AI, МАШИННОЕ И ГЛУБОКОЕ ОБУЧЕНИЕ

Сергей Николенко

ИТМО — Санкт-Петербург

7 сентября 2026 г.

Random facts:

- 7 сентября 1776 г. американская подводная лодка «Черепаха» попыталась установить часовую бомбу на британский флагман «Орёл» в гавани Нью-Йорка
- 7 сентября 1812 г. у села Бородино прошло самое кровопролитное в истории однодневное сражение; итог его считается неопределённым, а потери с каждой из сторон составили около 40000 человек
- 7 сентября 1911 г. Гийом Аполлинер был помещён в тюрьму по подозрению в похищении «Моны Лизы» из Лувра; впрочем, его выпустили уже через пять дней
- 7 сентября 1936 г. в Тасманийском зоопарке умер последний сумчатый волк
- 7 сентября 1947 г. отмечалось 800-летие Москвы, и в честь праздника в столице были одновременно заложены восемь «сталинских высоток»
- 7 сентября 1992 г. «Коммерсантъ» впервые использовал термин «новые русские»
- 7 сентября 1996 г. в Лас-Вегасе было совершено покушение на Тупака Шакура; 13 сентября он умер

О курсе

- Курс о глубоком обучении: что это такое, почему оно работает и что с ним делают сегодня.
- Сегодня — вводная: пройдём весь путь от механических автоматов XIII века до первых нейронных сетей.
- План на сегодня:
  (1) откуда взялась сама идея искусственного интеллекта и что с ней было в XX веке;
  (2) что такое машинное обучение и какие в нём бывают задачи;
  (3) что в мозге такого интересного и как из этого получились искусственные нейронные сети;
  (4) основы байесовского вывода: монетка, линейная и логистическая регрессия.
- А как обучать сети — в следующий раз.

Что такое машинное обучение

Первые мысли об искусственном интеллекте

- Гефест создавал себе роботов-андроидов, например гигантского человекоподобного робота Талоса.
- Пигмалион оживлял Галатею.
- Иегова и Аллах — куски глины.
- Особо мудрые раввины могли создавать големов.
- Альберт Великий изготовил искусственную говорящую голову (чем очень расстроил Фому Аквинского).

Fig. 432.—*The Talking Head of Albertus Magnus.*

Механические автоматы

- Роботы и автоматы вообще были весьма изобретательны: аль-Джазари (XII век)

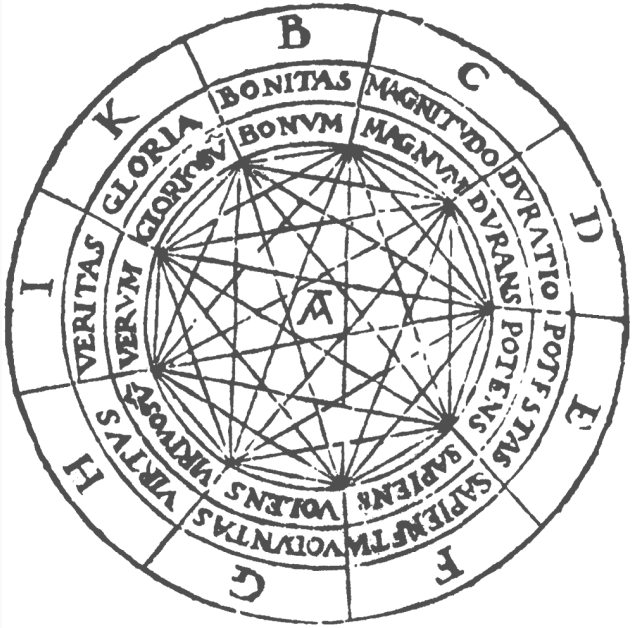
- Замок Эден Роберта II д'Артуа, механический рыцарь Леонардо да Винчи...

Механические автоматы

- Автоматоны Жаке-Дро (XVIII век):

Механические автоматы

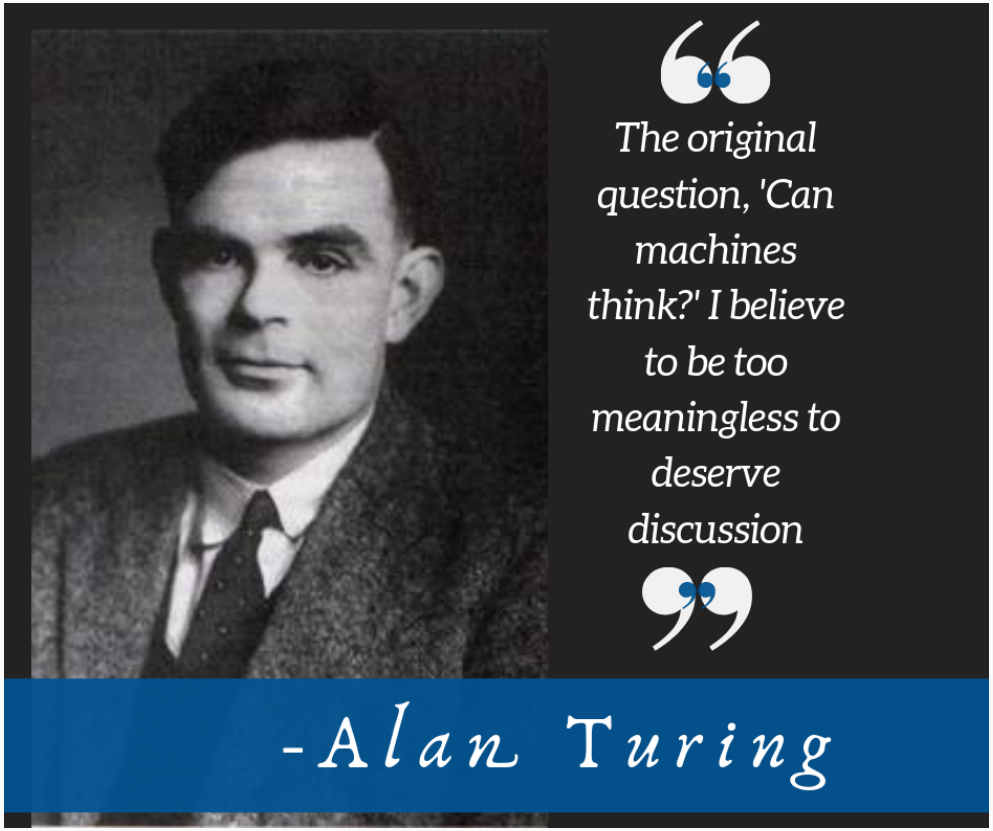
- Но это чистая механика, а в математике/логике AI долго был довольно рудиментарным
- Логическая машина Раймунда Луллия (рубеж XIII-XIV вв.):



- Начиная с доктора Франкенштейна, дальше AI в литературе появляется постоянно...

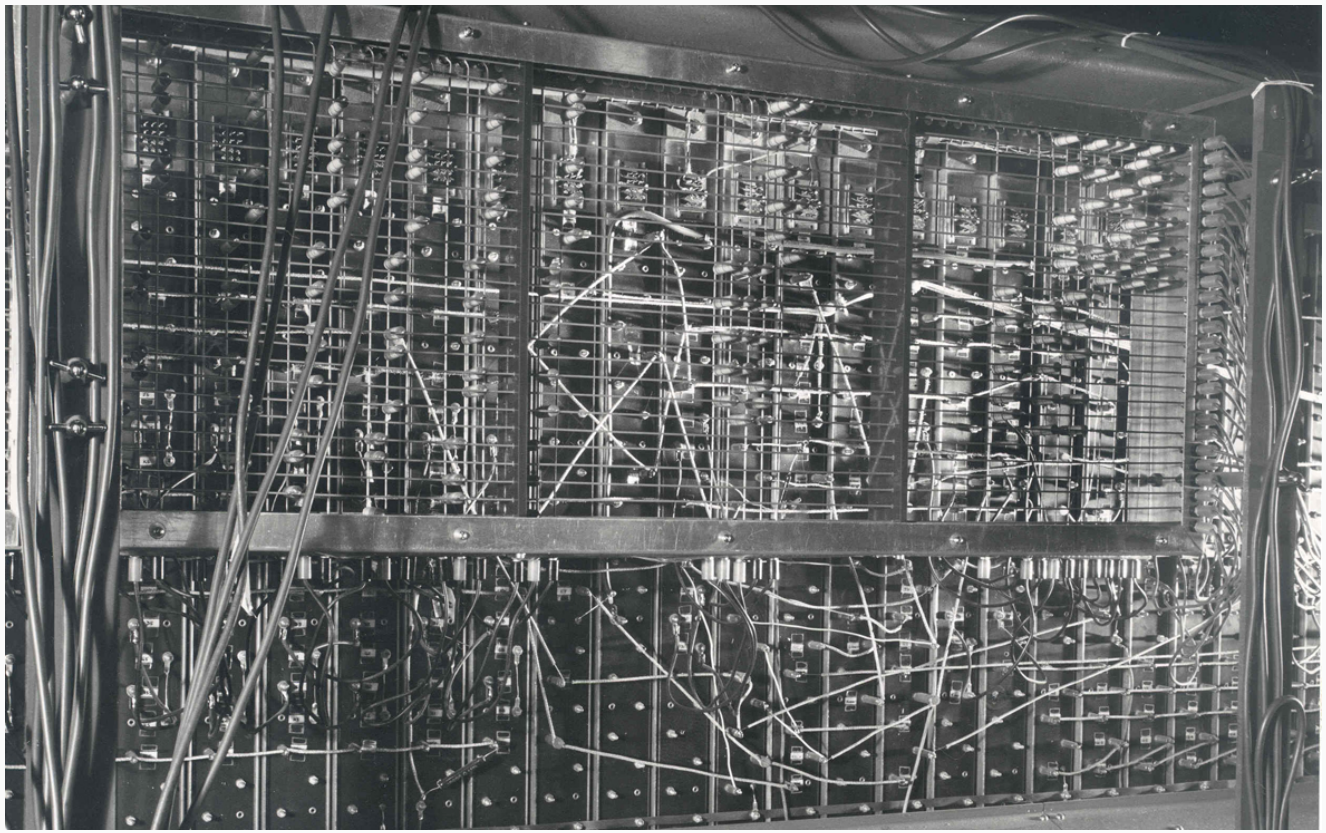
Тест Тьюринга

- AI как наука начался с *теста Тьюринга* (1950).
- Компьютер должен успешно выдать себя за человека в (письменном) диалоге между судьёй, человеком и компьютером. Правда, исходная формулировка была несколько тоньше и интереснее...



Тест Тьюринга

- Здесь уже очевидно, сколько всего надо, чтобы сделать AI:
  - обработка естественного языка;
  - представление знаний;
  - выводы из полученных знаний;
  - обучение на опыте (собственно machine learning).



Дартмутский семинар

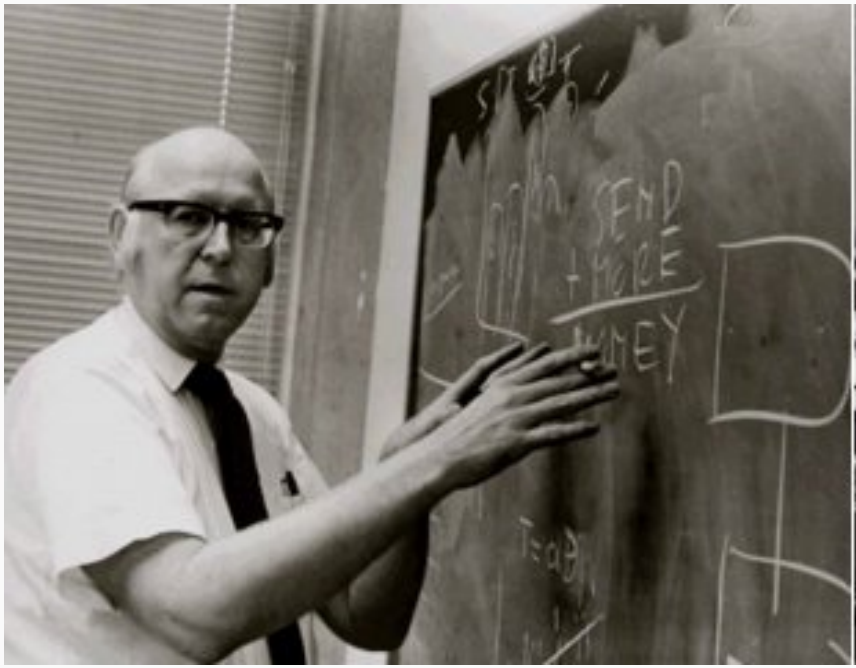
- AI как наука появился в 1956 на семинаре в Дартмуте.
- Его организовали Джон Маккарти (John McCarthy), Марвин Мински (Marvin Minsky), Клод Шеннон (Claude Shennon) и Натаниэль Рочестер (Nathaniel Rochester).
- Это была, наверное, самая амбициозная грантозаявка в истории информатики.

Дартмутский семинар

Мы предлагаем исследование искусственного интеллекта сроком в 2 месяца с участием 10 человек летом 1956 года в Дарттмутском колледже, Гановер, Нью–Гемпшир. Исследование основано на предположении, что всякий аспект обучения или любое другое свойство интеллекта может в принципе быть столь точно описано, что машина сможет его симулировать. Мы попытаемся понять, как обучить машины использовать естественные языки, формировать абстракции и концепции, решать задачи, сейчас подвластные только людям, и улучшать самих себя. Мы считаем, что существенное продвижение в одной или более из этих проблем вполне возможно, если специально подобранная группа учёных будет работать над этим в течение лета.

1956-1960: большие надежды

- Оптимистическое время. Казалось, что ещё немного, ещё чуть-чуть...
- Allen Newell, Herbert Simon: *Logic Theorist*.
  - Программа для логического вывода.
  - Смогла передоказать большую часть *Principia Mathematica*, кое-где даже изящнее, чем сами Рассел с Уайтхедом.

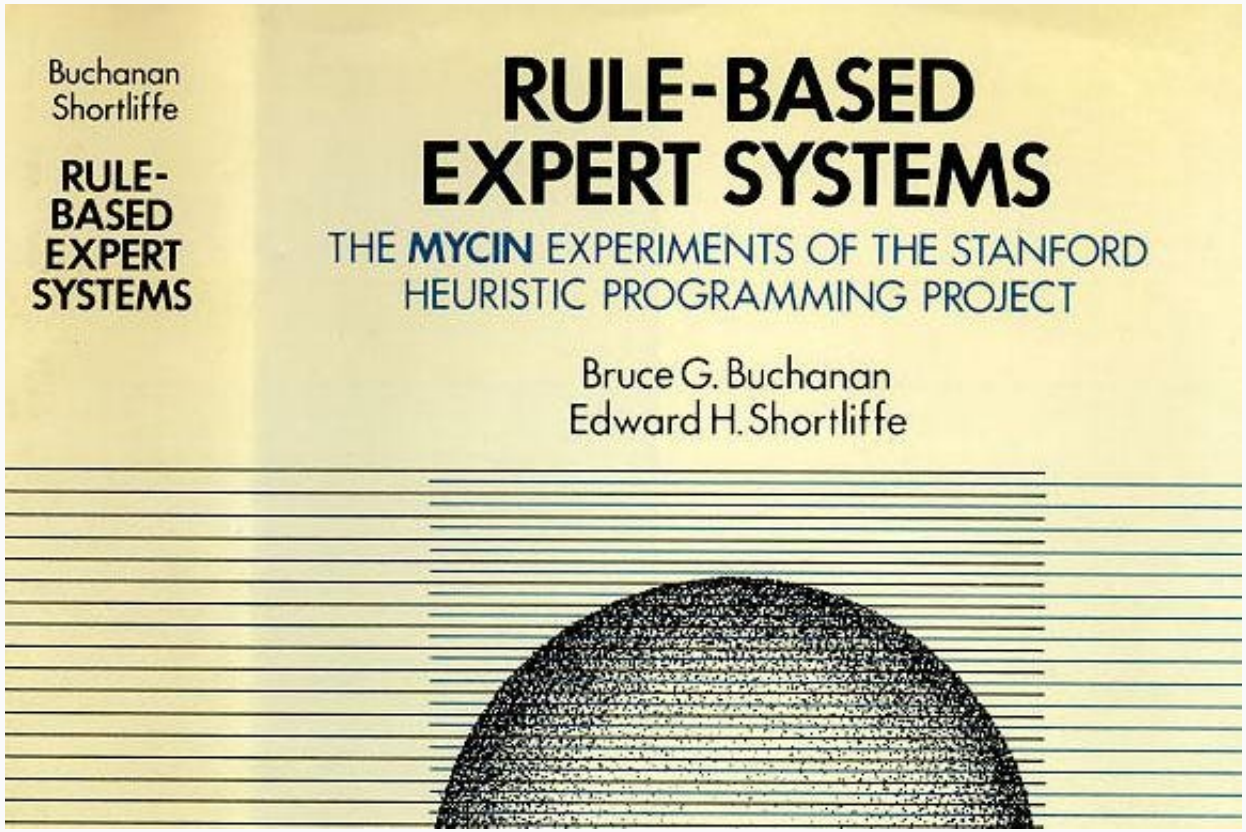


1956-1960: Большие надежды

- Оптимистическое время. Казалось, что ещё немного, ещё чуть-чуть...
- General Problem Solver – программа, которая пыталась думать как человек;
- Много программ, которые умели делать некоторые ограниченные вещи (microworlds):
  - Analogy (IQ-тесты на «выберите лишнее»);
  - Student (алгебраические словесные задачи);
  - Blocks World (переставляла 3D-блоки).

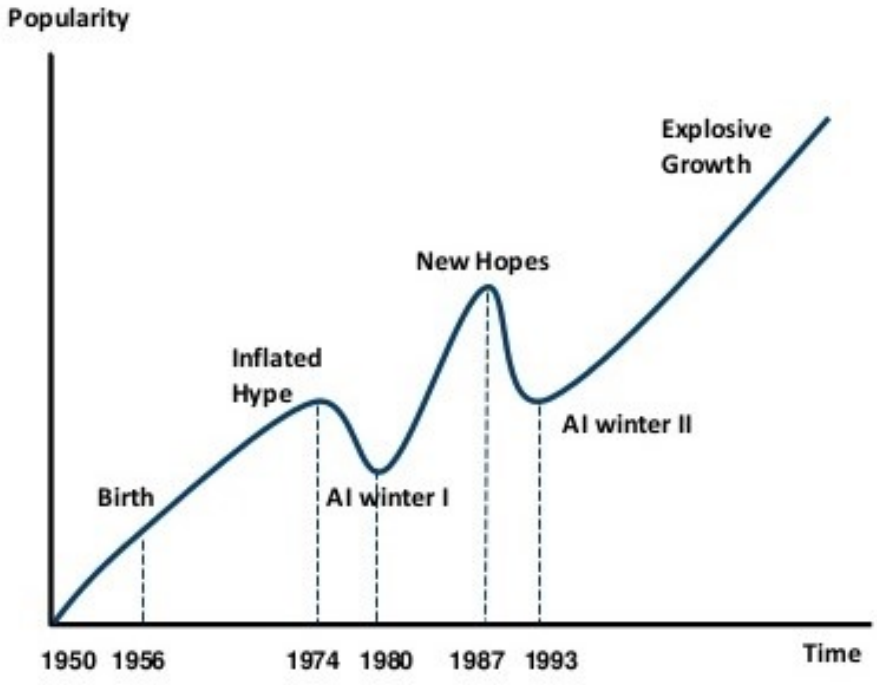
1970-е: knowledge-based systems

- Суть: накопить достаточно большой набор правил и знаний о предметной области, затем делать выводы.
- Первый успех: MYCIN – диагностика инфекций крови:
  - около 450 правил;
  - результаты как у опытного врача и существенно лучше, чем у начинающих врачей.



1980-е: коммерческие применения; индустрия AI

- Первый AI-отдел был в компании DEC (Digital Equipment Corporation); утверждают, что к 1986 году он экономил DEC около $10 млн. в год.
- Бум закончился к концу 80-х, когда многие компании не смогли оправдать завышенных ожиданий.



Timeline of AI Development

- **1950s-1960s**: First AI boom - the age of reasoning, prototype AI developed
- **1970s**: AI winter I
- **1980s-1990s**: Second AI boom: the age of Knowledge representation (appearance of expert systems capable of reproducing human decision-making)
- **1990s**: AI winter II
- **1997**: Deep Blue beats Gary Kasparov
- **2006**: University of Toronto develops Deep Learning
- **2011**: IBM's Watson won Jeopardy
- **2016**: Go software based on Deep Learning beats world's champions

1990-2010: DATA MINING, MACHINE LEARNING

- В последние десятилетия основной акцент сместился на машинное обучение и поиск закономерностей в данных.
- Особенно — с развитием интернета.
- Сейчас про AI в смысле трёх законов робототехники уже не очень вспоминают.
- // Но роботика — процветает и пользуется machine learning на каждом шагу.

Определение

- Что значит — обучающаяся машина? Как определить «обучаемость»?

Определение

- Что значит — обучающаяся машина? Как определить «обучаемость»?

Definition

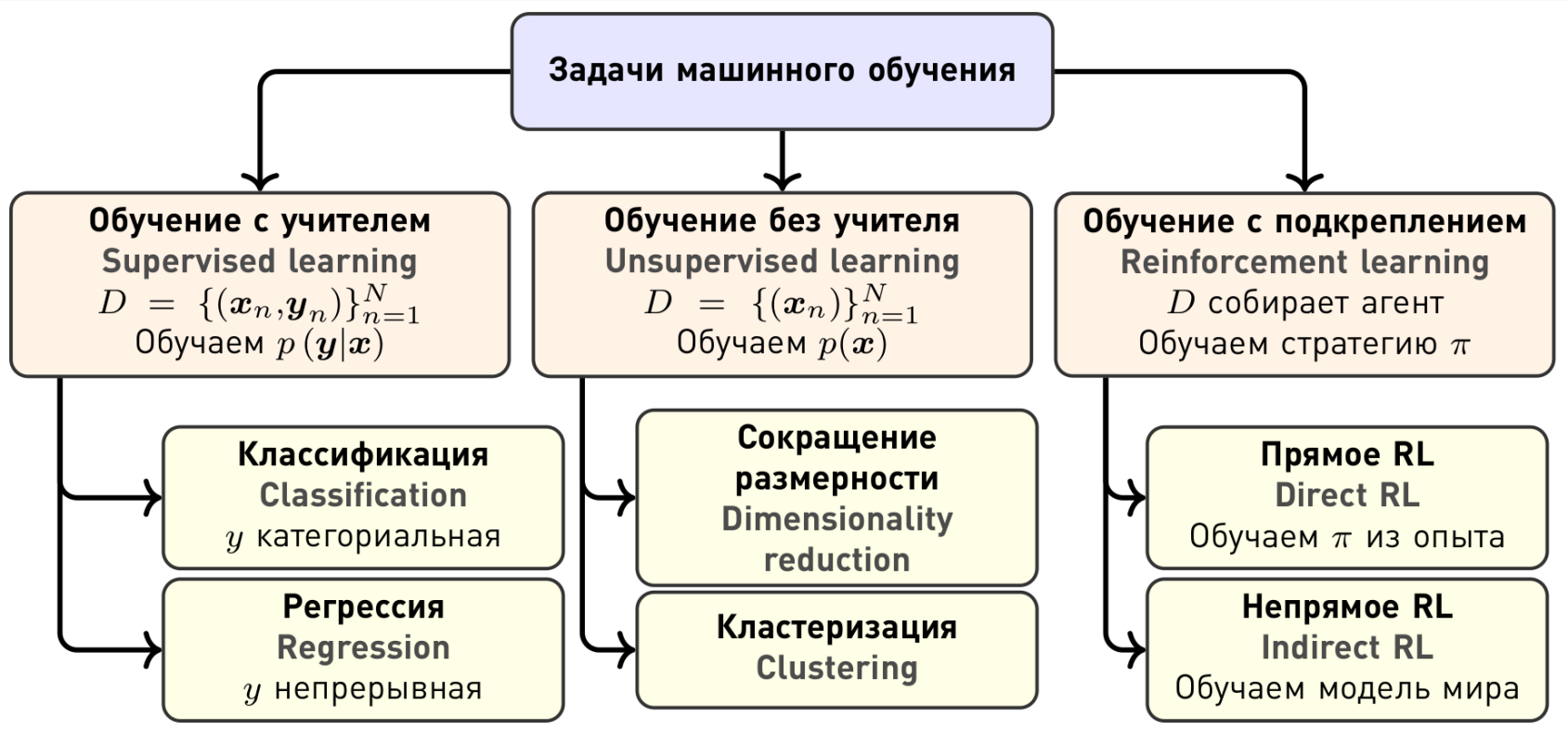
Компьютерная программа обучается по мере накопления опыта относительно некоторого класса задач T и целевой функции P, если качество решения этих задач (относительно P) улучшается с получением нового опыта.

- Определение очень (слишком?) общее.
- Какие конкретные примеры можно привести?

Чем мы будем заниматься

- Мы будем рассматривать разные алгоритмы, которые решают ту ли иную задачу, причём решают тем лучше, чем больше начальных (тестовых) данных ему дадут.
- Сегодня мы поговорим об общей теории байесовского вывода, в которую обычно можно погрузить любой алгоритм машинного обучения.
- Но сначала – краткий обзор основных задач машинного обучения в целом.

Основные задачи и понятия машинного обучения



Основные задачи и понятия машинного обучения

- *Обучение с учителем* (supervised learning) – обучение, в котором есть некоторое число примеров с правильными ответами:
  - *обучающая выборка* (training set) – набор примеров, каждый из которых состоит из *признаков* (features, attributes);
  - у примеров есть правильные ответы – переменная (response), которую мы предсказываем; она может быть категориальная (categorical), непрерывная или ординальная (ordinal);

Основные задачи и понятия машинного обучения

- *Обучение с учителем* (supervised learning) – обучение, в котором есть некоторое число примеров с правильными ответами:
  - модель *обучается* на этой выборке (training phase, learning phase), затем может быть применена к новым примерам (test set);
  - главное – обучить модель, которая не только точки из обучающей выборки объясняет, но и на новые примеры хорошо *обобщается* (generalizes);
  - иначе – оверфиттинг (overfitting);

Основные задачи и понятия машинного обучения

- *Обучение с учителем* (supervised learning) – обучение, в котором есть некоторое число примеров с правильными ответами:
  - обычно нам дают просто обучающую выборку – как тогда проверить, обобщаются ли модели?
  - кросс-валидация – разбиваем выборку на тренировочный и валидационный набор (validation set);
  - перед тем как подавать что-то на вход, обычно делают предобработку, стараясь выделить из входных данных самые содержательные аспекты (feature extraction).

Основные задачи и понятия машинного обучения

- *Обучение с учителем* (supervised learning) – обучение, в котором есть некоторое число примеров с правильными ответами:
  - *классификация*: есть некоторый дискретный набор категорий (классов), и надо новые примеры определить в какой-нибудь класс;
    - классификация текстов по темам, спам-фильтр;
    - распознавание лиц/объектов/текста;

Основные задачи и понятия машинного обучения

- *Обучение с учителем* (supervised learning) – обучение, в котором есть некоторое число примеров с правильными ответами:
  - *регрессия*: есть некоторая неизвестная функция, и надо предсказать её значения на новых примерах:
    - инженерные приложения (предсказать температуру, положение робота, whatever);
    - финансы – предсказать цену акций;
  - то же плюс изменения во времени – например, распознавание речи.

Основные задачи и понятия машинного обучения

- *Обучение без учителя* (unsupervised learning) – обучение, в котором нет правильных ответов, только данные:
  - *кластеризация* (clustering): надо разбить данные на заранее неизвестные классы по некоторой мере похожести:
    - выделить семейства генов из последовательностей нуклеотидов;
    - кластеризовать пользователей и персонализовать под них приложение;
    - кластеризовать масс-спектрометрическое изображение на части с разным составом;

Основные задачи и понятия машинного обучения

- *Обучение без учителя* (unsupervised learning) – обучение, в котором нет правильных ответов, только данные:
  - *снижение размерности* (dimensionality reduction): данные имеют огромную размерность (очень много признаков), нужно уменьшить её, выделить самые информативные признаки, чтобы все вышеописанные алгоритмы смогли работать;
  - *дополнение матриц* (matrix completion): есть разреженная матрица, надо предсказать, что на недостающих позициях.
  - Часто даны правильные ответы для небольшой части данных – semi-supervised learning.

Основные задачи и понятия машинного обучения

- *Обучение с подкреплением* (reinforcement learning) – обучение, в котором агент учится из собственных проб и ошибок:
  - *многорукие бандиты*: есть некоторый набор действий, каждое из которых ведёт к случайным результатам; нужно получить как можно больший доход;
  - *exploration vs. exploitation*: как и когда от исследования нового переходить к использованию того, что уже изучил;
  - *credit assignment*: конфетку дают в самом конце (выиграл партию), и надо как-то распределить эту конфетку по всем ходам, которые привели к победе.

Основные задачи и понятия машинного обучения

- *активное обучение* (active learning) – как выбрать следующий (относительно дорогой) тест;
- *обучение ранжированию* (learning to rank) – ординальная регрессия, как породить упорядоченный список (интернет-поиск);
- *бустинг* (boosting) – как скомбинировать несколько слабых классификаторов так, чтобы получился хороший;
- *выбор модели* (model selection) – где провести черту между моделями с многими параметрами и с немногими.

Вероятность в машинном обучении

- Во всех методах и подходах очень пригодится метод, который мог бы не просто выдавать ответ, а ещё оценивать, насколько модель уверена в этом ответе, насколько модель хорошо описывает данные, как изменятся эти величины при дальнейших экспериментах и т.д.
- Поэтому центральную роль в машинном обучении играет теория вероятностей – и мы тоже будем её активно применять.

Источники

- Christopher M. Bishop, *Pattern Recognition and Machine Learning*, Springer, 2007.
- Kevin Murphy, *Machine Learning: A Probabilistic Perspective*, MIT Press, 2013.
- Trevor Hastie, Robert Tibshirani, and Jerome Friedman, *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*, 2nd ed., Springer, 2009.

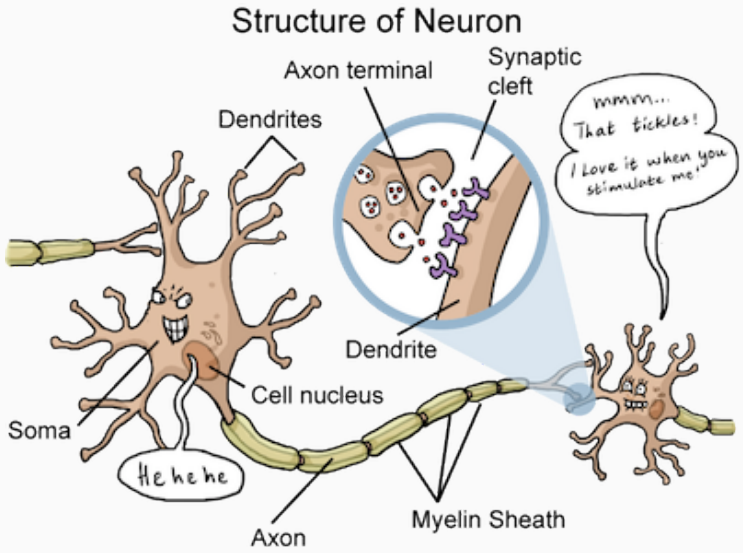
Человеческий мозг и история искусственных нейронных сетей

Компьютер у нас в голове

- Компьютер считает быстрее человека, но гораздо хуже
  - понимает естественный язык,
  - распознаёт изображения, узнаёт людей (это уже не совсем так),
  - обучается в широком смысле этого слова и т.д...
- Почему так? Как человек всего этого добивается?

Компьютер у нас в голове

- В мозге много нейронов; каждый нейрон:
  - через дендриты получает сигнал от других нейронов;
  - время от времени запускает сигнал по аксону;
  - через синапсы сигнал аксона доходит до дендритов других нейронов.

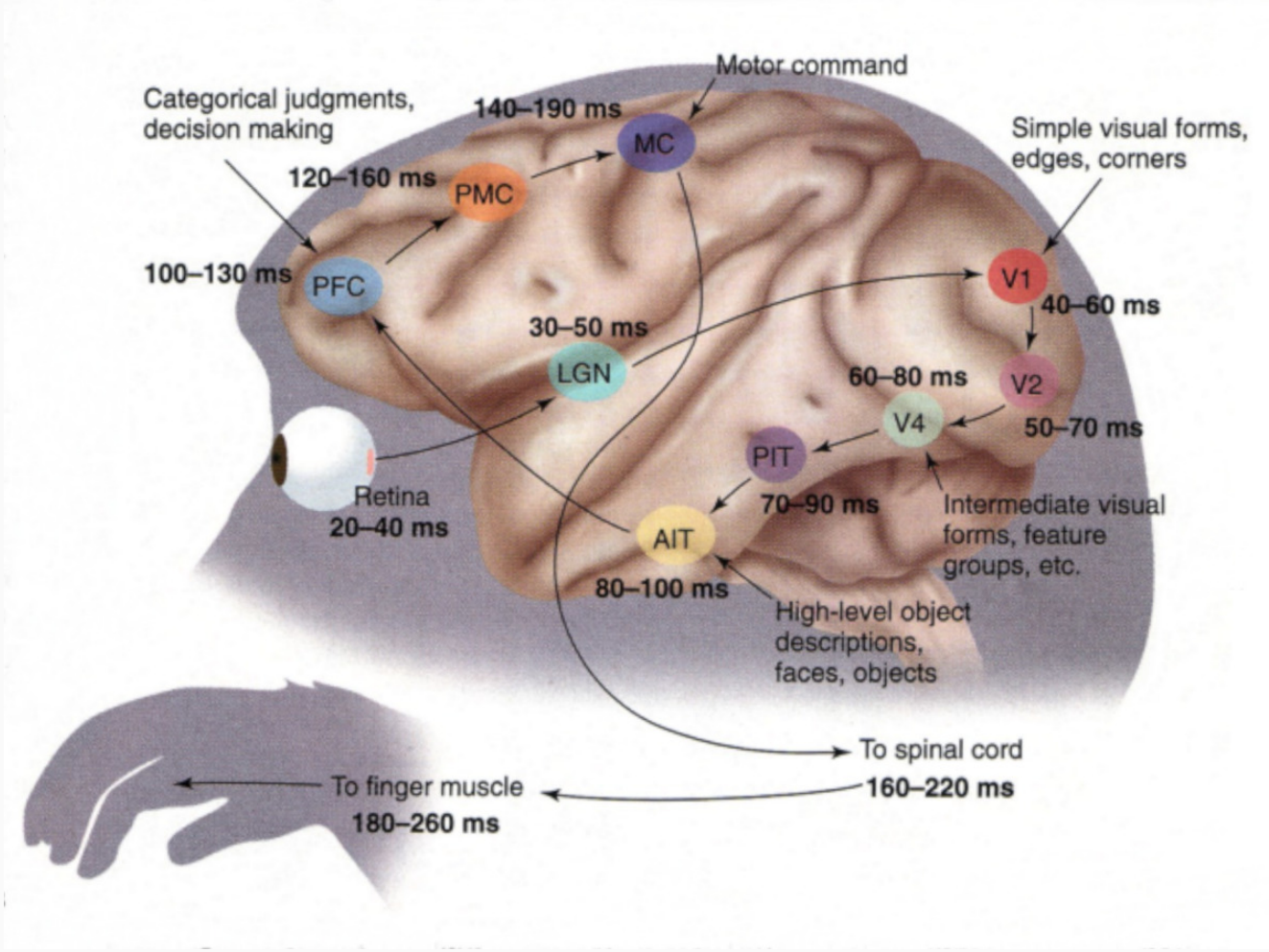


Компьютер у нас в голове

- Нейрон время от времени, стохастически, выдаёт сигналы.
- Всего нейронов очень много: 10^{11} нейронов, в среднем 7000 связей у каждого, т.е. 10^{15} синапсов.
- Но firing rate от 10 до 200 герц.
- А распознаём мы лицо за пару сотен миллисекунд.
- Это значит, что цепочки короткие, и вычисления скорее параллельные, чем последовательные.

Компьютер у нас в голове

- Вот как мы обрабатываем картинку:



Обучение признаков

- Другая сторона вопроса – обучение признаков.
- Мозг очень хорошо умеет обучаться на очень-очень маленькой выборке данных.
- И может адаптироваться к новым источникам информации.
- Как он это делает? Можем ли мы сделать так же?

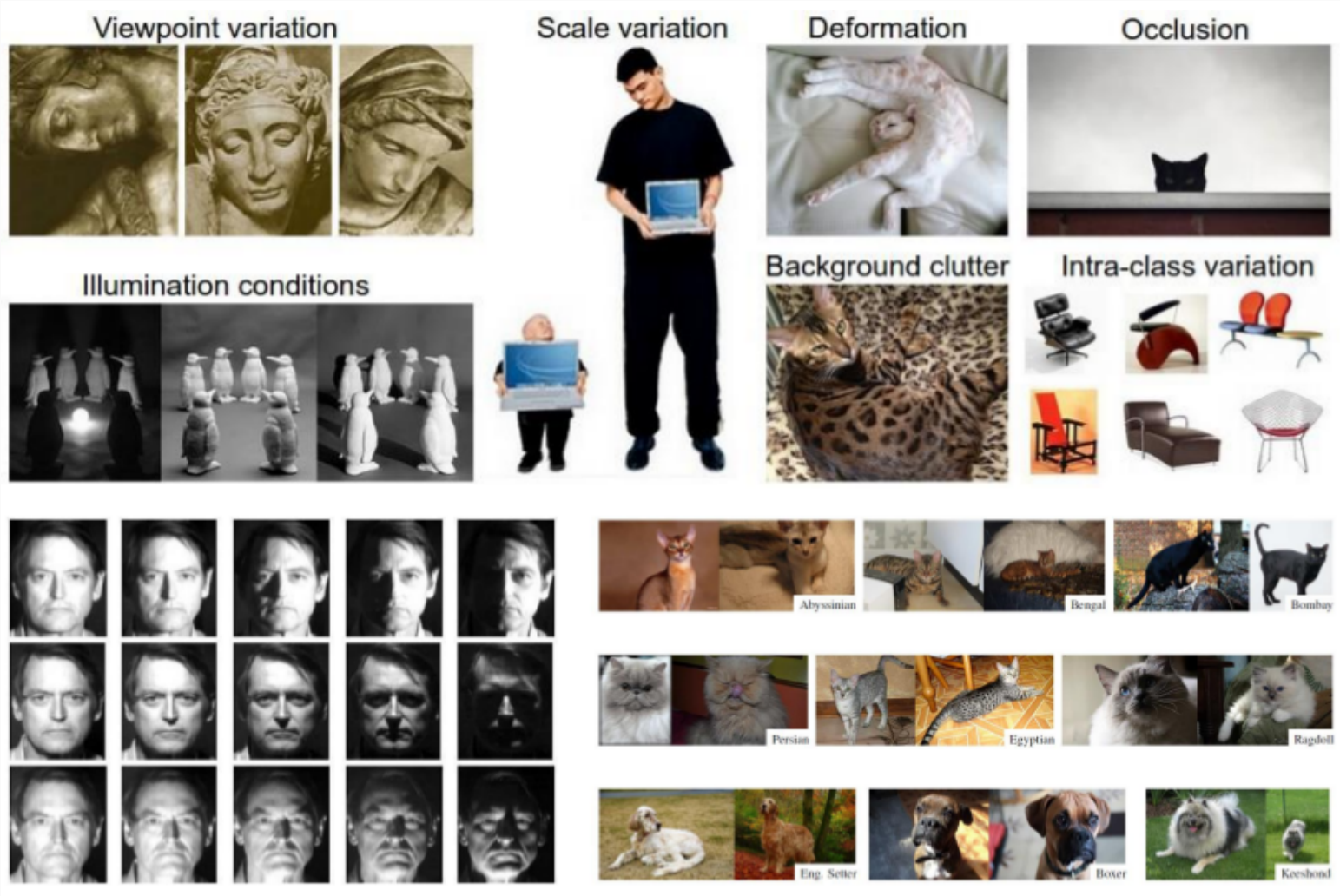
Обучение признаков

- Системы обработки неструктурированной информации выглядят обычно так:

$$\text{вход} \to \text{признаки} \to \text{классификатор}$$

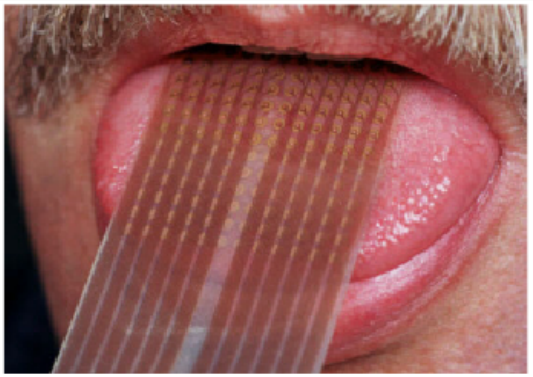
- Люди много десятилетий пытались придумать хорошие признаки:
  - MFCC для распознавания речи;
  - SIFT для обработки изображений;
  - ...
- Задача feature engineering: как сделать такие признаки?
- Но, может быть, можно найти признаки автоматически? Мозг ведь это как-то делает...
- Но это сложно!

Почему это сложно: распознавание изображений



Пластичность

- Третий аспект — пластичность мозга.



- Она показывает, что есть единый алгоритм обучения, который можно применять к самым разным ситуациям.

Предостережение

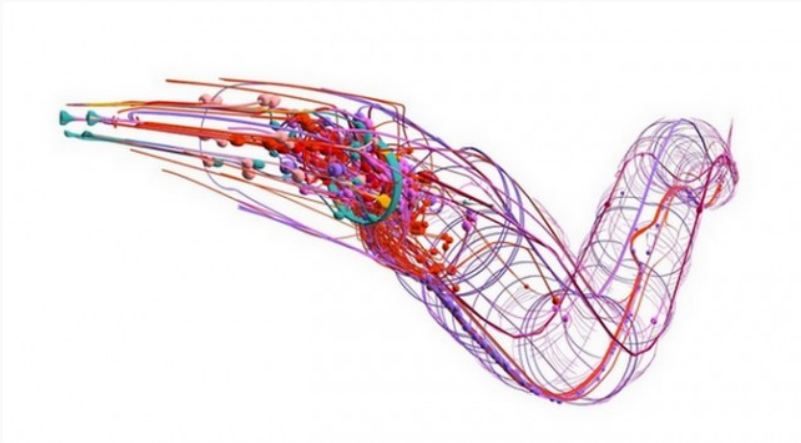
- Мы ещё не раз вернёмся к тому, как работает настоящий мозг и настоящие нейроны.
- Но лучше не слишком увлекаться:
  - 2004 – может ли биолог починить радиоприёмник?

  - 2016 – может ли нейробиолог понять микропроцессор?

Предостережение

- Мало что понимаем в целых нервных системах:
  - Human Brain Project Генри Маркрама провалился;

  - а получается пока OpenWorm
    (нематода *C. elegans*, 302 нейрона).



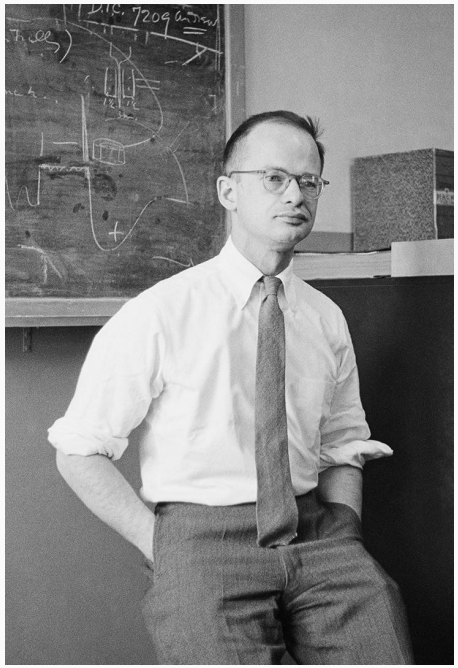
- Так что будем вдохновляться, но без фанатизма.

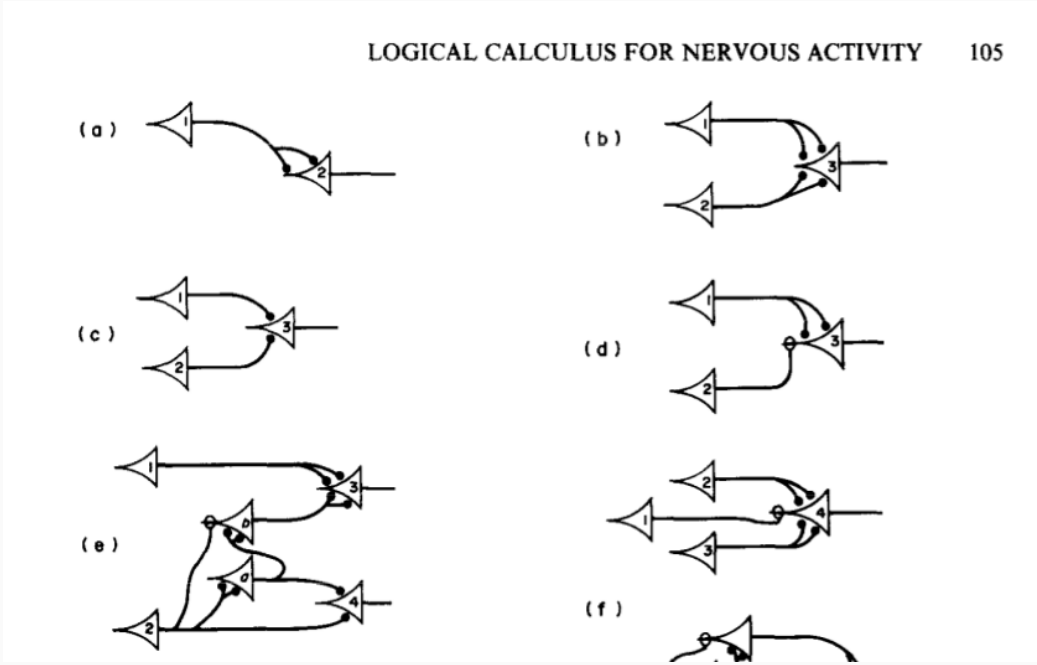
Суть происходящего

- Вообще говоря, всё, что мы делаем в машинном обучении, — это аппроксимация и оптимизация функций.
- Например:
  - есть функция из картинок в то, что на них изображено;
  - ещё проще: бинарная функция из пикселей картинки в «котик или не котик»;
  - функция, мягко скажем, довольно сложная; задана в виде датасета; как нам её реализовать?
- Обычный подход в машинном обучении:
  - строим какой-то класс моделей, обычно параметрический;
  - и подгоняем параметры так, чтобы модель хорошо описывала данные;
  - то есть оптимизируем какую-то функцию ошибки или функцию качества (правдоподобие, апостериорную вероятность).
- Нейронные сети — это очень мощный и гибкий класс таких моделей.
- Сложность в том, как же обучить их параметры.

История ANN и перцептрон

- Давайте попробуем как в природе: есть связанные между собой нейроны, которые передают друг другу сигналы.
- McCulloch, Pitts, 1943: идея.

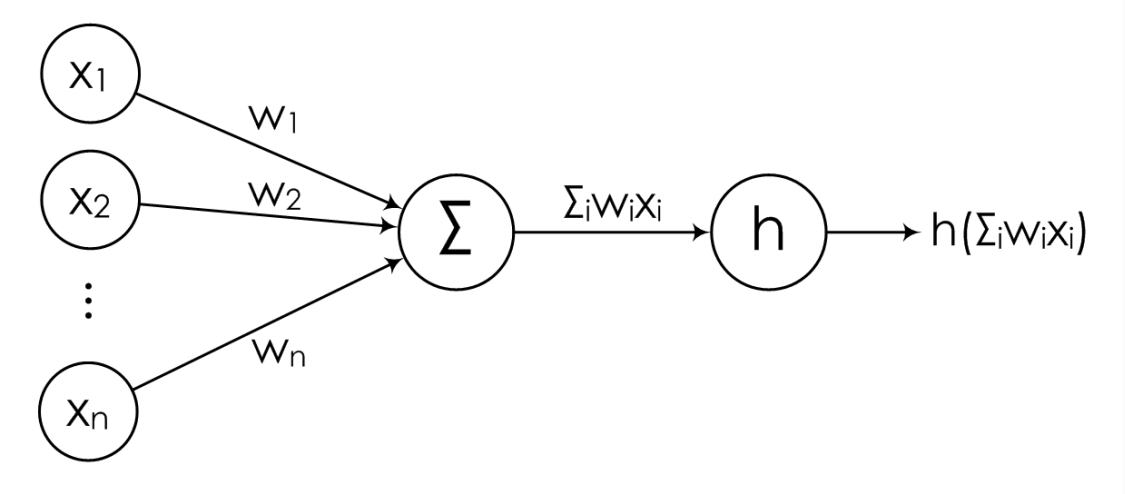




- Идею искусственных сетей, похожих на современные (даже с несколькими уровнями), предлагал ещё Алан Тьюринг (1948).
- Rosenblatt, 1958: перцептрон (один искусственный нейрон).
- Тогда же появился алгоритм обучения градиентным спуском.

История ANN и перцептрон

- Один нейрон обычно моделируется вот так:



- Линейная комбинация входов, потом нелинейность:

$$y = h(\mathbf{w}^\top \mathbf{x}) = h\left(\sum_i w_i x_i\right).$$

- Как обучить перцептрон, и каков будет результат?

История ANN и перцептрон

- Обучение *градиентным спуском*.
- Для исходного перцептрона Розенблатта ($h = \mathrm{id}$):

$$E_P(\mathbf{w}) = -\sum_{\mathbf{x}\in\mathcal{M}} y(\mathbf{x})\left(\mathbf{w}^\top \mathbf{x}\right),$$

$$\mathbf{w}^{(\tau+1)} = \mathbf{w}^{(\tau)} - \eta\nabla_{\mathbf{w}} E_P(\mathbf{w}) = \mathbf{w}^{(\tau)} + \eta t_n \mathbf{x}_n.$$

- Или, к примеру, можно делать классификацию, подставляя в качестве функции активации логистический сигмоид $h(x) = \sigma(x) = \frac{1}{1+e^{-x}}$:

$$E(\mathbf{w}) = -\frac{1}{N}\sum_{i=1}^{N}\left(y_i \log\sigma(\mathbf{w}^\top \mathbf{x}_i) + (1-y_i)\log\left(1-\sigma(\mathbf{w}^\top \mathbf{x}_i)\right)\right).$$

- По сути это просто логистическая регрессия.

История ANN и перцептрон

- Первый перцептрон (Rosenblatt, 1958) распознавал цифры.

История ANN и перцептрон

- И люди в него верили:

NEW NAVY DEVICE LEARNS BY DOING

Psychologist Shows Embryo of Computer Designed to Read and Grow Wiser

WASHINGTON, July 7 (UPI) —The Navy revealed the embryo of an electronic computer today that it expects will be able to walk, talk, see, write, reproduce itself and be conscious of its existence.

The embryo—the Weather Bureau's $2,000,000 "704" computer—learned to differentiate between right and left after fifty attempts in the Navy's demonstration for newsmen.

The service said it would use this principle to build the first of its Perceptron thinking machines that will be able to read and write. It is expected to be finished in about a year at a cost of $100,000.

Dr. Frank Rosenblatt, designer of the Perceptron, conducted the demonstration. He said the machine would be the first device to think as the human brain. As do human beings, Perceptron will make mistakes at first, but will grow wiser as it gains experience, he said.

Dr. Rosenblatt, a research psychologist at the Cornell Aeronautical Laboratory, Buffalo, said Perceptrons might be fired to the planets as mechanical space explorers.

Without Human Controls

The Navy said the perceptron would be the first non-living mechanism "capable of receiving, recognizing and identifying its surroundings without any human training or control."

The "brain" is designed to remember images and information it has perceived itself. Ordinary computers remember only what is fed into them on punch cards or magnetic tape.

Later Perceptrons will be able to recognize people and call out their names and instantly translate speech in one language to speech or writing in another language, it was predicted.

Mr. Rosenblatt said in principle it would be possible to build brains that could reproduce themselves on an assembly line and which would be conscious of their existence.

1958 New York Times...

In today's demonstration, the "704" was fed two cards, one with squares marked on the left side and the other with squares on the right side.

Learns by Doing

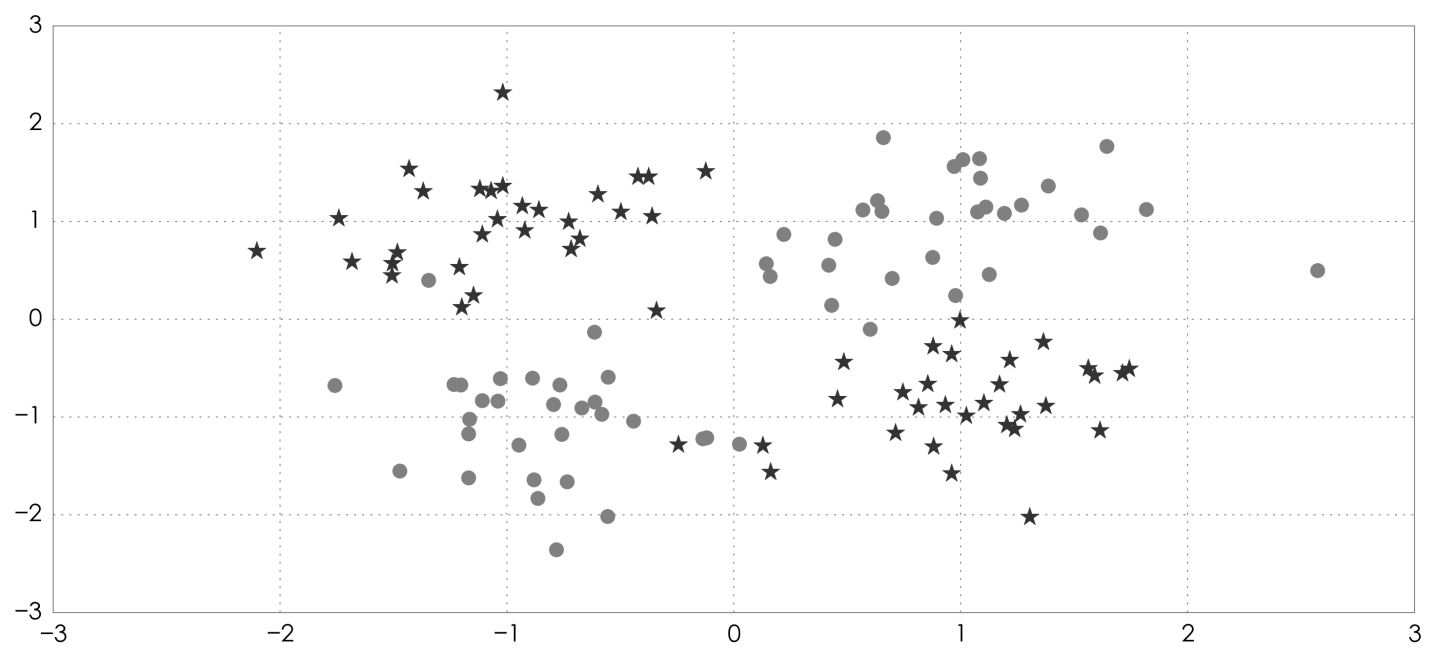
In the first fifty trials, the machine made no distinction between them. It then started registering a "Q" for the left squares and "O" for the right squares.

Dr. Rosenblatt said he could explain why the machine learned only in highly technical terms. But he said the computer had undergone a "self-induced change in the wiring diagram."

The first Perceptron will have about 1,000 electronic "association cells" receiving electrical impulses from an eye-like scanning device with 400 photo-cells. The human brain has 10,000,000,000 responsive cells, including 100,000,000 connections with the eyes.

История ANN и перцептрон

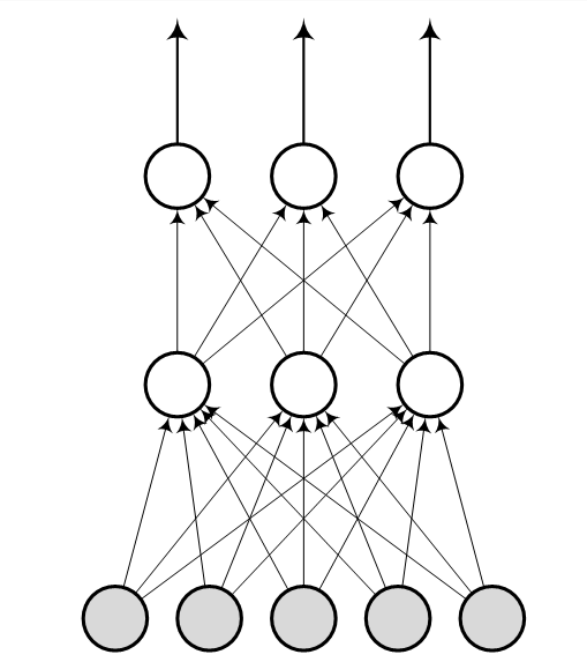
- 1960-е годы: изучали перцептроны.
- (Minsky, Papert, 1969): даже с нелинейностью один перцептрон задаёт только линейную разделяющую поверхность, XOR нельзя моделировать перцептроном...



- Это почему-то восприняли как большую проблему, которая ставит крест на нейронных сетях.

История ANN и перцептрон

- ...хм, ну конечно! Для нелинейных функций нужна сеть перцептронов:



История ANN и перцептрон

- (Brison, Hoback, 1969): предложили алгоритм обратного распространения ошибки (backpropagation).
- (Hinton, 1974): переоткрыл backpropagation, с тех пор он стал популярным.
- Во второй половине 1970-х появились многоуровневые ANN, была разработана современная теория.
- Глубокие модели появились в первой половине 1980-х гг.! Это вообще не очень хитрая идея сама по себе.

История ANN и перцептрон

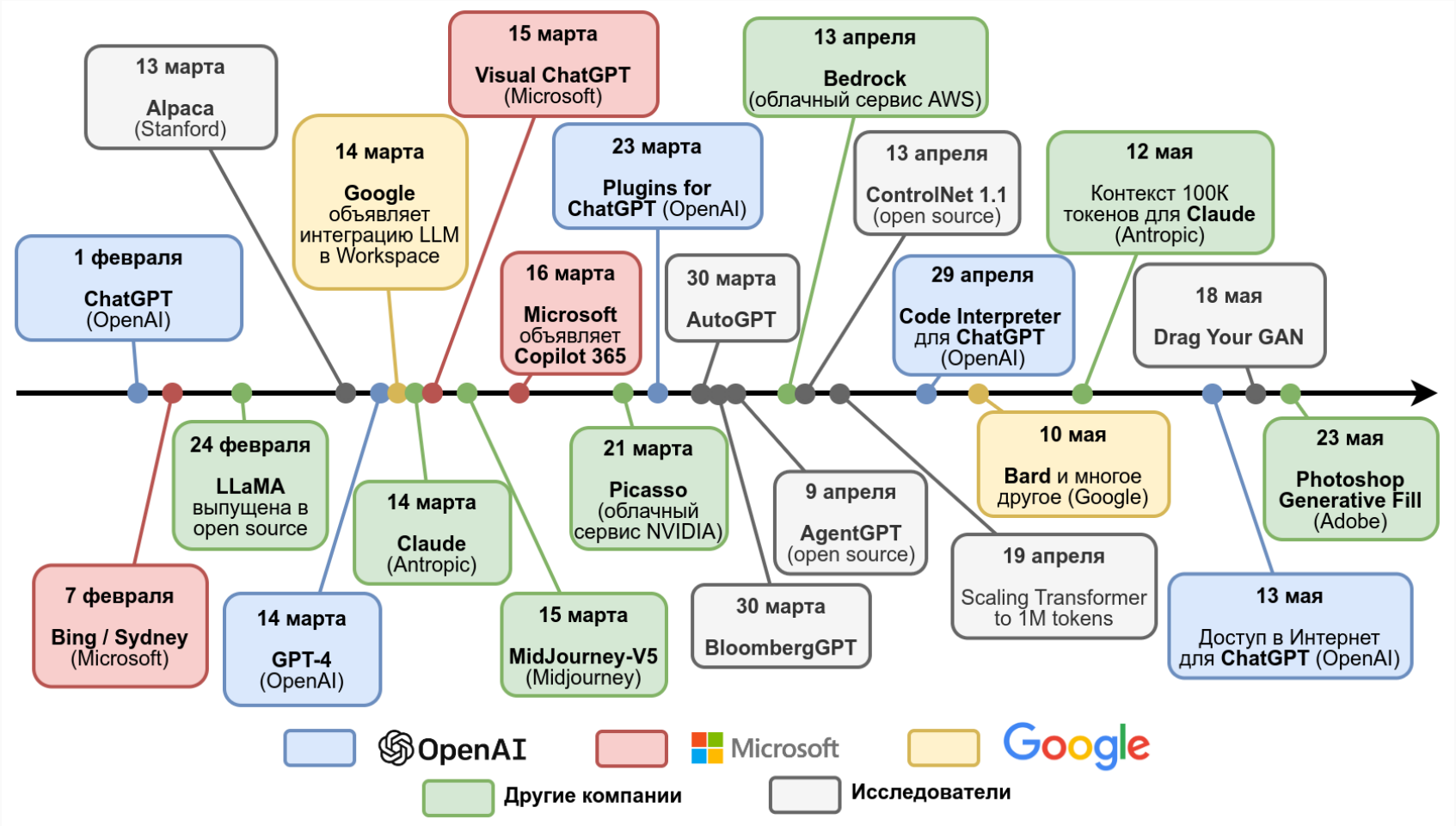
- Но к началу 1990-х решили, что нейронные сети использовать смысла нет, и началась «вторая зима».
- Это было потому, что тогда ни математически, ни вычислительно не могли нормально обучить большие модели.
- Нужно было обучать сети неделями, а качество, например, на распознавании речи проигрывало HMM, в других задачах проигрывало другим методам.
- John Denker, 1994: «neural networks are the second best way of doing just about anything».
- К концу 90-х на нейросетях остались только обработка изображений (Yann LeCun) и несколько других групп (Geoffrey Hinton, Yoshua Bengio).

История ANN и перцептрон

- Революция deep learning начинается в 2006: в группе Джеффри Хинтона изобрели Deep Belief Networks (DBN).
- Основная идея: научились делать *обучение без учителя*, выделять признаки (при помощи машин Больцмана, RBM; появились в 1983), использовать это как предобучение.
- Компьютеры стали мощнее, появились способы передать вычисления на GPU, и вычислительно мы смогли обрабатывать гораздо более объёмные датасеты, которые как раз стали доступными.
- А потом появились новые методы регуляризации (дропаут, затем batch normalization), и RBM с предобучением тоже стали не очень нужны.

История ANN и перцептрон

- Сейчас мы живём во время очередной мини-революции, на этот раз языковых моделей.



История ANN и перцептрон

- Многие считают, что AGI не за горами!
- Вот основные компоненты революции глубокого обучения.
- Осталось только понять, что всё это значит. :)
- Но сначала вспомним основы байесовского вывода: без них непонятно, откуда берутся функции ошибки, которые мы потом будем минимизировать.
- Начнём с монетки, а закончим линейной и логистической регрессией — и вот логистическая регрессия окажется ровно одним нейроном, с которого в следующий раз и начнём собирать сеть.

Байесовский подход

Основные определения

- Нам не понадобятся математические определения сигма-алгебры, вероятностной меры, борелевских множеств и т.п.
- Достаточно понимать, что бывают дискретные случайные величины (неотрицательные вероятности исходов в сумме дают единицу) и непрерывные случайные величины (интеграл неотрицательной функции плотности равен единице).

Основные определения

- *Совместная вероятность* – вероятность одновременного наступления двух событий, $p(x, y)$; маргинализация:

$$p(x) = \sum_y p(x, y).$$

- *Условная вероятность* – вероятность наступления одного события, если известно, что произошло другое, $p(x \mid y)$:

$$p(x, y) = p(x \mid y)p(y) = p(y \mid x)p(x).$$

- *Теорема Байеса* – из предыдущей формулы:

$$p(y|x) = \frac{p(x|y)p(y)}{p(x)} = \frac{p(x|y)p(y)}{\sum_{y'} p(x|y')p(y')}.$$

- *Независимость*: x и y независимы, если

$$p(x, y) = p(x)p(y).$$

Прямые и обратные задачи

- Прямая задача: в урне лежат 10 шаров, из них 3 чёрных. Какова вероятность выбрать чёрный шар?
- Или: в урне лежат 10 шаров с номерами от 1 до 10. Какова вероятность того, что номера трёх последовательно выбранных шаров дадут в сумме 12?
- Обратная задача: перед нами две урны, в каждой по 10 шаров, но в одной 3 чёрных, а в другой — 6. Кто-то взял из какой-то урны шар, и он оказался чёрным. Насколько вероятно, что он брал шар из первой урны?
- Заметьте, что в обратной задаче вероятности сразу стали байесовскими (хоть здесь и можно переформулировать через частоты).

Прямые и обратные задачи

- Иначе говоря, прямые задачи теории вероятностей описывают некий вероятностный процесс или модель и просят подсчитать ту или иную вероятность (т.е. фактически по модели предсказать поведение).
- Обратные задачи содержат *скрытые переменные* (в примере — номер урны, из которой брали шар). Они часто просят по известному поведению построить вероятностную модель.
- Задачи машинного обучения обычно являются задачами второй категории.

Вероятность как частота

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

Вероятность как степень доверия

- Мы можем рассуждать о том, «насколько вероятно» то, что
  - сборная России победит на чемпионате мира по футболу в 2026 году;
  - «Одиссею» написала женщина;
  - Александр Фёдорович Керенский бежал за границу в женском платье;
  - ...
- Но о «стремящемся к бесконечности количестве экспериментов» говорить бессмысленно — эксперимент здесь ровно один.

Вероятность как степень доверия

- Здесь вероятности уже выступают как *степени доверия* (degrees of belief). Это байесовский подход к вероятностям (Томас Байес так понимал).
- К счастью, и те, и другие вероятности подчиняются одним и тем же законам; есть результаты о том, что вполне естественные аксиомы вероятностной логики тут же приводят к весьма узкому классу функций (Cox, 19).

Определения

- Запишем теорему Байеса:

$$p(\theta|D) = \frac{p(\theta)p(D|\theta)}{p(D)}.$$

- Здесь $p(\theta)$ — *априорная вероятность* (prior probability), $p(D|\theta)$ — *правдоподобие* (likelihood), $p(\theta|D)$ — *апостериорная вероятность* (posterior probability), $p(D) = \int p(D \mid \theta)p(\theta)\mathrm{d}\theta$ — вероятность данных (evidence).

ML vs. MAP

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

$$\theta_{ML} = \arg\max_\theta p(D \mid \theta).$$

- В байесовском подходе ищут *апостериорное распределение* (posterior)

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

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

$$\theta_{MAP} = \arg\max_\theta p(\theta \mid D) = \arg\max_\theta p(D \mid \theta)p(\theta).$$

Байесовский вывод для монетки

ML vs. MAP

- Мы остановились на том, что в статистике обычно ищут *гипотезу максимального правдоподобия* (maximum likelihood):

$$\theta_{ML} = \arg\max_\theta p(D \mid \theta).$$

- В байесовском подходе ищут *апостериорное распределение* (posterior)

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

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

$$\theta_{MAP} = \arg\max_\theta p(\theta \mid D) = \arg\max_\theta p(D \mid \theta)p(\theta).$$

Постановка задачи

- Простая задача вывода: дана нечестная монетка, она подброшена N раз, имеется последовательность результатов падения монетки. Надо определить её «нечестность» и предсказать, чем она выпадет в следующий раз.

Постановка задачи

- Если у нас есть вероятность $p_h = \theta$ того, что монетка выпадет орлом (вероятность решки $p_t = 1 - \theta$), то вероятность того, что выпадет последовательность s, которая содержит n орлов и m решек, равна

$$p\left(s|p_h = \theta\right) = \theta^n (1-\theta)^m.$$

- Сделаем предположение: будем считать, что вероятность монетки до экспериметов для нас распределена равномерно, т.е. у нас нет априорных предпочтений насчёт θ.
- Теперь нужно использовать теорему Байеса и вычислить скрытые параметры.

Пример применения теоремы Байеса

- Правдоподобие: $p\left(\theta|s\right) = \frac{p(s|\theta)p(\theta)}{p(s)}$.
- Здесь $p(\theta)$ следует понимать как непрерывную случайную величину, сосредоточенную на интервале $[0, 1]$, коей она и является. Наше предположение о равномерном распределении в данном случае значит, что априорная вероятность $p(\theta) = 1$, $\theta \in [0, 1]$ (т.е. априори мы не знаем, насколько нечестна монетка, и предполагаем это равновероятным). А $p\left(s|\theta\right)$ мы уже знаем.
- Итого получается:

$$p\left(\theta|s\right) = \frac{\theta^n(1-\theta)^m}{p(s)}.$$

Пример применения теоремы Байеса

- Итого получается:

$$p\left(\theta|s\right) = \frac{\theta^n(1-\theta)^m}{p(s)}.$$

- $p(s)$ можно подсчитать как

$$p(s) = \int_0^1 \theta^n(1-\theta)^m d\theta = \\ = \frac{\Gamma(n+1)\Gamma(m+1)}{\Gamma(n+m+2)} = \frac{n!m!}{(n+m+1)!},$$

но найти $\arg\max_\theta p(\theta \mid s) = \frac{n}{n+m}$ можно и без этого.

Пример применения теоремы Байеса

- Итого получается:

$$p\left(\theta|s\right) = \frac{\theta^n(1-\theta)^m}{p(s)}.$$

- Но это ещё не всё. Чтобы предсказать следующий исход, надо найти $p\left(\text{heads}|s\right)$:

$$p(\text{heads}|s) = \int_0^1 p(\text{heads}|\theta)p(\theta|s)d\theta =$$

$$= \int_0^1 \frac{\theta^{n+1}(1-\theta)^m}{p(s)} dp_h =$$

$$= \frac{(n+1)!m!}{(n+m+2)!} \cdot \frac{(n+m+1)!}{n!m!} = \frac{n+1}{n+m+2}.$$

- Получили правило Лапласа.

Пример применения теоремы Байеса

- Итого получается:

$$p\left(\theta|s\right) = \frac{\theta^n(1-\theta)^m}{p(s)}.$$

- Это была иллюстрация двух основных задач байесовского вывода:
  (1) найти апостериорное распределение (posterior distribution) на гипотезах/параметрах:

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

  и/или найти максимальную апостериорную гипотезу (maximum a posteriori) $\theta_{\mathrm{MAP}} = \arg\max_\theta p(\theta \mid D)$;
  (2) найти предсказательное распределение (predictive distribution) исходов дальнейших экспериментов:

$$p(x \mid D) \propto \int_{\theta\in\Theta} p(x \mid \theta)p(D|\theta)p(\theta)\mathrm{d}\theta.$$

Сопряжённые априорные распределения

Напоминание

- Напоминаю, что основная наша задача – как обучить параметры распределения и/или предсказать следующие его точки по имеющимся данным.
- В байесовском выводе участвуют:
  - $p(x \mid \theta)$ – правдоподобие данных;
  - $p(\theta)$ – априорное распределение;
  - $p(x) = \int_\Theta p(x \mid \theta) p(\theta) d\theta$ – маргинальное правдоподобие;
  - $p(\theta \mid x) = \frac{p(x|\theta)p(\theta)}{p(x)}$ – апостериорное распределение;
  - $p(x' \mid x) = \int_\Theta p(x' \mid \theta) p(\theta \mid x) d\theta$ – предсказание нового x'.
- Задача обычно в том, чтобы найти $p(\theta \mid x)$ и/или $p(x' \mid x)$.

Априорные распределения

- Когда мы проводим байесовский вывод, у нас, кроме правдоподобия, должно быть ещё *априорное распределение* (prior distribution) по всем возможным значениям параметров.
- Мы раньше к ним специально не присматривались, но они очень важны.
- Задача байесовского вывода – как подсчитать $p(\theta \mid x)$ и/или $p(x' \mid x)$.
- Но чтобы это сделать, сначала надо выбрать $p(\theta)$. Как выбирать априорные распределения?

Сопряжённые априорные распределения

- Разумная цель: давайте будем выбирать распределения так, чтобы они оставались такими же и *a posteriori*.
- До начала вывода есть априорное распределение $p(\theta)$.
- После него есть какое-то новое апостериорное распределение $p(\theta \mid x)$.
- Я хочу, чтобы $p(\theta \mid x)$ тоже имело тот же вид, что и $p(\theta)$, просто с другими параметрами.

Сопряжённые априорные распределения

- Не слишком формальное определение: семейство распределений $p(\theta \mid \alpha)$ называется семейством *сопряжённых априорных распределений* для семейства правдоподобий $p(x \mid \theta)$, если после умножения на правдоподобие апостериорное распределение $p(\theta \mid x, \alpha)$ остаётся в том же семействе: $p(\theta \mid x, \alpha) = p(\theta \mid \alpha')$.
- α называются *гиперпараметрами* (hyperparameters), это «параметры распределения параметров».
- Тривиальный пример: семейство всех распределений будет сопряжённым чему угодно, но это не очень интересно.

Сопряжённые априорные распределения

- Разумеется, вид хорошего априорного распределения зависит от вида распределения собственно данных, $p(x \mid \theta)$.
- Сопряжённые априорные распределения подсчитаны для многих распределений, мы приведём несколько примеров.

Испытания Бернулли

- Каким будет сопряжённое априорное распределение для бросания нечестной монетки (испытаний Бернулли)?
- Ответ: это будет бета-распределение; плотность распределения нечестности монетки θ

$$p(\theta \mid \alpha, \beta) = \frac{\theta^{\alpha-1}(1-\theta)^{\beta-1}}{B(\alpha, \beta)}.$$

Испытания Бернулли

- Плотность распределения нечестности монетки θ

$$p(\theta \mid \alpha, \beta) = \frac{\theta^{\alpha-1}(1-\theta)^{\beta-1}}{B(\alpha, \beta)}.$$

- Тогда, если мы посэмплируем монетку, получив s орлов и f решек, получится

$$p(s, f \mid \theta) = \binom{s+f}{s}\theta^s(1-\theta)^f, \text{ и}$$

$$p(\theta|s, f) = \frac{\binom{s+f}{s}\theta^{s+\alpha-1}(1-\theta)^{f+\beta-1}/B(\alpha, \beta)}{\int_0^1 \binom{s+f}{s}x^{s+\alpha-1}(1-x)^{f+\beta-1}/B(\alpha, \beta)dx} =$$

$$= \frac{\theta^{s+\alpha-1}(1-\theta)^{f+\beta-1}}{B(s+\alpha, f+\beta)}.$$

Испытания Бернулли

- Итого получается, что сопряжённое априорное распределение для параметра нечестной монетки θ – это

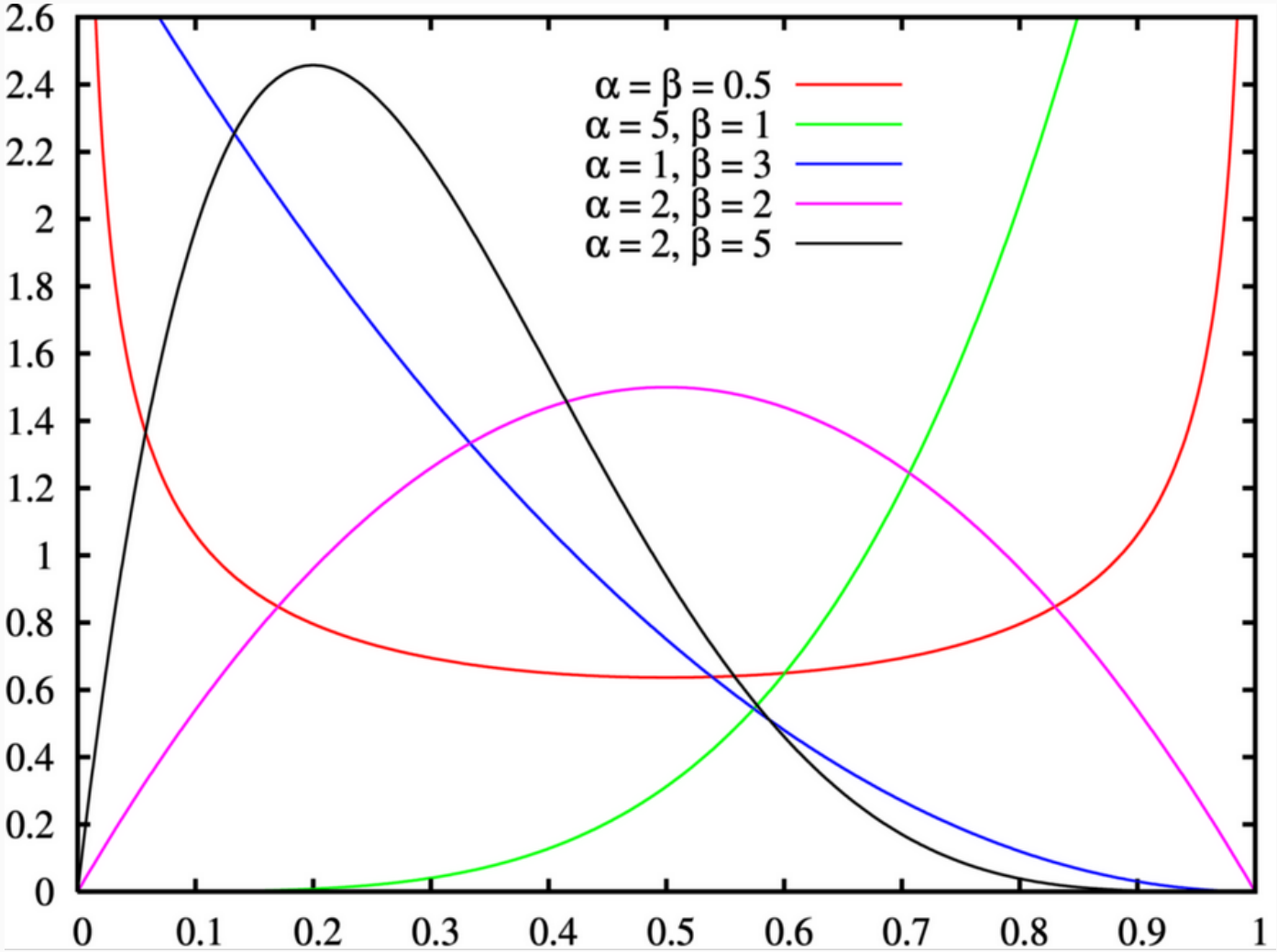
$$p(\theta \mid \alpha, \beta) \propto \theta^{\alpha-1}(1-\theta)^{\beta-1}.$$

- После получения новых данных с s орлами и f решками гиперпараметры меняются на

$$p(\theta \mid s+\alpha, f+\beta) \propto \theta^{s+\alpha-1}(1-\theta)^{f+\beta-1}.$$

- На этом этапе можно забыть про сложные формулы и выводы, получилось очень простое правило обучения (под обучением теперь понимается изменение гиперпараметров).

Бета--распределение



Мультиномиальное распределение

- Простое обобщение: рассмотрим мультиномиальное распределение с n испытаниями, k категориями и по x_i экспериментов дали категорию i.
- Параметры θ_i показывают вероятность попасть в категорию i:

$$p(x \mid \theta) = \binom{n}{x_1, \dots, x_n} \theta_1^{x_1} \theta_2^{x_2} \dots \theta_k^{x_k}.$$

- Сопряжённым априорным распределением будет распределение Дирихле:

$$p(\theta \mid \alpha) \propto \theta_1^{\alpha_1 - 1} \theta_2^{\alpha_2 - 1} \dots \theta_k^{\alpha_k - 1}.$$

Мультиномиальное распределение

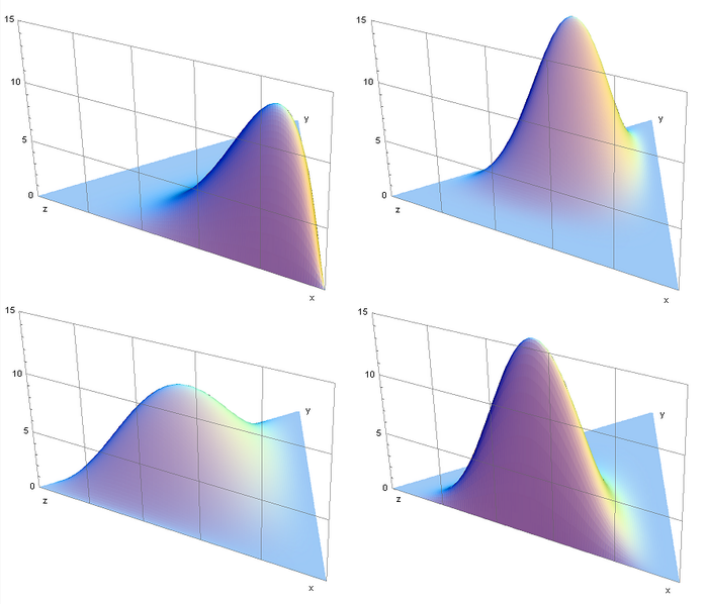
- Сопряжённым априорным распределением будет распределение Дирихле:

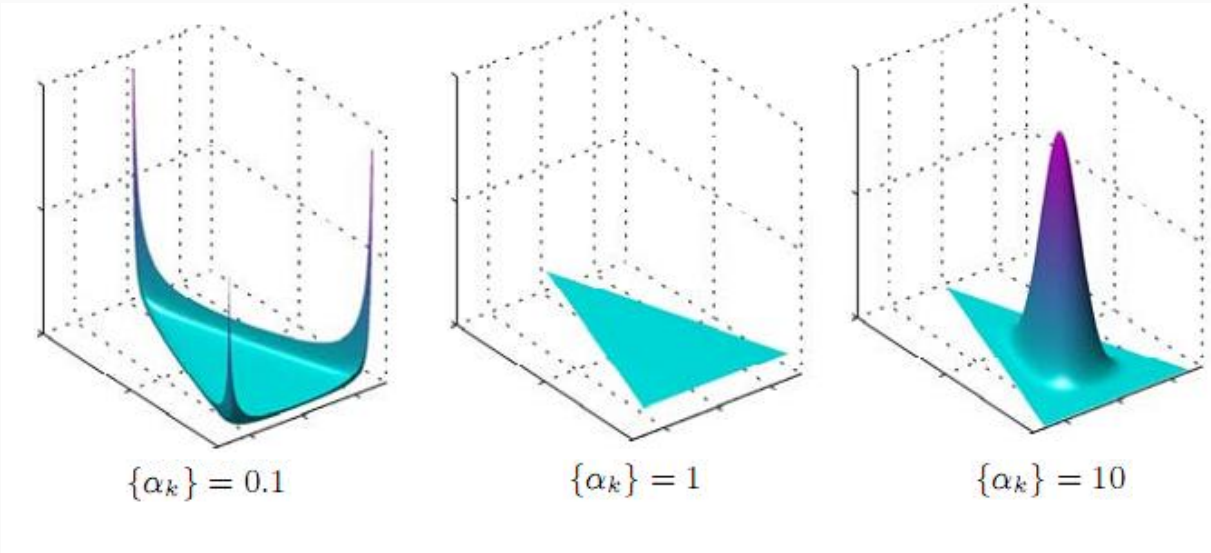
$$p(\theta \mid \alpha) \propto \theta_1^{\alpha_1 - 1} \theta_2^{\alpha_2 - 1} \dots \theta_k^{\alpha_k - 1}.$$

Упражнение. Докажите, что при получении данных $x_1, \dots, x_k$ гиперпараметры изменятся на

$$p(\theta \mid x, \alpha) = p(\theta \mid x + \alpha) \propto \theta_1^{x_1 + \alpha_1 - 1} \theta_2^{x_2 + \alpha_2 - 1} \dots \theta_k^{x_k + \alpha_k - 1}.$$

Распределение Дирихле





Линейная регрессия

Метод наименьших квадратов

- Линейная регрессия: рассмотрим линейную функцию

$$y(\mathbf{x}, \mathbf{w}) = w_0 + \sum_{j=1}^{p} x_j w_j = \mathbf{x}^\top \mathbf{w}, \quad \mathbf{x} = (1, x_1, \dots, x_p).$$

- Таким образом, по вектору входов $\mathbf{x}^\top = (x_1, \dots, x_p)$ мы будем предсказывать выход y как

$$\hat{y}(\mathbf{x}) = \hat{w}_0 + \sum_{j=1}^{p} x_j \hat{w}_j = \mathbf{x}^\top \hat{\mathbf{w}}.$$

Метод наименьших квадратов

- Как найти оптимальные параметры $\hat{\mathbf{w}}$ по тренировочным данным вида $(\mathbf{x}_i, y_i)_{i=1}^N$?
- Метод наименьших квадратов: будем минимизировать

$$\mathrm{RSS}(\mathbf{w}) = \sum_{i=1}^{N} (y_i - \mathbf{x}_i^\top \mathbf{w})^2.$$

- Как минимизировать?

Метод наименьших квадратов

- Можно на самом деле решить задачу точно – записать как

$$\mathrm{RSS}(\mathbf{w}) = (\mathbf{y} - \mathbf{X}\mathbf{w})^\top (\mathbf{y} - \mathbf{X}\mathbf{w}),$$

  где $\mathbf{X}$ – матрица $N \times p$, продифференцировать по $\mathbf{w}$, получится

$$\hat{\mathbf{w}} = (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{y},$$

  если матрица $\mathbf{X}^\top \mathbf{X}$ невырожденная.
- Замечание: $\left(\mathbf{X}^\top \mathbf{X}\right)^{-1} \mathbf{X}^\top$ называется *псевдообратной матрицей Мура–Пенроуза* (Moore–Penrose pseudo-inverse) матрицы $\mathbf{X}$; это обобщение понятия обратной матрицы на неквадратные матрицы.
- Много ли нужно точек, чтобы обучить такую модель?

Байесовская регрессия

- Теперь давайте поговорим о линейной регрессии по-байесовски.
- Основное наше предположение – в том, что шум (ошибка в данных) распределён нормально, т.е. переменная t, которую мы наблюдаем, получается как

$$t = y(\mathbf{x}, \mathbf{w}) + \epsilon, \quad \epsilon \sim N(0, \sigma^2).$$

  Иными словами,

$$p(t \mid \mathbf{x}, \mathbf{w}, \sigma^2) = N(t \mid y(\mathbf{x}, \mathbf{w}), \sigma^2).$$

- Здесь пока y – любая функция.

Байесовская регрессия

- Чтобы не повторять совсем уж то же самое, мы рассмотрим не в точности линейную регрессию, а её естественное обобщение – линейную модель с базисными функциями:

$$y(\mathbf{x}, \mathbf{w}) = w_0 + \sum_{j=1}^{M-1} w_j \phi_j(\mathbf{x}) = \mathbf{w}^\top \phi(\mathbf{x})$$

(M параметров, $M-1$ базисная функция, $\phi_0(\mathbf{x}) = 1$).

Байесовская регрессия

- Базисные функции ϕ_i – это, например:
  - результат feature extraction;
  - расширение линейной модели на нелинейные зависимости (например, $\phi_j(x) = x^j$);
  - локальные функции, которые существенно не равны нулю только в небольшой области (например, гауссовские базисные функции $\phi_j(\mathbf{x}) = e^{-\frac{(x-\mu_j)^2}{2s^2}}$);
  - ...

Байесовская регрессия

- Рассмотрим набор данных $\mathbf{X} = \{\mathbf{x}_1, \dots, \mathbf{x}_N\}$ со значениями $\mathbf{t} = \{t_1, \dots, t_N\}$.
- Будем предполагать, что данные взяты независимо по одному и тому же распределению:

$$p(\mathbf{t} \mid \mathbf{X}, \mathbf{w}, \sigma^2) = \prod_{n=1}^{N} N\left(t_n \mid \mathbf{w}^\top \phi(\mathbf{x}_n), \sigma^2\right).$$

- Прологарифмируем (опустим $\mathbf{X}$, т.к. по нему всегда условная вероятность будет):

$$\ln p(\mathbf{t} \mid \mathbf{w}, \sigma^2) = -\frac{N}{2} \ln(2\pi\sigma^2) - \frac{1}{2\sigma^2} \sum_{n=1}^{N} \left(t_n - \mathbf{w}^\top \phi(\mathbf{x}_n)\right)^2.$$

- Прологарифмируем (опустим $\mathbf{X}$, т.к. по нему всегда условная вероятность будет):

$$\ln p(\mathbf{t} \mid \mathbf{w}, \sigma^2) = -\frac{N}{2} \ln(2\pi\sigma^2) - \frac{1}{2\sigma^2} \sum_{n=1}^{N} \left(t_n - \mathbf{w}^\top \phi(\mathbf{x}_n)\right)^2.$$

- И вот мы получили, что для максимизации правдоподобия по $\mathbf{w}$ нам нужно как раз минимизировать среднеквадратичную ошибку!

$$\nabla_{\mathbf{w}} \ln p(\mathbf{t} \mid \mathbf{w}, \sigma^2) = \frac{1}{\sigma^2} \sum_{n=1}^{N} \left(t_n - \mathbf{w}^\top \phi(\mathbf{x}_n)\right) \phi(\mathbf{x}_n).$$

Байесовская регрессия

- Решая систему уравнений $\nabla \ln p(\mathbf{t} \mid \mathbf{w}, \sigma^2) = 0$, получаем то же самое, что и раньше:

$$\mathbf{w}_{ML} = \left(\Phi^\top \Phi\right)^{-1} \Phi^\top \mathbf{t}.$$

- Здесь $\Phi = (\phi_j(\mathbf{x}_i))_{i,j}$.

- Теперь можно и относительно σ^2 максимизировать правдоподобие; получим

$$\sigma^2_{ML} = \frac{1}{N}\sum_{n=1}^{N}\left(t_n - \mathbf{w}^\top_{ML}\phi(\mathbf{x}_n)\right)^2,$$

т.е. как раз выборочная дисперсия имеющихся данных вокруг предсказанного значения.

Пример: полиномиальная аппроксимация

Полиномиальная аппроксимация

- Мы говорили о регрессии с базисными функциями:

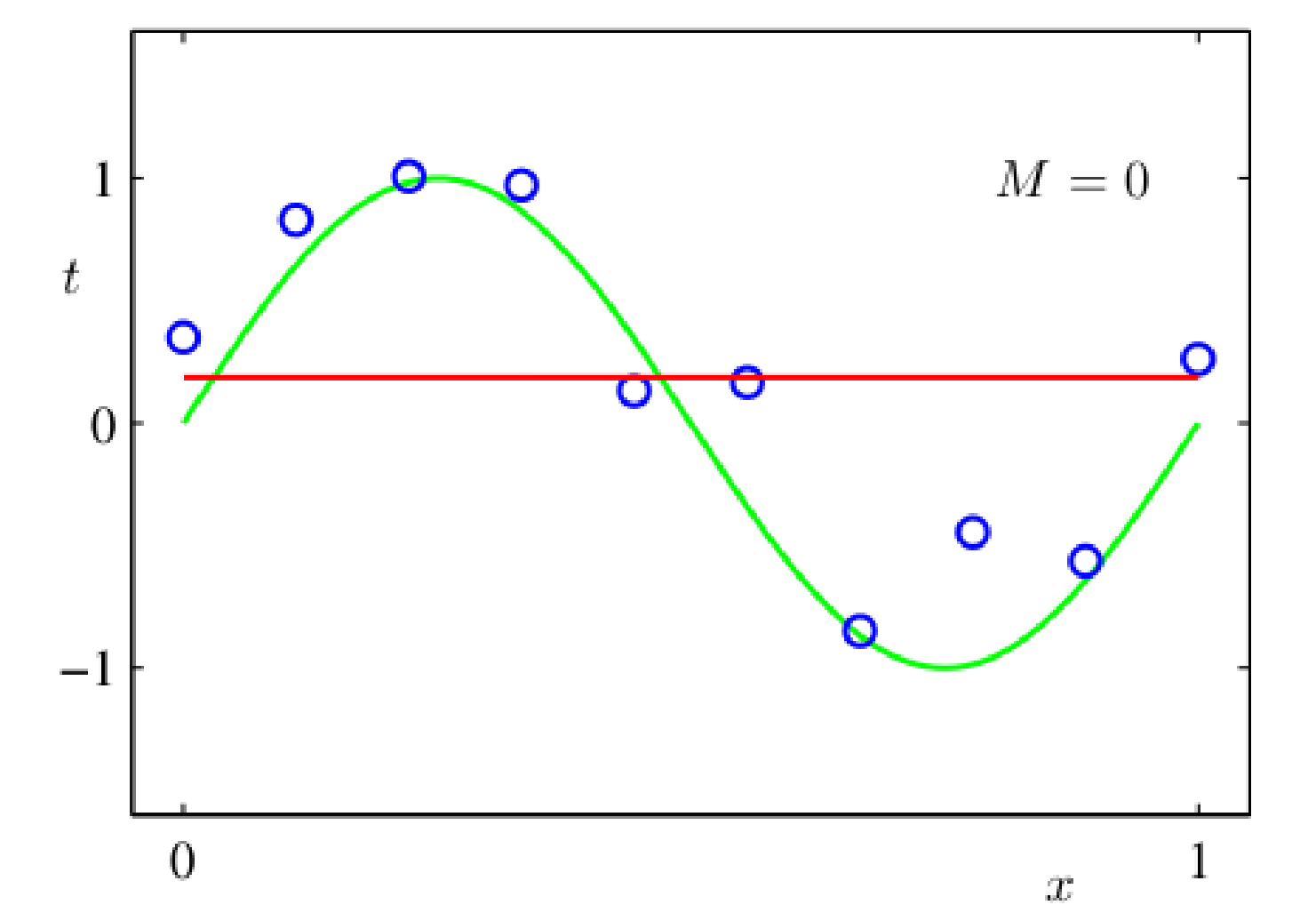
$$f(\mathbf{x}, \mathbf{w}) = w_0 + \sum_{j=1}^{M} w_j \phi_j(\mathbf{x}) = \mathbf{w}^\top \phi(\mathbf{x}).$$

- Давайте для примера рассмотрим такую регрессию для $\phi_j(x) = x^j$, т.е.

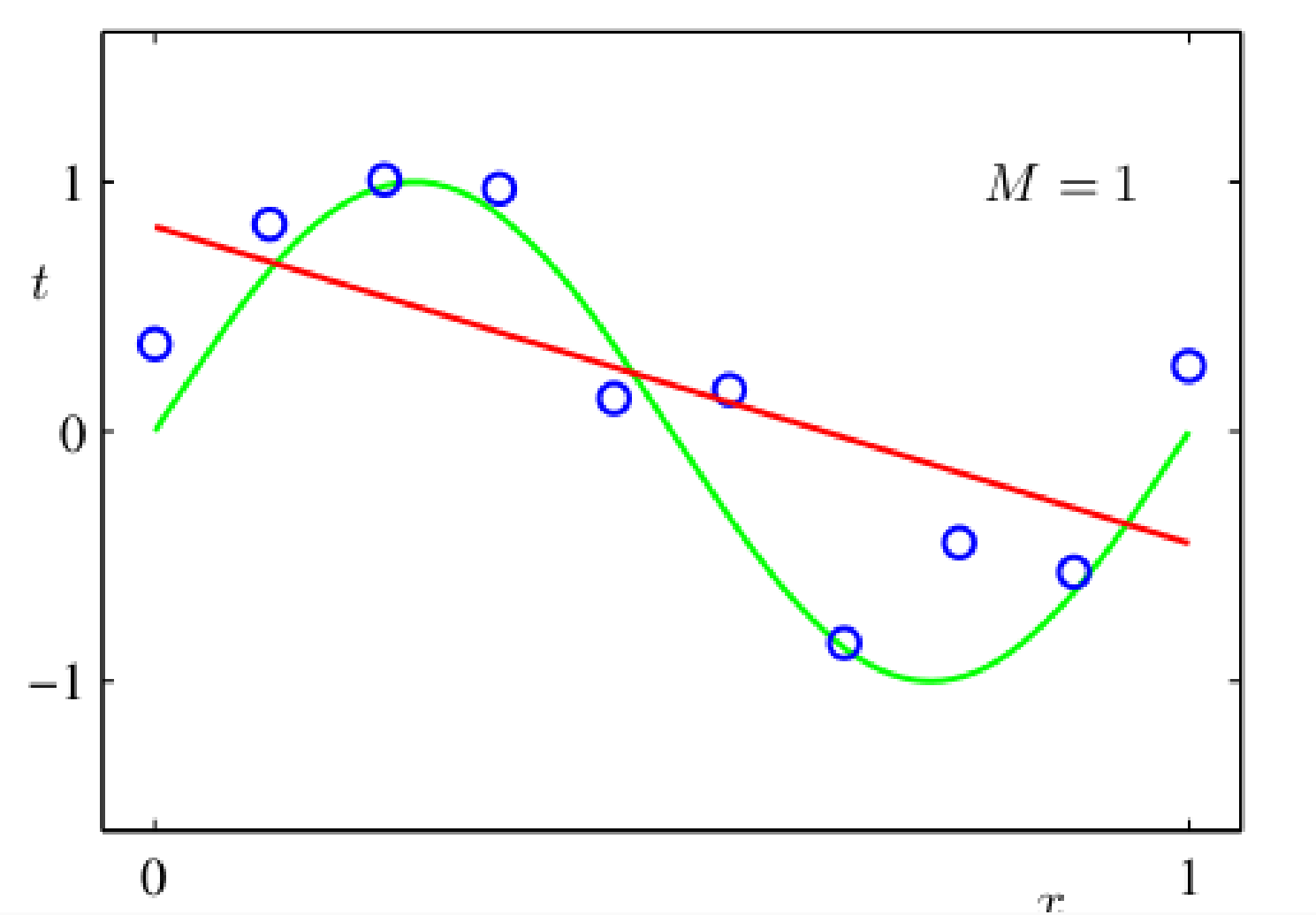
$$f(x, \mathbf{w}) = w_0 + w_1 x + w_2 x^2 + \ldots + w_M x^M.$$

- И будем, как раньше, минимизировать квадратичную ошибку.
- Пример с кодом.

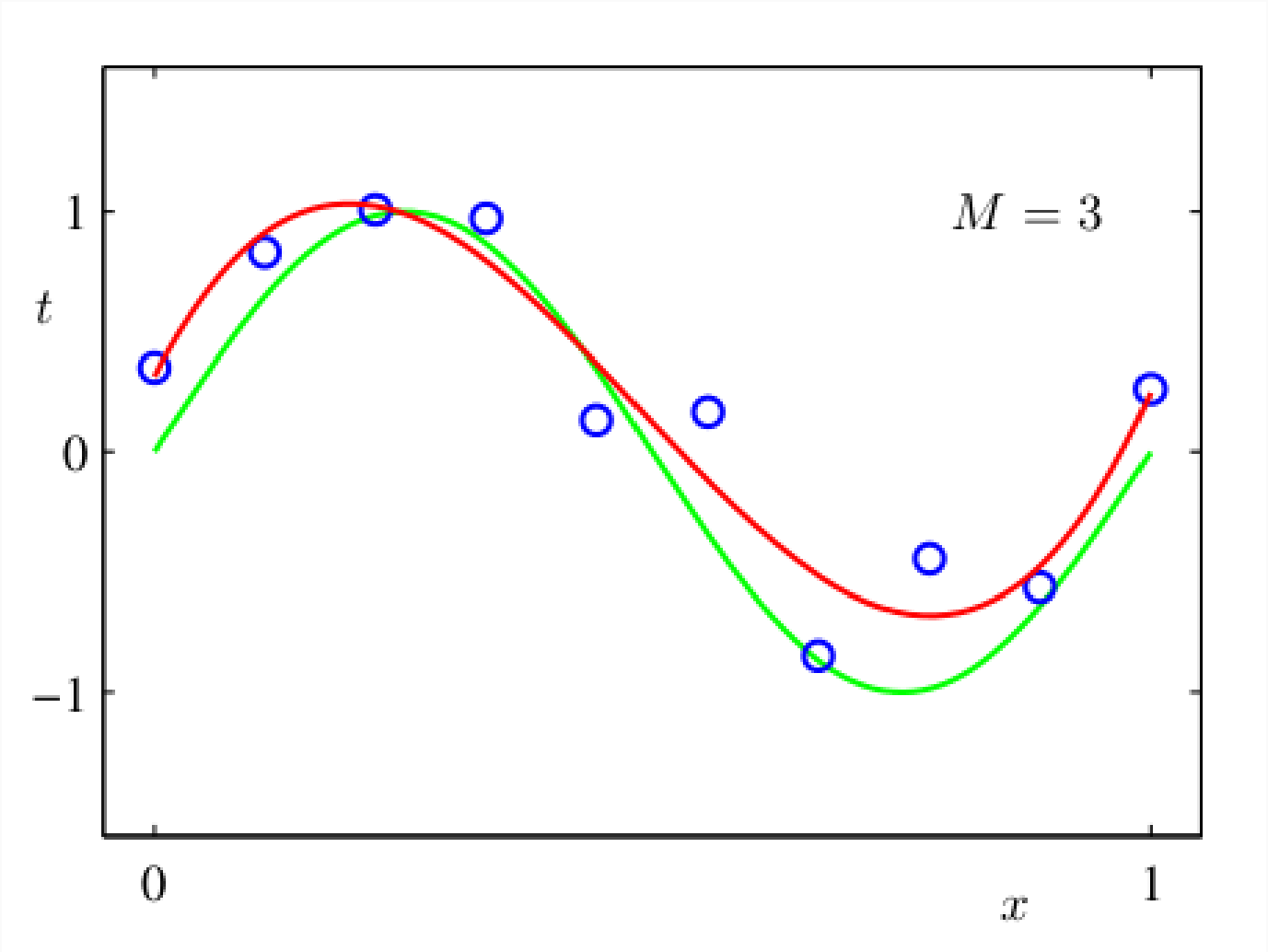
Полиномиальная аппроксимация



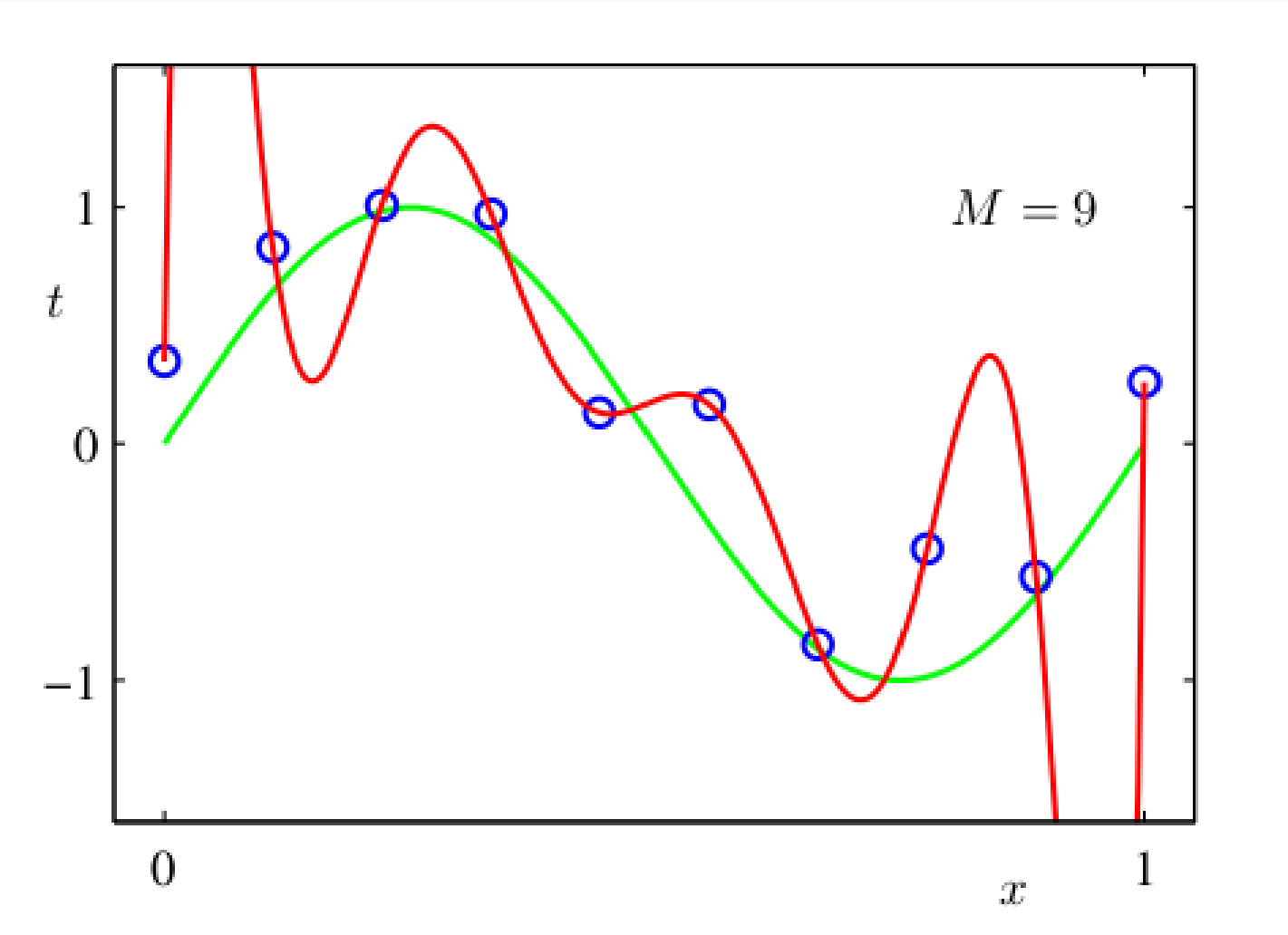
Полиномиальная аппроксимация



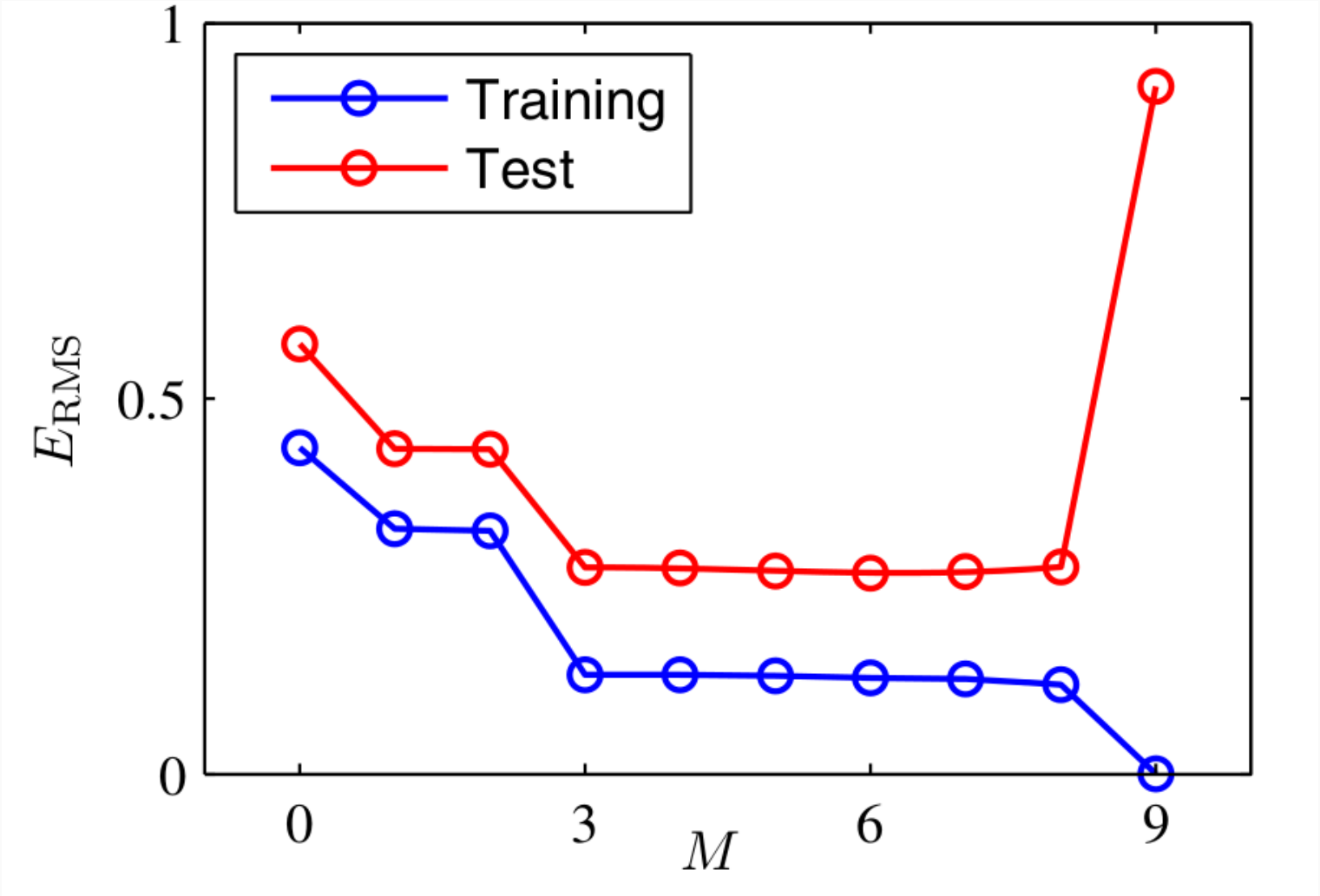
Полиномиальная аппроксимация



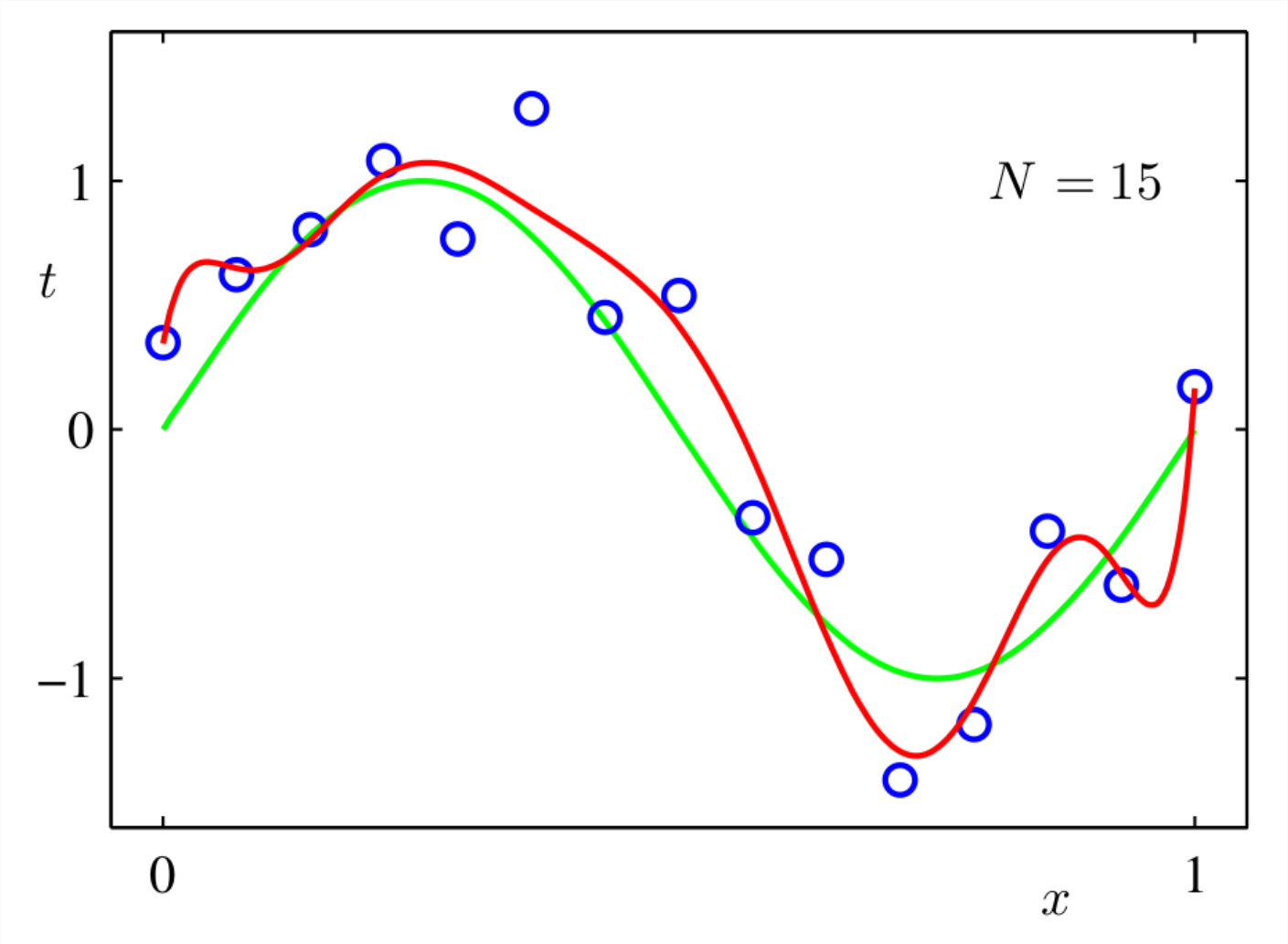
Полиномиальная аппроксимация



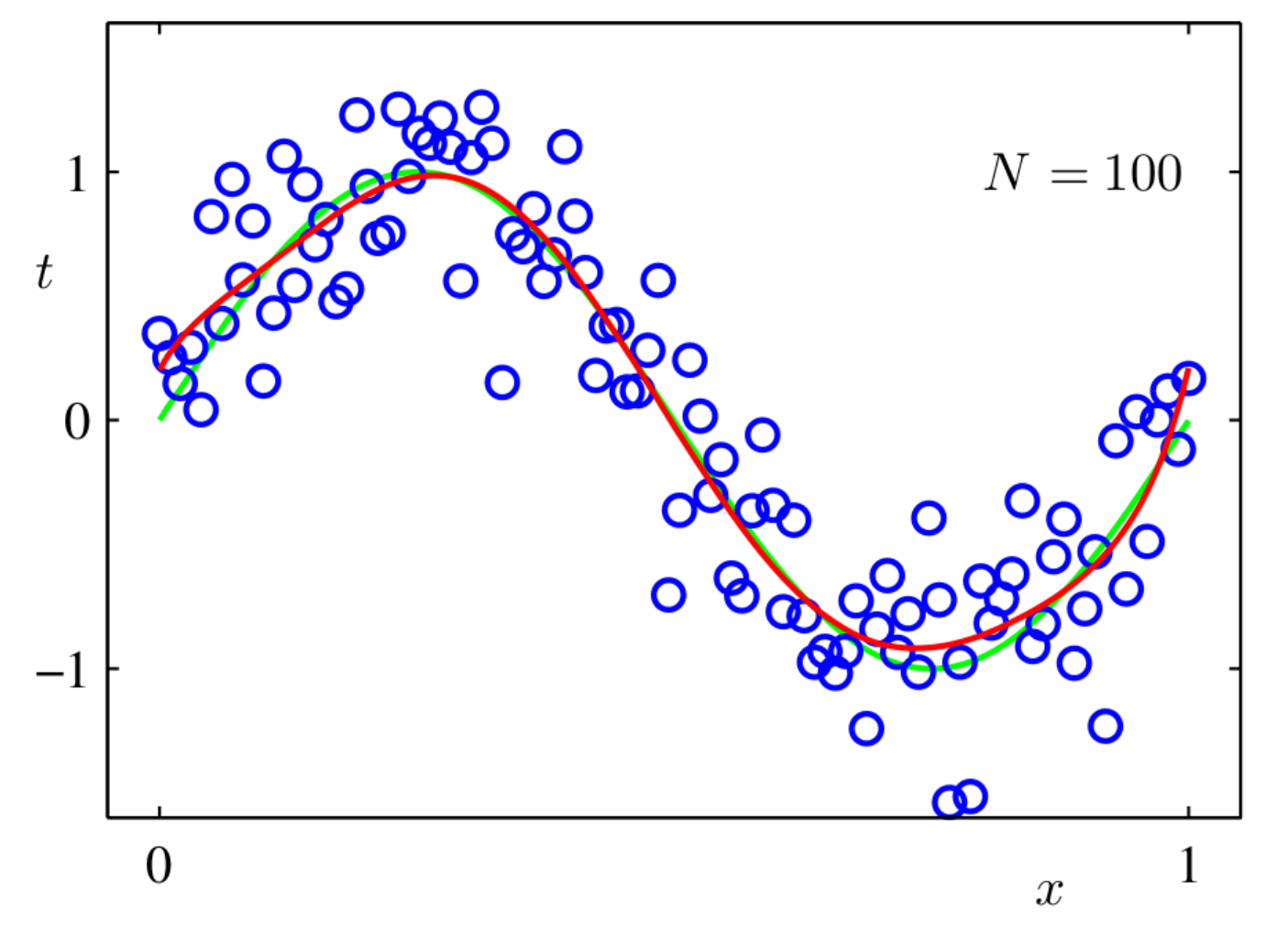
Значения RMS



Можно собрать больше данных...



Можно собрать больше данных...



Значения коэффициентов

	$M=0$	$M=1$	$M=6$	$M=9$
$w_0^\star$	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

Логистическая регрессия

В прошлый раз

- Итак, мы рассмотрели логистический сигмоид:

$$p(C_1 \mid \mathbf{x}) = \frac{p(\mathbf{x} \mid C_1)p(C_1)}{p(\mathbf{x} \mid C_1)p(C_1) + p(\mathbf{x} \mid C_2)p(C_2)} = \frac{1}{1 + e^{-a}} = \sigma(a),$$

$$\text{где } a = \ln \frac{p(\mathbf{x} \mid C_1)p(C_1)}{p(\mathbf{x} \mid C_2)p(C_2)}, \qquad \sigma(a) = \frac{1}{1 + e^{-a}}.$$

- Вывели из него LDA и QDA, обучив их методом максимального правдоподобия.

Два класса

- Два класса, и апостериорное распределение – логистический сигмоид на линейной функции:

$$p(C_1 \mid \phi) = y(\phi) = \sigma(\mathbf{w}^\top \phi), \quad p(C_2 \mid \phi) = 1 - p(C_1 \mid \phi).$$

- *Логистическая регрессия* – это когда мы напрямую оптимизируем $\mathbf{w}$.

Два класса

- Для датасета $\{\phi_n, t_n\}$, $t_n \in \{0, 1\}$, $\phi_n = \phi(\mathbf{x}_n)$:

$$p(\mathbf{t} \mid \mathbf{w}) = \prod_{n=1}^{N} y_n^{t_n} (1 - y_n)^{1 - t_n}, \quad y_n = p(C_1 \mid \phi_n).$$

- Ищем параметры максимального правдоподобия, минимизируя $-\ln p(\mathbf{t} \mid \mathbf{w})$:

$$E(\mathbf{w}) = -\ln p(\mathbf{t} \mid \mathbf{w}) = -\sum_{n=1}^{N} \left[t_n \ln y_n + (1 - t_n) \ln(1 - y_n) \right].$$

Два класса

- Пользуясь тем, что $\sigma' = \sigma(1 - \sigma)$, берём градиент (похоже на перцептрон):

$$\nabla E(\mathbf{w}) = \sum_{n=1}^{N} (y_n - t_n)\phi_n.$$

- Если теперь сделать градиентный спуск, получим как раз разделяющую поверхность.
- Заметим, правда, что если данные действительно разделимы, то может получиться жуткий оверфиттинг: $\|\mathbf{w}\| \to \infty$, и сигмоид превращается в функцию Хевисайда. Надо регуляризовать.

IRLS

- В логистической регрессии не получается замкнутого решения из-за сигмоида.
- Но функция $E(\mathbf{w})$ всё равно выпуклая, и можно воспользоваться методом Ньютона-Рапсона – на каждом шаге использовать локальную квадратичную аппроксимацию к функции ошибки:

$$\mathbf{w}^{\text{new}} = \mathbf{w}^{\text{old}} - \mathbf{H}^{-1}\nabla E(\mathbf{w}),$$

где $\mathbf{H}$ (Hessian) – матрица вторых производных $E(\mathbf{w})$.

IRLS

- Замечание: давайте применим Ньютона-Рапсона к обычной линейной регрессии с квадратической ошибкой:

$$\nabla E(\mathbf{w}) = \sum_{n=1}^{N} \left(\mathbf{w}^\top \phi_n - t_n\right) \phi_n = \Phi^\top \Phi \mathbf{w} - \Phi^\top \mathbf{t},$$

$$\nabla\nabla E(\mathbf{w}) = \sum_{n=1}^{N} \phi_n \phi_n^\top = \Phi^\top \Phi,$$

и шаг оптимизации будет

$$\mathbf{w}^{\text{new}} = \mathbf{w}^{\text{old}} - \left(\Phi^\top \Phi\right)^{-1} \left[\Phi^\top \Phi \mathbf{w}^{\text{old}} - \Phi^\top \mathbf{t}\right] = \\ = \left(\Phi^\top \Phi\right)^{-1} \Phi^\top \mathbf{t},$$

т.е. мы за один шаг придём к решению.

IRLS

- Для логистической регрессии:

$$\nabla E(\mathbf{w}) = \sum_{n=1}^{N} (y_n - t_n)\, \phi_n = \Phi^\top (\mathbf{y} - \mathbf{t}),$$

$$\mathbf{H} = \nabla\nabla E(\mathbf{w}) = \sum_{n=1}^{N} y_n(1 - y_n)\phi_n \phi_n^\top = \Phi^\top R \Phi$$

для диагональной матрицы R с $R_{nn} = y_n(1 - y_n)$.

IRLS

- Формула шага оптимизации:

$$\mathbf{w}^{\text{new}} = \mathbf{w}^{\text{old}} - \left(\Phi^\top R \Phi\right)^{-1} \Phi^\top (\mathbf{y} - \mathbf{t}) = \left(\Phi^\top R \Phi\right)^{-1} \Phi^\top R \mathbf{z},$$

  где $\mathbf{z} = \Phi \mathbf{w}^{\text{old}} - R^{-1} (\mathbf{y} - \mathbf{t})$.
- Получилось как бы решение взвешенной задачи минимизации квадратического отклонения с матрицей весов R.
- Отсюда название: iterative reweighted least squares (IRLS).

Несколько классов

- В случае нескольких классов

$$p(C_k \mid \phi) = y_k(\phi) = \frac{e^{a_k}}{\sum_j e^{a_j}} \text{ для } a_k = \mathbf{w}_k^\top \phi.$$

- Опять выпишем максимальное правдоподобие; во-первых,

$$\frac{\partial y_k}{\partial a_j} = y_k \left([k = j] - y_j \right).$$

Несколько классов

- Теперь запишем правдоподобие – для схемы кодирования 1-of-K будет целевой вектор $\mathbf{t}_n$ и правдоподобие

$$p(\mathbf{T} \mid \mathbf{w}_1, \dots, \mathbf{w}_K) = \prod_{n=1}^{N} \prod_{k=1}^{K} p(C_k \mid \phi_n)^{t_{nk}} = \prod_{n=1}^{N} \prod_{k=1}^{K} y_{nk}^{t_{nk}}$$

для $y_{nk} = y_k(\phi_n)$; берём логарифм:

$$E(\mathbf{w}_1, \dots, \mathbf{w}_K) = -\ln p(\mathbf{T} \mid \mathbf{w}_1, \dots, \mathbf{w}_K) = -\sum_{n=1}^{N} \sum_{k=1}^{K} t_{nk} \ln y_{nk}, \text{ и}$$

$$\nabla_{\mathbf{w}_j} E(\mathbf{w}_1, \dots, \mathbf{w}_K) = -\sum_{n=1}^{N} \left(y_{nj} - t_{nj}\right) \phi_n.$$

Несколько классов

- Оптимизировать опять можно по Ньютону-Рапсону; гессиан получится как

$$\nabla_{\mathbf{w}_k} \nabla_{\mathbf{w}_j} E(\mathbf{w}_1, \dots, \mathbf{w}_K) = -\sum_{n=1}^{N} y_{nk} \left([k = j] - y_{nj} \right) \phi_n \phi_n^\top.$$

Пробит-регрессия

- А что если у нас другая форма сигмоида?
- Мы по-прежнему в той же постановке: два класса, $p(t = 1 \mid a) = f(a)$, $a = \mathbf{w}^\top \phi$, f – функция активации.
- Давайте установим функцию активации с порогом θ: для каждого ϕ_n, вычисляем $a_n = \mathbf{w}^\top \phi_n$, и

$$\begin{cases} t_n = 1, & \text{если } a_n \geq \theta, \\ t_n = 0, & \text{если } a_n < \theta. \end{cases}$$

Пробит-регрессия

- Если θ берётся по распределению $p(\theta)$, это соответствует

$$f(a) = \int_{-\infty}^{a} p(\theta)\mathrm{d}\theta.$$

- Пусть, например, $p(\theta)$ – гауссиан с нулевым средним и единичной дисперсией. Тогда

$$f(a) = \Phi(a) = \int_{-\infty}^{a} N\left(\theta \mid 0, 1\right)\mathrm{d}\theta.$$

Пробит-регрессия

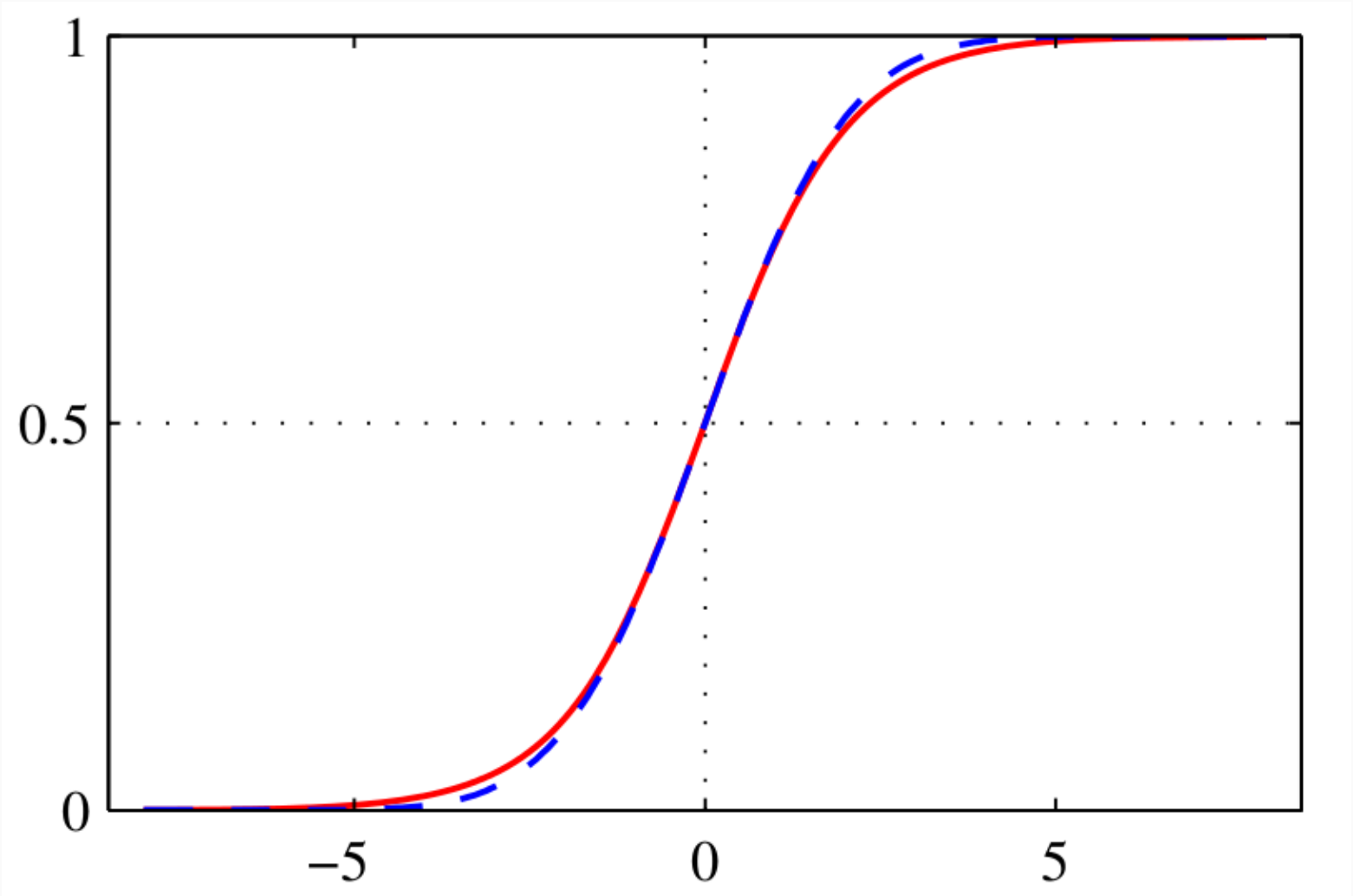
- Это называется *пробит-функцией* (probit); неэлементарная, но тесно связана с

$$\operatorname{erf}(a) = \frac{2}{\sqrt{\pi}} \int_0^a e^{-\frac{\theta^2}{2}} \mathrm{d}\theta :$$

$$\Phi(a) = \frac{1}{2}\left[1 + \frac{1}{\sqrt{2}}\operatorname{erf}(a)\right].$$

- Пробит-регрессия – это модель с пробит-функцией активации.

σ и Φ



Спасибо за внимание!