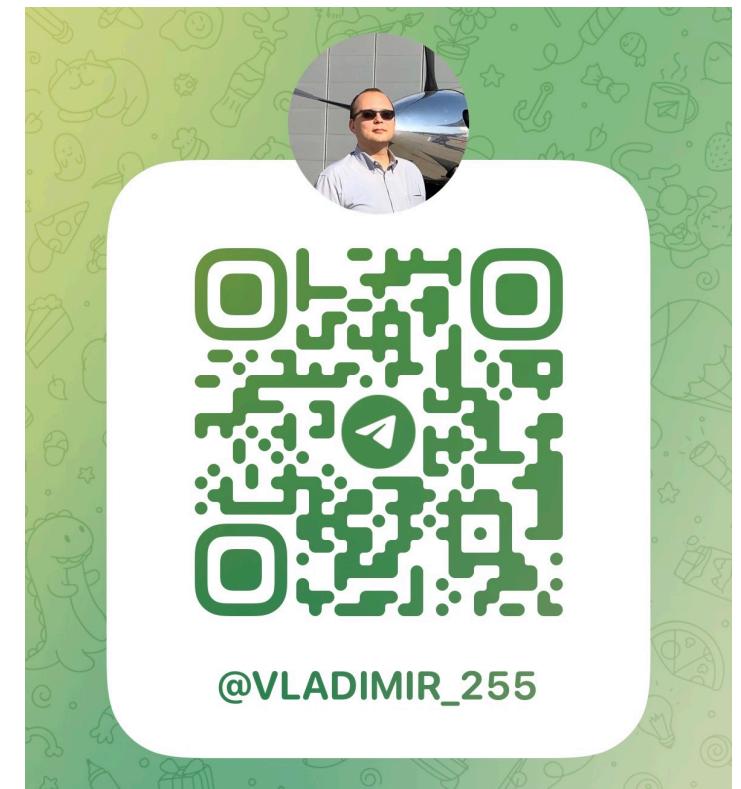


Машинное обучение (краткий курс)

Судаков Владимир Анатольевич,
Профессор кафедры 806
sudakov@ws-dss.com
2025



Содержание

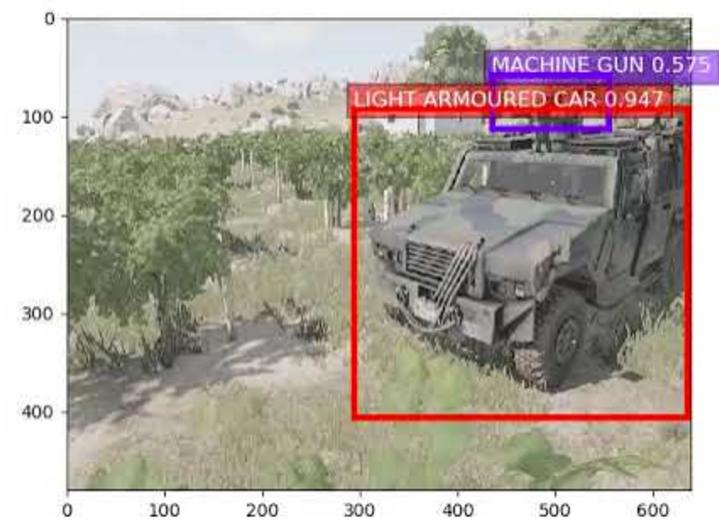
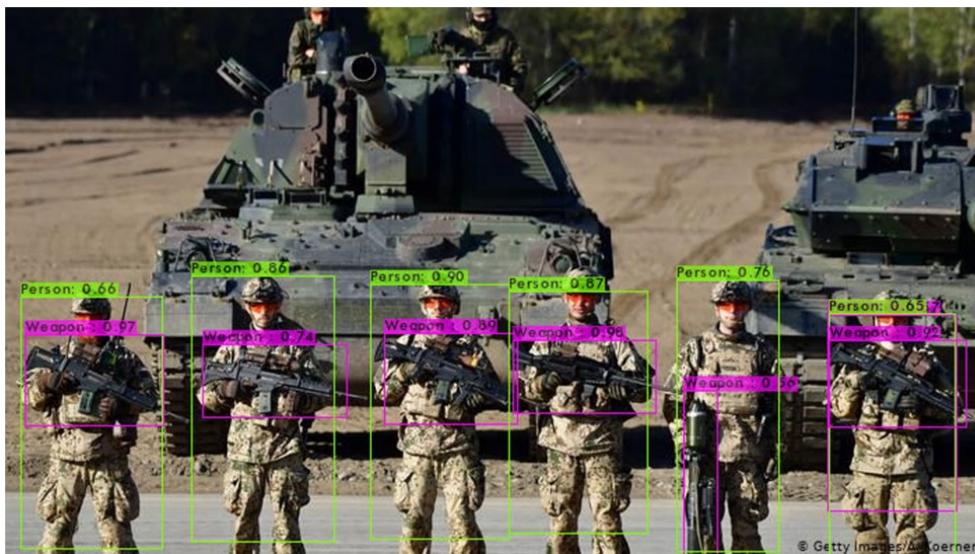
- Определение, классификация
- Линейные модели
 - регрессия
 - классификация
- Ядерные методы
- Деревья решений
- Графовые модели
- EM-алгоритм
- Обучение с подкреплением (маловероятно)

Литература

- Кристофер Бишоп. Распознавание образов и машинное обучение (PRML book)
- Кэвин Мэрфи. Вероятностное машинное обучение: Введение
- Джоэл Грас. Data science. Наука о данных с нуля
- Машинное обучение (курс лекций, К.В.Воронцов)
- Машинное обучение и анализ данных (курс лекций, А.Г.Дьяконов)
- Стюарт Рассел, Питер Норвиг. Искусственный интеллект: современный подход (AIMA-2)
- Судаков В.А., Титов Ю.П. Методы искусственного интеллекта в информационных системах: Учебник — М.: МИРЭА, 2024.
- Саттон Ричард С., Барто Эндрю Г. Обучение с подкреплением
- <https://github.com/sudakov/ai-lab>
- <https://github.com/sudakov/math-for-ds/tree/main/Bayes>

Некоторые популярные на сегодня задачи ИИ

- Обработка естественного языка
- Анализ изображений



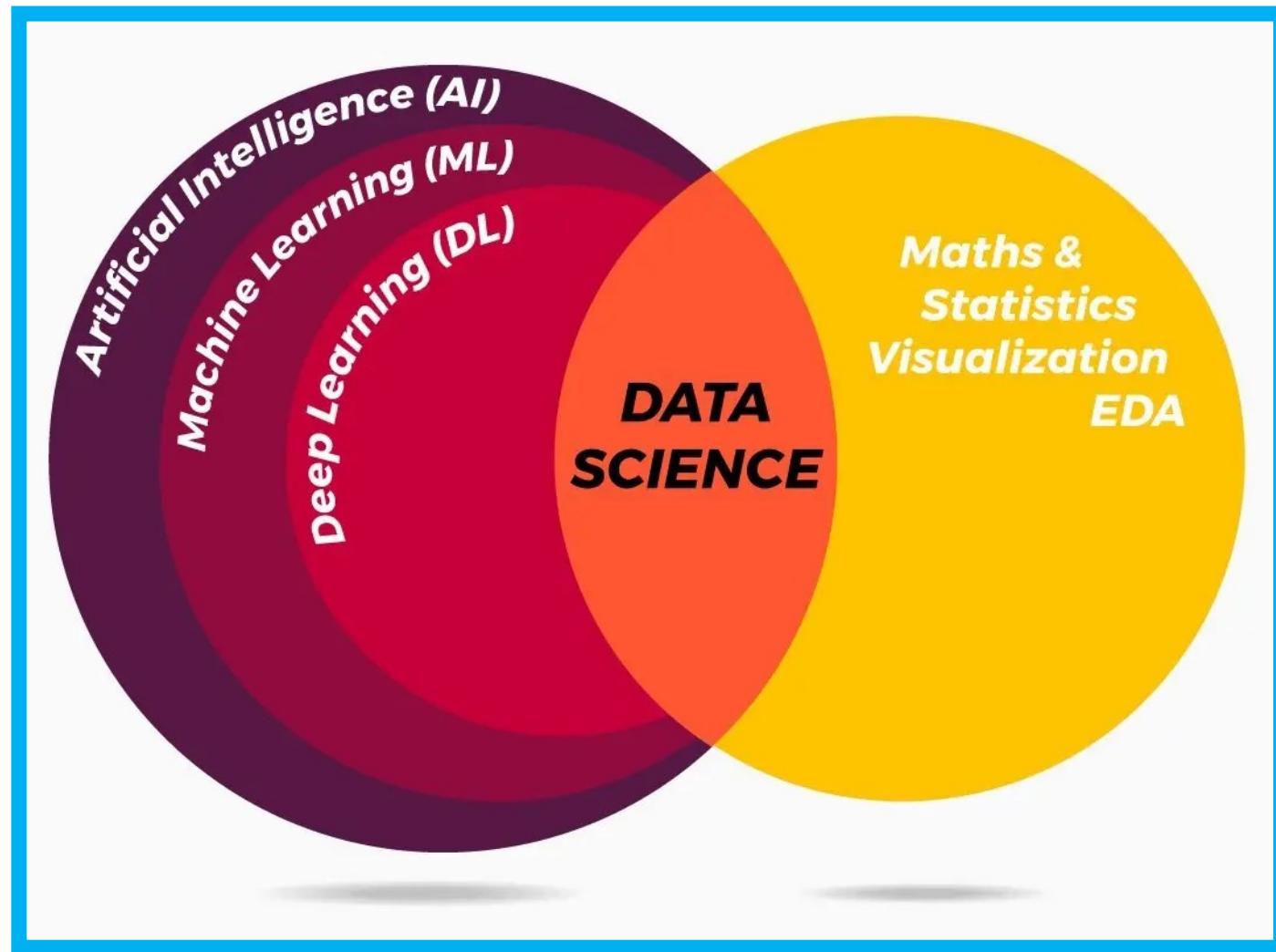
Машинное обучение.

Определение

Говорят, что *компьютерная программа обучается на опыте* E относительно некоторого класса задач T и меры качества P , если ее качество на задачах, принадлежащих T , измеренное в соответствии с P , улучшается с увеличением опыта E .

Алгоритмы машинного обучения создают модель на основе выборочных данных, известных как «обучающие данные», чтобы делать прогнозы или предлагать решения, не будучи явно запрограммированными на это.

Взаимосвязи между науками



Машинное обучение



Отличия

Обучение с учителем (supervised) vs

Обучение без учителя (unsupervised)



Вопрос для обсуждения

Многие методы Data Science и Machine Learning появились достаточно давно - 50-70 года прошлого века, но активно использоваться в бизнесе начали только сейчас

С чем это связано?

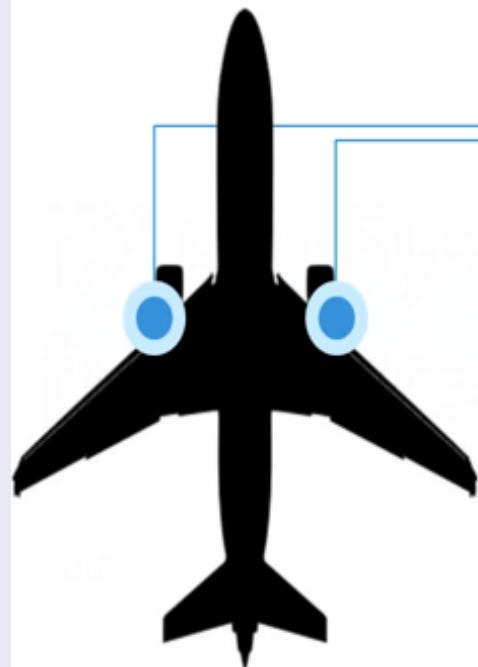
Что такого случилось?

Что есть сейчас и чего не было тогда?

Пример больших данных

Data in USA cross-country commercial flights

Sensor data from a cross-country flight



$$20 \text{ TB} \times 2 \times 6 \times 28,537 \times 365$$

20 terabytes of
information per
engine every hour

twin-engine
Boeing 737

six-hour, cross-
country flight from
New York to Los
Angeles

of commercial
flights in the sky in
the United States on
any given day.

days in a year

$$= 2,499,841,200 \text{ TB}$$

Некоторые задачи машинного обучения

Регрессия

Классификация

Ранжирование

Кластеризация

Понижение размерности

Некоторые задачи

Задача регрессии – прогноз на основе выборки объектов с различными признаками. На выходе - вещественное число (2, 35, 76.454 и др.). Например, цена квартиры, стоимость ценной бумаги через неделю, ожидаемый доход магазина на следующий месяц.

Задача классификации – получение категориального ответа на основе набора признаков. Имеет конечное количество ответов (часто, в формате «да» или «нет»): является ли изображение человеческим лицом, давать ли клиенту кредит, к какой категории отнести товар.

Дано

Множество объектов X .

Множество допустимых ответов Y .

Целевая функция (target function) $y^* : X \rightarrow Y$, значения которой $t_i = y^*(x_i)$ известны только на конечном подмножестве объектов $\{x_1, \dots, x_N\} \subset X$.

Пары «объект– ответ» (x_i, t_i) называются прецедентами.

Совокупность пар $(x_i, t_i)_{i=1}^N$ называется обучающей выборкой (training sample).

Требуется найти

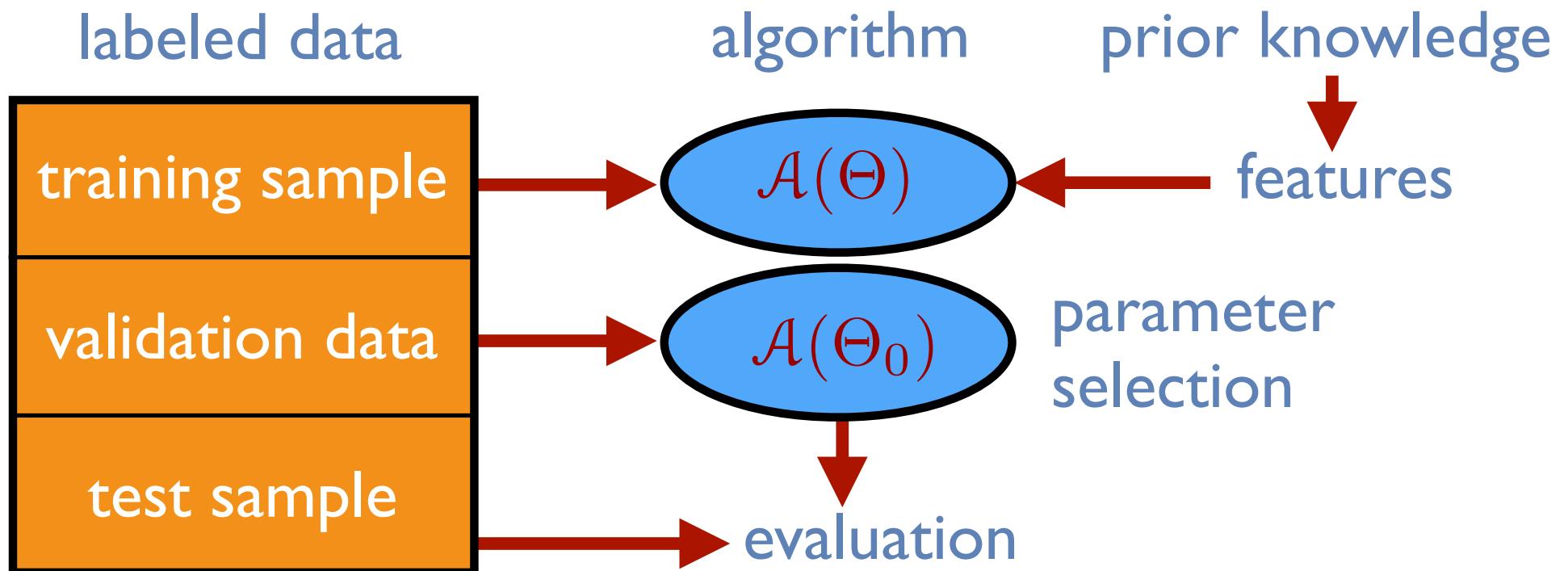
зависимость y^* по выборке $(x_i, t_i)_{i=1}^N$, то есть
построить решающую функцию (decision function)

$$y: X \rightarrow Y,$$

которая приближала бы целевую функцию $y^*(x)$,
причём не только на объектах обучающей выборки,
но и на всём множестве X (или некотором его
подмножестве).

Решающая функция должна допускать эффективную
компьютерную реализацию.

Этапы обучения



Рекомендуемые системы программирования и библиотеки

- Python
- Jupyter Lab
- NumPy
- Pandas
- Scikit-Learn
- XGBoost

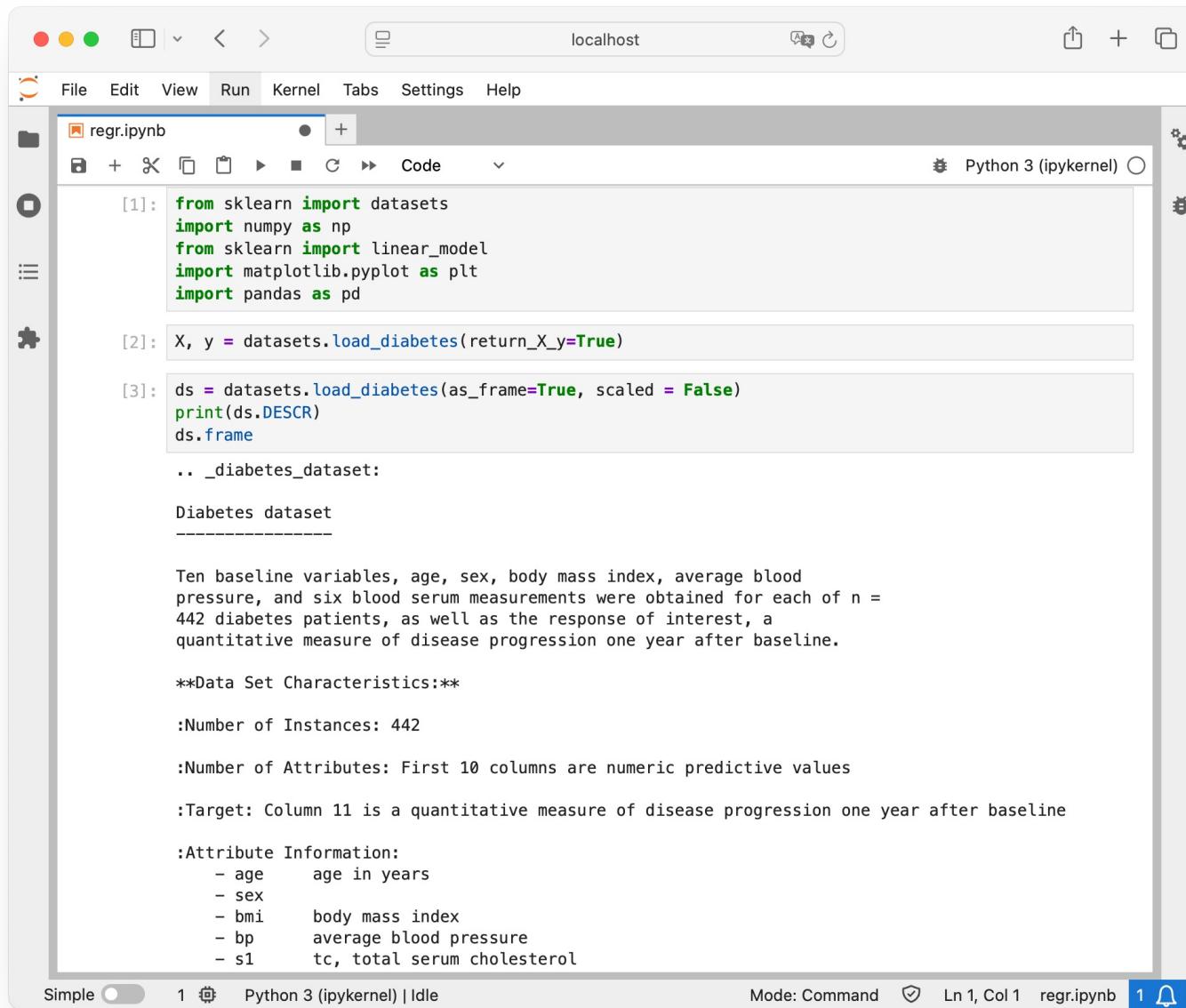


Следующий семестр

- PyTorch, TensorFlow, Transformers



Пример задачи



The screenshot shows a Jupyter Notebook interface with the following code and output:

```
File Edit View Run Kernel Tabs Settings Help
regr.ipynb Python 3 (ipykernel)
[1]: from sklearn import datasets
import numpy as np
from sklearn import linear_model
import matplotlib.pyplot as plt
import pandas as pd

[2]: X, y = datasets.load_diabetes(return_X_y=True)

[3]: ds = datasets.load_diabetes(as_frame=True, scaled = False)
print(ds.DESCR)
ds.frame
... _diabetes_dataset:

Diabetes dataset
-----
Ten baseline variables, age, sex, body mass index, average blood
pressure, and six blood serum measurements were obtained for each of n =
442 diabetes patients, as well as the response of interest, a
quantitative measure of disease progression one year after baseline.

**Data Set Characteristics:**

:Number of Instances: 442

:Number of Attributes: First 10 columns are numeric predictive values

:Target: Column 11 is a quantitative measure of disease progression one year after baseline

:Attribute Information:
- age      age in years
- sex
- bmi     body mass index
- bp      average blood pressure
- s1      tc, total serum cholesterol
```

The notebook is titled "regr.ipynb" and is running in "Python 3 (ipykernel)". The code imports necessary libraries and loads the "diabetes" dataset. The dataset is described as containing ten baseline variables, age, sex, body mass index, average blood pressure, and six blood serum measurements, along with the response of interest, a quantitative measure of disease progression one year after baseline. The target variable is the 11th column. The attribute information includes age (in years), sex, bmi (body mass index), bp (average blood pressure), and s1 (tc, total serum cholesterol).

Пример решения (не идеального)

The screenshot shows a Jupyter Notebook interface with the following details:

- Title Bar:** localhost
- File Menu:** File, Edit, View, Run, Kernel, Tabs, Settings, Help
- Code Cell [3]:** Displays a table of data with columns: age, sex, bmi, bp, s1, s2, s3, s4, s5, s6, target. The table has 442 rows.
- Code Cell [4]:** X_train, X_test = X[:-20], X[-20:]
y_train, y_test = y[:-20], y[-20:]
- Code Cell [5]:** ols = linear_model.LinearRegression()
r = ols.fit(X_train, y_train)
- Code Cell [6]:** ols.predict(X_test) - y_test
- Output Cell [6]:** An array of numerical values.
- Bottom Status Bar:** Simple, Python 3 (ipykernel) | Idle, Mode: Command, Ln 1, Col 1, regr.ipynb, 1

Что нужно
улучшить?

Пример

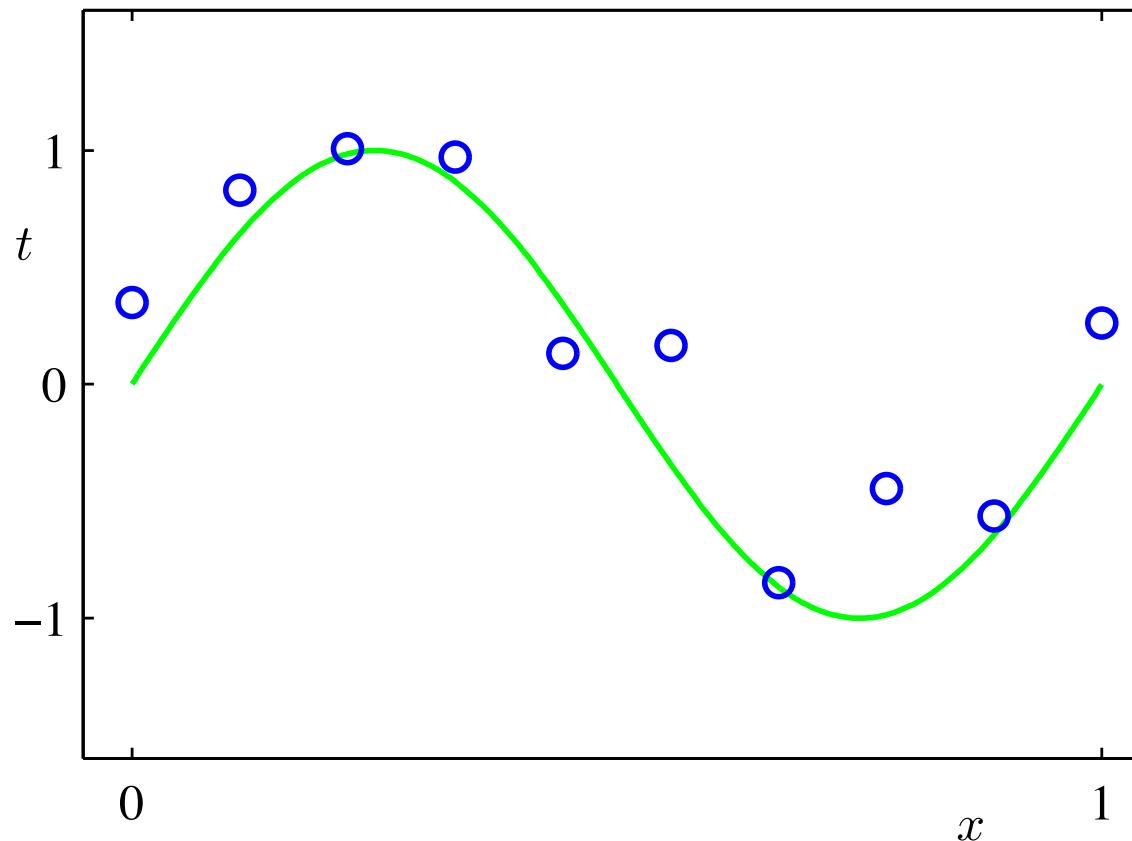
- Рассмотрим синтетически сгенерированный пример набора данных. Данные были получены из функции $\sin(2\pi x)$ путём добавления случайного гауссовского шума со стандартным отклонением $\sigma = 0.3$.
- Мы сгенерировали $N = 10$ наблюдения, равномерно распределенные в диапазоне $[0,1]$. Эти наблюдения составляют вектор входных данных,

$$x = (x_1, \dots, x_N)^T$$

- Для каждого сгенерированного наблюдения x мы получили соответствующее значение функции $\sin(2\pi x)$, а затем добавили случайный шум, чтобы зафиксировать реальную ситуацию с отсутствующей информацией.

$$t = (t_1, \dots, t_N)^T$$

Пример графика



Наша цель — предсказать значение \hat{t} для некоторого **нового** значения \hat{x} , при отсутствии каких-либо данных о зелёной кривой.

Пример

Попытаемся подгонять данные с помощью полиномиальной функции вида

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

где M — порядок полинома.

Такие функции, как $y(x, \mathbf{w})$, являющиеся линейными функциями неизвестных коэффициентов \mathbf{w} , называются *линейными моделями*.

ФУНКЦИЯ ОШИБКИ

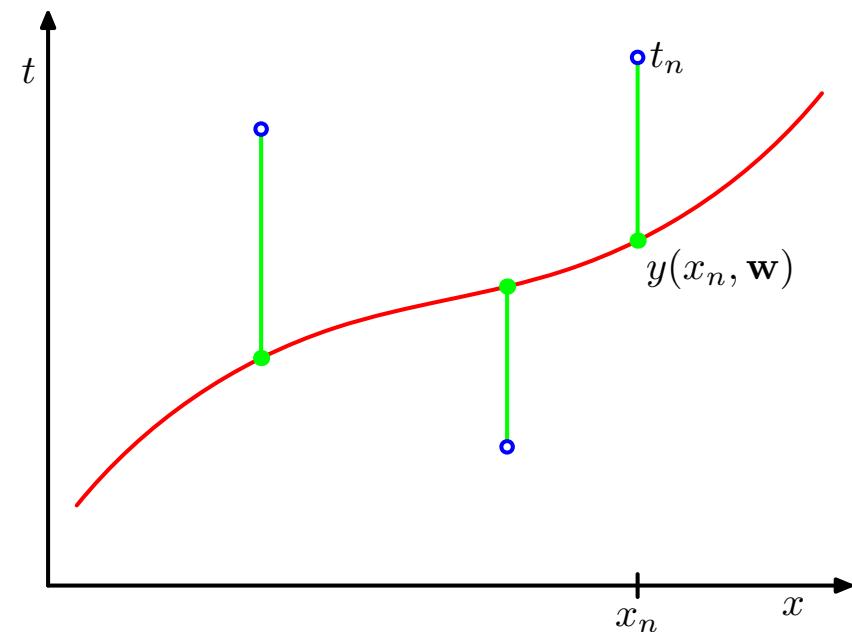
Необходимо определить значения коэффициентов \mathbf{w} , подгоняя полином к обучающим данным. Это можно сделать, минимизировав функцию ошибки, которая измеряет несоответствие между функцией $y(x, \mathbf{w})$, для заданного значения \mathbf{w} , и точками обучающих данных.

Одна простая функция ошибок представляет собой сумму квадратов ошибок между $y(x, \mathbf{w})$ и соответствующими целевыми значениями t_n

$$E(\mathbf{w}) = \frac{1}{2} \sum_{n=1}^N (y(x_n, \mathbf{w}) - t_n)^2 \geq 0$$

где функция становится равной нулю тогда и только тогда, когда функция $y(x, \mathbf{w})$ проходит точно через каждую точку обучающих данных.

Мы можем решить задачу аппроксимации кривой, выбрав значение \mathbf{w} при котором $E(\mathbf{w})$, как можно меньше. Поскольку функция погрешности квадратичная, её производные линейны, и, следовательно, минимизация функции имеет единственное решение в точке, обозначенное как \mathbf{w}^* .



Поиск решения

Для минимизации функции погрешности необходимо вывести вектор градиента, приравнять его к нулю и решить уравнение \mathbf{w}^* следующим образом:

$$\nabla E(\mathbf{w}^*) = \mathbf{0}$$

Сначала нам нужно подставить полином в функцию ошибки

$$E(\mathbf{w}) = \frac{1}{2} \sum_{n=1}^N \sum_{j=0}^M (w_j x_n^j - t_n)^2$$

Обратите внимание, что каждая точка данных из сгенерированного обучающего набора имеет размерность 1, то есть $x \in \mathbb{R}$. Однако полиномиальная функция заполняет M признаков для каждого входного значения x , по сути, преобразуя x в M -мерный вектор.

Таким образом, обучающий набор x можно записать в виде $N \times M$ матрицы $\mathbf{X} = \|x_{nj}\|$, где x_{nj} представляет x_n^j , то есть n -ое входное значение, возведенное в степень j . Чтобы найти вектор градиента, мы берём частную производную E по произвольному w_k .

Дифференцируя сумму почленно, получаем:

$$\begin{aligned} \nabla E(\mathbf{w}^*)_k &= \frac{\partial}{\partial w_k} E(\mathbf{w}) = \frac{1}{2} \sum_{n=1}^N 2 \left(\sum_{j=0}^M w_j x_n^j - t_n \right) x_n^k = \sum_{n=1}^N \left(\sum_{j=0}^M w_j x_n^j - t_n \right) x_n^k = \sum_{n=1}^N (\mathbf{X}\mathbf{w} - \mathbf{t})_n x_{nk} = \sum_{n=1}^N \mathbf{x}_{kn}^\top (\mathbf{X}\mathbf{w} - \mathbf{t})_n \\ &= (\mathbf{X}^\top (\mathbf{X}\mathbf{w} - \mathbf{t}))_k \end{aligned}$$

Используя частную производную для одного компонента, мы вычисляем вектор градиента по k индексу. Таким образом, \mathbf{w}^* должно удовлетворять

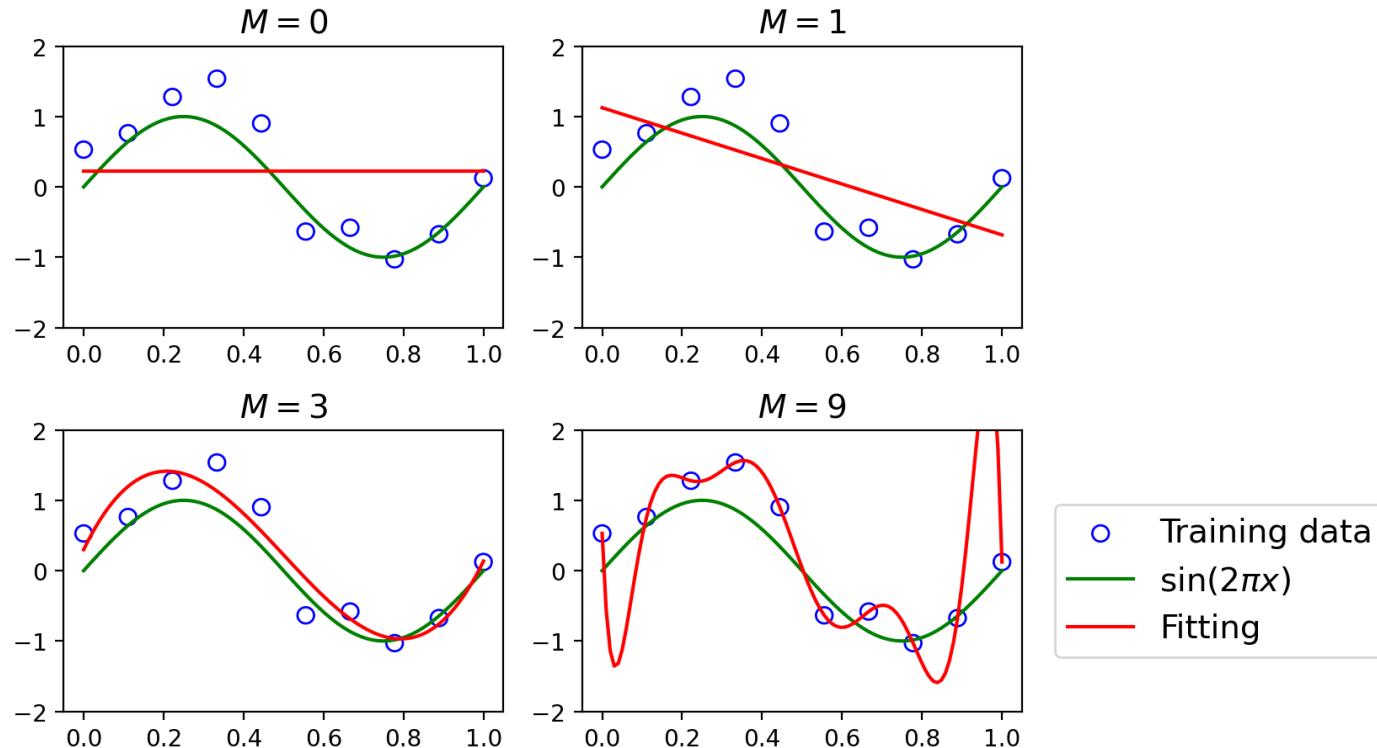
$$\nabla E(\mathbf{w}^*) = \mathbf{X}^\top (\mathbf{X}\mathbf{w}^* - \mathbf{t}) = \mathbf{0}$$

Решение \mathbf{w}^* дает единственное решение задачи подгонки кривой

$$\mathbf{X}^\top (\mathbf{X}\mathbf{w}^* - \mathbf{t}) = \mathbf{0} \Leftrightarrow \mathbf{X}^\top \mathbf{X}\mathbf{w}^* = \mathbf{X}^\top \mathbf{t} \Leftrightarrow \mathbf{w}^* = (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{t}$$

Полученный многочлен задается функцией $y(x, \mathbf{w}^*)$.

Выбор модели

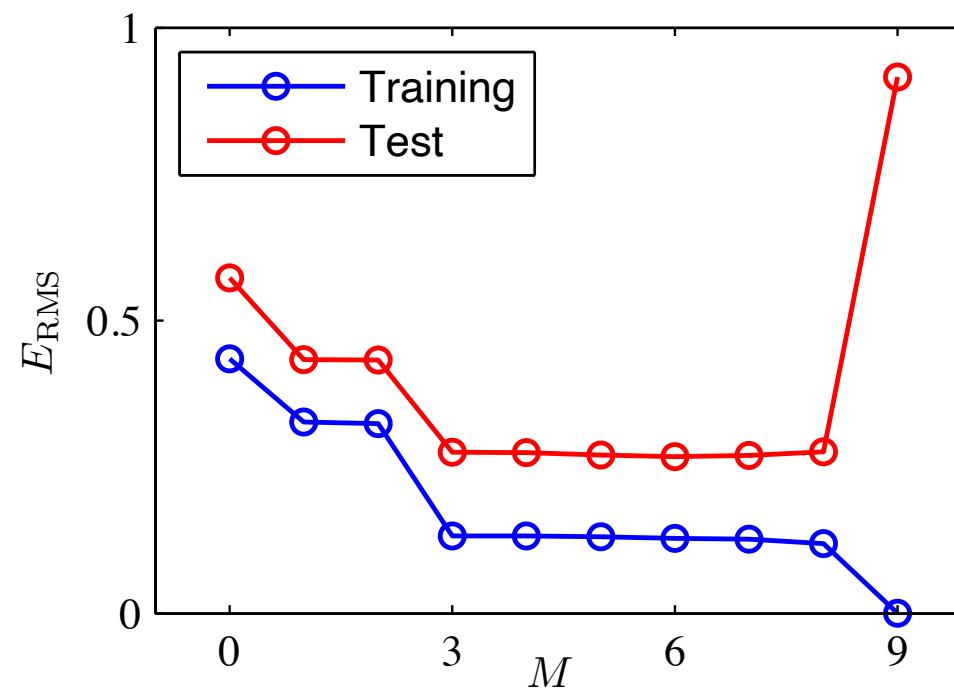


Обратите внимание, что константа ($M = 0$) и полиномы первого порядка ($M = 1$) дают довольно плохое соответствие данным. Полином третьего порядка ($M = 3$) даёт, по-видимому, наилучшее соответствие, в то время как полином более высокого порядка ($M = 9$) даёт отличное соответствие данным, то есть $E(\mathbf{w}^*) = \mathbf{0}$. Однако подобранная кривая плохо отражает базовую функцию $\sin(2\pi x)$. Это явление известно как *переобучение*.

RMS

Более количественное представление об эффективности обобщения M можно получить, используя среднеквадратичную ошибку (RMS), определяемую как

$$E_{RMS} = \sqrt{2 \frac{E(\mathbf{w}^*)}{N}}$$



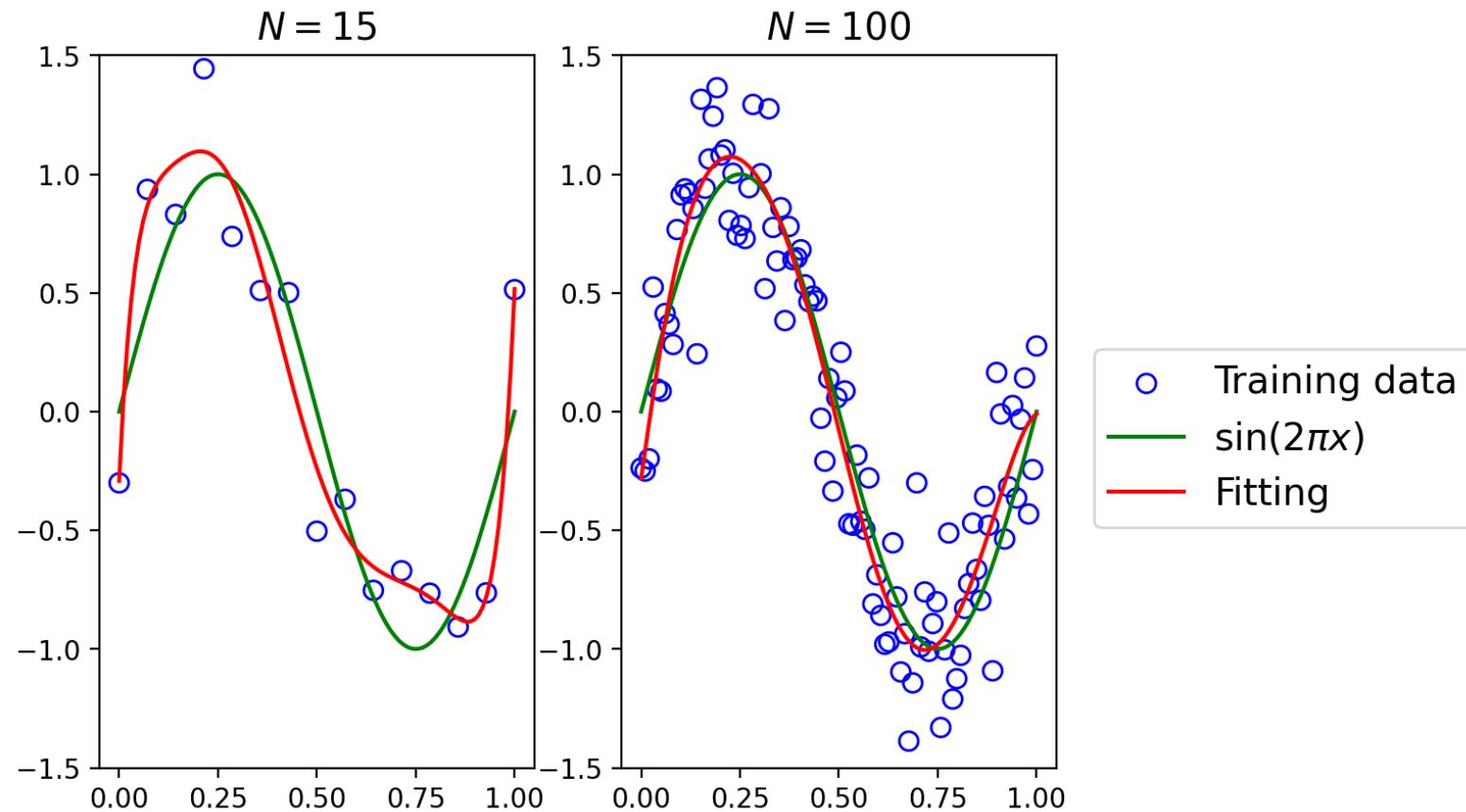
Кажется, что модель не должна работать хуже с увеличением M

Таблица коэффициентов

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

Увеличение объема данных

Полином 9-й степени



Регуляризация

Одним из часто используемых методов контроля явления переобучения является *регуляризация*, которая добавляет штрафной член к функции ошибки, чтобы предотвратить достижение коэффициентами больших значений. Простейшим таким штрафным членом является сумма квадратов всех коэффициентов, что приводит к модифицированной функции ошибки следующего вида:

$$\tilde{E}(\mathbf{w}) = \frac{1}{2} \sum_{n=1}^N (y(x_n, \mathbf{w}) - t_n)^2 + \frac{\lambda}{2} \|\mathbf{w}\|^2$$

Такие методы известны как методы сжатия, поскольку они уменьшают значение коэффициентов. Частный случай квадратичной регуляризации называется *гребневой регрессией* (ridge regression). В нейронных сетях этот подход также известен как *редукция весов* (weight decay).

Часто коэффициент w_0 исключается из регуляризатора, потому что из-за его включения результаты зависят от выбора начала координат для целевой переменной

Минимизация

Функцию ошибки гребня можно точно минимизировать следующим образом:

$$\begin{aligned}\nabla E(\mathbf{w}^*)_k &= \frac{\partial}{\partial w_k} E(\mathbf{w}) \\ &= \frac{1}{2} \sum_{n=1}^N 2 \left(\sum_{j=0}^M w_j x_n^j - t_n \right) x_n^k + \frac{1}{2} \lambda 2 w_k \\ &= \sum_{n=1}^N \left(\sum_{j=0}^M w_j x_n^j - t_n \right) x_n^k + \lambda w_k \\ &= \sum_{n=1}^N (\mathbf{X}\mathbf{w} - \mathbf{t})_n \mathbf{X}_{nk} + \lambda w_k = \sum_{n=1}^N \mathbf{X}_{kn}^\top (\mathbf{X}\mathbf{w} - \mathbf{t})_n + \lambda w_k \\ &= (\mathbf{X}^\top (\mathbf{X}\mathbf{w} - \mathbf{t}))_k + \lambda w_k\end{aligned}$$

Используя частную производную для одного компонента, мы вычисляем вектор градиента, отбрасывая k нижний индекс. Тогда \mathbf{w}^* должен удовлетворять

$$\nabla E(\mathbf{w}^*) = \mathbf{X}^\top (\mathbf{X}\mathbf{w}^* - \mathbf{t}) + \lambda \mathbf{w}^* \mathbf{I} = \mathbf{0}$$

Решение \mathbf{w}^* дает единственное решение, которое минимизирует ошибку гребня

$$\begin{aligned}\mathbf{X}^\top (\mathbf{X}\mathbf{w}^* - \mathbf{t}) + \lambda \mathbf{w}^* \mathbf{I} &= \mathbf{0} \Leftrightarrow \\ \mathbf{X}^\top \mathbf{X}\mathbf{w}^* - \mathbf{X}^\top \mathbf{t} + \lambda \mathbf{w}^* \mathbf{I} &= \mathbf{0} \Leftrightarrow \\ \mathbf{X}^\top \mathbf{X}\mathbf{w}^* + \lambda \mathbf{w}^* \mathbf{I} &= \mathbf{X}^\top \mathbf{t} \Leftrightarrow \\ \mathbf{w}^* (\mathbf{X}^\top \mathbf{X} + \lambda \mathbf{I}) &= \mathbf{X}^\top \mathbf{t} \Leftrightarrow \\ \mathbf{w}^* &= (\mathbf{X}^\top \mathbf{X} + \lambda \mathbf{I})^{-1} \mathbf{X}^\top \mathbf{t}\end{aligned}$$

Результаты для полинома $M=9$

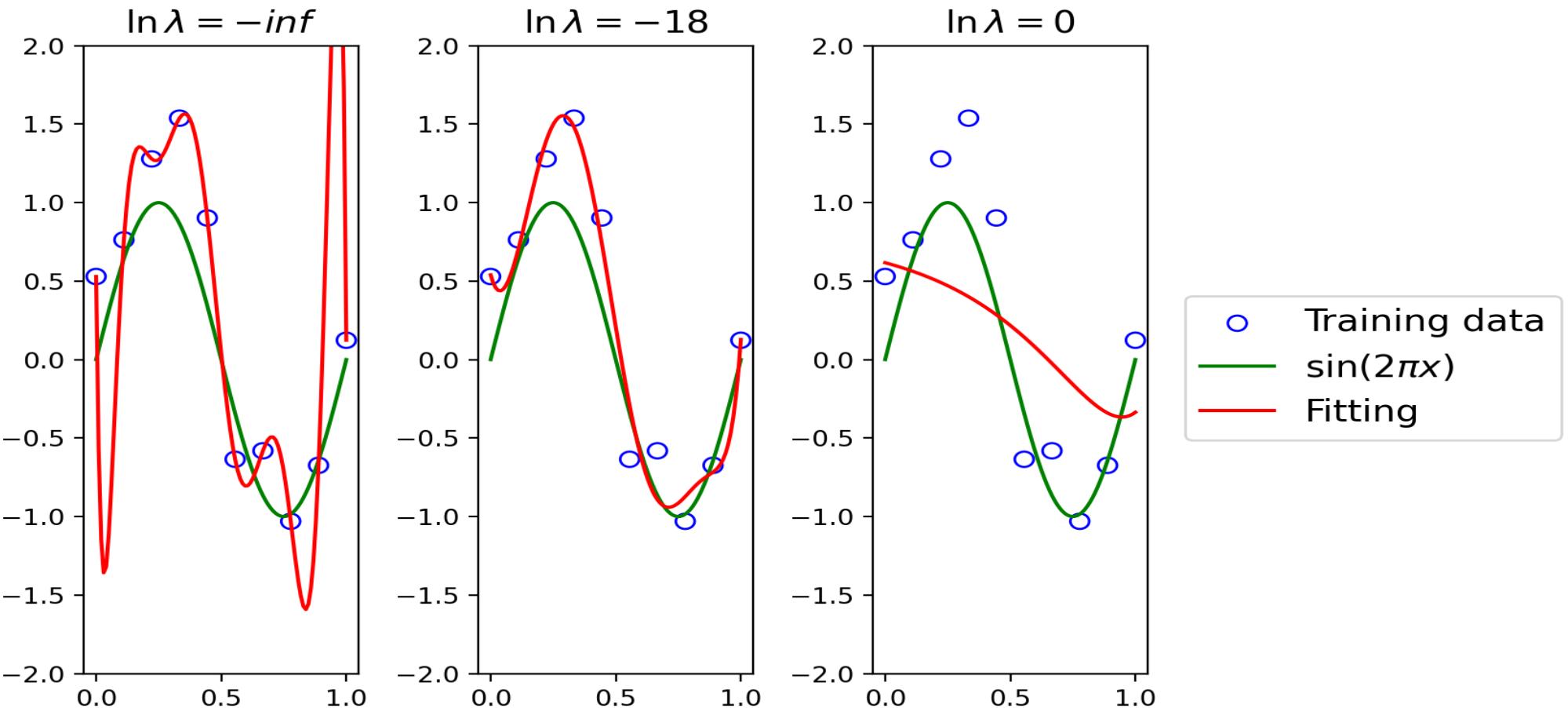
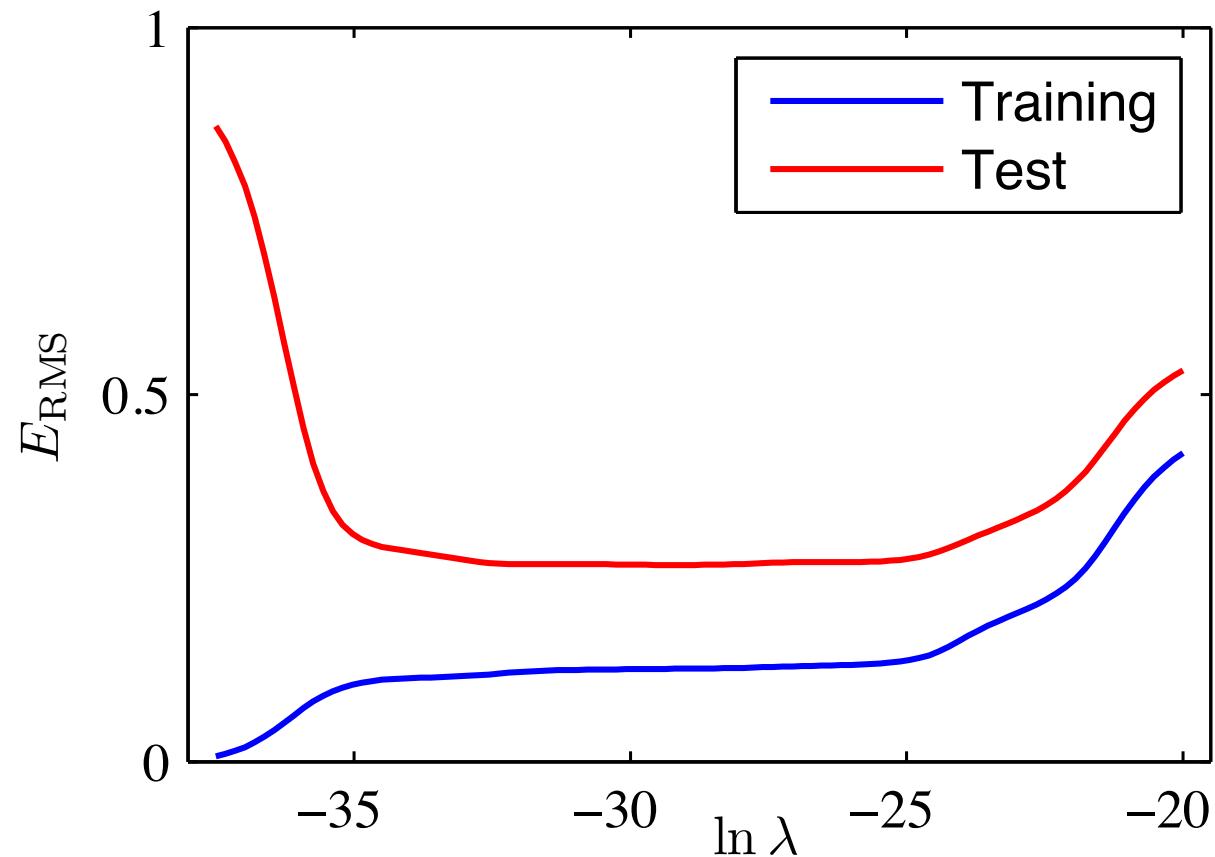


Таблица коэффициентов

	$\ln\lambda = -\infty$	$\ln\lambda = -18$	$\ln\lambda = 0$
w0	0.39	0.38	0.36
w1	-135.04	-2.14	-0.46
w2	3206.76	81.88	-0.39
w3	-29215.92	-390.51	-0.22
w4	139594.34	578.12	-0.05
w5	-388863.80	-31.48	0.07
w6	652373.24	-49.12	0.17
w7	-648124.69	-28.57	0.25
w8	350721.94	540.17	0.31
w9	-79556.29	-255.65	0.35

График среднеквадратической ошибки в зависимости от $\ln \lambda$ для полинома
степени $M = 9$



Теорема Байеса в ML

Пусть \mathbf{w} — параметры и \mathcal{D} — данные. Теорема Байеса задаётся формулой:

$$p(\mathbf{w}|\mathcal{D}) = \frac{p(\mathcal{D}|\mathbf{w})p(\mathbf{w})}{p(\mathcal{D})} \Leftrightarrow \text{posterior} = \frac{\text{likelihood} \times \text{prior}}{\text{evidence}}$$

Частотная парадигма обычно количественно оценивает свойства величин, определяемых данными, в свете фиксированных параметров модели, тогда как байесовская парадигма обычно количественно оценивает свойства неизвестных параметров модели в свете наблюдаемых данных.

$$\text{posterior} \propto \text{likelihood} \times \text{prior}$$

$p(\mathcal{D})$ можно найти из условия: $\int p(\mathbf{w}|\mathcal{D})d\mathbf{w} = 1 \Rightarrow p(\mathcal{D}) = \int p(\mathcal{D}|\mathbf{w})p(\mathbf{w})d\mathbf{w}$

Оценка правдоподобия

- Широко используемой частотной оценкой $p(\mathcal{D}|\mathbf{w})$ является оценка максимального правдоподобия, при котором значение \mathbf{w} максимизирует функцию правдоподобия $p(\mathcal{D}|\mathbf{w})$. Это соответствует выбору значения \mathbf{w} , для которого вероятность наблюдаемого набора данных максимизируется. В литературе по машинному обучению отрицательная логарифмическая функция правдоподобия называется функцией ошибок. Поскольку отрицательный логарифм является монотонно убывающей функцией, максимизация правдоподобия эквивалентна минимизации ошибки.
- Одним из подходов к определению частотной величины ошибки является бутстрэп, в котором несколько наборов данных создаются следующим образом. Предположим, наш исходный набор данных состоит из N точек данных $X = \{x_1, \dots, x_N\}$. Мы можем создать новый набор данных X_b , извлекая N точек случайным образом из набора X с возвращением, так что некоторые точки из X могут повторяться в X_b , тогда как другие точки в X могут отсутствовать в X_b . Этот процесс может быть повторен L раз для генерации L наборов данных, содержащих по N точек, и каждый из них получается путем выбора из исходного набора данных.
- Статистическую точность оценок параметров можно определить, посмотрев на изменчивость предсказаний между различными наборами данными.

Задача +1 балл

Монета бросается три раза и каждый раз выпадает орел. Какова вероятность получить орла в четвертом броске?

Не стесняйтесь задавать уточняющие вопросы от них многое зависит.

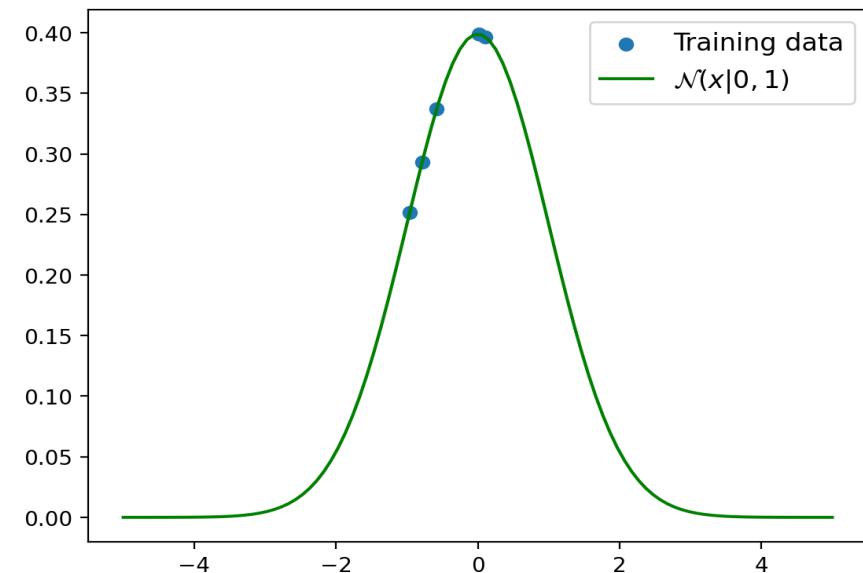
Нормальное распределение

$$\begin{aligned}\mathcal{N}(x|\mu, \sigma^2) \\ = \frac{1}{(2\pi\sigma^2)^{1/2}} \exp\left\{-\frac{1}{2\sigma^2}(x - \mu)^2\right\}\end{aligned}$$

Предположим, у нас есть набор данных наблюдений, $x = (x_1, \dots, x_N)^T$ полученных независимо из гауссовского распределения. Точки данных, полученные независимо из одного и того же распределения, называются независимыми и одинаково распределёнными.

Поскольку данные независимы, функция правдоподобия одномерного гауссовского распределения выглядит следующим образом:

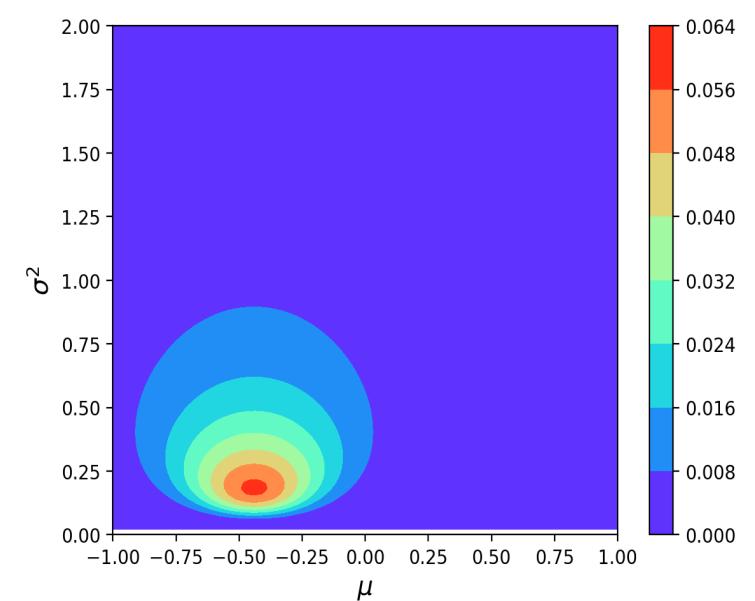
$$p(x|\mu, \sigma^2) = \prod_{n=1}^N \mathcal{N}(x_n|\mu, \sigma^2)$$



Нахождение параметров μ, σ^2

Для нахождения неизвестных параметров μ и σ^2 необходимо использовать наблюдаемый набор данных и найти значения параметров, максимизирующие функцию правдоподобия. Логарифмическую функцию правдоподобия можно записать следующим образом:

$$\begin{aligned} \ln p(x|\mu, \sigma^2) &= \ln[\prod_{n=1}^N \mathcal{N}(x_n|\mu, \sigma^2)] \\ &= \sum_{n=1}^N \ln \mathcal{N}(x_n|\mu, \sigma^2) \\ &= \sum_{n=1}^N \ln \left(\frac{1}{(2\pi\sigma^2)^{1/2}} \exp \left\{ -\frac{1}{2\sigma^2} (x_n - \mu)^2 \right\} \right) \\ &= \sum_{n=1}^N \ln \left(\frac{1}{(2\pi\sigma^2)^{1/2}} \right) + \sum_{n=1}^N \ln \left(\exp \left\{ -\frac{1}{2\sigma^2} (x_n - \mu)^2 \right\} \right) \\ &= N \ln \left(\frac{1}{(2\pi\sigma^2)^{1/2}} \right) - \sum_{n=1}^N \frac{1}{2\sigma^2} (x_n - \mu)^2 \\ &= N \ln 1 - N \ln (2\pi\sigma^2)^{1/2} - \frac{1}{2\sigma^2} \sum_{n=1}^N (x_n - \mu)^2 \\ &\stackrel{\ln 1 = 0}{=} -N \ln (2\pi\sigma^2)^{1/2} - \frac{1}{2\sigma^2} \sum_{n=1}^N (x_n - \mu)^2 \\ &\stackrel{\ln x^y = y \ln x}{=} -\frac{N}{2} \ln 2\pi\sigma^2 - \frac{1}{2\sigma^2} \sum_{n=1}^N (x_n - \mu)^2 \\ &= -\frac{N}{2} \ln 2\pi - \frac{N}{2} \ln \sigma^2 - \frac{1}{2\sigma^2} \sum_{n=1}^N (x_n - \mu)^2 \end{aligned}$$



Отрицательный логарифм функции правдоподобия называется функцией ошибки.

Поскольку отрицательный логарифм — монотонно убывающая функция, максимизация правдоподобия эквивалентна минимизации ошибки.

Нахождение параметров μ, σ^2

$$\frac{\partial \ln p(\mathbf{x}|\mu, \sigma^2)}{\partial \mu} = 0 \Leftrightarrow$$

$$\frac{\partial}{\partial \mu} \left(\frac{N}{2} \ln 2\pi - \frac{N}{2} \ln \sigma^2 - \frac{1}{2\sigma^2} \sum_{n=1}^N (x_n - \mu)^2 \right) = 0 \Leftrightarrow$$

$$\frac{\partial}{\partial \mu} \left(\frac{1}{2\sigma^2} \sum_{n=1}^N (x_n - \mu)^2 \right) = 0 \Leftrightarrow$$

$$\frac{\partial}{\partial \mu} \sum_{n=1}^N (x_n - \mu)^2 = 0 \Leftrightarrow$$

$$\sum_{n=1}^N (2\mu - 2x_n) = 0 \Leftrightarrow 2 \sum_{n=1}^N (\mu - x_n) = 0 \Leftrightarrow$$

$$N\mu - \sum_{n=1}^N x_n = 0 \Leftrightarrow N\mu = \sum_{n=1}^N x_n \Leftrightarrow$$

$$\mu_{ML} = \frac{1}{N} \sum_{n=1}^N x_n$$

$$\frac{\partial \ln p(\mathbf{x}|\mu, \sigma^2)}{\partial \sigma^2} = 0 \Leftrightarrow$$

$$\frac{\partial}{\partial \sigma^2} \left(\frac{N}{2} \ln 2\pi - \frac{N}{2} \ln \sigma^2 - \frac{1}{2\sigma^2} \sum_{n=1}^N (x_n - \mu)^2 \right) = 0 \Leftrightarrow$$

$$\frac{\partial}{\partial \sigma^2} \left(-\frac{1}{2\sigma^2} \sum_{n=1}^N (x_n - \mu)^2 - \frac{N}{2} \ln \sigma^2 \right) = 0 \Leftrightarrow$$

$$\frac{\partial}{\partial \sigma^2} \left(-\frac{1}{2\sigma^2} \sum_{n=1}^N (x_n - \mu)^2 \right) + \frac{\partial}{\partial \sigma^2} \left(-\frac{N}{2} \ln \sigma^2 \right) = 0 \Leftrightarrow$$

$$\frac{\partial}{\partial \sigma^2} \left(-(2\sigma^2)^{-1} \sum_{n=1}^N (x_n - \mu)^2 \right) + -\frac{N}{2\sigma^2} = 0 \Leftrightarrow$$

$$\frac{1}{4\sigma^4} \sum_{n=1}^N (x_n - \mu)^2 - \frac{N}{2\sigma^2} = 0 \Leftrightarrow$$

$$\frac{1}{4\sigma^4} \sum_{n=1}^N (x_n - \mu)^2 = \frac{N}{2\sigma^2} \Leftrightarrow$$

$$\sigma_{ML}^2 = \frac{1}{N} \sum_{n=1}^N (x_n - \mu_{ML})^2$$

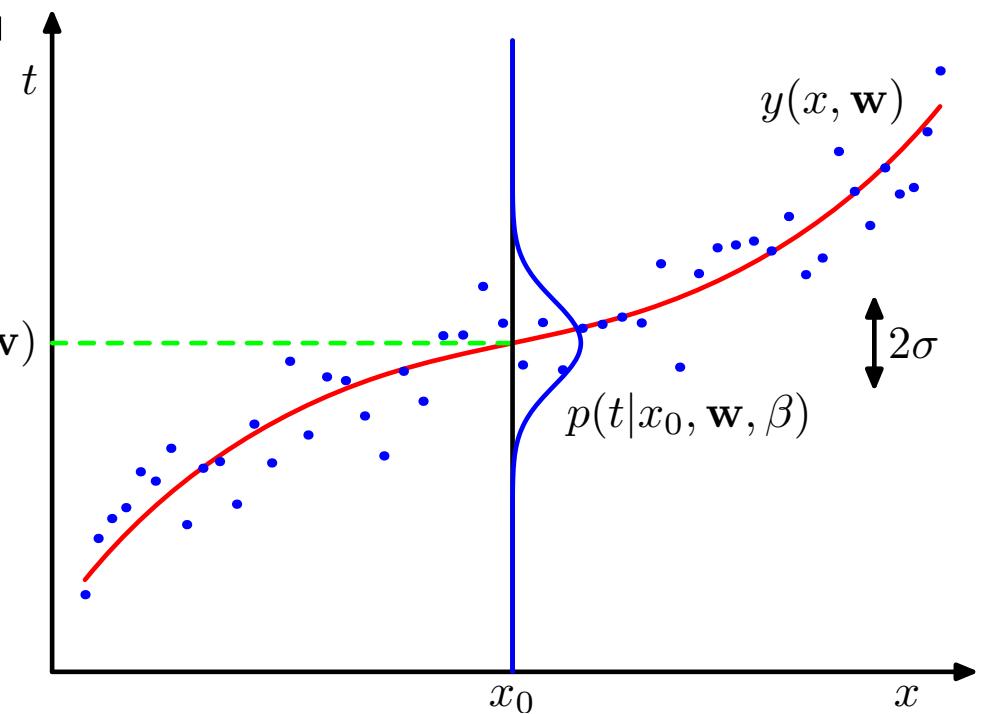
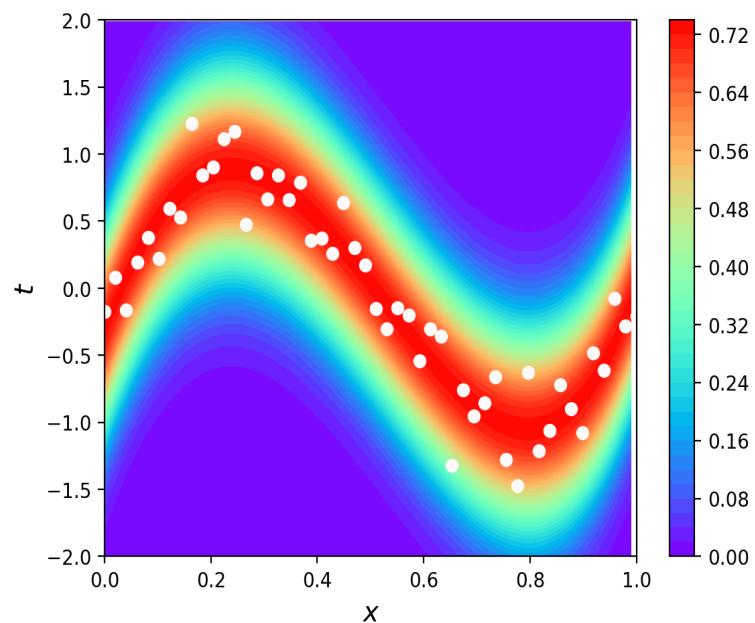
Проблема, возникающая в контексте наших решений для метода максимального правдоподобия, заключается в том, что он систематически недооценивает дисперсию гауссовского распределения. Это пример явления, называемого смещением, и связано с проблемой переобучения, возникающей в контексте полиномиальной подгонки кривой. $\tilde{\sigma}^2 = \frac{N}{N-1} \sigma_{ML}^2 = \frac{1}{N-1} \sum_{n=1}^N (x_n - \mu_{ML})^2$

Аппроксимация кривой

Предположим, что целевая переменная t имеет гауссовское распределение со средним значением $y(x, \mathbf{w})$:

$$p(t|x, \mathbf{w}, \beta) = \mathcal{N}(t|y(x, \mathbf{w}), \beta^{-1})$$

где β параметр, соответствующий обратной дисперсии.



Аппроксимация кривой

Используем обучающие данные x, t для определения значений неизвестных параметров w, β методом максимального правдоподобия. Если данные являются независимыми, функция правдоподобия задаётся следующим образом:

$$p(t|x, w, \beta) = \prod_{n=1}^N \mathcal{N}(t_n | y(x, w), \beta^{-1})$$

Подобно гауссовскому распределению ранее, мы максимизируем логарифм функции правдоподобия в форме

$$\ln p(t|x, w, \beta) = -\frac{\beta}{2} \sum_{n=1}^N (y(x, w) - t_n)^2 + \frac{N}{2} \ln \beta - \frac{N}{2} \ln 2\pi$$

Для коэффициентов полинома w_{ML} можно опустить члены, не зависящие от w . Поскольку β не меняет положения максимума функции, её можно заменить на 1. Следовательно, максимизация эквивалентна минимизации функции ошибки суммы квадратов, как показано в примере аппроксимации кривой полиномом. Функция ошибки суммы квадратов возникла как следствие максимизации правдоподобия в предположении гауссовского распределения шума.

Определение β

$$\begin{aligned}\frac{\partial}{\partial \beta} \ln p(t|x, \mathbf{w}, \beta) = 0 &\Leftrightarrow \\ \frac{\partial}{\partial \beta} \left[-\frac{\beta}{2} \sum_{n=1}^N (y(x, \mathbf{w}) - t_n)^2 + \frac{N}{2} \ln \beta - \frac{N}{2} \ln 2\pi \right] &= 0 \Leftrightarrow \\ \frac{\partial}{\partial \beta} \left[-\frac{\beta}{2} \sum_{n=1}^N (y(x, \mathbf{w}) - t_n)^2 \right] + \frac{\partial}{\partial \beta} \left[\frac{N}{2} \ln \beta \right] &= 0 \Leftrightarrow \\ -\frac{1}{2} \sum_{n=1}^N (y(x, \mathbf{w}) - t_n)^2 + \frac{N}{2} \frac{1}{\beta} &= 0 \Leftrightarrow \\ \sum_{n=1}^N (y(x, \mathbf{w}) - t_n)^2 &= \frac{N}{\beta} \Leftrightarrow \\ \frac{1}{\beta_{ML}} &= \frac{1}{N} \sum_{n=1}^N (y(x, \mathbf{w}) - t_n)^2\end{aligned}$$

Поскольку у нас есть вероятностная модель, мы можем использовать предсказательное распределение, которое дает распределение вероятностей по t , а не простую точечную оценку.

$$p(t|x, \mathbf{w}_{ML}, \beta_{ML}) = \mathcal{N}(t|y(x, \mathbf{w}_{ML}), \beta_{ML}^{-1})$$

Можно дополнительно ввести априорное распределение для коэффициентов полинома \mathbf{w} , чтобы сделать шаг к более байесовскому подходу.

Априорное распределение для коэффициентов полинома \mathbf{w}

Рассмотрим, например, гауссовское априорное распределение вида

$$p(\mathbf{w}|\alpha) = \mathcal{N}(\mathbf{w}|\mathbf{0}, \alpha^{-1}\mathbf{I}) = \left(\frac{\alpha}{2\pi}\right)^{(M+1)/2} \exp\left\{-\frac{\alpha}{2} \mathbf{w}^T \mathbf{w}\right\}$$

где α —точность многомерного гауссовского распределения, а M —порядок полинома, то есть размерность вектора параметров \mathbf{w} . Тогда из теоремы Байеса следует, что

$$p(\mathbf{w}|x, t, \alpha, \beta) \propto p(t|x, \mathbf{w}, \beta)p(\mathbf{w}|\alpha)$$

Следовательно, можно определить \mathbf{w} , максимизируя апостериорное распределение. Этот метод известен как *метод максимального апостериорного распределения*(MAP) или MAP-вывод. Максимум апостериорного распределения определяется минимумом отрицательного логарифма от $p(\mathbf{w}|x, t, \alpha, \beta)$:

$$\frac{\beta}{2} \sum_{n=1}^N (y(x, \mathbf{w}) - t_n)^2 + \frac{\alpha}{2} \mathbf{w}^T \mathbf{w}$$

Таким образом, мы видим, что максимизация апостериорного распределения эквивалентна минимизации регуляризованной суммы квадратов функции ошибок, встречающейся при регуляризации, где $\lambda = \alpha/\beta$.

Байесовская аппроксимация кривой

$$p(t|x, \mathbf{x}, \mathbf{t}) = \int p(t|x, \mathbf{w})p(\mathbf{w}|x, \mathbf{t})d\mathbf{w}$$

Интегрирование в данной формуле можно выполнить аналитически, в результате чего прогностическое распределение принимает вид нормального распределения:

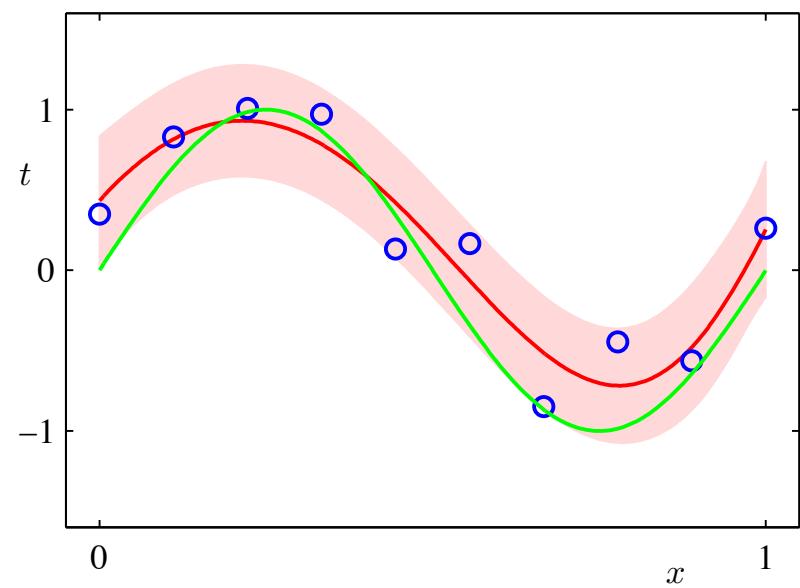
$$p(t|x, \mathbf{x}, \mathbf{t}) = N(t|m(x), s^2(x))$$

$$m(x) = \beta \phi(x)^T S \sum_{n=1}^N \phi(x_n) t_n$$

$$s^2(x) = \beta^{-1} + \phi(x)^T S \phi(x)$$

$$S^{-1} = \alpha \mathbf{I} + \beta \sum_{n=1}^N \phi(x_n) \phi(x_n)^T$$

$$\phi(x) = (x^i : i = \overline{0, M})$$



Выбор модели

В примере, посвященном аппроксимации полиномиальных кривых с использованием метода наименьших квадратов, существует оптимальный порядок полинома, который дает наилучшее обобщение.

Порядок полинома определяет количество свободных параметров в модели и тем самым определяет ее сложность.

В регуляризованном методе наименьших квадратов на реальную сложность модели влияет коэффициент регуляризации в то время как для более сложных моделей, таких как смеси распределений или нейронные сети, может существовать множество параметров, определяющих их сложность.

Для практического применения нам необходимо определить значения таких параметров, и главная цель при этом - достижение наилучших прогностических характеристик параметров сложности в данной модели.

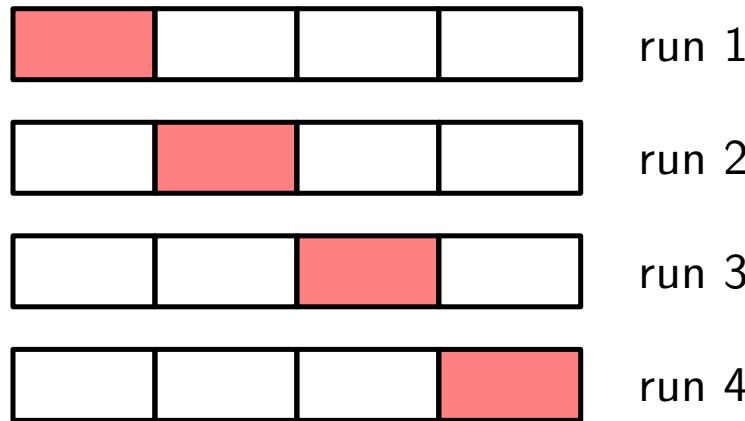
Следует рассмотреть ряд различных видов моделей, пытаясь найти лучшую для конкретного приложения.

No free lunch theorem

Все модели неверны, но некоторые из них полезны. — Джордж Бокс

В литературе описано множество самых разных моделей, поэтому естественно спросить, какая из них наилучшая. К сожалению, не существует одной модели, которая была бы оптимальна для всех задач, — иногда этот факт называют **теоремой об отсутствии бесплатных завтраков**. Причина в том, что набор допущений (иногда называемый **индуктивным смещением**), который хорошо работает в одной предметной области, может давать никуда не годные результаты в другой. При выборе подходящей модели лучше всего опираться на знание предметной области и (или) на метод проб и ошибок (т. е. применять такие способы выбора модели, как перекрестная проверка или байесовские методы). Поэтому так важно иметь в своем арсенале много моделей и алгоритмических методов.

Перекрёстная проверка



- Метод S -групповой перекрестной проверки, проиллюстрированный здесь для случая $S = 4$, включает в себя получение доступных данных и разбиение их на S групп (в простейшем случае они имеют одинаковый размер). Затем $S - 1$ групп используются для обучения набора моделей, а оценка точности осуществляется на оставшейся группе. После этого описанная процедура повторяется для всех возможных вариантов выбора S групп, обозначенных здесь красным цветом, а показатели точности, полученные в результате S сеансов обучения, усредняются
- Если данных мало то можно взять $S=N$
- Если параметров много, то изучение комбинаций таких параметров могло бы в худшем случае потребовать нескольких сеансов обучения, количество которых экспоненциально растет при увеличении количества параметров.

Штраф за сложность

Информационный критерий Акаике, или (Akaike, AIC):

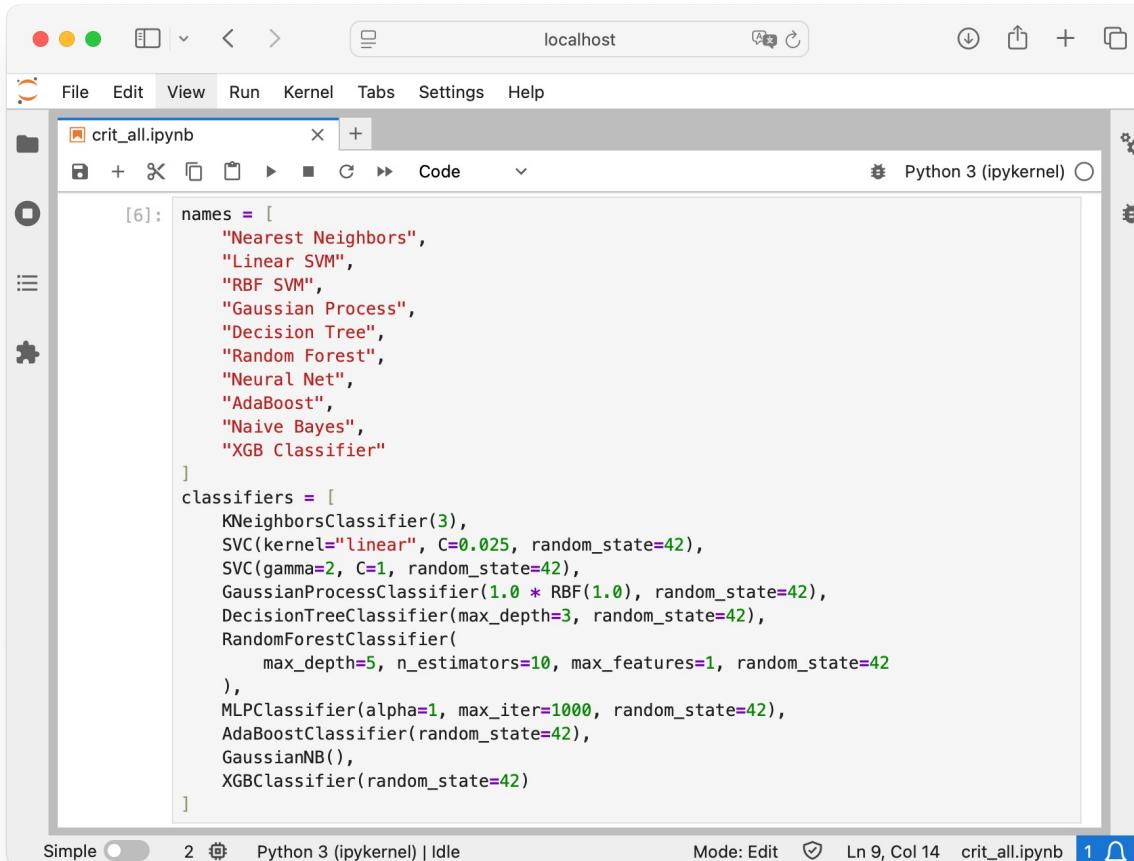
$$\ln p(D|w_{ML}) - M$$

$\ln p(D|w_{ML})$ - максимум логарифмической функции правдоподобия.

M – число параметров модели.

Однако такие критерии не учитывают неопределенность в параметрах модели, и на практике они, как правило, предпочитают слишком простые модели.

Пример выбора модели



The screenshot shows a Jupyter Notebook interface with the following details:

- Title Bar:** localhost
- File Menu:** File, Edit, View, Run, Kernel, Tabs, Settings, Help
- Toolbar:** Back, Forward, Home, Code, Python 3 (ipykernel)
- Code Cell:** [6]:

```
names = [
    "Nearest Neighbors",
    "Linear SVM",
    "RBF SVM",
    "Gaussian Process",
    "Decision Tree",
    "Random Forest",
    "Neural Net",
    "AdaBoost",
    "Naive Bayes",
    "XGB Classifier"
]
classifiers = [
    KNeighborsClassifier(3),
    SVC(kernel="linear", C=0.025, random_state=42),
    SVC(gamma=2, C=1, random_state=42),
    GaussianProcessClassifier(1.0 * RBF(1.0), random_state=42),
    DecisionTreeClassifier(max_depth=3, random_state=42),
    RandomForestClassifier(
        max_depth=5, n_estimators=10, max_features=1, random_state=42
    ),
    MLPClassifier(alpha=1, max_iter=1000, random_state=42),
    AdaBoostClassifier(random_state=42),
    GaussianNB(),
    XGBClassifier(random_state=42)
]
```
- Status Bar:** Simple, Python 3 (ipykernel) | Idle, Mode: Edit, Ln 9, Col 14, crit_all.ipynb, 1

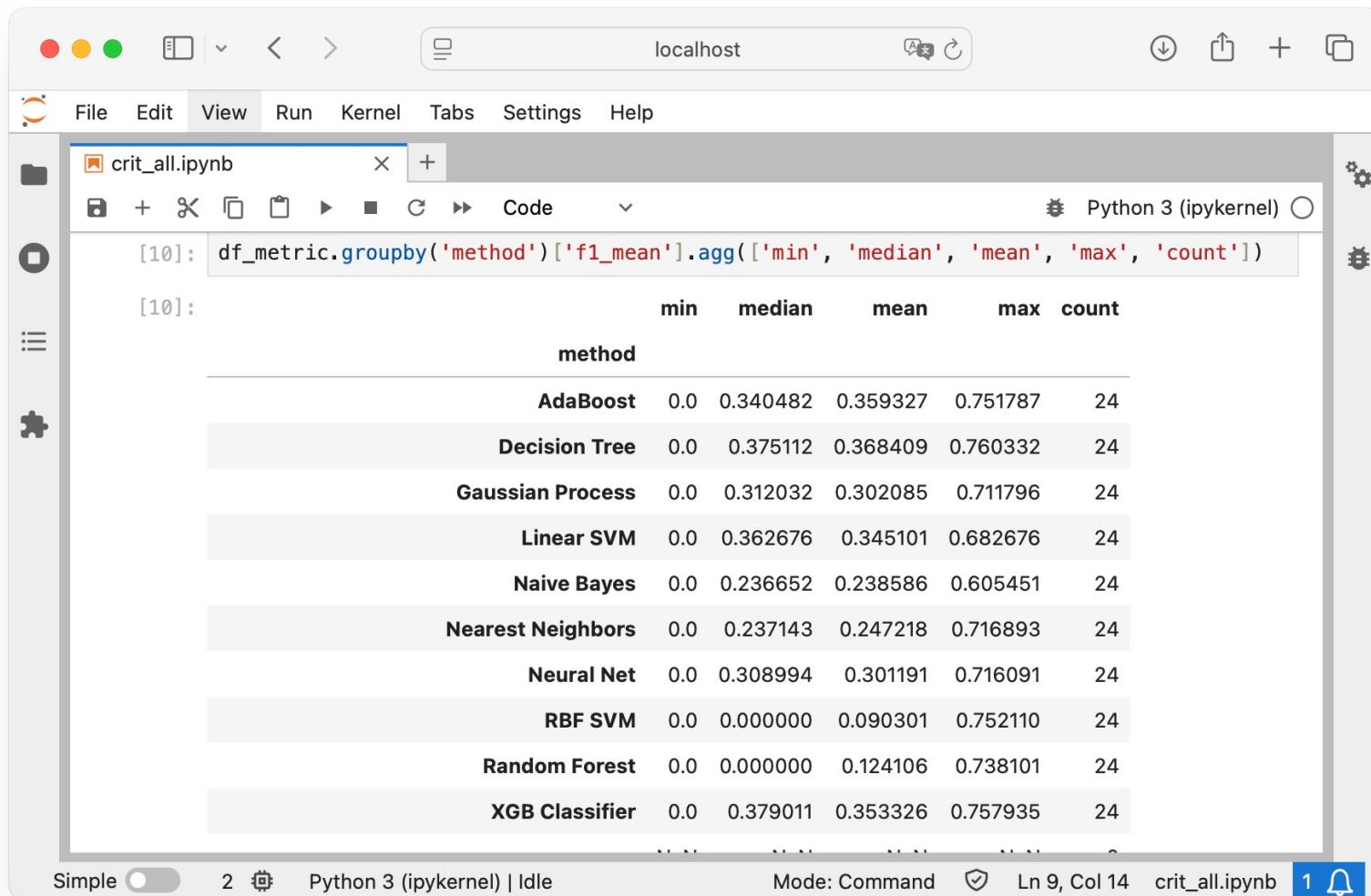
Пример выбора модели

The screenshot shows a Jupyter Notebook interface with the following details:

- Title Bar:** localhost
- File Menu:** File, Edit, View, Run, Kernel, Tabs, Settings, Help
- Code Cell:** [7]:
A code cell containing Python code for model selection. The code uses a StratifiedKFold cross-validation loop to calculate F1 scores for various classifiers across multiple columns of data.
- Output Area:** Shows a list of 12 personality traits or symptoms, numbered 1 to 12, each associated with a brief description.
- Bottom Status Bar:** Mode: Edit, Python 3 (ipykernel) | Idle, Ln 9, Col 14, crit_all.ipynb, 1

```
[7]: l = []
cv = StratifiedKFold(n_splits=3, shuffle=True, random_state=1)
for i, c in enumerate(df.columns[2:28]):
    crit = c[:-2]
    print(i+1, crit)
    y = df.loc[:,crit+'_X'].to_numpy().astype(int).ravel()
    if sum(y==0) > 5 and sum(y==1) > 5:
        # iterate over classifiers
        for i, (name, clf) in enumerate(zip(names, classifiers)):
            # Выполняем кросс-валидацию с метрикой F1
            f1_scores = cross_val_score(clf, x, y, cv=cv, scoring='f1')
            # Получаем среднее значение F1 и стандартное отклонение
            l.append({'i':i, 'col_name':crit, 'method': name, 'f1_mean': f1_scores.mean(), 'f1_std': f1_scores.std()})
    else:
        l.append({'col_name':crit, 'method': "невозможно оценить, т.к. мало данных"})
1 Бытовая ориентация
2 Снижение настроения
3 Импульсивность
4 Контроль
5 Астения
6 Тревожность личностная (один признак, без сочетаний)
7 Статусное тревоги (один признак, без сочетаний)
8 Творческая направленность
9 Признаки отклонения от нормы
10 Экстраверсия
11 Проблемы общения
12 Низкая социализированность
```

Пример выбора модели



The screenshot shows a Jupyter Notebook interface with the following details:

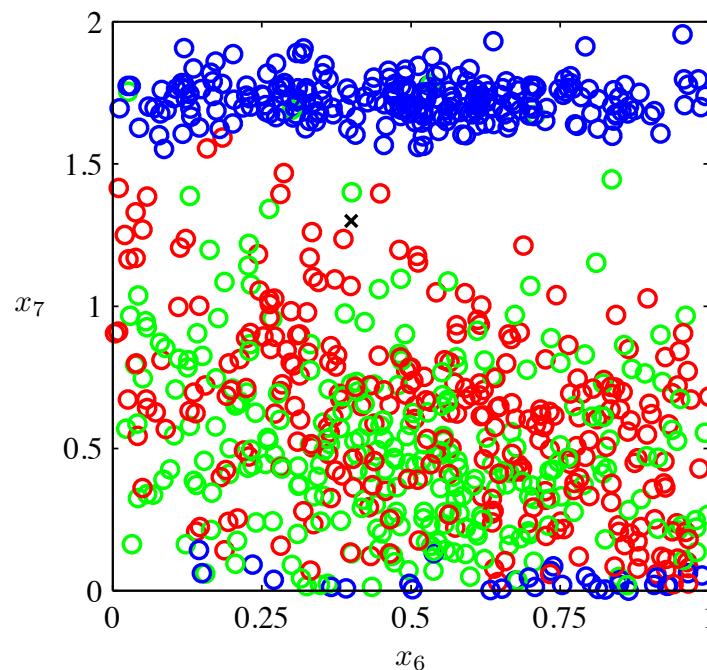
- File Bar:** File, Edit, View, Run, Kernel, Tabs, Settings, Help.
- Current Notebook:** crit_all.ipynb
- Kernel:** Python 3 (ipykernel)
- Code Cell [10]:** df_metric.groupby('method')[['f1_mean']].agg(['min', 'median', 'mean', 'max', 'count'])
- Output Cell [10]:** A table showing classifier performance metrics. The columns are min, median, mean, max, and count. The rows are grouped by method.

method	min	median	mean	max	count
AdaBoost	0.0	0.340482	0.359327	0.751787	24
Decision Tree	0.0	0.375112	0.368409	0.760332	24
Gaussian Process	0.0	0.312032	0.302085	0.711796	24
Linear SVM	0.0	0.362676	0.345101	0.682676	24
Naive Bayes	0.0	0.236652	0.238586	0.605451	24
Nearest Neighbors	0.0	0.237143	0.247218	0.716893	24
Neural Net	0.0	0.308994	0.301191	0.716091	24
RBF SVM	0.0	0.000000	0.090301	0.752110	24
Random Forest	0.0	0.000000	0.124106	0.738101	24
XGB Classifier	0.0	0.379011	0.353326	0.757935	24

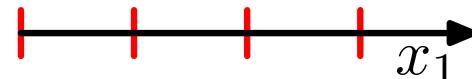
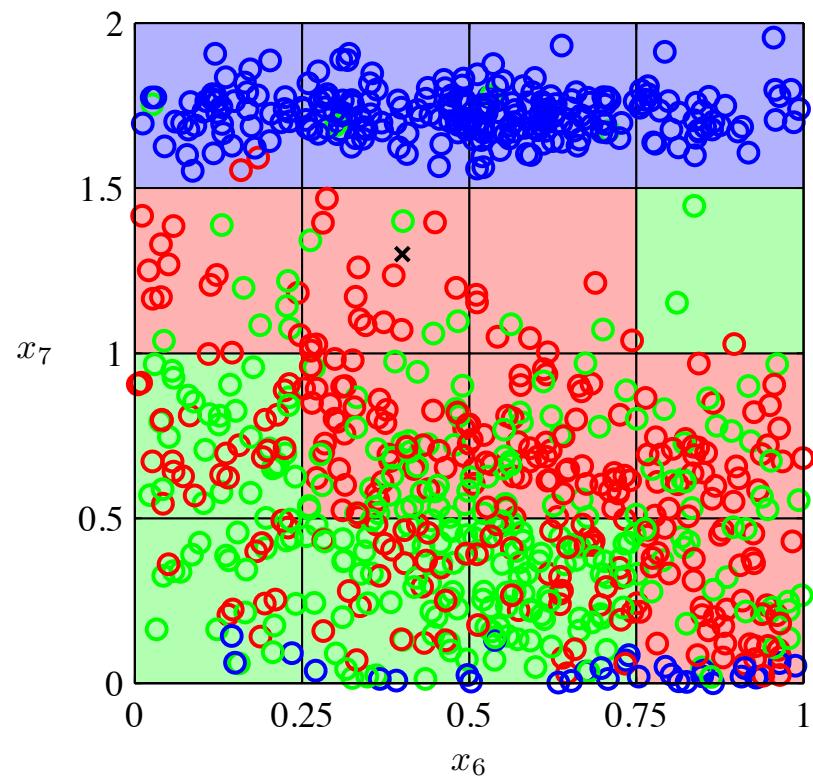
- Bottom Status Bar:** Simple (radio button), 2 (cell count), Python 3 (ipykernel) | Idle, Mode: Command, Ln 9, Col 14, crit_all.ipynb, 1 (bell icon).

Проклятие размерности

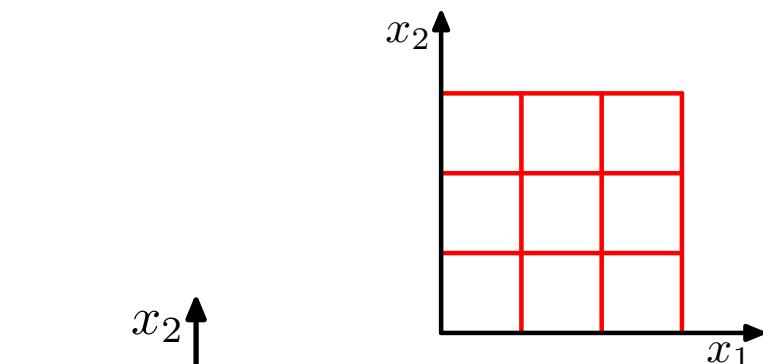
На рисунке показаны 100 точек из набора данных в виде диаграммы рассеяния, показывающей 2 из 12 измерений, x_6 и x_7 (остальные десять входных значений не указаны, чтобы не усложнять рисунок). Каждая точка данных помечена в соответствии с тем, какому из трех геометрических классов она принадлежит. Наша цель состоит в том, чтобы использовать эти данные в качестве обучающего множества и получить возможность классифицировать новое наблюдение (x_6 , x_7), обозначен крестиком.



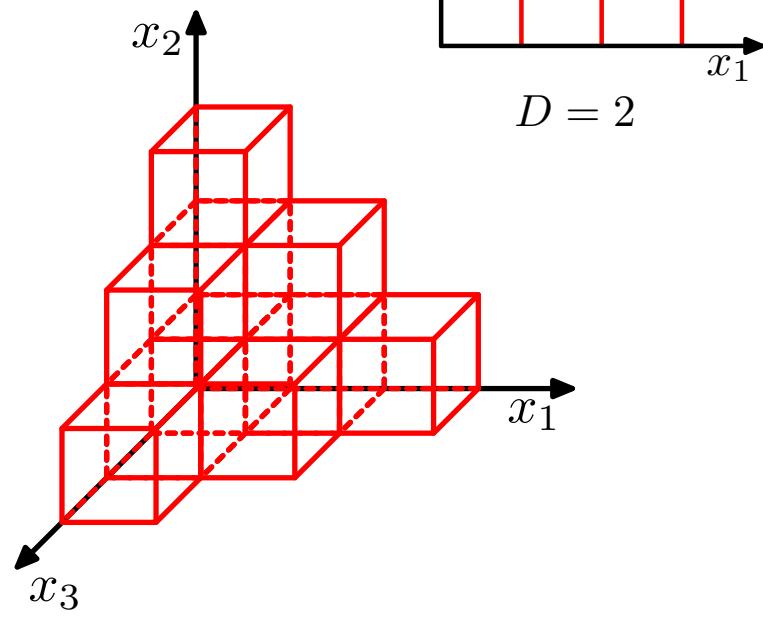
Проклятие размерности



$$D = 1$$



$$D = 2$$



$$D = 3$$

Теория принятия решений

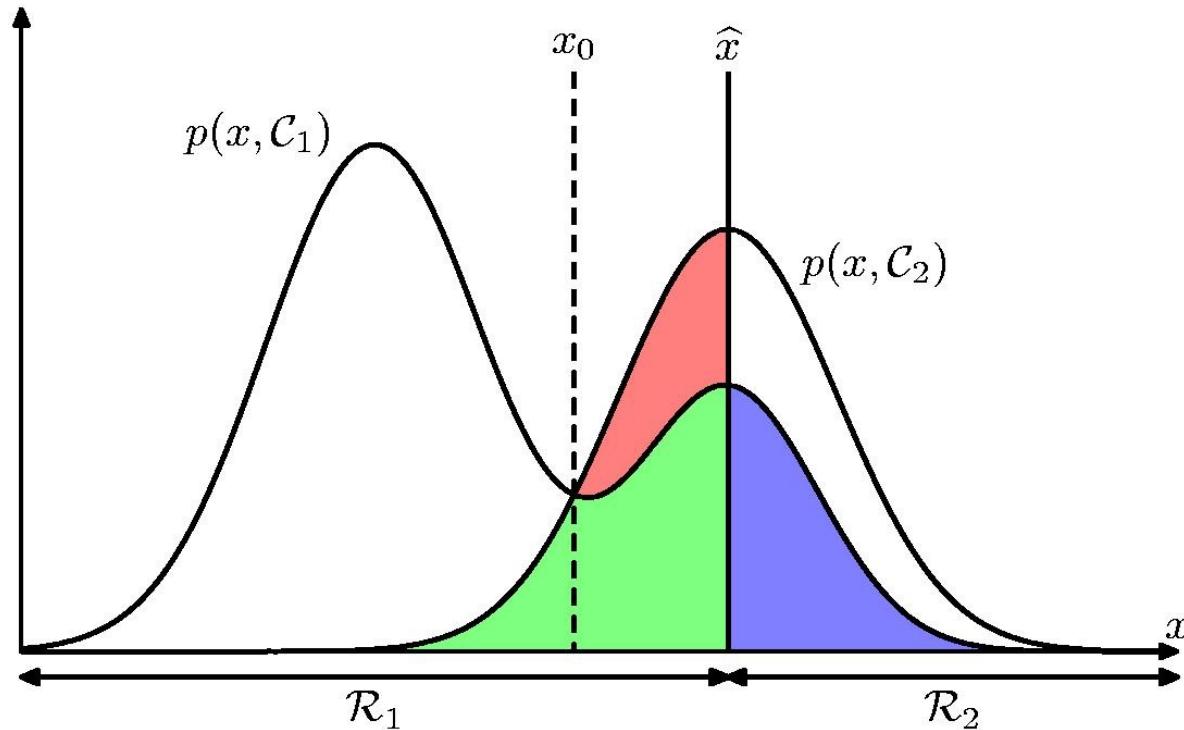
Шаг вывода

- Определите $p(t|x)$ или $p(x,t)$.

Шаг принятия решений

- Для заданного x , определите оптимальное t .

Минимальная частота ошибочной классификации



$$\begin{aligned} p(\text{mistake}) &= p(\mathbf{x} \in \mathcal{R}_1, \mathcal{C}_2) + p(\mathbf{x} \in \mathcal{R}_2, \mathcal{C}_1) \\ &= \int_{\mathcal{R}_1} p(\mathbf{x}, \mathcal{C}_2) d\mathbf{x} + \int_{\mathcal{R}_2} p(\mathbf{x}, \mathcal{C}_1) d\mathbf{x}. \end{aligned}$$

Минимальные ожидаемые потери

Пример: классифицировать медицинские изображения как «рак»(cancer) или «норма».

		Decision	
		cancer	normal
Truth	cancer	0	1000
	normal	1	0

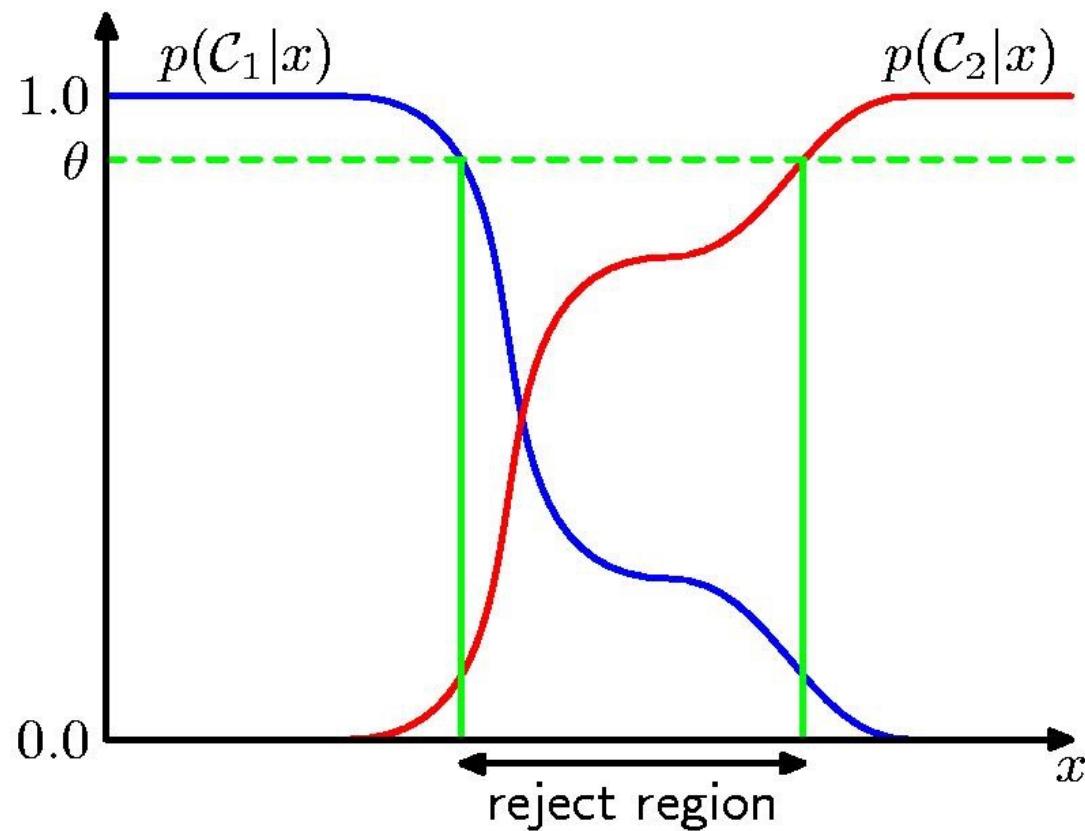
Минимальные ожидаемые потери

$$\mathbb{E}[L] = \sum_k \sum_j \int_{\mathcal{R}_j} L_{kj} p(\mathbf{x}, \mathcal{C}_k) d\mathbf{x}$$

Регионы \mathcal{R}_j выбираются так,
чтобы минимизировать

$$\mathbb{E}[L] = \sum_k L_{kj} p(\mathcal{C}_k | \mathbf{x})$$

Опция «не принимать решение»



3 подхода к принятию решений

Генеративные (порождающие) модели

- Определение условных вероятностей классов $p(x/C_k)$
- Вывод априорных вероятностей классов $p(C_k)$
- Использование теоремы Байеса для нахождения апостериорных вероятностей классов $p(C_k/x) = p(x/C_k)p(C_k)/p(x)$
- Или сразу определять $p(x, C_k)$ и далее определять условные вероятности через нормировку

Дискриминантные модели

- Определение апостериорных вероятностей классов $p(C_k/x)$
- Использование теории принятия решений для отнесения каждого нового x к одному из классов.

Нахождение дискриминантной функции

- $f(x)$ отображает x на метку класса
- В этом случае вероятности не играют никакой роли. Единая задача обучения.

Преимущества и недостатки

- Первый подход является наиболее требовательным, поскольку предполагает поиск совместного распределения по x и C_k . Во многих приложениях вектор x будет иметь большую размерность, и, следовательно, нам может понадобиться большой набор обучающих данных, чтобы иметь возможность определять условные по классу плотности (class-conditional densities) с разумной точностью.
- Априорные вероятности классов $p(C_k)$ часто можно вычислить просто как долю точек из множества обучающих данных, принадлежащих каждому из классов.
- Одним из преимуществ первого подхода является то, что он также позволяет определить маргинальную плотность данных $p(x)$ для обнаружения по модели новых точек с низкой вероятностью и точек, прогнозы которых могут иметь низкую точность. Это явление называется обнаружением выбросов или обнаружением новизны.

Зачем разделять вывод и решение?

Минимизация риска (матрица потерь может меняться со временем)

Опция «Не принимать решение»

Несбалансированные априорные классы.

- Рассмотрим нашу медицинскую рентгенологическую задачу еще раз и предположим, что мы собрали большое количество рентгеновских снимков населения в целом для использования в качестве обучающих данных с целью создания автоматизированной системы скрининга. Поскольку рак встречается редко среди населения в целом, то если бы мы использовали такой набор данных для обучения адаптивной модели, мы могли бы столкнуться с серьезными трудностями из-за небольшой доли класса рака. Например, классификатор, который назначает каждую точку нормальному классу, уже достиг бы 99,9% точности, и было бы трудно избежать этого тривиального решения. Кроме того, даже большой набор данных будет содержать очень мало примеров рентгеновских снимков, соответствующих раку, и поэтому алгоритм обучения не будет подвергаться широкому диапазону примеров таких изображений и, следовательно, вряд ли будет хорошо обобщать. Сбалансированный набор данных, в котором мы выбрали равное количество примеров из каждого из классов, позволил бы нам найти более точную модель. Однако затем нам необходимо компенсировать эффекты наших модификаций обучающих данных. Из теоремы Байеса мы видим, что апостериорные вероятности пропорциональны априорным вероятностям, которые мы можем интерпретировать как доли точек в каждом классе. Поэтому мы можем просто взять апостериорные вероятности, полученные из нашего искусственно сбалансированного набора данных, и сначала разделить их на доли классов в этом наборе данных, а затем умножить на доли классов в генеральной совокупности, к которой мы хотим применить модель. Наконец, нам необходимо выполнить нормализацию, чтобы гарантировать, что сумма новых апостериорных вероятностей равна единице. Обратите внимание, что эту процедуру нельзя применить, если мы обучили дискриминантную функцию напрямую, а не определяли апостериорные вероятности.

Объединение моделей.

- Для сложных приложений мы, возможно, захотим разбить задачу на ряд меньших подзадач, каждая из которых может решаться отдельным модулем. Например, в гипотетической проблеме медицинского диагноза у нас может быть информация, полученная из анализов крови, а также по рентгеновским снимкам. Вместо того чтобы объединять всю эту не однородную информацию в одном огромном пространстве входных данных, может быть более эффективным создание одной системы для интерпретации рентгеновских изображений и другой для интерпретации показателей крови. Поскольку каждая из двух моделей дает апостериорные вероятности для классов, мы можем систематически комбинировать выходы с использованием правил теории вероятностей.

Теория принятия решений для регрессии

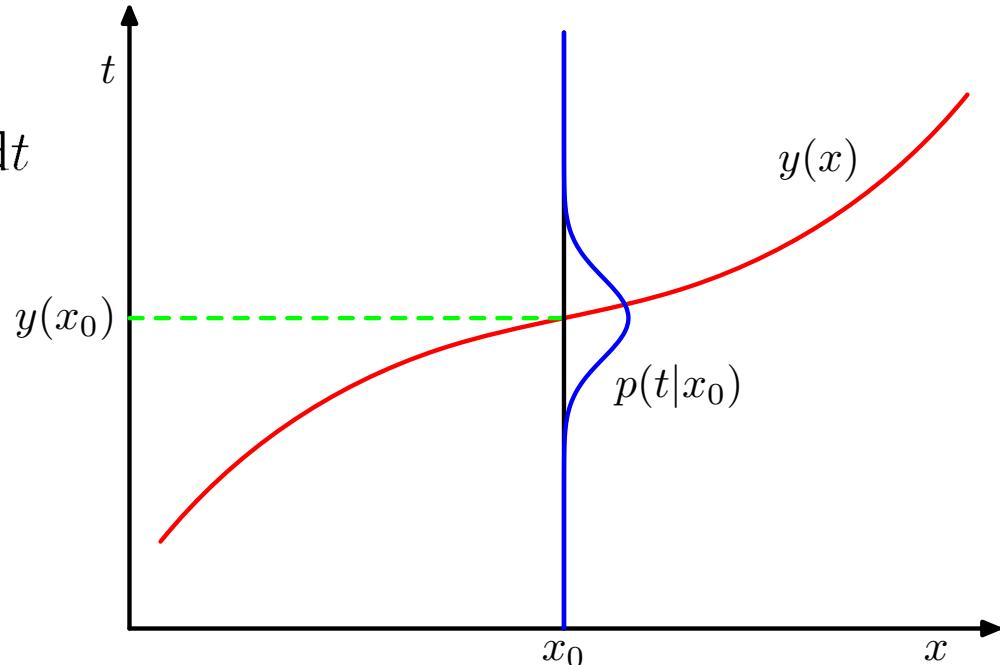
Шаг вывода

- Определить $p(\mathbf{x}, t)$.

Шаг решения

- Для заданного \mathbf{x} , построить оптимальный прогноз $y(\mathbf{x})$ для t .
- Функция потерь:

$$\mathbb{E}[L] = \iint L(t, y(\mathbf{x})) p(\mathbf{x}, t) d\mathbf{x} dt$$



Квадратичная функция потерь

$$\mathbb{E}[L] = \iint \{y(\mathbf{x}) - t\}^2 p(\mathbf{x}, t) \, d\mathbf{x} \, dt$$

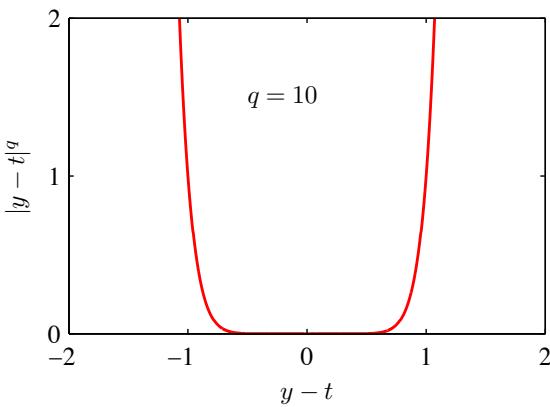
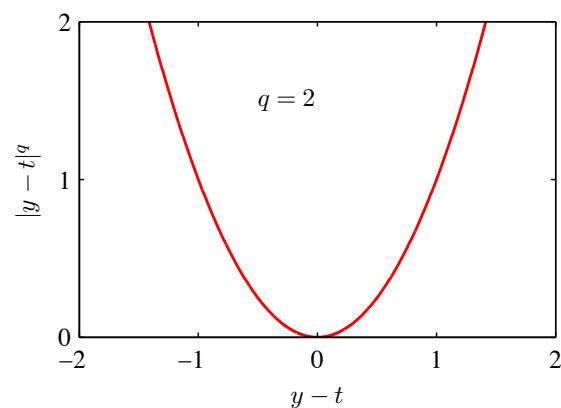
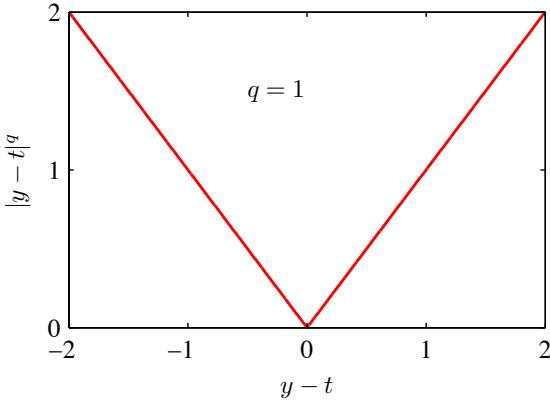
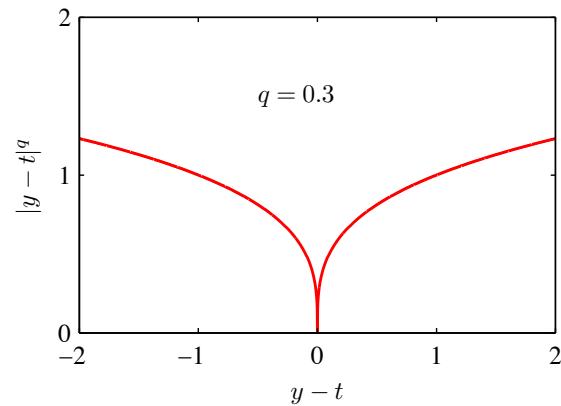
$$\begin{aligned}\{y(\mathbf{x}) - t\}^2 &= \{y(\mathbf{x}) - \mathbb{E}[t|\mathbf{x}] + \mathbb{E}[t|\mathbf{x}] - t\}^2 \\ &= \{y(\mathbf{x}) - \mathbb{E}[t|\mathbf{x}]\}^2 + 2\{y(\mathbf{x}) - \mathbb{E}[t|\mathbf{x}]\}\{\mathbb{E}[t|\mathbf{x}] - t\} + \{\mathbb{E}[t|\mathbf{x}] - t\}^2\end{aligned}$$

$$\mathbb{E}[L] = \int \{y(\mathbf{x}) - \mathbb{E}[t|\mathbf{x}]\}^2 p(\mathbf{x}) \, d\mathbf{x} + \int \text{var}[t|\mathbf{x}] p(\mathbf{x}) \, d\mathbf{x}$$

$$y(\mathbf{x}) = \mathbb{E}[t|\mathbf{x}]$$

Функция потерь Минковского

$$\mathbb{E}[L_q] = \iint |y(x) - t|^q p(x, t) dx dt$$



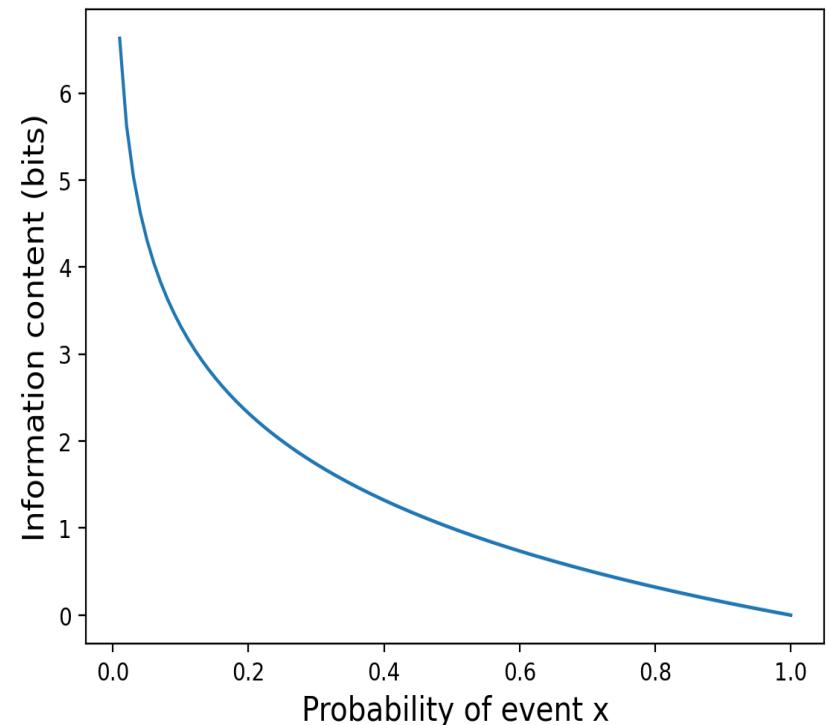
Теория информации

Сколько информации мы получаем, когда наблюдаем определенную переменную?

Рассмотрим, например, событие появления в небе инопланетного космического корабля. Мы никогда его не видели и были бы крайне удивлены, если бы такое событие когда-нибудь произошло, поскольку это неожиданно с учетом наших текущих знаний. Следовательно, количество информации можно рассматривать как степень удивления при получении значения x . Мера информационного содержания, зависит от распределения вероятностей $p(x)$ и, в частности, определяется логарифмом $p(x)$:

$$h(x) = -\log_2 p(x)$$

где знак «минус» гарантирует, что $h(x) \geq 0$. Если основание логарифма равно 2, то единицы измерения $h(x)$ — **биты**.

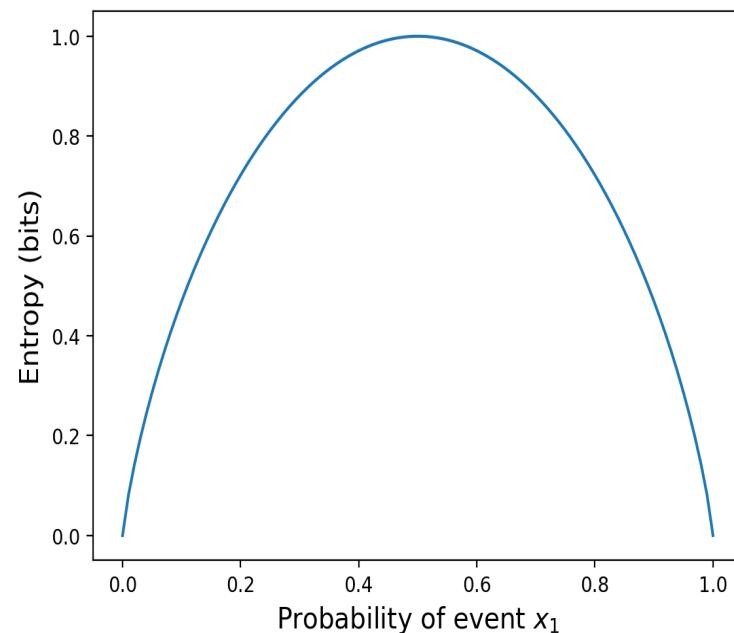


Энтропия

Средний объем информации получается путем умножения ожидаемого значения на содержание информации, заданное формулой:

$$H[x] = - \sum_x p(x)h(x) = - \sum_x p(x)\log_2 p(x)$$

Эта важная величина называется энтропией случайной величины X . Рассмотрим дискретную случайную величину, X имеющую два возможных значения x_1 с вероятностями p и $1 - p$:



Энтропия

- Теория кодирования: x дискретен с 8 возможными состояниями; сколько бит нужно для передачи состояния x ?
- Все состояния одинаково вероятны

$$H[x] = -8 \times \frac{1}{8} \log_2 \frac{1}{8} = 3 \text{ bits.}$$

Энтропия

x	a	b	c	d	e	f	g	h
$p(x)$	$\frac{1}{2}$	$\frac{1}{4}$	$\frac{1}{8}$	$\frac{1}{16}$	$\frac{1}{64}$	$\frac{1}{64}$	$\frac{1}{64}$	$\frac{1}{64}$
code	0	10	110	1110	111100	111101	111110	111111

$$\begin{aligned} H[x] &= -\frac{1}{2} \log_2 \frac{1}{2} - \frac{1}{4} \log_2 \frac{1}{4} - \frac{1}{8} \log_2 \frac{1}{8} - \frac{1}{16} \log_2 \frac{1}{16} - \frac{4}{64} \log_2 \frac{1}{64} \\ &= 2 \text{ bits} \end{aligned}$$

$$\begin{aligned} \text{average code length} &= \frac{1}{2} \times 1 + \frac{1}{4} \times 2 + \frac{1}{8} \times 3 + \frac{1}{16} \times 4 + 4 \times \frac{1}{64} \times 6 \\ &= 2 \text{ bits} \end{aligned}$$

Энтропия

- Сколькоими способами можно разместить N одинаковых объектов в M ячейках?

$$W = \frac{N!}{\prod_i n_i!}$$

- W – кратность.

$$H = \frac{1}{N} \ln W \simeq - \lim_{N \rightarrow \infty} \sum_i \left(\frac{n_i}{N} \right) \ln \left(\frac{n_i}{N} \right) = - \sum_i p_i \ln p_i$$

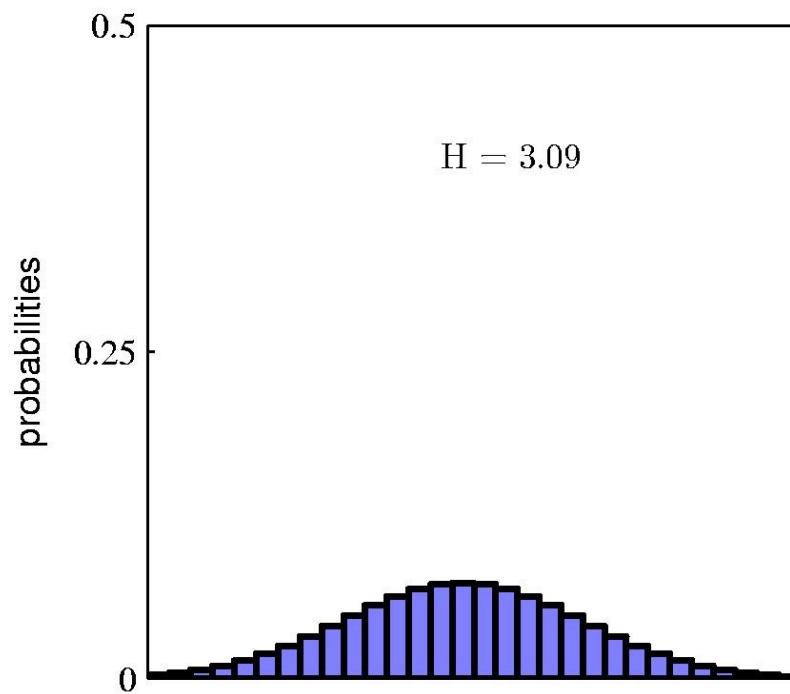
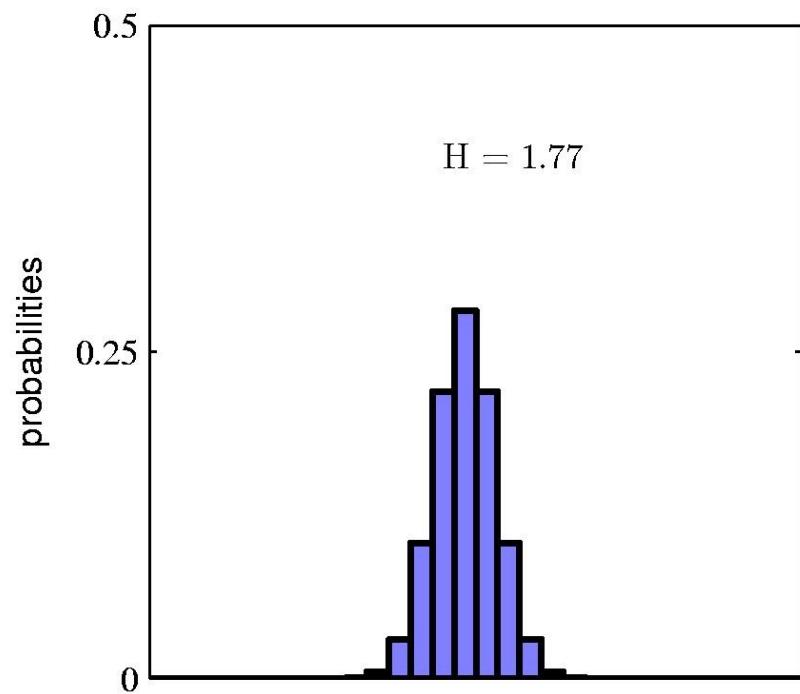
- В выводе использовано приближение Стирлинга:

$$\ln N! = N \ln N - N$$

- Энтропия максимизируется, когда

$$\forall i : p_i = \frac{1}{M}$$

Энтропия



Дифференциальная энтропия

- Разбейте на отрезки длиной Δ вещественную ось:

$$\lim_{\Delta \rightarrow 0} \left\{ - \sum_i p(x_i) \Delta \ln p(x_i) \right\} = - \int p(x) \ln p(x) dx$$

- Максимум дифференциальной энтропии на неограниченной вещественной оси (для фикс. σ^2)

$$p(x) = \mathcal{N}(x|\mu, \sigma^2)$$

- равен

$$H[x] = \frac{1}{2} \{ 1 + \ln(2\pi\sigma^2) \} .$$

Условная энтропия

В случае совместного распределения $p(\mathbf{x}, \mathbf{y})$ средняя дополнительная информация, необходимая для указания \mathbf{y} при том что значение \mathbf{x} уже известно, называется *условной энтропией* и определяется как

$$H[\mathbf{y}|\mathbf{x}] = - \int p(\mathbf{x}, \mathbf{y}) \ln p(\mathbf{y}|\mathbf{x}) d\mathbf{x} d\mathbf{y}$$

Более того, используя правило произведения, легко видеть, что условная энтропия удовлетворяет соотношению:

$$\begin{aligned} H[\mathbf{x}, \mathbf{y}] &= - \int \int p(\mathbf{x}, \mathbf{y}) \ln p(\mathbf{x}, \mathbf{y}) d\mathbf{x} d\mathbf{y} \\ &= - \int \int p(\mathbf{x}, \mathbf{y}) \ln(p(\mathbf{y}|\mathbf{x})p(\mathbf{x})) d\mathbf{x} d\mathbf{y} \\ &= - \int \int p(\mathbf{x}, \mathbf{y})(\ln p(\mathbf{y}|\mathbf{x}) + \ln p(\mathbf{x})) d\mathbf{x} d\mathbf{y} \\ &= - \int \int p(\mathbf{x}, \mathbf{y}) \ln p(\mathbf{y}|\mathbf{x}) d\mathbf{x} d\mathbf{y} - \int \int p(\mathbf{x}, \mathbf{y}) \ln p(\mathbf{x}) d\mathbf{x} d\mathbf{y} \\ &= - \int \int p(\mathbf{x}, \mathbf{y}) \ln p(\mathbf{y}|\mathbf{x}) d\mathbf{x} d\mathbf{y} - \int \int p(\mathbf{x}, \mathbf{y}) d\mathbf{y} \ln p(\mathbf{x}) d\mathbf{x} \\ &= - \int \int p(\mathbf{x}, \mathbf{y}) \ln p(\mathbf{y}|\mathbf{x}) d\mathbf{x} d\mathbf{y} - \int p(\mathbf{x}) \ln p(\mathbf{x}) d\mathbf{x} \\ &= H[\mathbf{y}|\mathbf{x}] + H[\mathbf{x}] \end{aligned}$$

Информация, необходимая для описания \mathbf{x}, \mathbf{y} представлена \mathbf{x} только информацией, необходимой для его описания, и дополнительной информацией, необходимой для уточнения \mathbf{y} .

Дивергенция Кульбака — Лейблера

Рассмотрим некоторое неизвестное распределение $p(x)$ и предположим, что смоделировали его, используя аппроксимирующее распределение $q(x)$. Если мы используем распределение $q(x)$ для создания схемы кодирования с целью передачи значений x получателю, тогда среднее дополнительное количество информации (измеренное в натах), необходимое для указания значения x (при условии, что мы выбираем эффективную схему кодирования), в результате использования $q(x)$ вместо истинного распределения $p(x)$ задается формулой

$$\text{KL}(p\|q) = - \int p(\mathbf{x}) \ln q(\mathbf{x}) d\mathbf{x} - \left(- \int p(\mathbf{x}) \ln p(\mathbf{x}) d\mathbf{x} \right)$$

$$= - \int p(\mathbf{x}) \ln \left\{ \frac{q(\mathbf{x})}{p(\mathbf{x})} \right\} d\mathbf{x}$$

$$\text{KL}(p\|q) \simeq \frac{1}{N} \sum_{n=1}^N \{ - \ln q(\mathbf{x}_n | \boldsymbol{\theta}) + \ln p(\mathbf{x}_n) \}$$

$$\text{KL}(p\|q) \geq 0$$

$$\text{KL}(p\|q) \neq \text{KL}(q\|p)$$

Взаимная информация

$$\begin{aligned} I[\mathbf{x}, \mathbf{y}] &\equiv \text{KL}(p(\mathbf{x}, \mathbf{y}) \| p(\mathbf{x})p(\mathbf{y})) \\ &= - \iint p(\mathbf{x}, \mathbf{y}) \ln \left(\frac{p(\mathbf{x})p(\mathbf{y})}{p(\mathbf{x}, \mathbf{y})} \right) d\mathbf{x} d\mathbf{y} \end{aligned}$$

$$I[\mathbf{x}, \mathbf{y}] = H[\mathbf{x}] - H[\mathbf{x}|\mathbf{y}] = H[\mathbf{y}] - H[\mathbf{y}|\mathbf{x}]$$

Итоги

- Определите цели
- Измеряйте свое приближение к целям
- Ведите разведочный анализ, готовьте данные
- Выбирайте модели
- Обучайте модели
- Рассчитывайте метрики
- Внедряйте хорошие модели в бизнес-процессы
- Постоянно повышайте качество

Спасибо за внимание!