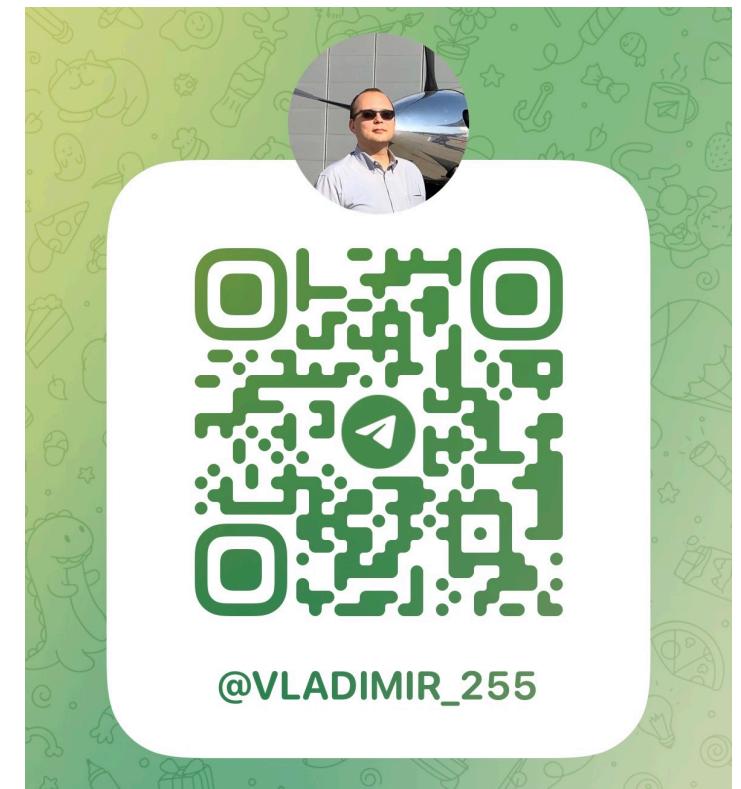


Машинное обучение (краткий курс)

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



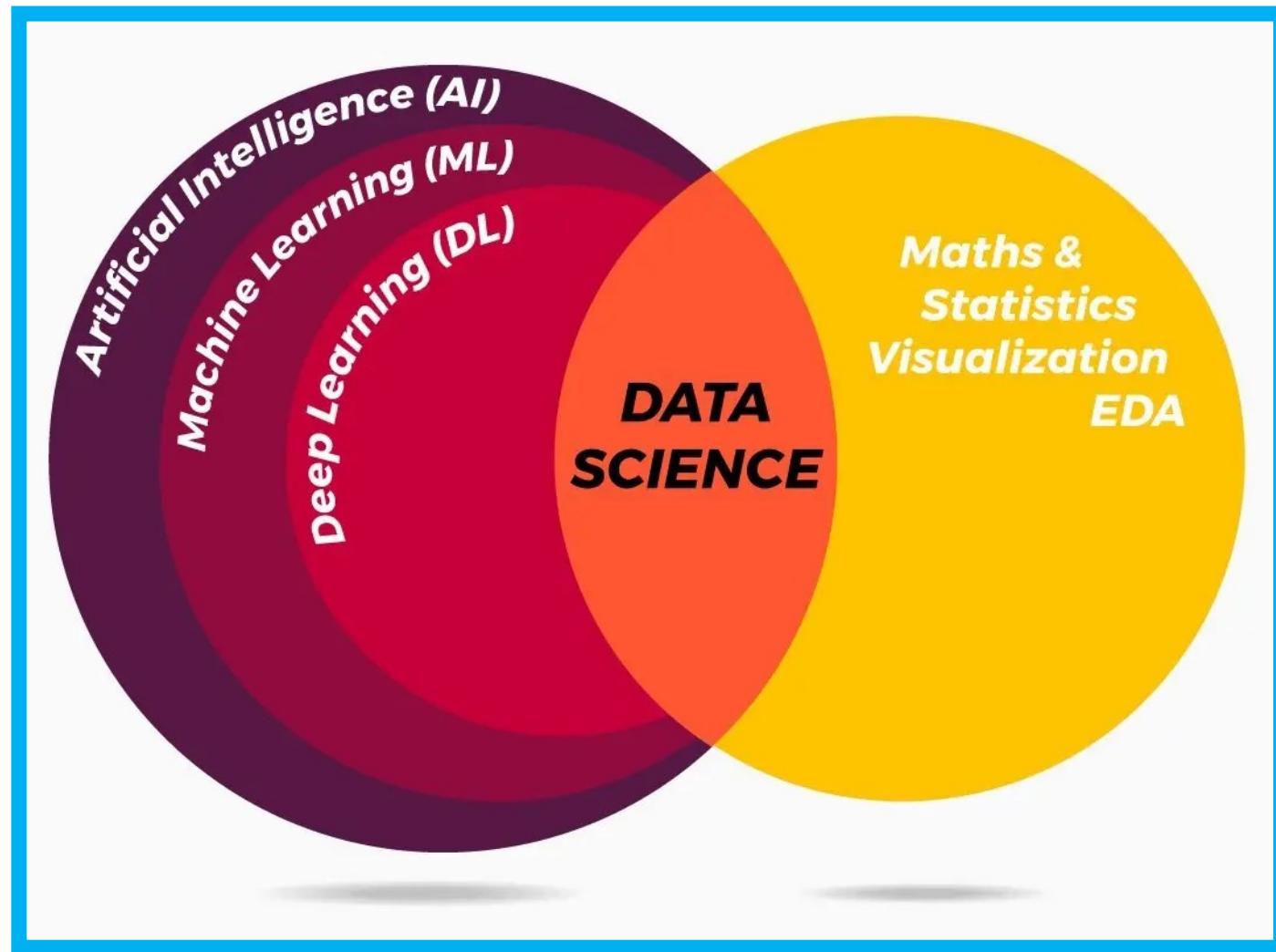
Машинное обучение.

Определение

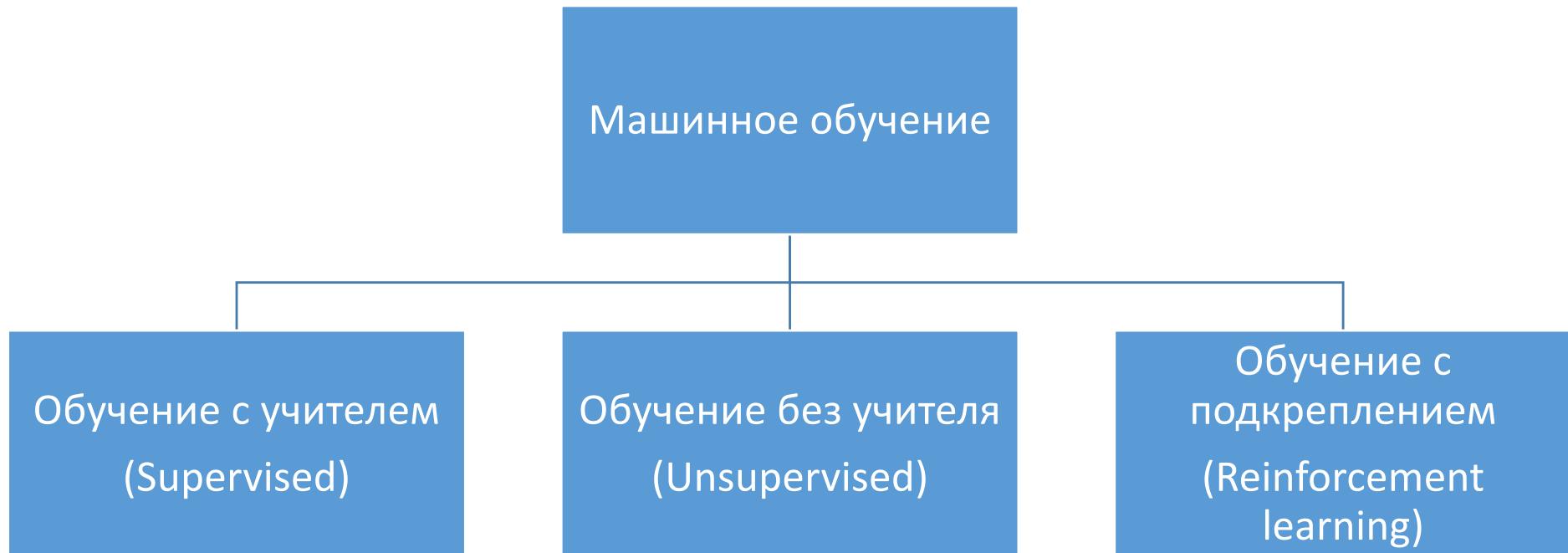
Говорят, что *компьютерная программа обучается на опыте* E относительно некоторого класса задач T и меры качества P , если ее качество на задачах, принадлежащих T , измеренное в соответствии с P , улучшается с увеличением опыта E .

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?

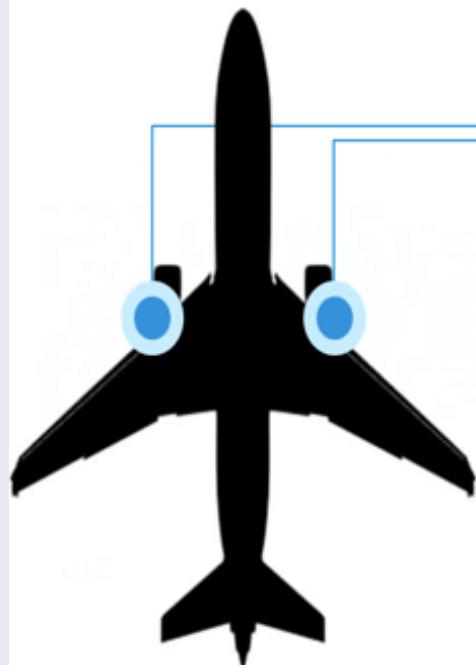
Что такого случилось?

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

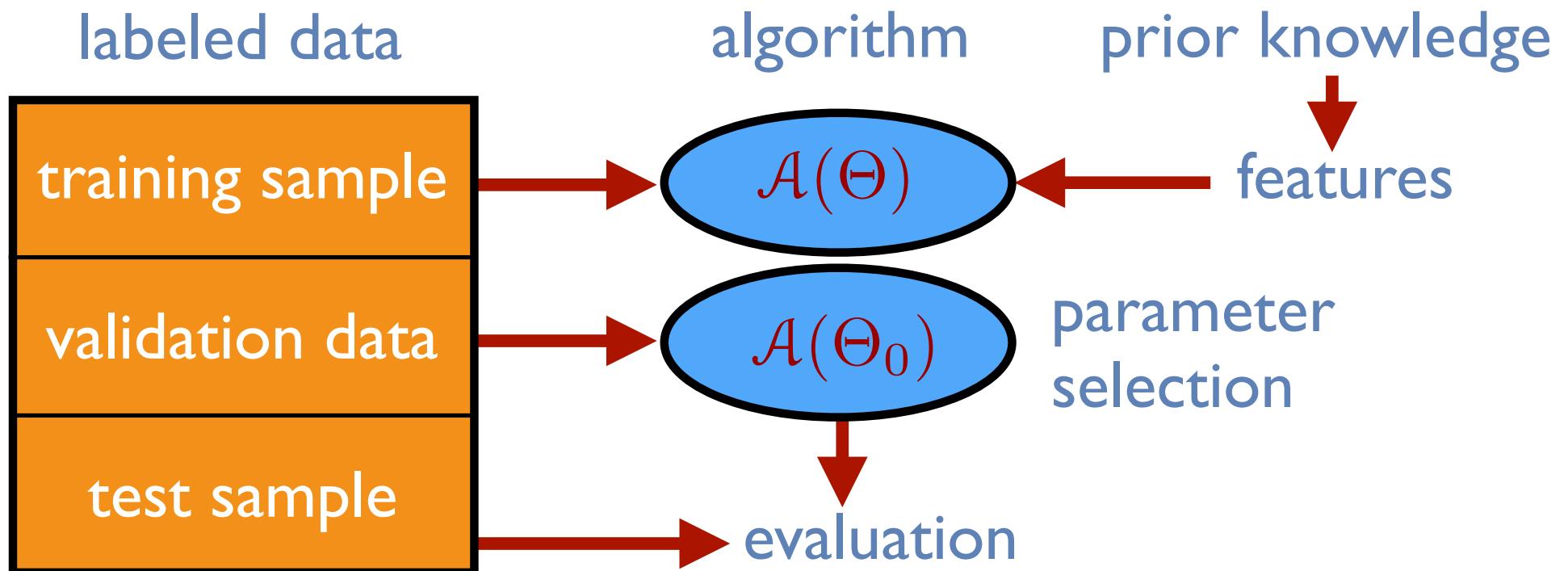
зависимость y^* по выборке $(x_i, t_i)_{i=1}^N$, то есть
построить решающую функцию (decision function)

$$y: X \rightarrow Y,$$

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 a должна допускать эффективную
компьютерную реализацию.

Этапы обучения



Рекомендуемые системы программирования и библиотеки

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

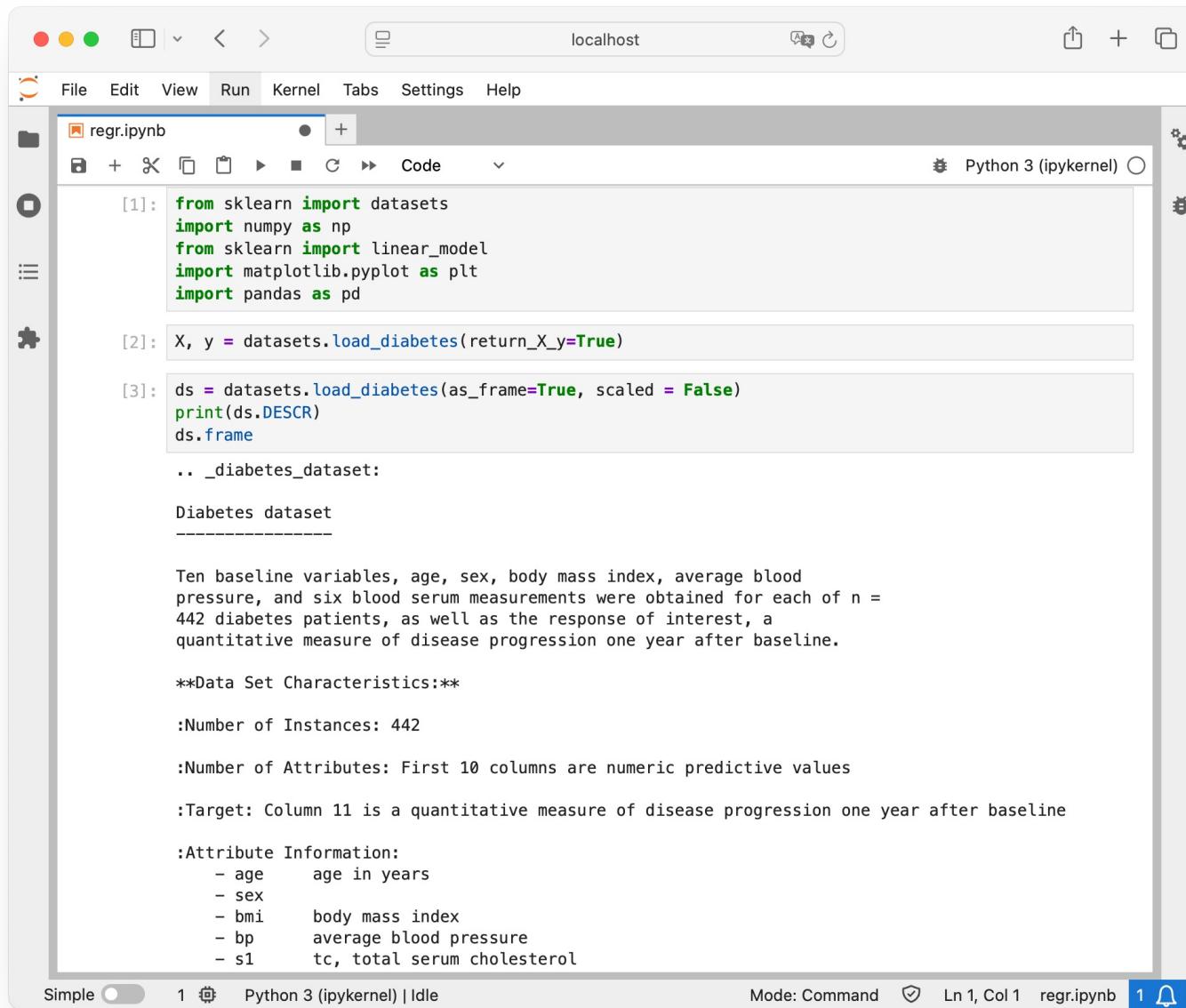


Следующий семестр

- 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 "load_diabetes" dataset. The output shows the dataset's characteristics, including 442 instances, 11 attributes, and a target variable representing disease progression. It also provides attribute information for the first five columns.

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

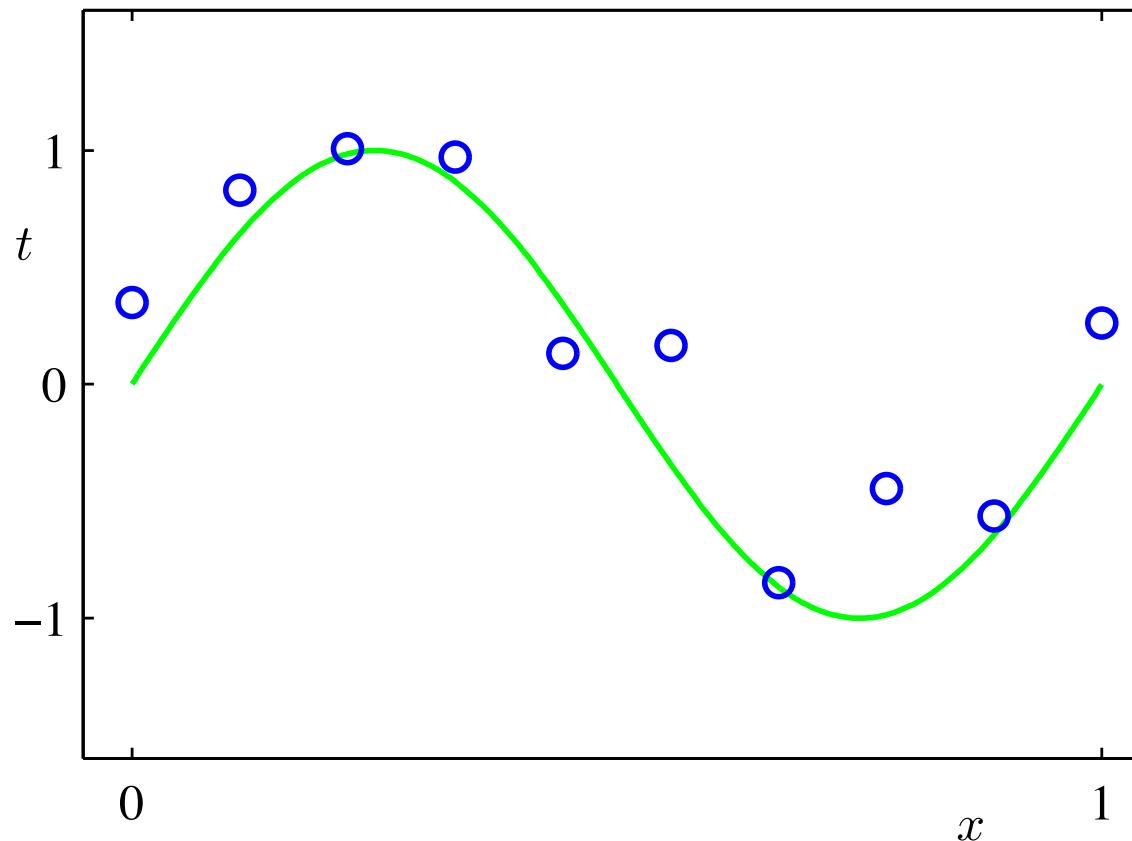
- Рассмотрим синтетически сгенерированный пример набора данных. Данные были получены из функции $\sin(2\pi x)$ путём добавления случайного гауссовского шума со стандартным отклонением $\sigma = 0.3$.
- Мы сгенерировали $N = 10$ наблюдения, равномерно распределенные в диапазоне $[0,1]$. Эти наблюдения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

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

- 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.

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

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

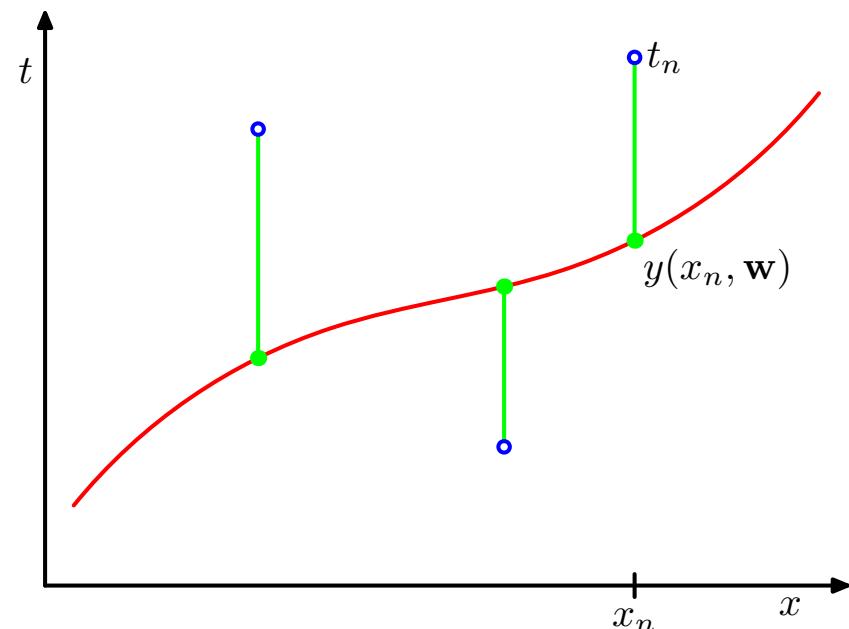
Необходимо определить значения коэффициентов \mathbf{w} , подгоняя полином к обучающим данным. Это можно сделать, минимизировав функцию ошибки, которая измеряет несоответствие между функцией $y(x, \mathbf{w})$, для заданного значения \mathbf{w} , и точками обучающих данных.

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 N точка данных из сгенерированного обучающего набора имеет 1 размерность, то есть $x \in \mathbb{R}$. Однако полиномиальная функция заполняет M признаки для каждого входного значения x , по сути, преобразуясь x в M -мерный вектор. Таким образом, обучающий набор x можно записать в виде $N \times M$ матрицы \mathbf{X} , где \mathbf{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:

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

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

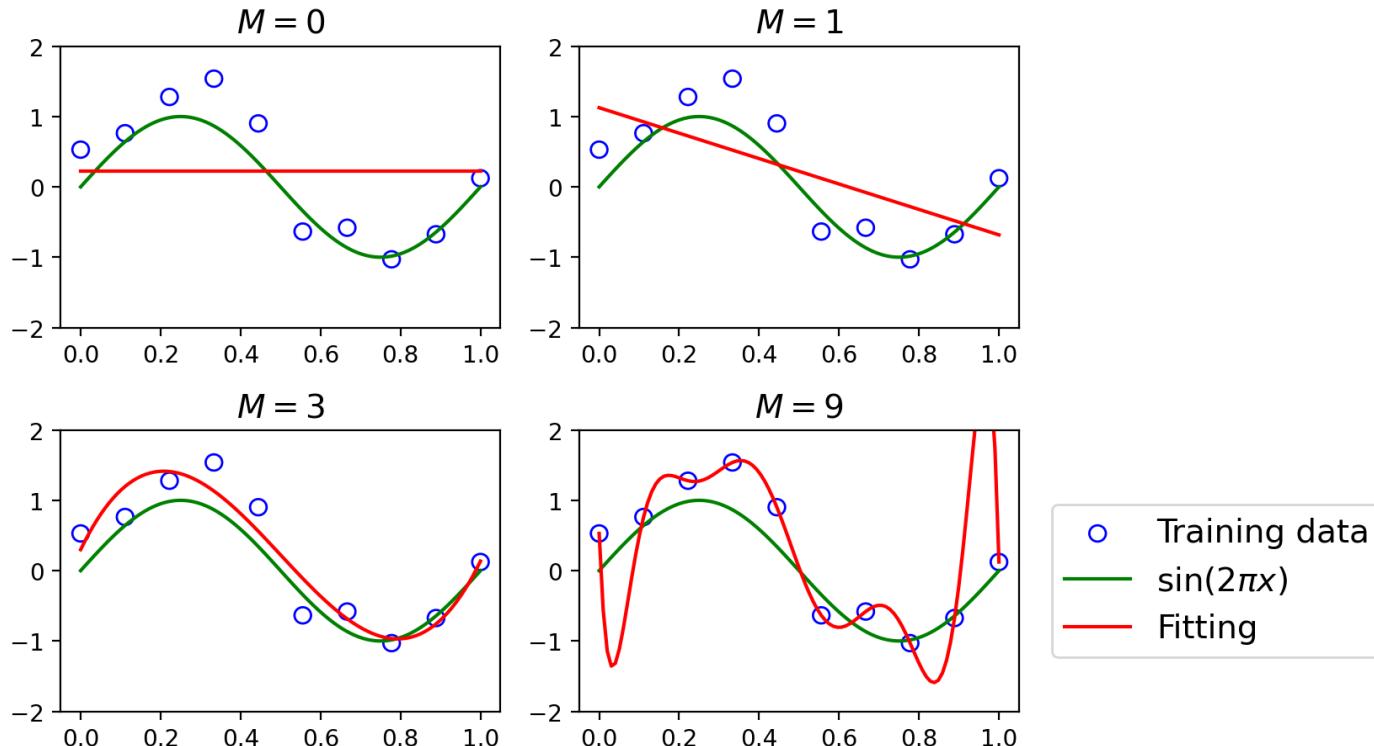
$$\nabla E(\mathbf{w}^*) = \mathbf{X}^\top (\mathbf{X}\mathbf{w}^* - \mathbf{t}) = \mathbf{0}$$

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

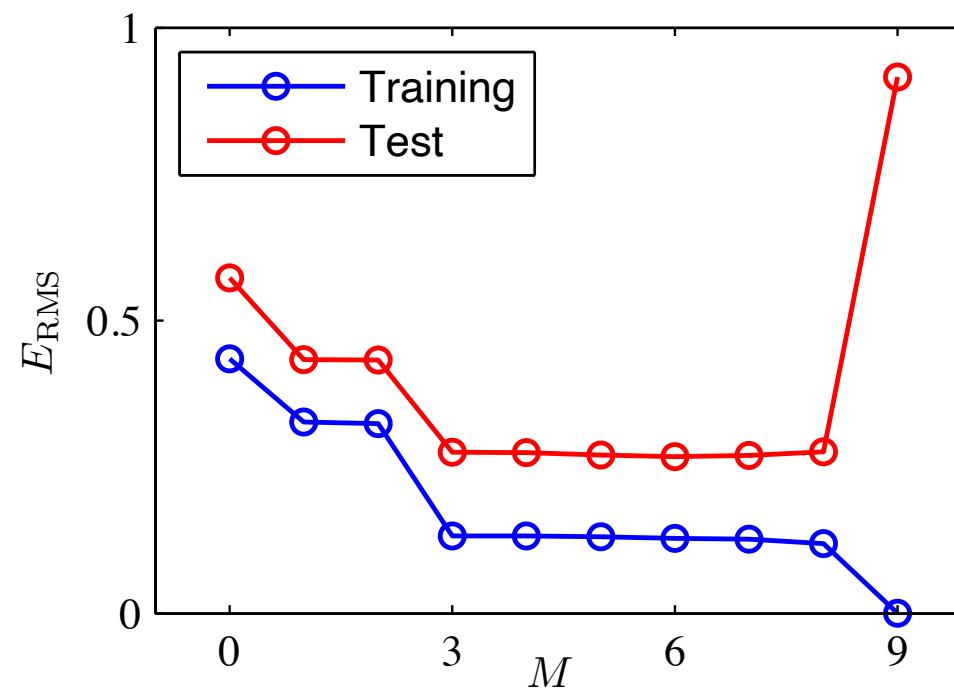


Обратите внимание, что константа ($M = 0$) и полиномы первого порядка ($M = 1$) дают довольно плохое соответствие данным. Полином третьего порядка ($M = 3$) даёт, по-видимому, наилучшее соответствие, в то время как полином более высокого порядка ($M = 9$) даёт отличное соответствие данным, то есть $E(\mathbf{w}^*) = 0$. 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

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

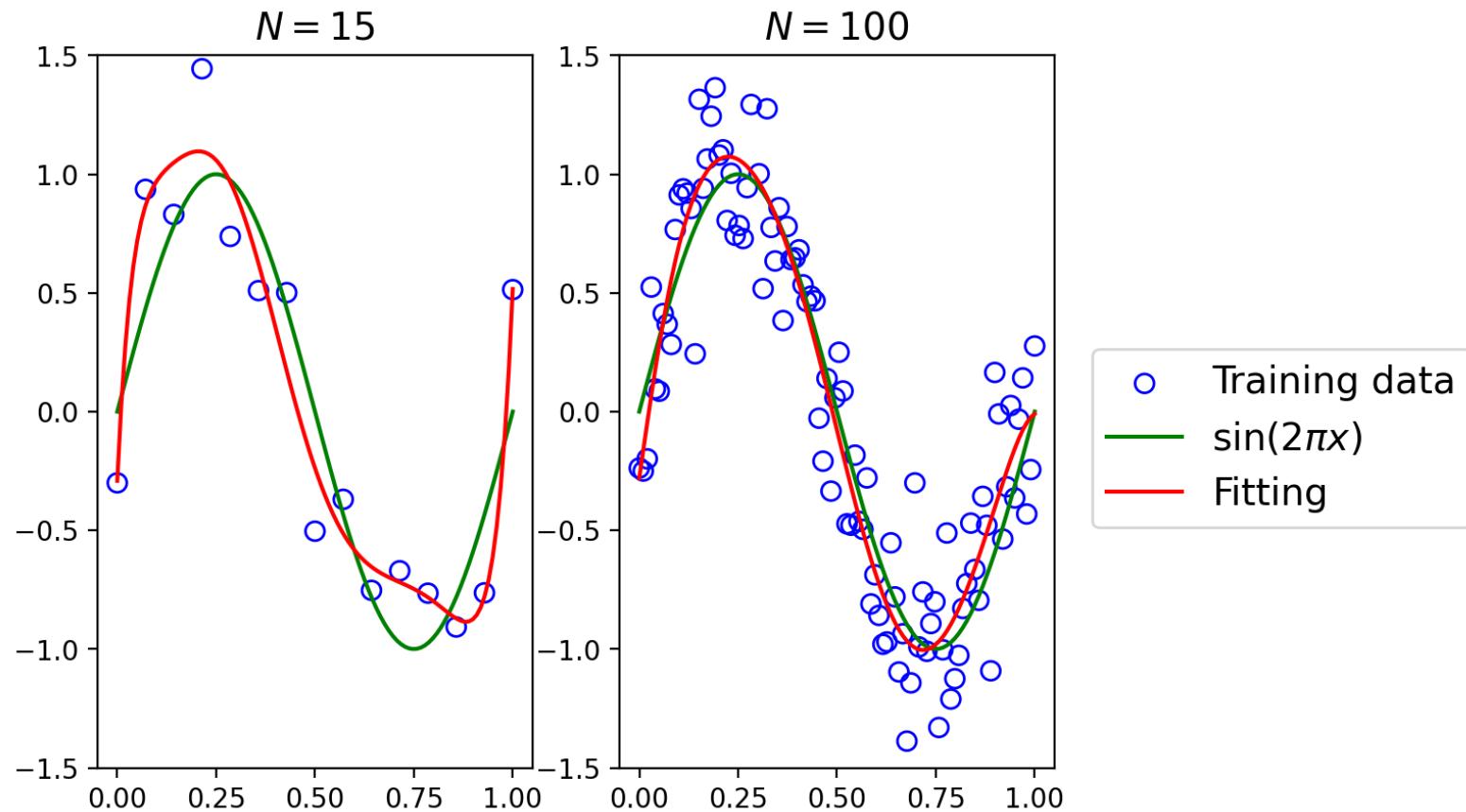


Кажется, что модель не должна работать хуже с увеличением M

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^*				-1061800,18
w_7^*				1042400,18
w_8^*				-557682,99
w_9^*				125201,43

Увеличение объема данных



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:

$$\tilde{E}(\mathbf{w}) = \frac{1}{2} \sum_{n=1}^N (y(x_n, \mathbf{w}) - t_n)^2 + \lambda \|\mathbf{w}\|_2^2$$

Такие методы известны как методы сжатия, поскольку они уменьшают значение коэффициентов. Частный случай квадратичной регуляризации называется *гребневой регрессией*. В нейронных сетях этот подход также известен как *метод уменьшения веса*.

Минимизация

Подобно предыдущему случаю, функцию ошибки гребня можно точно минимизировать в замкнутой форме 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 $M=9$

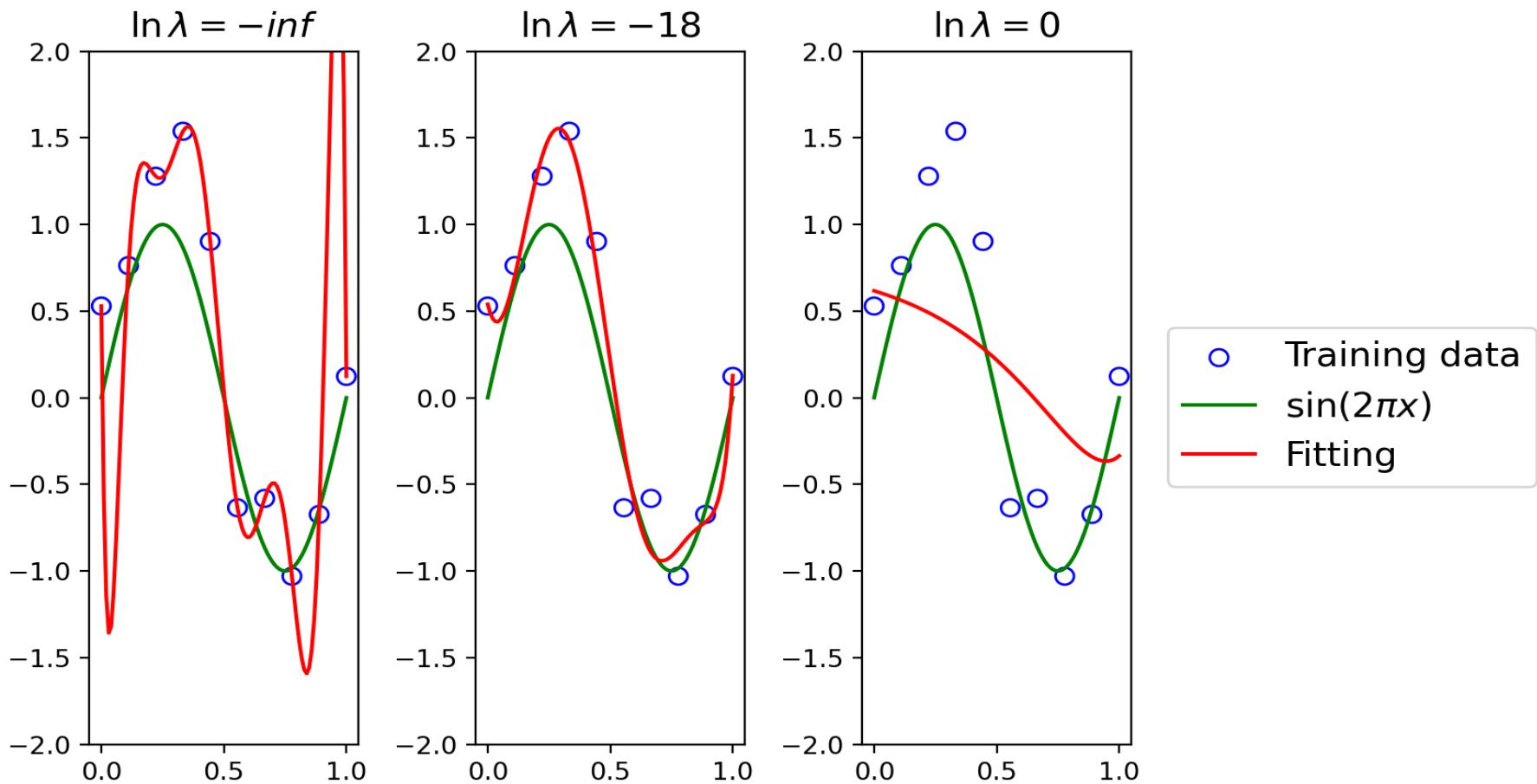
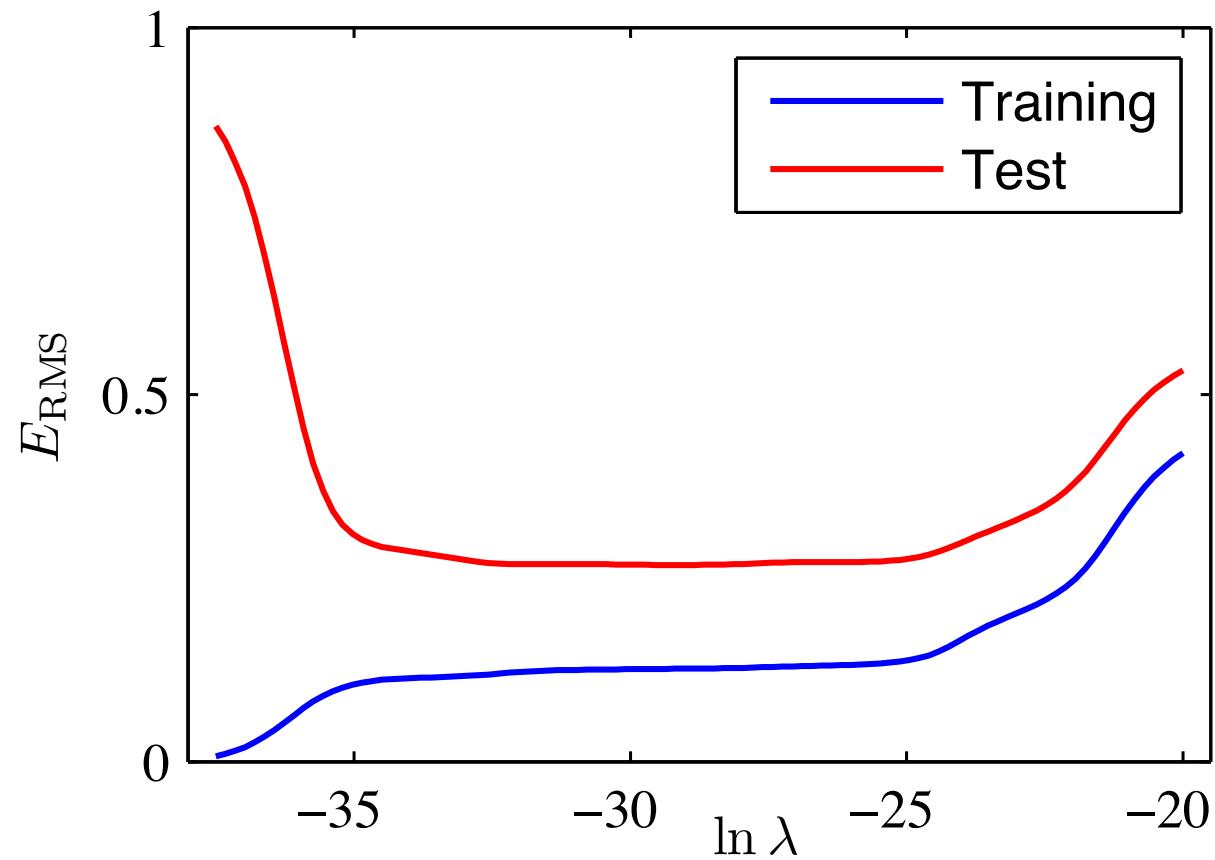


Таблица коэффициентов

j	$\ln\lambda = -\infty$	$\ln\lambda = -18$	$\ln\lambda = 0$
1	0.39	0.38	0.36
2	-135.04	-2.14	-0.46
3	3206.76	81.88	-0.39
4	-29215.92	-390.51	-0.22
5	139594.34	578.12	-0.05
6	-388863.80	-31.48	0.07
7	652373.24	-49.12	0.17
8	-648124.69	-28.57	0.25
9	350721.94	540.17	0.31
10	-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: $\int p(\mathbf{w}|\mathcal{D})d\mathbf{w} = 1 \Rightarrow p(\mathcal{D}) = \int p(\mathcal{D}|\mathbf{w})p(\mathbf{w})d\mathbf{w}$

Задача: монета бросается три раза и каждый раз выпадает орел. Какова вероятность получить орла в третьем броске?

Не стесняйтесь задавать уточняющие вопросы от них многое зависит.

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!