Introduction to Supervised Learning

Problem setup, feature types, assumptions about data

MISiS Mega Science, Spring Semester

Artem Maevskiy[1], Ekaterina Trofimova[1], Andrey Ustyuzhanin[1,2],

[1]National HSE University

[2]MISiS National University of Science and Technology

LAMBDA • HSE

March 2021

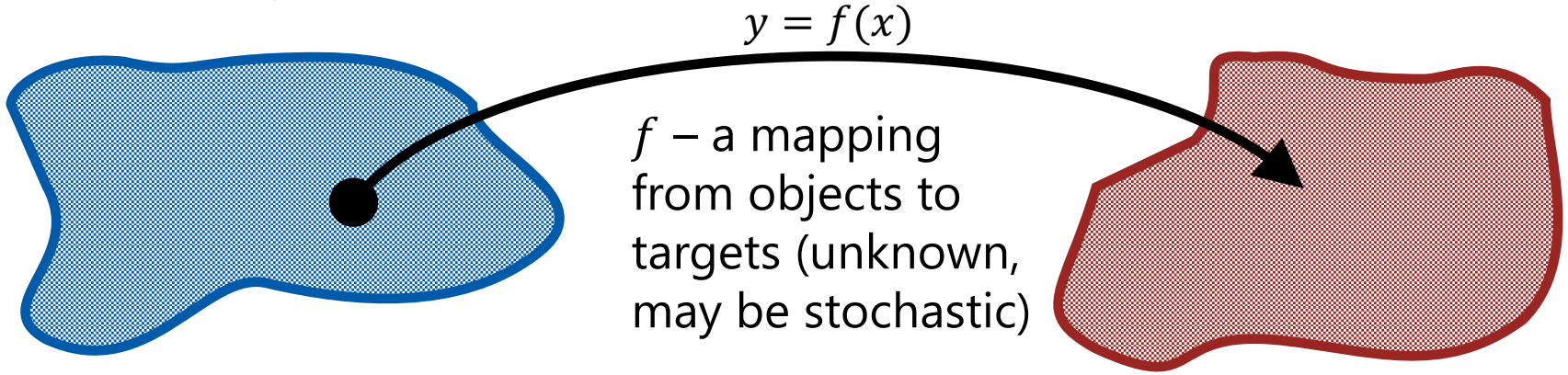
MISIS
National University of
Science and Technology

Supervised Learning

Problem setup

$\mathcal{X}$ – a set of objects

$\mathcal{Y}$ – a set of targets



A dataset: $D = \{(x_i,\ y_i) : i = 1,\ 2, \dots,\ N\}$

$$x_i \in \mathcal{X}, \qquad y_i = f(x_i) \in \mathcal{Y}$$

Goal: **approximate f given D**

i.e. learn to **recover targets from objects**

Examples

- Iris flower species classification

Objects

Individual flowers, described by the length and width of their sepals and petals

Targets

Species to which this particular flower belongs

Mapping

Different shapes of sepals and petals correspond to different species

(non-deterministic)



images source: wikipedia.org

Examples

- Spam filtering

Objects

E-mails (sequences of characters)

Targets

“spam” / “not spam”

Mapping

Message content defines whether it’s spam or not

(non-deterministic, varies from person to person)

Examples

- CAPTCHA recognition

Objects

CAPTCHA images (vectors of pixel brightness levels)

Targets

Sequences of characters

Mapping

Inverse of CAPTCHA generating algorithm

(almost deterministic, depending on the level of distortion)

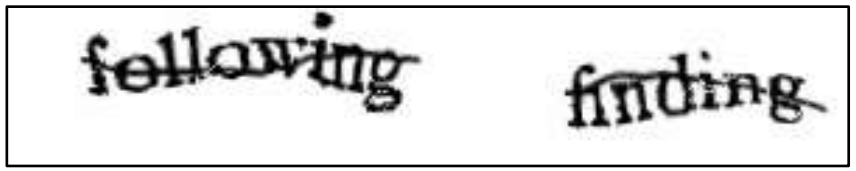


image source: wikipedia.org

Examples

- Particle identification in a HEP experiment

Objects

Particles, described by the detector responses (e.g. track parameters, calorimeter energy deposit, etc.)

Targets

Types of the particles ($e, \mu, p,$ etc…)

Mapping

Inverse of physical processes generating the detector response

(non-deterministic)

Features

Features

- Objects x_i are described by features x_i^j, i.e.:
  - $x_i = (x_i^1, x_i^2, \ldots, x_i^d)$
- many algorithms require that the dimensionality d of the data is same for all objects
  - In such case the objects may be organised in a design matrix:

$$X = \begin{bmatrix} x_1^1 & x_1^2 & \cdots & x_1^d \\ x_2^1 & x_2^2 & \cdots & x_2^d \\ \vdots & \vdots & \ddots & \vdots \\ x_N^1 & x_N^2 & \cdots & x_N^d \end{bmatrix}$$

features (→)

objects (↓)

Example: Iris dataset

sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)
5.1	3.5	1.4	0.2
4.9	3.0	1.4	0.2
4.7	3.2	1.3	0.2
4.6	3.1	1.5	0.2
5.0	3.6	1.4	0.2
...	...	...	...
6.7	3.0	5.2	2.3
6.3	2.5	5.0	1.9
6.5	3.0	5.2	2.0
6.2	3.4	5.4	2.3
5.9	3.0	5.1	1.8

In this example, all featuers are real numbers

Feature types

- Individual features x_i^j may be of various nature
- Common cases:
  - **Numeric features**, e.g.:
    - Sepal length
    - Height of a building
    - Particle transverse momentum
    - Number of hits on the particle track

Feature types

- Individual features x_i^j may be of various nature
- Common cases:
  - **Categorical**

nominal (no order implied), e.g.:

Color
City of birth
Particle type

ordinal (values can be compared, though pairwise differences are not defined), e.g.:

Level of education
Particle passing loose, medium or tight selection criteria
Age category (child, teen, adult, etc.)

Feature types

- Individual features x_i^j may be of various nature
- Common cases:
  - **Binary**, e.g.:
    - True / False
  - Can be treated as numeric (0/1 or –1/+1)

Learning Algorithms

Machine Learning Algorithm

Algorithm $\mathcal{A}$:

given a dataset $D = \{(x_i,\ y_i) : i = 1,\ 2, \dots,\ N\}$

$$x_i \in \mathcal{X},\ y_i = f(x_i) \in \mathcal{Y}$$

returns an approximation $\hat{f} = \mathcal{A}(D)$ to the true dependence f.

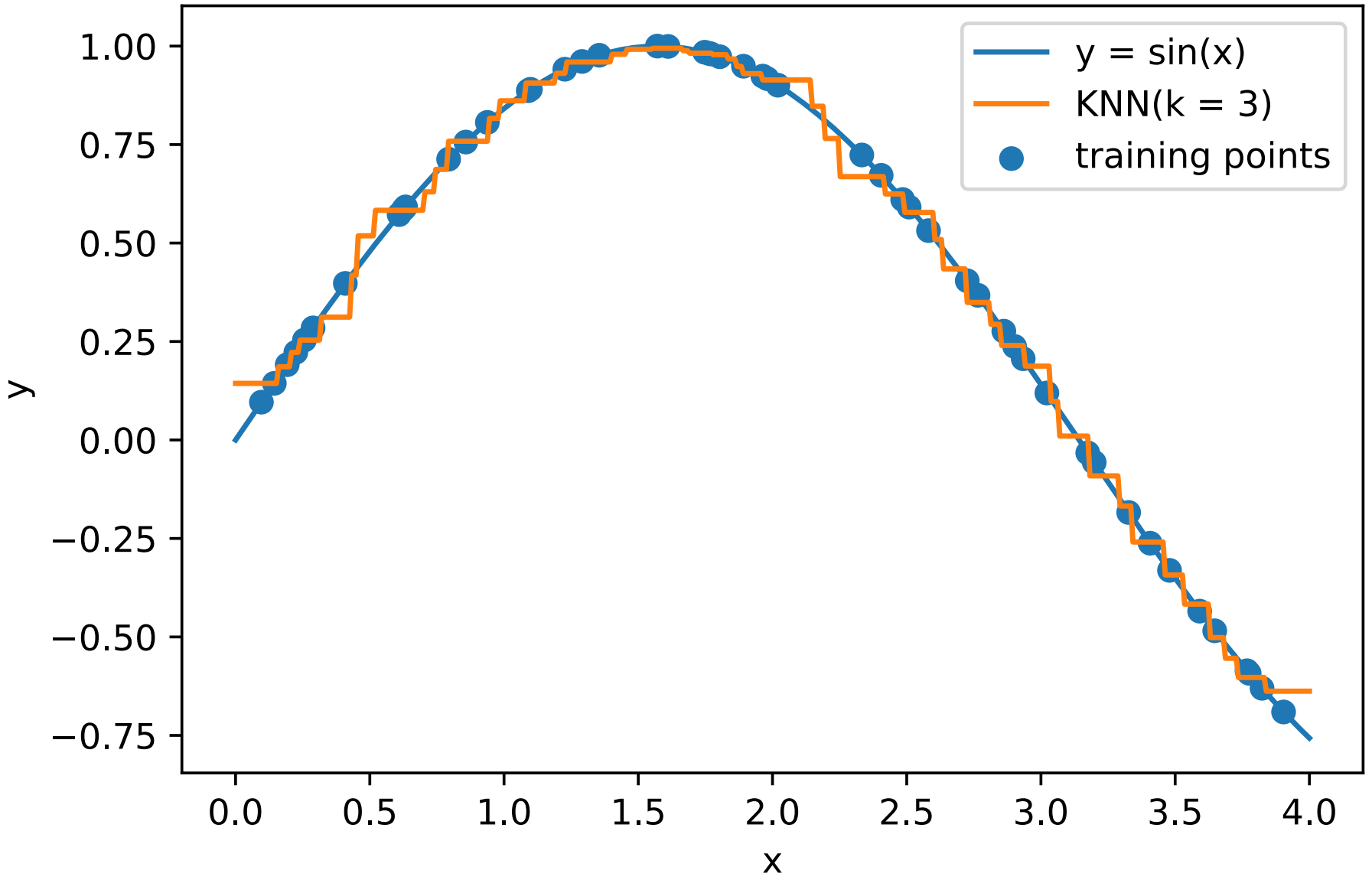
Example: k nearest neighbors (kNN)

- Idea: close objects should have similar targets
- Why don't we look up k closest (by some metric of the feature space) objects in the dataset and average their targets:

$$\hat{f}(x) = \frac{1}{k} \sum_{i:\, x_i \in D_x^k} y_i$$

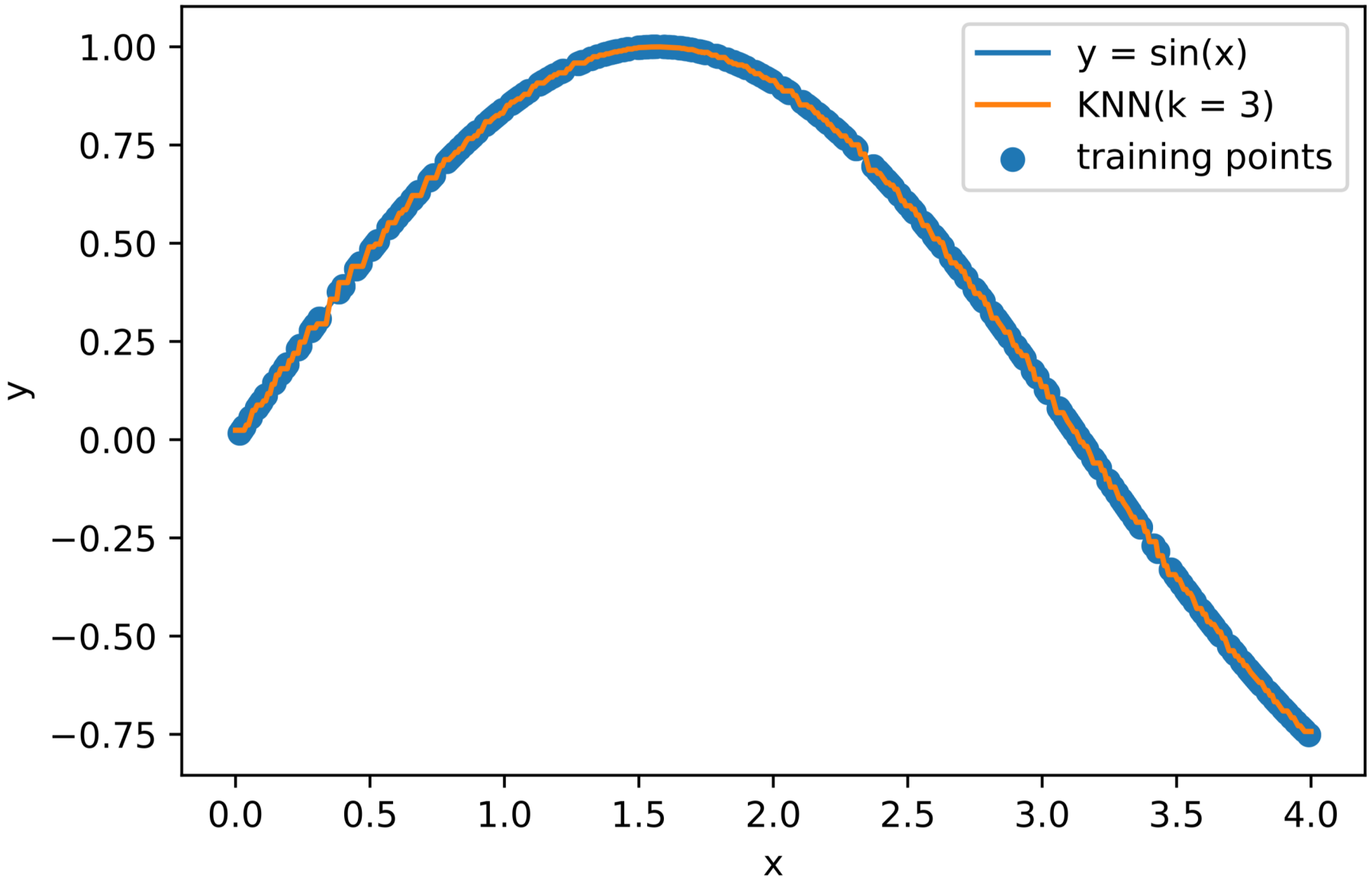
D_x^k – set of k objects from D closest to x

Example: k nearest neighbors



training points: 50

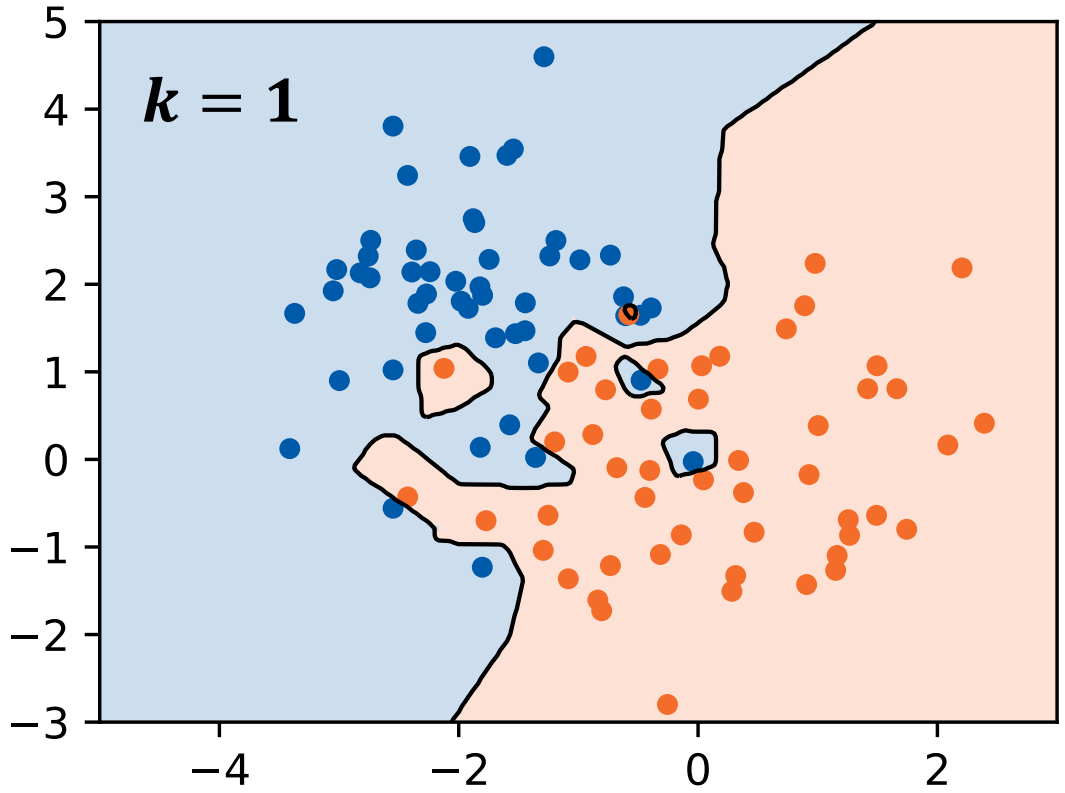
Example: k nearest neighbors

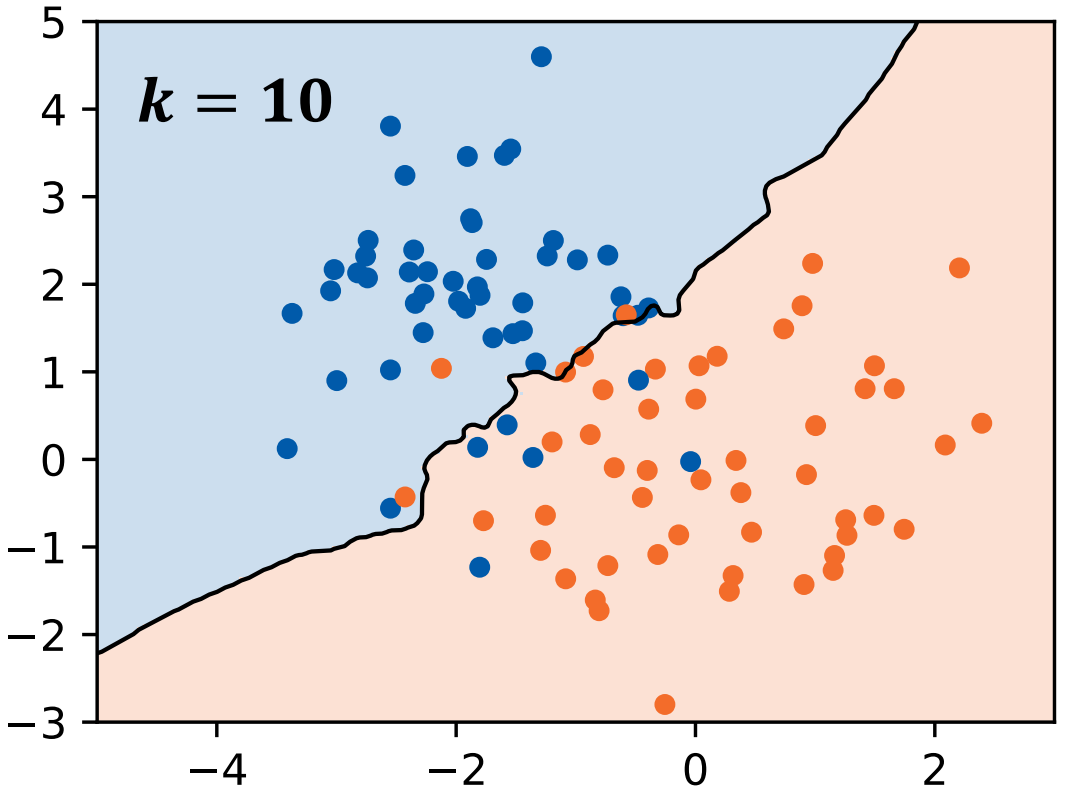


training points: 250

More data = better approximation

Example: k nearest neighbors





Classification example

Loss function

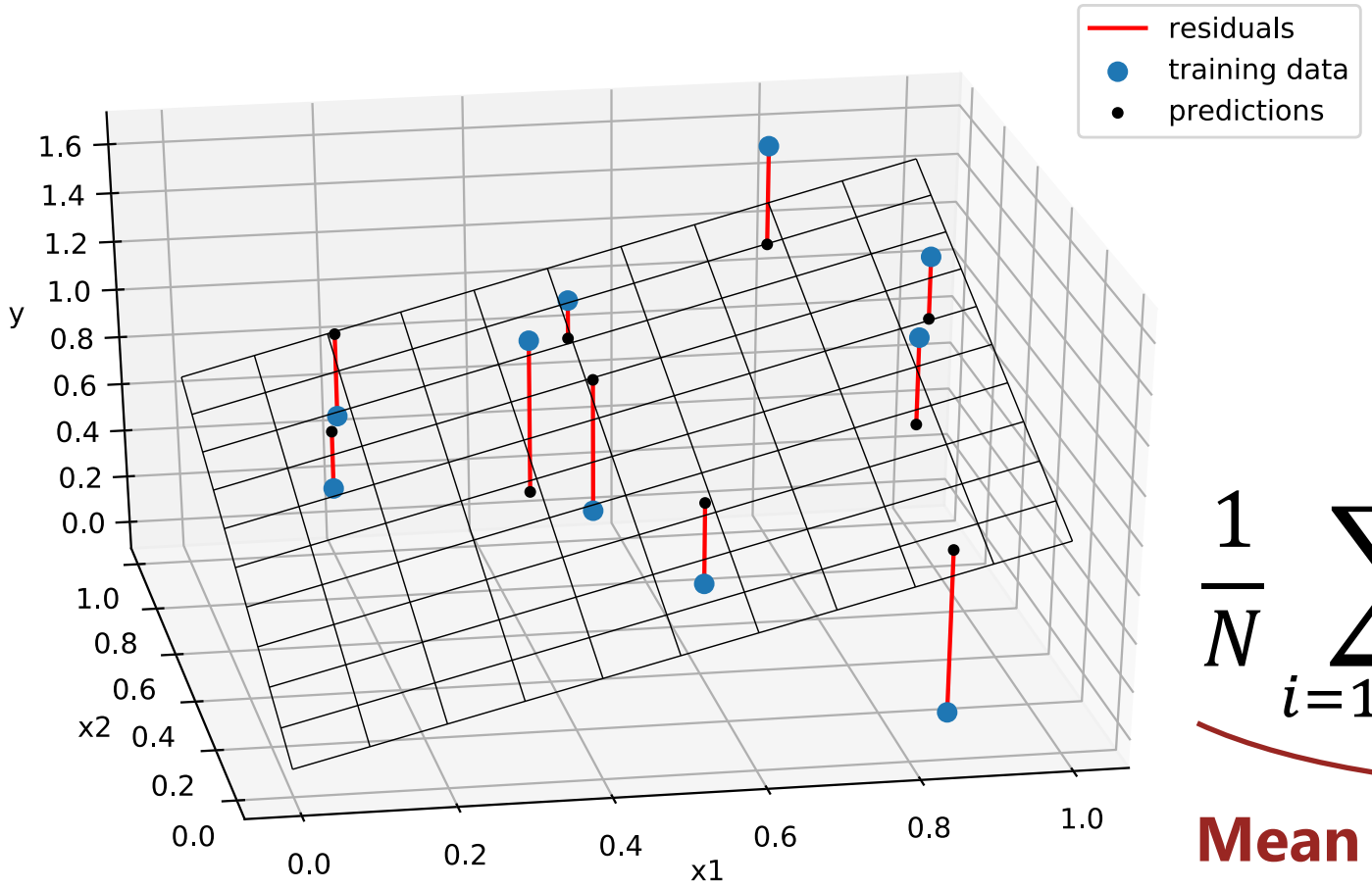
- How does an algorithm find the approximation $\hat{f} = \mathcal{A}(D)$ to the true mapping function?
- Many algorithms work by solving an **optimization task**
- We can measure the quality of a prediction for a single object x_i with a **loss function** $\mathcal{L} = \mathcal{L}\left(y_i, \hat{f}(x_i)\right)$
- Then, learning can be formulated as a **loss minimization** problem:

$$\hat{f} = \operatorname{argmin}_{\tilde{f}} \mathop{\mathbb{E}}_{(x,y) \in D} \mathcal{L}\left(y, \tilde{f}(x)\right)$$

E.g. squared error:

$$\mathcal{L} = \left(y_i - \hat{f}(x_i)\right)^2$$

Example: linear regression



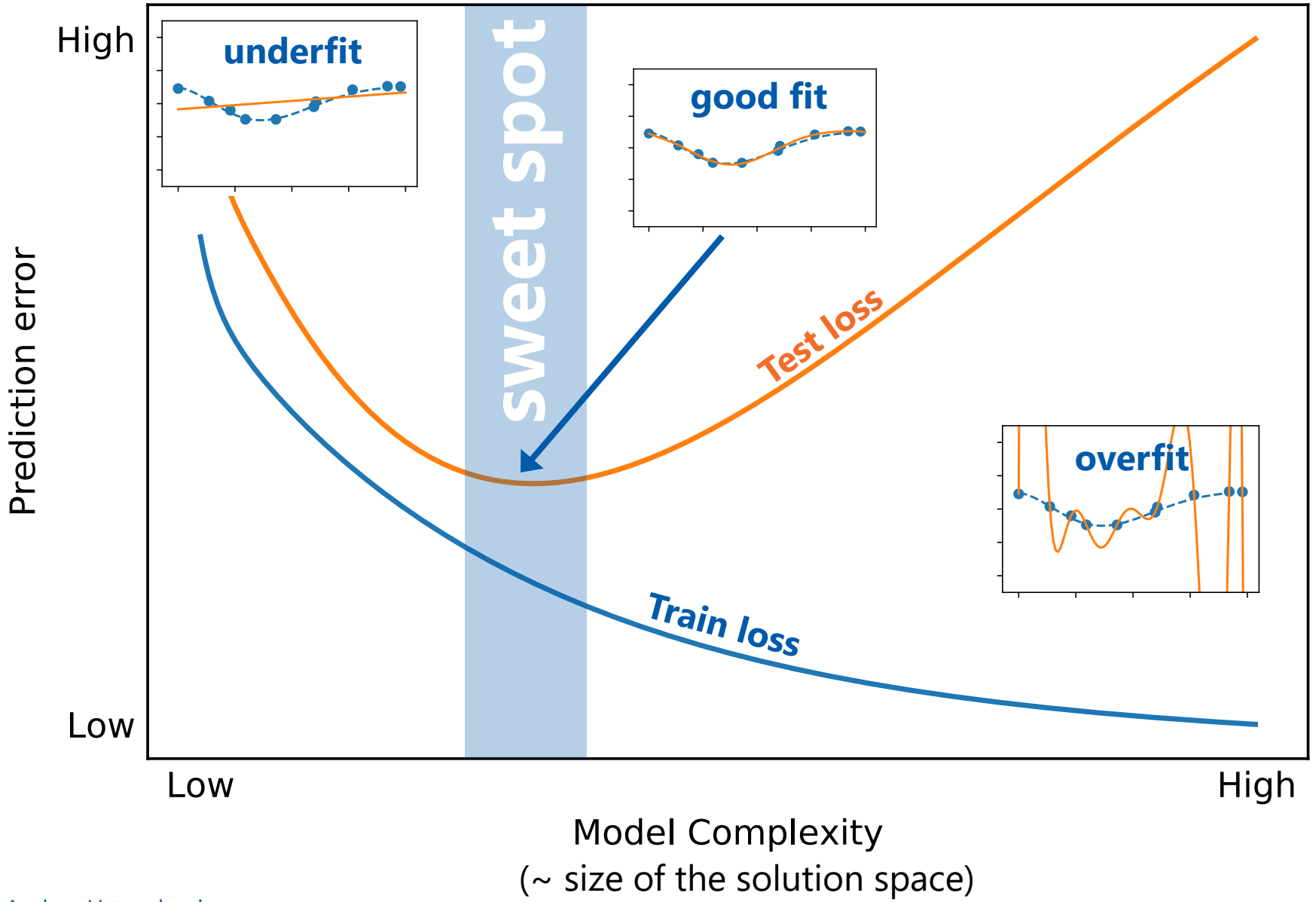
$$\hat{f}_{w,b}(x) = w^{\mathrm{T}}x + b$$

$$w \in \mathbb{R}^d$$
$$b \in \mathbb{R}$$
$$x \in \mathcal{X} \subset \mathbb{R}^d$$

$$\frac{1}{N}\sum_{i=1\ldots N}\left(y_i - \hat{f}_{w,b}(x_i)\right)^2 \longrightarrow \min_{w,b}$$

Mean Squared Error (MSE loss)

How to check whether a model is good?



Check the loss on the **test data** – i.e. data that the learning algorithm "hasn't seen"

The goal is to find the **right level of limitations** – not too strict, not too loose

Summary

- Supervised Machine Learning algorithms build approximations $\hat{f} = \mathcal{A}(D)$ to the true dependence f
- Features may be of various nature: real, categorical, binary
- Machine Learning algorithms can be defined as expected risk minimization tasks
- Choosing the right model = applying the right assumptions about the data
- Use test data to detect underfitting and overfitting

- Food for thought: how can Linear Regression model be used to fit a n-th degree polynomial?

Quiz / Questions

Among the models below which one has the lowest complexity for a fixed training set size (i.e. restricts the solution space the most)?

A. kNN (k = 1)
B. kNN (k = 5)
C. kNN (k = 20)
D. All of these are of same complexity

Thank you!


austyuzhanin@hse.ru


anaderiRu

Andrey Ustyuzhanin