Data Analytics and Mining

Intro to Machine Learning, Supervised Learning, Regression

Data Analytics and Mining, 2024

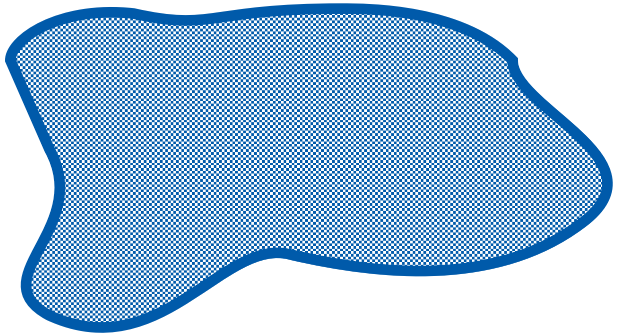
Majid Sohrabi
National Research University Higher School of Economics

November 8, 2024

Supervised Learning

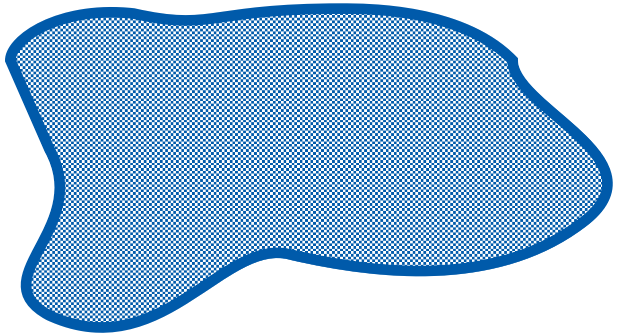
Problem setup

$\mathcal{X}$ – a set of objects

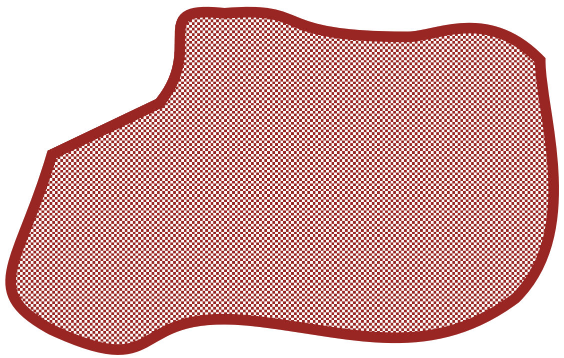


Problem setup

$\mathcal{X}$ – a set of objects



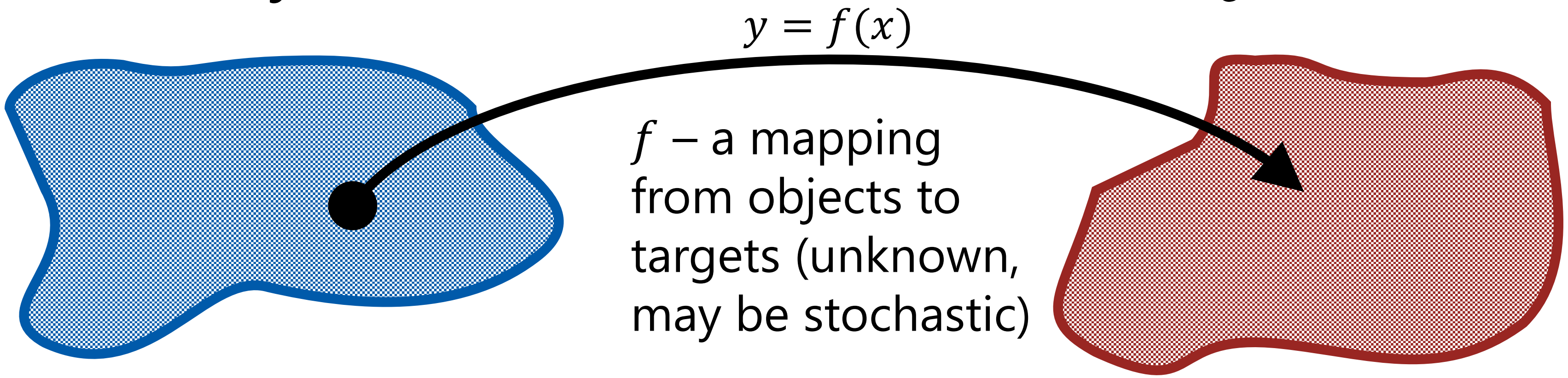
$\mathcal{Y}$ – a set of targets



Problem setup

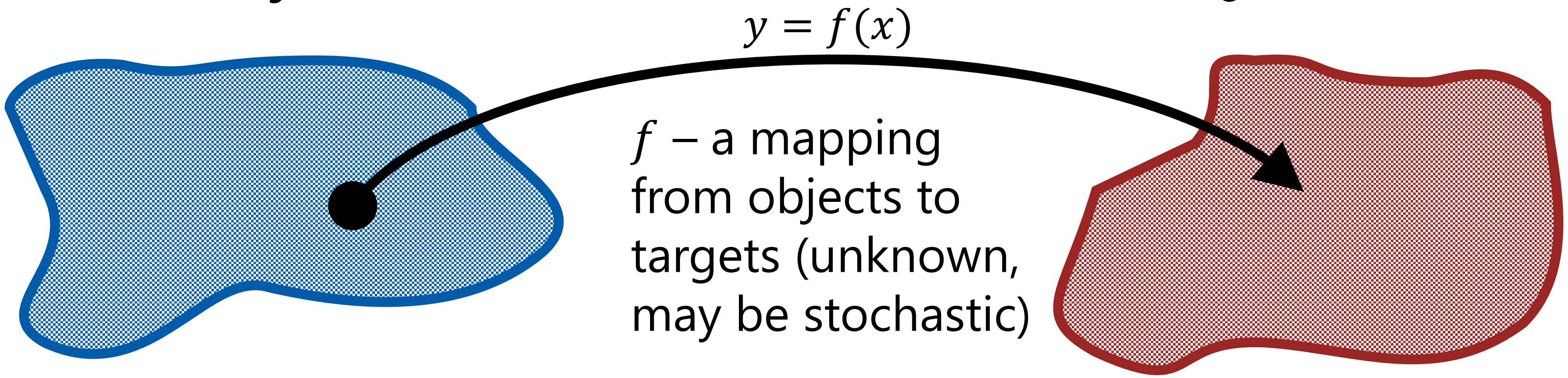
$\mathcal{X}$ – a set of objects

$\mathcal{Y}$ – a set of targets



Problem setup

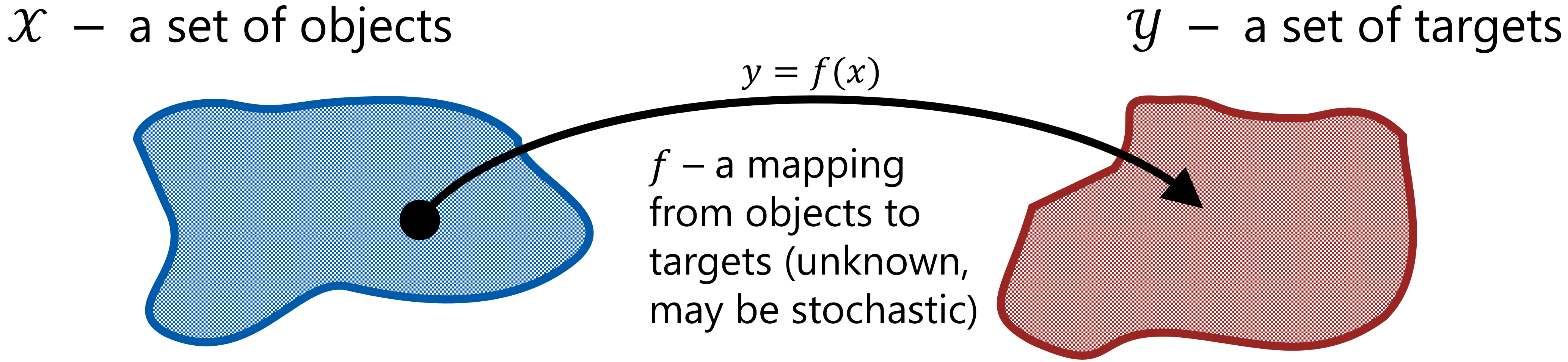
$\mathcal{X}$ – a set of objects $\qquad$ $\mathcal{Y}$ – a set of targets



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

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

Problem setup



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)

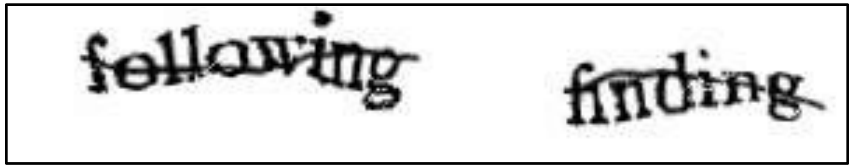


image source: wikipedia.org

Features

Features

Objects x_i are described by features x_i^j, i.e.:

– It's a vector $x_i = (x_i^1, x_i^2, \dots, x_i^d)$

Features

Objects x_i are described by features x_i^j, i.e.:

- It's a vector $x_i = (x_i^1, x_i^2, \dots, x_i^d)$

many algorithms require that the dimensionality d of the data is same for all objects

Features

Objects x_i are described by features x_i^j, i.e.:

– It's a vector $x_i = (x_i^1, x_i^2, \dots, x_i^d)$

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:

$$X = \begin{array}{cc} \xrightarrow{\textit{features}} & \\ \begin{bmatrix} x_1^1 & x_1^2 & \cdots & x_1^d \\ x_2^1 & x_2^2 & \cdots & x_2^d \\ \vdots & \vdots & \ddots & \vdots \\ x_N^1 & x_N^2 & \cdots & x_N^d \end{bmatrix} & \Big\downarrow \textit{objects} \end{array}$$

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
  - Temperature
  - Price
  - Age
  - Etc.

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
- Name

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

- Level of education
- 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)

One-hot encoding

How does one convert categorical feature to numeric?

One-hot encoding

How does one convert categorical feature to numeric?

- Assigning each category a number (e.g. ”red” = 1, “green” = 2, etc.) may have negative effect on the learning algorithm

One-hot encoding

How does one convert categorical feature to numeric?

– Assigning each category a number (e.g. "red" = 1, "green" = 2, etc.) may have negative effect on the learning algorithm

One-hot encoding – simple trick to convert categorical feature to numeric:

color
"red"
"red"
"blue"
"green"
"blue"

→

is_blue	**is_red**	**is_green**
0	1	0
0	1	0
1	0	0
0	0	1
1	0	0

A trick for ordinal features

One-hot encoding may be used, though it loses the information about the relations between the categories

A trick for ordinal features

One-hot encoding may be used, though it loses the information about the relations between the categories

Similar trick:

Academic degree		**is_bachelor**	**is_master**	**is_PhD**
“none”	→	0	0	0
“bachelor”		1	0	0
“master”		1	1	0
“PhD”		1	1	1
“master”		1	1	0

More advanced encoding techniques

See https://contrib.scikit-learn.org/category_encoders/index.html

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)

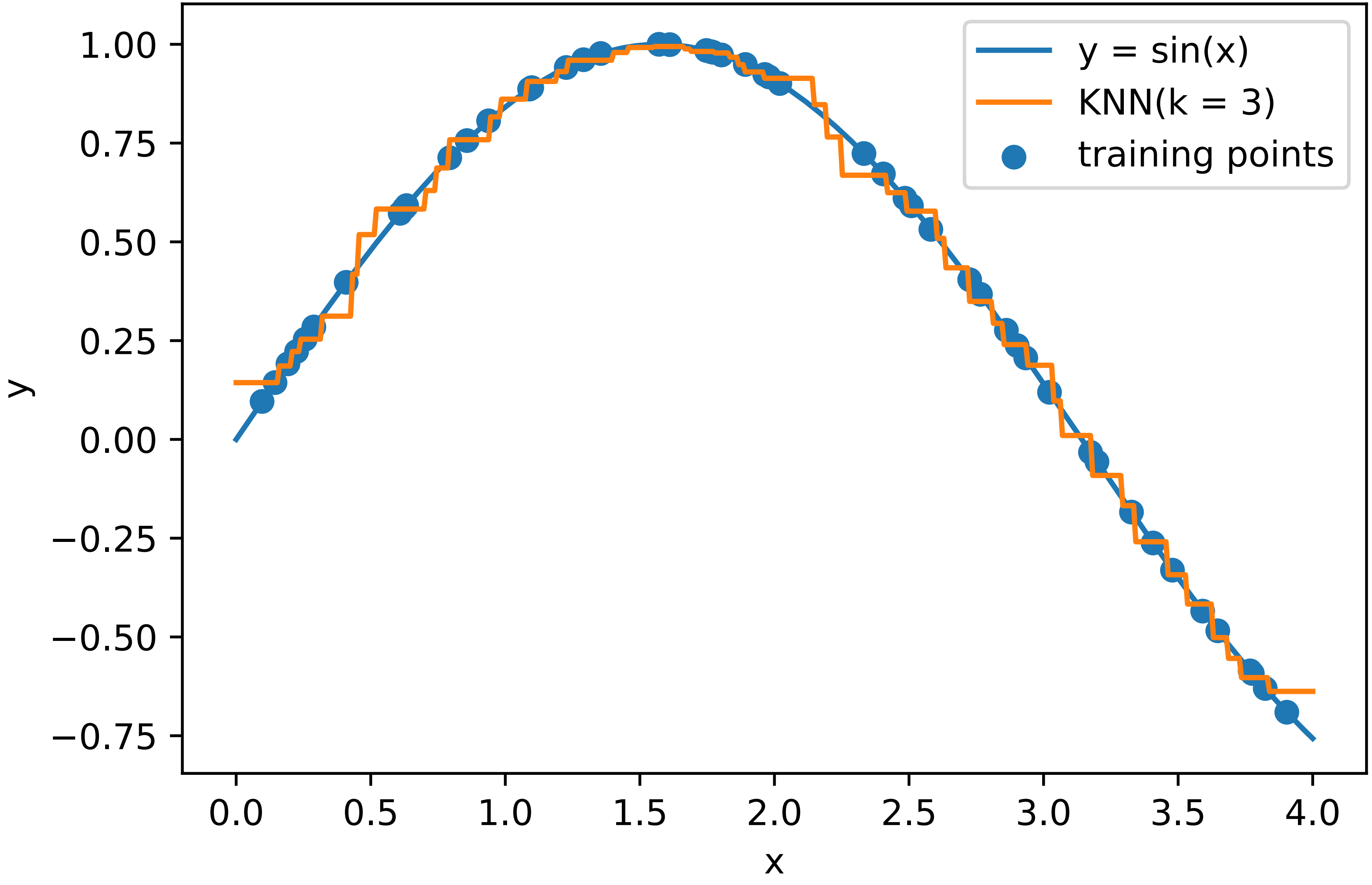
Idea: close objects should have similar targets

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

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

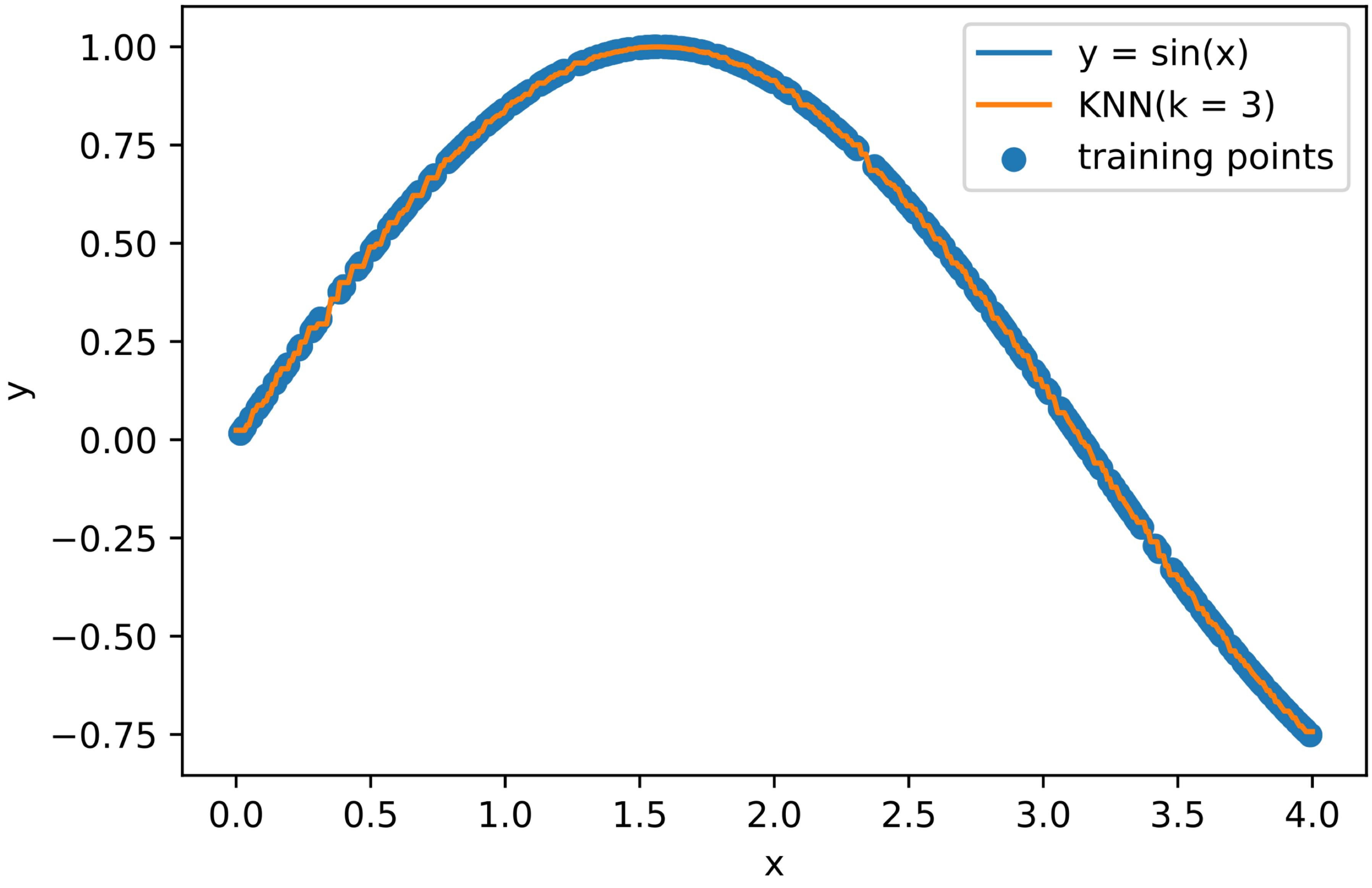
D_x^k – set of k objects from D closest to x

Example: k nearest neighbors



training points: 50

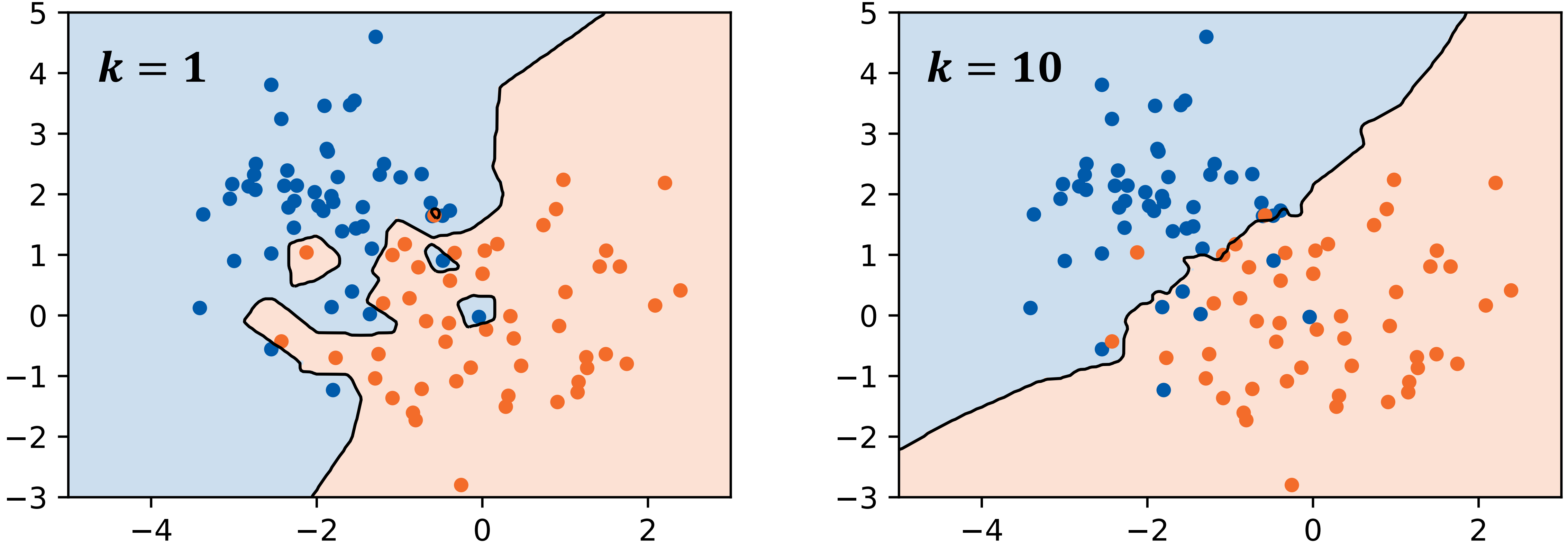
Example: k nearest neighbors



training points: 250

More data = **better approximation**

Example: k nearest neighbors



$$\hat{f}(x) = \underset{C}{\operatorname{argmax}} \sum_{i:\, x_i \in D_x^k} \mathbb{I}[y_i = C]$$

Classification example

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

Loss function

How does an algorithm find the approximation $\hat{f} = \mathcal{A}(D)$ to the true mapping function?

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**

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** $\mathcal{L} = \mathcal{L}\left(y_i, \hat{f}(x_i)\right)$

E.g. squared error:
$\boldsymbol{\mathcal{L} = \left(y_i - \hat{f}(x_i)\right)^2}$

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** $\mathcal{L} = \mathcal{L}\left(y_i, \hat{f}(x_i)\right)$

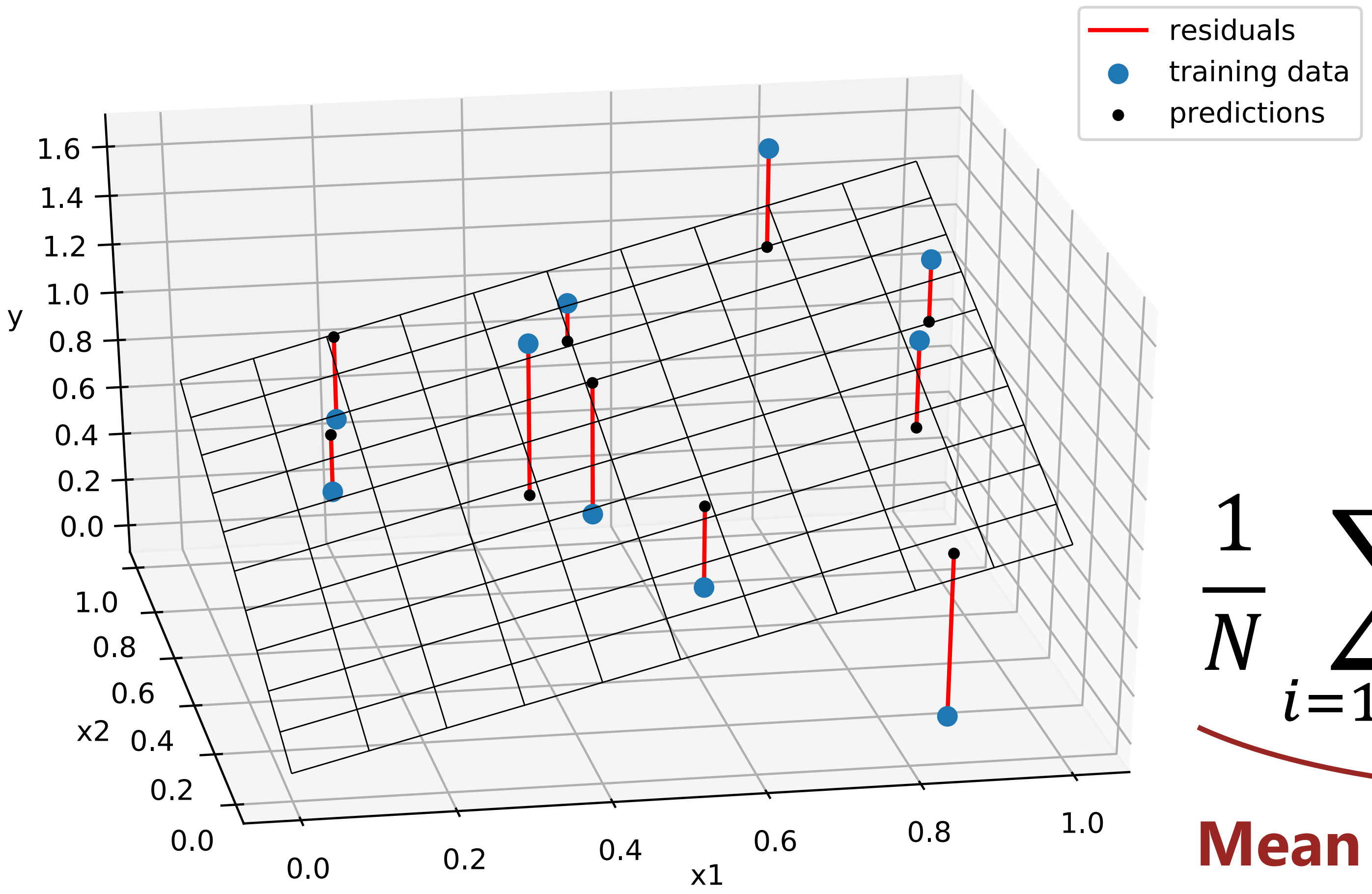
E.g. squared error:

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

Then, learning (or training) can be formulated as a **loss minimization** problem:

$$\hat{f} = \underset{\tilde{f}}{\operatorname{argmin}} \underset{(x,y) \in D}{\mathbb{E}} \mathcal{L}\big(y, \tilde{f}(x)\big)$$

Example: linear regression



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

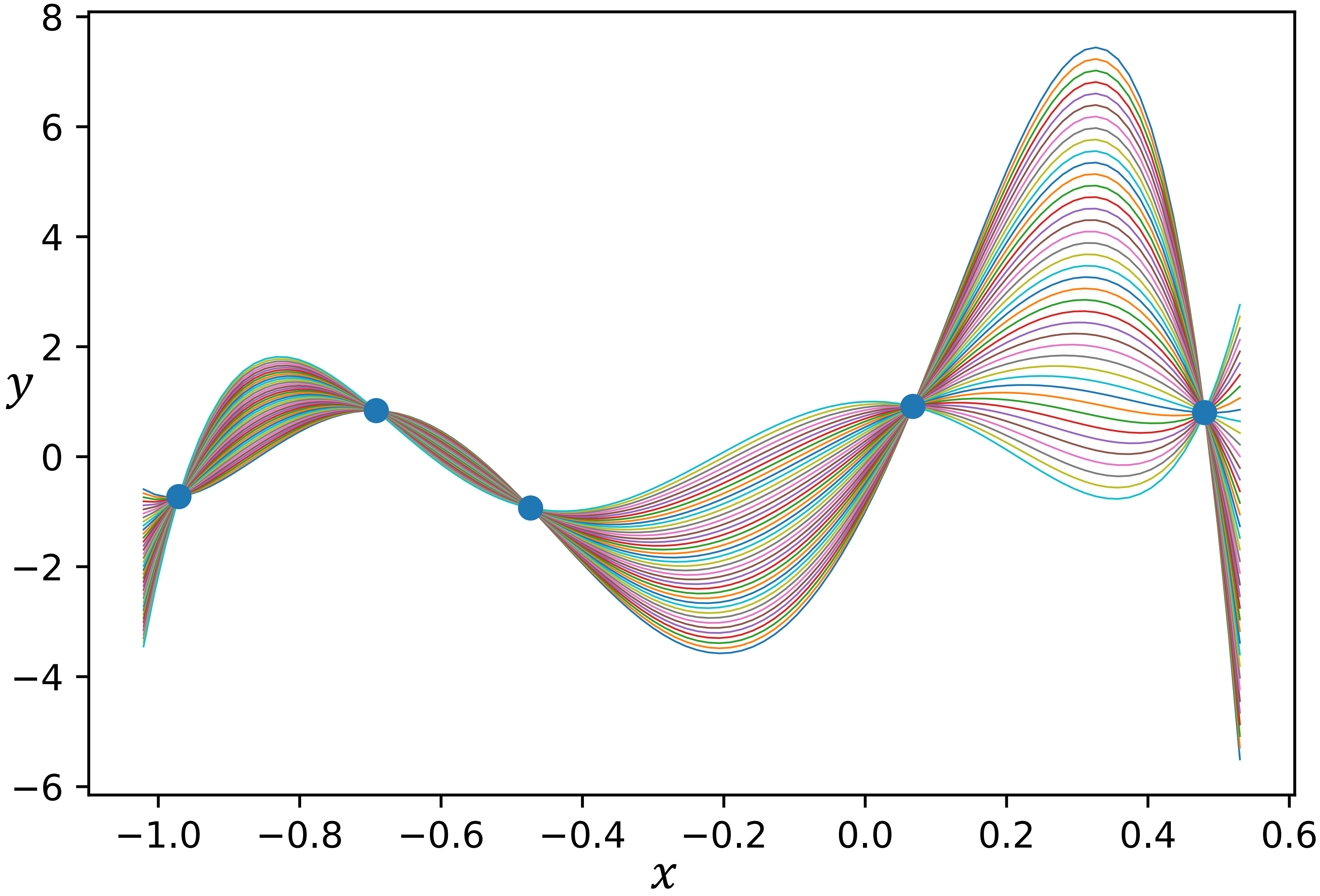
$$w \in \mathbb{R}^d$$
$$b \in \mathbb{R}$$
$$x \in \mathcal{X} \subset \mathbb{R}^d$$

$$\underbrace{\frac{1}{N}\sum_{i=1\ldots N}\left(y_i - \hat{f}_{w,b}(x_i)\right)^2}_{\text{Mean Squared Error (MSE loss)}} \xrightarrow[w,b]{} \min$$

Mean Squared Error
(MSE loss)

Assumptions about data

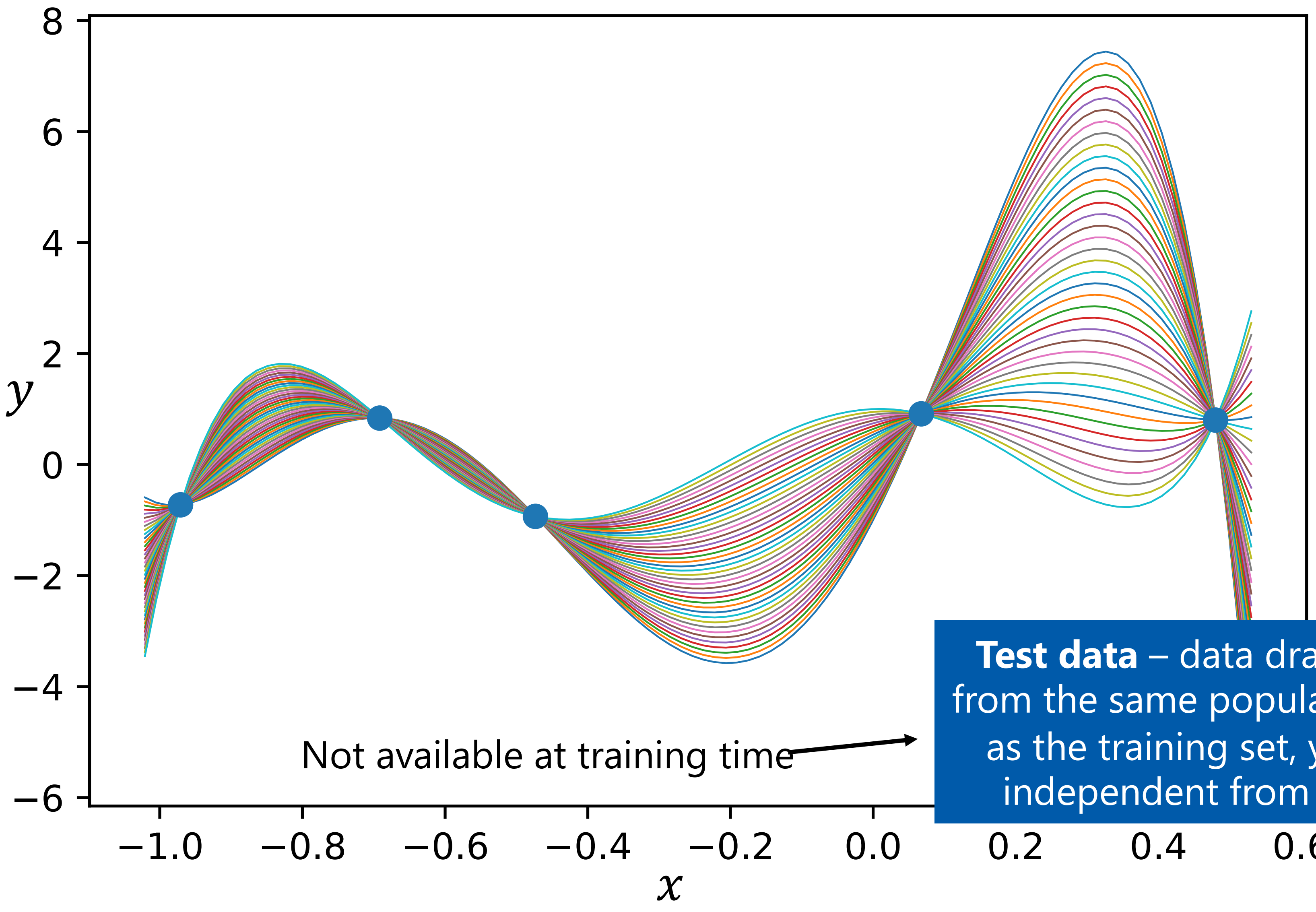
No assumptions = Infinitely many solutions



Any of these curves minimizes the loss

We want **expected loss over population** to be minimized

No assumptions = Infinitely many solutions

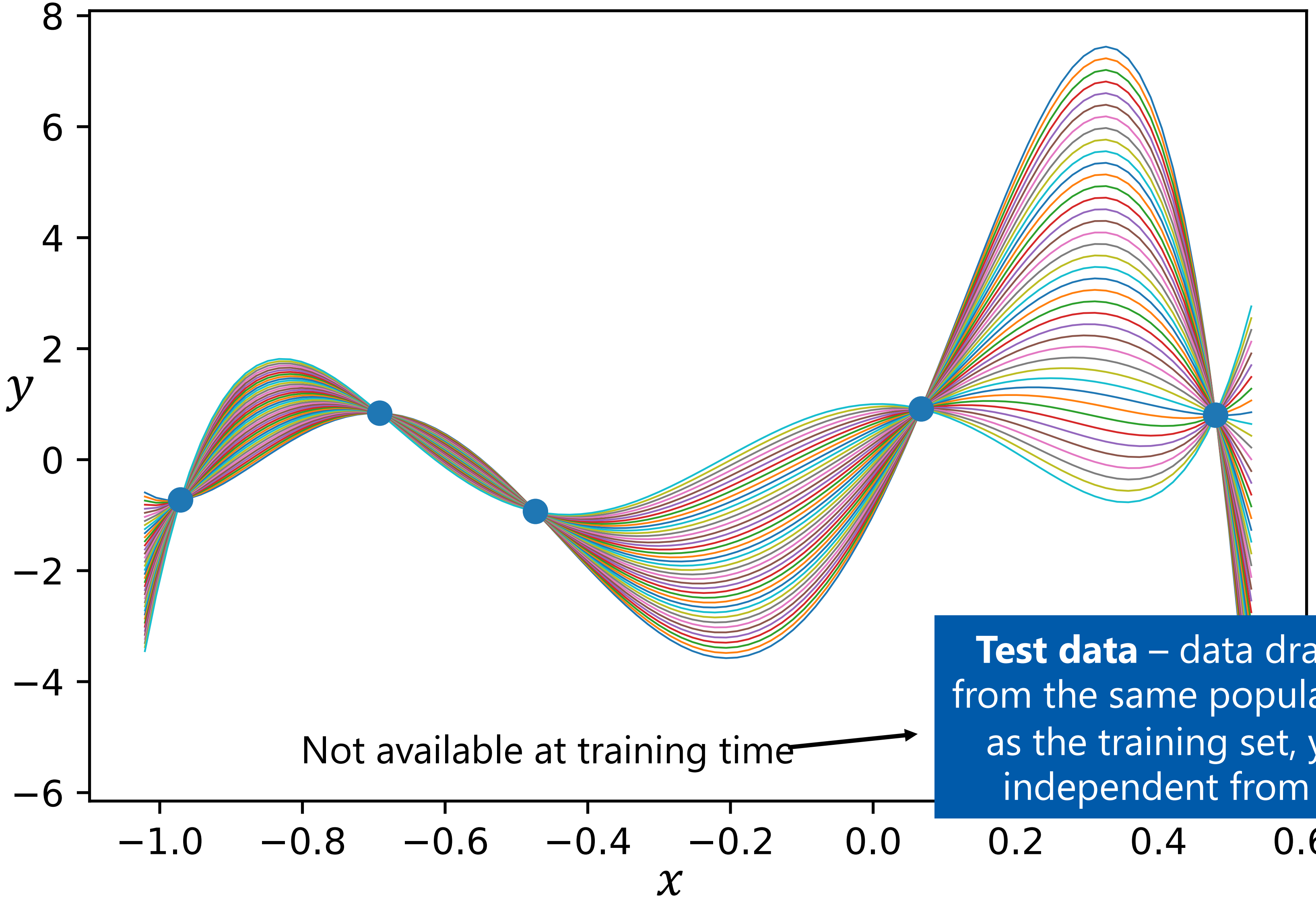


Test data – data drawn from the same population as the training set, yet independent from it

Any of these curves minimizes the loss

We want **expected loss over population** to be minimized

No assumptions = Infinitely many solutions



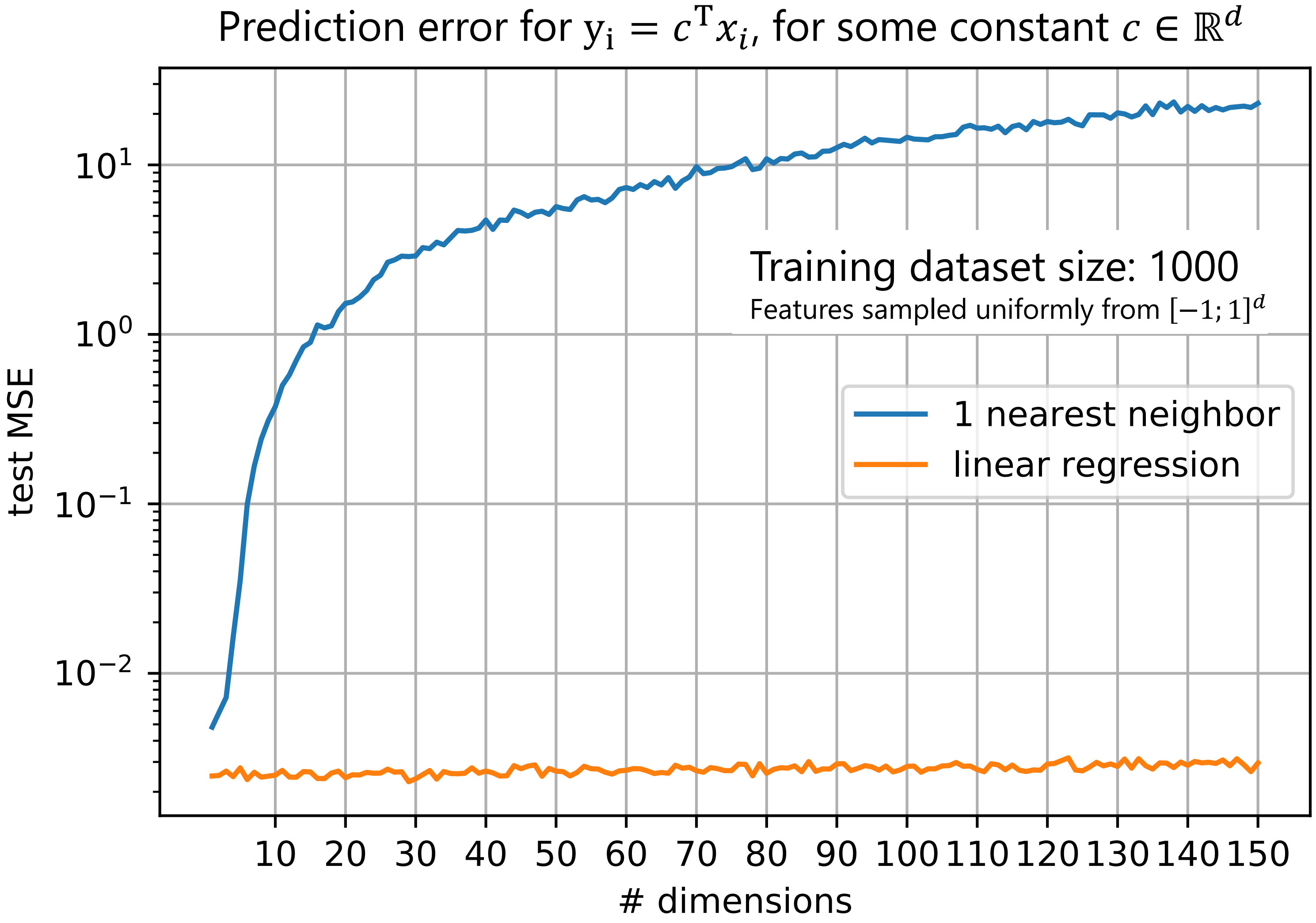
Test data – data drawn from the same population as the training set, yet independent from it

Any of these curves minimizes the loss

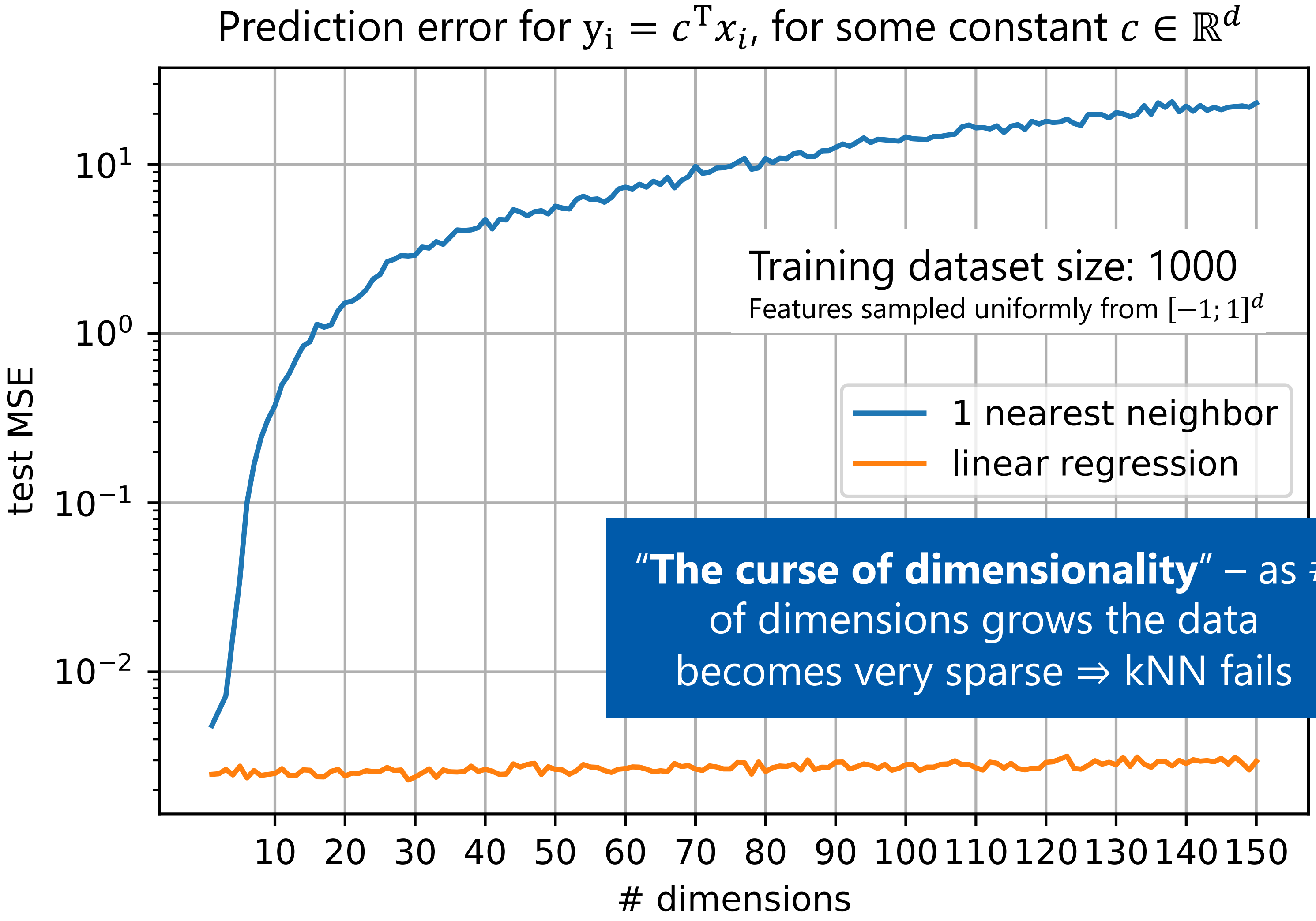
We want **expected loss over population** to be minimized

Need to **assume some structure** of the data, common to **training** and **testing** data

Example: kNN(k = 1) VS Linear Regression

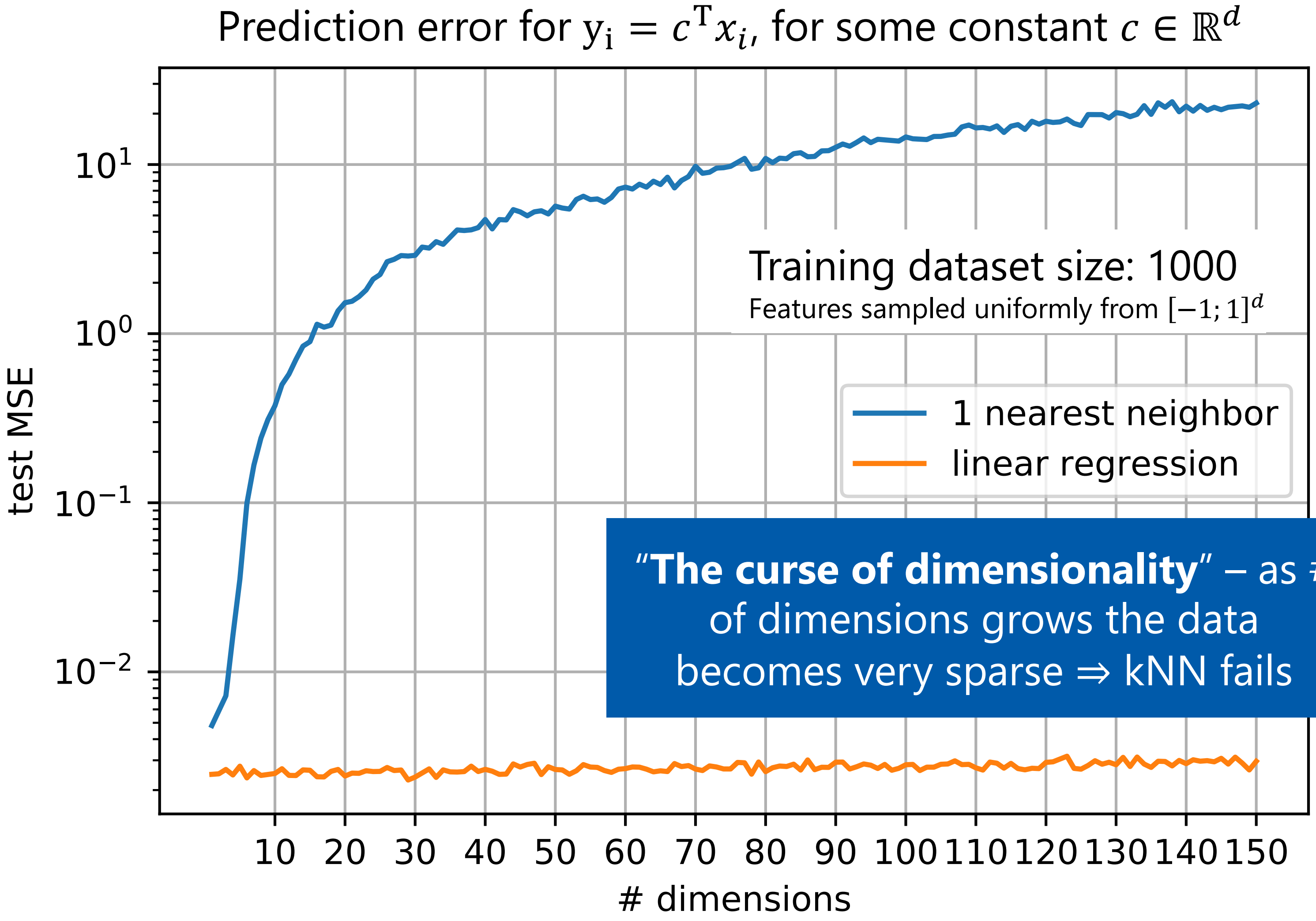


Example: kNN(k = 1) VS Linear Regression



"**The curse of dimensionality**" – as # of dimensions grows the data becomes very sparse ⇒ kNN fails

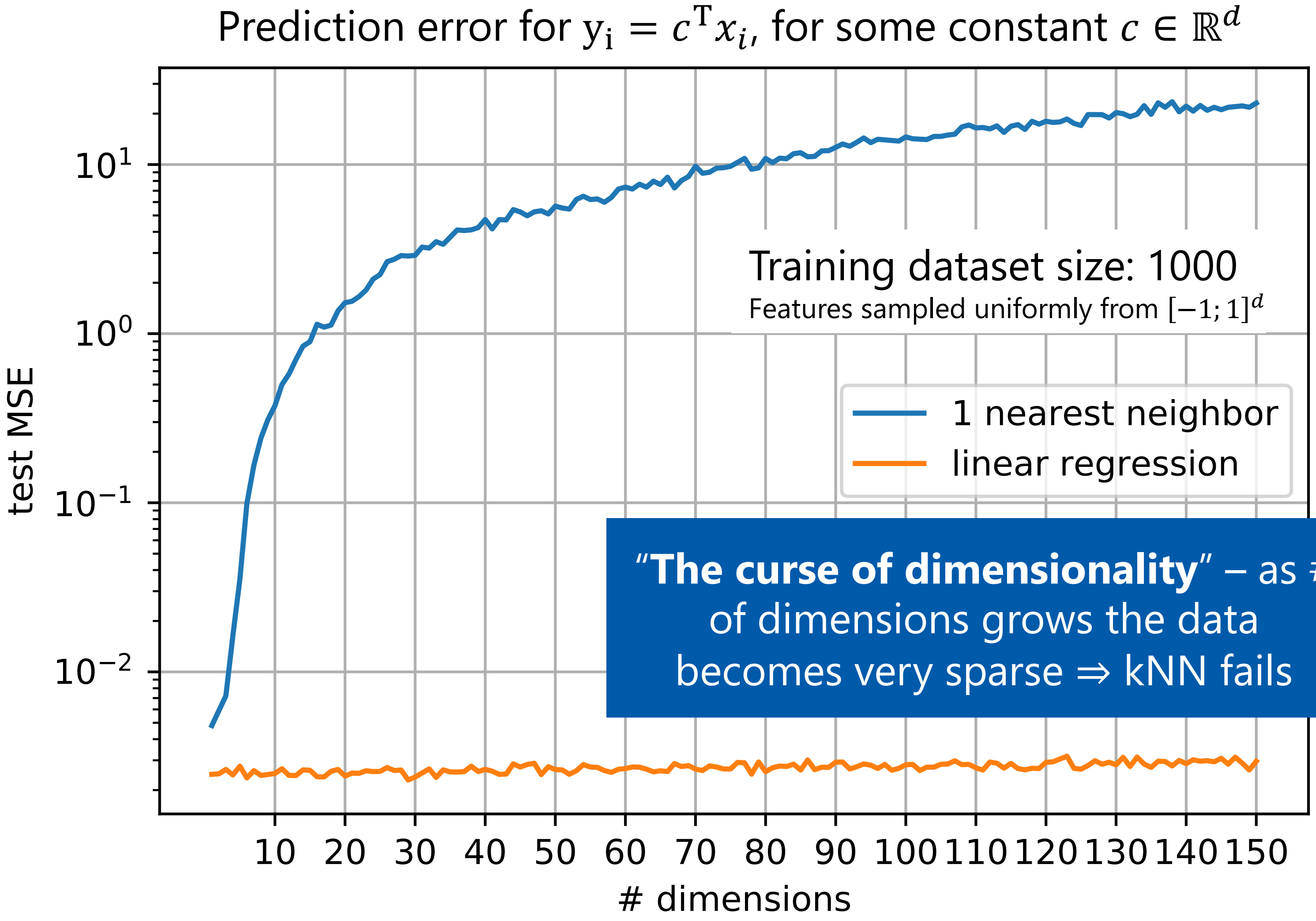
Example: kNN(k = 1) VS Linear Regression



"**The curse of dimensionality**" – as # of dimensions grows the data becomes very sparse ⇒ kNN fails

Assumption for kNN:
"**similar objects have similar targets**"

Example: kNN(k = 1) VS Linear Regression



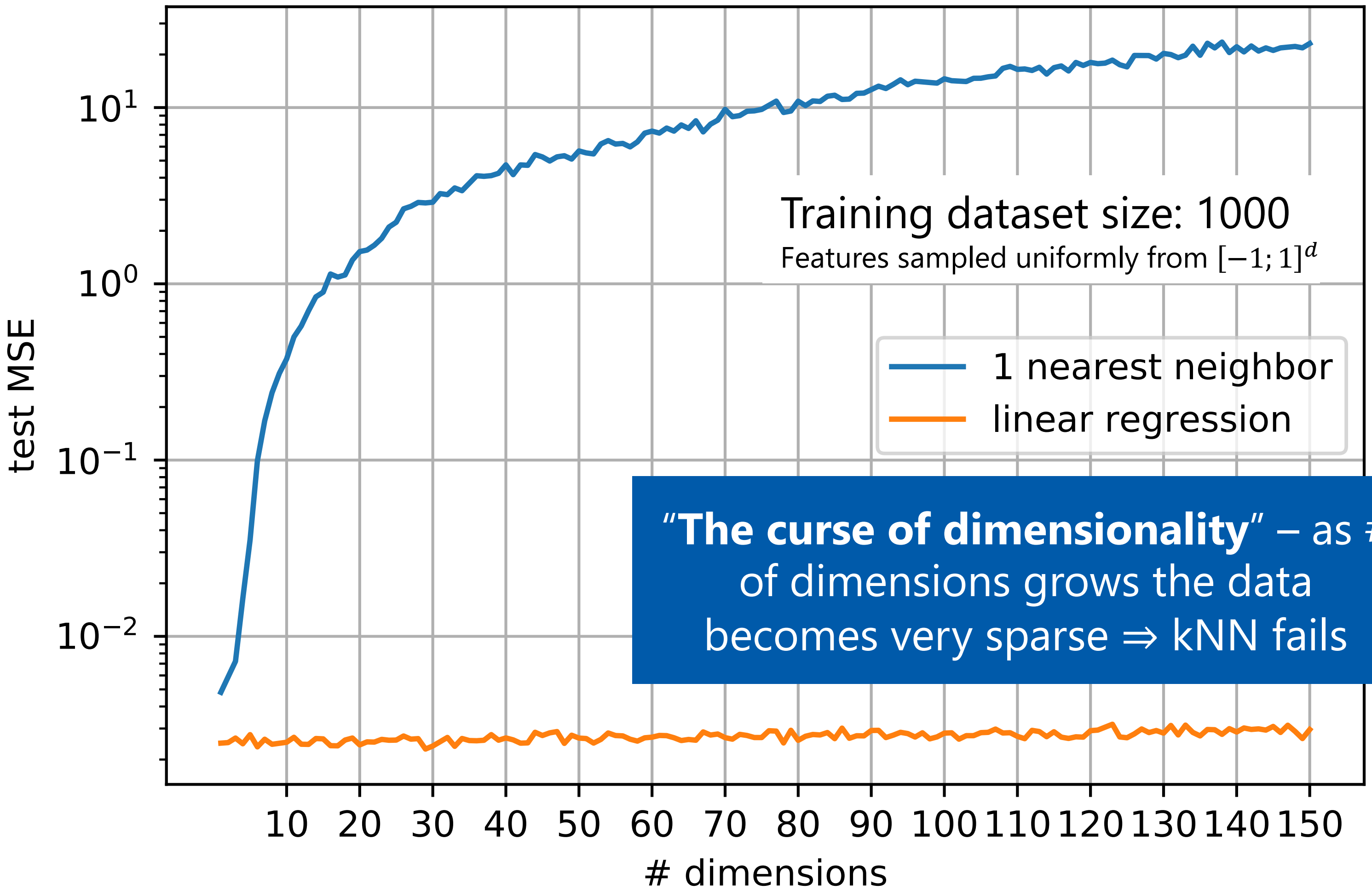
"**The curse of dimensionality**" – as # of dimensions grows the data becomes very sparse ⇒ kNN fails

Assumption for kNN:
"**similar objects have similar targets**"

Assumption for Linear Regression:
"**targets are linear in features**"

Example: kNN(k = 1) VS Linear Regression

Prediction error for $y_i = c^T x_i$, for some constant $c \in \mathbb{R}^d$



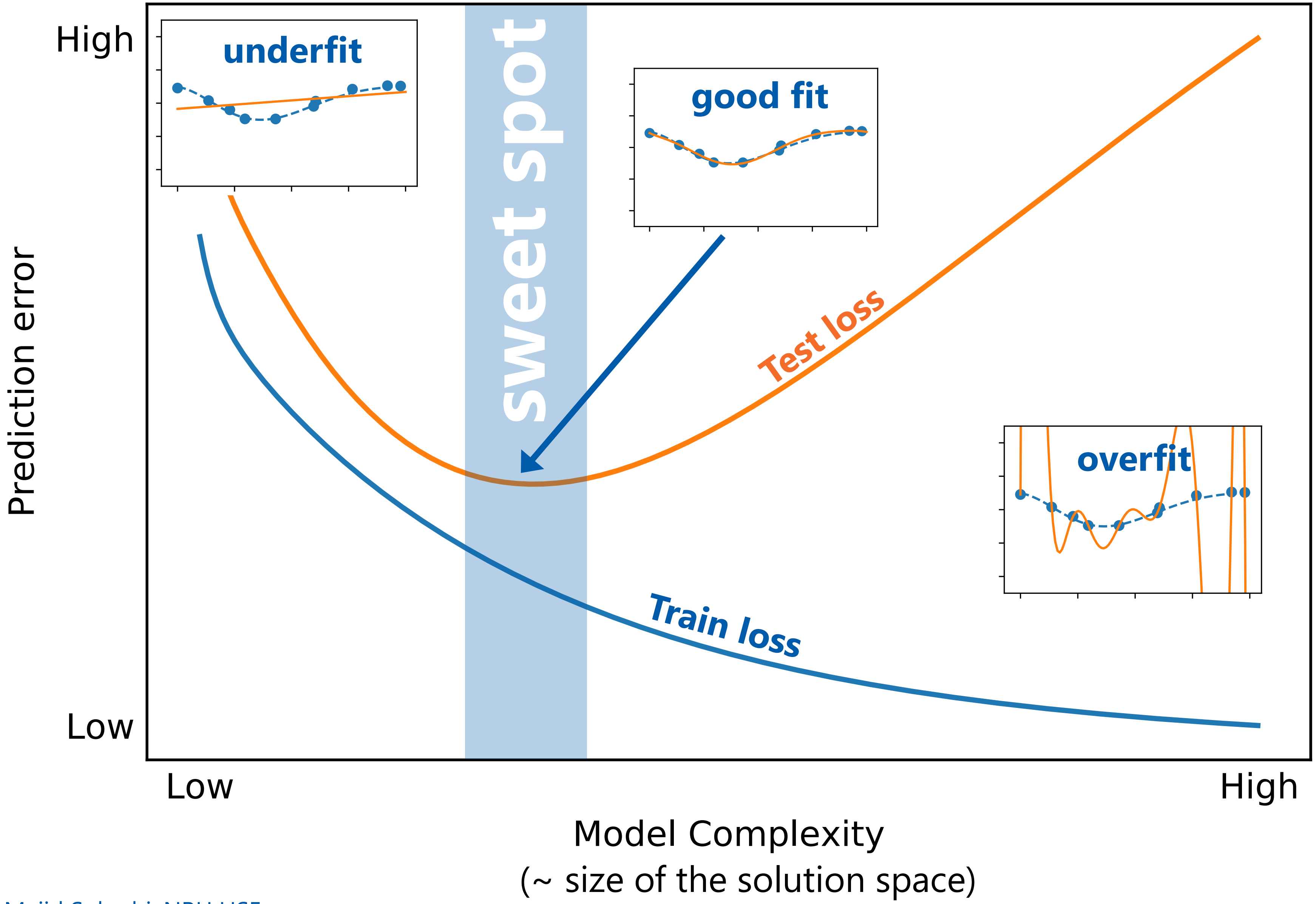
"**The curse of dimensionality**" – as # of dimensions grows the data becomes very sparse ⇒ kNN fails

Assumption for kNN:
"**similar objects have similar targets**"

Assumption for Linear Regression:
"**targets are linear in features**"

For this example, both assumptions are correct, but one is **stronger** than the other

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, one-hot encoding is useful to convert categorical features to numeric

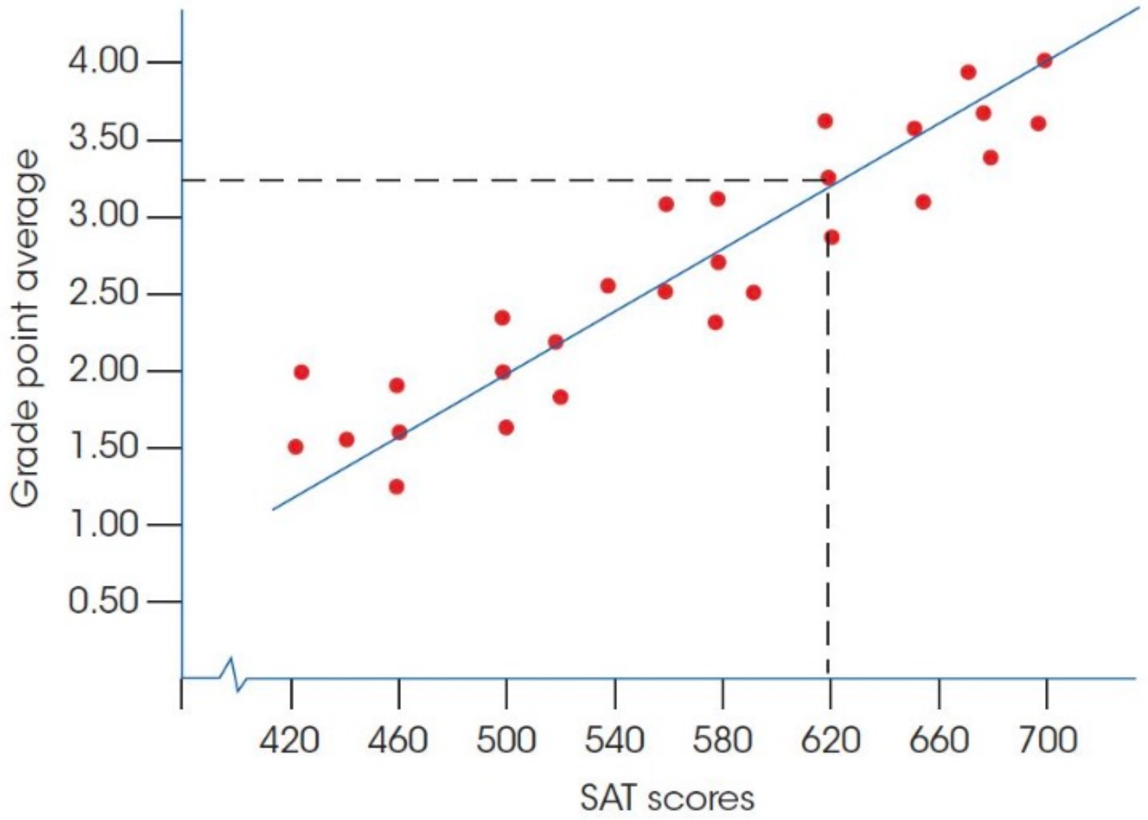
Machine Learning algorithms can be defined as expected loss minimization tasks

Choosing the right model = applying the right assumptions about the data

Use test data to detect underfitting and overfitting

Regression

Linear trend -> Regression Line



This line serves several purposes.

1. The line makes the relationship between SAT and GPA easier to see.
2. The line identifies the center, or central tendency, of the relationship, just as the mean describes central tendency for a set of scores. Thus, the line provides a simplified description of the relationship.
3. The line can be used for prediction of unknown values

Regression & Regression Line

The statistical technique for finding the best-fitting straight line for a set of data is called regression, and the resulting straight line is called the regression line.

In general, a linear relationship between two variables X and Y can be expressed by the equation

$$Y = bX + a$$

where a and b are fixed constants.

$$b = \frac{SP}{SS_X}$$

$$a = MY - bMX$$

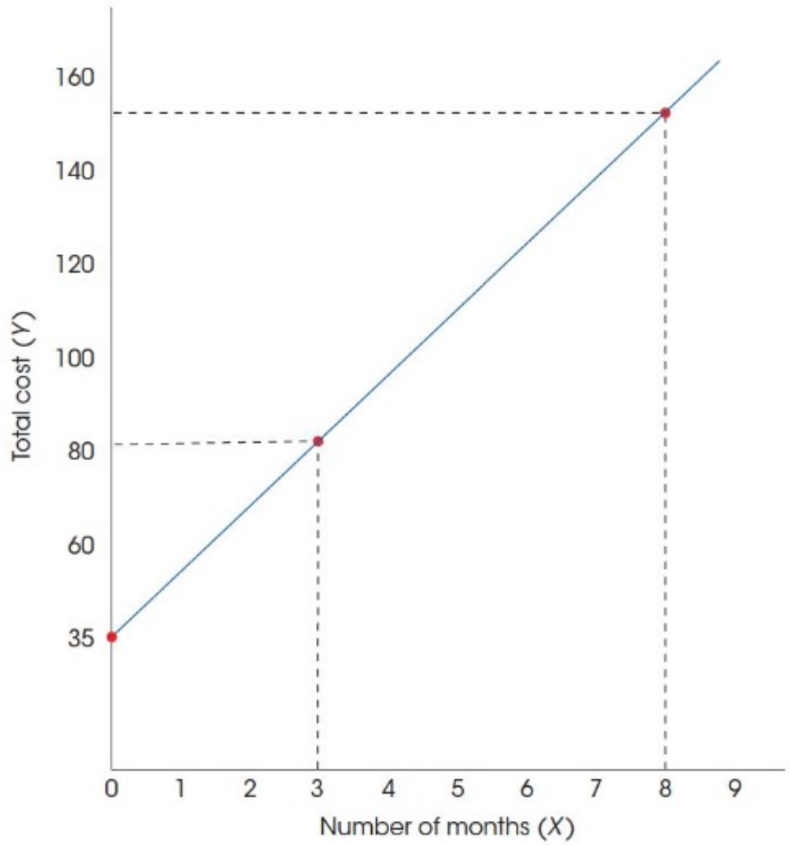
For example, a local gym charges a membership fee of $35 and a monthly fee of $15 for unlimited use of the facility. With this information, the total cost for the gym can be computed using a linear equation that describes the relationship between the total cost (Y) and the number months (X).

$$Y = 15X + 35$$

Linear Equation

In the general linear equation, the value of **b** is called the **slope**. The slope determines how much the Y variable changes when X is increased by one point. For the gym membership example, the slope is b = \$15 and indicates that your total cost increases by \$15 each month.

The value of **a** in the general equation is called the **Y-intercept** because it determines the value of Y when X = 0.



Calculating coefficients for the equation

X	Y	$X - M_X$	$Y - M_Y$	$(X - M_X)^2$	$(Y - M_Y)^2$	$(X - M_X)(Y - M_Y)$
5	10	1	3	1	9	3
1	4	−3	−3	9	9	9
4	5	0	−2	0	4	0
7	11	3	4	9	16	12
6	15	2	8	4	64	16
4	6	0	−1	0	1	0
3	5	−1	−2	1	4	2
2	0	−2	−7	4	49	14

$$b = \frac{SP}{SS_X}$$

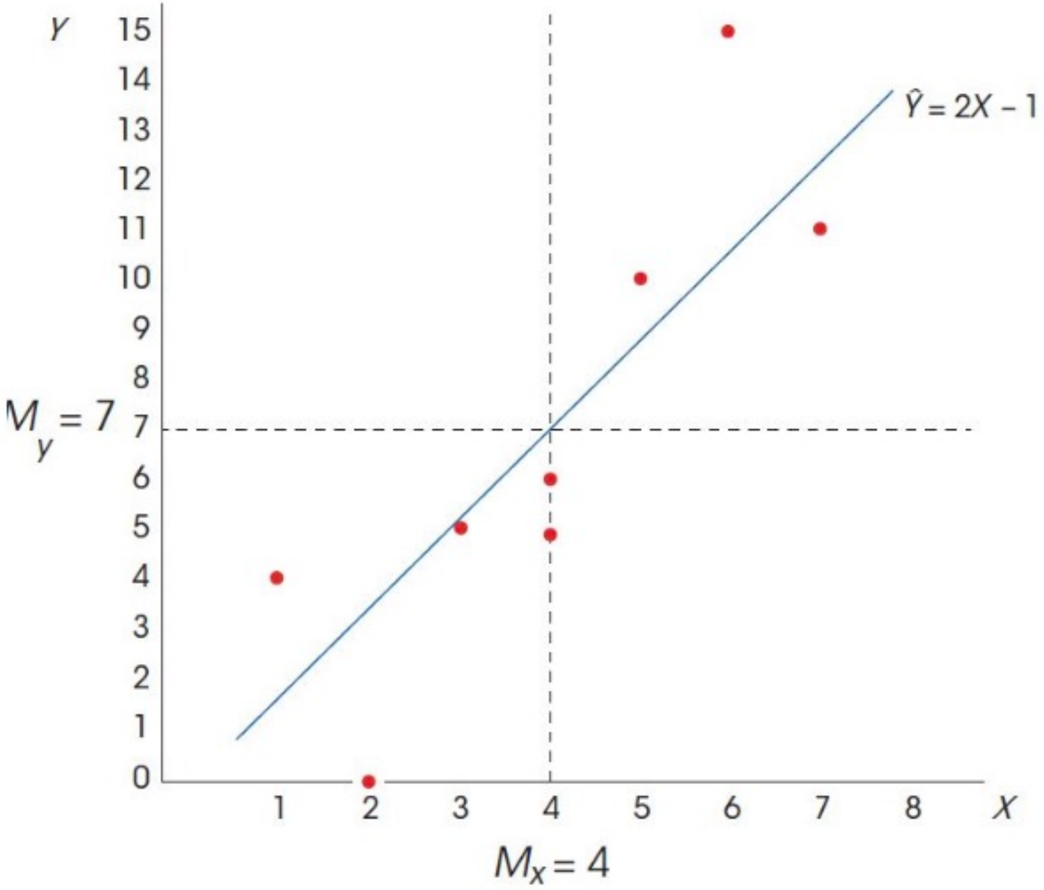
$$a = MY - bMX$$

Calculating coefficients for the equation

X	Y	$X - M_X$	$Y - M_Y$	$(X - M_X)^2$	$(Y - M_Y)^2$	$(X - M_X)(Y - M_Y)$
5	10	1	3	1	9	3
1	4	–3	–3	9	9	9
4	5	0	–2	0	4	0
7	11	3	4	9	16	12
6	15	2	8	4	64	16
4	6	0	–1	0	1	0
3	5	–1	–2	1	4	2
2	0	–2	–7	4	49	14
				$SS_X = 28$	$SS_Y = 156$	$SP = 56$

$$b = \frac{SP}{SS_X} = \frac{56}{28} = 2 \qquad a = M_Y - bM_X = 7 - 2(4) = -1 \qquad \hat{Y} = 2X - 1$$

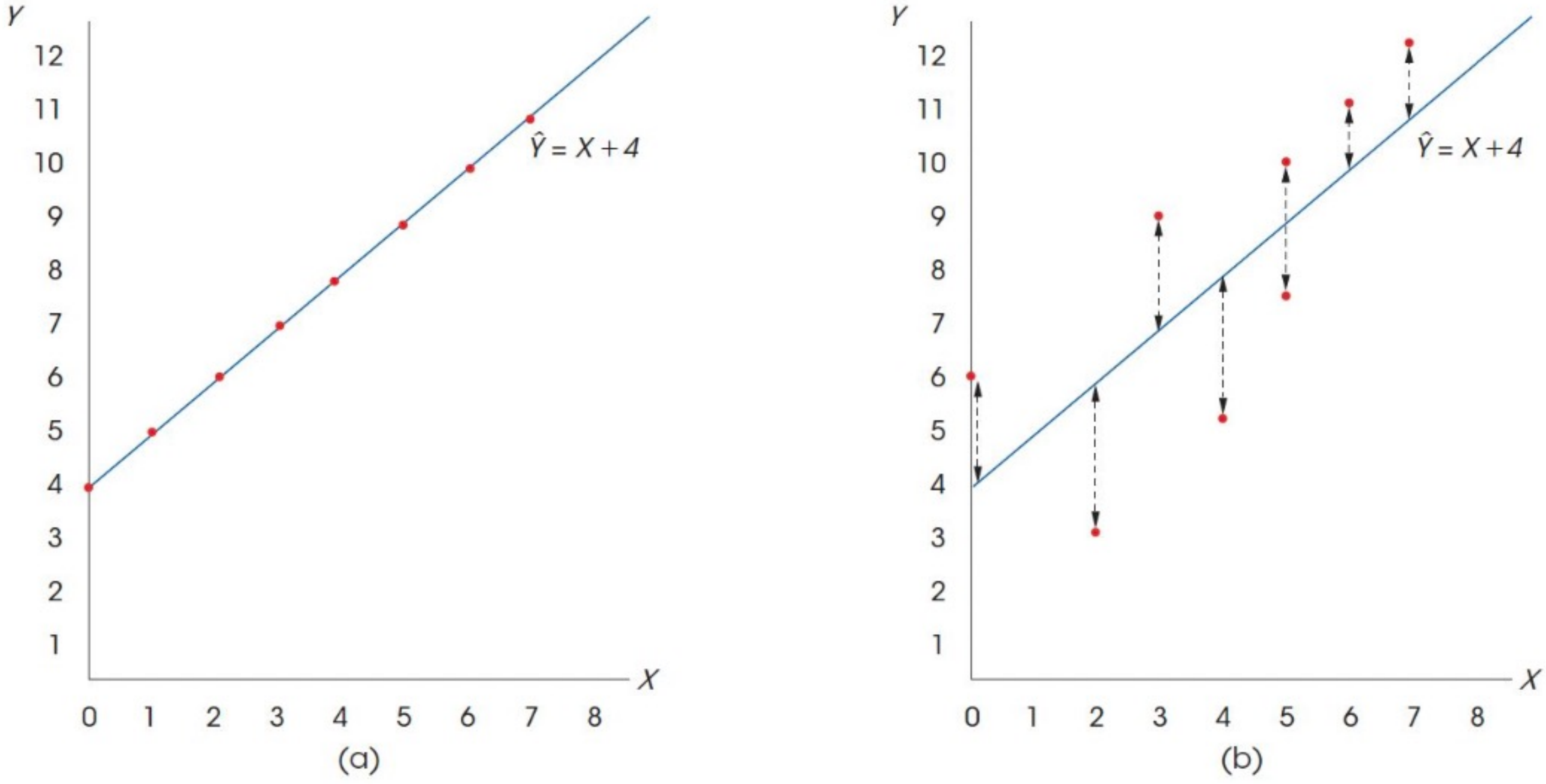
Fitting the Line to the data



First, the calculation of the Y-intercept ensures that the regression line passes through the point defined by the mean for X and the mean for Y. That is, the point identified by the coordinates M_X, M_Y will always be on the line.

Second, the sign of the correlation (+ or –) is the same as the sign of the slope of the regression line. E.g., if the correlation is positive, then the slope is also positive and the regression line slopes up to the right. A correlation of zero means that the slope is also zero and the regression equation produces a horizontal line that passes through the data at a level equal to the mean for the Y values.

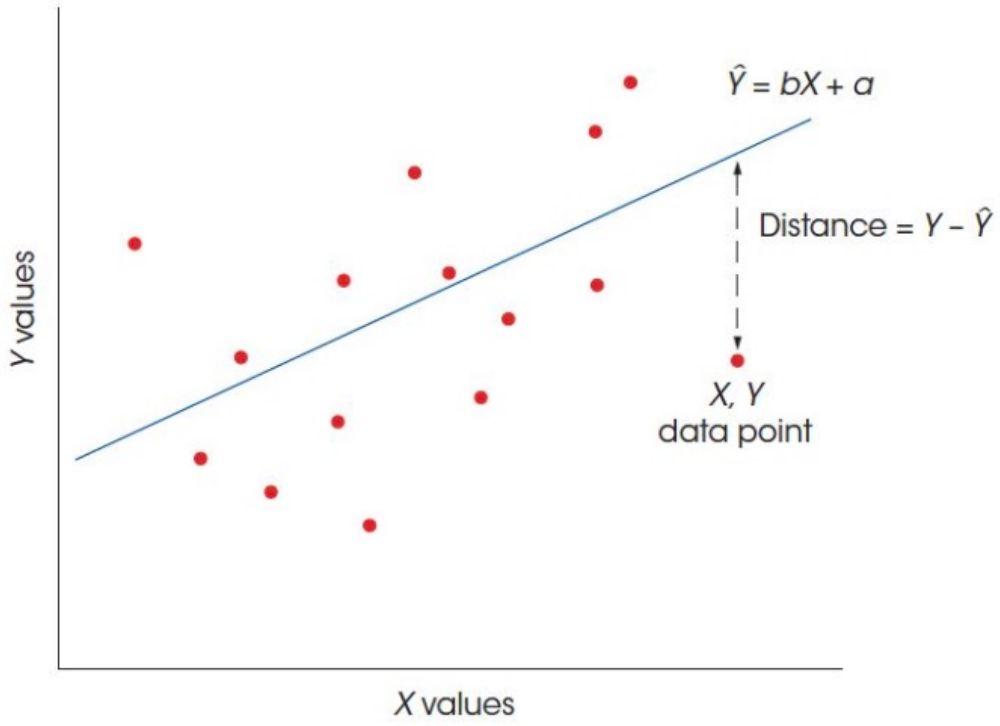
Regression line precision



The least squares regression

To determine how well a line fits the data points, the first step is to define mathematically the distance between the line and each data point. For every X value in the data, the linear equation determines a Y value on the line. This value is the predicted Y and is called ŷ ("Y hat"). The distance between this predicted value and the actual Y value in the data is determined by

$$distance = Y - \hat{Y}$$



Root Mean Square Error (RMSE)

There are different metrics to measure the distance between the predicted Y values on the regression line and the actual Y values in the data.

We will be using the RMSE — **root mean squared error**. It sums all the squared distances and divides it by the number of observation. Finally, it derives a root from the variance. It shows you, how far is on average the prediction from the actual data points.

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$$

Calculating RMSE example

Apartment size	**Distance to Subway**	**District**	**Price**
50 m^2	1 km	Cheremushki	5 000 000 rub
100 m^2	2 km	Shchukino	9 000 000 rub
50 m^2	1 km	Lubyanka	20 000 000 rub
100 m^2	0.5 km	Khamovniki	50 000 000 rub

Calculating RMSE example

Apartment size	Distance to Subway	District	Price	Price Predicted
50 m^2	1 km	Cheremushki	5 000 000 rub	5 000 000 rub
100 m^2	2 km	Shchukino	9 000 000 rub	10 000 000 rub
50 m^2	1 km	Lubyanka	20 000 000 rub	5 000 000 rub
100 m^2	0.5 km	Khamovniki	50 000 000 rub	10 000 00 rub

$$\hat{y} = 100\ 000 \times \boldsymbol{Apartment\_size}$$

Calculating MSE example

Apartment size	Distance to Subway	District	Price	Price Predicted
50 m^2	1 km	Cheremushki	5 000 000 rub	5 000 000 rub
100 m^2	2 km	Shchukino	9 000 000 rub	10 000 000 rub
50 m^2	1 km	Lubyanka	20 000 000 rub	5 000 000 rub
100 m^2	0.5 km	Khamovniki	50 000 000 rub	10 000 00 rub

$\hat{y} = 100\,000 \times \boldsymbol{Apartment\_size}$ $\quad$ $RMSE = (((1kk)^2 + (15kk)^2 + (40kk)^2)/4$ ^ 0.5 = 21.37kk

Calculating MSE example

Apartment size	Distance to Subway	District	Price	Price Predicted
50 m^2	1 km	Cheremushki	5 000 000 rub	5 000 000 rub
100 m^2	2 km	Shchukino	9 000 000 rub	9 000 000 rub
50 m^2	1 km	Lubyanka	20 000 000 rub	20 500 000 rub
100 m^2	0.5 km	Khamovniki	50 000 000 rub	40 000 000 rub

$$\hat{y} = 100\,000 \times \boldsymbol{Apartment\ size} - 1\,000\,000 \times \boldsymbol{Distance\ to\ Subway} + 300\,000 \times \boldsymbol{Central\ District} \times \mathbf{Apartment\ size} + 1\,000\,000$$

$$\boldsymbol{RMSE = 5kk}$$

Thank you!

Majid Sohrabi

msohrabi@hse.ru