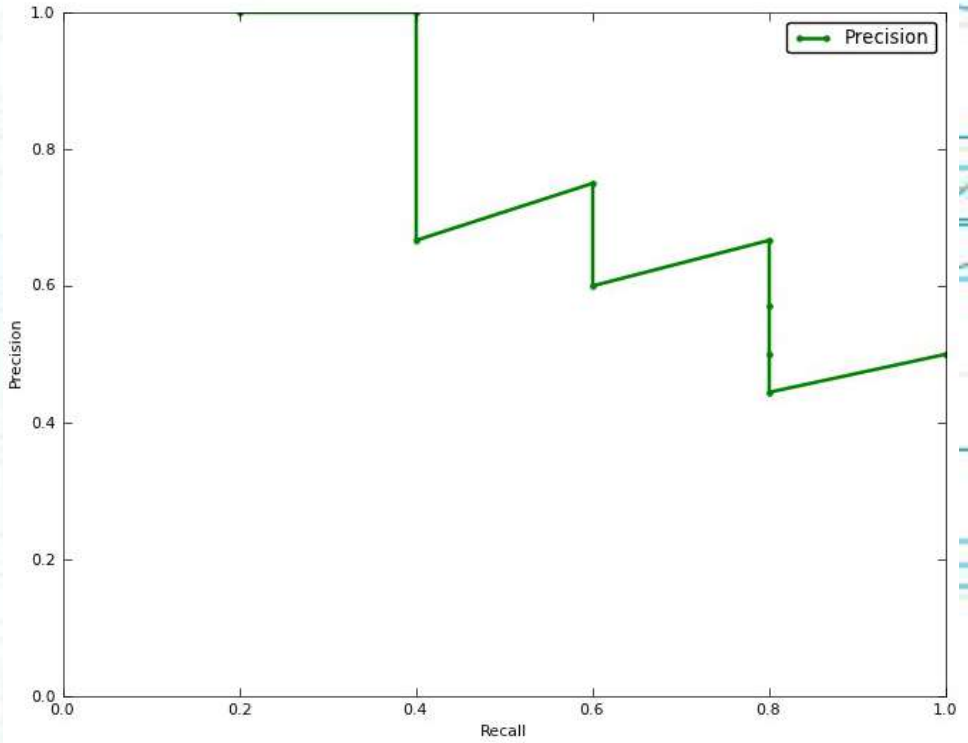
Средняя точность (AveP) – площадь под PR-кривой.

Как выглядит график для случайного порядка?

Пример расчета PR-кривой в задаче определения самолетов

Результат ранжирования

Порог	Precision	Recall
Top 1	100%	20%
Top 2	100%	40%
Top 3	66%	40%
Top 4	75%	60%
Top 5	60%	60%
Top 6	66%	80%
Top 7	57%	80%
Top 8	50%	80%
Top 9	44%	80%
Top 10	50%	100%



Оценки качества

- F-мера (F-measure, мера Ван Ризбергена)

$$F = \frac{2 \cdot \mathrm{precision} \cdot \mathrm{recall}}{(\mathrm{precision} + \mathrm{recall})}$$

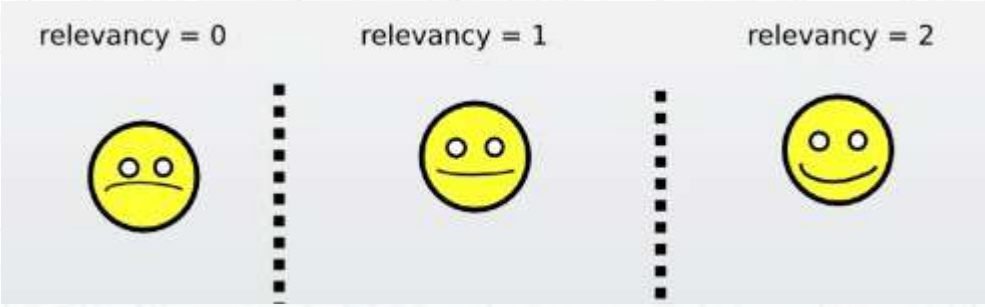
- CG (Cumulative gain) $\mathrm{CG_p} = \sum_{i=1}^{p} rel_i$
- DCG (Discounted cumulative gain)

$$\mathrm{DCG_p} = \sum_{i=1}^{p} \frac{2^{rel_i} - 1}{\log_2(i+1)}$$

- Нормированный DCG (NDCG)

$$\mathrm{nDCG_p} = \frac{DCG_p}{IDCG_p}$$

Пример вычисления nDCG



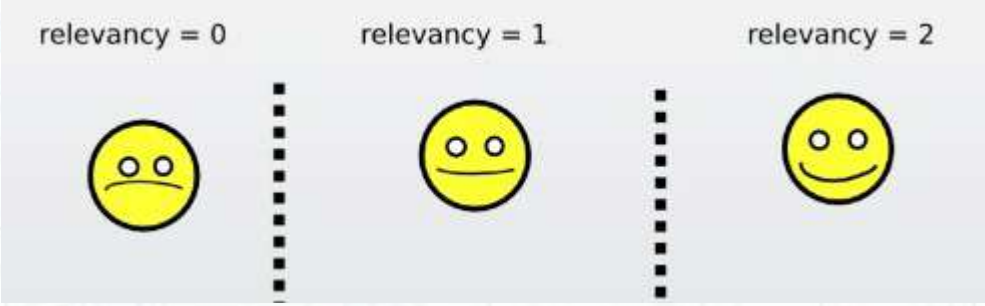
1 http://lemonde.fr
2 http://figaro.fr
3 http://leparisien.fr
4 http://ouestfrance.fr
5 http://liberation.fr

NOT TOO BAD

2 http://figaro.fr
1 http://lemonde.fr
4 http://ouestfrance.fr
3 http://leparisien.fr
5 http://liberation.fr

PERFECT

Пример вычисления nDCG



1 http://lemonde.fr
2 http://figaro.fr
3 http://leparisien.fr
4 http://ouestfrance.fr
5 http://liberation.fr

NOT TOO BAD

2 http://figaro.fr
1 http://lemonde.fr
4 http://ouestfrance.fr
3 http://leparisien.fr
5 http://liberation.fr

PERFECT

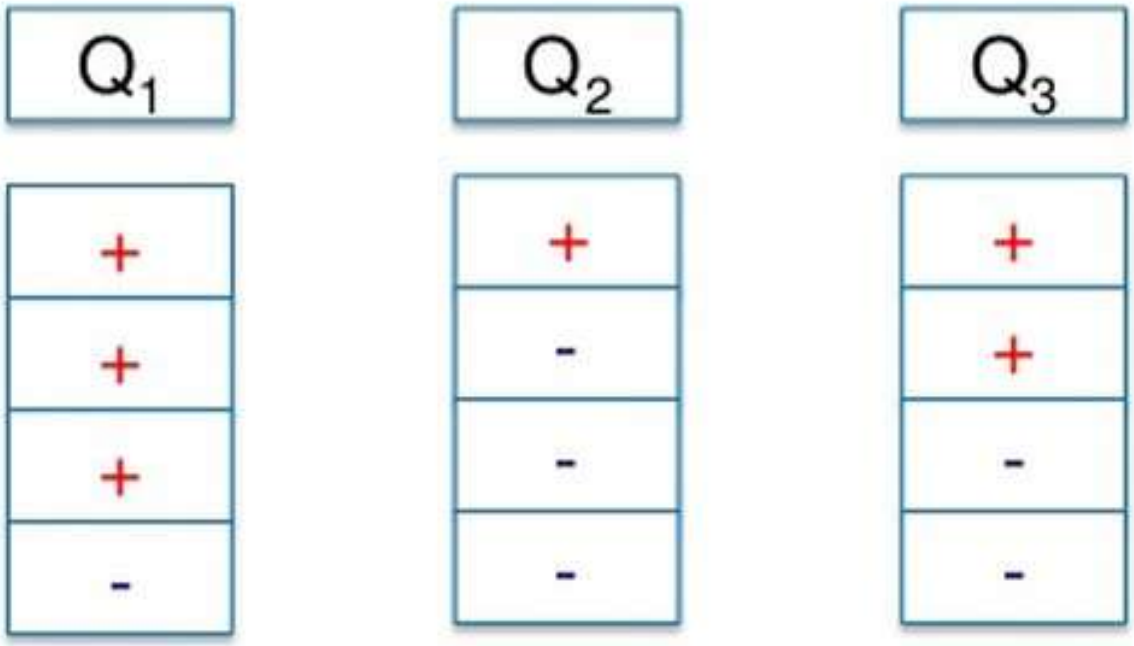
DCG = 1/log(2) + 3/log(3) + 0 + 1/log(5) + 0
IDCG = 3/log(2) + 1/log(3) + 1/log(4) + 0 + 0

Подходы к решению задачи ранжирования

- Point-wise — поточечный: предсказывается ранг объекта
- Pair-wise — попарный: моделируется функция, ранжирующая пары объектов
- List-wise — списочный: объект – упорядоченный набор; оптимизируются параметры ранжирующей списки функции для максимизации некоторой меры порядка

Point-wise

- Предположение: для обучающей выборки известны абсолютные значения ранга
- Сведем задачу ранжирования к задаче предсказания ранга (классификации или регрессии)
- Пример: Объект – (q,d); два класса: документ d релевантен запросу q или нет. Обучающую выборку должны готовить специально обученные асессоры.



Недостатки поточечного подхода

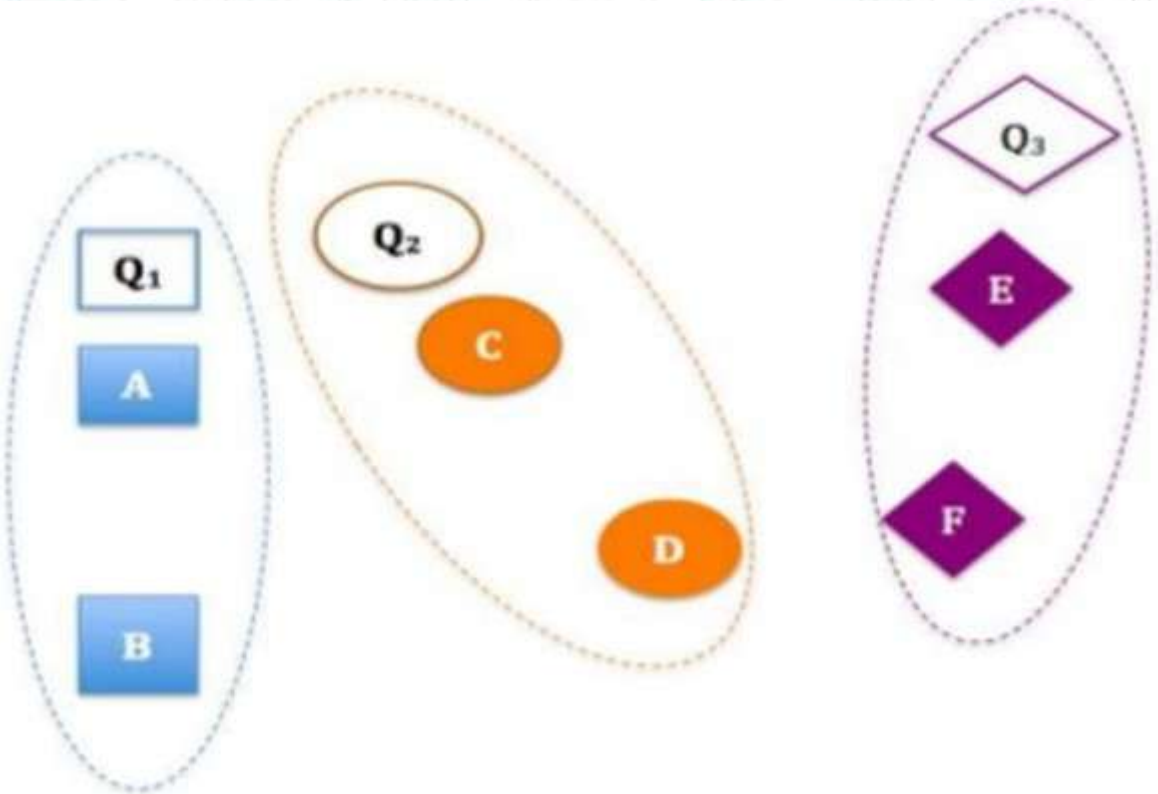
- Проблема: алгоритм рассматривает документы из разных запросов вместе, сравнивая между собой

  Если один человек предпочитает классику, а другой – рок. Зачем определять силу их предпочтения?

- Любой (правильный и неправильный) порядок объектов внутри одного списка с приближенно равными рангами штрафуется функционалом качества одинаково:
  ……..; (q1,d1); (q1,d2); (q1,d3);……
  Если пар очень много, то от порядка указанных мера почти не зависит т.к. они идут рядом в гигантском списке

Pair-wise

- Объект – пара ранжируемых элементов (x_1,x_2). Требуется предсказать порядок: $x_1>x_2$ или $x_1<x_2$.
- Обучающая выборка: множество известных отранжированных пар



Pair-wise

- Метод обучения учится на парах. Например, SVM:

$$Q(a) = \tfrac{1}{2}\|w\|^2 + C\sum_{i \prec j} \mathscr{L}\big(\underbrace{a(x_j) - a(x_i)}_{\text{Margin}(i,j)}\big) \quad \to \quad \min_a,$$

где $a(x) = \langle w, x\rangle$ — функция ранжирования,
$\mathscr{L}(M) = (1 - M)_+$ — функция потерь,
$M = \text{Margin}(i,j) = \langle w, x_j - x_i\rangle$ — отступ,

$$\begin{cases} \tfrac{1}{2}\|w\|^2 + C\sum_{i \prec j} \xi_{ij} \to \min\limits_{w,\xi}; \\ \langle w, x_j - x_i\rangle \geqslant 1 - \xi_{ij}, \quad i \prec j; \\ \xi_{ij} \geqslant 0, \quad i \prec j. \end{cases}$$

Pair-wise

- Пример 2: метод стохастического градиента для логистической регрессии (RankNet)

$$Q(a) = \sum_{i \prec j} \mathscr{L}\big(a(x_j) - a(x_i)\big) \quad \to \quad \min$$

$$a(x) = \langle w, x \rangle \qquad \mathscr{L}(M) = \log(1 + e^{-\sigma M})$$

На каждой итерации берем случайно группу и пару i<j:

$$w := w + \eta \cdot \frac{\sigma}{1 + \exp(\sigma \langle x_j - x_i, w \rangle)} \cdot (x_j - x_i)$$

Недостатки попарного подхода

- Оптимизируемый фунционал качества оценивает глобальный порядок, а не порядок для одной группы (list)
- Не учитываются зависимости между сравниваемыми парами в общей группе

Пример проявления недостатка

- В магазине 3 товара: a,b,c. Посетители сайта магазина ранжируют товары по убыванию предпочтений (обучающая выборка).
- Pair-wise подход посчитал вероятности: P(a>b) = 0.6; P(a>c) = 0.3; P(b>c) = 0.7
- Вычислим вероятности всех возможных порядков

P(>)	a	b	c
a		0.6	0.3
b	0.4		0.7
c	0.7	0.3	

Пример проявления недостатка

- P(a>b>c) = P(a>b)*P(a>c)*P(b>c) = 0.126
- P(a>c>b) = 0.3*0.6*0.3 = 0.054
- P(b>a>c) = 0.4*0.7*0.3 = 0.084
- P(b>c>a) = 0.7*0.4*0.7 = 0.196
- P(c>a>b) = 0.7*0.3*0.6 = 0.126
- P(c>b>a) = 0.3*0.7*0.4 = 0.084
- Все правильно?

P(>)	a	b	c
a		0.6	0.3
b	0.4		0.7
c	0.7	0.3	

Пример проявления недостатка

- P(a>b>c) = P(a>b)*P(a>c)*P(b>c) = 0.126
- P(a>c>b) = 0.3*0.6*0.3 = 0.054
- P(b>a>c) = 0.4*0.7*0.3 = 0.084
- P(b>c>a) = 0.7*0.4*0.7 = 0.196
- P(c>a>b) = 0.7*0.3*0.6 = 0.126
- P(c>b>a) = 0.3*0.7*0.4 = 0.084
- Сумма всех вероятностей = 0.67
- Парадокс Кондорсе'. Метод Шульце?
- P(a>b, b>c, c>a) = 0.294

P(>)	a	b	c
a		0.6	0.3
b	0.4		0.7
c	0.7	0.3	

Пример проявления недостатка

- P(a>b>c) = P(a>b)*P(a>c)*P(b>c) = 0.126
- P(a>c>b) = 0.3*0.6*0.3 = 0.054
- P(b>a>c) = 0.4*0.7*0.3 = 0.084
- P(b>c>a) = 0.7*0.4*0.7 = 0.196
- P(c>a>b) = 0.7*0.3*0.6 = 0.126
- P(c>b>a) = 0.3*0.7*0.4 = 0.084

P(>)	a	b	c
a		0.6	0.3
b	0.4		0.7
c	0.7	0.3	

- А на самом деле в обучающей выборке все пользователи сайта делились на три группы:
  30% голосовало за порядок a>b>c, 30% - за c>a>b,
  40% - за b>c>a
  Других вариантов пользователи не предлагали!

List-wise

- Объект – группа (list), внутри которой нужно произвести ранжирование
- Оптимизируемый функционал оценивает качество каждой группы
- Было в методе стохастического градиента для Pair-wise:

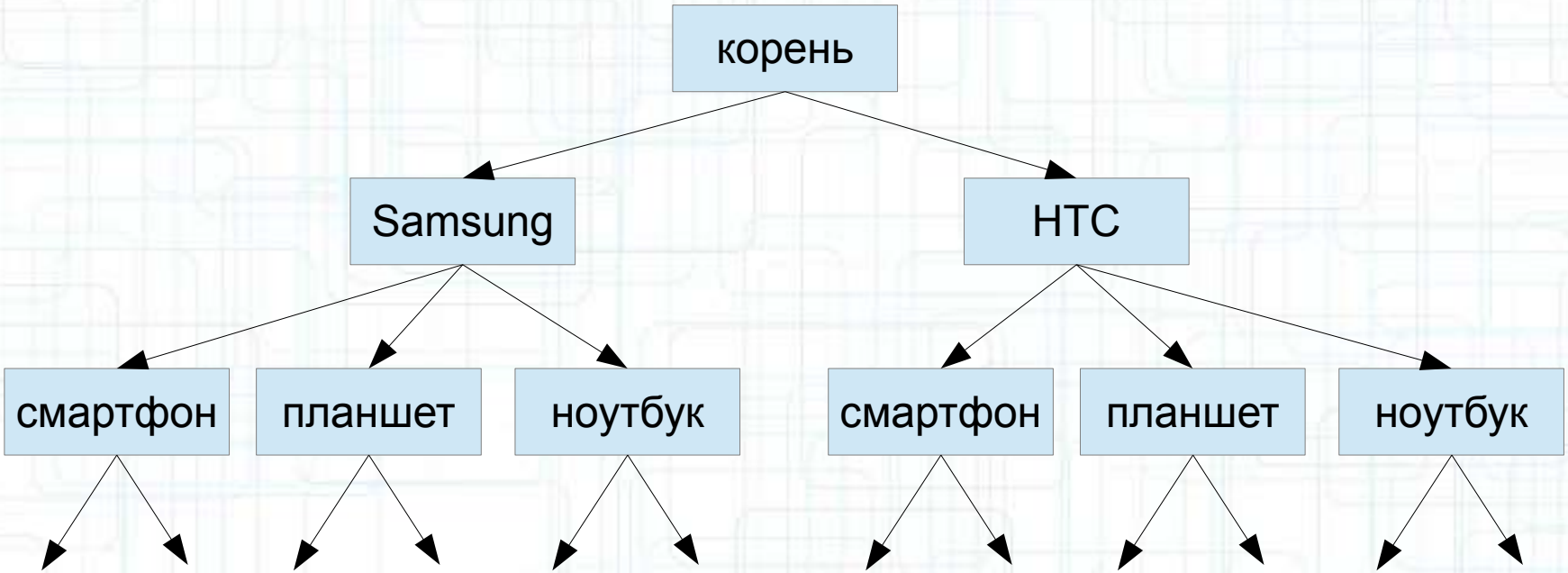
$$w := w + \eta \cdot \frac{\sigma}{1 + \exp(\sigma \langle x_j - x_i, w \rangle)} \cdot (x_j - x_i)$$

- Модифицируем:

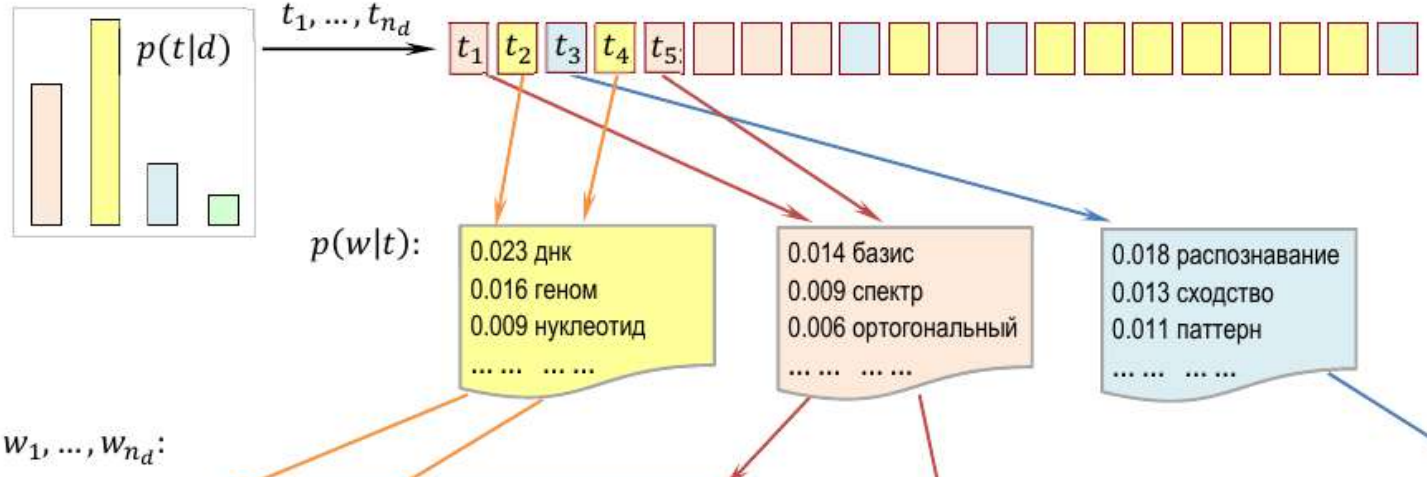
$$w := w + \eta \cdot \frac{\sigma}{1 + \exp(\sigma \langle x_j - x_i, w \rangle)} \cdot |\Delta NDCG_{ij}| \cdot (x_j - x_i)$$

Extreme Multilabel Classification

- Трактуем группу (list) в качестве класса
- Один объект может входить в несколько групп => многозначная классификация
- Иерархия классов:



Машинное обучение
Тематическое моделирование



Разработан спектрально-аналитический подход к выявлению размытых протяженных повторов в геномных последовательностях. Метод основан на разномасштабном оценивании сходства нуклеотидных последовательностей в пространстве коэффициентов разложения фрагментов кривых GC- и GA-содержания по классическим ортогональным базисам. Найдены условия оптимальной аппроксимации, обеспечивающие автоматическое распознавание повторов различных видов (прямых и инвертированных, а также тандемных) на спектральной матрице сходства. Метод одинаково хорошо работает на разных масштабах данных. Он позволяет выявлять следы сегментных дупликаций и мегасателлитные участки в геноме, районы синтении при сравнении пары геномов. Его можно использовать для детального изучения фрагментов хромосом (поиска размытых участков с умеренной длиной повторяющегося паттерна).

https://yandexdataschool.ru/edu-process/courses/machine-learning

Содержание лекции

- Постановка задачи
- Предыстория
- Латентный семантический анализ (LSA)
- Вероятностный LSA (PLSA)
- Латентное размещение Дирихле (LDA)
- Учет контекста

Предыстория

- Векторная модель документов:
  $d_j = (w_{1j}, w_{2j}, \ldots, w_{nj})$
- w_{ij} – вес i-того слова в j-том документе
- Методы взвешивания термов:
  - булевский вес (0,1)
  - Tf - term frequency (функция от количества вхождений слова в документ)
  - Tf-idf = TF*IDF
- Близость между документами (или запросом и документом) вычислялась про правилу косинуса:

$$\text{similarity} = \cos(\theta) = \frac{A \cdot B}{\|A\|\|B\|} = \frac{\sum_{i=1}^{n} A_i \times B_i}{\sqrt{\sum_{i=1}^{n} (A_i)^2} \times \sqrt{\sum_{i=1}^{n} (B_i)^2}}$$

TF-IDF

- Мера важности слов в контексте документа: TF-IDF = TF*IDF

$$\mathrm{TF} = \frac{n_i}{\sum_k n_k} \qquad \mathrm{IDF} = \log \frac{|D|}{|(d_i \supset t_i)|}$$

  - n_i – число вхождений i-того термина в документ
  - $|d_i \supset t_i|$ - число документов с термином t_i
  - $|D|$ - количество документов

Недостатки векторной модели

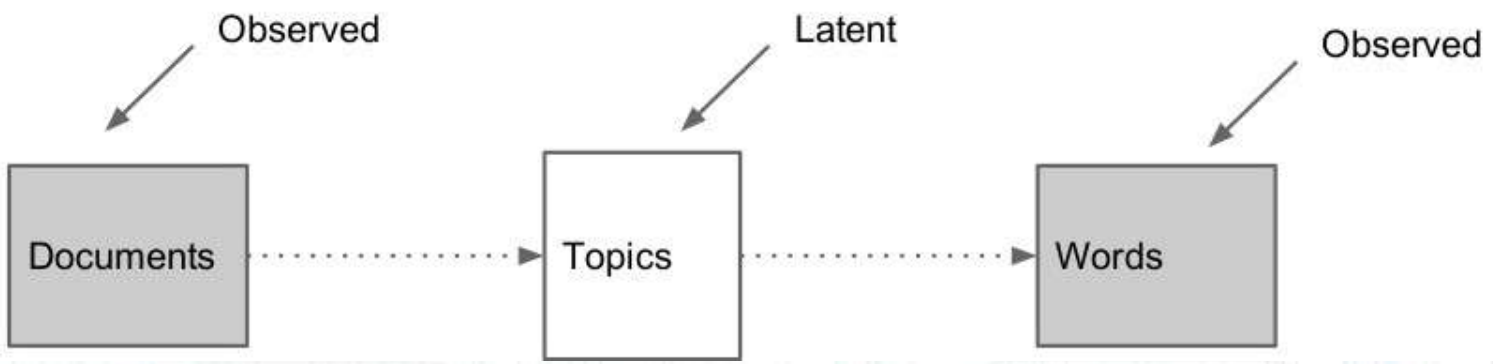
- Проблемы с большими документами (они дают приближенно равные маленькие скалярные произведения) – это “проклятие размерности”
- Поисковая система находит документы только со словами из запроса. Документы на ту же тему, но другими словами – не находятся

Тематическое моделирование

- Для анализа текстов удобно описывать их небольшим набором тем (математически: понизить размерность)
- Это позволит легко:
  - проводить категоризацию документов
  - аннотировать тексты
  - вычислять близость документов
- Сферы применения: информационный поиск, категоризация, поиск рецензентов/экспертов, рекомендательные системы, аннотирование изображений,...

Латентный семантический анализ

- Ключевая идея: любой текст является смесью небольшого количества скрытых (latent) составных элементов (тем)



- Пример: наша лекция сегодня состоит из линейной алгебры, теории вероятностей, моделирования
  Значит, она с большой вероятностью должна содержать слова:
  ЛА: вектор, скалярное произведение, ортогональный, SVD-разложение,...
  ТВ: вероятность, Байес, при условии, распределение,…
  М: модель, соответствие, проверка,...

Латентный семантический анализ

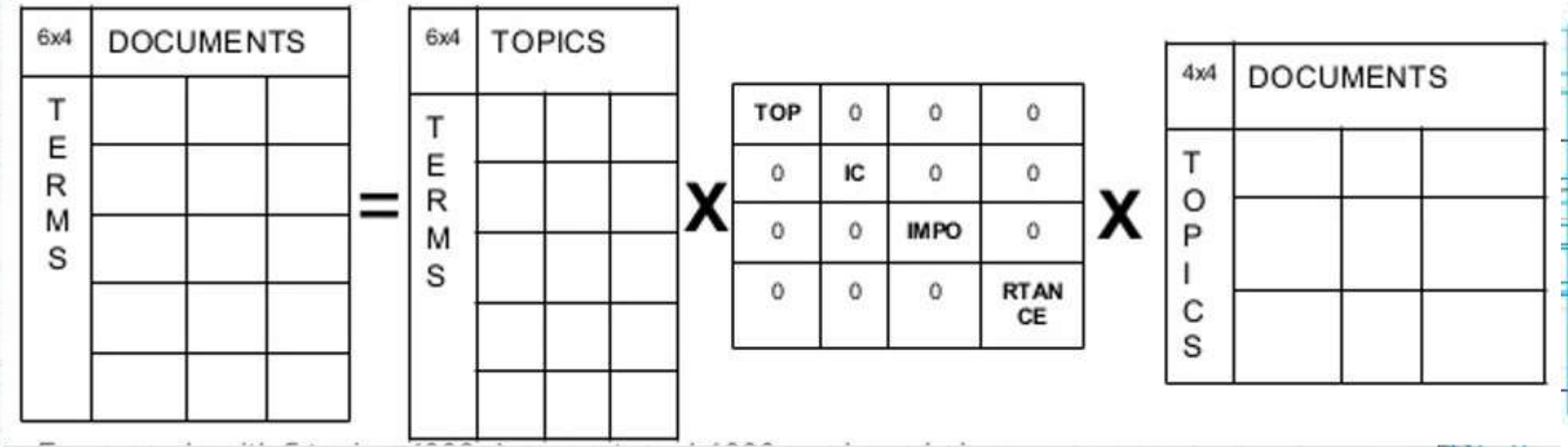
- [Deerwester и др. ‘90]:
- Считает частоты встречаемости слов в каждом документе
- Записывает их в term-document матрицу
- Понижает размерность по методу PCA: SVD-разложение + сокращение числа компонент
- Темы – небольшой набор ортогональных векторов, лучшим образом приближающий исходную линейную оболочку документов

Пример

$$\begin{array}{cc} & \mathbf{d}_j \\ & \downarrow \\ \mathbf{t}_i^T \rightarrow & \begin{bmatrix} x_{1,1} & \cdots & x_{1,n} \\ \vdots & \ddots & \vdots \\ x_{m,1} & \cdots & x_{m,n} \end{bmatrix} \end{array}$$

	D1	D2	D3	D4
linux	3	4	1	0
modem	4	3	0	1
the	3	4	4	3
clutch	0	1	4	3
steering	2	0	3	3
petrol	0	1	3	4

SVD-разложение



Числа в диагональной матрице имеют смысл “важностей” тем в нашей коллекции документов

Пример

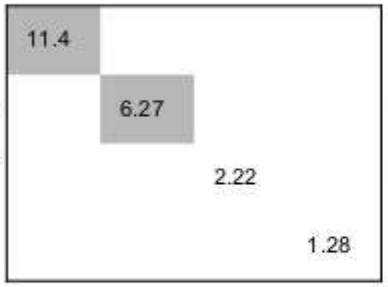
3	4	1	0
4	3	0	1
3	4	4	3
0	1	4	3
2	0	3	3
0	1	3	4

=

	To1	To2	To3	To4
Te1	-0.33	-0.53	0.37	-0.14
Te2	-0.32	-0.54	-0.49	0.35
Te3	-0.62	-0.10	0.26	-0.14
Te4	-0.38	0.42	0.30	-0.24
Te5	-0.36	0.25	-0.68	-0.47
Te6	-0.37	0.42	0.02	0.75

X

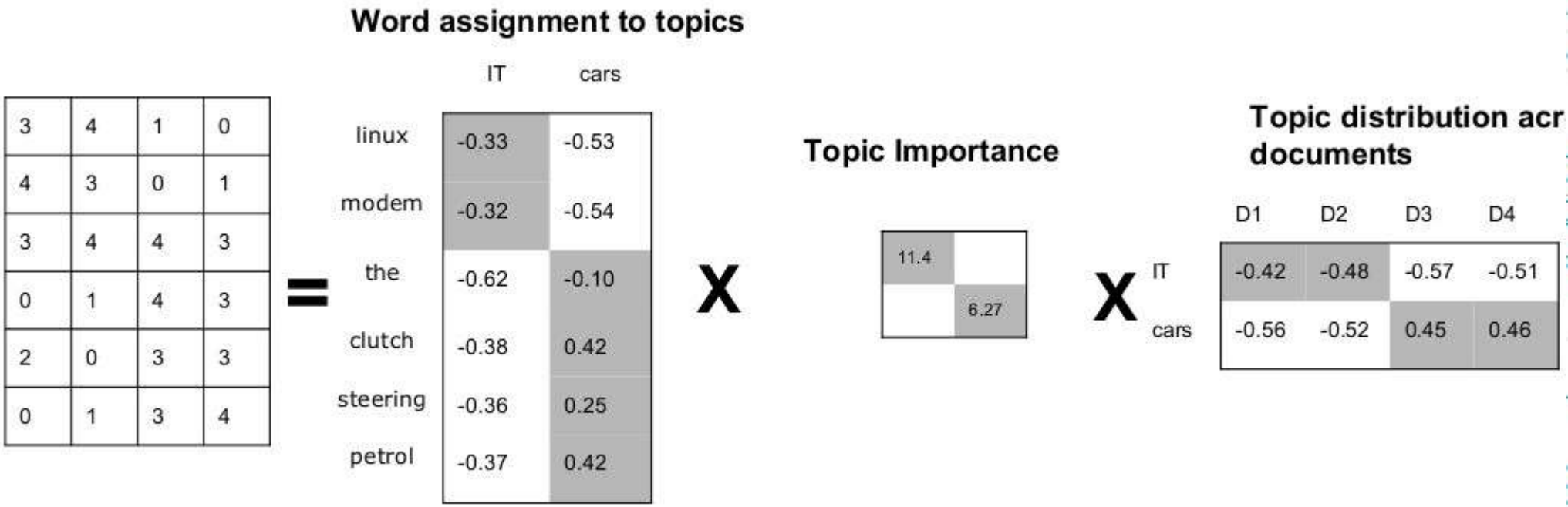
Topic Importance



X

	D1	D2	D3	D4
To1	-0.42	-0.48	-0.57	-0.51
To2	-0.56	-0.52	0.45	0.46
To3	-0.65	0.62	0.28	-0.35
To4	-0.30	0.34	-0.63	0.63

Оставили только главные компоненты



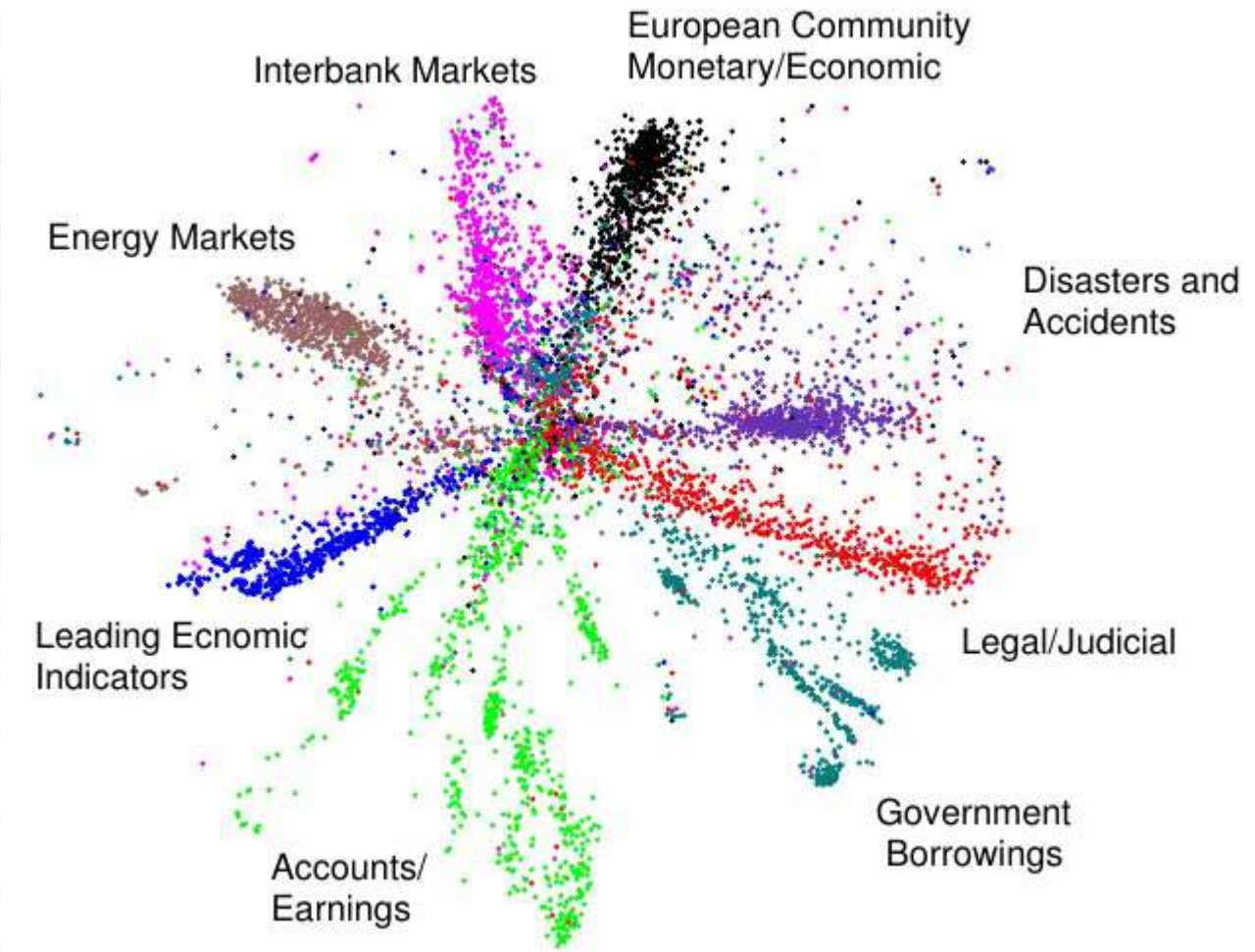
3	4	1	0
4	3	0	1
3	4	4	3
0	1	4	3
2	0	3	3
0	1	3	4

Пример работы
Матрица: слова-темы

Блоки – темы. Представлены слова с максимальными координатами в темах.

music band songs rock album jazz pop song singer night	book life novel story books man stories love children family	art museum show exhibition artist artists paintings painting century works	game knicks nets points team season play games night coach	show film television movie series says life man character know
theater play production show stage street broadway director musical directed	clinton bush campaign gore political republican dole presidential senator house	stock market percent fund investors funds companies stocks investment trading	restaurant sauce menu food dishes street dining dinner chicken served	budget tax governor county mayor billion taxes plan legislature fiscal

Пример работы 2
Визуализация документов

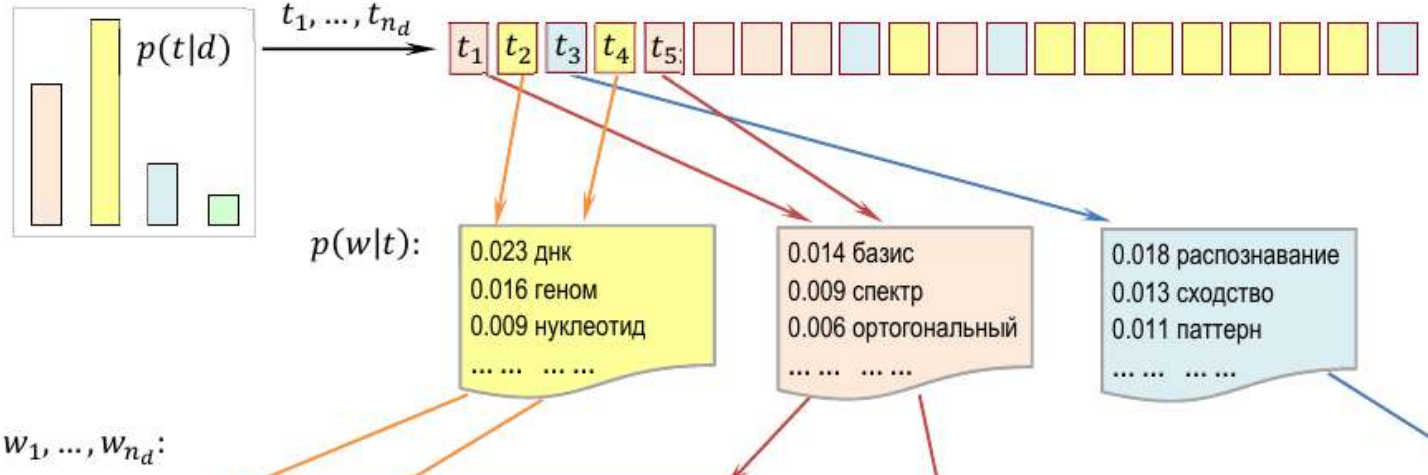


Вероятностный подход к LSA (PLSA)

- Вместо сокращения размерности документов предположим, что документы – случайны и порождены совместными вероятностными распределениями:
(слова, темы) и (темы, документы)
- Найдем параметры этого распределения
- Математически получается та же формула, что и в SVD. Но SVD находит темы оптимизируя евклидово расстояние, а PLSA - правдоподобие

Случайный процесс порождения документов в модели PLSA

$$p(w|d) = \sum_{t \in T} p(w|t)\, p(t|d)$$



Разработан спектрально-аналитический подход к выявлению размытых протяженных повторов в геномных последовательностях. Метод основан на разномасштабном оценивании сходства нуклеотидных последовательностей в пространстве коэффициентов разложения фрагментов кривых GC- и GA-содержания по классическим ортогональным базисам. Найдены условия оптимальной аппроксимации, обеспечивающие автоматическое распознавание повторов различных видов (прямых и инвертированных, а также тандемных) на спектральной матрице сходства. Метод одинаково хорошо работает на разных масштабах данных. Он позволяет выявлять следы сегментных дупликаций и мегасателлитные участки в геноме, районы синтении при сравнении пары геномов. Его можно использовать для детального изучения фрагментов хромосом (поиска размытых участков с умеренной длиной повторяющегося паттерна).

Восстановление плотности p(w|t) и p(t|d)

- Дано: n_{dw} – количество вхождений слова w в документ d, n_d – размер документа

$$\frac{n_{dw}}{n_d} \approx p(w|d)$$

- Найти: вероятности терминов в темах, вероятности тем в документах

$$\phi_{wt} = p(w|t)$$
$$\theta_{td} = p(t|d)$$

так, чтобы выполнялось равенство:

$$p(w|d) = \sum_{t \in T} \phi_{wt} \theta_{td}$$

Принцип максимума правдоподобия

- Правдоподобие коллекции документов:

$$\prod_{i=1}^{n} p(d_i, w_i) = \prod_{d \in D} \prod_{w \in d} p(d, w)^{n_{dw}}$$

- Максимизация логарифма правдоподобия

$$\sum_{d \in D} \sum_{w \in d} n_{dw} \ln p(w|d) p(d) \to \max_{\Phi, \Theta}$$

эквивалентна максимизации функционала:

$$\mathscr{L}(\Phi, \Theta) = \sum_{d \in D} \sum_{w \in d} n_{dw} \ln \sum_{t} \phi_{wt} \theta_{td} \to \max_{\Phi, \Theta}$$

EM-алгоритм

- E-шаг 1:
  Оцениваем число слов в документе d, порожденных темой t. По формуле Байеса:

$$p(t|d,w) = \frac{p(w,t|d)}{p(w|d)} = \frac{p(w|t)p(t|d)}{p(w|d)} = \frac{\phi_{wt}\theta_{td}}{\sum_s \phi_{ws}\theta_{sd}}$$

  отсюда:

$$n_{td} = \sum_w n_{wd} \frac{\phi_{wt}\theta_{td}}{\sum_s \phi_{ws}\theta_{sd}}$$

- M-шаг 1:
  оценка вероятности темы в документе

$$\theta_{td} = p(t|d) = \frac{n_{td}}{n_d}$$

EM-алгоритм

- E-шаг 2:
  Оцениваем количество вхождений слова w в тему t

$$n_{wt} = \sum_d n_{wd} \frac{\phi_{wt}\theta_{td}}{\sum_s \phi_{ws}\theta_{sd}}$$

- M-шаг 2:
  оценка вероятности вхождения слова в тему

$$\phi_{wt} = p(w|t) = \frac{n_{wt}}{n_t}$$

Неединственность решения

- Если задача разрешима, то решений бесконечно много:

$$\underset{W\times D}{\left(\frac{n_{dw}}{n_d}\right)} \approx \underset{W\times T}{\Phi} \cdot \underset{T\times D}{\Theta} = (\Phi S)(S^{-1}\Theta) = \underset{W\times T}{\Phi'} \cdot \underset{T\times D}{\Theta'}$$

S – произвольная невырожденная матрица

Недостатки PLSA

- Переобучение
- Некоторые документы имеют пару выраженных тем, а для многих – почти все темы по чуть-чуть присутствуют
- Если уменьшать число тем – получится плохая модель, если увеличивать – много документов с тематической неопределенностью
- Как сделать тем много и решить задачу с условием: в каждом документе не более 3-5 тем?
  Именно для этого служит LDA.

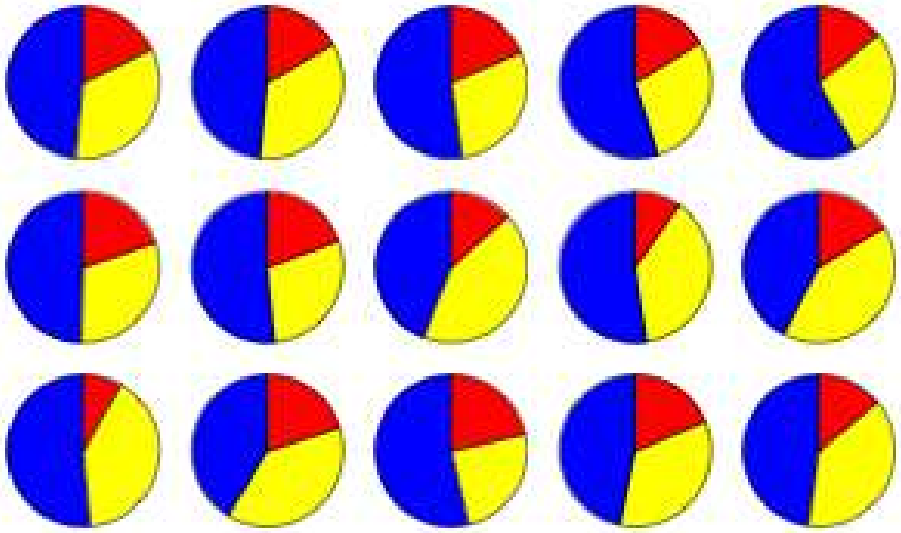
Распределение Дирихле

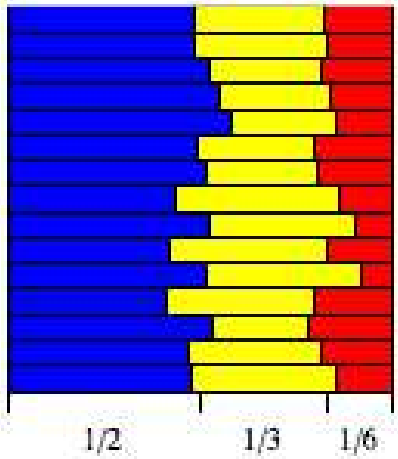
- Пусть отрезок длины 1 разрезан на K частей. Рассмотрим эксперимент с фиксированным исходом: случайно (равномерно) бросили несколько точек на отрезок и в каждой части оказалось ровно α_i-1 штук. Какими могут быть длины частей?
- Ответ: они распределены по закону Дирихле!
- Вероятность того, что вероятность каждого из K взаимоисключающих событий равна x_i при условии, что каждое событие наблюдалось α_i-1 раз

$$f(x_1,\ldots,x_K;\alpha_1,\ldots,\alpha_K) = \frac{1}{\mathrm{B}(\alpha)}\prod_{i=1}^{K} x_i^{\alpha_i-1}$$

$$\sum_{i=1}^{K} x_i = 1$$

Наглядная трактовка





Другая трактовка: урны и шары

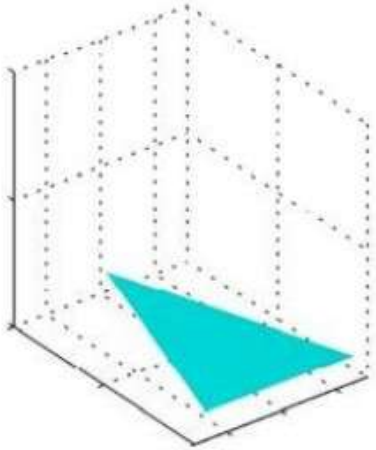
- Рассмотрим урну с шарами К различных цветов. Изначально в ней α_1 шаров цвета 1, α_2 – цвета 2,…
- Возьмем случайно из урны шар и положим его назад вместе с шаром того же цвета
- Если повторять это бесконечно много раз, то пропорции цветов в урне будут подчинены распределению Дирихле $\mathrm{Dir}(\alpha_1,\ldots,\alpha_K)$

Распределение Дирихле

- Распределение тем по документам и слов по темам можно моделировать распределением Дирихле!

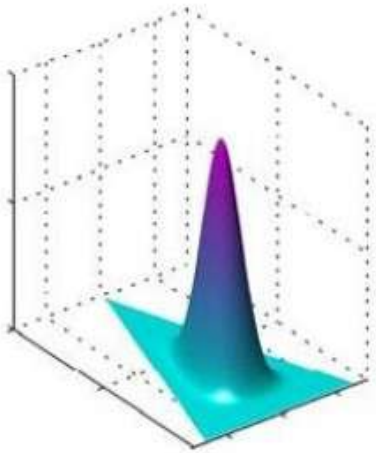
N=3:

Params = [1, 1, 1]



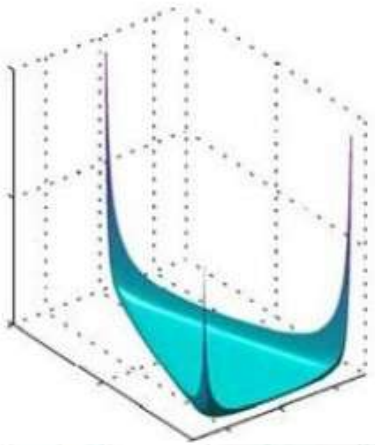
Bigger than 1

Params = [10, 10, 10]



Less than 1

Params = [.1, .1, .1]



Латентное размещение Дирихле

- Upgade PLSA:
  приближенно оценим (или зададим) среднее число тем в документе K и среднее число ключевых слов в теме V
- Смоделируем распределение тем по документам и слов по темам размещениями Дирихле с параметрами:
  K-мерный вектор α (чем меньше α, тем меньше выраженных тем в документе)
  V-мерный вектор β (чем меньше β, тем меньше слов, характеризующих тему)
- Обычно все координаты векторов α и β берут одинаковыми

Случайный процесс порождения документов в модели LDA

- Дано: количество тем K в документе и слов в теме V, параметры α и β
- Для каждого документа генерируем вероятности тем из распределения Дирихле с параметром α
- Для каждой темы генерируем вероятности слов из распределения Дирихле с параметром β
- Для каждой позиции в документе
  - Выбираем случайно тему согласно сгенерированным вероятностям
  - Вибираем случайно слово, согласно вероятностям слов в теме

Принцип максимума апостериорной вероятности

- Принцип максимума правдоподобия модели f(x|θ) для случайной величины x:

$$\hat{\theta}_{\mathrm{ML}}(x) = \arg\max_{\theta} f(x|\theta)$$

- Если параметр θ – случайная величина с известным априорным распределением g, то по формуле Байеса можно вычислить апостериорное распределение θ.
- Принцип максимума апостериорной вероятности:

$$\hat{\theta}_{\mathrm{MAP}}(x) = \arg\max_{\theta} \frac{f(x|\theta)\, g(\theta)}{\int_{\Theta} f(x|\theta')\, g(\theta')\, d\theta'} = \arg\max_{\theta} f(x|\theta)\, g(\theta)$$

Принцип максимума апостериорной вероятности

$$\ln \prod_{d \in D} \prod_{w \in d} p(d,w)^{n_{dw}} \prod_{t \in T} \mathrm{Dir}(\phi_t | \beta) \prod_{d \in D} \mathrm{Dir}(\theta_d | \alpha) \to \max_{\Phi, \Theta}$$

$$\sum_{d \in D} \sum_{w \in d} n_{dw} \ln \sum_{t \in T} \phi_{wt} \theta_{td} +$$

$$+ \sum_{t \in T} \sum_{w \in W} \ln \phi_{wt}^{\beta_w - 1} + \sum_{d \in D} \sum_{t \in T} \ln \theta_{td}^{\alpha_t - 1} \to \max_{\Phi, \Theta}$$

Регуляризованный EM-алгоритм

$$\underbrace{\sum_{d,w} n_{dw} \ln \sum_t \phi_{wt}\theta_{td}}_{\ln \text{ правдоподобия } \mathscr{L}(\Phi,\Theta)} + \underbrace{\sum_{t,w} \tilde{\beta}_w \ln \phi_{wt} + \sum_{d,t} \tilde{\alpha}_t \ln \theta_{td}}_{\text{критерий регуляризации } R(\Phi,\Theta)} \to \max_{\Phi,\Theta}$$

Если коэффициенты регуляризации >0, тогда чем больше логарифмы тем лучше. А когда мы берем α и β < 1, то получается чем больше нулевых вероятностей, тем лучше!

- В PLSA:

$$\phi_{wt} = \frac{n_{wt}}{n_t} \qquad \theta_{td} = \frac{n_{td}}{n_d}$$

- В LDA:

$$\phi_{wt} = \frac{n_{wt} + \beta_w}{n_t + \beta_0} \qquad \theta_{td} = \frac{n_{td} + \alpha_t}{n_d + \alpha_0}$$

Контекстные модели

- BagOfWords не учитывает порядок слов => машина не может полностью понять смысл предложений.
- Словосочетания сильно увеличивают словарь и требуют очень большой выборки для обучения
- Разложение документов и слов по темам не позволяет достаточно хорошо понять текст, чтобы, например, перевести его на другой язык
- Если в представлении слова закодировать не темы, а его контекст – машина будет лучше его понимать

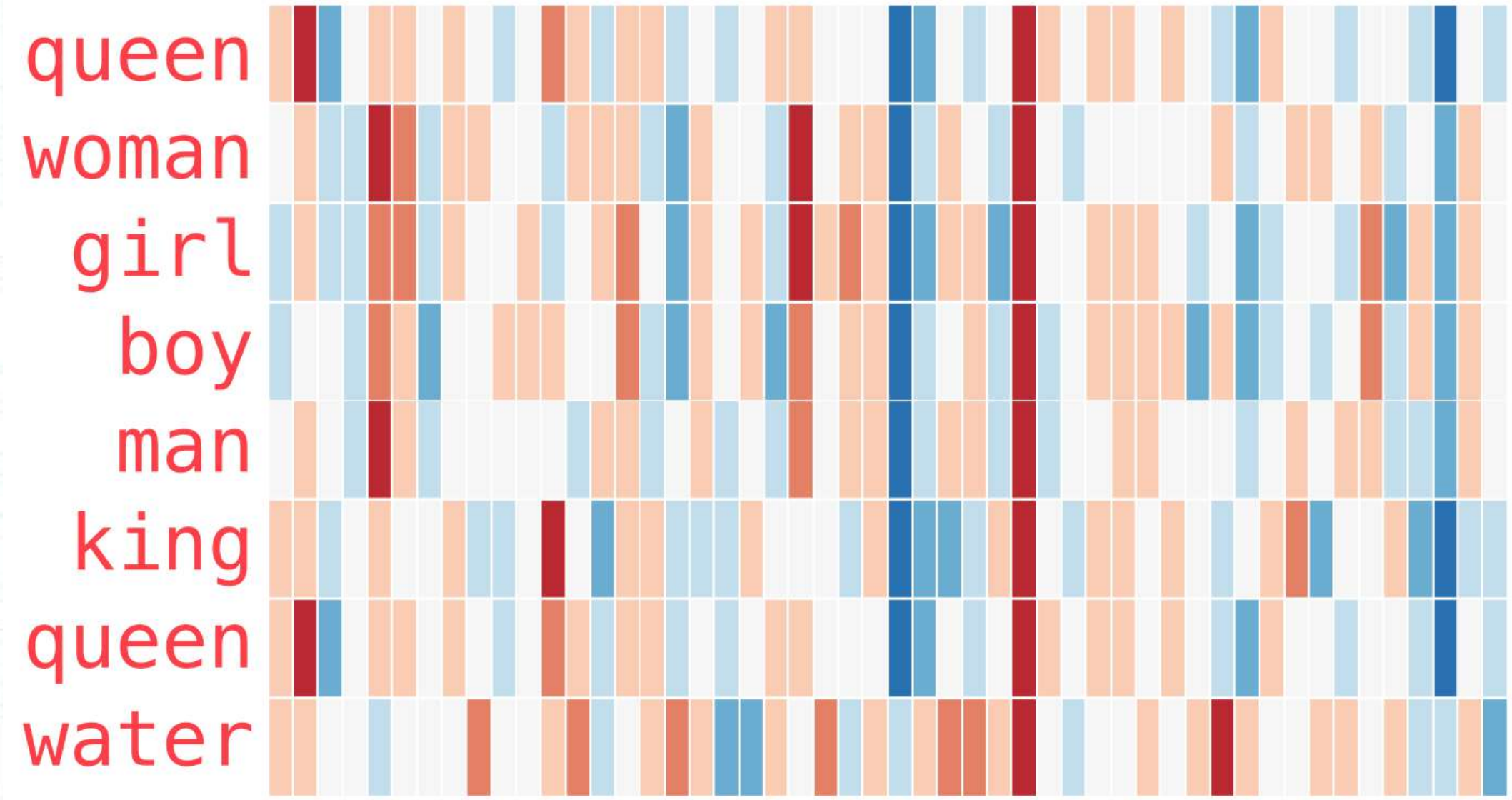
Word2vec

“Я увлекаюсь NLP и я люблю собак”

Составим матрицу встречаемости пар слов

	Я	увлекаюсь	NLP	и	люблю	собак
Я	0	1	0	1	1	0
увлекаюсь	1	0	1	0	0	0
NLP	0	1	0	1	0	0
и	1	0	1	0	0	0
люблю	1	0	0	0	0	1
собак	0	0	0	0	1	0

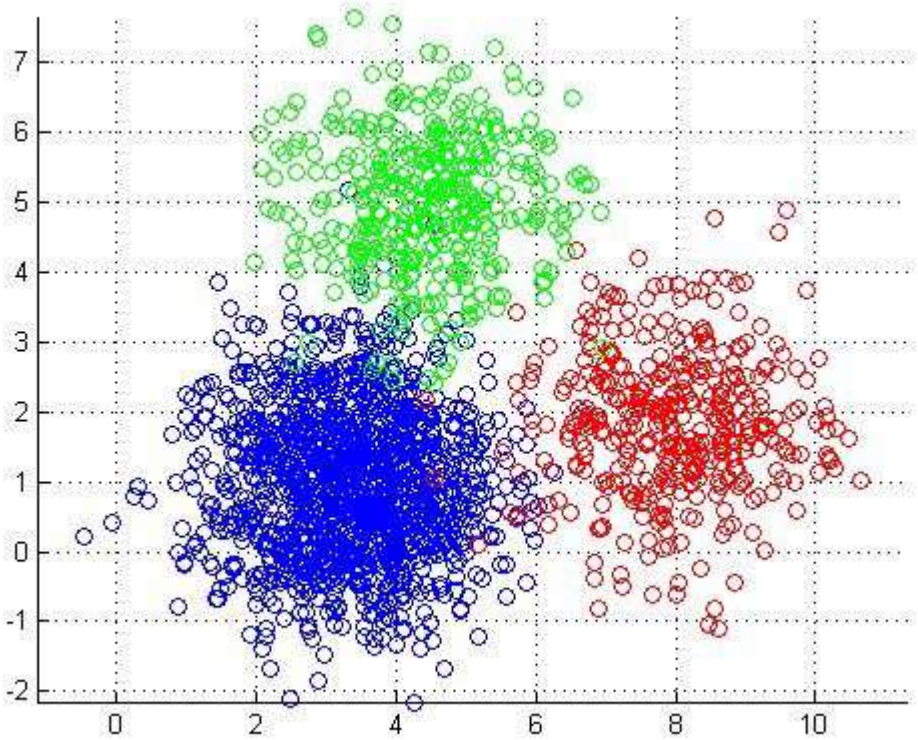
Word2vec – понижение размерности для матрицы встречаемости



Word2vec

king − man + woman ~= queen

Машинное обучение
Кластеризация



https://yandexdataschool.ru/edu-process/courses/machine-learning

Содержание лекции

- Постановка задачи
- EM-алгоритм
- Метод k-средних
- DBSCAN

Постановка задачи

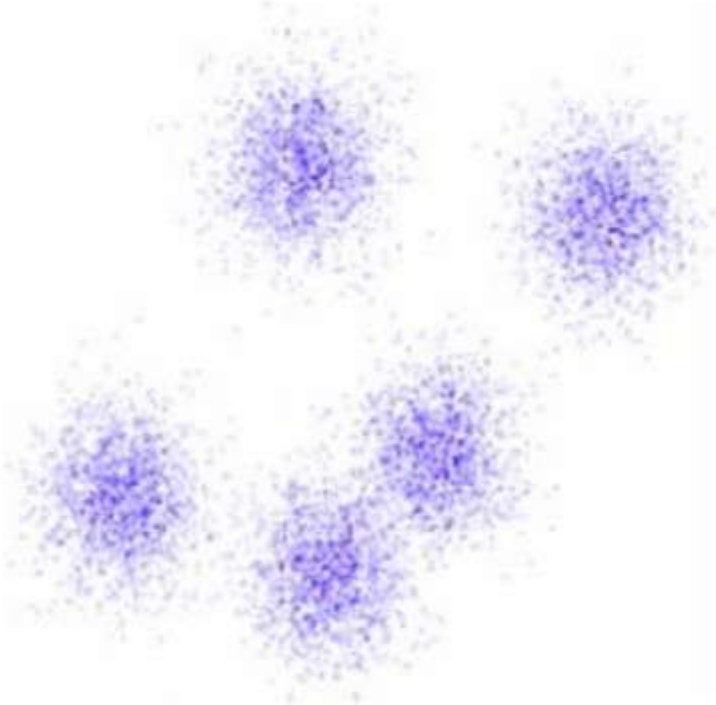
- Дано:
  - пространство объектов X
  - обучающая выборка X^ℓ
  - метрика между объектами
- Найти:
  - множество кластеров Y
  - алгоритм кластеризации a : X→Y
- Каждый кластер должен состоять из близких объектов
- Объекты разных кластеров должны быть существенно различны

Классификация и кластеризация

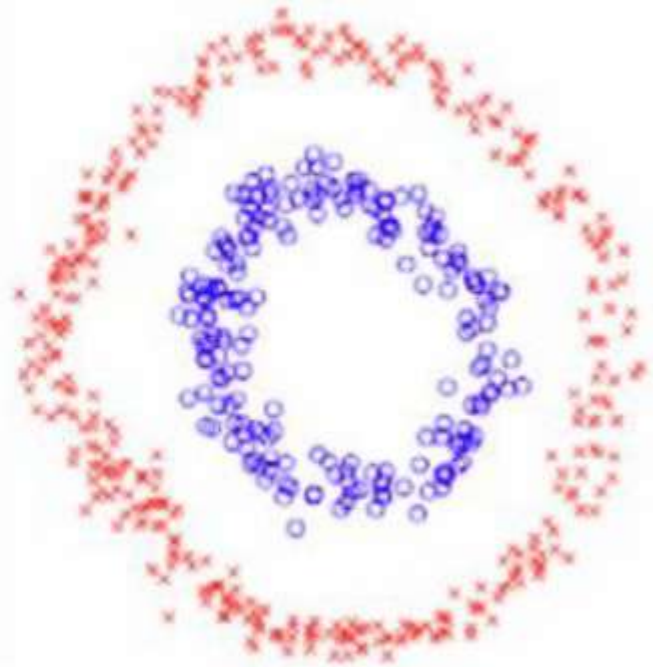
Классификация	Кластеризация
• Известное количество классов	• Неизвестно количество классов
• Классы известны для объектов обучающей выборки	• Нет данных о классах в обучающей выборке
• Используется для классификации объектов “в будущем”	• Используется для исследования множества объектов
• Классификация – это обучение с учителем	• Кластеризация – это обучение без учителя

Близость или связанность?

- Compactness, e.g., k-means, mixture models
- Connectivity, e.g., spectral clustering

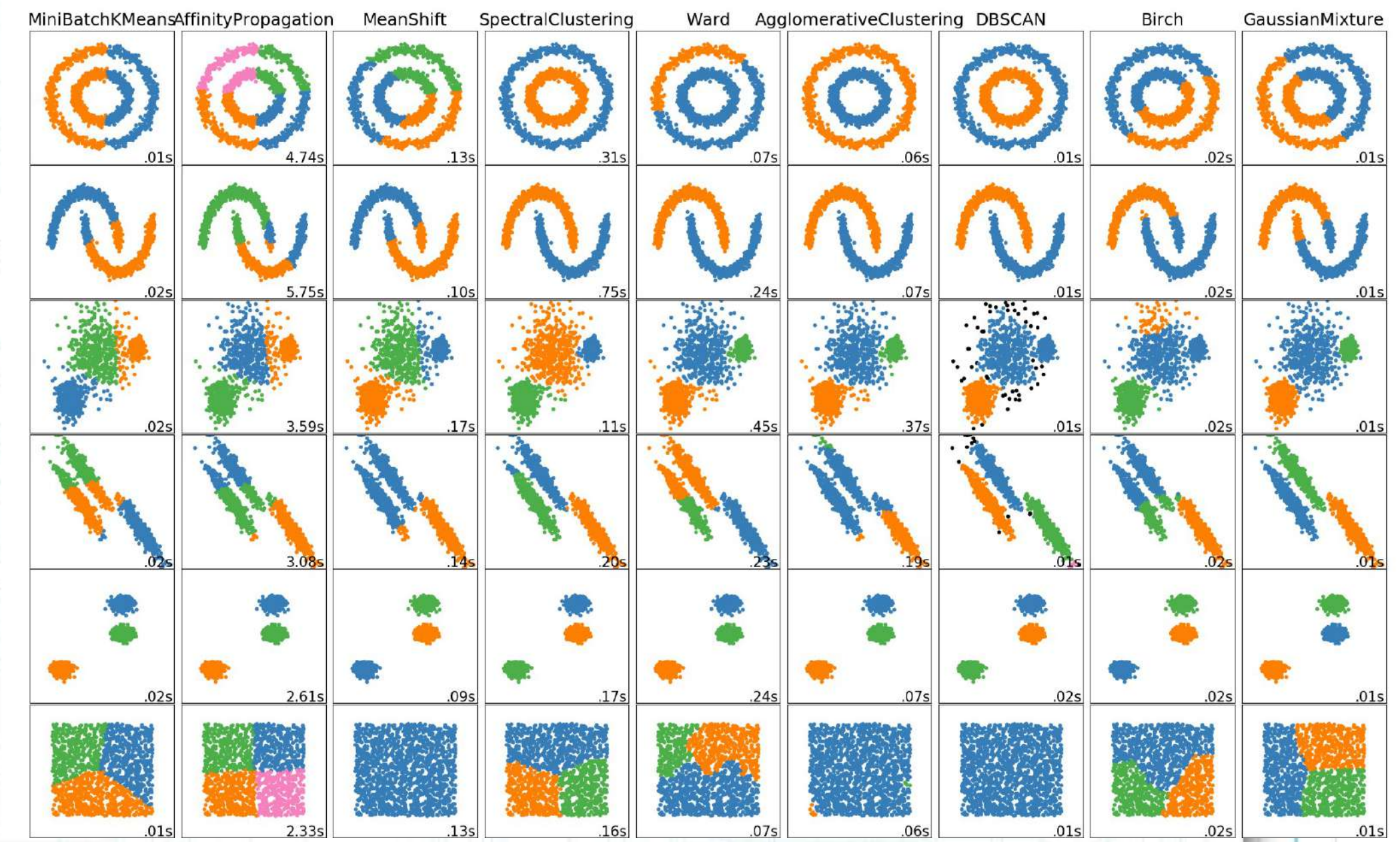


Compactness

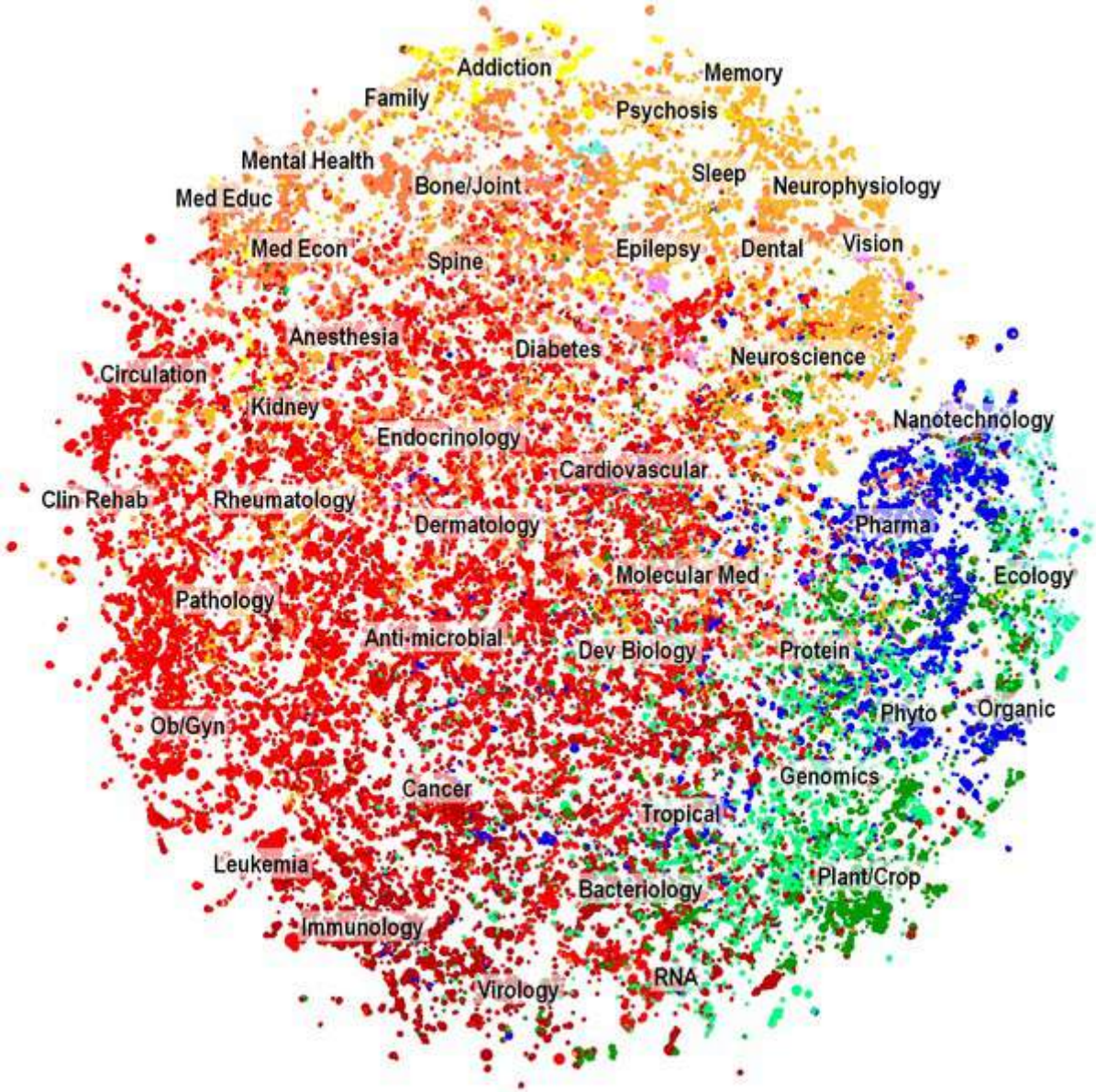


Connectivity

Пример: результаты работы алгоритмов кластеризации



Пример: кластеризация статей по медицине



EM-кластеризация

Гипотеза: выборка X^ℓ порождена смесью гауссовских случайных распределений

$$p(x) = \sum_{y \in Y} w_y p_y(x), \qquad \sum_{y \in Y} w_y = 1,$$

$$p_y(x) = (2\pi)^{-\frac{n}{2}} (\sigma_{y1} \cdots \sigma_{yn})^{-1} \exp\left(-\tfrac{1}{2}\rho_y^2(x, \mu_y)\right)$$

$\mu_y = (\mu_{y1}, \ldots, \mu_{yn})$ — центр кластера y;
$\Sigma_y = \operatorname{diag}(\sigma_{y1}^2, \ldots, \sigma_{yn}^2)$ — диагональная матрица ковариаций;

$$\rho_y^2(x, x') = \sum_{j=1}^{n} \sigma_{yj}^{-2} \, |f_j(x) - f_j(x')|^2.$$

EM-кластеризация

1: начальное приближение w_y, μ_y, Σ_y для всех $y \in Y$;

2: **повторять**

3: E-шаг (expectation):

$$g_{iy} := P(y|x_i) \equiv \frac{w_y p_y(x_i)}{\sum_{z \in Y} w_z p_z(x_i)}, \quad y \in Y, \quad i = 1, \ldots, \ell;$$

4: M-шаг (maximization):

$$w_y := \frac{1}{\ell} \sum_{i=1}^{\ell} g_{iy}, \quad y \in Y;$$

$$\mu_{yj} := \frac{1}{\ell w_y} \sum_{i=1}^{\ell} g_{iy} f_j(x_i), \quad y \in Y, \quad j = 1, \ldots, n;$$

$$\sigma^2_{yj} := \frac{1}{\ell w_y} \sum_{i=1}^{\ell} g_{iy} (f_j(x_i) - \mu_{yj})^2, \quad y \in Y, \quad j = 1, \ldots, n;$$

5: $y_i := \arg\max_{y \in Y} g_{iy}, \quad i = 1, \ldots, \ell;$

6: **пока** y_i не перестанут изменяться;

Метод k-средних

1: начальное приближение центров μ_y, $y \in Y$;

2: **повторять**

3: аналог E-шага:

отнести каждый x_i к ближайшему центру:

$y_i := \arg\min_{y \in Y} \rho(x_i, \mu_y)$, $i = 1, \ldots, \ell$;

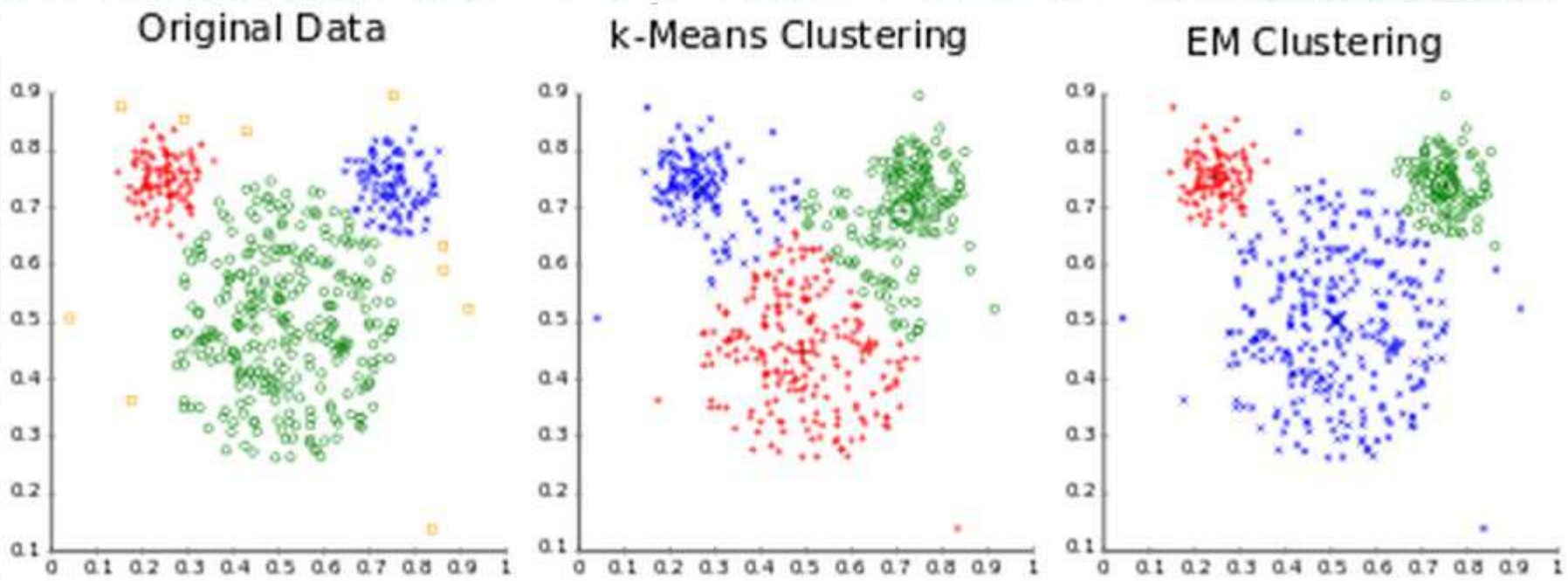
4: аналог M-шага:

вычислить новые положения центров:

$$\mu_{yj} := \frac{\sum_{i=1}^{\ell} [y_i = y] f_j(x_i)}{\sum_{i=1}^{\ell} [y_i = y]}, \quad y \in Y, \quad j = 1, \ldots, n;$$

5: **пока** y_i не перестанут изменяться;

Сравнение k-средних и EM-кластеризации



DBSCAN

- Density-Based Spatial Clustering of Applications with Noise – самый популярный алгоритм кластеризации
- Ключевые понятия:
  - Внутренняя точка – имеет более MinPts соседей (r<Eps)
  - Граничная точка – имеет меньше соседей, но является соседней к какой-либо внутренней точке
  - Остальные точки - шумовые
  - Достижимость по плотности: точка q достижима из внутренней точки p, если существует последовательность Eps-соседних внутренних точек от p к q

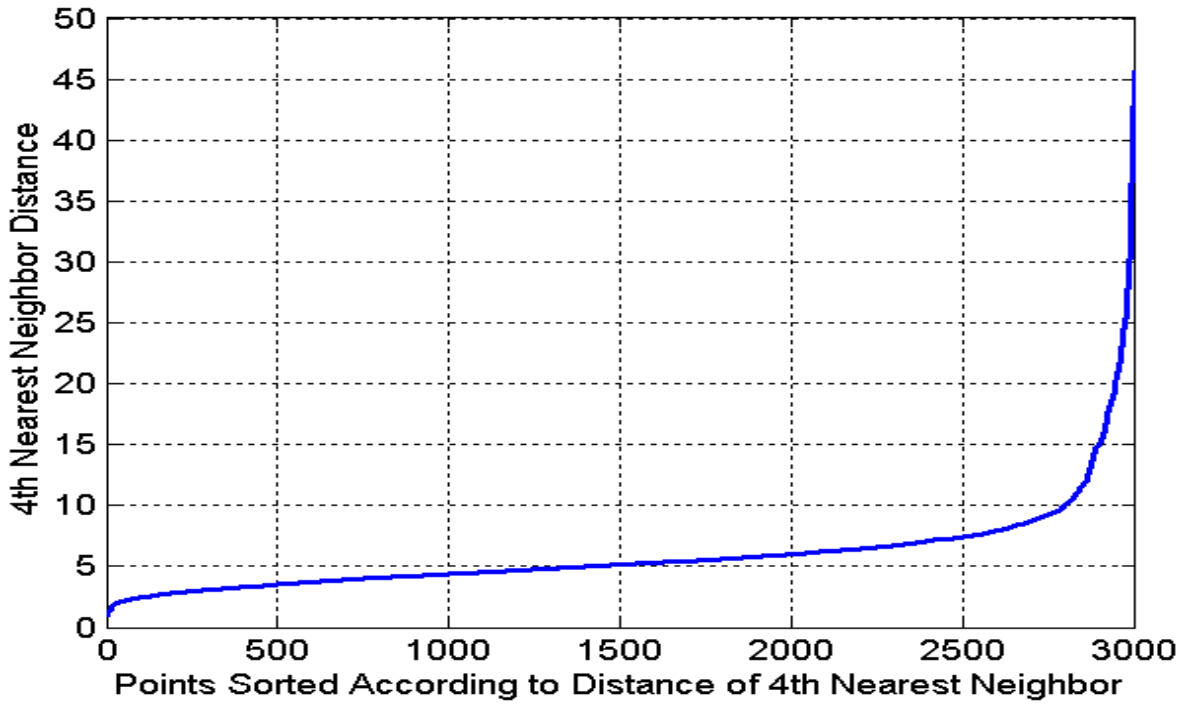
Алгоритм DBSCAN

- Выбрать точку p
- Если p-внутренняя, то
  - Найти все достижимые по плотности точки из p
  - Сформировать кластер
- Иначе – перейти к следующей точке

- Результат не зависит от порядка просмотра точек

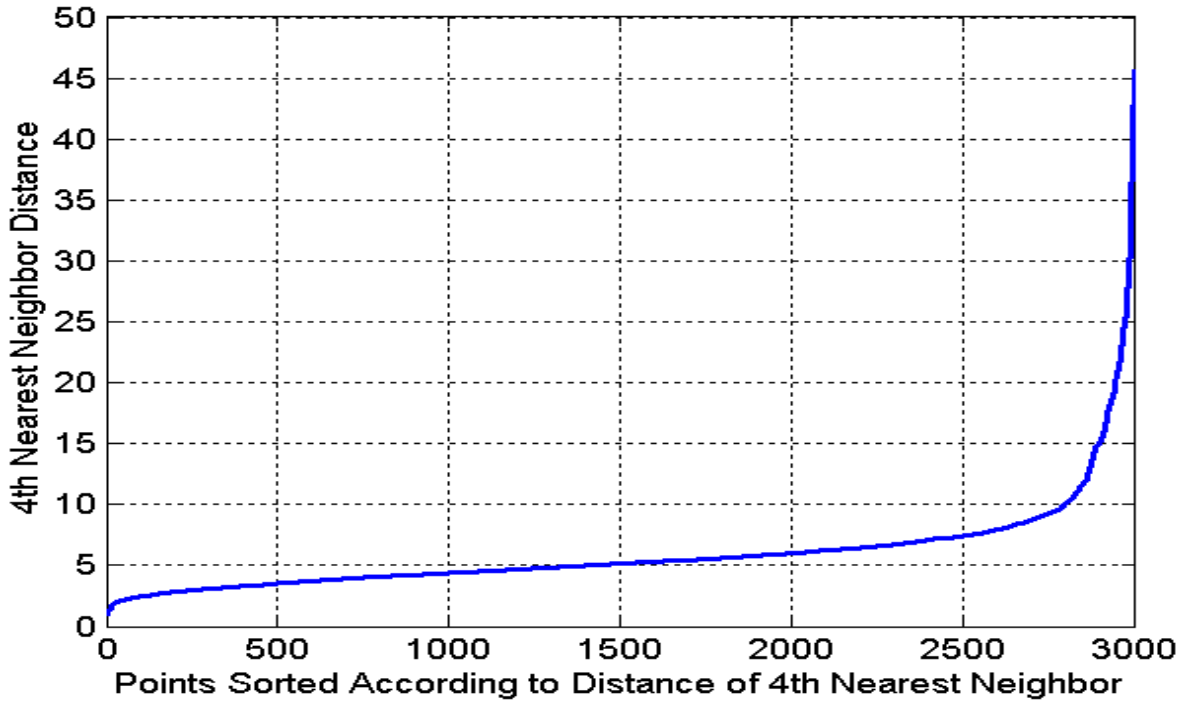
DBSCAN: выбор Eps и MinPts

- Ключевая идея: для всех точек одного кластера их k-тый сосед (k<размера кластера) находится на приблизительно одном и том же расстоянии
- Соседи шумовых точек – далеко
- График отсортированных расстояний:



DBSCAN: выбор Eps и MinPts

- Искомое Eps - начало крутого подъема на графике расстояний до соседа с фиксированным номером
- MinPts – номер соседа



Применение кластеризации в Feature engineering

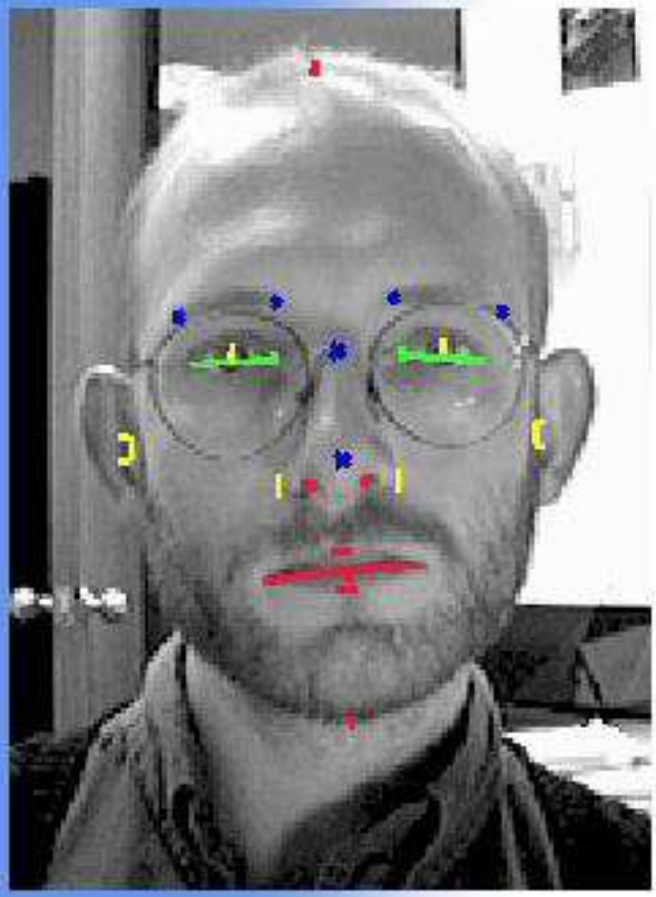
- Создание информативного признака (номер кластера) по заданному набору других признаков
- Сокращение размерности: большой набор признаков сводим к одному номеру кластера

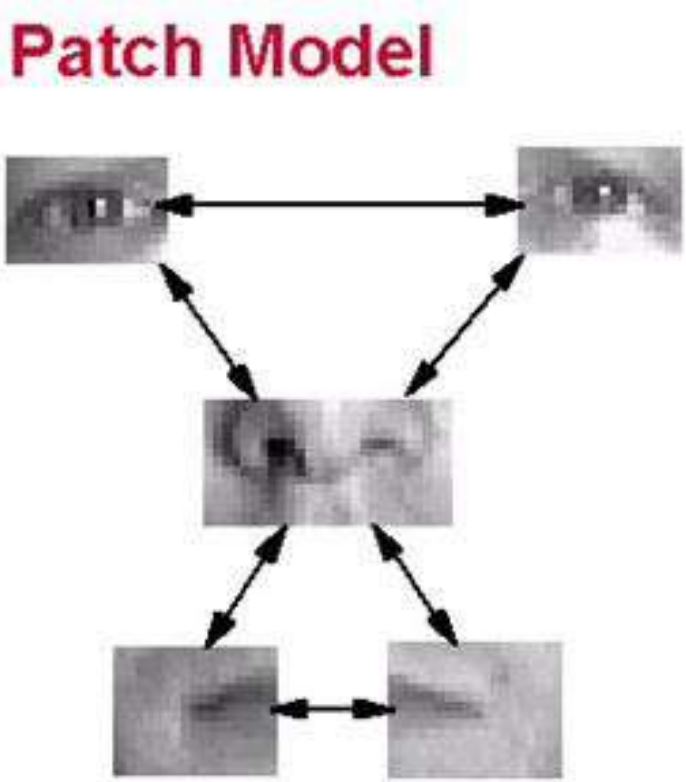
Машинное обучение
Компьютерное зрение

Задачи компьютерного зрения

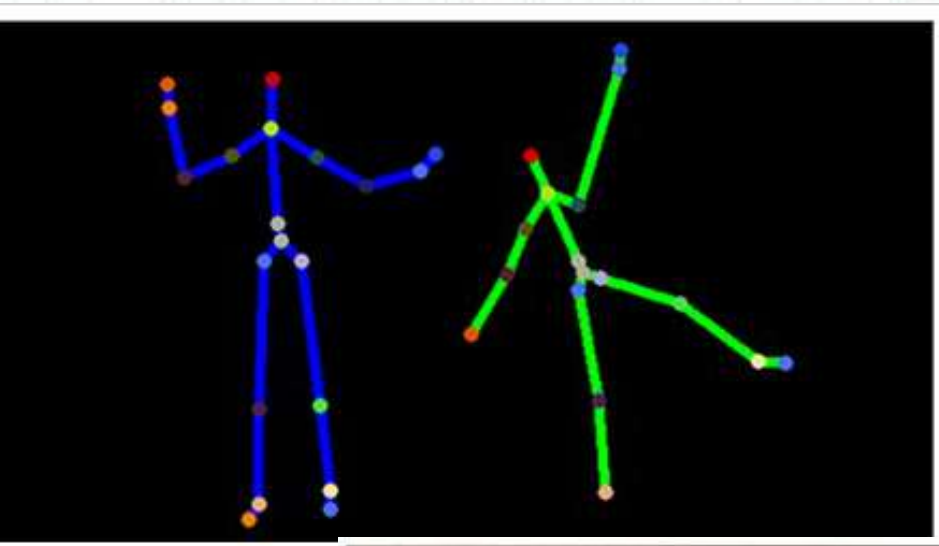
- **Классификация:**
  - **распознавание лиц, объектов, жестов**
  - **распознавание рукописного текста**
  - **поиск по изображению**
- **Регрессия:**
  - **определение положения объекта на фото**
  - **определение ориентации объекта в пространстве**
  - **стерео-реконструкция**
  - **восстановление зашумленных изображений**
- **Выбросы:**
  - **выявление аномалий**
  - **реферирование видео**
- **Кластеризация:**
  - **сегментация изображений**
  - **поиск дубликатов (фотоподделки)**

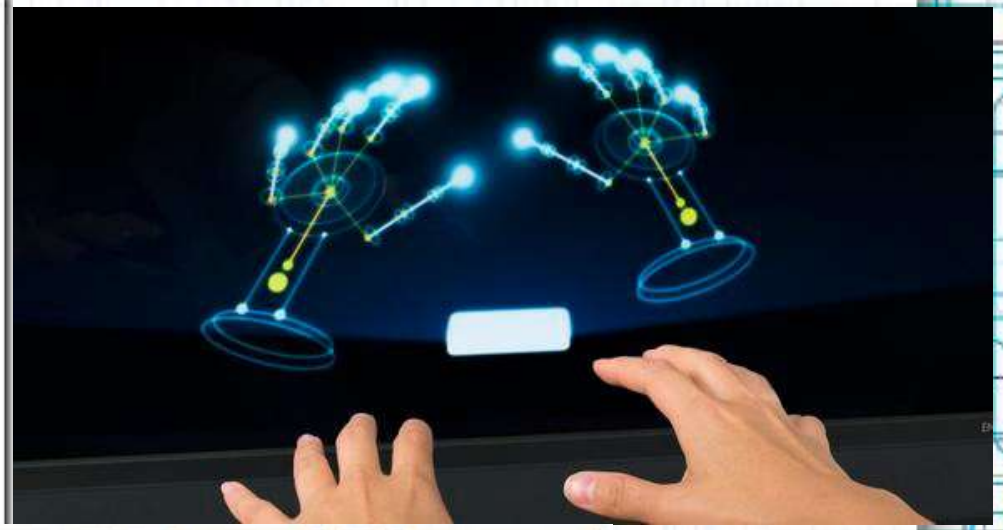
Распознавание лиц



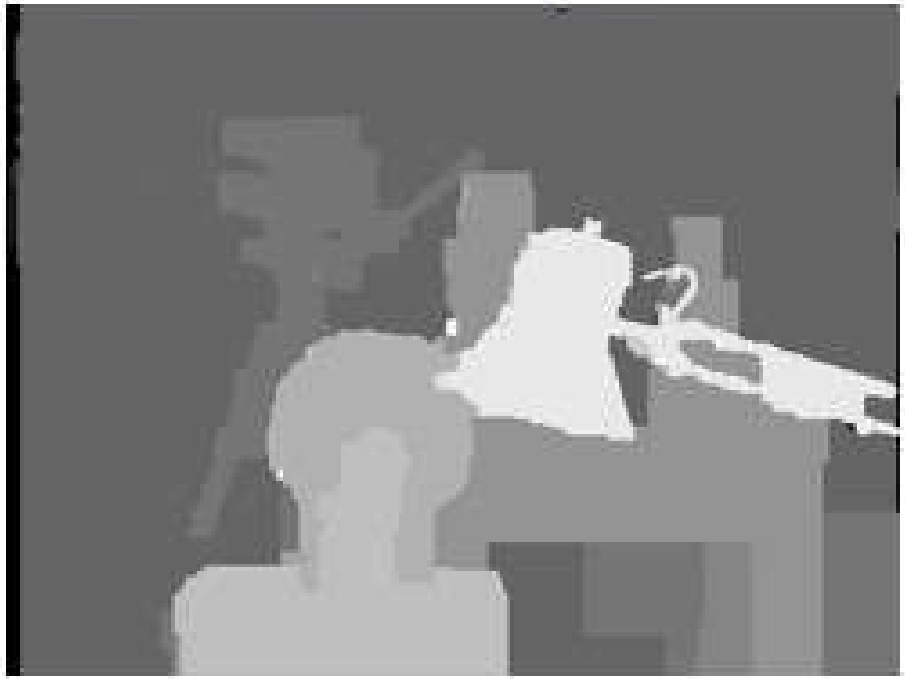


Распознавание жестов

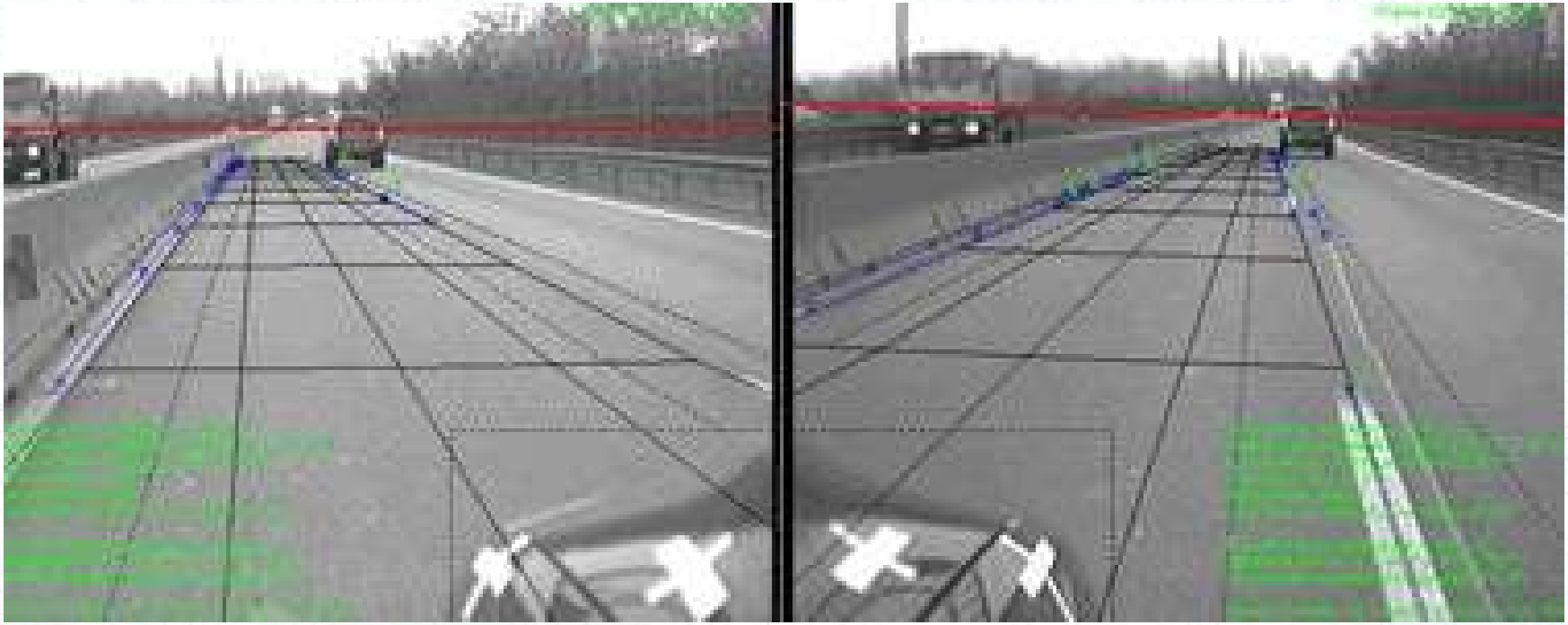




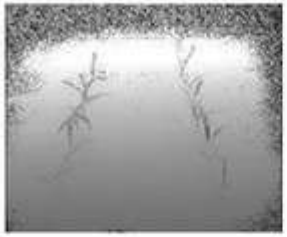
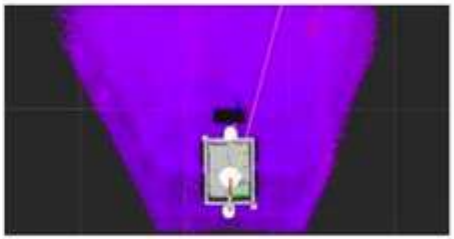
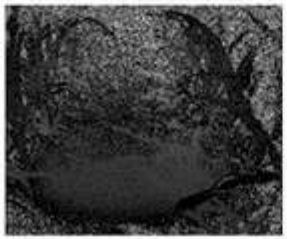
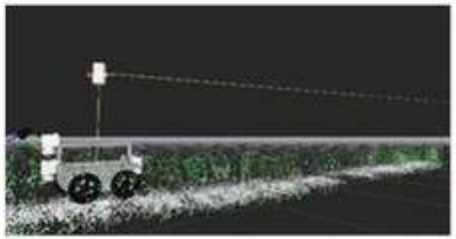
Стерео реконструкция



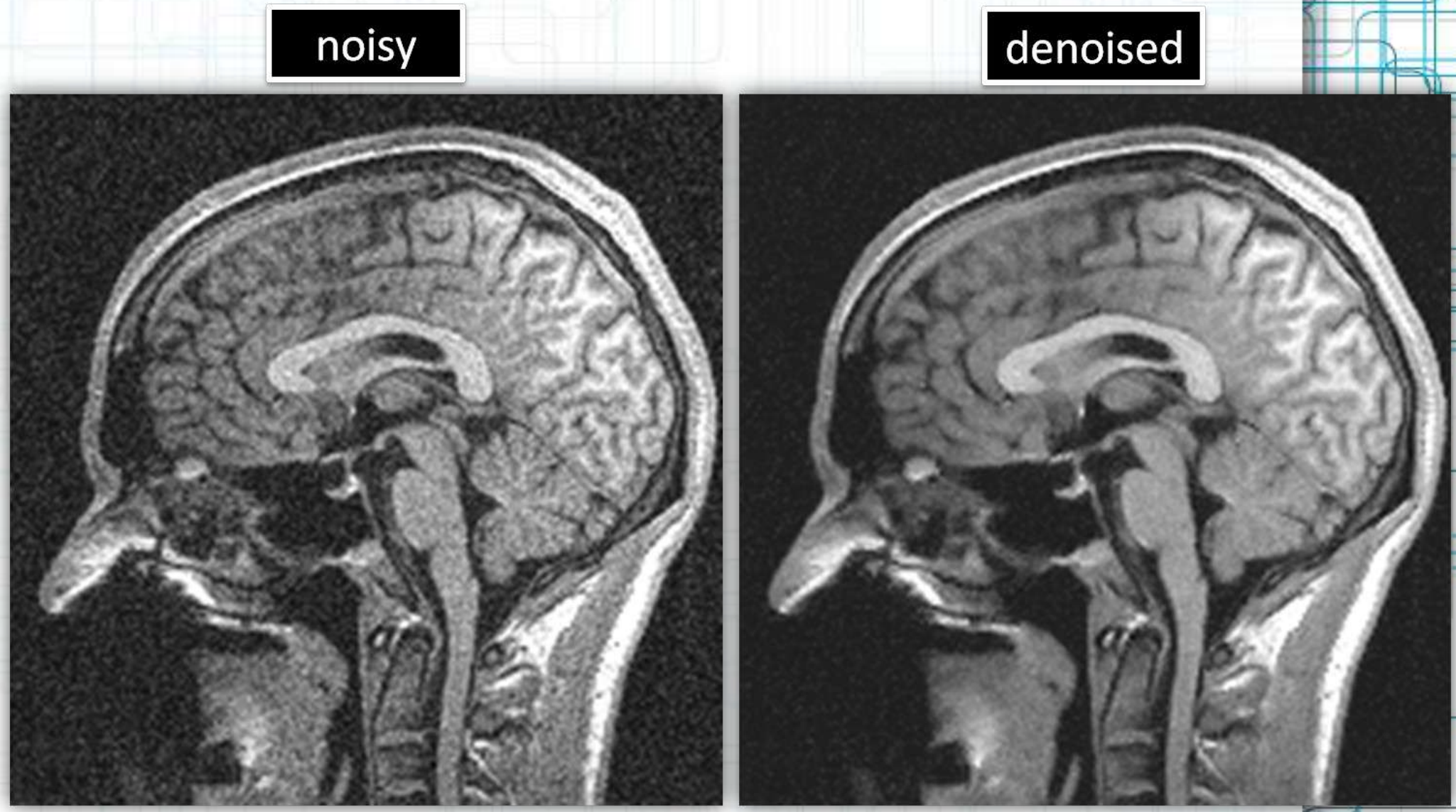
Стерео реконструкция

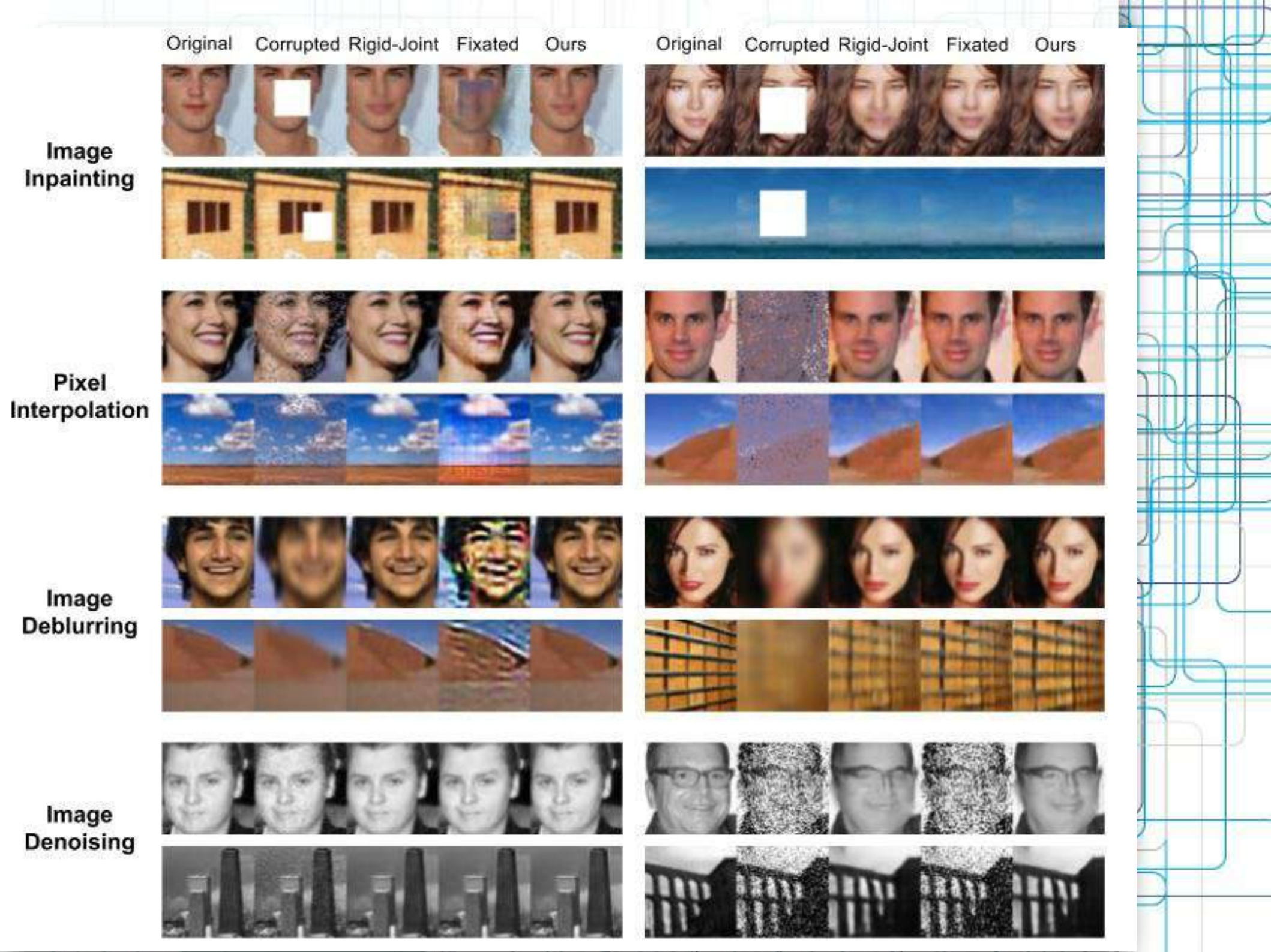


Стерео реконструкция

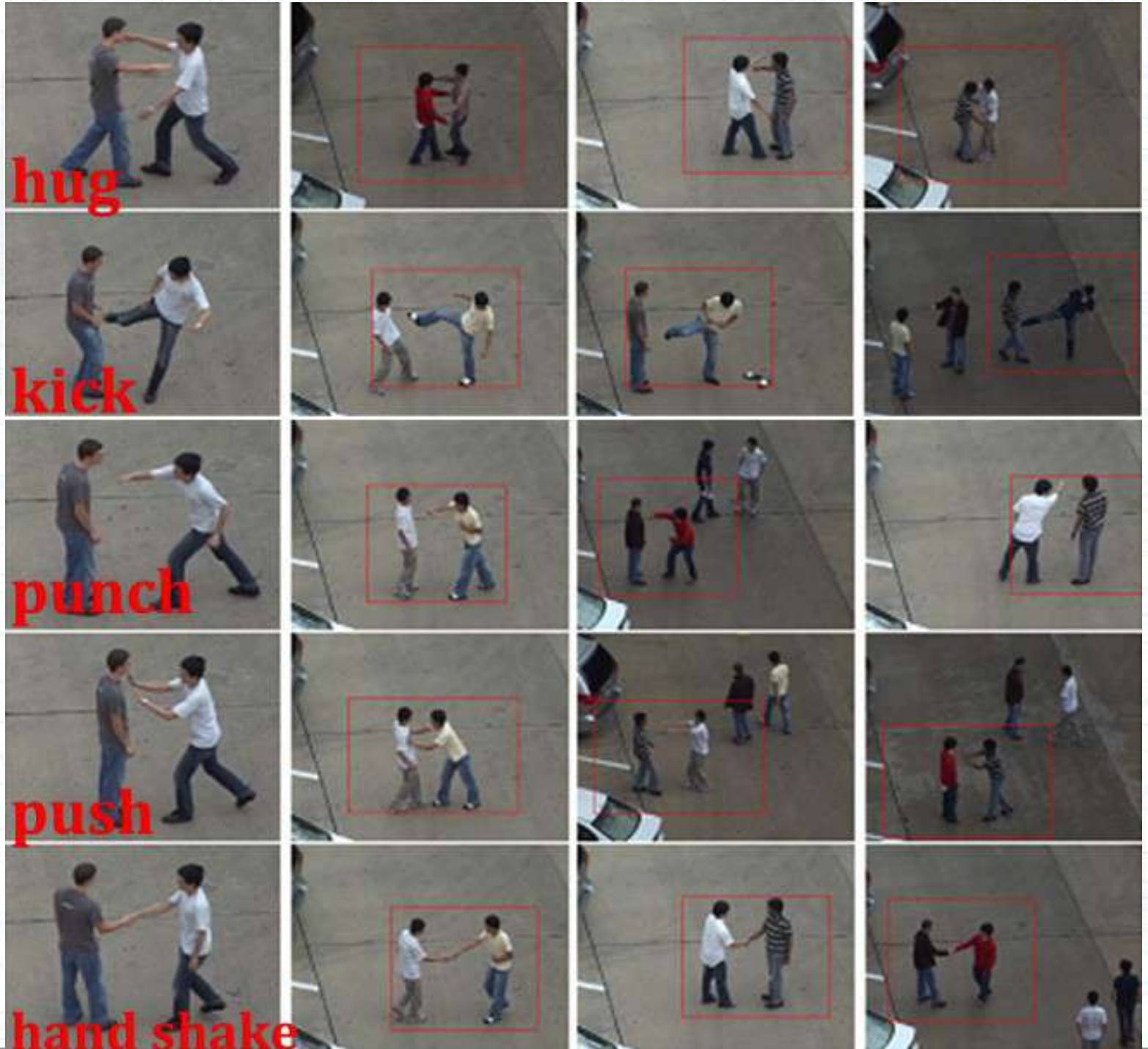
Environment	Lighting	RGB image	NIR image	Depth image	Point clouds (3-D reconstruction)
Greenhouse	Sun shadow				
Greenhouse	Night				
Open field	Sunny				

Восстановление зашумленных изображений



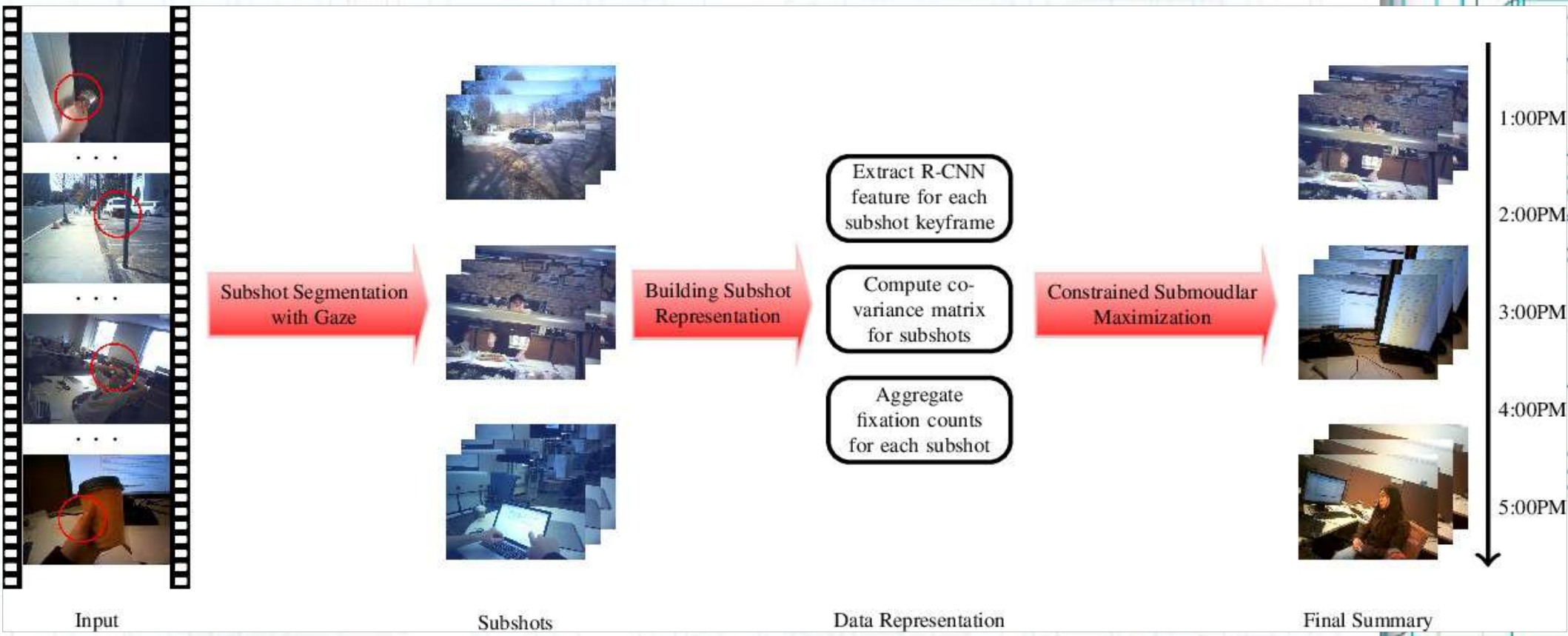

Original
Corrupted
Rigid-Joint
Fixated
Ours
Original
Corrupted
Rigid-Joint
Fixated
Ours
Image Inpainting
Pixel Interpolation
Image Deblurring
Image Denoising

Выявление аномалий

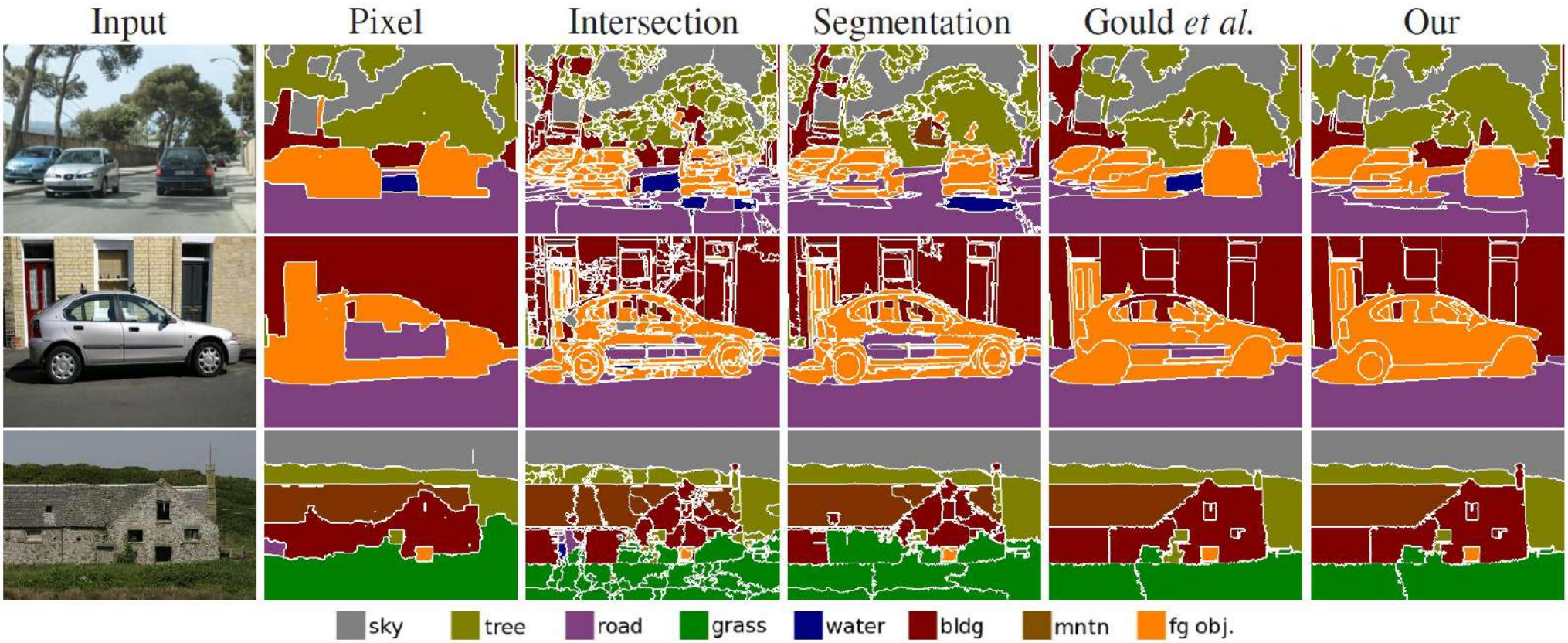


Реферирование видео (video summarization)

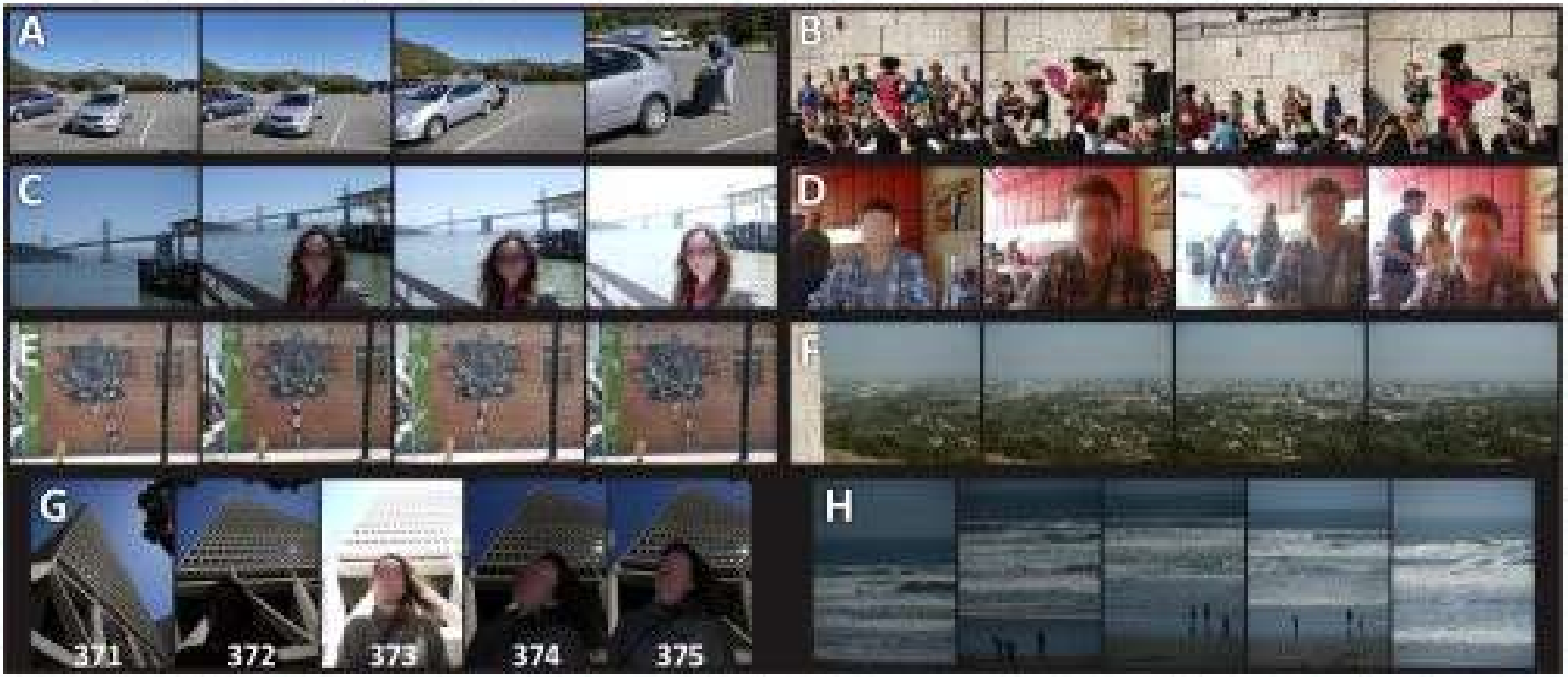
Реферирование видео (video summarization)



Сегментация изображений



Поиск дубликатов

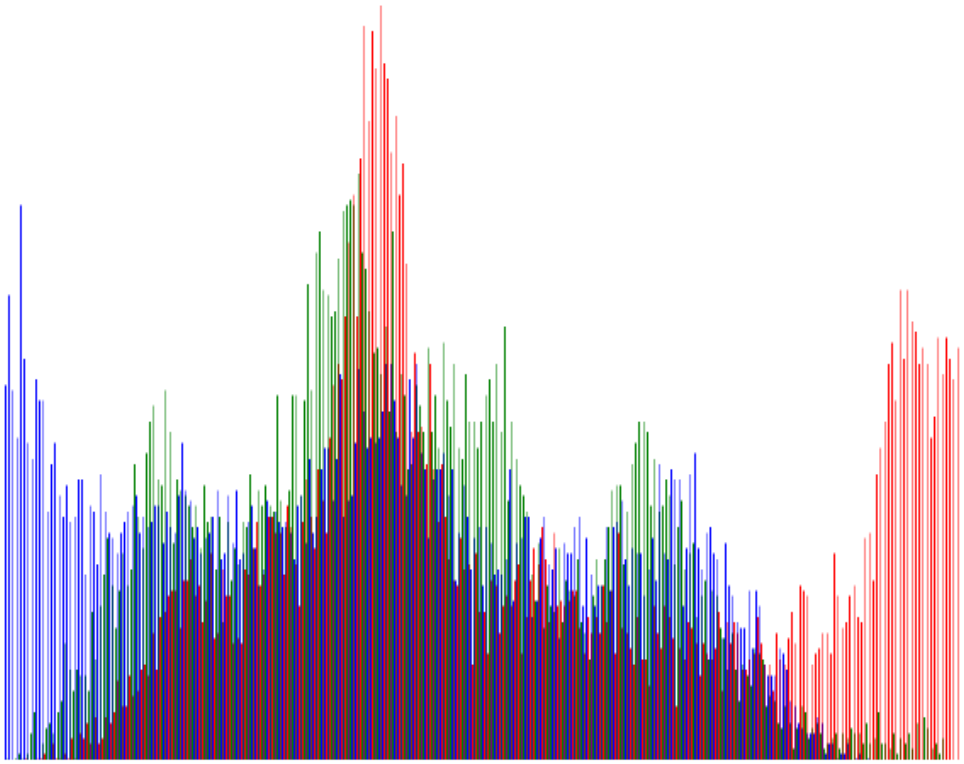


Создание и обнаружение фейковых фото и видео

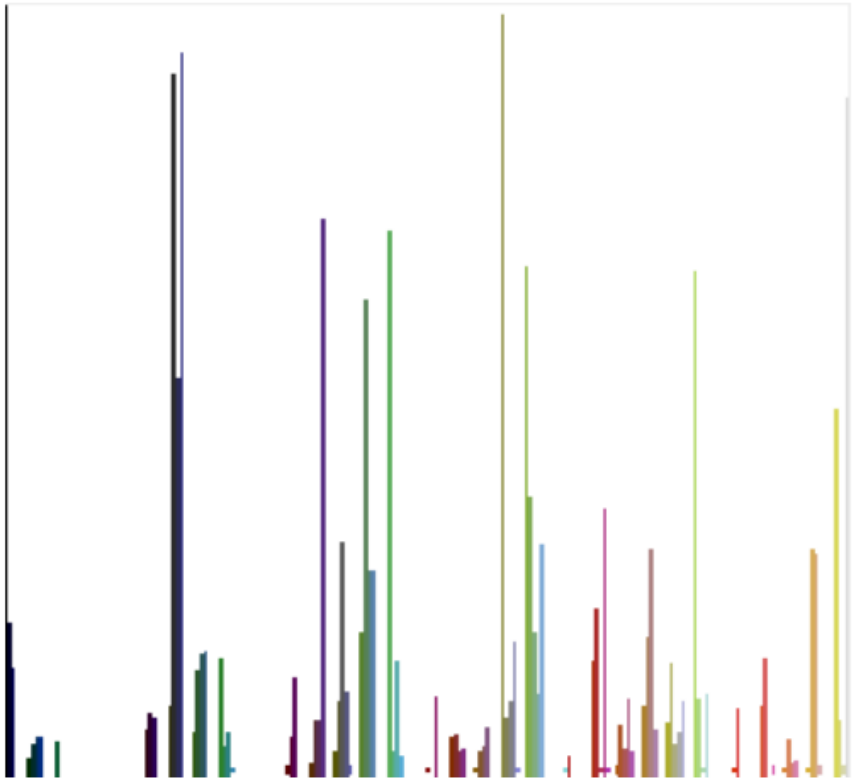
Признаки изображений

- Глобальные:
  - полноцветные гистограммы
  - контекст формы
  - GIST
- Локальные:
  - Детекторы: LoG, DoG, DoH, MSER, Hessian Affine, KAZE, FAST
  - Дескрипторы: SIFT, GLOH, SURF, LIOP, BRIEF, ORB, FREAK, BRISK, CARD
- Свертки с ядрами

Полноцветная гистограмма

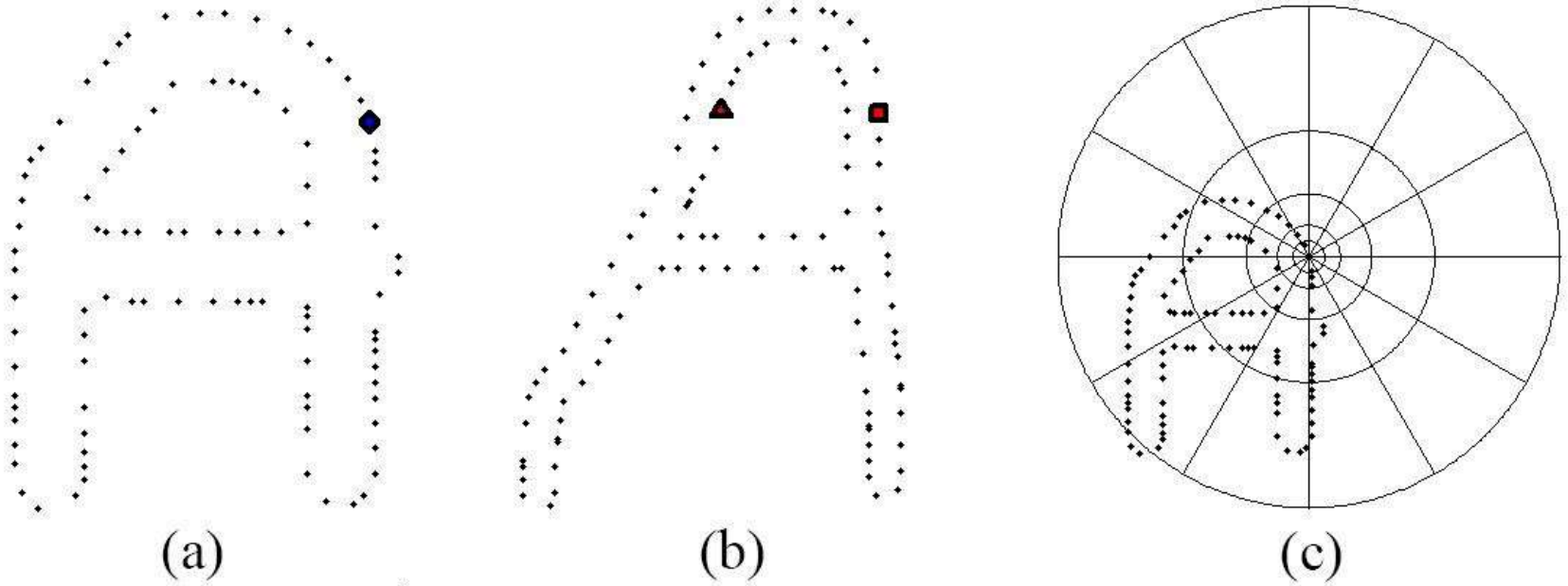


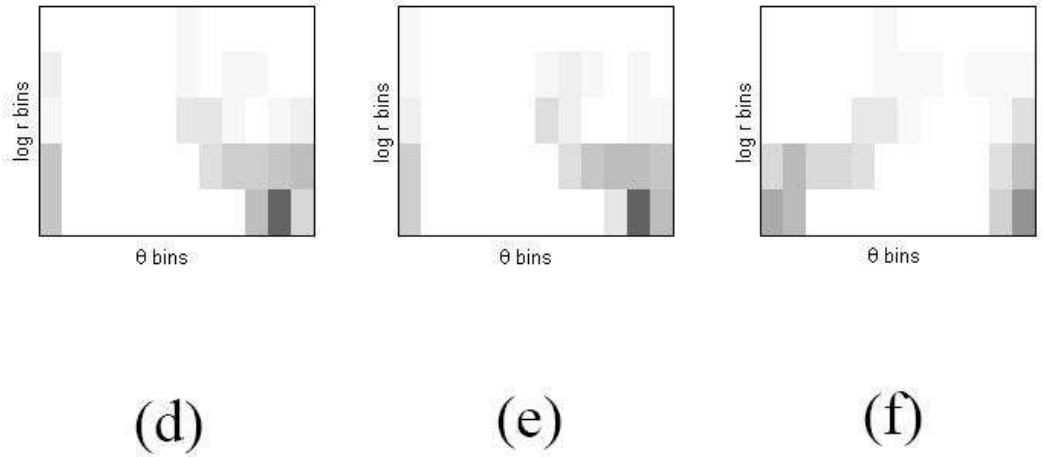
Обычная RGB-гистограмма



Полноцветная гистограмма

Контекст формы

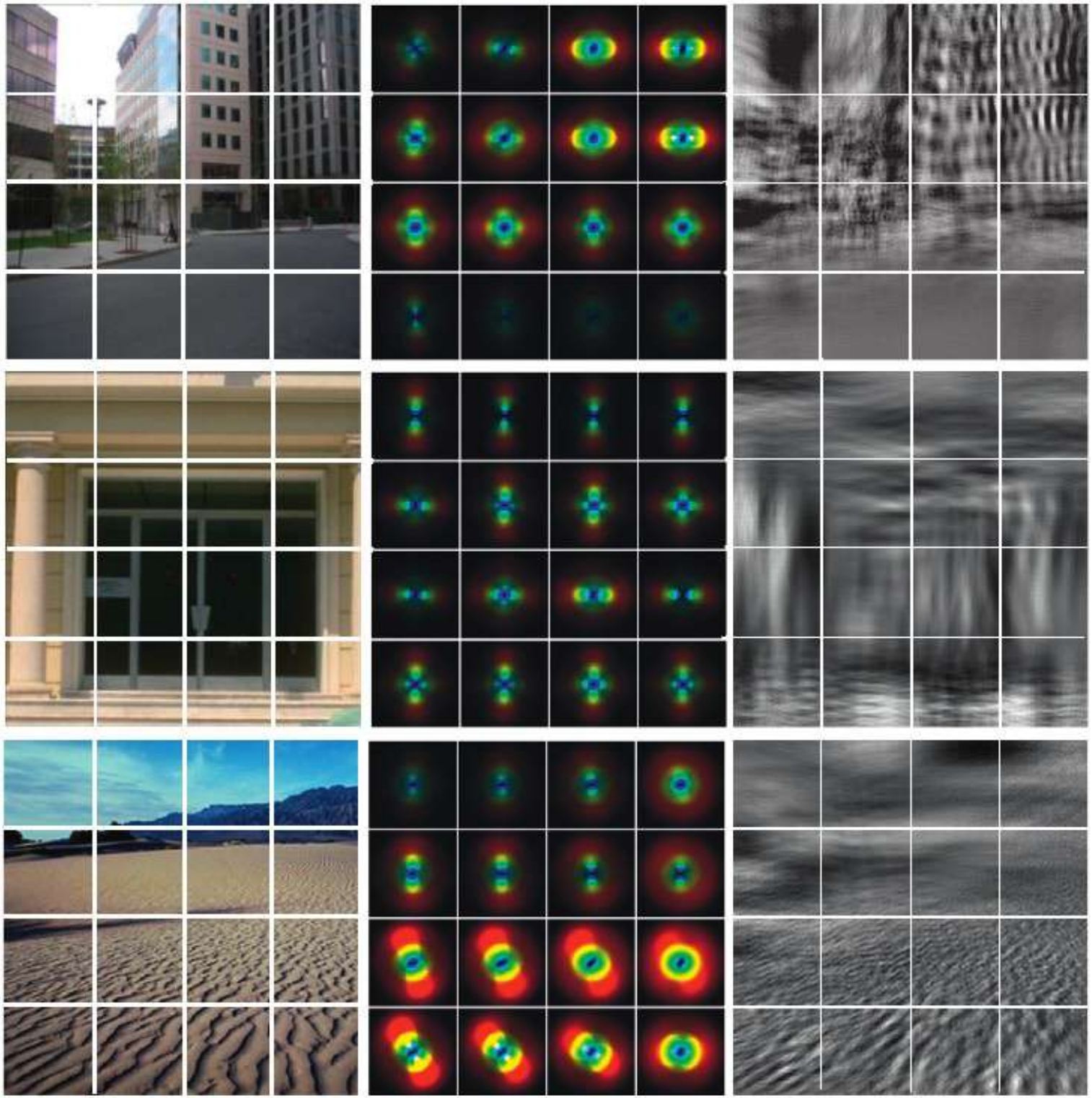




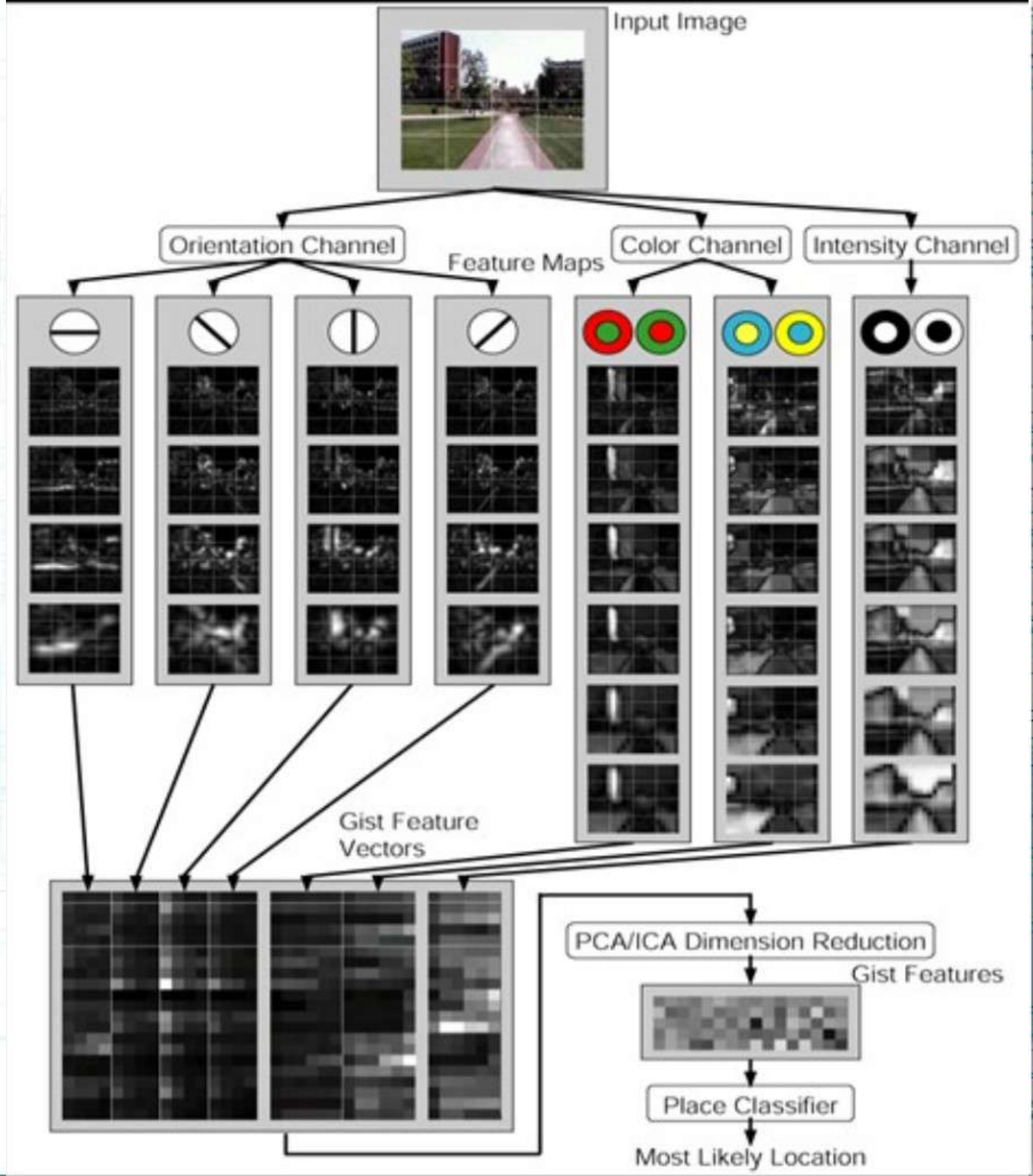
GIST (суть)

$$g_k = \sum_{x,y} w_k(x,y) \times \left| I(x,y) \otimes h_k(x,y) \right|^2$$

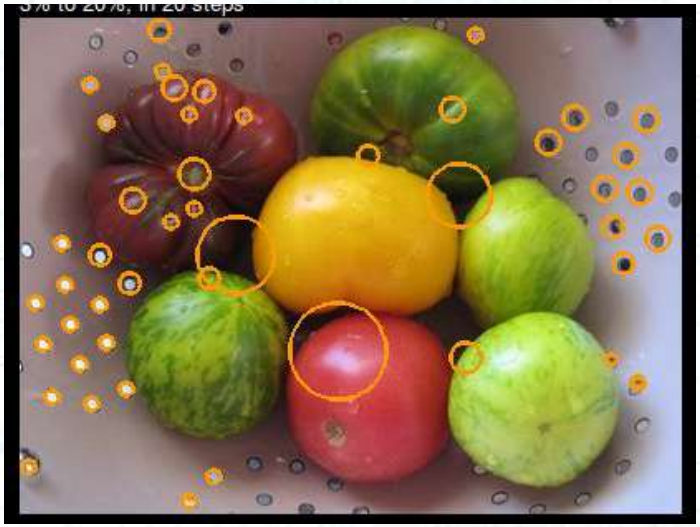
- $\otimes$ - свертка
- I(x,y) – яркость
- h_k(x,y) – фильтр Габора (6 ориентаций, 4 масштаба)
- w_k(x,y) – окно (делит изображение на 16 частей)
- Итого: 16x6x4 = 384 признака

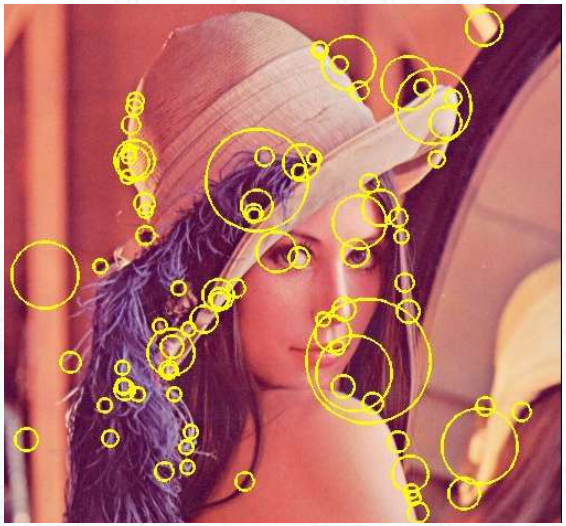


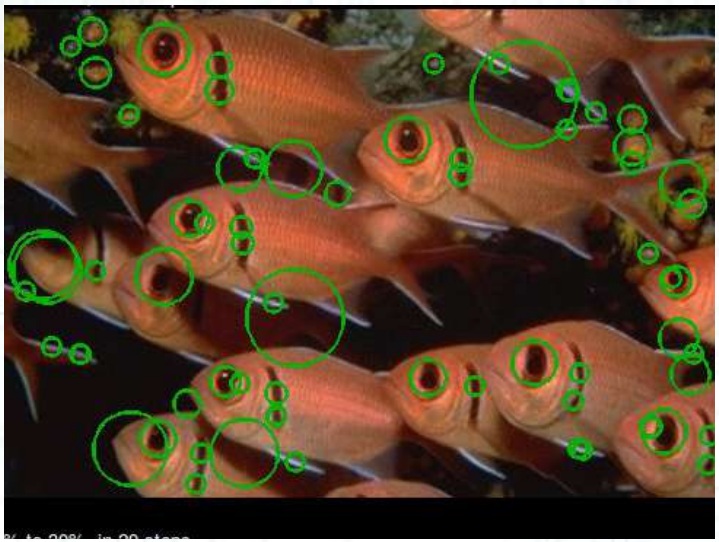
GIST


Input Image
Orientation Channel
Feature Maps
Color Channel
Intensity Channel
Gist Feature
Vectors
PCA/ICA Dimension Reduction
Gist Features
Place Classifier
Most Likely Location

Детектирование особых точек (Blob detection)







Детектирование особых точек (Blob detection)

- LoG - Laplacian of Gaussian

$$g(x,y,t) = \frac{1}{2\pi t^2} e^{\frac{x^2+y^2}{2t^2}} \qquad G = g(x,y,t) * \mathrm{image}(x,y)$$

$$LG = \Delta G = G_{xx} + G_{yy}$$

- DoG – Difference of Gaussians

$$G_t = \Delta G \qquad LG \approx \frac{G(x,y,t_2) - G(x,y,t_1)}{t_2 - t_1}$$

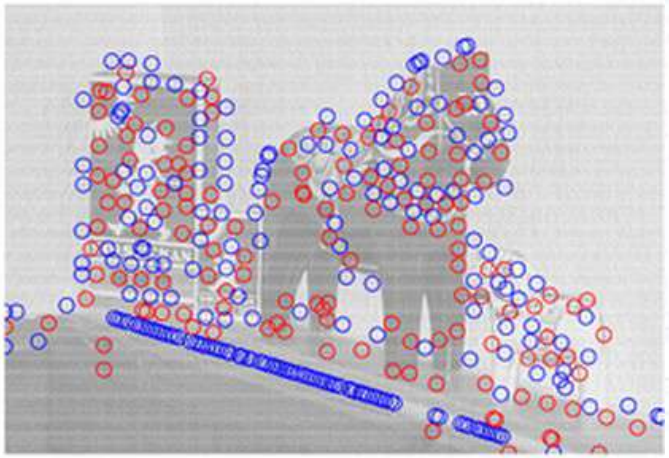
- DoH - Determinant of Hessian – инвариантен относительно аффинных преобразований

$$\det H = G_{xx} G_{yy} - G_{xy}^2$$

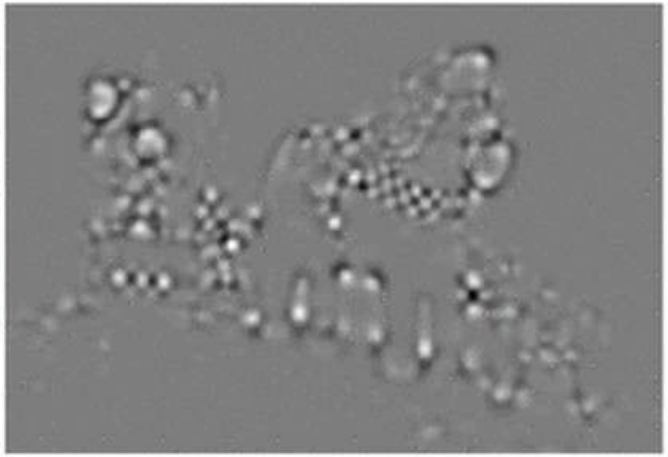
original image f

$\nabla^2 L$

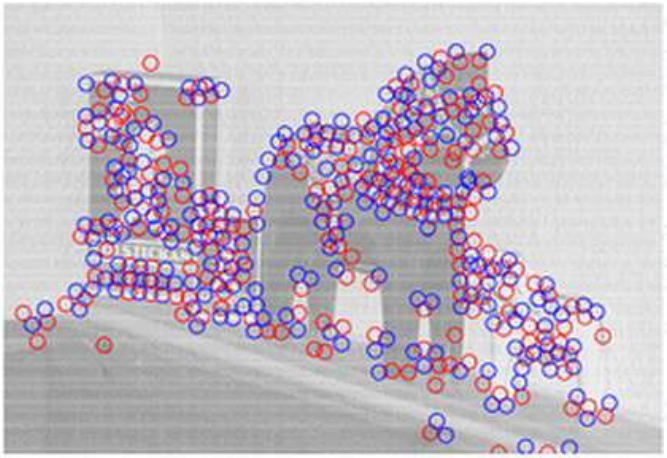
local extrema of $\nabla^2 L$



$\det \mathcal{H}L$



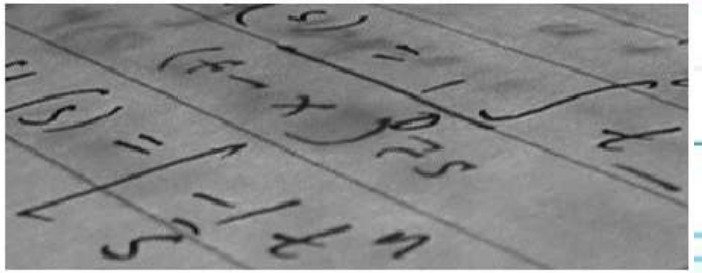
local extrema of $\det \mathcal{H}L$



Детектирование угловых точек

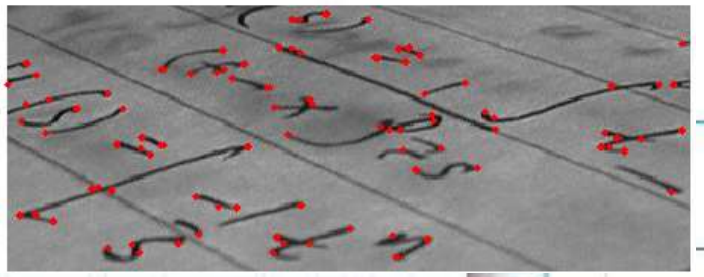
- Моравек сравнивал окрестность каждой точки изображения, смещенную в разных направлениях.
  - Если не отличается, тогда это внутренняя точка
  - Если отличается только в одном направлении (и противоположном) – это кусок прямой границы
  - Если отличается по всем направлениям – это угол
- Обобщение – структурный тензор:

$$A = \sum_u \sum_v w(u,v) \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix} = \begin{bmatrix} \langle I_x^2 \rangle & \langle I_x I_y \rangle \\ \langle I_x I_y \rangle & \langle I_y^2 \rangle \end{bmatrix}$$



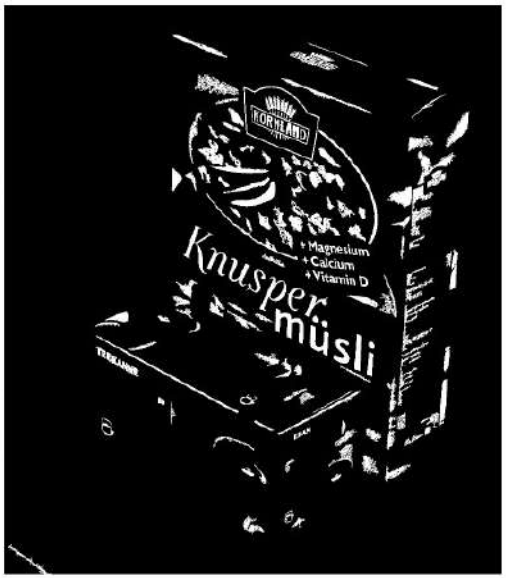
точка угловая ттт, когда оба с.з. велики

$$M_c = \lambda_1 \lambda_2 - \kappa (\lambda_1 + \lambda_2)^2 = \det(A) - \kappa \, \mathrm{trace}^2(A)$$



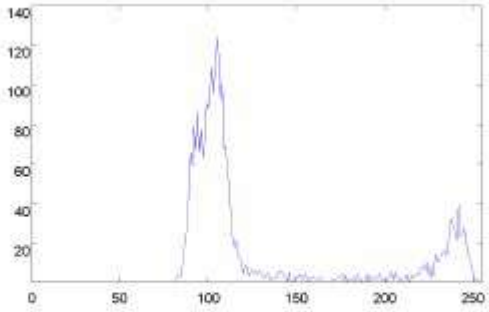
MSER - Maximally stable extremal regions

(a) Gray scale input image



(b) Detected MSERs

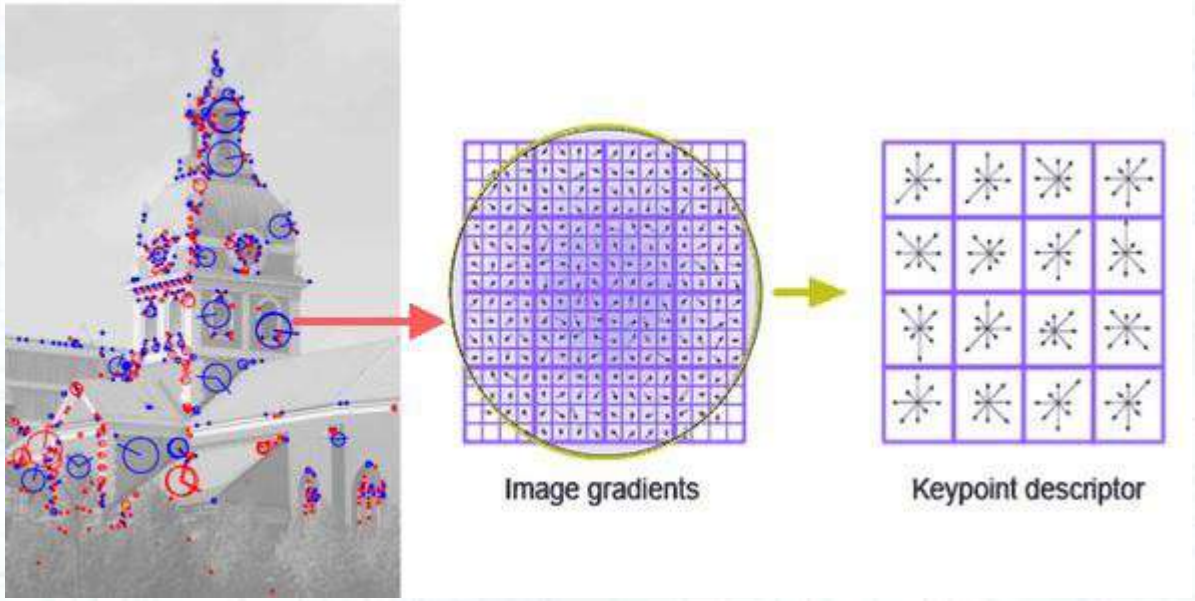
(a) Input Image

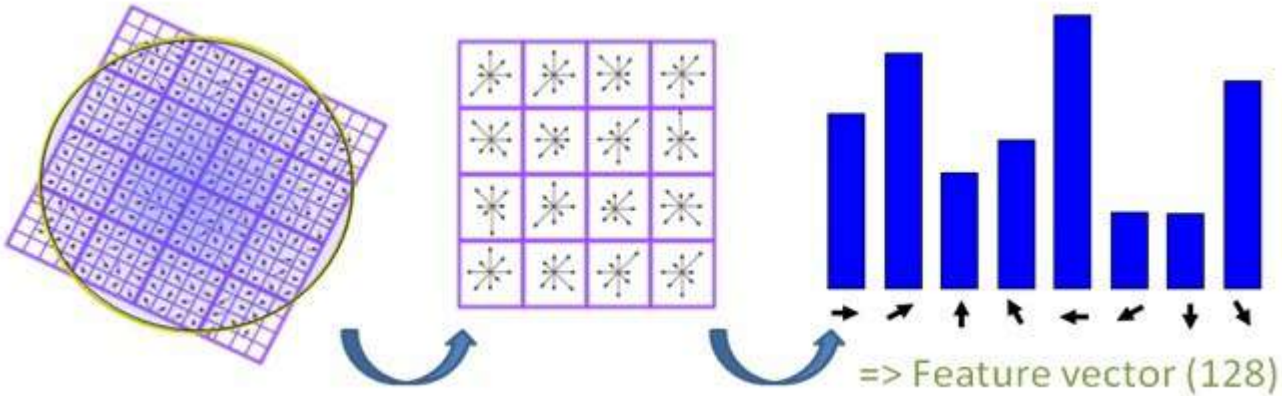


(b) Image histogram

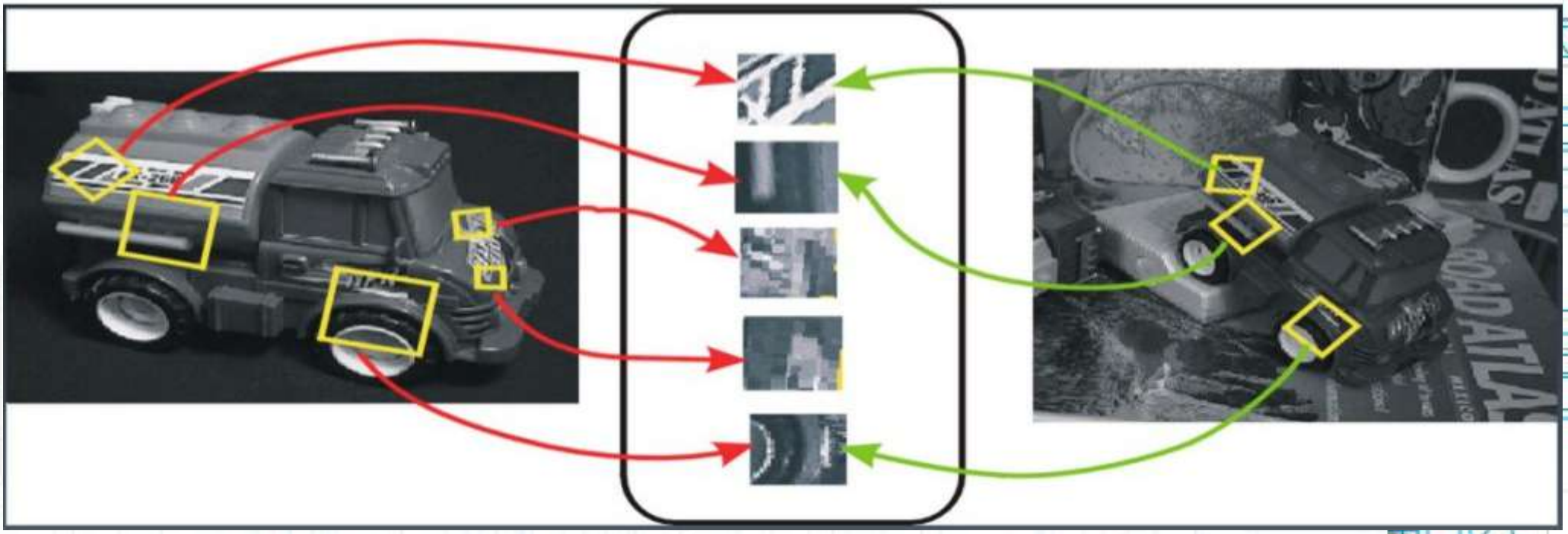
(c) MSER result

SIFT – Scale Invariant Feature Transform





SIFT – Scale Invariant Feature Transform

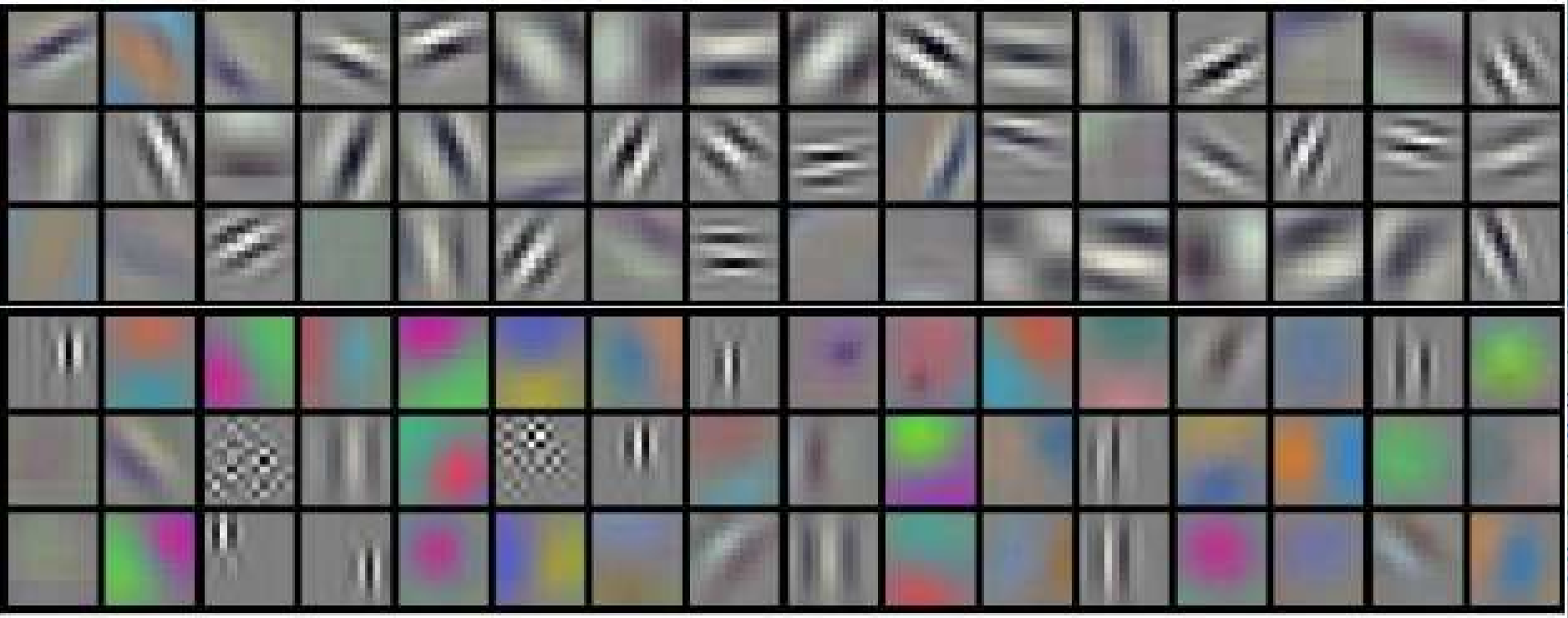


Поиск аффинного преобразования конфигураций особых точек – решаем СЛАУ методом наименьших квадратов:

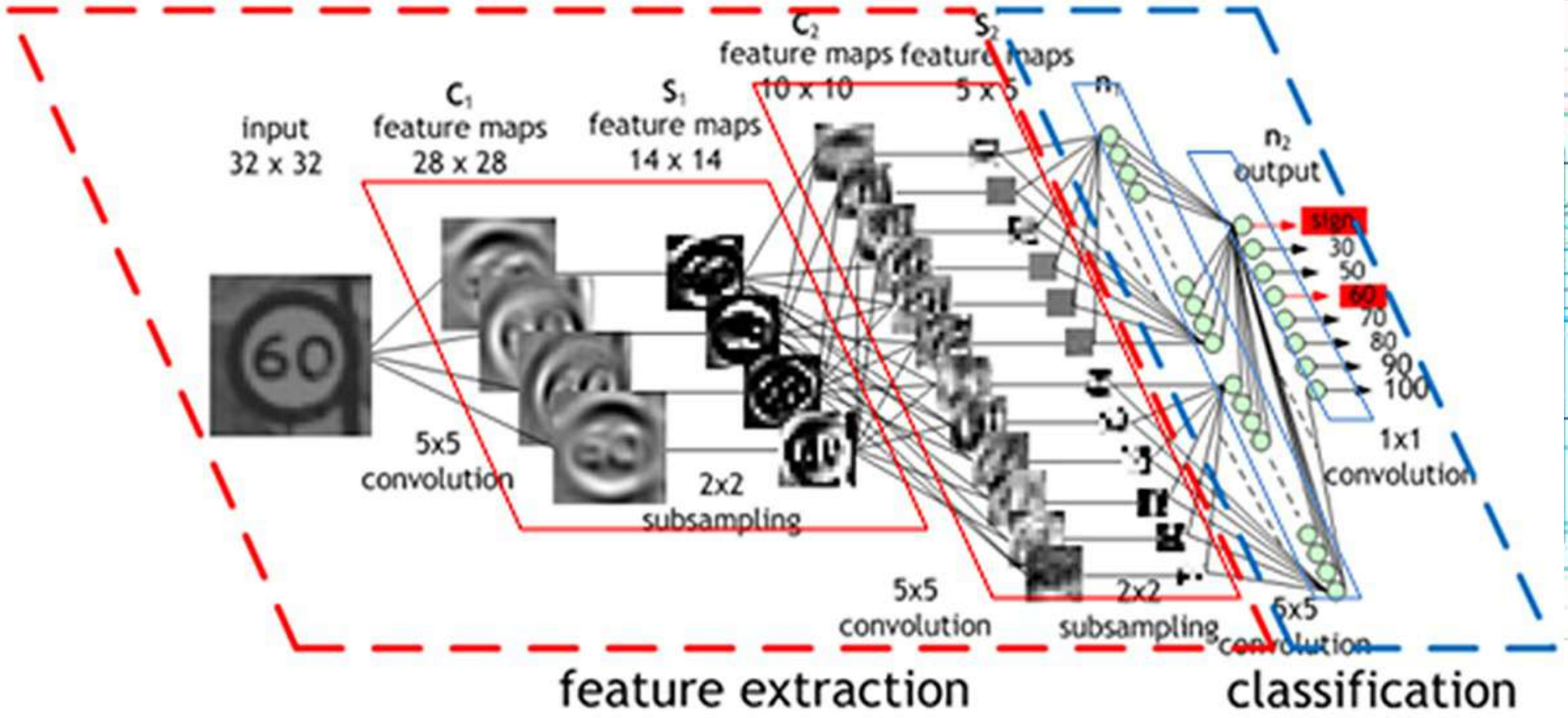
$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} m1 & m2 \\ m3 & m4 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} tx \\ ty \end{bmatrix}$$

$$\begin{bmatrix} x & y & 0 & 0 & 1 & 0 \\ 0 & 0 & x & y & 0 & 1 \\ \dots & & & & & \\ \dots & & & & & \end{bmatrix} \begin{bmatrix} m1 \\ m2 \\ m3 \\ m4 \\ tx \\ ty \end{bmatrix} = \begin{bmatrix} u \\ v \\ . \\ . \end{bmatrix}$$

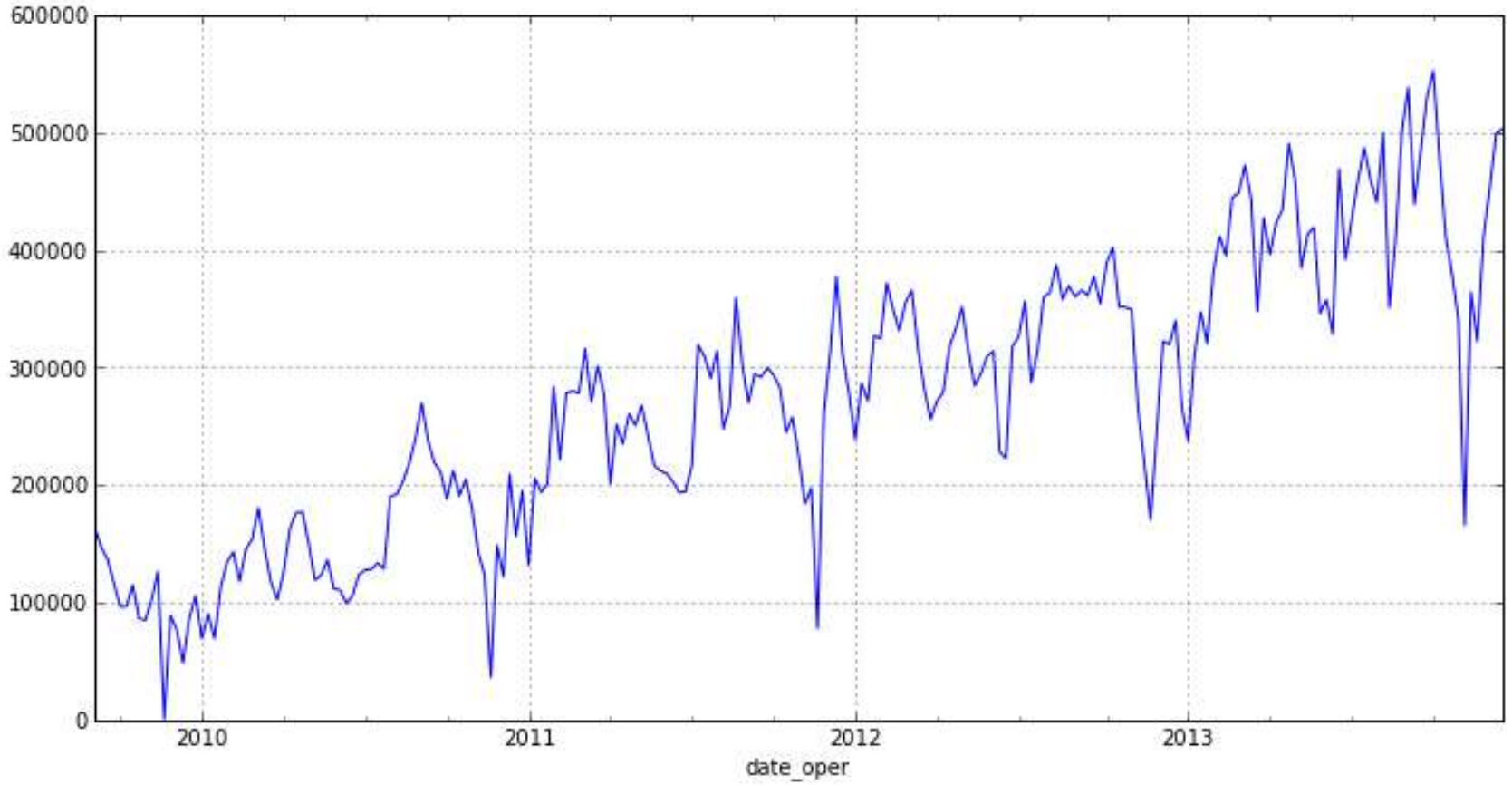
Свертки с ядрами



Сверточные нейронные сети



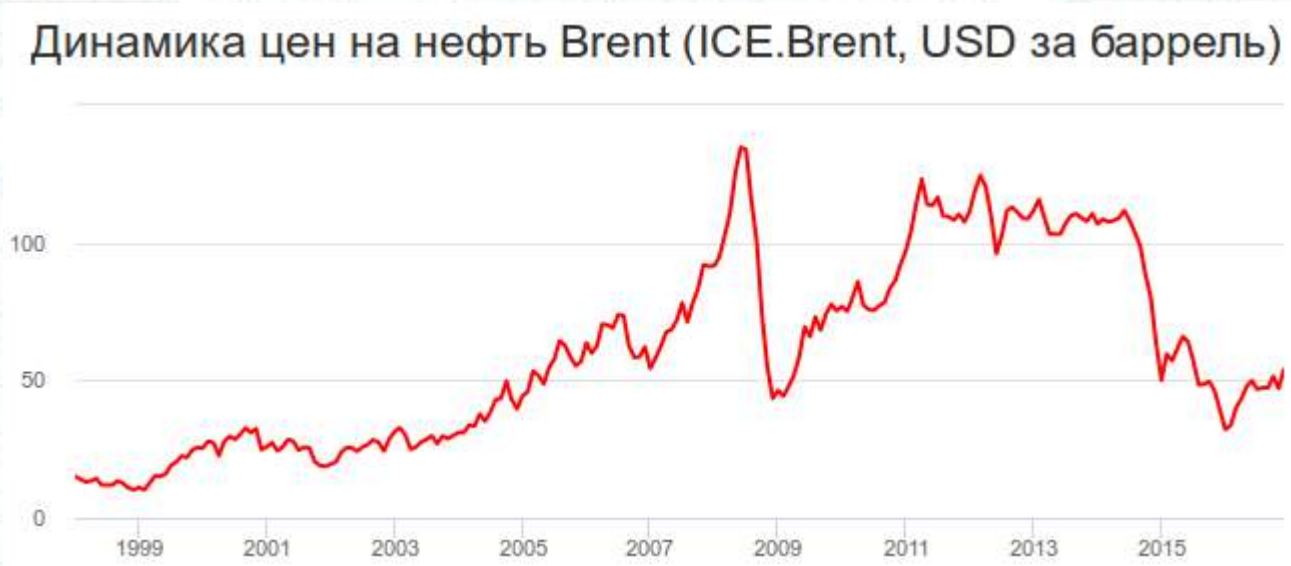
Прогнозирование временных рядов



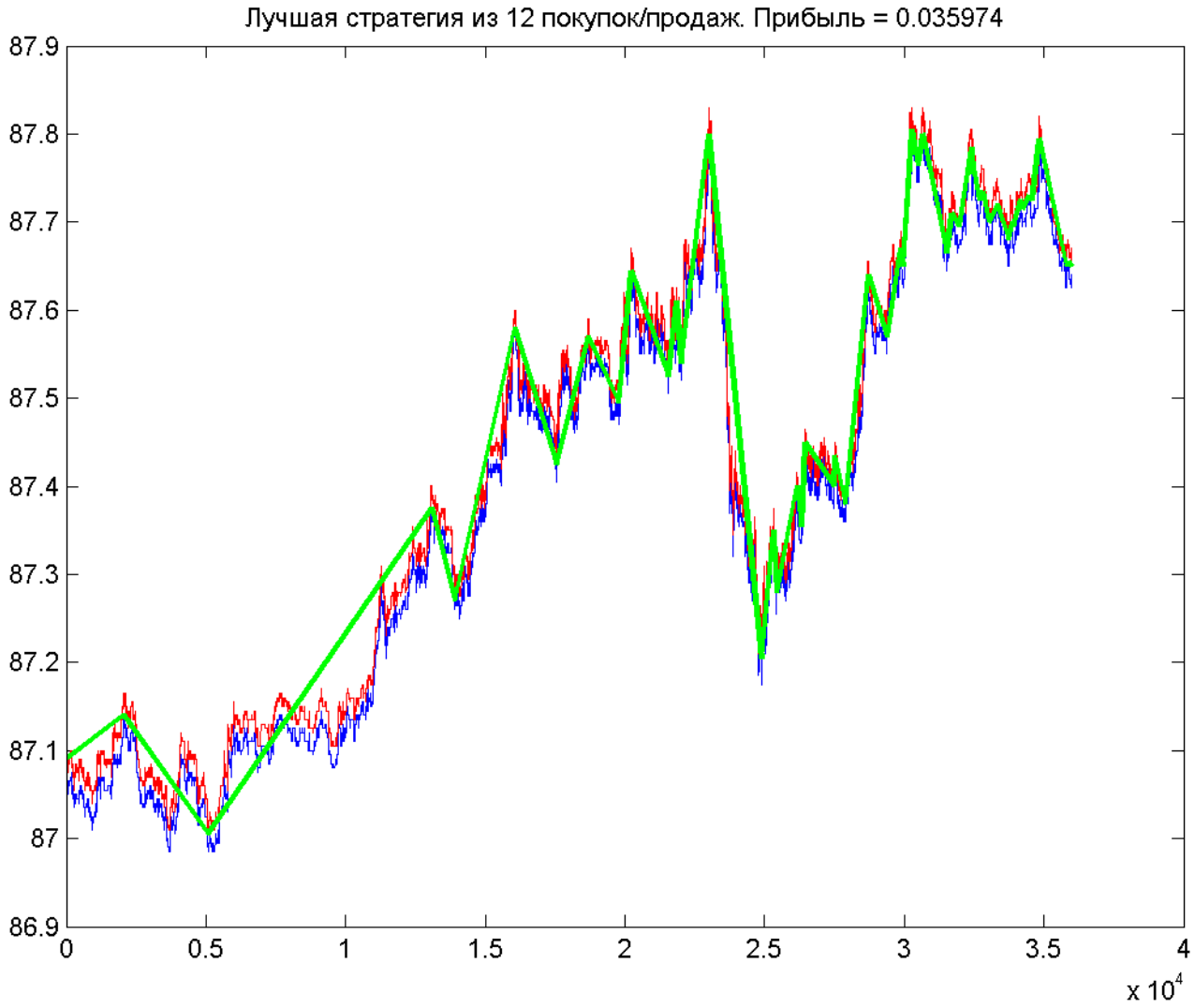
Содержание

- Примеры задач
- Модель авторегрессии
- Модель скользящего среднего
- Модель ARMA
- ARIMA - интегрированная ARMA
- Подбор параметров модели. Авторегрессионный спектр

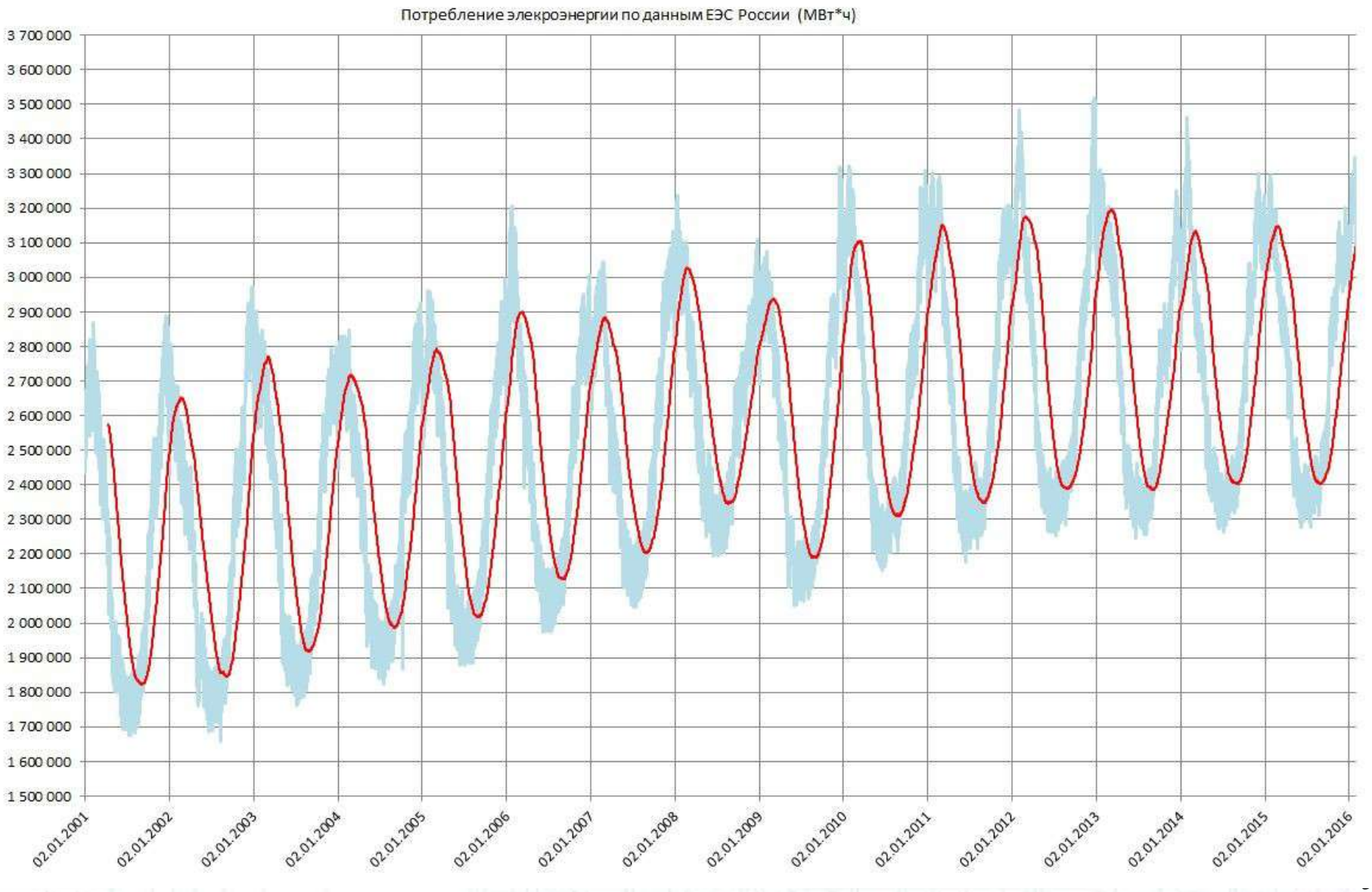
Примеры задач



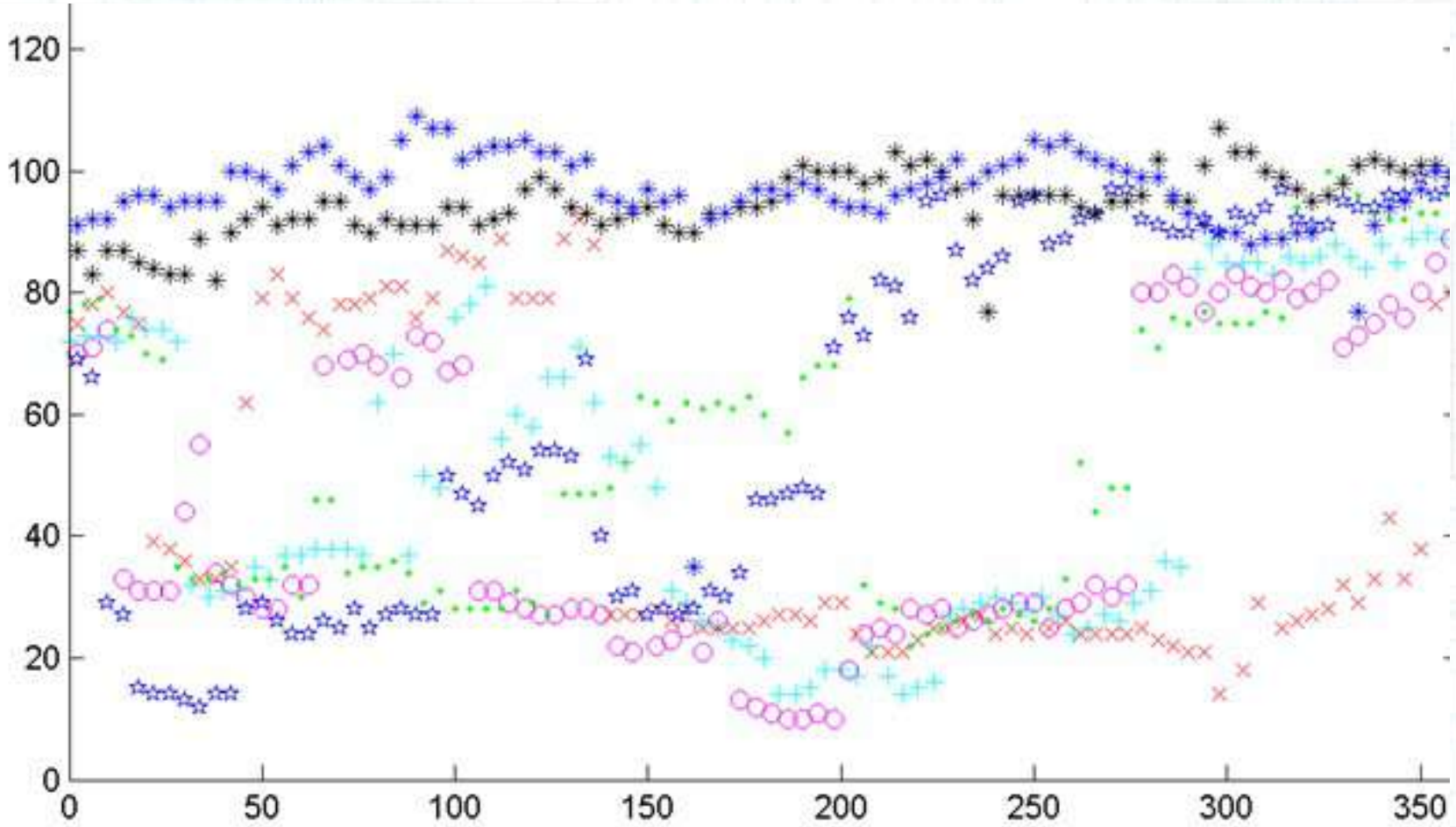
Тиковые данные



Потребление электроэнергии

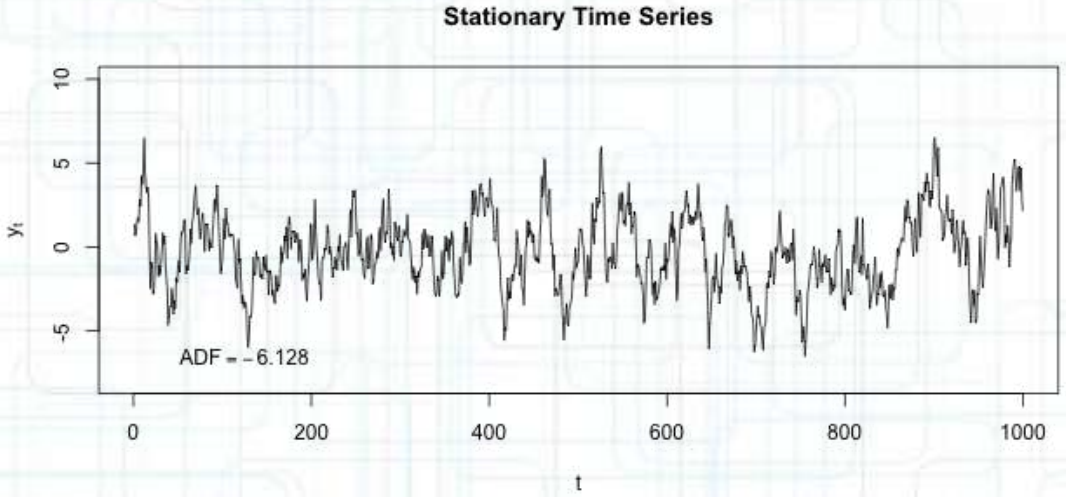


Скорость движения автомобилей в разные дни недели с 15:00 до 21:00

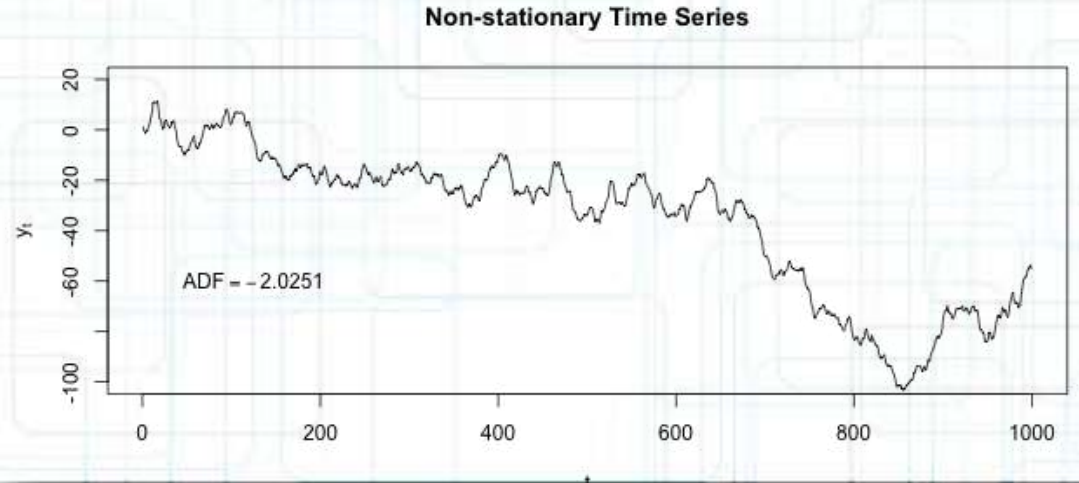


Авторегрессионная модель

- Случайный процесс называется стационарным, если случайное распределение значений функции зависит только от предыдущих значений и расстояния по времени до них, но не от самих значений времени

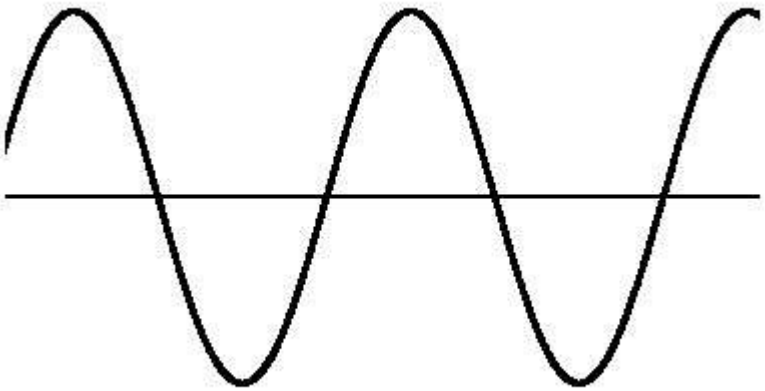


$$X_t = c + \sum_{i=1}^{p} a_i X_{t-i} + \varepsilon_t,$$



Авторегрессионная модель

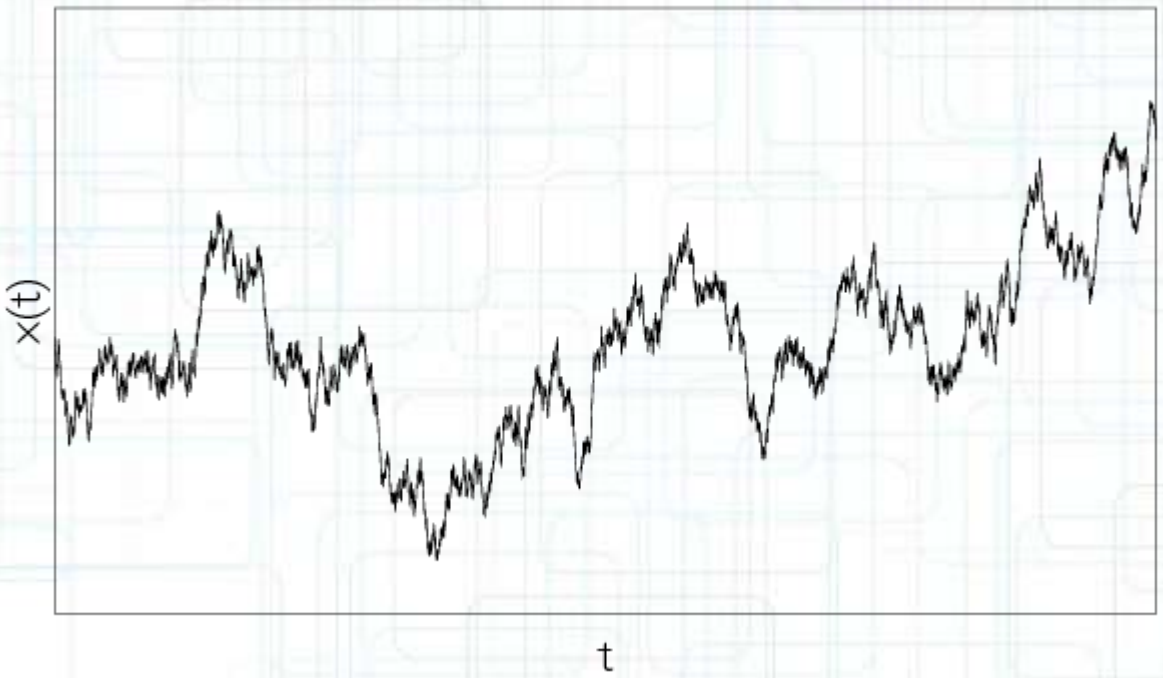
- Реализации стационарного случайного процесса могут быть периодическими
  Пример: процесс X(t)=sin(t+s), где s – равномерно распределенная на [0;2π] случайная величина
- X(t) = X(t-2π)



Авторегресионный процесс первого порядка AR(1)

- Стационарный процесс – марковский, если значение зависит только от ближайшего предыдущего значения
- Пример: случайное блуждание

$$X_t = c + rX_{t-1} + \varepsilon_t$$



Поиск коэффициентов авторегрессии

- Метод наименьших квадратов:

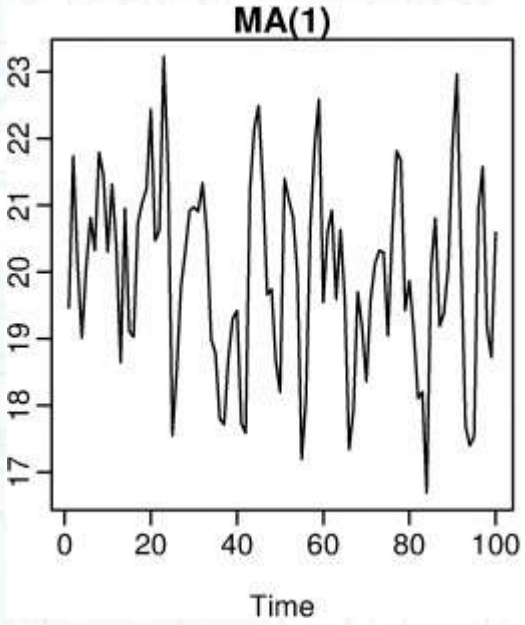
$$\hat{y}_{t+1}(w) = \sum_{j=1}^{n} w_j y_{t-j+1},$$

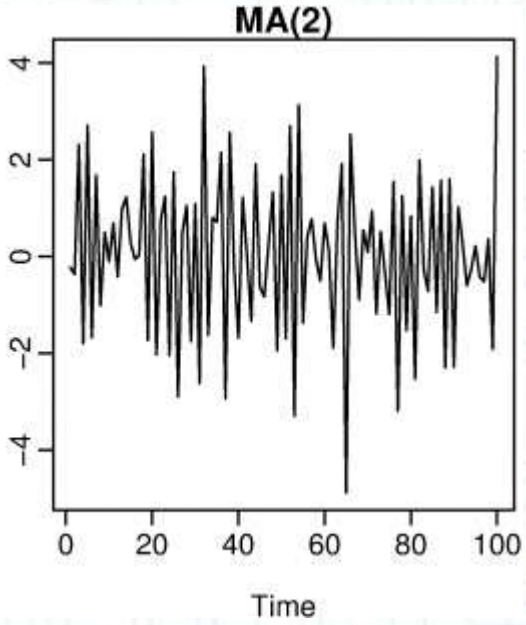
$$\underset{\ell \times n}{F} = \begin{pmatrix} y_{t-1} & y_{t-2} & y_{t-3} & \cdots & y_{t-n} \\ y_{t-2} & y_{t-3} & y_{t-4} & \cdots & y_{t-n-1} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ y_n & y_{n-1} & y_{n-2} & \cdots & y_1 \\ y_{n-1} & y_{n-2} & y_{n-3} & \cdots & y_0 \end{pmatrix}, \quad \underset{\ell \times 1}{y} = \begin{pmatrix} y_t \\ y_{t-1} \\ \cdots \\ y_{n+1} \\ y_n \end{pmatrix}$$

$$Q_t(w, X^\ell) = \sum_{i=n}^{t} \left(\hat{y}_i(w) - y_i\right)^2 = \|Fw - y\|^2 \to \min_w$$

Модель скользящего среднего (Moving Average - MA)

$$X_t = \mu + \varepsilon_t + \theta_1 \varepsilon_{t-1} + \cdots + \theta_q \varepsilon_{t-q}$$





Модель ARMA (autoregressive moving average)

$$X_t = c + \varepsilon_t + \sum_{i=1}^{p} \alpha_i X_{t-i} + \sum_{i=1}^{q} \beta_i \varepsilon_{t-i}.$$

ARIMA - интегрированная модель авторегрессии скользящего среднего (Box-Jenkins model)

Временной ряд называется интегрированным порядка k, если разности ряда порядка k являются стационарными

$$\triangle^d X_t = c + \sum_{i=1}^{p} a_i \triangle^d X_{t-i} + \sum_{j=1}^{q} b_j \varepsilon_{t-j} + \varepsilon_t$$

Y_t → AR filter → Integration filter → MA filter → ε_t

(long term) (stochastic trend) (short term) (white noise error)

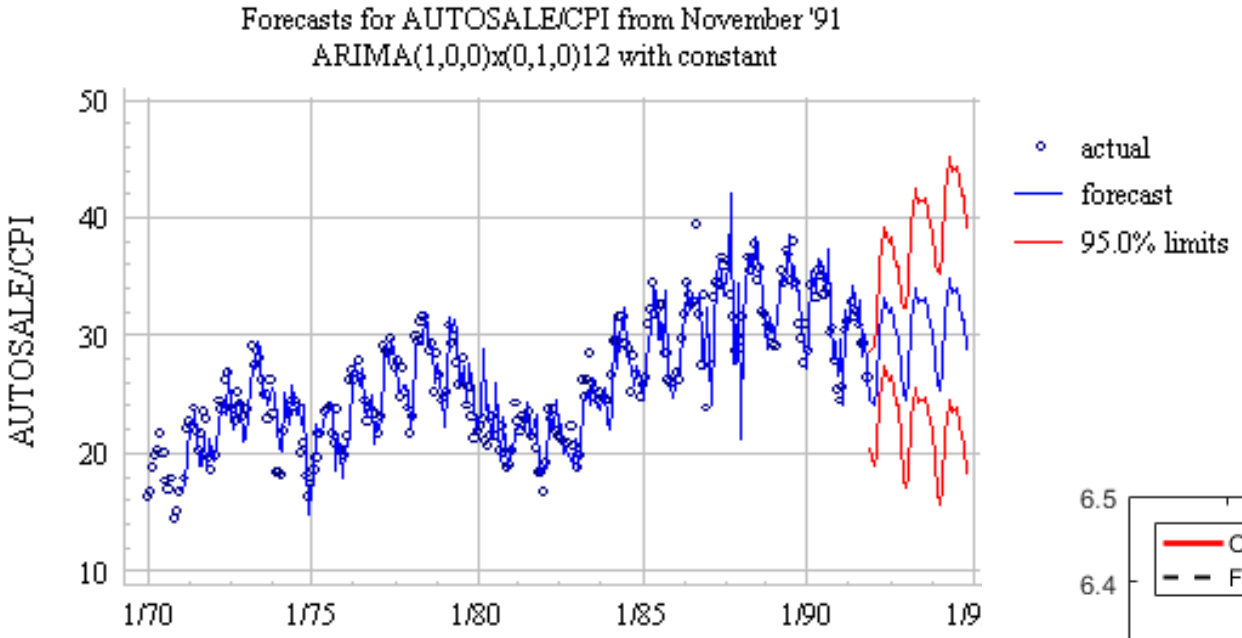
ARIMA (2,0,1) $y_t = a_1 y_{t-1} + a_2 y_{t-2} + b_1 \varepsilon_{t-1}$

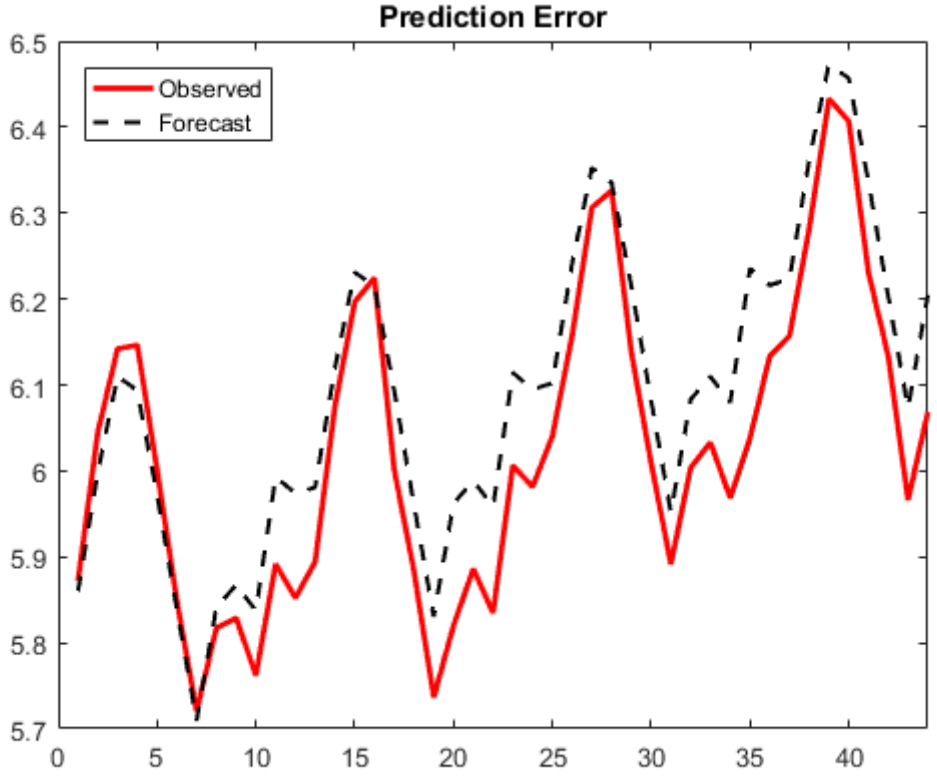
ARIMA (3,0,1) $y_t = a_1 y_{t-1} + a_2 y_{t-2} + a_3 y_{t-3} + b_1 \varepsilon_{t-1}$

ARIMA (1,1,0) $\Delta y_t = a_1 \Delta y_{t-1} + \varepsilon_t$, where $\Delta y_t = y_t - y_{t-1}$

ARIMA (2,1,0) $\Delta yt = a_1 \Delta y_{t-1} + a_2 \Delta y_{t-2} + \varepsilon_t$ where $\Delta yt = yt - yt\text{-}1$

ARIMA

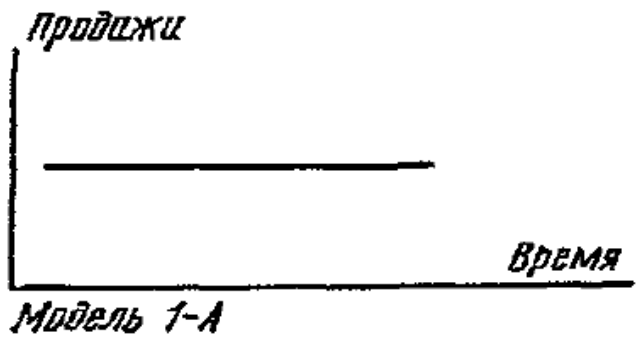

Forecasts for AUTOSALE/CPI from November '91
ARIMA(1,0,0)x(0,1,0)12 with constant
AUTOSALE/CPI
50
40
30
20
10
1/70
1/75
1/80
1/85
1/90
1/9
actual
forecast
95.0% limits


Prediction Error
Observed
Forecast
6.5
6.4
6.3
6.2
6.1
6
5.9
5.8
5.7
0
5
10
15
20
25
30
35
40

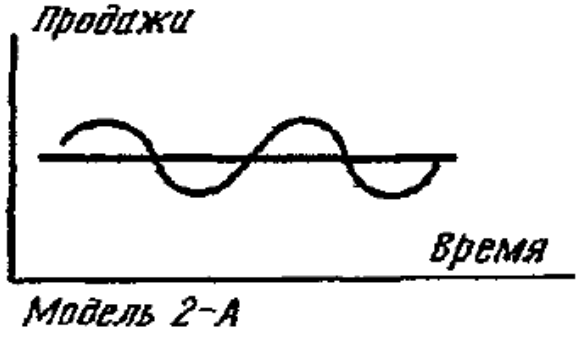
Общий план решения задачи предсказания

- Подготовка данных – сведение к стационарному случайному процессу
- Определение типа модели
- Оценка параметров
- Предсказание

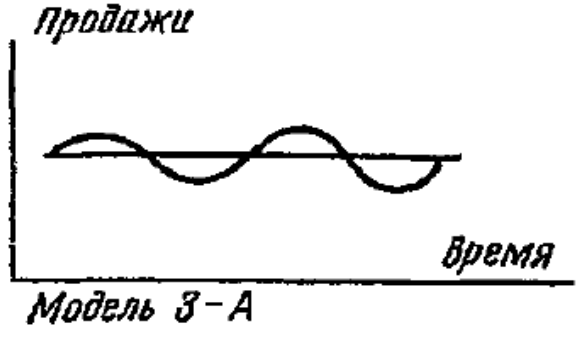
Модели с трендом и сезонным эффектом



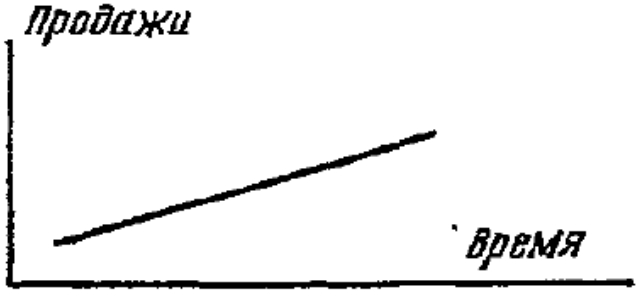
Модель 1-А



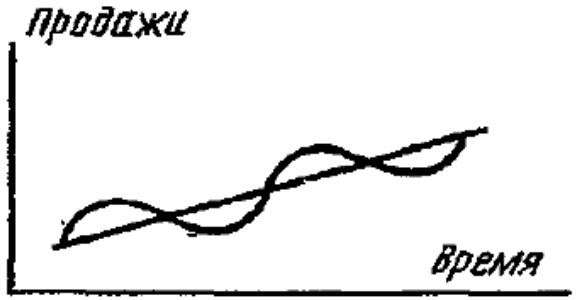
Модель 2-А



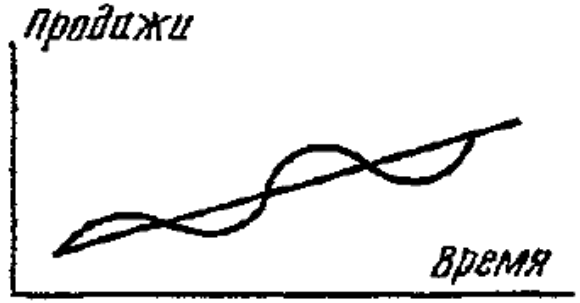
Модель 3-А



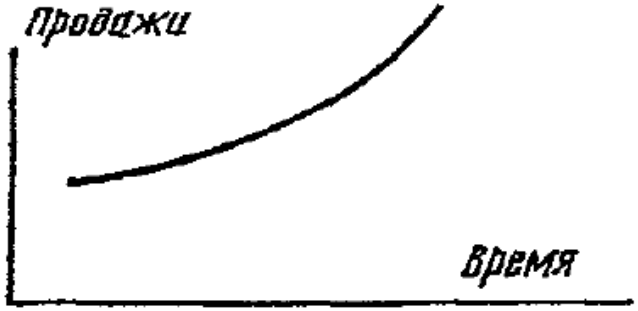
Модель 1-В



Модель 2-В

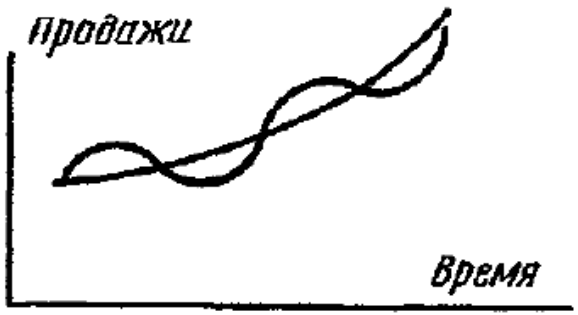


Модель 3-В



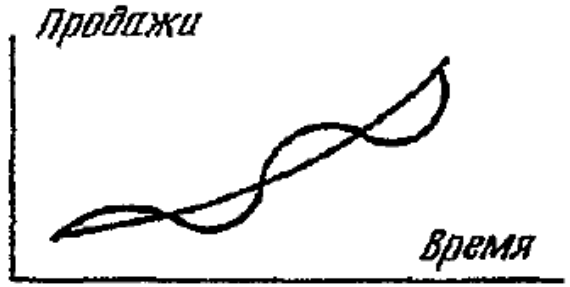
Модель 1-С

а)



Модель 2-С

б)



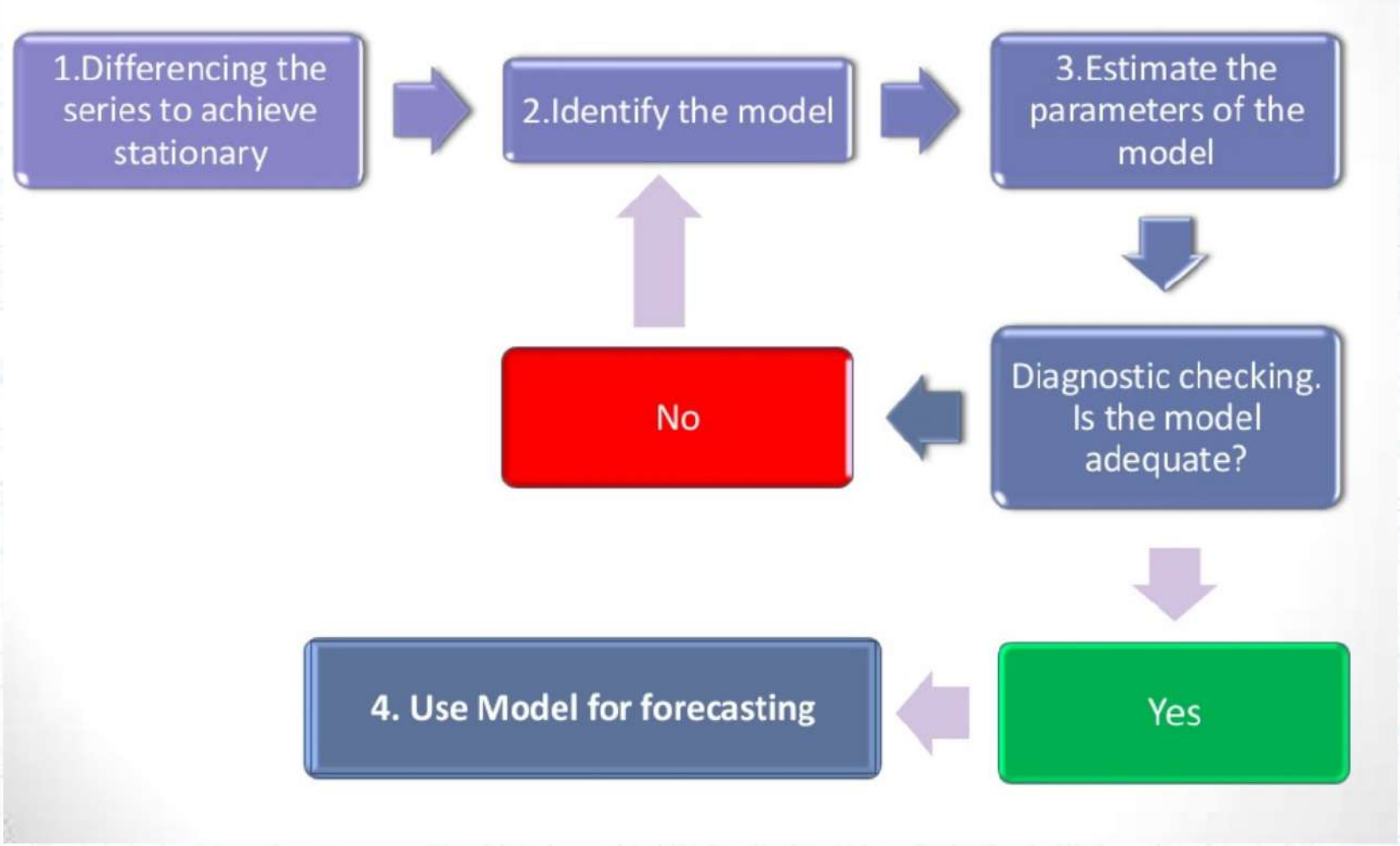
Модель 3-С

в)

Моделирование с помощью ARIMA(p,d,q)

- Стационарность – определение правильного d, исключение сезонности
- Подбор p и q, используя ACF, PACF и unit root тесты
- Проверка – расчет оценки качества
- Оценка невязки – является ли она белым шумом?
- Предсказание

Авторы Бокс и Дженкинс предлагают схему:



Стационарность

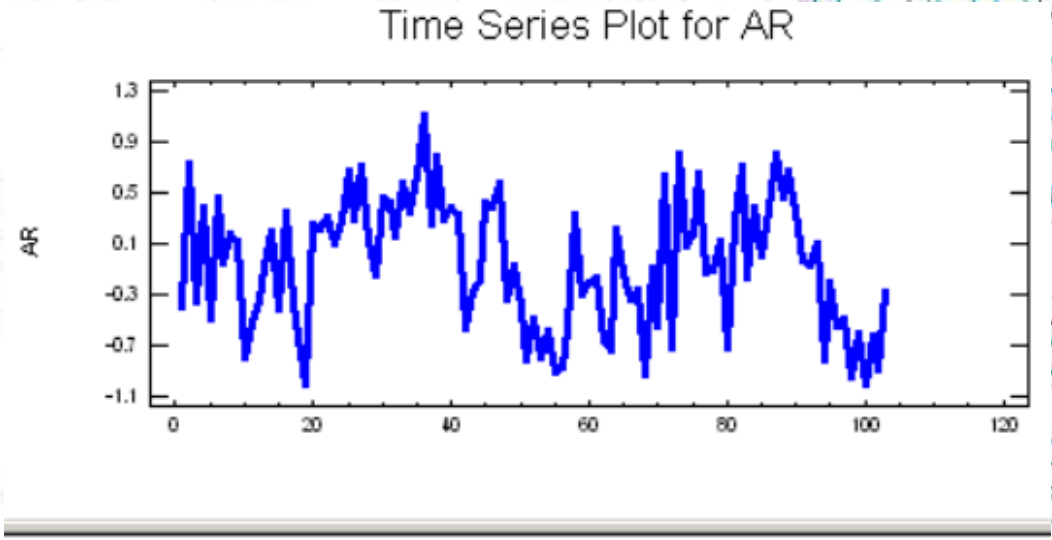
- Процесс из разностей какого порядка является стационарным?
- Исключить сезонность, используя
  - сезонные добавки/множители к среднему значению за этот сезон
  - сезонную ARIMA(p,d,q)x(P,D,Q) модель, например ARIMA(0,0,0)x(0,1,0): $\hat{Y}_t = Y_{t-12} + \mu$

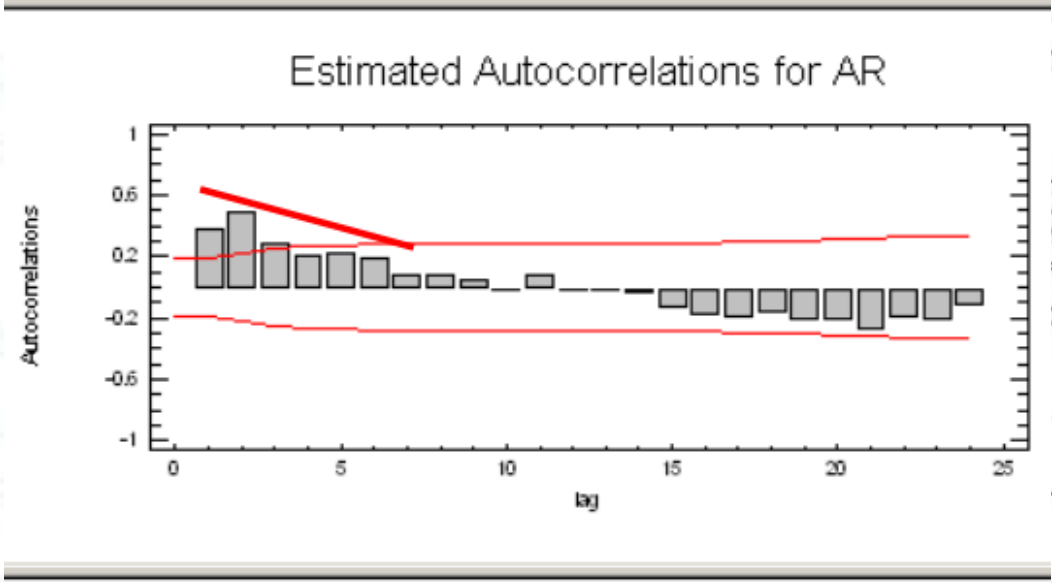
Автокорреляция (ACF)

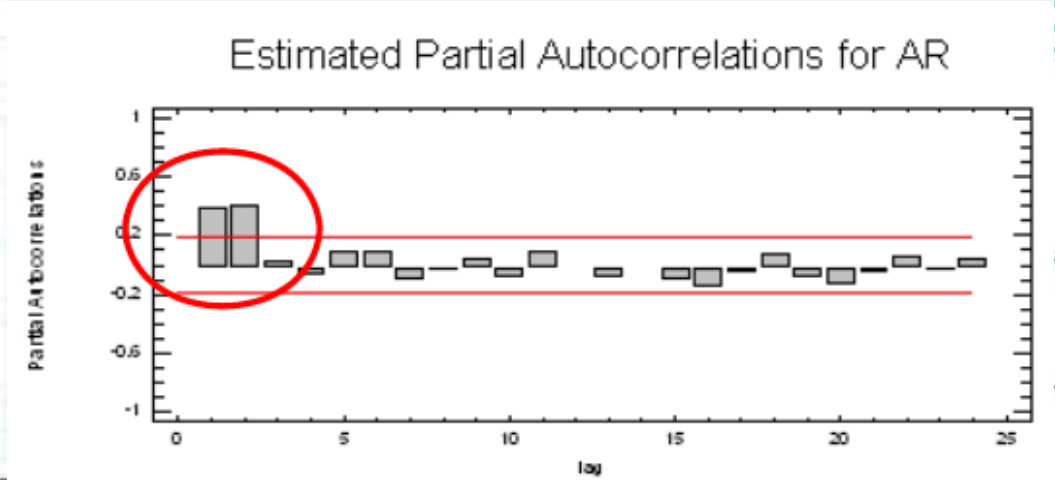
- Корреляция между значениями процесса, с зафиксированным расстоянием по времени между ними
- Частичная автокорреляция (PACF) - “часть корреляции между Y_t и Y_{t-k}, которая не обясняется промежуточными корреляциями”. Коэффициент в AR-модели

Признаки AR модели

- Процесс стремится вернуться к некоторому среднему значению
- ACF убывает плавно, PACF - резко

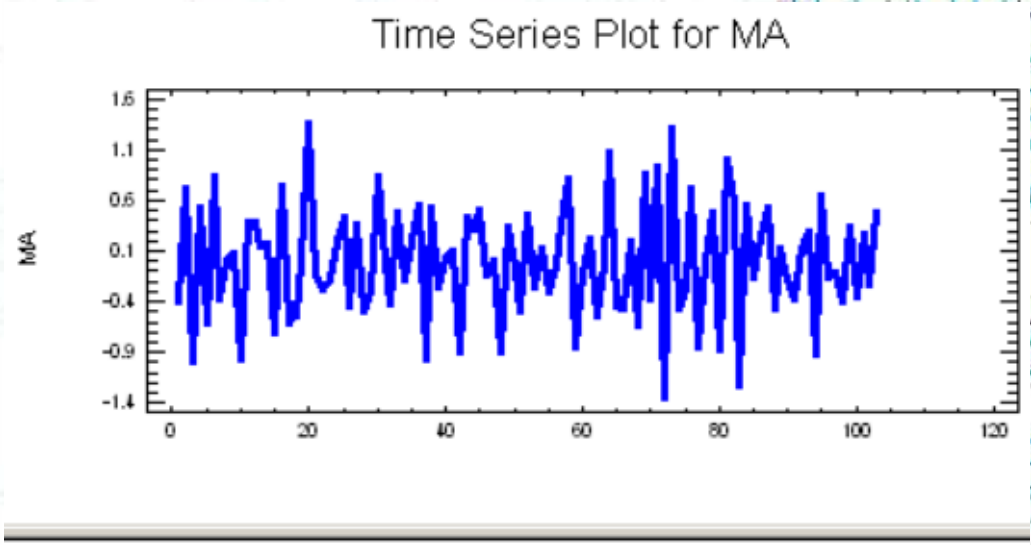


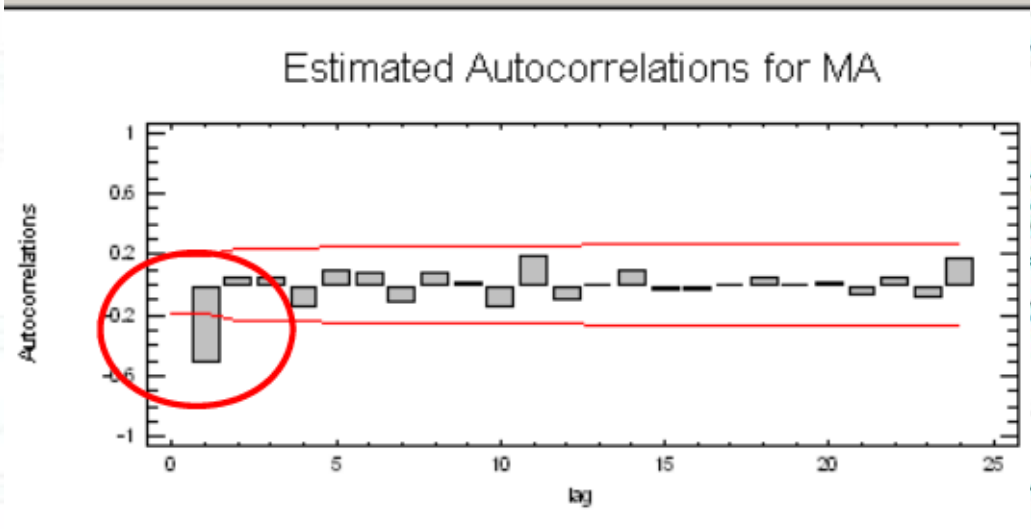


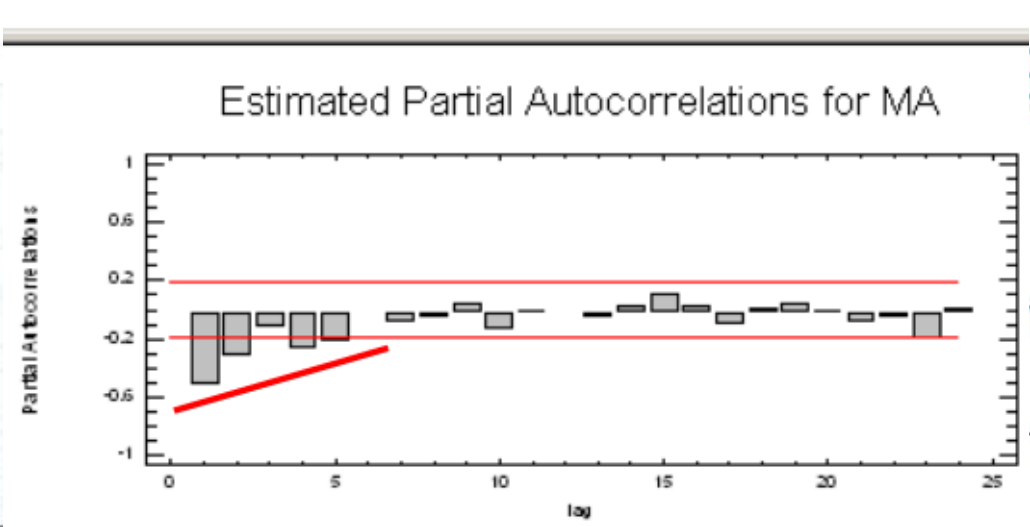


Признаки MA модели

- Похожа на белый шум
- ACF убывает резко, PACF - постепенно



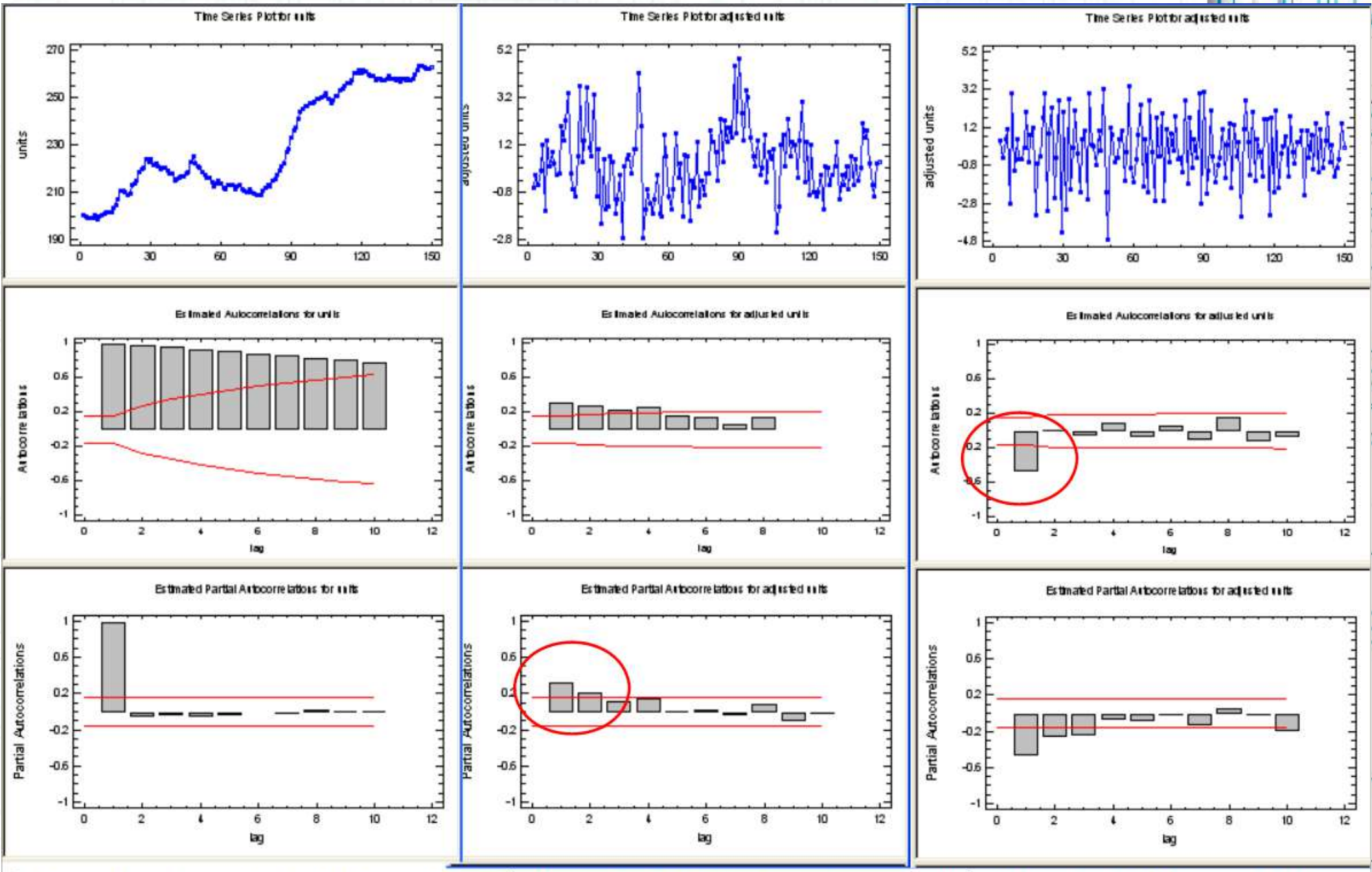




AR или MA

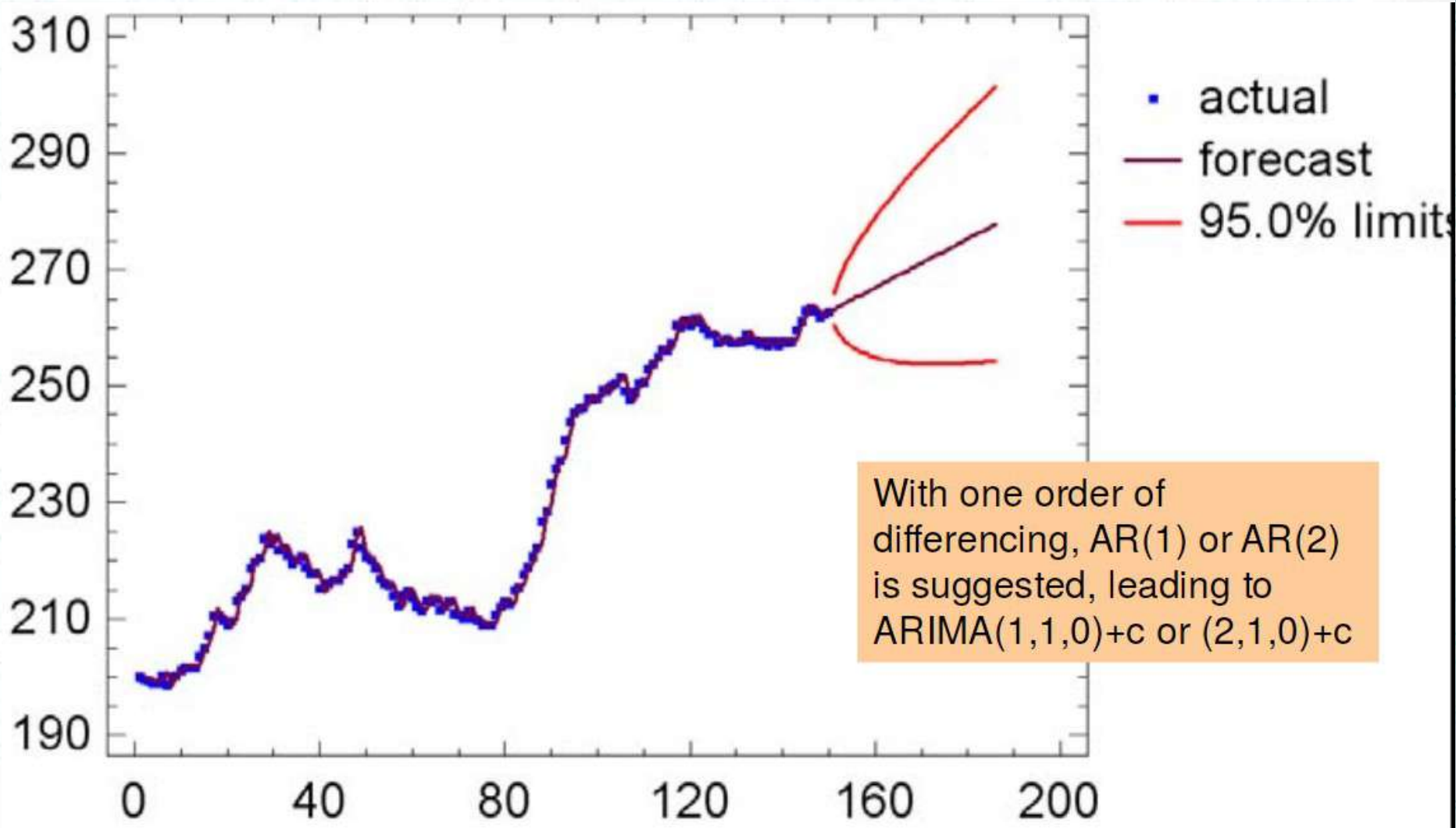
- Все зависит от порядка d дифференцирования процесса
- Исходный процесс обычно похож на AR
- После вычисления нескольких разностей он превращается в MA-процесс
- Не нужно дифференцировать слишком много раз – это переобучение

Пример



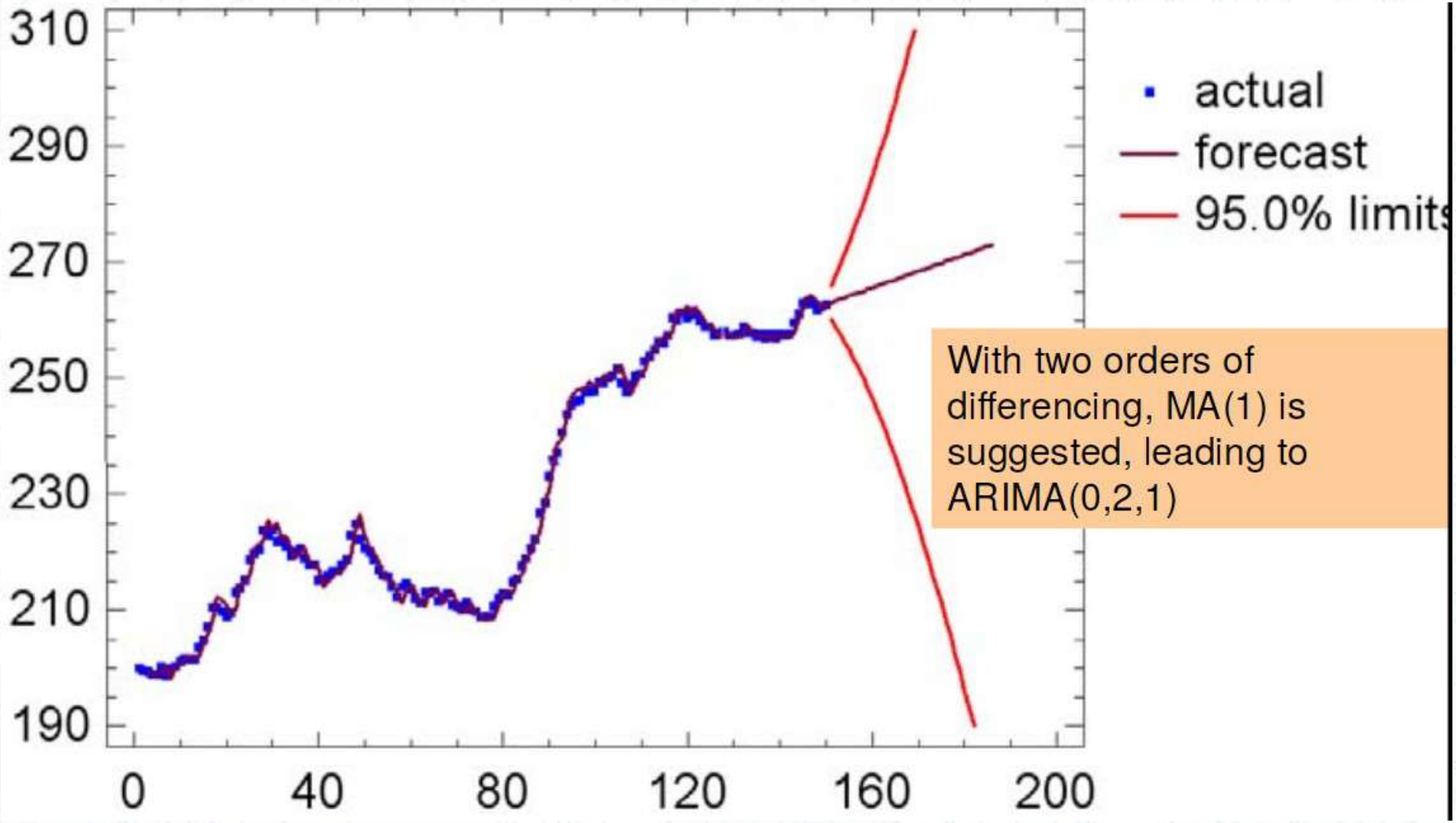
Original series: nonstationary | 1st difference: AR signature | 2nd difference: MA signature

Пример ARIMA(1,1,0)



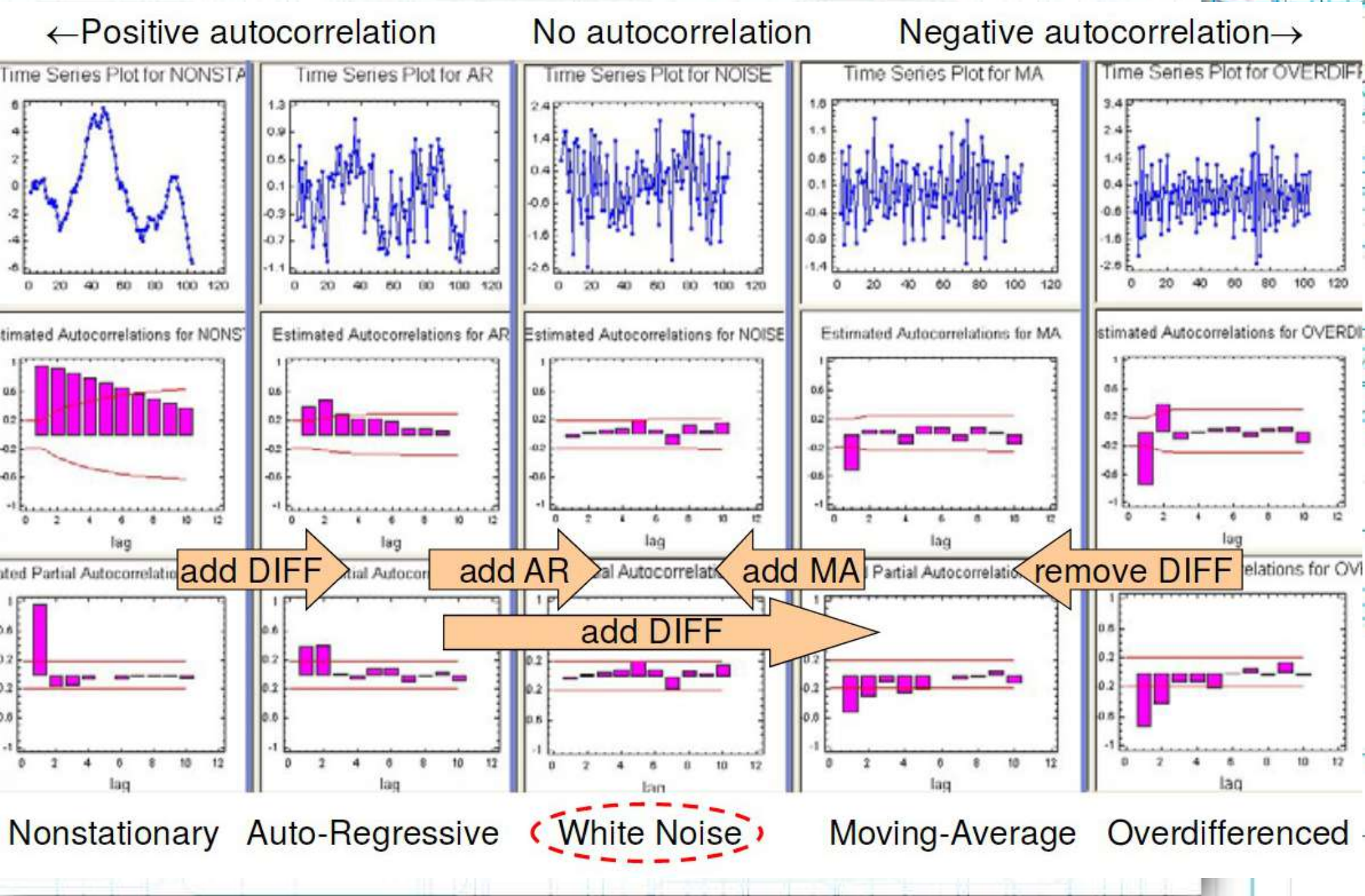
With one order of differencing, AR(1) or AR(2) is suggested, leading to ARIMA(1,1,0)+c or (2,1,0)+c

Пример ARIMA(0,2,1)

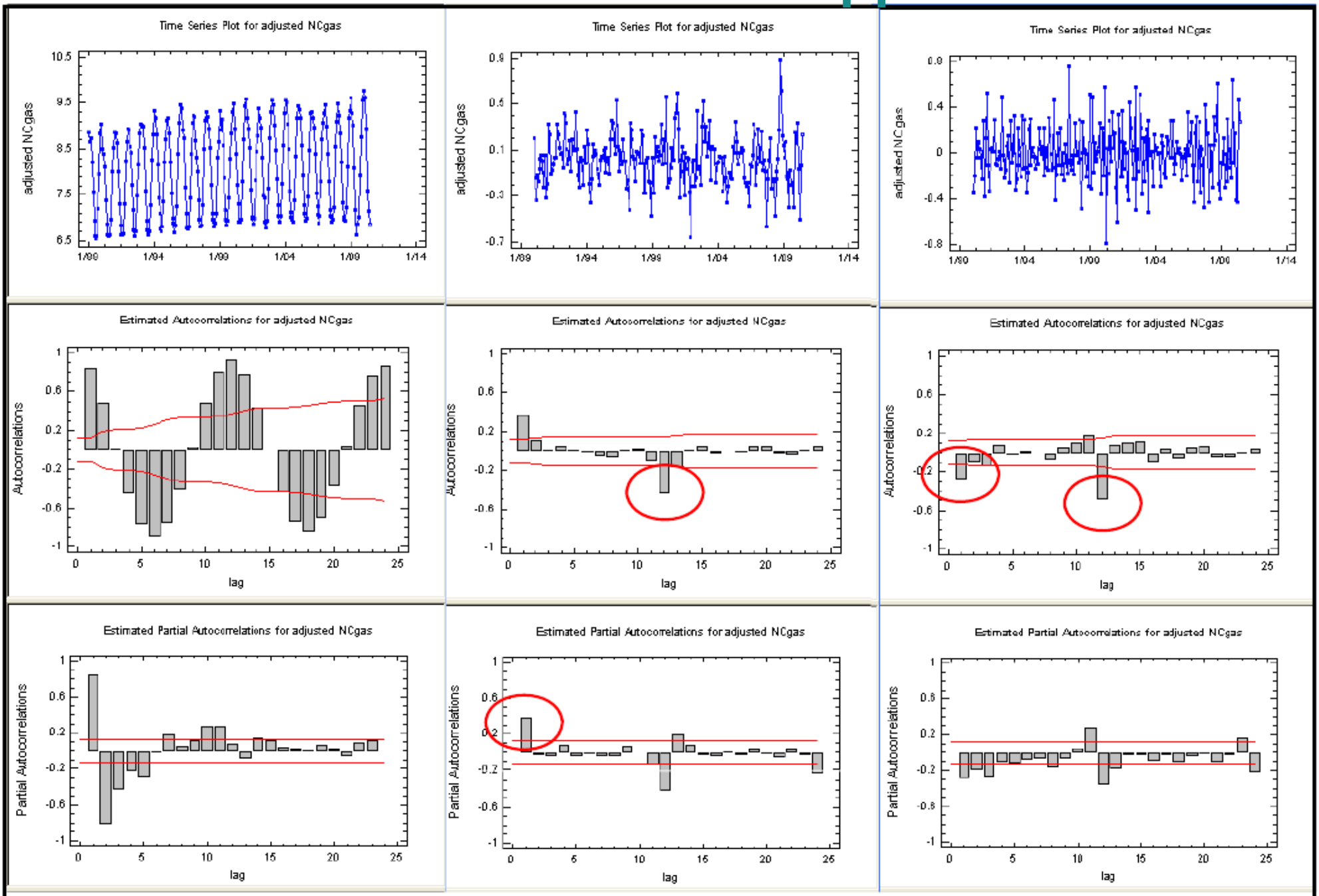


With two orders of differencing, MA(1) is suggested, leading to ARIMA(0,2,1)

Подбор параметров модели



Сезонная ARIMA-модель



Original series: nonstationary Seas. diff: need AR(1) & SMA(1) Both diff: need MA(1) & SMA(1)