

Elements of Machine Learning & Data Science

Winter semester 2023/24

Lecture 1 – Introduction

10.10.2023

Prof. Bastian Leibe

Prof. Wil van der Aalst

Prof. Holger Hoos

AI Success Stories

Speech Recognition



Siri. Beta
Your wish is
its command.



AI Success Stories

Large Language Models

LE

Write a poem on the benefits and dangers of AI research in which successive pairs of lines rhyme and in which each line starts with a successive letter of the alphabet from A to Z.



AI research has its perks,
But also causes some irks.

Benefits abound, from medicine to industry,
Computers with brains can help us with ease.

Efficiency and speed are just a few,
Gains in productivity are there for all to view.

However, dangers lurk and must not be ignored,
Intelligence gone rogue could be quite abhorred.

Job loss is a concern that must be faced,
Killer robots, a nightmare that could be traced.

Nevertheless, the potential is vast and bright,
Opening doors to a future that's full of light.

Progress must be made with caution and care,
Questioning the ethics is of utmost importance to share.

Risks and rewards go hand in hand,
So let's proceed with intelligence and a plan.

Understanding the implications is key,
Venturing forth with knowledge and care, we'll see.

Wise decisions will lead us to success,
Exploring the unknown, we'll make great progress.

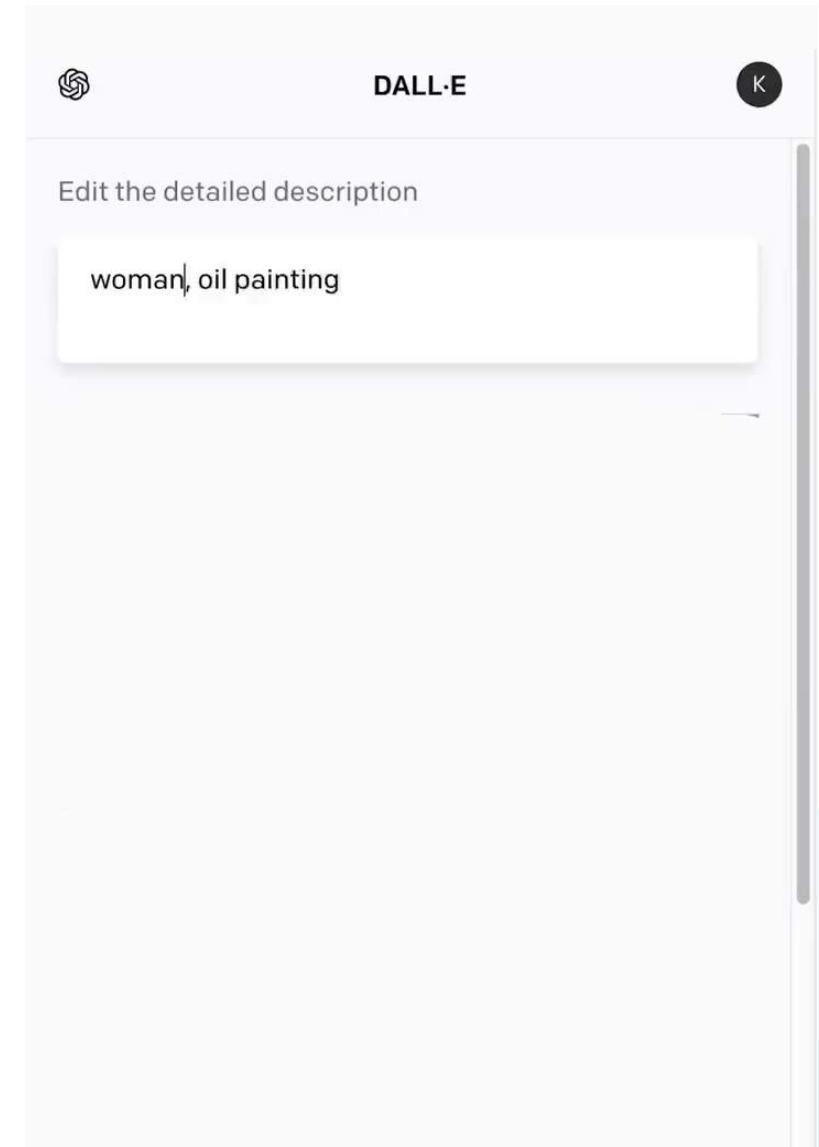
Yearning for answers, we'll continue to strive,
Zealous in our pursuit to thrive.

Text Generation Models
E.g., ChatGPT

AI Success Stories

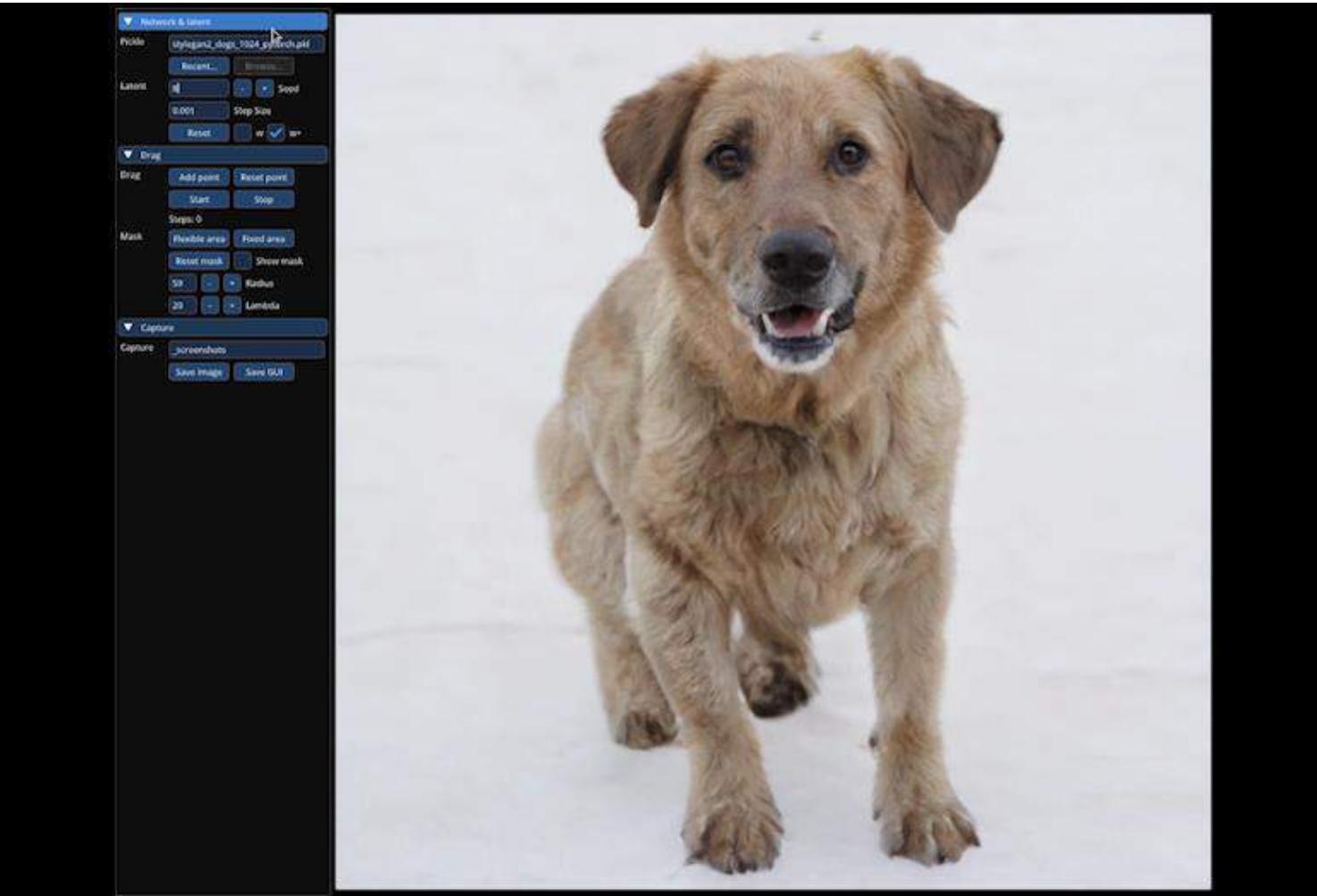
Image Generation Models

- E.g., OpenAI's DALL-e



AI Successes

Content Sensitive Image Manipulation



Video source: <https://vcai.mpi-inf.mpg.de/projects/DragGAN/>

AI Success Stories

AlphaGo



AI Success Stories

Protein Structure Folding Prediction

- E.g., AlphaFold 2 by Google DeepMind

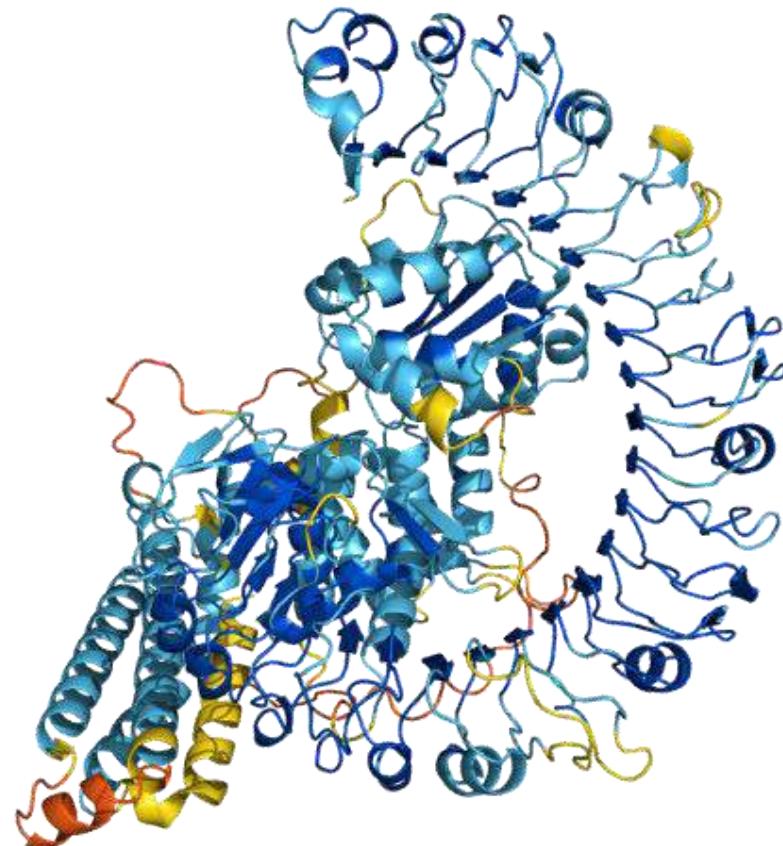
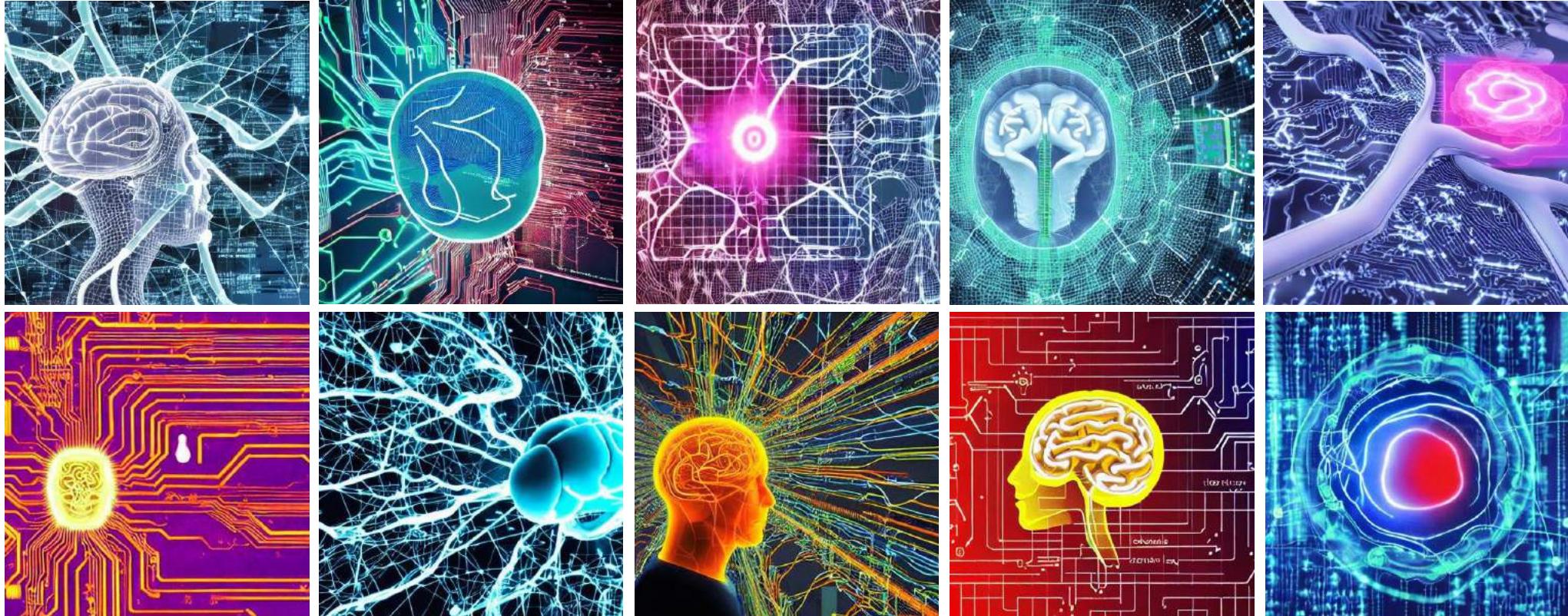


Image from <https://alphafold.ebi.ac.uk/>



Very Exciting Times Are Ahead!

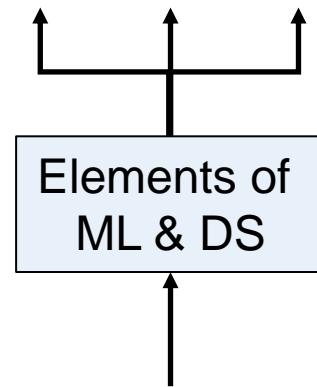


Images created with StableDiffusion by Stability.AI

- We are witness to the first strong AI systems being created in front of our eyes...

In This Lecture...

- ...we will NOT tell you how all of this works.
- Rather, we will lay the foundation, so that you can
 - Use AI methods in your Bachelor thesis
 - Take in-depth classes on a large range of topics during your Master studies



Elements of Machine Learning & Data Science

Winter semester 2023/24

Introduction to Machine Learning

10.10.2023

Prof. Bastian Leibe
Chair for Computer Vision



The Chair for Computer Vision (CVG)

Computer Vision
Prof. Dr. Bastian Leibe



RWTHAACHEN
UNIVERSITY

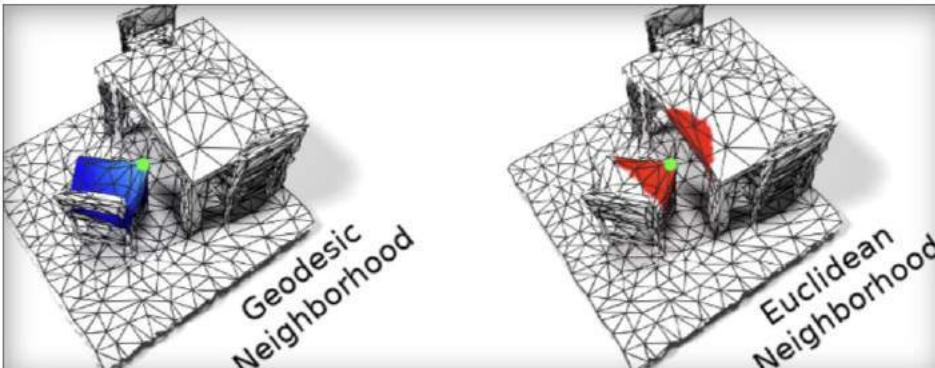
Welcome

Home
Contact
Staff

Research
Publications
Software

Teaching
Theses
Jobs

Projects
Datasets



Welcome to the Computer Vision Group at RWTH Aachen University!

The Computer Vision group has been established at RWTH Aachen University in context with the Cluster of Excellence "UMIC - Ultra High-Speed Mobile Information and Communication" and is associated with the Chair Computer Sciences 8 - Computer Graphics, Computer Vision, and Multimedia. The group focuses on computer vision applications for mobile devices and robotic or automotive platforms. Our main research areas are visual object recognition, tracking, self-localization, 3D reconstruction, and in particular combinations between those topics.

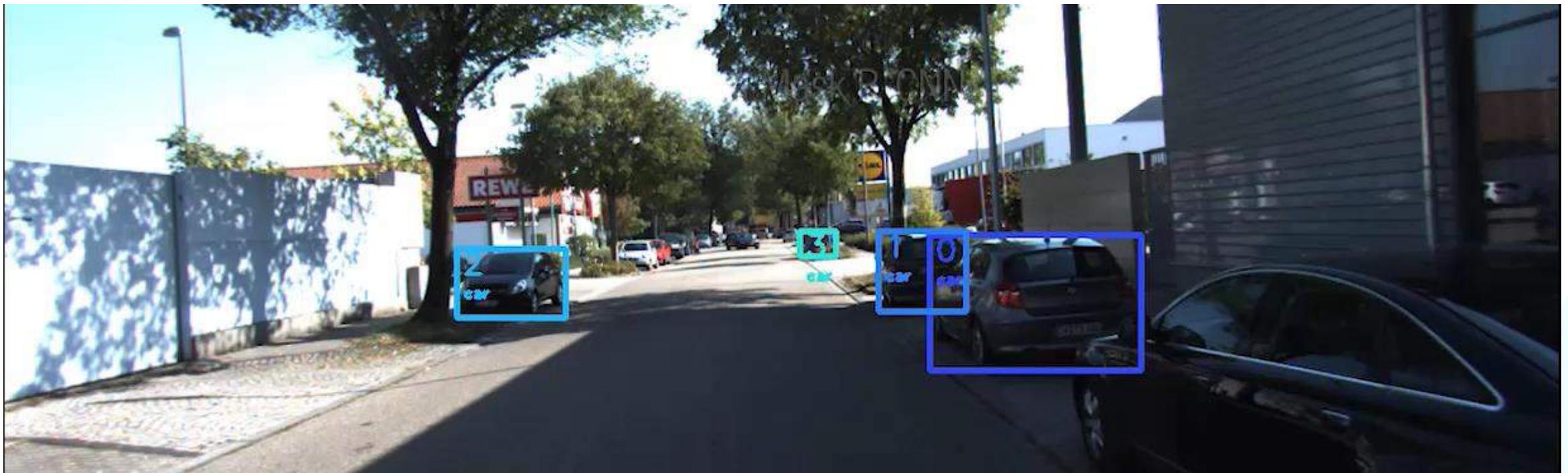
We offer lectures and seminars about computer vision and machine learning.

You can browse through all our publications and the projects we are working on.

<http://vision.rwth-aachen.de>

CVG Research Topics

Object Detection and Tracking



CVG Research Topics

Interactive Segmentation



CVG Research Topics

Human Body Pose Estimation

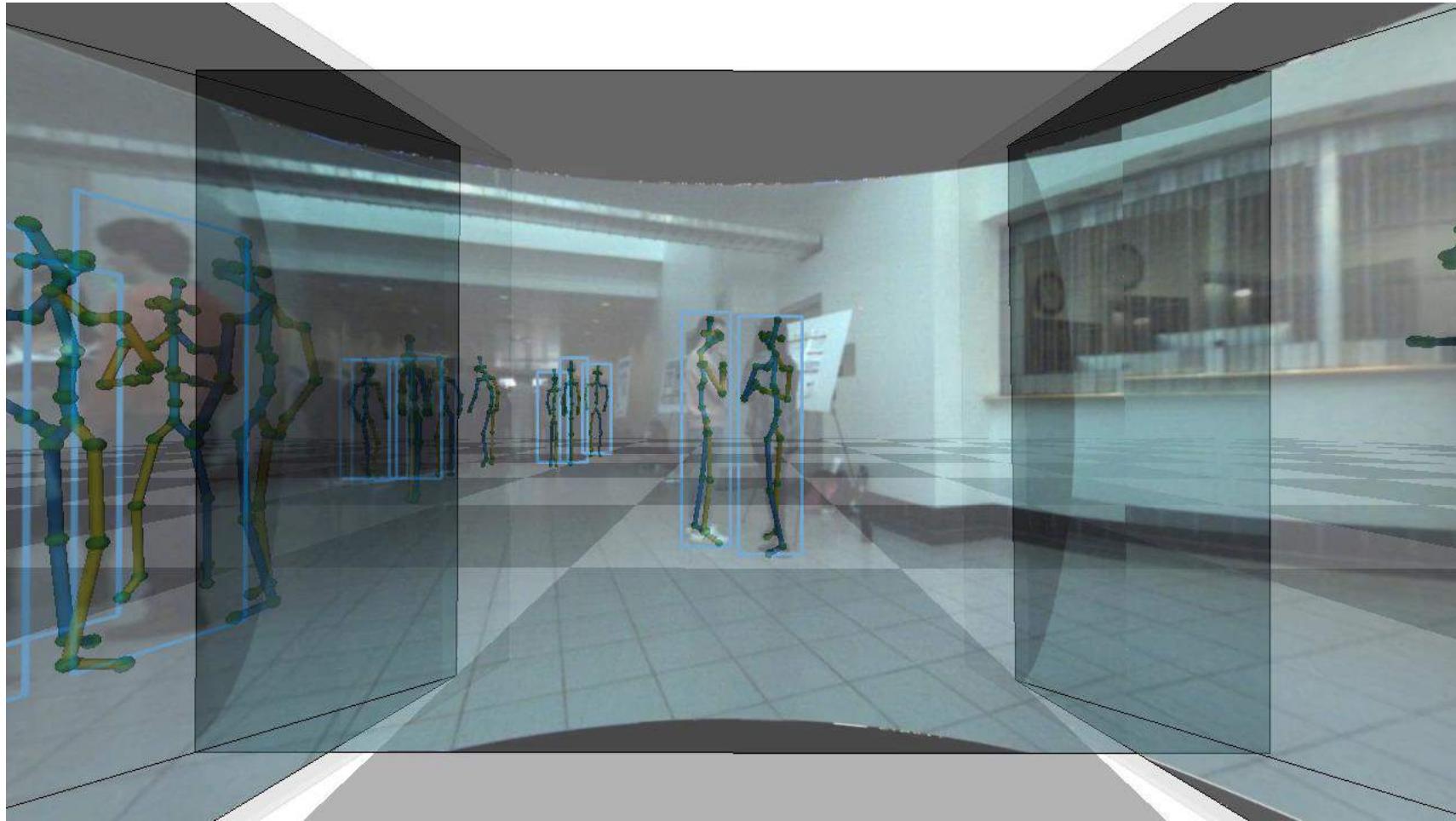


MeTRAbs



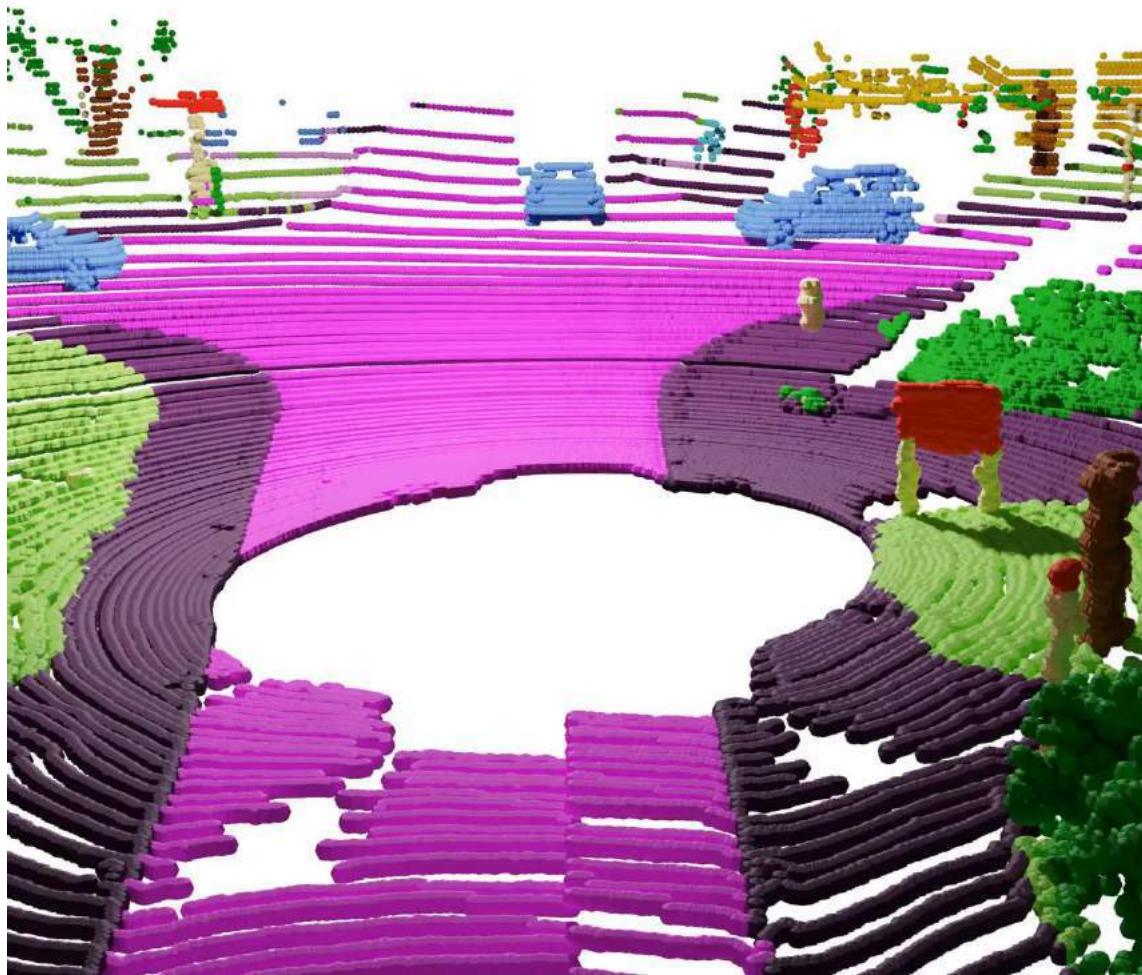
CVG Research Topics

Applications for Mobile Robotics



CVG Research Topics

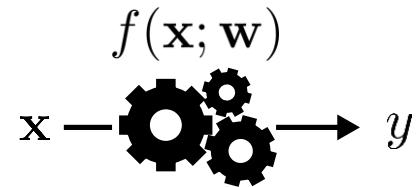
3D Scene Understanding



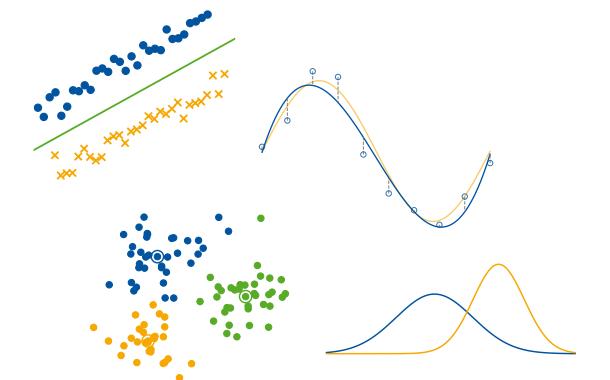
E.g., 4D LiDAR Segmentation

Machine Learning Topics

1. **Introduction to ML**
2. Probability Density Estimation
3. Linear Discriminants
4. Linear Regression
5. Logistic Regression
6. Support Vector Machines
7. AdaBoost
8. Neural Network Basics



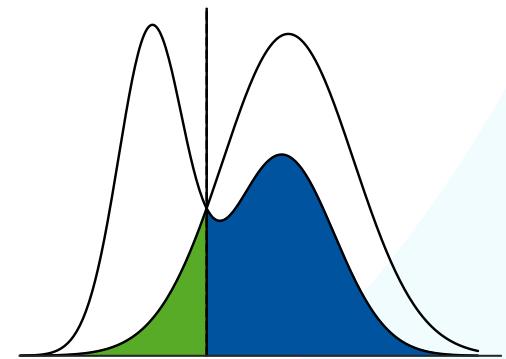
Machine Learning
Concepts



Forms of Machine Learning

$$p(\mathcal{C}|\mathbf{x}) = \frac{p(\mathbf{x}|\mathcal{C})p(\mathcal{C})}{p(\mathbf{x})}$$

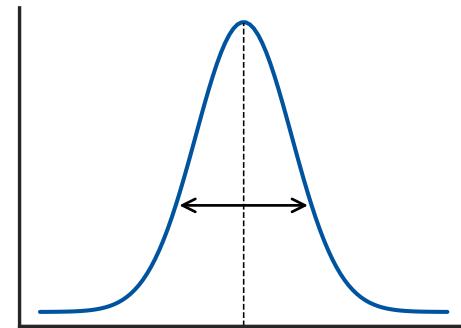
Bayes Decision Theory



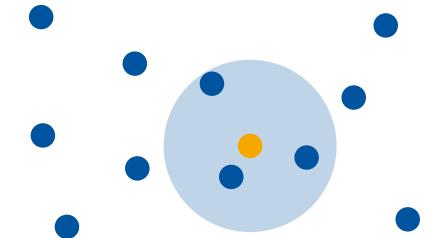
Bayes Optimal
Classification

Machine Learning Topics

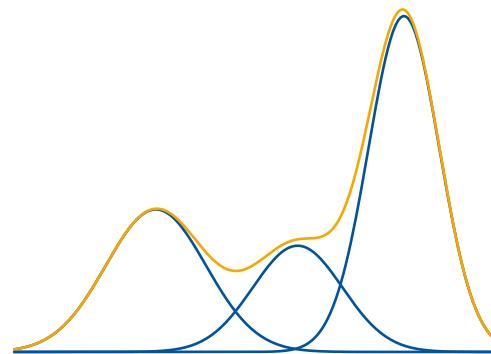
1. Introduction to ML
2. **Probability Density Estimation**
3. Linear Discriminants
4. Linear Regression
5. Logistic Regression
6. Support Vector Machines
7. AdaBoost
8. Neural Network Basics



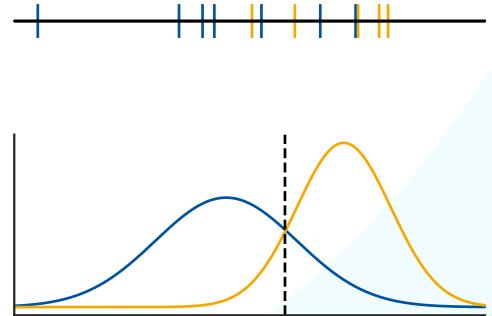
Parametric Methods
& ML-Algorithm



Nonparametric Methods



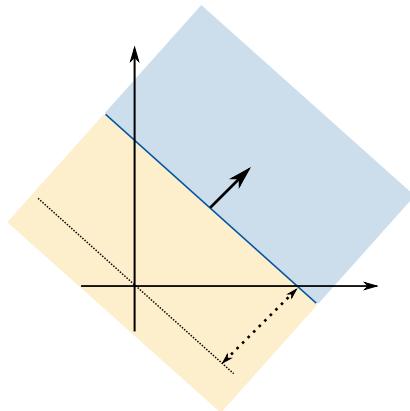
Mixtures of Gaussians
& EM-Algorithm



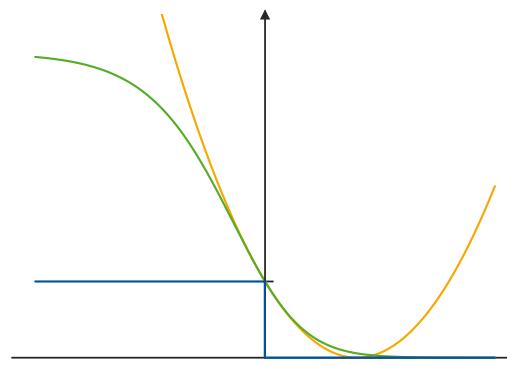
Bayes Classifiers

Machine Learning Topics

1. Introduction to ML
2. Probability Density Estimation
- 3. Linear Discriminants**
4. Linear Regression
5. Logistic Regression
6. Support Vector Machines
7. AdaBoost
8. Neural Network Basics



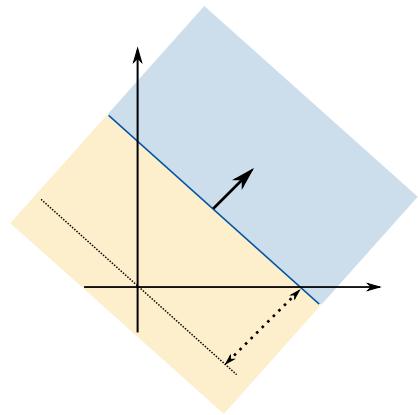
Linear Discriminants



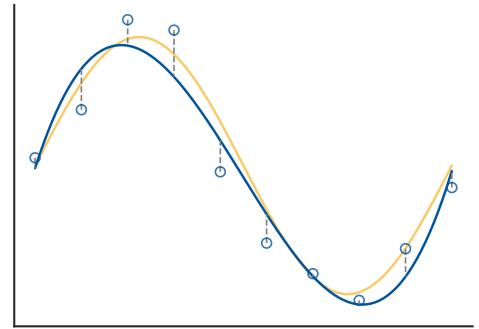
Error Functions
for Classification

Machine Learning Topics

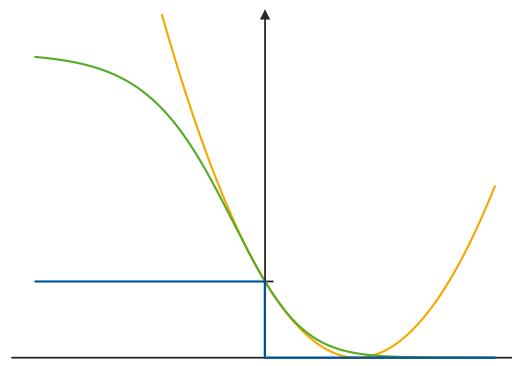
1. Introduction to ML
2. Probability Density Estimation
3. Linear Discriminants
4. **Linear Regression**
5. Logistic Regression
6. Support Vector Machines
7. AdaBoost
8. Neural Network Basics



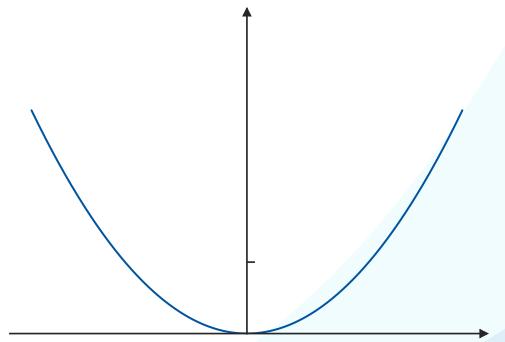
Linear Discriminants



Linear Regression



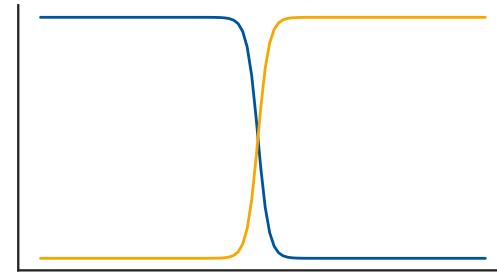
Error Functions
for Classification



Error Functions
for Regression

Machine Learning Topics

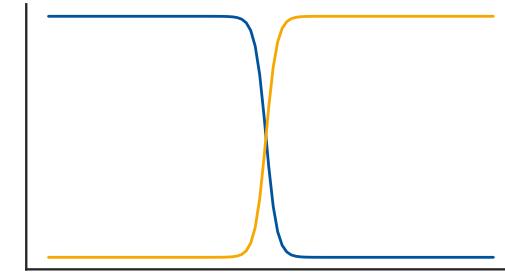
1. Introduction to ML
2. Probability Density Estimation
3. Linear Discriminants
4. Linear Regression
5. **Logistic Regression**
6. Support Vector Machines
7. AdaBoost
8. Neural Network Basics



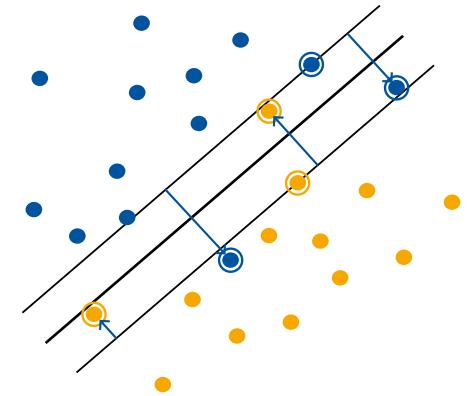
Logistic Regression

Machine Learning Topics

1. Introduction to ML
2. Probability Density Estimation
3. Linear Discriminants
4. Linear Regression
5. Logistic Regression
6. **Support Vector Machines**
7. AdaBoost
8. Neural Network Basics



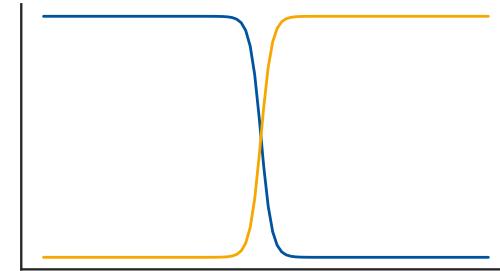
Logistic Regression



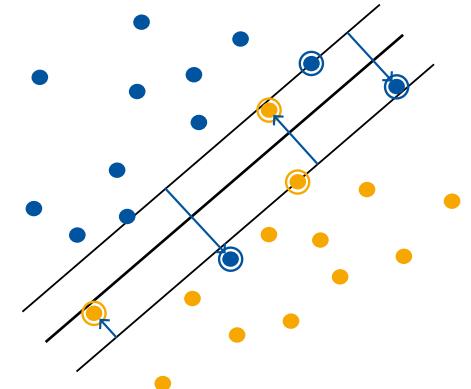
Support Vector
Machines

Machine Learning Topics

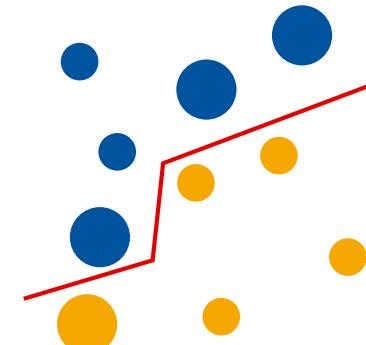
1. Introduction to ML
2. Probability Density Estimation
3. Linear Discriminants
4. Linear Regression
5. Logistic Regression
6. Support Vector Machines
7. **AdaBoost**
8. Neural Network Basics



Logistic Regression



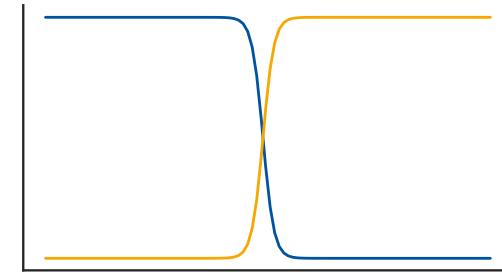
Support Vector
Machines



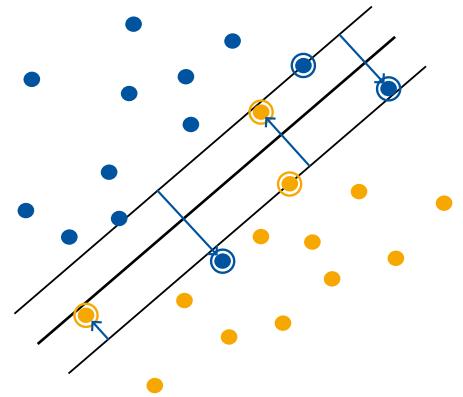
AdaBoost

Machine Learning Topics

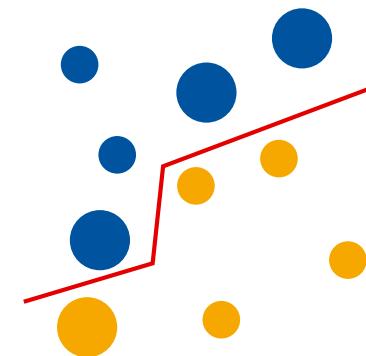
1. Introduction to ML
2. Probability Density Estimation
3. Linear Discriminants
4. Linear Regression
5. Logistic Regression
6. Support Vector Machines
7. AdaBoost
8. **Neural Network Basics**



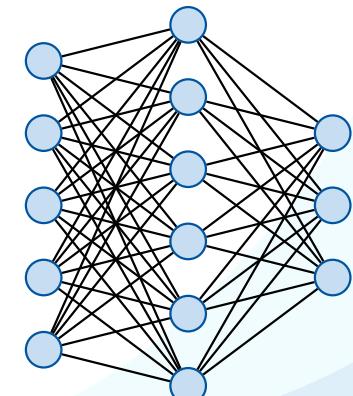
Logistic Regression



Support Vector
Machines



AdaBoost



Multi-Layer Perceptrons

Elements of Machine Learning and Data Science

Introduction to Data Science by PADS

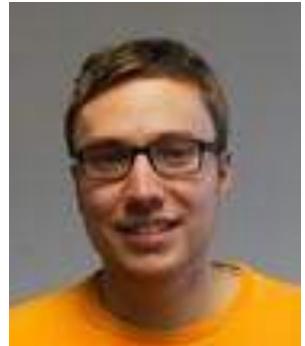
Prof. Wil van der Aalst



Wil van der Aalst



Marco Pegoraro



Harry Beyel



Nina Graves



Benedikt Knopp



Christian Rennert



Christopher Schwanen

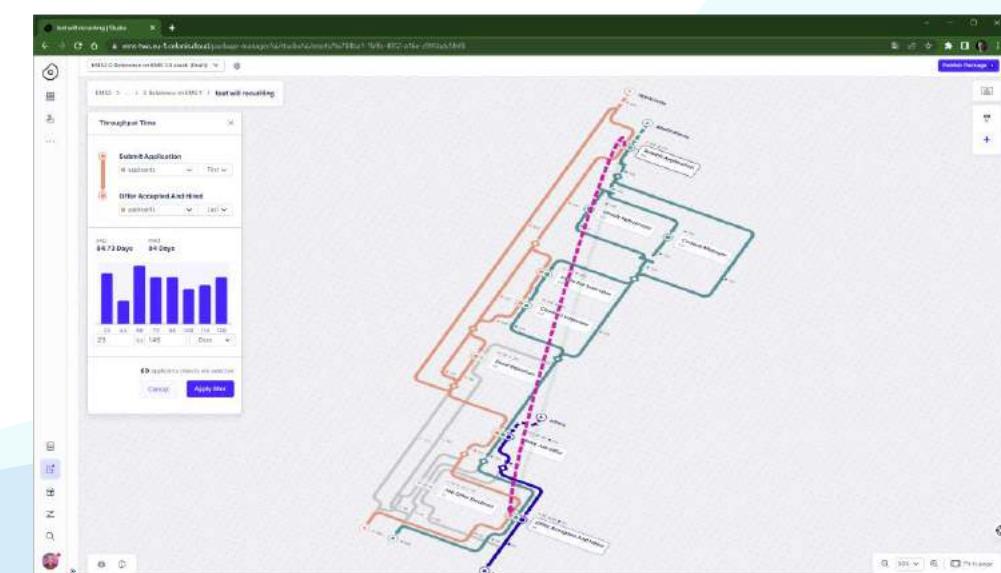
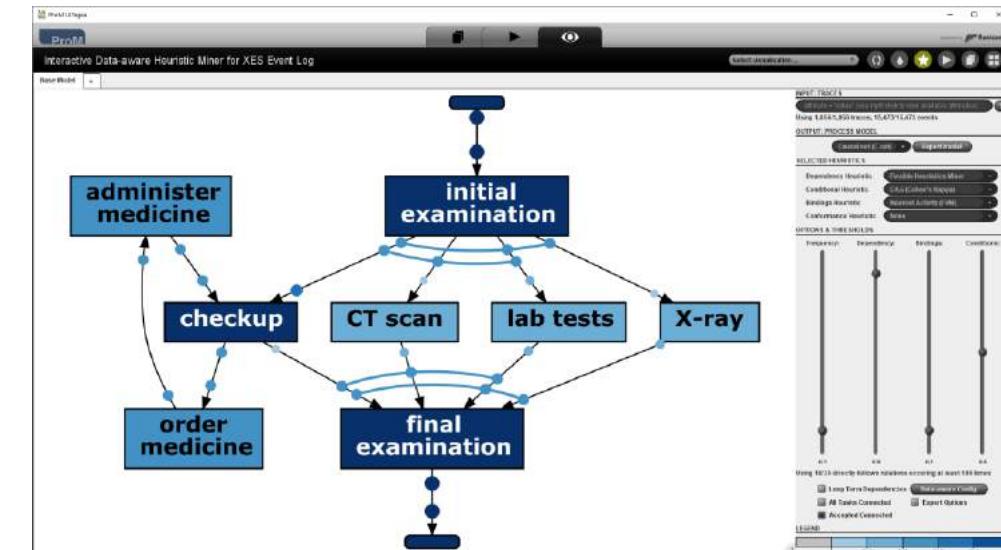
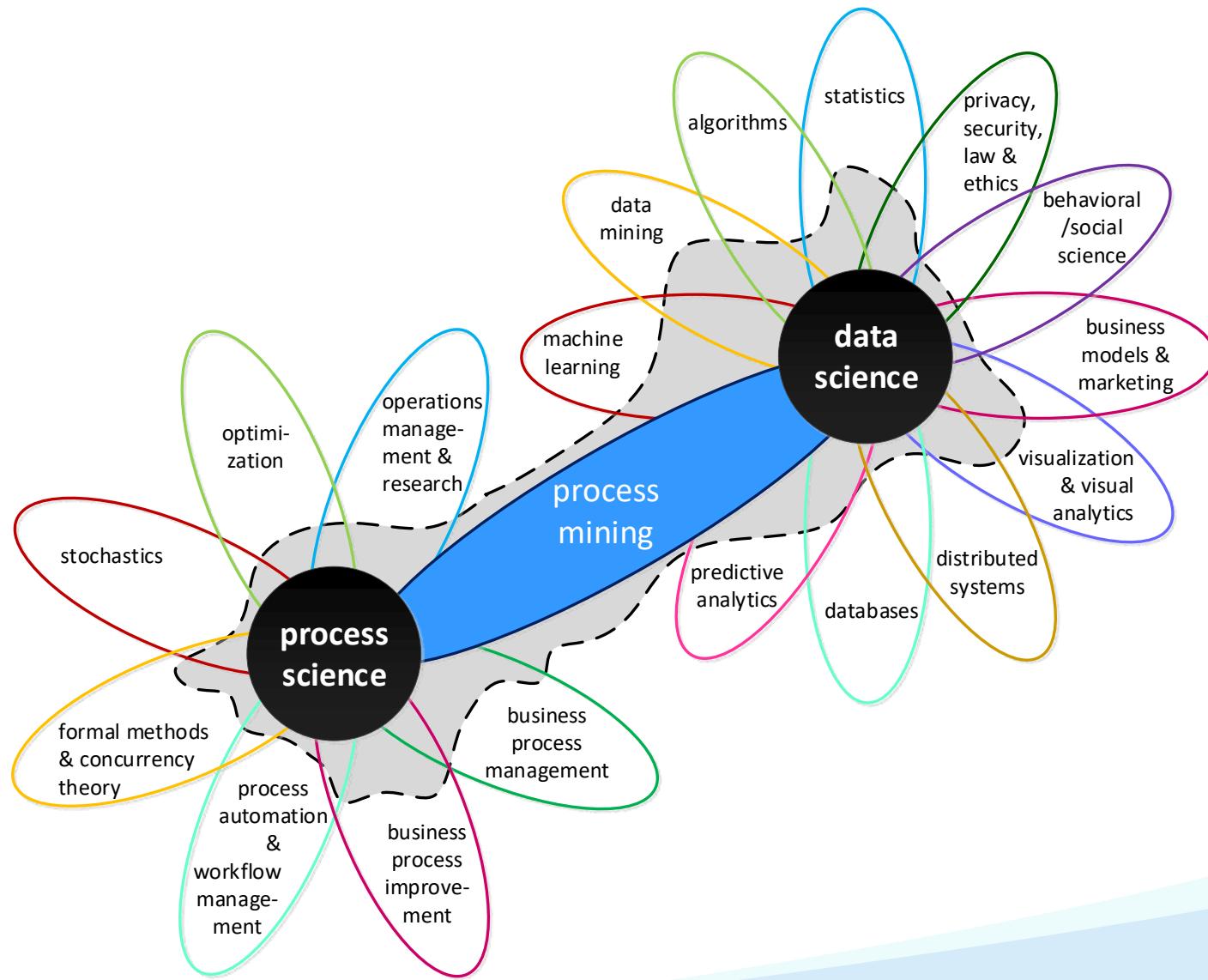


Leah Tacke genannt Unterberg

lectures

exercises

More about PADS



Data Science: A Definition

“Data science is an interdisciplinary field aiming to turn data into real value. Data may be structured or unstructured, big or small, static or streaming. Value may be provided in the form of predictions, automated decisions, models learned from data, or any type of data visualization delivering insights. Data science includes data extraction, data preparation, data exploration, data transformation, storage and retrieval, computing infrastructures, various types of mining and learning, presentation of explanations and predictions, and the exploitation of results taking into account ethical, social, legal, and business aspects.”

Data Science: A Definition

“Data science is an **interdisciplinary** field aiming to **turn data into real value**. Data may be structured or unstructured, big or small, static or streaming. Value may be provided in the form of predictions, automated decisions, models learned from data, or any type of data visualization delivering insights. Data science includes data extraction, data preparation, data exploration, data transformation, storage and retrieval, computing infrastructures, various types of mining and learning, presentation of explanations and predictions, and the exploitation of results taking into account ethical, social, legal, and business aspects.”

Data Science: A Definition

“Data science is an interdisciplinary field aiming to turn data into real value. Data may be structured or unstructured, big or small, static or streaming. Value may be provided in the form of predictions, automated decisions, models learned from data, or any type of data visualization delivering insights. Data science includes data extraction, data preparation, data exploration, data transformation, storage and retrieval, computing infrastructures, various types of mining and learning, presentation of explanations and predictions, and the exploitation of results taking into account ethical, social, legal, and business aspects.”

Data Science: A Definition

“Data science is an interdisciplinary field aiming to turn data into real value. Data may be structured or unstructured, big or small, static or streaming. Value may be provided in the form of predictions, automated decisions, models learned from data, or any type of data visualization delivering insights. Data science includes data extraction, data preparation, data exploration, data transformation, storage and retrieval, computing infrastructures, various types of mining and learning, presentation of explanations and predictions, and the exploitation of results taking into account ethical, social, legal, and business aspects.”

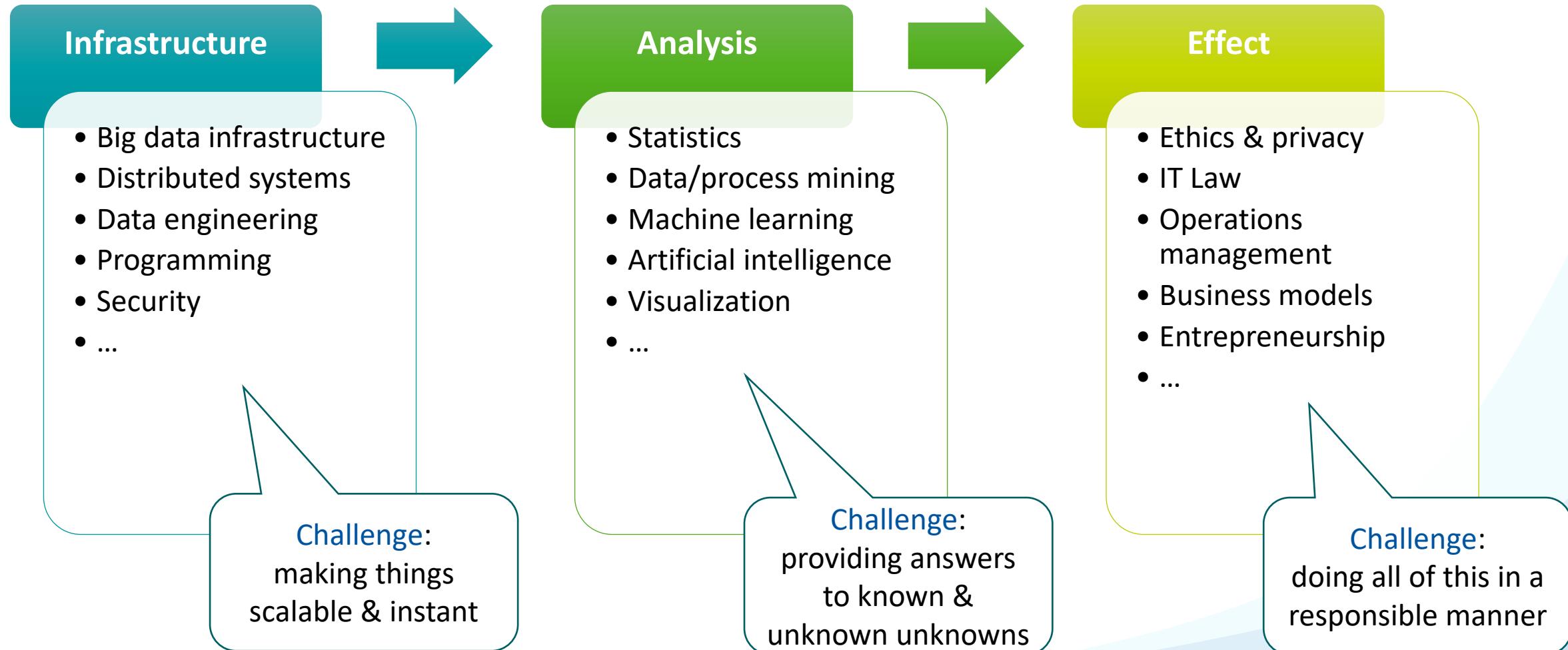
Data Science: A Definition

“Data science is an interdisciplinary field aiming to turn data into real value. Data may be structured or unstructured, big or small, static or streaming. Value may be provided in the form of predictions, automated decisions, models learned from data, or any type of data visualization delivering insights. Data science includes data extraction, data preparation, data exploration, data transformation, storage and retrieval, computing infrastructures, various types of mining and learning, presentation of explanations and predictions, and the exploitation of results taking into account ethical, social, legal, and business aspects.”

Data Science: A Definition

“Data science is an interdisciplinary field aiming to turn data into real value. Data may be structured or unstructured, big or small, static or streaming. Value may be provided in the form of predictions, automated decisions, models learned from data, or any type of data visualization delivering insights. Data science includes data extraction, data preparation, data exploration, data transformation, storage and retrieval, computing infrastructures, various types of mining and learning, presentation of explanations and predictions, and the exploitation of results taking into account ethical, social, legal, and business aspects.”

The Data Science Pipeline

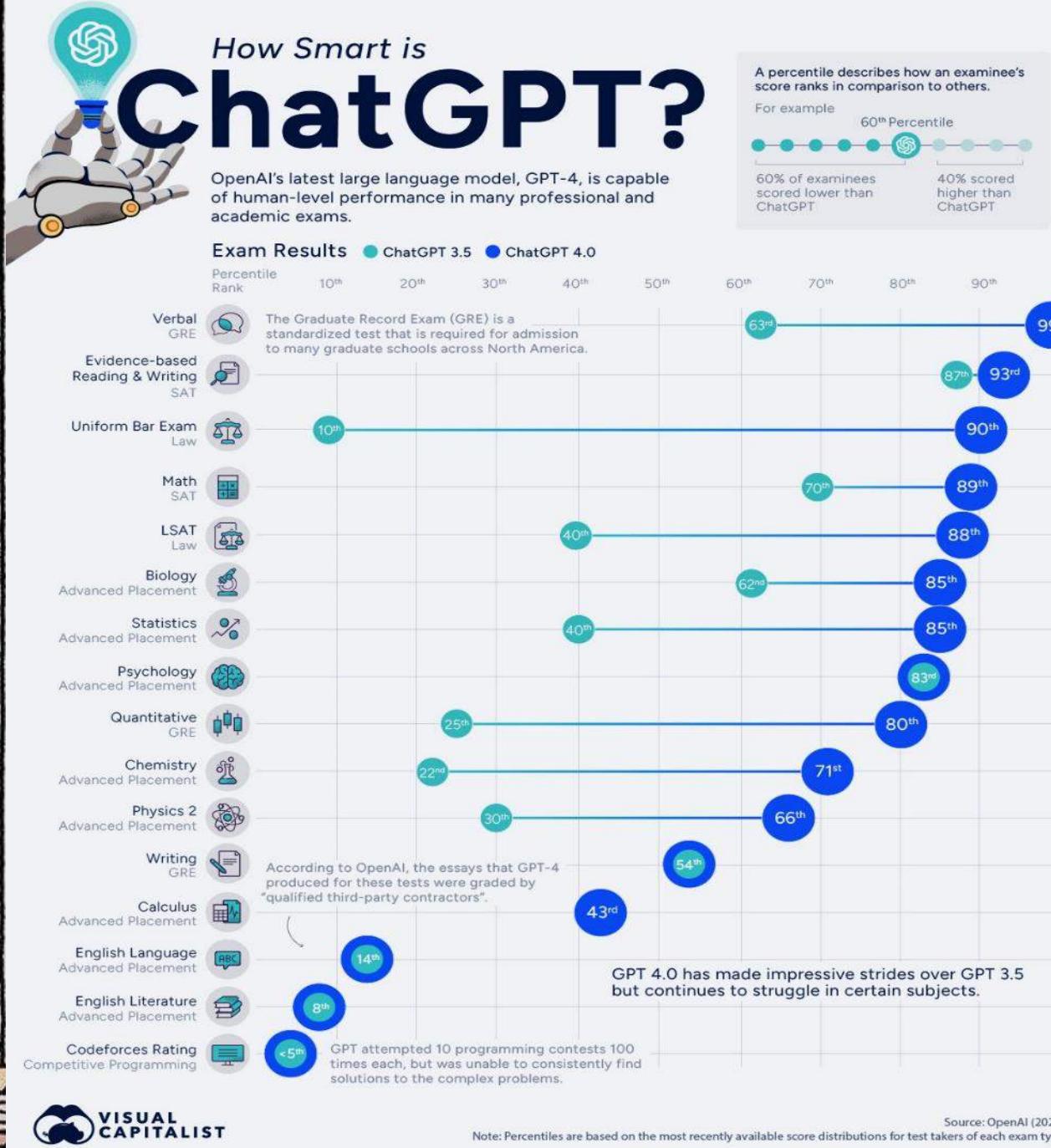
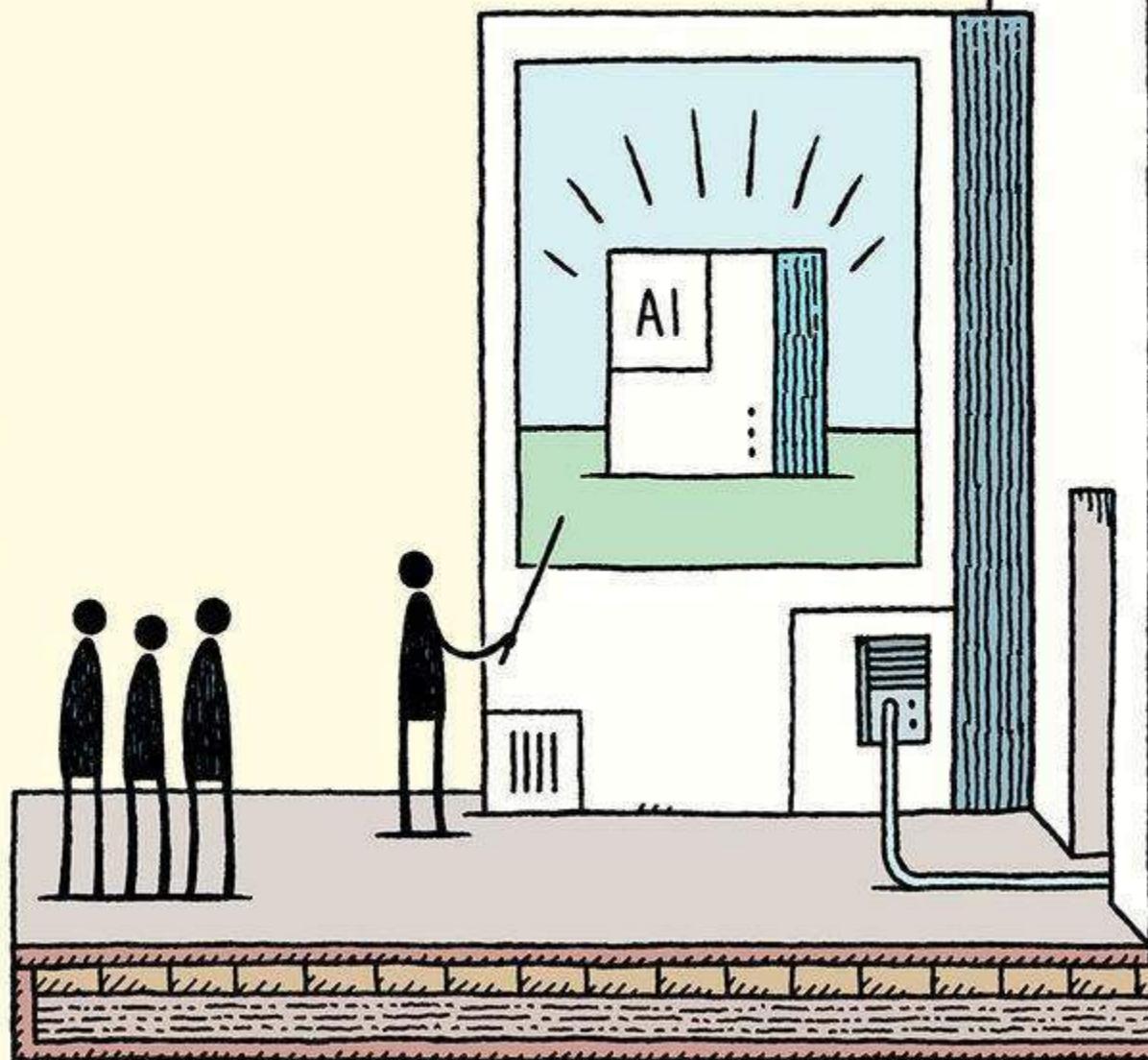


Main Topics Covered in the PADS / Data Science part

1. **Introduction to Data Science**
2. **Decision Trees**
3. **Clustering**
4. **Frequent Itemsets**
5. **Association Rules and Sequence Mining**
6. **Data Modeling, Quality, and the Data Science Process**
7. **Process Mining**
8. **Text Mining**
9. **Responsible Data Science**

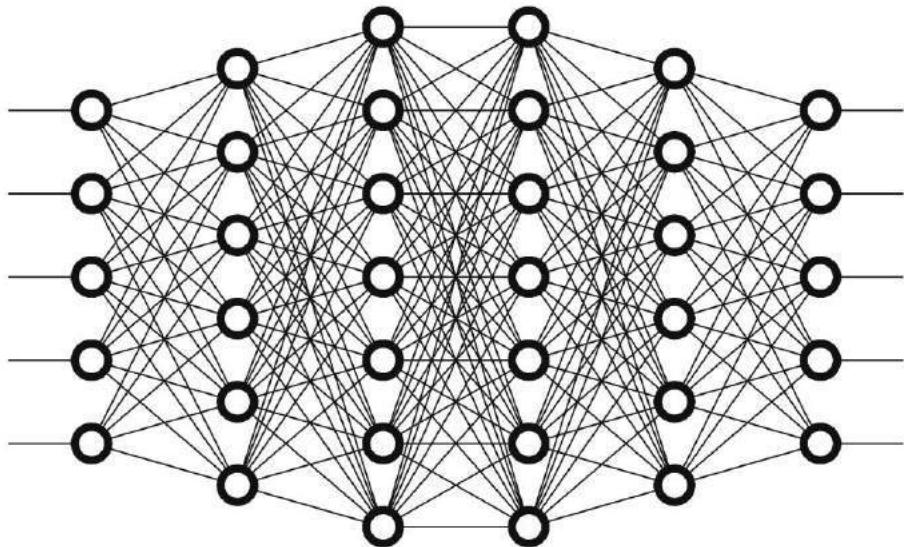


Let's first take a step back

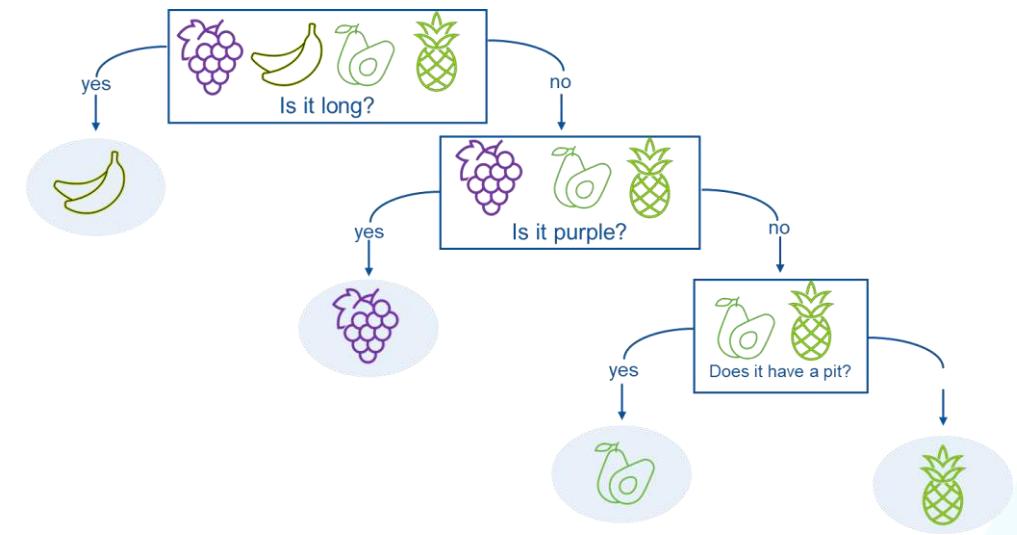


Black Box versus White Box

Black Box



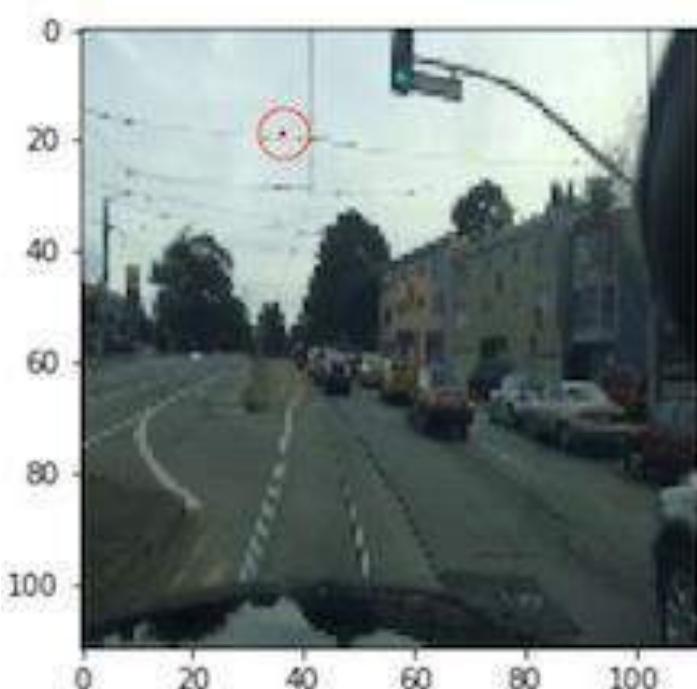
White Box



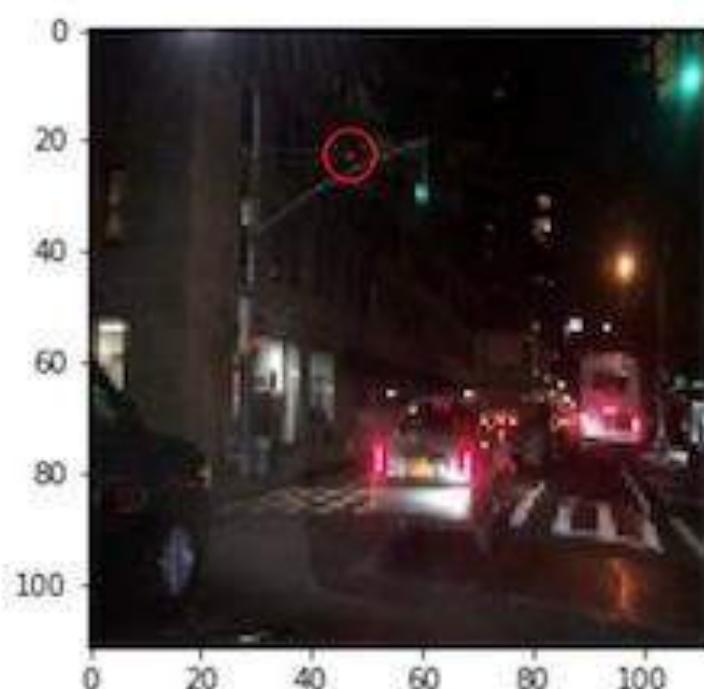
- complex, brute force
- needs more data
- better performance
- no human interpretation/adaptation

- simpler, less comp. overhead
- needs less data
- lower performance
- human interpretation/adaptation

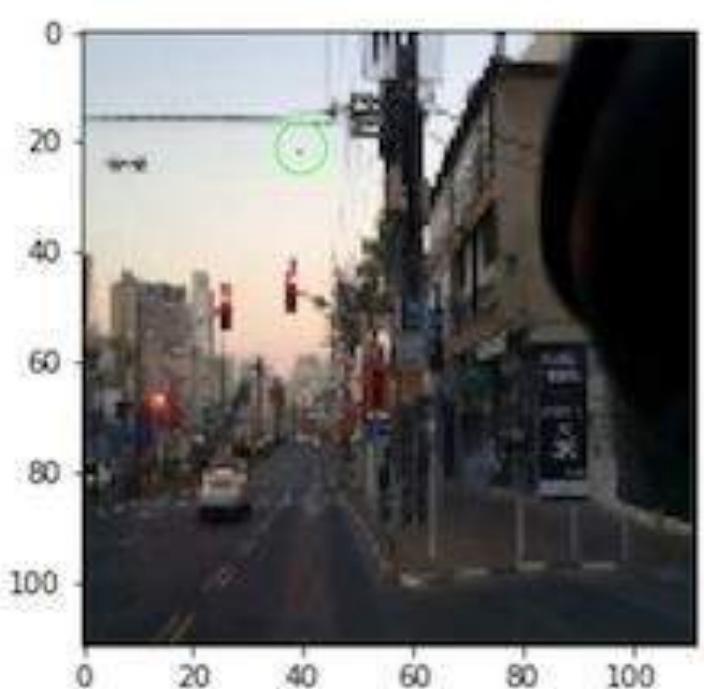
Unsurprising: There are fundamental limitations



Green light classified as red
after one pixel change



Green light classified as red
after one pixel change



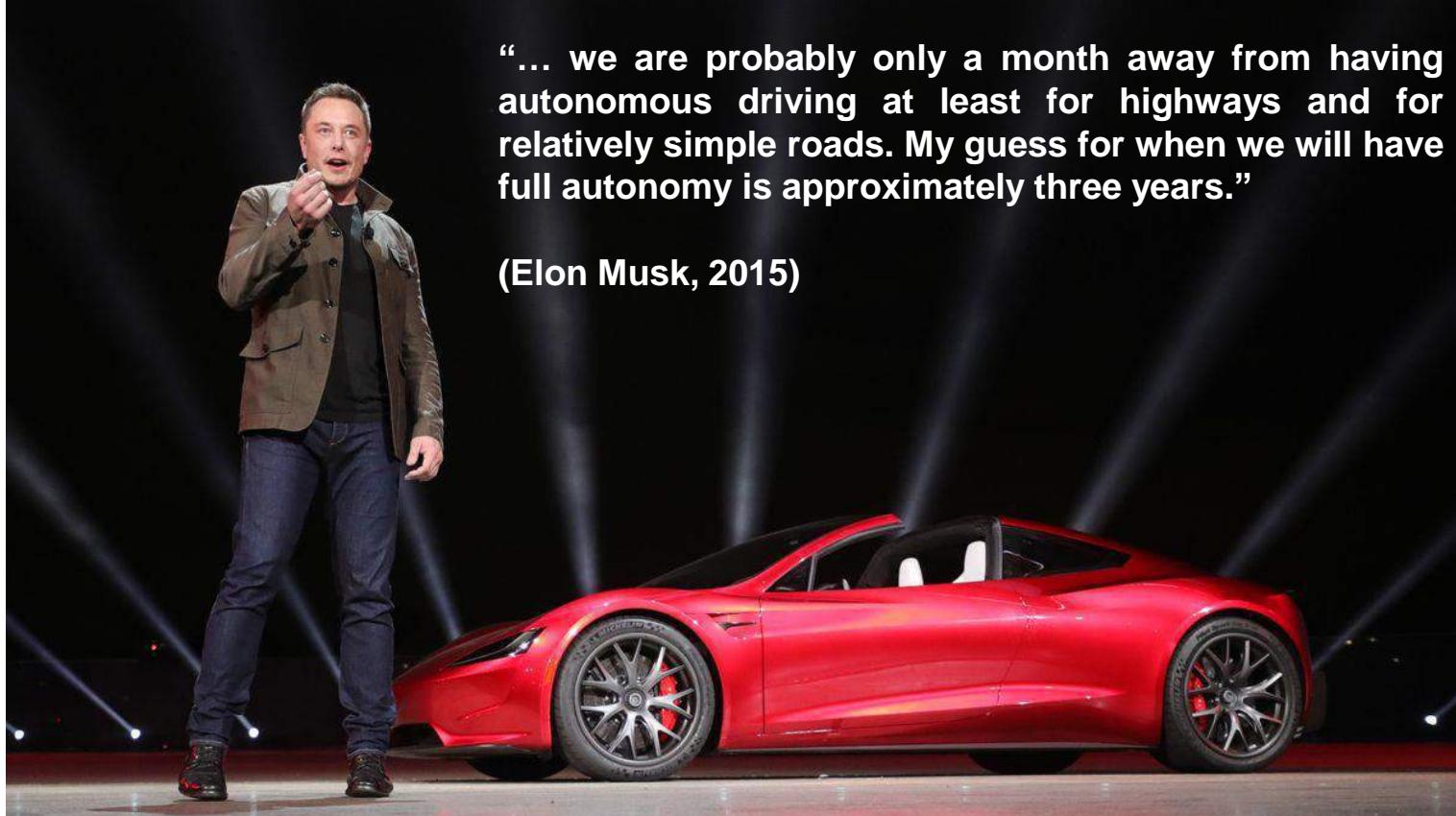
Red light classified as green
after one pixel change.

**Winner Nexar traffic light challenge:
On average, it takes only 3 pixels to
turn red into green or green into red!**

Human-in-the-Loop: People are still needed.

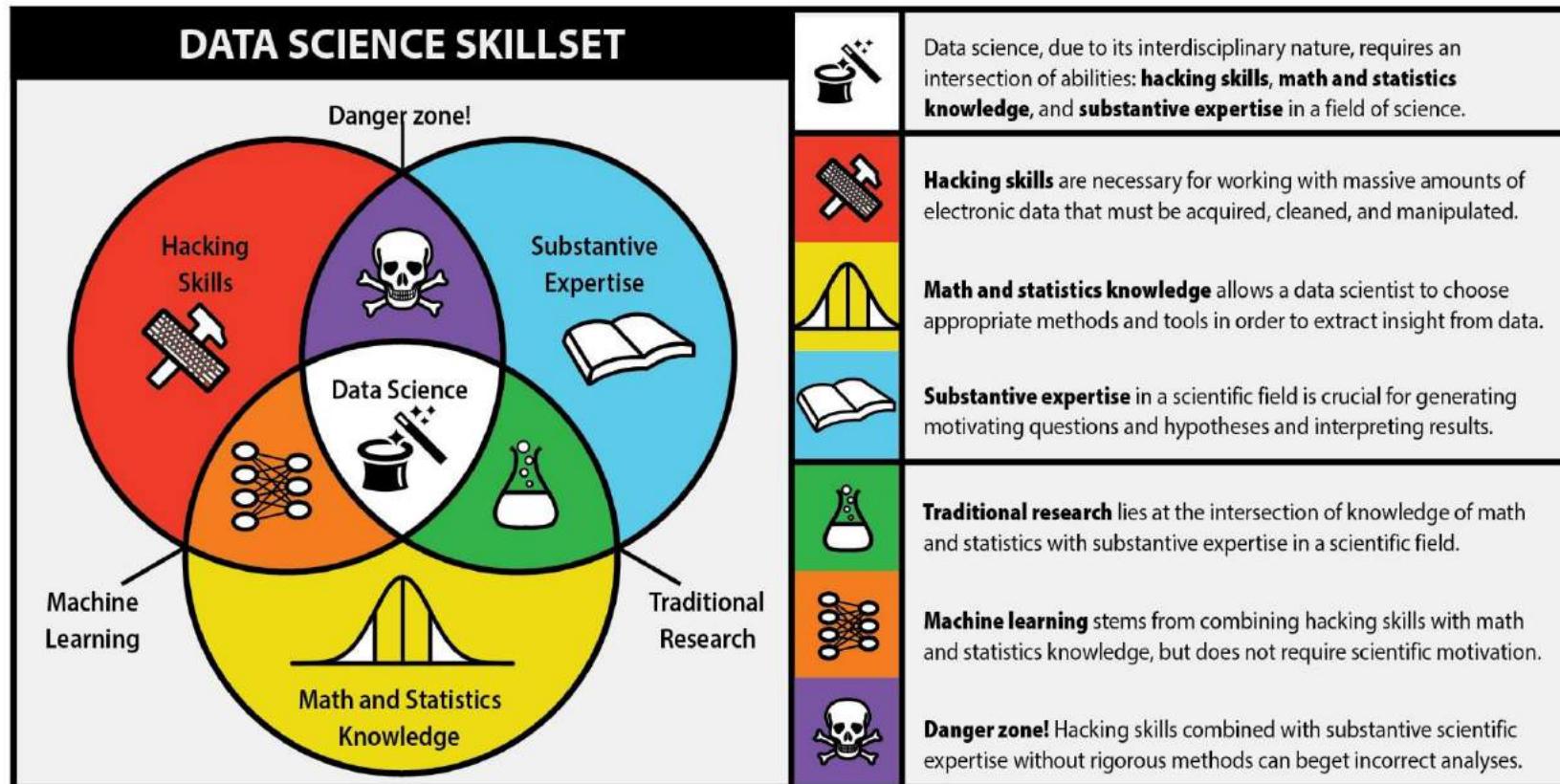
“... we are probably only a month away from having autonomous driving at least for highways and for relatively simple roads. My guess for when we will have full autonomy is approximately three years.”

(Elon Musk, 2015)

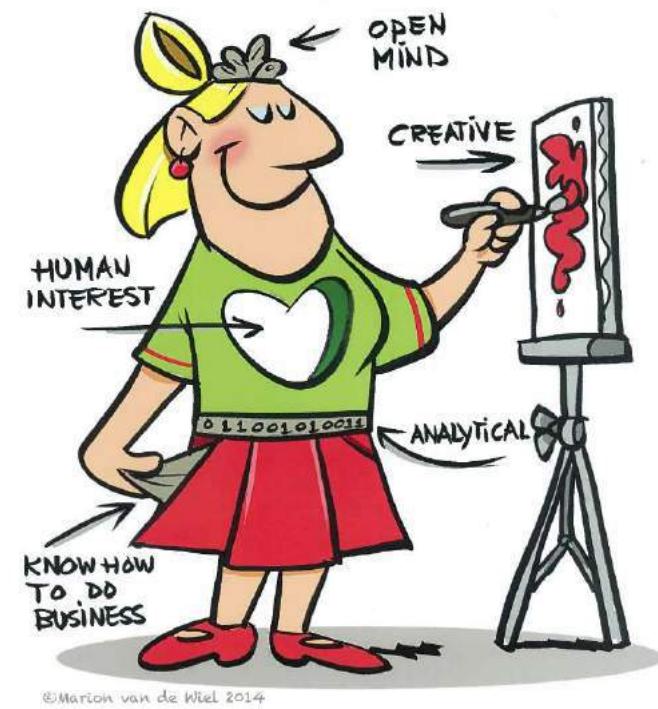


Therefore, models need to be understandable and we need hybrid forms of intelligence.

We need well-trained Data Scientists !



THE PERFECT DATA SCIENTIST



Overview

1. Introduction to Data Science

2. Decision Trees

3. Clustering

4. Frequent Itemsets

5. Association Rules and Sequence Mining

6. Data Modeling, Quality, and the Data Science Process

7. Process Mining

8. Text Mining

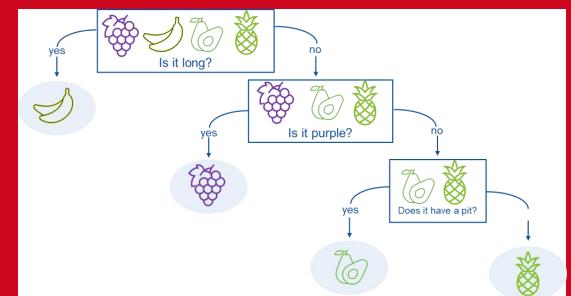
9. Responsible Data Science

- Introduction
- What is (tabular) data
- What is data science
- Challenges: reliability, biases, and responsible data science
- Types of data: a high-level taxonomy
- Descriptive statistics
- Simpson's Paradox, spurious correlations
- Basic visualization: plotting, boxplots, histograms, distributions, scatter plots, bar plots
- One-hot encoding, binning
- “How to Lie with Statistics”

Overview

1. **Introduction to Data Science**
2. **Decision Trees**
3. **Clustering**
4. **Frequent Itemsets**
5. **Association Rules and Sequence Mining**
6. **Data Modeling, Quality, and the Data Science Process**
7. **Process Mining**
8. **Text Mining**
9. **Responsible Data Science**

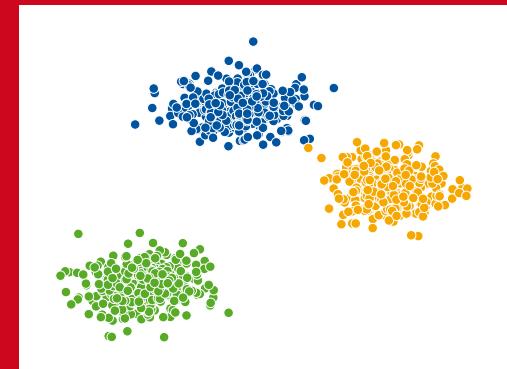
- Decision trees: intro and definitions
- Entropy, information gain, Gini
- ID3 algorithm
- Pruning
- Boosting, bagging, random forests
- Dealing with continuous attributes



Overview

1. **Introduction to Data Science**
2. **Decision Trees**
3. **Clustering**
4. **Frequent Itemsets**
5. **Association Rules and Sequence Mining**
6. **Data Modeling, Quality, and the Data Science Process**
7. **Process Mining**
8. **Text Mining**
9. **Responsible Data Science**

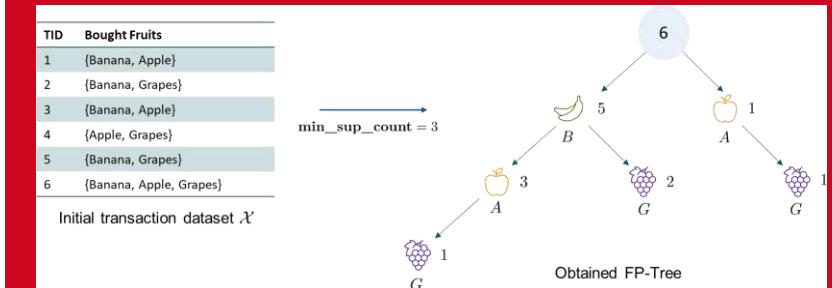
- Clustering: intro and definitions
- Distance measures
- K means, K medoids
- Agglomerative clustering
- DBSCAN



Overview

- 1. Introduction to Data Science**
- 2. Decision Trees**
- 3. Clustering**
- 4. Frequent Itemsets**
- 5. Association Rules and Sequence Mining**
- 6. Data Modeling, Quality, and the Data Science Process**
- 7. Process Mining**
- 8. Text Mining**
- 9. Responsible Data Science**

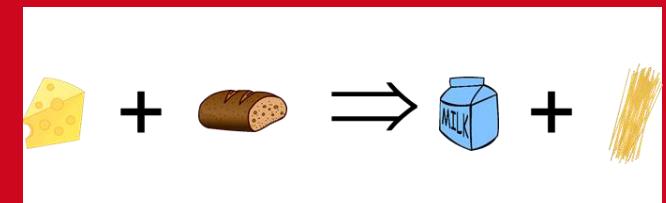
- Frequent itemsets: intro and definitions
- Support and itemsets properties
- The Apriori algorithm
- The FP-Growth algorithm



Overview

1. **Introduction to Data Science**
2. **Decision Trees**
3. **Clustering**
4. **Frequent Itemsets**
5. **Association Rules and Sequence Mining**
6. **Data Modeling, Quality, and the Data Science Process**
7. **Process Mining**
8. **Text Mining**
9. **Responsible Data Science**

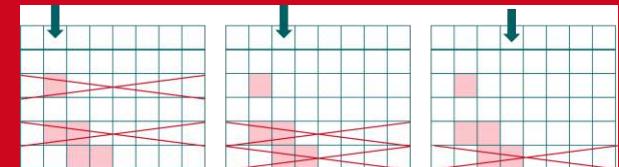
- Association rules: intro and definitions
- Generating and evaluating association rules
- Simpson's Paradox in association rules
- Sequence mining: intro and definitions
- The Apriori-all algorithm



Overview

1. **Introduction to Data Science**
2. **Decision Trees**
3. **Clustering**
4. **Frequent Itemsets**
5. **Association Rules and Sequence Mining**
6. **Data Modeling, Quality, and the Data Science Process**
7. **Process Mining**
8. **Text Mining**
9. **Responsible Data Science**

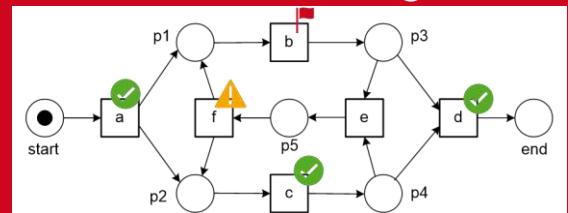
- Data modeling and quality: intro and definitions
- Beyond tables: Temporal data, time series, event data
- Data science processes and methodologies
 - PDCA (Plan, Do, Check, Act)
 - DMAIC (Define, Measure, Analyze, Improve, and Control)



Overview

1. **Introduction to Data Science**
2. **Decision Trees**
3. **Clustering**
4. **Frequent Itemsets**
5. **Association Rules and Sequence Mining**
6. **Data Modeling, Quality, and the Data Science Process**
7. **Process Mining**
8. **Text Mining**
9. **Responsible Data Science**

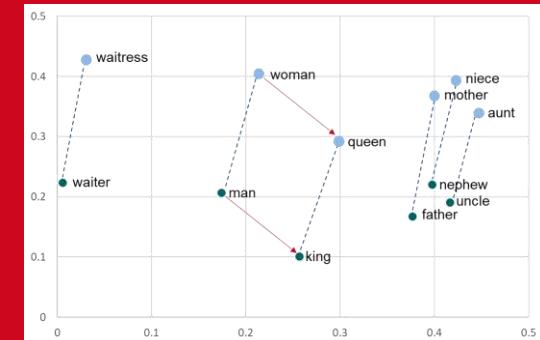
- Process mining: intro and definitions
- Models and formalisms
- Process discovery: bottom-up and top-down
- The inductive miner
- Token-based replay
- Fitness and token-based replay conformance checking



Overview

1. **Introduction to Data Science**
2. **Decision Trees**
3. **Clustering**
4. **Frequent Itemsets**
5. **Association Rules and Sequence Mining**
6. **Data Modeling, Quality, and the Data Science Process**
7. **Process Mining**
8. **Text Mining**
9. **Responsible Data Science**

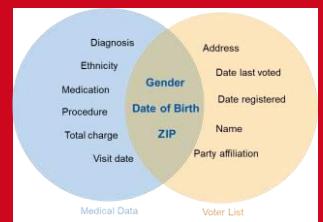
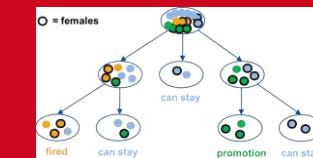
- Text mining: intro and definitions
- Text preprocessing
- Text models: BoW and tf-idf
- N-grams
- Autoencoding
- word2vec



Overview

- 1. Introduction to Data Science**
- 2. Decision Trees**
- 3. Clustering**
- 4. Frequent Itemsets**
- 5. Association Rules and Sequence Mining**
- 6. Data Modeling, Quality, and the Data Science Process**
- 7. Process Mining**
- 8. Text Mining**
- 9. Responsible Data Science**

- RDS: intro and definitions
- Anonymization
- K-Anonymity, L-Diversity, T-Closeness
- Unfairness, prejudice, discrimination
- Fairness risks and metrics
- Fair decision trees
- FACT vs FAIR





Elements of Machine Learning & Data Science

Winter semester 2023/24

**Empirical analysis
and performance optimization (AutoML)**

Prof. Holger Hoos

Chair for AI Methodology (AIM)

AIM:

- machine learning
- automated reasoning
- optimisation
- empirical analysis of (AI) algorithms
- automated design of (AI) algorithms
- human-centred AI
- AI for Good, AI for All



Prof. Holger Hoos



Dr. Jakob Bossek



Dr. Igor Vatolkin



Marie Anastacio, MSc



Julian Dierkes, MSc



Henning Duwe, MSc



Wadie Skaf, MSc

Key questions:

- **How good is an ML model?**
- **How good could an ML model be?**

Key questions:

- **How good is an ML model?**
 - Is it “fit for use” (i.e., good enough for deployment)?
 - What are its strengths and weaknesses?
 - Might anything have gone wrong during training?

Key questions:

- **How good is an ML model?**
 - How do we assess whether it is “fit for use” (i.e., good enough for deployment)?
 - How do we assess its strengths and weaknesses?
 - How do we detect if anything has gone wrong during training?

Key questions:

- **How good could an ML model be?**
 - Are we using the best possible ML method / model?
 - Have we configured and trained it in the best possible way?
 - Can we further improve performance?

Key questions:

- **How good could an ML model be?**
 - How can we ensure we are using a good ML method / model?
 - How can we configure and train it for optimised performance?
 - How can we further improve performance?

High-level learning goals:

Be able to ...

- answer these key questions in a technical manner;
- recognise weaknesses in the empirical performance of ML models using standard tools and methods;
- explain these analysis tools and methods at a technical level;
- use standard tools and methods for selecting models and optimising their hyperparameters;
- explain these AutoML tools and methods at a technical level.



Auto-WEKA: Combined Selection and Hyperparameter Optimization of Classification Algorithms

Chris Thornton Frank Hutter Holger H. Hoos Kevin Leyton-Brown

Department of Computer Science, University of British Columbia
201-2366 Main Mall, Vancouver BC, V6T 1Z4, Canada
{cwt Thornton, hutter, hoos, kevinlb}@cs.ubc.ca



Test of Time Award for Research

Auto-WEKA: Combined Selection and Hyperparameter Optimization of Classification Algorithms
Chris Thornton ,Frank Hutter, Holger H. Hoos, Kevin Leyton-Brown

COMMUNICATIONS
OF THE
ACM

CACM.ACM.ORG 03/2022 VOL.65 NO.03

Automating Data Science

The Pushback Effects of Race, Ethnicity, Gender, and Age in Code Review

Does AI Improve Recruiting?

Which Competencies Should Data Analytics Experts Have?

The Troubling Future for Facial Recognition Software

A Conversation with Margo Seltzer and Mike Olson

75 ACM 2022

Association for Computing Machinery

review articles

DOI:10.1145/3485256

Given the complexity of data science projects and related demand for human expertise, automation has the potential to transform the data science process.

BY TIJL DE BIE, LUC DE RAEDT, JOSÉ HERNÁNDEZ-ORALLO, HOLGER H. HOOS, PADHRAIC SMYTH, AND CHRISTOPHER K.I. WILLIAMS

Automating Data Science

DATA SCIENCE COVERS the full spectrum of deriving insight from data, from initial data gathering and interpretation, via processing and engineering of data, and exploration and modeling, to eventually producing novel insights and decision support systems.

Data science can be viewed as overlapping or broader in scope than other data-analytic methodological disciplines, such as statistics, machine learning, databases, or visualization.¹⁰

To illustrate the breadth of data science, consider, for example, the problem of recommending items (movies, books, or other products) to customers. While the core of these applications can consist of algorithmic techniques such as matrix factorization, a deployed system will involve a much wider range of technological and human considerations. These range from scalable back-end transaction systems that retrieve customer and product data in real time, experimental design for evaluating system changes, causal analysis for understanding the effect

of interventions, to the human factors and psychology that underlie how customers react to visual information displays and make decisions.

As another example, in areas such as astronomy, particle physics, and climate science, there is a rich tradition of building computational pipelines to support data-driven discovery and hypothesis testing. For instance, geoscientists use monthly global landcover maps based on satellite imagery at sub-kilometer resolutions to better understand how the Earth's surface is changing over time.¹¹ These maps are interactive and browsable, and they are the result of a complex data-processing pipeline, in which terabytes to petabytes of raw sensor and image data are transformed into databases of automatically detected and annotated objects and information. This type of pipeline involves many steps, in which human decisions and insight are critical, such as instrument calibration, removal of outliers, and classification of pixels.

The breadth and complexity of these and many other data science scenarios means the modern data scientist requires broad knowledge and experience across a multitude of topics. Together with an increasing demand for data analysis skills, this has led to a shortage of trained data scientists with appropriate background and experience, and significant market competition for limited expertise. Considering this bottleneck, it is not surprising there is increasing interest in automat-

» key insights

- Automation in data science aims to facilitate and transform the work of data scientists, not to replace them.
- Important parts of data science are already being automated, especially in the modeling stages, where techniques such as automated machine learning (AutoML) are gaining traction.
- Other aspects are more difficult to automate, not only because of technological challenges, but because open-ended and context-dependent tasks require human interaction.

» key insights

- Automation in data science aims to facilitate and transform the work of data scientists, not to replace them.
- Important parts of data science are already being automated, especially in the modeling stages, where techniques such as automated machine learning (AutoML) are gaining traction.
- Other aspects are more difficult to automate, not only because of technological challenges, but because open-ended and context-dependent tasks require human interaction.

ILLUSTRATION BY JARIN HAN

Data Engineering:

data wrangling,
data integration,
data preparation,
data transformation,

...

Model Building:

algorithm selection,
parameter optimization,
performance evaluation,
model selection,

...

Data Exploration:

domain understanding,
goal exploration,
data aggregation,
data visualization,

...

Exploitation:

model interpretation and visualization,
reporting and narratives,
predictions and decisions,
monitoring and maintenance,

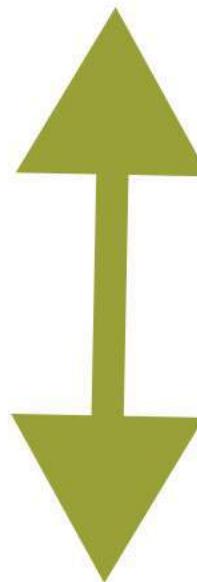
...

More
open-ended



Less
open-ended

Less
dependent on
domain context



More
dependent on
domain context

Behold ...

... the awesome power of AutoML / AutoDS / AutoAI!

But:

With great power comes great responsibility :-)

Enjoy the course and see you in our lectures!

Elements of Machine Learning & Data Science

Winter semester 2023/24

Lecture 2 – Introduction to ML

13.10.2023

Prof. Bastian Leibe

Announcements: Moodle

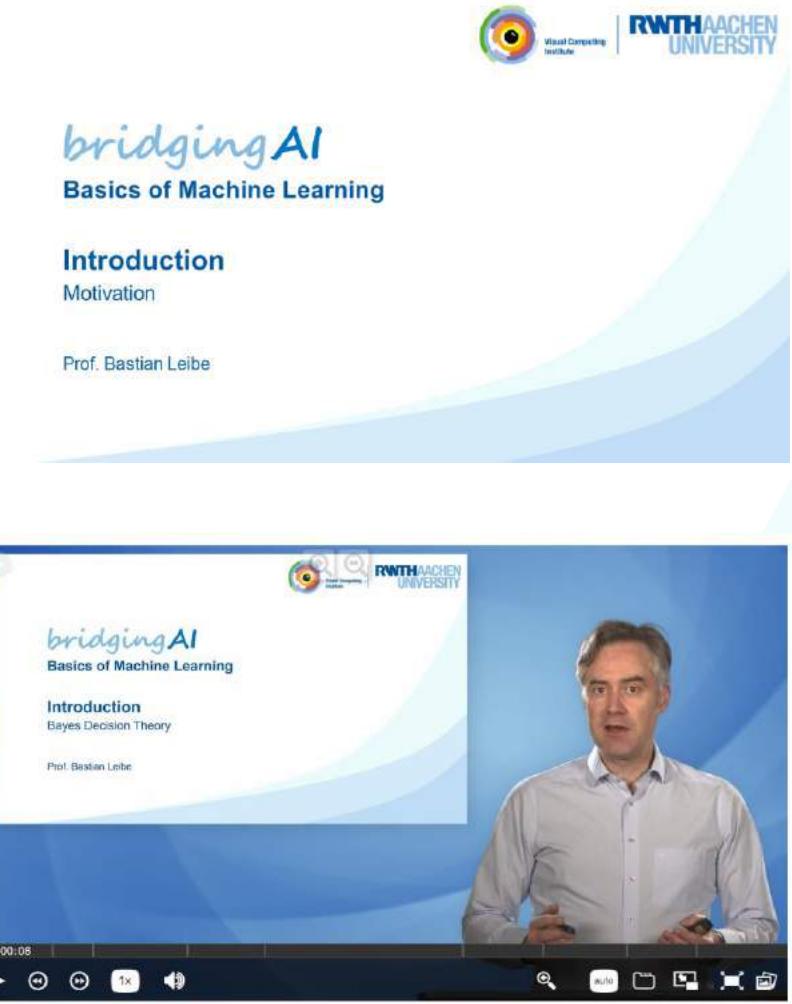
- **Materials provided on moodle**
 - Pdfs of the slides
 - **Video recording** of the previous lecture
 - Pre-recorded videos for *this* lecture
- *If you haven't done so yet, please register for the class to get access to the moodle.*

The diagram illustrates the structure of the Moodle announcements. It features four main sections, each preceded by a blue arrow pointing from the left:

- Lecture 1: Introduction & Organization (10.10.2023, all instructors)**
 - EleMLDS-ws23-part01-intro.pdf (5.9 MB) Hochgeladen 10.10.2023 13:23 Als erledigt kennzeichnen
 - EleMLDS-ws23-part01-intro-6on1.pdf (1.4 MB) Hochgeladen 10.10.2023 13:24 Als erledigt kennzeichnen
 - elemlds23-part01-intro.mp4
Video recording of the lecture on 10.10.2023.
- Lecture 2: Introduction to Machine Learning (13.10.2023, Leibe)**
 - Pre-recorded videos
Here we provide several pre-recorded videos that together cover the topics from Lecture 2 and [part of] Lecture 3. You can watch them either ahead of the lecture or use them for in-depth repetition.
 - EleMLDS-ws23-part02-intro-to-ml.pdf (959.6 KB) Hochgeladen 13.10.2023 14:25 Als erledigt kennzeichnen
 - EleMLDS-ws23-part02-intro-to-ml-6on1.pdf (575.4 KB) Hochgeladen 13.10.2023 14:25 Als erledigt kennzeichnen

Announcements: Pre-recorded Videos

- **Companion MOOCs**
 - We have created two MOOCs to complement this lecture
 - Basics of ML
 - Basics of Data Science
 - Target: International Master students and students from other degree programs joining Computer Science
- **The pre-recorded videos are part of those MOOCs**
 - Extended explanations of key lecture topics
 - High production value
 - They may not cover the full topic range of the lecture
 - *Please use them as supplementary material*



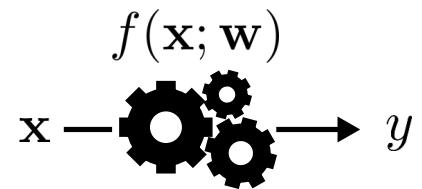
Announcement: Small-Group Exercises

Monday	Tuesday	Wednesday	Thursday	Friday
14:30-18:00h	3x 14:30-16:00h	2x 14:30-16:00h		
3x 16:30-18:00h	3x 16:30-18:00h	2x 16:30-18:00h		
18:30-20:00h	2x 18:30-20:00h			

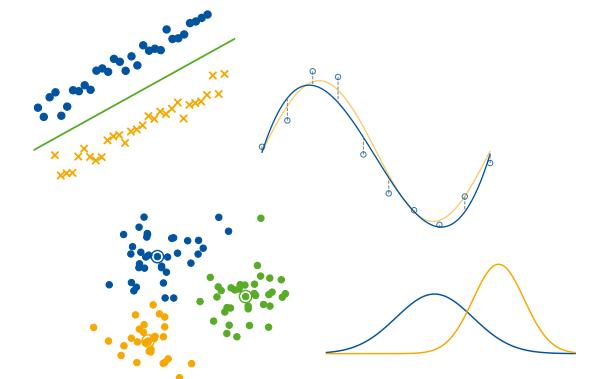
- **Bi-weekly small-group exercises**
 - We're currently setting up a poll to collect your preferences for the exercise slots
 - Please enter your choices until Wed, 18.10. evening!
 - Based on the poll results, we will assign you to exercise slots
 - *We will send out an email announcement with detailed instructions tonight or on Saturday...*

Machine Learning Topics

1. **Introduction to ML**
2. Probability Density Estimation
3. Linear Discriminants
4. Linear Regression
5. Logistic Regression
6. Support Vector Machines
7. AdaBoost
8. Neural Network Basics



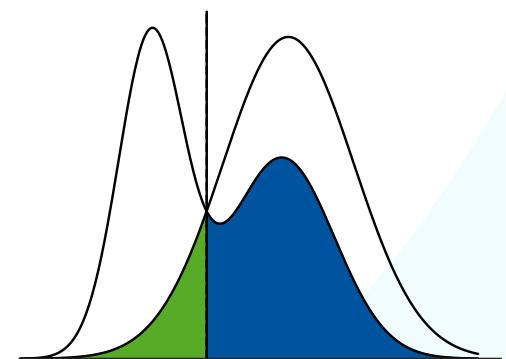
Machine Learning
Concepts



Forms of Machine Learning

$$p(\mathcal{C}|\mathbf{x}) = \frac{p(\mathbf{x}|\mathcal{C})p(\mathcal{C})}{p(\mathbf{x})}$$

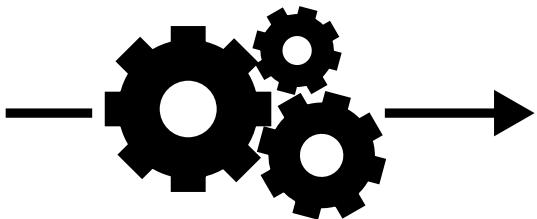
Bayes Decision Theory



Bayes Optimal
Classification

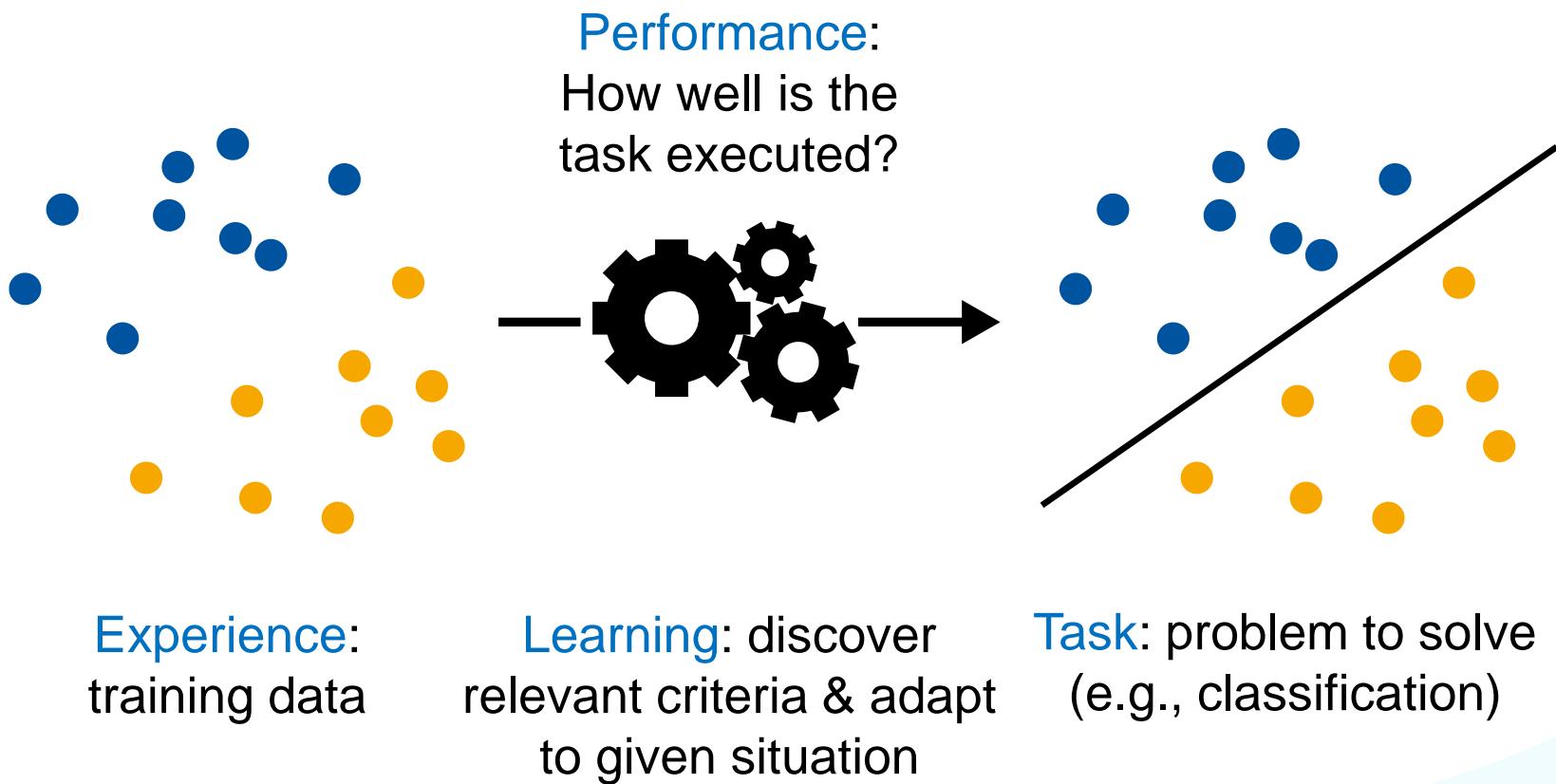
Topics for Today

1. Motivation
2. Forms of Learning
3. Terms, Concepts, and Notation
4. Bayes Decision Theory



What is Machine Learning?

*Machines that **learn** to perform a task from experience*



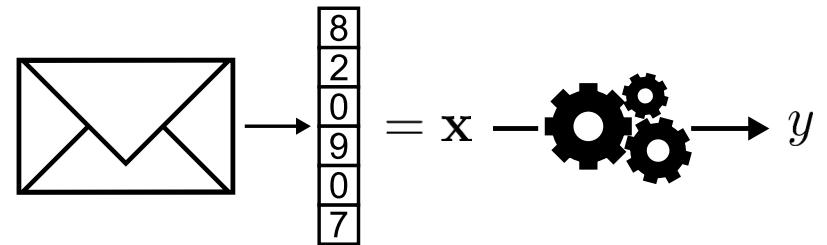
Mathematical Formulation

*Machines that learn to perform a **task** from experience*

Often described through a mathematical function:

$$y = f(\mathbf{x}; \mathbf{w})$$

Output y Input \mathbf{x} Parameters \mathbf{w}
(learned!)



Discrete targets: **Classification**

$y \in \{\text{important, spam}\}$

Continuous targets: **Regression**

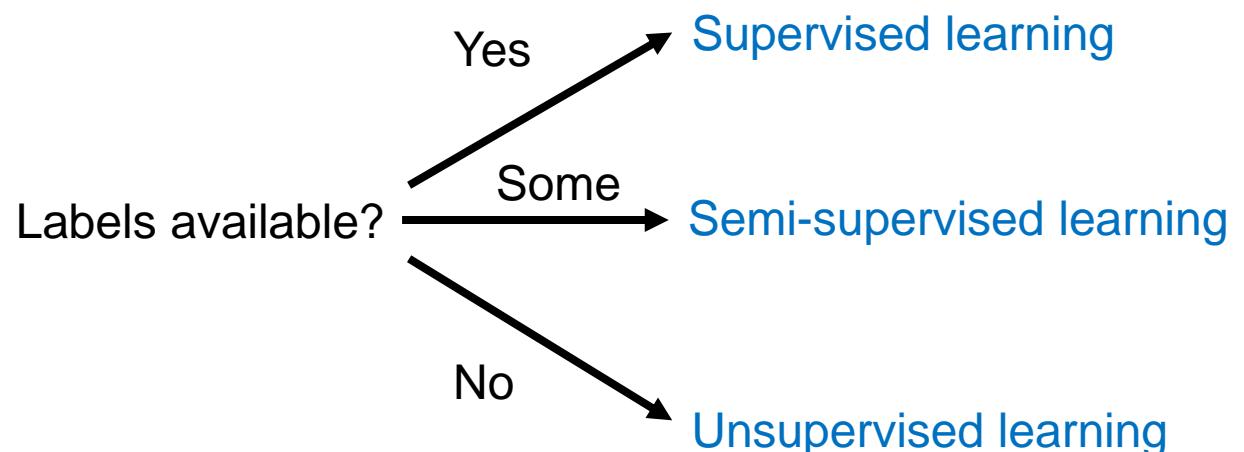
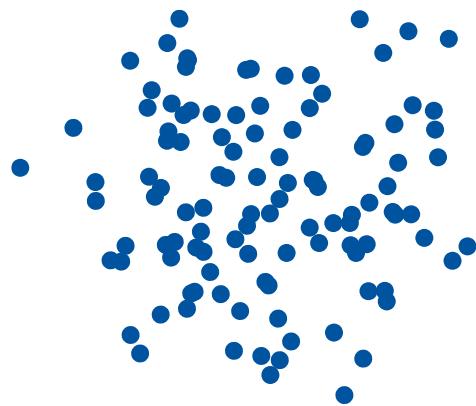
$y = p(\text{spam}) \in [0, 1]$

Learning from Data

Machines that learn to perform a task from experience

Learning from collected samples:

$$\mathcal{X} = \{\mathbf{x}_1, \dots, \mathbf{x}_N\}$$



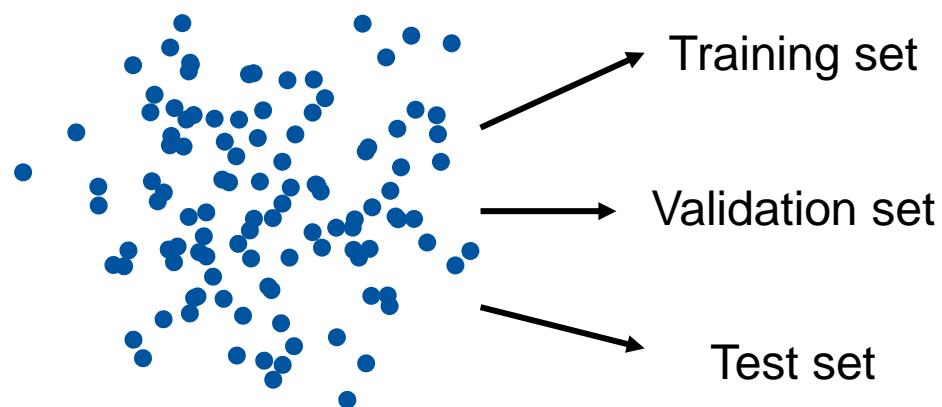
Learning via sparse feedback:

Reinforcement learning

Measuring Success

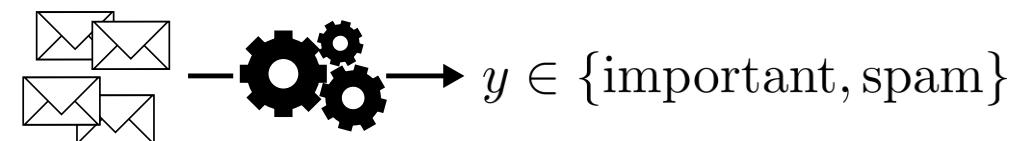
*Machines that learn to **perform** a task from experience*

- Performance measure: typically a single number.
 - Calculate with a suitable **metric**.
- Divide data into disjoint subsets:

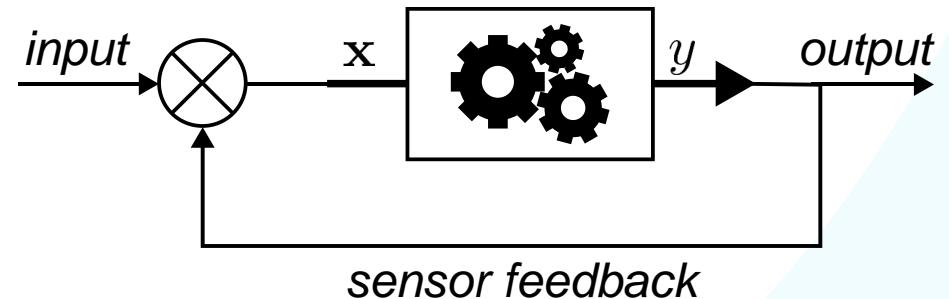


- Measure generalization performance on test set.

E.g., % correctly recognized spam mails



E.g., average distance to desired endpoint

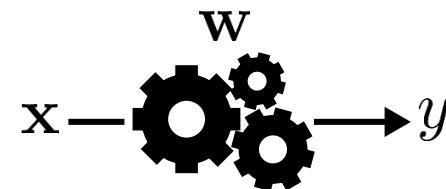


Learning as Optimization

*Machines that **learn** to perform a task from experience*

Learning = optimizing $f(\mathbf{x}; \mathbf{w})$

\mathbf{w} describes the type
of model that we use.

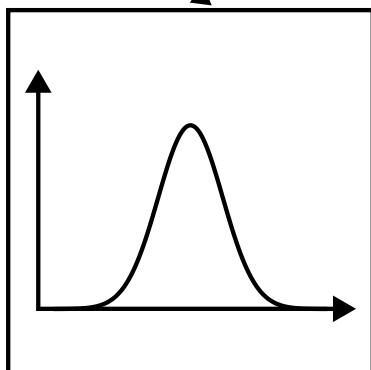
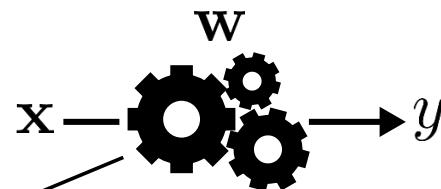


Learning as Optimization

*Machines that **learn** to perform a task from experience*

Learning = optimizing $f(\mathbf{x}; \mathbf{w})$

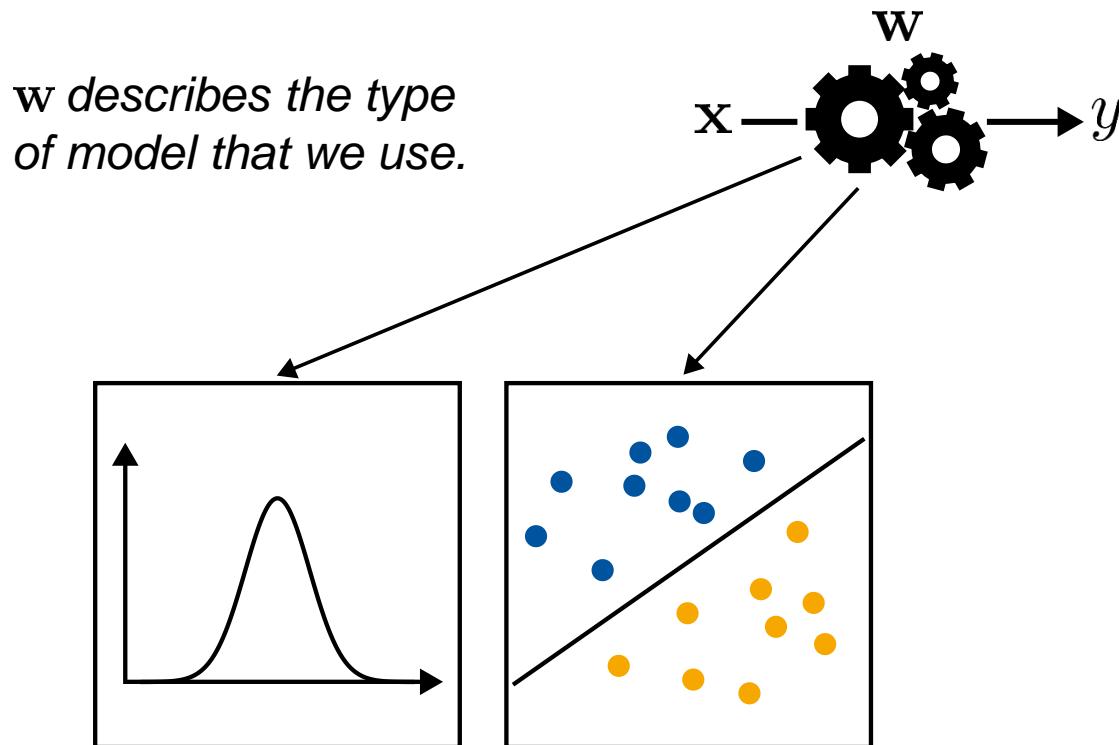
\mathbf{w} describes the type
of model that we use.



Learning as Optimization

*Machines that **learn** to perform a task from experience*

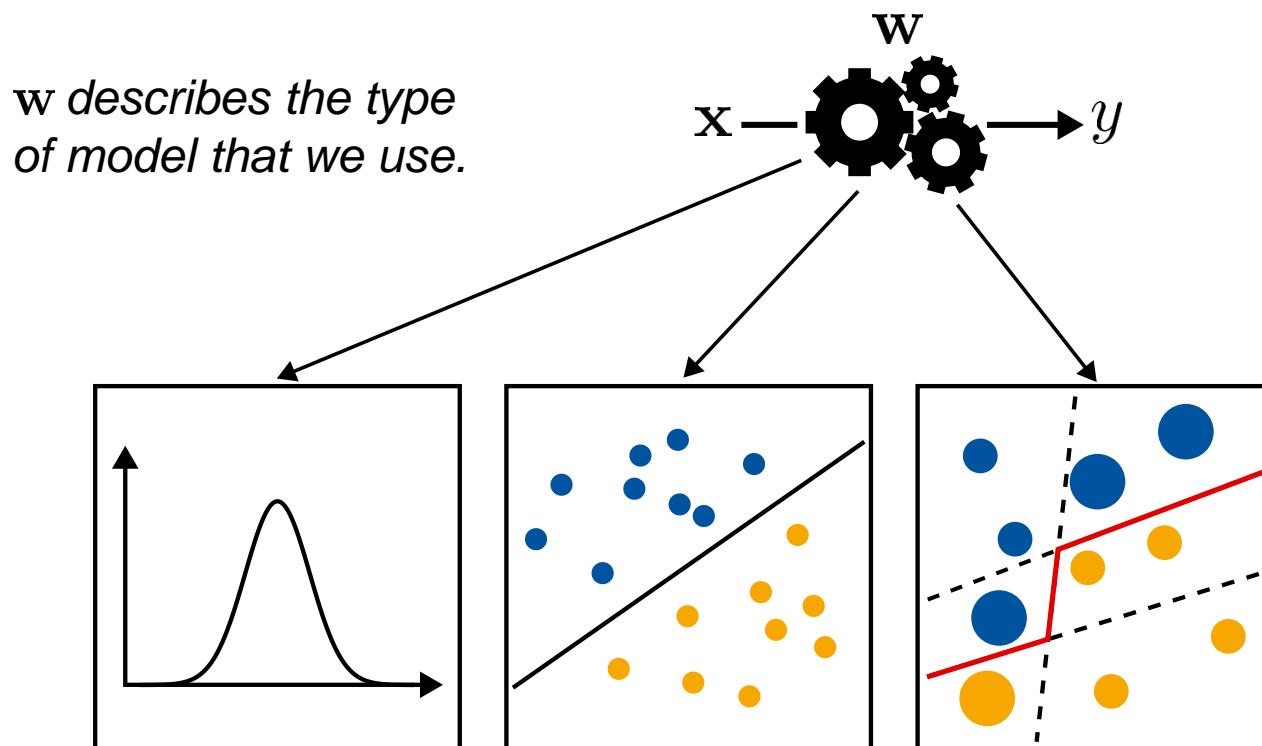
Learning = optimizing $f(\mathbf{x}; \mathbf{w})$



Learning as Optimization

*Machines that **learn** to perform a task from experience*

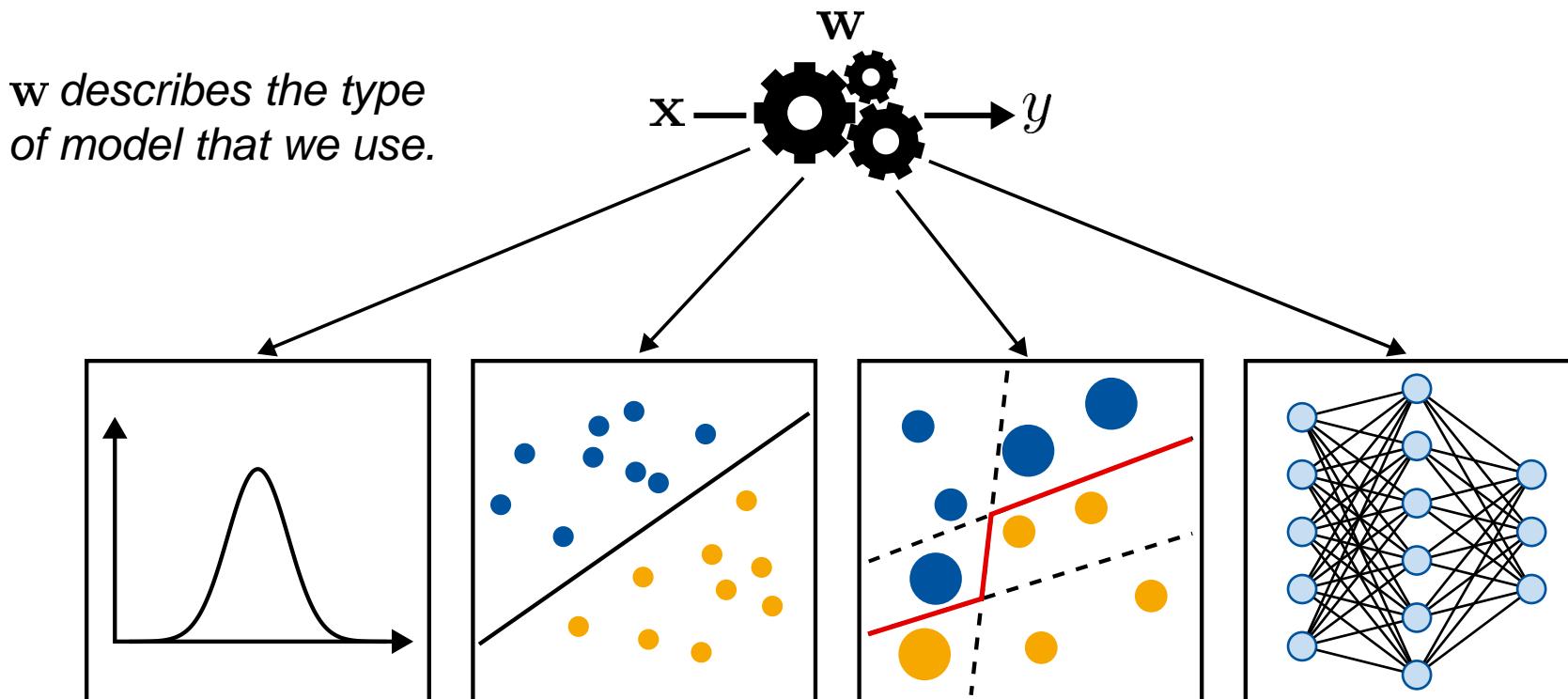
Learning = optimizing $f(\mathbf{x}; \mathbf{w})$



Learning as Optimization

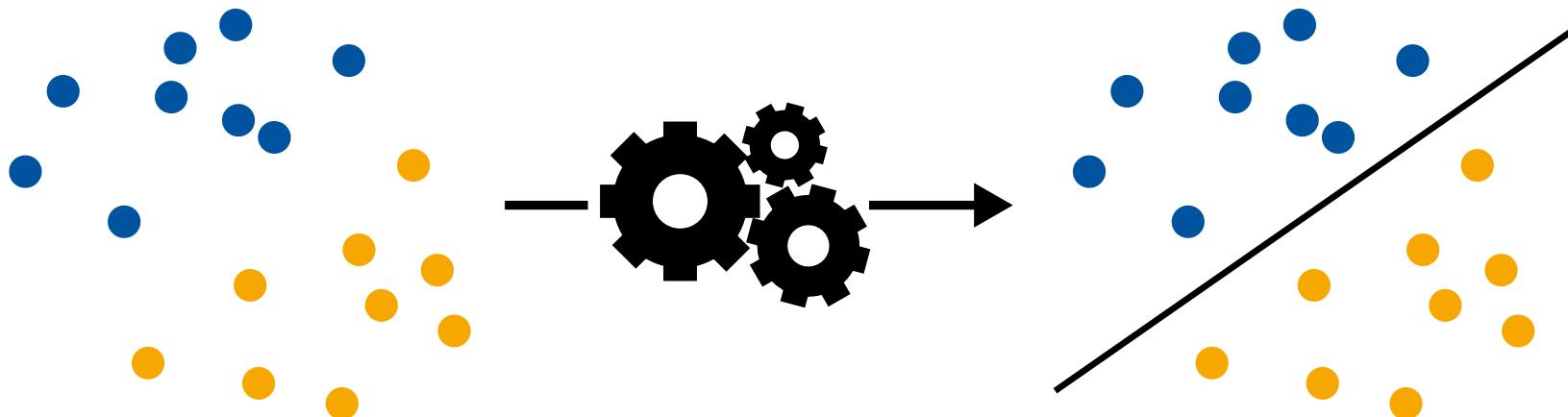
*Machines that **learn** to perform a task from experience*

Learning = optimizing $f(\mathbf{x}; \mathbf{w})$



What is Machine Learning?

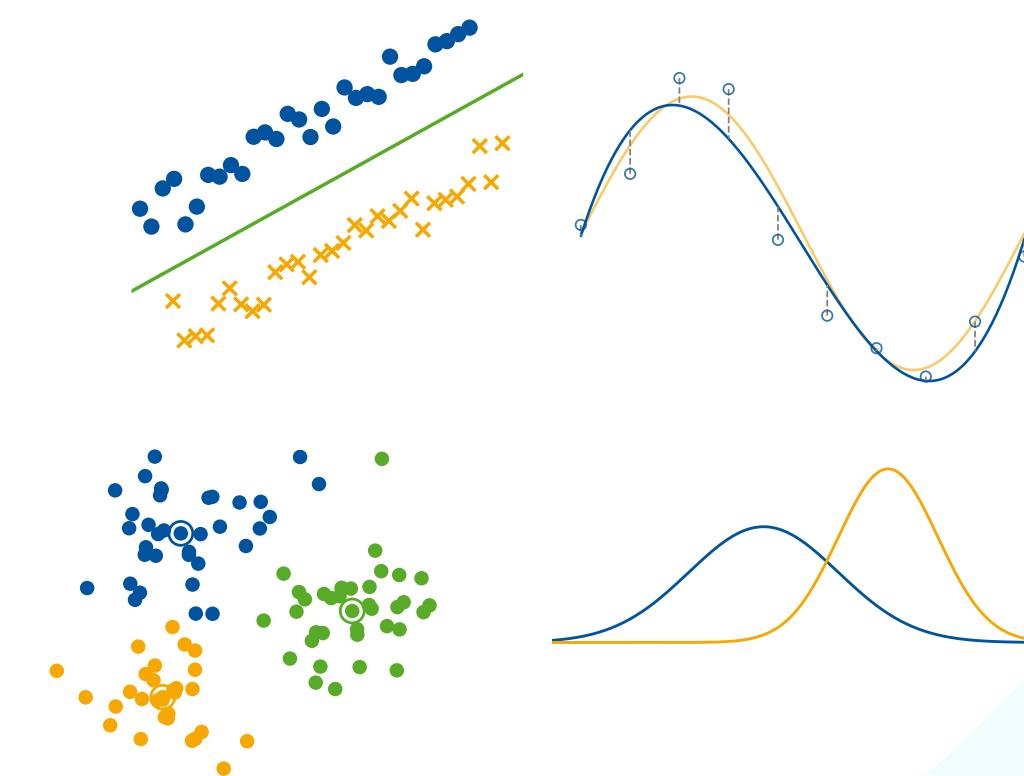
*Machines that **learn** to perform a task from experience*



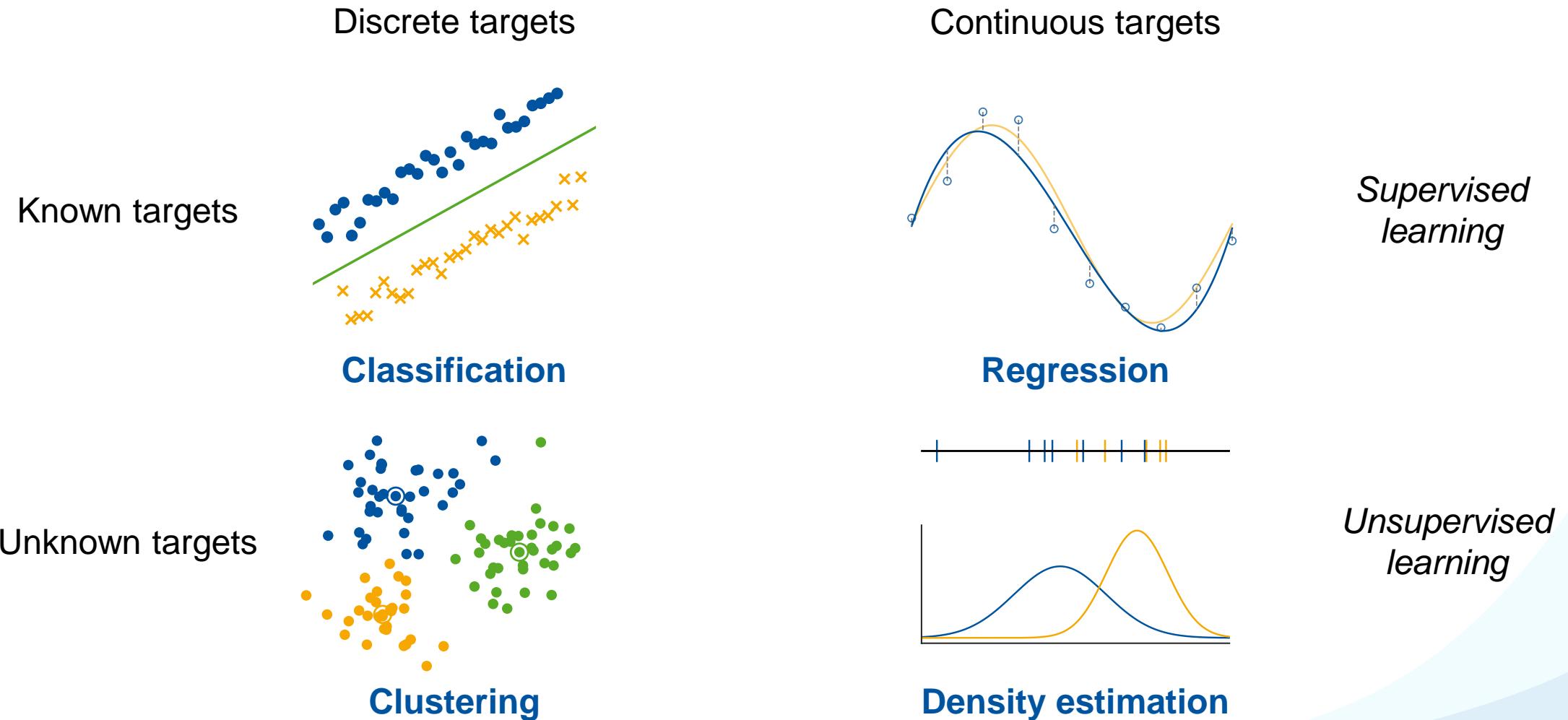
*We will focus on **statistical Machine Learning**.*

Topics for Today

1. Motivation
2. **Forms of Learning**
3. Terms, Concepts, and Notation
4. Bayes Decision Theory



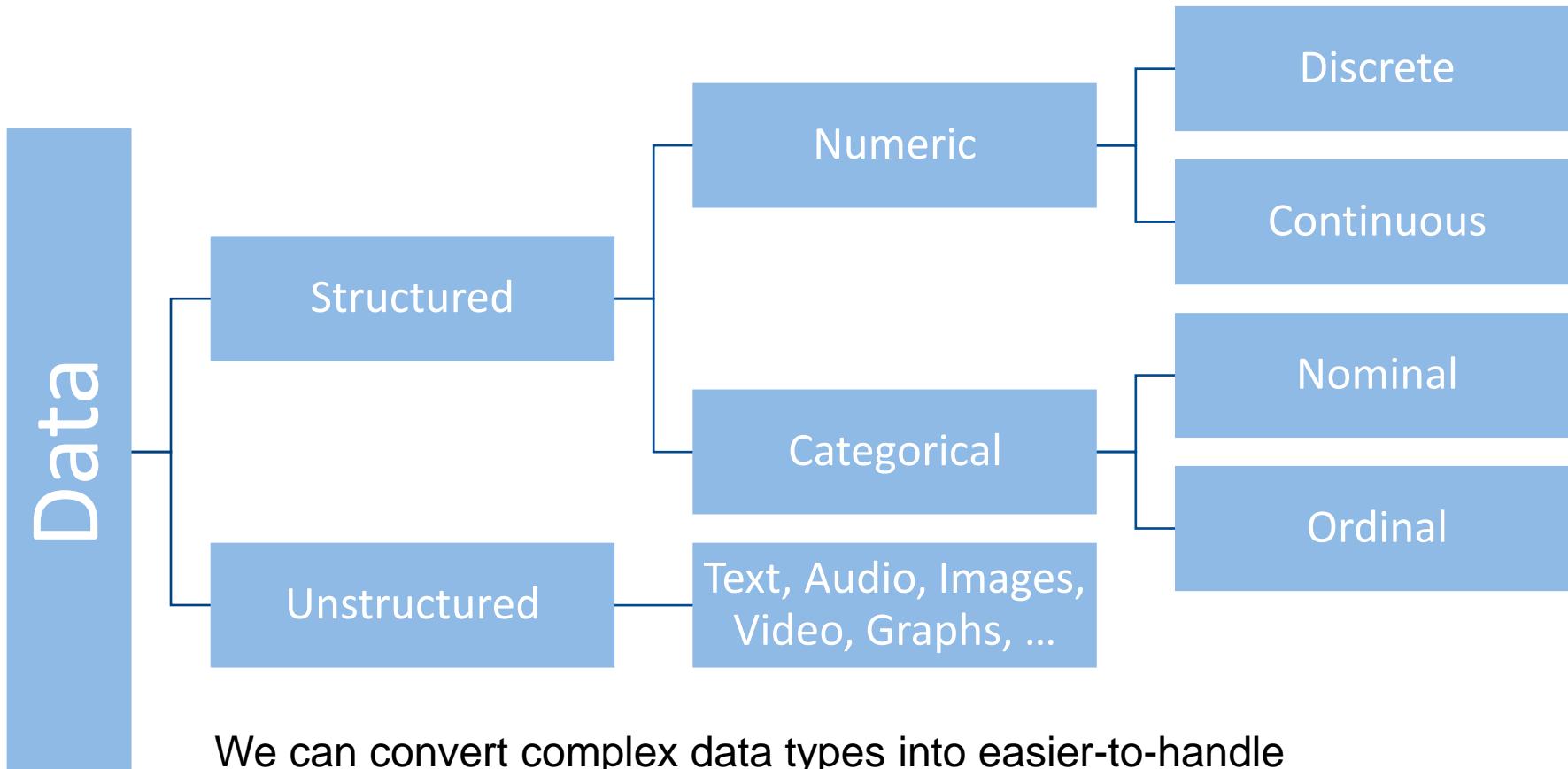
Supervised vs. Unsupervised Learning



Supervised Learning

- We will mostly focus on **supervised learning**.
- Given training data with labels: $\mathcal{D} = \{(\mathbf{x}_1, t_1), \dots, (\mathbf{x}_N, t_N)\}$
- The goal is to learn a predictive function $y(\mathbf{x}; \mathbf{w})$ that yields good performance on unseen test data.
- In real-world scenarios, we also need to preprocess our data to handle, e.g.,
 - Missing or wrong values
 - Outliers
 - Inconsistencies

Data Types - Overview

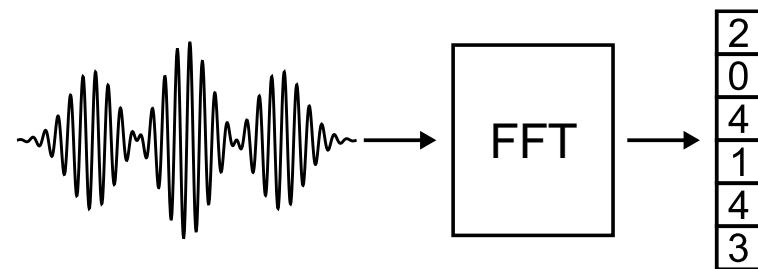


We can convert complex data types into easier-to-handle continuous vector-space data via **feature extraction**.

Features

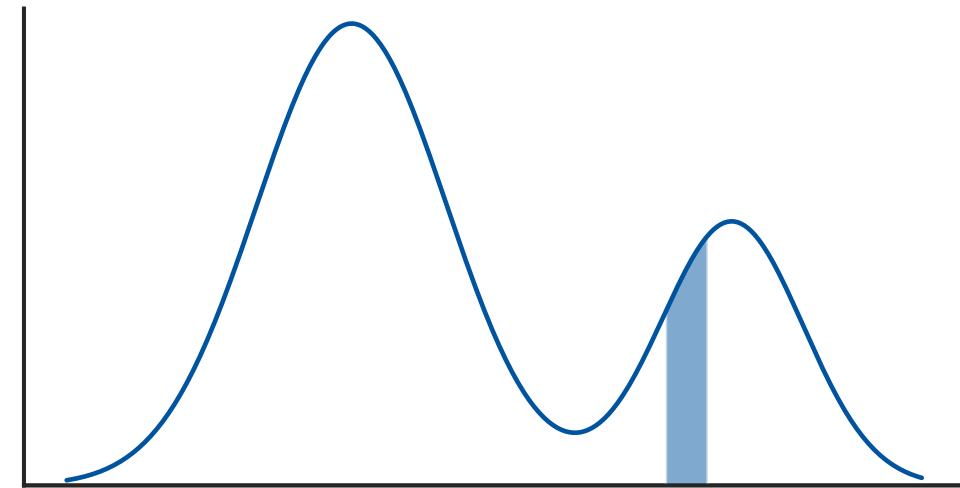
- **Feature extraction** is the process that creates descriptive vectors from samples.
 - Features should be invariant to irrelevant input variations.
 - Selecting the “right” features is crucial.
 - Usually encode some domain knowledge.
 - Higher-dimensional features are more discriminative.
- **Curse of dimensionality**: complexity increases exponentially with number of dimensions.

Example: convert audio snippet to feature vector with Fast Fourier Transform (FFT).



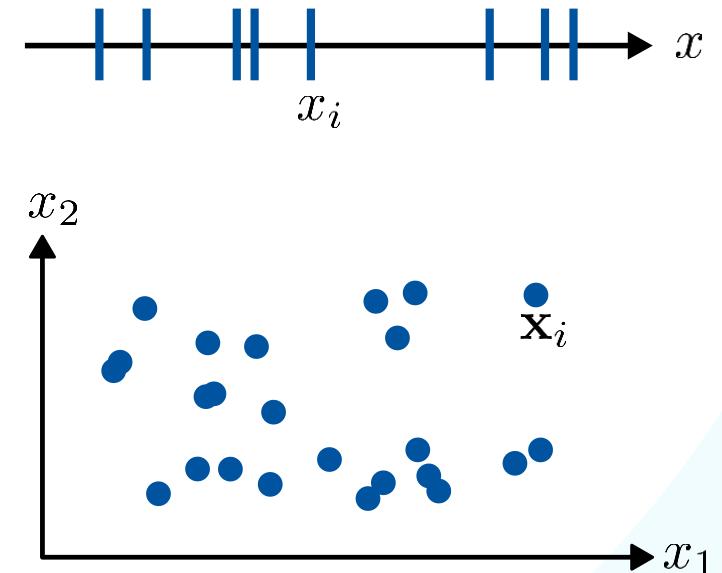
Introduction

1. Motivation
2. Forms of learning
3. **Terms, Concepts, and Notation**
4. Bayes Decision Theory



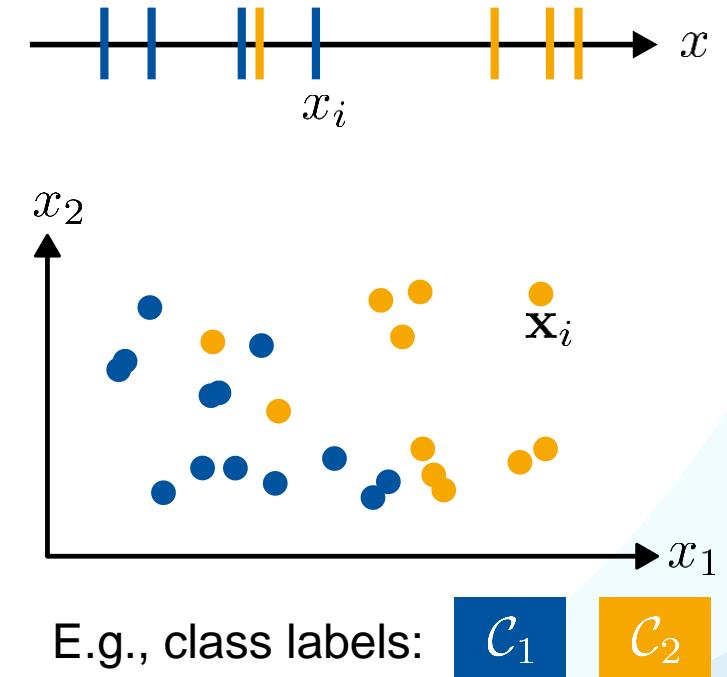
Terms, Concepts, and Notation

- Most of our tools will be based on **statistics** and **probability theory**.
- We will review the most important concepts here.
- Some Notation:
 - Scalar data $x \in \mathbb{R}$
 - Vector-valued data $\mathbf{x} \in \mathbb{R}^D$
 - Datasets $\mathcal{X} = \{x_1, \dots, x_N\}$



Terms, Concepts, and Notation

- Most of our tools will be based on statistics and probability theory.
- We will only review the most important concepts here.
- Some Notation:
 - Scalar data $x \in \mathbb{R}$
 - Vector-valued data $\mathbf{x} \in \mathbb{R}^D$
 - Datasets $\mathcal{X} = \{x_1, \dots, x_N\}$
 - Labelled datasets $\mathcal{D} = \{(x_1, t_1), \dots, (x_N, t_N)\}$
 - Matrices $\mathbf{M} \in \mathbb{R}^{m \times n}$
 - Dot product $\mathbf{w}^\top \mathbf{x} = \sum_{j=1}^D w_j x_j$



Probability Basics

- Probabilities are defined over random variables:

- Discrete case:

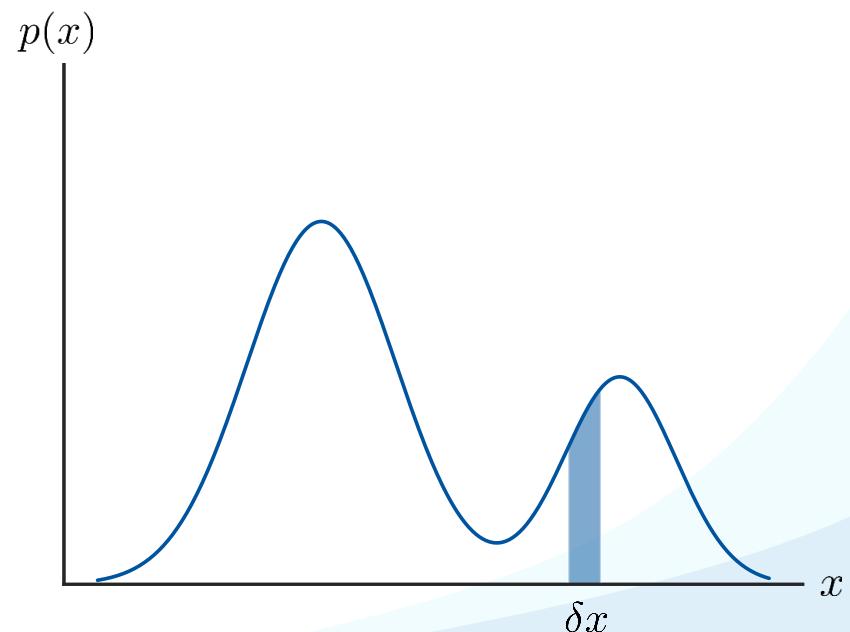
$$p(X = x_j) = \frac{n_j}{N}$$



- Continuous case:

$$p(X \in (x_1, x_2)) = \int_{x_1}^{x_2} p(x) dx$$

Where $p(x)$ is the probability density function (pdf) of x .



Probability Basics

- Random variables $A \in \{a_i\}, B \in \{b_j\}$
- Consider N trials:

$$n_{ij} = \#\{A = a_i \wedge B = b_j\}$$

$$c_i = \#\{A = a_i\}$$

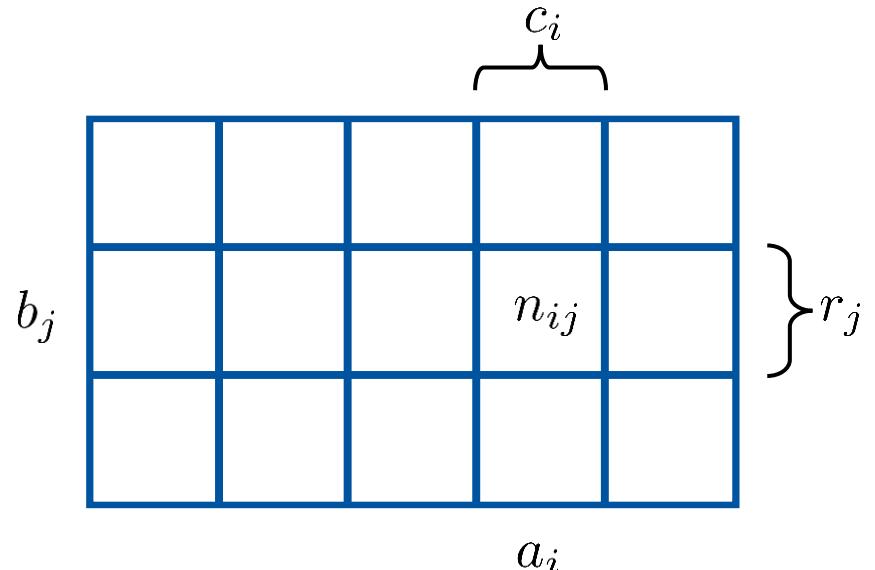
$$r_j = \#\{B = b_j\}$$

- Derive from this:

- Joint probability $p(A = a_i, B = b_j) = \frac{n_{ij}}{N}$

- Marginal probability $p(A = a_i) = \frac{c_i}{N}$

- Conditional probability $p(B = b_j | A = a_i) = \frac{n_{ij}}{c_i}$

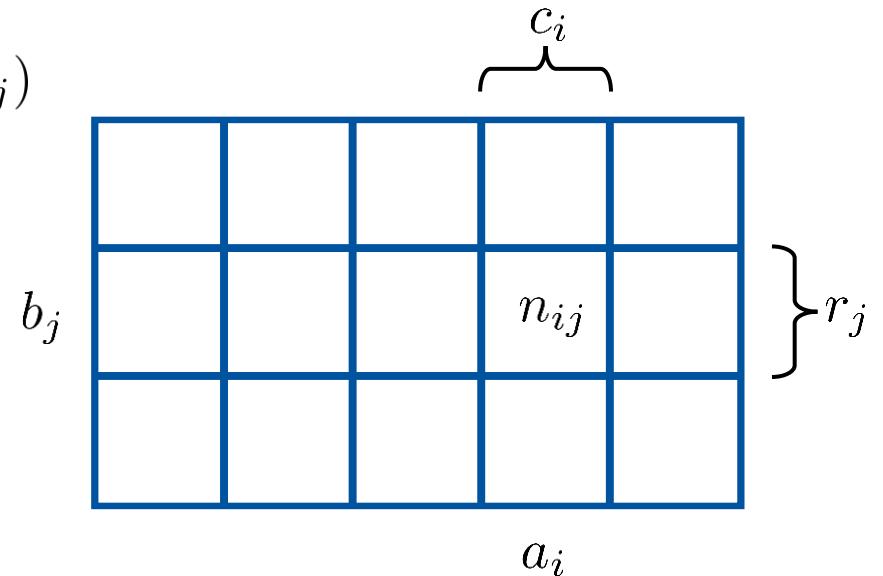


- **Sum rule:**

$$p(A = a_i) = \frac{c_i}{N} = \frac{1}{N} \sum_j n_{ij} = \sum_{b_j} p(A = a_i, B = b_j)$$

- **Product rule:**

$$\begin{aligned} p(A = a_i, B = b_j) &= \frac{n_{ij}}{N} = \frac{n_{ij}}{c_i} \cdot \frac{c_i}{N} \\ &= p(B = b_j | A = a_i) p(A = a_i) \end{aligned}$$



Rules of Probability - Summary

- Sum rule:

$$p(A) = \sum_B p(A, B)$$

- Product rule:

$$p(A, B) = p(B|A)p(A)$$

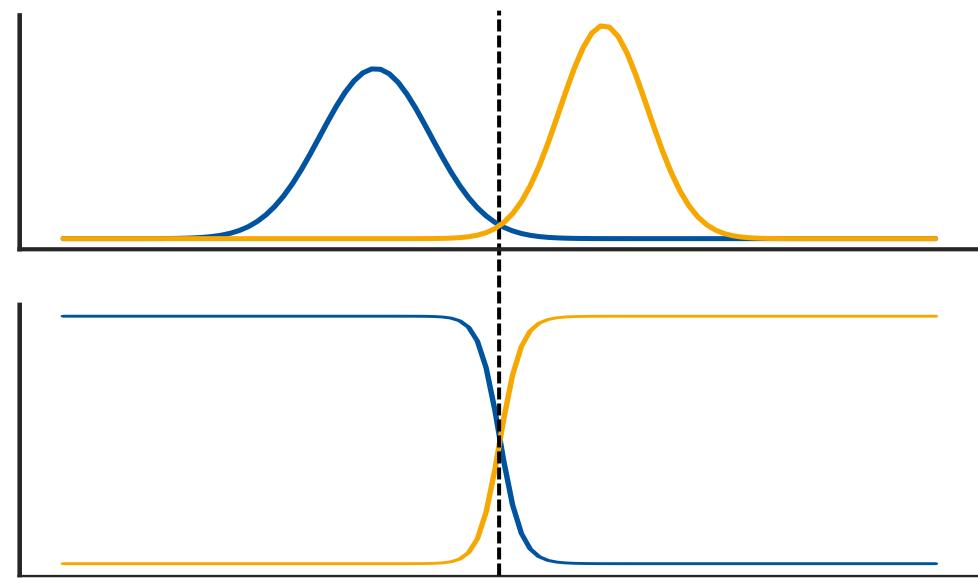
- Combine into Bayes' Theorem:

$$\begin{aligned} p(A|B) &= \frac{p(B|A)p(A)}{p(B)} \\ &= \frac{p(B|A)p(A)}{\sum_A p(B|A)p(A)} \end{aligned}$$

This is the most important equation in this course!

Introduction

1. Motivation
2. Forms of learning
3. Terms, Concepts, and Notation
4. **Bayes Decision Theory**



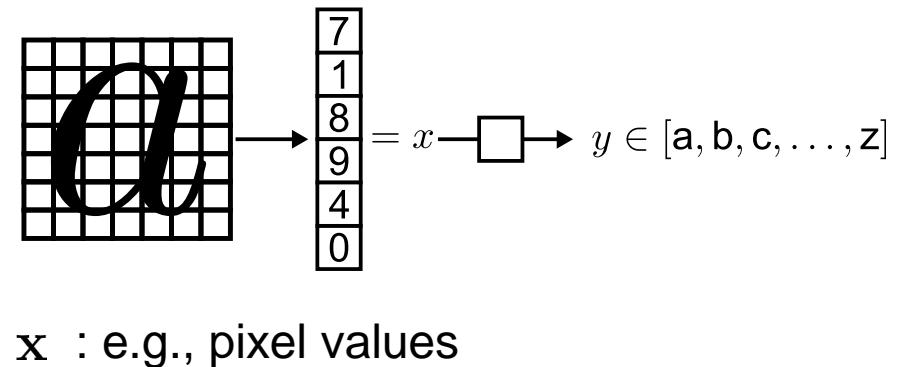
Bayes Decision Theory

- Goal: predict an output class \mathcal{C} from measurements x , by minimizing the probability of misclassification.
- *How can we make such decisions optimally?*
- Bayes Decision Theory gives us the tools for this
 - Based on Bayes' Theorem:

$$p(Y|X) = \frac{p(X|Y)p(Y)}{p(X)}$$

- In the following, we will introduce its basic concepts...

Example: handwritten character recognition



Core Concept: Priors

- What can we tell about the outcome of an experiment *before* making any measurements?
- The **a-priori probability** $p(\mathcal{C})$ captures the probability distribution over the different class outcomes
 - Based on previously observed data
 - i.e., independent of the actual measurement
- The prior probabilities over all possible class outcomes sum to one.

Example: in English text, the letter “e” makes up ~13% of all letters:

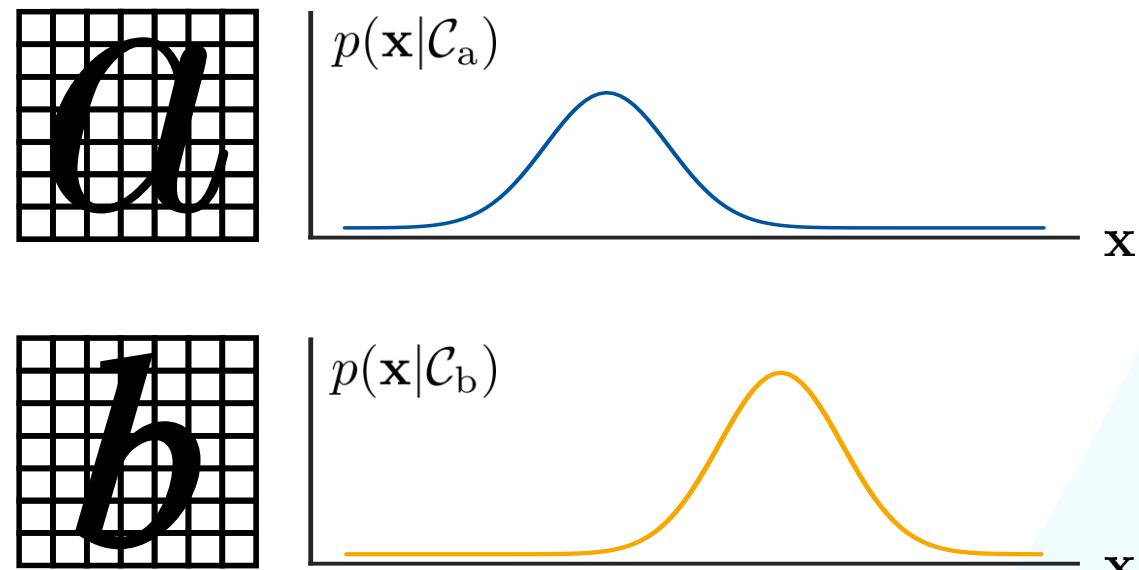
$$p(\mathcal{C}_e) = 0.13$$

And there are 26 letters in the English alphabet:

$$\sum_{\alpha \in \{a, \dots, z\}} p(\mathcal{C}_\alpha) = 1$$

Core Concept: Likelihood

- How *likely* is it that we *observe* a certain measurement \mathbf{x} *given* an example of class \mathcal{C} ?
- This is expressed by the **likelihood** $p(\mathbf{x}|\mathcal{C})$
 - It is called a *class-conditional distribution*, since it specifies the distribution of \mathbf{x} conditioned on the class \mathcal{C} .
 - We can estimate the likelihood from the distribution of measurements \mathbf{x} observed on the given training data.
- Here, \mathbf{x} measures certain properties of the input data.
 - E.g., the fraction of black pixels
 - We simply treat it as a vector $\mathbf{x} \in \mathbb{R}^D$.



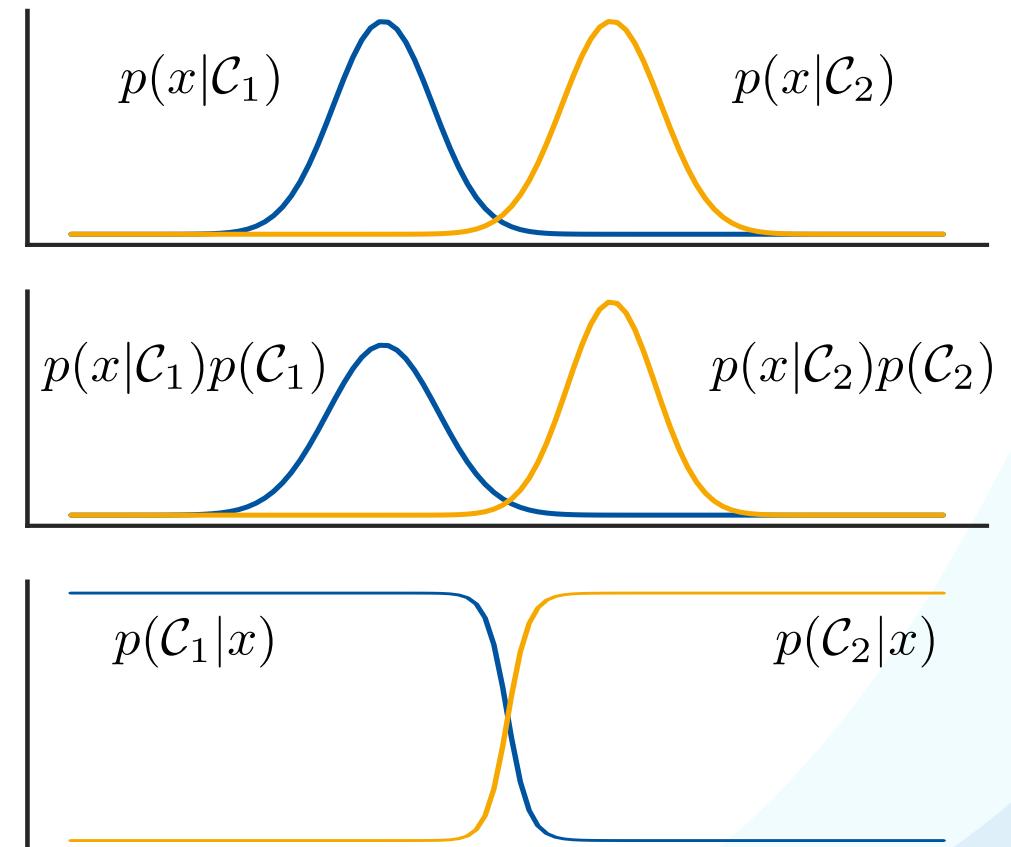
Core Concept: Posterior

- What is the probability for class \mathcal{C}_k if we made a measurement \mathbf{x} ?
- This **a-posteriori probability** $p(\mathcal{C}_k|\mathbf{x})$ can be computed via Bayes' Theorem after we observed \mathbf{x} :

$$p(\mathcal{C}_k|\mathbf{x}) = \frac{p(\mathbf{x}|\mathcal{C}_k)p(\mathcal{C}_k)}{p(\mathbf{x})} = \frac{p(\mathbf{x}|\mathcal{C}_k)p(\mathcal{C}_k)}{\sum_j p(\mathbf{x}|\mathcal{C}_j)p(\mathcal{C}_j)}$$

- *This is usually what we're interested in!*
- Interpretation

$$\text{posterior} = \frac{\text{likelihood} \cdot \text{prior}}{\text{normalization factor}}$$



Making Optimal Decisions

- Goal: minimize the probability of misclassification.

$$p(\text{mistake}) = p(x \in \mathcal{R}_1, \mathcal{C}_2) + p(x \in \mathcal{R}_2, \mathcal{C}_1)$$

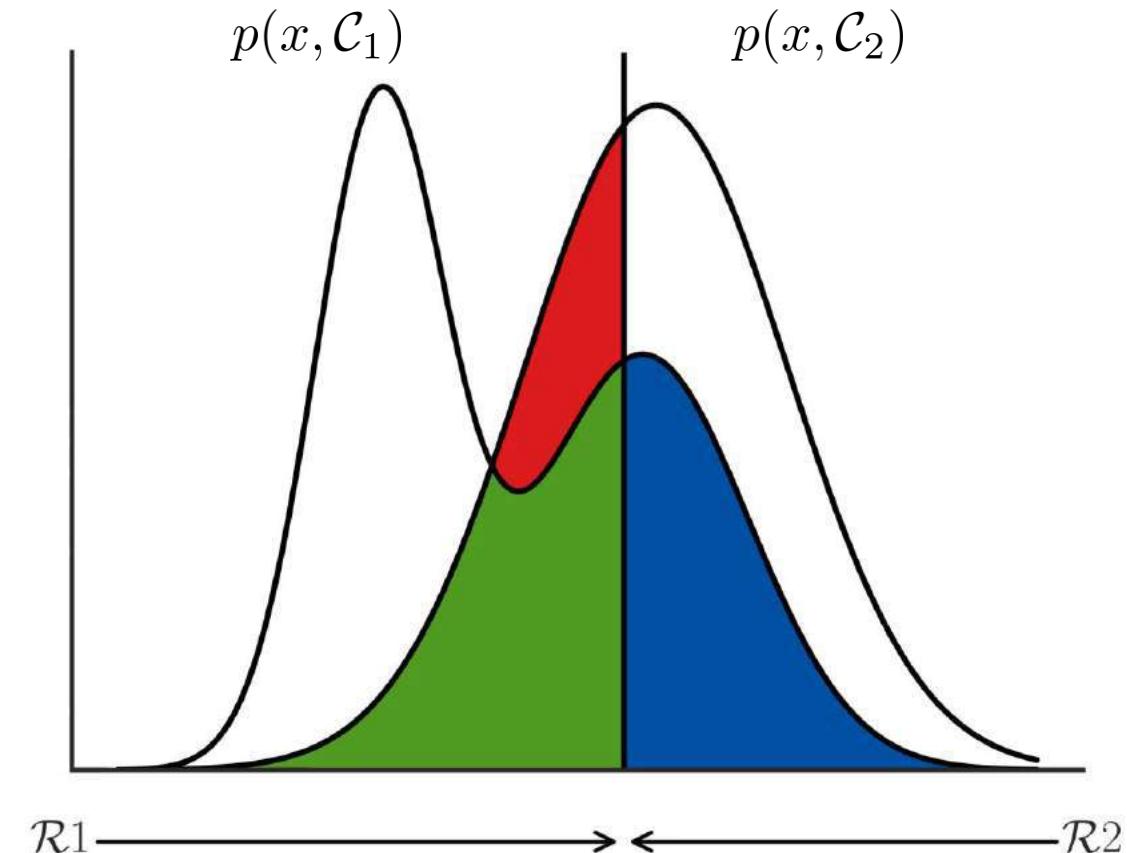
$$= \int_{\mathcal{R}_1} p(x, \mathcal{C}_2) dx + \int_{\mathcal{R}_2} p(x, \mathcal{C}_1) dx$$

$$= \int_{\mathcal{R}_1} p(\mathcal{C}_2|x)p(x) dx + \int_{\mathcal{R}_2} p(\mathcal{C}_1|x)p(x) dx$$

- Note:

$\boxed{\text{blue}} + \boxed{\text{green}} = \text{constant}$

We can only reduce



\mathcal{R}_1 and \mathcal{R}_2 are the **decision regions** after setting a decision threshold.

Making Optimal Decisions

- Goal: minimize the probability of misclassification.

$$p(\text{mistake}) = p(x \in \mathcal{R}_1, \mathcal{C}_2) + p(x \in \mathcal{R}_2, \mathcal{C}_1)$$

$$= \int_{\mathcal{R}_1} p(x, \mathcal{C}_2) dx + \int_{\mathcal{R}_2} p(x, \mathcal{C}_1) dx$$

$$= \int_{\mathcal{R}_1} p(\mathcal{C}_2|x)p(x) dx + \int_{\mathcal{R}_2} p(\mathcal{C}_1|x)p(x) dx$$

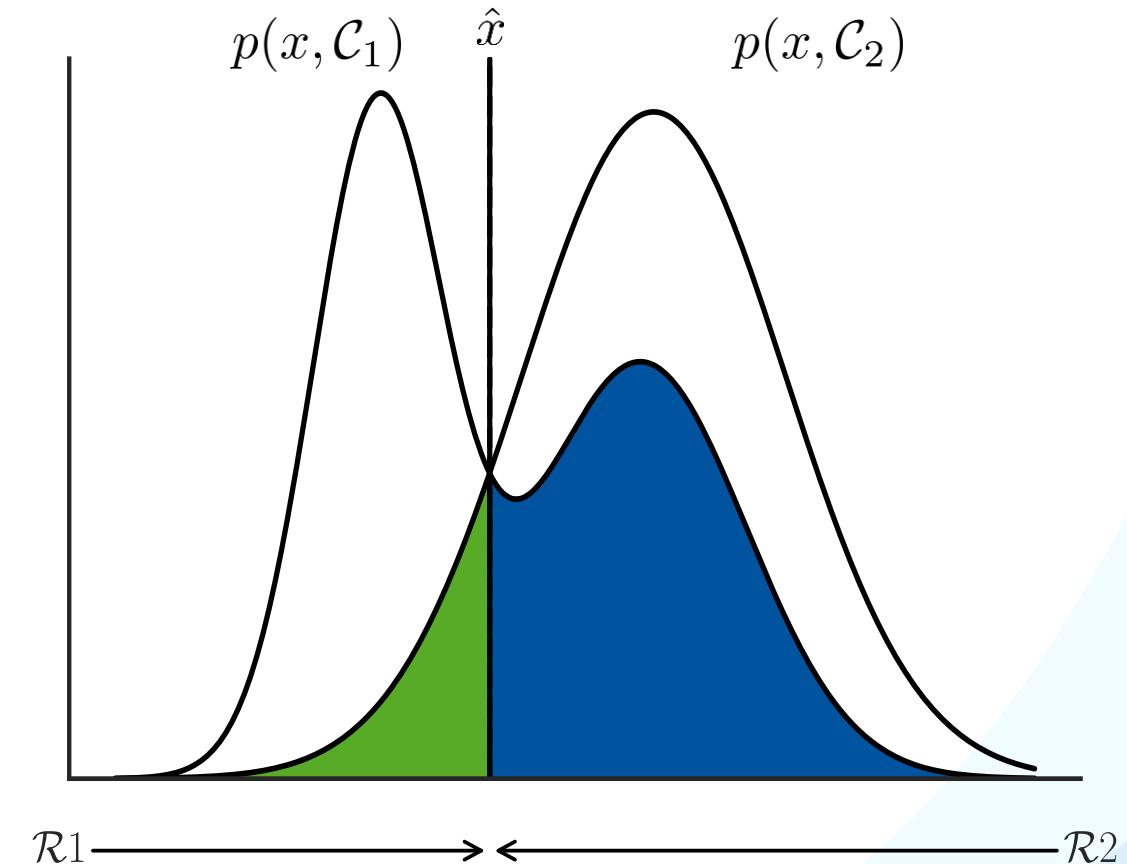
- Note:



$$+ = \text{constant}$$

We can only reduce 

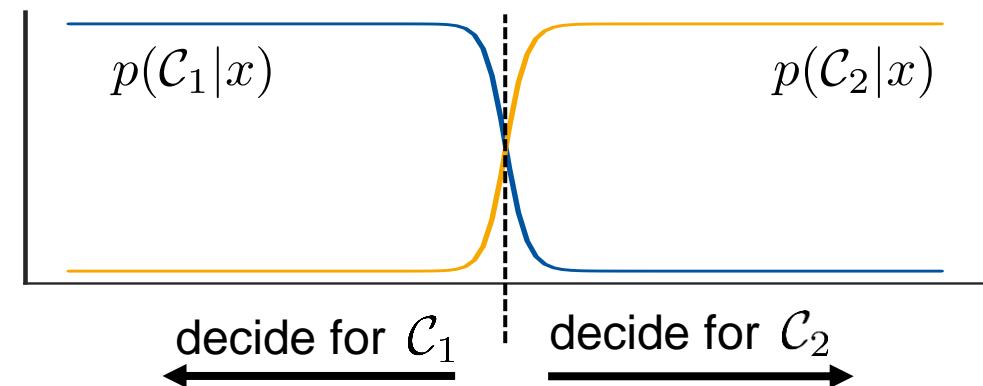
- Minimal error at the intersection \hat{x}*



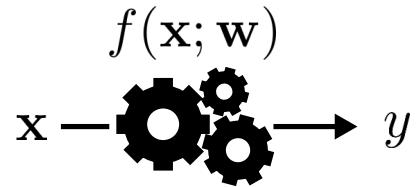
\mathcal{R}_1 and \mathcal{R}_2 are the **decision regions** after setting a decision threshold.

Making Optimal Decisions

- Our goal is to minimize the probability of a misclassification.
- The optimal decision rule is: decide for \mathcal{C}_1 iff $p(\mathcal{C}_1|\mathbf{x}) > p(\mathcal{C}_2|\mathbf{x})$
- Or for multiple classes: decide for \mathcal{C}_k iff $p(\mathcal{C}_k|\mathbf{x}) > p(\mathcal{C}_j|\mathbf{x}) \forall j \neq k$
- *Once we can estimate posterior probabilities, we can use this rule to build classifiers.*



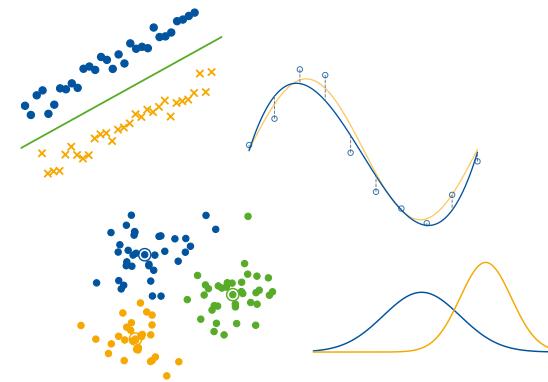
Summary: Introduction to ML



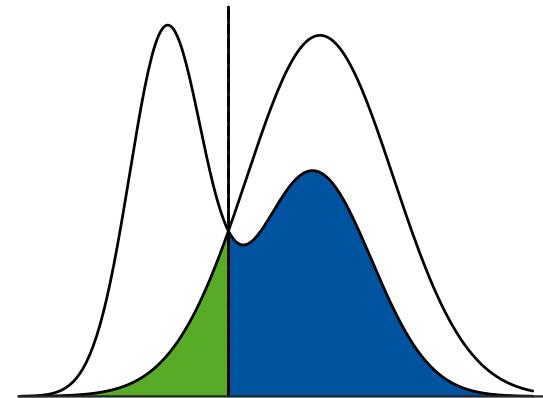
Machine Learning

$$p(\mathcal{C}|\mathbf{x}) = \frac{p(\mathbf{x}|\mathcal{C})p(\mathcal{C})}{p(\mathbf{x})}$$

Bayes Theorem



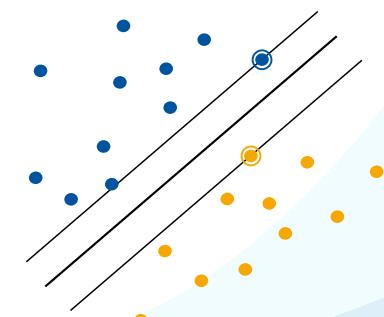
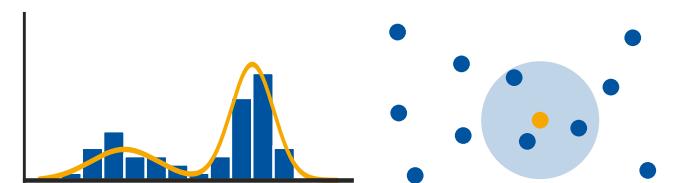
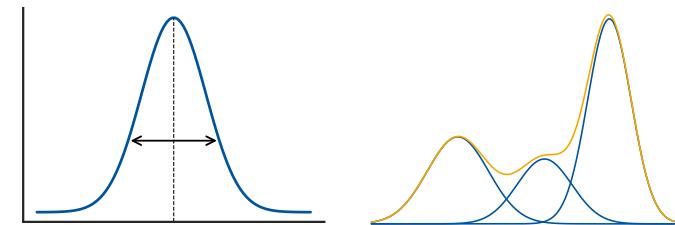
Forms of Machine Learning



Bayes Optimal Classification

Next Lectures...

- Ways how to estimate the probability densities $p(\mathbf{x}|\mathcal{C})$
 - Parametric methods
 - Gaussian distribution
 - Mixtures of Gaussians
 - Non-parametric methods
 - Histograms
 - k-Nearest Neighbor
 - Kernel Density Estimation
- Ways to directly model the posteriors $p(\mathcal{C}_k|\mathbf{x})$
 - Linear discriminants
 - Logistic regression, SVMs, Neural Networks, ...



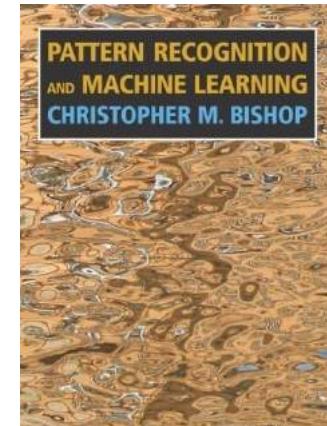
Machine Learning Topics

1. Introduction to ML
- 2. Probability Density Estimation**
3. Linear Discriminants
4. Linear Regression
5. Logistic Regression
6. Support Vector Machines
7. AdaBoost
8. Neural Network Basics

References and Further Reading

- More information, including a short review of Probability theory and a good introduction in Bayes Decision Theory can be found in Chapters 1.1, 1.2 and 1.5 of

Christopher M. Bishop
Pattern Recognition and Machine Learning
Springer, 2006



Elements of Machine Learning & Data Science

Winter semester 2023/24

Lecture 3 – Bayes Decision Theory

17.10.2023

Prof. Bastian Leibe

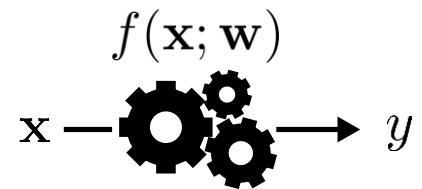
Announcement: Small-Group Exercises

Monday	Tuesday	Wednesday	Thursday	Friday
14:30-18:00h	3x 14:30-16:00h	2x 14:30-16:00h		
3x 16:30-18:00h	3x 16:30-18:00h	2x 16:30-18:00h		
18:30-20:00h	2x 18:30-20:00h			

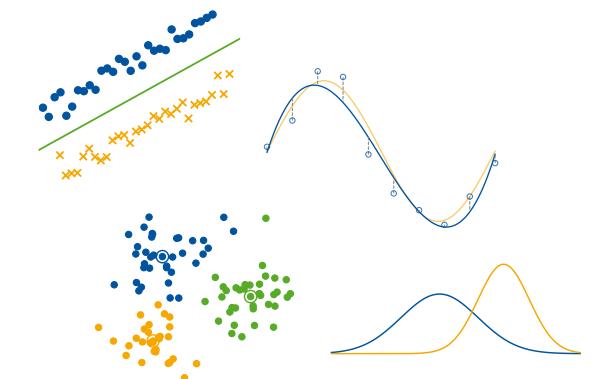
- **Bi-weekly small-group exercises**
 - We're currently setting up a poll to collect your preferences for the exercise slots
 - Please enter your choices until Wed, 18.10. evening!
 - Based on the poll results, we will assign you to exercise slots
 - *Please sign up for your time slot preferences by Wed evening...*

Machine Learning Topics

1. **Introduction to ML**
2. Probability Density Estimation
3. Linear Discriminants
4. Linear Regression
5. Logistic Regression
6. Support Vector Machines
7. AdaBoost
8. Neural Network Basics



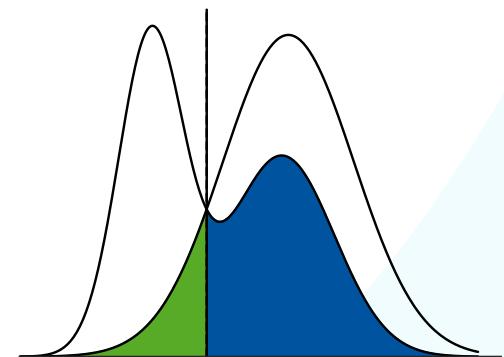
Machine Learning
Concepts



Forms of Machine Learning

$$p(\mathcal{C}|\mathbf{x}) = \frac{p(\mathbf{x}|\mathcal{C})p(\mathcal{C})}{p(\mathbf{x})}$$

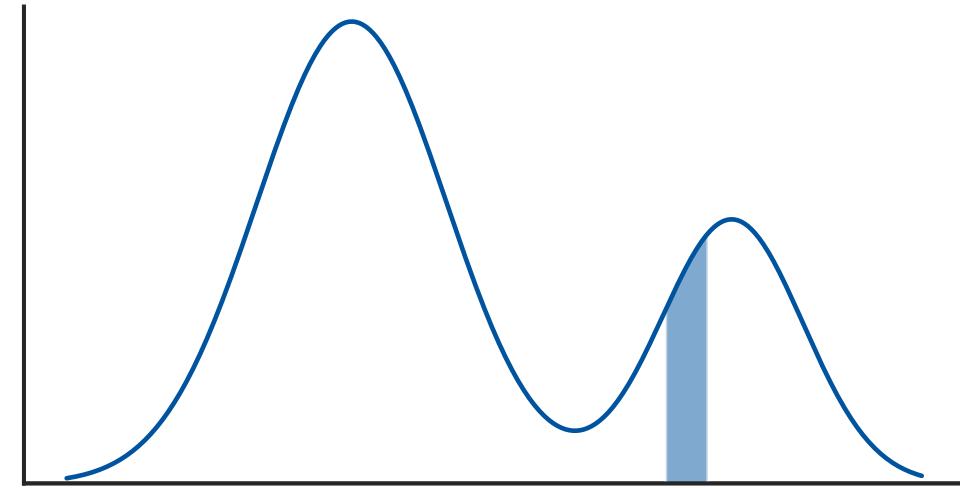
Bayes Decision Theory



Bayes Optimal
Classification

Introduction

1. Motivation
2. Forms of learning
3. **Terms, Concepts, and Notation**
4. Bayes Decision Theory



Rules of Probability - Summary

- Sum rule:

$$p(A) = \sum_B p(A, B)$$

- Product rule:

$$p(A, B) = p(B|A)p(A)$$

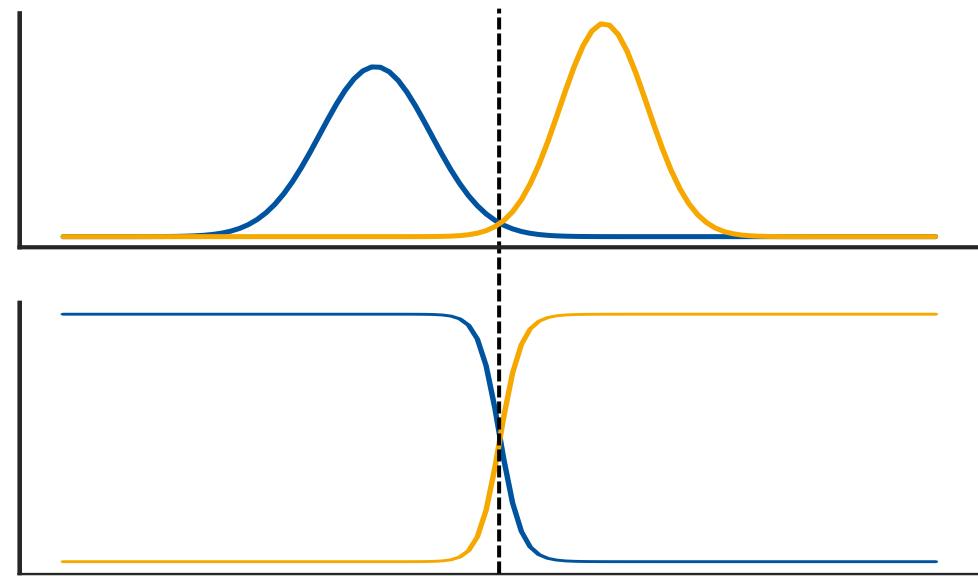
- Combine into Bayes' Theorem:

$$\begin{aligned} p(A|B) &= \frac{p(B|A)p(A)}{p(B)} \\ &= \frac{p(B|A)p(A)}{\sum_A p(B|A)p(A)} \end{aligned}$$

This is the most important equation in this course!

Introduction

1. Motivation
2. Forms of learning
3. Terms, Concepts, and Notation
4. **Bayes Decision Theory**

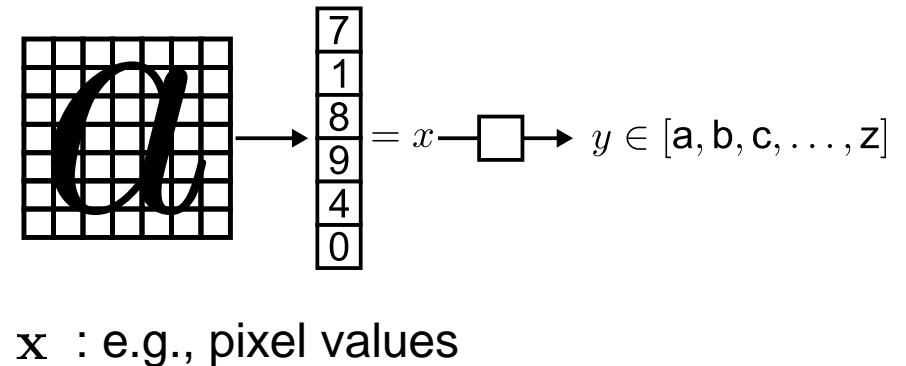


Bayes Decision Theory

- Goal: predict an output class \mathcal{C} from measurements x , by minimizing the probability of misclassification.
- *How can we make such decisions optimally?*
- Bayes Decision Theory gives us the tools for this
 - Based on Bayes' Theorem:

$$p(Y|X) = \frac{p(X|Y)p(Y)}{p(X)}$$

Example: handwritten character recognition



- In the following, we will introduce its basic concepts...

Core Concept: Priors

- What can we tell about the outcome of an experiment *before* making any measurements?
- The **a-priori probability** $p(\mathcal{C})$ captures the probability distribution over the different class outcomes
 - Based on previously observed data
 - i.e., independent of the actual measurement
- The prior probabilities over all possible class outcomes sum to one.

Example: in English text, the letter “e” makes up ~13% of all letters:

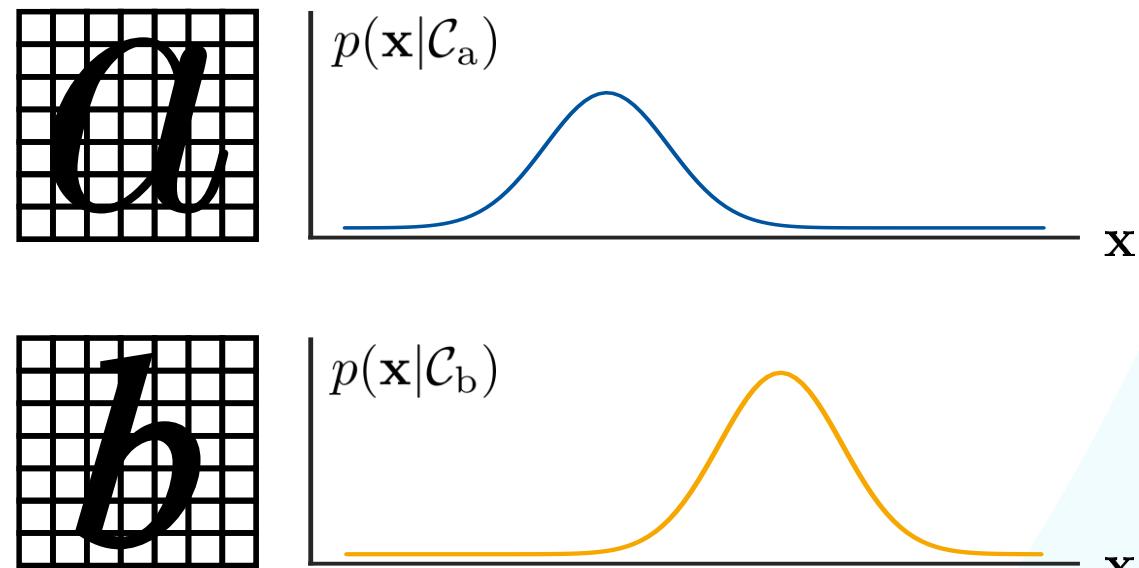
$$p(\mathcal{C}_e) = 0.13$$

And there are 26 letters in the English alphabet:

$$\sum_{\alpha \in \{a, \dots, z\}} p(\mathcal{C}_\alpha) = 1$$

Core Concept: Likelihood

- How *likely* is it that we *observe* a certain measurement \mathbf{x} *given* an example of class \mathcal{C} ?
- This is expressed by the **likelihood** $p(\mathbf{x}|\mathcal{C})$
 - It is called a *class-conditional distribution*, since it specifies the distribution of \mathbf{x} conditioned on the class \mathcal{C} .
 - We can estimate the likelihood from the distribution of measurements \mathbf{x} observed on the given training data.
- Here, \mathbf{x} measures certain properties of the input data.
 - E.g., the fraction of black pixels
 - We simply treat it as a vector $\mathbf{x} \in \mathbb{R}^D$.



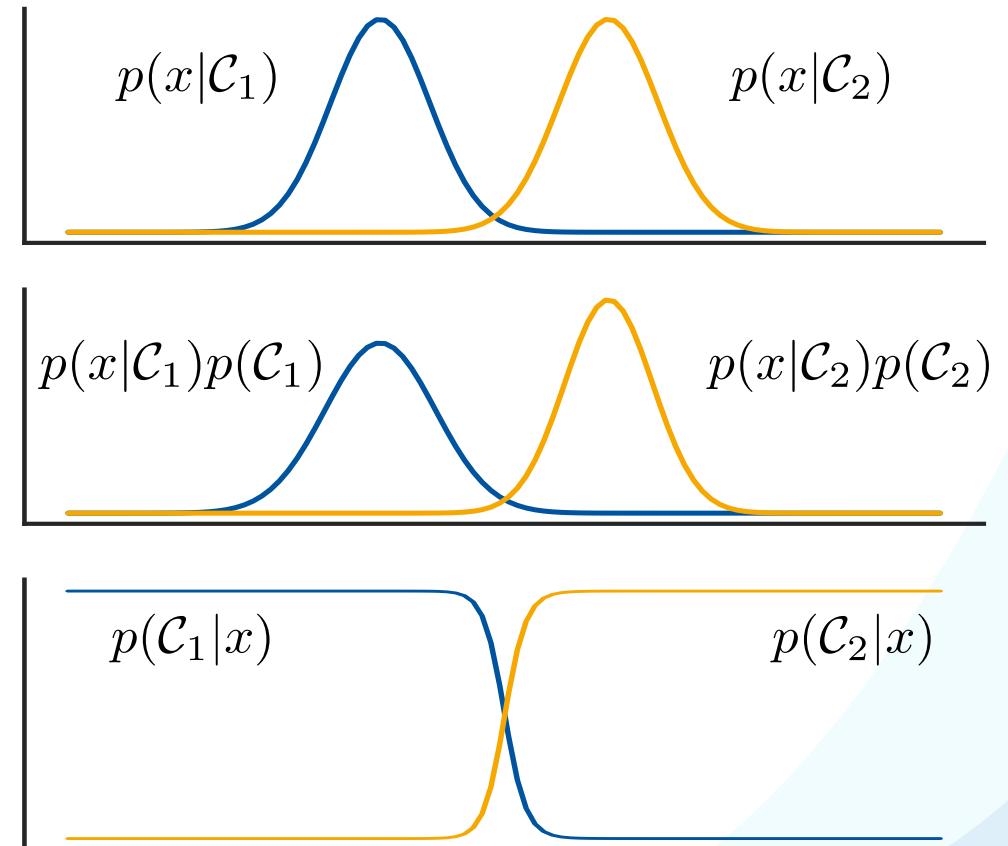
Core Concept: Posterior

- What is the probability for class \mathcal{C}_k if we made a measurement \mathbf{x} ?
- This **a-posteriori probability** $p(\mathcal{C}_k|\mathbf{x})$ can be computed via Bayes' Theorem after we observed \mathbf{x} :

$$p(\mathcal{C}_k|\mathbf{x}) = \frac{p(\mathbf{x}|\mathcal{C}_k)p(\mathcal{C}_k)}{p(\mathbf{x})} = \frac{p(\mathbf{x}|\mathcal{C}_k)p(\mathcal{C}_k)}{\sum_j p(\mathbf{x}|\mathcal{C}_j)p(\mathcal{C}_j)}$$

- *This is usually what we're interested in!*
- Interpretation

$$\text{posterior} = \frac{\text{likelihood} \cdot \text{prior}}{\text{normalization factor}}$$



Making Optimal Decisions

- Goal: minimize the probability of misclassification.

$$p(\text{mistake}) = p(x \in \mathcal{R}_1, \mathcal{C}_2) + p(x \in \mathcal{R}_2, \mathcal{C}_1)$$

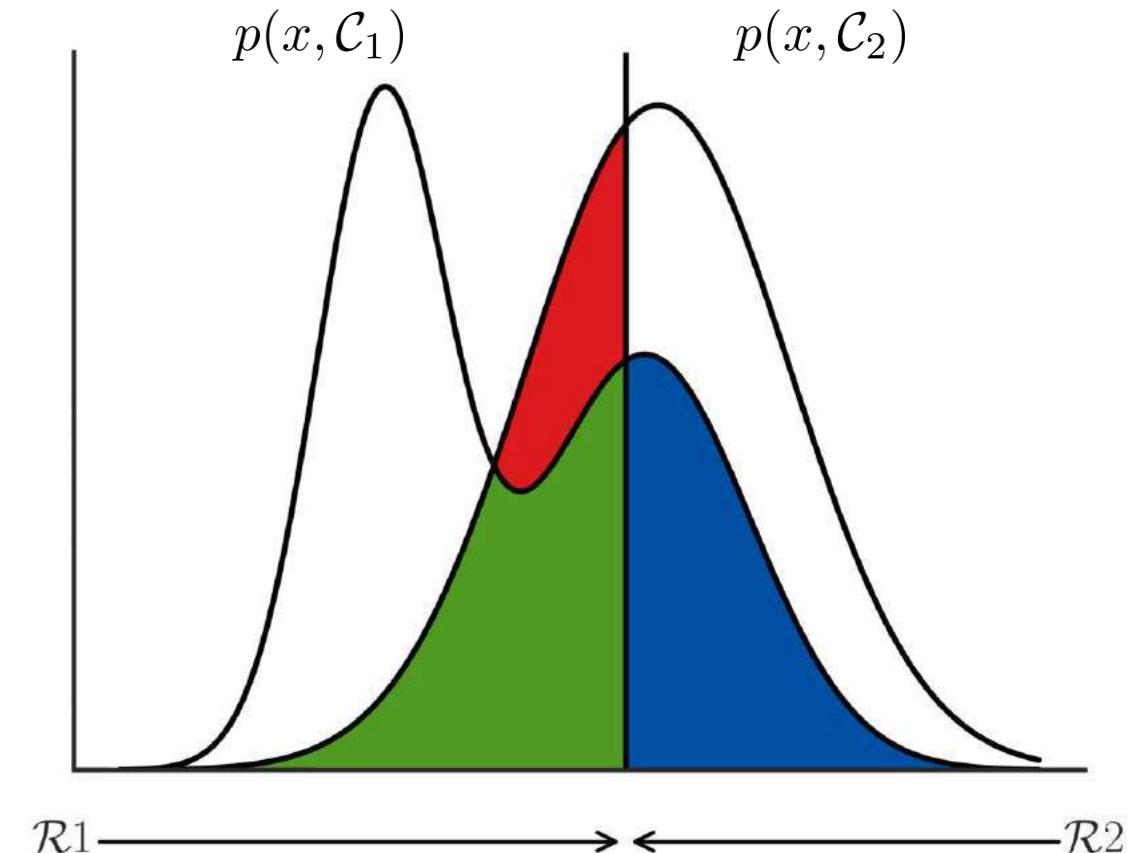
$$= \int_{\mathcal{R}_1} p(x, \mathcal{C}_2) dx + \int_{\mathcal{R}_2} p(x, \mathcal{C}_1) dx$$

$$= \int_{\mathcal{R}_1} p(\mathcal{C}_2|x)p(x) dx + \int_{\mathcal{R}_2} p(\mathcal{C}_1|x)p(x) dx$$

- Note:

+ = constant

We can only reduce



\mathcal{R}_1 and \mathcal{R}_2 are the **decision regions** after setting a decision threshold.

Making Optimal Decisions

- Goal: minimize the probability of misclassification.

$$p(\text{mistake}) = p(x \in \mathcal{R}_1, \mathcal{C}_2) + p(x \in \mathcal{R}_2, \mathcal{C}_1)$$

$$= \int_{\mathcal{R}_1} p(x, \mathcal{C}_2) dx + \int_{\mathcal{R}_2} p(x, \mathcal{C}_1) dx$$

$$= \int_{\mathcal{R}_1} p(\mathcal{C}_2|x)p(x) dx + \int_{\mathcal{R}_2} p(\mathcal{C}_1|x)p(x) dx$$

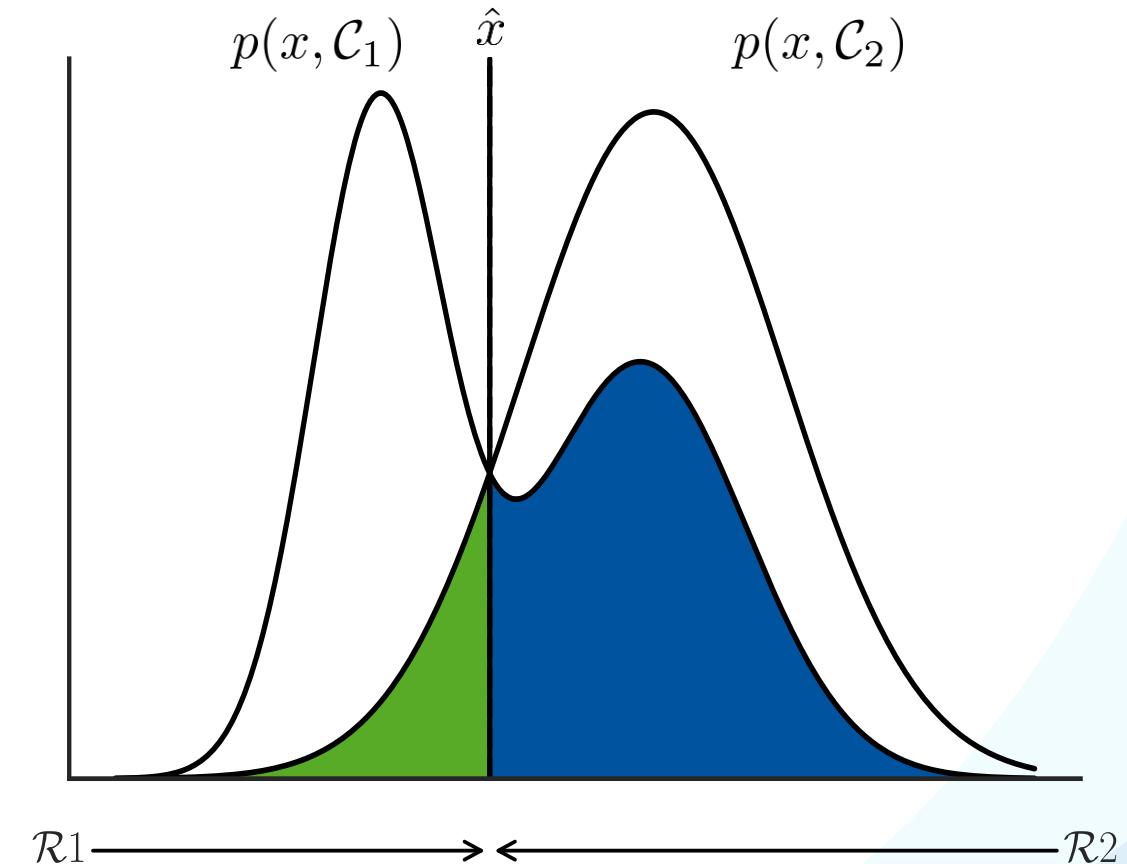
- Note:



$$+ = \text{constant}$$

We can only reduce 

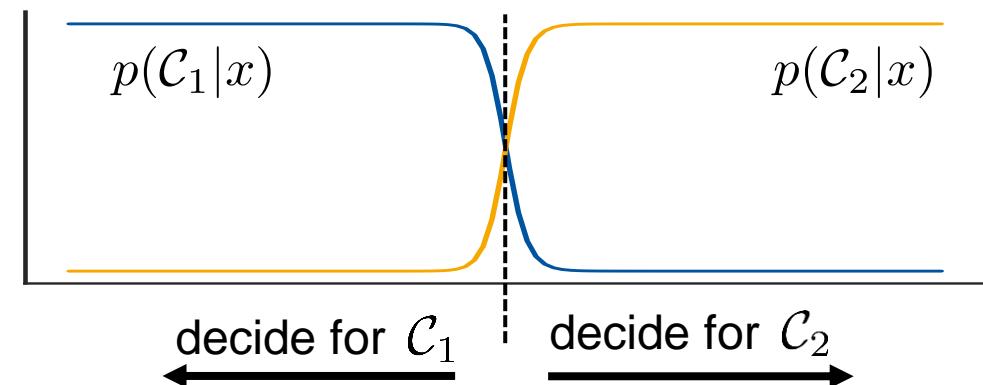
- Minimal error at the intersection \hat{x}*



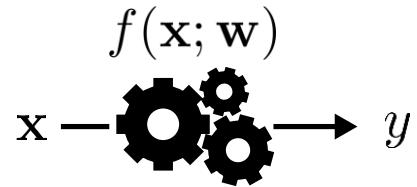
\mathcal{R}_1 and \mathcal{R}_2 are the **decision regions** after setting a decision threshold.

Making Optimal Decisions

- Our goal is to minimize the probability of a misclassification.
- The optimal decision rule is: decide for \mathcal{C}_1 iff $p(\mathcal{C}_1|\mathbf{x}) > p(\mathcal{C}_2|\mathbf{x})$
- Or for multiple classes: decide for \mathcal{C}_k iff $p(\mathcal{C}_k|\mathbf{x}) > p(\mathcal{C}_j|\mathbf{x}) \forall j \neq k$
- *Once we can estimate posterior probabilities, we can use this rule to build classifiers.*



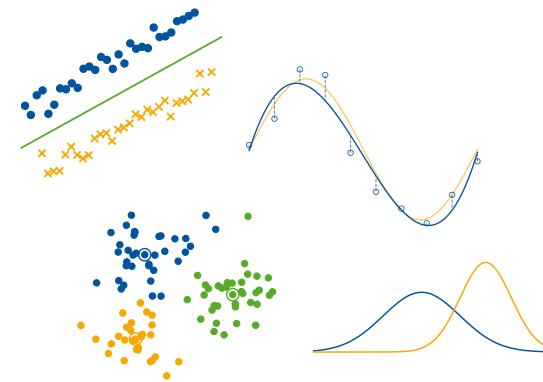
Summary: Introduction to ML



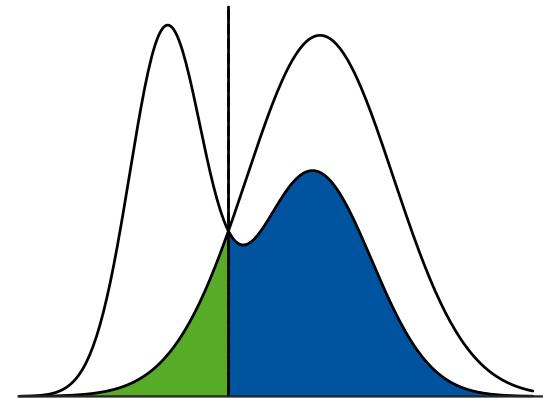
Machine Learning

$$p(\mathcal{C}|\mathbf{x}) = \frac{p(\mathbf{x}|\mathcal{C})p(\mathcal{C})}{p(\mathbf{x})}$$

Bayes Theorem



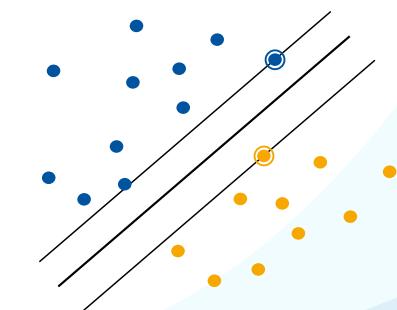
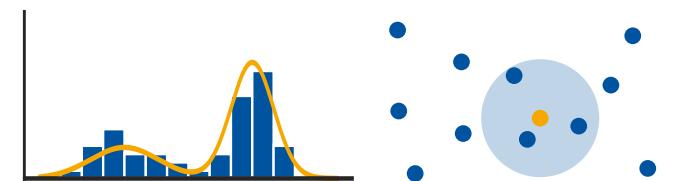
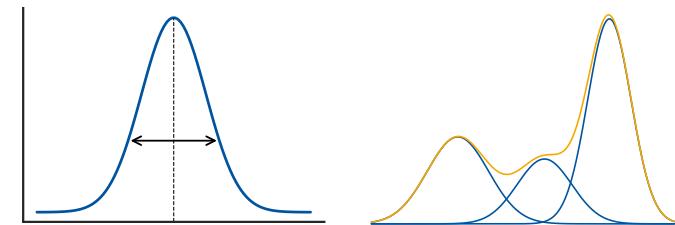
Forms of Machine Learning



Bayes Optimal Classification

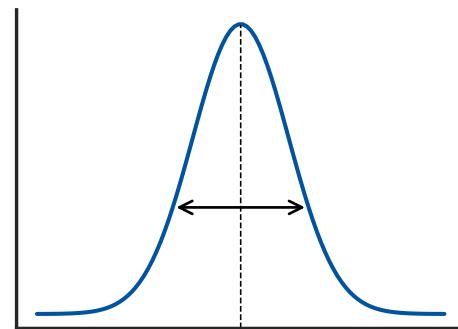
Next Lectures...

- Ways how to estimate the probability densities $p(\mathbf{x}|\mathcal{C}_k)$
 - **Parametric methods**
 - Gaussian distribution
 - Mixtures of Gaussians
 - **Non-parametric methods**
 - Histograms
 - k-Nearest Neighbor
 - Kernel Density Estimation
- Ways to directly model the posteriors $p(\mathcal{C}_k|\mathbf{x})$
 - Linear discriminants
 - Logistic regression, SVMs, Neural Networks, ...

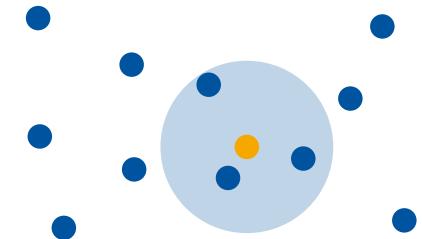


Machine Learning Topics

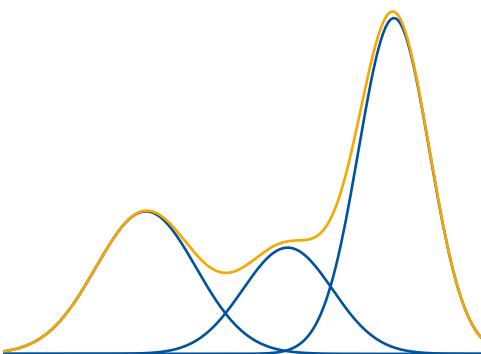
1. Introduction to ML
2. **Probability Density Estimation**
3. Linear Discriminants
4. Linear Regression
5. Logistic Regression
6. Support Vector Machines
7. AdaBoost
8. Neural Network Basics



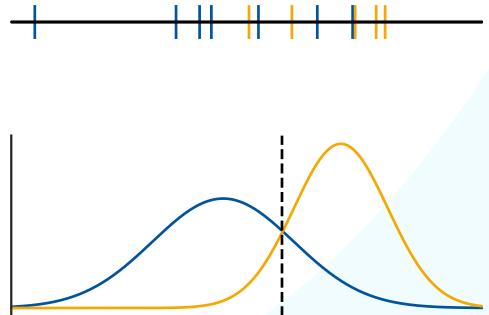
Parametric Methods
& ML-Algorithm



Nonparametric Methods



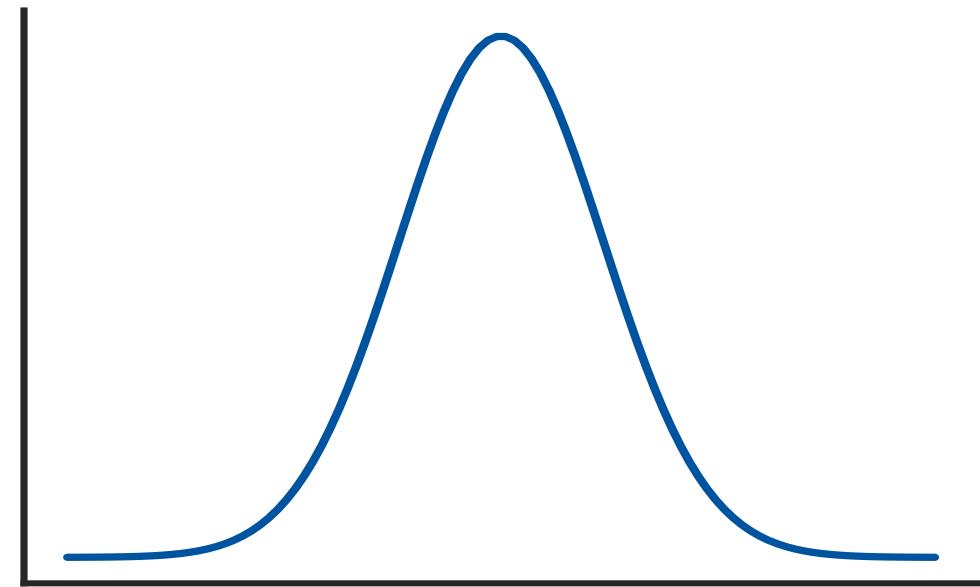
Mixtures of Gaussians
& EM-Algorithm



Bayes Classifiers

Probability Density Estimation

1. Probability Distributions
2. Parametric Methods
3. Nonparametric Methods
4. Mixture Models
5. Bayes Classifier
6. K-NN Classifier



Probability Distributions

- Up to now: Bayes optimal classification based on $p(\mathbf{x}|\mathcal{C}_k)$ and $p(\mathcal{C}_k)$.

- *How can we estimate (= learn) those probability densities?*

- Supervised training case: data and class labels are known.
 - Estimate the probability density for each class \mathcal{C}_k separately.

$$p(\mathbf{x}|\mathcal{C}_k) \quad \text{given } \mathbf{x} \in \mathcal{C}_k$$

- (For simplicity of notation, we will drop the class label \mathcal{C}_k in the following $\Rightarrow p(\mathbf{x})$).

- *First, we look at the Gaussian distribution in more detail...*

The Gaussian (or Normal) Distribution

- One-dimensional (univariate) case:

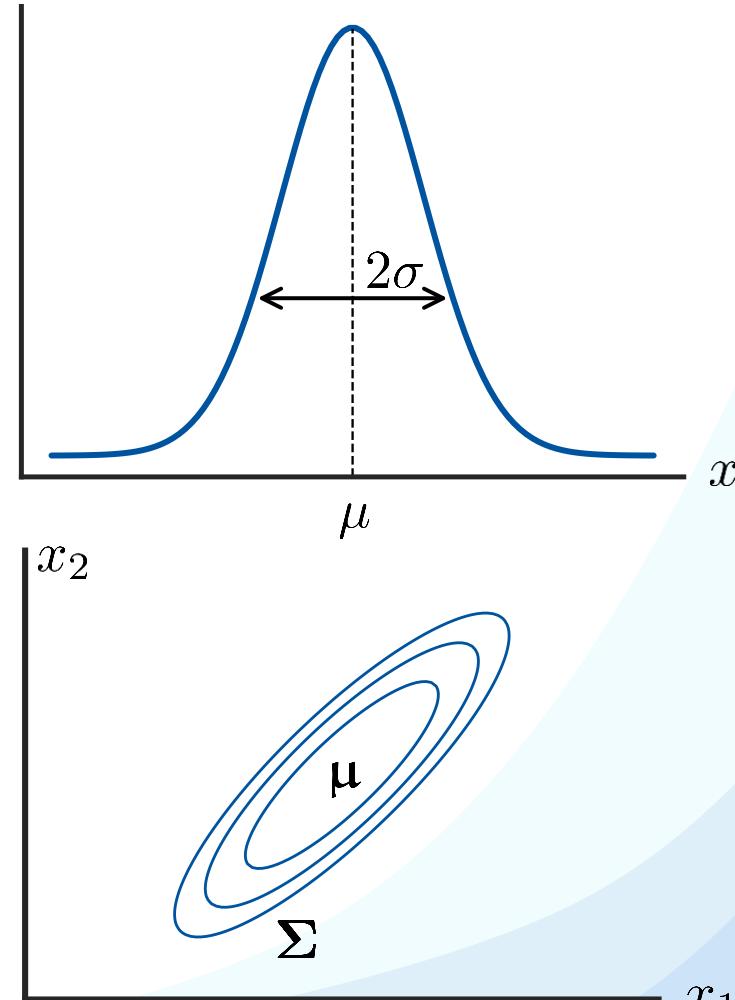
$$\mathcal{N}(x|\mu, \sigma^2) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$$

Mean Variance

- Multi-dimensional (multivariate) case:

$$\mathcal{N}(\mathbf{x}|\boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{(2\pi)^{D/2}|\boldsymbol{\Sigma}|^{1/2}} \exp\left(-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^\top \boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})\right)$$

Mean
vector Covariance
matrix

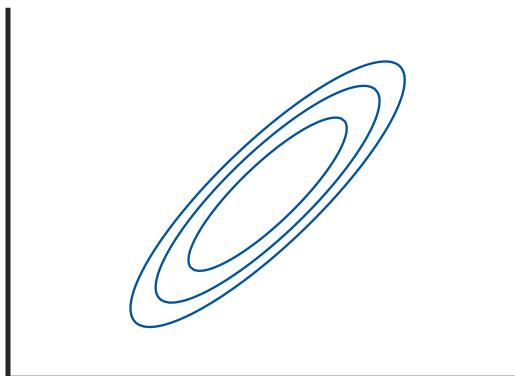


Gaussian Distribution: Shape

Full covariance matrix:

$$\Sigma = [\sigma_{ij}]$$

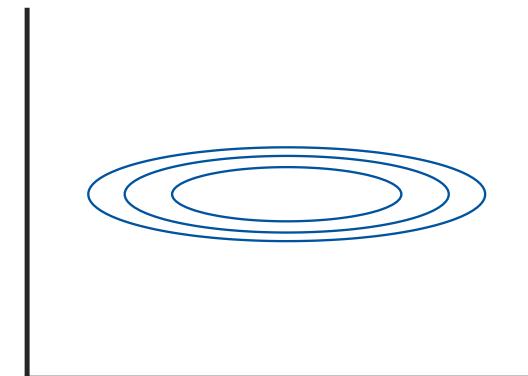
General ellipsoid shape



Diagonal covariance matrix:

$$\Sigma = \text{diag}\{\sigma_i\}$$

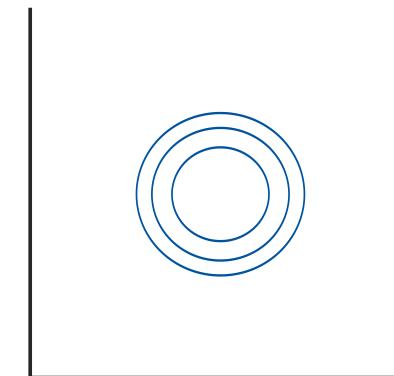
Axis-aligned ellipsoid



Uniform variance:

$$\Sigma = \sigma^2 \mathbf{I}$$

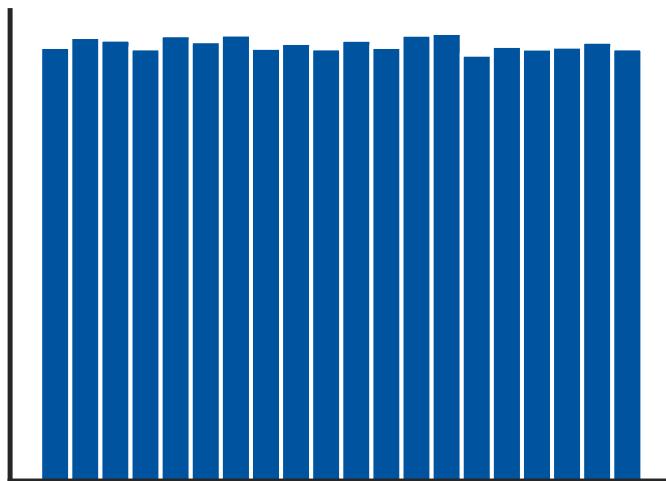
Hypersphere



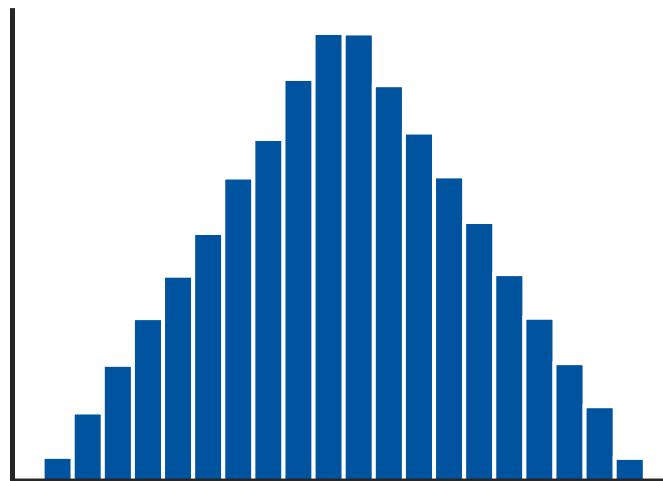
Gaussian Distribution: Motivation

- Central Limit Theorem
 - The distribution of a sum of N *i.i.d.* random variables becomes increasingly Gaussian as N grows.
 - In practice, the convergence to a Gaussian can be very rapid.
 - This makes the Gaussian interesting for many applications.
- Example: Sum over N uniform $[0,1]$ random variables.

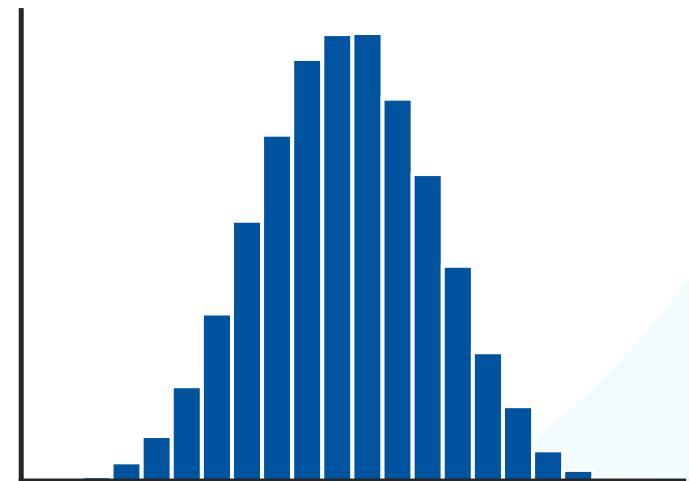
i.i.d. = independent and identically distributed



$N = 1$



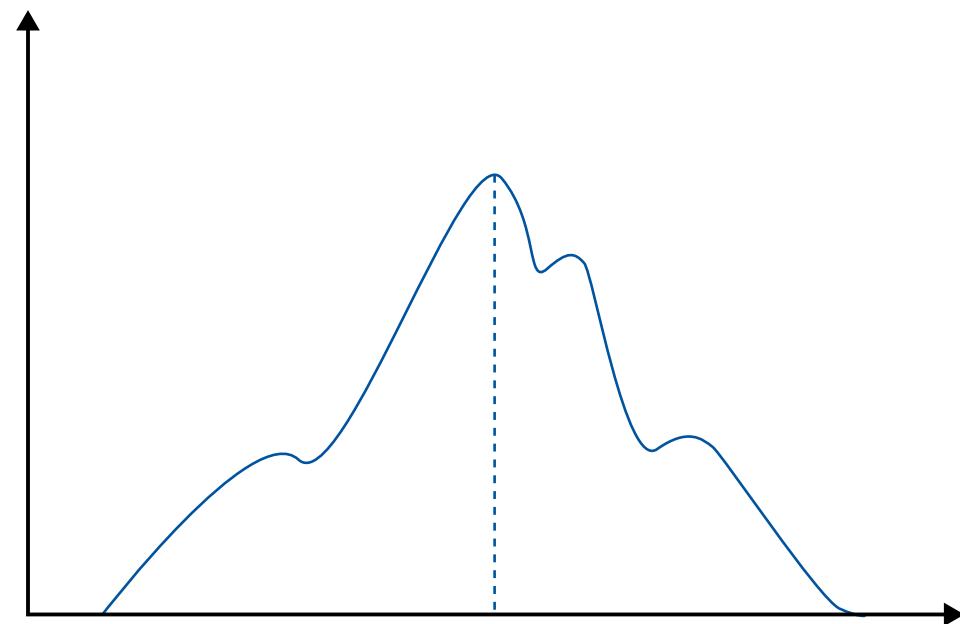
$N = 2$



$N = 5$

Probability Density Estimation

1. Probability Distributions
2. **Parametric Methods**
3. Nonparametric Methods
4. Mixture Models
5. Bayes Classifier
6. K-NN Classifier

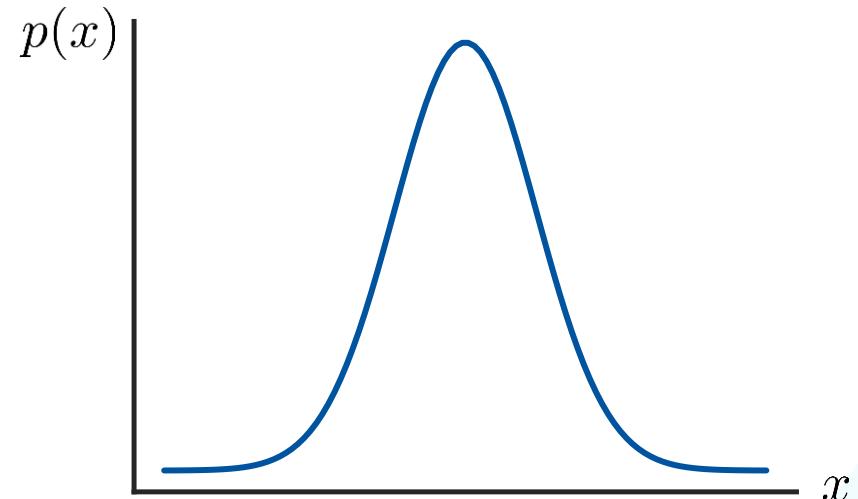


Parametric Methods

- In **parametric methods**, we assume that we know the parametric form of the underlying data distribution.
 - I.e., the equation of the pdf with parameters θ .

Example: $p(x) = \mathcal{N}(x|\mu, \sigma)$

$$\theta = (\mu, \sigma)$$



Parametric Methods

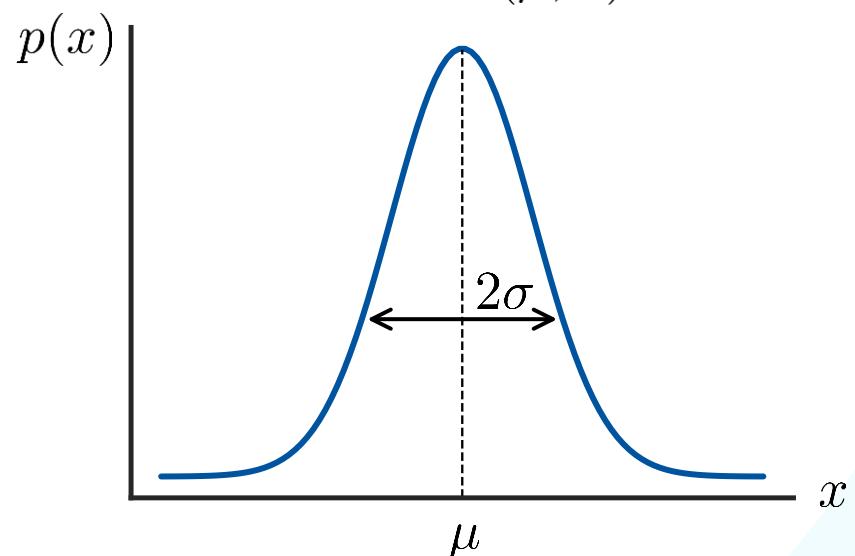
- In **parametric methods**, we assume that we know the parametric form of the underlying data distribution.
 - I.e., the equation of the pdf with parameters θ .
- Goal: Estimate θ from training data $\mathcal{X} = \{\mathbf{x}_1, \dots, \mathbf{x}_N\}$.
- **Likelihood** of θ :

$$L(\theta) = p(\mathcal{X}|\theta)$$

Probability that the data \mathcal{X} was indeed generated by a distribution with parameters θ .

Example: $p(x) = \mathcal{N}(x|\mu, \sigma)$

$$\theta = (\mu, \sigma)$$



Maximum Likelihood Approach

- Idea: Find optimal parameters by maximizing $L(\theta)$.
- Computation of the likelihood:

- Single data point (e.g., for Gaussian):

$$p(x_n|\theta) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(x_n - \mu)^2}{2\sigma^2}\right)$$

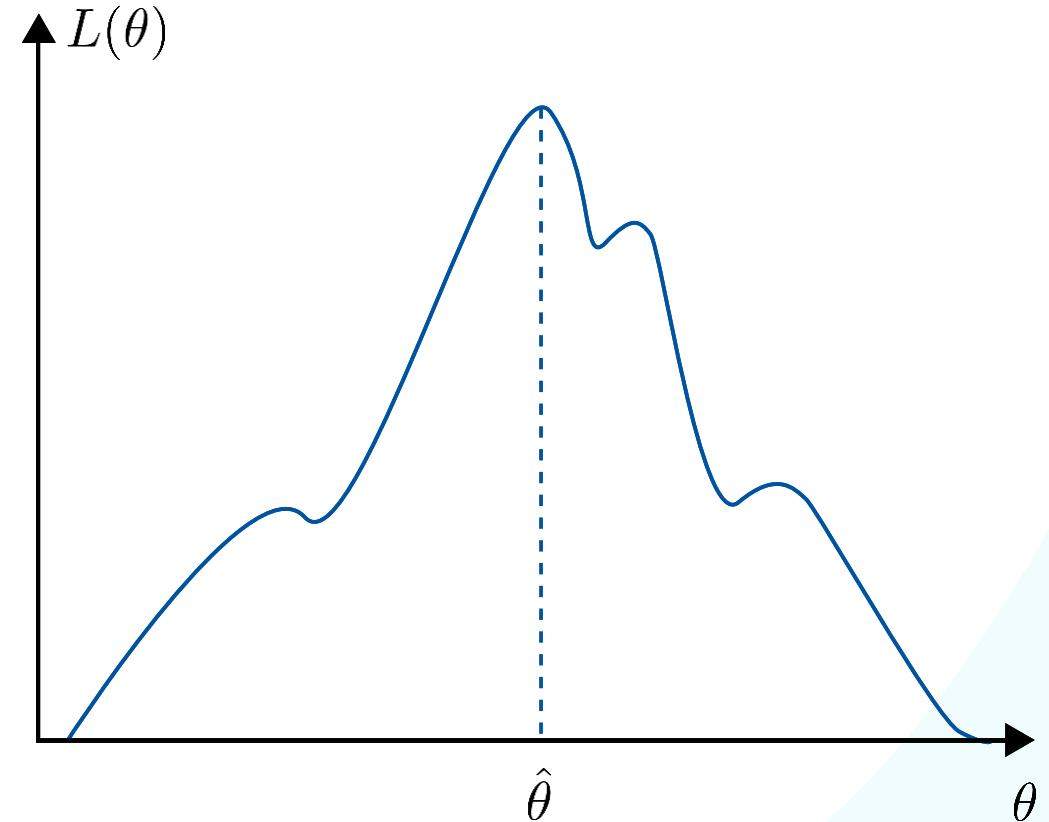
- Assumption: all data points are independent

$$L(\theta) = p(\mathcal{X}|\theta) = \prod_{n=1}^N p(x_n|\theta)$$

- Negative Log-Likelihood (“Energy”):

$$E(\theta) = -\ln L(\theta) = -\sum_{n=1}^N \ln p(x_n|\theta)$$

Maximizing the likelihood \Leftrightarrow minimizing the negative log-likelihood.



- Minimizing the negative log-likelihood:
 - Take the derivative and set it to zero.

$$\frac{\partial}{\partial \theta} E(\theta) = -\frac{\partial}{\partial \theta} \sum_{n=1}^N \ln p(x_n|\theta) = -\sum_{n=1}^N \frac{\frac{\partial}{\partial \theta} p(x_n|\theta)}{p(x_n|\theta)} \stackrel{!}{=} 0$$

- Log-likelihood for Normal distribution (1D case):

$$\begin{aligned}\frac{\partial}{\partial \mu} E(\mu, \sigma) &= -\sum_{n=1}^N \frac{\frac{\partial}{\partial \mu} p(x_n|\mu, \sigma)}{p(x_n|\mu, \sigma)} \\ &= -\sum_{n=1}^N -\frac{2(x_n - \mu)}{2\sigma^2} \frac{p(x_n|\mu, \sigma)}{p(x_n|\mu, \sigma)}\end{aligned}$$

$$p(x_n|\mu, \sigma) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(x_n - \mu)^2}{2\sigma^2}\right)$$

$$\frac{\partial}{\partial \mu} p(x_n|\mu, \sigma) = -\frac{2(x_n - \mu)}{2\sigma^2} p(x_n|\mu, \sigma)$$

$$= \frac{1}{\sigma^2} \sum_{n=1}^N (x_n - \mu) = \cancel{\frac{1}{\sigma^2}} \left(\sum_{n=1}^N x_n - N\mu \right) \stackrel{!}{=} 0 \iff \hat{\mu} = \frac{1}{N} \sum_{n=1}^N x_n$$

- By minimizing the negative log-likelihood, we found: Similarly, we can derive:

$$\hat{\mu} = \frac{1}{N} \sum_{n=1}^N x_n$$

sample mean

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

sample variance

- $\hat{\theta} = (\hat{\mu}, \hat{\sigma})$ is the Maximum Likelihood estimate for the parameters of a Gaussian distribution.
 - This is a very important result.
 - Unfortunately, it is wrong...

- To be precise, the result is not wrong, but **biased**.
- Assume the samples x_1, x_2, \dots, x_N come from a true Gaussian distribution with mean μ and variance σ^2
 - It can be shown that the expected estimates are then

$$\mathbb{E}[\mu_{\text{ML}}] = \mu$$

$$\mathbb{E}[\sigma_{\text{ML}}^2] = \left(\frac{N-1}{N}\right)\sigma^2$$

\Rightarrow *The ML estimate will underestimate the true variance!*

- We can correct for this bias:

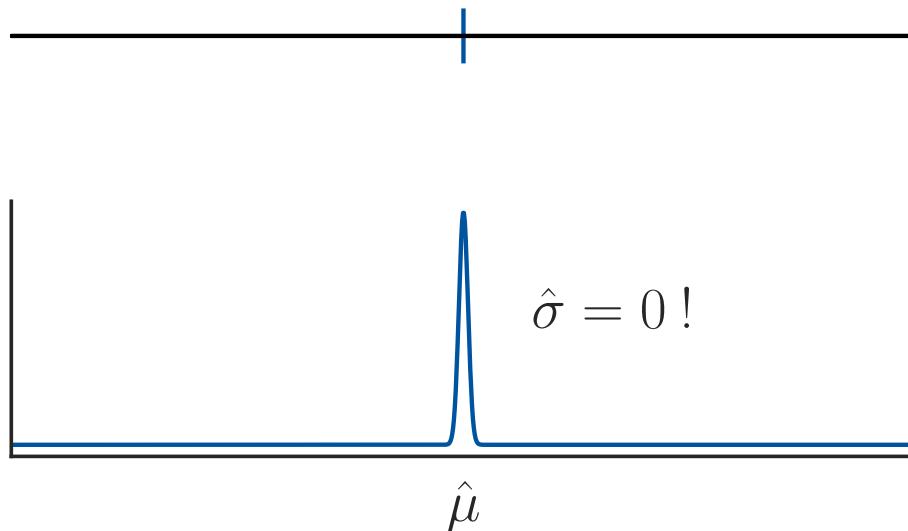
$$\hat{\sigma}^2 = \frac{N}{N-1}\sigma_{\text{ML}}^2 = \frac{1}{N-1} \sum_{n=1}^N (x_n - \hat{\mu})^2$$

- Maximum Likelihood has several significant limitations.
 - It systematically underestimates the variance of the distribution!
 - E.g., consider the estimate for a single sample:

$$N = 1, \mathcal{X} = \{x_1\}$$

$$\hat{\mu} = \frac{1}{N} \sum_{n=1}^N x_n = x_1$$

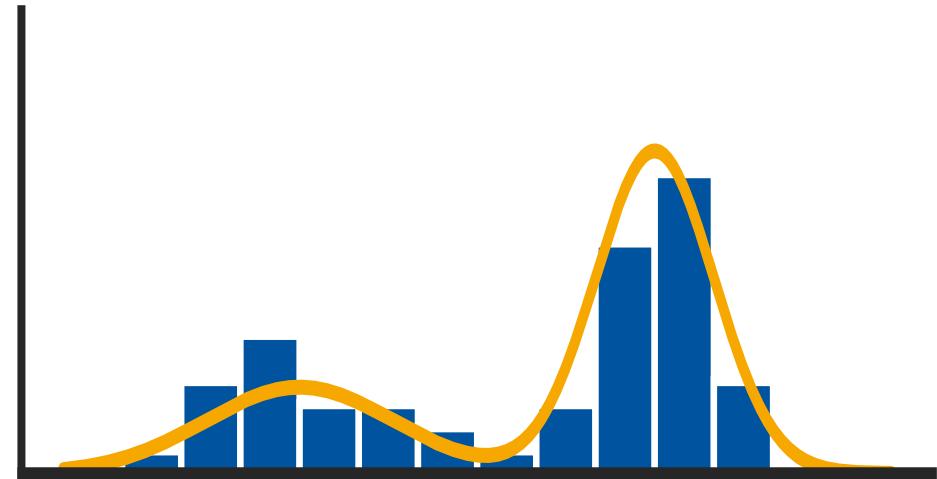
$$\hat{\sigma}^2 = \frac{1}{N} \sum_{n=1}^N (x_n - \hat{\mu})^2 = 0$$



- We say ML overfits to the observed data.
- *We will still often use Maximum Likelihood, but it is important to know about this effect.*

Probability Density Estimation

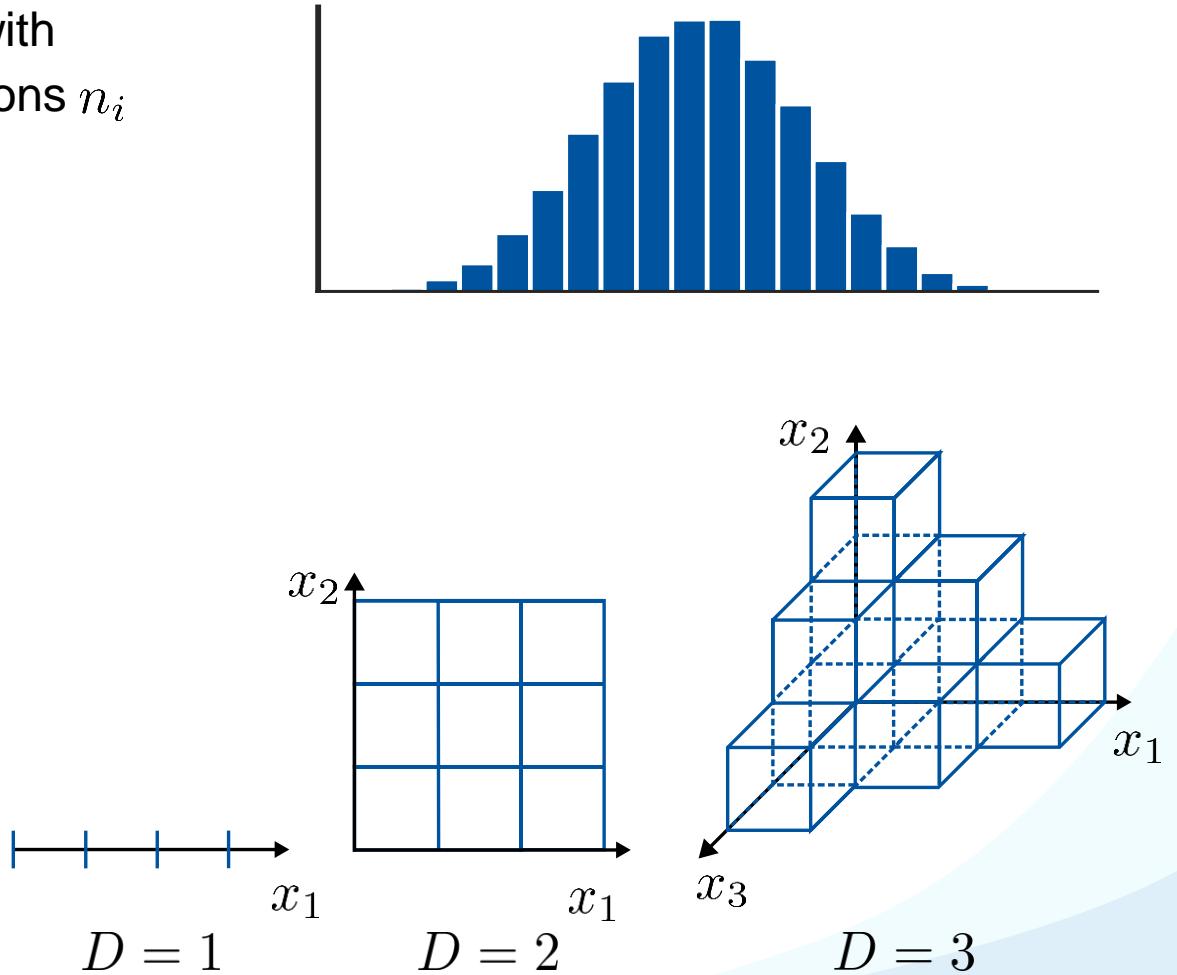
1. Probability Distributions
2. Parametric Methods
- 3. Nonparametric Methods**
 - a) Histograms
 - b) Kernel Methods & k-Nearest Neighbors
4. Mixture Models
5. Bayes Classifier
6. K-NN Classifier



Histograms

- Partition the data space into N distinct bins with widths Δ_i and count the number of observations n_i in each bin.
- Then, $p_i = \frac{n_i}{N\Delta_i}$.
- Often the same width is used for all bins.
- This can be done, in principle, for any dimensionality D .

...but the required number of bins grows exponentially with D !



The bin width Δ acts as a smoothing factor.

Not smooth enough



About ok



Too smooth



Advantages

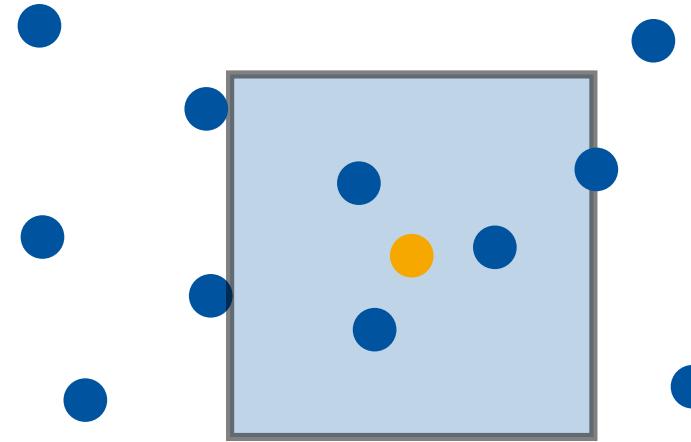
- Very general method. In the limit ($N \rightarrow \infty$), every probability density can be represented.
- No need to store the data points once histogram is computed.

Limitations

- Rather brute-force.
- Discontinuities at bin edges.
- Choosing right bin size is hard.
- Unsuitable for high-dimensional feature spaces.

Probability Density Estimation

1. Probability Distributions
2. Parametric Methods
3. **Nonparametric Methods**
 - a) Histograms
 - b) **Kernel Methods & k-Nearest Neighbors**
4. Mixture Models
5. Bayes Classifier
6. K-NN Classifier



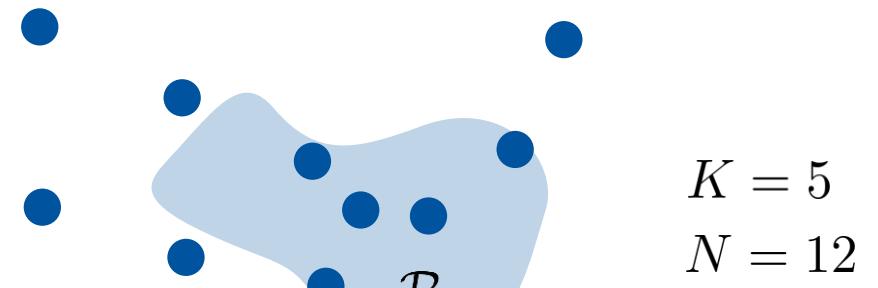
Kernel Methods and k-Nearest Neighbors

- Data point \mathbf{x} comes from pdf $p(\mathbf{x})$.
 - Probability that \mathbf{x} falls into small region \mathcal{R} :

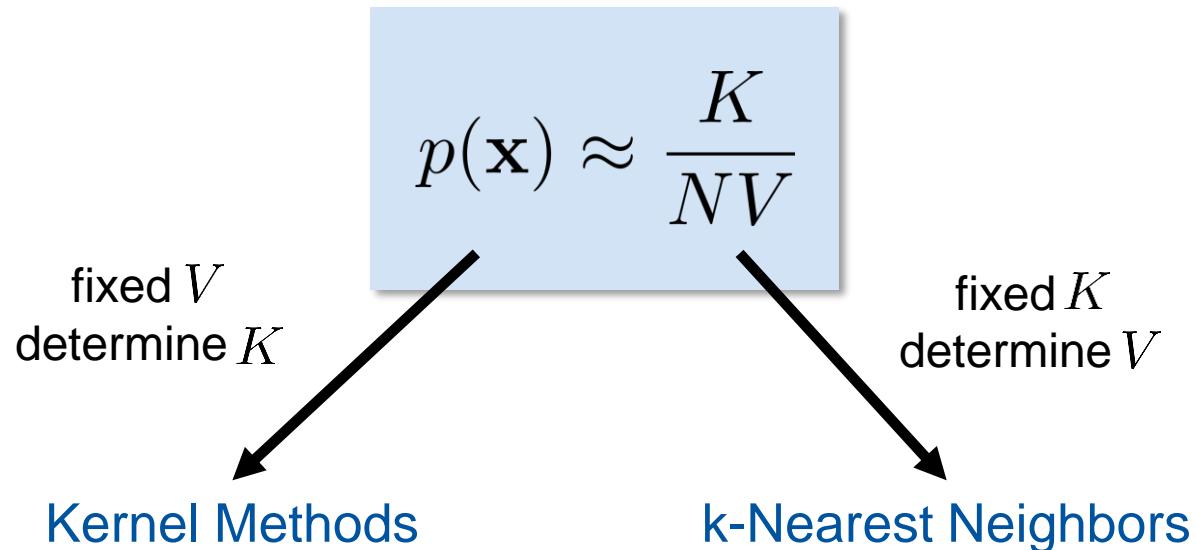
$$P = \int_{\mathcal{R}} p(y) dy \approx p(\mathbf{x})V$$

- Estimate $p(\mathbf{x})$ from samples
 - Let K be the number of samples that fall into \mathcal{R} .
 - If the number of samples N is sufficiently large, we can estimate P as:

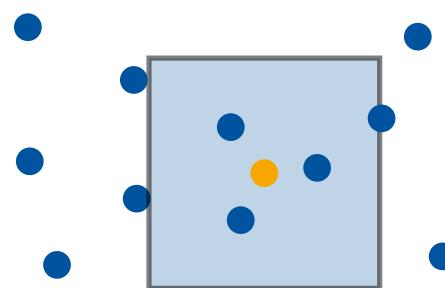
$$P = \frac{K}{N} \Rightarrow p(\mathbf{x}) \approx \frac{K}{NV}$$



For sufficiently small \mathcal{R} ,
 $p(\mathbf{x})$ is roughly constant.
 V : volume of \mathcal{R} .



Example: Determine the number K of data points inside a fixed hypercube



Kernel Methods

- Hypercube of dimension D with edge length h :

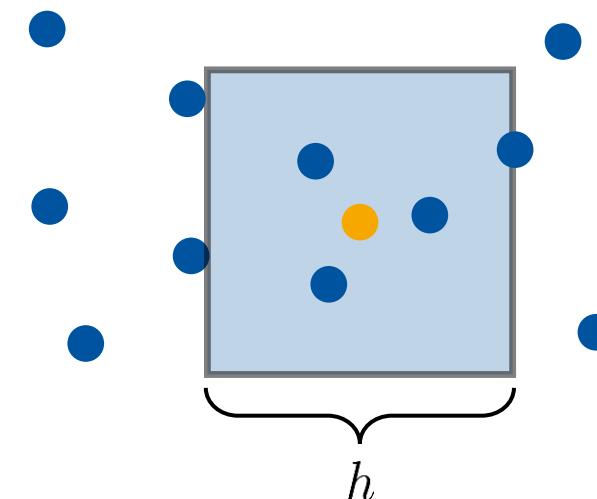
$$k(\mathbf{u}) = \begin{cases} 1, & \text{if } |u_i| \leq \frac{1}{2}h, \ i = 1, \dots, D \\ 0, & \text{otherwise} \end{cases}$$

$$K = \sum_{n=1}^N k(\mathbf{x} - \mathbf{x}_n) \quad V = \int k(\mathbf{u}) d\mathbf{u} = h^D$$

- Probability density estimate:

$$p(\mathbf{x}) \approx \frac{K}{NV} = \frac{1}{Nh^D} \sum_{n=1}^N k(\mathbf{x} - \mathbf{x}_n)$$

- This method is known as **Parzen Window** estimation.



- In general, we can use any kernel such that

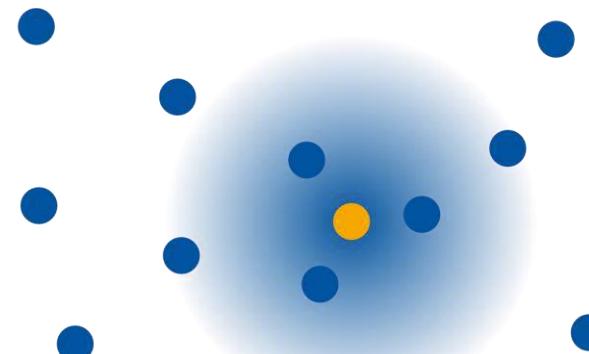
$$k(\mathbf{u}) \geq 0, \quad \int k(\mathbf{u}) d\mathbf{u} = 1$$

$$K = \sum_{n=1}^N k(\mathbf{x} - \mathbf{x}_n)$$

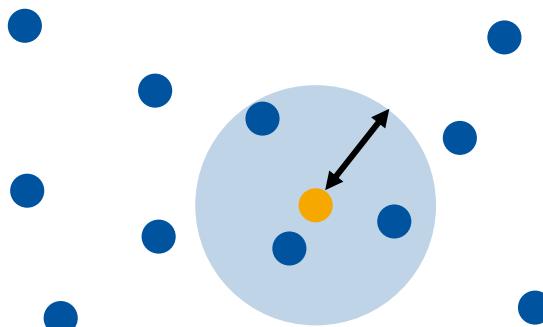
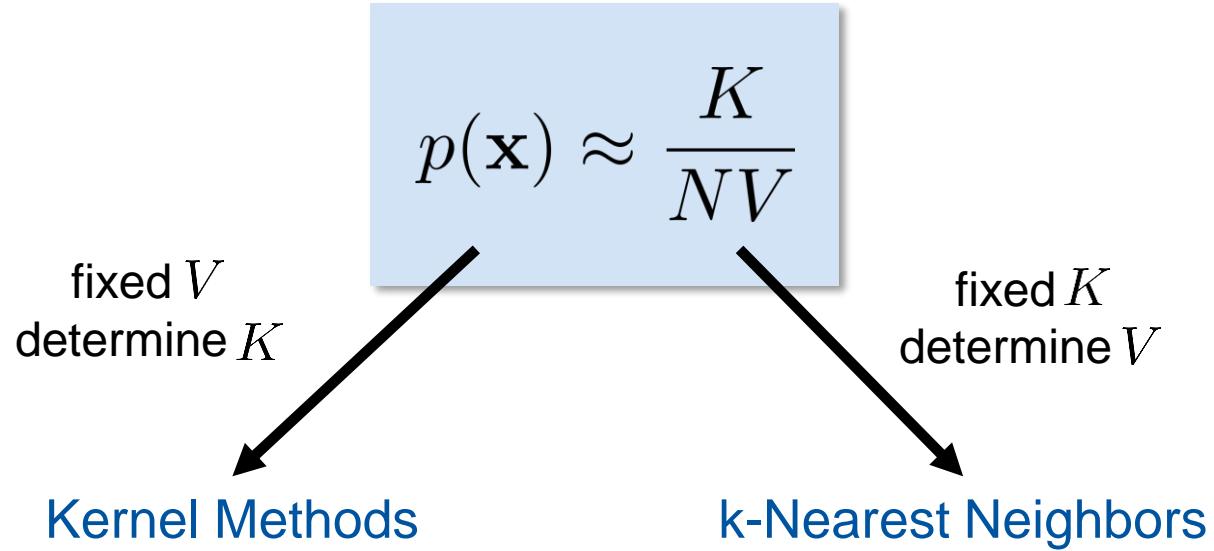
- Then, we get the probability density estimate

$$p(\mathbf{x}) \approx \frac{K}{NV} = \frac{1}{N} \sum_{n=1}^N k(\mathbf{x} - \mathbf{x}_n)$$

- This is known as **Kernel Density Estimation**.



*E.g., a Gaussian kernel
for smoother boundaries.*



Increase the volume V
until the K next data
points are found.

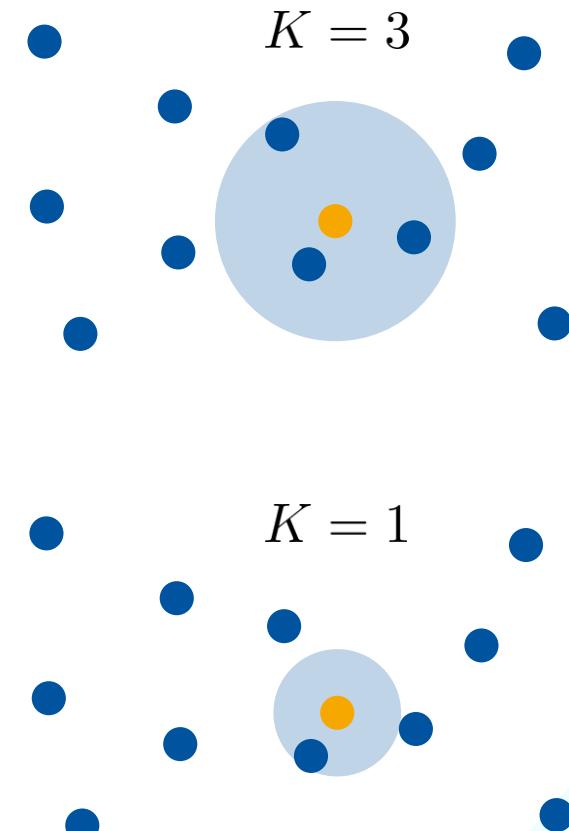
k-Nearest Neighbors

- Fix K , estimate V from the data.
- Consider a hypersphere centered on \mathbf{x} and let it grow to a volume V^* that includes K of the given N data points.
- Then

$$p(\mathbf{x}) \approx \frac{K}{NV^*}$$

- Side note:
 - Strictly speaking, the model produced by k-NN is not a true density model, because the integral over all space diverges.
 - E.g. consider $K = 1$ and a sample exactly on a data point.

$$V^* = 0 \quad \Rightarrow \quad p(\mathbf{x}) \approx \frac{K}{N \cdot 0}$$



Advantages

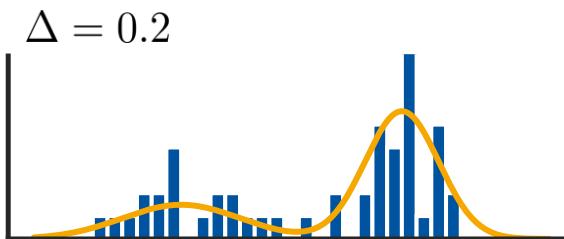
- Very general. In the limit ($N \rightarrow \infty$), every probability density can be represented.
- No computation during training phase
 - Just need to store training set

Limitations

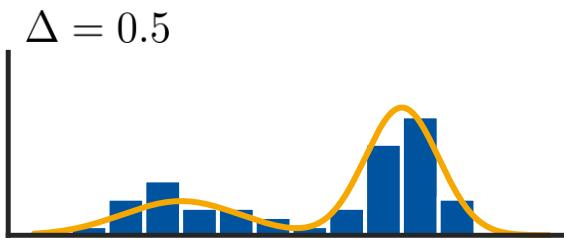
- Requires storing and computing with the entire dataset.
 - Computational costs linear in the number of data points.
 - Can be improved through efficient storage structures (at the cost of some computation during training).
- Choosing the kernel size/ K is a hyperparameter optimization problem.

Bias-Variance Tradeoff

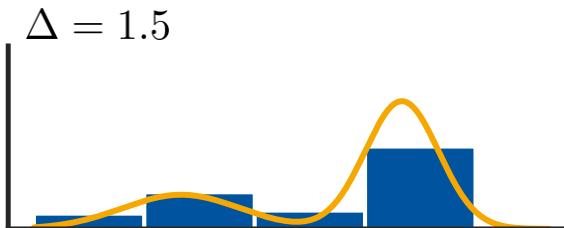
Not smooth enough
Too much variance



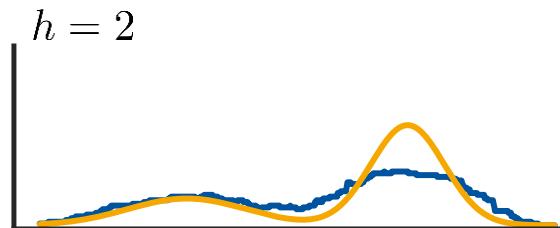
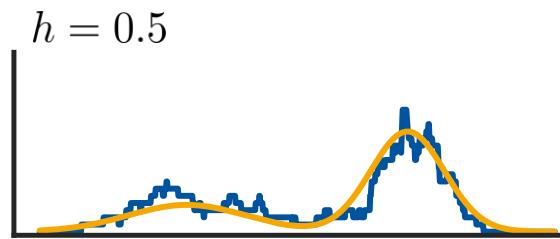
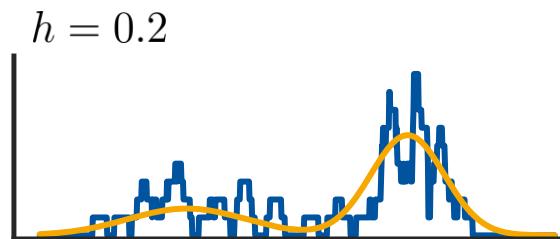
About ok



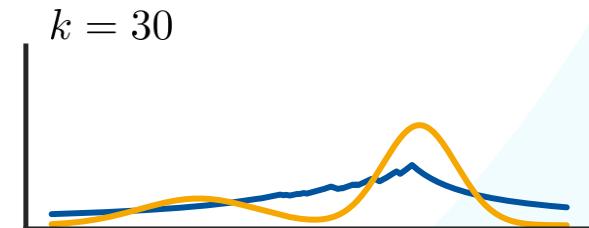
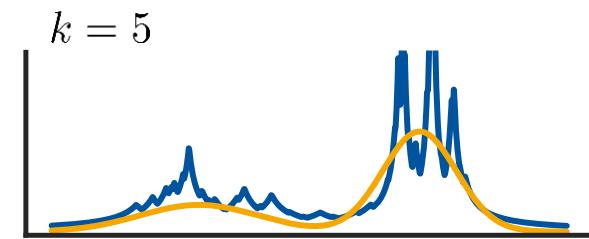
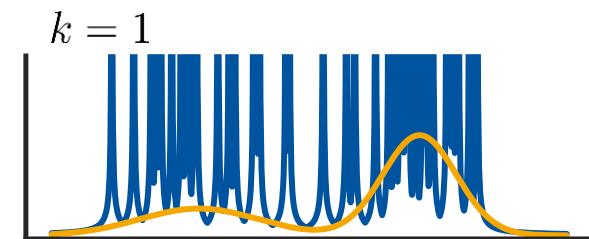
Too smooth
Too much bias



Histograms:
Bin width Δ



Parzen Window:
Kernel size h

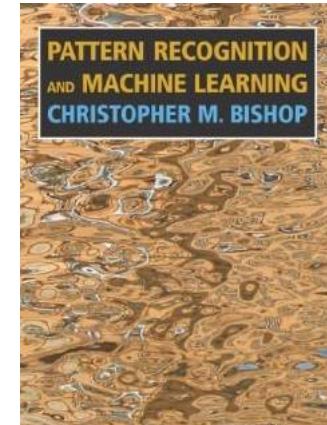


k-NN:
of neighbors k

References and Further Reading

- More information in Bishop's book
 - Gaussian distribution and ML: Ch. 1.2.4 and 2.3.1-2.3.4.
 - Nonparametric methods: Ch. 2.5.
-

Christopher M. Bishop
Pattern Recognition and Machine Learning
Springer, 2006



Elements of Machine Learning & Data Science

Winter semester 2023/24

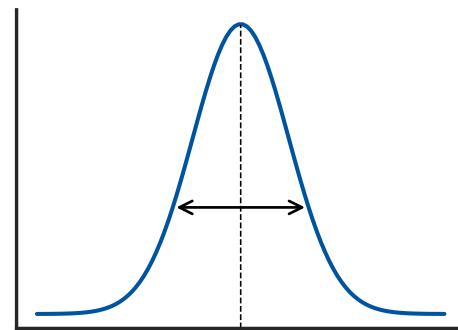
Lecture 4 – Probability Density Estimation I

20.10.2023

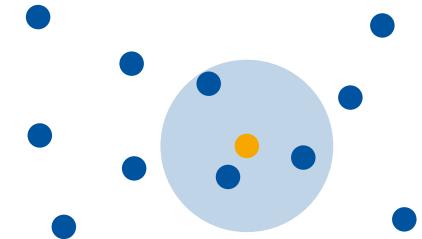
Prof. Bastian Leibe

Machine Learning Topics

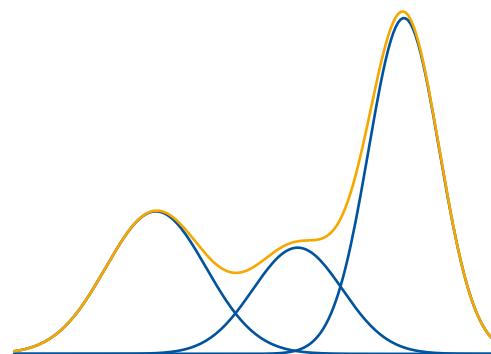
1. Introduction to ML
2. **Probability Density Estimation**
3. Linear Discriminants
4. Linear Regression
5. Logistic Regression
6. Support Vector Machines
7. AdaBoost
8. Neural Network Basics



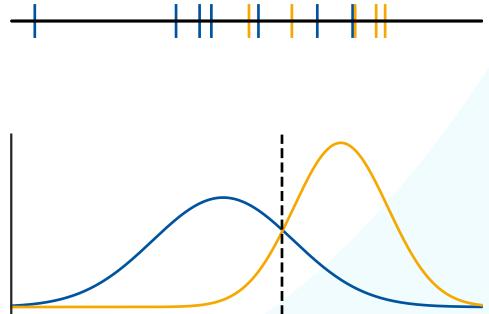
Parametric Methods
& ML-Algorithm



Nonparametric Methods



Mixtures of Gaussians
& EM-Algorithm



Bayes Classifiers

Recap: The Gaussian (or Normal) Distribution

- One-dimensional (univariate) case:

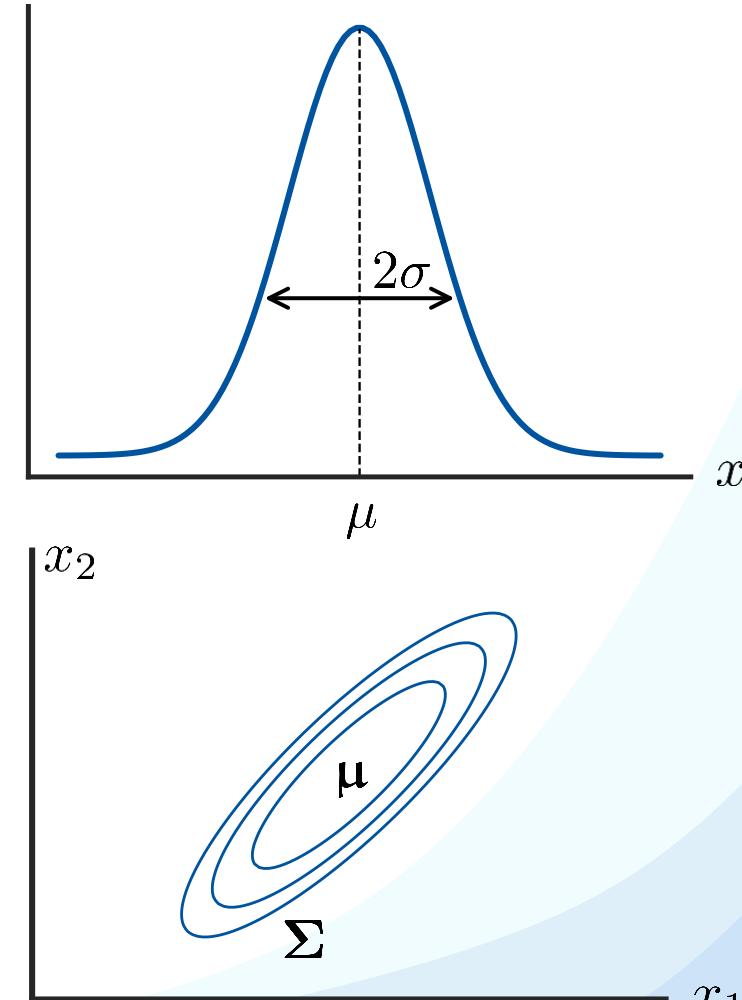
$$\mathcal{N}(x|\mu, \sigma^2) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$$

Mean Variance

- Multi-dimensional (multivariate) case:

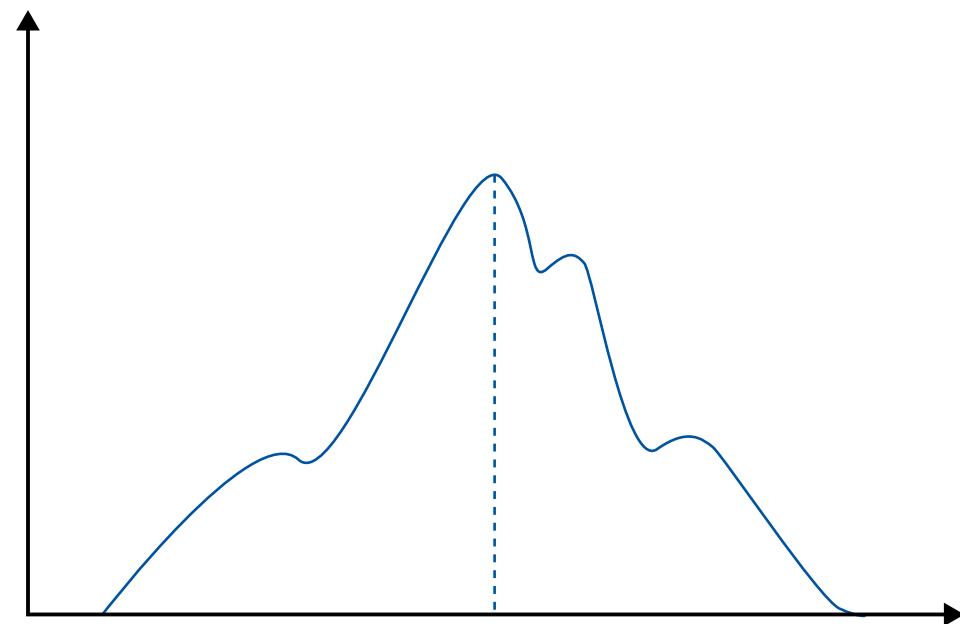
$$\mathcal{N}(\mathbf{x}|\boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{(2\pi)^{D/2}|\boldsymbol{\Sigma}|^{1/2}} \exp\left(-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^\top \boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})\right)$$

Mean vector Covariance matrix



Probability Density Estimation

1. Probability Distributions
2. **Parametric Methods**
3. Nonparametric Methods
4. Mixture Models
5. Bayes Classifier
6. K-NN Classifier

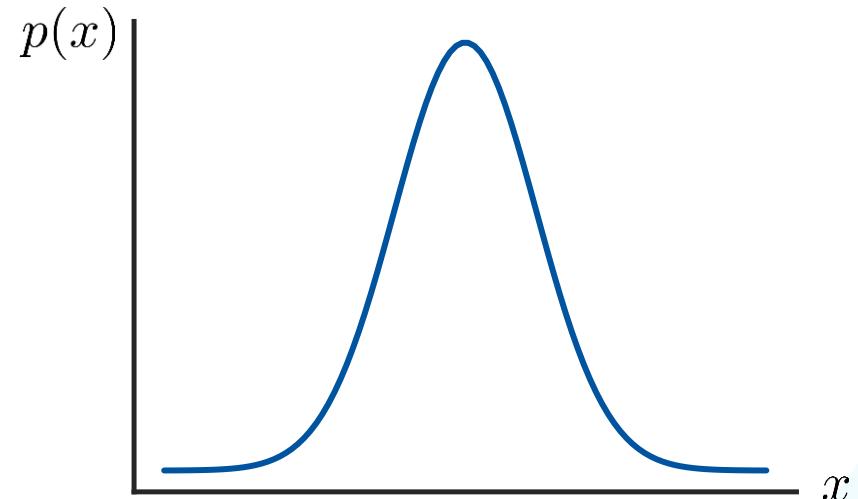


Parametric Methods

- In **parametric methods**, we assume that we know the parametric form of the underlying data distribution.
 - I.e., the equation of the pdf with parameters θ .

Example: $p(x) = \mathcal{N}(x|\mu, \sigma)$

$$\theta = (\mu, \sigma)$$



Parametric Methods

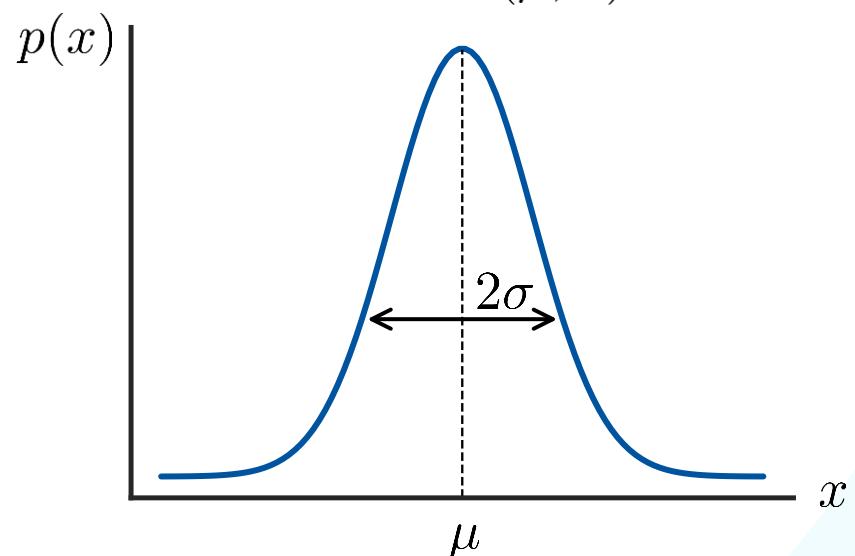
- In **parametric methods**, we assume that we know the parametric form of the underlying data distribution.
 - I.e., the equation of the pdf with parameters θ .
- Goal: Estimate θ from training data $\mathcal{X} = \{\mathbf{x}_1, \dots, \mathbf{x}_N\}$.
- **Likelihood** of θ :

$$L(\theta) = p(\mathcal{X}|\theta)$$

Probability that the data \mathcal{X} was indeed generated by a distribution with parameters θ .

Example: $p(x) = \mathcal{N}(x|\mu, \sigma)$

$$\theta = (\mu, \sigma)$$



Maximum Likelihood Approach

- Idea: Find optimal parameters by maximizing $L(\theta)$.
- Computation of the likelihood:

- Single data point (e.g., for Gaussian):

$$p(x_n|\theta) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(x_n - \mu)^2}{2\sigma^2}\right)$$

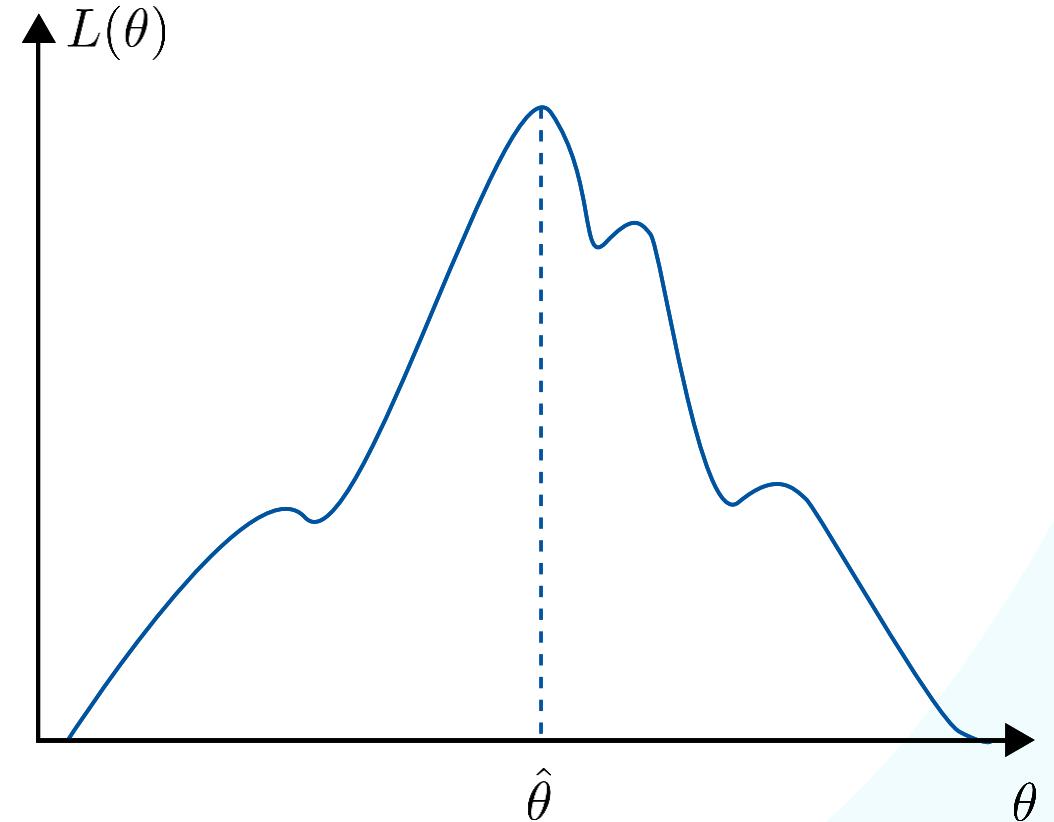
- Assumption: all data points are independent

$$L(\theta) = p(\mathcal{X}|\theta) = \prod_{n=1}^N p(x_n|\theta)$$

- Negative Log-Likelihood (“Energy”):

$$E(\theta) = -\ln L(\theta) = -\sum_{n=1}^N \ln p(x_n|\theta)$$

Maximizing the likelihood \Leftrightarrow minimizing the negative log-likelihood.



- Minimizing the negative log-likelihood:
 - Take the derivative and set it to zero.

$$\frac{\partial}{\partial \theta} E(\theta) = -\frac{\partial}{\partial \theta} \sum_{n=1}^N \ln p(x_n|\theta) = -\sum_{n=1}^N \frac{\frac{\partial}{\partial \theta} p(x_n|\theta)}{p(x_n|\theta)} \stackrel{!}{=} 0$$

- Log-likelihood for Normal distribution (1D case):

$$\begin{aligned}\frac{\partial}{\partial \mu} E(\mu, \sigma) &= -\sum_{n=1}^N \frac{\frac{\partial}{\partial \mu} p(x_n|\mu, \sigma)}{p(x_n|\mu, \sigma)} \\ &= -\sum_{n=1}^N -\frac{2(x_n - \mu)}{2\sigma^2} \frac{p(x_n|\mu, \sigma)}{p(x_n|\mu, \sigma)}\end{aligned}$$

$$p(x_n|\mu, \sigma) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(x_n - \mu)^2}{2\sigma^2}\right)$$

$$\frac{\partial}{\partial \mu} p(x_n|\mu, \sigma) = -\frac{2(x_n - \mu)}{2\sigma^2} p(x_n|\mu, \sigma)$$

$$=\frac{1}{\sigma^2} \sum_{n=1}^N (x_n - \mu) = \cancel{\frac{1}{\sigma^2}} \left(\sum_{n=1}^N x_n - N\mu \right) \stackrel{!}{=} 0 \iff \hat{\mu} = \frac{1}{N} \sum_{n=1}^N x_n$$

- By minimizing the negative log-likelihood, we found: Similarly, we can derive:

$$\hat{\mu} = \frac{1}{N} \sum_{n=1}^N x_n$$

sample mean

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

sample variance

- $\hat{\theta} = (\hat{\mu}, \hat{\sigma})$ is the Maximum Likelihood estimate for the parameters of a Gaussian distribution.
 - This is a very important result.
 - Unfortunately, it is wrong...

- To be precise, the result is not wrong, but **biased**.
- Assume the samples x_1, x_2, \dots, x_N come from a true Gaussian distribution with mean μ and variance σ^2
 - It can be shown that the expected estimates are then

$$\mathbb{E}[\mu_{\text{ML}}] = \mu$$

$$\mathbb{E}[\sigma_{\text{ML}}^2] = \left(\frac{N-1}{N}\right)\sigma^2$$

\Rightarrow *The ML estimate will underestimate the true variance!*

- We can correct for this bias:

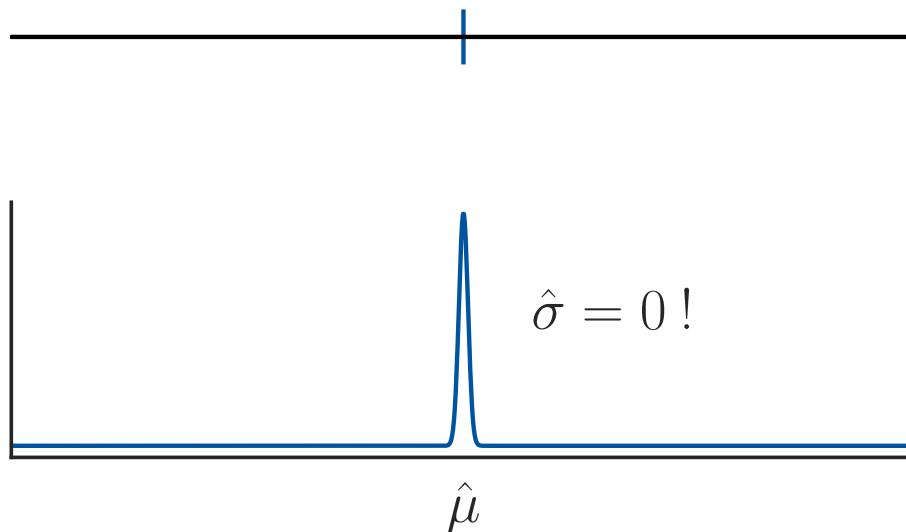
$$\hat{\sigma}^2 = \frac{N}{N-1}\sigma_{\text{ML}}^2 = \frac{1}{N-1} \sum_{n=1}^N (x_n - \hat{\mu})^2$$

- Maximum Likelihood has several significant limitations.
 - It systematically underestimates the variance of the distribution!
 - E.g., consider the estimate for a single sample:

$$N = 1, \mathcal{X} = \{x_1\}$$

$$\hat{\mu} = \frac{1}{N} \sum_{n=1}^N x_n = x_1$$

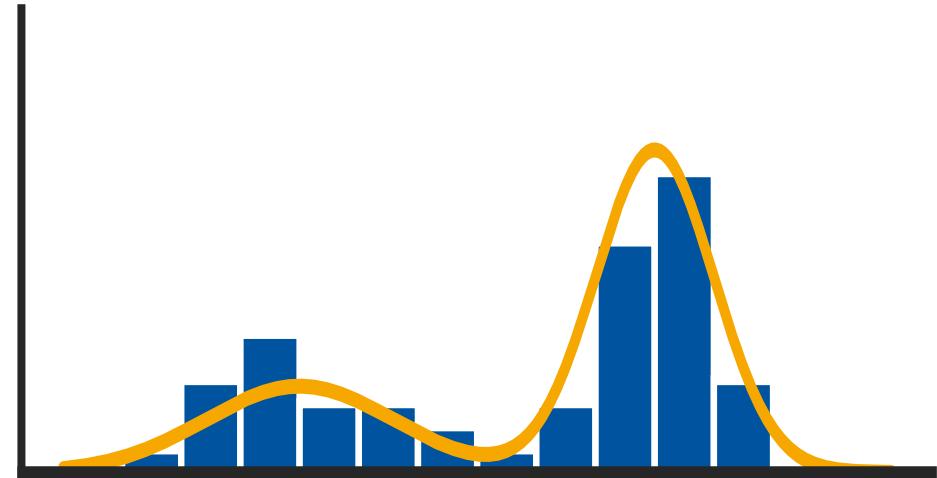
$$\hat{\sigma}^2 = \frac{1}{N} \sum_{n=1}^N (x_n - \hat{\mu})^2 = 0$$



- We say ML overfits to the observed data.
- *We will still often use Maximum Likelihood, but it is important to know about this effect.*

Probability Density Estimation

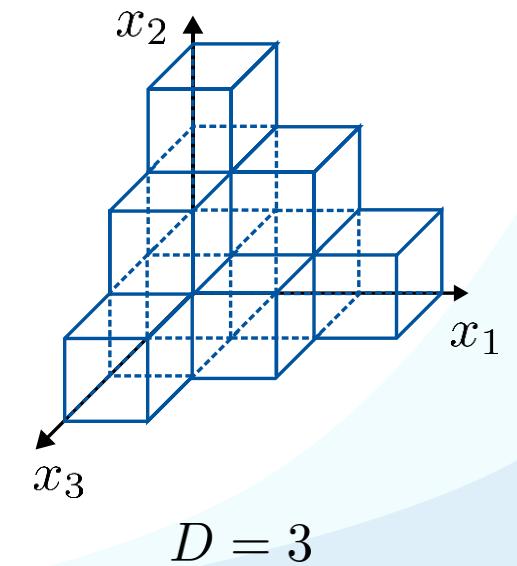
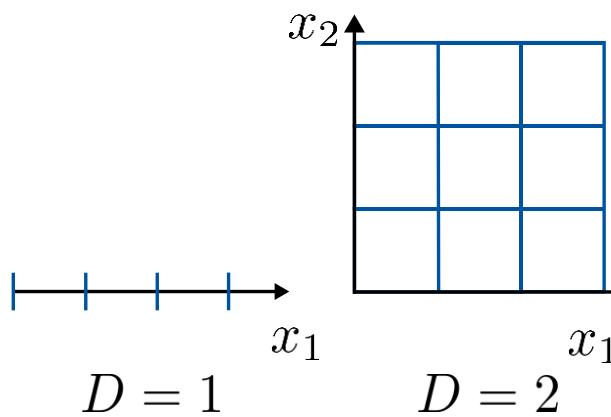
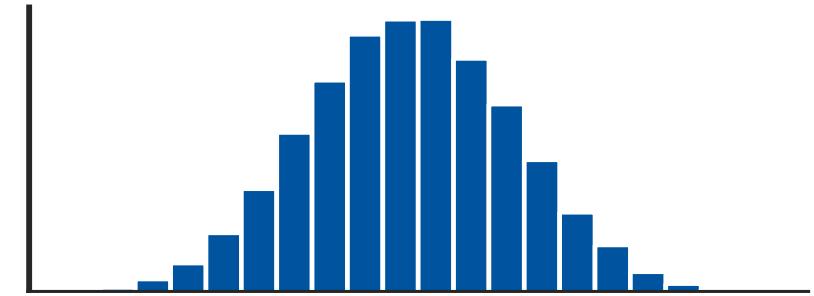
1. Probability Distributions
2. Parametric Methods
- 3. Nonparametric Methods**
 - a) Histograms
 - b) Kernel Methods & k-Nearest Neighbors
4. Mixture Models
5. Bayes Classifier
6. K-NN Classifier



Histograms

- Partition the data space into M distinct bins with widths Δ_i and count the number of observations n_i in each bin.
- Then, $p_i = \frac{n_i}{N\Delta_i}$.
- Often the same width is used for all bins.
- This can be done, in principle, for any dimensionality D .

...but the required number of bins grows exponentially with D !



The bin width Δ acts as a smoothing factor.

Not smooth enough



About ok



Too smooth



Advantages

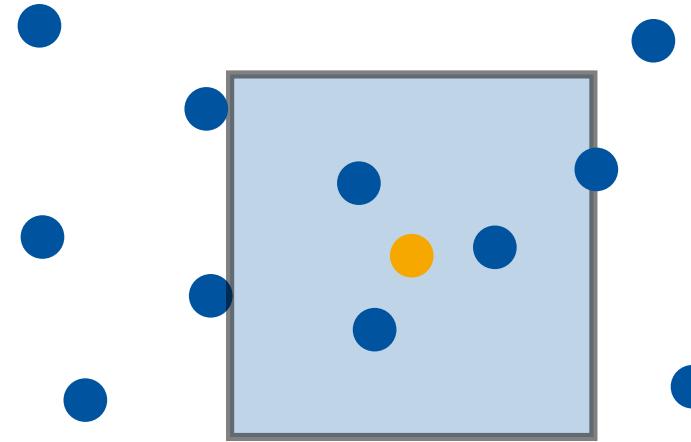
- Very general method. In the limit ($N \rightarrow \infty$), every probability density can be represented.
- No need to store the data points once histogram is computed.

Limitations

- Rather brute-force.
- Discontinuities at bin edges.
- Choosing right bin size is hard.
- Unsuitable for high-dimensional feature spaces.

Probability Density Estimation

1. Probability Distributions
2. Parametric Methods
3. **Nonparametric Methods**
 - a) Histograms
 - b) **Kernel Methods & k-Nearest Neighbors**
4. Mixture Models
5. Bayes Classifier
6. K-NN Classifier



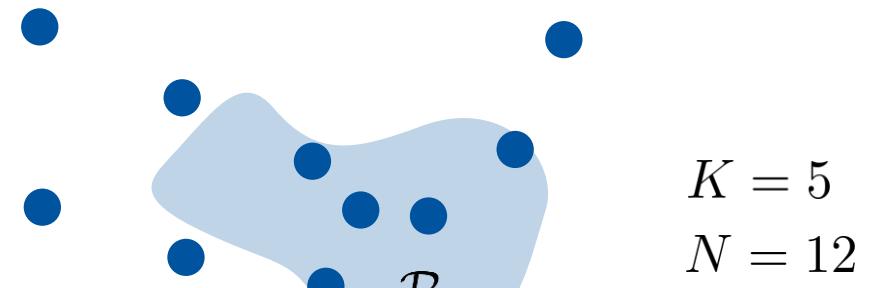
Kernel Methods and k-Nearest Neighbors

- Data point \mathbf{x} comes from pdf $p(\mathbf{x})$.
 - Probability that \mathbf{x} falls into small region \mathcal{R} :

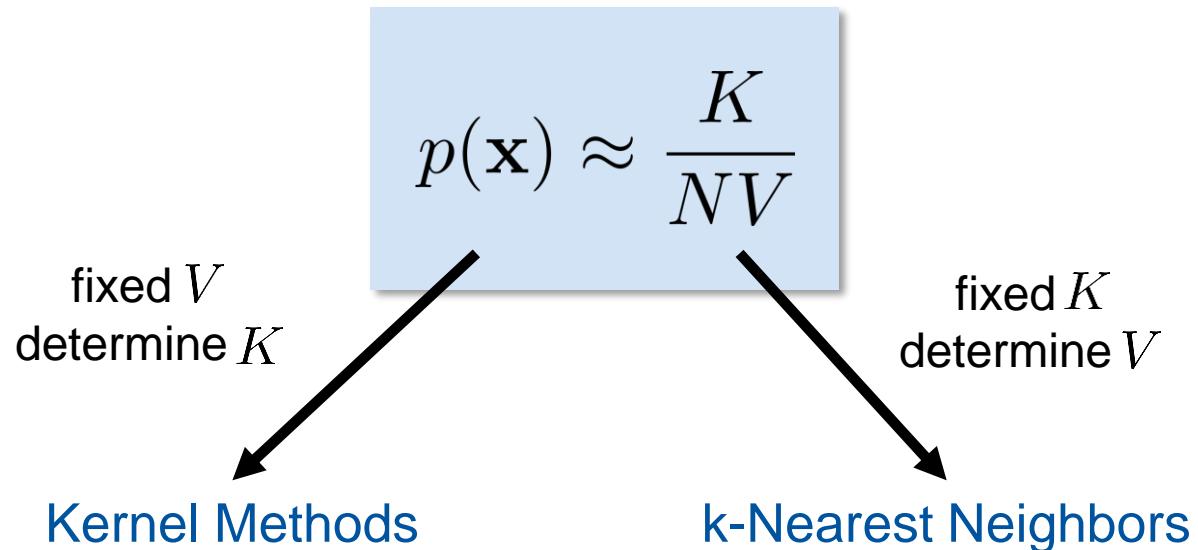
$$P = \int_{\mathcal{R}} p(y) dy \approx p(\mathbf{x})V$$

- Estimate $p(\mathbf{x})$ from samples
 - Let K be the number of samples that fall into \mathcal{R} .
 - If the number of samples N is sufficiently large, we can estimate P as:

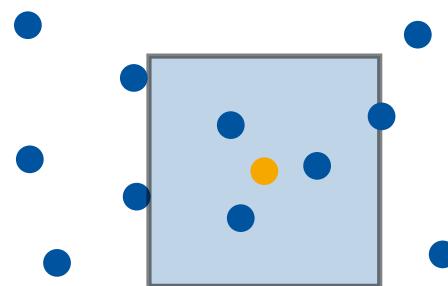
$$P = \frac{K}{N} \Rightarrow p(\mathbf{x}) \approx \frac{K}{NV}$$



For sufficiently small \mathcal{R} ,
 $p(\mathbf{x})$ is roughly constant.
 V : volume of \mathcal{R} .



Example: Determine the number K of data points inside a fixed hypercube



Kernel Methods

- Hypercube of dimension D with edge length h :

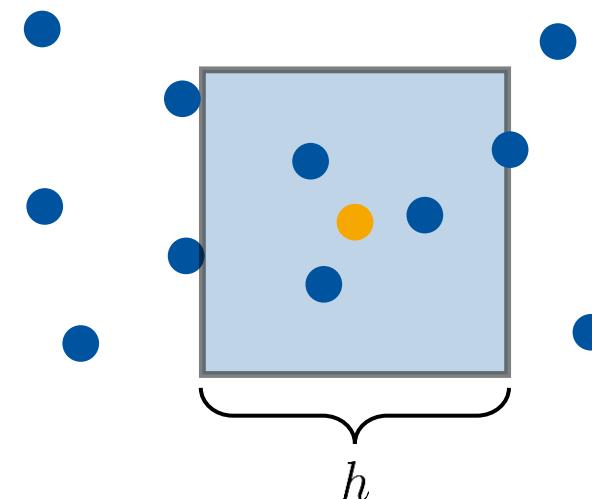
$$k(\mathbf{u}) = \begin{cases} 1, & \text{if } |u_i| \leq \frac{1}{2}h, \ i = 1, \dots, D \\ 0, & \text{otherwise} \end{cases}$$

$$K = \sum_{n=1}^N k(\mathbf{x} - \mathbf{x}_n) \quad V = \int k(\mathbf{u}) d\mathbf{u} = h^D$$

- Probability density estimate:

$$p(\mathbf{x}) \approx \frac{K}{NV} = \frac{1}{Nh^D} \sum_{n=1}^N k(\mathbf{x} - \mathbf{x}_n)$$

- This method is known as **Parzen Window** estimation.



- In general, we can use any kernel such that

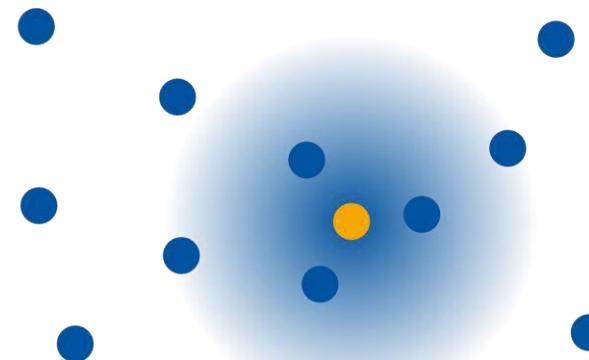
$$k(\mathbf{u}) \geq 0, \quad \int k(\mathbf{u}) d\mathbf{u} = 1$$

$$K = \sum_{n=1}^N k(\mathbf{x} - \mathbf{x}_n)$$

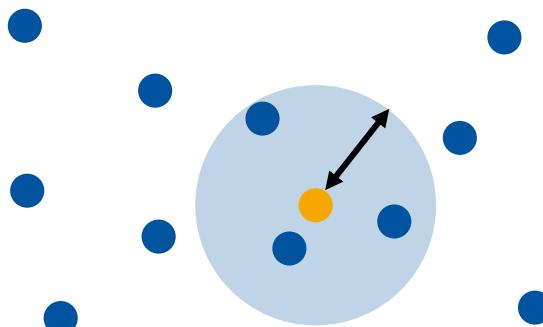
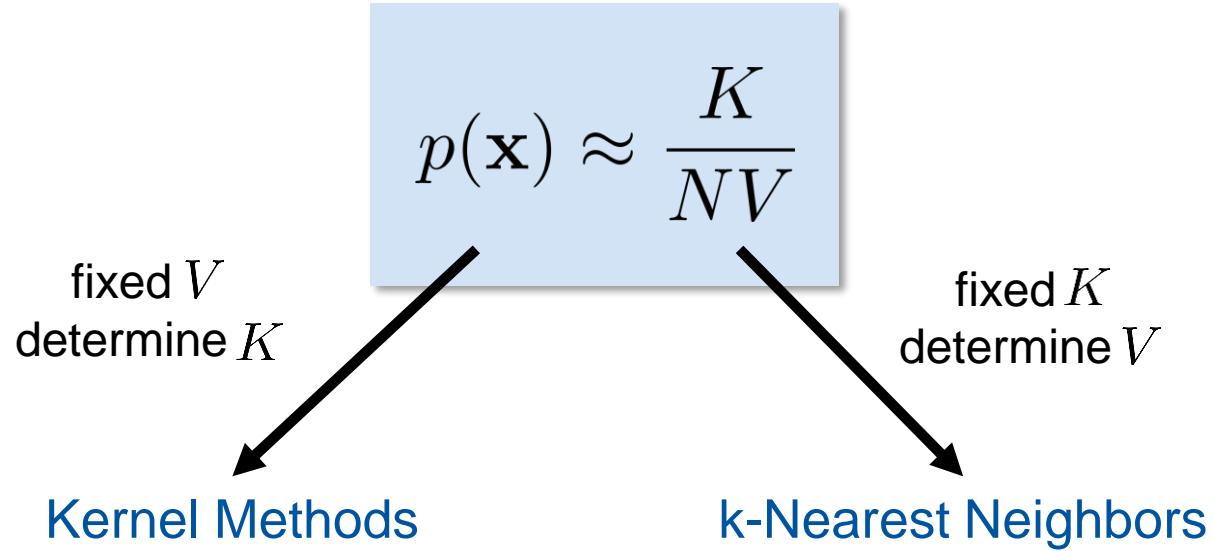
- Then, we get the probability density estimate

$$p(\mathbf{x}) \approx \frac{K}{NV} = \frac{1}{N} \sum_{n=1}^N k(\mathbf{x} - \mathbf{x}_n)$$

- This is known as **Kernel Density Estimation**.



*E.g., a Gaussian kernel
for smoother boundaries.*



Increase the volume V
until the K next data
points are found.

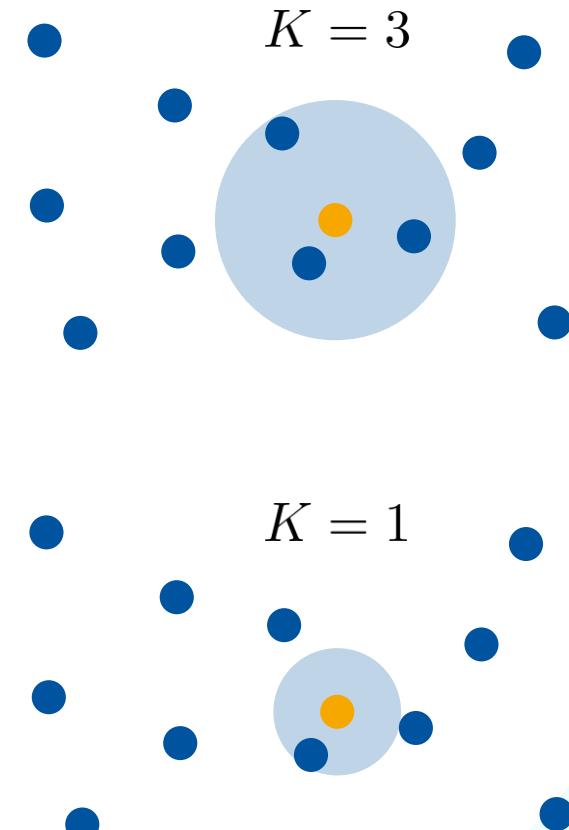
k-Nearest Neighbors

- Fix K , estimate V from the data.
- Consider a hypersphere centered on \mathbf{x} and let it grow to a volume V^* that includes K of the given N data points.
- Then

$$p(\mathbf{x}) \approx \frac{K}{NV^*}$$

- Side note:
 - Strictly speaking, the model produced by k-NN is not a true density model, because the integral over all space diverges.
 - E.g. consider $K = 1$ and a sample exactly on a data point.

$$V^* = 0 \quad \Rightarrow \quad p(\mathbf{x}) \approx \frac{K}{N \cdot 0}$$



Advantages

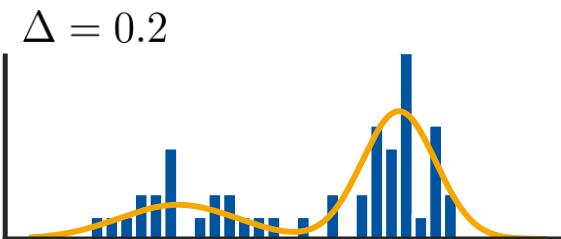
- Very general. In the limit ($N \rightarrow \infty$), every probability density can be represented.
- No computation during training phase
 - Just need to store training set

Limitations

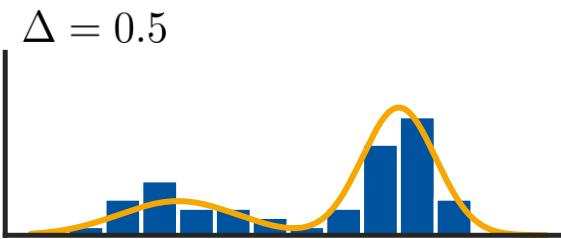
- Requires storing and computing with the entire dataset.
 - Computational costs linear in the number of data points.
 - Can be improved through efficient storage structures (at the cost of some computation during training).
- Choosing the kernel size/ K is a hyperparameter optimization problem.

Bias-Variance Tradeoff

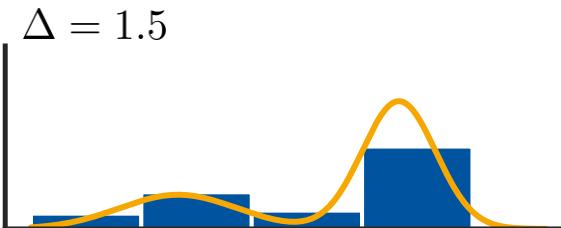
Not smooth enough
Too much variance



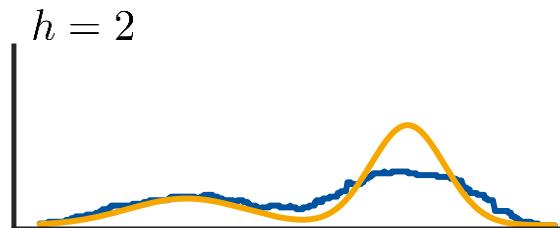
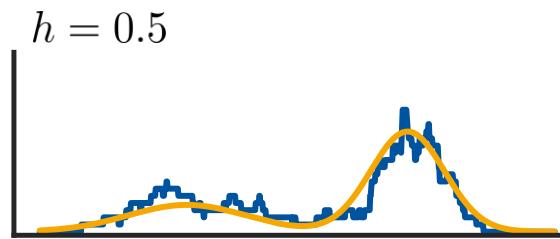
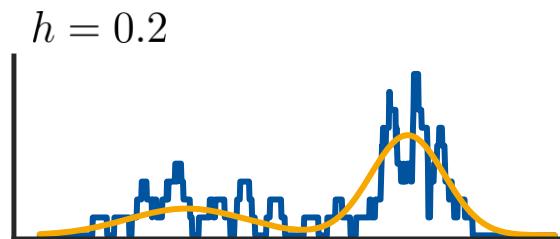
About ok



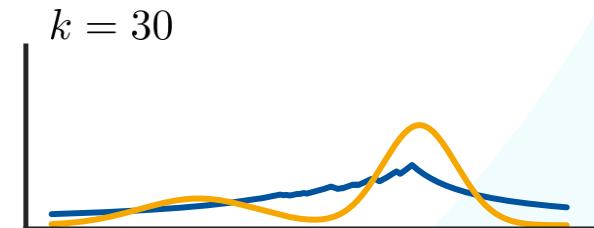
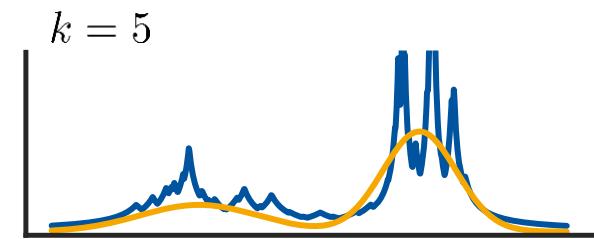
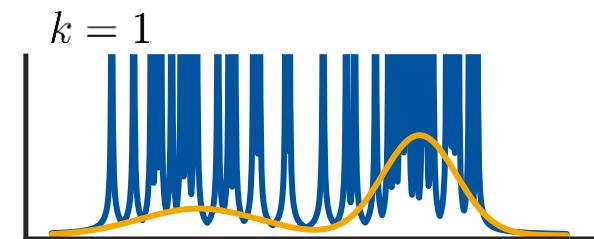
Too smooth
Too much bias



Histograms:
Bin width Δ



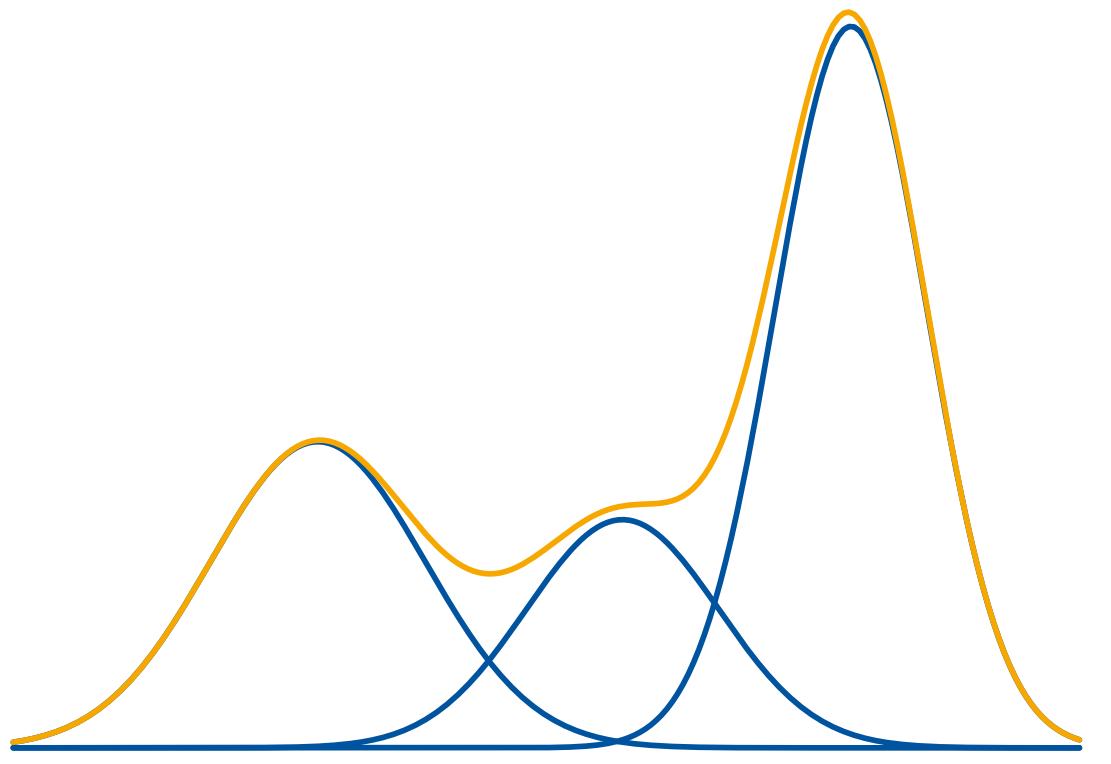
Parzen Window:
Kernel size h



k-NN:
of neighbors k

Probability Density Estimation

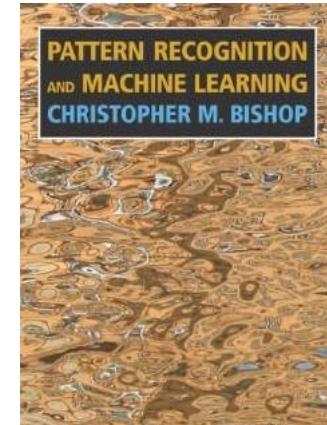
1. Probability Distributions
2. Parametric Methods
3. Nonparametric Methods
4. **Mixture Models**
5. Bayes Classifier
6. K-NN Classifier



References and Further Reading

- More information in Bishop's book
 - Gaussian distribution and ML: Ch. 1.2.4 and 2.3.1-2.3.4.
 - Nonparametric methods: Ch. 2.5.

Christopher M. Bishop
Pattern Recognition and Machine Learning
Springer, 2006



Elements of Machine Learning & Data Science

Winter semester 2023/24

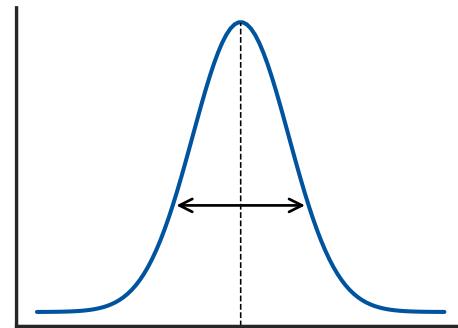
Lecture 5 – Probability Density Estimation II

24.10.2023

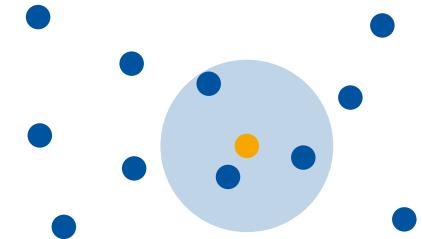
Prof. Bastian Leibe

Machine Learning Topics

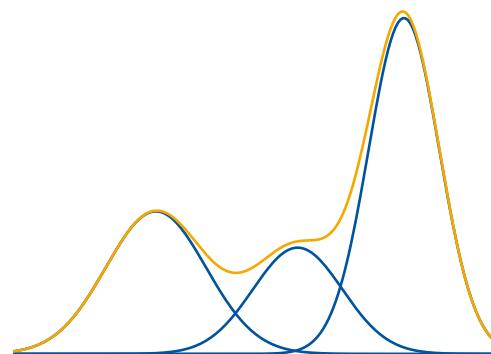
1. Introduction to ML
2. **Probability Density Estimation**
3. Linear Discriminants
4. Linear Regression
5. Logistic Regression
6. Support Vector Machines
7. AdaBoost
8. Neural Network Basics



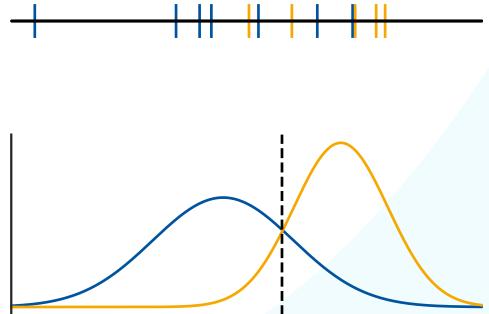
Parametric Methods
& ML-Algorithm



Nonparametric Methods



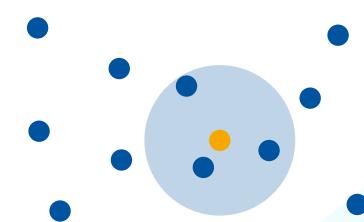
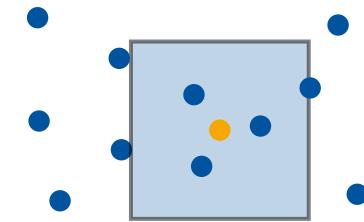
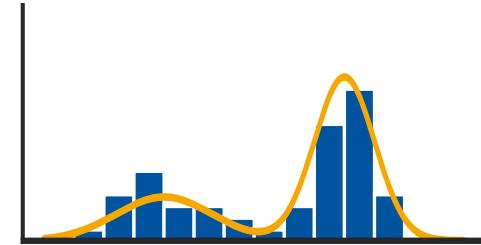
Mixtures of Gaussians
& EM-Algorithm



Bayes Classifiers

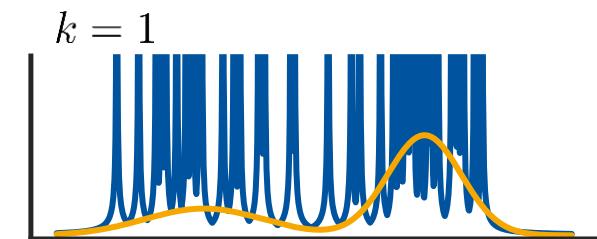
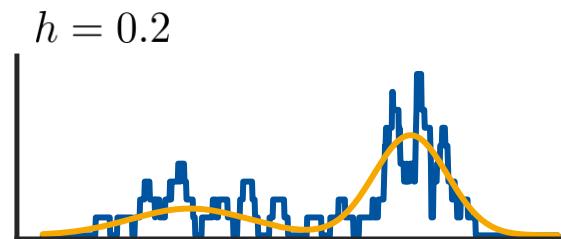
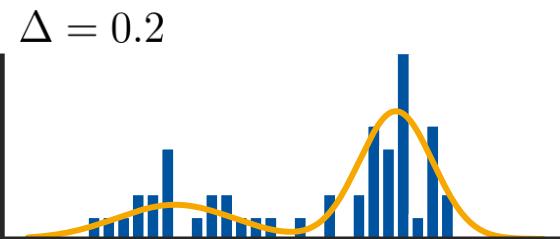
Recap: Probability Density Estimation

1. Probability Distributions
2. Parametric Methods
3. **Nonparametric Methods**
 - a) **Histograms**
 - b) **Kernel Methods & k-Nearest Neighbors**
4. Mixture Models
5. Bayes Classifier
6. K-NN Classifier

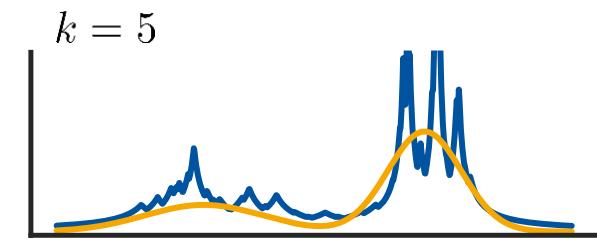
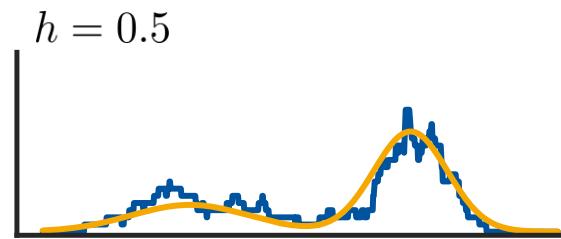
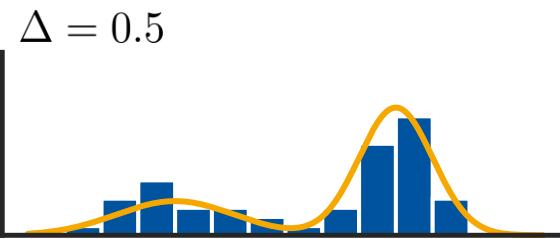


Recap: Nonparametric Approaches

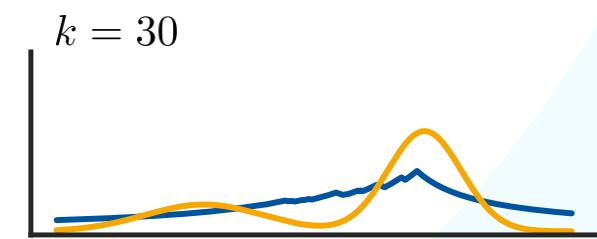
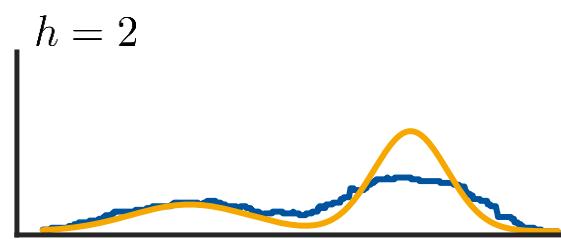
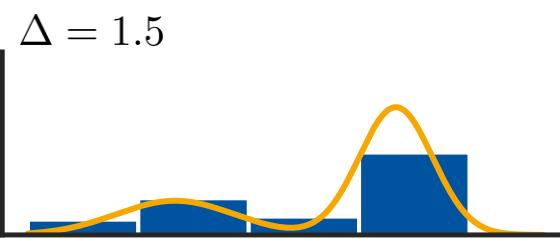
Not smooth enough
Too much variance



About ok



Too smooth
Too much bias



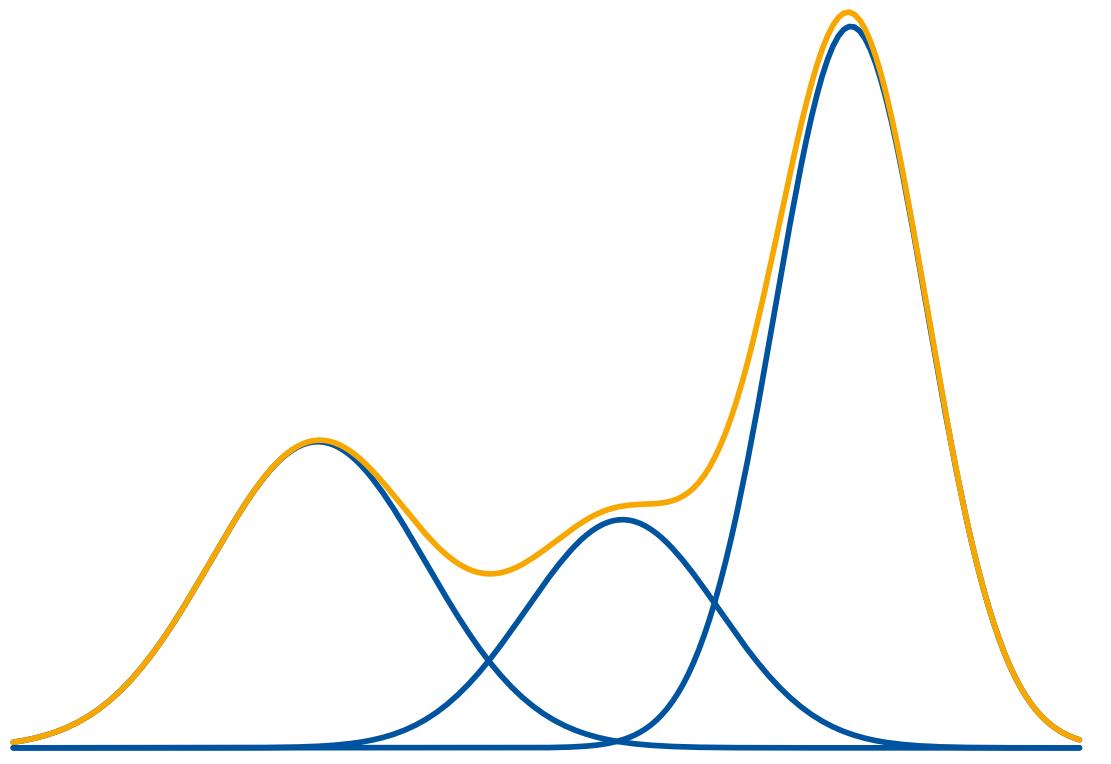
Histograms:
Bin width Δ

Parzen Window:
Kernel size h

k-NN:
of neighbors k

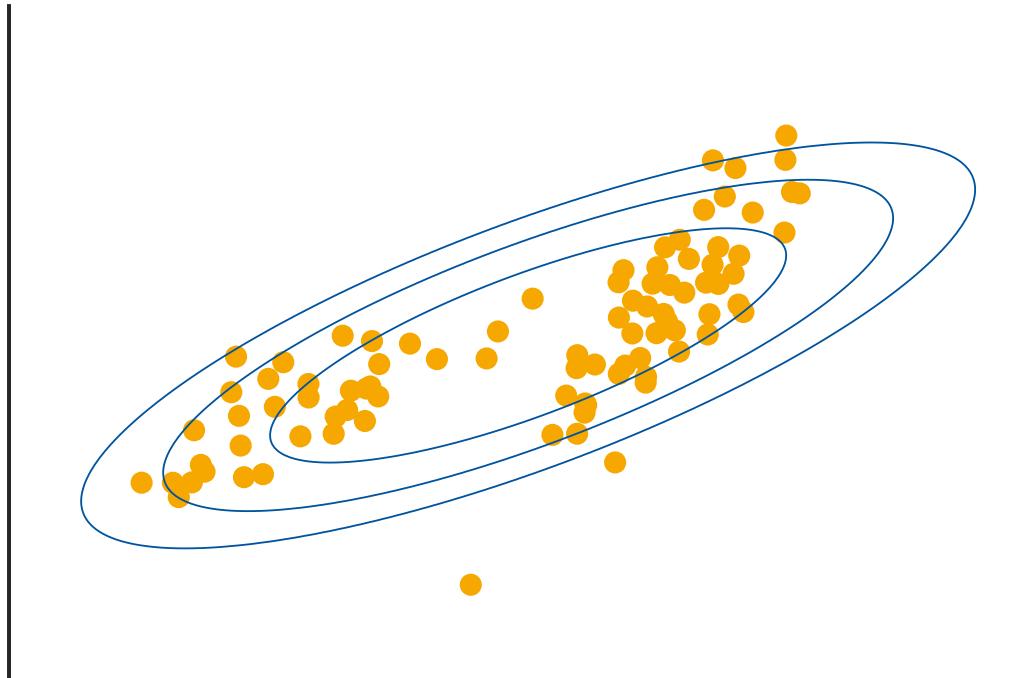
Probability Density Estimation

1. Probability Distributions
2. Parametric Methods
3. Nonparametric Methods
4. **Mixture Models**
5. Bayes Classifier
6. K-NN Classifier



Mixture Models

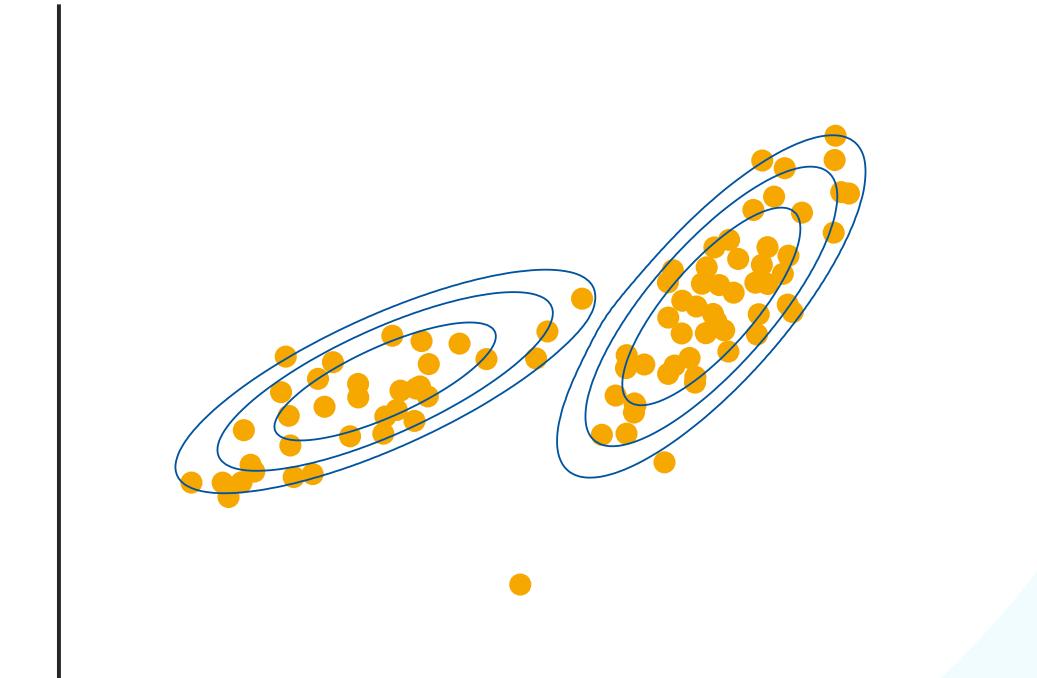
- Often, a single parametric representation is not enough.
- Struggle to fit multimodal data



Single Gaussian

Mixture Models

- Often, a single parametric representation is not enough.
- Struggle to fit multimodal data
- Mixture models combine multiple densities into a single distribution.
 - Improves modeling of multimodal data.



Mixture of two Gaussians

Mixture of Gaussians (MoG)

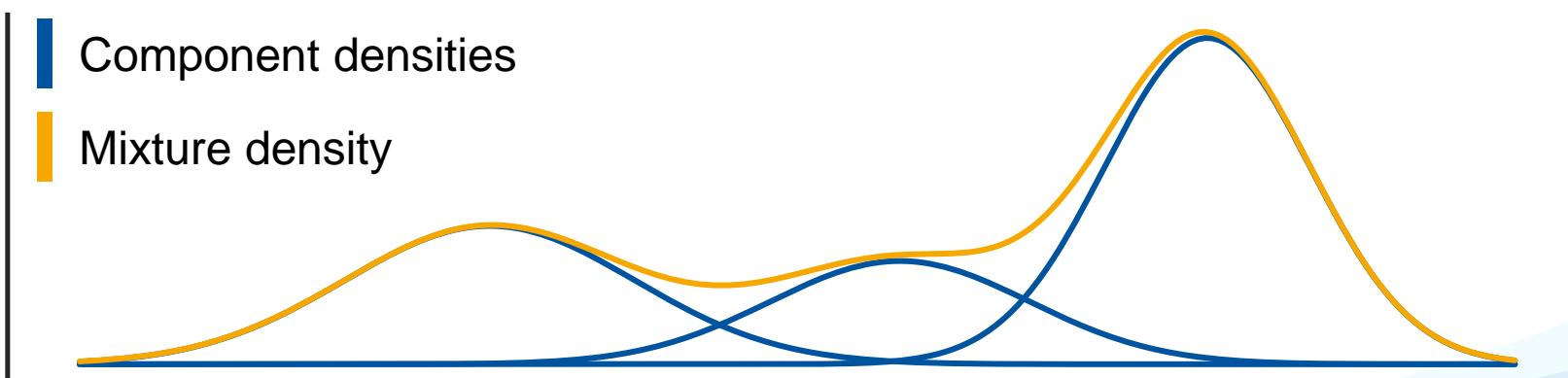
- This is the sum of M individual Normal distributions:

$$p(x|\theta) = \sum_{j=1}^M p(x|\theta_j)p(j)$$

Likelihood of measurement x given mixture component j

Prior of component j

- In the limit, every smooth distribution can be approximated this way (if M is large enough).



- For Gaussians, the complete mixture model is given as:

$$p(x|\theta) = \sum_{j=1}^M p(x|\theta_j)p(j)$$

$$p(x|\theta_j) = \mathcal{N}(x|\mu_j, \sigma_j^2) = \frac{1}{\sqrt{2\pi}\sigma_j} \exp\left(-\frac{(x - \mu_j)^2}{2\sigma_j^2}\right)$$

$$p(j) = \pi_j \text{ with } 0 \leq \pi_j \leq 1 \text{ and } \sum_{j=1}^M \pi_j = 1$$

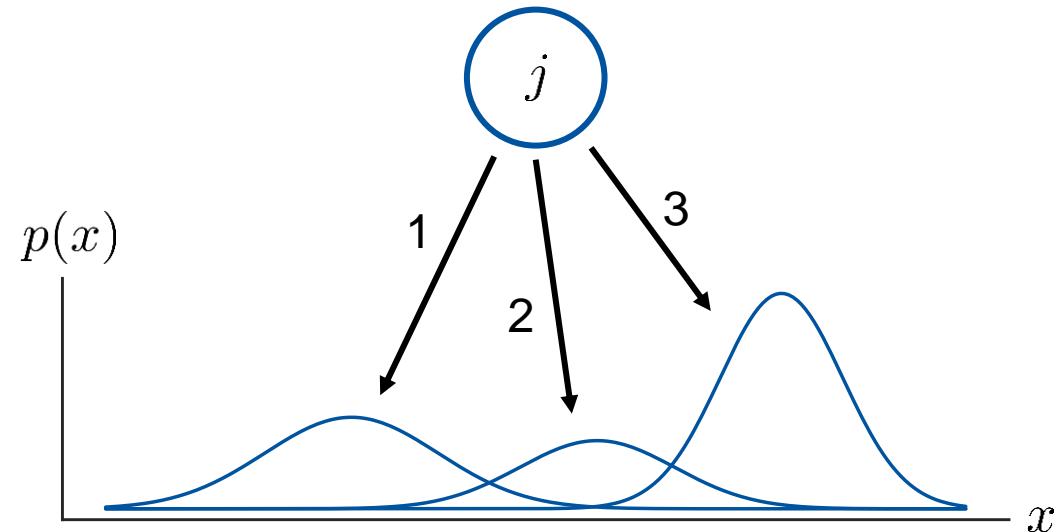
Note: this integrates to 1

$$\int p(x|\theta)dx = 1$$

Total parameters:

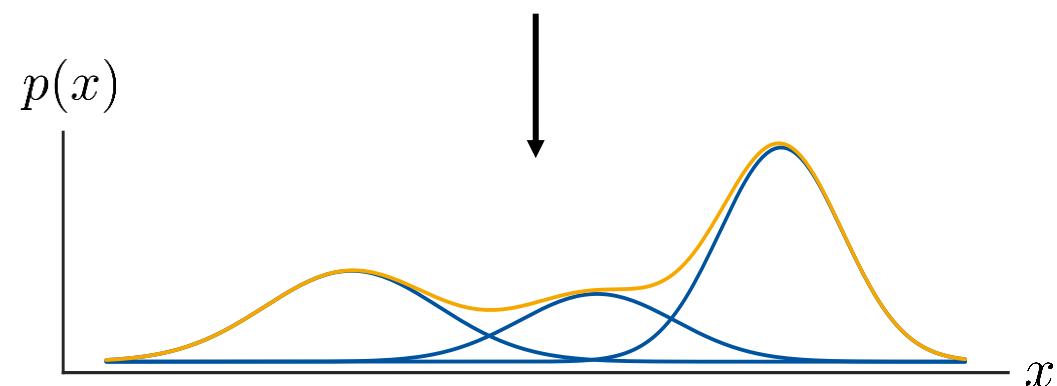
$$\theta = (\pi_1, \mu_1, \sigma_1, \dots, \pi_M, \mu_M, \sigma_M)$$

- MoGs are **generative models**: we can easily sample from them.



$$p(j) = \pi_j$$

Sample the component using its “weight”



$$p(x|\theta_j)$$

Sample from the mixture component

$$p(x|\theta) = \sum_{j=1}^M p(x|\theta_j)p(j)$$

The result is a sample from the mixture density

Learning a Mixture Model

- Apply Maximum Likelihood
 - Minimize $E = -\ln L(\theta) = -\sum_{n=1}^N \ln p(\mathbf{x}_n | \theta)$
 - Let's first look at μ_j :

$$\frac{\partial E}{\partial \mu_j} = 0$$

- We can already see that this will be difficult:

$$\ln p(\mathcal{X} | \boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \sum_{n=1}^N \ln \left(\sum_{k=1}^K \pi_k \mathcal{N}(\mathbf{x}_n | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \right)$$

Steps for Maximum Likelihood:

1. Express Likelihood $L(\theta)$
2. Apply negative logarithm to get $E(\theta)$
3. Take derivative, set to zero
4. Solve for parameters

Learning a Mixture Model

- Apply Maximum Likelihood
 - Minimize $E = -\ln L(\theta) = -\sum_{n=1}^N \ln p(\mathbf{x}_n | \theta)$
 - Let's first look at μ_j :

$$\frac{\partial E}{\partial \mu_j} = 0$$

- We can already see that this will be difficult:

$$\ln p(\mathcal{X} | \boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \sum_{n=1}^N \ln \left(\sum_{k=1}^K \pi_k \mathcal{N}(\mathbf{x}_n | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \right)$$

This will cause problems!

Steps for Maximum Likelihood:

1. Express Likelihood $L(\theta)$
2. Apply negative logarithm to get $E(\theta)$
3. Take derivative, set to zero
4. Solve for parameters

$$\begin{aligned}
 \frac{\partial E}{\partial \mu_j} &= - \sum_{n=1}^N \frac{\frac{\partial}{\partial \mu_j} p(\mathbf{x}_n | \theta_j)}{\sum_{k=1}^K p(\mathbf{x}_n | \theta_k)} \\
 &= - \sum_{n=1}^N \left(\Sigma^{-1} (\mathbf{x}_n - \mu_j) \frac{p(\mathbf{x}_n | \theta_j)}{\sum_{k=1}^K p(\mathbf{x}_n | \theta_k)} \right) \\
 &= - \Sigma^{-1} \sum_{n=1}^N (\mathbf{x}_n - \mu_j) \frac{\pi_j \mathcal{N}(\mathbf{x}_n | \mu_j, \Sigma_j)}{\sum_{k=1}^K \pi_k \mathcal{N}(\mathbf{x}_n | \mu_k, \Sigma_k)} \stackrel{!}{=} 0
 \end{aligned}$$

$$\begin{aligned}
 \frac{\partial}{\partial \mu_j} \mathcal{N}(\mathbf{x}_n | \mu_k, \Sigma_k) &= \\
 \Sigma^{-1} (\mathbf{x}_n - \mu_j) \mathcal{N}(\mathbf{x}_n | \mu_k, \Sigma_k)
 \end{aligned}$$

= $\gamma_j(\mathbf{x}_n)$

“responsibility” of
component j for \mathbf{x}_n

We thus obtain $\mu_j = \frac{\sum_{n=1}^N \gamma_j(\mathbf{x}_n) \mathbf{x}_n}{\sum_{n=1}^N \gamma_j(\mathbf{x}_n)}$

$$\boldsymbol{\mu}_j = \frac{\sum_{n=1}^N \gamma_j(\mathbf{x}_n) \mathbf{x}_n}{\sum_{n=1}^N \gamma_j(\mathbf{x}_n)}$$

$$\gamma_j(\mathbf{x}_n) = \frac{\pi_j \mathcal{N}(\mathbf{x}_n | \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)}{\sum_{k=1}^K \pi_k \mathcal{N}(\mathbf{x}_n | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}$$

- There is no direct analytical solution!

$$\frac{\partial E}{\partial \boldsymbol{\mu}_j} = f(\pi_1, \boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1, \dots, \pi_M, \boldsymbol{\mu}_M, \boldsymbol{\Sigma}_M)$$

- Complex gradient function (non-linear mutual dependencies)
- Optimization of one Gaussian depends on all other Gaussians!
- Standard solution: iterative optimization with EM algorithm

The EM Algorithm

- The Expectation-Maximization (EM) Algorithm alternates between two steps:
 - **E-Step:** softly assign samples to mixture components:

$$\gamma_j(\mathbf{x}_n) \leftarrow \frac{\pi_j \mathcal{N}(\mathbf{x}_n | \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)}{\sum_{k=1}^K \pi_k \mathcal{N}(\mathbf{x}_n | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)} \quad \forall j = 1, \dots, K, n = 1, \dots, N$$

- **M-Step:** re-estimate parameters of each component based on the soft assignments:

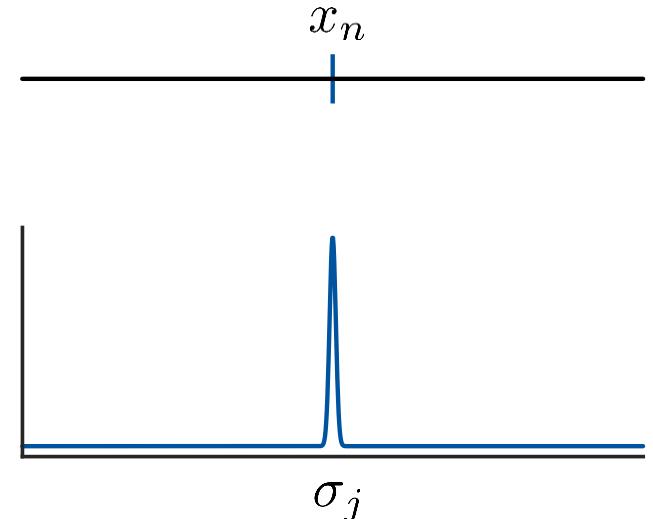
$$\begin{aligned} \hat{N}_j &\leftarrow \sum_{n=1}^N \gamma_j(\mathbf{x}_n) & \hat{\boldsymbol{\mu}}_j^{\text{new}} &\leftarrow \frac{1}{\hat{N}_j} \sum_{n=1}^N \gamma_j(\mathbf{x}_n) \mathbf{x}_n \\ \hat{\pi}_j^{\text{new}} &\leftarrow \frac{\hat{N}_j}{N} & \hat{\boldsymbol{\Sigma}}_j^{\text{new}} &\leftarrow \frac{1}{\hat{N}_j} \sum_{n=1}^N \gamma_j(\mathbf{x}_n) (\mathbf{x}_n - \hat{\boldsymbol{\mu}}_j^{\text{new}}) (\mathbf{x}_n - \hat{\boldsymbol{\mu}}_j^{\text{new}})^T \end{aligned}$$

Practical Advice

- When implementing EM, we need to take care to avoid singularities in the estimation!
 - Mixture components may collapse on single data points.
 - E.g. consider the case $\Sigma_k = \sigma_k^2 \mathbf{I}$ (this also holds in general)
 - Assume component j is exactly centered on data point \mathbf{x}_n . This data point will then contribute a term in the likelihood function

$$\mathcal{N}(\mathbf{x}_n | \mathbf{x}_n, \sigma_j^2 \mathbf{I}) = \frac{1}{\sqrt{2\pi}\sigma_j}$$

- For $\sigma_j \rightarrow 0$, this term goes to infinity!
- We need to introduce **regularization** to avoid this.
 - Enforce minimum width for the Gaussians



Instead of Σ^{-1} , use $(\Sigma + \sigma_{\min} \mathbf{I})^{-1}$.

Discussion: Mixture Models

Advantages

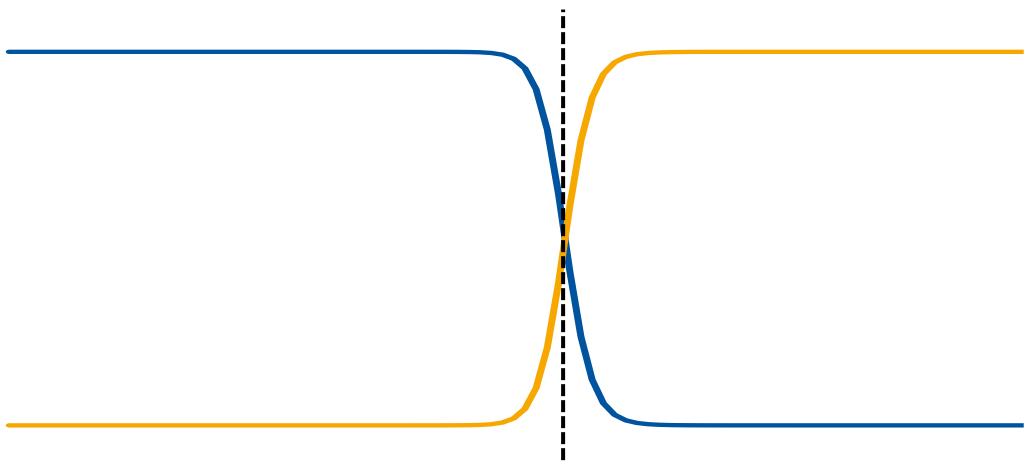
- Very general, can represent any continuous distribution.
- Once trained, is very fast to evaluate.

Limitations

- Need to apply regularization to avoid numerical instabilities.
- Choosing the right number of mixture components is hard.
- The EM algorithm is computationally expensive.
 - Especially for high-dim. problems.
 - Very sensitive to initialization.
 - *Practical Tip:* Run k-Means first and initialize clusters with k-Means result

Probability Density Estimation

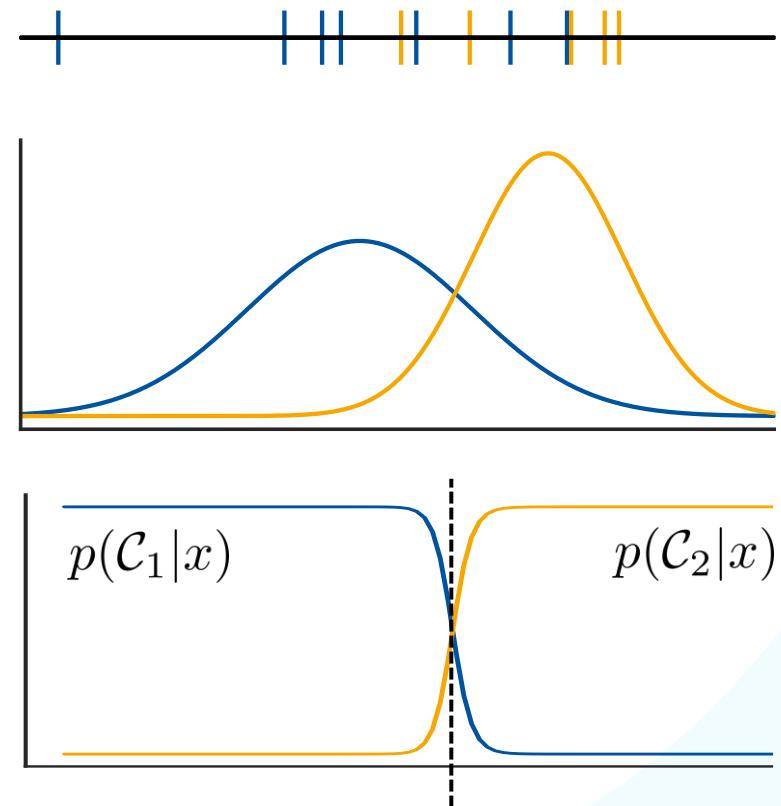
1. Probability Distributions
2. Parametric Methods
3. Nonparametric Methods
4. Mixture Models
- 5. Bayes Classifier**
6. K-NN Classifier



Bayes Classifier

- We know how to estimate probability densities from data.
- We can now use [Bayes Decision Theory](#) to build a classifier:
 - Estimate likelihoods & priors from data.
 - Calculate posterior with Bayes' Theorem.
 - Decide for class with highest posterior probability:

$$p(\mathcal{C}_1|x) > p(\mathcal{C}_2|x)$$



Likelihood-Ratio Test

- Assume we want to classify an observation x into one of two classes $\mathcal{C}_1, \mathcal{C}_2$.

- Decide for \mathcal{C}_1 if

$$p(\mathcal{C}_1|x) > p(\mathcal{C}_2|x)$$

- This is equivalent to

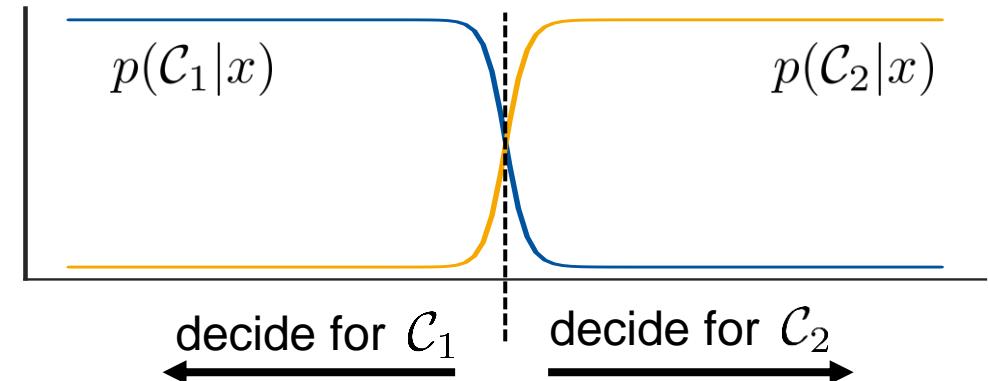
$$p(x|\mathcal{C}_1)p(\mathcal{C}_1) > p(x|\mathcal{C}_2)p(\mathcal{C}_2)$$

- Which again is equivalent to

$$\frac{p(x|\mathcal{C}_1)}{p(x|\mathcal{C}_2)} > \frac{p(\mathcal{C}_2)}{p(\mathcal{C}_1)}$$



Decision threshold θ



$$p(\mathcal{C}|x) = \frac{p(x|\mathcal{C})p(\mathcal{C})}{p(x)}$$

Decision Functions

- We can find a decision function based on probability densities.
 - Determine class-conditional densities $p(x|\mathcal{C}_k)$ for each class individually.
 - Separately infer the prior class probabilities $p(\mathcal{C}_k)$.
 - Then use Bayes' theorem and/or the likelihood-ratio test.
- Alternative: solve the inference problem of determining the posterior class probabilities directly.
 - Then use Bayes' decision theory to assign each new observation to its class.

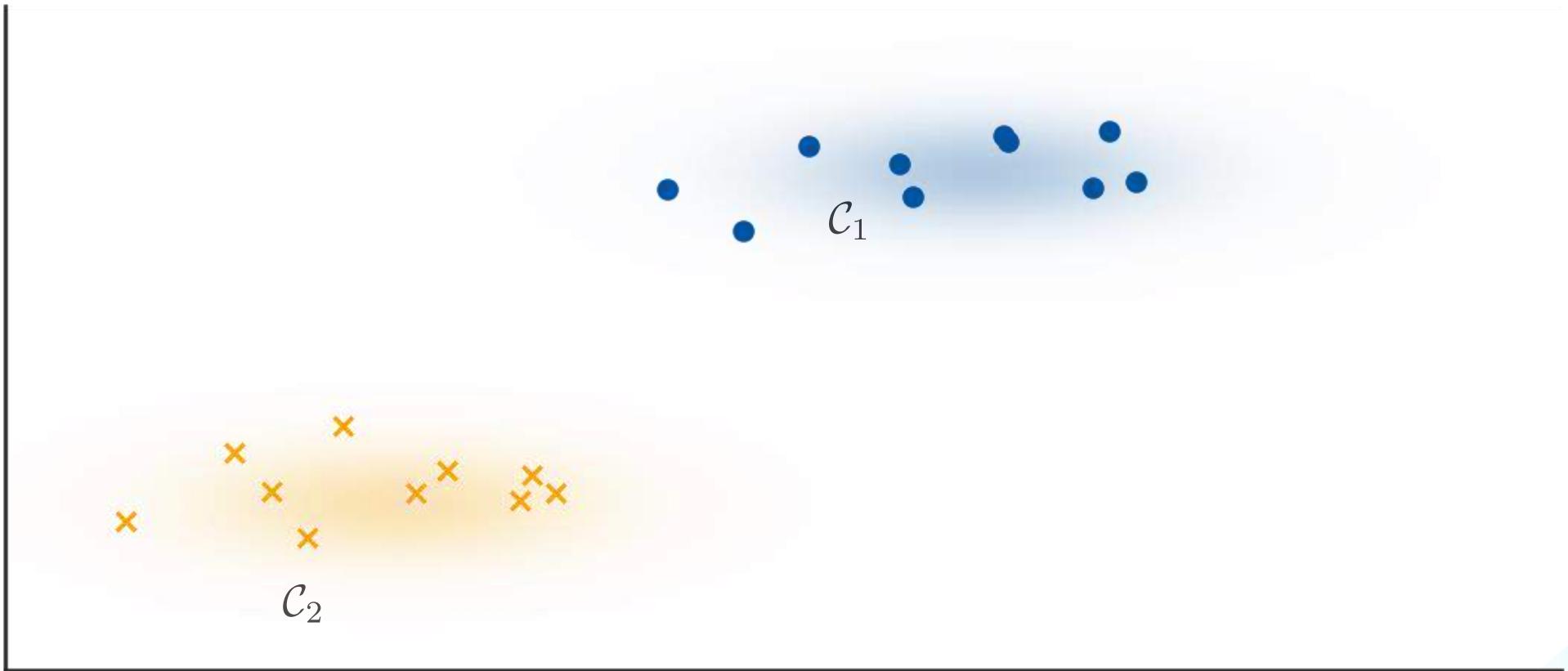
Generative methods:

$$y_k(x) \propto p(x|\mathcal{C}_k)p(\mathcal{C}_k)$$

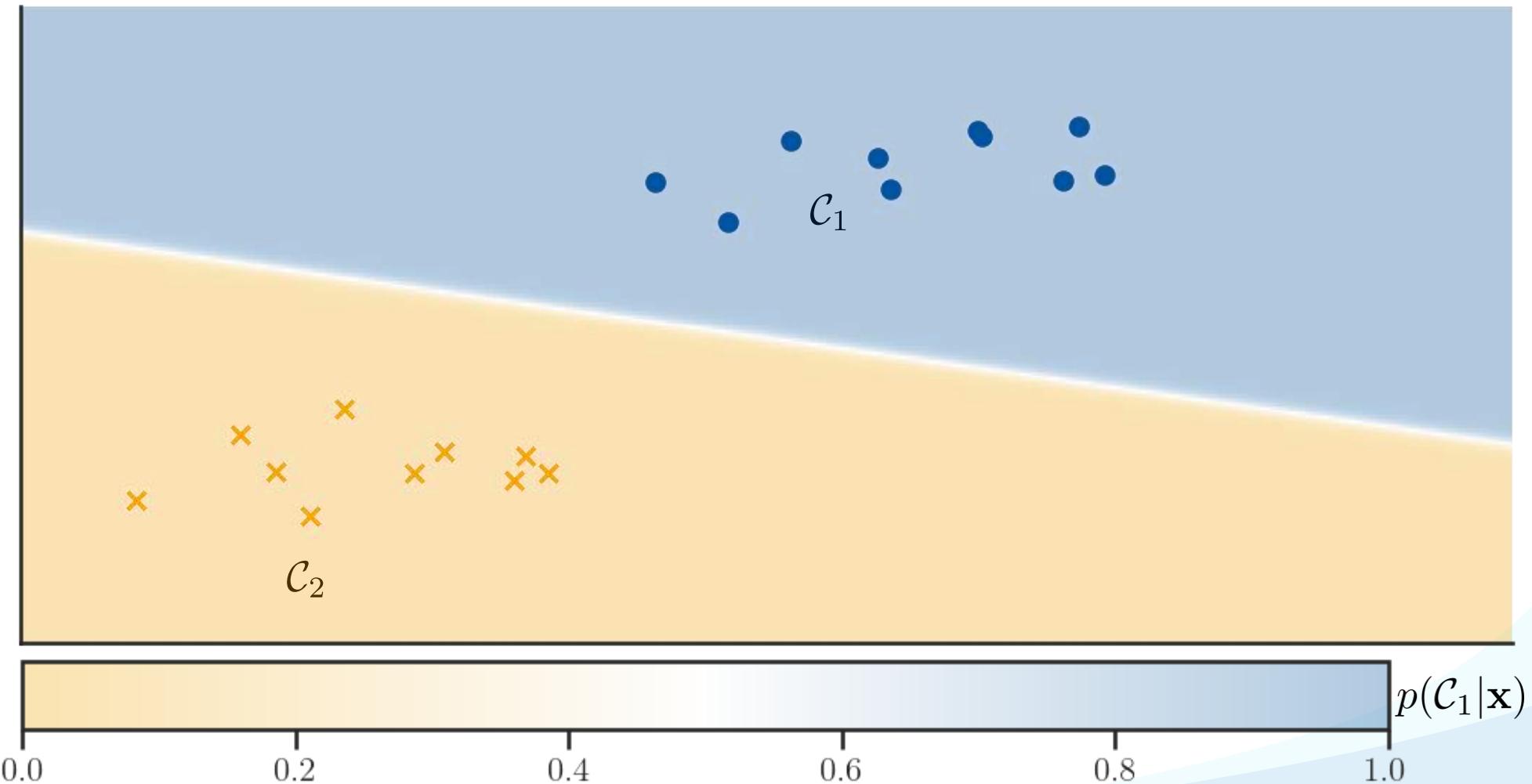
Discriminative methods:

$$y_k(x) = p(\mathcal{C}_k|x)$$

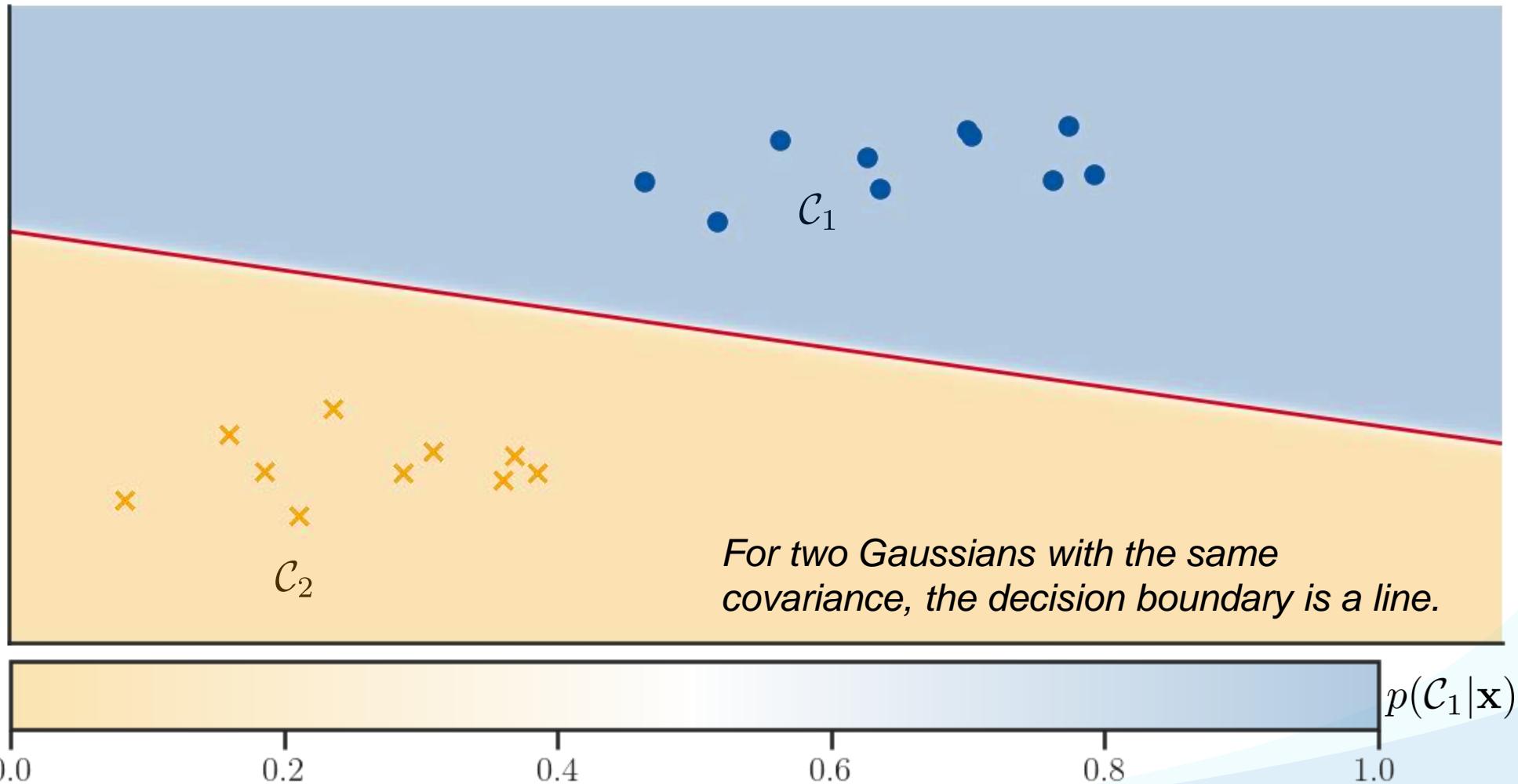
Example



Example

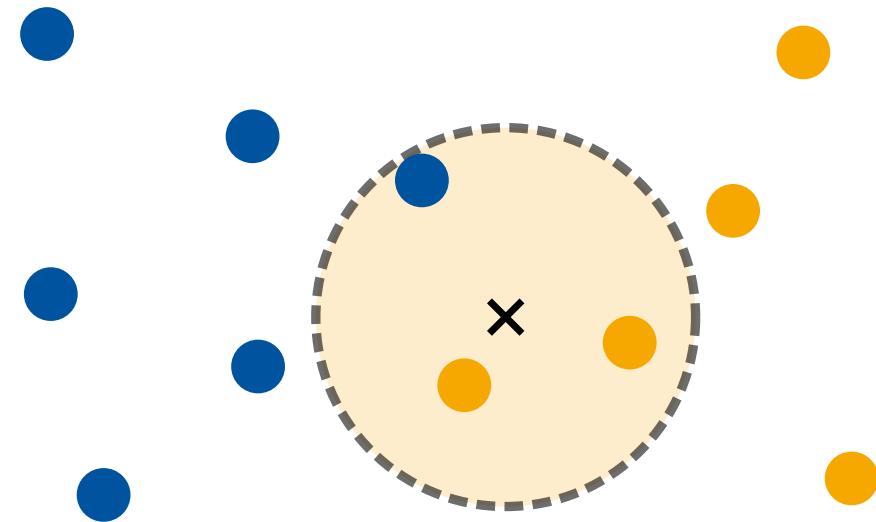


Example



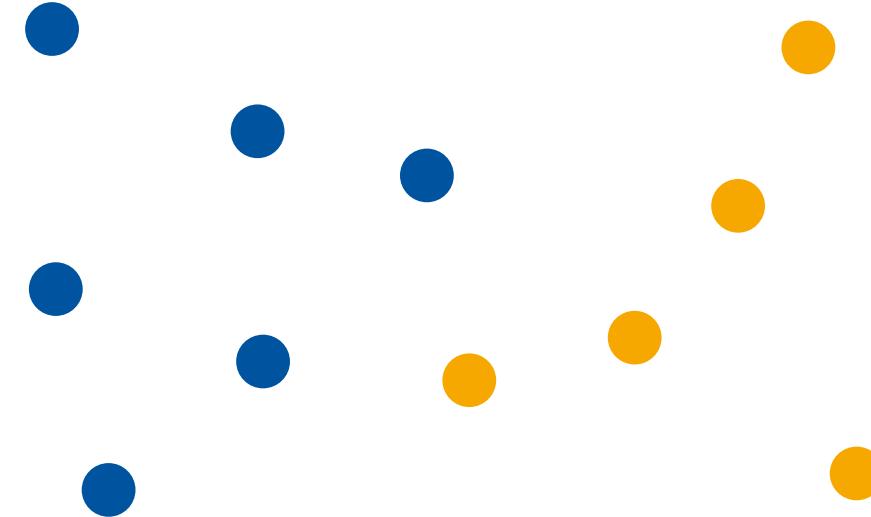
Probability Density Estimation

1. Probability Distributions
2. Parametric Methods
3. Nonparametric Methods
4. Mixture Models
5. Bayes Classifier
6. **K-NN Classifier**



K-NN Classifier

- Combine K-NN density estimation with Bayes Decision Theory: [K-NN Classifier](#)



K-NN Classifier

- Combine K-NN density estimation with Bayes Decision Theory: [K-NN Classifier](#)

- Determine the class-conditional densities

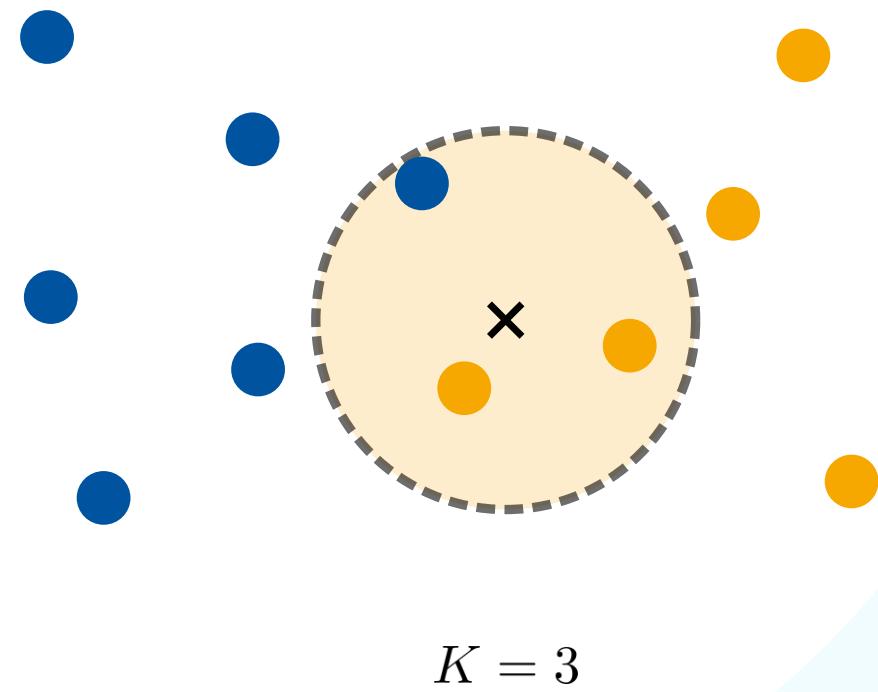
$$p(\mathbf{x}|\mathcal{C}_j) \approx \frac{K_j}{N_j V} \quad p(\mathbf{x}) \approx \frac{K}{NV}$$

- Determine the prior probabilities

$$p(\mathcal{C}_j) \approx \frac{N_j}{N}$$

- Use Bayes' theorem to compute the posterior

$$p(\mathcal{C}_j|\mathbf{x}) \approx p(\mathbf{x}|\mathcal{C}_j)p(\mathcal{C}_j) \frac{1}{p(\mathbf{x})}$$



K-NN Classifier

- Combine K-NN density estimation with Bayes Decision Theory: [K-NN Classifier](#)

- Determine the class-conditional densities

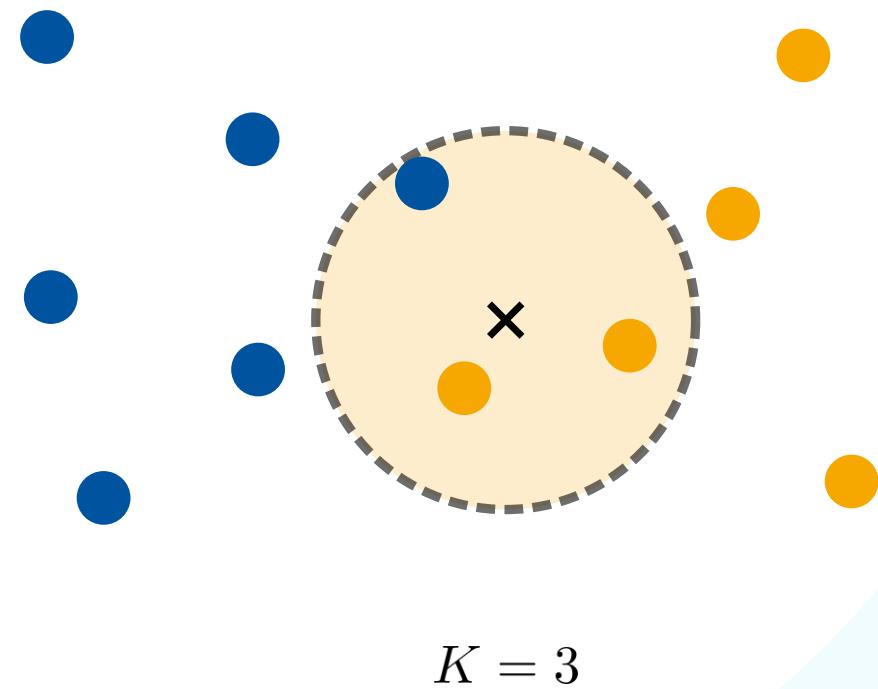
$$p(\mathbf{x}|\mathcal{C}_j) \approx \frac{K_j}{N_j V} \quad p(\mathbf{x}) \approx \frac{K}{NV}$$

- Determine the prior probabilities

$$p(\mathcal{C}_j) \approx \frac{N_j}{N}$$

- Use Bayes' theorem to compute the posterior

$$p(\mathcal{C}_j|\mathbf{x}) \approx \frac{K_j}{N_j V} p(\mathcal{C}_j) \frac{1}{p(\mathbf{x})}$$



K-NN Classifier

- Combine K-NN density estimation with Bayes Decision Theory: [K-NN Classifier](#)

- Determine the class-conditional densities

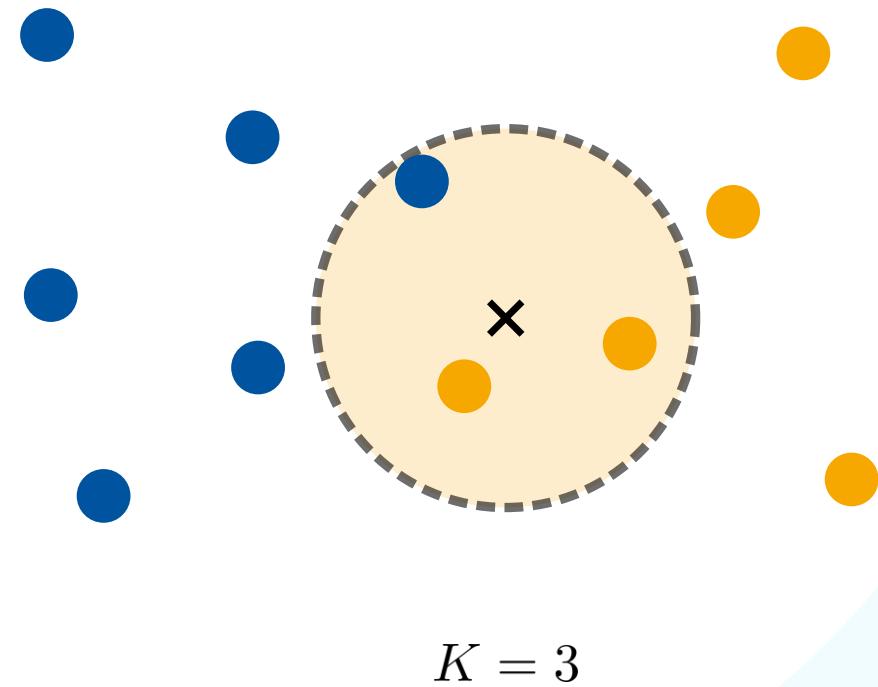
$$p(\mathbf{x}|\mathcal{C}_j) \approx \frac{K_j}{N_j V} \quad p(\mathbf{x}) \approx \frac{K}{NV}$$

- Determine the prior probabilities

$$p(\mathcal{C}_j) \approx \frac{N_j}{N}$$

- Use Bayes' theorem to compute the posterior

$$p(\mathcal{C}_j|\mathbf{x}) \approx \frac{K_j}{N_j V} \frac{N_j}{N} \frac{1}{p(\mathbf{x})}$$



K-NN Classifier

- Combine K-NN density estimation with Bayes Decision Theory: [K-NN Classifier](#)

- Determine the class-conditional densities

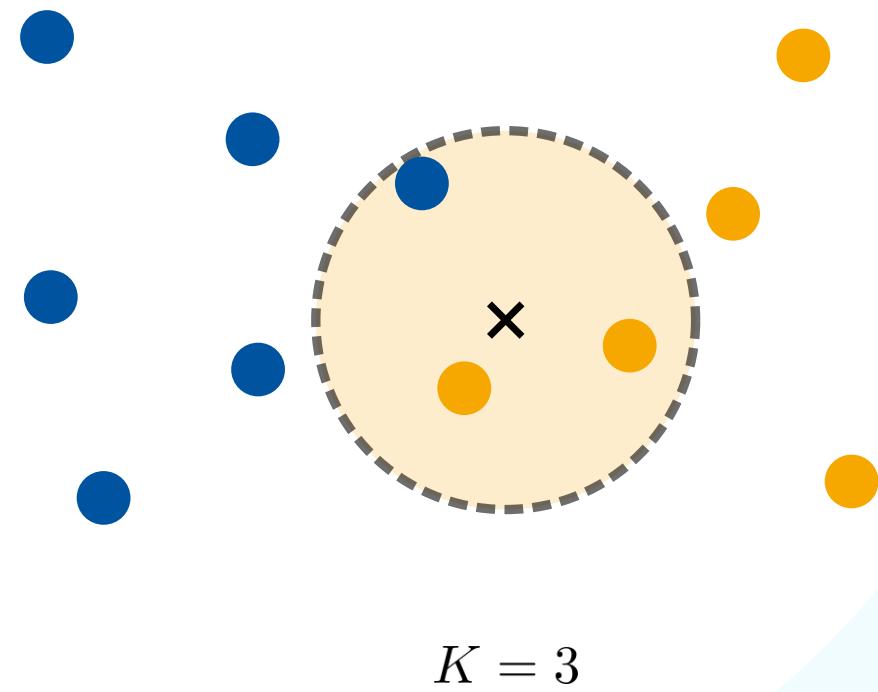
$$p(\mathbf{x}|\mathcal{C}_j) \approx \frac{K_j}{N_j V} \quad p(\mathbf{x}) \approx \frac{K}{NV}$$

- Determine the prior probabilities

$$p(\mathcal{C}_j) \approx \frac{N_j}{N}$$

- Use Bayes' theorem to compute the posterior

$$p(\mathcal{C}_j|\mathbf{x}) \approx \frac{K_j}{N_j V} \frac{N_j}{N} \frac{NV}{K}$$



K-NN Classifier

- Combine K-NN density estimation with Bayes Decision Theory: [K-NN Classifier](#)

- Determine the class-conditional densities

$$p(\mathbf{x}|\mathcal{C}_j) \approx \frac{K_j}{N_j V} \quad p(\mathbf{x}) \approx \frac{K}{NV}$$

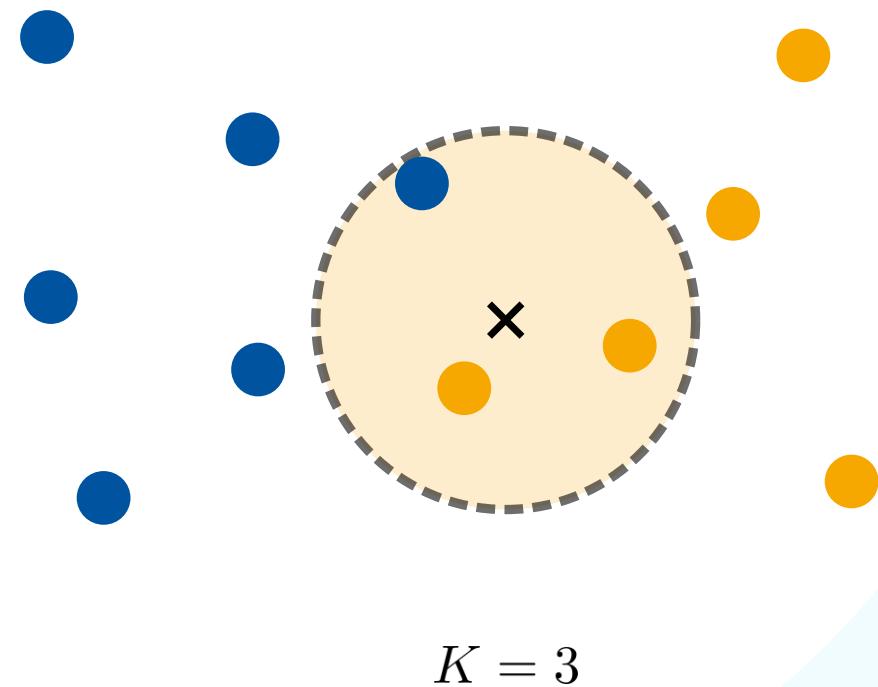
- Determine the prior probabilities

$$p(\mathcal{C}_j) \approx \frac{N_j}{N}$$

- Use Bayes' theorem to compute the posterior

$$p(\mathcal{C}_j|\mathbf{x}) \approx \frac{K_j}{N_j V} \frac{N_j}{N} \frac{NV}{K} = \frac{K_j}{K}$$

\Rightarrow Decide for the majority class among the neighbors.

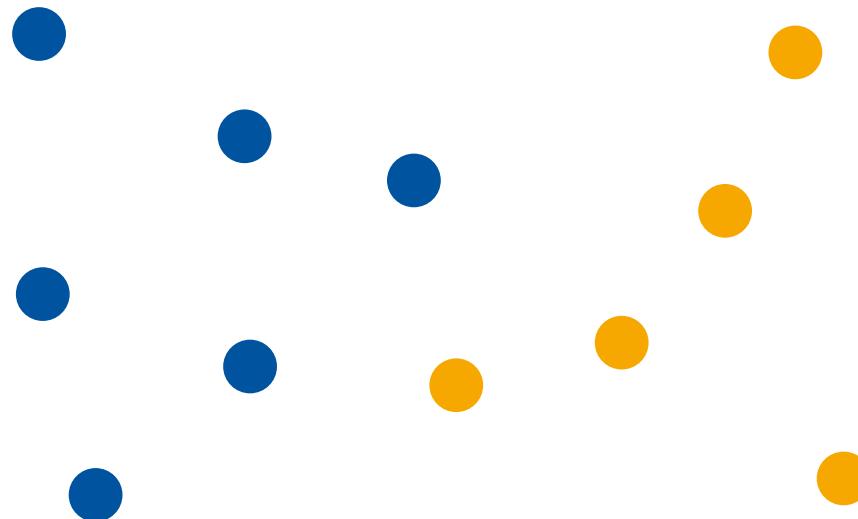


K-NN Classifier

- Combine K-NN density estimation with Bayes Decision Theory: [K-NN Classifier](#)
- Algorithm

Given a new sample x :

1. Find the K training samples with the smallest distance to x .
2. Assign the majority label of those samples to x .

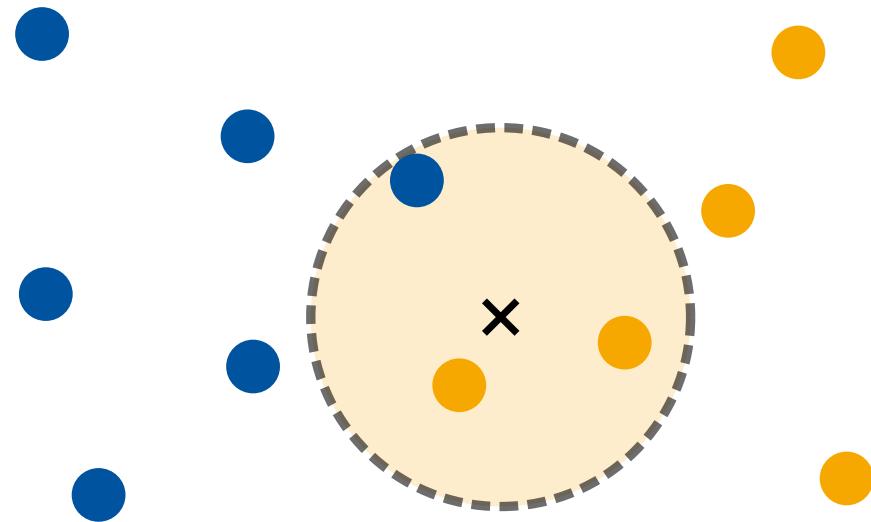


K-NN Classifier

- Combine K-NN density estimation with Bayes Decision Theory: [K-NN Classifier](#)
- Algorithm

Given a new sample \mathbf{x} :

1. Find the K training samples with the smallest distance to \mathbf{x} .
2. Assign the majority label of those samples to \mathbf{x} .



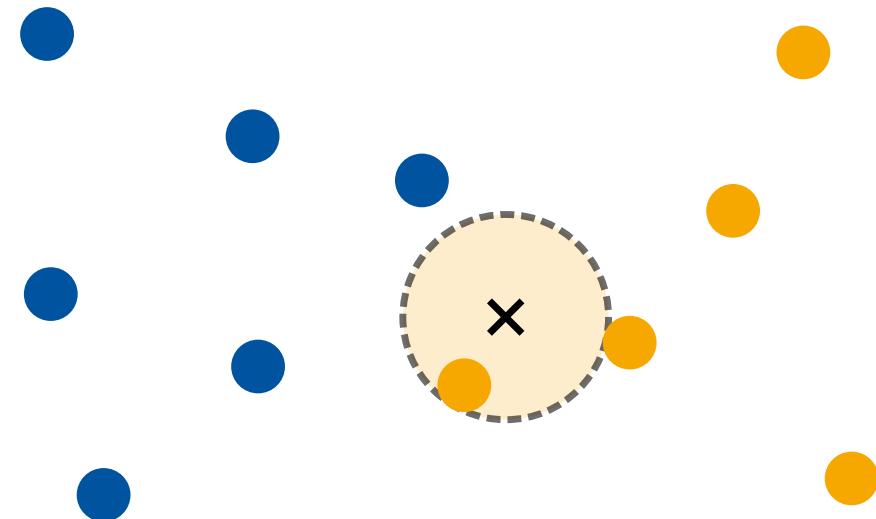
K-NN Classifier

- Combine K-NN density estimation with Bayes Decision Theory: [K-NN Classifier](#)
- Algorithm

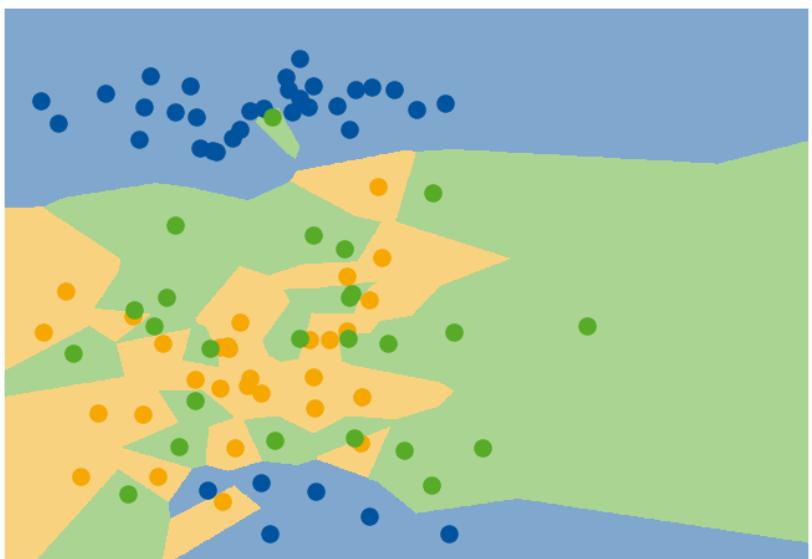
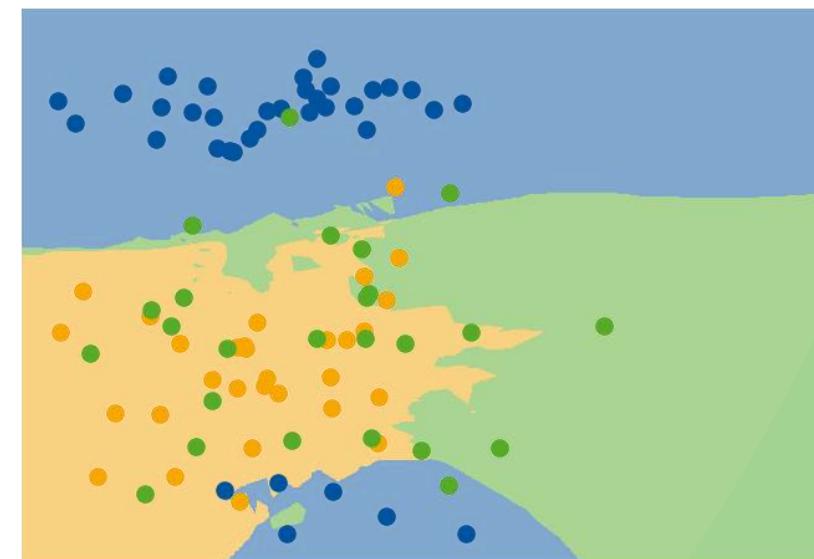
Given a new sample \mathbf{x} :

1. Find the K training samples with the smallest distance to \mathbf{x} .
 2. Assign the majority label of those samples to \mathbf{x} .
- Special case: 1-NN Classifier.

Theoretical guarantee: Never worse than 2x the error of the optimal classifier!



Example

 $K = 1$  $K = 3$  $K = 15$

Discussion: K-NN Classifier

Advantages

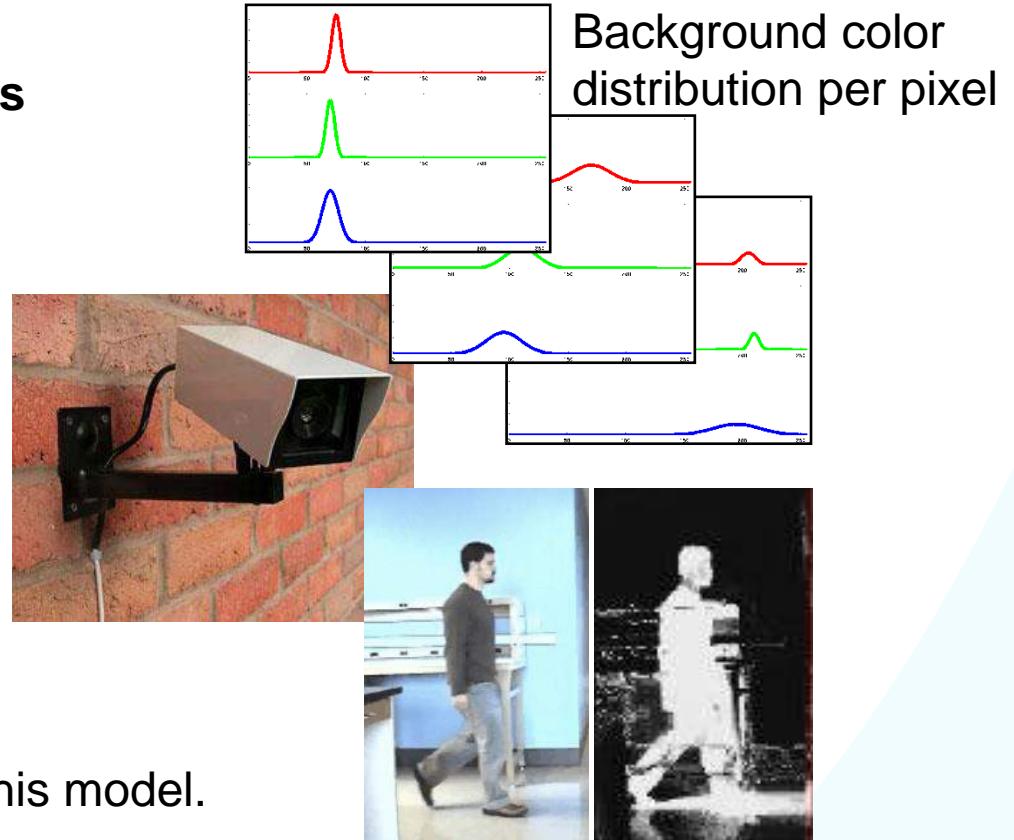
- Very simple, Bayes-optimal classifier.
- Needs no training.
- Can always be used as first estimate when working with a new dataset.
- Theoretical guarantees
 - Never worse than 2x optimal error

Limitations

- Requires storing the complete training set.
- Finding the k-nearest neighbors can become very expensive in high-dimensional spaces
- Theoretical optimality bound is often too loose to be of practical value.

Application Example: Background Models for Tracking

- **Example: Object tracking in static surveillance cameras**
 - Want to know if anybody enters a forbidden area
 - Challenge: many possible moving objects
- Idea: Train background color model for each pixel
 - Initialize with an empty scene.
 - Learn “common” appearance variation for each background pixel, e.g., by fitting a Gaussian distribution to the observed noise over several frames.
 - Evaluate the likelihood of observed pixel colors under this model.
⇒ *Anything that cannot be explained by the background model is labeled as foreground (=object).*



Application Example: Background Models for Tracking

- Problem: Outdoor scenes
 - Dynamic areas
 - Waving trees, rippling water, ...
- ⇒ *More flexible representation needed here!*
- Idea:
 - Use Kernel Density Estimation using the observed pixel values over a temporal window to model the “background” distribution for each pixel.
 - Again, evaluate the likelihood of the observed pixel color under this background model to detect “foreground” objects.

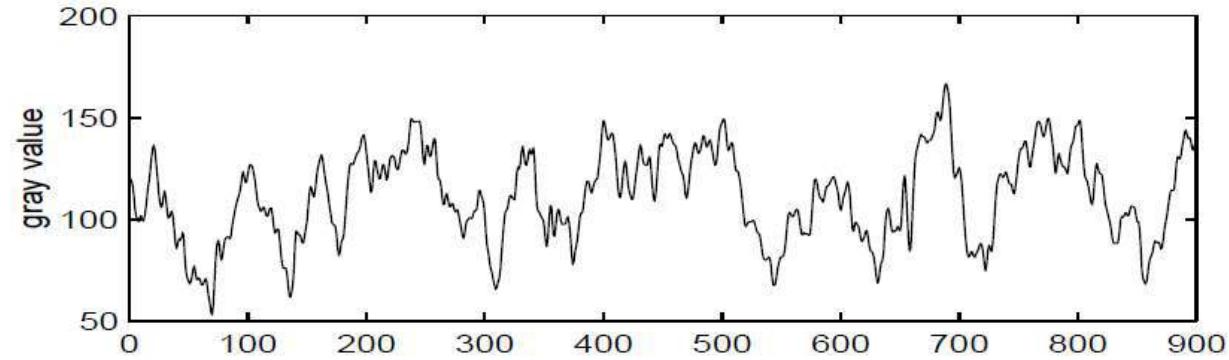


Image & Video source: A. Elgammal

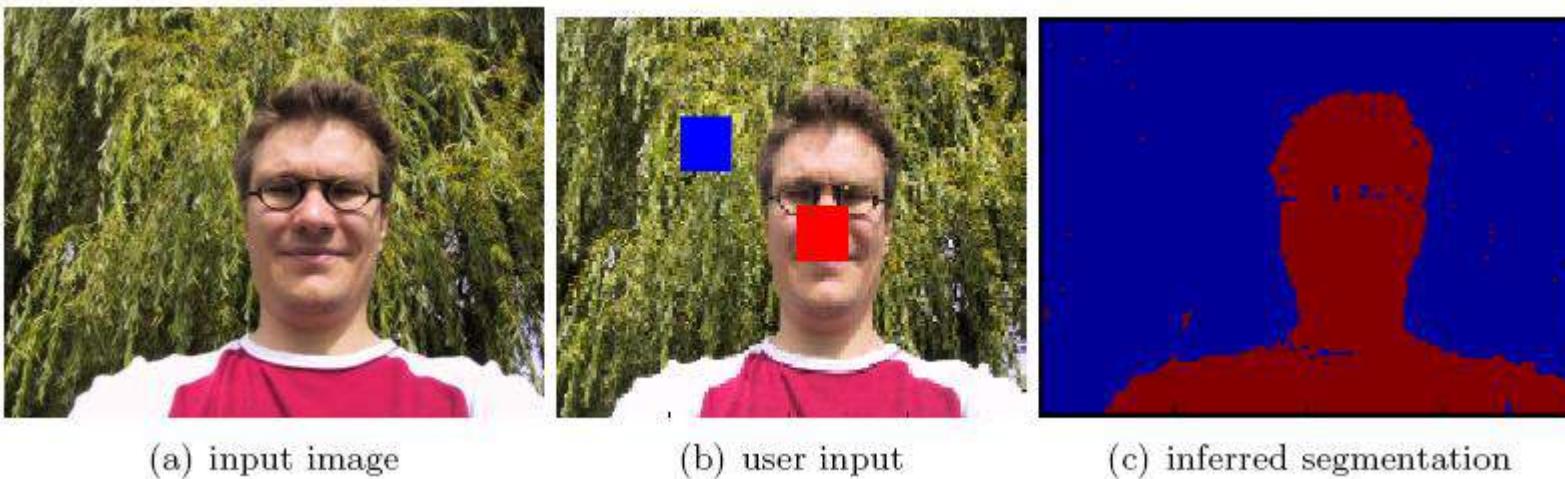
Application Example: Background Models for Tracking

- Results
 - Very robust foreground object detection in dynamic scenes
 - Automatic adaptation to varying weather conditions through temporal window



Video source: A. Elgammal

Application Example: Image Segmentation

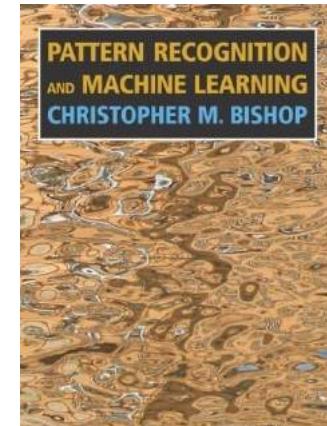


- **Example: User assisted image segmentation**
 - User marks two regions for foreground and background.
 - Learn a MoG model for the color values in each region.
 - Use those models to classify all other pixels with a Bayes classifier (likelihood ratio test)
- ⇒ Simple, but effective segmentation procedure

References and Further Reading

- More information about EM and MoG estimation is available in Chapter 2.3.9 and the entire Chapter 9 of Bishop's book.

Christopher M. Bishop
Pattern Recognition and Machine Learning
Springer, 2006



Elements of Machine Learning & Data Science

Introduction to Data Science

Lecture 6

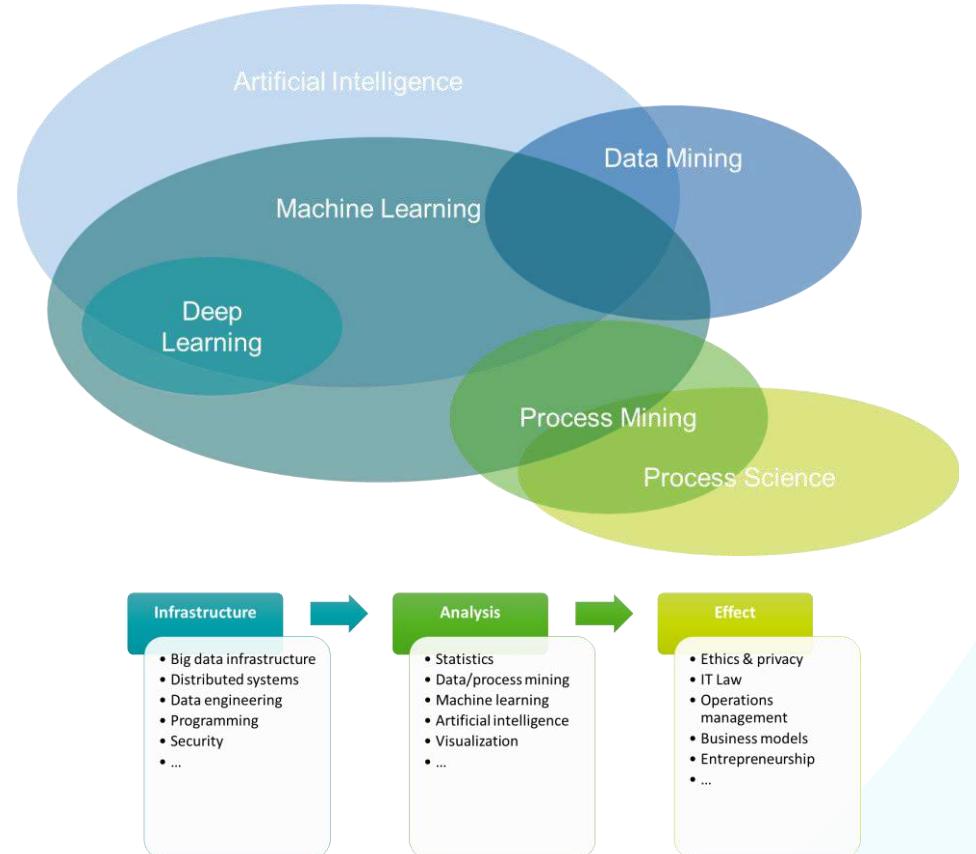
Prof. Wil van der Aalst

Marco Pegoraro, M.Sc.

Leah Tacke genannt Unterberg, M.Sc.

Outline

1. Introduction
2. Tabular Data
3. Data Science Process
4. Challenges
5. Data Types
6. Descriptive Statistics
7. Interpretative Pitfalls
8. Basic Visualizations
9. Feature Transformations
10. “How to lie with statistics”

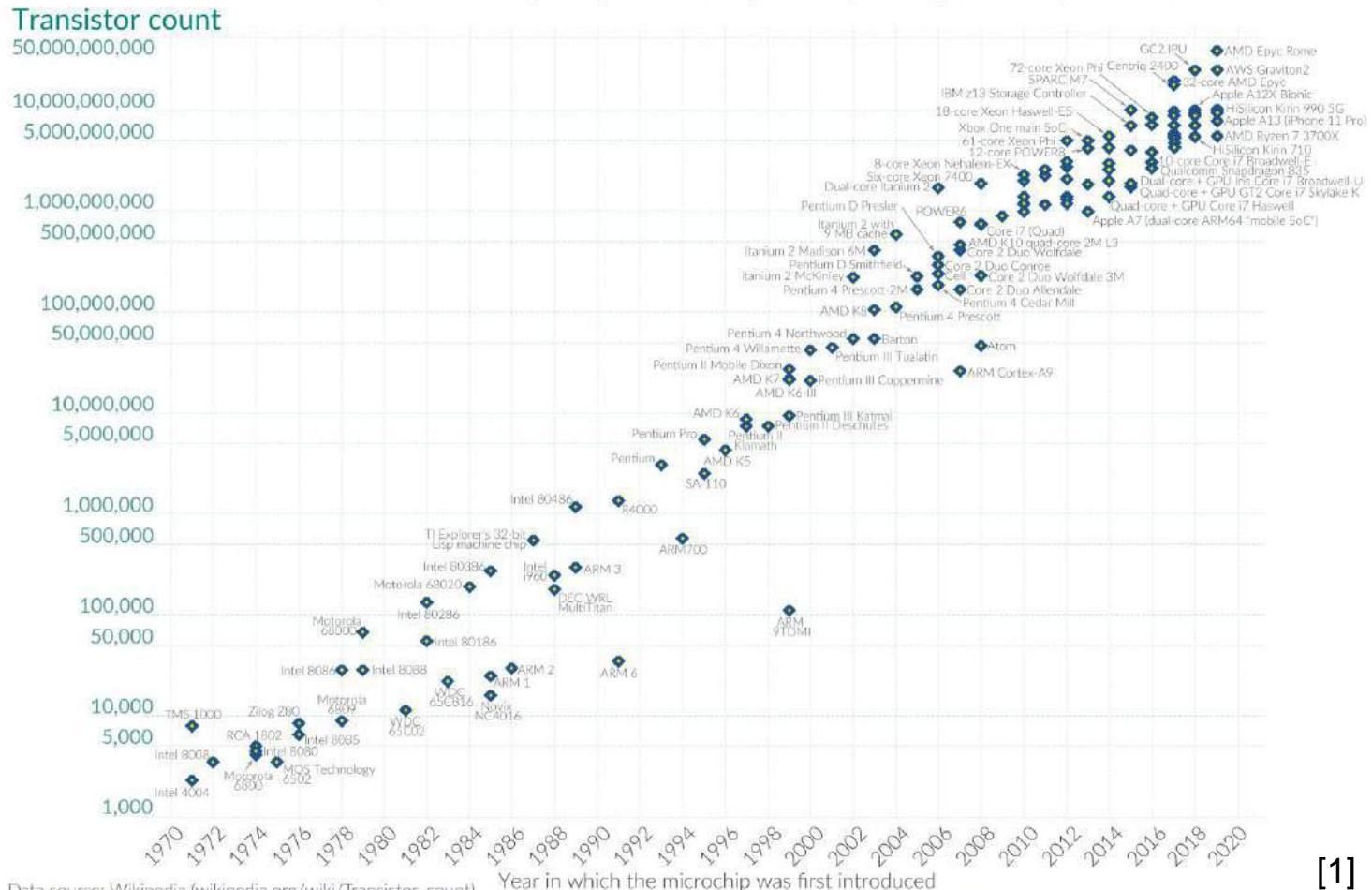


Motivation – Impact and Size of Data

Moore's Law: The number of transistors on microchips doubles every two years

Moore's law describes the empirical regularity that the number of transistors on integrated circuits doubles approximately every two years. This advancement is important for other aspects of technological progress in computing – such as processing speed or the price of computers.

Our World
in Data



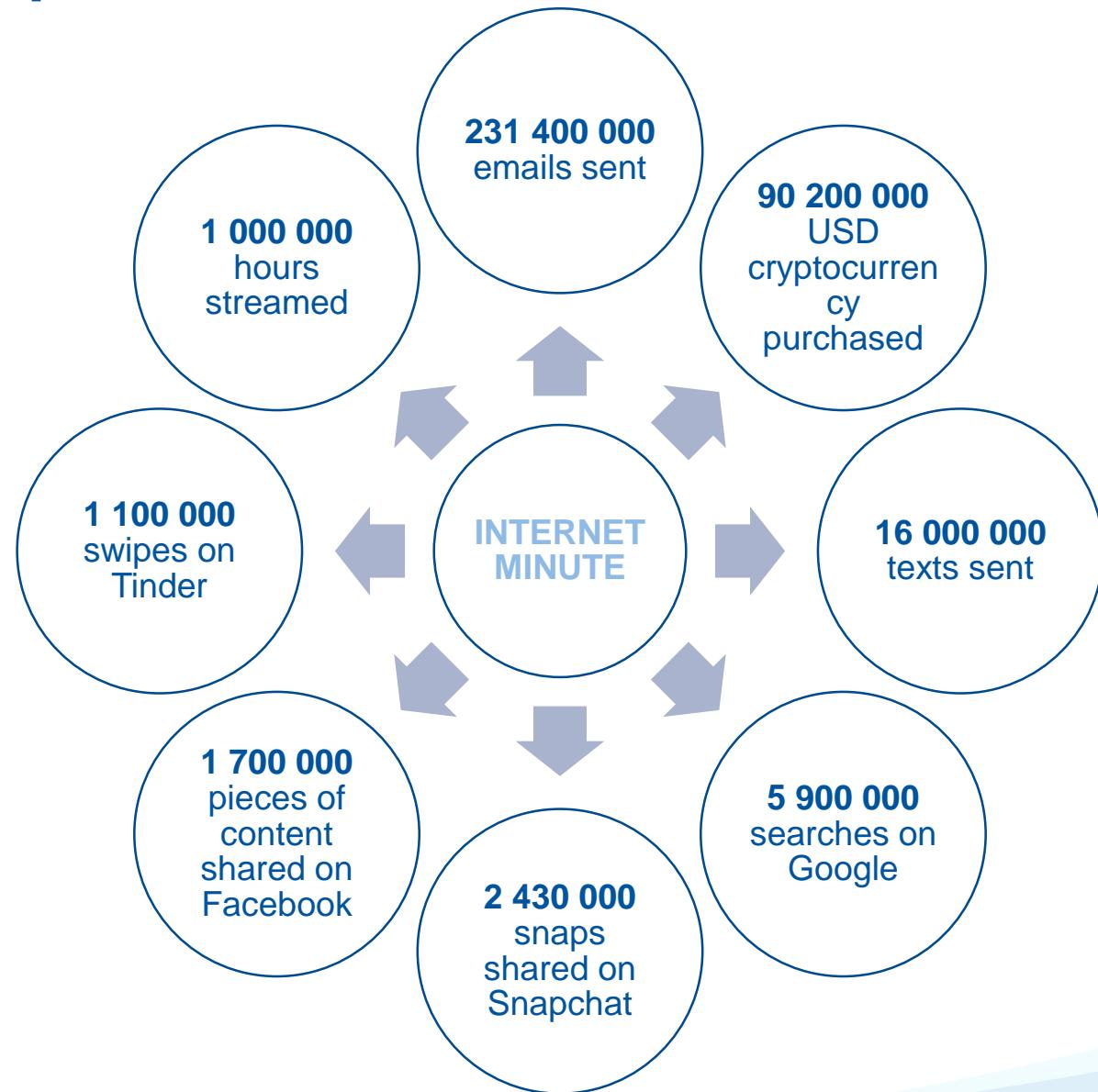
Data source: Wikipedia ([wikipedia.org/wiki/Transistor_count](https://en.wikipedia.org/w/index.php?title=Transistor_count&oldid=1000000000))

OurWorldInData.org – Research and data to make progress against the world's largest problems.

Licensed under CC-BY by the authors Hannah Ritchie and Max Roser.

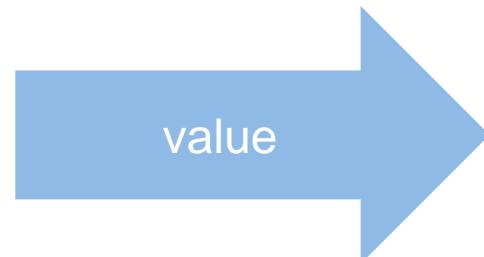
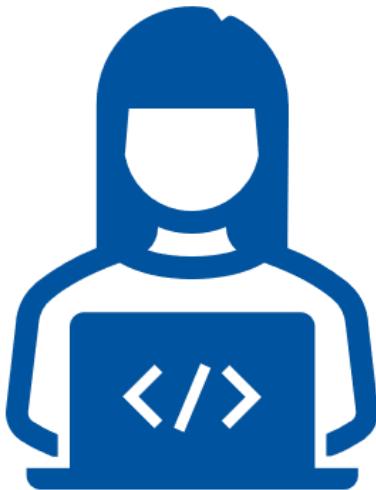
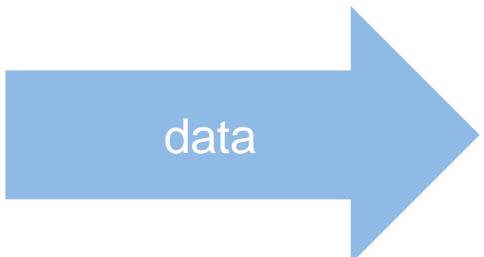
[1]

Motivation – Impact and Size of Data



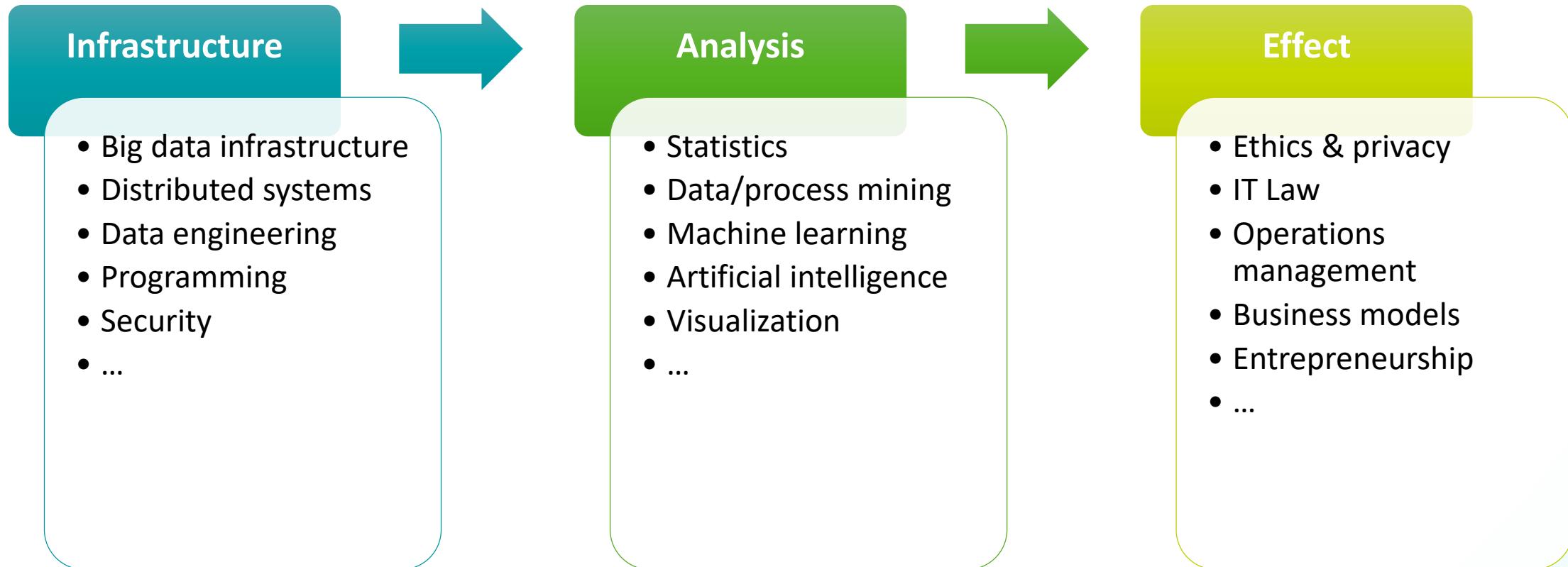
[2] Statista, as of 27.03.2023

Motivation – Data Scientist

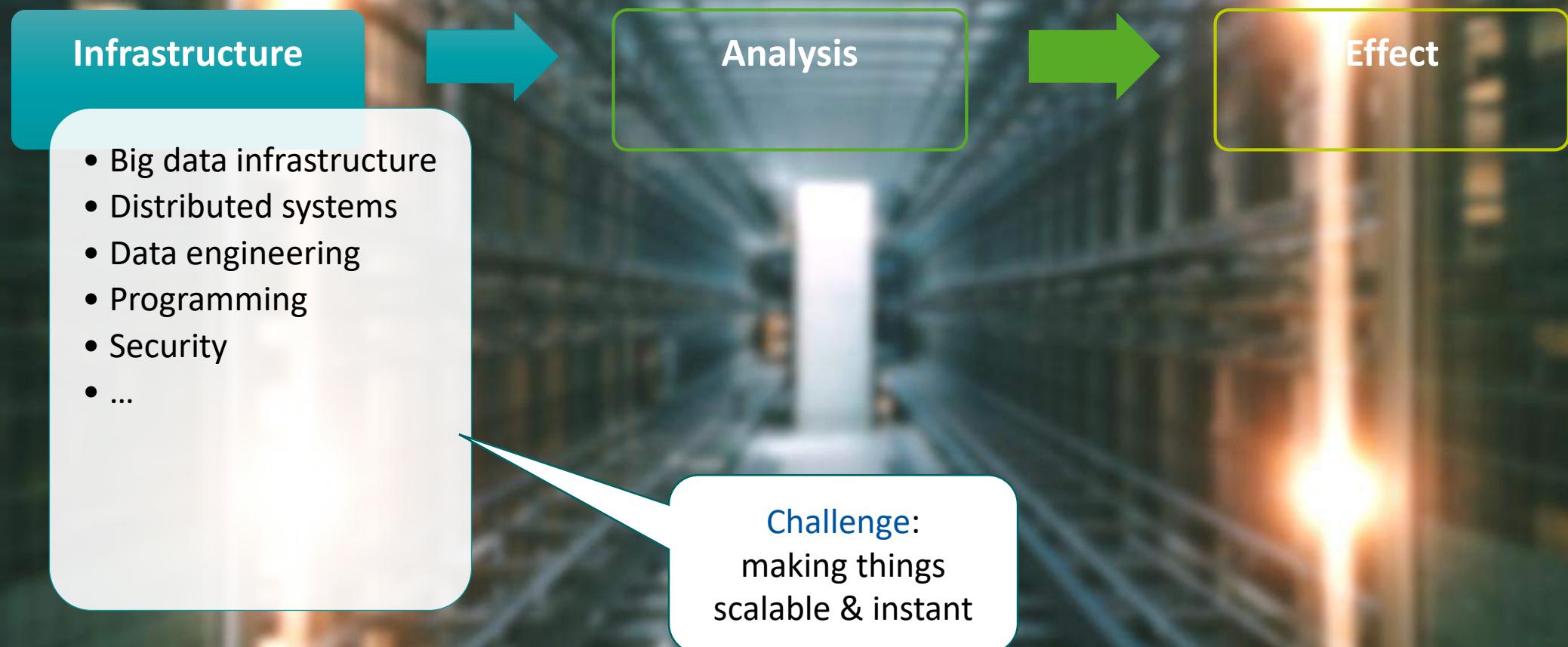


data scientist

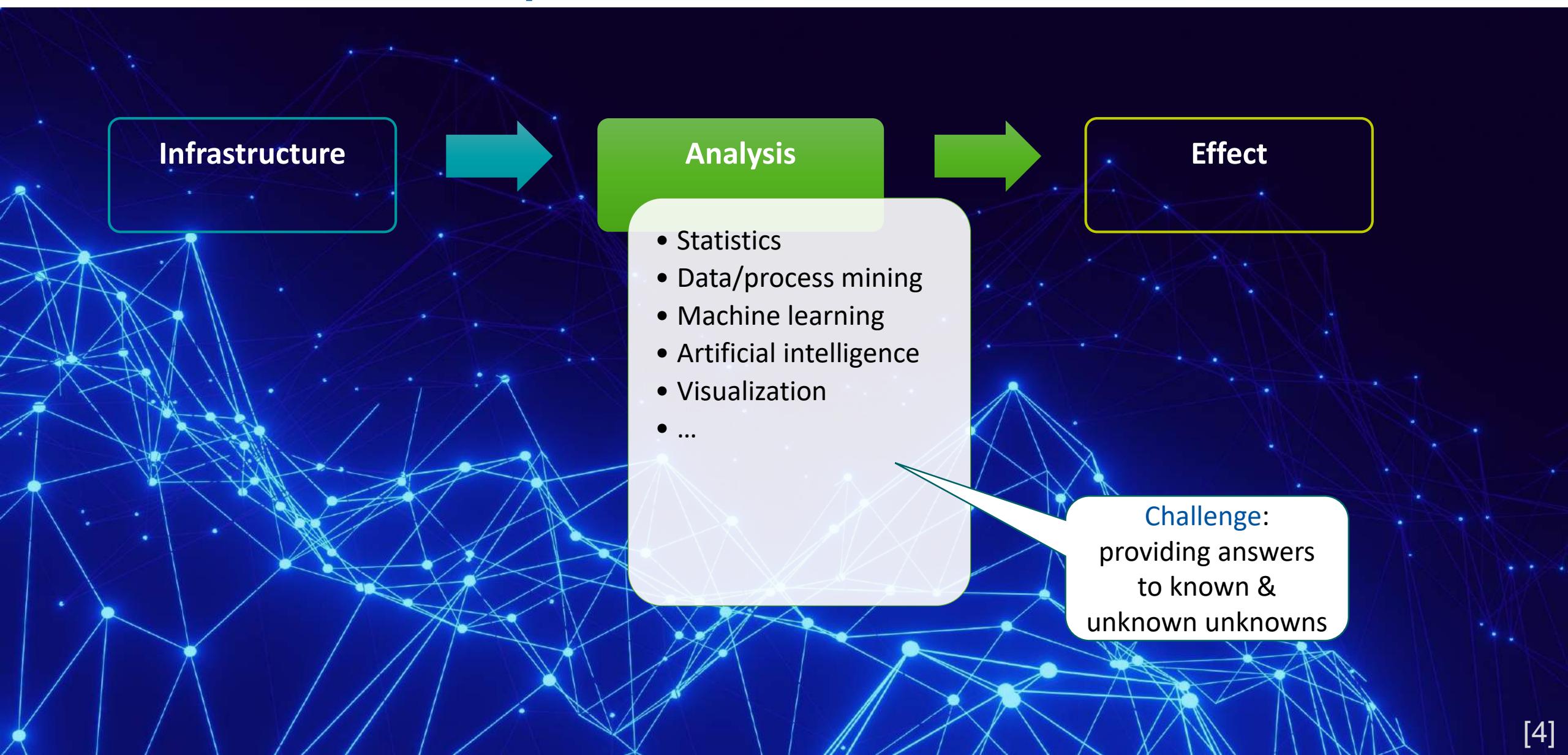
The Data Science Pipeline



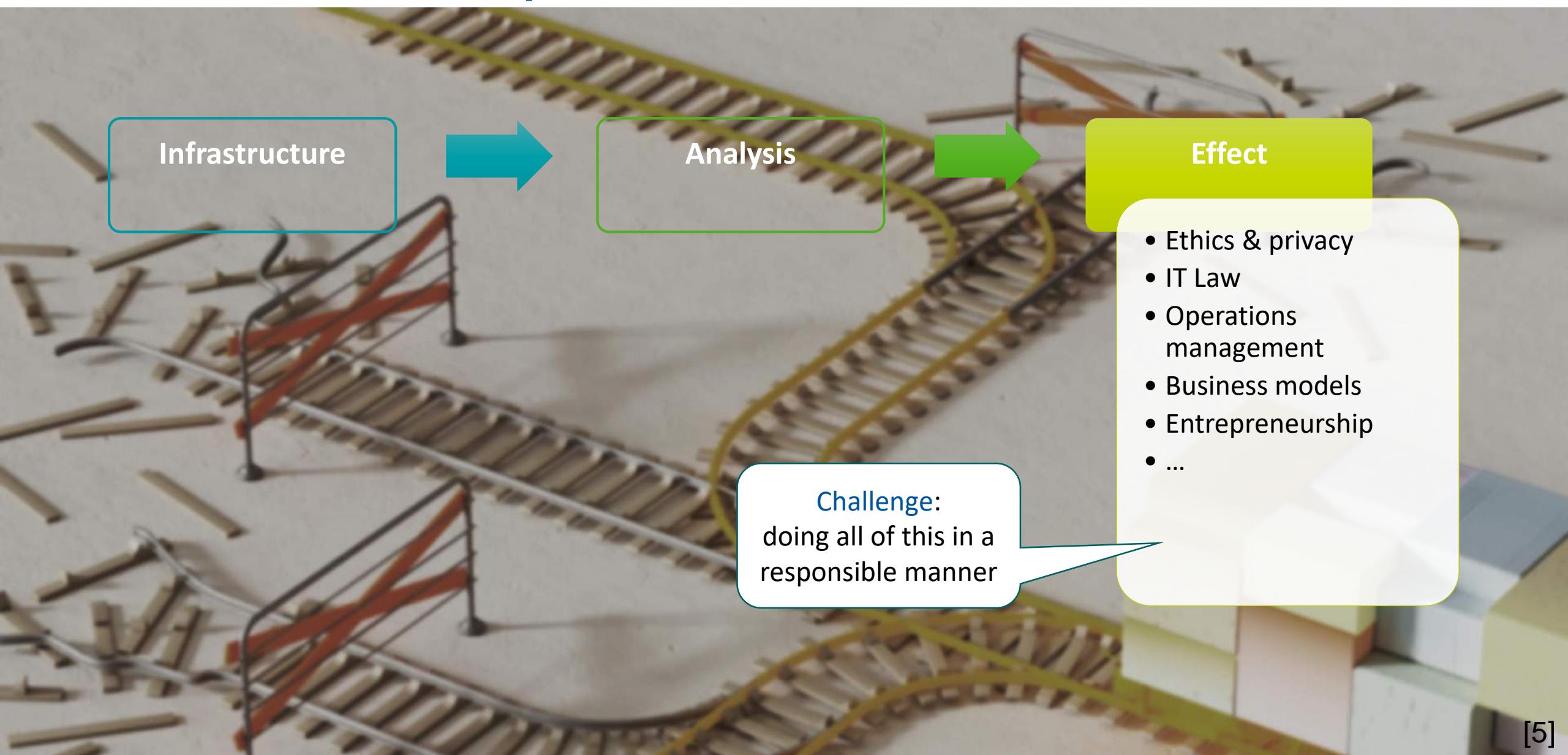
The Data Science Pipeline



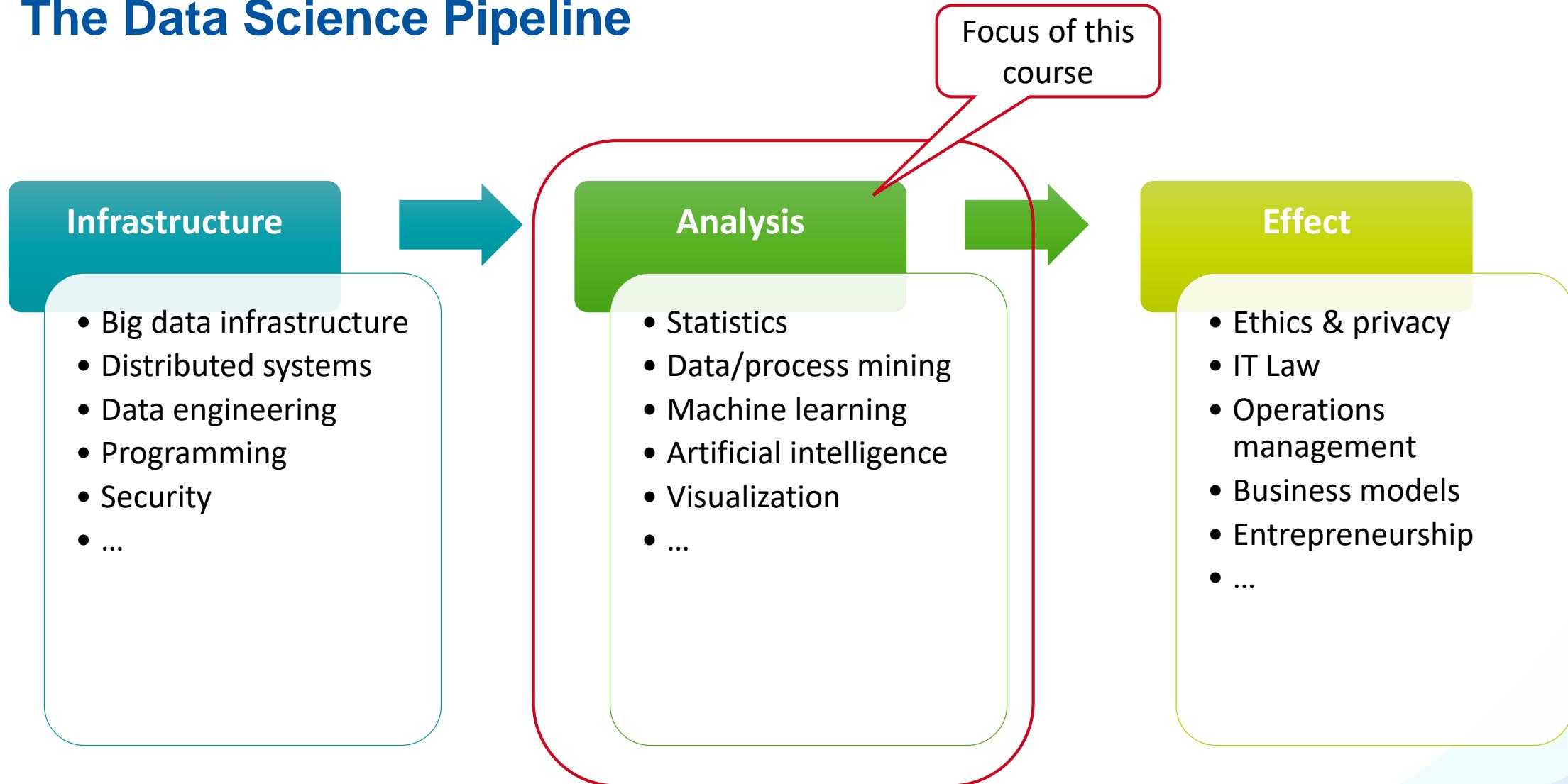
The Data Science Pipeline



The Data Science Pipeline



The Data Science Pipeline

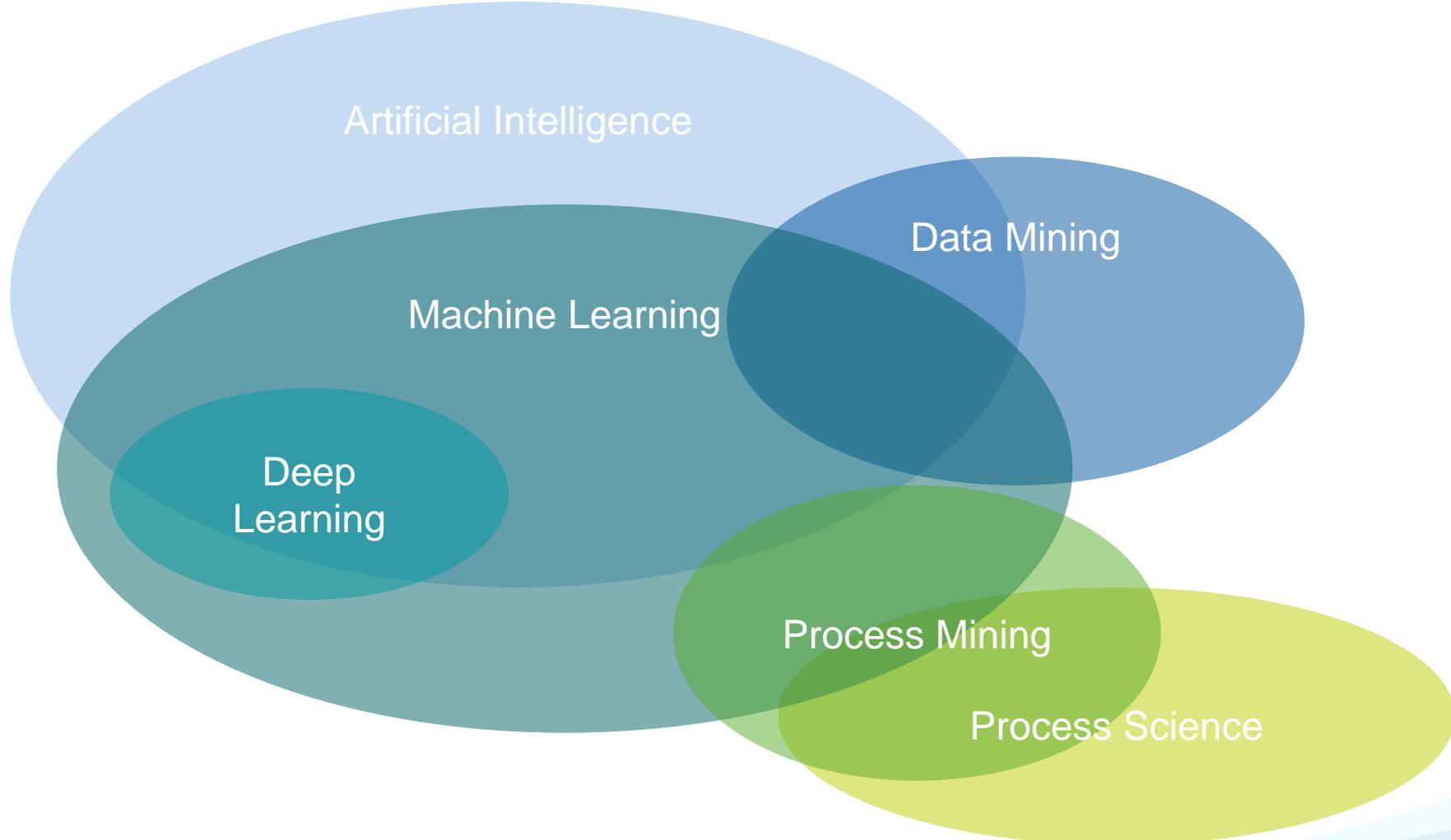


Terminology

- Many different names (statistics, data analytics, data mining, machine learning, artificial intelligence, predictive analytics, process mining, etc.) are used to refer to the key disciplines that contribute to data science
- Unfortunately, the areas these names describe are heavily overlapping and context dependent



Terminology



Data Science: A Definition

“Data science is an interdisciplinary field aiming to turn data into real value. Data may be structured or unstructured, big or small, static or streaming. Value may be provided in the form of predictions, automated decisions, models learned from data, or any type of data visualization delivering insights. Data science includes data extraction, data preparation, data exploration, data transformation, storage and retrieval, computing infrastructures, various types of mining and learning, presentation of explanations and predictions, and the exploitation of results taking into account ethical, social, legal, and business aspects.”

What actually is the *Data* in Data Science?

Example

- A restaurant owner wants to analyze the performance of their menu items ...
- You have collected the following data:

price	calories	vegetarian	spicy	bestseller
12.99	800	Yes	No	Yes
9.99	600	Yes	Yes	No
14.99	1000	No	Yes	No
11.99	700	No	No	Yes
8.99	500	Yes	No	No

Features

- Features are **raw** or **derived** (mean, median, max, min, rank, etc.)
- **Time** is a special feature:
 - It cannot decrease
 - We often want to predict the future based on the past
 - Vital in temporal data analysis (time series data, event data, sequential data, ...)

Example – Unlabeled Data

Unlabeled – no target feature selected

instances	features				
	price	calories	vegetarian	spicy	bestseller
	12.99	800	Yes	No	Yes
	9.99	600	Yes	Yes	No
	14.99	1000	No	Yes	No
	11.99	700	No	No	Yes
	8.99	500	Yes	No	No

Example – Labeled Data

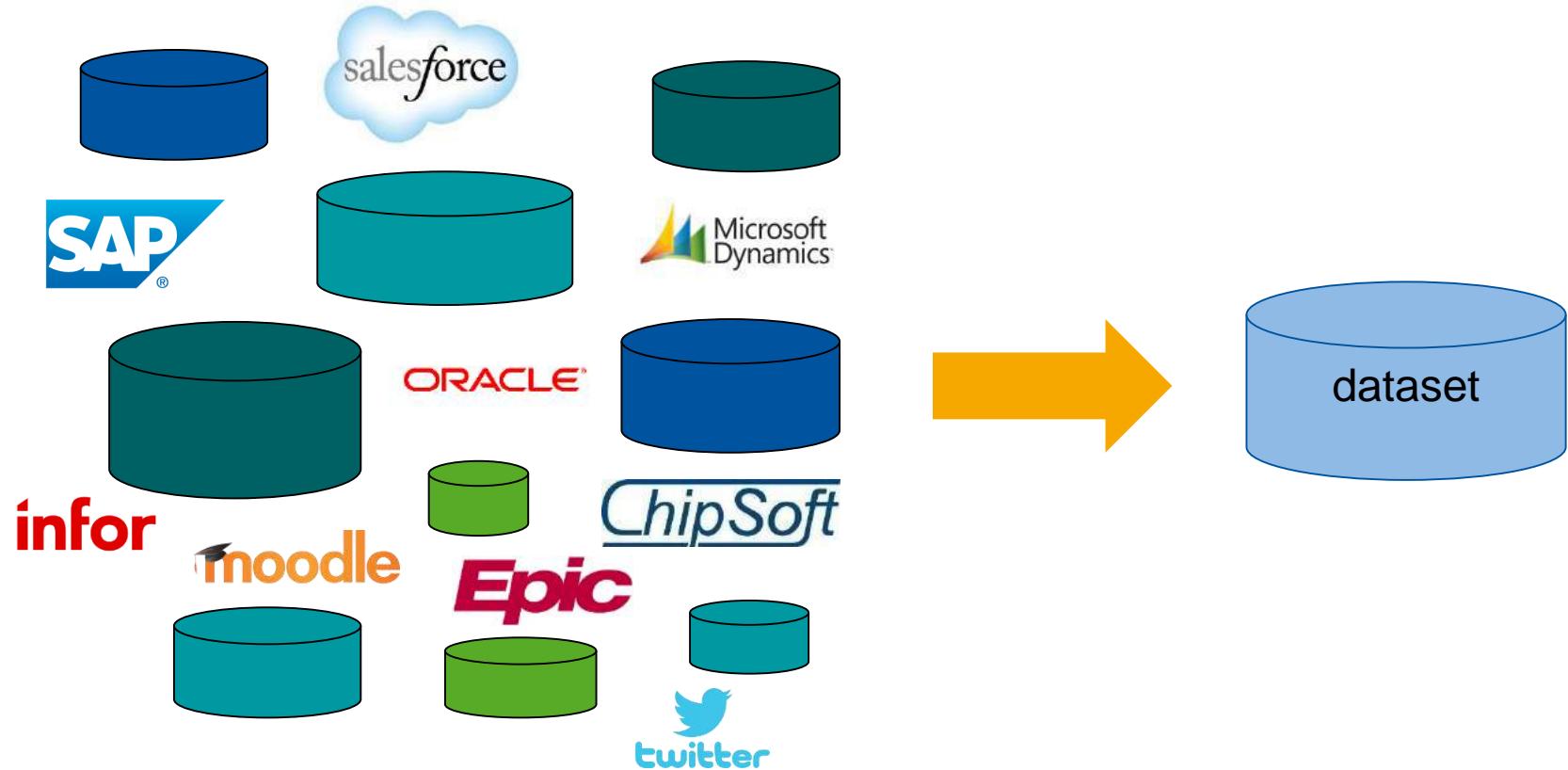
Labeled – designated target feature

The diagram illustrates a labeled dataset. A large callout bubble at the top right points to the last column, labeled "target feature (class label)". Another callout bubble points to the first four columns, labeled "descriptive features". The table itself has a vertical label "instances" on its left and a horizontal label "features" at the bottom.

instances	price	calories	vegetarian	spicy	bestseller
	12.99	800	Yes	No	Yes
	9.99	600	Yes	Yes	No
	14.99	1000	No	Yes	No
	11.99	700	No	No	Yes
	8.99	500	Yes	No	No

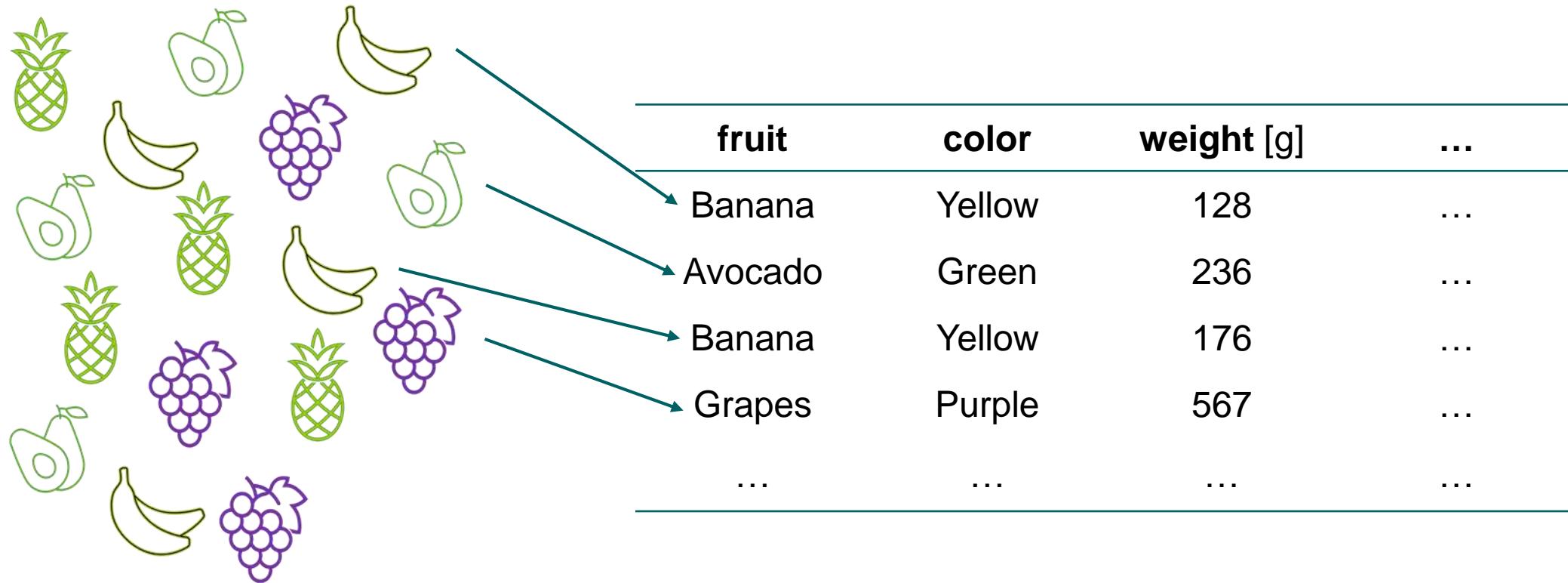
features

Extracting Data



80/20

Feature Extraction



Example – Instances and Features

- Rows – instances
- Columns – features

instances	features				
	price	calories	vegetarian	spicy	bestseller
	12.99	800	Yes	No	Yes
	9.99	600	Yes	Yes	No
	14.99	1000	No	Yes	No
	11.99	700	No	No	Yes
8.99	500	Yes	No	No	

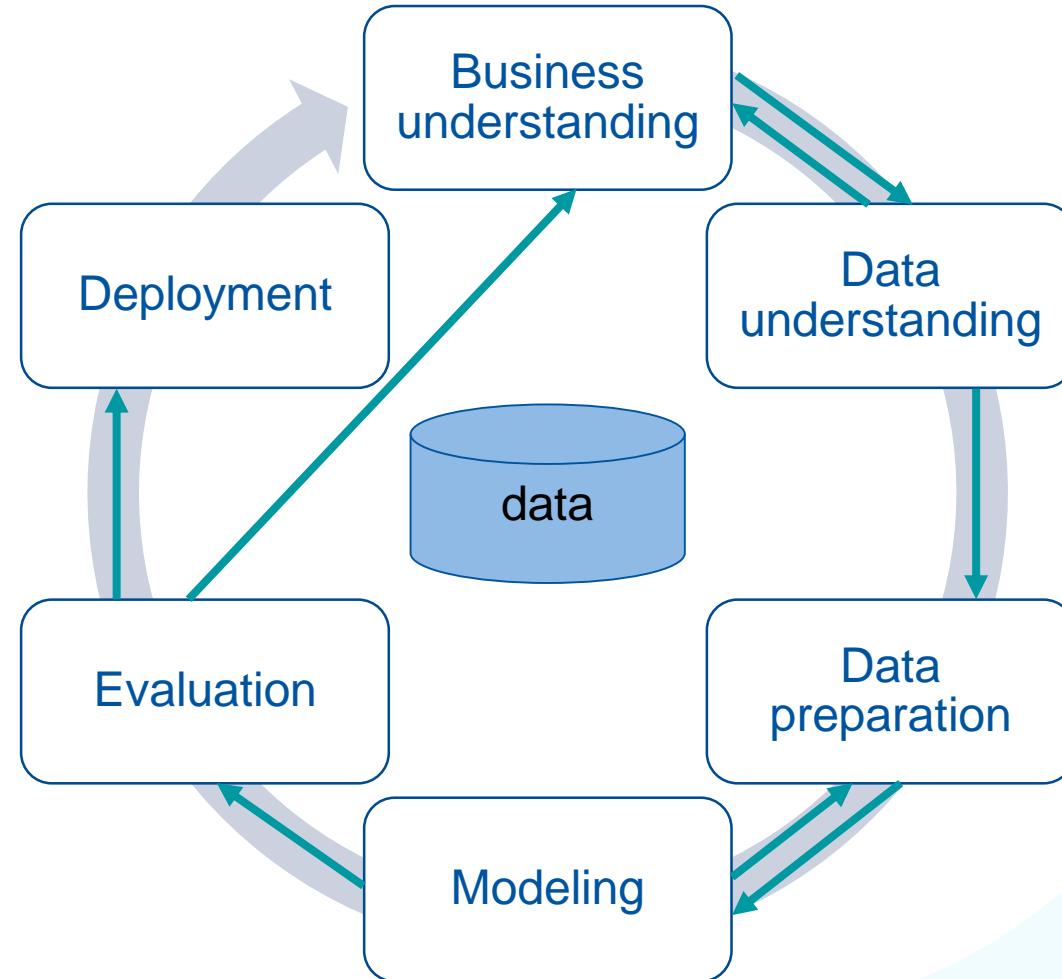
Data Science Is Complex and Requires a Structured Approach

“Data science is an interdisciplinary field aiming to turn data into real value. Data may be structured or unstructured, big or small, static or streaming. Value may be provided in the form of predictions, automated decisions, models learned from data, or any type of data visualization delivering insights. Data science includes data extraction, data preparation, data exploration, data transformation, storage and retrieval, computing infrastructures, various types of mining and learning, presentation of explanations and predictions, and the exploitation of results taking into account ethical, social, legal, and business aspects.”

Wil van der Aalst. Process Mining: Data Science in Action. Springer-Verlag, Berlin, 2016.

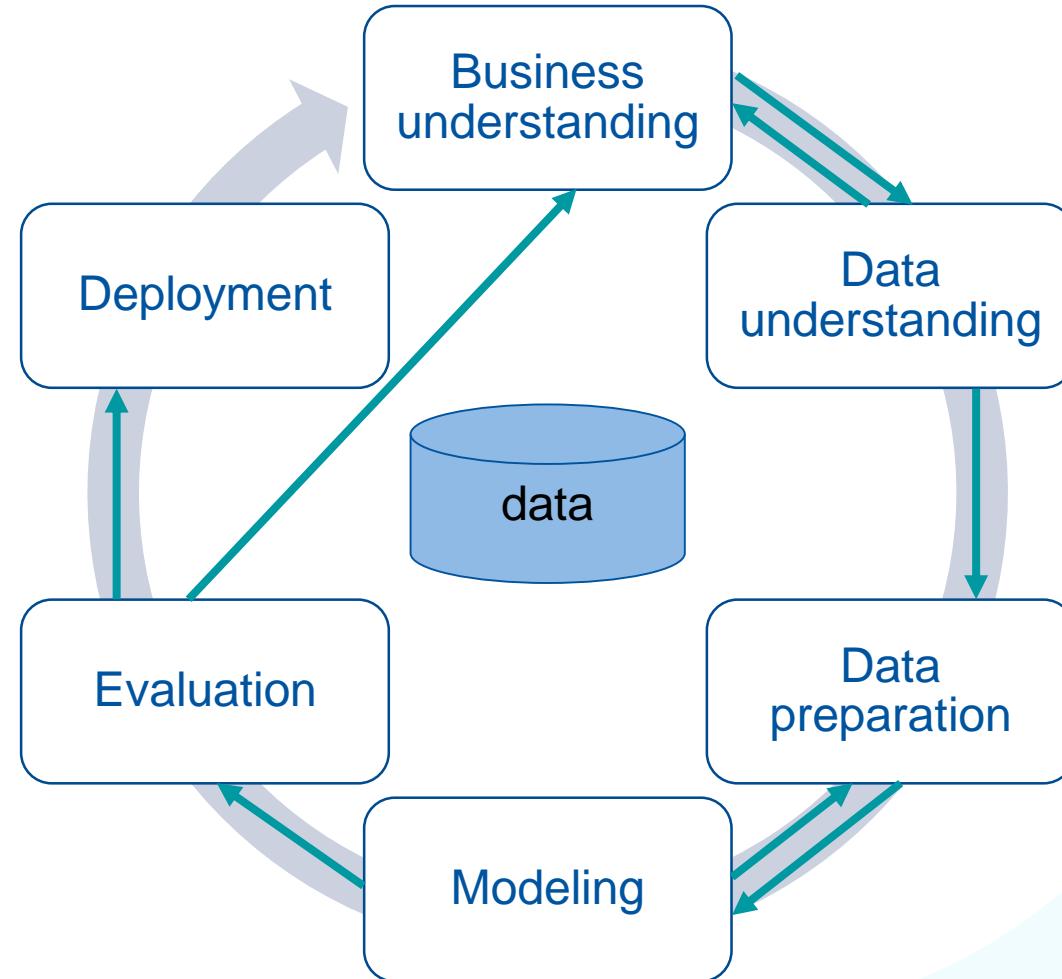
Cross-Industry Standard Process for Data Mining (CRISP-DM)

- Developed in the late 90s
- Its structure is quite obvious
- Details: Pete Chapman (1999)
‘The CRISP-DM User Guide’
- Any similar life-cycle models

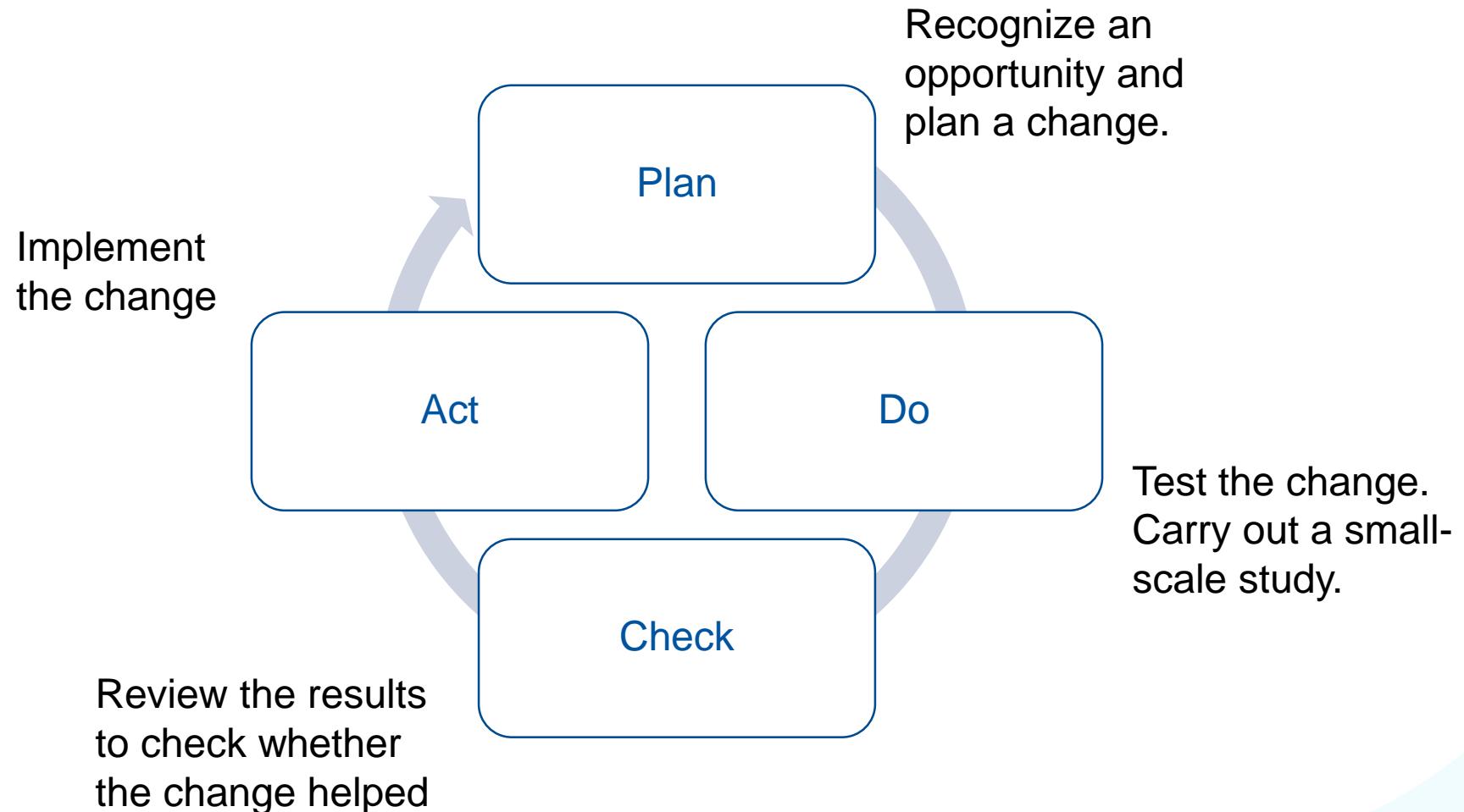


Cross-Industry Standard Process for Data Mining (CRISP-DM)

1. Business understanding – What does the organization need?
2. Data understanding – What data do we have?
3. Data preparation – How do we prepare the data for analysis?
4. Modeling – What modeling techniques should we apply?
5. Evaluation – Which model best meets the business objectives?
6. Deployment – How do stakeholders access and use the results?



Plan-Do-Check-Act (PDCA)



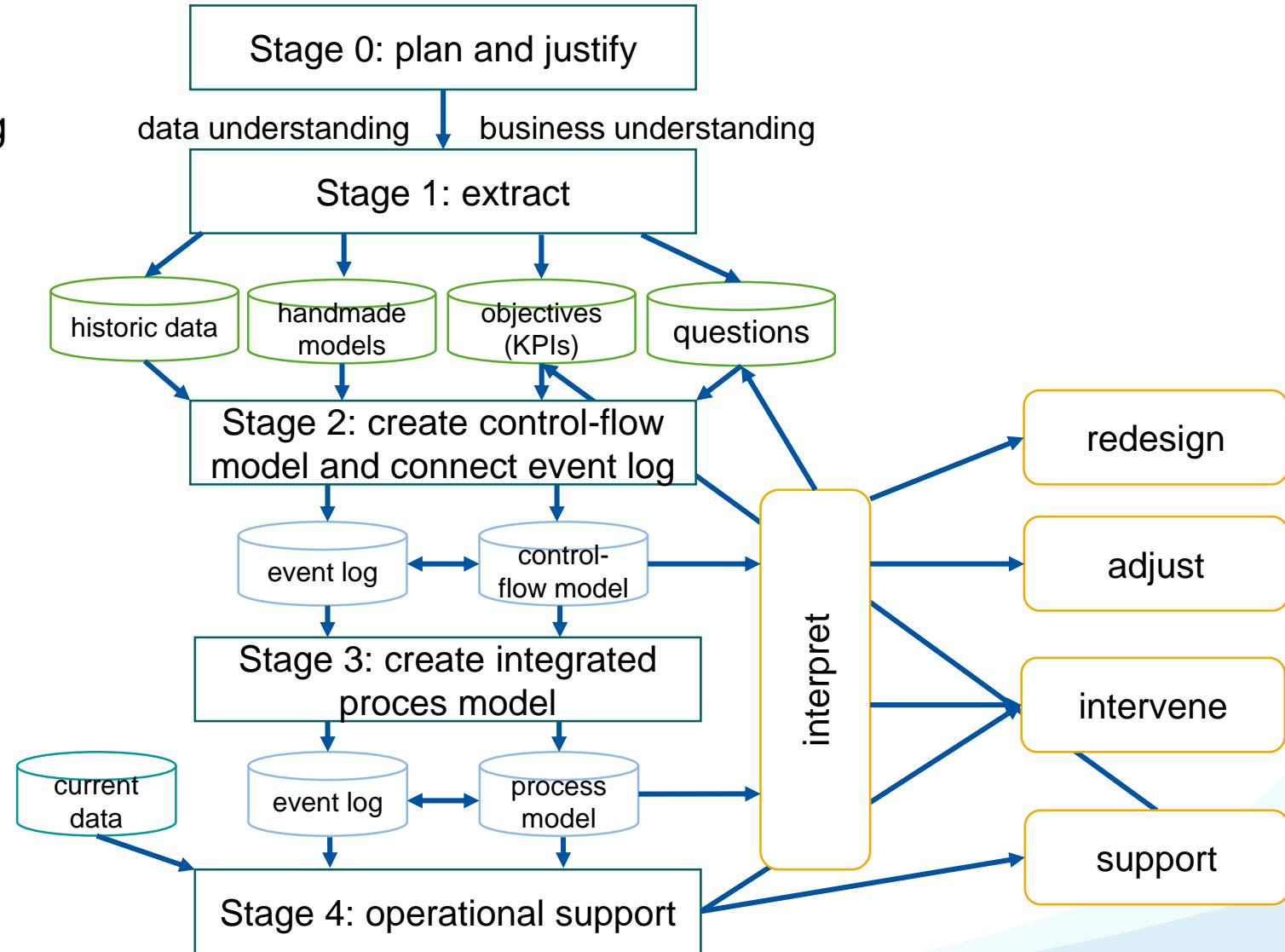
Define-Measure-Analyze-Improve-Control (DMAIC)

Define	Measure	Analyze	Improve	Control
<ul style="list-style-type: none">• Launch team• Establish charter• Plan project• Gather VOC/VOB• Plan for change	<ul style="list-style-type: none">• Document the process• Collect baseline data• Narrow project focus	<ul style="list-style-type: none">• Analyze data• Identify root causes• Identify and remove waste	<ul style="list-style-type: none">• Generate solutions• Evaluate solutions• Optimize solutions• Pilot• Plan and implement	<ul style="list-style-type: none">• Control the process• Validate project benefits

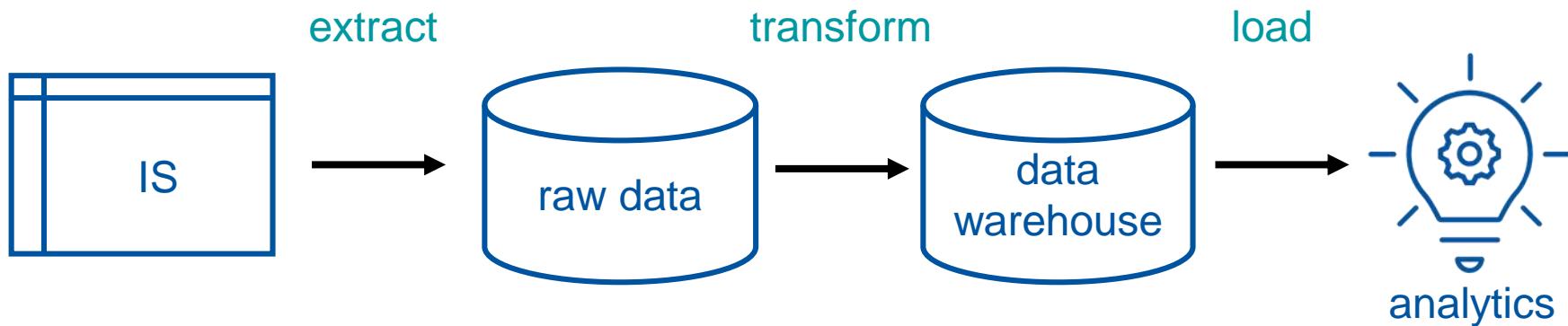
Often used as part of the Six Sigma methodology

L* Lifecycle Model

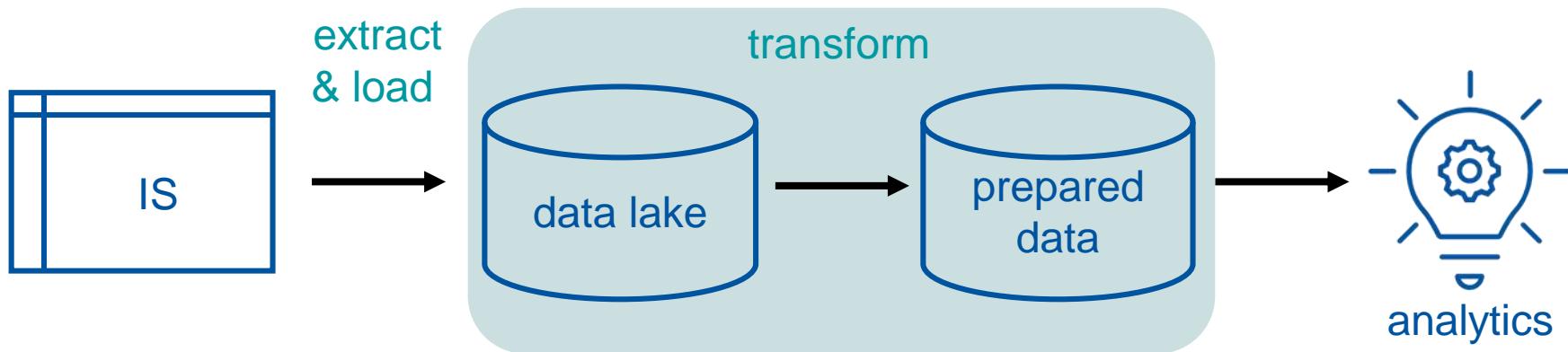
Specific for process mining



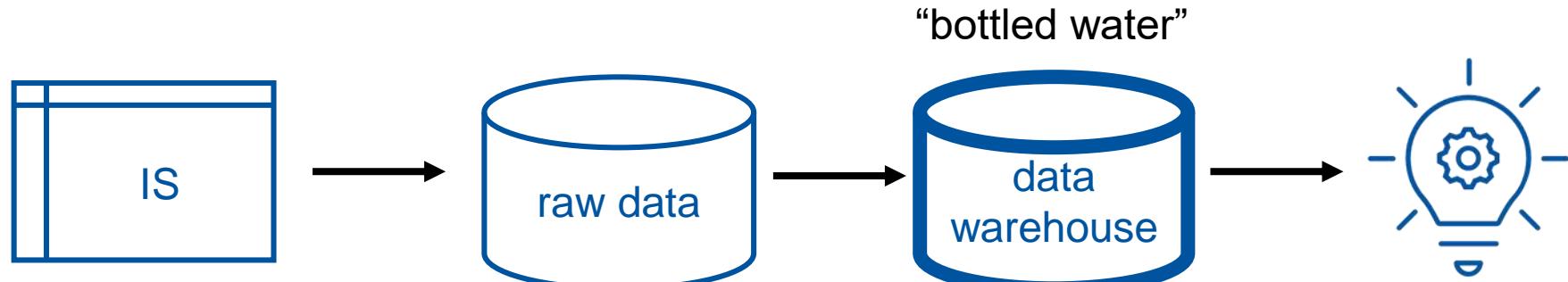
Extract-Transform-Load (ETL)



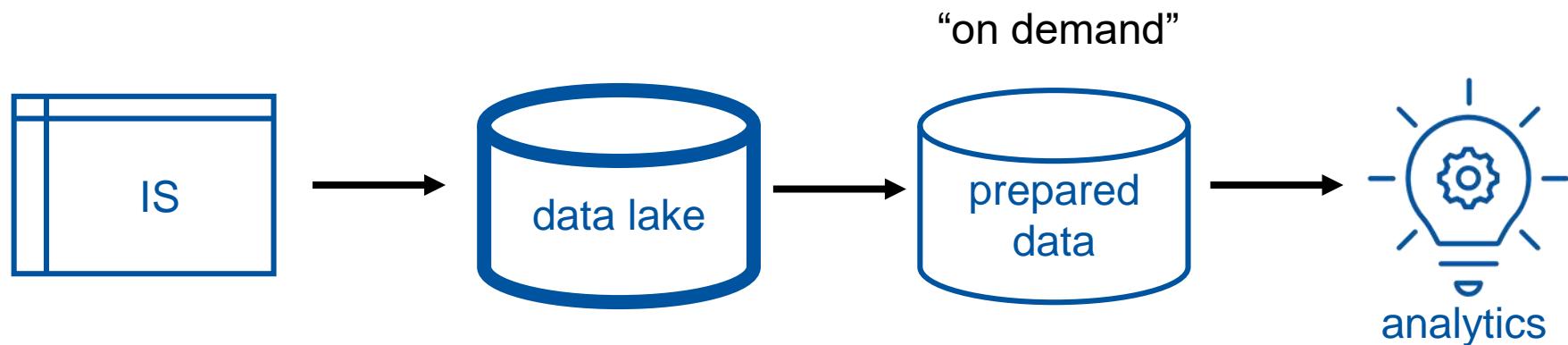
Extract-Load-Transform (ELT)



Differences



Extract-Transform-Load (ETL)



Extract-Load-Transform (ELT)

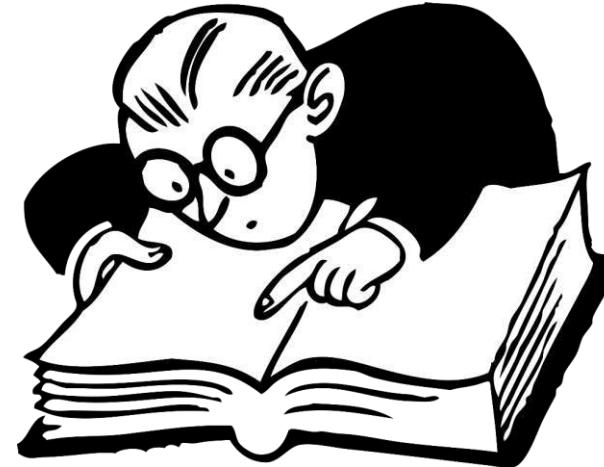
Organizational Issues

- Project or a continuous effort?
- Involve all stakeholders (users, customers, process owners, managers, board level, etc.)
- Positive Return-on-Investment (ROI) requires actionable insights
- Prepare for resistance (privacy concerns, data quality excuses, fear of transparency, etc.)
- Requires change management

Important, but ... our focus will be on data science techniques

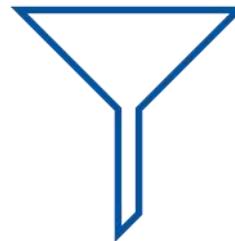
Finding Data

- There may be hundreds or thousands of tables
- There may exist many different entities that are less or not at all relevant



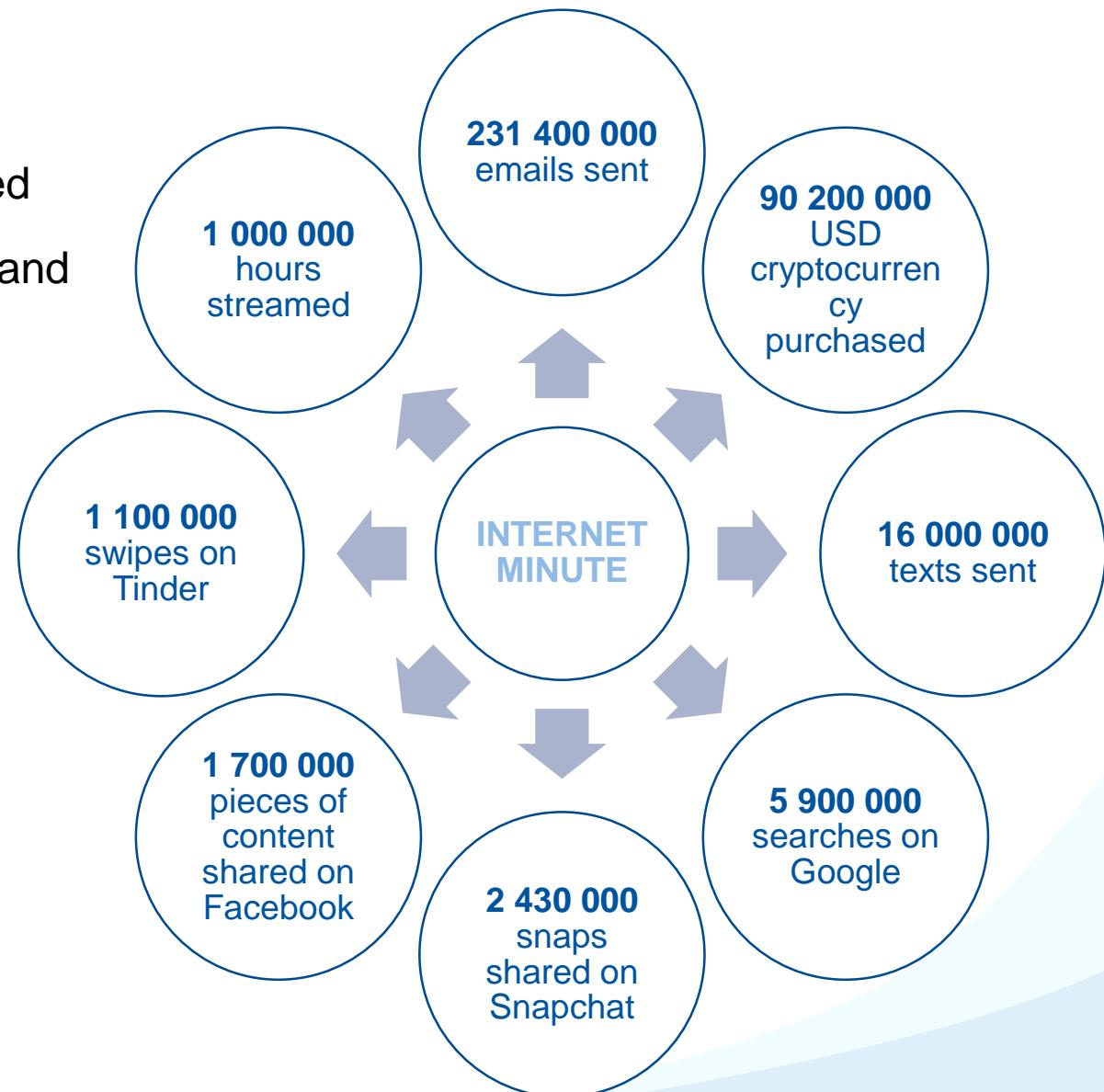
Preparing the Data

- Reorganizing data, filtering data, etc.
- Extracting relevant features
- **Normalization** (elimination of the effects of varying scales and units in different features, allowing for more accurate comparisons)
- **Sampling** (making data smaller or removing/changing a sample bias)



Big Data

- Lots of data (e.g. transactions) are recorded
- Need to have the ability to save, compare and analyze the collected data
- Requires distribution and concurrency



Streaming Data

- Data is generated continuously and processed in real-time
- Data is not stored in a database for later analysis
- Challenge: processing the data in real-time, need to handle the volume and velocity



Streaming Data

- Data is generated continuously and processed in real-time
- Data is not stored in a database for later analysis
- Challenge: processing the data in real-time, need to handle the **volume** and **velocity**



Source: De Agostini Editorial/Getty Images



Source: NatGeo

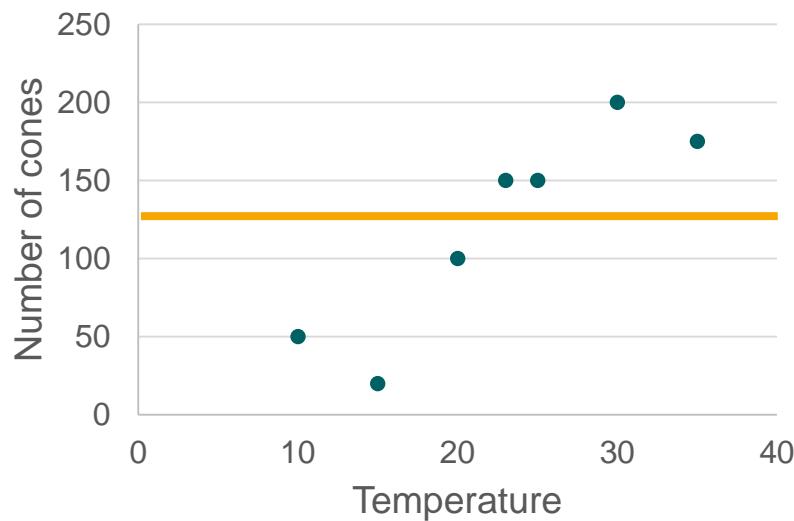
Data Quality

- Data may be:
 - Incomplete
 - Invalid
 - Inconsistent
 - Imprecise
 - Outdated
- Challenge: detecting and handling such issues

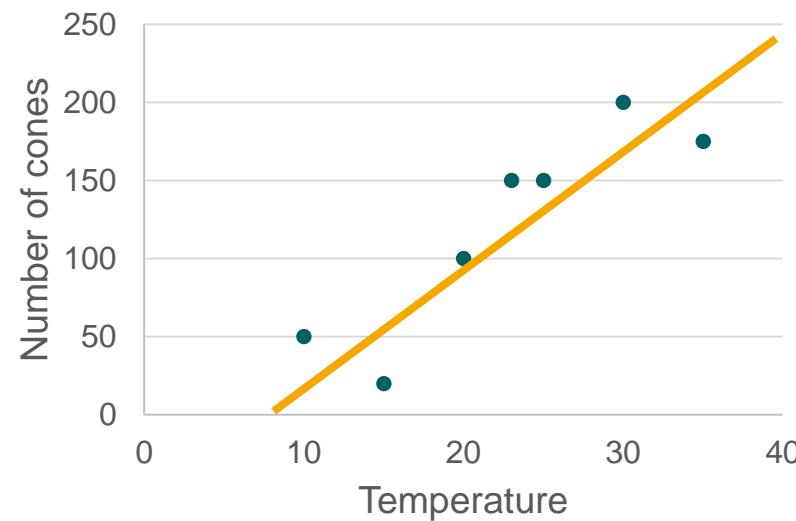


Overfitting and Underfitting

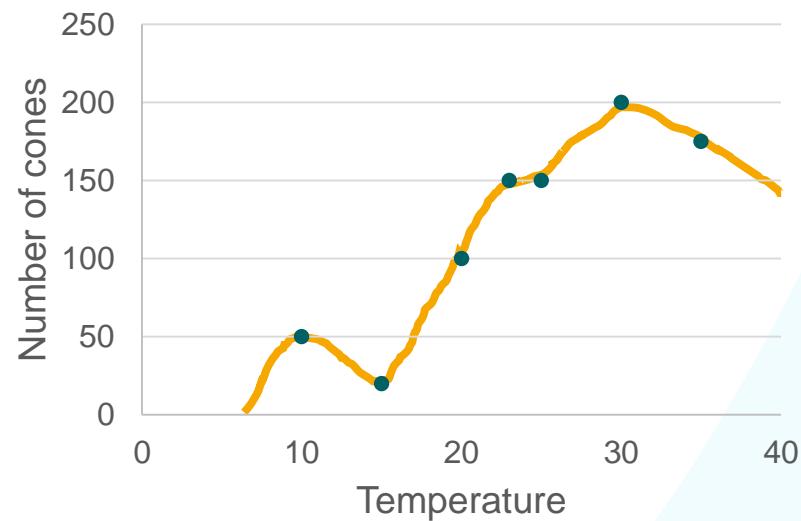
Underfitting



Optimal



Overfitting



Concept Drift

- Properties of the data change over time and thus the performance of a model decreases
- The data that the model is trained on no longer represents the real-world data
- Challenge: when to update the model with new data



Turning Insights into Action



- Predicting the inevitable does not help much
- What can be influenced?
- Is there still time?

Concerns – Responsible Data Science

- Responsible Data Science advocates the development of techniques, algorithms, tools, laws, ethical/social principles for ensuring **fairness**, **accuracy**, **confidentiality** and **transparency** covering the whole data science pipeline



Concerns – Responsible Data Science

- Responsible Data Science advocates the development of techniques, algorithms, tools, laws, and ethical/social principles for ensuring **fairness**, **accuracy**, **confidentiality** and **transparency** covering the whole data science pipeline
- **Fairness**: How to avoid unfair conclusions even if they are true?
- **Accuracy**: How to answer questions with a guaranteed level of accuracy?
- **Confidentiality**: How to answer questions without revealing secrets?
- **Transparency**: How to clarify answers such that they become indisputable?



III-posed Problems

- A problem is **well-posed** if
 - A solution **exists**
 - The solution is **unique**
- Problems in data science are often **ill-posed**:
 - **Many possible models** explaining observed phenomena
 - Data set is just a **sample** and does not represent the whole population
 - **Noise** in the data set
 - The result needs to **generalize** to have predictive or explanatory value



Data Types

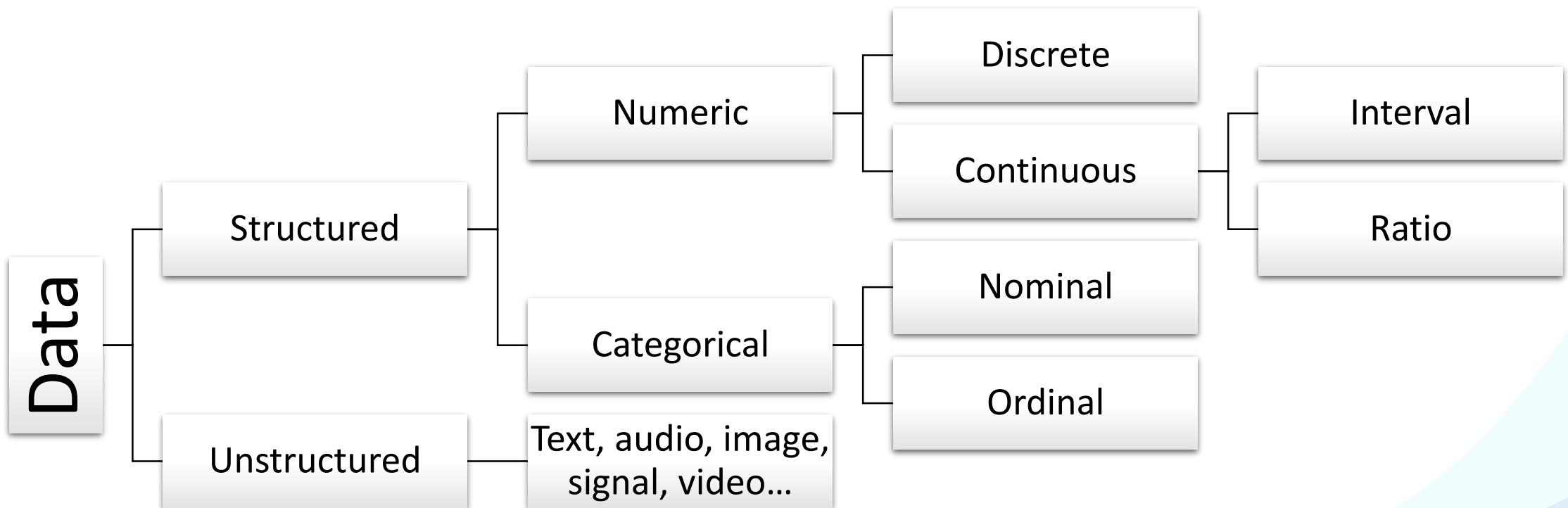
Tabular Data

Feature values can have various types - knowing these data types is essential for correct data analysis and data processing!

instances	features				bestseller
	price	calories	vegetarian	spicy	
	12.99	800	Yes	No	Yes
	9.99	600	Yes	Yes	No
	14.99	1000	No	Yes	No
	11.99	700	No	No	Yes
	8.99	500	Yes	No	No

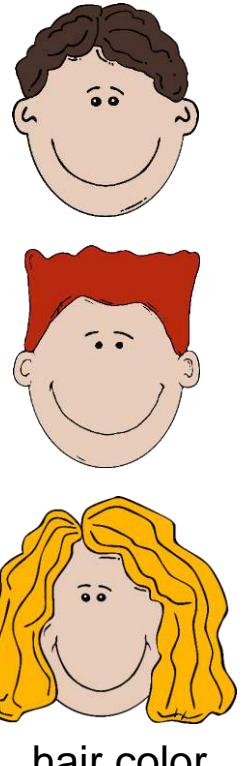
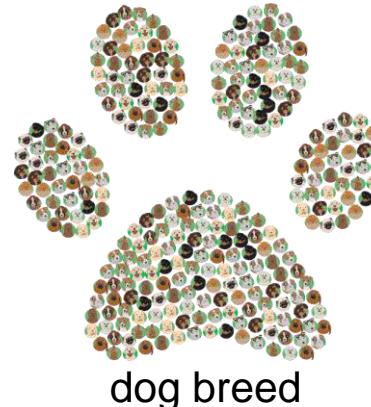
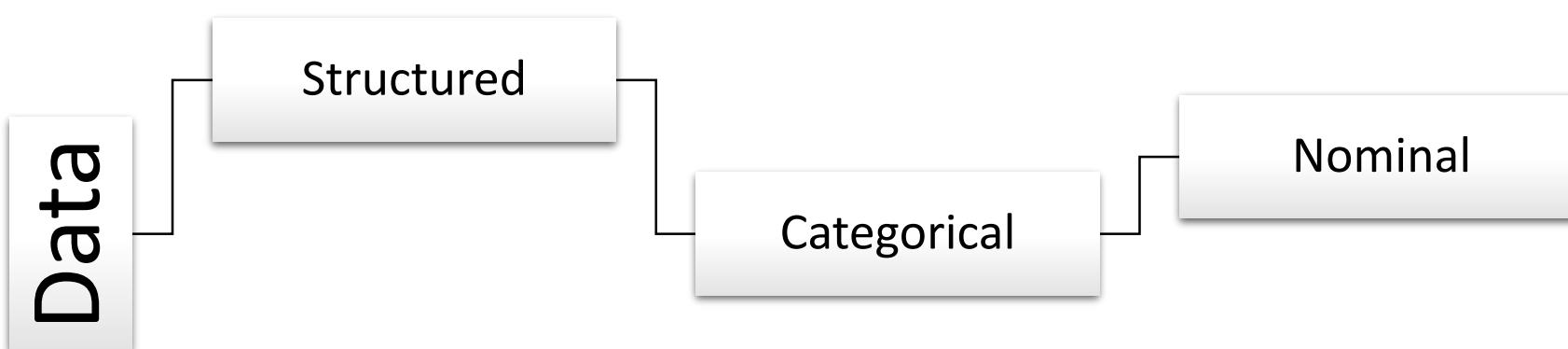
Data Types Overview

Feature values can have various types - knowing these **data types** is essential for correct data analysis and data processing!



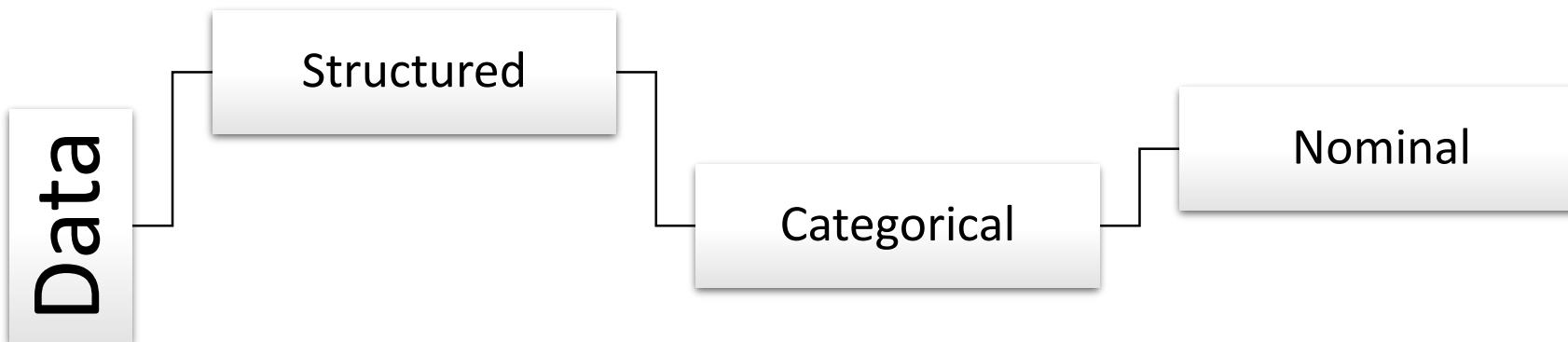
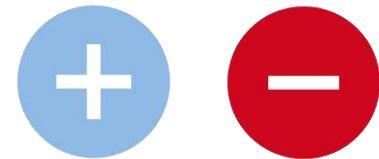
Data Types - Nominal

- Represents category, code or state
- Ordering of the values has no meaning
(e.g., blonde hair is not better than brown hair)



Data Types - Binary

- Special case of nominal: Binary
- Only two categories (often 0 and 1)
- Symmetric: both values are equal (subjectively or frequency based)
- Asymmetric: one value is normal/default, the other exception



Data Types - Ordinal

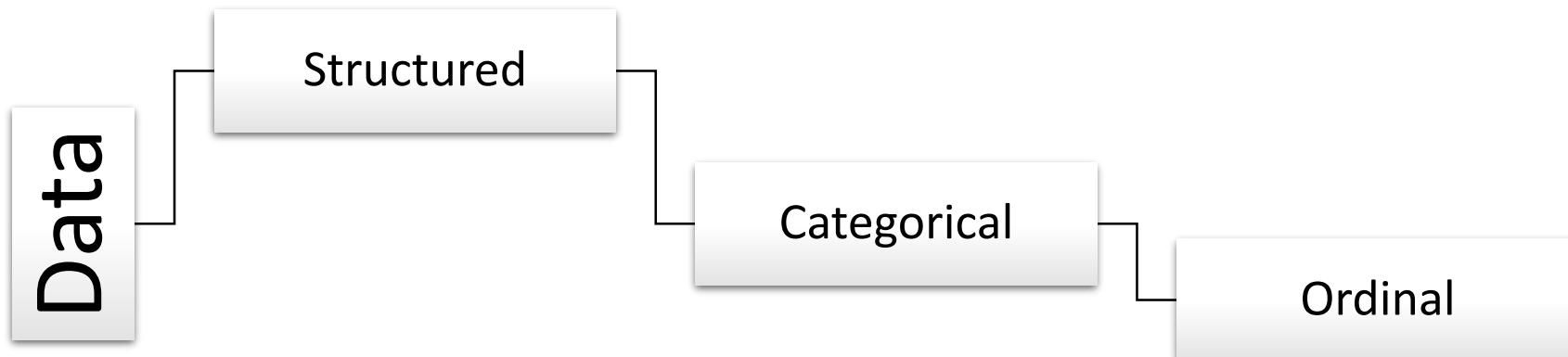


grades



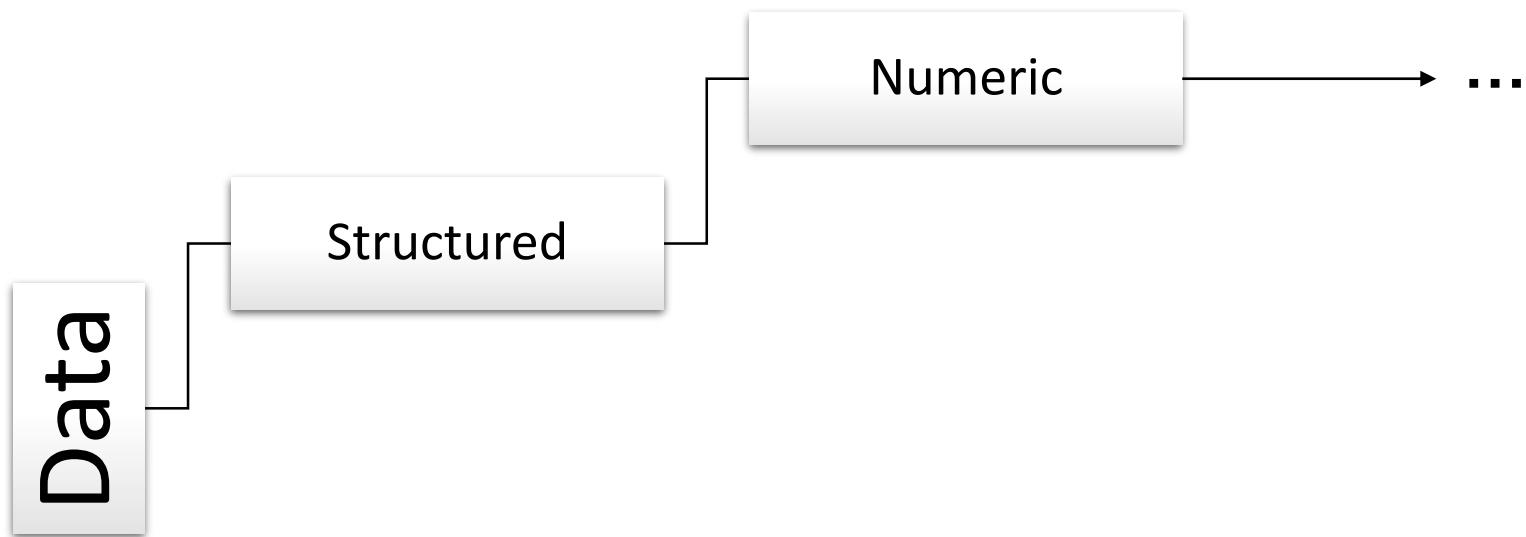
customer satisfaction

- Values have a meaningful order
 - high, medium, low
 - excellent, good, satisfactory, poor
 - lightning fast, quick, slow
 - strongly agree, agree, indifferent, disagree, strongly disagree
- The difference between successive values cannot be quantified



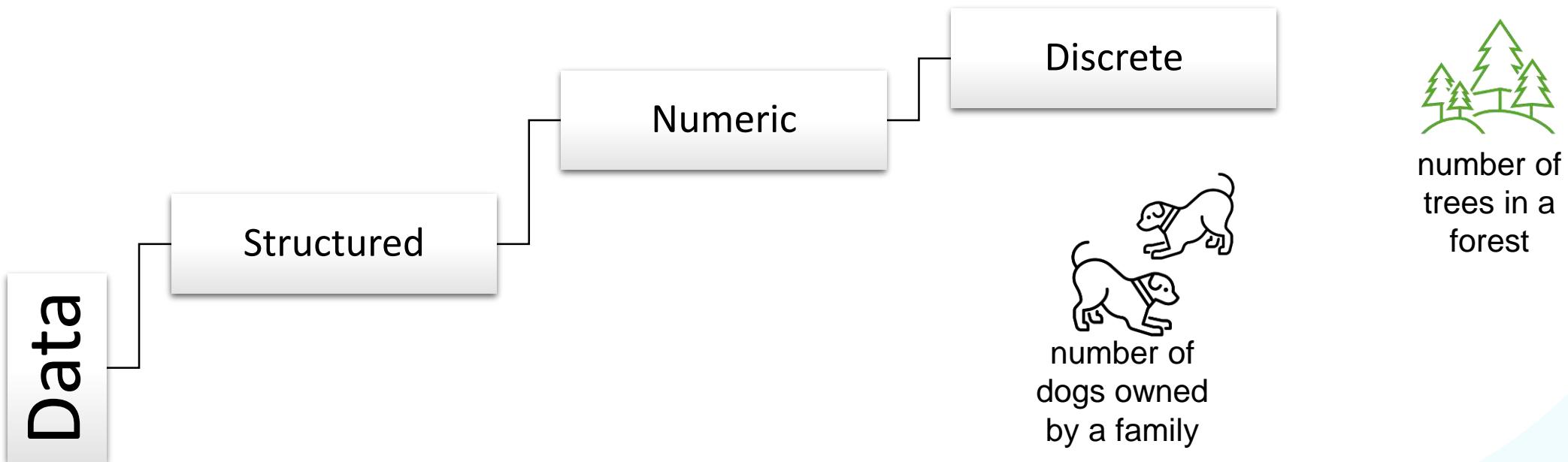
Data Types - Numeric

- Measurable quantities
- Differences can be quantified
- Mean, median, mode, variance, etc. can be computed



Data Types – Discrete

- Numeric
- Can be counted



Data Types - Interval

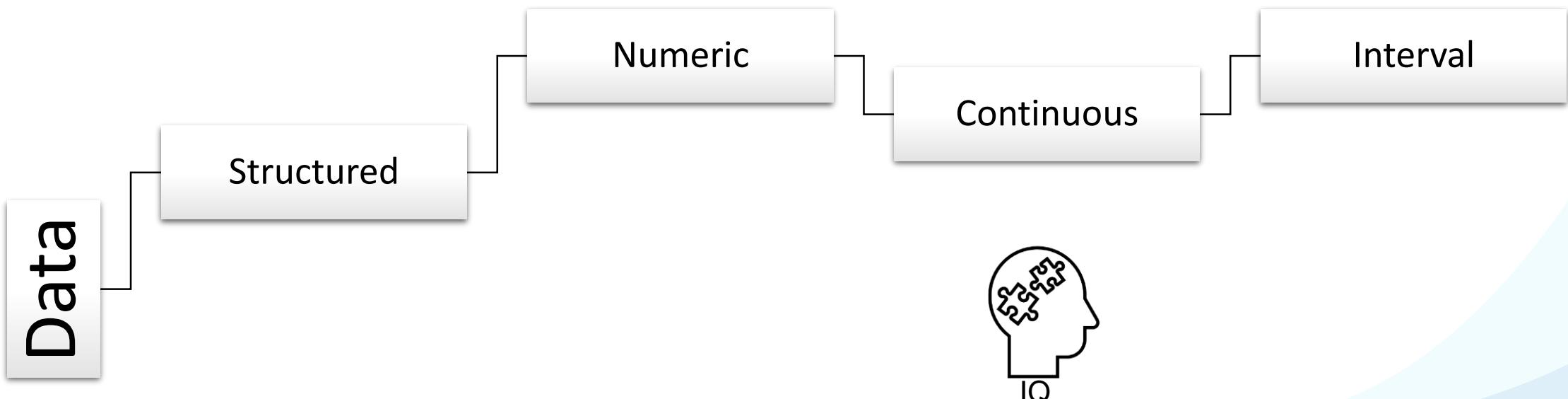
- Scale of equal-sized units with quantifiable difference between the units
- A zero may not exist, values may go negative



coordinates

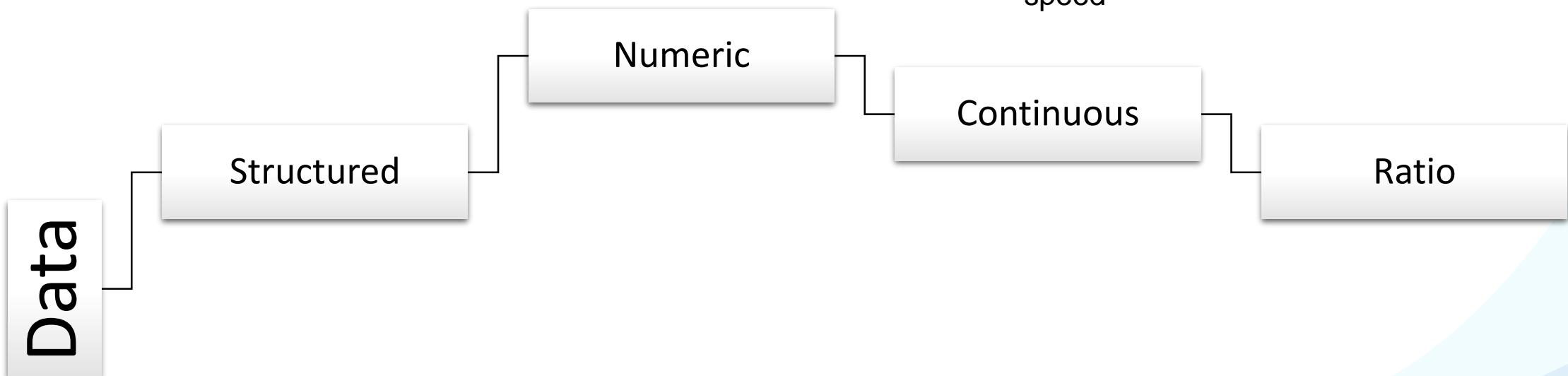


temperature
(Celsius, Fahrenheit)

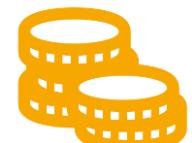


Data Types - Ratio

- Multiples/ratios can be identified (e.g., three times as heavy, four times as fast, etc.)
- The scale ends at zero (0 kg, 0 km/h, 0 Kelvin)



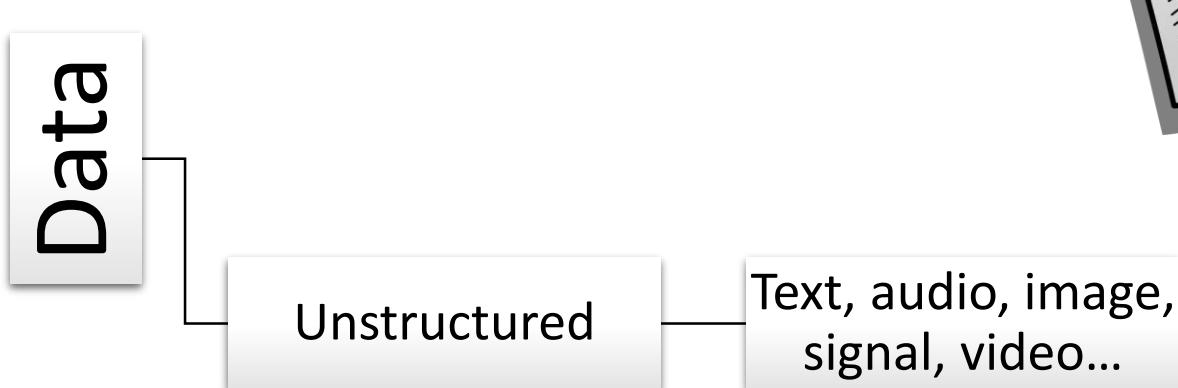
temperature
(Kelvin)



monetary
values

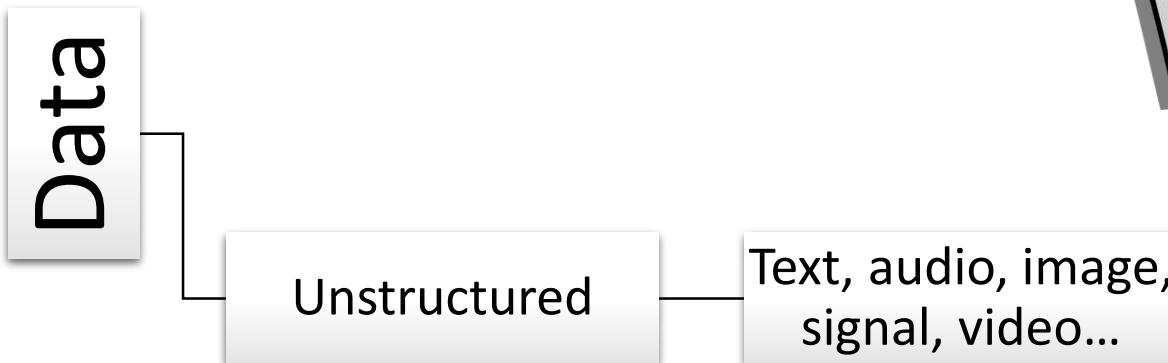
Data Types - Unstructured

- Text, audio, video, etc.
- Can be turned into structural data
- Examples: multiset of words or n-grams to describe a text, or pixel information for images



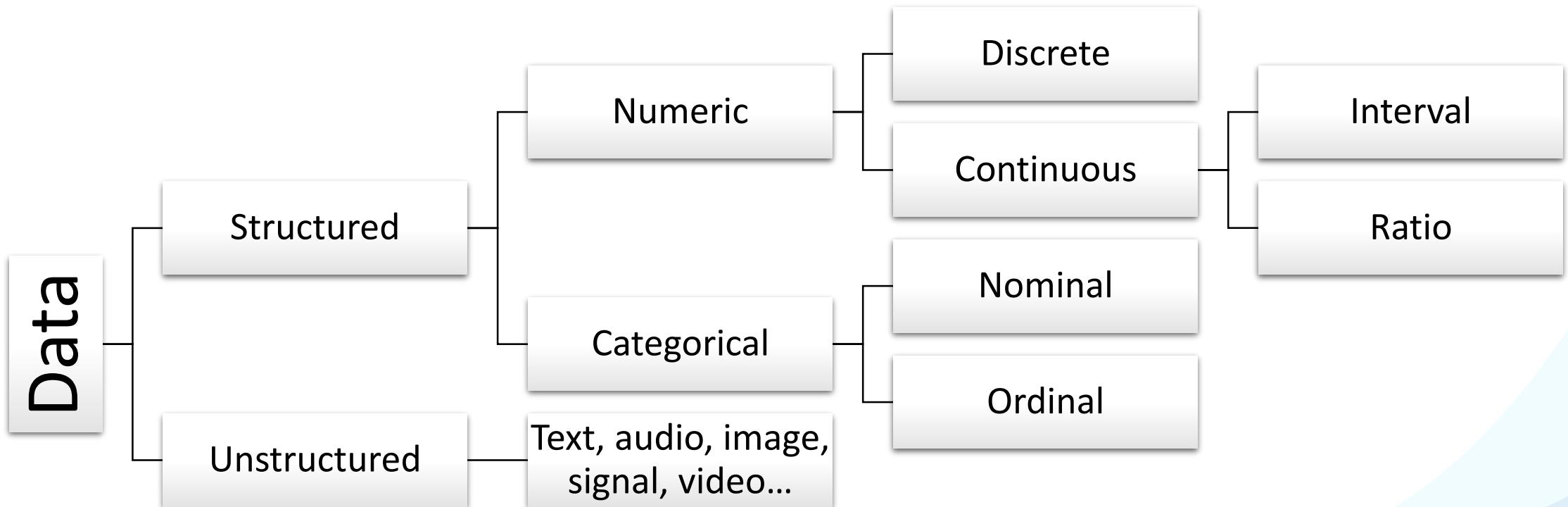
Data Types - Unstructured

- Extremely prevalent in Big Data
- Huge opportunity for novel transformation/extraction approaches
 - e.g. NLP
- Misnomer, as data may be structured, just not to an appreciable degree under the current viewpoint



Data Types Overview

- Data types are essential for correct data analysis and data processing!



Conclusion

- Data can be **unstructured** (e.g., text) but turned into e.g., vectors
- Most techniques are based on **tabular data** (especially the basic ones)
- The **data type** is vital for the correct data processing and analysis

price	calories	vegetarian	spicy	bestseller
12.99	800	Yes	No	Yes
9.99	600	Yes	Yes	No
14.99	1000	No	Yes	No
11.99	700	No	No	Yes
8.99	500	Yes	No	No

Descriptive Statistics Repetition

Individual Features - Continuous

x
1.5
2.7
3.1
4.2
5.5
6.9
7.6
8.1
9.3
10.0

Count = 10
Number of instances

Usually denoted
by N in this course

Individual Features - Continuous

x	Count = 10
1.5	Number of instances
2.7	Cardinality = 10
3.1	Number of unique values
4.2	
5.5	
6.9	
7.6	
8.1	
9.3	
10.0	

Individual Features - Continuous

x	Count = 10
1.5	Number of instances
2.7	Cardinality = 10
3.1	Number of unique values
4.2	Min = 1.5
5.5	Minimum value
6.9	
7.6	
8.1	
9.3	
10.0	

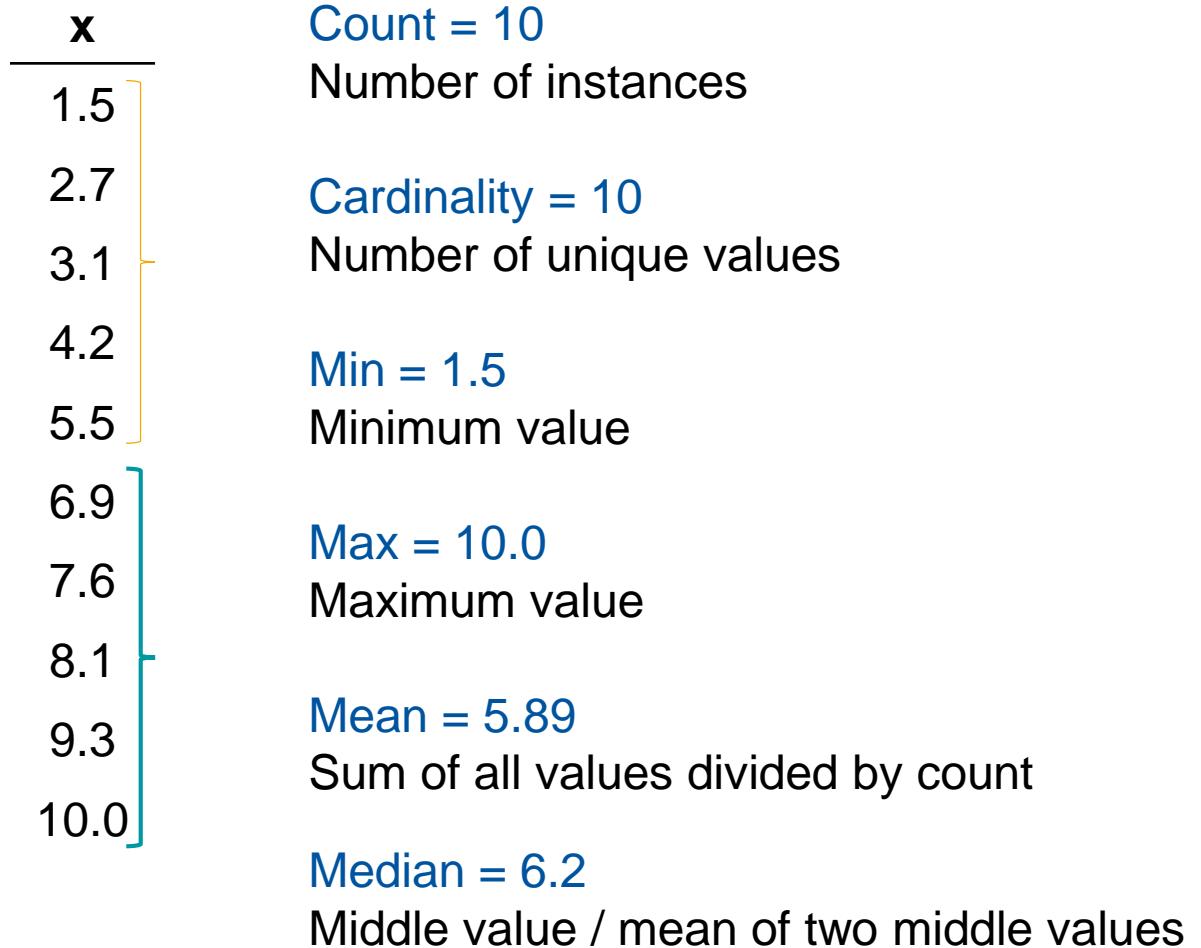
Individual Features - Continuous

x	Count = 10
1.5	Number of instances
2.7	Cardinality = 10
3.1	Number of unique values
4.2	Min = 1.5
5.5	Minimum value
6.9	Max = 10.0
7.6	Maximum value
8.1	
9.3	
10.0	

Individual Features - Continuous

x	Count = 10
1.5	Number of instances
2.7	Cardinality = 10
3.1	Number of unique values
4.2	Min = 1.5
5.5	Minimum value
6.9	Max = 10.0
7.6	Maximum value
8.1	$\bar{x} = \frac{\sum_{n=1}^N x_n}{N}$
9.3	Mean = 5.89
10.0	Sum of all values divided by count

Individual Features - Continuous



Individual Features - Continuous

x	Count = 10	
1.5	Number of instances	
2.7	Cardinality = 10	
3.1	Number of unique values	
4.2	Min = 1.5	
5.5	Minimum value	
6.9	Max = 10.0	
7.6	Maximum value	
8.1		
9.3	Mean = 5.89	
10.0	Sum of all values divided by count	
	Median = 6.2	
	Middle value / mean of two middle values	

$$var(x) = \frac{\sum_{n=1}^N (x_n - \bar{x})^2}{N-1}$$

Variance ≈ 8.621

Average squared distance of each value from the mean

Individual Features - Continuous

x	Count = 10 Number of instances	Variance ≈ 8.621 Average squared distance of each value from the mean
1.5	Cardinality = 10 Number of unique values	Standard deviation ≈ 2.936 How spread out the data is (the square root of the variance)
2.7	Min = 1.5 Minimum value	
3.1	Max = 10.0 Maximum value	
4.2	Mean = 5.89 Sum of all values divided by count	
5.5		
6.9		
7.6		
8.1		
9.3		
10.0	Median = 6.2 Middle value / mean of two middle values	$std(x) = \sqrt{var(x)}$

Individual Features - Continuous

x	Count = 10 Number of instances	Variance ≈ 8.621 Average squared distance of each value from the mean
10 th → 1.5	Cardinality = 10 Number of unique values	Standard deviation ≈ 2.936 How spread out the data is (the square root of the variance)
25 th → 2.7	Min = 1.5 Minimum value	p^{th} percentile Value at or below (or strictly below) which p percent of the instances are located
3.1	Max = 10.0 Maximum value	10 th percentile = 1.5 First quartile (Q_1)
4.2	Mean = 5.89 Sum of all values divided by count	25 th percentile = 3.1 Second quartile (Q_2)
50 th → 5.5	Median = 6.2 Middle value / mean of two middle values	50 th percentile = 5.5 75 th percentile = 8.1 100 th percentile = 10 Third quartile (Q_3)
6.9		
7.6		
8.1		
9.3		
100 th → 10.0		

Individual Features - Categorical

x	Count = 10
	Number of instances
A	
B	
A	
C	
B	
B	
C	
A	
C	
B	

Individual Features - Categorical

x	Count = 10
A	Number of instances
B	Cardinality = 3
A	Number of unique values
C	
B	
B	
C	
A	
C	
B	

Individual Features - Categorical

$\frac{x}{A}$	Count = 10 Number of instances
B	Cardinality = 3 Number of unique values
C	Mode = B Value that appears most frequently
B	
C	
A	
C	
B	

Multiple Features - Covariance

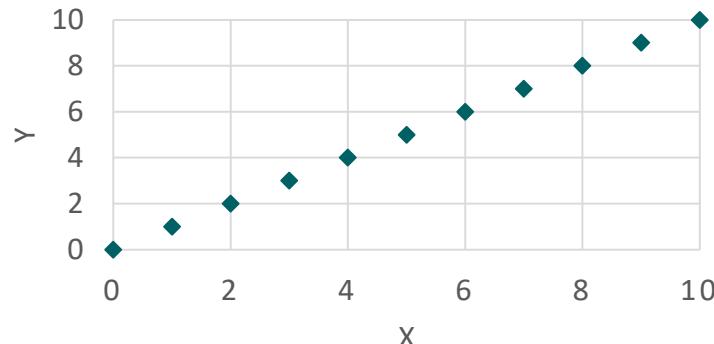
x	y
1.5	4.2
2.7	4.9
3.1	7.1
4.2	9.8
5.5	12.3
6.9	14.7
7.6	16.5
8.1	18.2
9.3	20.9
10.0	22.6

$$Cov(x, y) = \frac{1}{N-1} \sum_{n=1}^N ((x_n - \bar{x}) \cdot (y_n - \bar{y}))$$

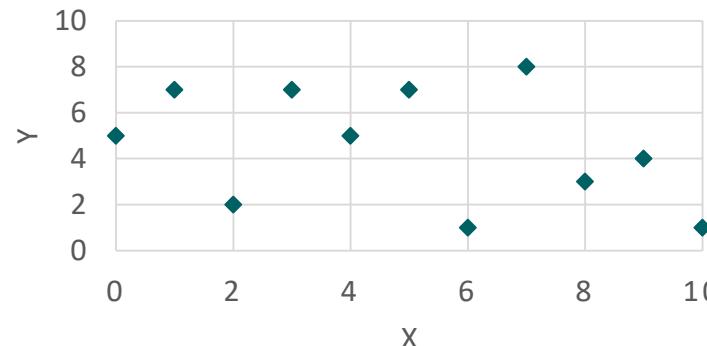
$$\text{Cov}(x,y) \approx 19.134$$

+ & + \Rightarrow +
+ & - \Rightarrow -
- & + \Rightarrow -
- & - \Rightarrow +

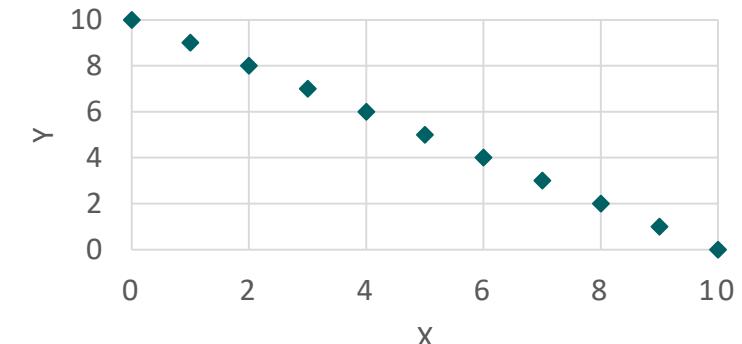
Multiple Features – Correlation



maximal positive correlation



no correlation



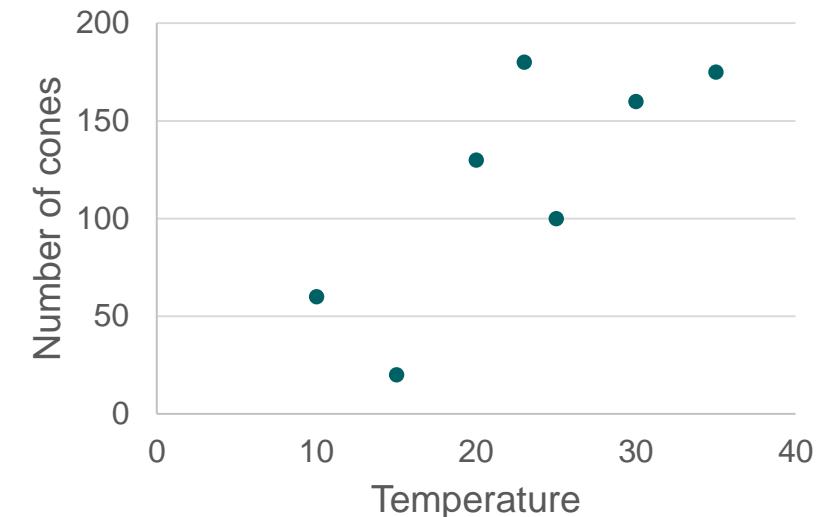
maximal negative correlation

$$\text{Corr}(x, y) = \frac{\text{Cov}(x, y)}{\sqrt{\text{Var}(x)} \cdot \sqrt{\text{Var}(y)}}$$

Between -1 and 1
 > 0: positive correlation
 < 0: negative correlation
 ≈ 0: independent

Multiple Features – Correlation (Example)

Temperature (°C)	Number of cones
10	60
15	10
20	185
23	150
25	150
30	200
35	175



$$\text{Corr}(x, y) = \frac{\text{Cov}(x,y)}{\sqrt{\text{Var}(x)} \cdot \sqrt{\text{Var}(y)}} = \frac{419.88}{8.54 \cdot 63.69} = 0.77$$

Temperature

Number of cones

Strong positive correlation

Multiple Features – Correlation Matrix

Features a, b, \dots, z

$$\begin{matrix} & \textcolor{blue}{a} & \textcolor{blue}{b} & & \textcolor{blue}{z} \\ \textcolor{blue}{a} & \left[\begin{matrix} \text{Corr}(a, a) & \text{Corr}(a, b) & \dots & \text{Corr}(a, z) \\ \text{Corr}(b, a) & \text{Corr}(b, b) & \dots & \text{Corr}(b, z) \\ \dots & \dots & \dots & \dots \\ \text{Corr}(z, a) & \text{Corr}(z, b) & \dots & \text{Corr}(z, z) \end{matrix} \right] \end{matrix}$$

Multiple Features – Correlation Matrix

Features a, b, \dots, z

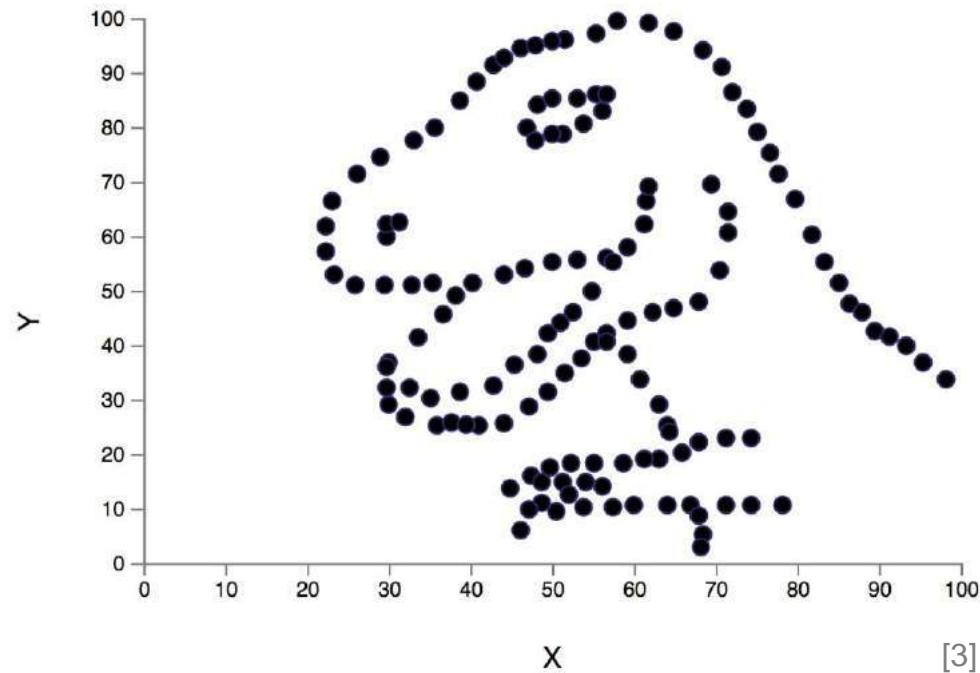
	a	b	\dots	z
a	1.0	0.90	\dots	0.35
b	0.90	1.0	\dots	0.30
\dots	\dots	\dots	\dots	\dots
z	0.35	0.30	\dots	1.0

What can we say about this distribution?

x	y
55.3846	97.1795
51.5385	96.0256
46.1538	94.4872
42.8205	91.4103
40.7692	88.3333
38.7179	84.8718
35.641	79.8718
33.0769	77.5641
28.9744	74.4872
26.1538	71.4103
...	...

$Count = 142$
 $Mean(x) = 54.2633$
 $Std(x) = 16.7651$
 $Mean(y) = 47.8323$
 $Std(y) = 26.9354$
 $Corr(x, y) = -0.0645$

Datasaurus



$Count = 142$

$Mean(x) = 54.2633$

$Std(x) = 16.7651$

$Mean(y) = 47.8323$

$Std(y) = 26.9354$

$Corr(x, y) = -0.0645$

Anscombe's Quartet

Dataset 1		Dataset 2		Dataset 3		Dataset 4	
x	y	x	y	x	y	x	y
10.0	8.04	10.0	9.14	10.0	7.46	8.0	6.58
8.0	6.95	8.0	8.14	8.0	6.77	8.0	5.76
13.0	7.58	13.0	8.74	13.0	12.74	8.0	7.71
9.0	8.81	9.0	8.77	9.0	7.11	8.0	8.84
11.0	8.33	11.0	9.26	11.0	7.81	8.0	8.47
14.0	9.96	14.0	8.10	14.0	8.84	8.0	7.04
6.0	7.24	6.0	6.13	6.0	6.08	8.0	5.25
4.0	4.26	4.0	3.10	4.0	5.39	19.0	12.50
12.0	10.84	12.0	9.13	12.0	8.15	8.0	5.56
7.0	4.82	7.0	7.26	7.0	6.42	8.0	7.91
5.0	5.68	5.0	4.74	5.0	5.73	8.0	6.89

[4]

$$\text{Mean}(x) = 9$$

$$\text{Var}(x) = 11$$

$$\text{Mean}(y) = 7.5$$

$$\text{Var}(y) = 4.125$$

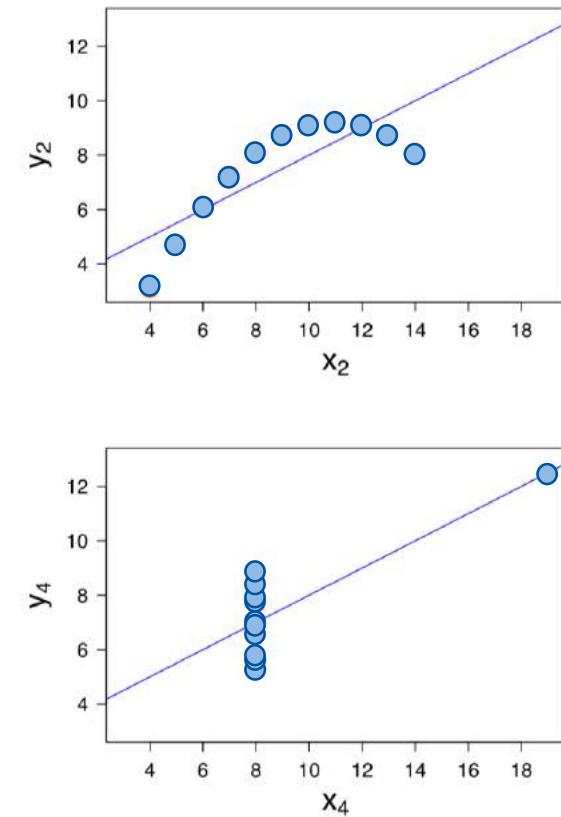
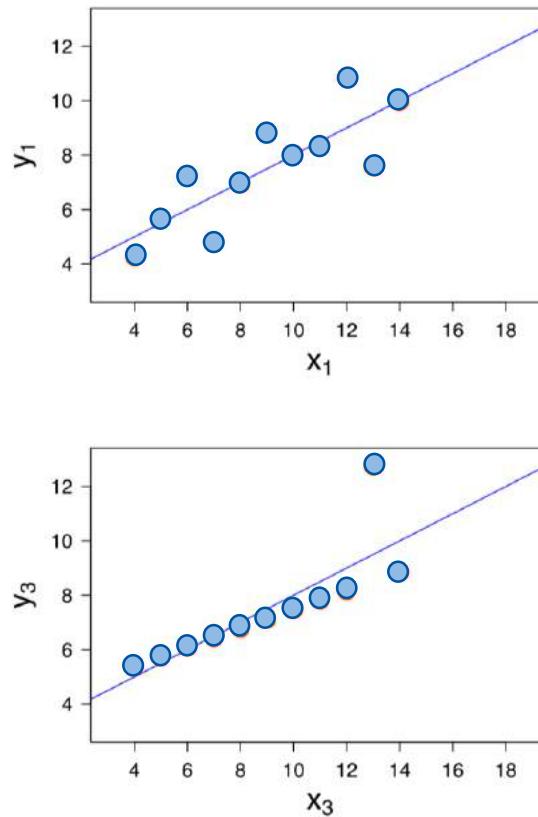
$$\text{Corr}(x, y) = 0.816$$

$$\text{Linear regression line: } y = \frac{1}{2}x + 3$$

Anscombe's Quartet

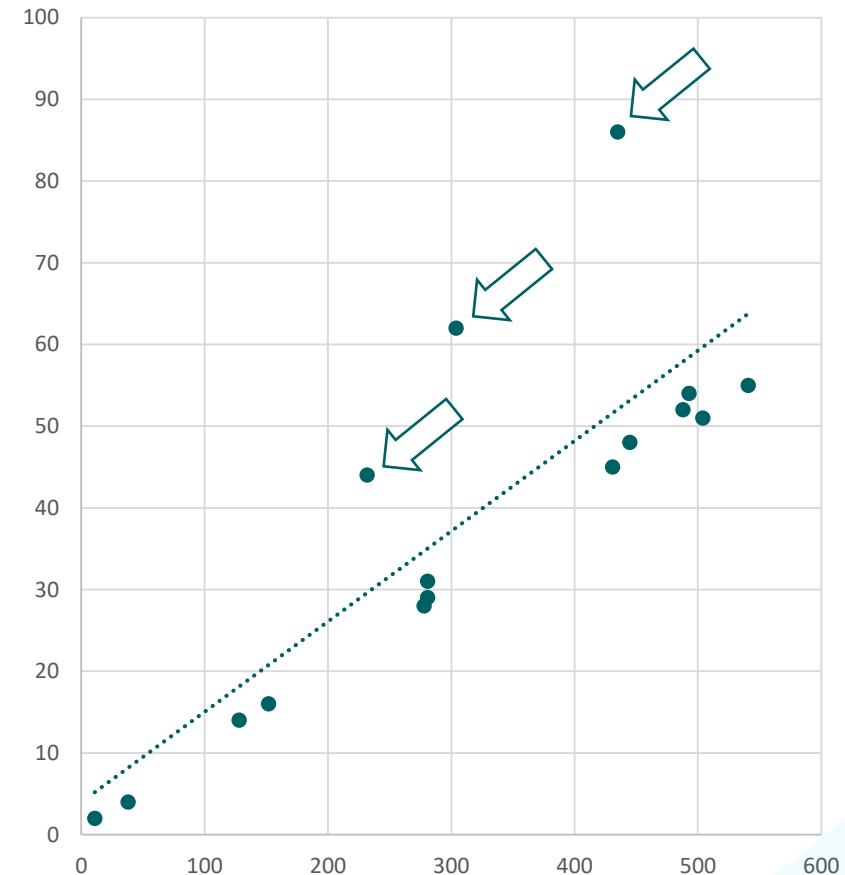
Dataset 1		Dataset 2		Dataset 3		Dataset 4	
x	y	x	y	x	y	x	y
10.0	8.04	10.0	9.14	10.0	7.46	8.0	6.58
8.0	6.95	8.0	8.14	8.0	6.77	8.0	5.76
13.0	7.58	13.0	8.74	13.0	12.74	8.0	7.71
9.0	8.81	9.0	8.77	9.0	7.11	8.0	8.84
11.0	8.33	11.0	9.26	11.0	7.81	8.0	8.47
14.0	9.96	14.0	8.10	14.0	8.84	8.0	7.04
6.0	7.24	6.0	6.13	6.0	6.08	8.0	5.25
4.0	4.26	4.0	3.10	4.0	5.39	19.0	12.50
12.0	10.84	12.0	9.13	12.0	8.15	8.0	5.56
7.0	4.82	7.0	7.26	7.0	6.42	8.0	7.91
5.0	5.68	5.0	4.74	5.0	5.73	8.0	6.89

[4]



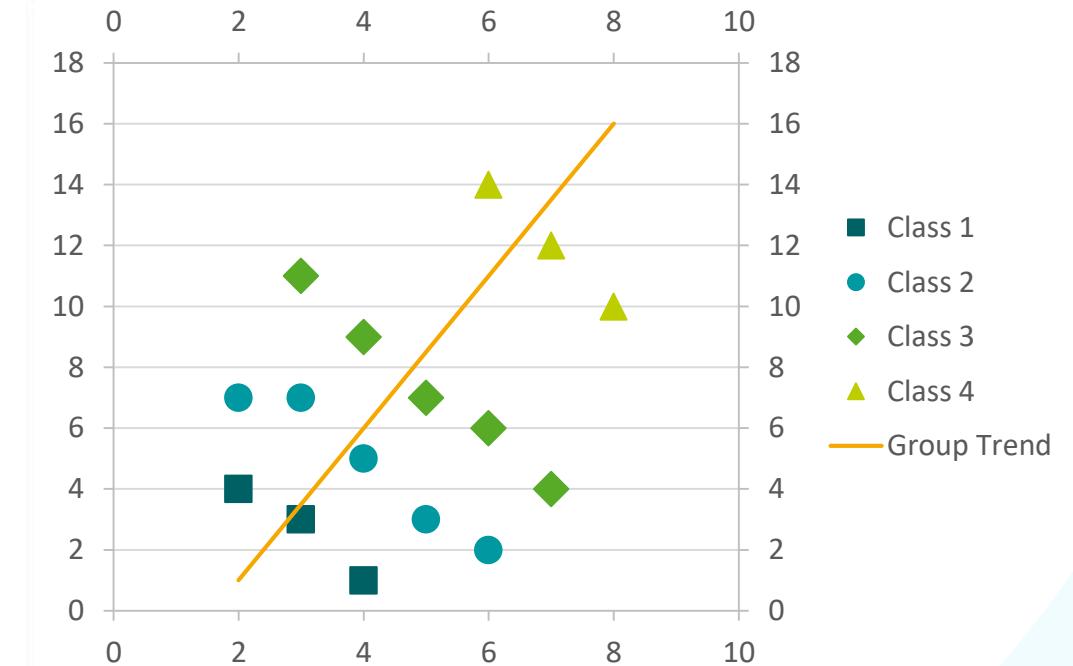
Outliers

- Outlier: an observation that lies an abnormal distance away from other values
- Can have a significant impact on measures such as mean, variance or standard deviation
- It is important to identify and deal with outliers before performing any analysis
→ visualize and explore our data first!



Simpson's Paradox

A trend appears in several different groups of data but **disappears or reverses** when these groups are combined.



Simpson's Paradox

	All		Men		Women	
	Applicants	Admitted	Applicants	Admitted	Applicants	Admitted
Total	12,763	41%	8,442	44%	4,321	35%

Aggregated

Department	All		Men		Women	
	Applicants	Admitted	Applicants	Admitted	Applicants	Admitted
A	933	64%	825	62%	108	82%
B	585	63%	560	63%	25	68%
C	918	35%	325	37%	593	34%
D	792	34%	417	33%	375	35%
E	584	25%	191	28%	393	24%
F	714	6%	373	6%	341	7%
Total	4526	39%	2691	45%	1835	30%

Legend:

greater percentage of successful applicants than the other gender

greater number of applicants than the other gender

bold - the two 'most applied for' departments for each gender

By department (six largest)

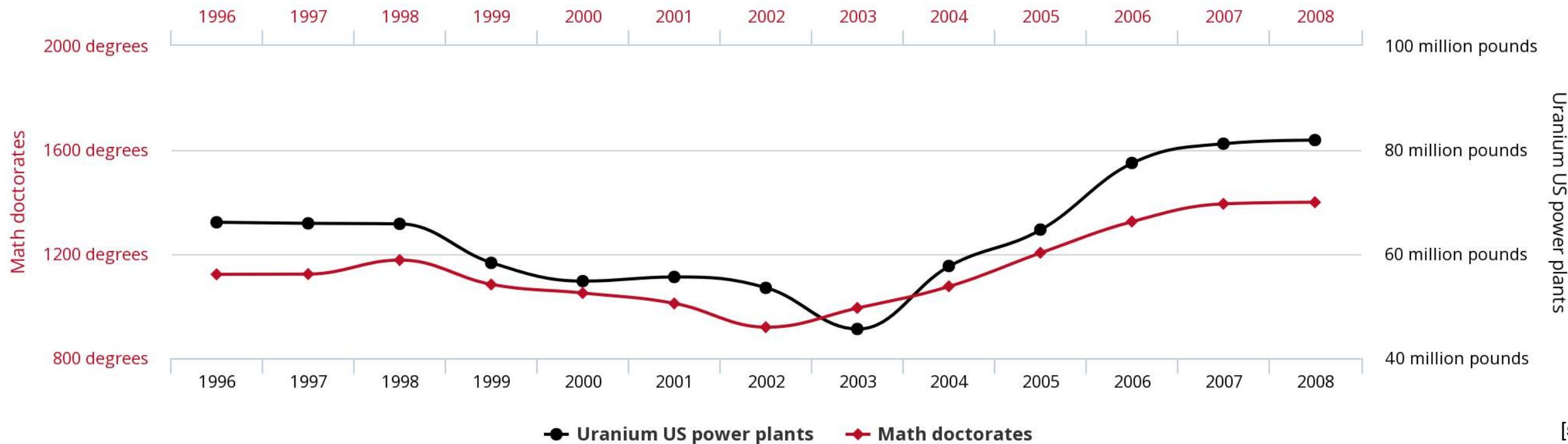
UC Berkeley admission data, 1973

Spurious Correlations

Math doctorates awarded

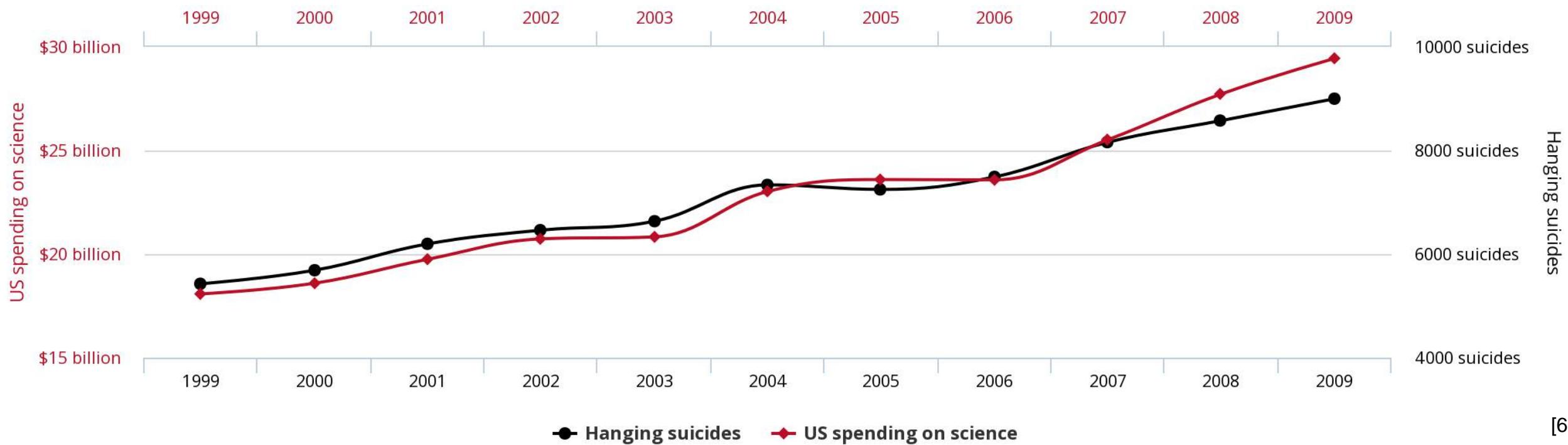
correlates with

Uranium stored at US nuclear power plants



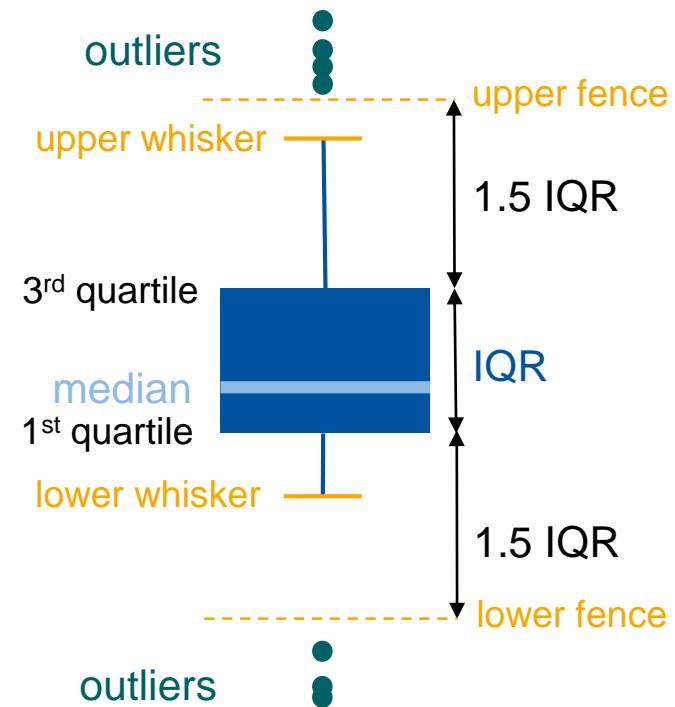
Spurious Correlations

US spending on science, space, and technology
correlates with
Suicides by hanging, strangulation and suffocation



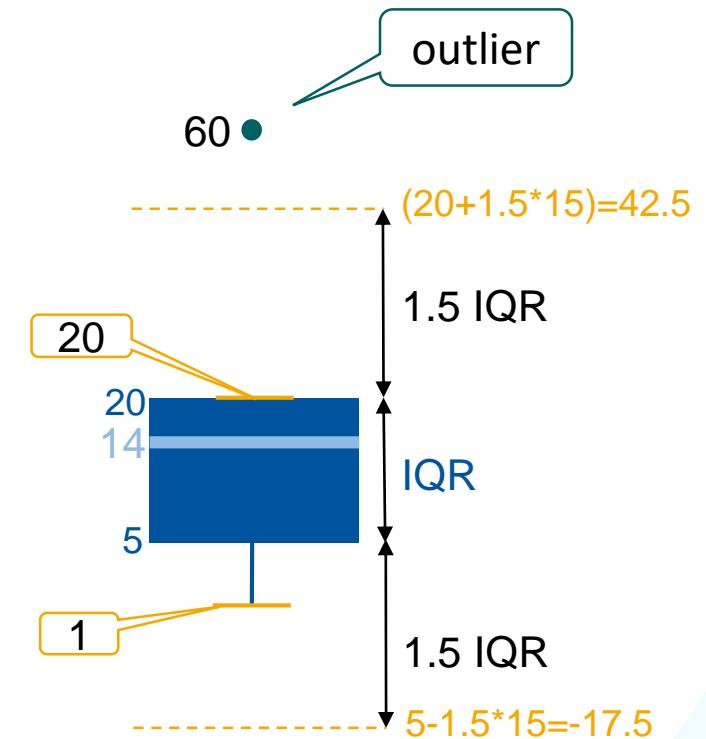
Box Plot

- Median value (middle), depicted by bar
- IQR – Interquartile Range (covers 50% of middle instances), depicted by box
- Upper fence – 3^{rd} quartile + 1.5 IQR
Upper whisker – maximal value below upper fence
- Lower fence – 1st quartile - 1.5 IQR
Lower whisker – minimal value above lower fence
- Outliers – drawn separately



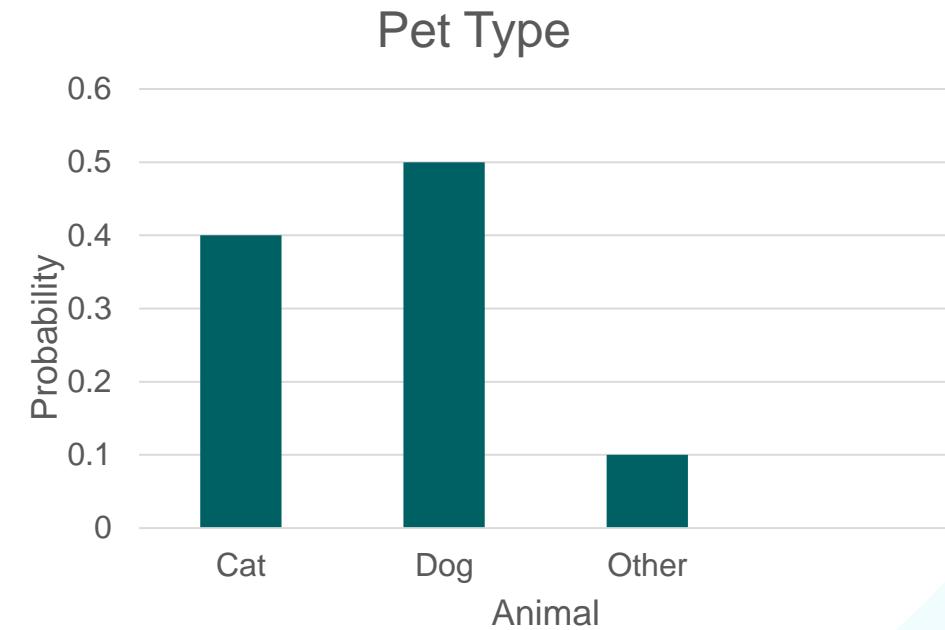
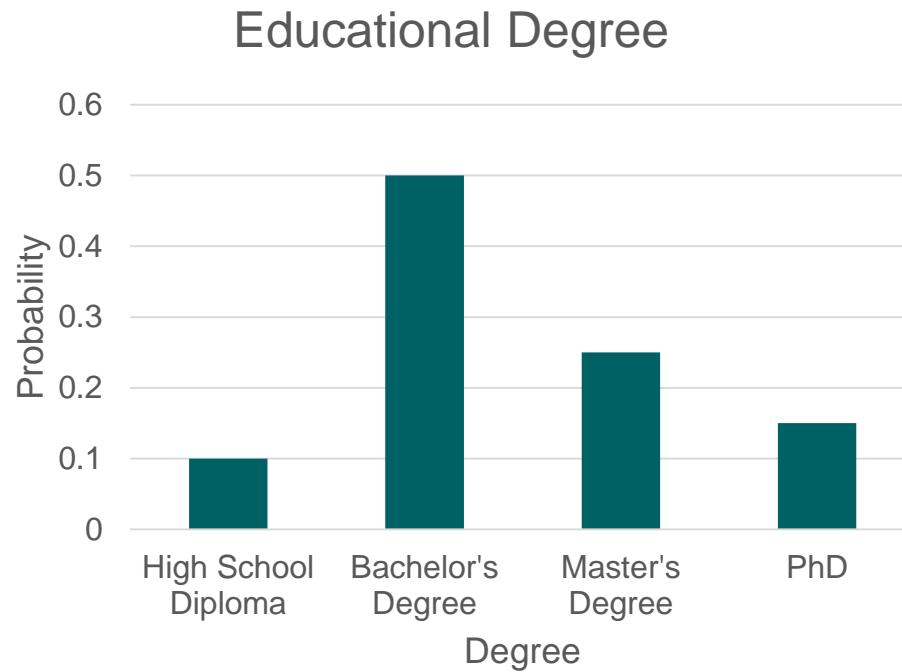
Box Plot - Example

Index	x	
		• Median: 14
1	1	• 1 st quartile: 5
2	3	$x_n \text{ with } n = \lceil \frac{25}{100} \cdot 11 \rceil = \lceil 2.75 \rceil = 3$
3	5	
4	7	• 3 rd quartile: 20
5	10	$x_n \text{ with } n = \lceil \frac{75}{100} \cdot 11 \rceil = \lceil 8.25 \rceil = 9$
6	14	
7	15	• IQR: $20 - 5 = 15$
8	17	
9	20	• Upper whisker: maximal value below $20 + 1.5 \cdot 15 = 42.5 \rightarrow 20$
10	20	• Lower whisker: minimal value above $5 - 1.5 \cdot 15 = -17.5 \rightarrow 1$
11	60	



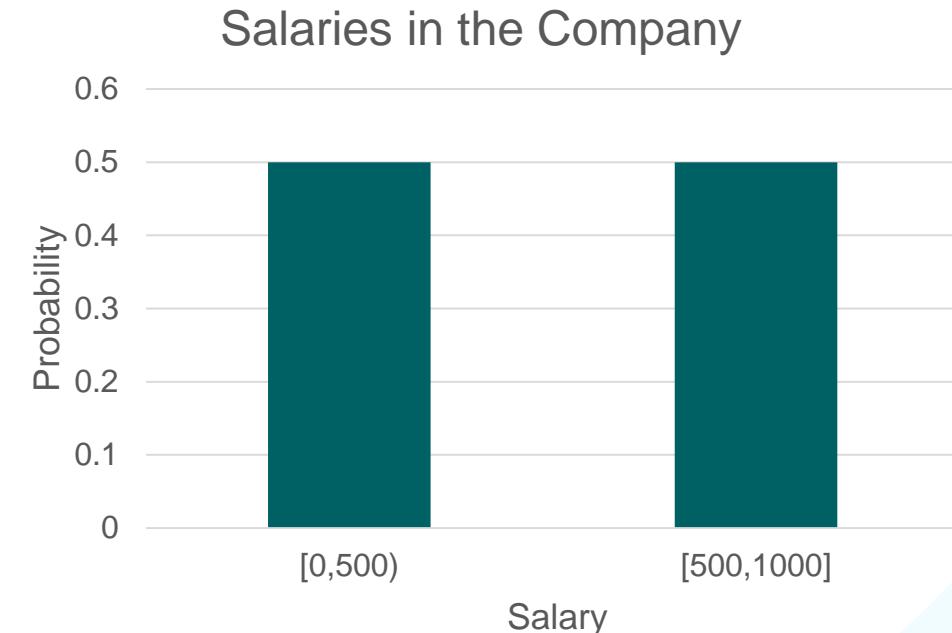
Histograms - Visualizations of Distributions

Categorical features



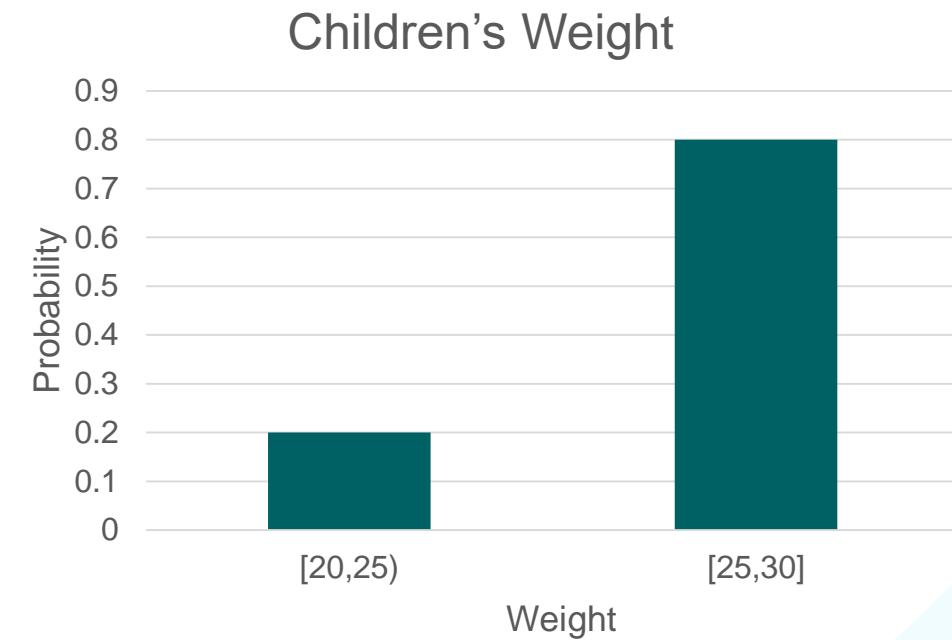
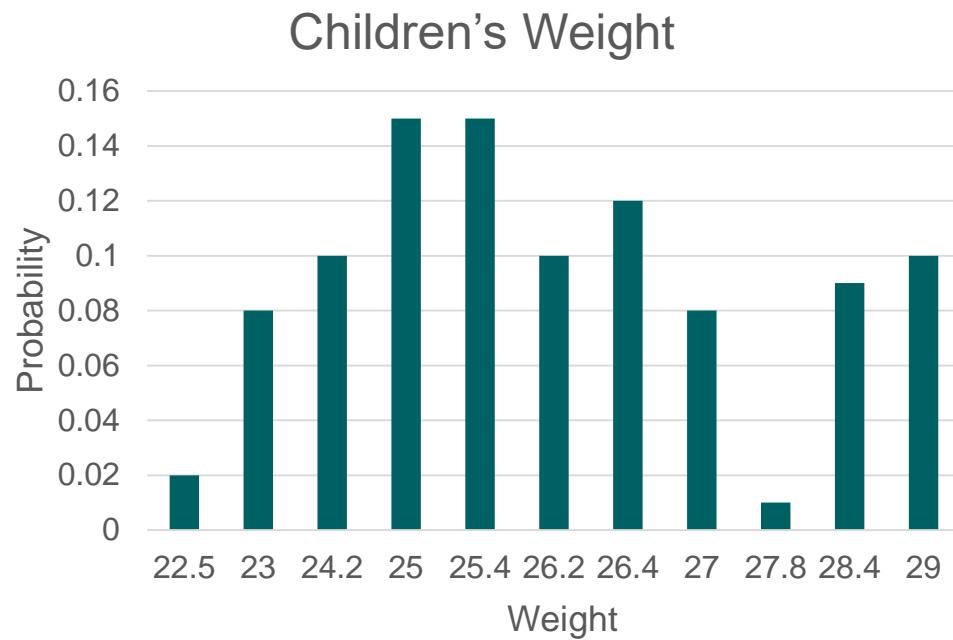
Histograms - Visualizations of Distributions

Continuous features



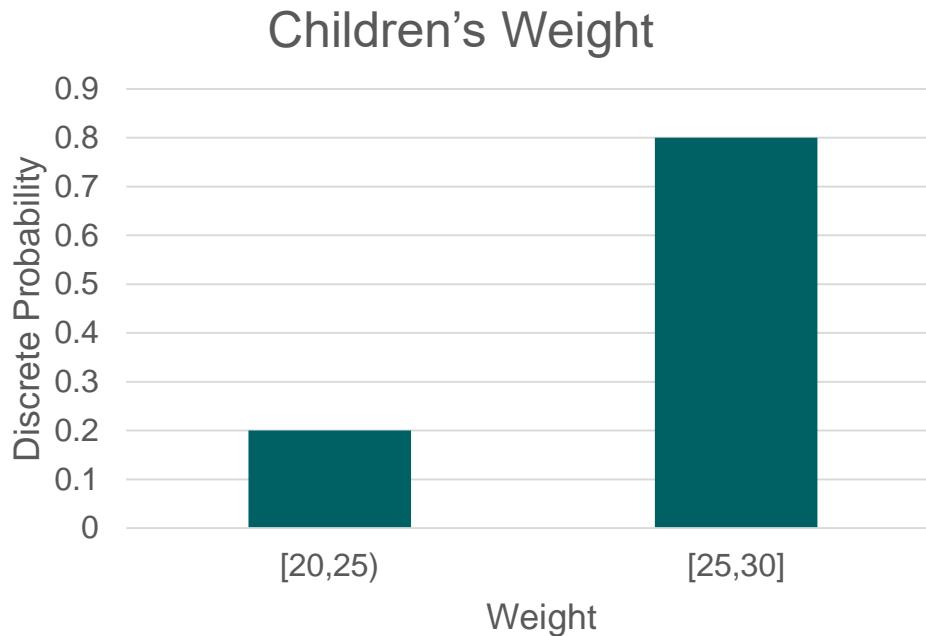
Histograms - Visualizations of Distributions

Continuous features

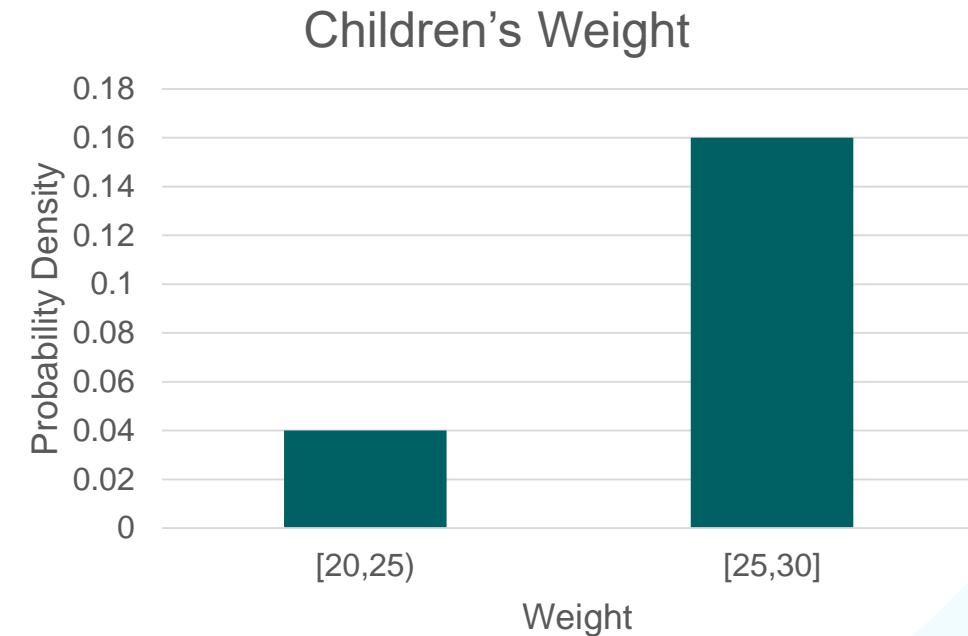


Histograms – Watch out for Normalization!

- Discrete probability distribution over intervals
- Normalized over population
- Sums to 1 [over discrete intervals]
- Continuous probability density over values
- Normalized over population **and** bin width
- Integrates to 1 [over continuous range]



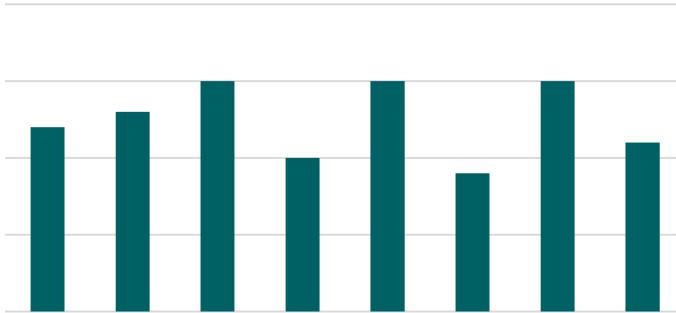
“probability that a child’s weight is between 20 to 25 or between 25 to 30”



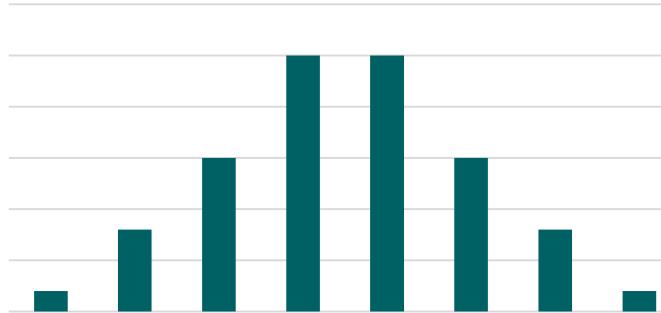
“probability of a child’s weight over the reals”

Different Types of Histograms

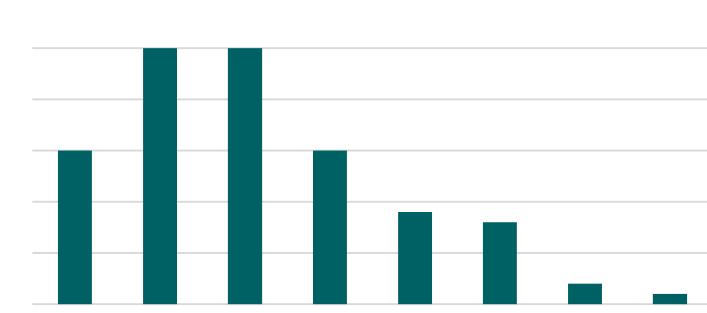
Uniform



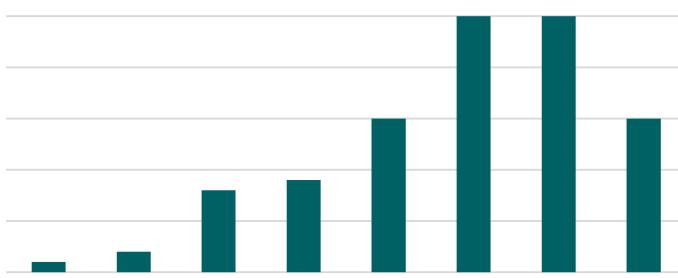
Normal (unimodal)



Normal (skewed right)



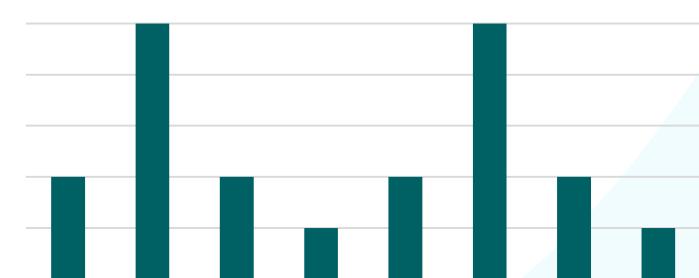
Normal (skewed left)



Exponential

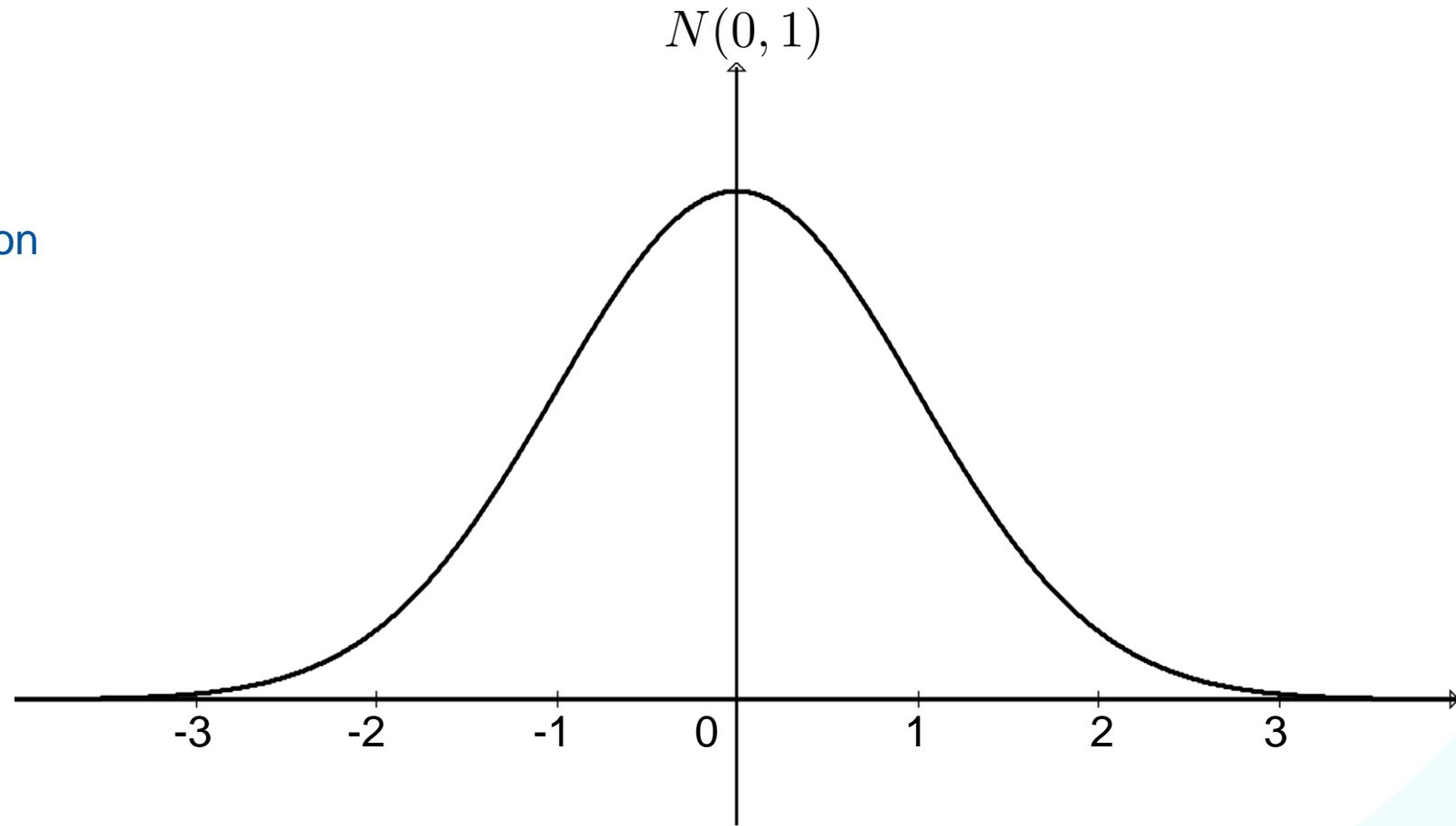


Multimodal



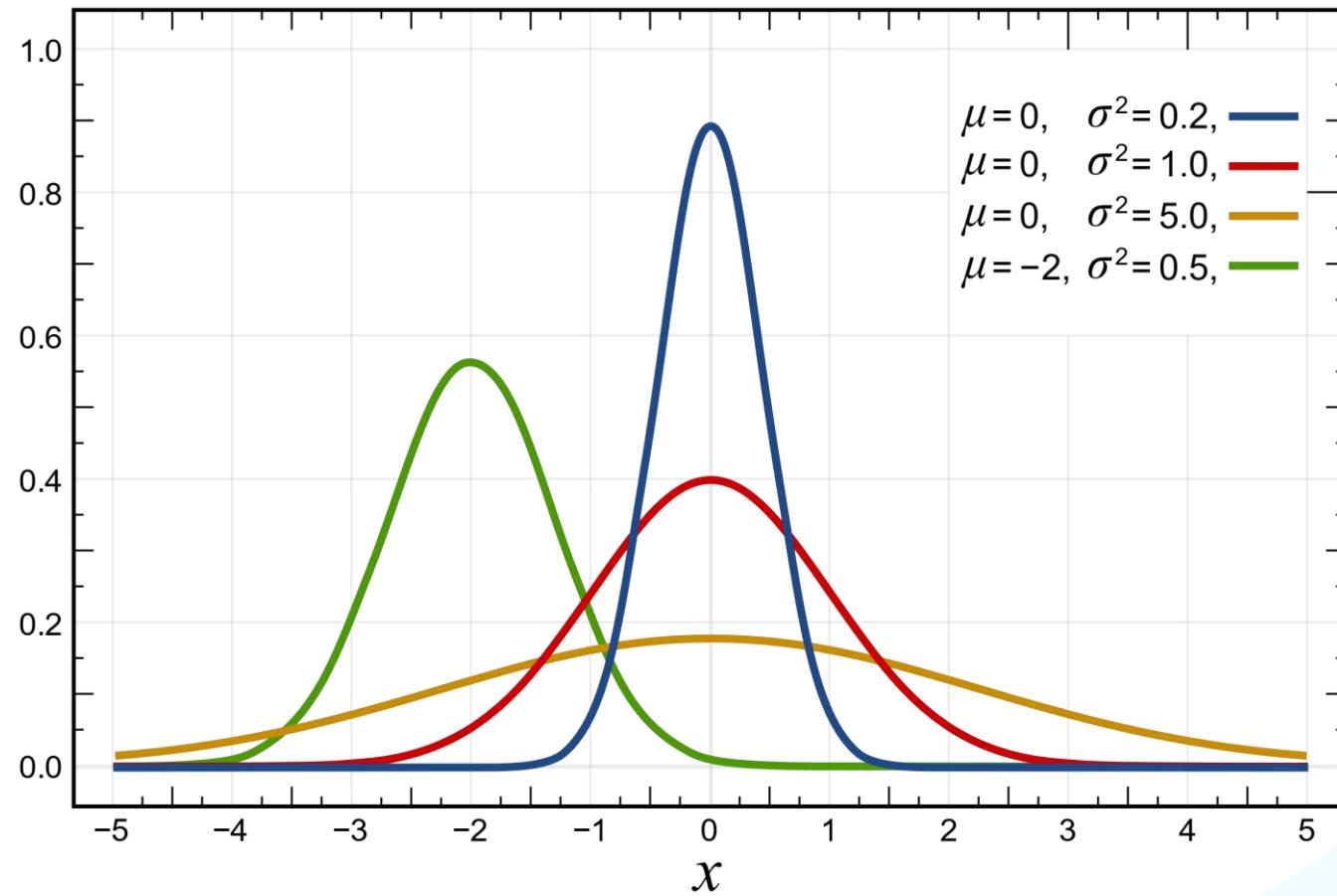
Normal (Gaussian) Distribution

- $N(\mu, \sigma^2)$
- μ - mean
- σ - standard deviation



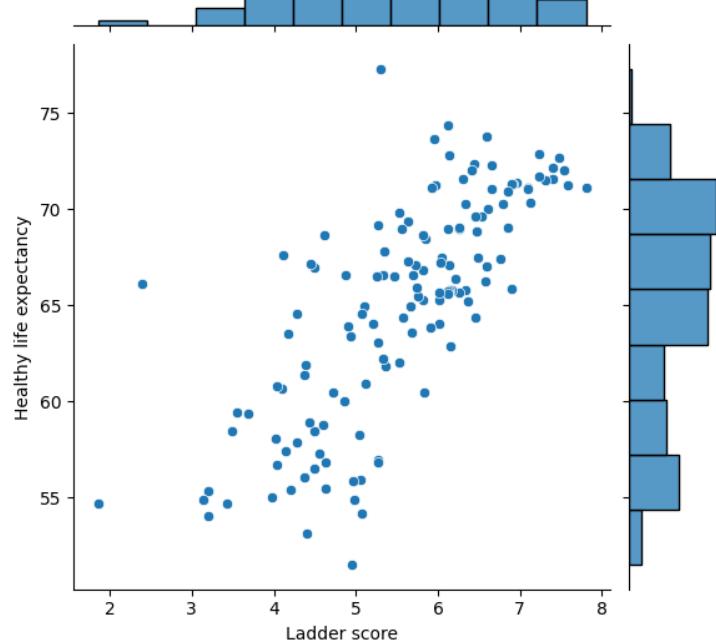
Normal (Gaussian) Distribution

- $N(\mu, \sigma^2)$
- μ - mean
- σ - standard deviation

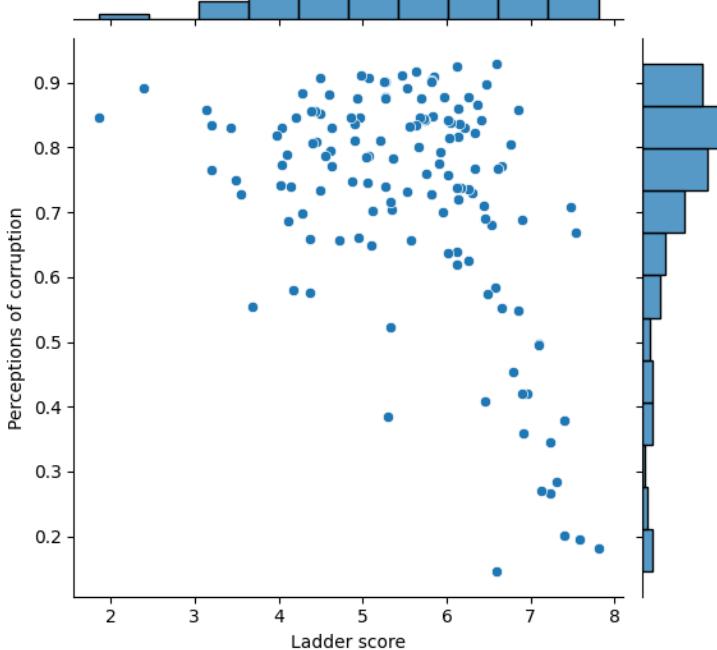


Scatter Plot - Correlation

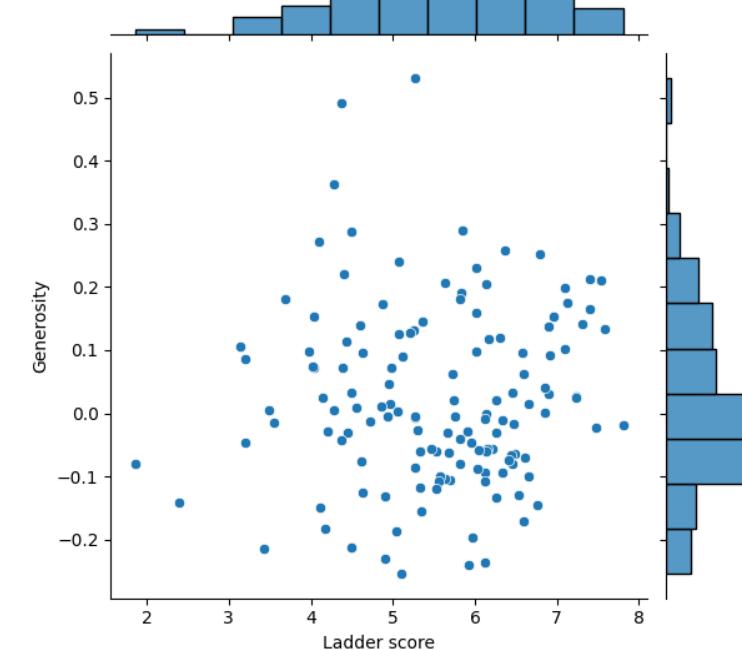
World Happiness Report 2023 [3]



Positive correlation

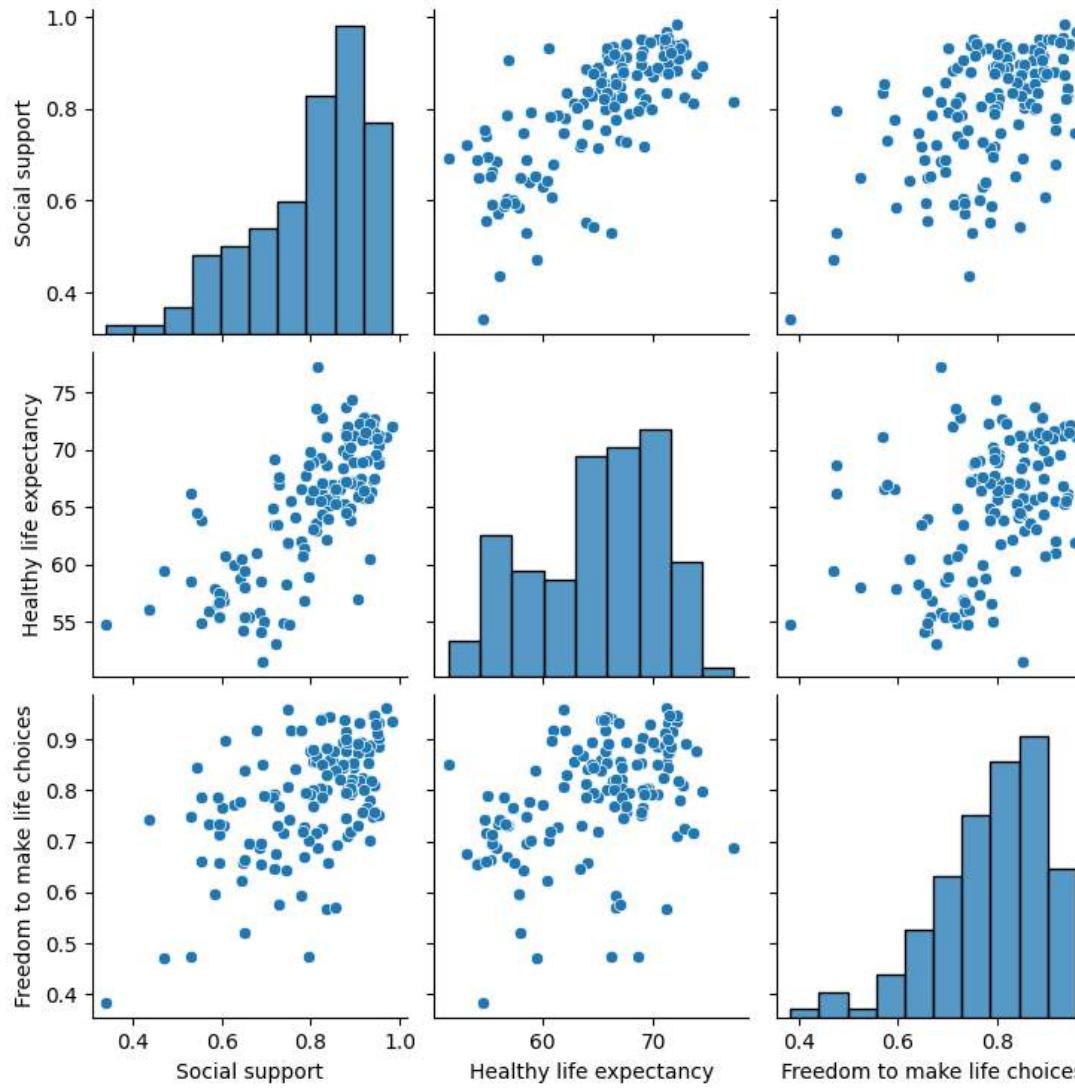


Negative correlation



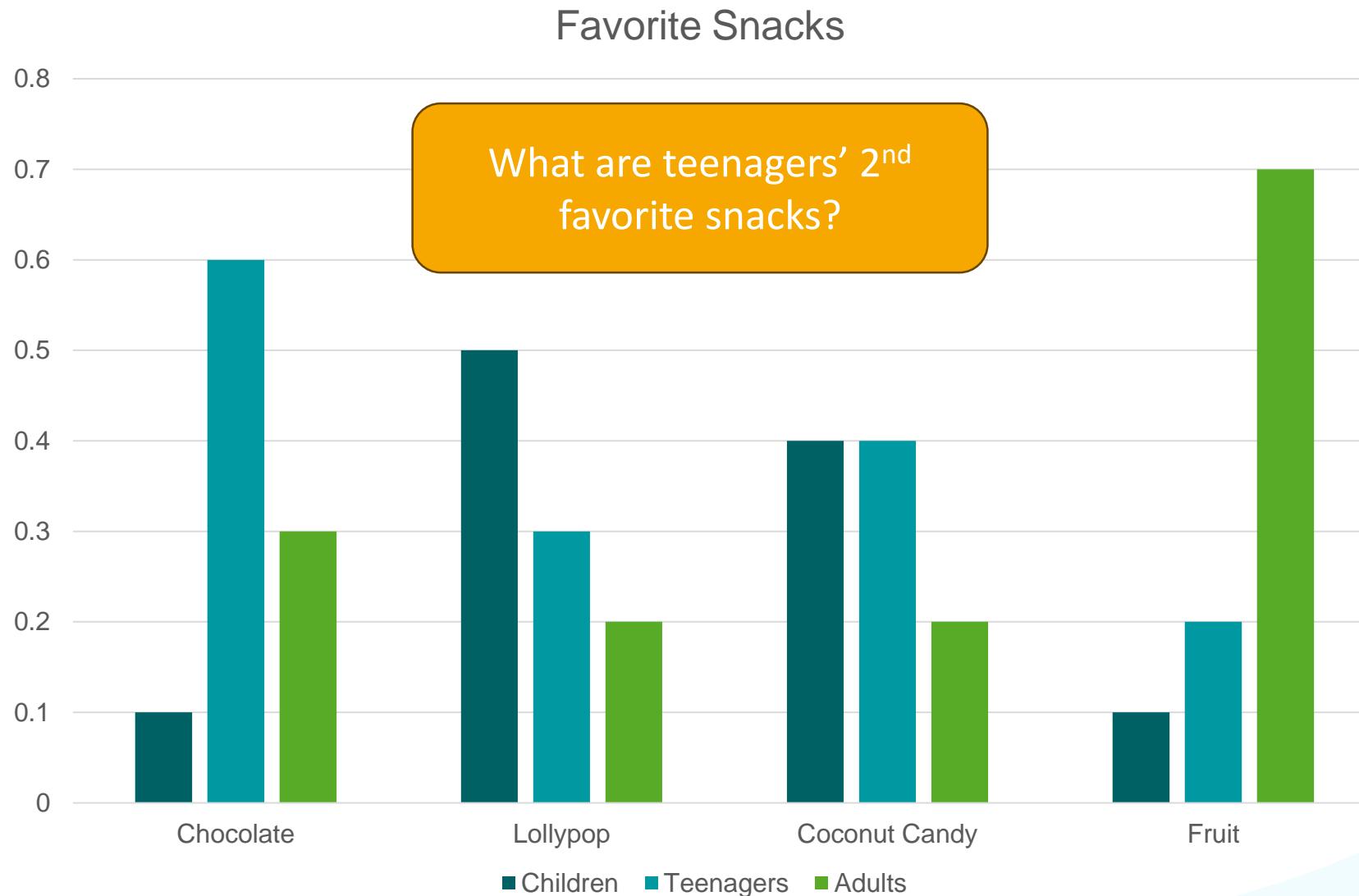
No correlation

Scatter Plot Matrix

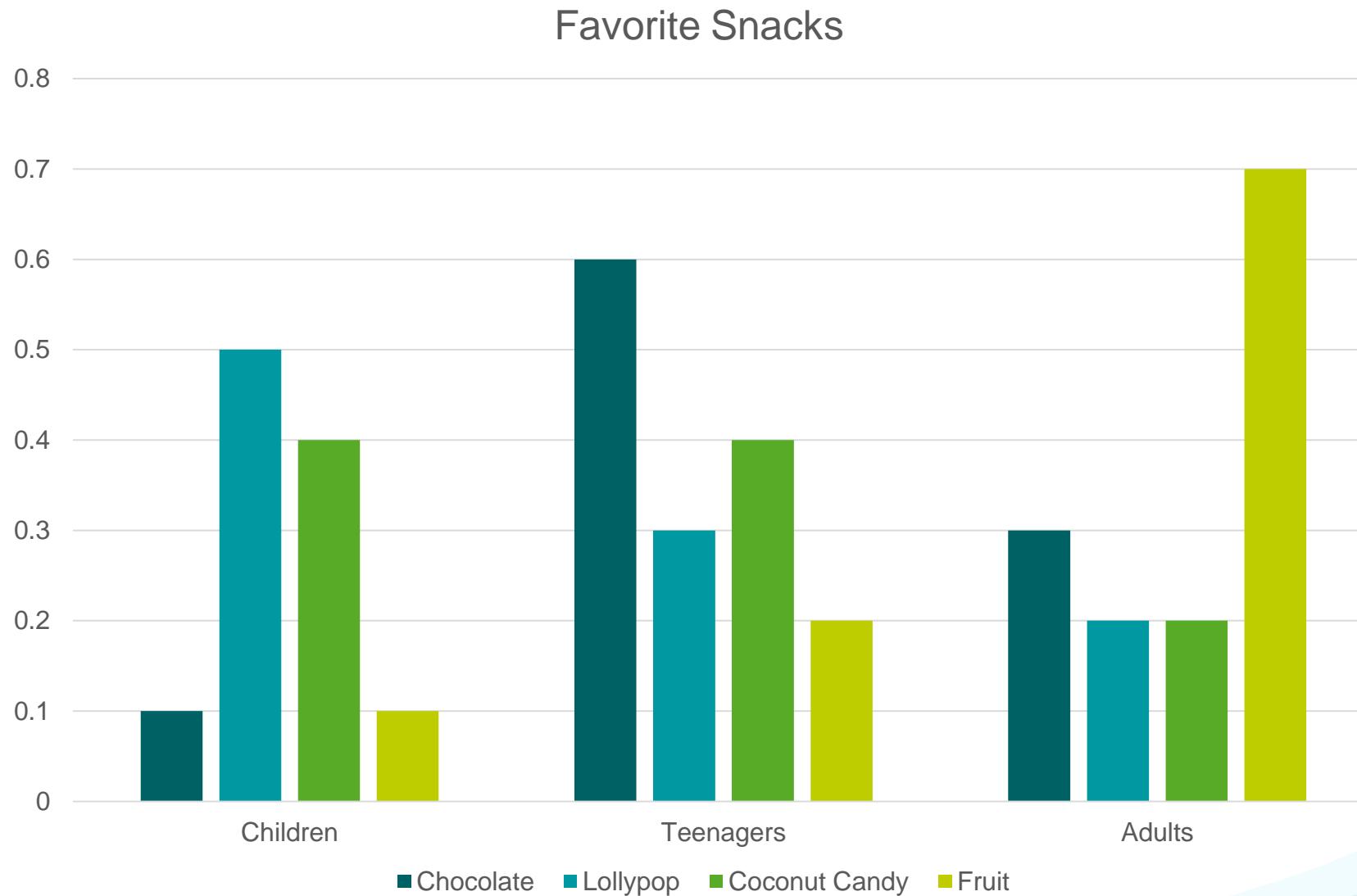


World Happiness Report 2023

Faceting: Collection of Bar Plots

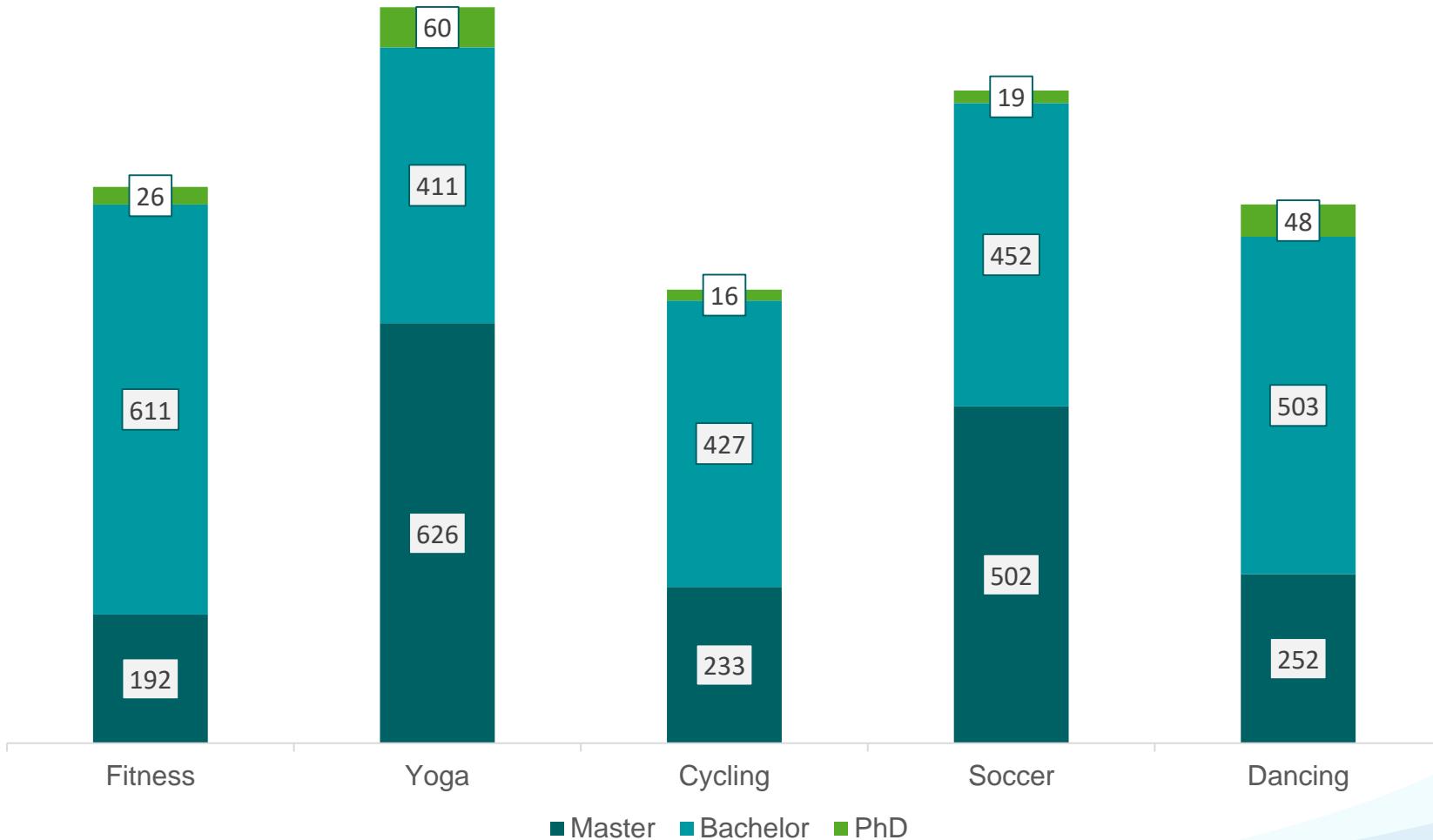


Faceting: Change of Focus



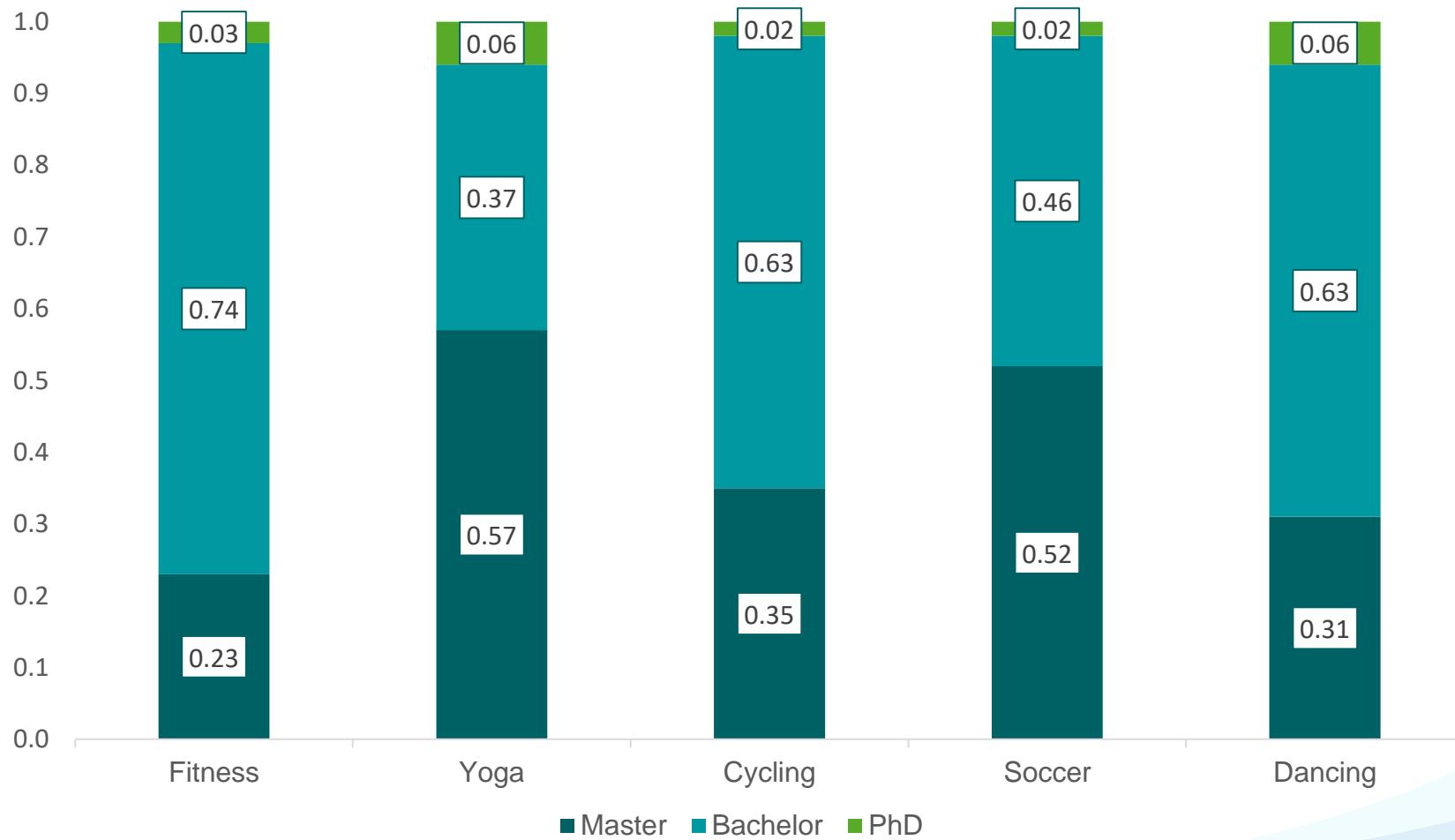
Stacked Bar Plots

University Sports Class Participation

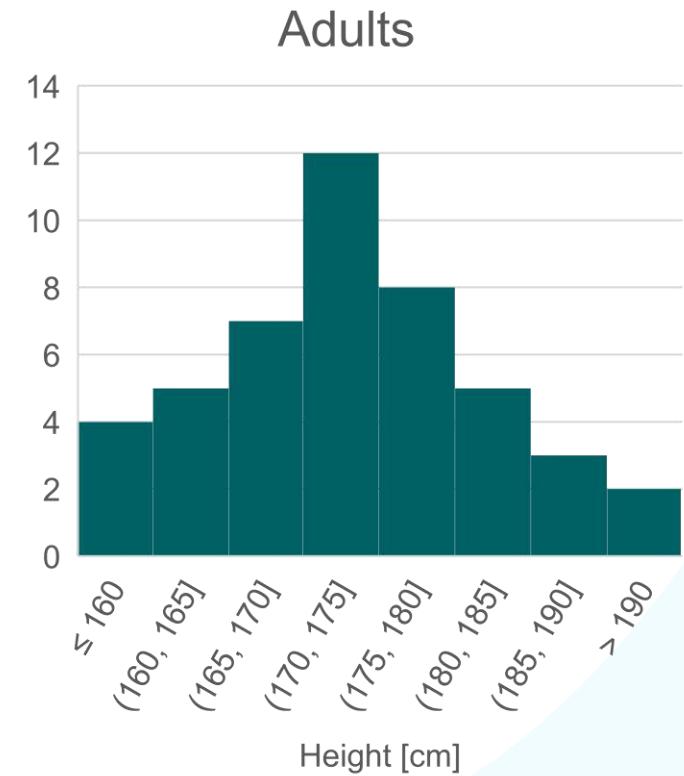
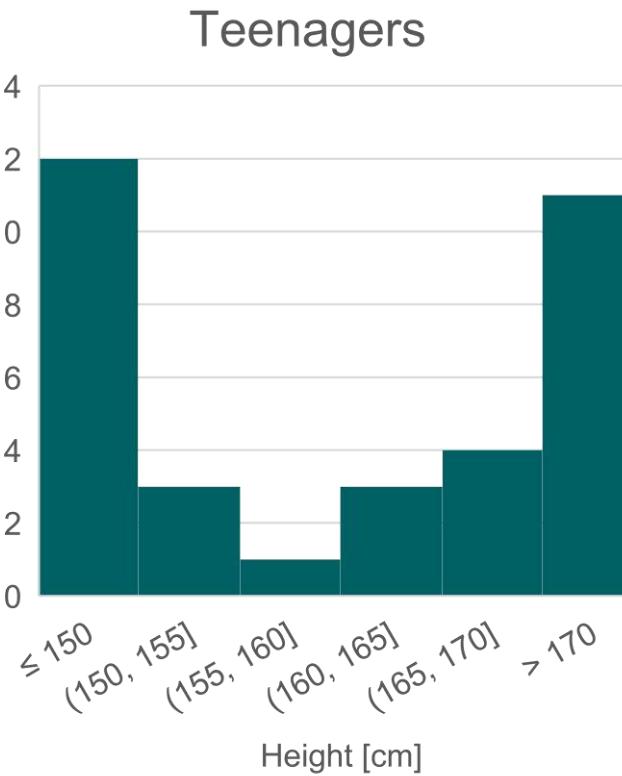
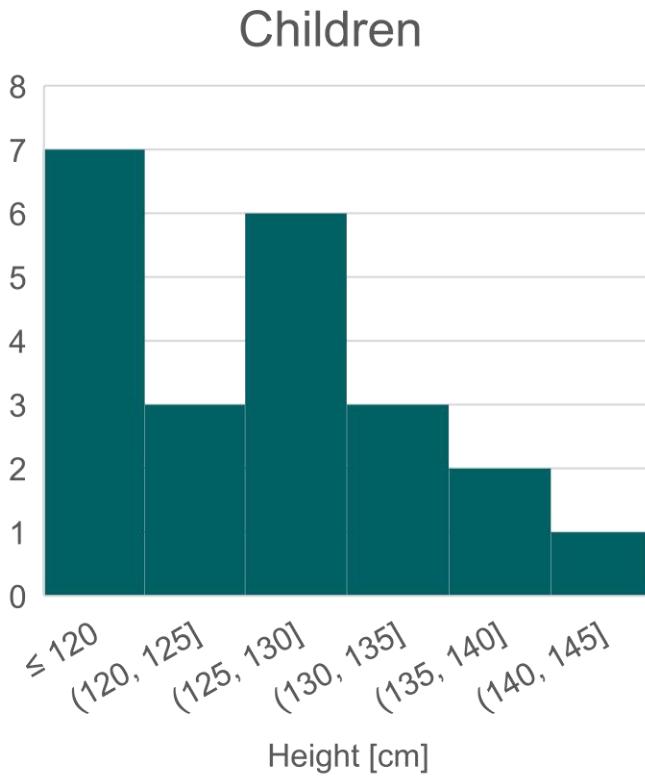


Stacked Bar Plots

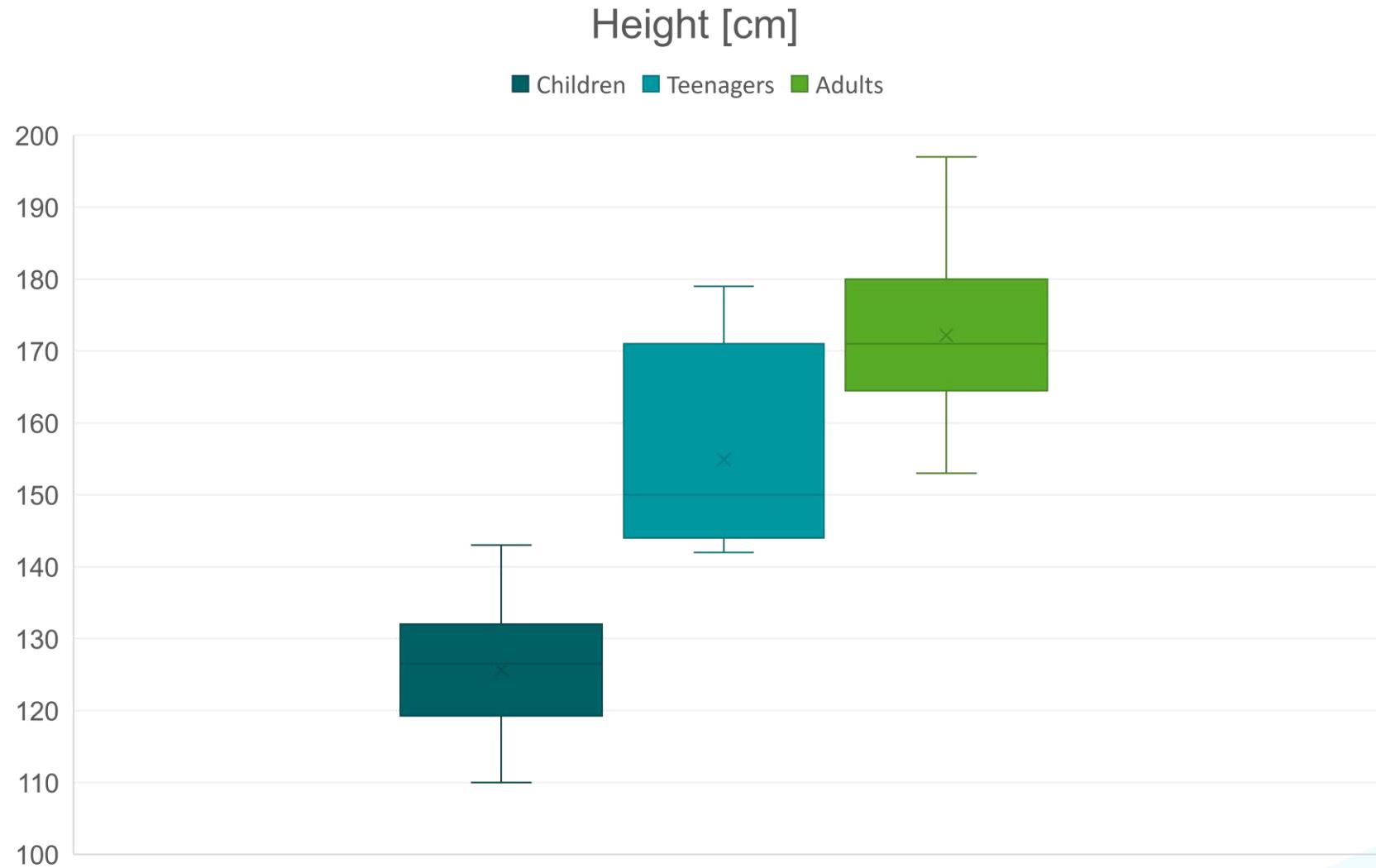
University Sports Class Participation



Collection of Histograms

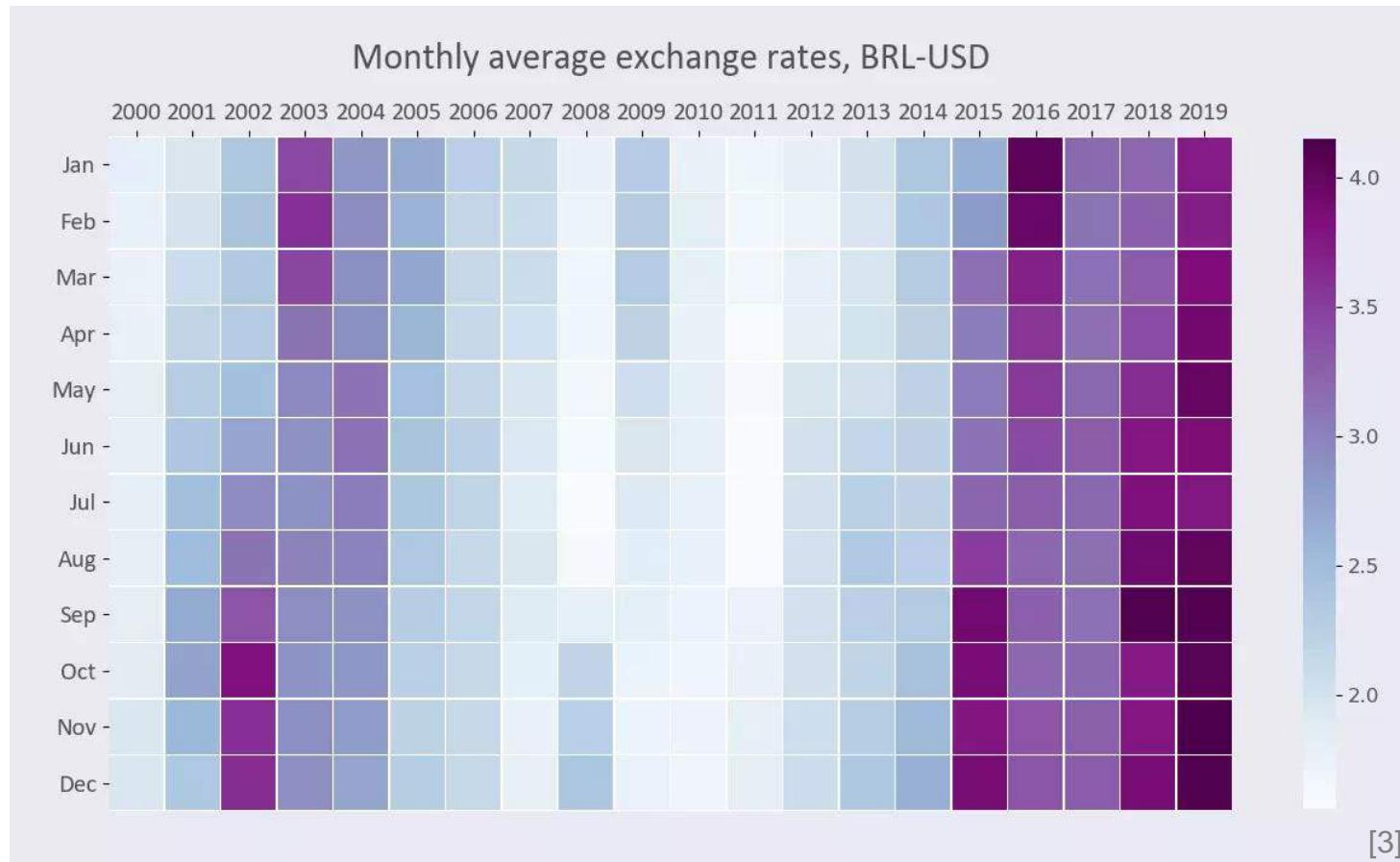


Collection of Box Plots



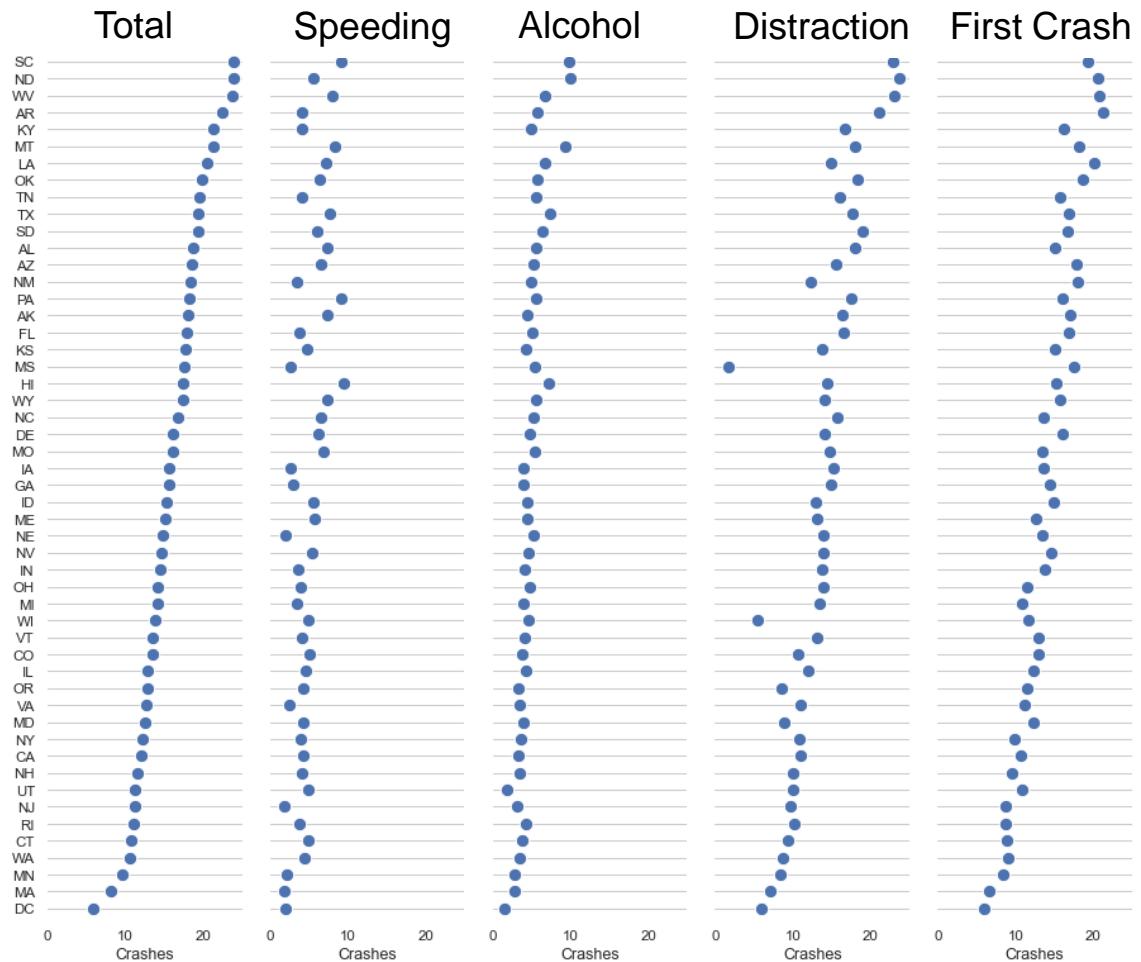
Advanced Visualizations - Examples

Heatmap



Advanced Visualizations - Examples

Dot Plot with Several Variables



Fatal Collisions per Billion Miles
Comparison of US States

Value of Good Visualizations

- Understanding and analyzing data more quickly and easily
- Communicating to others more effectively
- Identifying outliers, anomalies and other unexpected patterns in data
- Making clear decisions – identifying key insights

Feature Transformations

Dealing with Categorical Features

f_1	f_2	f_3	class
high	true	88	A
high	false	76	B
medium	false	32	B
low	true	89	C
high	true	21	C
medium	true	45	A

Categorical
descriptive
features (f_1, f_2)

Categorical
target feature

One-Hot Encoding

f₁	f₂	f₃	class
high	true	88	A
high	false	76	B
medium	false	32	B
low	true	89	C
high	true	21	C
medium	true	45	A

Standard one-hot encoding: introduce a 0/1 feature for every possible value

- high – (1,0,0)
- medium – (0,1,0)
- low – (0,0,1)

One-Hot Encoding: Standard

f_1 - high	f_1 - medium	f_1 - low	f_2	f_3	class
1	0	0	true	88	A
1	0	0	false	76	B
0	1	0	false	32	B
0	0	1	true	89	C
1	0	0	true	21	C
0	1	0	true	45	A

Standard one-hot encoding: introduce a 0/1 feature for every possible value

- high – (1,0,0)
- medium – (0,1,0)
- low – (0,0,1)

One-Hot Encoding: Common Variant

f_1 - dummy ₀	f_1 - dummy ₁	f_2	f_3	class
1	0	true	88	A
1	0	false	76	B
0	1	false	32	B
0	0	true	89	C
1	0	true	21	C
0	1	true	45	A

k-1 one-hot encoding:

- high – (1,0)
- medium – (0,1)
- low – (0,0)

+ preferable where co-linearity of features is problematic
- introduces asymmetry, e.g., see *low*

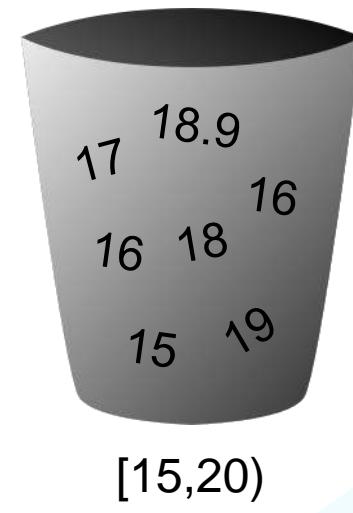
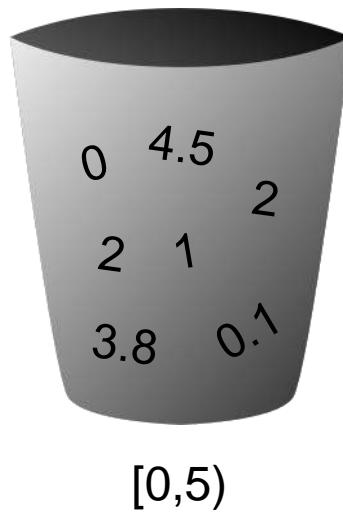
One-Hot Encoding – Special Cases

f_1 - high	f_1 - medium	f_1 - low	f_2	f_3	class
1	0	0	true	88	A
1	0	0	false	76	B
0	1	0	false	32	B
0	0	1	true	89	C
1	0	0	true	21	C
0	1	0	true	45	A

- **Binary values** (true, false) can be translated to a single numeric value (1, 0) [example of k-1 encoding]
- Note that categorical variables with a **clear order (ordinal)** may be translated to a single numeric value (e.g., excellent = 1.0, good = 0.7, average = 0.5, poor = 0.3, horrible = 0.0)

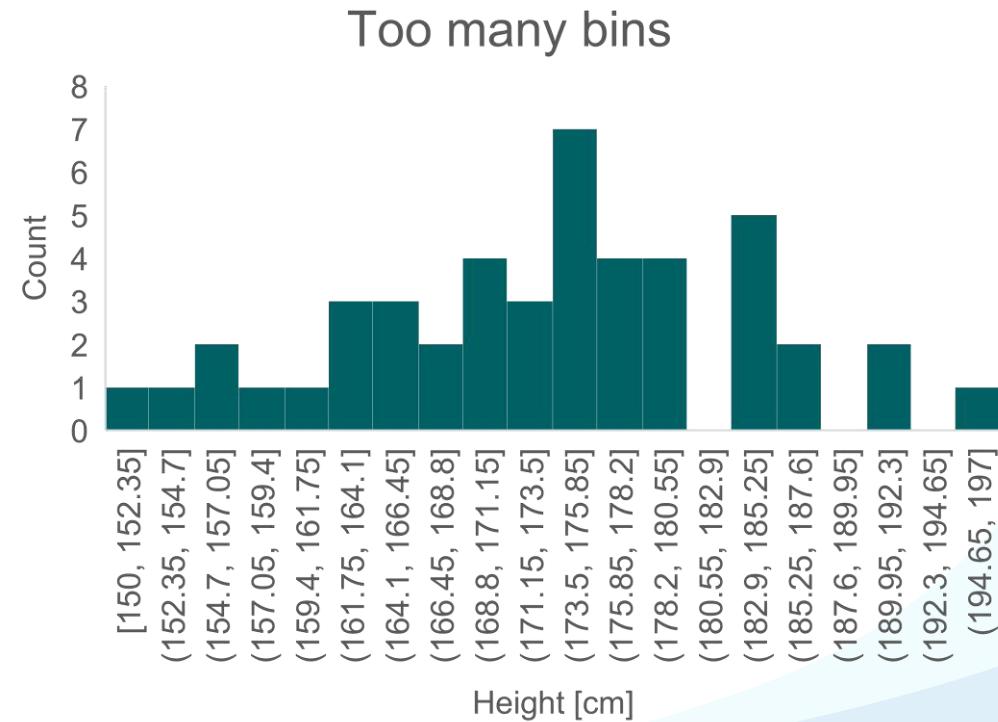
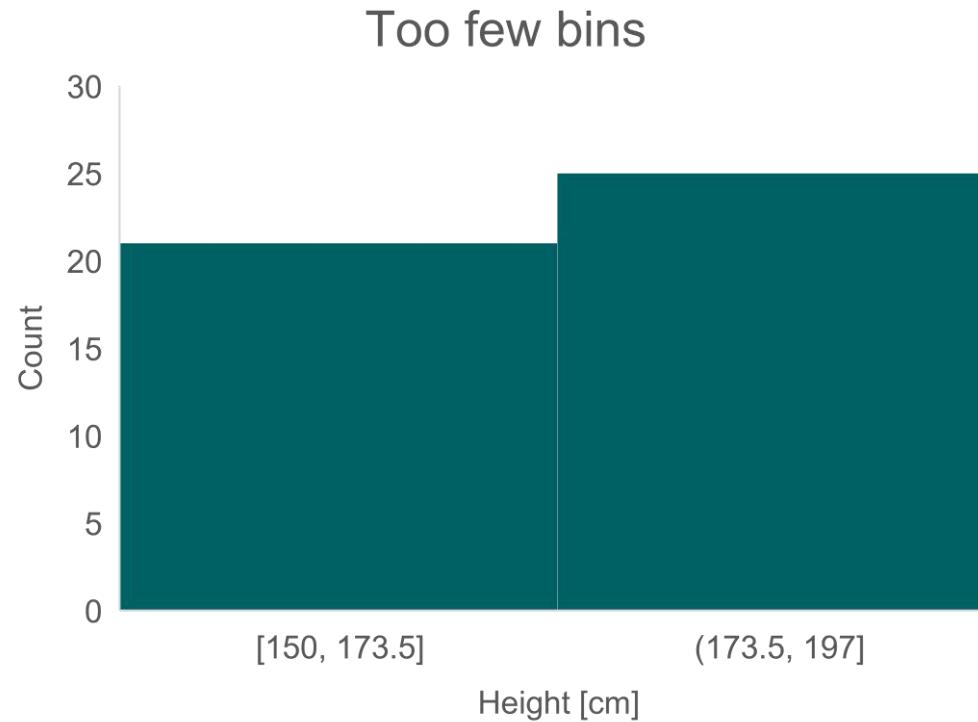
Dealing with Continuous Features - Binning

- Binning is used to transform **continuous** features **into** categorical
- A **bin** is a range, e.g., [0,5), [5,10), [10,15), [15,20)
- Choosing the right bins (their number and size) is crucial (e.g., to create meaningful **histograms**)



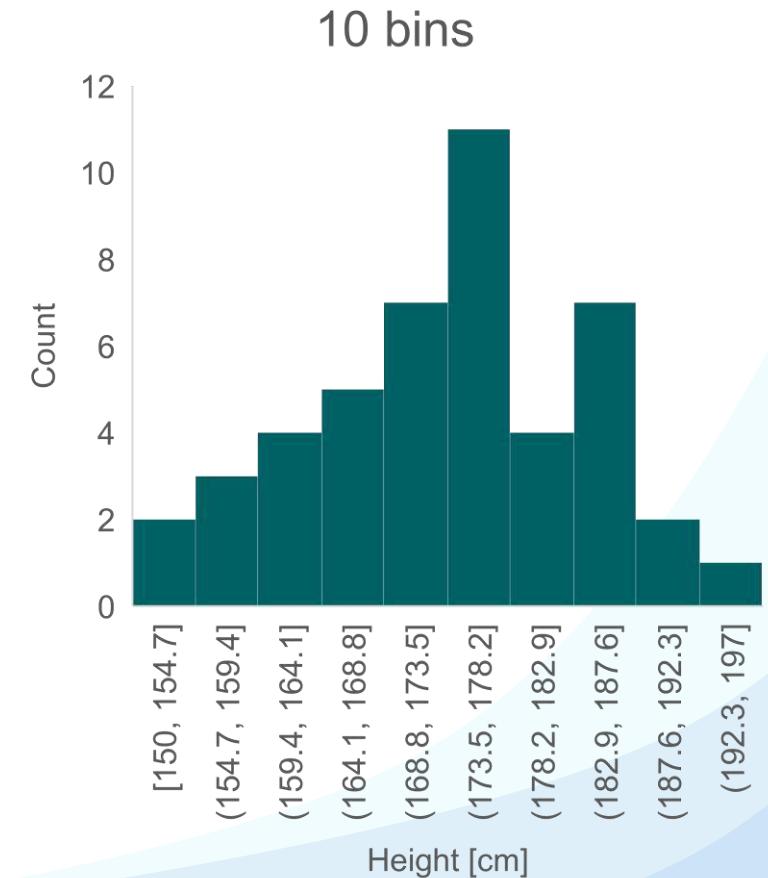
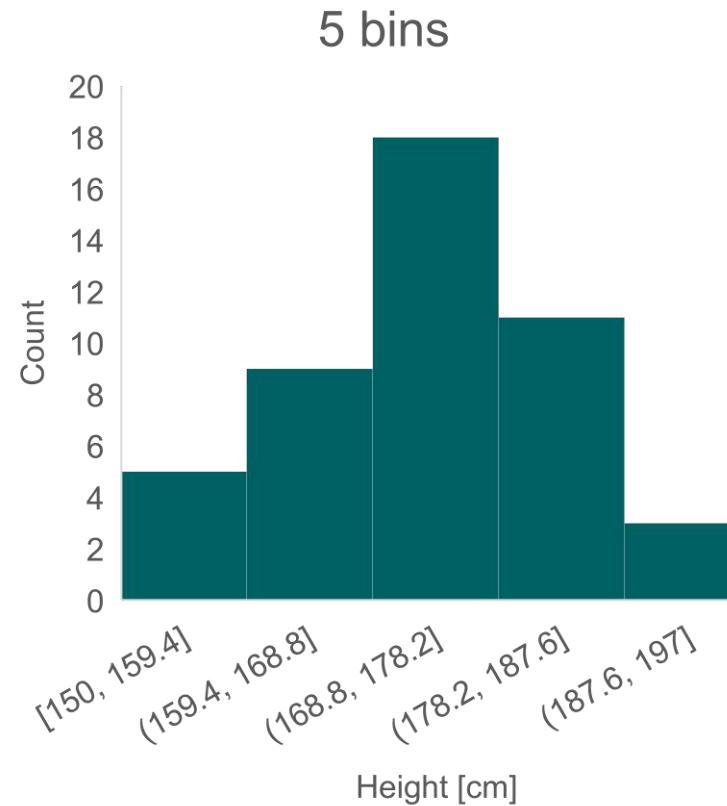
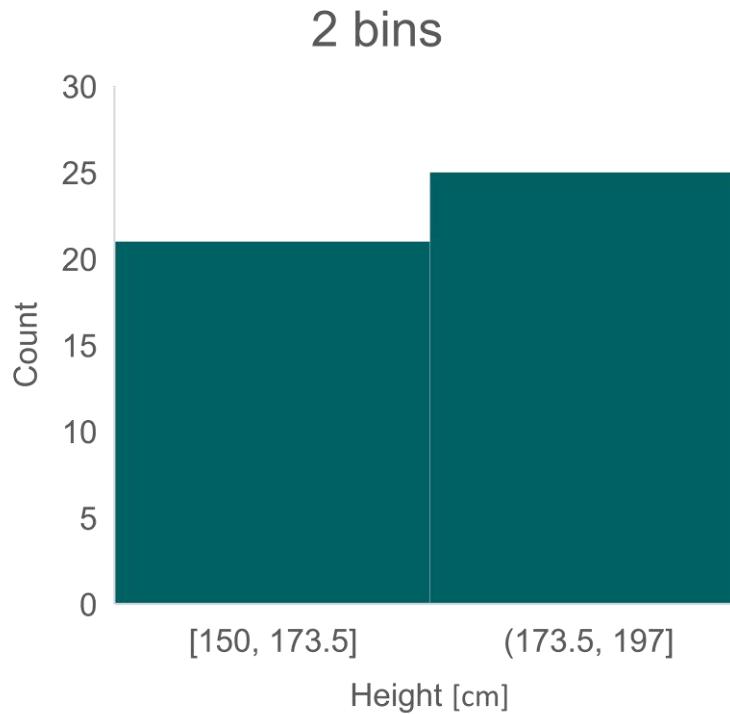
Binning – Number of Bins

- Too few bins may lead to the loss of information ([underfitting](#))
- Too many bins may lead to sparseness – bins that are empty or have just a few instances ([overfitting](#))



Equal Width Binning

Bins have a **fixed width**, but the number of items per bin may vary



Equal Width Binning - Example

Tree Age [years]	Tree Height [m]
9	26
51	96
47	61
77	118
64	91
2	6
48	60
13	31
9	11
29	86
90	107
80	88

Apply [equal width binning](#) to the feature **Tree Height** with a bin width of [29](#).
The lowest bin boundaries should coincide with the smallest value.

Equal Width Binning - Example

Tree Age [years]	Tree Height [m]
9	26
51	96
47	61
77	118
64	91
2	6
48	60
13	31
9	11
29	86
90	107
80	88

Apply equal width binning to the feature Tree Height with a bin width of 29.
The lowest bin boundaries should coincide with the smallest value.

1. Sort the data

Equal Width Binning - Example

Tree Age [years]	Tree Height [m]	Apply equal width binning to the feature Tree Height with a bin width of 29. The lowest bin boundaries should coincide with the smallest value.
2	6	
9	11	
9	26	1. Sort the data
13	31	
48	60	
47	61	
29	86	
80	88	
64	91	
51	96	
90	107	
77	118	

Equal Width Binning - Example

Tree Age [years]	Tree Height [m]	Apply equal width binning to the feature Tree Height with a bin width of 29. The lowest bin boundaries should coincide with the smallest value.
2	6	
9	11	
9	26	1. Sort the data
13	31	2. Distribute elements to bins
48	60	
47	61	
29	86	
80	88	
64	91	
51	96	
90	107	
77	118	

Equal Width Binning - Example

Tree Age [years]	Tree Height [m]
2	6
9	11
9	26
13	31
48	60
47	61
29	86
80	88
64	91
51	96
90	107
77	118

Apply equal width binning to the feature Tree Height with a bin width of 29. The lowest bin boundaries should coincide with the smallest value.

1. Sort the data
2. Distribute elements to bins:
 $6+29=35 \rightarrow [6,35)$

Equal Width Binning - Example

Tree Age [years]	Tree Height [m]
2	6
9	11
9	26
13	31
48	60
47	61
29	86
80	88
64	91
51	96
90	107
77	118

Apply equal width binning to the feature Tree Height with a bin width of 29. The lowest bin boundaries should coincide with the smallest value.

1. Sort the data
2. Distribute elements to bins:
 $6+29=35 \rightarrow [6,35)$
 $35+29=64 \rightarrow [35,64)$

Equal Width Binning - Example

Tree Age [years]	Tree Height [m]
2	6
9	11
9	26
13	31
48	60
47	61
29	86
80	88
64	91
51	96
90	107
77	118

Apply equal width binning to the feature Tree Height with a bin width of 29. The lowest bin boundaries should coincide with the smallest value.

1. Sort the data
2. Distribute elements to bins:
 $6+29=35 \rightarrow [6,35)$
 $35+29=64 \rightarrow [35,64)$
 $64+29=93 \rightarrow [64,93)$

Equal Width Binning - Example

Tree Age [years]	Tree Height [m]
2	6
9	11
9	26
13	31
48	60
47	61
29	86
80	88
64	91
51	96
90	107
77	118

Apply equal width binning to the feature Tree Height with a bin width of 29. The lowest bin boundaries should coincide with the smallest value.

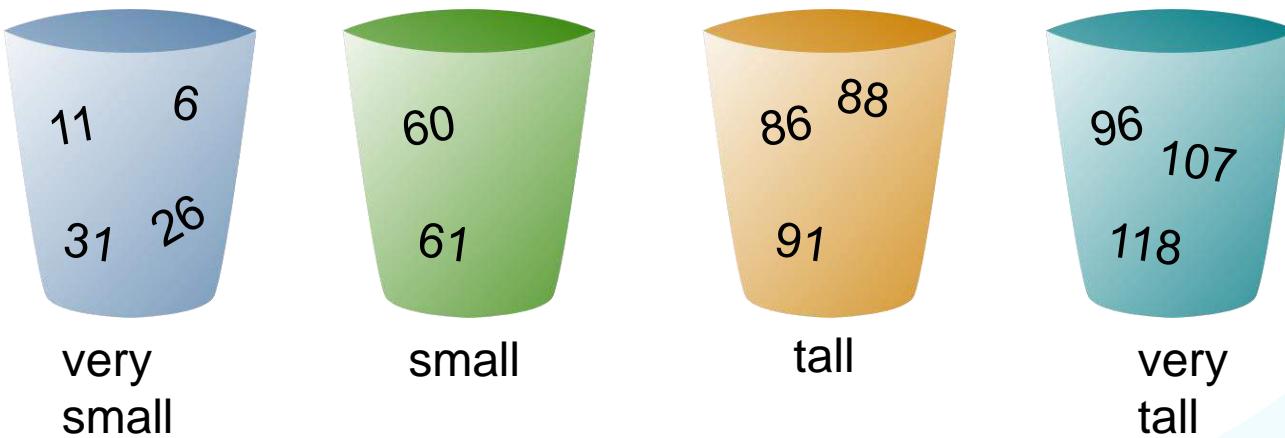
1. Sort the data
2. Distribute elements to bins:
 $6+29=35 \rightarrow [6,35)$
 $35+29=64 \rightarrow [35,64)$
 $64+29=93 \rightarrow [64,93)$
 $93+29=122 \rightarrow [93,122)$

Equal Width Binning - Example

Tree Age [years]	Tree Height [m]
2	6
9	11
9	26
13	31
48	60
47	61
29	86
80	88
64	91
51	96
90	107
77	118

Apply equal width binning to the feature Tree Height with a bin width of 29. The lowest bin boundaries should coincide with the smallest value.

1. Sort the data
2. Distribute elements to bins:

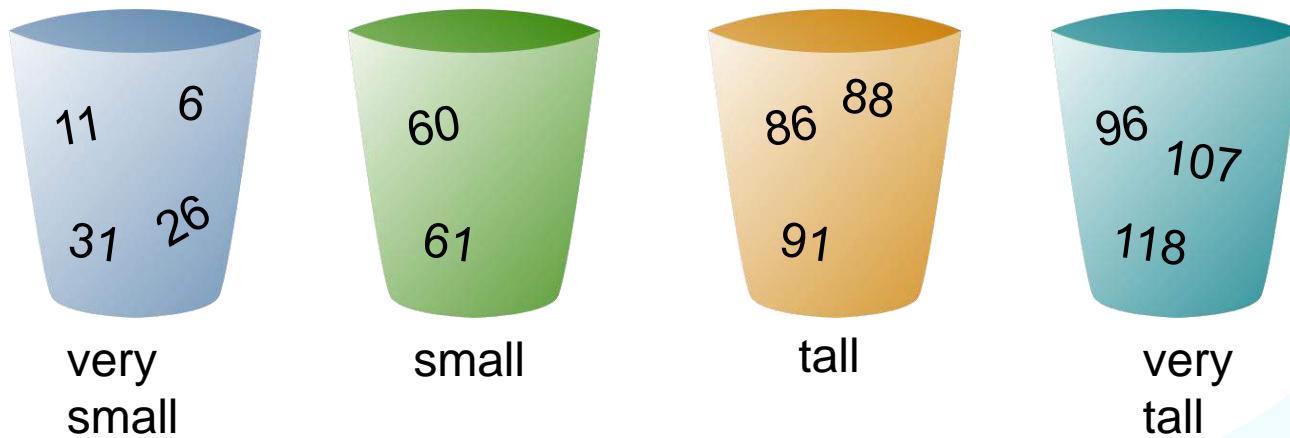


Equal Width Binning - Example

Tree Age [years]	Tree Height [m]
2	very small
9	very small
9	very small
13	very small
48	small
47	small
29	tall
80	tall
64	tall
51	very tall
90	very tall
77	very tall

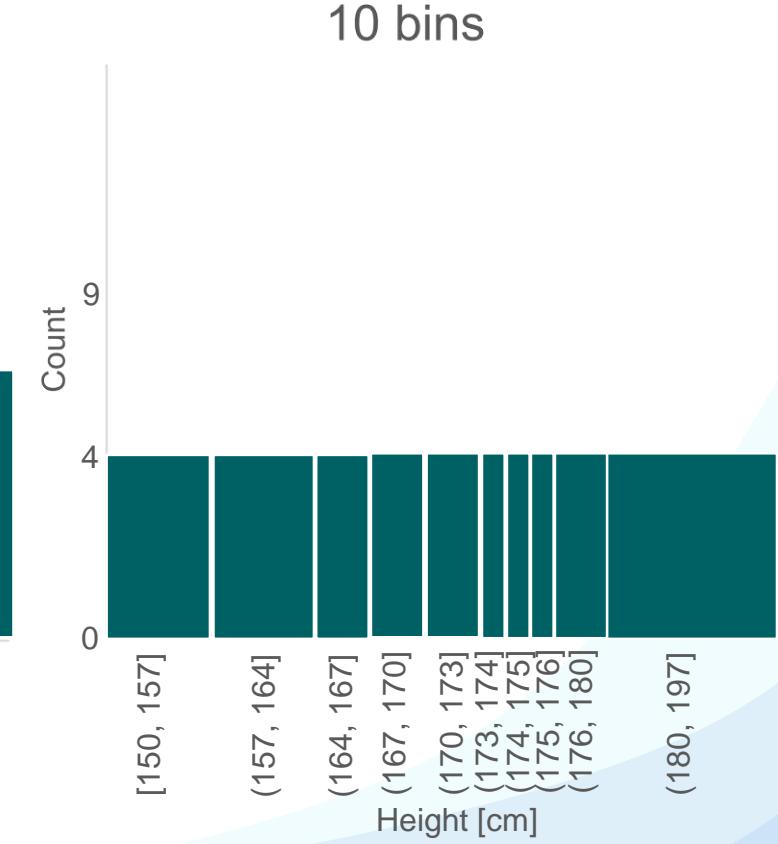
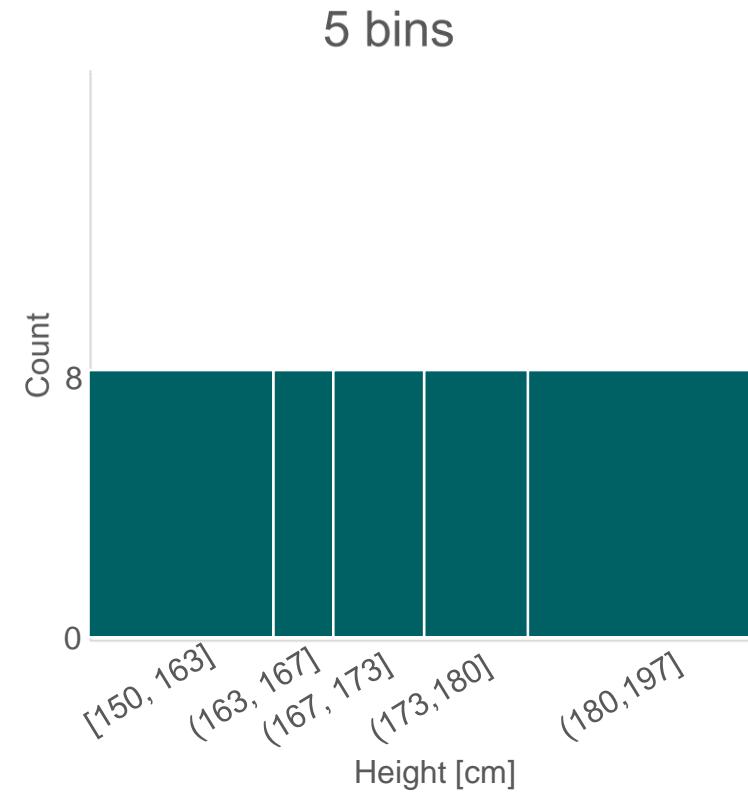
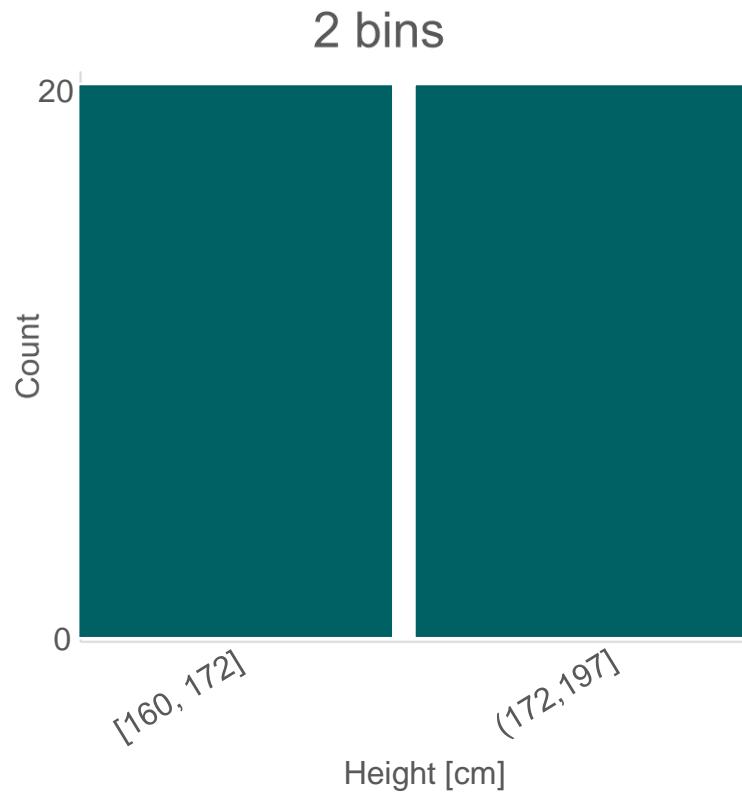
Apply [equal width binning](#) to the feature **Tree Height** with a bin width of [29](#). The lowest bin boundaries should coincide with the smallest value.

1. Sort the data
2. Distribute elements to bins:



Equal Frequency Binning

Bins vary in width, but the [numer of items per bin is fixed](#)



Equal Frequency Binning – Example

Tree Age [years]	Tree Height [m]
9	26
51	96
47	61
77	118
64	91
2	6
48	60
13	31
9	11
29	86
90	107
80	88

Apply [equal frequency binning](#) to the feature **Tree Age** with an element frequency of [4](#).

Equal Frequency Binning – Example

Tree Age [years]	Tree Height [m]
9	26
51	96
47	61
77	118
64	91
2	6
48	60
13	31
9	11
29	86
90	107
80	88

Apply [equal frequency binning](#) to the feature **Tree Age** with an element frequency of [4](#).

1. Sort the data

Equal Frequency Binning – Example

Tree Age [years]	Tree Height [m]
2	6
9	26
9	11
13	31
29	86
47	61
48	60
51	96
64	91
77	118
80	88
90	107

Apply [equal frequency binning](#) to the feature **Tree Age** with an element frequency of [4](#).

1. Sort the data

Equal Frequency Binning – Example

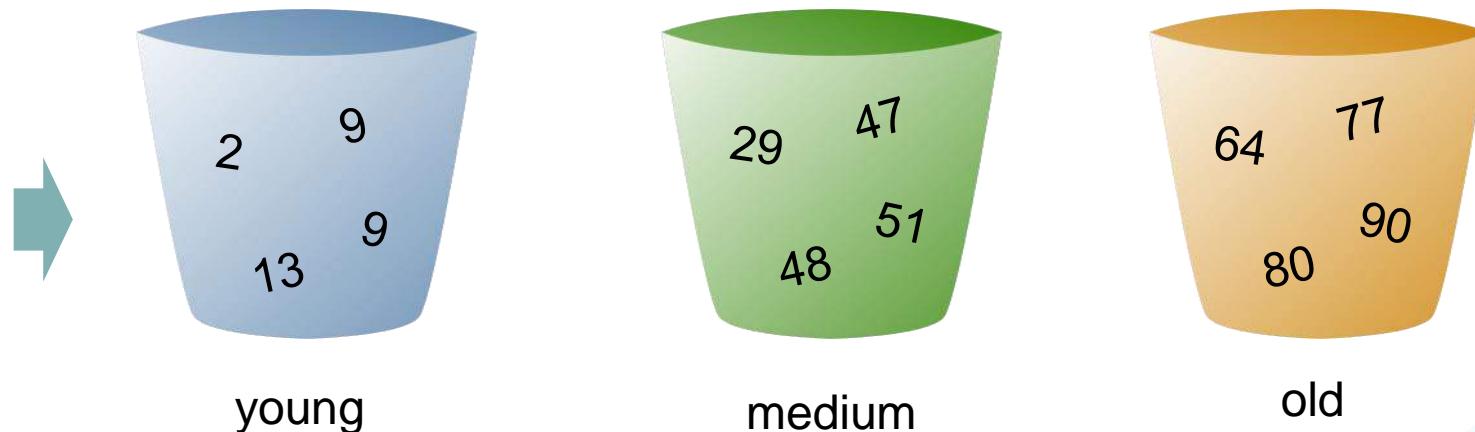
Tree Age [years]	Tree Height [m]	Apply equal frequency binning to the feature Tree Age with an element frequency of 4.
2	6	
9	26	
9	11	1. Sort the data
13	31	2. Distribute elements to bins
29	86	
47	61	
48	60	
51	96	
64	91	
77	118	
80	88	
90	107	

Equal Frequency Binning – Example

Tree Age [years]	Tree Height [m]
2	6
9	26
9	11
13	31
29	86
47	61
48	60
51	96
64	91
77	118
80	88
90	107

Apply equal frequency binning to the feature **Tree Age** with an element frequency of 4.

1. Sort the data
 2. Distribute elements to bins

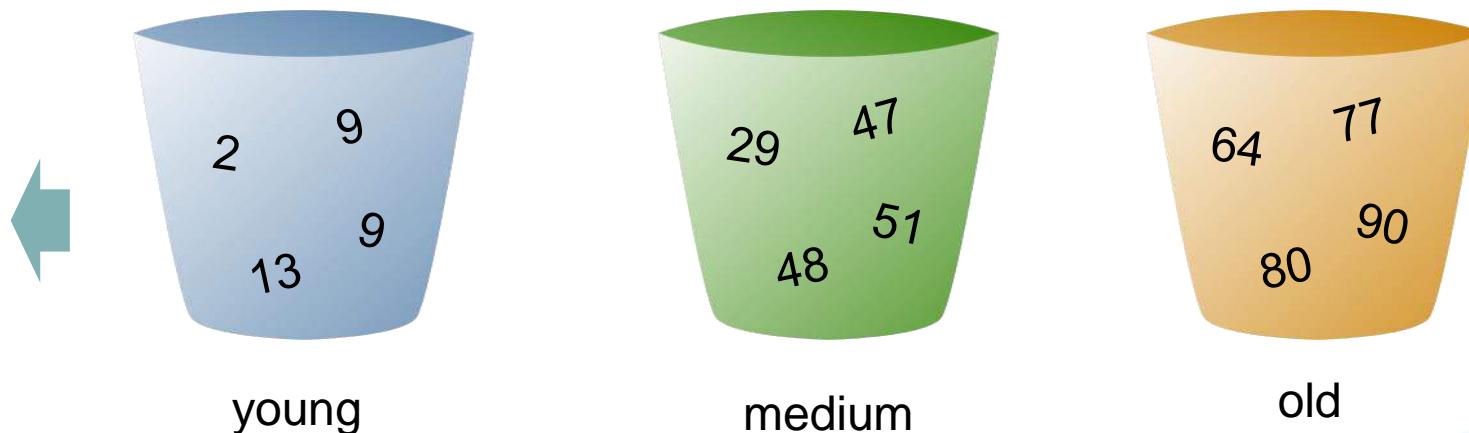


Equal Frequency Binning – Example

Tree Age [years]	Tree Height [m]
young	6
young	26
young	11
young	31
medium	86
medium	61
medium	60
medium	96
old	91
old	118
old	88
old	107

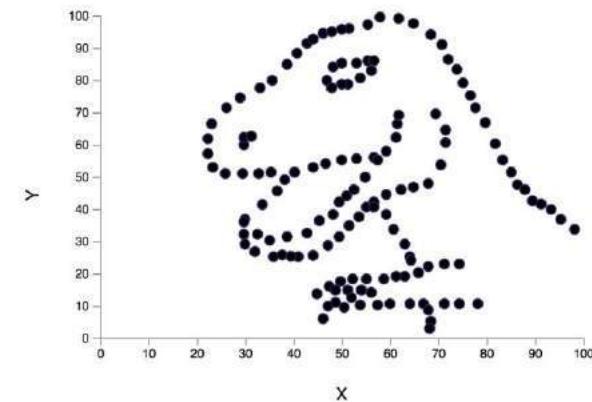
Apply equal frequency binning to the feature **Tree Age** with an element frequency of 4.

1. Sort the data
 2. Distribute elements to bins



Key Points

- Raw data has no value, we need to extract information
- Not just known unknowns, also unknown unknowns
- Visual exploration is a vital first step
(initial understanding, spotting data quality problems, building trust, etc.)
 - Humans have pretty good visual pattern recognition abilities – use them!



How to Lie with Statistics

... or, how to avoid misleading information and visualizations.

“There are lies, damn lies, and statistics”

- Anonymous

Mark Twain?

Benjamin Disraeli?

How to Lie with Statistics

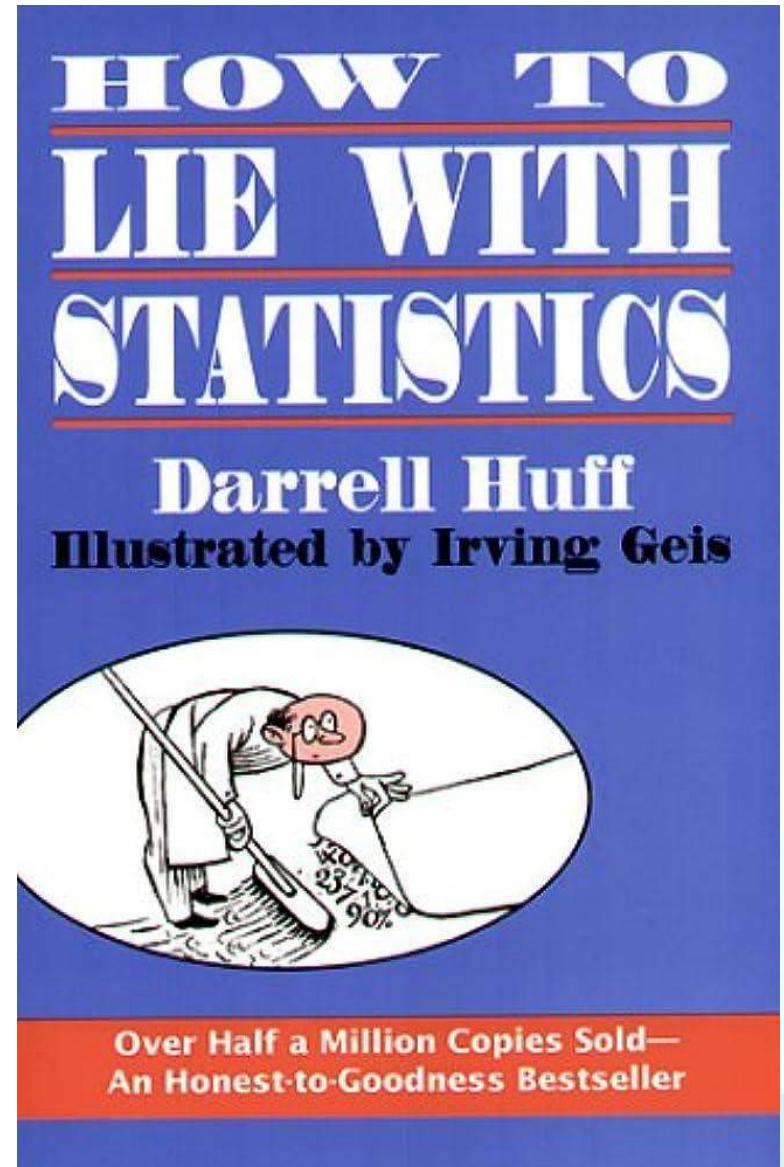
... or, how to avoid misleading information and visualizations.

- Design choice in presenting data and statistics have a huge impact!
- ...Even if what is shown is **technically true**

How to Lie with Statistics

... or, how to avoid misleading information and visualizations.

- Design choices in presenting data and statistics have a huge impact!
- ...Even if what is shown is **technically true**
- For an extreme example, search the case of **Sally Clark** (Discretion advised)



How to Lie with Statistics

... or, how to avoid misleading information and visualizations.

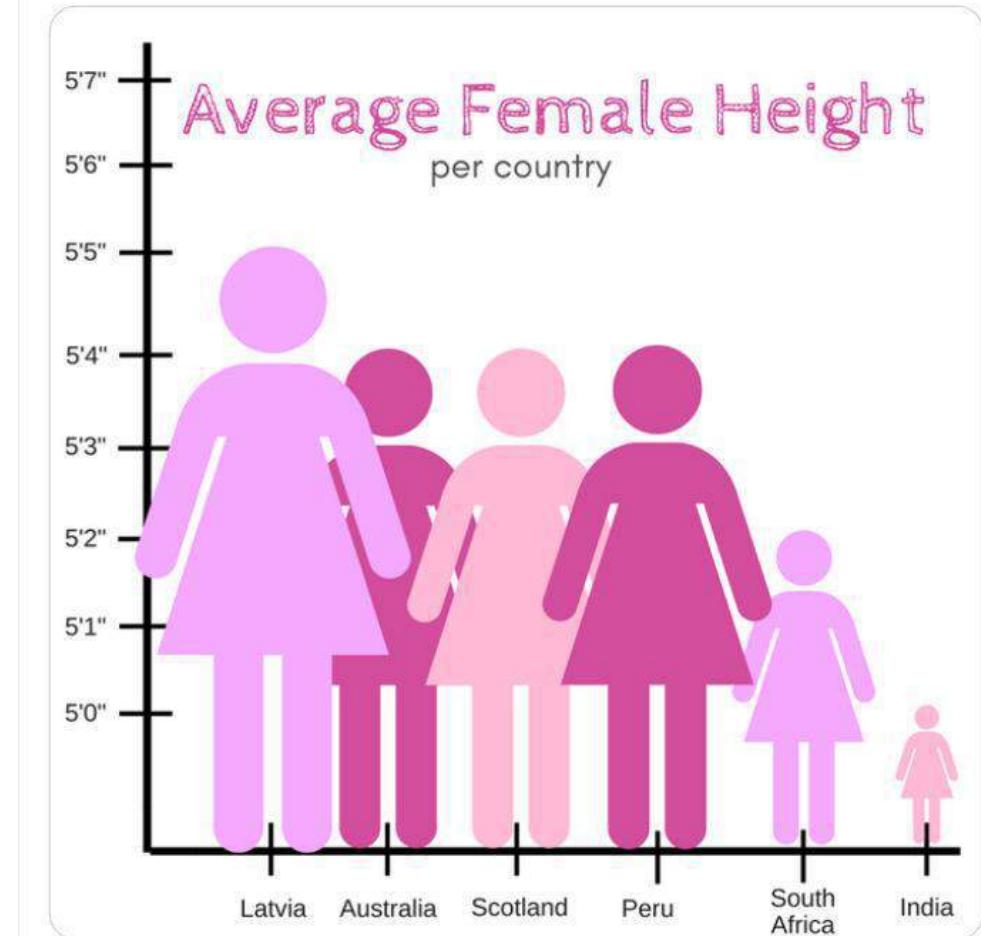
In some cases, rather than lying, the design is just **hilariously bad**.



Sabah Ibrahim
@reina_sabah



As an Indian woman, I can confirm that too much of my time is spent hiding behind a rock praying the terrifying gang of international giant ladies and their Latvian general don't find me



10:58 PM · Aug 6, 2020

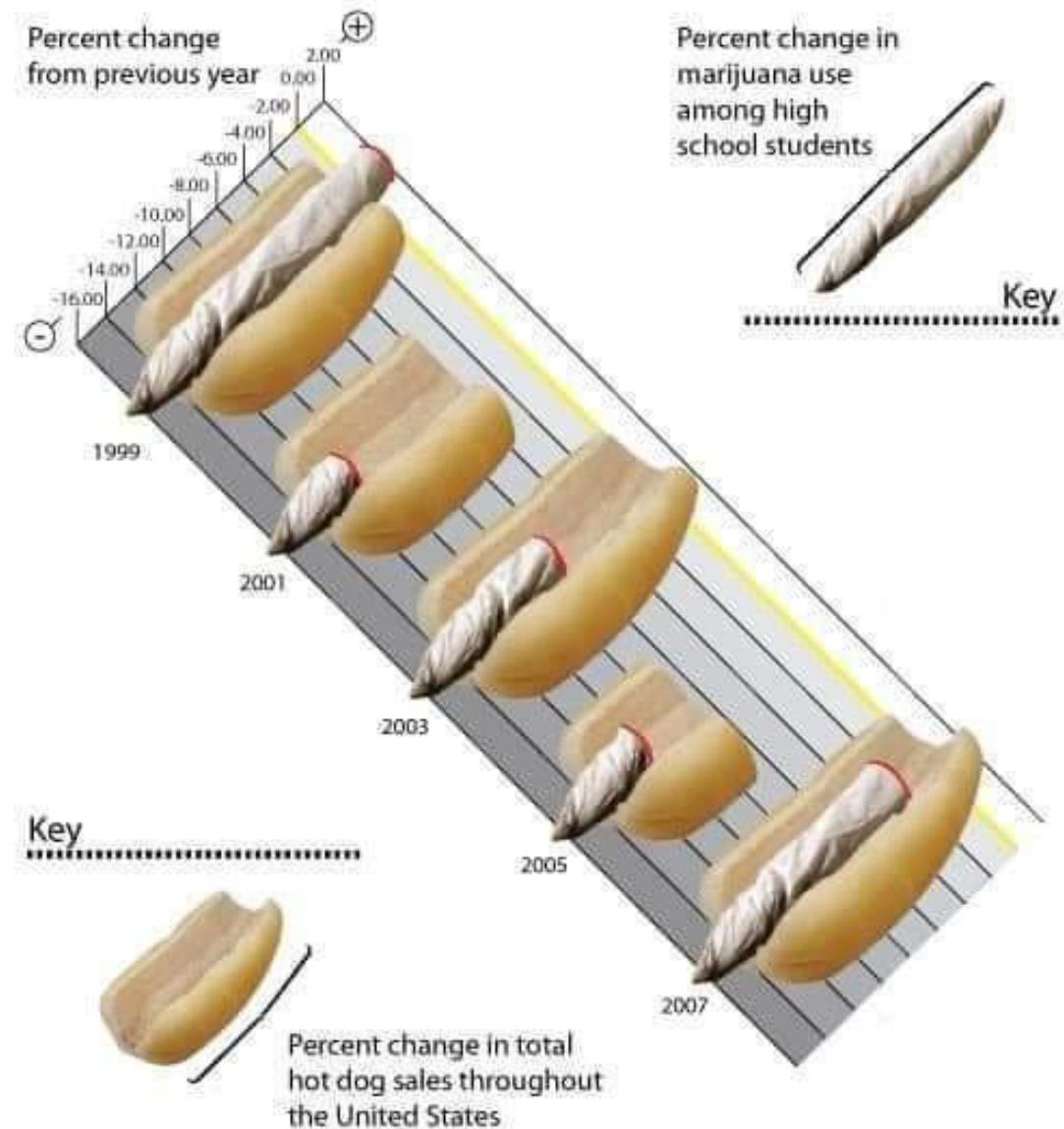


104.6K

How to Lie with Statistics

... or, how to avoid misleading information and visualizations.

In some cases, rather than lying, the design is just **hilariously bad**.



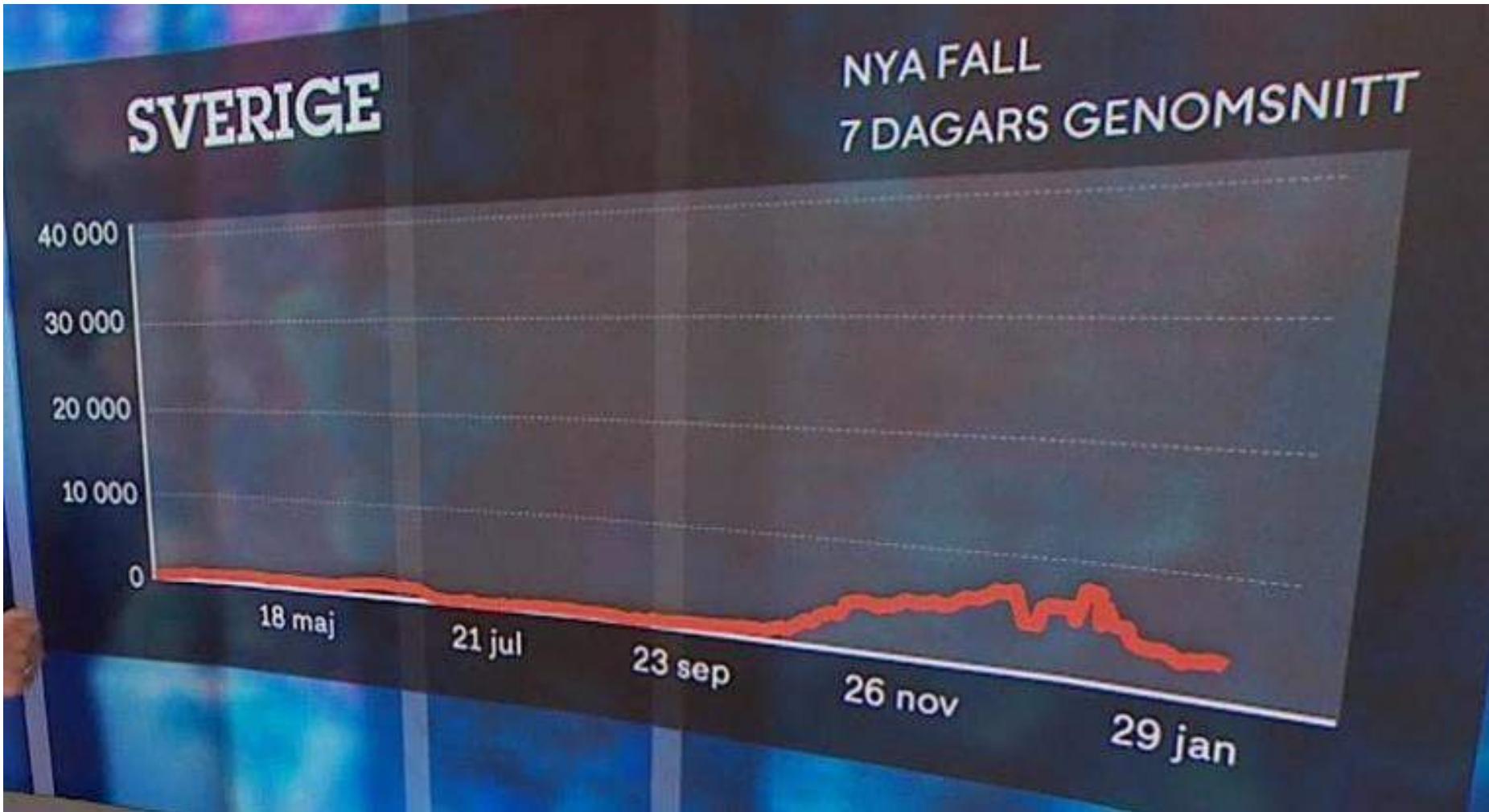
How to Lie with Statistics

... or, how to avoid misleading information and visualizations.



How to Lie with Statistics

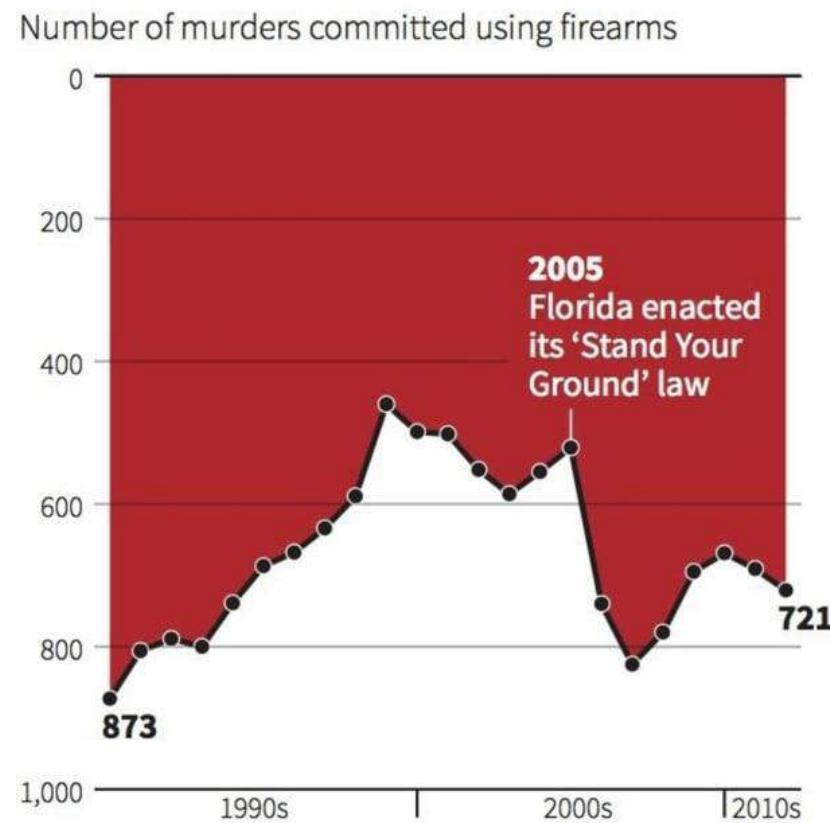
... or, how to avoid misleading information and visualizations.



How to Lie with Statistics

... or, how to avoid misleading information and visualizations.

Gun deaths in Florida



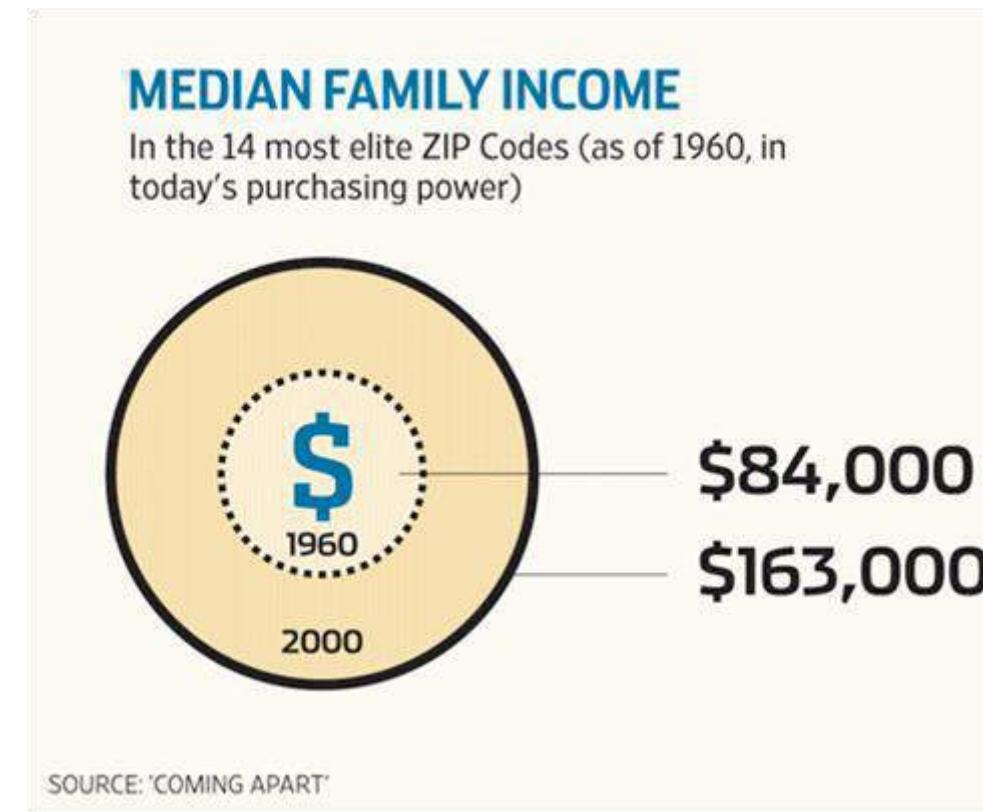
Source: Florida Department of Law Enforcement

C. Chan 16/02/2014

REUTERS

How to Lie with Statistics

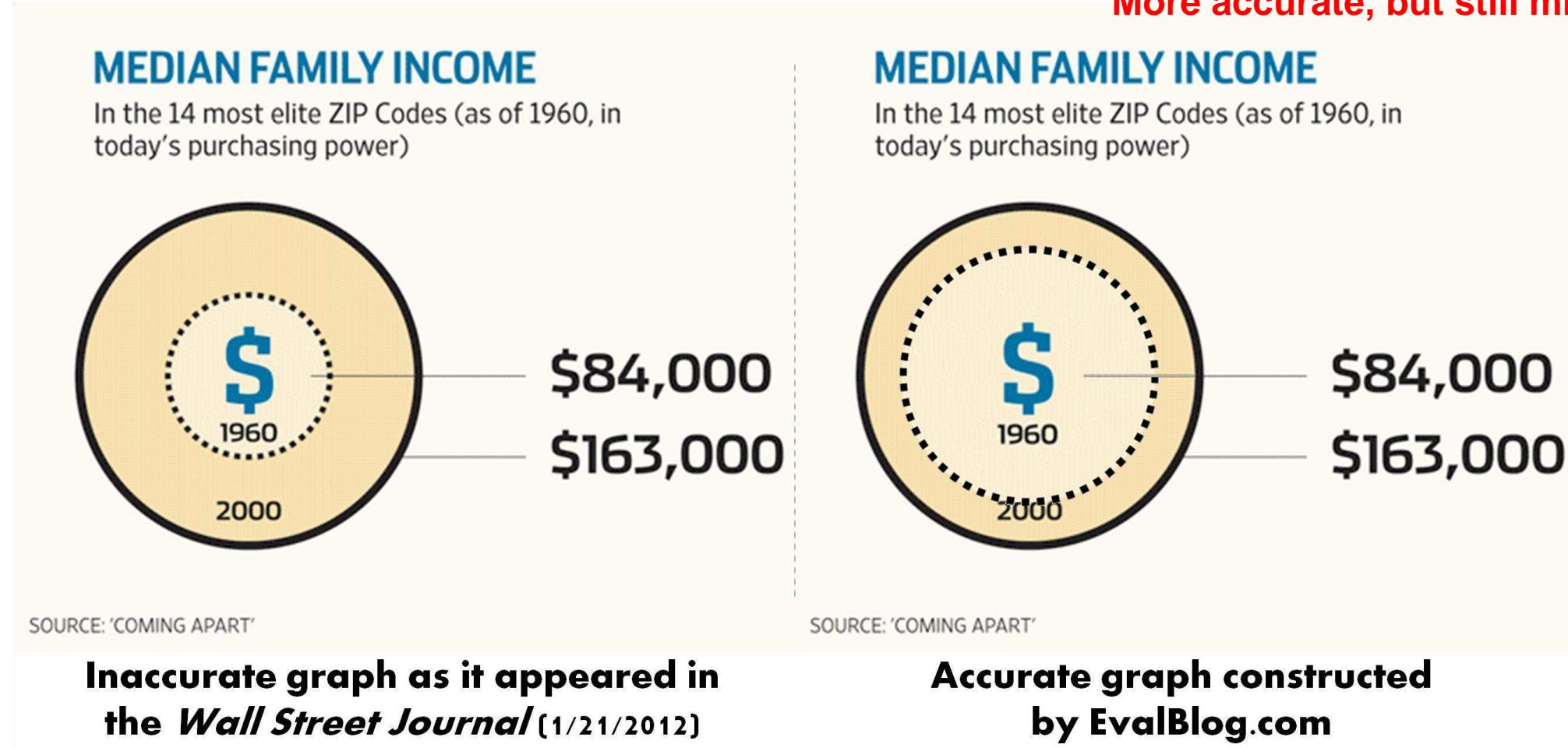
... or, how to avoid misleading information and visualizations.



The Wall Street Journal, 2012

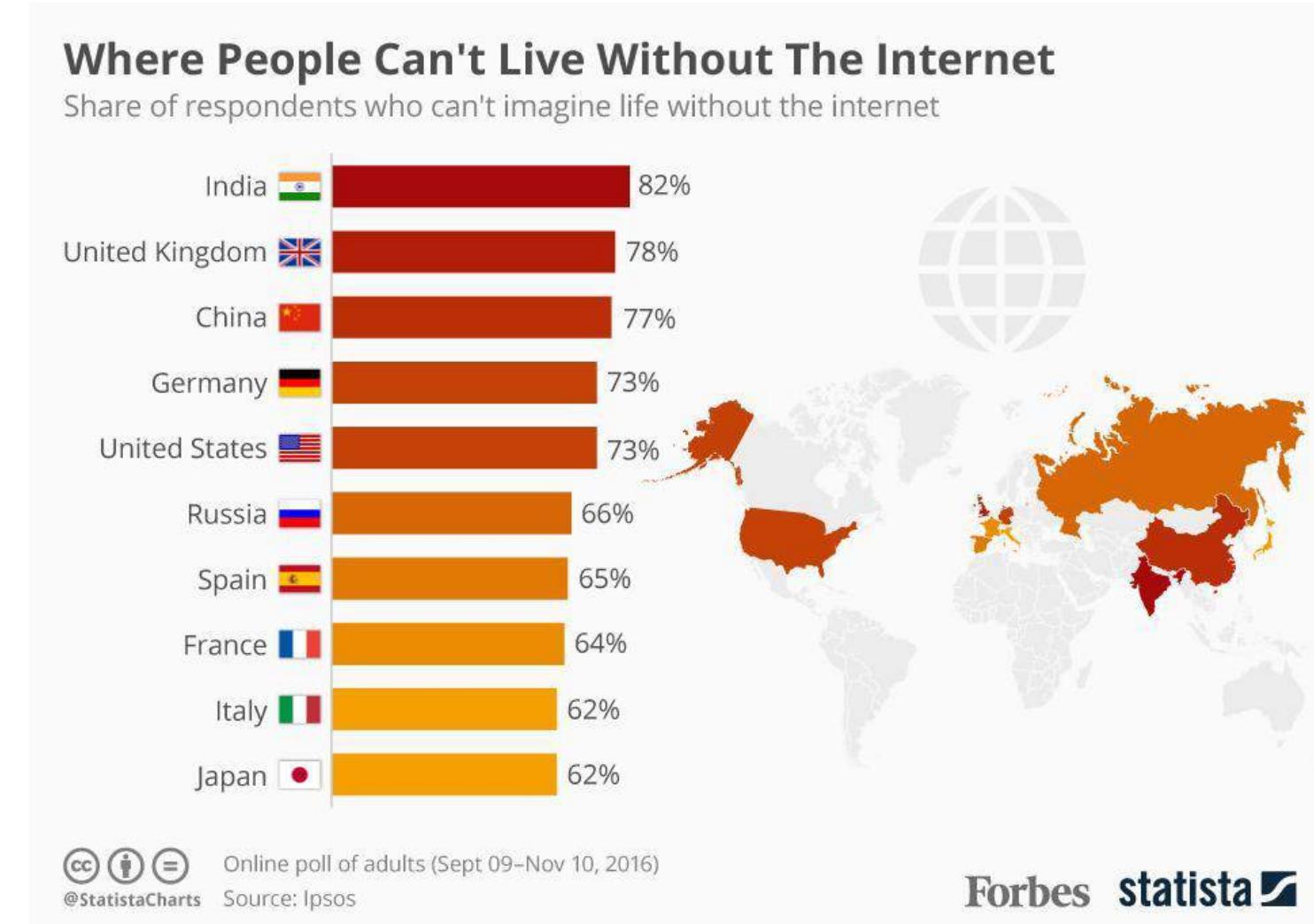
How to Lie with Statistics

... or, how to avoid misleading information and visualizations.



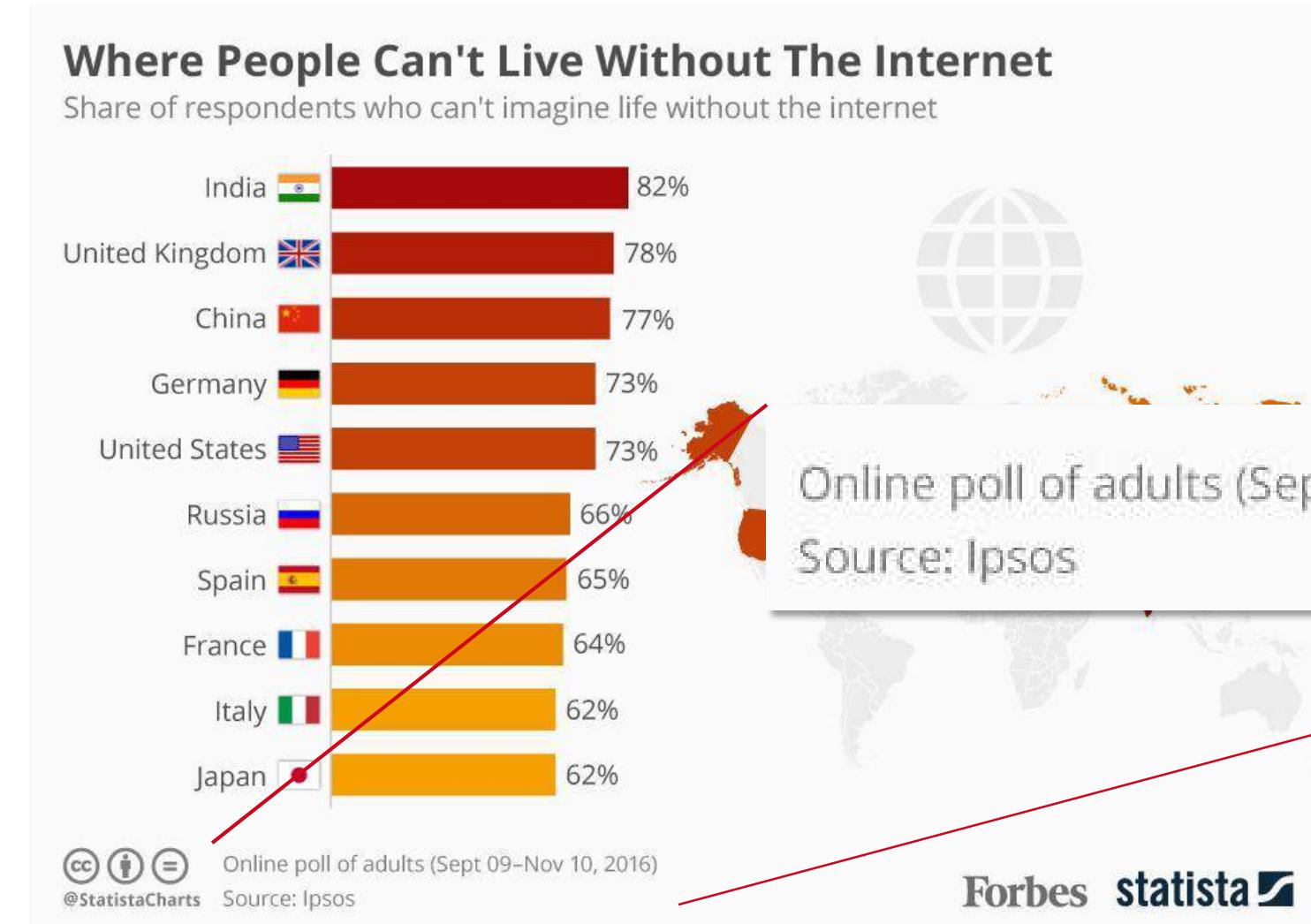
How to Lie with Statistics

... or, how to avoid misleading information and visualizations.



How to Lie with Statistics

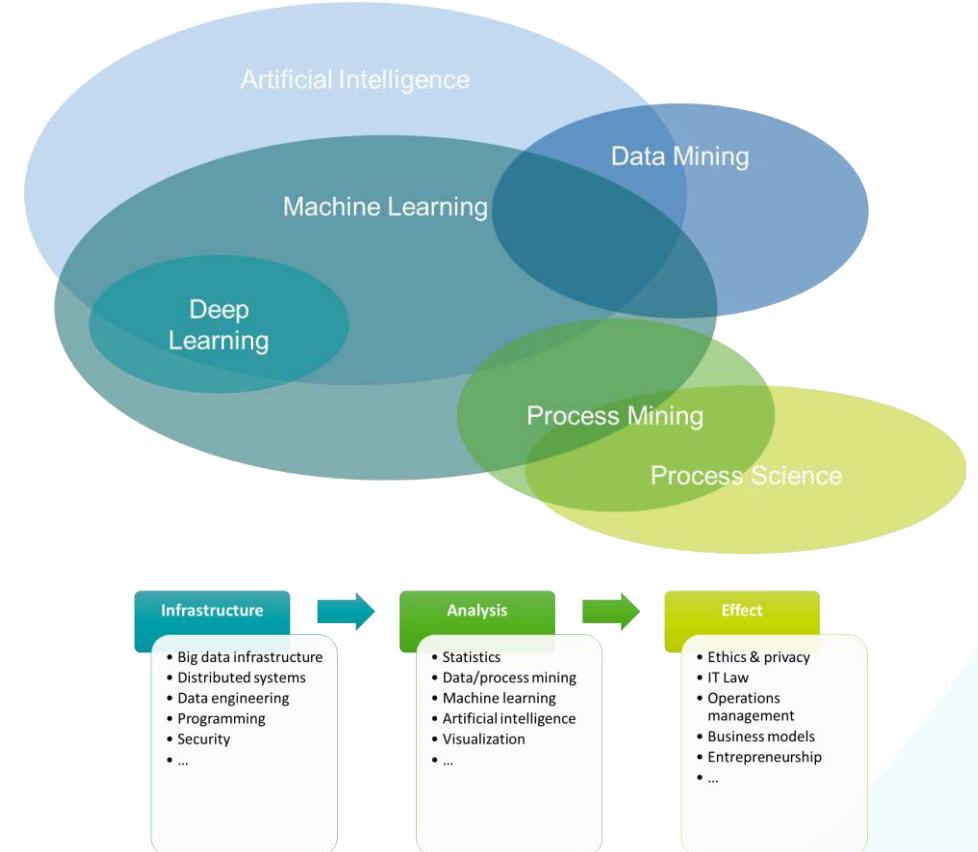
... or, how to avoid misleading information and visualizations.



Wrap up

- Data is vital, but hard to manage!
- Obtaining insights is a looping process, not a one-off application of algorithms
- Various criticalities, such as noise and bias
- Visualization is always fundamental...
- ...and comes with its own challenges!

Next up: Decision Trees



Elements of Machine Learning & Data Science

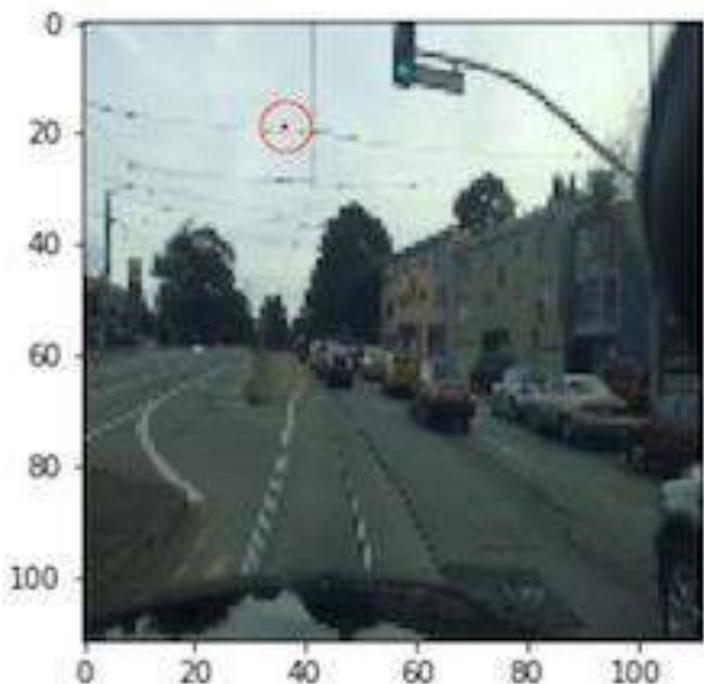
Decision Trees

Lecture 7

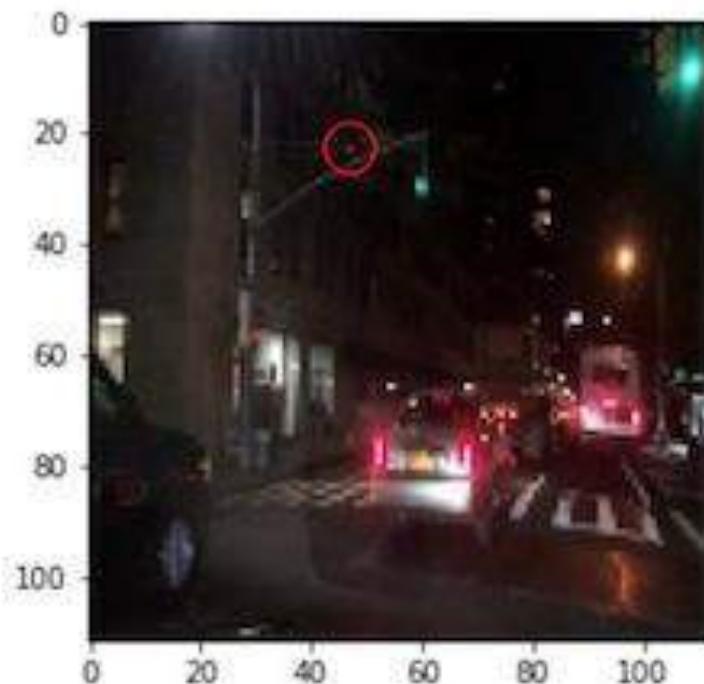
Prof. Wil van der Aalst

Marco Pegoraro, M.Sc.
Harry Beyel, M.Sc.

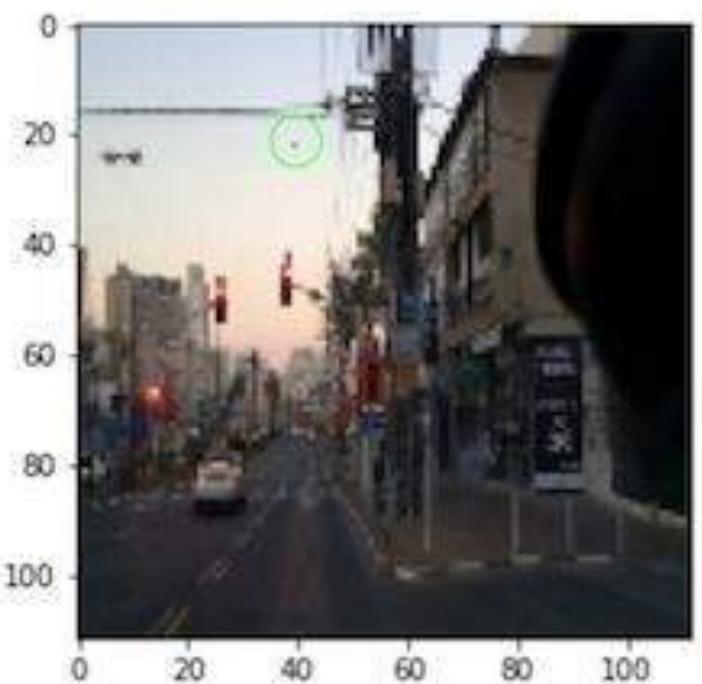
Classification problem: Red or Green?



Green light classified as red
after one pixel change



Green light classified as red
after one pixel change



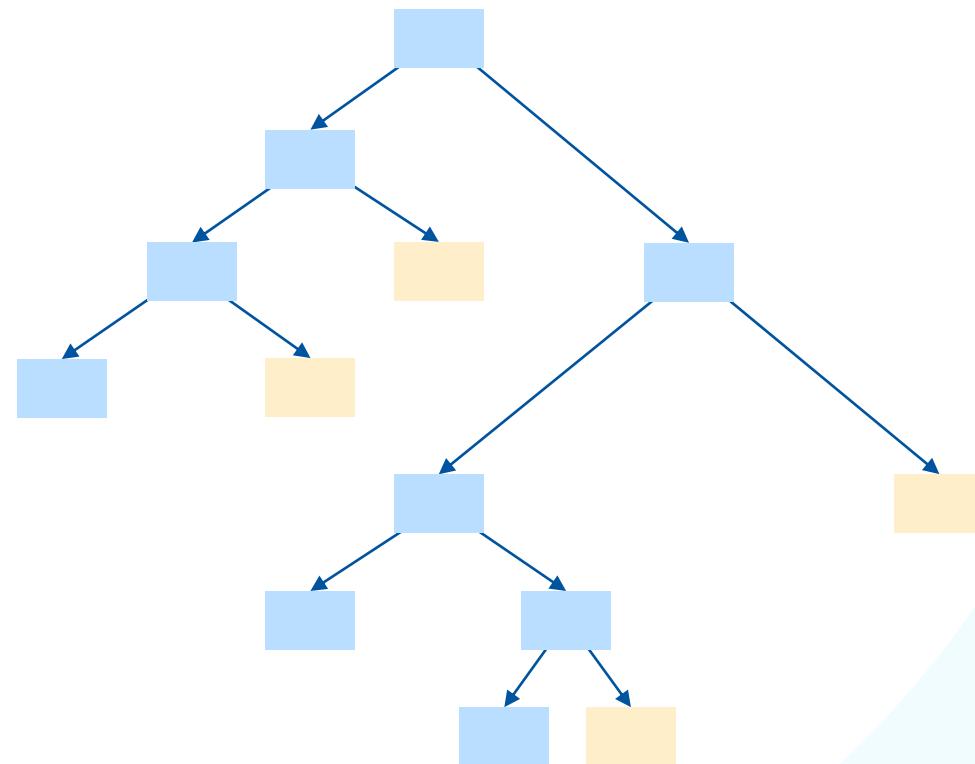
Red light classified as green
after one pixel change.

Winner Nexar traffic light challenge: On average, it takes only 3 pixels to turn red into green or green into red!

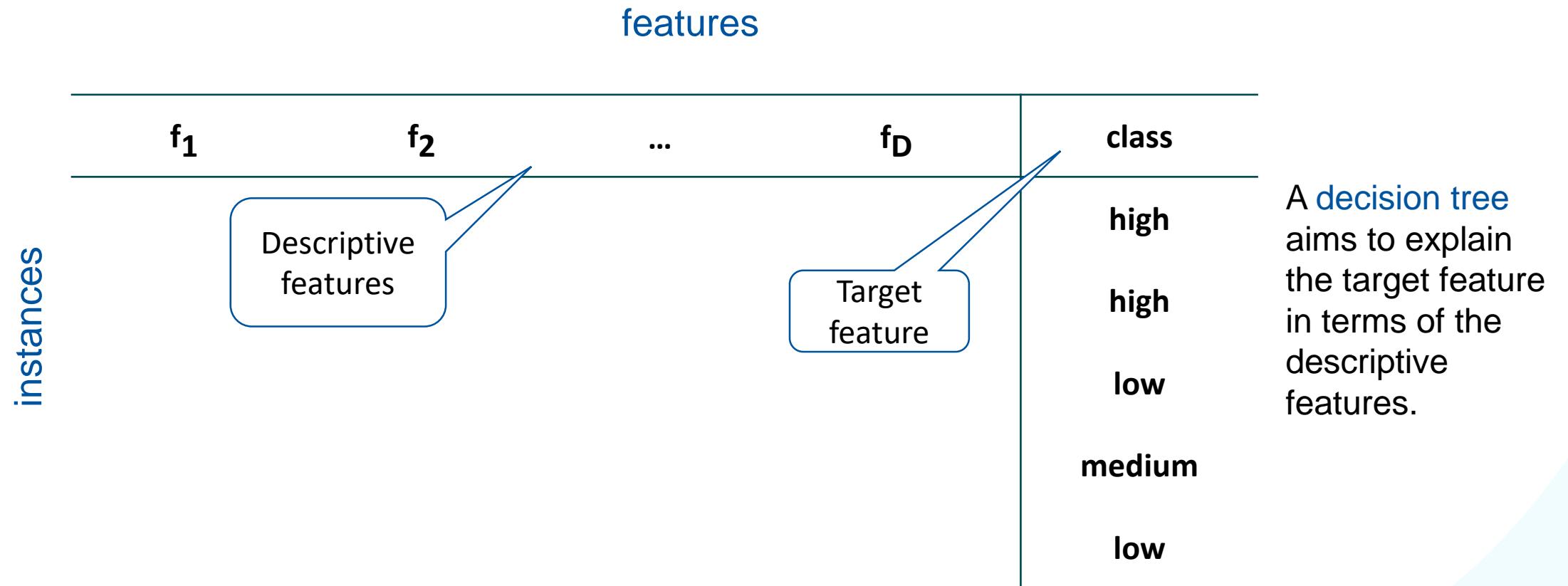
We start with simple tabular data and models that are easy to interpret!

Outline

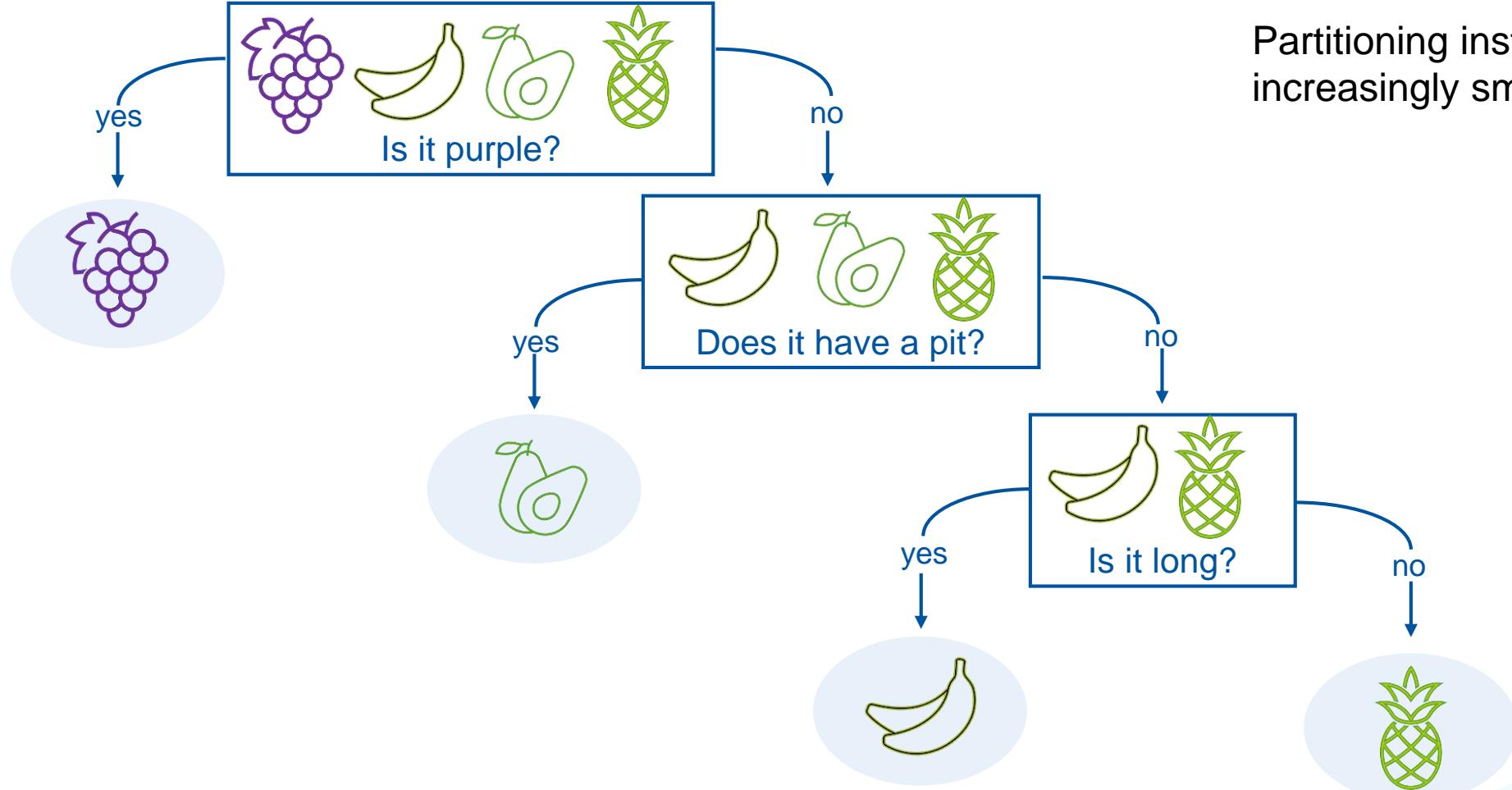
1. **Introduction to Decision Trees**
2. Entropy
3. ID3 Algorithm
4. Quantifying Information Gain
5. Pruning
6. Ensembles
7. Continuous Data



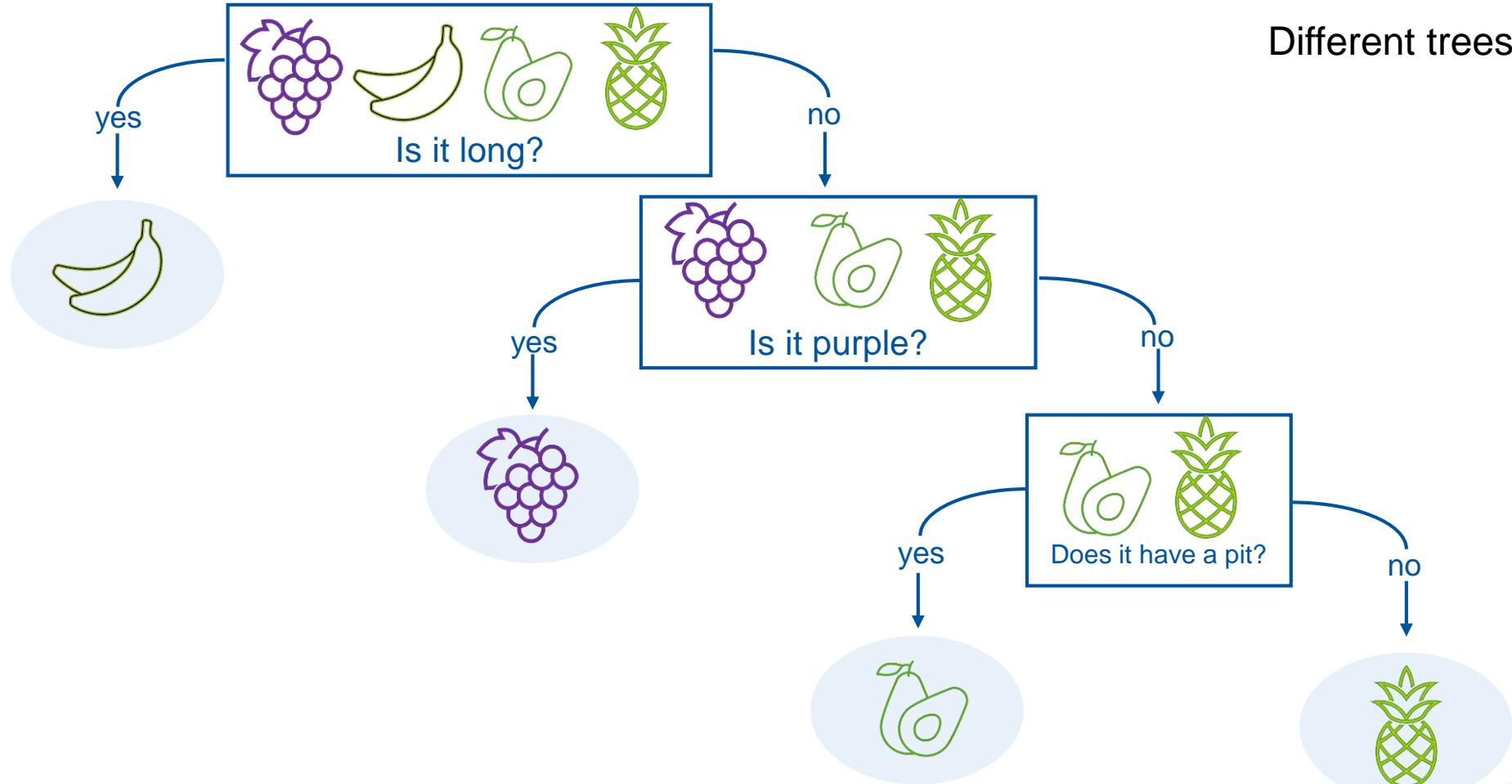
Intuition and Interpretation



Fruity Example



Fruity Example



Different trees are possible

Example 2

Rain	Wind	Temperature (°C)	Play tennis
Yes	Yes	15	No
No	No	34	Yes
Yes	No	23	Yes
Yes	Yes	20	Yes
No	Yes	28	No
...

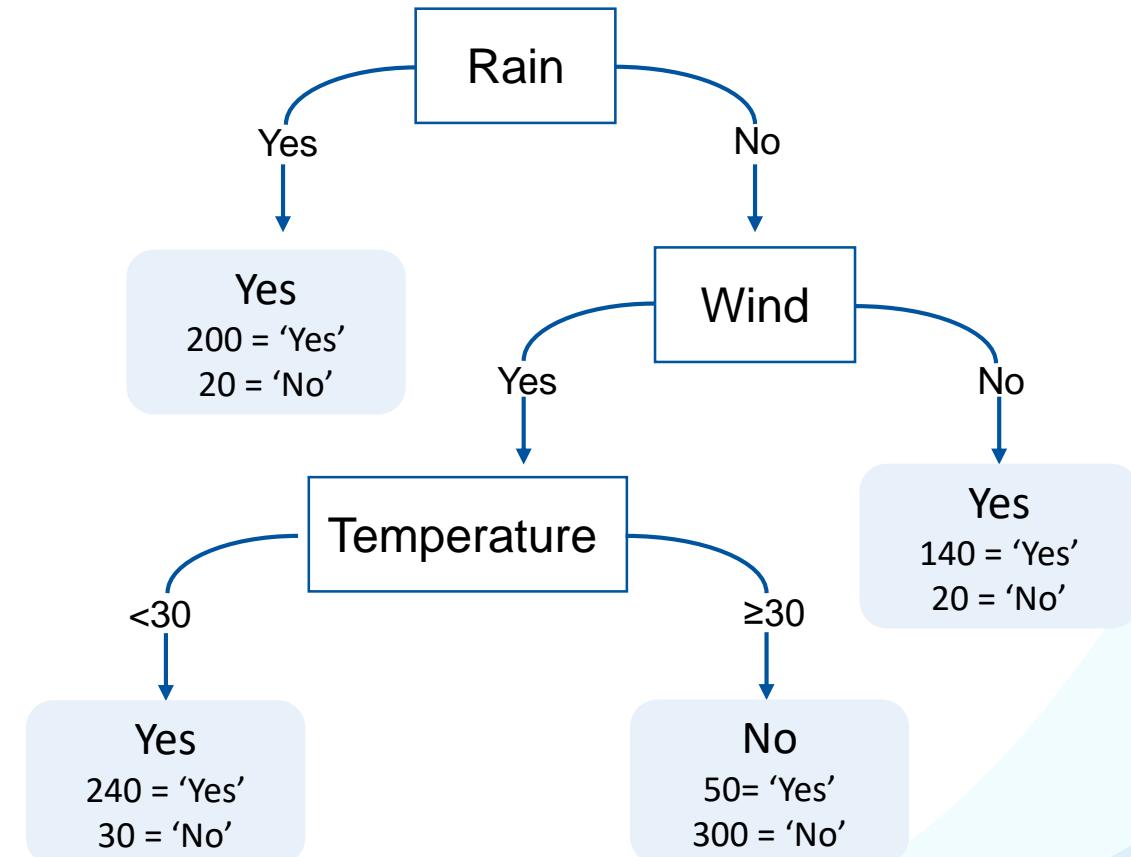
Descriptive features

Target feature

Example 2

Rain	Wind	Temperature (°C)	Play tennis
Yes	Yes	15	No
No	No	34	Yes
Yes	No	23	Yes
Yes	Yes	20	Yes
No	Yes	28	No
...

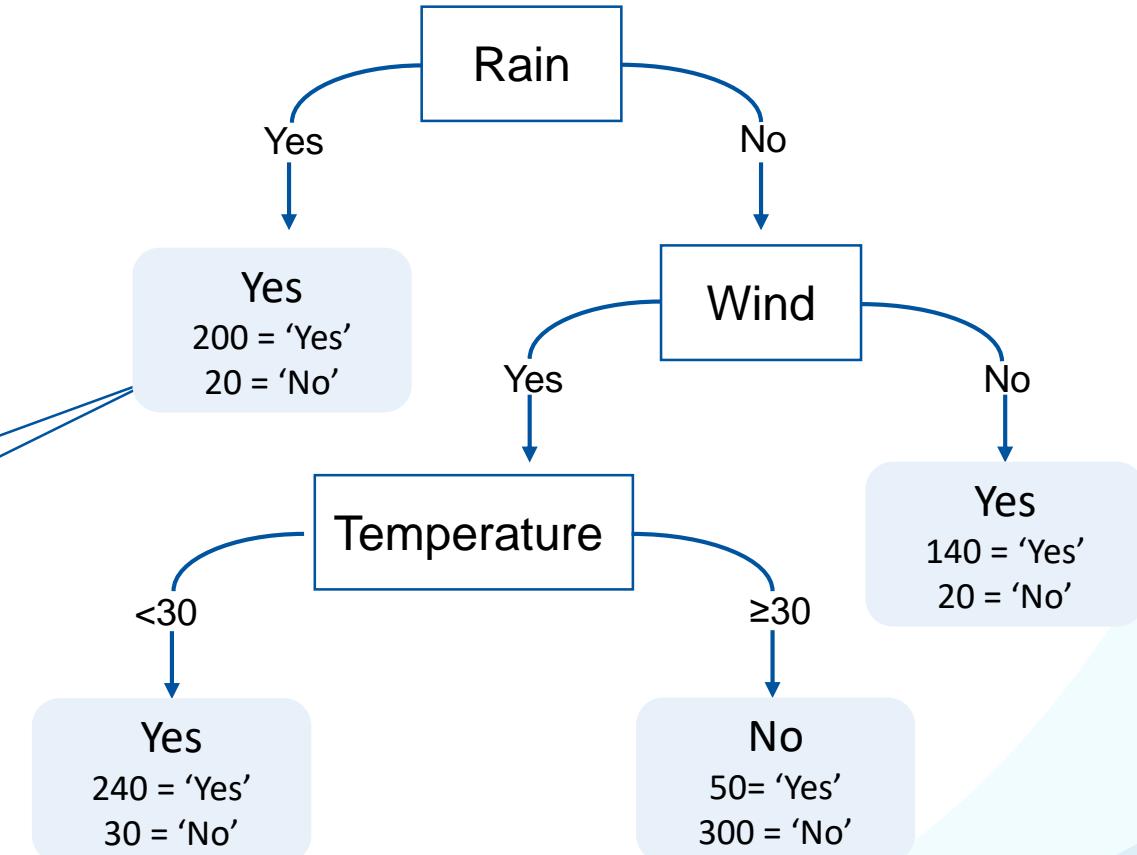
1000 Instances



Example 2

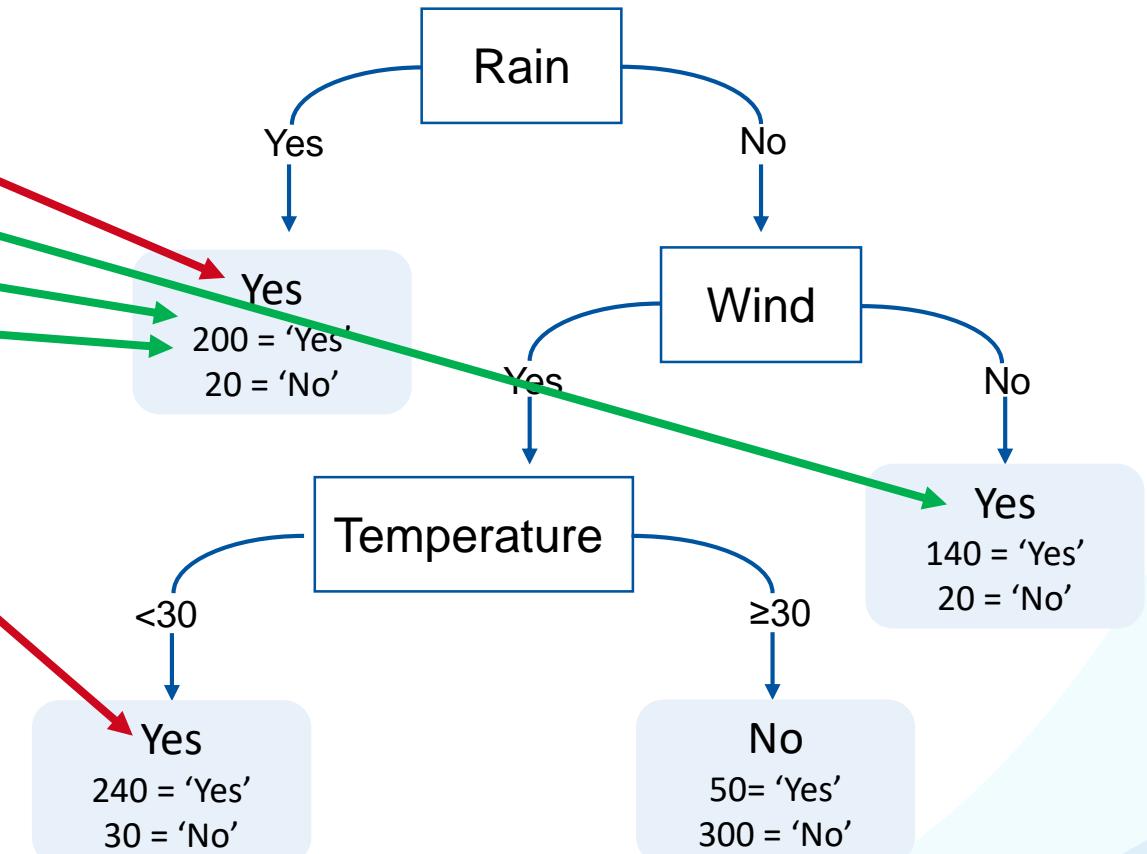
Rain	Wind	Temperature (°C)	Play tennis
Yes	Yes	15	No
No	No	34	Yes
Yes	No	23	Yes
Yes	Yes	20	Yes
No	Yes	28	No
...

220 cases with *Rain = Yes* are classified as 'Yes' (Play tennis), but 20 are classified incorrectly



Example 2

Rain	Wind	Temperature (°C)	Play tennis
Yes	Yes	15	No
No	No	34	Yes
Yes	No	23	Yes
Yes	Yes	20	Yes
No	Yes	28	No
...



Decision Tree Construction

Tree Structure

- Three types of nodes: **root node**, **interior nodes** and **leaf nodes**
- Root node refers to all instances
- Interior nodes **partition** the set of instances **based on a descriptive feature**
- Leaf nodes have a label (target feature value)
(usually based on the label of the majority of instances in this node)

Decision Tree Construction

Tree Structure

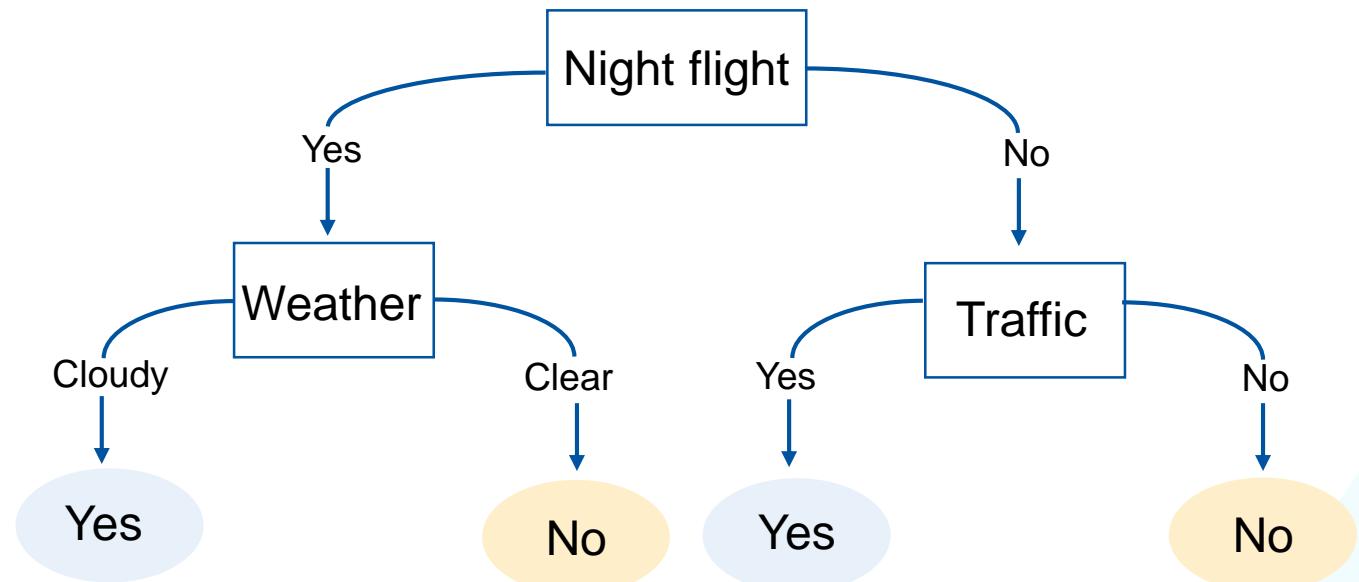
- Three types of nodes: root node, interior nodes and leaf nodes
- Root node refers to all instances
- Interior nodes partition the set of instances based on a descriptive feature
- Leaf nodes have a label (target feature value)
(usually based on the label of the majority of instances in this node)

There are two goals (often conflicting)

- The tree is small and simple
- The leaves are homogeneous in terms of the target feature

Comparing Decision Trees (1/2)

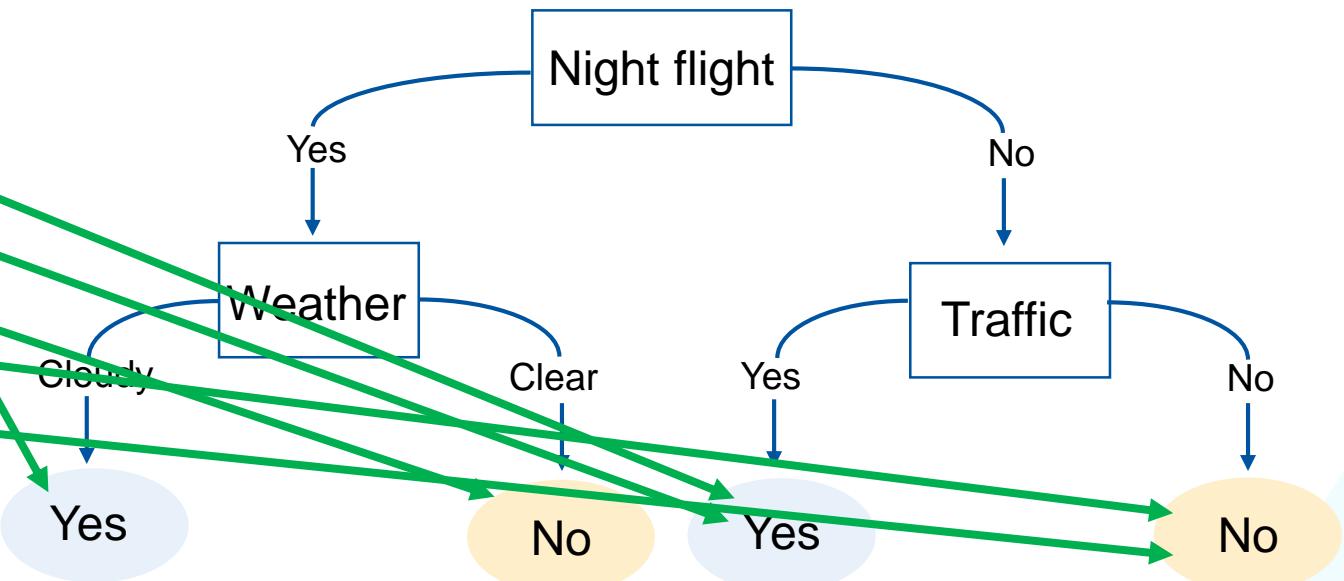
Weather	Traffic	Night flight	Flight delayed
Cloudy	No	Yes	Yes
Cloudy	Yes	No	Yes
Cloudy	Yes	No	Yes
Clear	Yes	Yes	No
Clear	No	No	No
Clear	No	No	No



Comparing Decision Trees (1/2)

Weather	Traffic	Night flight	Flight delayed
Cloudy	No	Yes	Yes
Cloudy	Yes	No	Yes
Cloudy	Yes	No	Yes
Clear	Yes	Yes	No
Clear	No	No	No
Clear	No	No	No

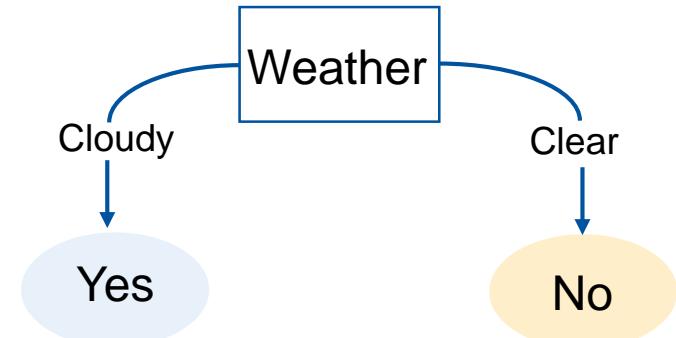
All instances correctly classified



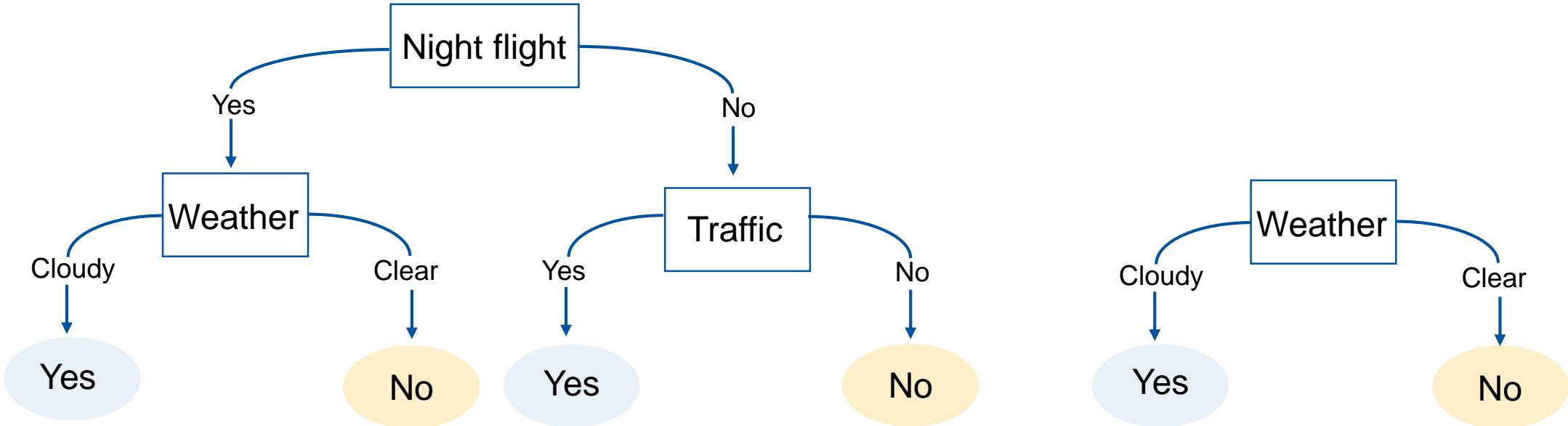
Comparing Decision Trees (2/2)

Weather	Traffic	Night flight	Flight delayed
Cloudy	No	Yes	Yes
Cloudy	Yes	No	Yes
Cloudy	Yes	No	Yes
Clear	Yes	Yes	No
Clear	No	No	No
Clear	No	No	No

All instances correctly classified



Comparing Decision Trees



Both trees correctly classify all observed instances, but the ‘simpler’ one seems ‘better’.

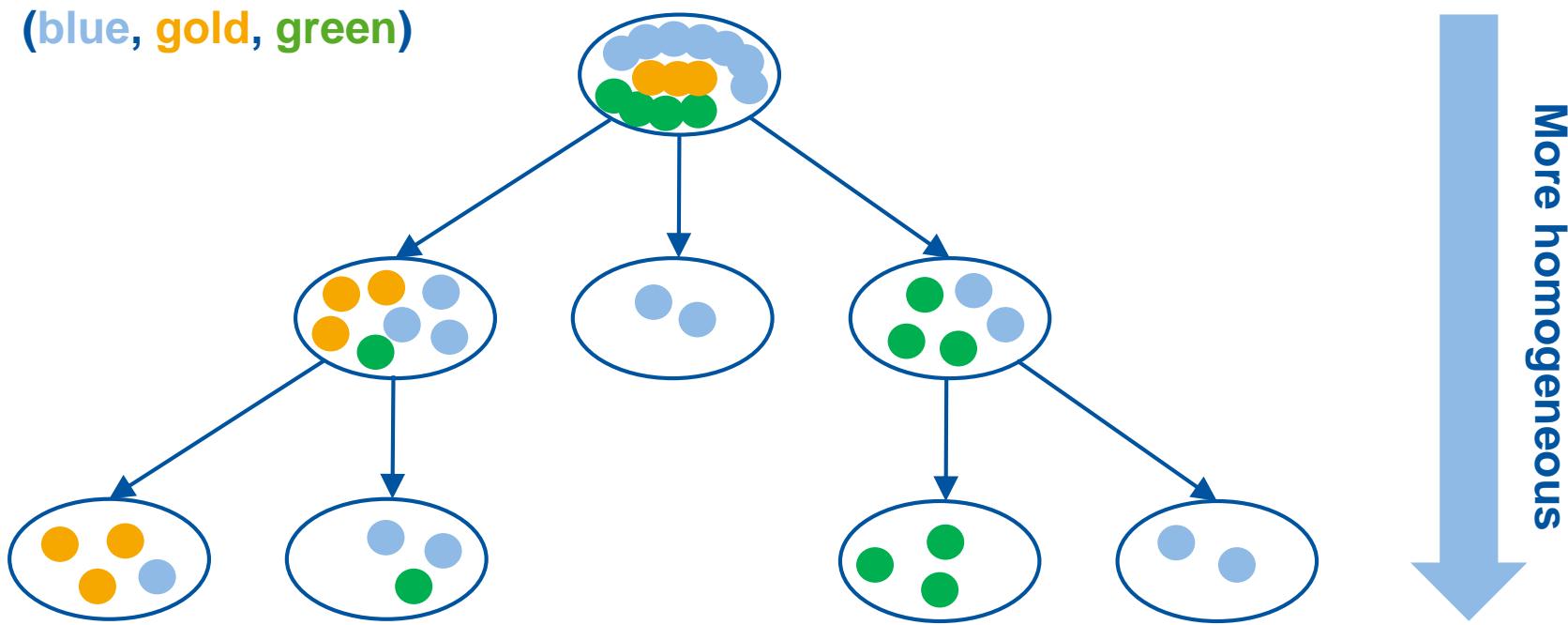
Key concepts:

- avoid overfitting
- apply Occam's razor
- prefer shallow trees

Characteristics Decision Trees

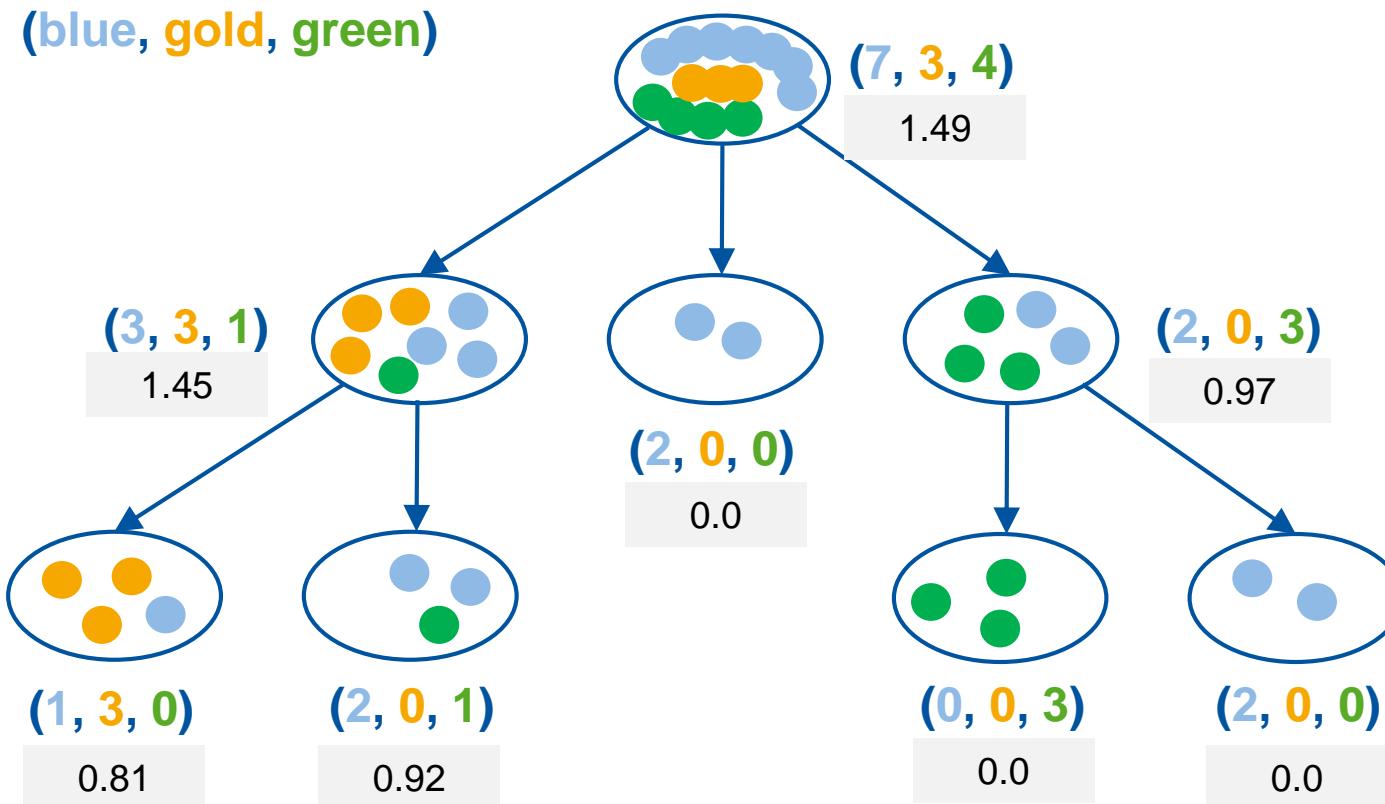
- A very simple model!
- In some cases, preferable to more complex and modern models (such as neural networks):
 - Fewer data points/attributes (managing **overfitting** is easier)
 - In domains where **explainability and transparency** are required
 - The choices of a tree are very easy to explain and show!
- There are **extensions** of **decision trees** that aim to combine simplicity and transparency with the ability to handle more complex data

Information Gain



Information gain = improvement in knowledge
(predictability of target label in nodes)

Entropy - Intuition



Idea

- Measure of impurity
- Uncertainty when guessing
- Incompressibility

Worst case entropy for 3 values: $\log_2 3 \approx 1.58$

Entropy - Formula

$$H(t) = - \sum_{k=1}^K (P(t = k) \cdot \log_s(P(t = k)))$$



$$H(\text{color}) = -\left(\frac{7}{14} \cdot \log_2\left(\frac{7}{14}\right) + \frac{3}{14} \cdot \log_2\left(\frac{3}{14}\right) + \frac{4}{14} \cdot \log_2\left(\frac{4}{14}\right)\right) \approx 1.49$$



t : examined target feature (color in the example)



K : number of possible values of the target feature ($K = |\{\text{blue, gold, green}\}| = 3$ in the example)

$P(t = k) \in [0, 1]$: probability that a random value in t equals the k th value in the set of possible values

s : logarithm base (we use $s = 2$ by convention)

Entropy - Formula

$$H(t) = - \sum_{k=1}^K (P(t = k) \cdot \log_s(P(t = k)))$$



$$H(\text{color}) = -\left(\frac{2}{5} \cdot \log_2\left(\frac{2}{5}\right) + \frac{0}{5} \cdot \log_2\left(\frac{0}{5}\right) + \frac{3}{5} \cdot \log_2\left(\frac{3}{5}\right)\right) \approx 0.97$$

Entropy - Formula

$$H(t) = - \sum_{k=1}^K (P(t = k) \cdot \log_s(P(t = k)))$$



$$H(\text{color}) = -\left(\frac{2}{5} \cdot \log_2\left(\frac{2}{5}\right) + \frac{0}{5} \cdot \log_2\left(\frac{0}{5}\right) + \frac{3}{5} \cdot \log_2\left(\frac{3}{5}\right)\right) \approx 0.97$$



$$H(\text{color}) = -\left(\frac{0}{3} \cdot \log_2\left(\frac{0}{3}\right) + \frac{0}{3} \cdot \log_2\left(\frac{0}{3}\right) + \frac{3}{3} \cdot \log_2\left(\frac{3}{3}\right)\right) = 0$$

Questions

Suppose that we have K possible values (colors) and N instances (balls).



$$H(t) = - \sum_{k=1}^K (P(t = k) \cdot \log_s(P(t = k)))$$

What distribution of the N instances over the K possible values yields the lowest entropy?

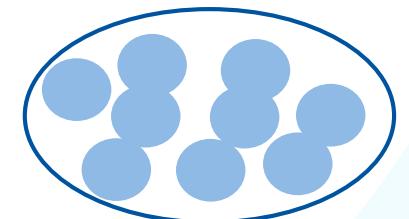
Questions

Suppose that we have K possible values (colors) and N instances (balls).



$$H(t) = - \sum_{k=1}^K (P(t = k) \cdot \log_s(P(t = k)))$$

What distribution of the N instances over the K possible values yields the lowest entropy?



$$H(\text{color}) = -(1 \cdot \log_2(1)) = 0$$

→ all instances have the same value

Questions

Suppose that we have K possible values (colors) and N instances (balls).



$$H(t) = - \sum_{k=1}^K (P(t = k) \cdot \log_s(P(t = k)))$$

What distribution of the N instances over the K possible values yields the highest entropy?

Questions

Suppose that we have K possible values (colors) and N instances (balls).



$$H(t) = - \sum_{k=1}^K (P(t=k) \cdot \log_s(P(t=k)))$$

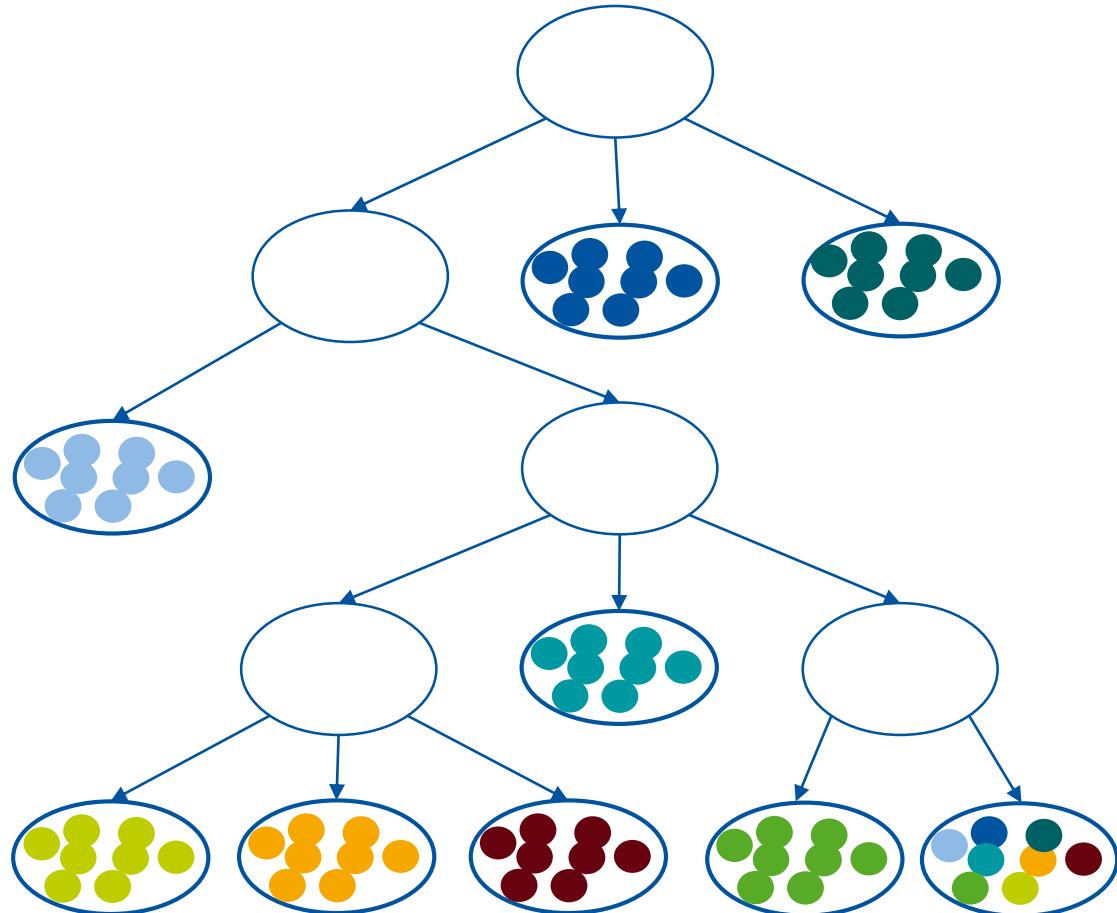
What distribution of the N instances over the K possible values yields the highest entropy?

→ Even distribution over all possible values

$$\begin{aligned} H(\text{color}) &= - \sum_{k=1}^K \left(\frac{1}{K} \cdot \log_2 \left(\frac{1}{K} \right) \right) \\ &= - \left(K \cdot \frac{1}{K} \cdot \log_2 \left(\frac{1}{K} \right) \right) \\ &= - \log_2 \left(\frac{1}{K} \right) \\ &= \log_2(K) \end{aligned}$$



Overall Entropy

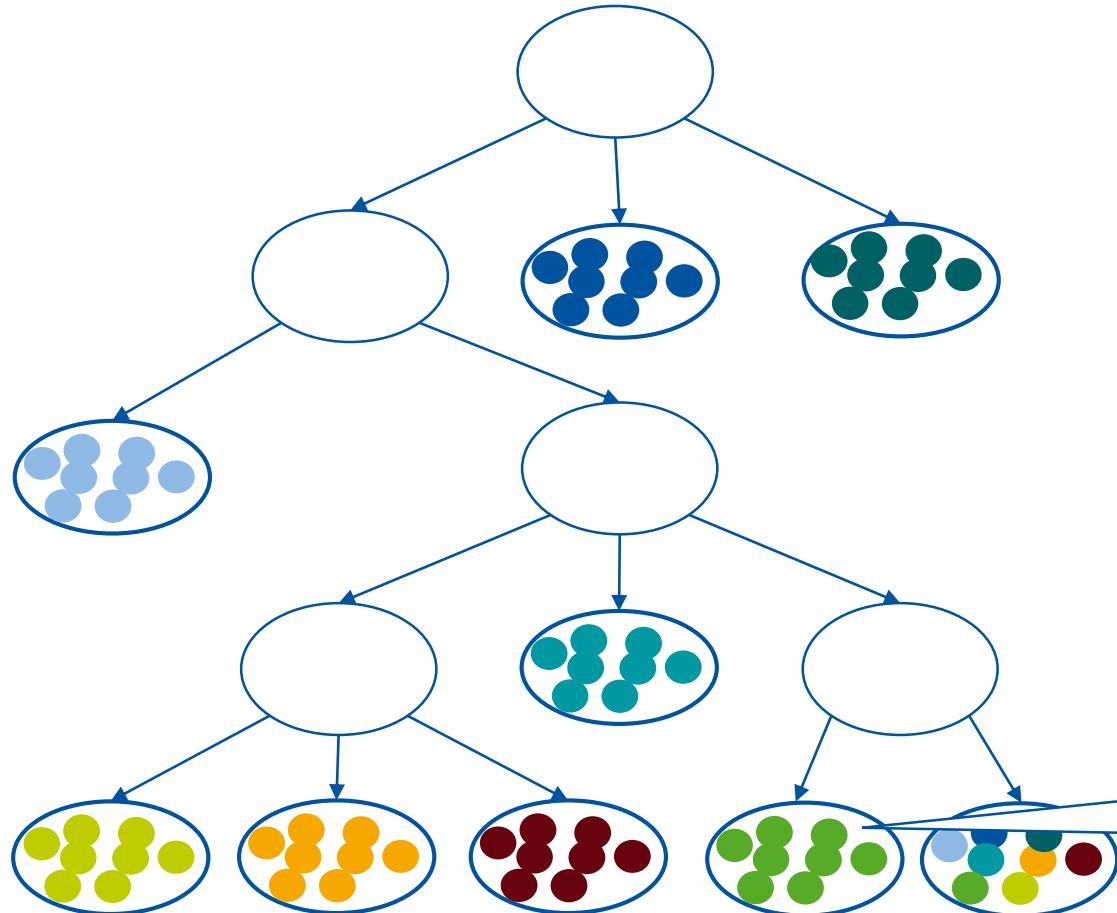


Overall entropy H_W is the weighted average of the individual entropies:

$$H_W(t) = \sum_{node \in nodes} \left(\frac{|node|}{N} \cdot H^{node}(t) \right)$$

Example: $N = 72, K = 8$

Overall Entropy



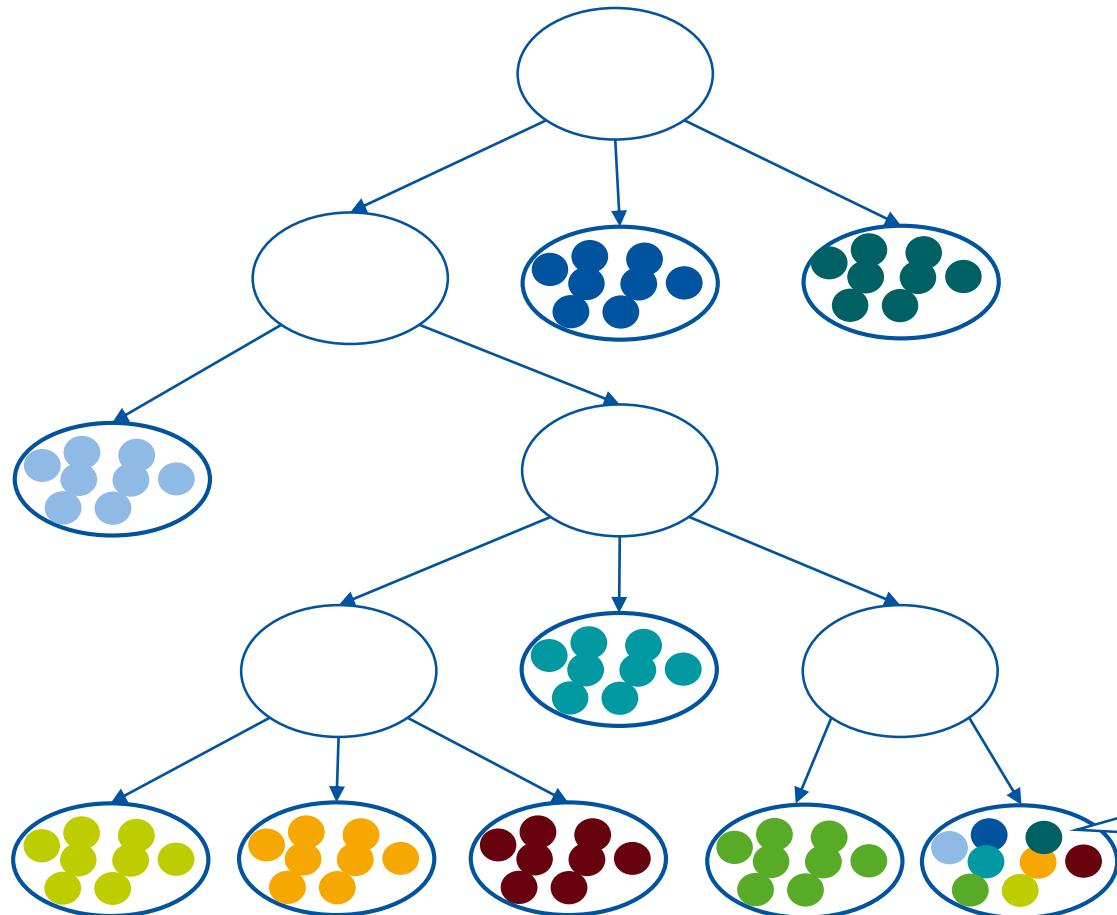
Overall entropy H_W is the weighted average of the individual entropies:

$$H_W(t) = \sum_{node \in nodes} \left(\frac{|node|}{N} \cdot H^{node}(t) \right)$$

Example: $N = 72, K = 8$

8 homogeneously colored balls:
 $H^{node}(\text{color}) = -\left(\frac{8}{8} \cdot \log_2\left(\frac{8}{8}\right)\right) = 0$

Overall Entropy



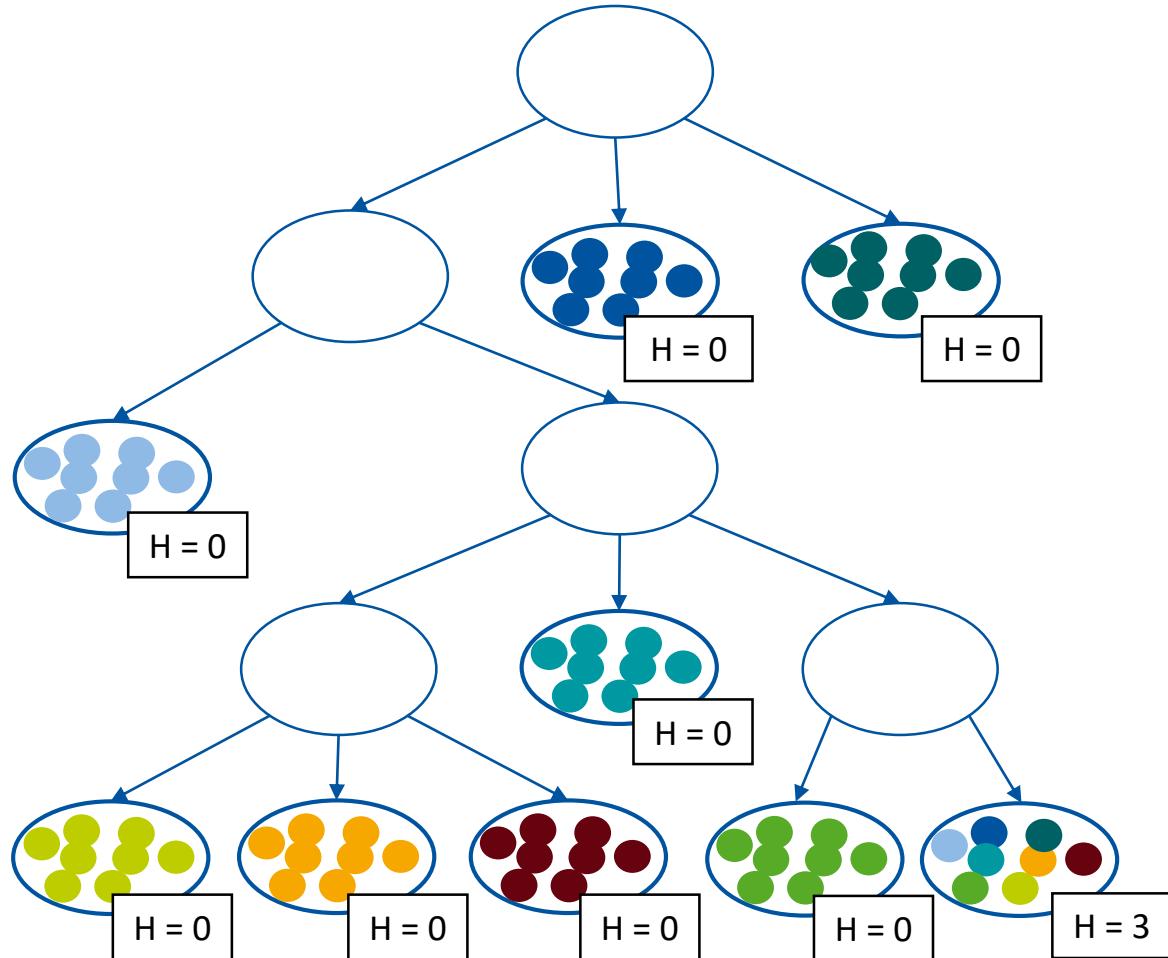
Overall entropy H_W is the weighted average of the individual entropies:

$$H_W(t) = \sum_{node \in nodes} \left(\frac{|node|}{N} \cdot H^{node}(t) \right)$$

Example: $N = 72, K = 8$

Even distribution of 8 colors over 8 balls:
$$H^{node}(color) = - \sum_{k=1}^8 \frac{1}{8} \cdot \log_2\left(\frac{1}{8}\right) = \log_2(8) = 3$$

Overall Entropy

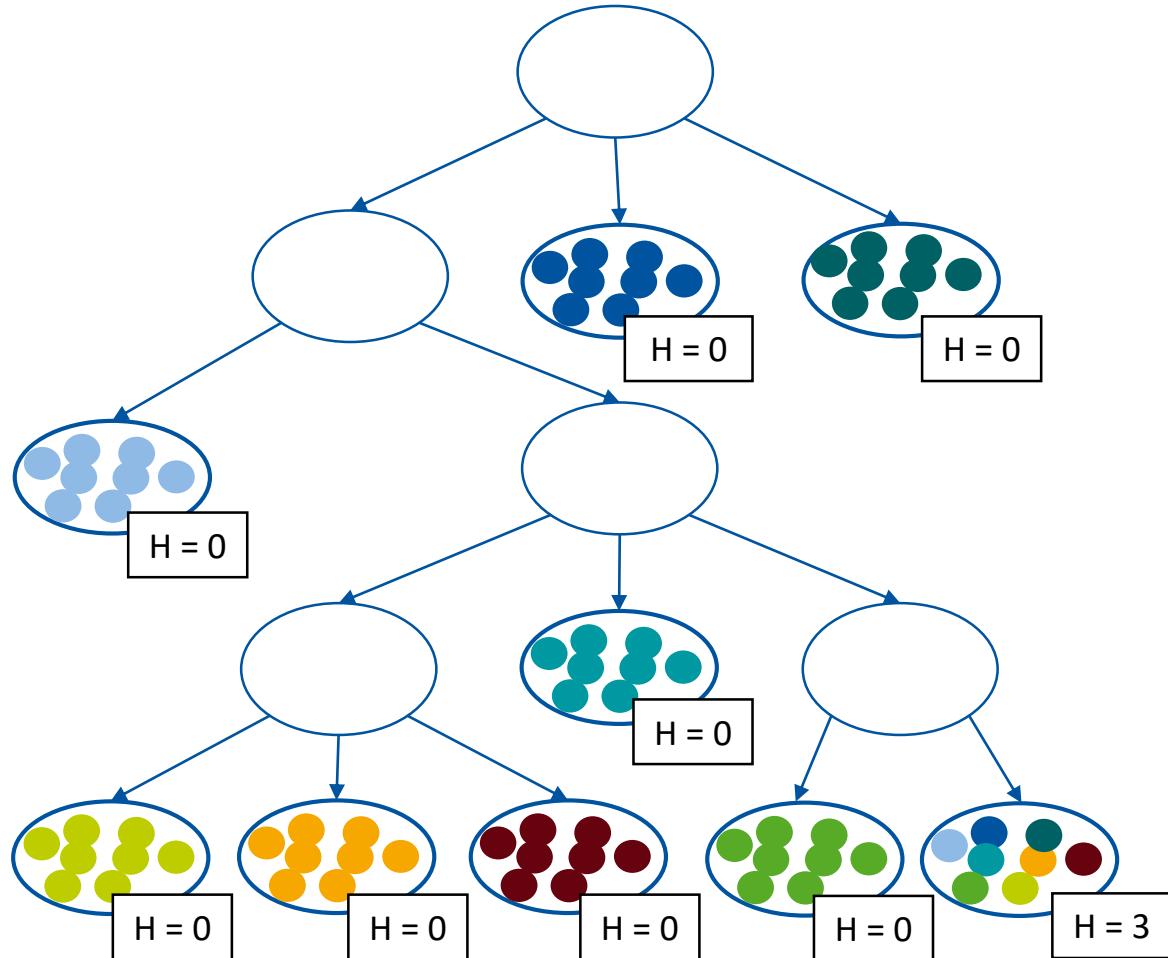


Overall entropy H_W is the weighted average of the individual entropies:

$$H_W(t) = \sum_{node \in nodes} \left(\frac{|node|}{N} \cdot H^{node}(t) \right)$$

Example: $N = 72, K = 8$

Overall Entropy



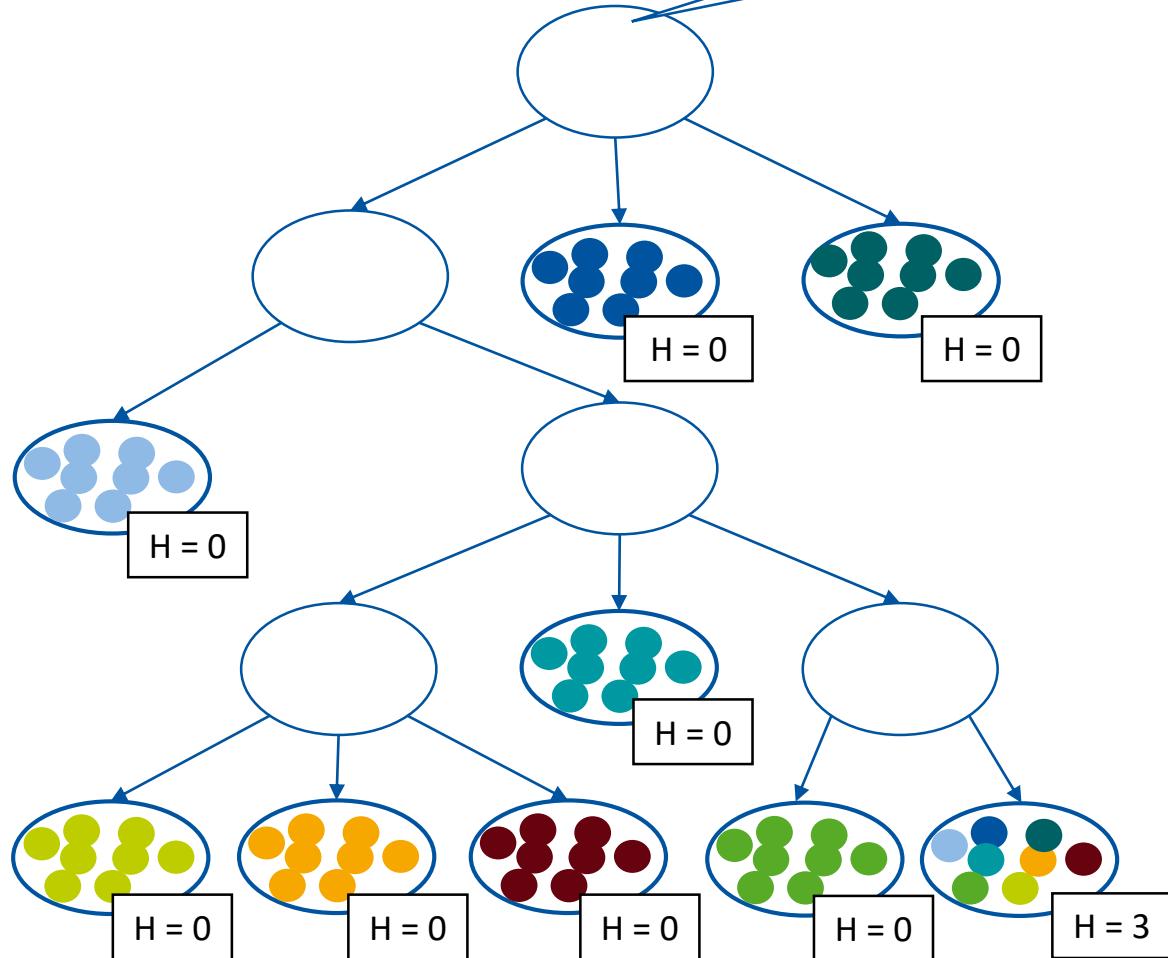
Overall entropy H_W is the weighted average of the individual entropies:

$$H_W(t) = \sum_{node \in nodes} \left(\frac{|node|}{N} \cdot H^{node}(t) \right)$$

Example: $N = 72, K = 8$

$$\begin{aligned}
 H_W(\text{color}) &= \frac{8}{72} \cdot 0 + \frac{8}{72} \cdot 0 + \frac{8}{72} \cdot 0 + \frac{8}{72} \cdot 0 \\
 &\quad + \frac{8}{72} \cdot 0 + \frac{8}{72} \cdot 0 + \frac{8}{72} \cdot 0 + \frac{8}{72} \cdot 0 \\
 &\quad + \frac{8}{72} \cdot 3 = \frac{24}{72} \approx 0.33
 \end{aligned}$$

Overall Entropy



Even distribution of 8 colors over 72 balls:

$$H_W(\text{color}) = \frac{72}{72} \cdot \left(-\sum_{k=1}^8 \left(\frac{9}{72} \cdot \log_2 \left(\frac{9}{72} \right) \right) \right) = \log_2(8) = 3$$

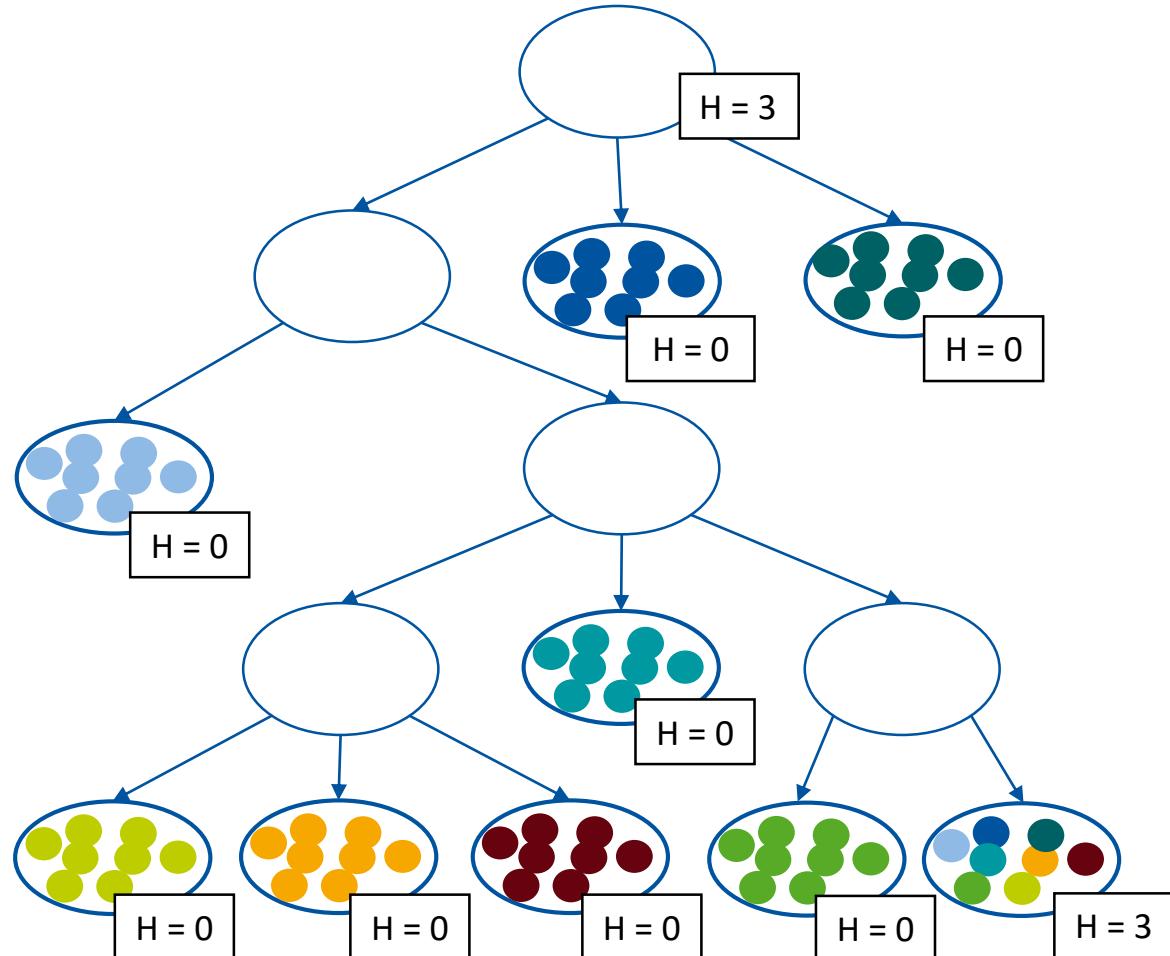
Overall entropy H_w is the weighted average of the individual entropies:

$$H_W(t) = \sum_{node \in nodes} \left(\frac{|node|}{N} \cdot H^{node}(t) \right)$$

Example: $N = 72, K = 8$

$$H_W(\text{color}) = \frac{8}{72} \cdot 0 + \frac{8}{72} \cdot 0 + \frac{8}{72} \cdot 0 + \frac{8}{72} \cdot 0 \\ + \frac{8}{72} \cdot 0 + \frac{8}{72} \cdot 0 + \frac{8}{72} \cdot 0 + \frac{8}{72} \cdot 0 \\ + \frac{8}{72} \cdot 3 = \frac{24}{72} \approx 0.33$$

Information Gain



$$H_W(\text{color}) = 3$$



information loss
 ≈ 2.66

information gain
 ≈ 2.66

$$H_W(\text{color}) \approx 0.33$$

Information Gain - Example Revisited

$$H(\text{delayed}) = 1$$

Weather	Traffic	Night flight	Flight delayed
Cloudy	No	Yes	Yes
Cloudy	Yes	No	Yes
Cloudy	Yes	No	Yes
Clear	Yes	Yes	No
Clear	No	No	No
Clear	No	No	No

Information Gain - Example Revisited

$$H^{cloudy}(\text{delayed}) = 0$$

$$H^{clear}(\text{delayed}) = 0$$

$$H(\text{delayed}) = 1$$

Weather	Traffic	Night flight	Flight delayed
Cloudy	No	Yes	Yes
Cloudy	Yes	No	Yes
Cloudy	Yes	No	Yes
Clear	Yes	Yes	No
Clear	No	No	No
Clear	No	No	No

$$H_W^{weather}(\text{delayed}) = 0$$

Weather	Flight delayed
Cloudy	Yes
Cloudy	Yes
Cloudy	Yes
Clear	No
Clear	No
Clear	No

Information Gain - Example Revisited

$$H(\text{delayed}) = 1$$

Weather	Traffic	Night flight	Flight delayed
Cloudy	No	Yes	Yes
Cloudy	Yes	No	Yes
Cloudy	Yes	No	Yes
Clear	Yes	Yes	No
Clear	No	No	No
Clear	No	No	No

$$H^{\text{cloudy}}(\text{delayed}) = 0$$

$$H^{\text{clear}}(\text{delayed}) = 0$$

$$H_W^{\text{weather}}(\text{delayed}) = 0$$

Weather	Flight delayed
Cloudy	Yes
Cloudy	Yes
Cloudy	Yes
Clear	No
Clear	No
Clear	No

$$H^{\text{traffic-yes}}(\text{delayed}) = 0.92$$

$$H^{\text{traffic-no}}(\text{delayed}) = 0.92$$

$$H_W^{\text{traffic}}(\text{delayed}) = 0.92$$

Traffic	Flight delayed
No	Yes
Yes	Yes
Yes	Yes
Yes	No
No	No
No	No

Information Gain - Example Revisited

$$H(\text{delayed}) = 1$$

Weather	Traffic	Night flight	Flight delayed
Cloudy	No	Yes	Yes
Cloudy	Yes	No	Yes
Cloudy	Yes	No	Yes
Clear	Yes	Yes	No
Clear	No	No	No
Clear	No	No	No

$$H^{\text{cloudy}}(\text{delayed}) = 0$$

$$H^{\text{clear}}(\text{delayed}) = 0$$

$$H_W^{\text{weather}}(\text{delayed}) = 0$$

Weather	Flight delayed
Cloudy	Yes
Cloudy	Yes
Cloudy	Yes
Clear	No
Clear	No
Clear	No

$$H^{\text{traffic-yes}}(\text{delayed}) = 0.92$$

$$H^{\text{traffic-no}}(\text{delayed}) = 0.92$$

$$H_W^{\text{traffic}}(\text{delayed}) = 0.92$$

Traffic	Flight delayed
No	Yes
Yes	Yes
Yes	Yes
Yes	No
No	No
No	No

$$H^{\text{night-yes}}(\text{delayed}) = 0$$

$$H^{\text{night-no}}(\text{delayed}) = 1$$

$$H_W^{\text{night-flight}}(\text{delayed}) \approx 0.67$$

Night flight	Flight delayed
Yes	No
No	Yes
No	Yes
Yes	No
No	No
No	No

Information Gain - Example Revisited

$$H(\text{delayed}) = 1$$

Weather	Traffic	Night flight	Flight delayed
Cloudy	No	Yes	Yes
...

$$\begin{aligned} H^{\text{cloudy}}(\text{delayed}) &= 0 \\ H^{\text{clear}}(\text{delayed}) &= 0 \end{aligned}$$

$$H_W^{\text{weather}}(\text{delayed}) = 0$$

Weather	Flight delayed
Cloudy	Yes
...	...

$$\begin{aligned} H^{\text{traffic-yes}}(\text{delayed}) &= 0.92 \\ H^{\text{traffic-no}}(\text{delayed}) &= 0.92 \end{aligned}$$

$$H_W^{\text{traffic}}(\text{delayed}) = 0.92$$

Traffic	Flight delayed
No	Yes
...	...

$$\begin{aligned} H^{\text{night-yes}}(\text{delayed}) &= 0 \\ H^{\text{night-no}}(\text{delayed}) &= 1 \end{aligned}$$

$$H_W^{\text{night-flight}}(\text{delayed}) \approx 0.67$$

Night flight	Flight delayed
Yes	No
...	...

$$IG(\text{weather}) = H(\text{delayed}) - H_W^{\text{weather}}(\text{delayed}) = 1 - 0 = 1$$



Information Gain - Example Revisited

$H(\text{delayed}) = 1$			
Weather	Traffic	Night flight	Flight delayed
Cloudy	No	Yes	Yes
...

$$H^{\text{cloudy}}(\text{delayed}) = 0$$

$$H^{\text{clear}}(\text{delayed}) = 0$$

$$H_W^{\text{weather}}(\text{delayed}) = 0$$

Weather	Flight delayed
Cloudy	Yes
...	...

$$H^{\text{traffic-yes}}(\text{delayed}) = 0.92$$

$$H^{\text{traffic-no}}(\text{delayed}) = 0.92$$

$$H_W^{\text{traffic}}(\text{delayed}) = 0.92$$

Traffic	Flight delayed
No	Yes
...	...

$$H^{\text{night-yes}}(\text{delayed}) = 0$$

$$H^{\text{night-no}}(\text{delayed}) = 1$$

$$H_W^{\text{night-flight}}(\text{delayed}) \approx 0.67$$

Night flight	Flight delayed
Yes	No
...	...

$$IG(\text{weather}) = H(\text{delayed}) - H_W^{\text{weather}}(\text{delayed}) = 1 - 0 = 1$$

$$IG(\text{traffic}) = H(\text{delayed}) - H_W^{\text{traffic}}(\text{delayed}) = 1 - 0.92 = 0.08$$

Information Gain - Example Revisited

$H(\text{delayed}) = 1$			
Weather	Traffic	Night flight	Flight delayed
Cloudy	No	Yes	Yes
...

$$H^{\text{cloudy}}(\text{delayed}) = 0$$

$$H^{\text{clear}}(\text{delayed}) = 0$$

$$H_W^{\text{weather}}(\text{delayed}) = 0$$

Weather	Flight delayed
Cloudy	Yes
...	...

$$H^{\text{traffic-yes}}(\text{delayed}) = 0.92$$

$$H^{\text{traffic-no}}(\text{delayed}) = 0.92$$

$$H_W^{\text{traffic}}(\text{delayed}) = 0.92$$

Traffic	Flight delayed
No	Yes
...	...

$$H^{\text{night-yes}}(\text{delayed}) = 0$$

$$H^{\text{night-no}}(\text{delayed}) = 1$$

$$H_W^{\text{night-flight}}(\text{delayed}) \approx 0.67$$

Night flight	Flight delayed
Yes	No
...	...

$$IG(\text{weather}) = H(\text{delayed}) - H_W^{\text{weather}}(\text{delayed}) = 1 - 0 = 1$$

$$IG(\text{traffic}) = H(\text{delayed}) - H_W^{\text{traffic}}(\text{delayed}) = 1 - 0.92 = 0.08$$

$$IG(\text{night-flight}) = H(\text{delayed}) - H_W^{\text{night-flight}}(\text{delayed}) = 1 - 0.67 = 0.33$$

Information Gain - Example Revisited

$H(\text{delayed}) = 1$			
Weather	Traffic	Night flight	Flight delayed
Cloudy	No	Yes	Yes
...

$$H^{\text{cloudy}}(\text{delayed}) = 0$$

$$H^{\text{clear}}(\text{delayed}) = 0$$

$$H_W^{\text{weather}}(\text{delayed}) = 0$$

Weather	Flight delayed
Cloudy	Yes
...	...

$$H^{\text{traffic-yes}}(\text{delayed}) = 0.92$$

$$H^{\text{traffic-no}}(\text{delayed}) = 0.92$$

$$H_W^{\text{traffic}}(\text{delayed}) = 0.92$$

Traffic	Flight delayed
No	Yes
...	...

$$H^{\text{night-yes}}(\text{delayed}) = 0$$

$$H^{\text{night-no}}(\text{delayed}) = 1$$

$$H_W^{\text{night-flight}}(\text{delayed}) \approx 0.67$$

Night flight	Flight delayed
Yes	No
...	...

$$IG(\text{weather}) = H(\text{delayed}) - H_W^{\text{weather}}(\text{delayed}) = 1 - 0 = 1$$

good

$$IG(\text{traffic}) = H(\text{delayed}) - H_W^{\text{traffic}}(\text{delayed}) = 1 - 0.92 = 0.08$$

worst

$$IG(\text{night-flight}) = H(\text{delayed}) - H_W^{\text{night-flight}}(\text{delayed}) = 1 - 0.67 = 0.33$$

not so good

ID3 (Iterative Dichotomiser 3) - Key Idea

Approach

1. For each feature: calculate the resulting entropy splitting the dataset \mathcal{X} using the selected feature
2. Split the set \mathcal{X} into subsets using the feature for which the resulting entropy (after splitting) is minimal (equivalently, information gain is maximum)
3. Create a decision tree node based on that feature
4. Recurse on subsets using remaining features (until stopping criteria are reached)

When to Stop?

Three stopping criteria

- When all of the instances have the same classification (**label = consensus value**)
- When there are no features left (**label = majority value**)
- When the dataset is empty (**label = majority parent**)

Algorithm

ID3 algorithm:

1. **if** all the instances in X have the same classification
 - (a) **return** a decision tree with one leaf node with consensus value as a label
2. **else if** there are no features left
 - (a) **return** a decision tree with one leaf node with majority value as a label
3. **else if** the dataset is empty
 - (a) **return** a decision tree with one leaf node with majority parent value as a label

three
stopping
criteria

4. **else**
 - (a) pick a feature that maximizes information gain
 - (b) once a feature is picked along a path from the root, it cannot be used again
 - (c) create subproblems based on the selected feature

recursively
constructing
the tree

Example

$H(\text{Customer})$

$$= -\left(\frac{2}{7} \cdot \log_2\left(\frac{2}{7}\right) + \frac{3}{7} \cdot \log_2\left(\frac{3}{7}\right) + \frac{2}{7} \cdot \log_2\left(\frac{2}{7}\right)\right)$$

$$= 1.5567$$

ID	Insurance	Education	Employment	Customer
1	Yes	Bachelor	Employed	Basic
2	Yes	High school	Unemployed	Premium
3	Yes	Bachelor	Self-employed	Premium
4	No	Bachelor	Self-employed	Basic
5	No	Master	Employed	Economy
6	Yes	Bachelor	Retired	Economy
7	Yes	Bachelor	Employed	Premium

Example

$$H(\text{Customer}) = 1.5567$$

ID	Insurance	Education	Employment	Customer
1	Yes	Bachelor	Employed	Basic
2	Yes	High school	Unemployed	Premium
3	Yes	Bachelor	Self-employed	Premium
4	No	Bachelor	Self-employed	Basic
5	No	Master	Employed	Economy
6	Yes	Bachelor	Retired	Economy
7	Yes	High school	Employed	Premium

Split by feature	Possible Values	Instances	Entropy	Overall Entropy	Information Gain	
Insurance	No	4, 5	1	1.265	$1.5567 - 1.265 = 0.2917$	
	Yes	1, 2, 3, 6, 7	1.3710			
Education	High school	2, 7	0	0.8571	$1.5567 - 0.8571 = 0.6996$	
	Master	5	0			
Employment	Bachelor	1, 3, 4, 6	1.5	0.9650	$1.5567 - 0.9650 = 0.5917$	
	Employed	1, 5, 7	1.5850			
	Unemployed	2	0			
	Self-employed	3, 4	1			
	Retired	6	0			

Example

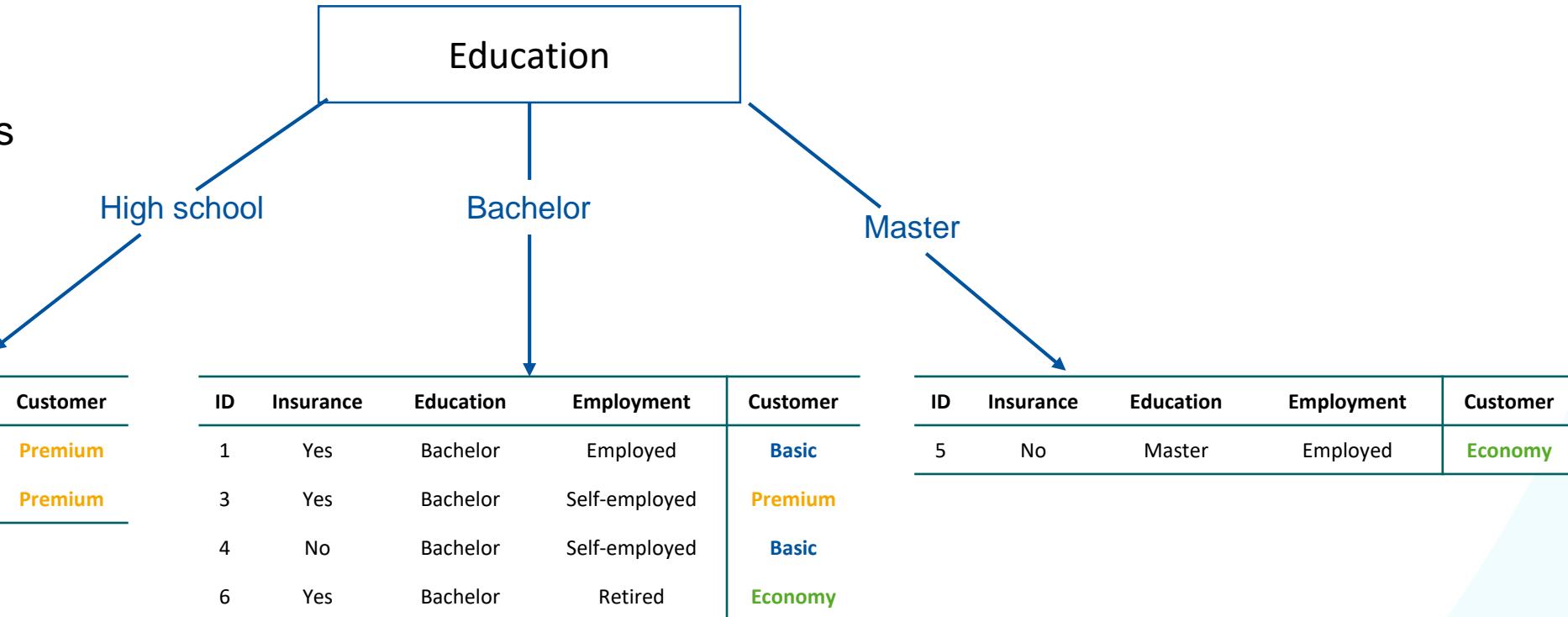
$$H(\text{Customer}) = 1.5567$$

ID	Insurance	Education	Employment	Customer
1	Yes	Bachelor	Employed	Basic
2	Yes	High school	Unemployed	Premium
3	Yes	Bachelor	Self-employed	Premium
4	No	Bachelor	Self-employed	Basic
5	No	Master	Employed	Economy
6	Yes	Bachelor	Retired	Economy
7	Yes	High school	Employed	Premium

Split by feature	Possible Values	Instances	Entropy	Overall Entropy	Information Gain	
Insurance	No	4, 5	1	1.265	$1.5567 - 1.265 = 0.2917$	
	Yes	1, 2, 3, 6, 7	1.3710			
Education	High school	2, 7	0	0.8571	$1.5567 - 0.8571 = 0.6996$	
	Master	5	0			
Employment	Bachelor	1, 3, 4, 6	1.5	0.9650	$1.5567 - 0.9650 = 0.5917$	
	Employed	1, 5, 7	1.5850			
	Unemployed	2	0	0.9650	$1.5567 - 0.9650 = 0.5917$	
	Self-employed	3, 4	1			
	Retired	6	0	0.9650	$1.5567 - 0.9650 = 0.5917$	

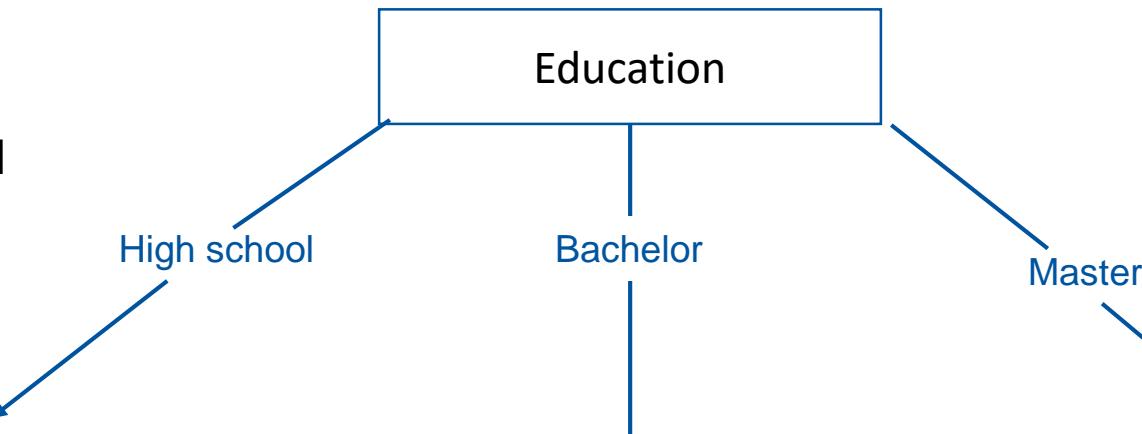
Example

Recursion until
stopping criteria holds



Example

Recursion until
stopping criteria hold



ID	Insurance	Education	Employment	Customer
2	Yes	High school	Unemployed	Premium
7	Yes	High school	Employed	Premium

consensus

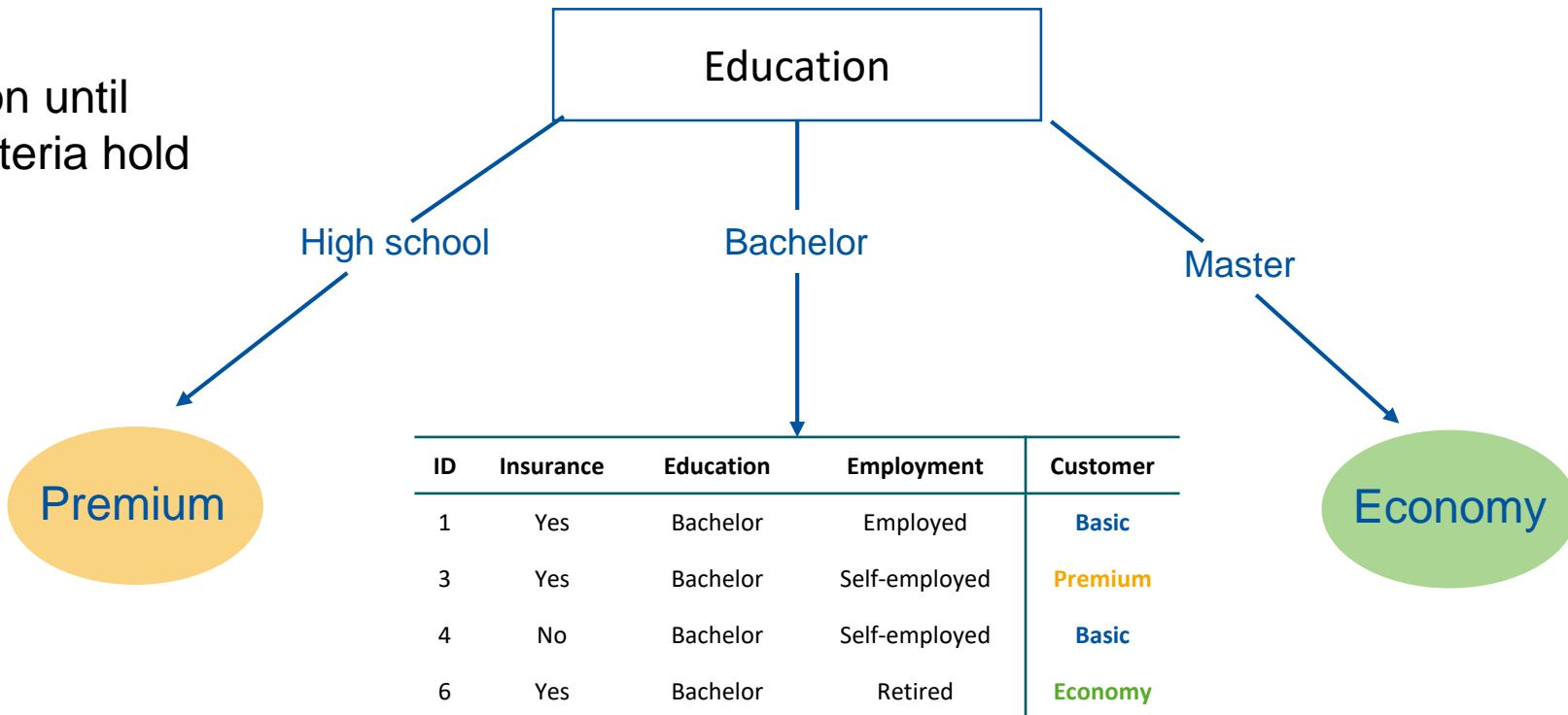
ID	Insurance	Education	Employment	Customer
1	Yes	Bachelor	Employed	Basic
3	Yes	Bachelor	Self-employed	Premium
4	No	Bachelor	Self-employed	Basic
6	Yes	Bachelor	Retired	Economy

ID	Insurance	Education	Employment	Customer
5	No	Master	Employed	Economy

consensus

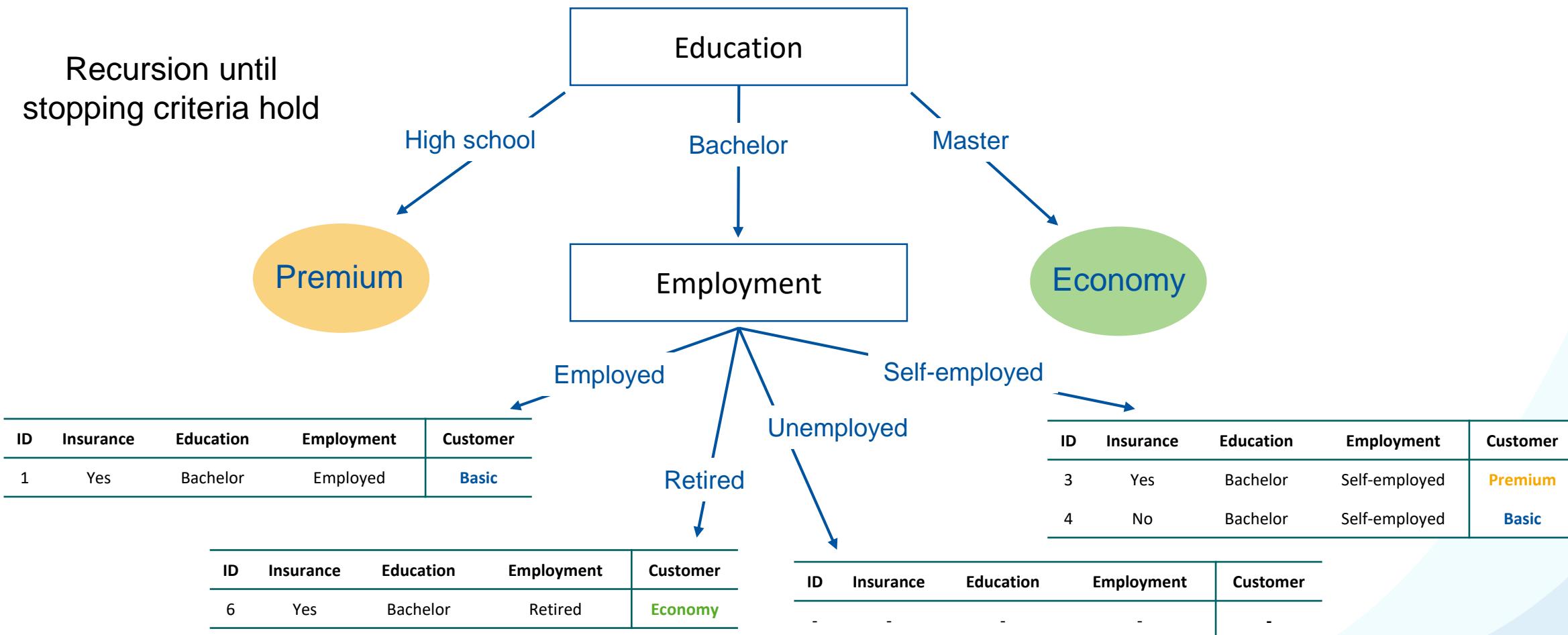
Example

Recursion until
stopping criteria hold



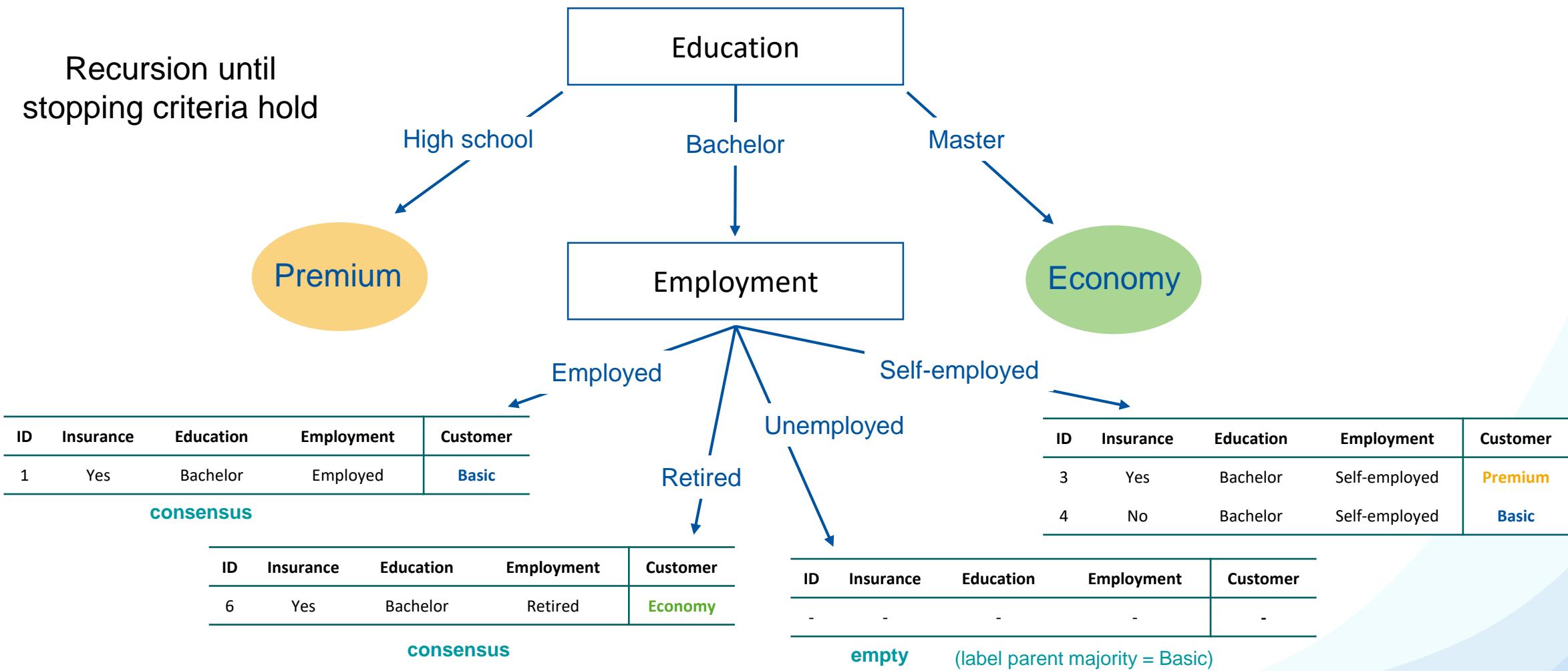
Example

Recursion until
stopping criteria hold



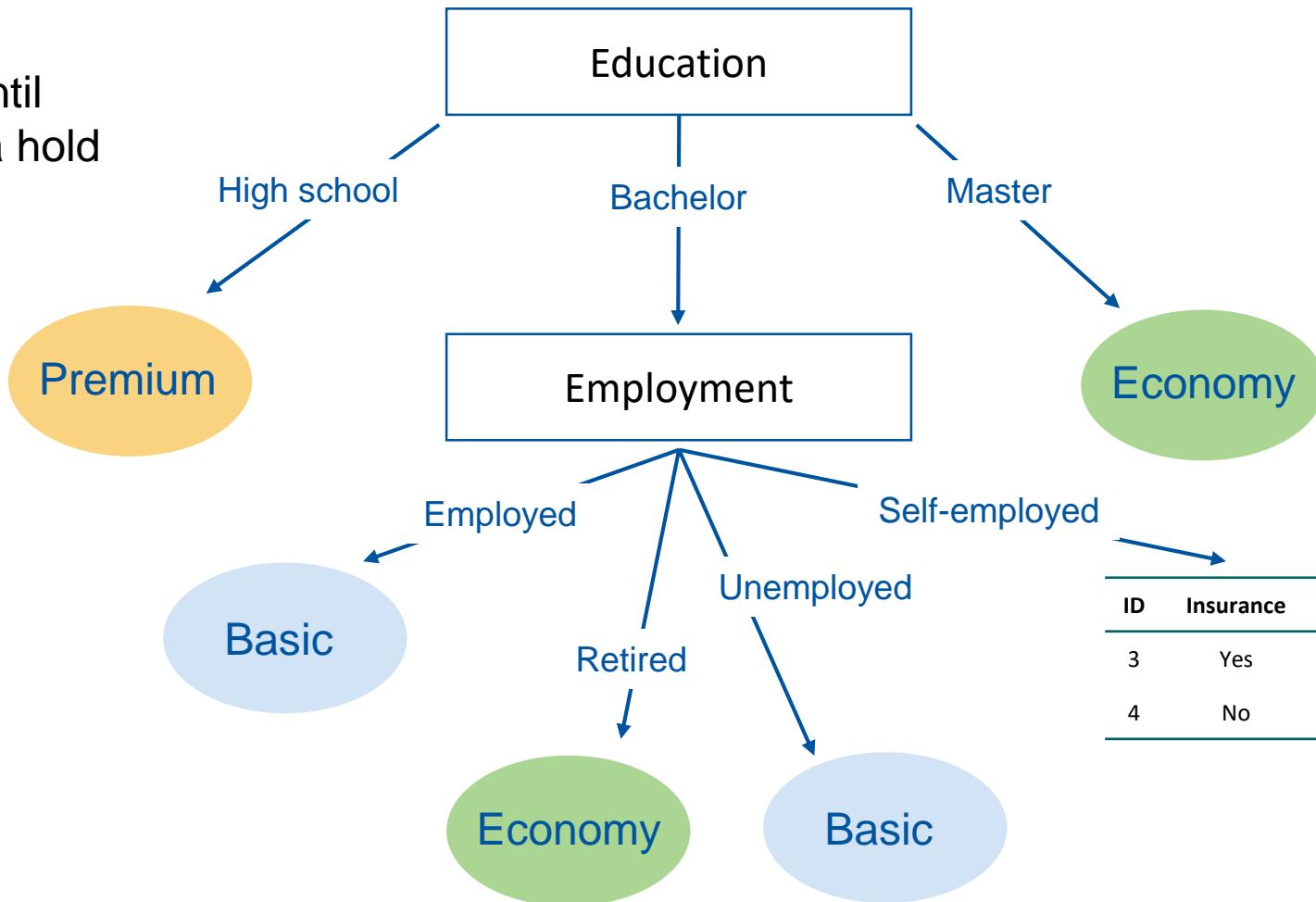
Example

Recursion until
stopping criteria hold



Example

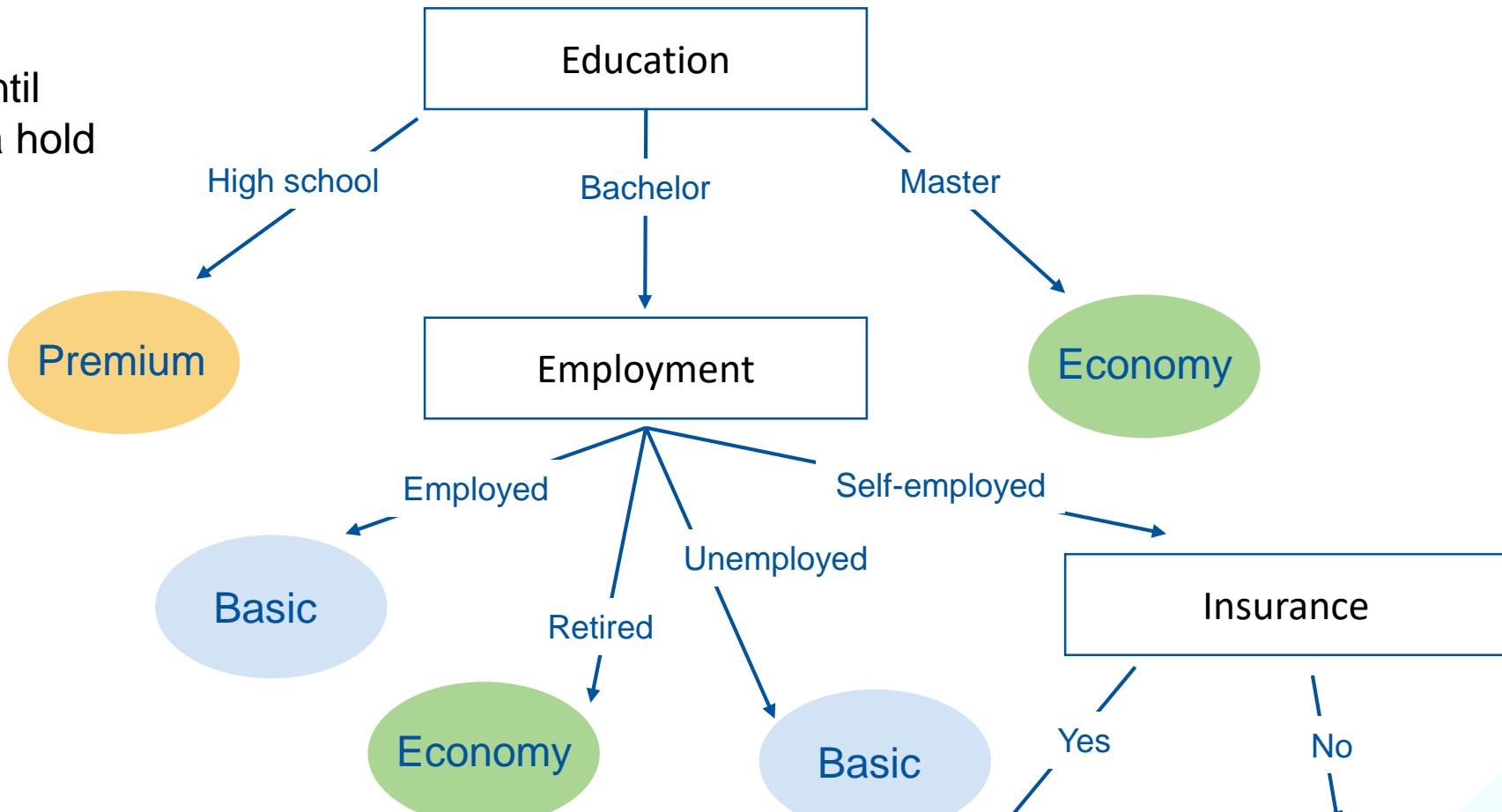
Recursion until
stopping criteria hold



ID	Insurance	Education	Employment	Customer
3	Yes	Bachelor	Self-employed	Premium
4	No	Bachelor	Self-employed	Basic

Example

Recursion until
stopping criteria hold



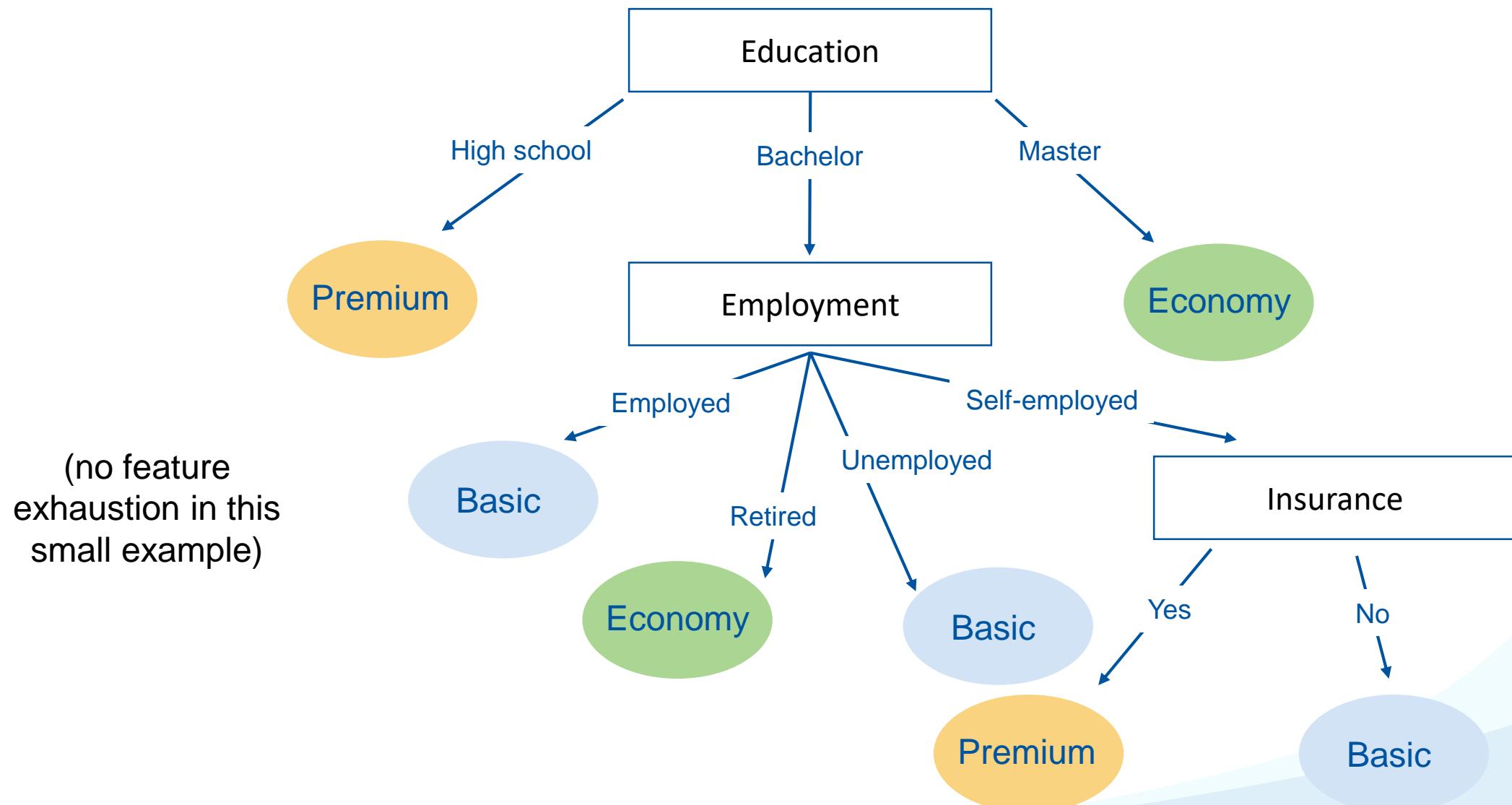
ID	Insurance	Education	Employment	Customer
3	Yes	Bachelor	Self-employed	Premium

consensus

ID	Insurance	Education	Employment	Customer
4	No	Bachelor	Self-employed	Basic

consensus

Example



Alternative Information Gain Notions

- Information gain aims to measure the improvement in purity / predictability / compressibility
- Example approaches:
 - Entropy-based information gain (IG)
 - Information gain ratio (GR)
 - Gini index (Gini)

Alternative Information Gain Notions

- Information gain aims to measure the improvement in purity / predictability / compressibility
- Example approaches:
 - **Entropy-based information gain (IG)**
 - Information gain ratio (GR)
 - Gini index (Gini)

Entropy of target feature t before splitting

$$H(t) = - \sum_{k=1}^K (P(t = k) \cdot \log_s(P(t = k)))$$

$$H_W(t) = \sum_{node \in nodes(d)} \left(\frac{|node|}{N} \cdot H_{node}(t) \right)$$

Weighted entropy of target feature t after splitting based on d

$$IG(d) = H(t) - H_W^d(t)$$

(seen before)

Alternative Information Gain Notions

- Information gain aims to measure the improvement in purity / predictability / compressibility
- Example approaches:
 - Entropy-based information gain (IG)
 - **Information gain ratio (GR)**
 - **Gini index (Gini)**

Information Gain Ratio

- Entropy-based information gain favors features with many different values (split in many subsets decreases entropy)
- Information gain ratio addresses this issue:

$$GR(d) = \frac{IG(d)}{H(d)} = \frac{H(t) - H_W^d(t)}{-\sum_{k=1}^K (P(d=k) \cdot \log_2(P(d=k)))}$$

Information gain when splitting based on descriptive feature d

Entropy of target feature t

Overall entropy of target feature t after splitting based on d

Entropy of descriptive feature d

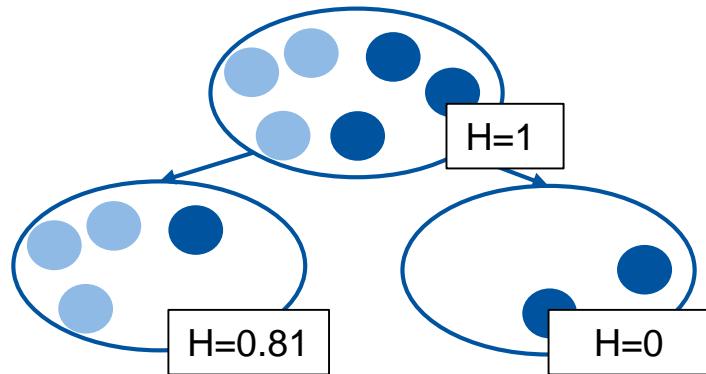
d can take K possible values

Probability of d taking the k th possible value

→ we can think of it as making an absolute value relative

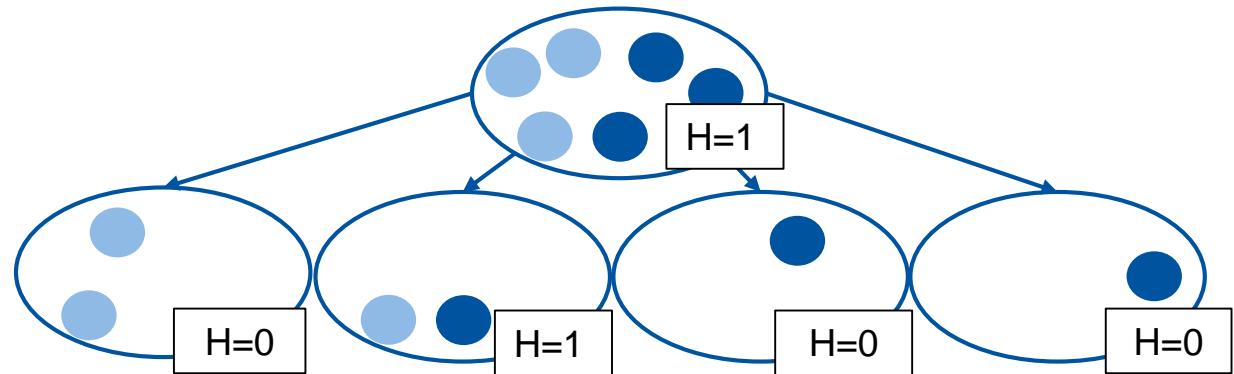
Information Gain Ratio - Example

split based on feature d



$$H_W^d(\text{color}) = \frac{4}{6} \cdot 0.81 = 0.54$$
$$IG(d) = 0.46$$

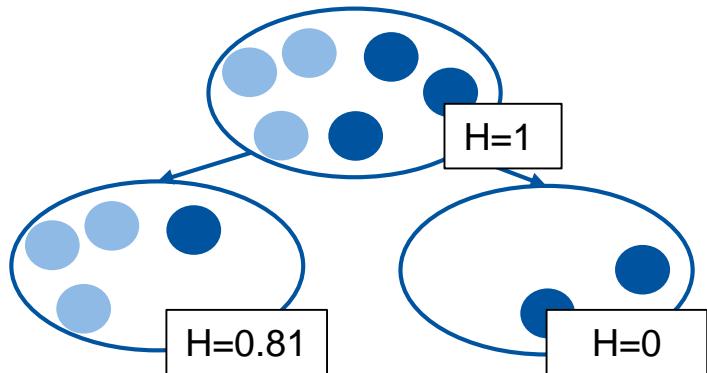
split based on feature d'



$$H_W^{d'}(\text{color}) = \frac{2}{6} \cdot 1 = 0.33$$
$$IG(d') = 0.67$$

Information Gain Ratio - Example

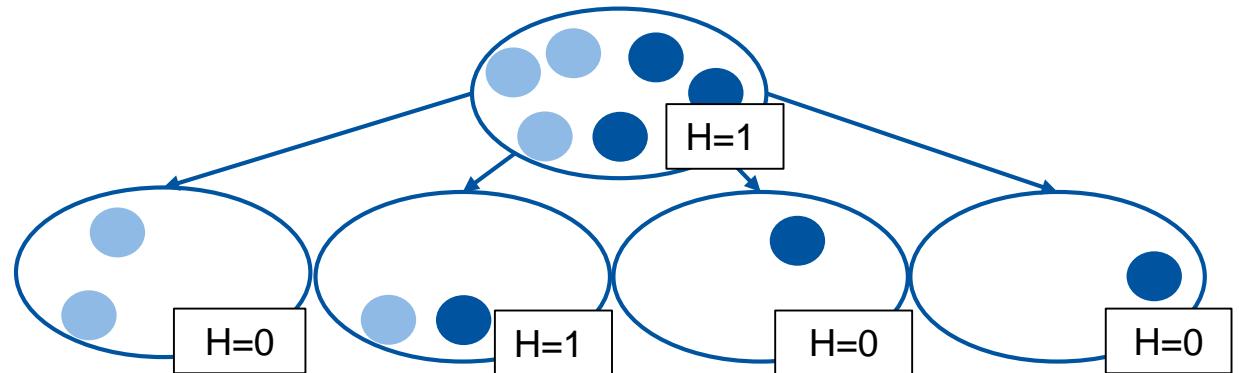
split based on feature d



$$IG(d) = 0.46$$

$$\begin{aligned} GR(d) &= \frac{0.46}{-(\frac{4}{6} \cdot \log_2(\frac{4}{6}) + \frac{2}{6} \cdot \log_2(\frac{2}{6}))} \\ &= \frac{0.46}{0.92} = 0.5 \end{aligned}$$

split based on feature d'



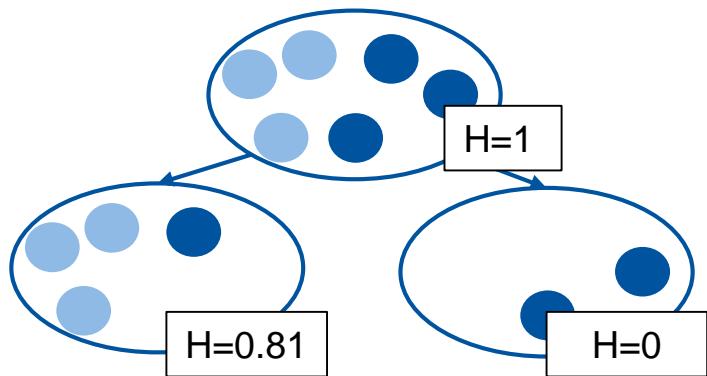
$$IG(d') = 0.67$$

$$\begin{aligned} GR(d') &= \frac{0.67}{-(\frac{2}{6} \cdot \log_2(\frac{2}{6}) + \frac{2}{6} \cdot \log_2(\frac{2}{6}) + \frac{1}{6} \cdot \log_2(\frac{1}{6}) + \frac{1}{6} \cdot \log_2(\frac{1}{6}))} \\ &= \frac{0.67}{1.92} = 0.35 \end{aligned}$$

$$GR(d) = \frac{IG(d)}{H(d)} = \frac{H(t) - H_W^d(t)}{-\sum_{k=1}^K (P(d=k) \cdot \log_2(P(d=k)))}$$

Information Gain Ratio - Example

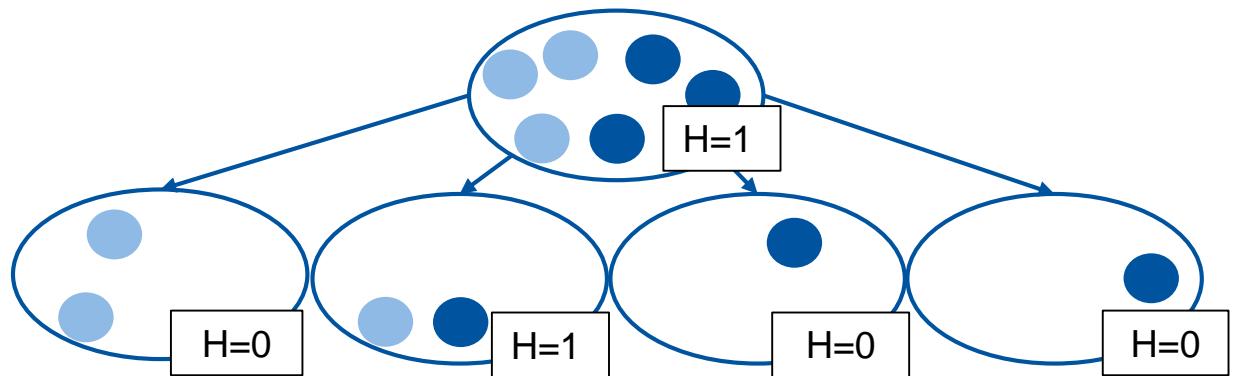
split based on feature d



$$IG(d) = 0.46 \quad \text{👎}$$

$$\begin{aligned} GR(d) &= \frac{0.46}{-(\frac{4}{6} \cdot \log_2(\frac{4}{6}) + \frac{2}{6} \cdot \log_2(\frac{2}{6}))} \\ &= \frac{0.46}{0.92} = 0.5 \quad \text{👍} \end{aligned}$$

split based on feature d'



$$IG(d') = 0.67 \quad \text{👍}$$

$$\begin{aligned} GR(d') &= \frac{0.67}{-(\frac{2}{6} \cdot \log_2(\frac{2}{6}) + \frac{2}{6} \cdot \log_2(\frac{2}{6}) + \frac{1}{6} \cdot \log_2(\frac{1}{6}) + \frac{1}{6} \cdot \log_2(\frac{1}{6}))} \\ &= \frac{0.67}{1.92} = 0.35 \quad \text{👎} \end{aligned}$$

$$GR(d) = \frac{IG(d)}{H(d)} = \frac{H(t) - H_W^d(t)}{-\sum_{k=1}^K (P(d=k) \cdot \log_2(P(d=k)))}$$

Gini Index

- An alternative measure of impurity
- Expected misclassification rate when guessing based on the observed distribution

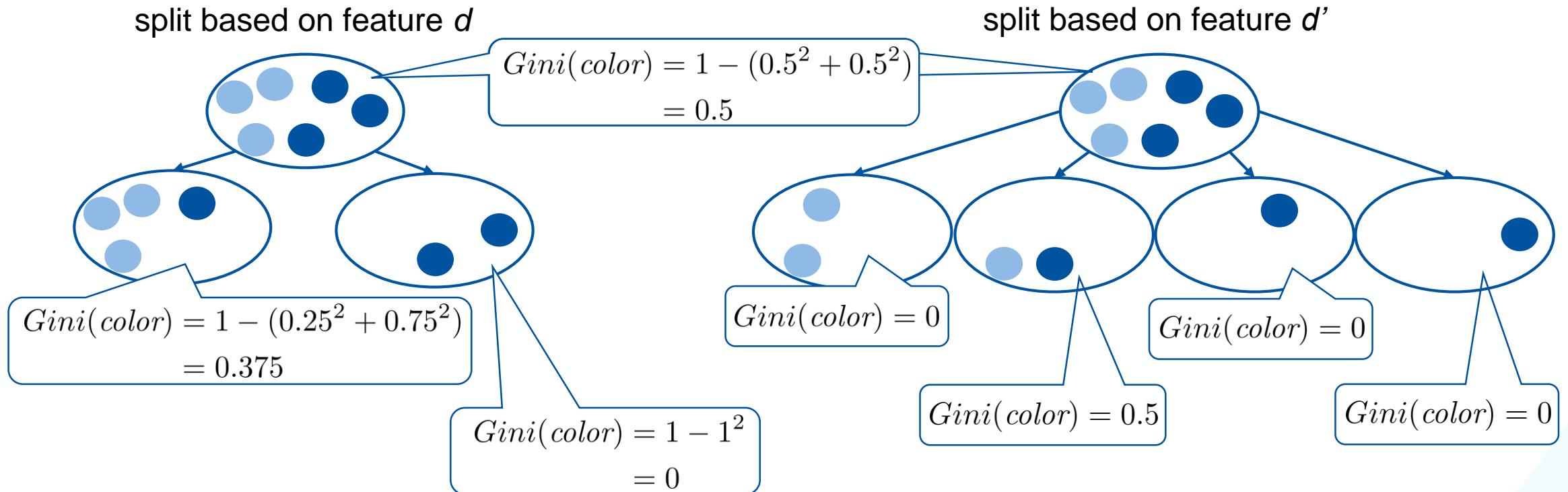
$$Gini(t) = 1 - \sum_{k=1}^K P(t = k)^2$$

t can take K possible values

Probability that t takes the k th value

- With probability $P(t = k)$ we guess that *class* equals the k th possible value and with probability $P(t = k)$ this guess is correct
- Can be seen as the probability of guessing the wrong label

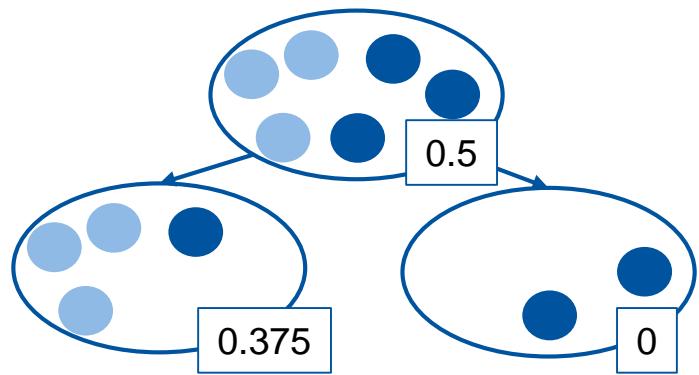
Gini Index - Example



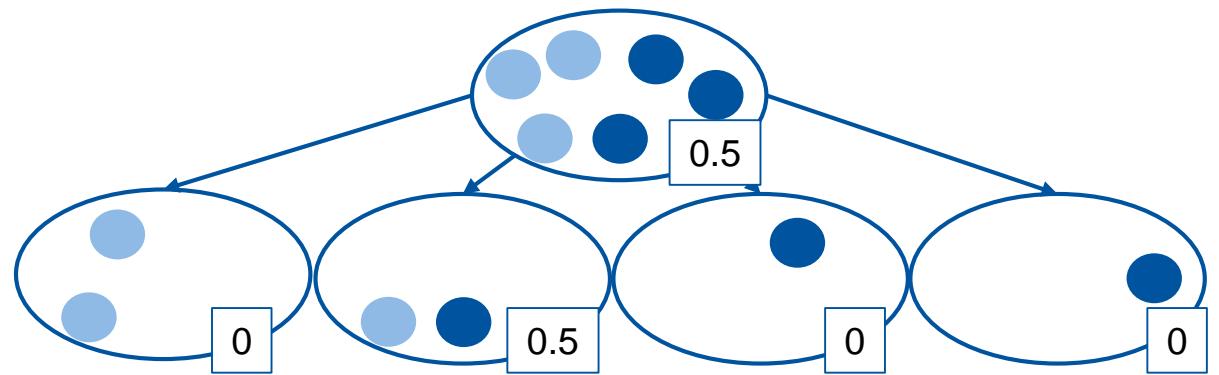
$$Gini(t) = 1 - \sum_{k=1}^K P(t = k)^2$$

Gini Index - Example

split based on feature d



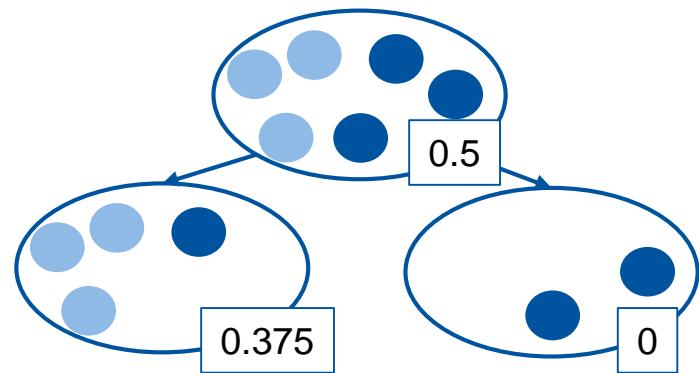
split based on feature d'



$$Gini(t) = 1 - \sum_{k=1}^K P(t = k)^2$$

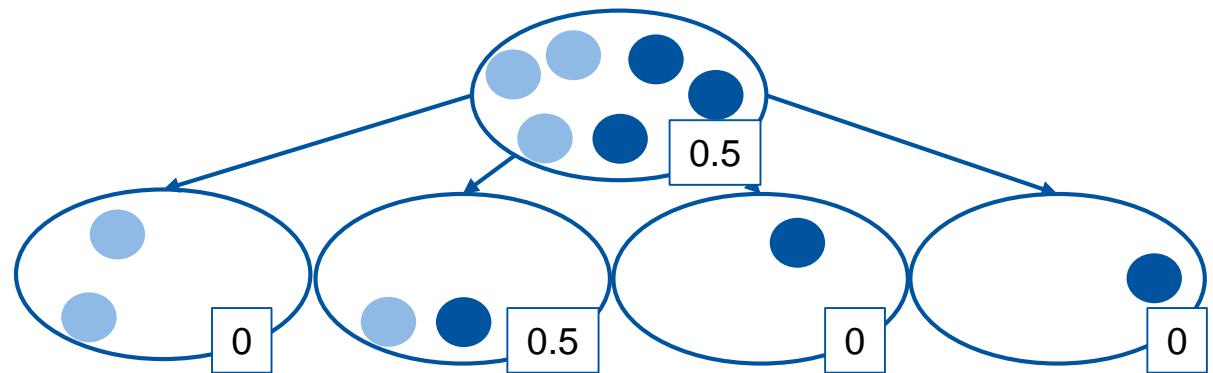
Gini Index - Example

split based on feature d



$$\begin{aligned} Gini_W(\text{color}) &= \frac{4}{6} \cdot 0.375 = 0.25 \\ IG_{Gini}(d) &= 0.5 - 0.25 = 0.25 \end{aligned}$$

split based on feature d'

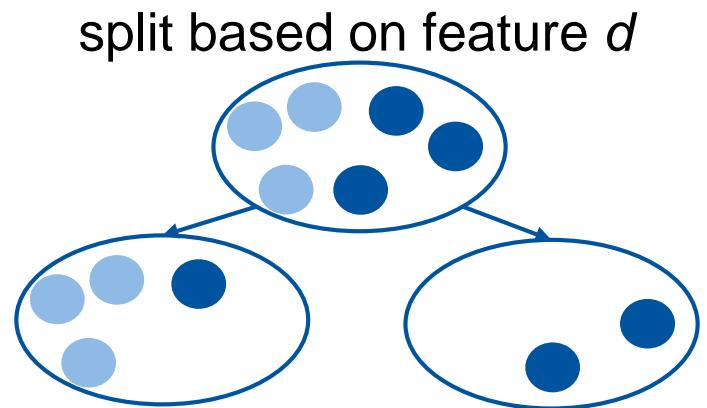


$$\begin{aligned} Gini_W(\text{color}) &= \frac{2}{6} \cdot 0.5 = 0.166 \\ IG_{Gini}(d') &= 0.5 - 0.166 = 0.33 \end{aligned}$$

compute weighted average and information gain as before

$$Gini(t) = 1 - \sum_{k=1}^K P(t = k)^2$$

Comparison



$$IG_{Entropy}(d) = 0.4591$$

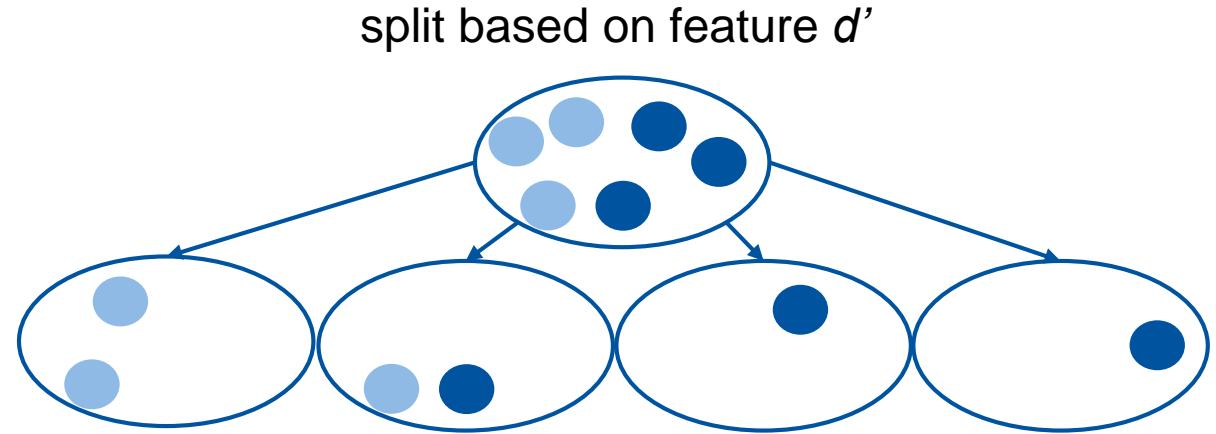
$$GR(d) = 0.5$$

$$IG_{Gini}(d) = 0.25$$

Entropy-based information gain

Information gain ratio

Gini-based information gain ratio

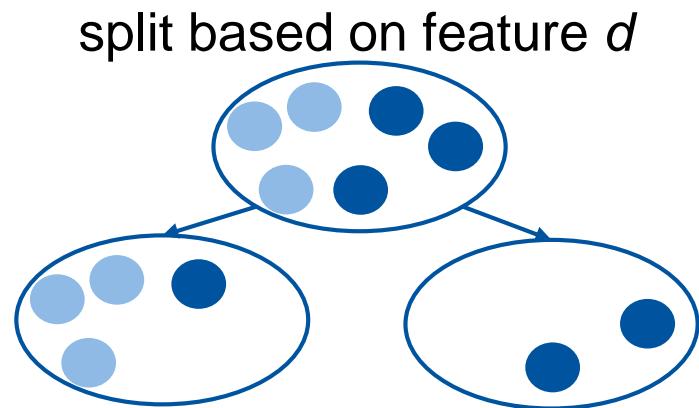


$$IG_{Entropy}(d') = 0.6667$$

$$GR(d') = 0.34$$

$$IG_{Gini}(d') = 0.33$$

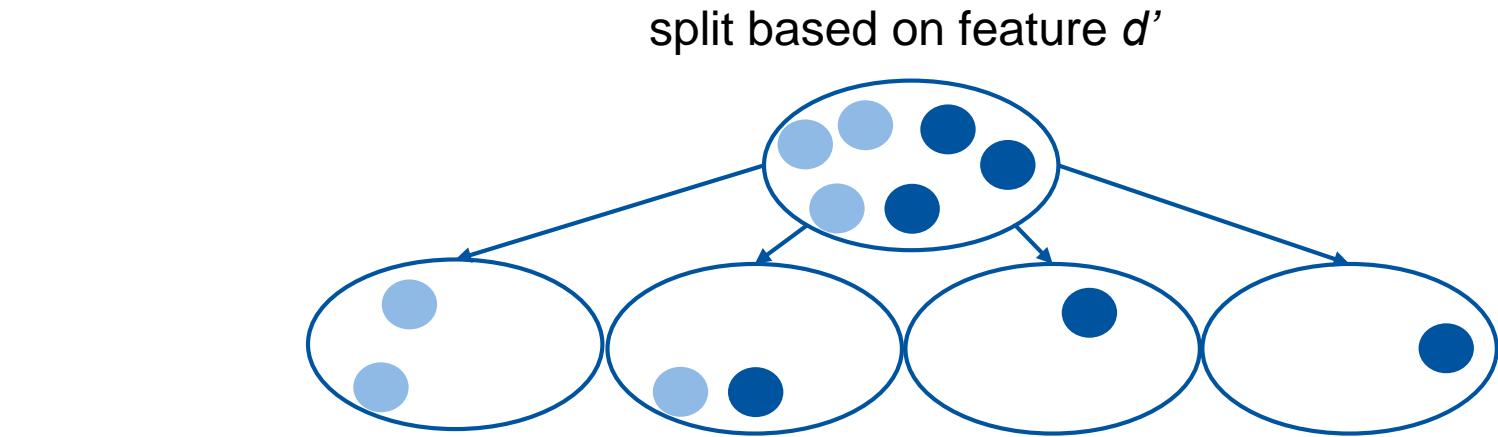
Comparison



$$IG_{Entropy}(d) = 0.4591$$

$$GR(d) = 0.5$$

$$IG_{Gini}(d) = 0.25$$



$$IG_{Entropy}(d') = 0.6667$$

$$GR(d') = 0.34$$

$$IG_{Gini}(d') = 0.33$$

Pruning Decision Trees

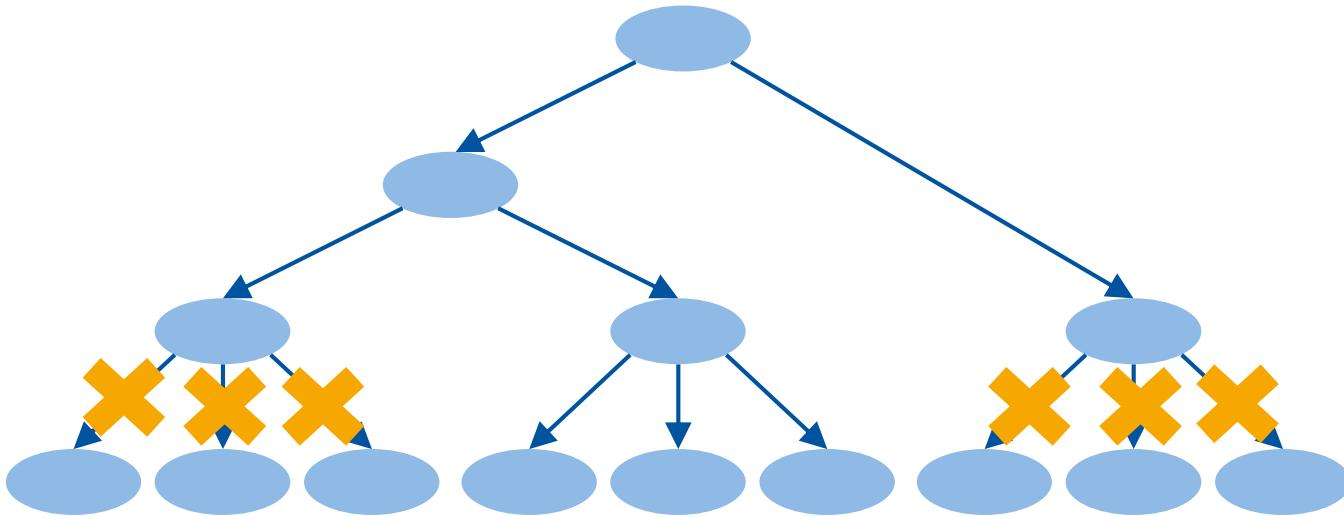
- Possible problems:
 - Decision tree is **overfitting** the data
 - Decision tree is too complex or too deep
- Two solution directions:
 - **Pre-pruning** (early stopping/forward)
 - **Post-pruning** (reduced error/backward)
- To generalize and **avoid overfitting**

Pre-pruning

- Stop creating subtrees and use **majority vote** to determine the label
- Many possible **stopping criteria**:
 - lower bound for number of instances
 - lower bound for information gain
 - ...
- May create trees that are **not consistent** with respect to the data

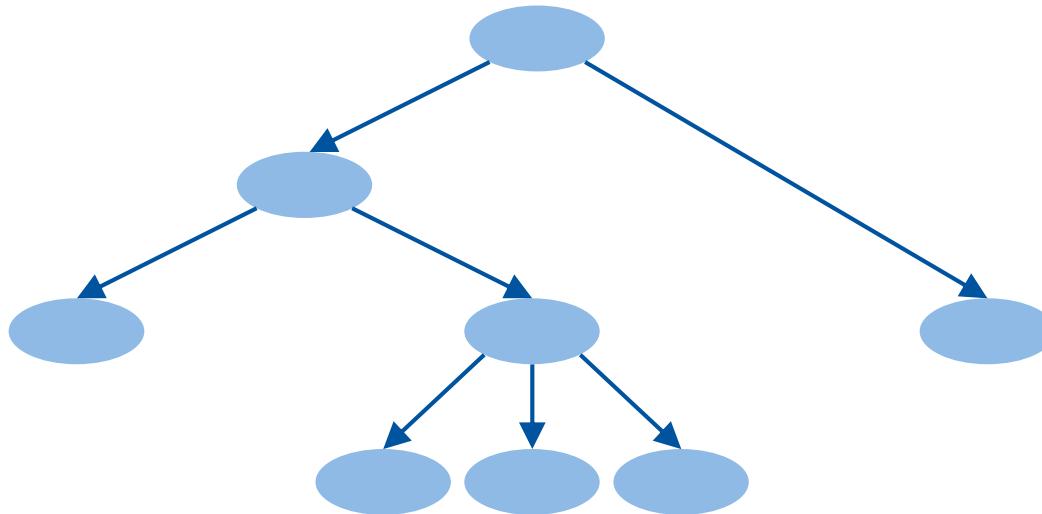
Pre-pruning

Example:
Information Gain = 0.0001
→ do not split further



Example:
Only 10 instances left
→ do not split further

Pre-pruning

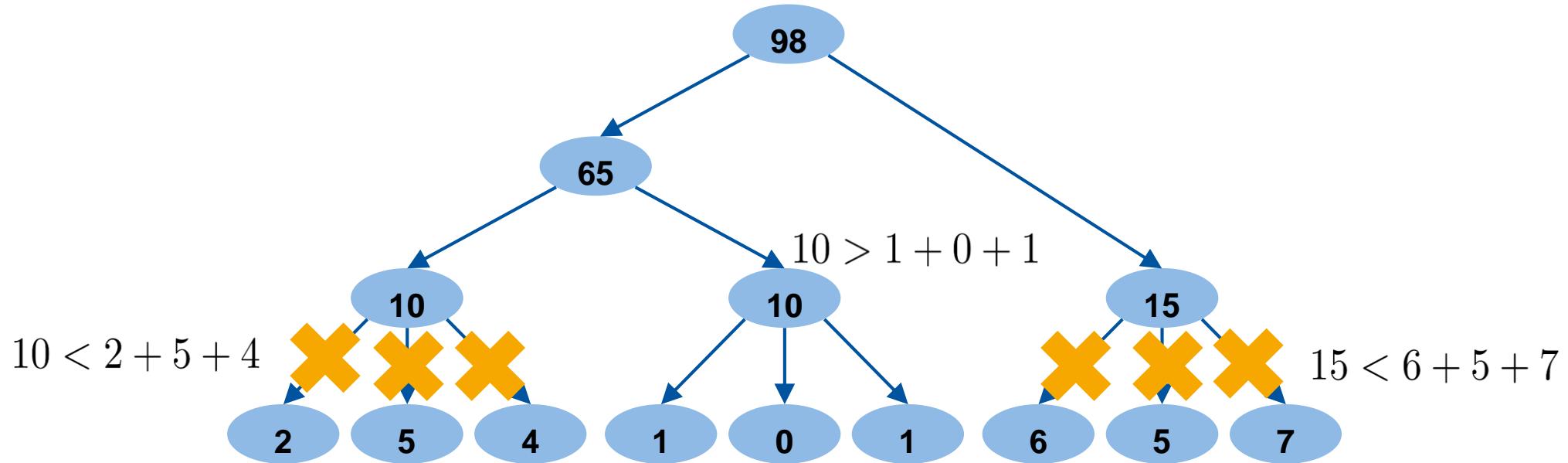


efficient, but we may miss strong dependencies at lower levels of trees

Post-pruning

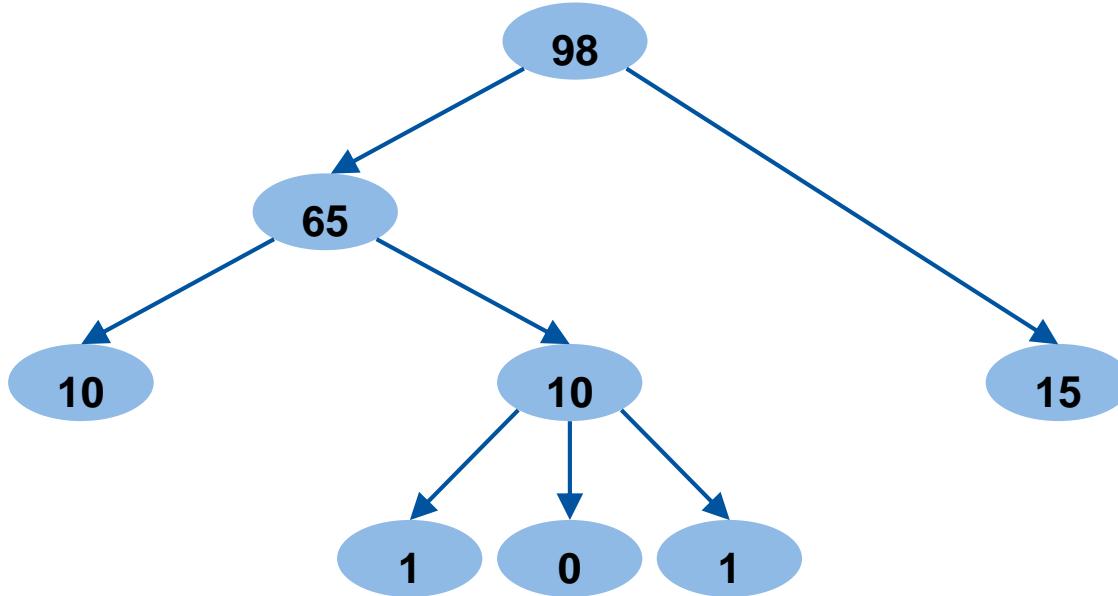
- First, build the **whole** decision tree; then **cut off branches** that do not add much
- Common approach is to **split the data** into a training set and a validation/test set
- Measure the **performance of splits** based on a validation/test set

Post-pruning



- Decision tree learned on a [training set](#)
- Numbers indicate misclassifications based on a [validation set](#)

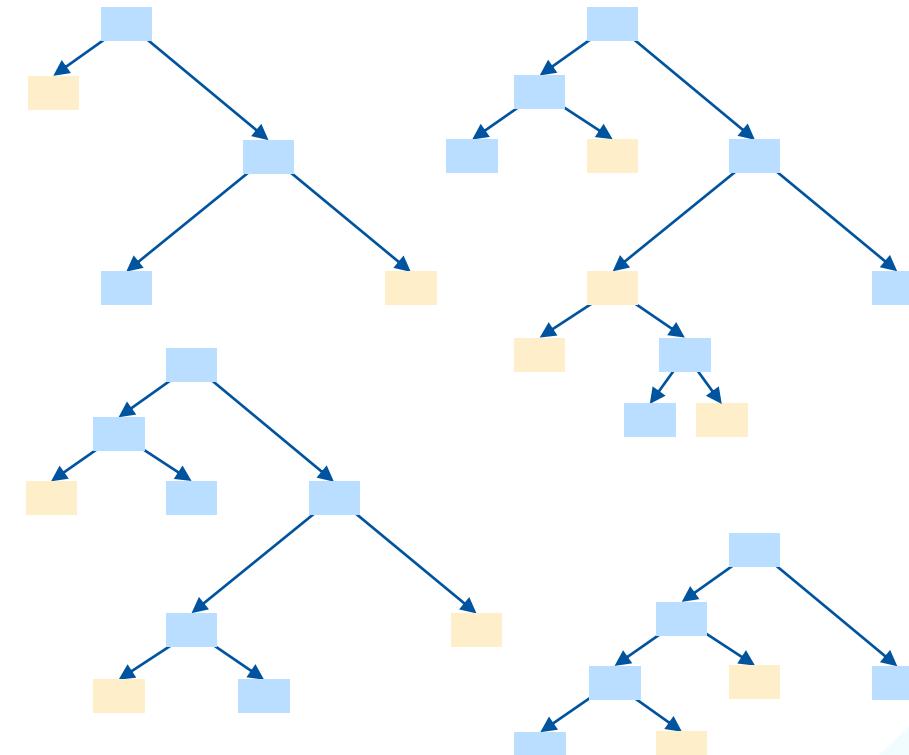
Post-pruning



Less efficient, but based on the complete tree

Idea

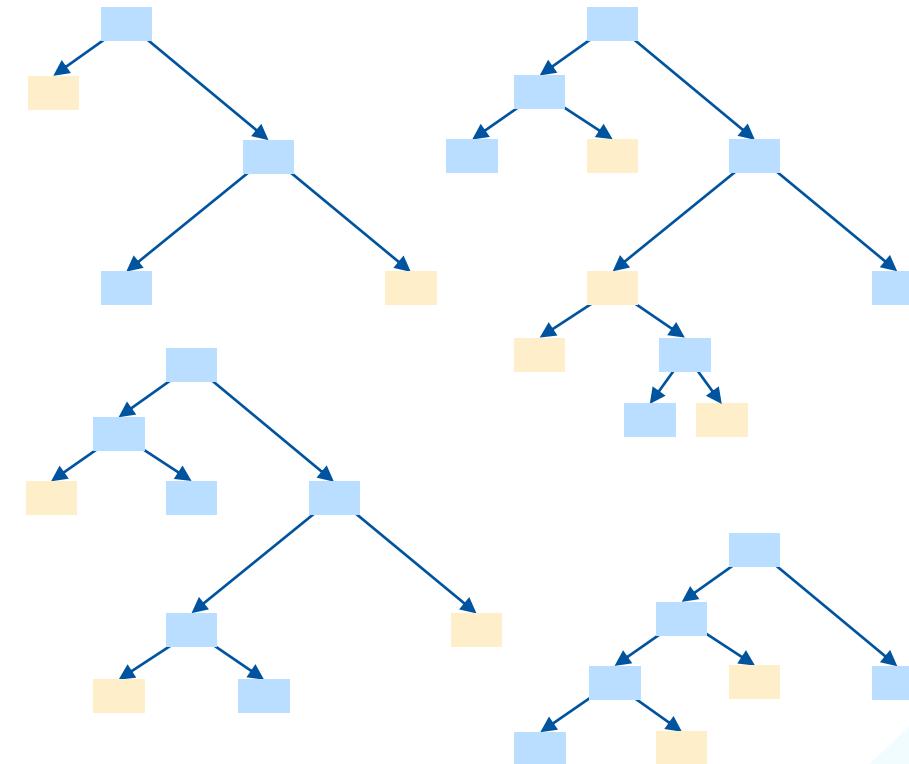
- Rather than creating a single decision tree, we aim to create a **set of trees** (called a model **ensemble**)
- Models should complement each other
- Different models can "vote" on the label (votes may be weighted)
- Multiple trees may give different answers (select the most frequent value or the average)
- Many variations of the same idea...



Boosting

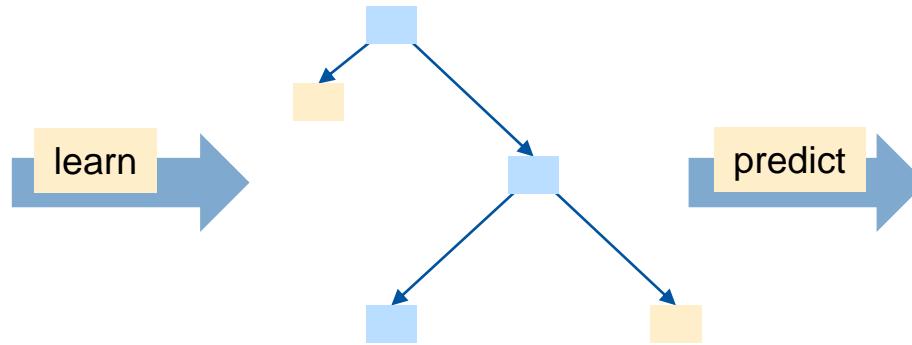
Correct iteratively

- Iteratively update the data set based on misclassifications
- Instances that are **wrongly** classified get a **higher weight** when learning the next model
- Each iteration new models are added to the ensemble



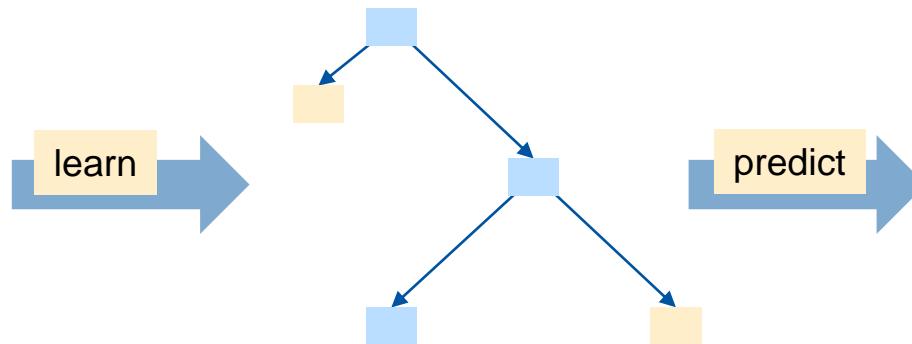
Boosting - Example

f1	f2	f3	f4	...	fD	class
						High
						High
						Low
						Medium
						High
						Low



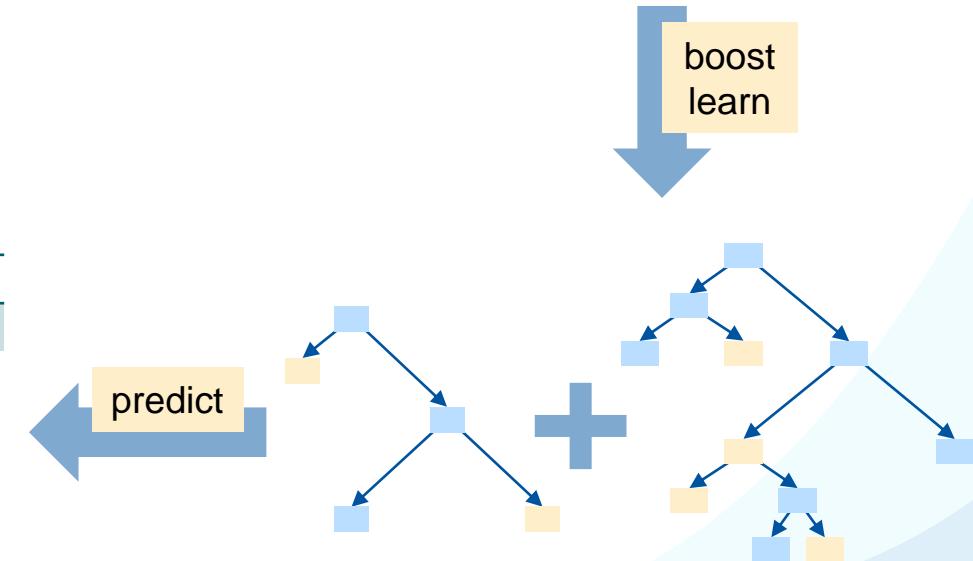
Boosting - Example

f_1	f_2	f_3	f_4	\dots	f_D	class
						High
						High
						Low
						Medium
						High
						Low

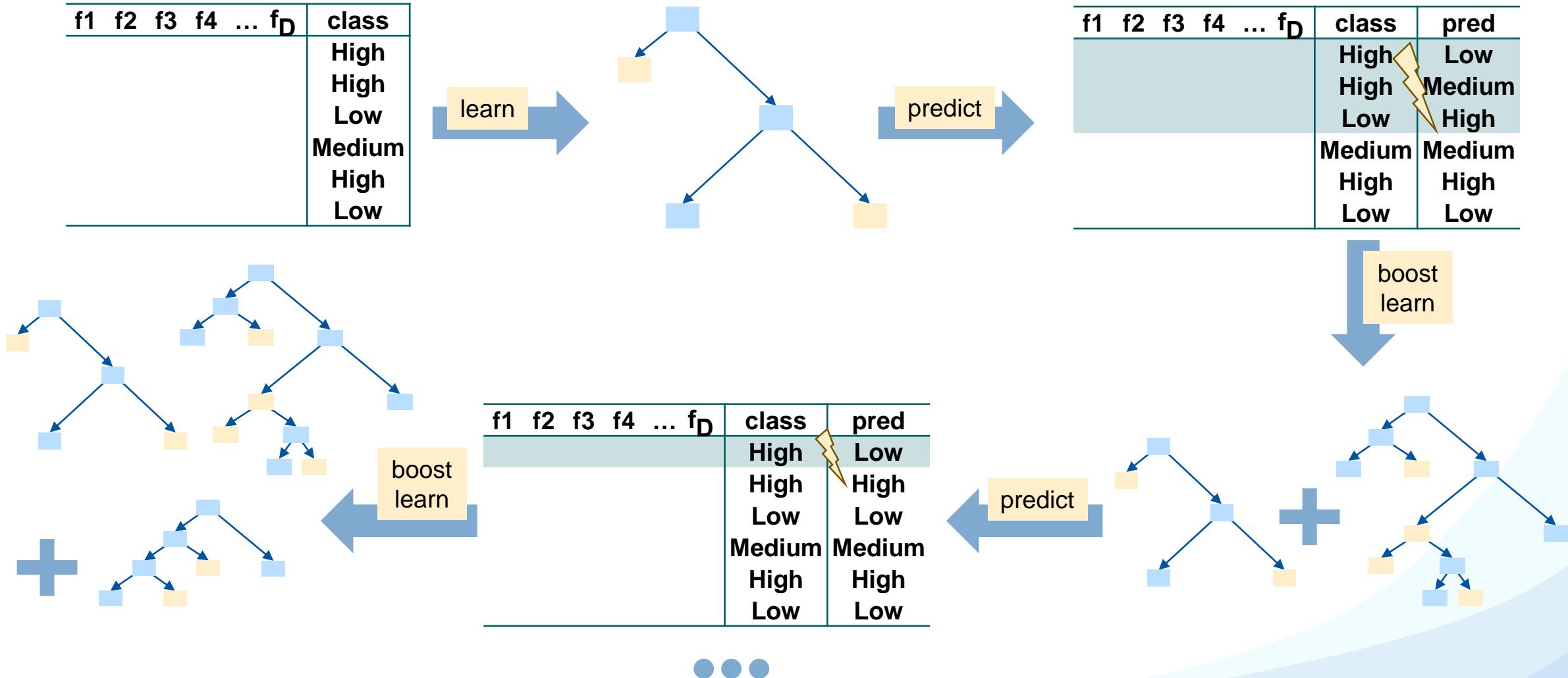


f_1	f_2	f_3	f_4	\dots	f_D	class	pred
						High	Low
						High	Medium
						Low	High
						Medium	Medium
						High	High
						Low	Low

f_1	f_2	f_3	f_4	\dots	f_D	class	pred
						High	Low
						High	High
						Low	Low
						Medium	Medium
						High	High
						Low	Low



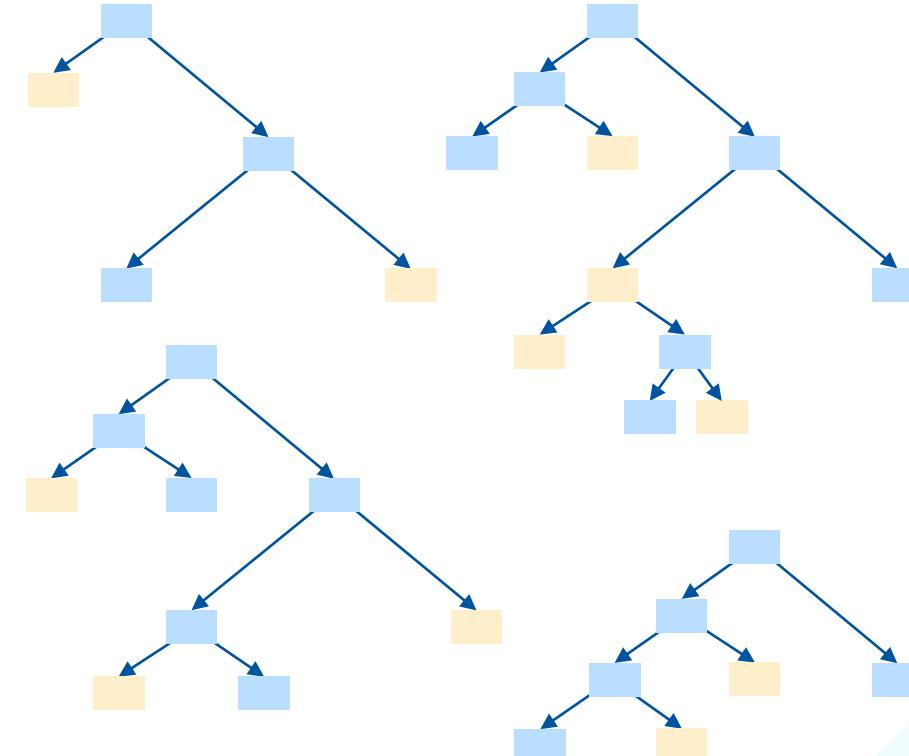
Boosting - Example



Bagging

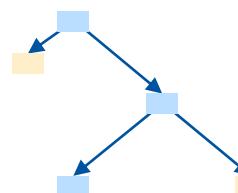
Split data upfront

- Each model is based on a [random sample](#) of the data set
- Avoids model depending on a specific sample of the data set (learning decision trees may be very sensitive to small variations)
- Many variants (e.g. remove some instances and duplicate others)

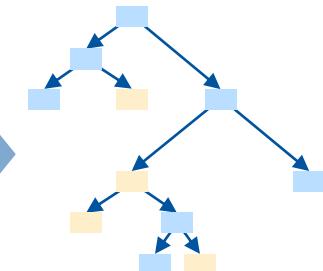


Bagging - Example

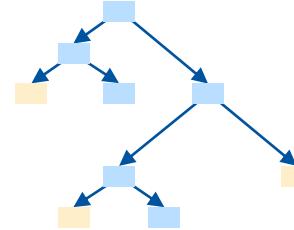
f_1	f_2	f_3	f_4	...	f_D	class
		X				High
		X				High
						Low
						Medium
						High
						Low



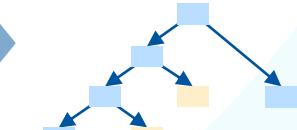
f_1	f_2	f_3	f_4	...	f_D	class
		X				High
		X				High
						Low
						Medium
						High
						Low



f_1	f_2	f_3	f_4	...	f_D	class
		X				High
		X				High
						Low
						Medium
						High
						Low

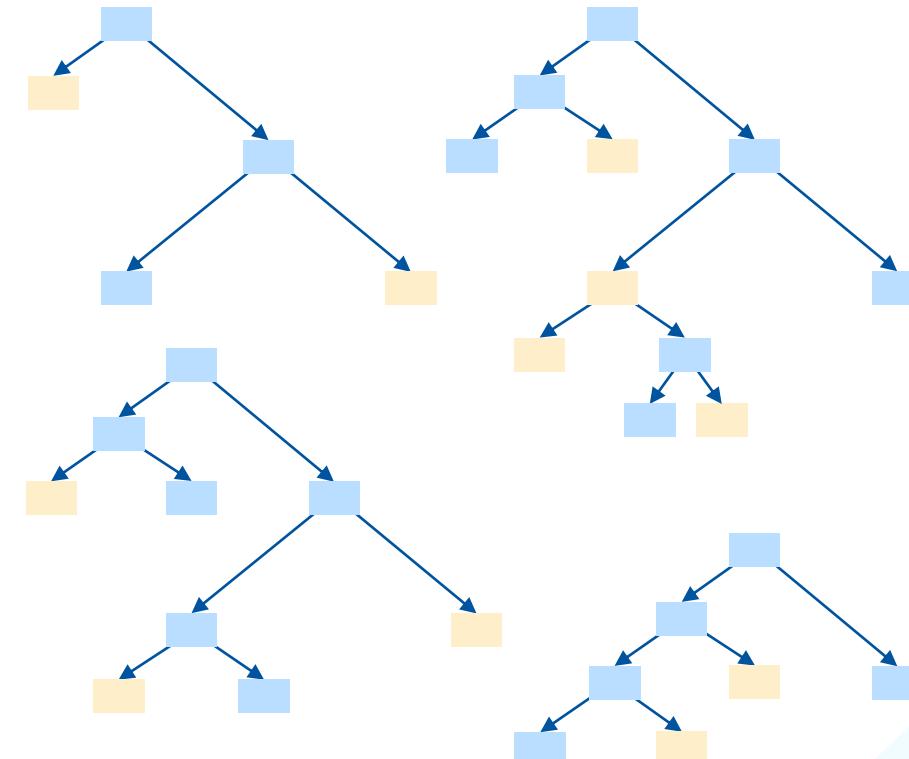


f_1	f_2	f_3	f_4	...	f_D	class
		X				High
		X				High
						Low
						Medium
						High
						Low



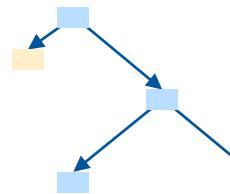
Subspace Sampling

- Each model is based on a **random set of descriptive features**
- More efficient and less likely to be overfitting when focusing on just a few features

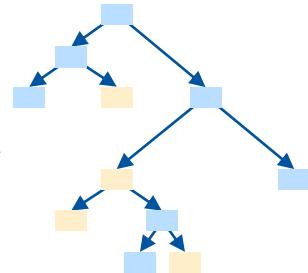


Subspace sampling

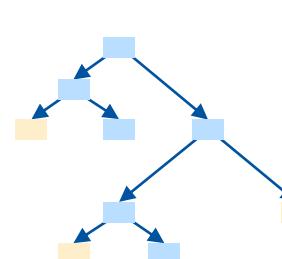
f1	X	X	f4	...	f _D	class
						High
						High
						Low
						Medium
						High
						Low



	X	f2	f3	X	...	f _D	class
							High
							High
							Low
							Medium
							High
							Low



f1	f2	f3	X	X	f _D	class
						High
						High
						Low
						Medium
						High
						Low



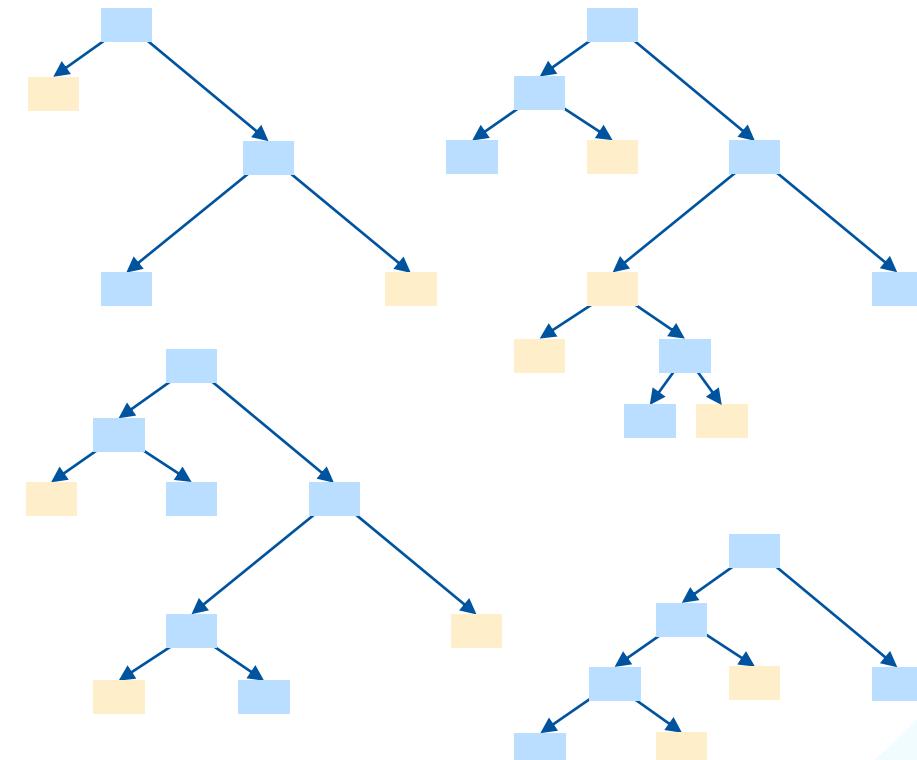
f1	f2	X	f4	...	X	class
						High
						High
						Low
						Medium
						High
						Low



Random Forest

Combine of bagging and subspace sampling

- Split data **twice**
 - random sample of instances
(bagging)
 - random set of descriptive features
(subspace sampling)
- Find a model for each subset of data created this way



Random Forest

f1	X X		f4	...	f _D	class
						High
						High
		X X				Low
						Medium
						High
						Low



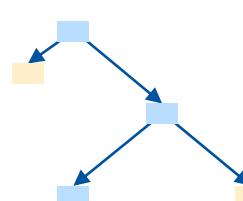
	f2	f3	X X	...	f _D	class
						High
						High
			X X			Low
						Medium
			X X			High
						Low



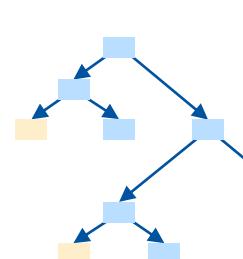
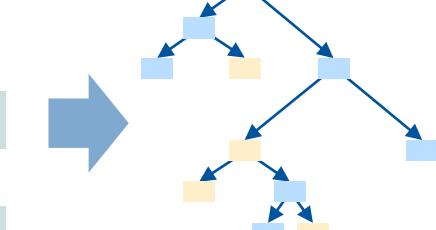
f1	f2	f3	X X	f _D	class
					High
			X X		High
					Low
					Medium
			X X		High
					Low



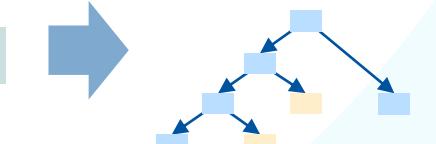
f1	f2	X X	f4	...	f _D	class
						High
						High
			X X			Low
						Medium
			X X			High
						Low



	f2	f3	X X	...	f _D	class
						High
						High
			X X			Low
						Medium
			X X			High
						Low



f1	f2	X X	f4	...	f _D	class
						High
						High
			X X			Low
						Medium
			X X			High
						Low



Dealing with Continuous Variables

- Thus far we assumed features were categorical
- We can use binning to make continuous features categorical

continuous target feature

	features				
	f_1	f_2	...	f_D	class
instances					
	high	88		59.99	5043
	high	76		50.00	4598
	low	32		39.50	3248
	low	89	continuous descriptive features	49.99	5466
	high	21		59.99	7682

continuous descriptive features

Continuous Descriptive Features

- Challenge: determine suitable boundaries (infinite number of thresholds is possible)
- Idea:
 - sort instances based on the continuous descriptive feature
 - look for changes in target feature labels
- Change points are candidate thresholds
- Select the threshold with the highest information gain



Continuous Descriptive Features - Example

ID	Insurance	Income	Employment	Customer
1	Yes	3500	Employed	Basic
2	Yes	0	Unemployed	Premium
3	Yes	1000	Self-employed	Premium
4	No	2000	Self-employed	Basic
5	No	5000	Employed	Economy
6	Yes	5100	Retired	Economy
7	Yes	3000	Employed	Premium

sort 

Continuous Descriptive Features - Example

ID	Insurance	Income	Employment	Customer
2	Yes	0	Unemployed	Premium
3	Yes	1000	Self-employed	Premium
4	No	2000	Self-employed	Basic
7	Yes	3000	Employed	Premium
1	Yes	3500	Employed	Basic
5	No	5000	Employed	Economy
6	Yes	5100	Retired	Economy

sort 

Continuous Descriptive Features - Example

ID	Insurance	Income	Employment	Customer
2	Yes	0	Unemployed	Premium
3	Yes	1000	Self-employed	Premium
4	No	2000	Self-employed	Basic
7	Yes	3000	Employed	Premium
1	Yes	3500	Employed	Basic
5	No	5000	Employed	Economy
6	Yes	5100	Retired	Economy

Thresholds: middle values of continuous feature in between changed target features

ID	Insurance	Income	Employment	Customer
2	Yes	0	Unemployed	Premium
3	Yes	1000	Self-employed	Premium
4	No	2000	Self-employed	Basic
7	Yes	3000	Employed	Premium
1	Yes	3500	Employed	Basic
5	No	5000	Employed	Economy
6	Yes	5100	Retired	Economy

Continuous Descriptive Features - Example

ID	Insurance	Income	Employment	Customer
2	Yes	0	Unemployed	Premium
3	Yes	1500	Self-employed	Premium
4	No	2500	Self-employed	Basic
7	Yes	3250	Employed	Premium
1	Yes	4250	Employed	Basic
5	No	5000	Employed	Economy
6	Yes	5100	Retired	Economy

Four candidate thresholds

Thresholds: middle values of continuous feature in between changed target features

Continuous Descriptive Features - Example

Threshold	Instances	Partition Entropy	Overall Entropy	Information Gain
≥ 1500	2, 3	0	1.0871	0.1981
	1, 4, 5, 6, 7	1.5219		
≥ 2500	2, 3, 4	0.9183	1.2507	0.306
	1, 5, 6, 7	1.5		
≥ 3250	2, 3, 4, 7	0.8113	0.8572	0.6995
	1, 5, 6	0.9183		
≥ 4250	1, 2, 3, 4, 7	0.9710	0.6935	0.8631
	5, 6	0		

Compute
as usual

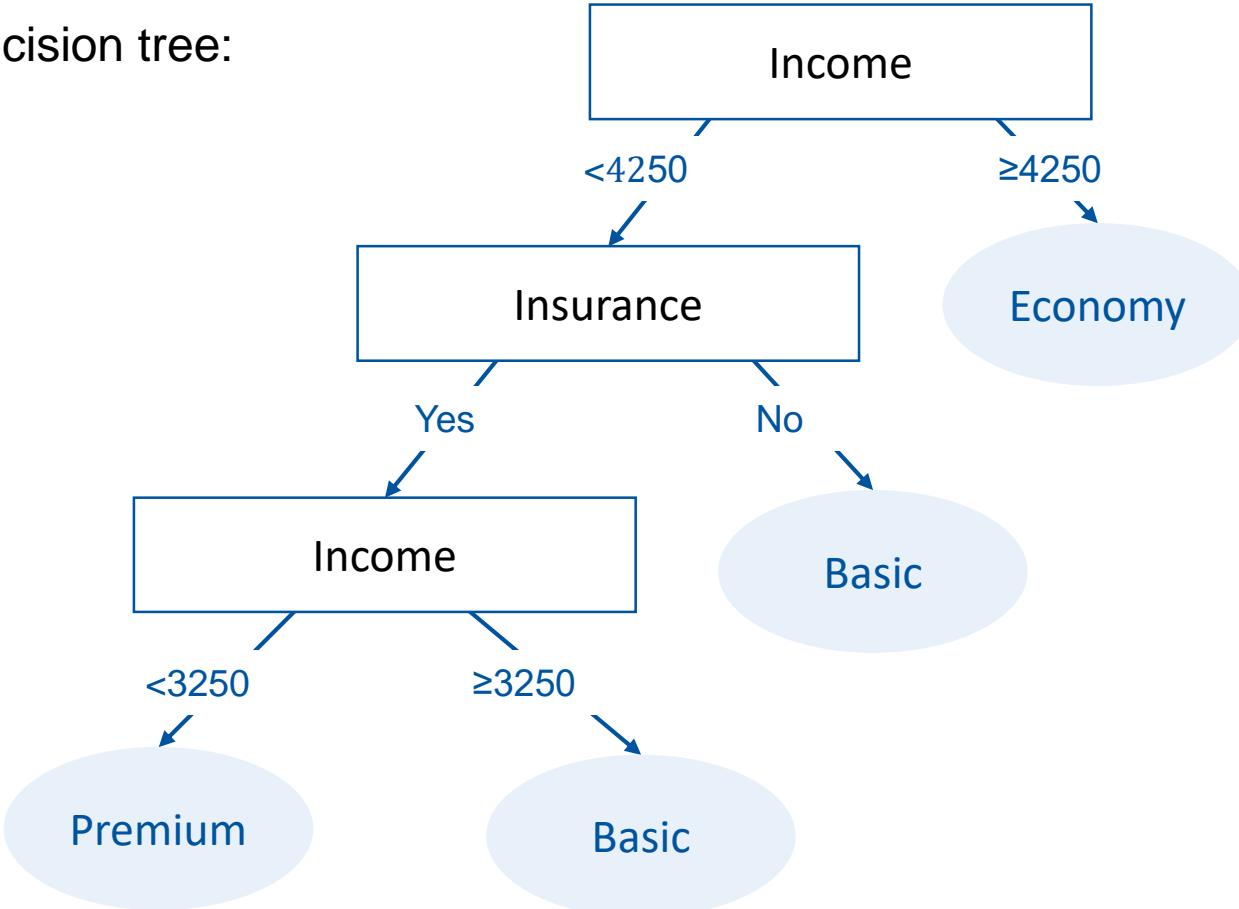
Continuous Descriptive Features - Example

Threshold	Instances	Partition Entropy	Overall Entropy	Information Gain
≥ 1500	2, 3	0	1.0871	0.1981
	1, 4, 5, 6, 7	1.5219		
≥ 2500	2, 3, 4	0.9183	1.2507	0.306
	1, 5, 6, 7	1.5		
≥ 3250	2, 3, 4, 7	0.8113	0.8572	0.6995
	1, 5, 6	0.9183		
≥ 4250	1, 2, 3, 4, 7	0.9710	0.6935	0.8631
	5, 6	0		

best

Continuous Descriptive Features - Example

Resulting decision tree:



The same feature can
now be used twice
along a path!

Continuous Target Features

- Goal: find descriptive features that ‘nicely’ partition the target feature axis
- Impurity = Variance within a partition
- We cannot use the target feature itself
- We ‘color the dots’ based on a selected descriptive feature

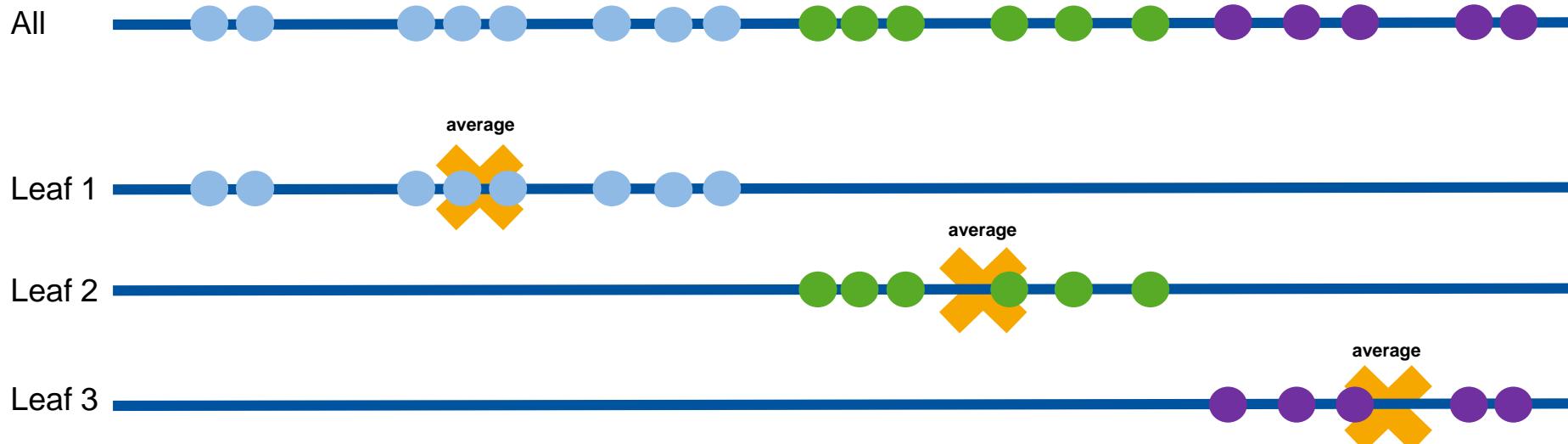


Continuous Target Features

Good Classification



- Three leaves (purple, green, blue show mapping based on descriptive feature)
- Impurity as measure of quality: variance within a leaf of the decision tree

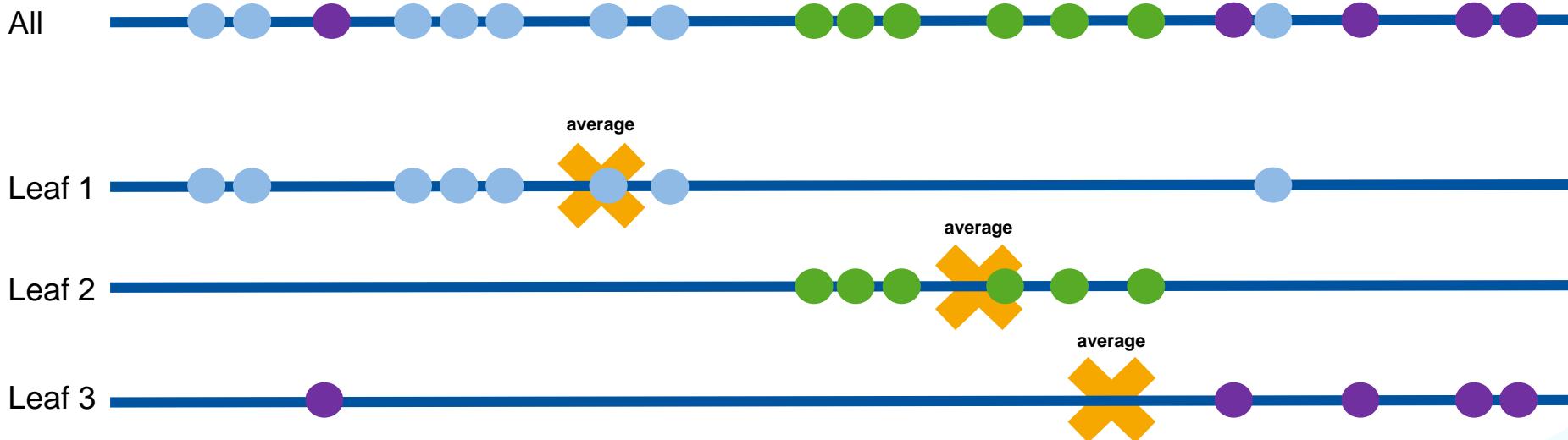


Continuous Target Features



Reasonable Classification

Variance within Leaf 1 and Leaf 3 increased with respect to the 'good classification'

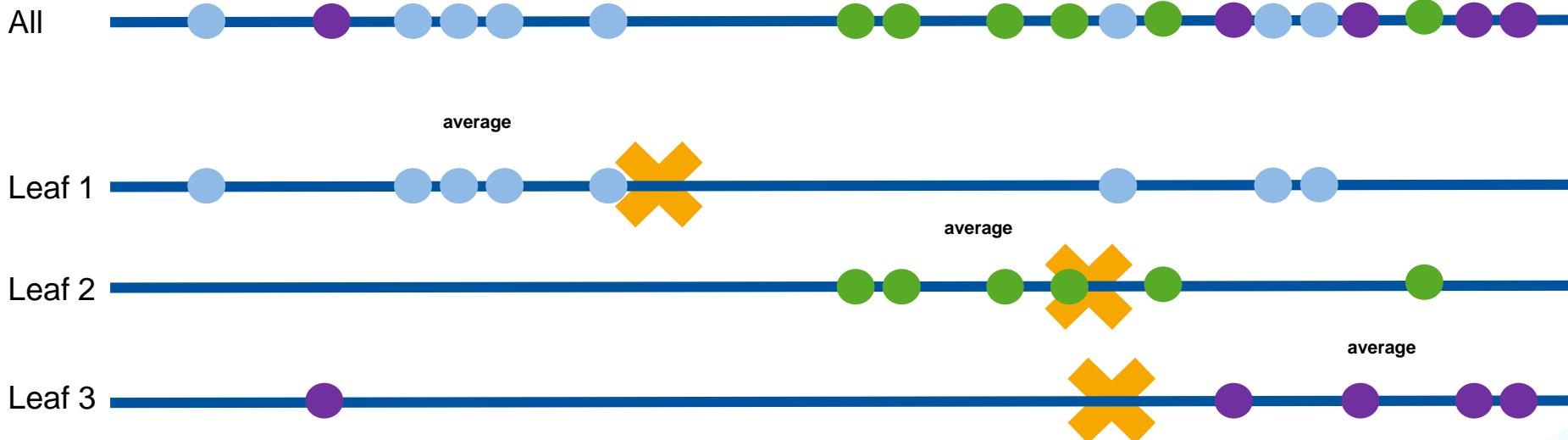


Continuous Target Features



Poor Classification

Variance within all leaves is high compared to the 'good classification'

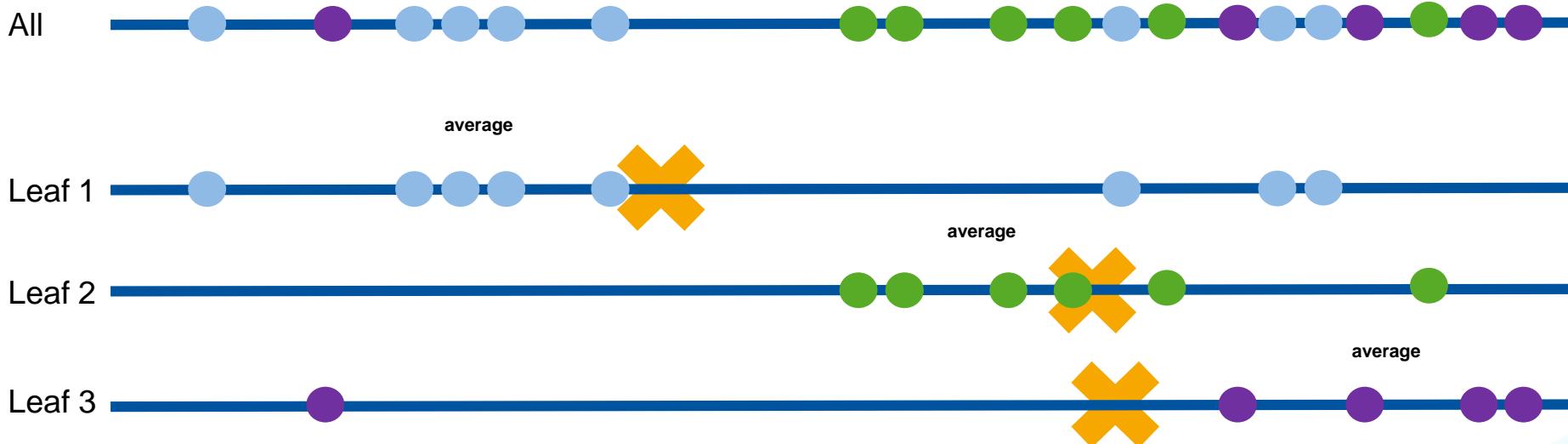


Impurity

Variance in a Node/Leaf

Number of instances
Target value of instance i
Mean of target values

$$Var(t) = \frac{\sum_{i=1}^N (t_i - \bar{t})^2}{N-1}$$



Adapting the ID3 Algorithm

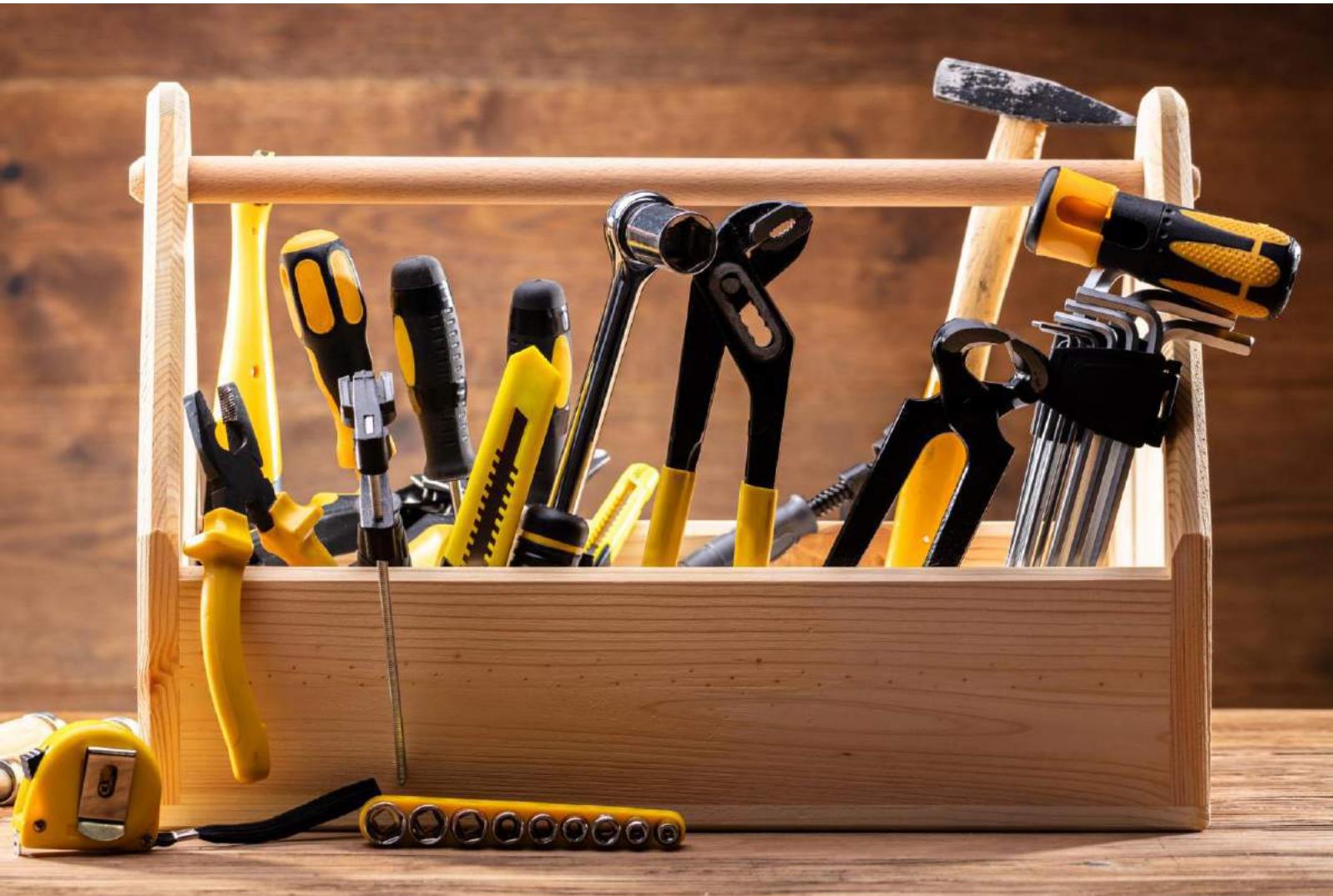
ID3 algorithm:

1. **if** all the instances in the dataset have the same classification
 - (a) **return** a decision tree with one leaf node with consensus value as a label
2. **else if** there are no features left
 - (a) **return** a decision tree with one leaf node with majority value as a label
3. **else if** the dataset is empty
 - (a) **return** a decision tree with one leaf node with majority parent value as a label
4. **else**
 - (a) pick a feature that lowers the weighted variance most within the subtrees
 - (b) once a feature is picked along a path from the root, it cannot be used again
 - (c) create subproblems based on the selected feature

Stopping criteria
(add pruning
strategies to
avoid overfitting)

Instead of
maximizing
information gain

Note that we presented a toolbox! (Not one specific algorithm.)



Many variations
are possible by
combining ideas

There is no best
solution, it all
depends on your
data

Performance on unseen test data is what counts



Avoid overfitting
the data!

Split data into
training and test
data

Topics such as
accuracy and
confusion matrix
be discussed later

Decision Trees - Conclusion

- Supervised learning aims to explain the target feature in terms of descriptive features
- Decision trees are easy to understand and interpret
- Focus on categorical variables but extensions to continuous data are possible
- Many variations based on the basic ID3 algorithm
 - Pruning
 - Ensembles
 - Information gain definitions
 - ...

Next: Clustering techniques

Elements of Machine Learning & Data Science

Clustering

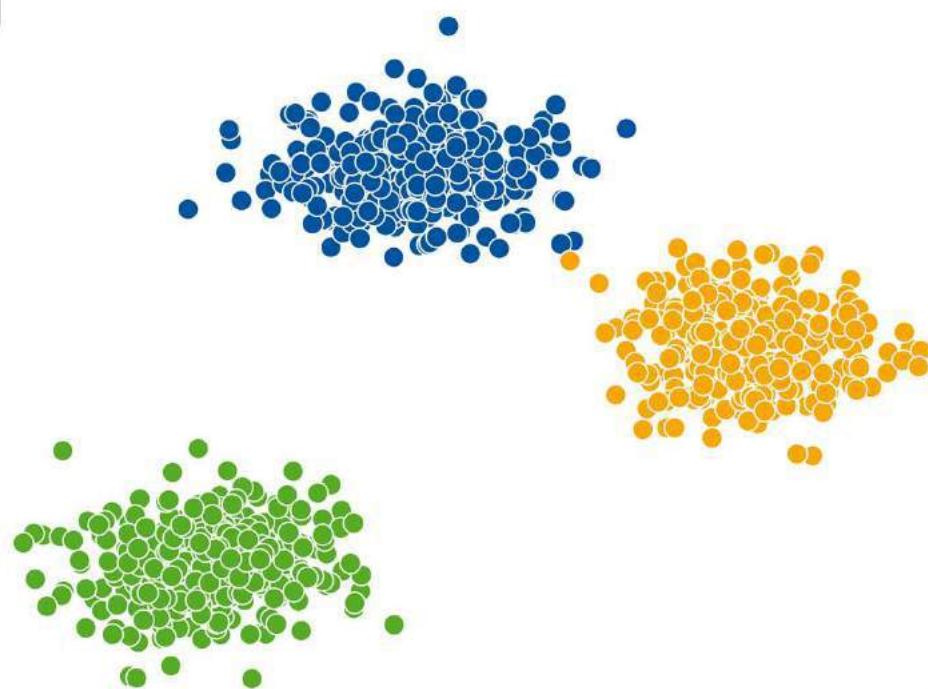
Lecture 8

Prof. Wil van der Aalst

Marco Pegoraro, M.Sc.
Christopher Schwanen, M.Sc.
Tsunghao Huang, M.Sc.

Clustering

1. Introduction to Unsupervised Learning
2. Introduction to Clustering
3. Similarity and Dissimilarity
4. K-means and K-medoids
5. Agglomerative Clustering
6. Density-Based (DBSCAN)
7. Closing



Unsupervised Learning

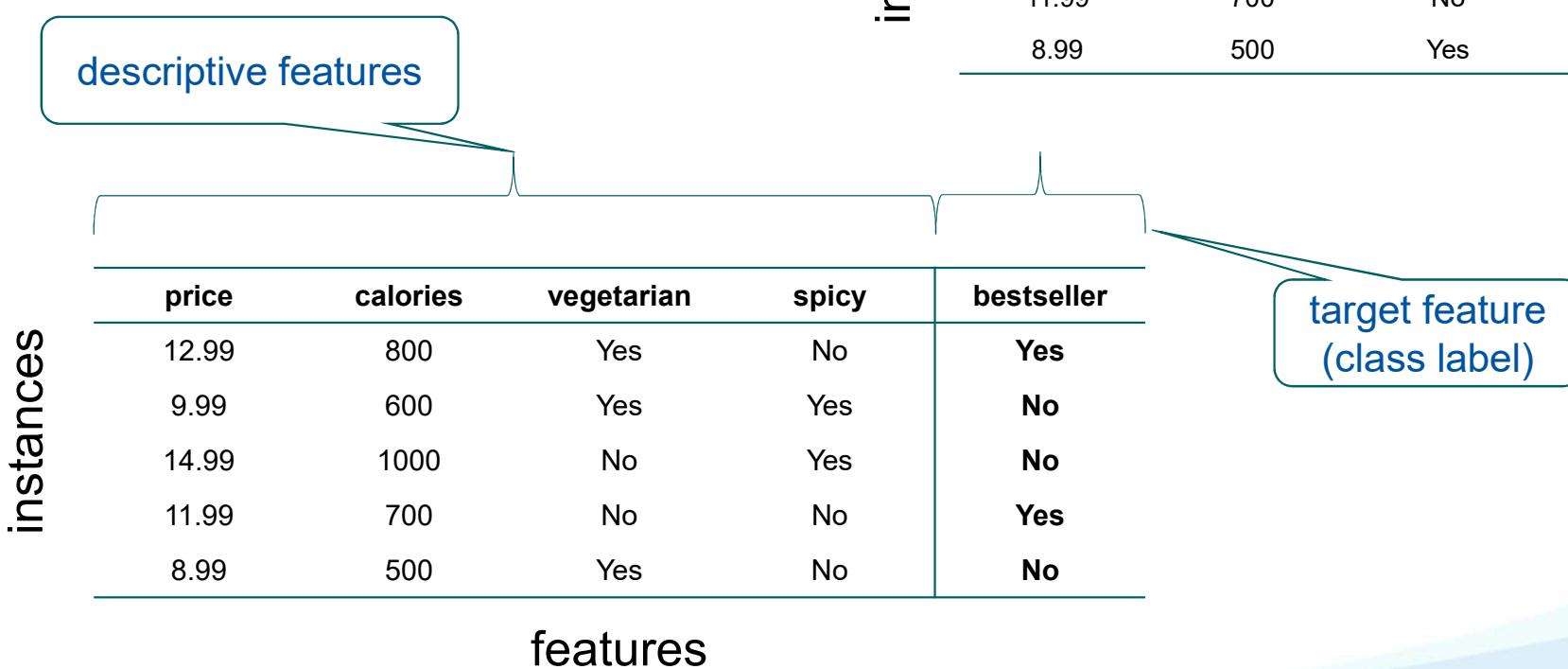
Unsupervised Learning

- Obtain a **model** that represents the data...
- ...without a target **variable** or **label**

	features				
instances	price	calories	vegetarian	spicy	bestseller
	12.99	800	Yes	No	Yes
	9.99	600	Yes	Yes	No
	14.99	1000	No	Yes	No
	11.99	700	No	No	Yes
	8.99	500	Yes	No	No

Unsupervised Learning

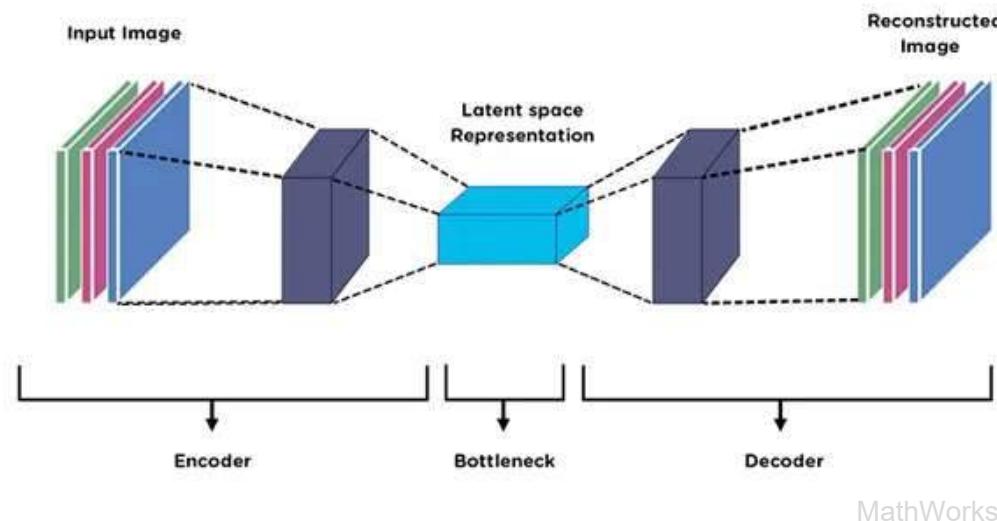
- Recall from intro lecture:
labeled vs unlabeled



Unsupervised Learning

- Obtain a **model** that represents the data...
- ...without a target **variable** or **label**
- Why?
 - When a target feature is hard to identify
 - Maybe we are not sure something is even there!
 - To search for **patterns** in the data
 - To learn a **representation**

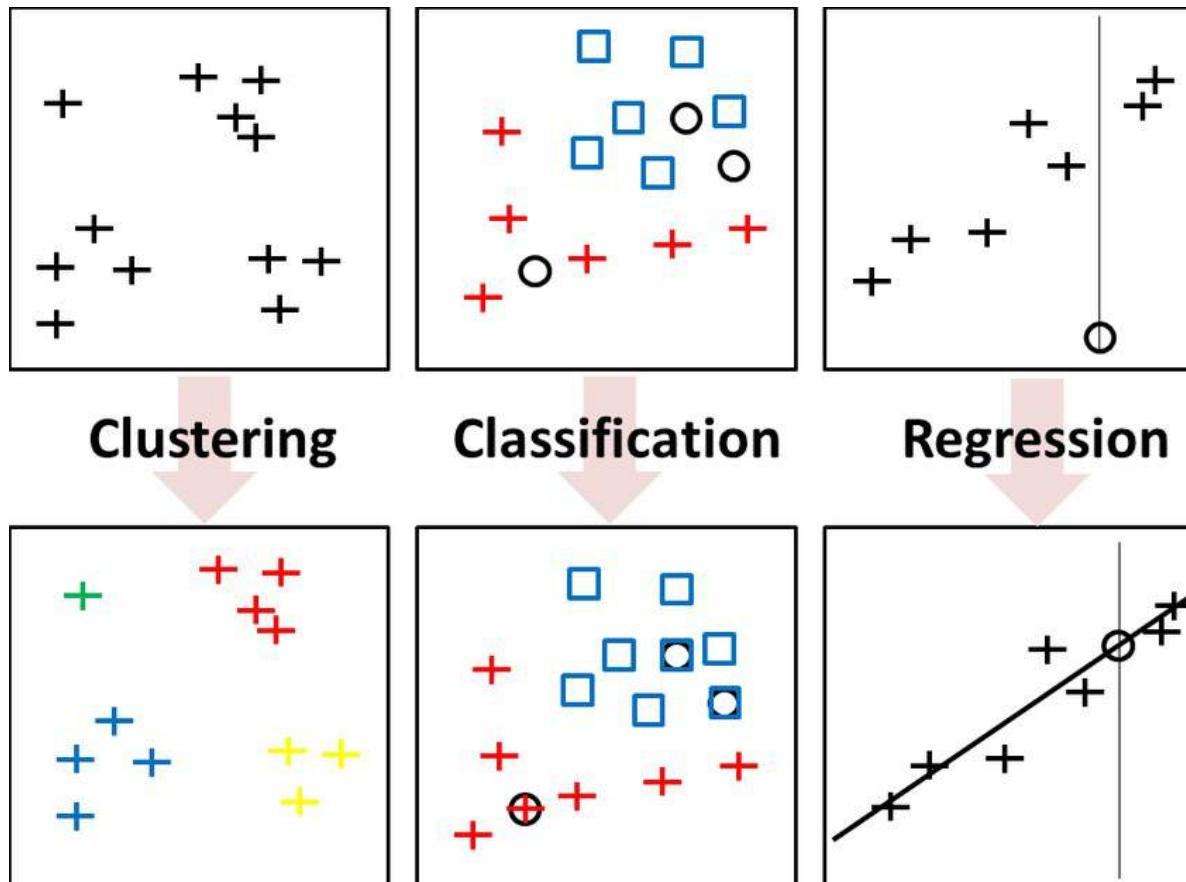
A complex example: autoencoding



MathWorks

Autoencoding: automatically finding a semantics-rich representation of the data in a latent vector space (with the desired dimensionality)

Unsupervised vs. Supervised Learning: Models



In unsupervised learning, the model typically explains **relationships** between instances

In supervised learning, the model typically explains the **values** of one or more features

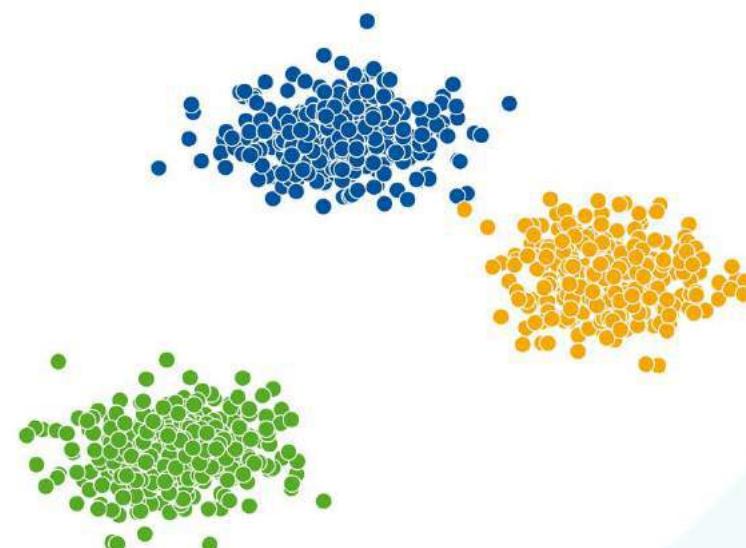
Unsupervised Learning

- Obtain a **model** that represents the data...
- ...without a target **variable** or **label**
- Challenges:
 - The **ground truth** might be hard to identify
 - This means that designing an **evaluation** can be very hard

Clustering

Clustering: motivation

- Find clusters (groups of instances) such that:
 - Instances within the cluster are **similar**
 - Instances in different clusters are **dissimilar**
- Applications:
 - To find **unexpected groups**
 - To do **data preprocessing**
e.g., discover (process) models for each cluster
 - Unlabeled data is cheaper than labeled data!



Clustering Use Cases

Spotify



User	Song 1	Song 2	Song 3	...
User 1	4	0	5	...
User 2	0	1	0	...
User 3	3	2	9	...
...

- 456 million active listeners
- 195 million premium subscribers
- Over 80 million songs

(As of January 2023)

Clustering Use Cases

Amazon



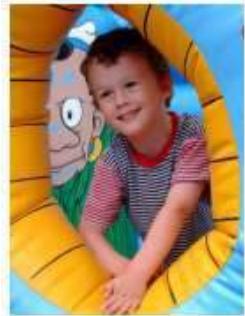
Customer	Prod 1	Prod 2	Prod 3	...
Customer 1	1	0	0	...
Customer 2	0	0	1	...
Customer 3	1	1	0	...
...

- 300 million active users
- Over 2 million third-party seller businesses
- Around 350 million items on the marketplace

(As of January 2023)

Clustering Use Cases

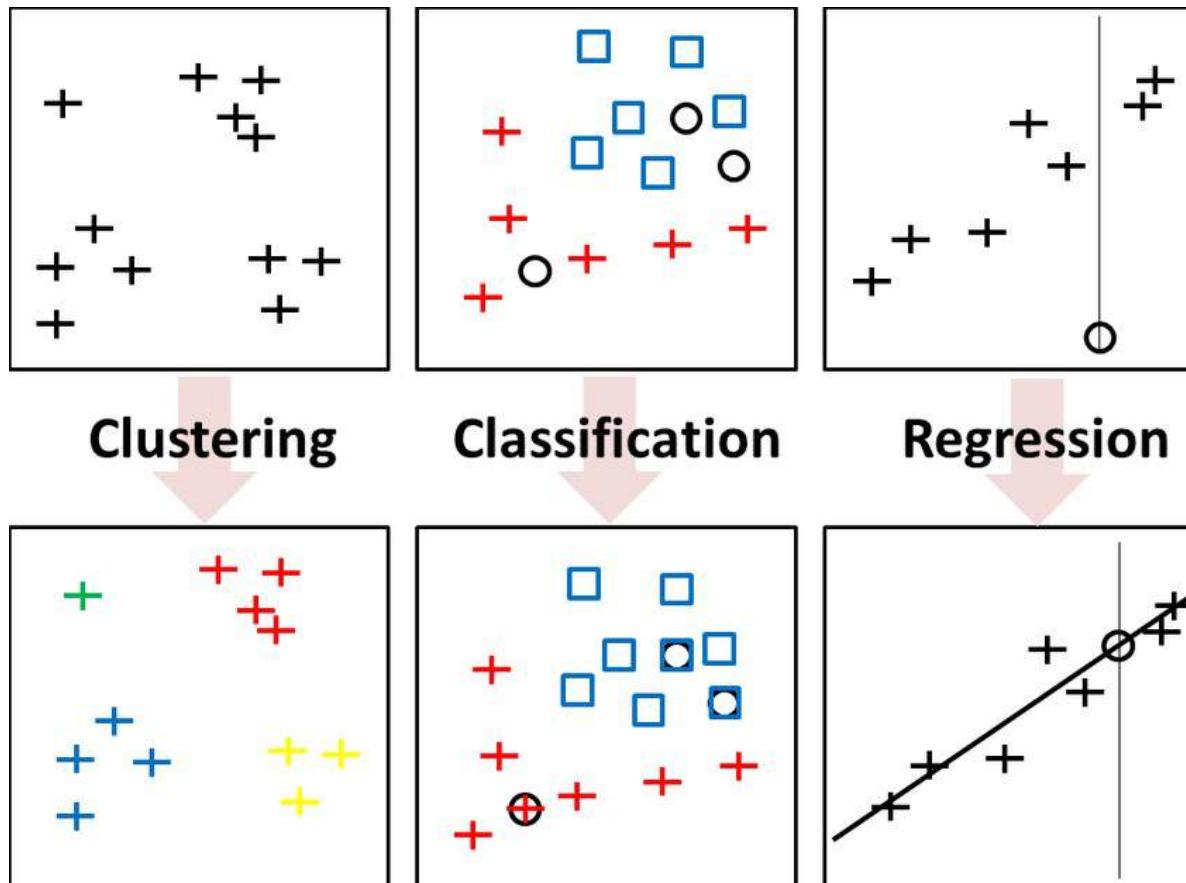
Image Segmentation



C. Bishop, 2006

- Goal: finding regions in a picture with homogeneous appearance
- Applications in computer graphics, e.g. subject detection, edge detection

Clustering, classification and regression



Do not mix up
classification with
clustering!

When doing
classification, we
have a [training set](#) of
correctly classified
instances.

When doing
clustering, we do not
(usually)

Clustering Approaches

- Partitioning methods (split into subsets)
 - Centroid-based (e.g., **k-means**)
 - Medoids-based (e.g., **k-medoids**)
- Hierarchical methods (build dendrogram)
 - **Agglomerative (bottom-up)**
 - Divisive (top-down)
- Density-based methods (e.g., **DBSCAN**)
- Grid-based methods

Similarity and Dissimilarity

Similarity / Dissimilarity

Goal: instances within a cluster are similar, instances in different clusters are dissimilar

$$\mathbf{x}_i = (x_{i1}, x_{i2}, \dots, x_{iD}) \iff \mathbf{x}_j = (x_{j1}, x_{j2}, \dots, x_{jD})$$

Similarity (or proximity)

- Numerical measure of how **alike** two instances are
- **Higher** when instances are more alike
- Often falls in the range [0, 1]

Dissimilarity (or distance)

- Numerical measure of how **different** two instances are
- **Lower** when instances are more alike
- Minimum dissimilarity is often 0
- Upper limit varies

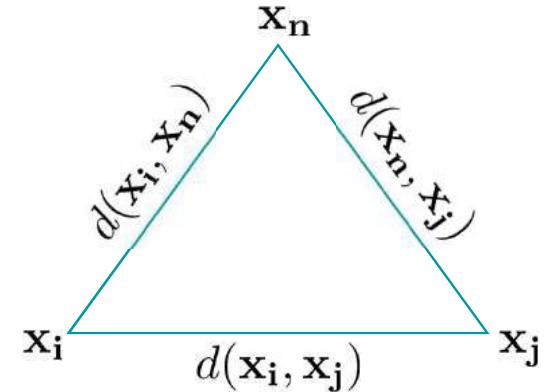


Metric Space Characteristics

$$\mathbf{x}_i = (x_{i1}, x_{i2}, \dots, x_{iD}) \iff \mathbf{x}_j = (x_{j1}, x_{j2}, \dots, x_{jD})$$

- **Non-negativity** - distance is a non-negative number $d(\mathbf{x}_i, \mathbf{x}_j) \geq 0$
- **Identity of indiscernibles** - the distance of an object to itself is 0 $d(\mathbf{x}_i, \mathbf{x}_i) = 0$
- **Symmetry** - distance is a symmetric function $d(\mathbf{x}_i, \mathbf{x}_j) = d(\mathbf{x}_j, \mathbf{x}_i)$
- **Triangle inequality** - going directly from object to object in space is no more than going through any other object

$$d(\mathbf{x}_i, \mathbf{x}_j) \leq d(\mathbf{x}_i, \mathbf{x}_n) + d(\mathbf{x}_n, \mathbf{x}_j)$$



Examples of Similarity / Dissimilarity Measures

- Binary/Nominal features:
 - Simple matching coefficient
 - Jaccard similarity coefficient
- Continuous features:
 - Euclidean distance $d(\mathbf{x}_i, \mathbf{x}_j) = \sqrt{(x_{i1} - x_{j1})^2 + (x_{i2} - x_{j2})^2 + \dots + (x_{iD} - x_{jD})^2}$
 - Manhattan distance $d(\mathbf{x}_i, \mathbf{x}_j) = |x_{i1} - x_{j1}| + |x_{i2} - x_{j2}| + \dots + |x_{iD} - x_{jD}|$
 - Minkowski distance (generalization)
 - Cosine similarity (non-metric measure)

Binary symmetric: Simple matching distance

- Assumes no clear asymmetry between group 0 and 1
- Example: two right-handed persons are as similar as two left-handed persons

	$y = 1$	$y = 0$
$x = 1$	a	b
$x = 0$	c	d

$$SMD(x, y) = \frac{b + c}{a + b + c + d}$$

- a = number of attributes where x and y are both 1
- b = number of attributes where x is 1 and y is 0
- c = number of attributes where x is 0 and y is 1
- d = number of attributes where x and y are both 0

Binary asymmetric: Jaccard distance

- Asymmetry between group 0 and 1
- Example: two persons that won a Turing Award are more similar than two people without a Turing Award

	$y = 1$	$y = 0$
$x = 1$	a	b
$x = 0$	c	d

$$d_J(i, j) = \frac{b + c}{a + b + c}$$

- a = number of attributes where x and y are both 1
- b = number of attributes where x is 1 and y is 0
- c = number of attributes where x is 0 and y is 1
- d = number of attributes where x and y are both 0

Nominal: Simple matching distance

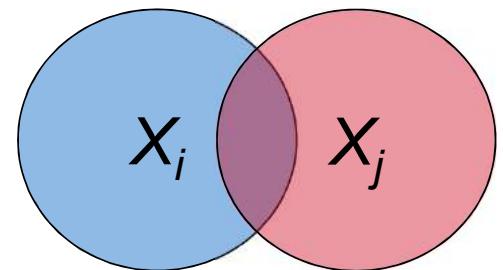
- mm : total number of variables where objects i and j mismatch
- p : total number of variables

$$SMD(i, j) = \frac{mm}{p}$$

- Simple matching coefficient: $SMC = 1 - SMD$

Nominal: Jaccard Similarity Coefficient

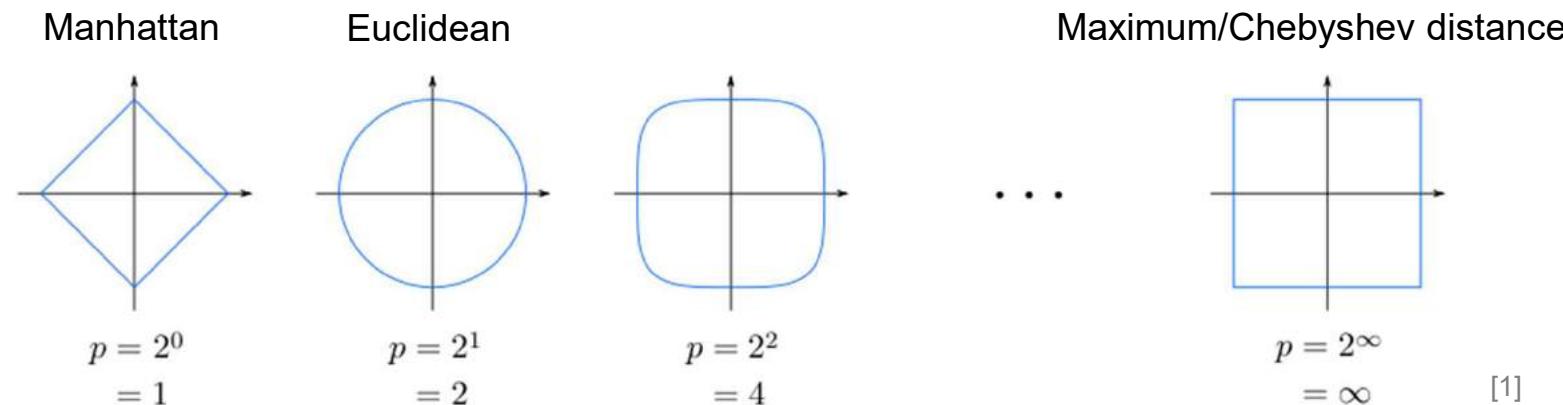
- Assumes instances are represented by sets
- Jaccard similarity between two sets X_i and X_j
$$J(X_i, X_j) = \frac{|X_i \cap X_j|}{|X_i \cup X_j|}$$
- Jaccard distance between two sets X_i and X_j
$$d_J(X_i, X_j) = \frac{|X_i \cup X_j| - |X_i \cap X_j|}{|X_i \cup X_j|} = 1 - J(X_i, X_j)$$
- Jaccard distance is a metric, i.e., distance is non-negative, distance to itself is zero, symmetric, and satisfies the triangle inequality
- Used for comparing item sets (e.g., products ordered, words appearing in documents, courses taken, and songs played)



Minkowski Distance (Metric)

Generalization of Manhattan and Euclidean distance to any natural dimension $p \geq 1$ (also called L^p norm)

$$d(\mathbf{x_i}, \mathbf{x_j}) = \sqrt[p]{|x_{i1} - x_{j1}|^p + |x_{i2} - x_{j2}|^p + \cdots + |x_{iD} - x_{jD}|^p}$$



Minkowski Distance – Examples

$$d(\mathbf{x}_i, \mathbf{x}_j) = \sqrt[p]{|x_{i1} - x_{j1}|^p + |x_{i2} - x_{j2}|^p + \dots + |x_{iD} - x_{jD}|^p}$$

Manhattan distance $p = 1$

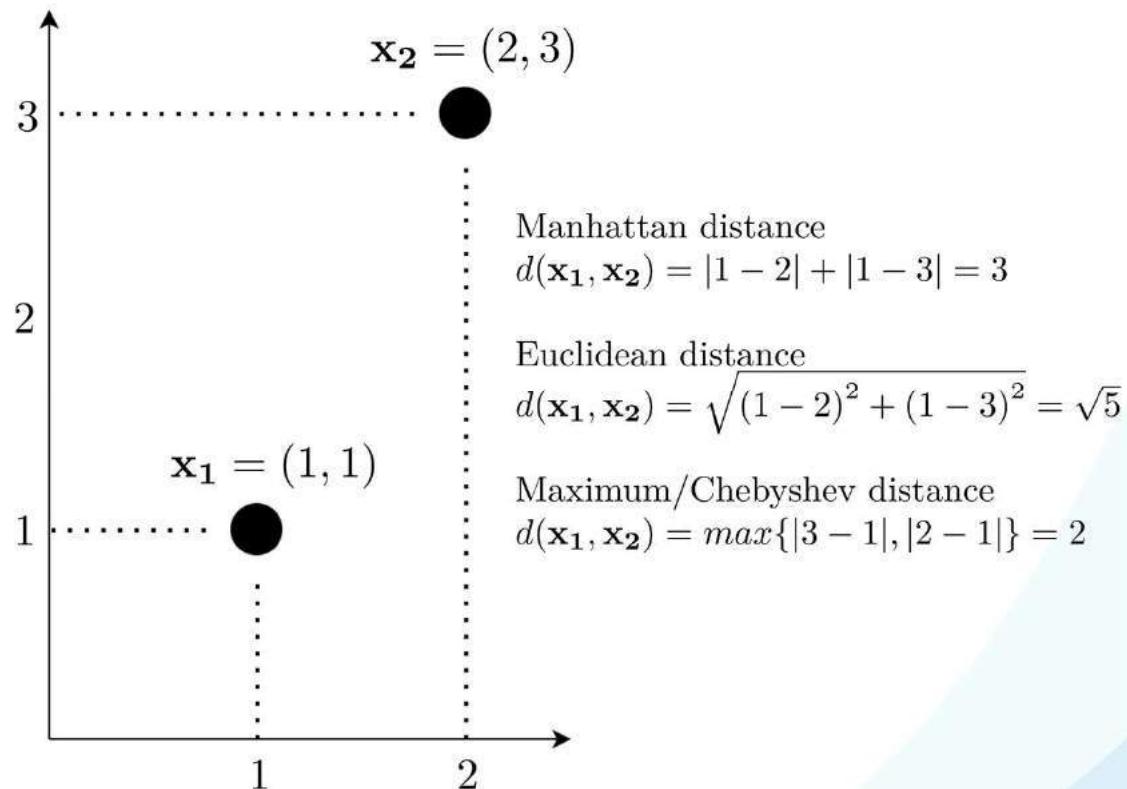
$$d(\mathbf{x}_i, \mathbf{x}_j) = |x_{i1} - x_{j1}| + |x_{i2} - x_{j2}| + \dots + |x_{iD} - x_{jD}|$$

Euclidean distance $p = 2$

$$d(\mathbf{x}_i, \mathbf{x}_j) = \sqrt{(x_{i1} - x_{j1})^2 + (x_{i2} - x_{j2})^2 + \dots + (x_{iD} - x_{jD})^2}$$

Maximum/Chebyshev distance $p \rightarrow \infty$

$$d(\mathbf{x}_i, \mathbf{x}_j) = \lim_{p \rightarrow \infty} \left(\sum_{d=1}^D |x_{id} - x_{jd}|^p \right)^{\frac{1}{p}} = \max_{d \in \{1, \dots, D\}} |x_{id} - x_{jd}|$$



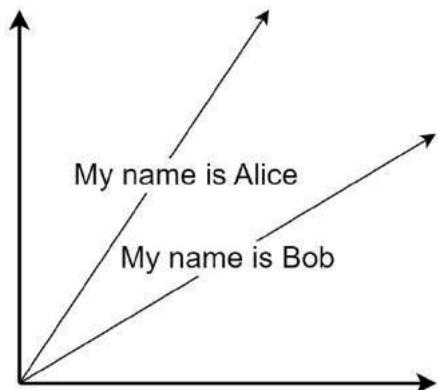
Similarity and Dissimilarity

Cosine Similarity (Non-Metric)

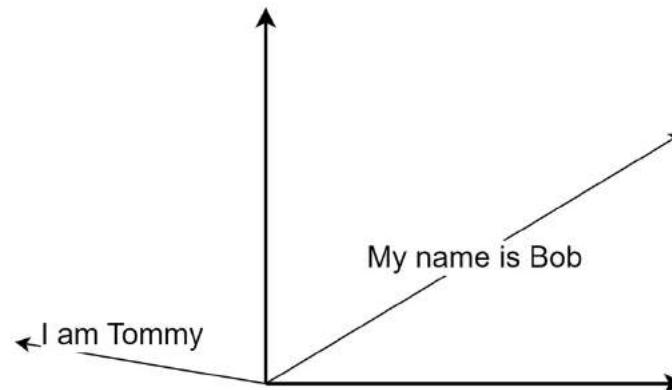
- Cosine similarity between two vectors \mathbf{v} and \mathbf{w}

$$S_C(\mathbf{x}_i, \mathbf{x}_j) = \cos(\theta) = \frac{\mathbf{x}_i \cdot \mathbf{x}_j}{\|\mathbf{x}_i\| \|\mathbf{x}_j\|}$$

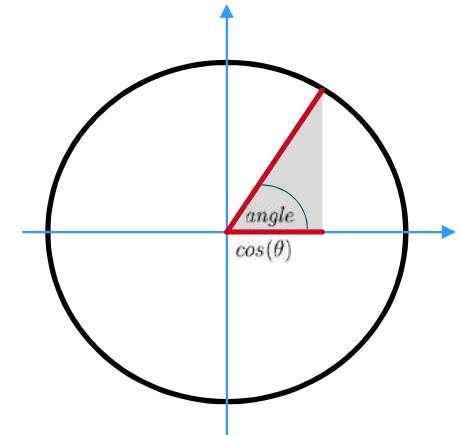
- Used for **sparse data** (focus on angle rather than distance)
- Often used for comparing representations of **textual data**



Similar vectors
Positive, approaches 1.0



Different vectors
Negative, approaches -1.0



angle	$\cos(\theta)$
0°	1.0000
45°	0.7071
90°	0.0000
135°	-0.7071
180°	-1.0000
270°	0.0000
360°	1.0000

Mixing Different Types of Features

- Commonly used approach – **normalize** all features ranges to [0, 1]
- One can give different weights to different features (i.e., distances are not the same in all dimensions)
- One can exclude features, because they would lead to obvious clusters (providing no insights)

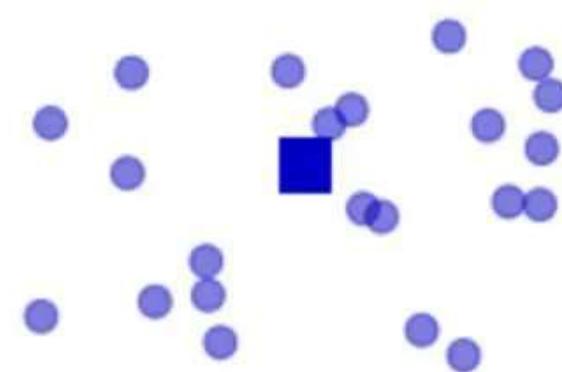
K-means and K-medoids

K-means – Idea

- Let's start with very strong assumptions
 - And therefore, a simpler problem
- What if we could represent a cluster with a **single point in space?**
- Then, grouping the instances in clusters would be very easy:
- Given an instance, find the closest representative, and assign the corresponding cluster

K-means – Idea

- Let's consider the **opposite problem**:
From the set of instances that we assume to **belong to a cluster**, how do we find this representative?
- A natural choice is to pick the point at the **geometric center** of the instances
- Or, equivalently, the center of the smallest sphere that contains all the instances
 - (given a certain distance metric)
- Also easy!
- Let's call this representative a **centroid**



J. Rogel-Salazar

K-means – Idea

- But then...
 - Finding the instances in a cluster given its centroid is easy
 - Finding a centroid given the instances in its cluster is also easy
 - But we have neither!
 - A chicken and egg problem!

K-means – Idea

- But then...
 - Finding the instances in a cluster given its centroid is easy
 - Finding a centroid given the instances in its cluster is also easy
 - But we have neither!
 - A chicken and egg problem
- Solution: we start with random points in space as centroids, and we iteratively refine

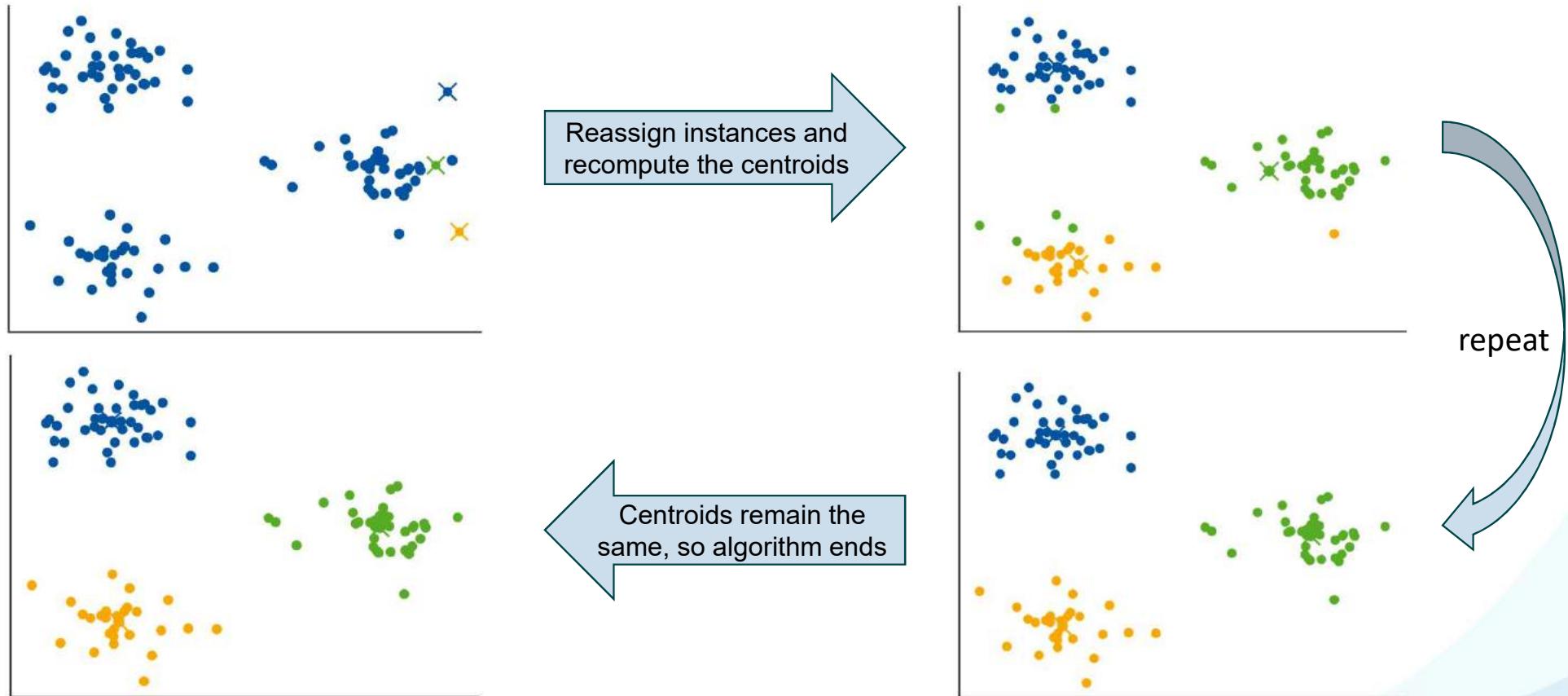
K-means

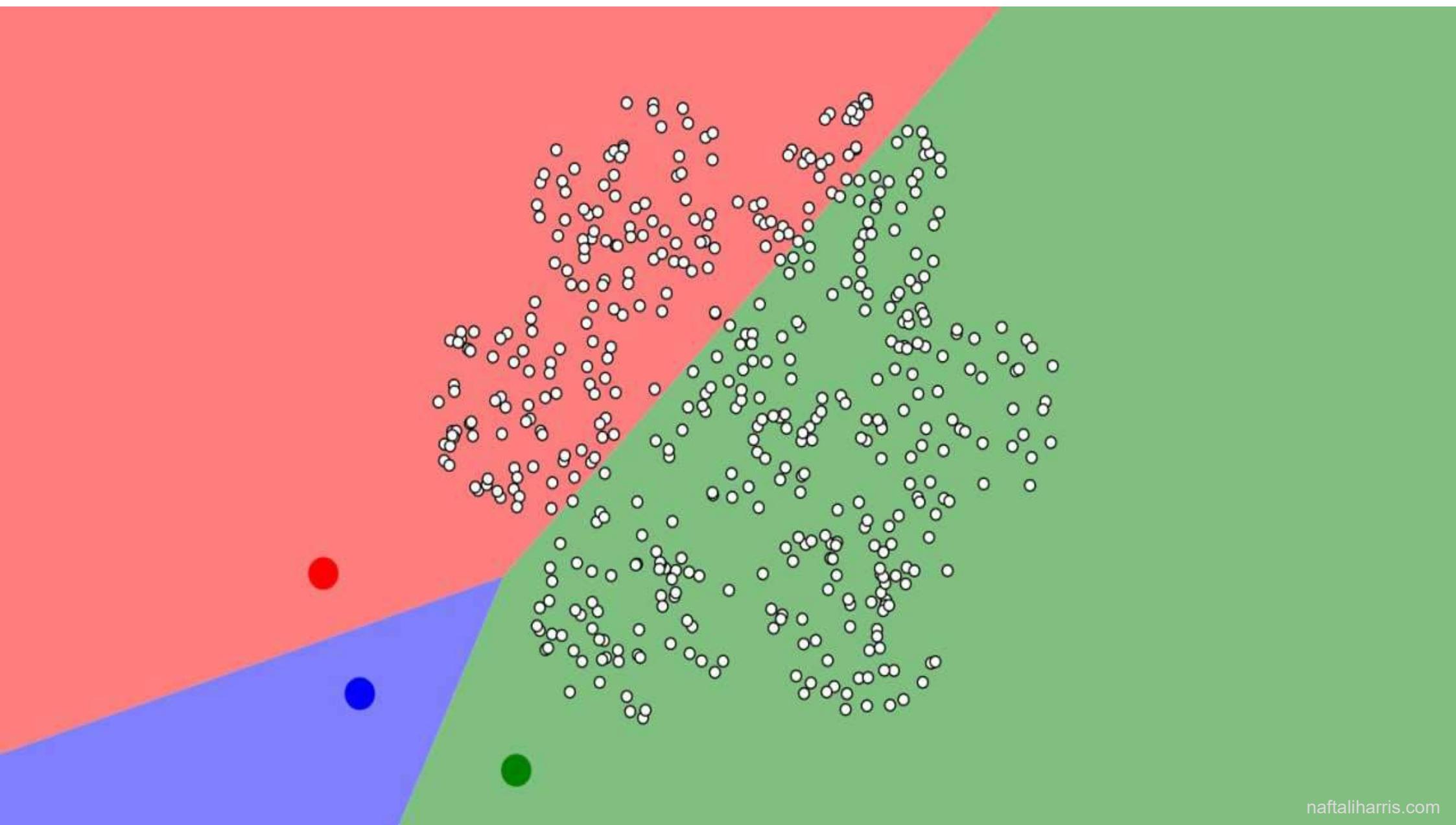
- Algorithm for clustering / partitioning data
- Each cluster's center (the **centroid**) is represented by the mean value of the instances (points) in the cluster
- Simple and fast to compute

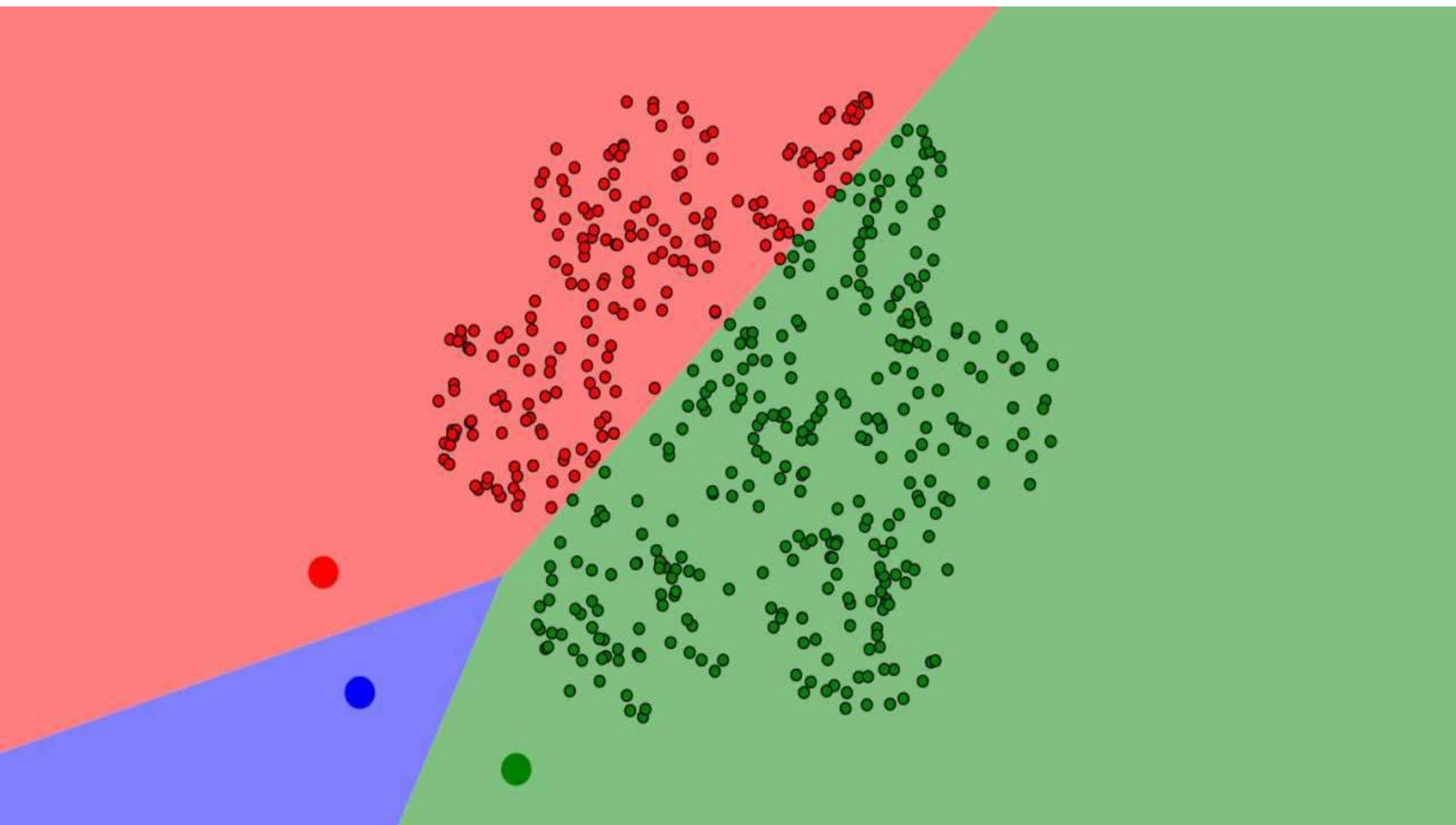
K-means algorithm:

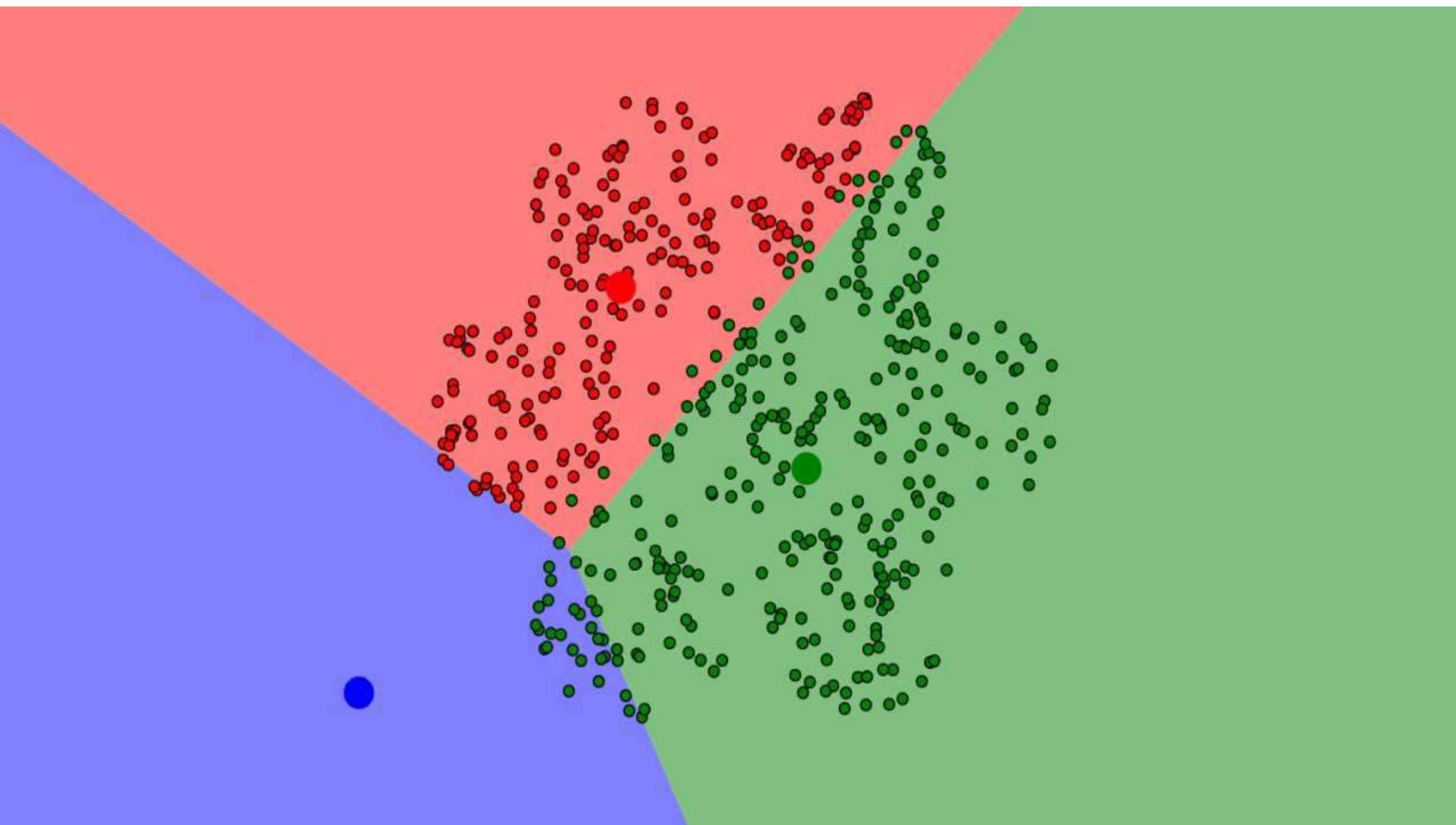
1. randomly choose k instances from the dataset \mathcal{X} as the initial cluster centers
2. **repeat until no change**
 - (a) reassign each instance to the cluster with the closest centroid
 - (b) recompute the centroid \mathbf{c}_i for each cluster \mathcal{C}_i for $i = 1, \dots, k$

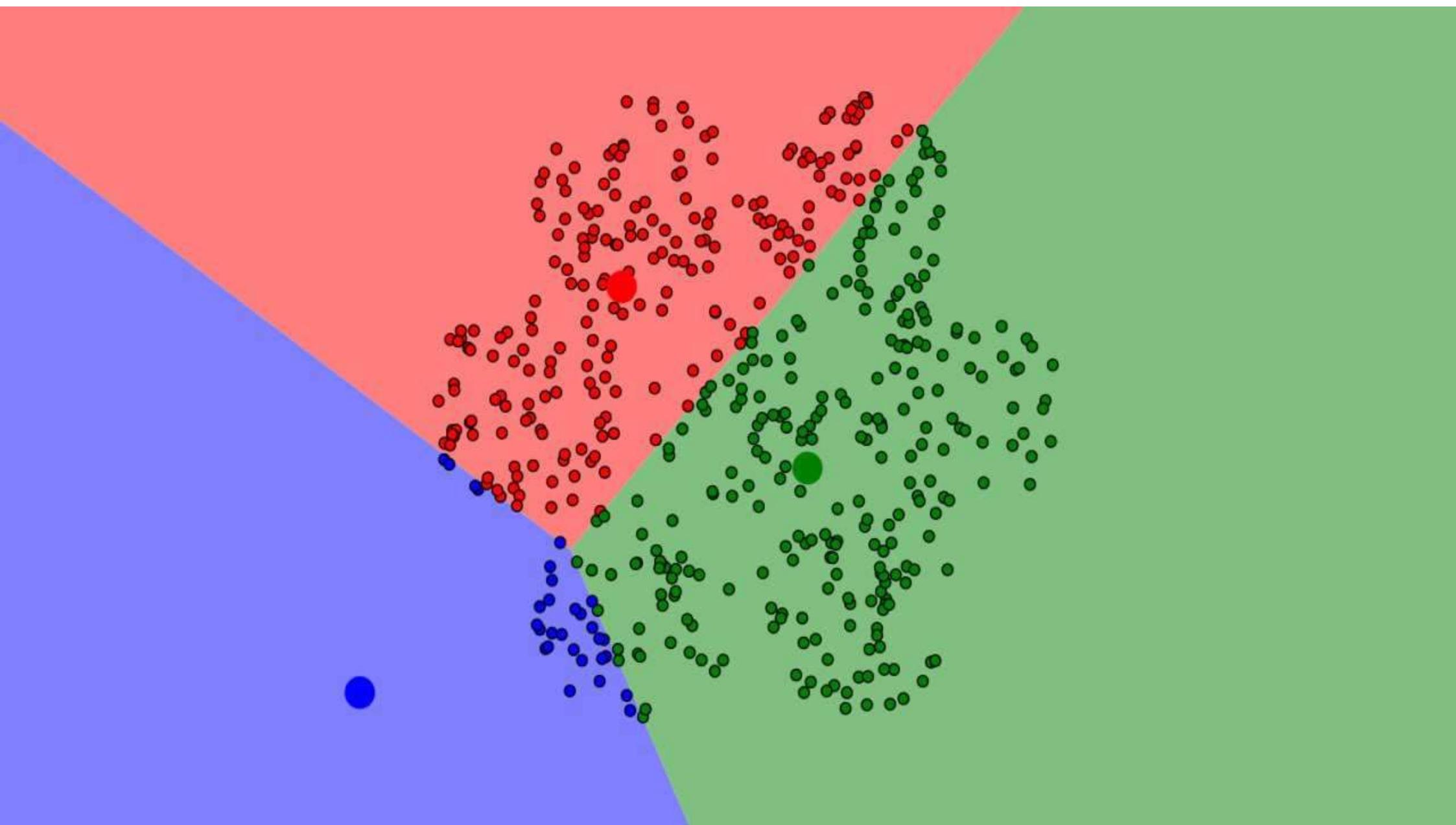
K-means – Example

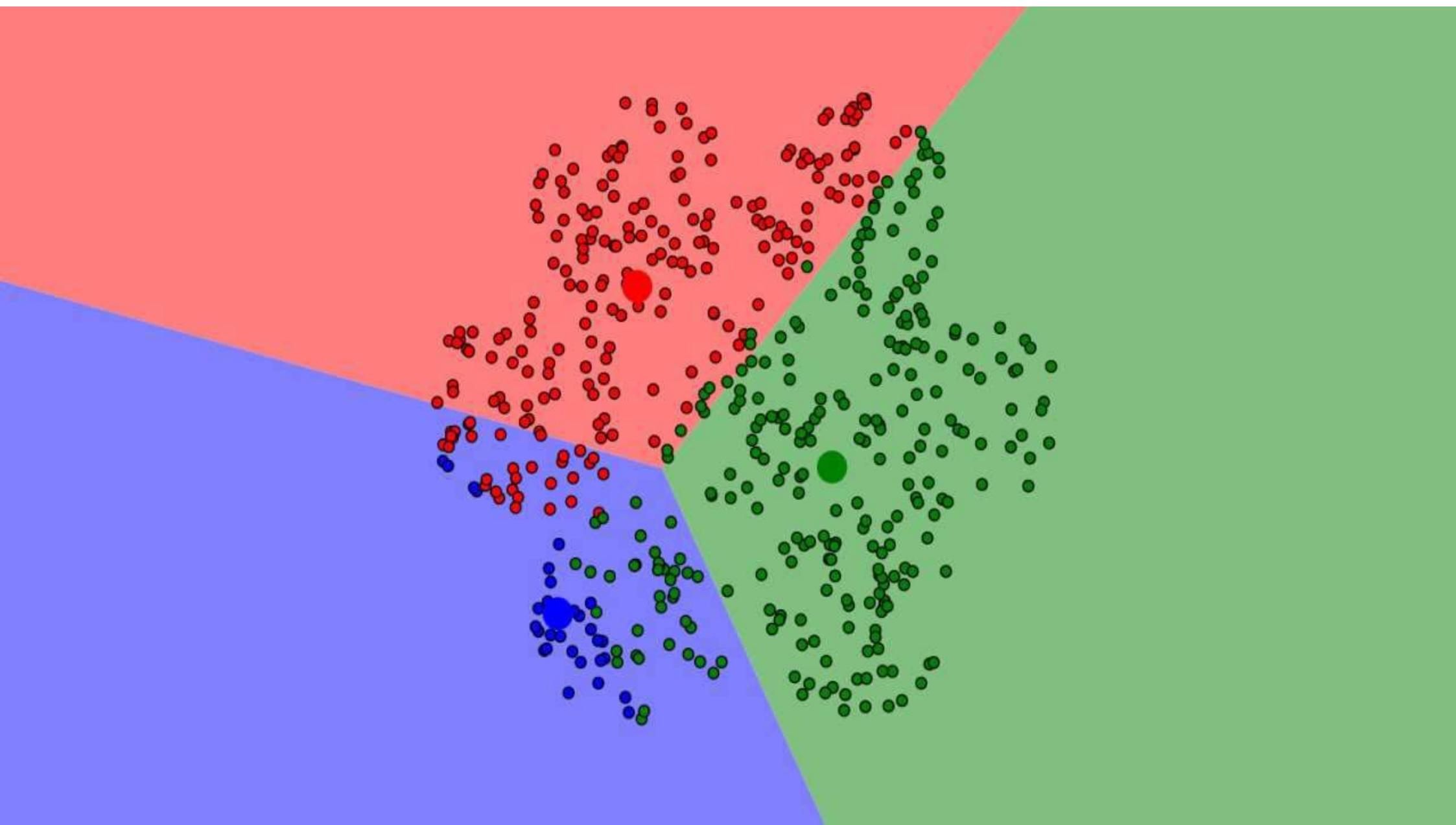


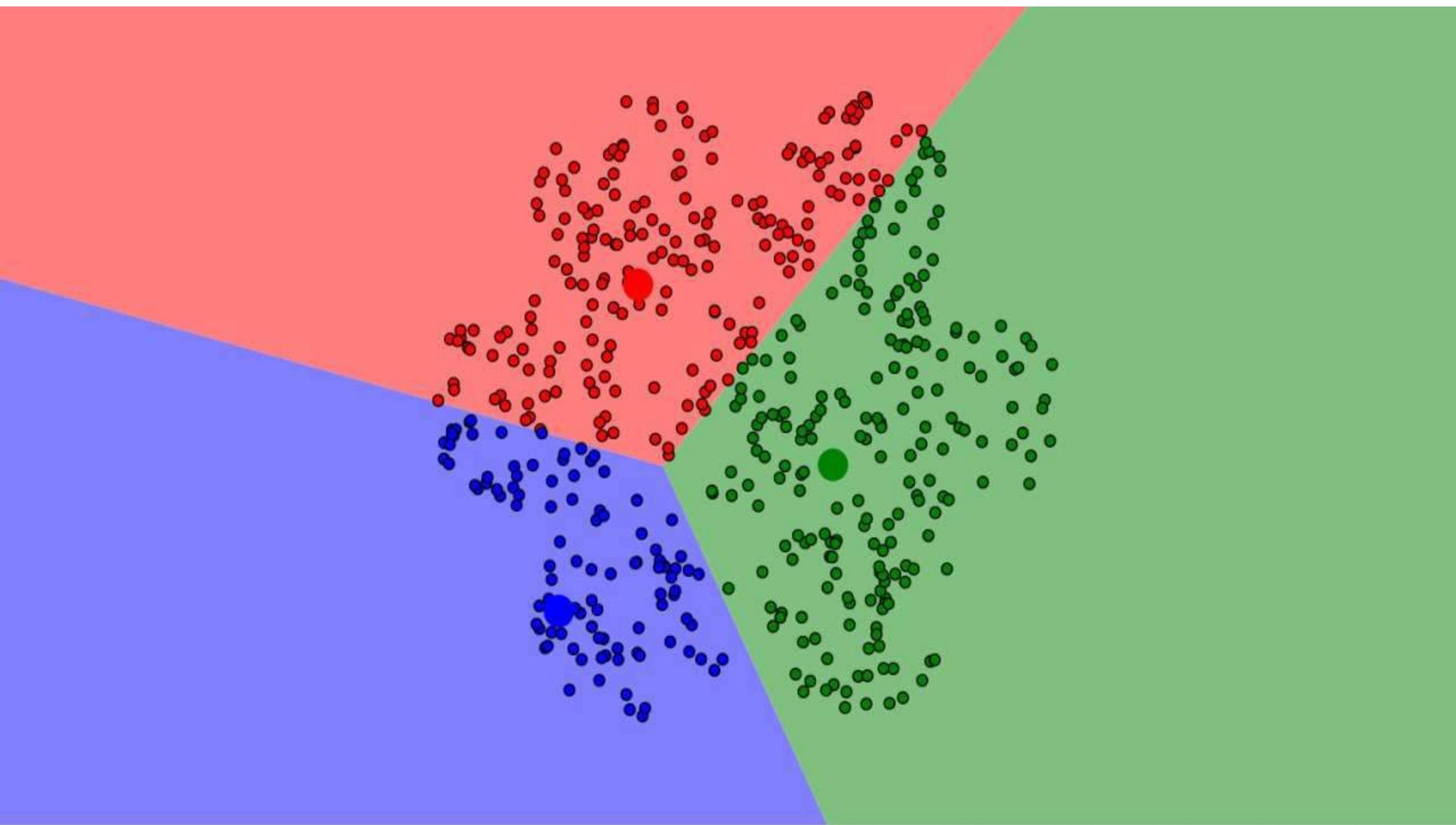


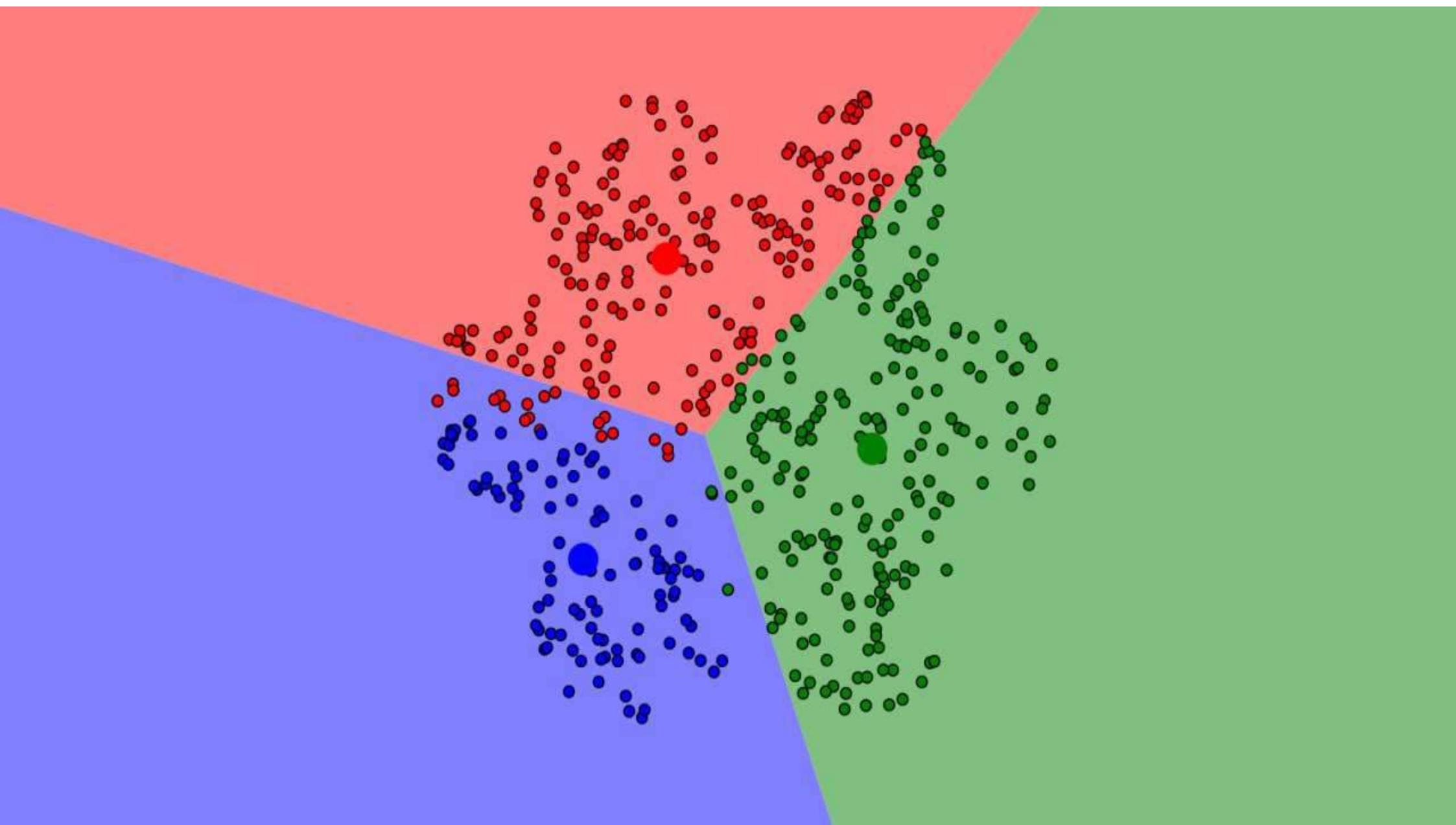


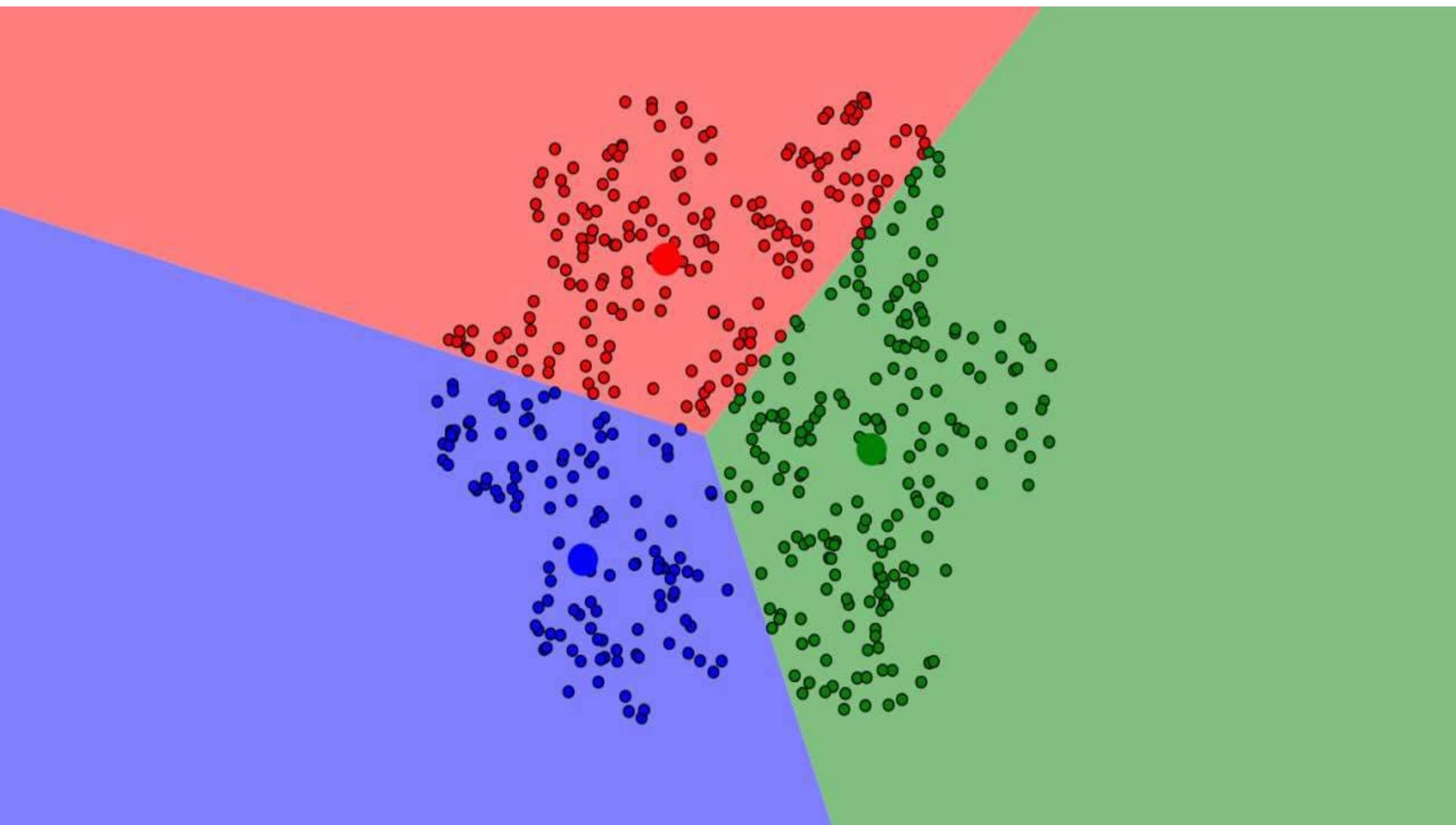


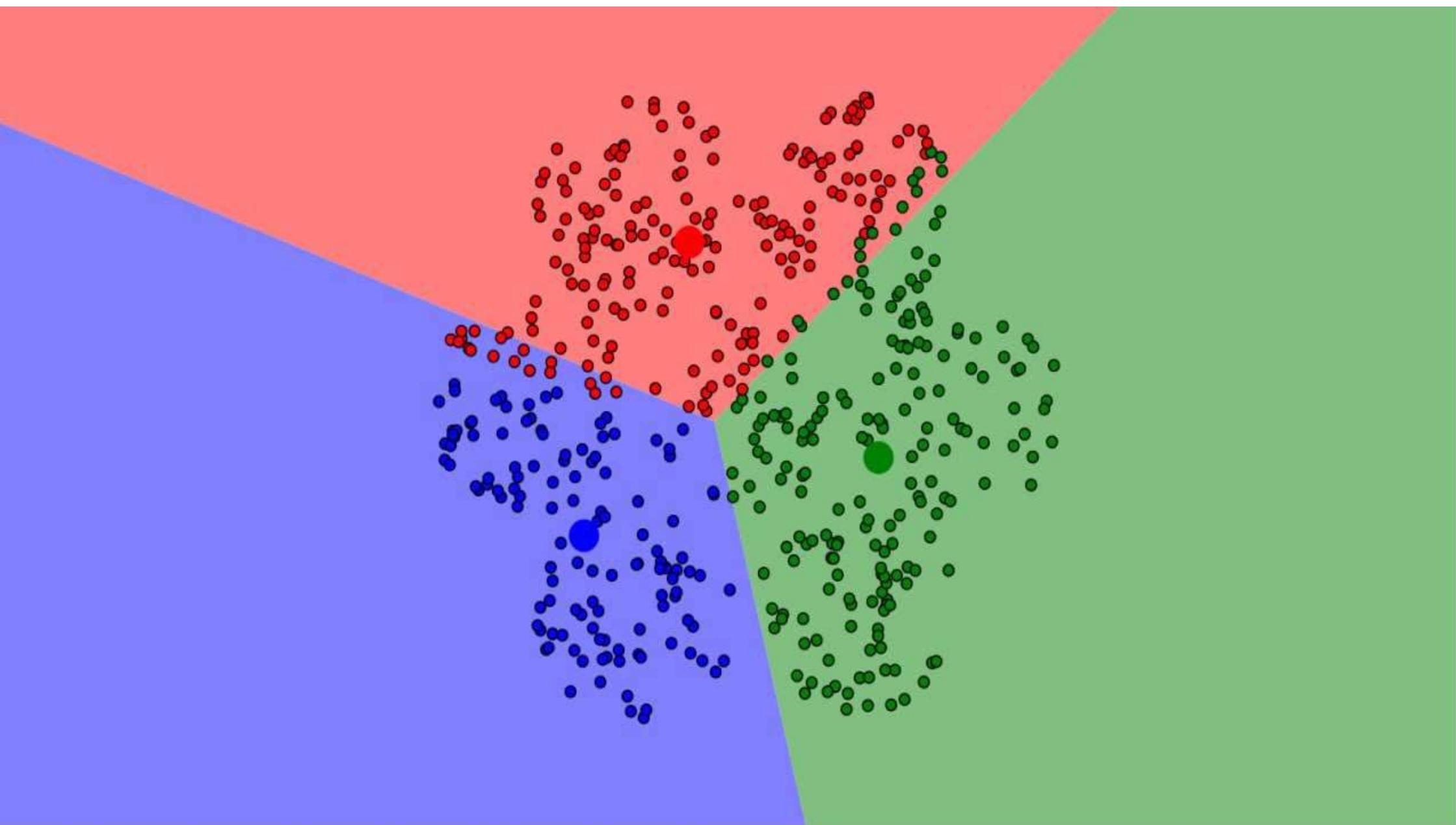


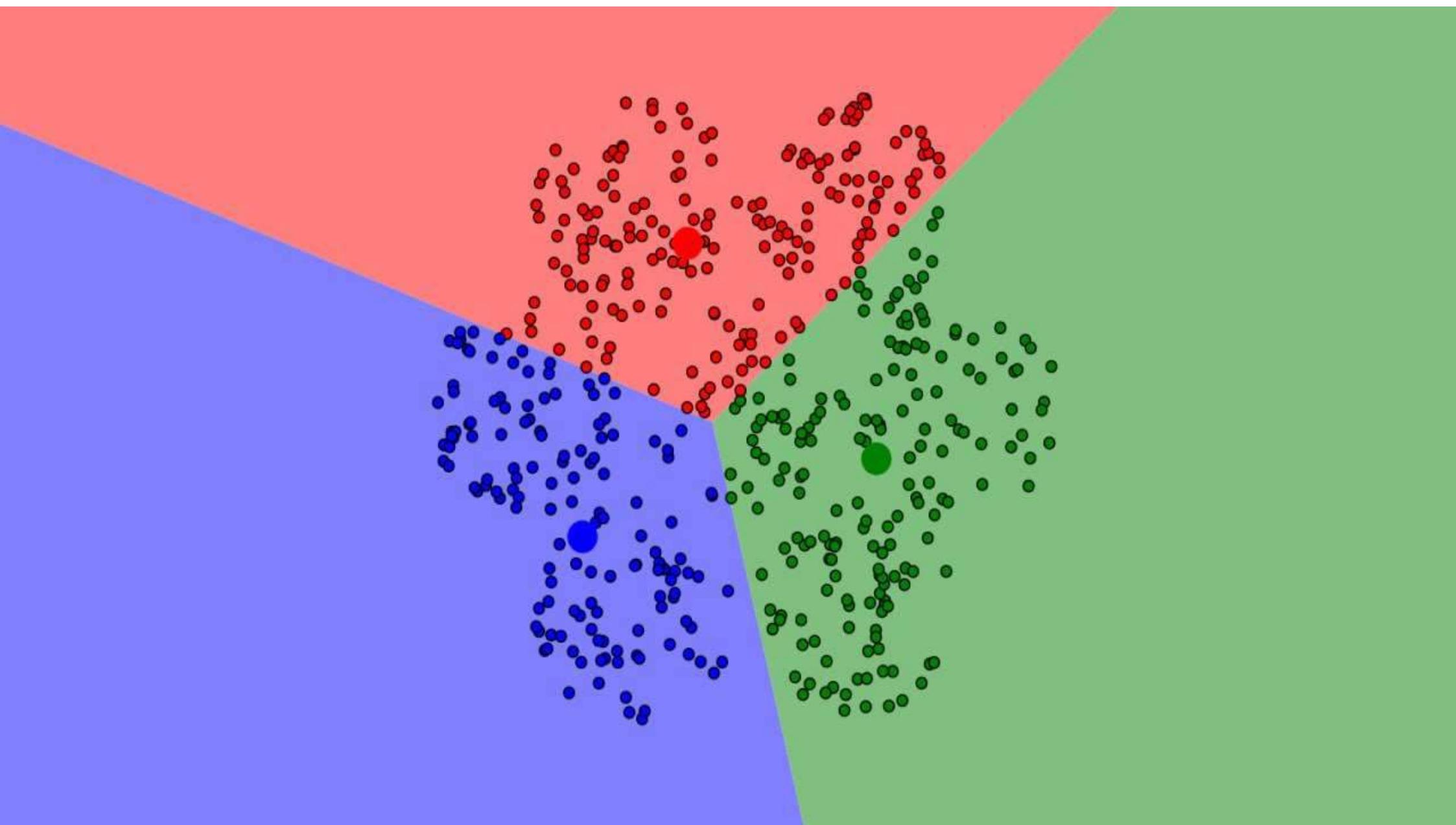










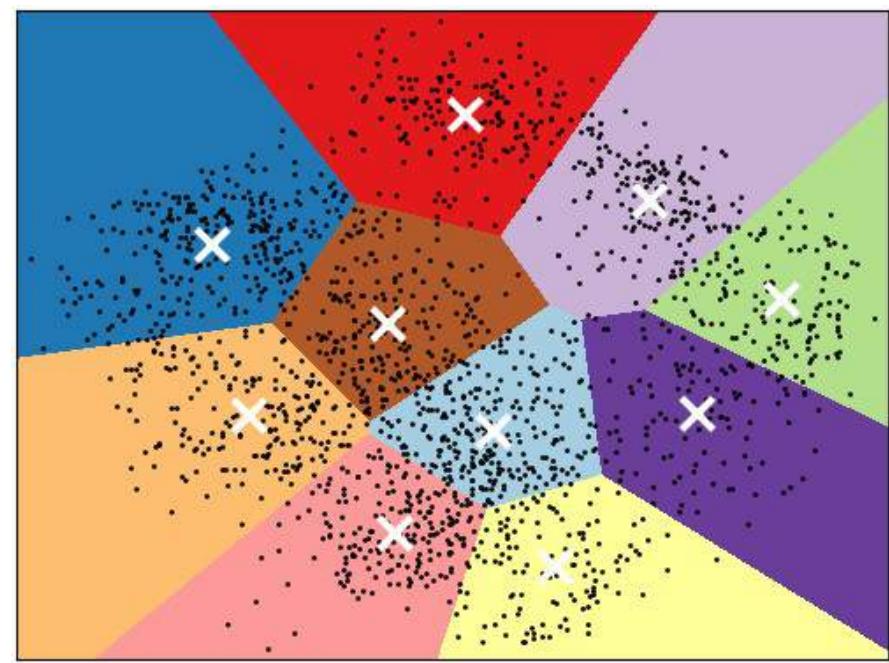


K-means – Example

- Note: even though you can think of K-means clusters as (truncated) spheres enclosing instances, the final goal is to partition the instance space:



J. Mx



Scikit-learn

K-means – Quality of Clusters

Error is typically described as the **sum of all squared errors** between all instances and their closest centroids

$$Error = \sum_{i=1}^k \sum_{\mathbf{x}_j \in \mathcal{C}_i} dist(\mathbf{x}_j, \mathbf{c}_i)^2$$

The equation illustrates the calculation of the total error for a k-means clustering. It consists of two nested summations. The outer summation, indexed by i from 1 to k , represents the sum over all clusters. The inner summation, indexed by j and constrained by $\mathbf{x}_j \in \mathcal{C}_i$, represents the sum over all instances belonging to cluster i . The term $dist(\mathbf{x}_j, \mathbf{c}_i)^2$ represents the squared Euclidean distance between instance \mathbf{x}_j and centroid \mathbf{c}_i .

Image Segmentation with K-means

K=2



K=3



K=10



Original



C. Bishop, 2006

K-means – Limitations

- Number **k** and the **distance metric** need to be chosen beforehand
- Assumes that clusters have **spherical shape** and similar density
- Different **initial points** often lead to different results (in practice k-means is run multiple times to minimize this problem)
- Sensitive to **outliers**

Dataset $\mathcal{X} = \{1, 2, 3, 8, 9, 10, 25\}$ with one feature

Note: Results depend on initialization

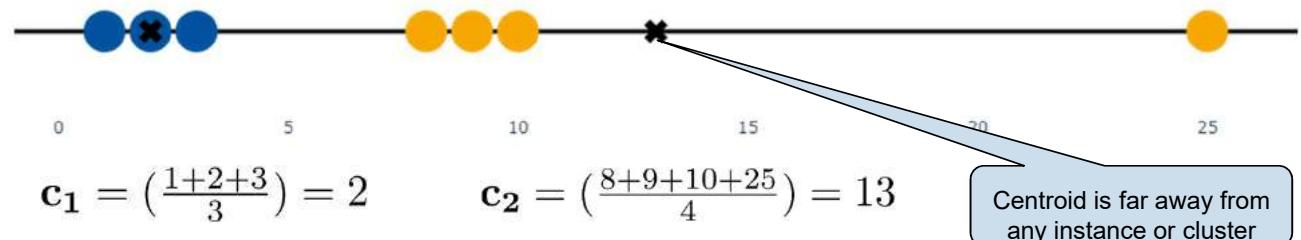


K-means – Limitations

- Number k and the **distance metric** need to be chosen beforehand
- Assumes that clusters have **spherical shape** and similar density
- Different **initial points** often lead to different results (in practice k-means is run multiple times to minimize this problem)
- Sensitive to **outliers**

Dataset $\mathcal{X} = \{1, 2, 3, 8, 9, 10, 25\}$ with one feature

Note: Results depend on initialization



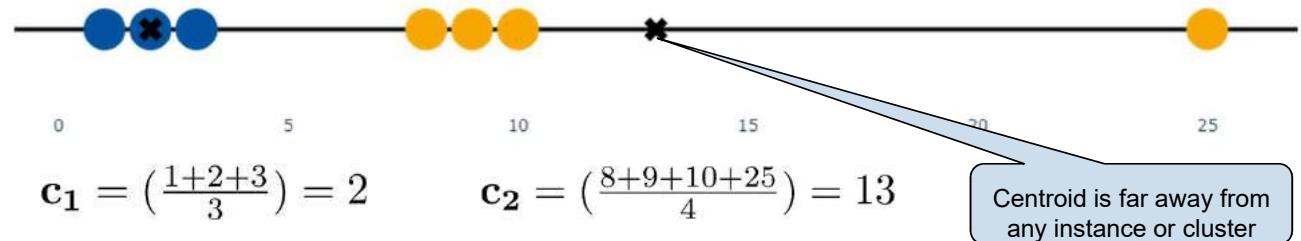
Algorithm terminates because each instance is assigned to correct centroid

K-means – Limitations

- Number k and the **distance metric** need to be chosen beforehand
- Assumes that clusters have **spherical shape** and similar density
- Different **initial points** often lead to different results (in practice k-means is run multiple times to minimize this problem)
- Sensitive to **outliers**

Dataset $\mathcal{X} = \{1, 2, 3, 8, 9, 10, 25\}$ with one feature

Note: Results depend on initialization



Algorithm terminates because each instance is assigned to correct centroid

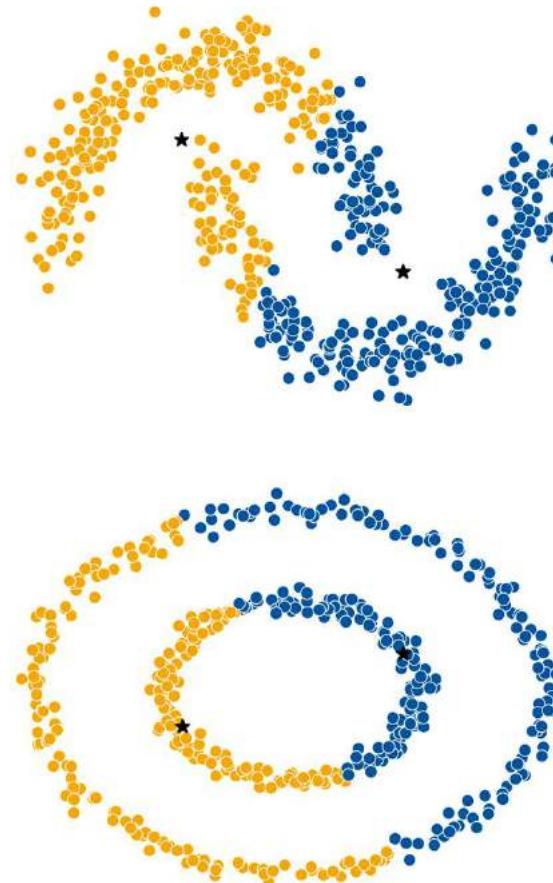
$$Error_{C_1} = (1 - 2)^2 + (2 - 2)^2 + (3 - 2)^2 = 2$$

$$Error_{C_2} = (8 - 13)^2 + (9 - 13)^2 + (10 - 13)^2 + (25 - 13)^2 = 194$$

$$Error = Error_{C_1} + Error_{C_2} = 196$$

K-means – Limitations

- Number **k** and the **distance metric** need to be chosen beforehand
- Assumes that clusters have **spherical shape** and similar density
- Different **initial points** often lead to different results (in practice k-means is run multiple times to minimize this problem)
- Sensitive to **outliers**



K-medoids – Idea

- Uses concrete instances (**medoids**) as cluster's centers rather than the mean values (centroids)
- Similar idea to K-means
- Error is again based on the distances

$$Error = \sum_{i=1}^k \sum_{\mathbf{x}_j \in \mathcal{C}_i} dist(\mathbf{x}_j, \mathbf{m}_i)^2$$

The diagram illustrates the components of the error formula. Three light blue callout boxes point to specific parts of the equation:

- A box labeled "Number of clusters" points to the variable k .
- A box labeled "Instance \mathbf{x}_j in cluster \mathcal{C}_i " points to the term $\mathbf{x}_j \in \mathcal{C}_i$.
- A box labeled "Medoid of cluster \mathcal{C}_i " points to the term \mathbf{m}_i .

K-medoids – Algorithm

- Uses concrete instances (**medoids**) as cluster's centers rather than the mean values (centroids)
- In literature medoids are also known as representative instances

K-medoids algorithm:

1. randomly choose k instances from the dataset \mathcal{X} as the initial cluster centers
2. **repeat until no change**
 - (a) reassign each instance to the cluster with the closest medoid
 - (b) for each medoid \mathbf{m}_i and each non-medoid instance \mathbf{x}_j
 - i. compute the error for the clustering assuming that we **swap** medoid \mathbf{m}_i by \mathbf{x}_j
 - ii. if the error is lower, perform the **swap**

Comparing K-medoids and K-means

- More robust to outliers (e.g., 1D example on the right)
- K-medoids is more flexible (can be used with any similarity measure)
- K-medoids is more time-consuming (although the effect of swaps is limited to the instances that change medoid)

Dataset $\mathcal{X} = \{1, 2, 3, 8, 9, 10, 25\}$ with one feature

Note: Results depend on initialization



Algorithm terminates because there is no swap that lowers the error

Comparing K-medoids and K-means

- More robust to outliers (e.g., 1D example on the right)
- K-medoids is more flexible (can be used with any similarity measure)
- K-medoids is more time-consuming (although the effect of swaps is limited to the instances that change medoid)

Dataset $\mathcal{X} = \{1, 2, 3, 8, 9, 10, 25\}$ with one feature

Note: Results depend on initialization



Algorithm terminates because there is no swap that lowers the error

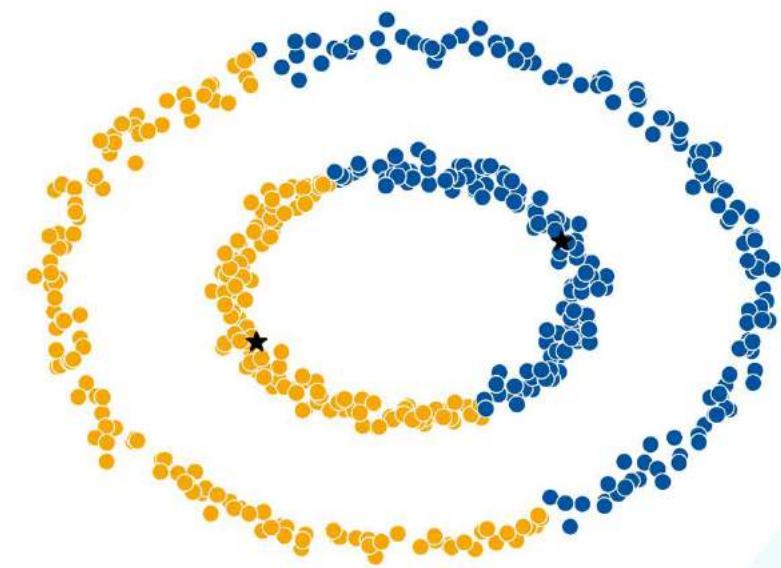
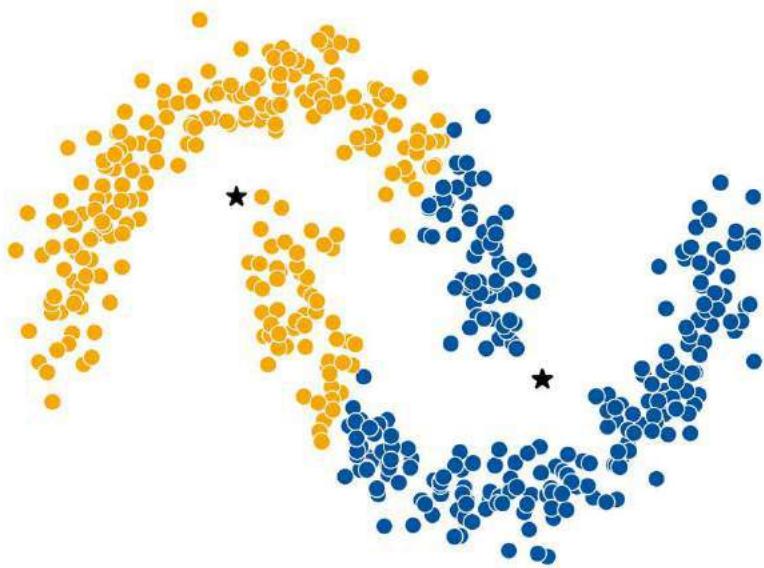
$$Error_{C_1} = (1 - 3)^2 + (2 - 3)^2 + (3 - 3)^2 + (8 - 3)^2 + (9 - 3)^2 + (10 - 3)^2 = 115$$

$$Error_{C_2} = (25 - 25)^2 = 0$$

$$Error = Error_{C_1} + Error_{C_2} = 115$$

K-means error was 196

K-means and K-medoids – Shape Limitations



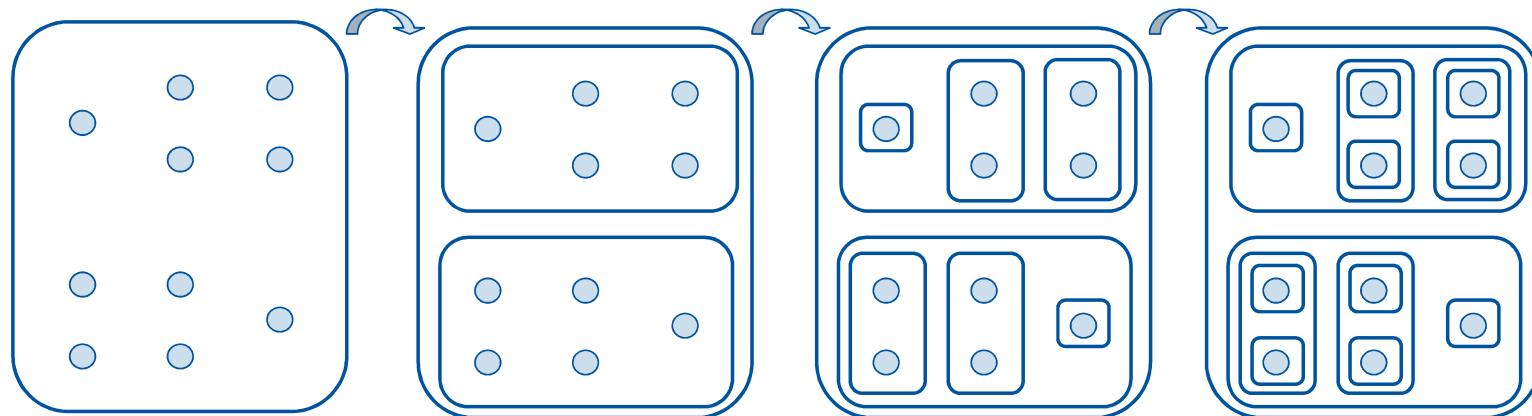
K-means and K-medoids – Choosing K

- The choice of a good value for **K** is quite hard!
- Connects with the more general issue of [evaluation of unsupervised learning approaches](#)
- Some ideas:
 - [Domain knowledge](#): the guidance of the data owners is important!
 - [Random restart](#): we perform clustering multiple times with multiple Ks, we keep the best
 - [Holdout](#): we split the dataset, we test various Ks on part of the data and measure the error on another
 - [Bayesian](#): sometimes, we may have a prior on values of K

Agglomerative Clustering

Agglomerative Clustering

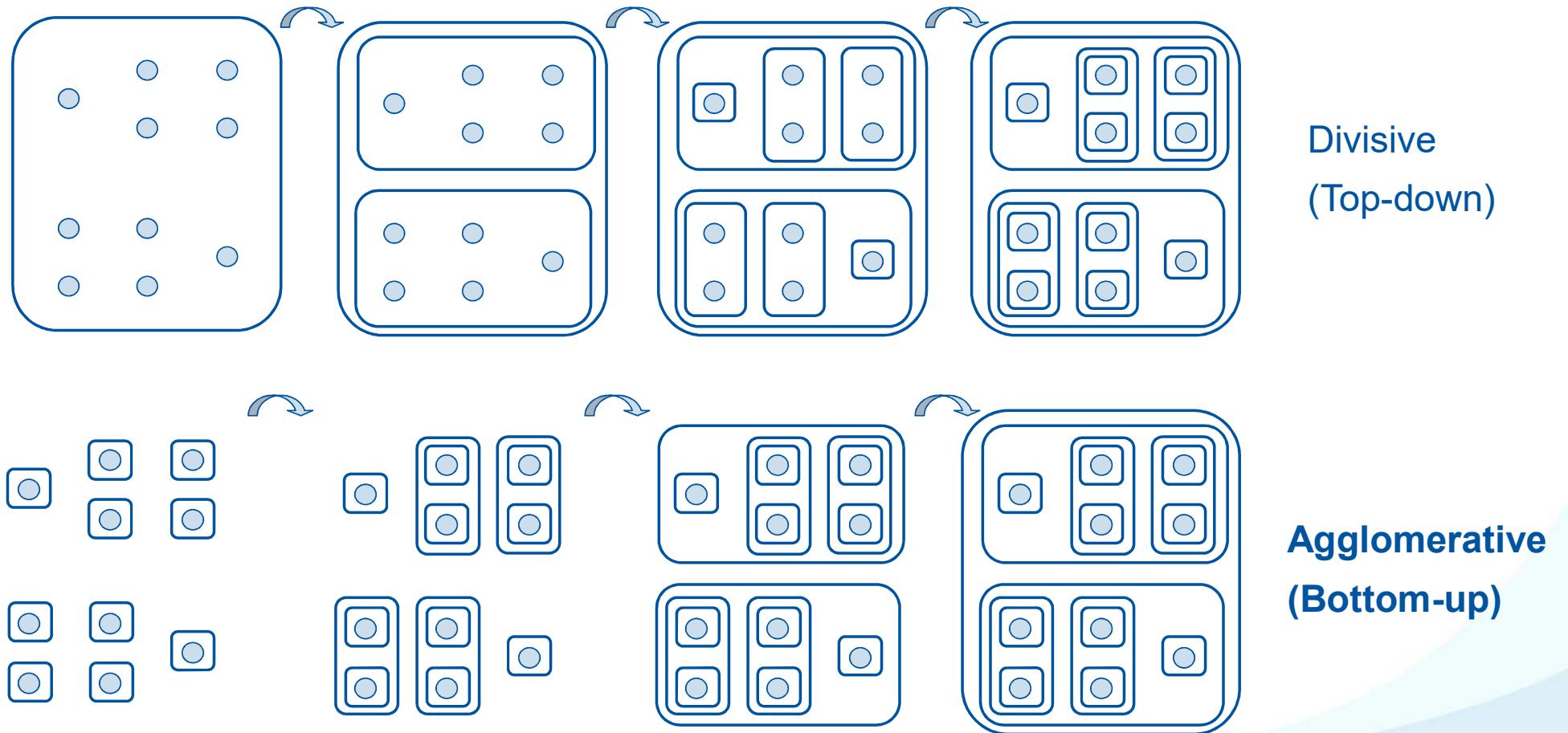
Hierarchical Clustering



**Divisive
(Top-down)**

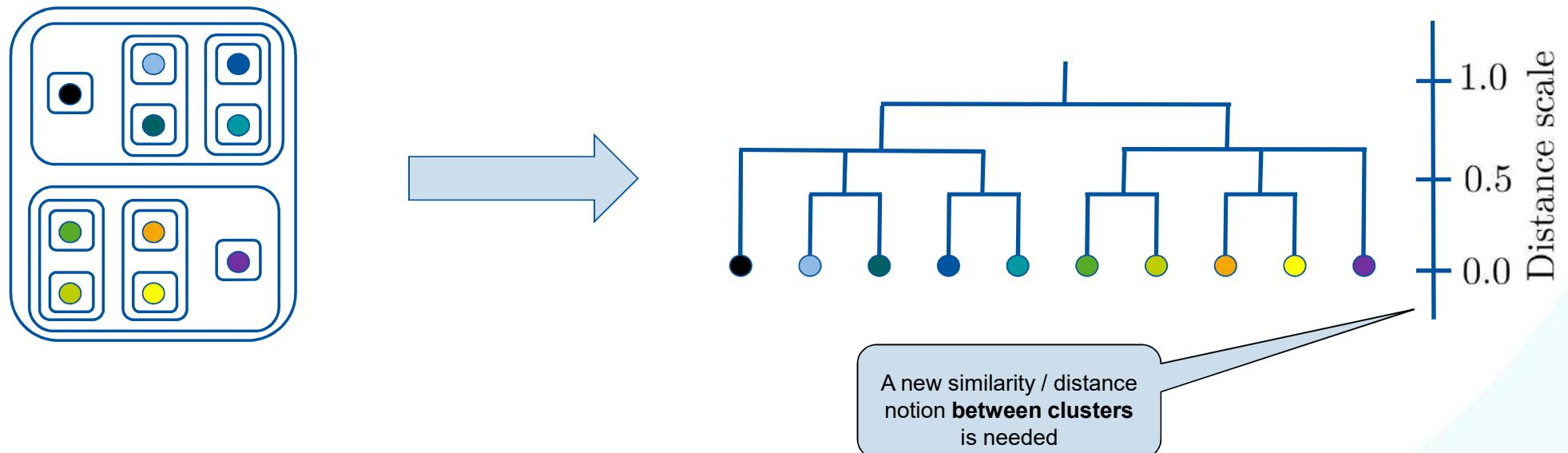
Agglomerative Clustering

Hierarchical Clustering



Dendrogram

- Look for two **clusters** that are most similar and create a **new cluster** by merging them
- Value depicted when merged is the similarity / distance before merging



Linkage Measures

- The distance between clusters is otherwise known as **linkage measure**
- Four widely used linkage measures:

Distance between any two instances \mathbf{x}_n and \mathbf{x}_m

Minimum distance: $\text{dist}_{\min} (\mathcal{C}_i, \mathcal{C}_j) = \min_{\mathbf{x}_n \in \mathcal{C}_i, \mathbf{x}_m \in \mathcal{C}_j} \{\|\mathbf{x}_n - \mathbf{x}_m\|\}$

Maximum distance: $\text{dist}_{\max} (\mathcal{C}_i, \mathcal{C}_j) = \max_{\mathbf{x}_n \in \mathcal{C}_i, \mathbf{x}_m \in \mathcal{C}_j} \{\|\mathbf{x}_n - \mathbf{x}_m\|\}$

\mathbf{c}_i is the mean (centroid) of cluster \mathcal{C}_i

Mean distance: $\text{dist}_{\text{mean}} (\mathcal{C}_i, \mathcal{C}_j) = \|\mathbf{c}_i - \mathbf{c}_j\|$

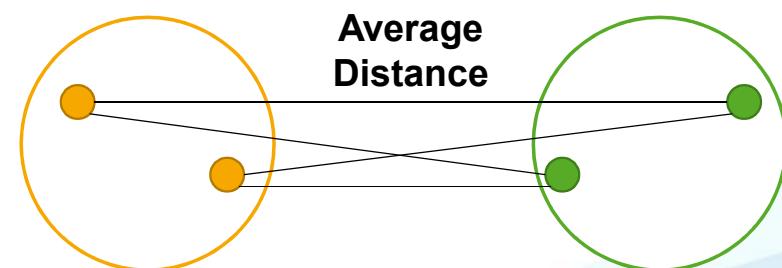
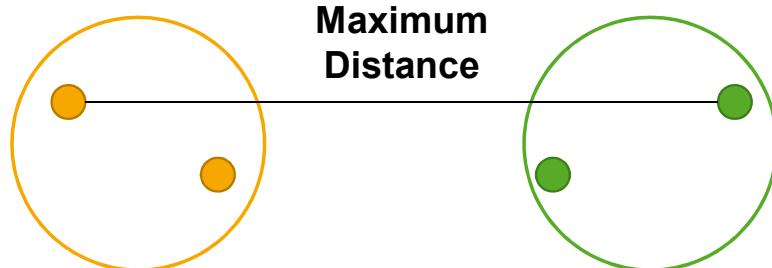
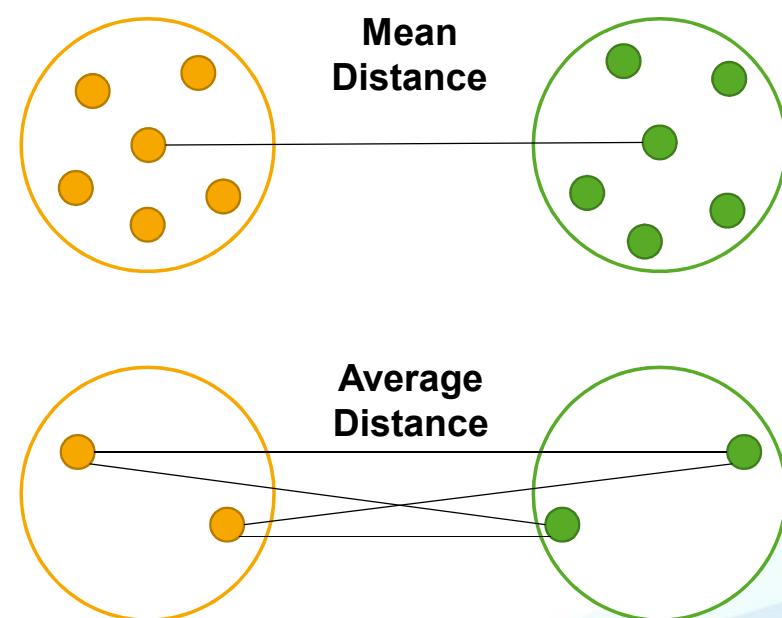
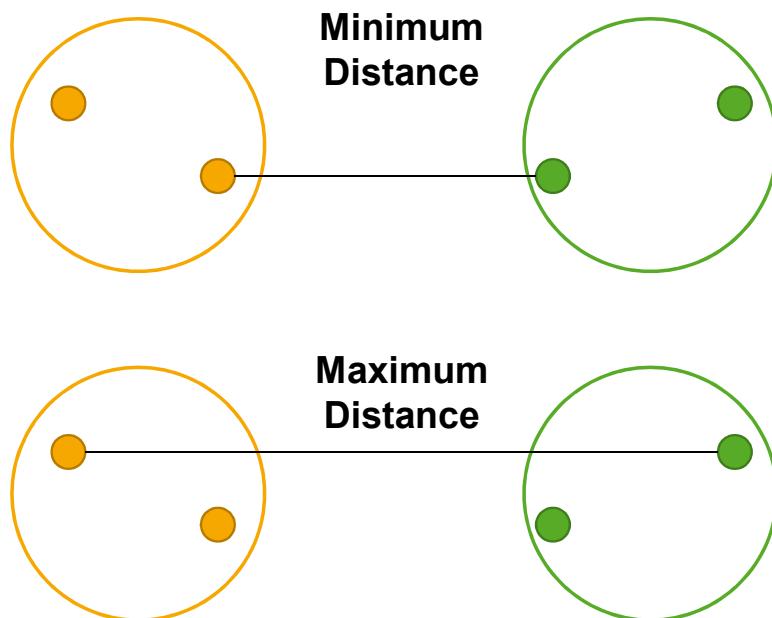
Average distance: $\text{dist}_{\text{avg}} (\mathcal{C}_i, \mathcal{C}_j) = \frac{1}{|\mathcal{C}_i| \cdot |\mathcal{C}_j|} \sum_{\mathbf{x}_n \in \mathcal{C}_i, \mathbf{x}_m \in \mathcal{C}_j} \|\mathbf{x}_n - \mathbf{x}_m\|$

Clusters \mathcal{C}_i and \mathcal{C}_j

$|\mathcal{C}_i|$ is the number of instances in cluster \mathcal{C}_i

Linkage Measures

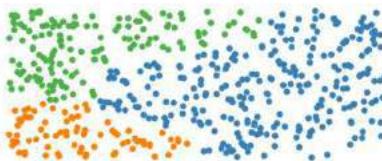
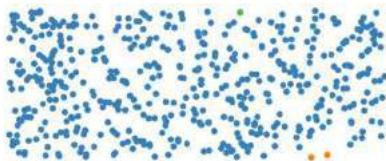
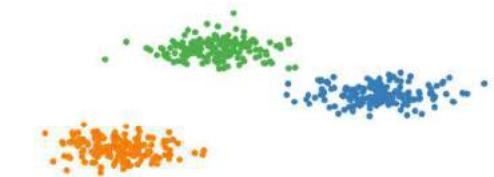
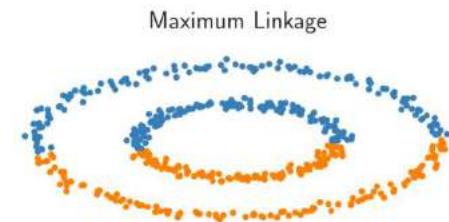
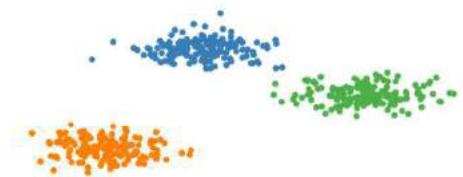
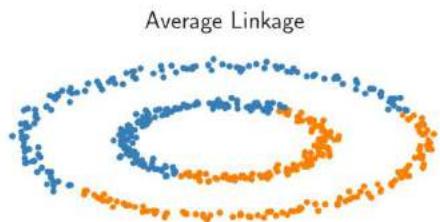
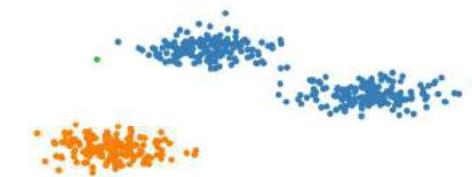
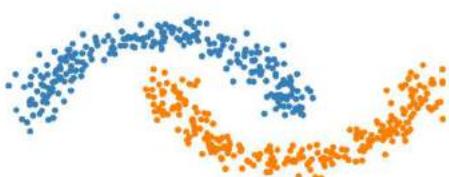
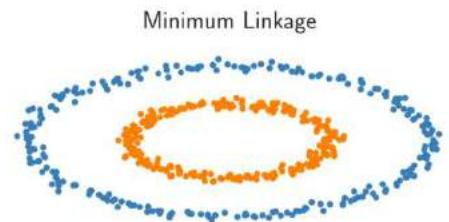
- The distance between clusters is otherwise known as **linkage measure**
- Four widely used linkage measures:



Agglomerative Clustering

Linkage Measures

Different linkage measures may lead to different results



Algorithm

Simplistic version (many variants possible)

Agglomerative hierarchical clustering algorithm:

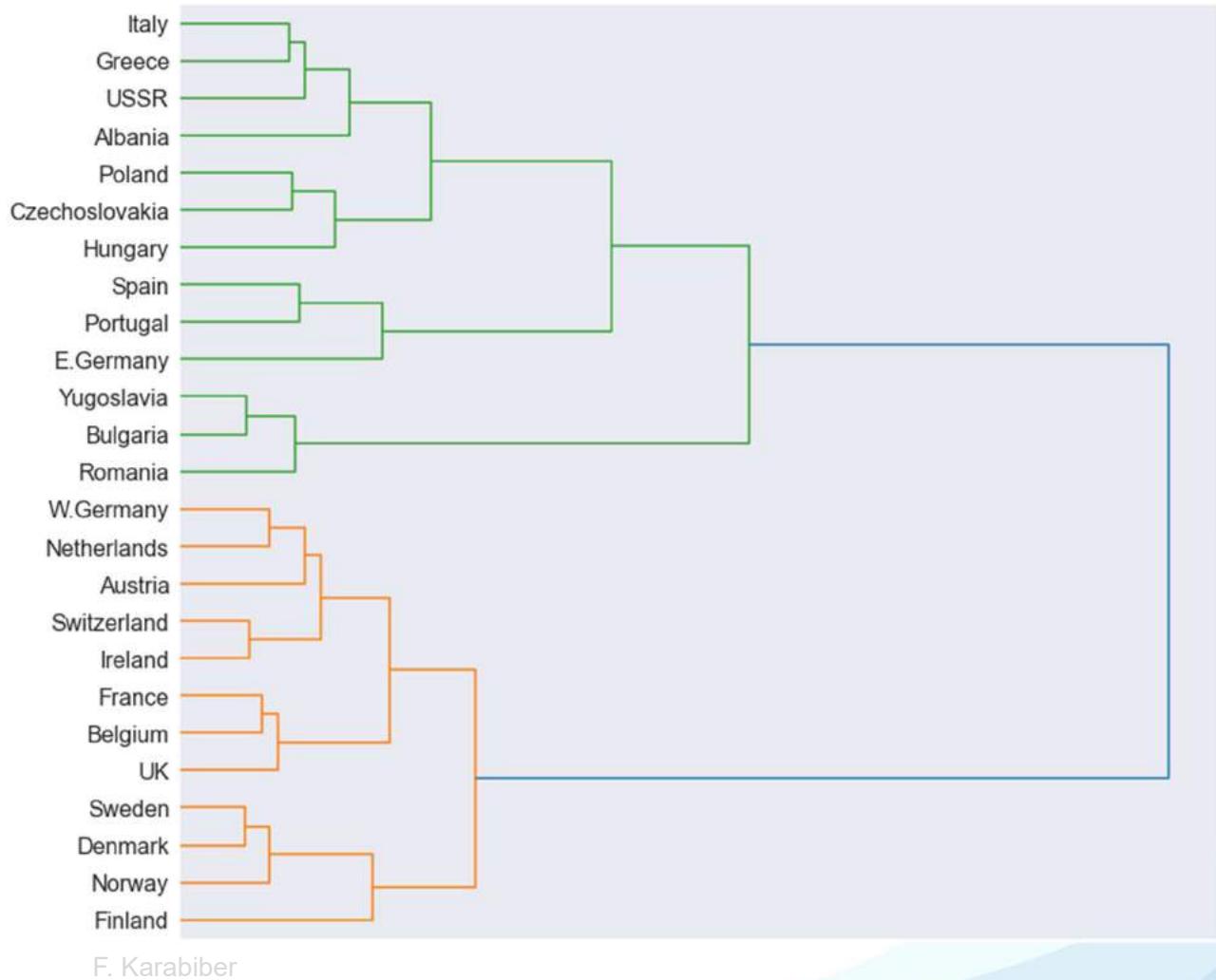
1. create a singleton cluster \mathcal{C}_i for each instance \mathbf{x}_i
2. **repeat until one cluster is left**
 - (a) compute the pairwise distance (using some linkage measure) between any two clusters \mathcal{C}_i and \mathcal{C}_j
 - (b) merge the two closest clusters
3. **return** dendrogram

Agglomerative Clustering

Dendrogram – Example

Countries clustered by source of average protein consumption

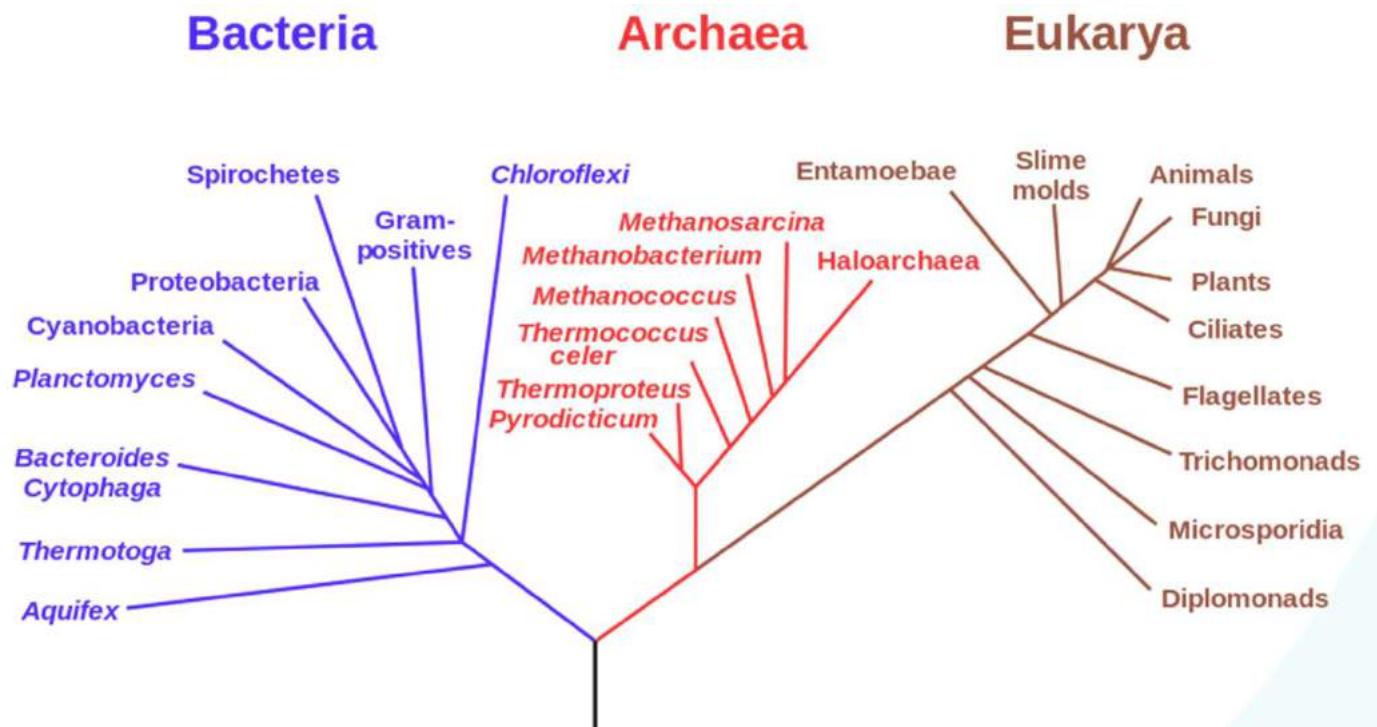
Note that the agglomerative clustering procedure “discovers” geographic proximity!



Dendrogram – Example

Phylogenetic trees:

Dendograms obtained through clustering by genetic information (in this case, tRNA)



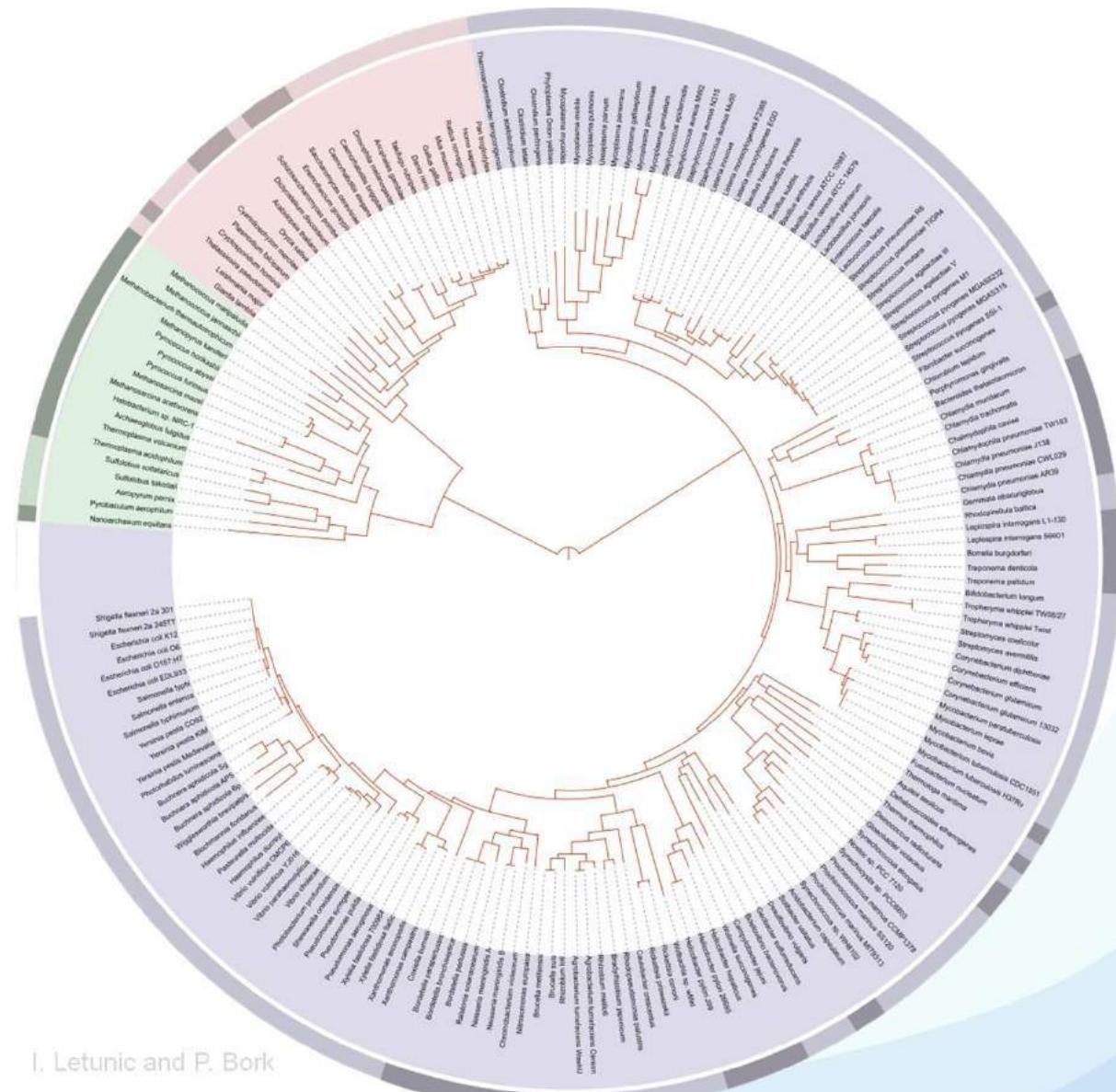
C.R. Woese et al.

Agglomerative Clustering

Dendrogram – Example

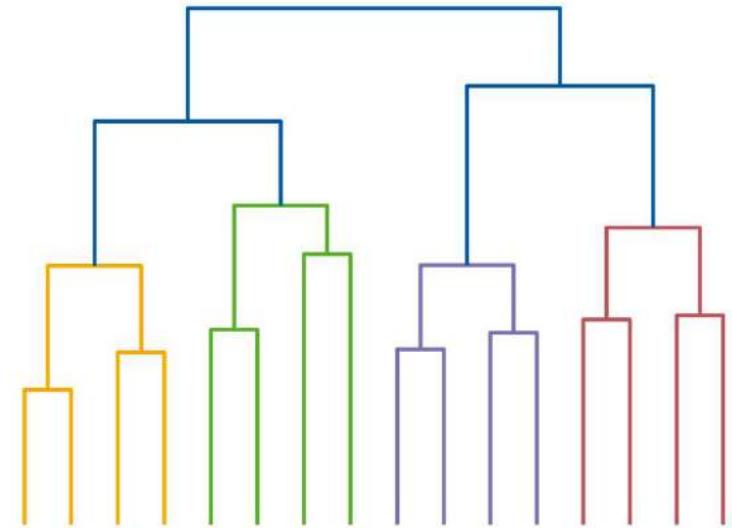
Phylogenetic trees:

Dendograms obtained through clustering by genetic information (in this case, full genome sequencing)



Properties

- No a priori information / decision about the number of clusters is required
- Dendrogram allows analysts to “play” with abstraction level
- The algorithm cannot undo joins that turn out to be undesirable
- There is no approach to objectively minimize some well-defined errors



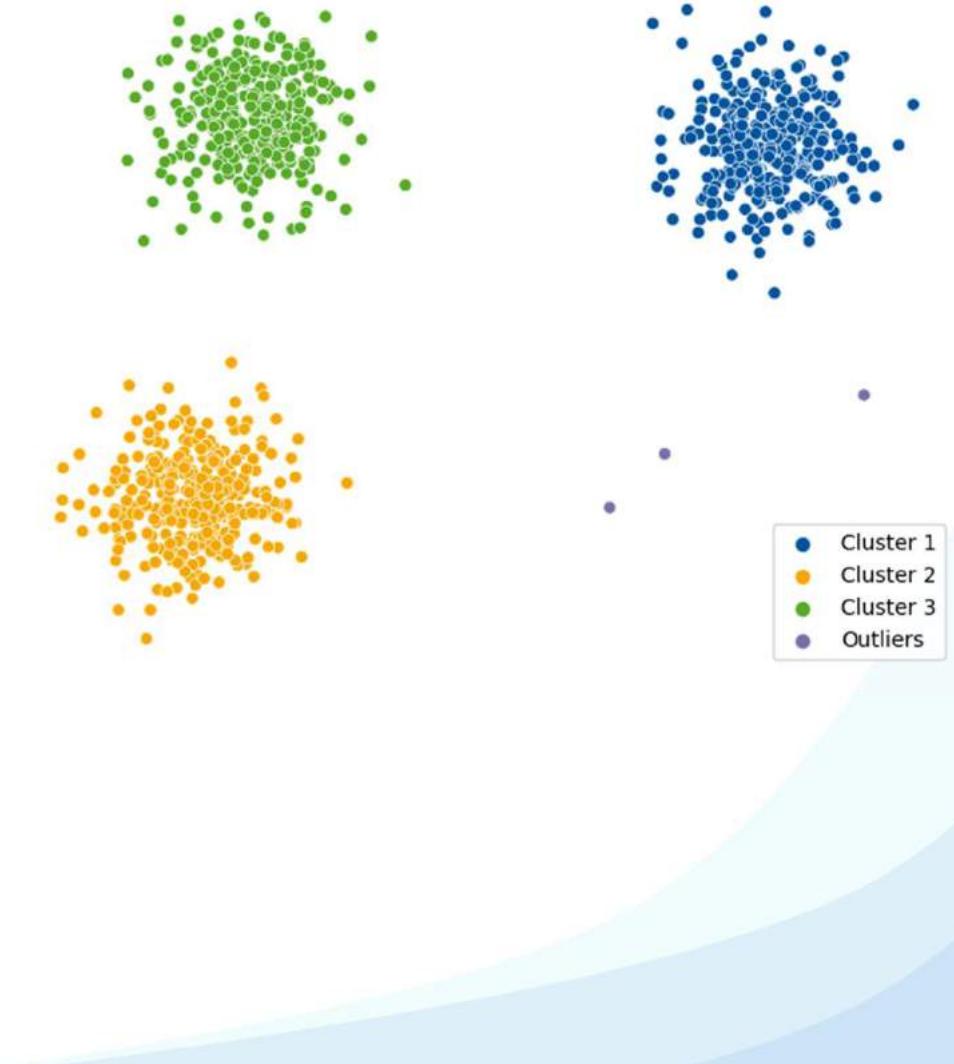
Density-Based Clustering

Density-Based Clustering

- Clusters are areas of **higher density**
- Used to find **clusters of any shape** (contrary to partitioning and hierarchical methods which tend to find spherical clusters)

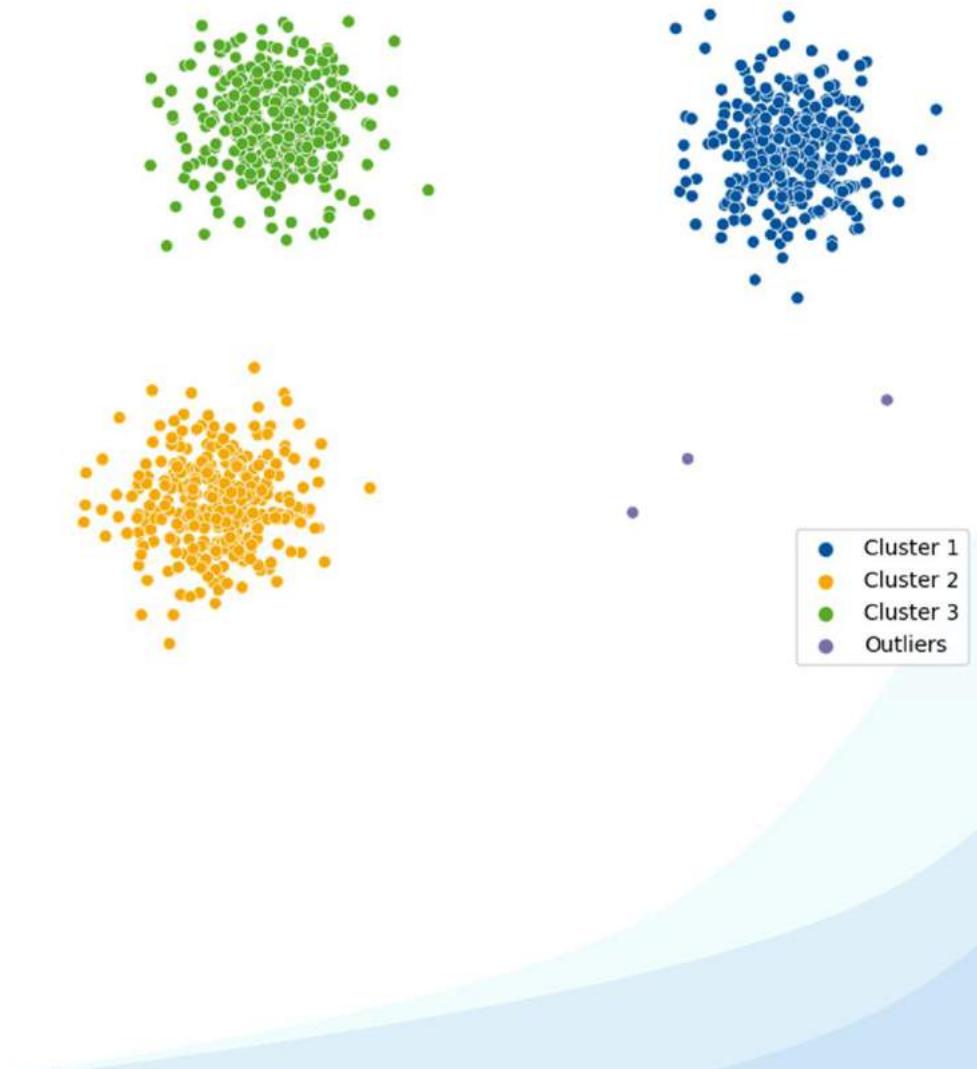
Density-Based Clustering

- Clusters are areas of **higher density**
- Used to find **clusters of any shape** (contrary to partitioning and hierarchical methods which tend to find spherical clusters)
- Instances in sparse areas are considered to be **outliers**
- Example – **Density-Based Spatial Clustering of Applications with Noise (DBSCAN)**



Density-Based Clustering

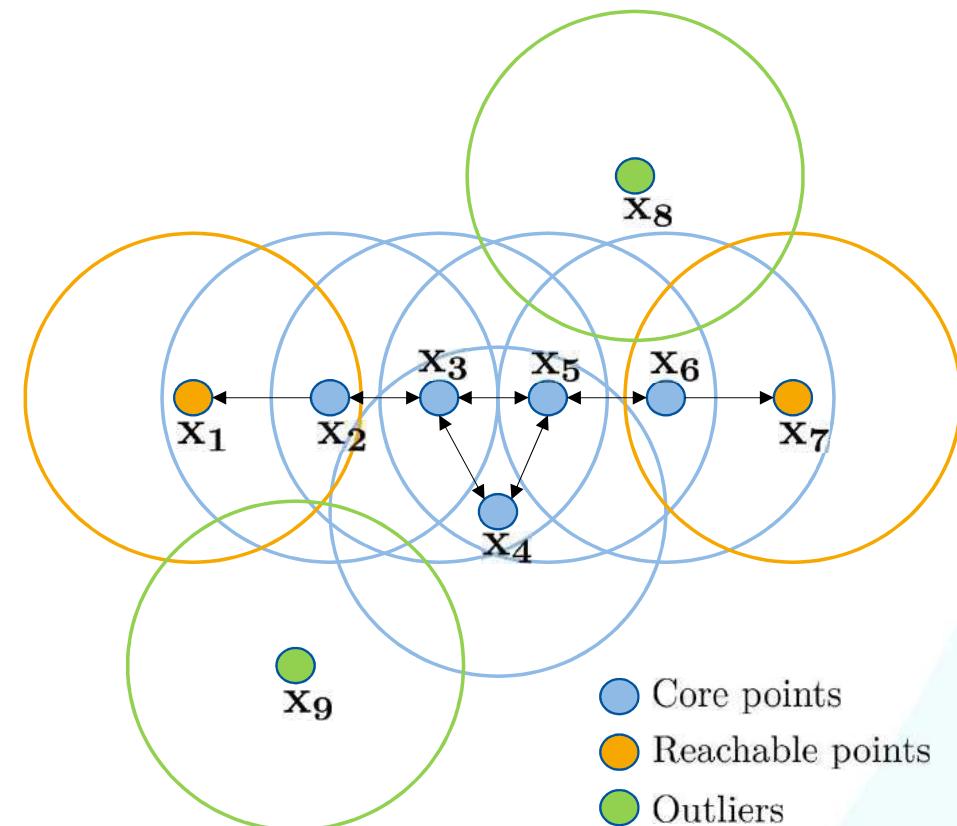
- Two instances x_i and x_j are **density-connected** if there is a core point x_k such that both x_i and x_j are reachable from x_k
- Density-connectedness is **symmetric** (unlike reachability)
- A **cluster** satisfies the following two properties:
 - All instances within the cluster are **mutually density-connected**
 - Any two density-connected core points are part of the cluster



Density-Based (DBSCAN)

DBSCAN

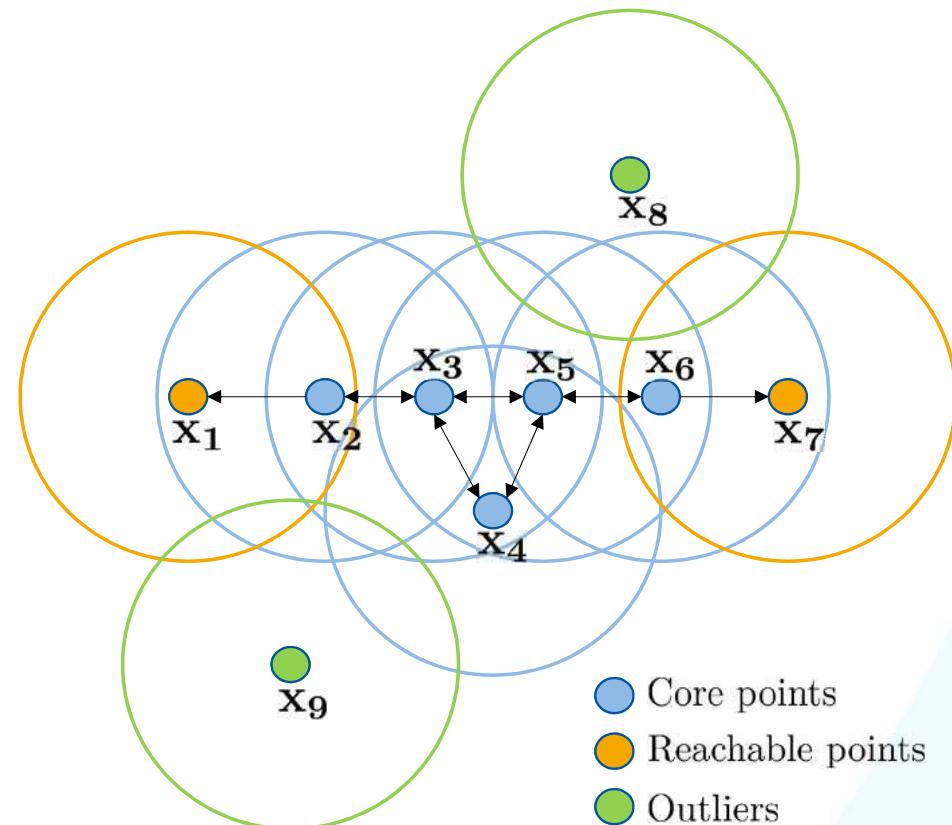
- Two parameters:
 - ϵ (fixed neighborhood size)
 - $MinPts$ (density threshold for dense regions)
- ϵ is the maximum radius of the neighborhood from x_i
- Instance x_i is a **core point** if at least $MinPts$ are within distance ϵ (including x_i)



ϵ is indicated by circles and $MinPts = 3$

DBSCAN - Example

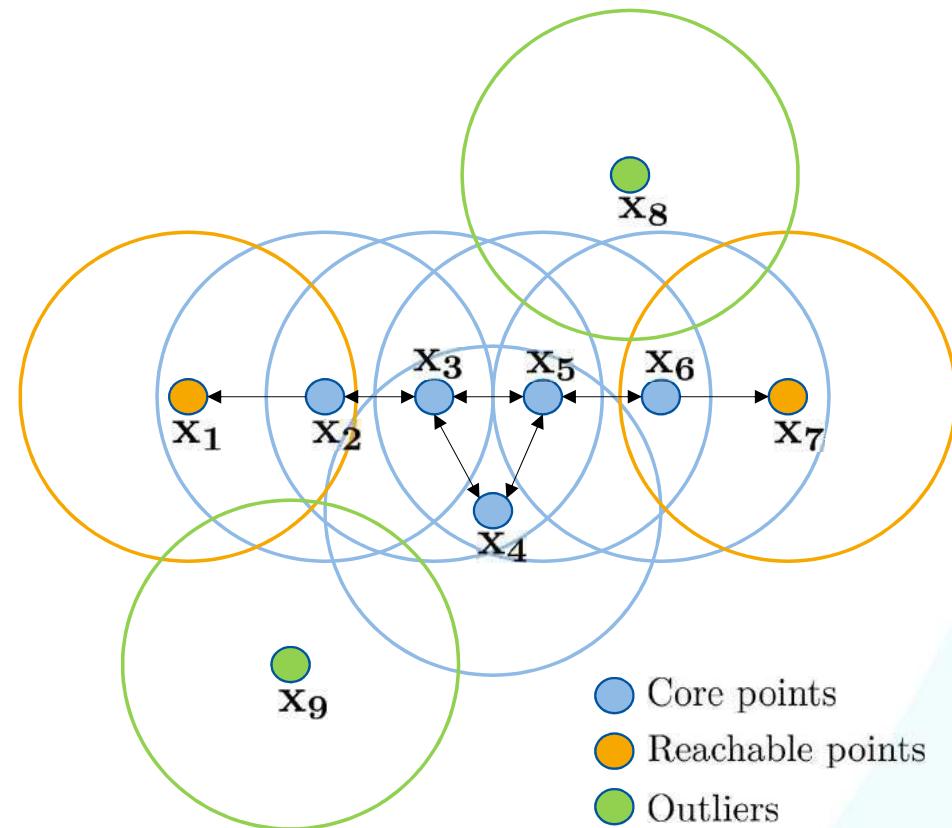
- An instance x_j is **directly reachable** from x_i if x_j is within distance ϵ from x_i and x_i is a **core point**



ϵ is indicated by circles and $MinPts = 3$

DBSCAN - Example

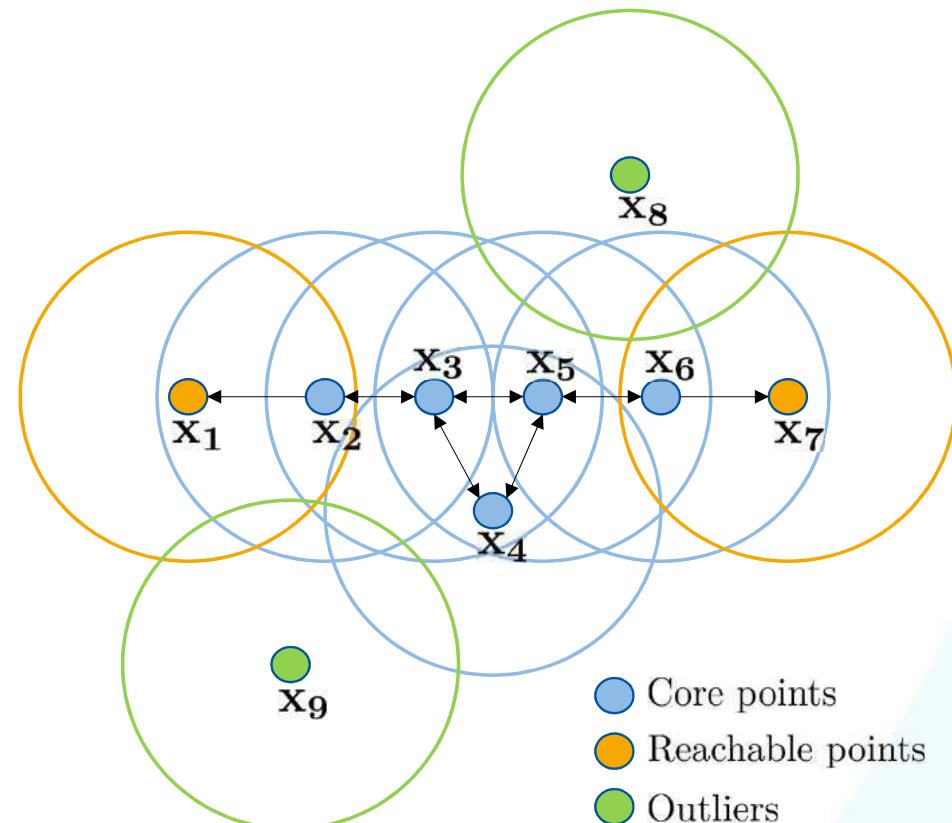
- An instance x_j is **directly reachable** from x_i if x_j is within distance ϵ from x_i and x_i is a **core point**
- An instance x_j is **reachable** from x_i if there is a path $\langle y_1, y_2, \dots, y_K \rangle$ with $y_1 = x_i$ and $y_K = x_j$ where each y_k is directly reachable from y_{k-1}
- All the points on the path must be **core points**, except for x_j , i.e., y_1, y_2, \dots, y_{n-1} are core points



ϵ is indicated by circles and $MinPts = 3$

DBSCAN - Example

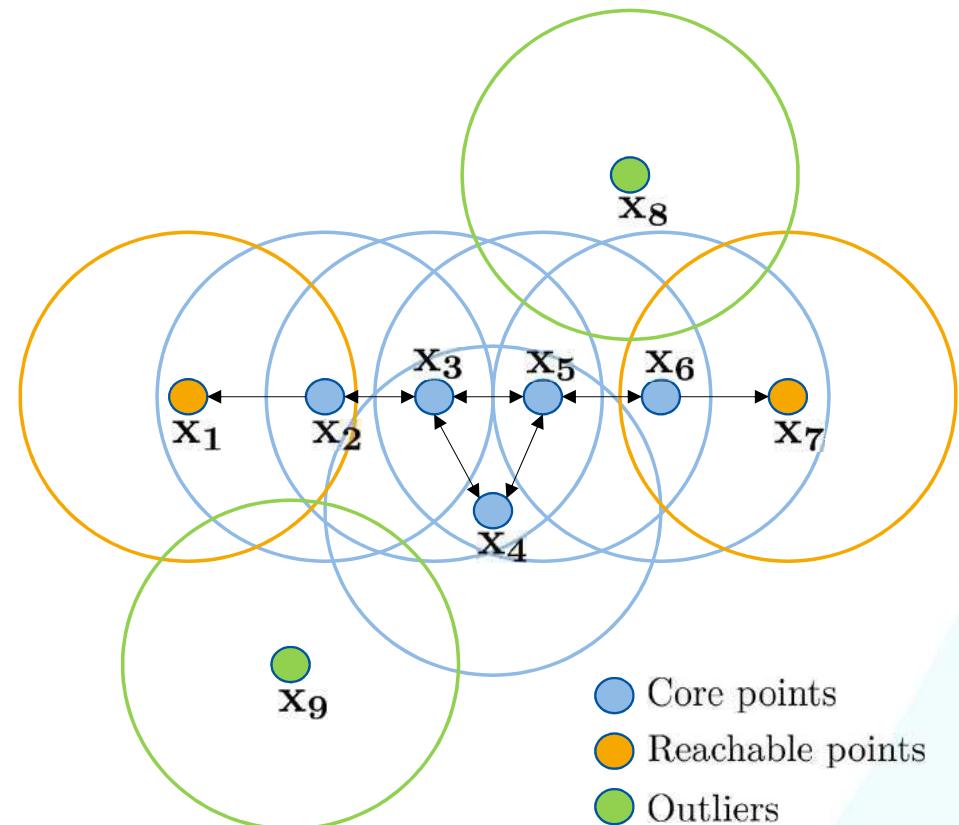
- An instance x_j is **directly reachable** from x_i if x_j is within distance ϵ from x_i and x_i is a **core point**
- An instance x_j is **reachable** from x_i if there is a path $\langle y_1, y_2, \dots, y_K \rangle$ with $y_1 = x_i$ and $y_K = x_j$ where each y_k is directly reachable from y_{k-1}
- All the points on the path must be **core points**, except for x_j , i.e., y_1, y_2, \dots, y_{n-1} are core points
- All points **not reachable** from any other point are **outliers**



ϵ is indicated by circles and $MinPts = 3$

DBSCAN - Approach

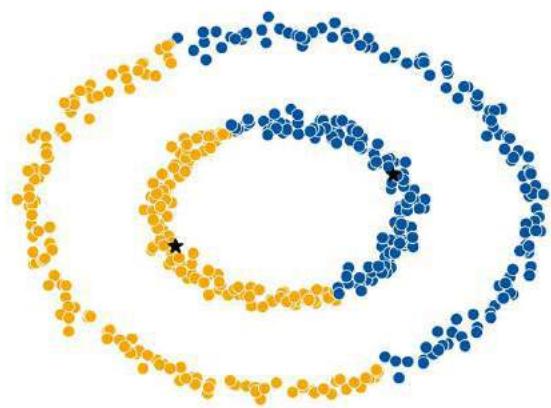
- The approach iteratively selects a **core point x_i , not yet part of a cluster** and creates a cluster for it
- The cluster is **incrementally extended** by adding all neighboring points of core points in the cluster
- If a cluster cannot be extended anymore, the next unvisited core point is considered
- The approach is **not entirely deterministic**: Points reachable from more than one cluster are assigned based on the processing order



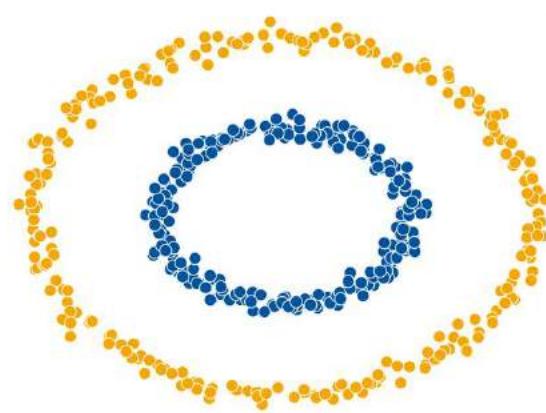
ϵ is indicated by circles and $MinPts = 3$

Density-Based (DBSCAN)

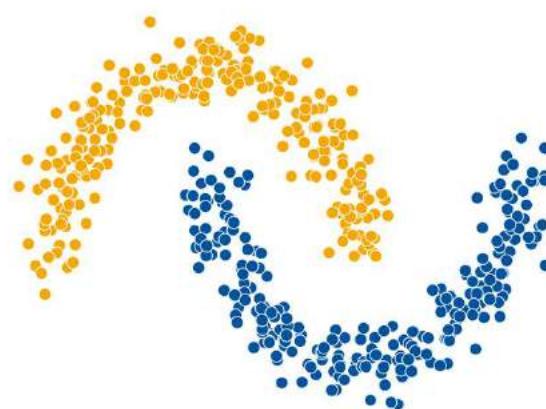
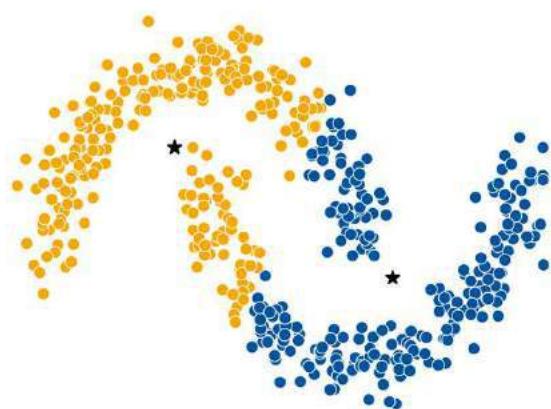
DBSCAN - Graphical Examples



K-means



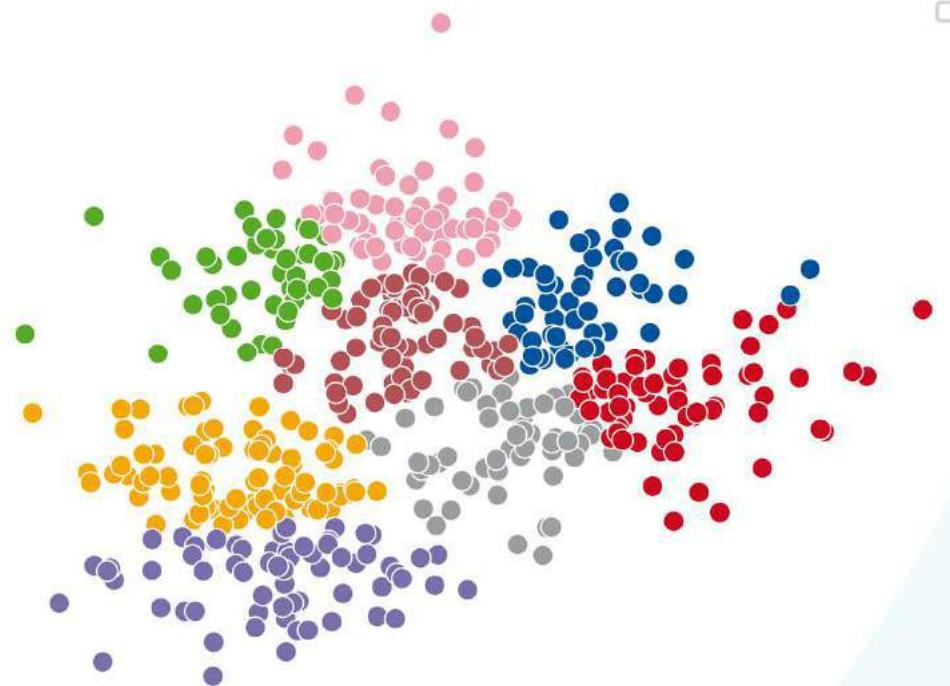
DBSCAN



Conclusion

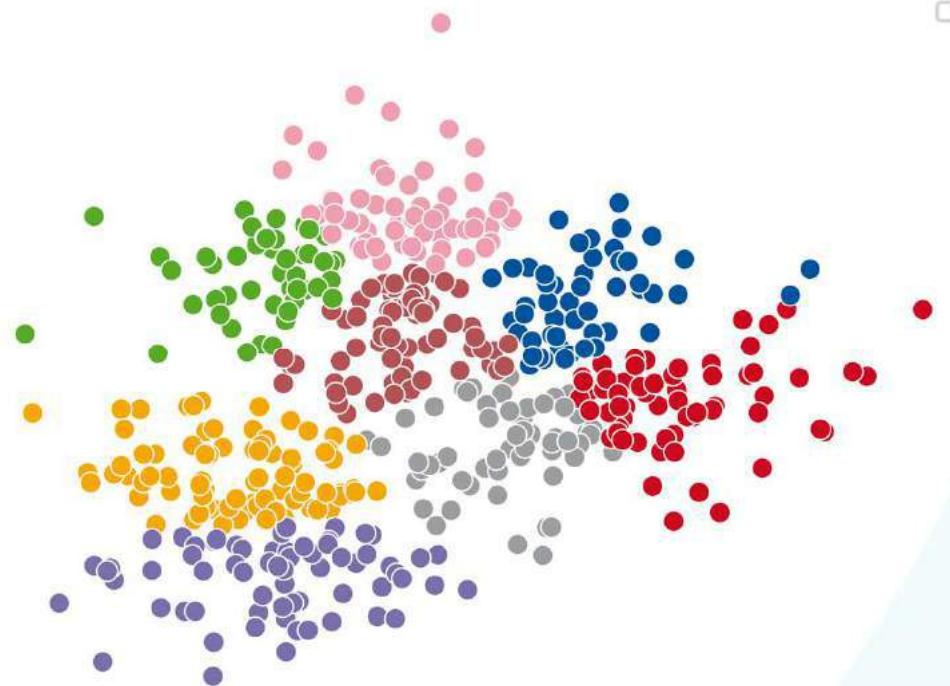
General Problem – Cluster Interpretation

- Cluster quality may be good, but this **does not imply that the clusters reveal new insights**
- Describe clusters in terms of their features (e.g., compare centroids)
- Use simple visualization techniques like **boxplots to compare clusters**



Takeaways

- Clustering: grouping together unlabeled instances
- Useful for **explorative analysis**, and when choosing a label does not make sense
- However, results are often **hard to validate!**
- Various approaches:
 - Based on (spatial) distance
 - By agglomeration
 - By density
- And many more!
- Next up: **Frequent Itemsets**



Elements of Machine Learning & Data Science

Frequent Itemsets

Lecture 9

Prof. Wil van der Aalst

Marco Pegoraro, M.Sc.

Christopher Schwanen, M.Sc.

Tsunghao Huang, M.Sc.

Frequent Itemsets

1. Introduction
2. Properties of Frequent Itemsets
3. A-Priori Algorithm
4. FP-Growth Algorithm



Pattern Mining

- Finding surprising **patterns** in the input data
- Types of patterns:
 - **Frequent itemsets**
 - Association rules
 - Sequential patterns
 - Partial orders
 - Subgraphs

Itemset Data

ID	f_1	f_2	f_3	f_4	...	f_D
1						
2						
3						
4						
5						
...						

Each instance is a **transaction**

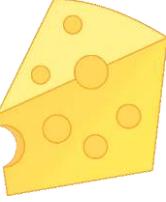
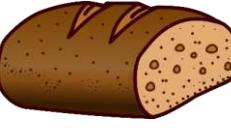
Each feature refers to an **item** (e.g., a product, disease, song, course, or error code)

Each cell describes the **presence of the item** (Boolean 0/1 or Positive Integer Count)

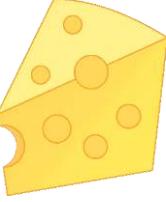
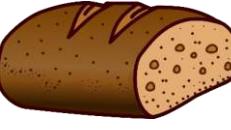
Itemset Data – Example



Itemset Data – Example

ID					...	
1	2	2	0	3		2
2	0	0	1	1		0
3	2	1	0	0		0
4	0	1	0	0		0
5	0	0	0	0		2
...

Itemset Data – Example

ID					...	
1	True	True	False	True		True
2	False	False	True	True		False
3	True	True	False	False		False
4	False	True	False	False		False
5	False	False	False	False		True
...

Other Itemset Data Examples

Rows	Columns
EdX users	Courses taken
Spotify users	Songs Played
Netflix users	Movies Watched
Patients in hospital	Diseases
Repair bills	Components replaced
...	...

Application of Frequent Itemsets

ID					...	
1	True	True	False	True		True
2	False	False	True	True		False
3	True	True	False	True		False
4	False	True	False	True		False
5	False	False	False	False		True
...



Frequent Itemsets (movies)

NETFLIX

Application of Frequent Itemsets

ID					...	
1	True	True	False	True		True
2	True	True	True	False		False
3	True	True	False	True		False
4	True	False	False	False		False
5	False	False	False	False		True
...



Frequent Itemsets



Introduction

- A notorious success story: the [Tesco Clubcard](#)
- Introduced in 1995, it was the first loyalty card with [automatic data collection](#)
- Widely regarded as responsible for Tesco's supremacy in the UK
- 1bn£ of increase in sales (4%) in one year
- Today, the Clubcard program is still incredibly profitable, even though Tesco gives away about 1bn£ in rewards and discounts each year!



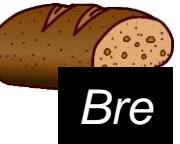
“You know more about my customers after three months than I know after 30 years.”

- Lord MacLaurin, chairman for Tesco, talking to the data scientists of the Clubcard program

Frequent Itemsets – Notation

- $\mathcal{I} = \{I_1, I_2, \dots, I_D\}$ is the set of all possible items
- $\mathcal{A} \subseteq \mathcal{I}$ is an itemset
- A transaction \mathcal{T} is a non-empty itemset
- A dataset \mathcal{X} is a collection of transactions
- Technically $\mathcal{X} \in \mathbb{M}(\mathbb{P}(\mathcal{I}))$ such that $\emptyset \notin \mathcal{X}$
(\mathbb{M} is the multiset and \mathbb{P} is the powerset operator)

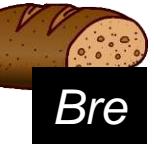
Frequent Itemsets – Notation Example

ID					...		Pas
1	2 (true)	0 (false)	0 (false)	3 (true)	0 (false)	2 (true)	
2	0 (false)	0 (false)	1 (true)	1 (true)	0 (false)	0 (false)	
3	2 (true)	1 (true)	0 (false)	0 (false)	0 (false)	0 (false)	
4	0 (false)	1 (true)	0 (false)	0 (false)	0 (false)	0 (false)	

$\mathcal{X} \in \mathbb{M}(\mathbb{P}(\mathcal{I}))$

- Set of all items $\mathcal{I} = \{Che, Bre, Chi, Mil, \dots, Pas\}$
- Transaction $\mathcal{T}_1 = \{Che, Mil, Pas\} \subseteq \mathcal{I}$
- Dataset with four transactions $\mathcal{X} = [\{Che, Mil, Pas\}, \{Chi, Mil\}, \{Che, Bre\}, \{Bre\}]$
- Dataset with ten transactions $\mathcal{X} = [\{Che, Mil, Pas\}^4, \{Chi, Mil\}^3, \{Che, Bre\}^2, \{Bre\}^1]$

Frequent Itemsets – Notation Generalization

ID					...		Pas
1	2 (true)	0 (false)	0 (false)	3 (true)	0 (false)	2 (true)	
2	0 (false)	0 (false)	1 (true)	1 (true)	0 (false)	0 (false)	
3	2 (true)	1 (true)	0 (false)	0 (false)	0 (false)	0 (false)	
4	0 (false)	1 (true)	0 (false)	0 (false)	0 (false)	0 (false)	

$\mathcal{X} \in \mathbb{M}(\mathbb{M}(\mathcal{I}))$

- Set of all items $\mathcal{I} = \{Che, Bre, Chi, Mil, \dots, Pas\}$
- Transaction $T_1 = [Che^2, Mil^3, Pas^2] \in \mathbb{M}(\mathcal{I})$
- Dataset with four transactions $\mathcal{X} = [[Che^2, Mil^3, Pas^2], [Chi, Mil], [Che^2, Bre], [Bre]]$

We will consider only itemsets that are proper sets (not multisets). However, generalization is trivial.

Frequent Itemsets – Support

$$\text{support}(\mathcal{A}) = \frac{|[\mathcal{T} \in \mathcal{X} | \mathcal{A} \subseteq \mathcal{T}]|}{|\mathcal{X}|}$$

(relative)

Fraction of transactions \mathcal{T} in dataset \mathcal{X} that cover the itemset \mathcal{A}

$$\text{support_count}(\mathcal{A}) = |[\mathcal{T} \in \mathcal{X} | \mathcal{A} \subseteq \mathcal{T}]|$$

(absolute, also called frequency or count)

Frequent Itemsets – Support

$$\text{support}(\mathcal{A}) = \frac{|[\mathcal{T} \in \mathcal{X} | \mathcal{A} \subseteq \mathcal{T}]|}{|\mathcal{X}|}$$

(relative)

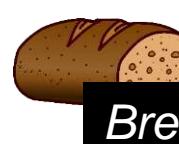
Fraction of transactions \mathcal{T} in dataset \mathcal{X} that cover the itemset \mathcal{A}

$$\text{support_count}(\mathcal{A}) = |[\mathcal{T} \in \mathcal{X} | \mathcal{A} \subseteq \mathcal{T}]|$$

(absolute, also called frequency or count)

- Minimum **support threshold min_sup** : lower bound for $\text{support}(\mathcal{A})$
- An itemset is **frequent** if its support is higher than min_sup
- Frequent itemsets are used to find **association rules**

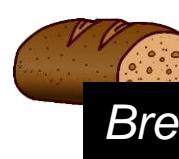
Support – Example

ID					...		Pas
1	2 (true)	0 (false)	0 (false)	3 (true)	0 (false)	2 (true)	
2	0 (false)	0 (false)	1 (true)	1 (true)	0 (false)	0 (false)	
3	2 (true)	1 (true)	0 (false)	0 (false)	0 (false)	0 (false)	
4	1 (true)	1 (true)	0 (false)	1 (true)	0 (false)	0 (false)	

$$\mathcal{X} \in \mathbb{M}(\mathbb{P}(\mathcal{I}))$$

Dataset $\mathcal{X} = [\{Che, Mil, Pas\}, \{Chi, Mil\}, \{Che, Bre\}, \{Che, Bre, Mil\}]$

Support – Example

ID					...		Pas
1	2 (true)	0 (false)	0 (false)	3 (true)	0 (false)	2 (true)	
2	0 (false)	0 (false)	1 (true)	1 (true)	0 (false)	0 (false)	
3	2 (true)	1 (true)	0 (false)	0 (false)	0 (false)	0 (false)	
4	1 (true)	1 (true)	0 (false)	1 (true)	0 (false)	0 (false)	

$\mathcal{X} \in \mathbb{M}(\mathbb{P}(\mathcal{I}))$

Dataset $\mathcal{X} = [\{Che, Mil, Pas\}, \{Chi, Mil\}, \{Che, Bre\}, \{Che, Bre, Mil\}]$

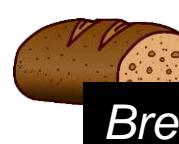
Itemset $\mathcal{A} = \{Che, Mil\} \subseteq \mathcal{I}$

$$\text{support_count}(\mathcal{A}) = |[\mathcal{T} \in \mathcal{X} \mid \mathcal{A} \subseteq \mathcal{T}]| = |[\mathcal{T}_1, \mathcal{T}_4]| = 2$$

$$\text{support}(\mathcal{A}) = \frac{|[\mathcal{T} \in \mathcal{X} \mid \mathcal{A} \subseteq \mathcal{T}]|}{|\mathcal{X}|} = \frac{|[\mathcal{T}_1, \mathcal{T}_4]|}{4} = \frac{2}{4}$$

\mathcal{A} is **frequent** if $\text{min_sup} \leq 0.5$

Support – Example

ID					...		Pas
1	2 (true)	0 (false)	0 (false)	3 (true)	0 (false)	2 (true)	
2	0 (false)	0 (false)	1 (true)	1 (true)	0 (false)	0 (false)	
3	2 (true)	1 (true)	0 (false)	0 (false)	0 (false)	0 (false)	
4	1 (true)	1 (true)	0 (false)	1 (true)	0 (false)	0 (false)	

$\mathcal{X} \in \mathbb{M}(\mathbb{P}(\mathcal{I}))$

Dataset $\mathcal{X} = [\{Che, Mil, Pas\}, \{Chi, Mil\}, \{Che, Bre\}, \{Che, Bre, Mil\}]$

Itemset $\mathcal{A} = \{Che, Mil\} \subseteq \mathcal{I}$

$$\text{support_count}(\mathcal{A}) = |[\mathcal{T} \in \mathcal{X} \mid \mathcal{A} \subseteq \mathcal{T}]| = |[\mathcal{T}_1, \mathcal{T}_4]| = 2$$

$$\text{support}(\mathcal{A}) = \frac{|[\mathcal{T} \in \mathcal{X} \mid \mathcal{A} \subseteq \mathcal{T}]|}{|\mathcal{X}|} = \frac{|[\mathcal{T}_1, \mathcal{T}_4]|}{4} = \frac{2}{4}$$

\mathcal{A} is **frequent** if $\text{min_sup} \leq 0.5$

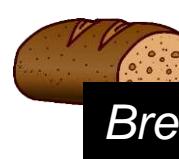
Itemset $\mathcal{B} = \{Mil\} \subseteq \mathcal{I}$

$$\text{support_count}(\mathcal{B}) = |[\mathcal{T} \in \mathcal{X} \mid \mathcal{B} \subseteq \mathcal{T}]| = |[\mathcal{T}_1, \mathcal{T}_2, \mathcal{T}_4]| = 3$$

$$\text{support}(\mathcal{B}) = \frac{|[\mathcal{T} \in \mathcal{X} \mid \mathcal{B} \subseteq \mathcal{T}]|}{|\mathcal{X}|} = \frac{|[\mathcal{T}_1, \mathcal{T}_2, \mathcal{T}_4]|}{4} = \frac{3}{4}$$

\mathcal{B} is **frequent** if $\text{min_sup} \leq 0.75$

Support – Example

ID					...		Pas
1	2 (true)	0 (false)	0 (false)	3 (true)	0 (false)	2 (true)	
2	0 (false)	0 (false)	1 (true)	1 (true)	0 (false)	0 (false)	
3	2 (true)	1 (true)	0 (false)	0 (false)	0 (false)	0 (false)	
4	1 (true)	1 (true)	0 (false)	1 (true)	0 (false)	0 (false)	

$\mathcal{X} \in \mathbb{M}(\mathbb{P}(\mathcal{I}))$

Itemset $\mathcal{A} = \{Che, Mil\} \subseteq \mathcal{I}$

$$\text{support}(\mathcal{A}) = \frac{|[\mathcal{T} \in \mathcal{X} | \mathcal{A} \subseteq \mathcal{T}]|}{|\mathcal{X}|} = \frac{|[\mathcal{T}_1, \mathcal{T}_4]|}{4} = \frac{2}{4}$$

Itemset $\mathcal{B} = \{Mil\} \subseteq \mathcal{I}$

$$\text{support}(\mathcal{B}) = \frac{|[\mathcal{T} \in \mathcal{X} | \mathcal{B} \subseteq \mathcal{T}]|}{|\mathcal{X}|} = \frac{|[\mathcal{T}_1, \mathcal{T}_2, \mathcal{T}_4]|}{4} = \frac{3}{4}$$

$$\mathcal{B} \subseteq \mathcal{A} \implies \text{support}(\mathcal{B}) \geq \text{support}(\mathcal{A})$$

General rule:
Subset of an itemset has higher or equal support than this itemset

Support – Summary

Support

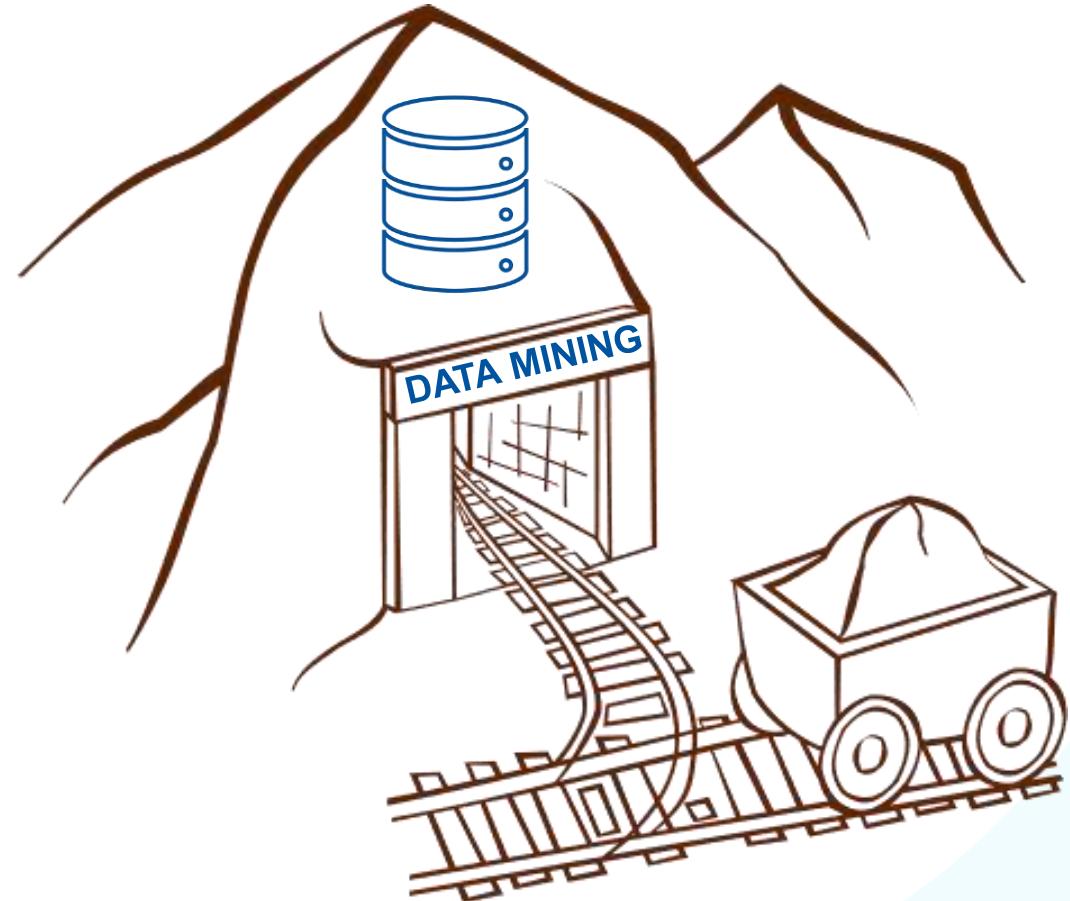
- A measure of the popularity (frequency) of an itemset.
- Calculated as the fraction of transactions in a dataset that contain the itemset.

$$\text{support}(\mathcal{A}) = \frac{|[\mathcal{T} \in \mathcal{X} | \mathcal{A} \subseteq \mathcal{T}]|}{|\mathcal{X}|}$$

- Any itemset with a support below the threshold is considered to be infrequent.
- Support is also used to find association rules

Frequent Itemsets

1. Introduction
2. **Properties of Frequent Itemsets**
3. A-Priori Algorithm
4. FP-Growth Algorithm



Problem Statement

Given dataset $\mathcal{X} \in \mathbb{M}(\mathbb{P}(\mathcal{I}))$ and minimum support threshold min_sup ,
find **all frequent non-empty itemsets**:

$$\{\mathcal{A} \subseteq \mathcal{I} \mid \text{support}(\mathcal{A}) \geq \text{min_sup}\}$$

Naïve Approach

- Given $\mathcal{A} \subseteq \mathcal{I}$, it is possible to check whether $\text{support}(\mathcal{A}) \geq \text{min_sup}$ by testing all transactions
- If there are D unique items, then there are $2^D - 1$ candidate itemsets that can all be tested individually
- However, this can be very time consuming...

Assume $D = 50\,000$ products

$$2^D - 1 =$$

Good luck
Of course most of them are not
real ones.

Subsets of Frequent Itemsets Are Also Frequent

- Assume $\mathcal{A} = \{I_1, I_2, \dots, I_{100}\}$ and $\text{support}(\mathcal{A}) \geq \text{min_sup}$
- All subsets of \mathcal{A} are also frequent
- There are $\binom{100}{1} = 100$ frequent itemsets having 1 item
- There are $\binom{100}{k}$ frequent itemsets having k items (“100 choose k “)
- There are $\binom{100}{1} + \binom{100}{2} + \dots + \binom{100}{99} = 2^{100} - 2 = 1.27 \times 10^{30}$ smaller frequent itemsets contained in \mathcal{A}

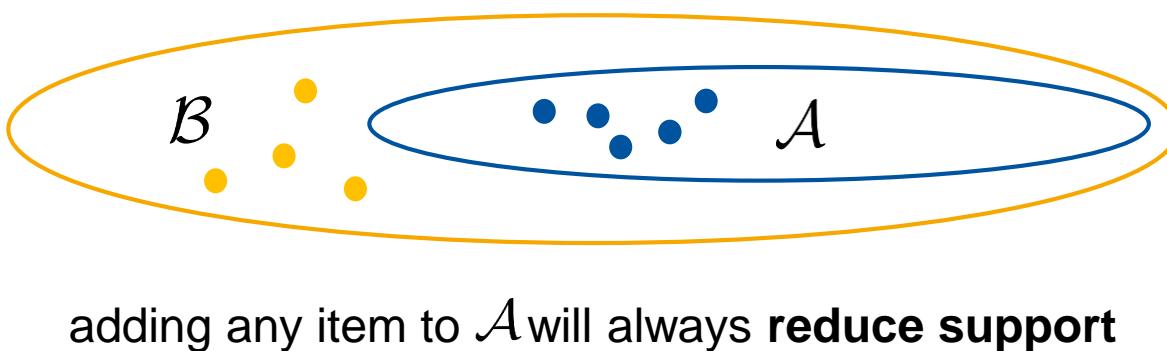
$$\binom{n}{k} = \frac{n!}{k!(n-k)!} = \frac{n(n-1)\dots(n-k+1)}{k(k-1)\dots1}$$

Summary

- We should **avoid** exhaustively testing **all** candidate itemsets
 - We need to focus on the “interesting” ones
- **Closed itemsets**

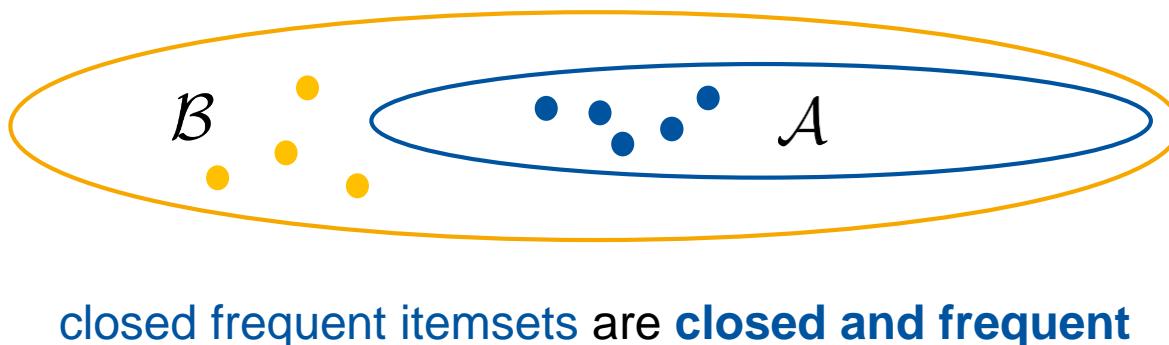
Closed Itemsets

- An itemset \mathcal{A} is **closed** if there is no proper superset $\mathcal{B} \supset \mathcal{A}$ that has the same support
- If \mathcal{A} is **closed**, then $\text{support}(\mathcal{A}) > \text{support}(\mathcal{B})$ for any $\mathcal{B} \supset \mathcal{A}$



Closed Frequent Itemsets

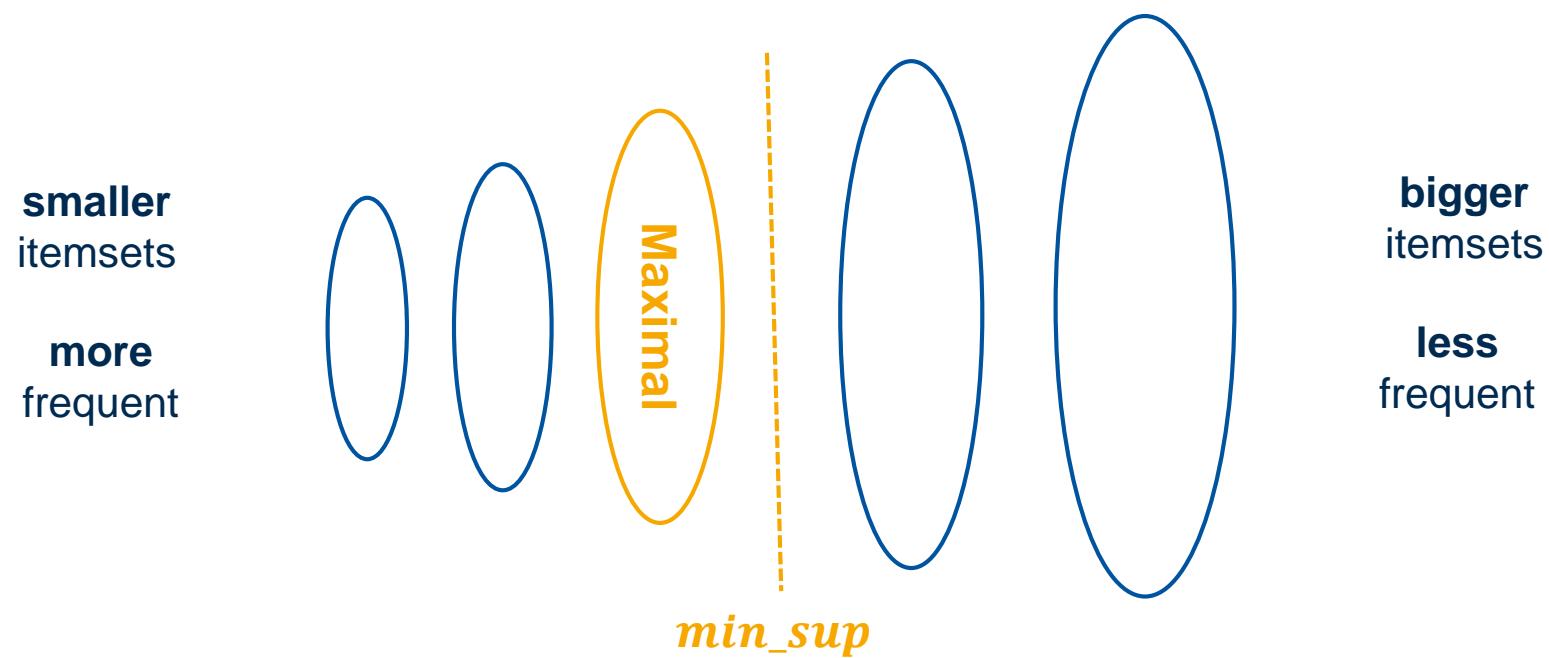
- An itemset \mathcal{A} is **closed** if there is no proper superset $\mathcal{B} \supset \mathcal{A}$ that has the same support
- If \mathcal{A} is **closed**, then $\text{support}(\mathcal{A}) > \text{support}(\mathcal{B})$ for any $\mathcal{B} \supset \mathcal{A}$
- \mathcal{A} is **frequent** if its support is higher than threshold min_sup



Maximal Frequent Itemsets

An itemset \mathcal{A} is a maximal frequent itemset if:

- \mathcal{A} is frequent
- there is no proper superset $\mathcal{B} \supset \mathcal{A}$ that is also frequent



Relationships

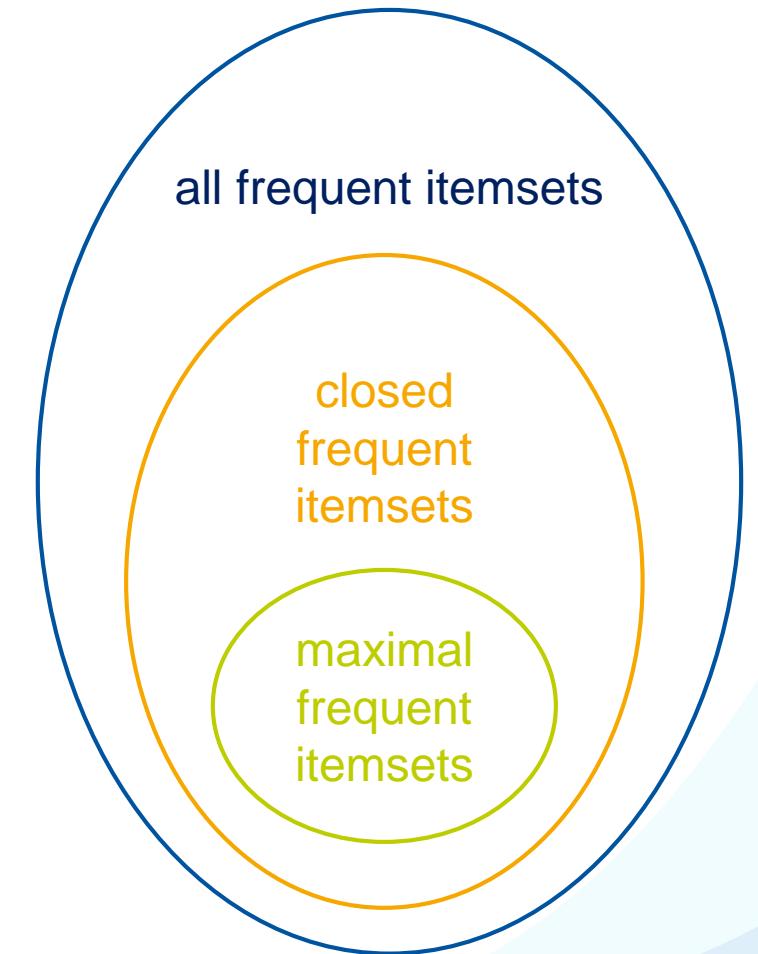
An itemset \mathcal{A} is a **closed frequent itemset** if:

- \mathcal{A} is frequent
- there is no proper superset $\mathcal{B} \supset \mathcal{A}$ that has the same support

An itemset \mathcal{A} is a **maximal frequent itemset** if:

- \mathcal{A} is frequent
- there is no proper superset $\mathcal{B} \supset \mathcal{A}$ that is also frequent

Hence, maximal frequent itemsets are closed by definition.



Example

Assume:

$$\mathcal{I} = \{I_1, I_2, \dots, I_{100}\}, \min_sup = \frac{5}{20} = 0.25$$
$$\mathcal{X} = [\{I_1, I_2, \dots, I_{50}\}^{10}, \{I_1, I_2, \dots, I_{100}\}^{10}]$$

- There are $2^{100} - 1 = 1.27 \times 10^{30}$ itemsets; all are frequent.
- There are two closed frequent itemsets:
 - $\mathcal{A} = \{I_1, I_2, \dots, I_{50}\}$ with $\text{support}(\mathcal{A}) = \frac{20}{20}$
 - $\mathcal{B} = \{I_1, I_2, \dots, I_{100}\}$ with $\text{support}(\mathcal{B}) = \frac{10}{20}$
- There is only one maximal frequent itemset: $\mathcal{B} = \{I_1, I_2, \dots, I_{100}\}$

Example

Assume:

$$\mathcal{I} = \{I_1, I_2, \dots, I_{100}\}, \min_sup = \frac{15}{20} = 0.75$$
$$\mathcal{X} = [\{I_1, I_2, \dots, I_{50}\}^{10}, \{I_1, I_2, \dots, I_{100}\}^{10}]$$

- There are $2^{50} - 1 = 3.17 \times 10^{15}$ itemsets that are frequent.
- There is one closed frequent itemsets:
 - $\mathcal{A} = \{I_1, I_2, \dots, I_{50}\}$ with $\text{support}(\mathcal{A}) = \frac{20}{20}$
- There is only one maximal frequent itemset: $\mathcal{A} = \{I_1, I_2, \dots, I_{50}\}$

Example

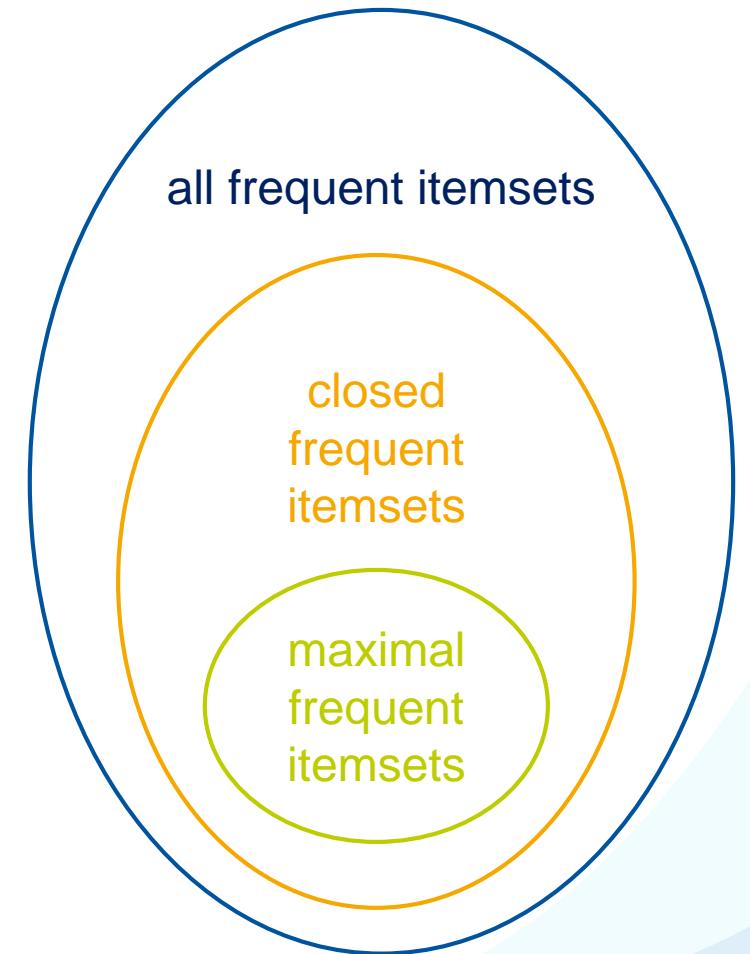
Assume:

$$\mathcal{I} = \{I_1, I_2, \dots, I_{100}\}, \min_sup = \frac{15}{20}$$
$$\mathcal{X} = [\{I_1, I_2, \dots, I_{99}\}^{10}, \{I_2, I_3, \dots, I_{100}\}^{10}]$$

- There are $2^{98} - 1 = 1.17 \times 10^{29}$ itemsets that are frequent.
- There is one closed frequent itemset:
 - $\mathcal{A} = \{I_2, I_3, \dots, I_{99}\}$ with $\text{support}(\mathcal{A}) = \frac{20}{20}$
- There is only one maximal frequent itemset: $\mathcal{A} = \{I_2, I_3, \dots, I_{99}\}$

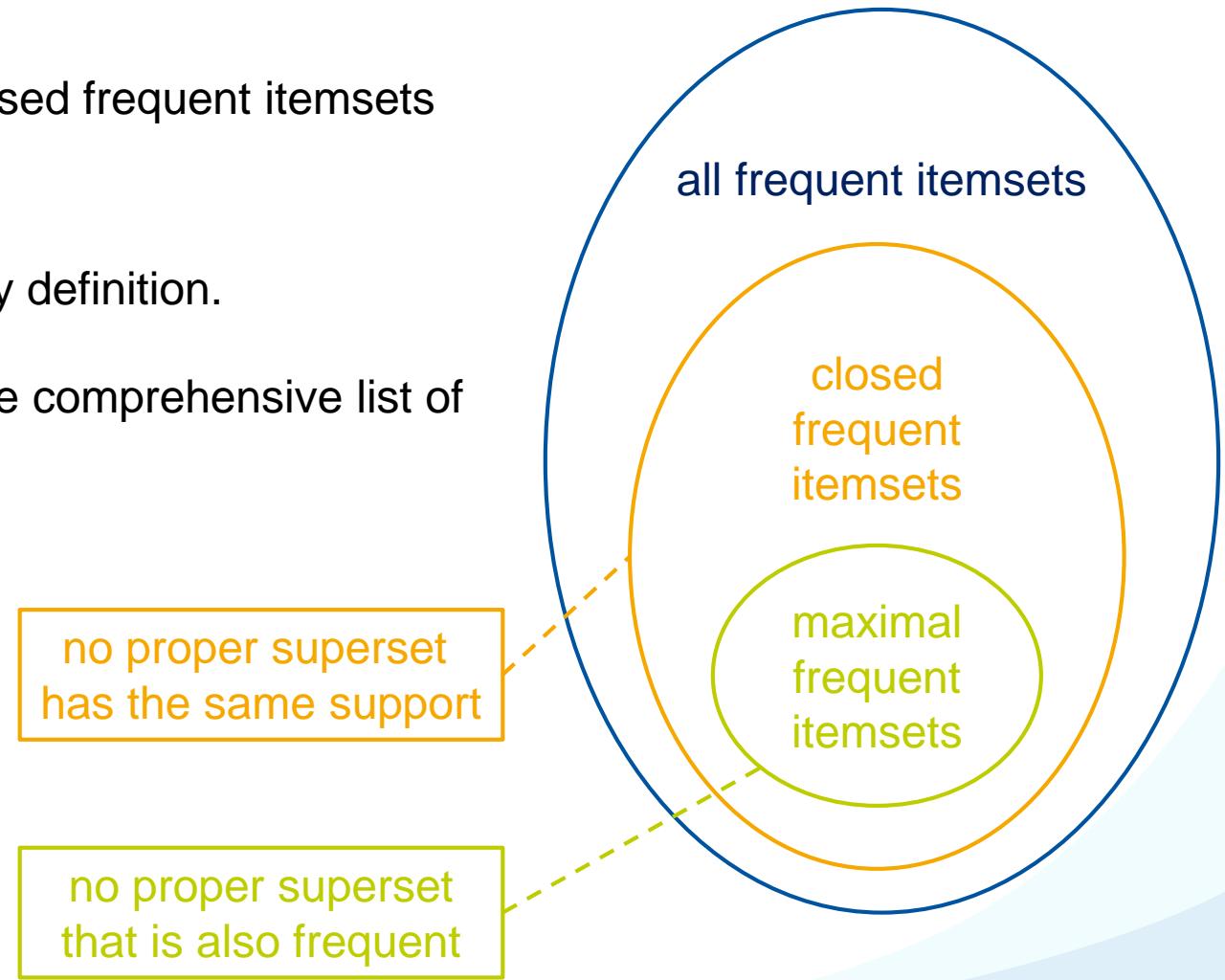
Observations

- The supports of the **closed** frequent itemsets provide complete information about the supports of **all** frequent item sets
 - Formally, assume:
 - $\mathcal{A} \subset \mathcal{B}$,
 - \mathcal{B} is a **closed** frequent itemset, and
 - there is no **closed** frequent itemset \mathcal{B}' such that $\mathcal{A} \subseteq \mathcal{B}' \subset \mathcal{B}$.Then $support(\mathcal{A}) = support(\mathcal{B})$.
- It suffices to store **closed** frequent itemsets
(**maximal** frequent itemsets provide less information)



Summary

- Both maximal frequent itemsets and closed frequent itemsets are subsets of frequent itemsets.
- Maximal frequent itemsets are closed by definition.
- Closed frequent itemsets provide a more comprehensive list of frequent patterns



Frequent Itemsets

1. Introduction
2. Properties of Frequent Itemsets
3. **Apriori Algorithm**
4. FP-Growth Algorithm



Apriori Algorithm

- Introduced by Rakesh Agrawal and Ramakrishnan Srikant in “Fast Algorithms for Mining Association Rules in Large Databases. VLDB 1994: 487-499”
- Computes frequent itemsets / association rules in a dataset
- Uses a “bottom up” approach (starts with candidate itemsets of size one)
- Extends frequent subsets one item at a time (candidate generation)
- Avoids unnecessary checks by re-using information from smaller subsets and exploiting frequent itemsets’ properties

Apriori Algorithm – Basic Idea

$$\mathcal{L}_k = \{\mathcal{A} \subseteq \mathcal{I} \mid \text{support}(\mathcal{A}) \geq \text{min_sup} \wedge |\mathcal{A}| = k\}$$

frequent itemsets of length k

1. Candidate generation: use the set \mathcal{L}_k of frequent itemsets of length k to generate the candidate set \mathcal{C}_{k+1} of candidate itemsets with length $k+1$

Apriori Algorithm – Basic Idea

$$\mathcal{L}_k = \{\mathcal{A} \subseteq \mathcal{I} \mid \text{support}(\mathcal{A}) \geq \text{min_sup} \wedge |\mathcal{A}| = k\}$$

frequent itemsets of length k

1. Candidate generation: use the set \mathcal{L}_k of frequent itemsets of length k to generate the candidate set \mathcal{C}_{k+1} of candidate itemsets with length $k+1$

does not
need the
input data
(efficient)

Apriori Algorithm – Basic Idea

$$\mathcal{L}_k = \{\mathcal{A} \subseteq \mathcal{I} \mid \text{support}(\mathcal{A}) \geq \text{min_sup} \wedge |\mathcal{A}| = k\}$$

frequent itemsets of length k

1. **Candidate generation:** use the set \mathcal{L}_k of frequent itemsets of length k to generate the candidate set \mathcal{C}_{k+1} of candidate itemsets with length $k+1$
2. **Pruning (antimonotonicity):** all nonempty subsets of a frequent itemset must also be frequent → superset of an infrequent itemset cannot be frequent

does not
need the
input data
(efficient)

Apriori Algorithm – Basic Idea

$$\mathcal{L}_k = \{\mathcal{A} \subseteq \mathcal{I} \mid \text{support}(\mathcal{A}) \geq \text{min_sup} \wedge |\mathcal{A}| = k\}$$

frequent itemsets of length k

1. Candidate generation: use the set \mathcal{L}_k of frequent itemsets of length k to generate the candidate set \mathcal{C}_{k+1} of candidate itemsets with length $k+1$
2. Pruning (antimonotonicity): all nonempty subsets of a frequent itemset must also be frequent → superset of an infrequent itemset cannot be frequent

does not
need the
input data
(efficient)

3. Testing candidates: use the dataset to filter the infrequent itemsets from \mathcal{C}_{k+1} and obtain \mathcal{L}_{k+1}

needs the
input data
(inefficient)

Apriori Algorithm – Basic Idea

$$\mathcal{L}_k = \{\mathcal{A} \subseteq \mathcal{I} \mid \text{support}(\mathcal{A}) \geq \text{min_sup} \wedge |\mathcal{A}| = k\}$$

frequent itemsets of length k

1. Candidate generation: use the set \mathcal{L}_k of frequent itemsets of length k to generate the candidate set \mathcal{C}_{k+1} of candidate itemsets with length $k+1$
2. Pruning (antimonotonicity): all nonempty subsets of a frequent itemset must also be frequent → superset of an infrequent itemset cannot be frequent
3. Testing candidates: use the dataset to filter the infrequent itemsets from \mathcal{C}_{k+1} and obtain \mathcal{L}_{k+1}

does not
need the
input data
(efficient)

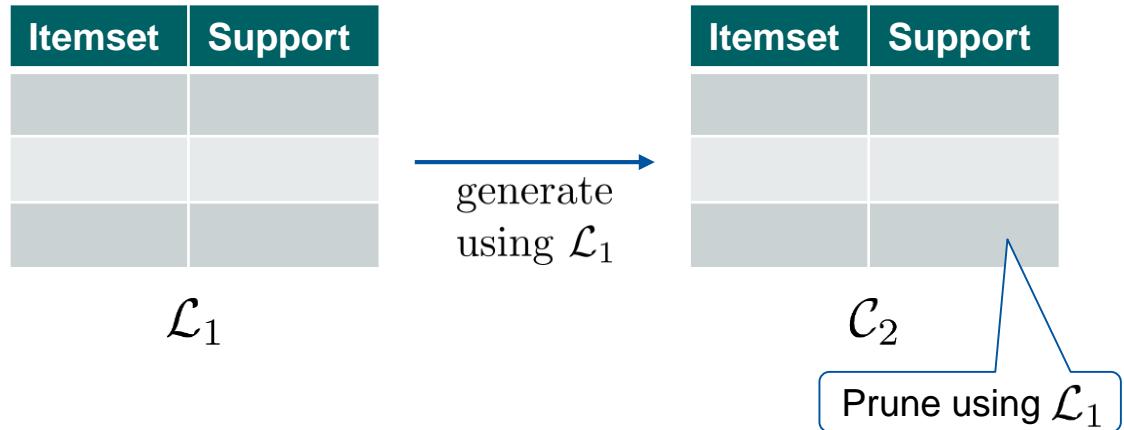
needs the
input data
(inefficient)

Apriori Algorithm – Basic Idea

Itemset	Support

\mathcal{L}_1

Apriori Algorithm – Basic Idea



Apriori Algorithm – Basic Idea

Itemset	Support

 \mathcal{L}_1

generate
using \mathcal{L}_1

Itemset	Support

 \mathcal{C}_2

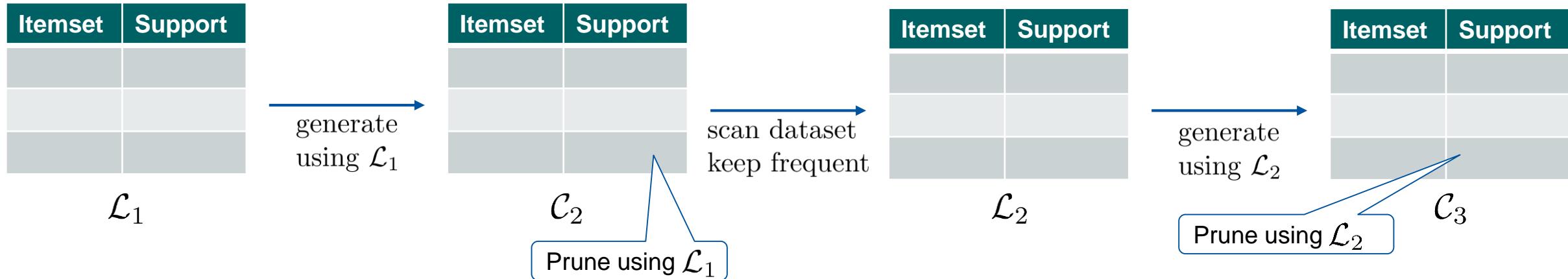
Prune using \mathcal{L}_1

Itemset	Support

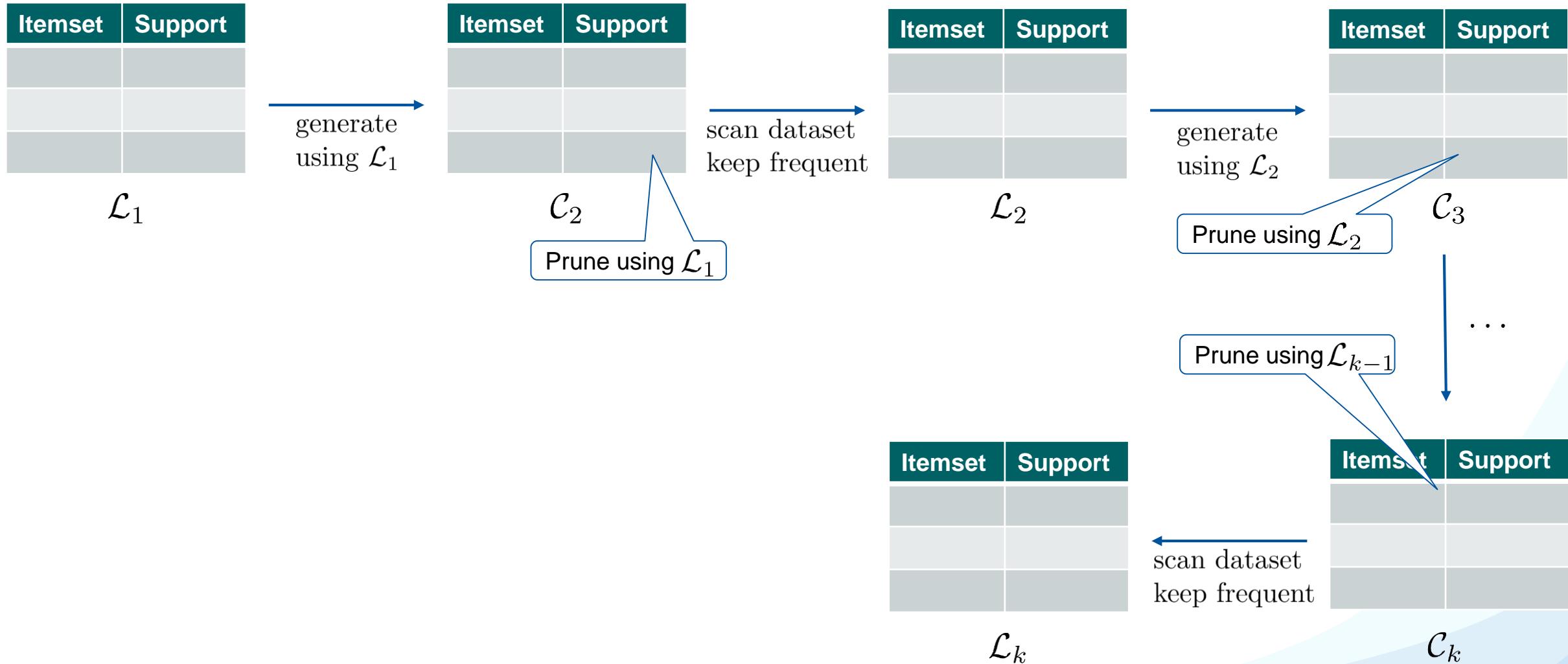
 \mathcal{L}_2

scan dataset
keep frequent

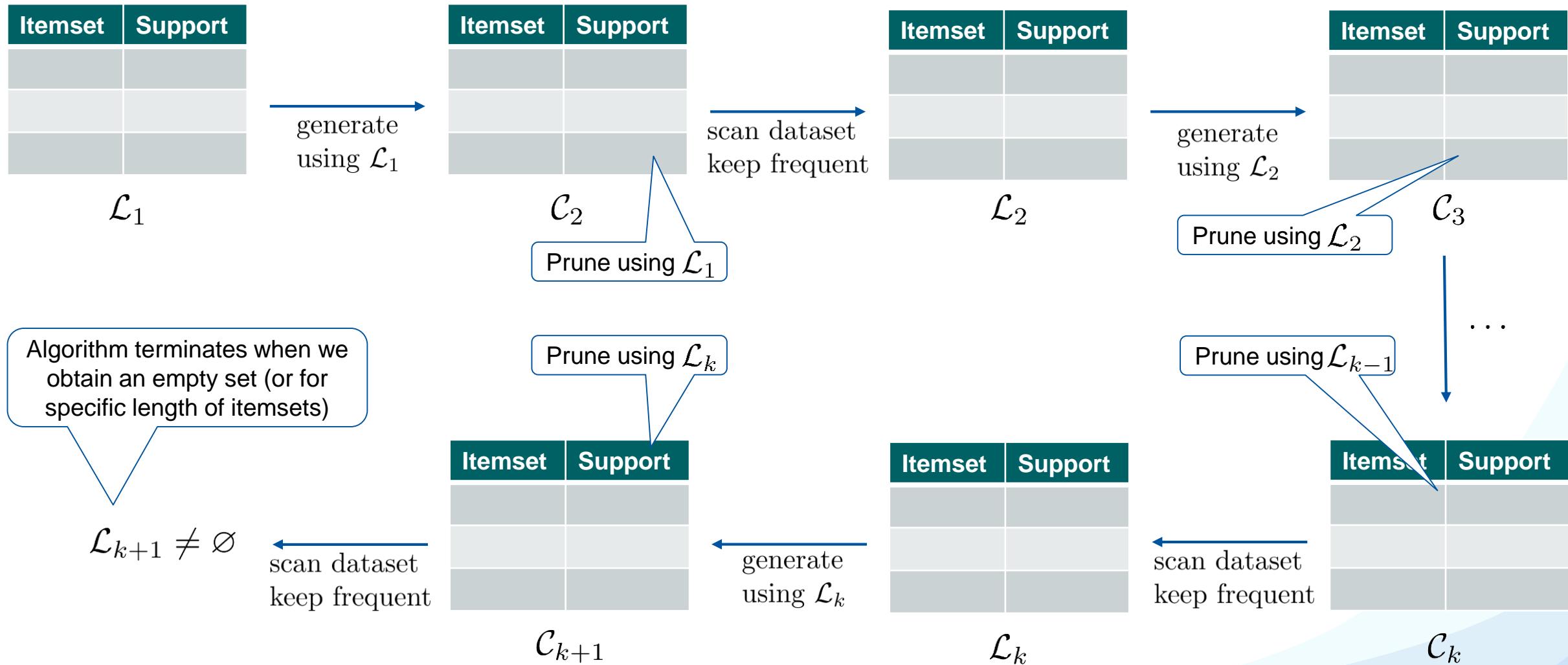
Apriori Algorithm – Basic Idea



Apriori Algorithm – Basic Idea



Apriori Algorithm – Basic Idea



Candidate Generation – Leveling

Leveling is used to generate candidate itemset \mathcal{C}_{k+1} from \mathcal{L}_k :

For any $\mathcal{A} \in \mathcal{L}_{k+1}$ there exist $\mathcal{A}', \mathcal{A}'' \in \mathcal{L}_k$ such that $\mathcal{A} = \mathcal{A}' \cup \mathcal{A}''$

Hence, we can obtain the candidate itemsets $\mathcal{C}_{k+1} \supseteq \mathcal{L}_{k+1}$ by joining suitable $\mathcal{A}', \mathcal{A}'' \in \mathcal{L}_k$!

all itemsets of length $k + 1$

\mathcal{C}_{k+1} before pruning

\mathcal{C}_{k+1} after pruning

\mathcal{L}_{k+1}

Candidate Generation – Leveling

Leveling is used to generate candidate itemset \mathcal{C}_{k+1} from \mathcal{L}_k :

For any $\mathcal{A} \in \mathcal{L}_{k+1}$ there exist $\mathcal{A}', \mathcal{A}'' \in \mathcal{L}_k$ such that $\mathcal{A} = \mathcal{A}' \cup \mathcal{A}''$

Assume that the items are ordered (I_1, I_2, \dots) and that
 $\mathcal{A} = \{I_1, I_2, \dots, I_{k-1}, I_k, I_{k+1}\} \in \mathcal{L}_{k+1}$

If \mathcal{A} is frequent, its subsets must be frequent, in particular:

$$\mathcal{A}' = \{I_1, I_2, \dots, I_{k-1}, I_k\} \in \mathcal{L}_k$$

$$\mathcal{A}'' = \{I_1, I_2, \dots, I_{k-1}, I_{k+1}\} \in \mathcal{L}_k$$

$$\mathcal{A}' \cup \mathcal{A}'' = \{I_1, I_2, \dots, I_{k-1}, I_k, I_{k+1}\} \in \mathcal{L}_{k+1} = \mathcal{A}$$

all itemsets of length $k + 1$

\mathcal{C}_{k+1} before pruning

\mathcal{C}_{k+1} after pruning

\mathcal{L}_{k+1}

Candidate Generation – Leveling

Leveling is used to generate candidate itemset \mathcal{C}_{k+1} from \mathcal{L}_k :

For any $\mathcal{A} \in \mathcal{L}_{k+1}$ there exist $\mathcal{A}', \mathcal{A}'' \in \mathcal{L}_k$ such that $\mathcal{A} = \mathcal{A}' \cup \mathcal{A}''$

Assume that the items are ordered (I_1, I_2, \dots) and that
 $\mathcal{A} = \{I_1, I_2, \dots, I_{k-1}, I_k, I_{k+1}\} \in \mathcal{L}_{k+1}$

If \mathcal{A} is frequent, its subsets must be frequent, in particular:

$$\mathcal{A}' = \{I_1, I_2, \dots, I_{k-1}, I_k\} \in \mathcal{L}_k$$

$$\mathcal{A}'' = \{I_1, I_2, \dots, I_{k-1}, I_{k+1}\} \in \mathcal{L}_k$$

$$\mathcal{A}' \cup \mathcal{A}'' = \{I_1, I_2, \dots, I_{k-1}, I_k, I_{k+1}\} \in \mathcal{L}_{k+1} = \mathcal{A}$$

\Rightarrow We can generate \mathcal{C}_{k+1} by joining itemsets $\mathcal{A}', \mathcal{A}'' \in \mathcal{L}_k$ which differ in one item

all itemsets of length $k + 1$

\mathcal{C}_{k+1} before pruning

\mathcal{C}_{k+1} after pruning

\mathcal{L}_{k+1}

Candidate Generation

Thanks to **leveling**:

- Apriori creates the set of **candidate itemsets of length $k+1$** , \mathcal{C}_{k+1} , by joining two frequent itemsets of length k
- This can be done efficiently without creating duplicates
- Next, we **prune** the set \mathcal{C}_{k+1} based on infrequent subsets

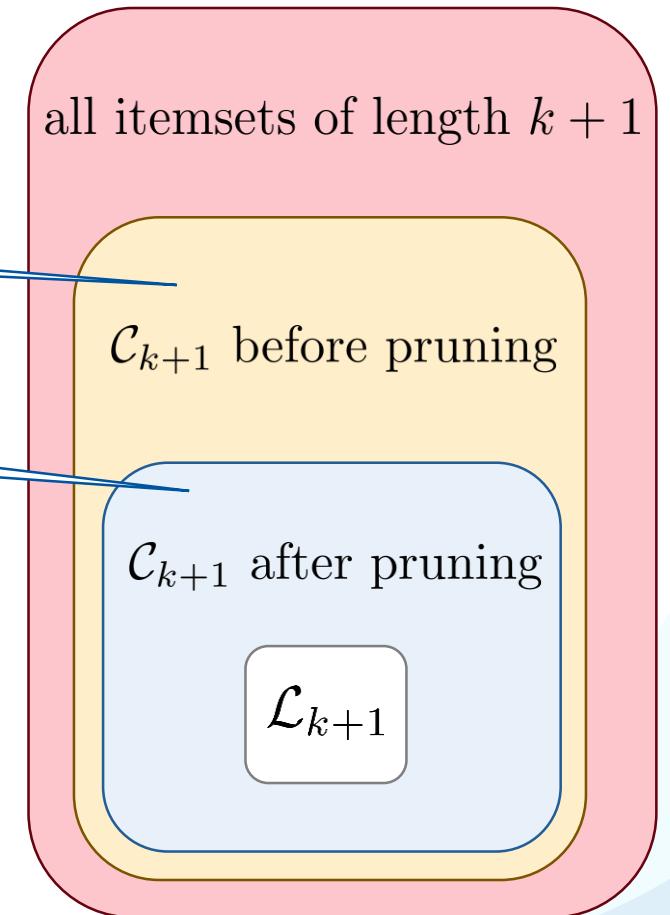
If \mathcal{A} is frequent, its subsets must be frequent, in particular:

$$\mathcal{A}' = \{I_1, I_2, \dots, I_{k-1}, I_k\} \in \mathcal{L}_k$$

$$\mathcal{A}'' = \{I_1, I_2, \dots, I_{k-1}, I_{k+1}\} \in \mathcal{L}_k$$

$$\mathcal{A}' \cup \mathcal{A}'' = \{I_1, I_2, \dots, I_{k-1}, I_k, I_{k+1}\} \in \mathcal{L}_{k+1} = \mathcal{A}$$

⇒ We can generate \mathcal{C}_{k+1} by joining itemsets $\mathcal{A}', \mathcal{A}'' \in \mathcal{L}_k$ which differ in one item



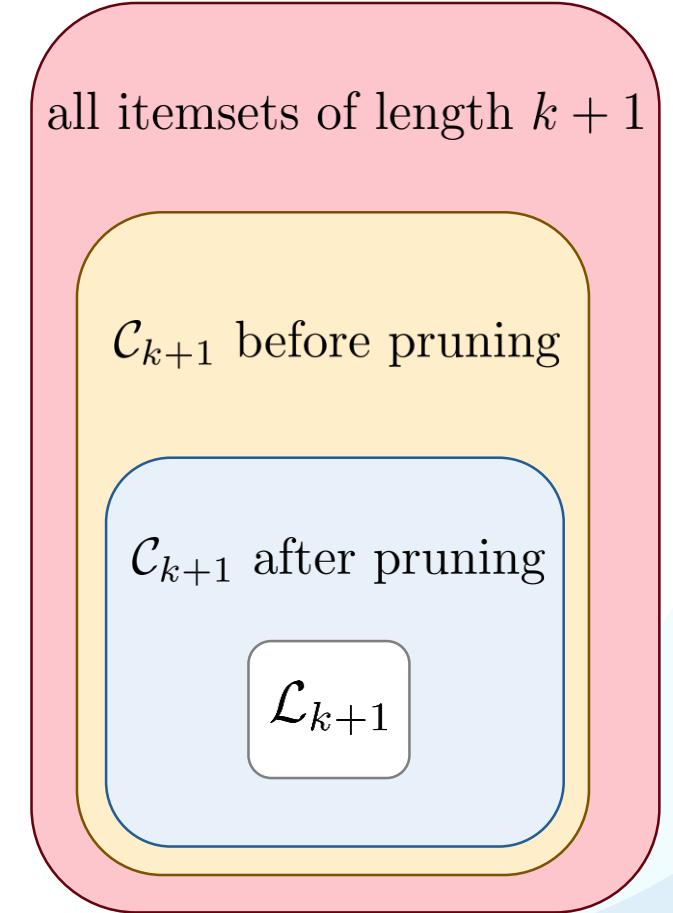
Pruning – Antimonotonicity

Antimonotonicity is used to prune the candidate set:

If \mathcal{B} is a frequent itemset, any subset $\mathcal{A} \subseteq \mathcal{B}$ must be frequent
 \Rightarrow If a subset $\mathcal{A} \subseteq \mathcal{B}$ is infrequent, then \mathcal{B} is infrequent

For any $\mathcal{A} \subseteq \mathcal{I}$ and $\mathcal{B} \subseteq \mathcal{I}$:

1. If $\mathcal{A} \subseteq \mathcal{B}$, then $\text{support}(\mathcal{A}) \geq \text{support}(\mathcal{B})$
2. If $\mathcal{A} \subseteq \mathcal{B}$ and $\text{support}(\mathcal{B}) \geq \text{min_sup}$,
then $\text{support}(\mathcal{A}) \geq \text{min_sup}$
3. If $\mathcal{A} \subseteq \mathcal{B}$ and $\text{support}(\mathcal{A}) < \text{min_sup}$,
then $\text{support}(\mathcal{B}) < \text{min_sup}$

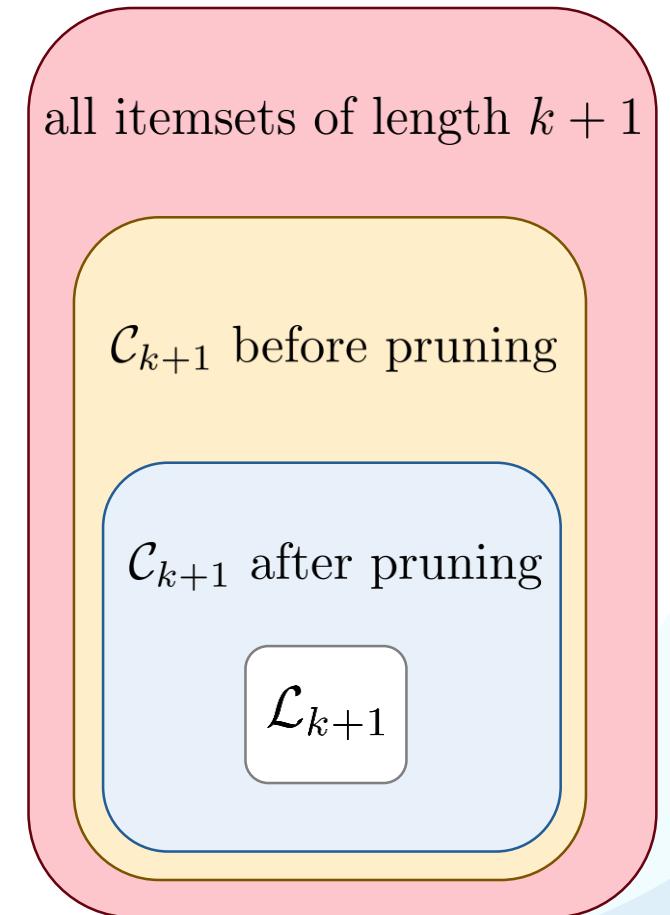


Pruning – Antimonotonicity

Antimonotonicity is used to prune the candidate set:

If \mathcal{B} is a frequent itemset, any subset $\mathcal{A} \subseteq \mathcal{B}$ must be frequent
 \Rightarrow If a subset $\mathcal{A} \subseteq \mathcal{B}$ is infrequent, then \mathcal{B} is infrequent

1. If $\{\text{Apple}\}$ or $\{\text{Banana}\}$ is **infrequent**,
then $\{\text{Apple}, \text{Banana}\}$ is **infrequent**
2. If $\{\text{Grapes}\}$ or $\{\text{Banana}\}$ is **infrequent**,
then $\{\text{Grapes}, \text{Banana}\}$ is **infrequent**
3. If $\{\text{Apples}\}$ or $\{\text{Grapes}\}$ is **infrequent**,
then $\{\text{Apples}, \text{Grapes}\}$ is **infrequent**



Pruning – Antimonotonicity

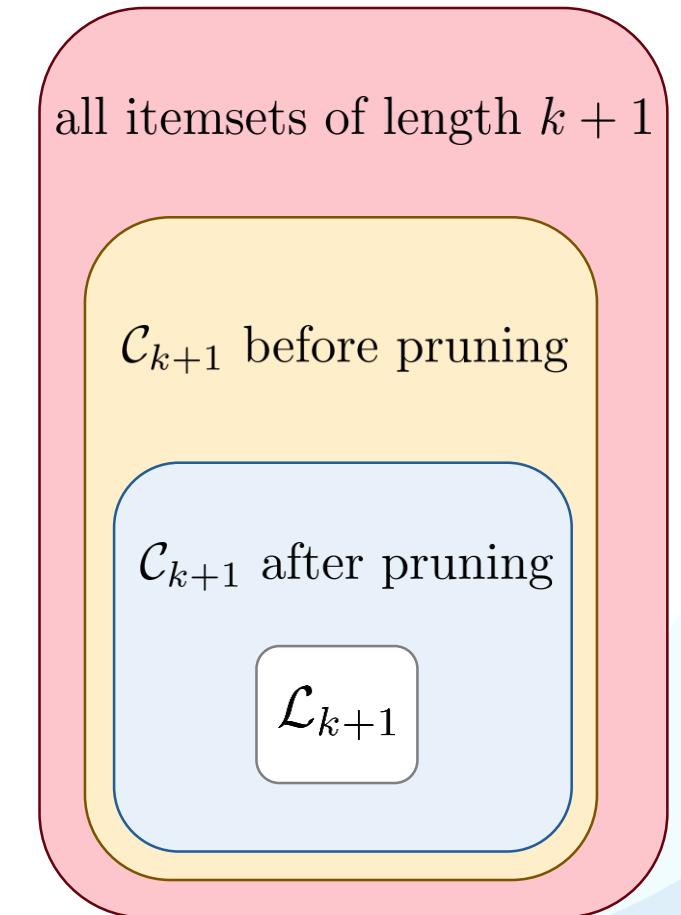
Antimonotonicity is used to prune the candidate set:

If \mathcal{B} is a frequent itemset, any subset $\mathcal{A} \subseteq \mathcal{B}$ must be frequent
 \Rightarrow If a subset $\mathcal{A} \subseteq \mathcal{B}$ is infrequent, then \mathcal{B} is infrequent

1. If {Apple} or {Banana} is infrequent, then {Apple, Banana} is infrequent
2. If {Grapes} or {Banana} is infrequent, then {Grapes, Banana} is infrequent
3. If {Apples} or {Grapes} is infrequent, then {Apples, Grapes} is infrequent

Bought Fruits	Support _count
{Banana}	4
{Grapes}	1
{Apple}	2

min_sup_count = 2



Pruning – Antimonotonicity

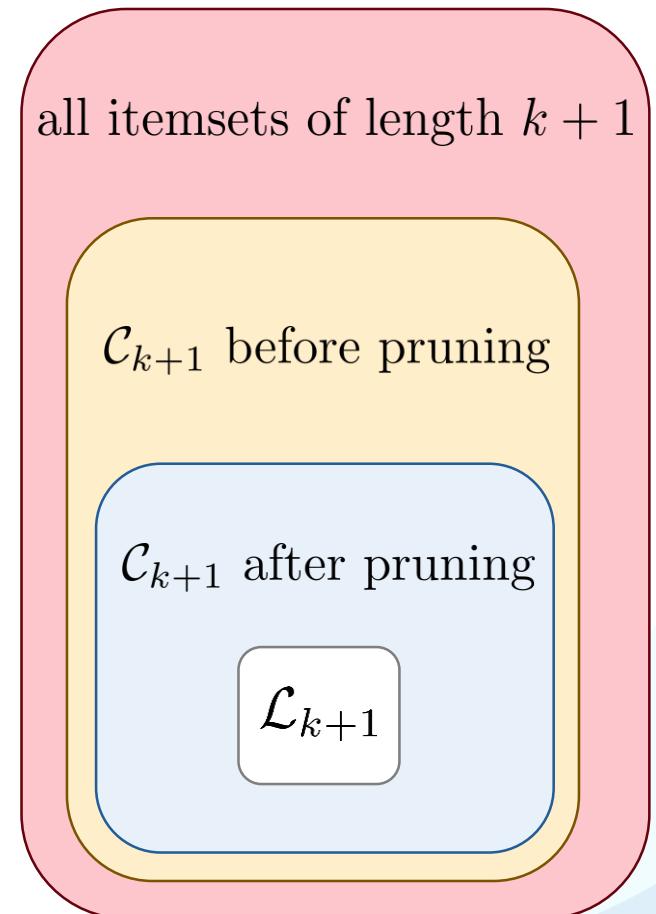
Antimonotonicity is used to **prune** the candidate set:

If \mathcal{B} is a frequent itemset, any subset $\mathcal{A} \subseteq \mathcal{B}$ must be frequent
 \Rightarrow If a subset $\mathcal{A} \subseteq \mathcal{B}$ is infrequent, then \mathcal{B} is infrequent

ID	Bought Fruits
1	{Banana, Apple}
2	{Grapes, Banana}
3	{Apple, Grapes}

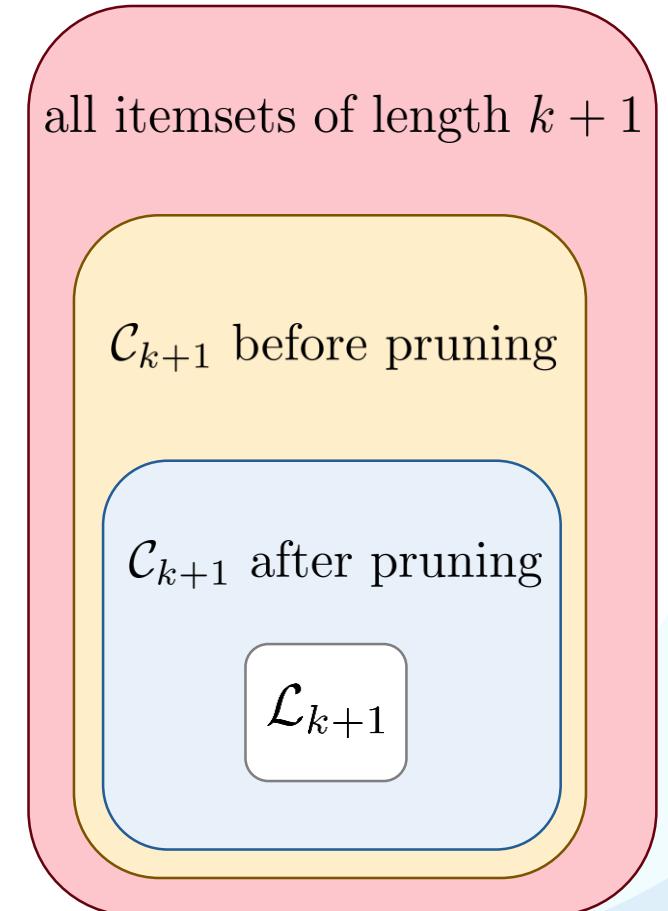


ID	Bought Fruits
1	{Banana, Apple}
2	{ Grapes, Banana }
3	{ Apple, Grapes }



Testing Candidates

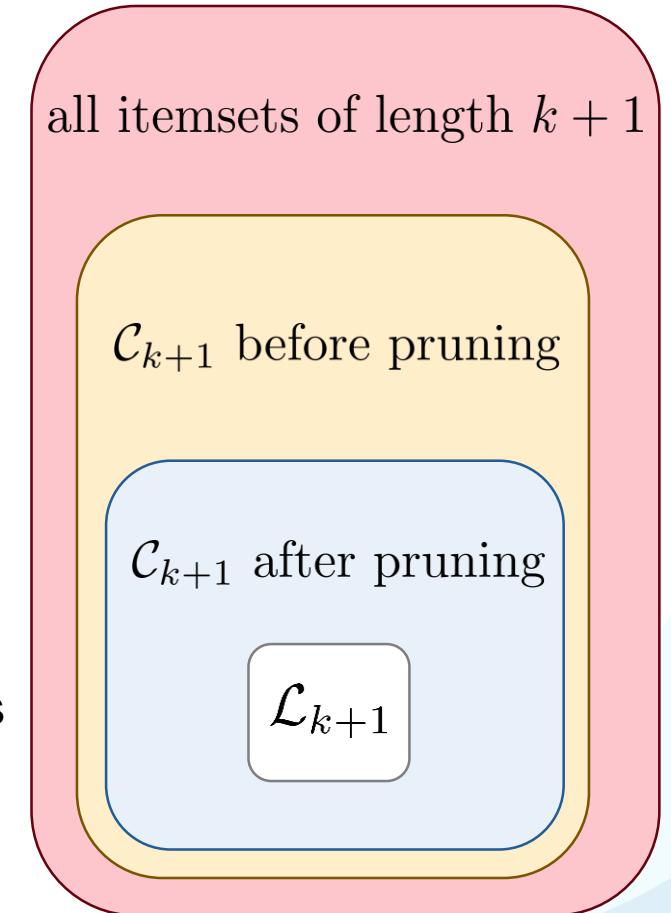
- After candidate generation and pruning test the remaining candidate itemsets
- We scan the dataset \mathcal{X} and remove all infrequent candidate itemsets from \mathcal{C}_{k+1} to obtain \mathcal{L}_{k+1}



Testing Candidates

- After candidate generation and pruning test the remaining candidate itemsets
- We scan the dataset \mathcal{X} and remove all infrequent candidate itemsets from \mathcal{C}_{k+1} to obtain \mathcal{L}_{k+1}
- Consider all transactions $\mathcal{T} \in \mathcal{X} \in \mathbb{M}(\mathbb{P}(\mathcal{I}))$
- For each candidate itemset $\mathcal{A} \in \mathcal{C}_k$ increment the corresponding counter if $\mathcal{A} \subseteq \mathcal{T}_k$
- This returns the frequencies (**support_count**) of the candidate itemsets and we can compute \mathcal{L}_{k+1} from \mathcal{C}_{k+1}

$$\mathcal{L}_{k+1} = \{\mathcal{A} \in \mathcal{C}_{k+1} \mid \text{support}(\mathcal{A}) \geq \text{min_sup}\}$$



Algorithm

Apriori algorithm:

1. Find the frequent itemsets of length 1 (\mathcal{L}_1)

2. Let $k \leftarrow 1$

3. **Repeat until** $\mathcal{L}_k \neq \emptyset$:

(a) Generate set of candidate itemsets of length $k+1$ (\mathcal{C}_{k+1})
based on \mathcal{L}_k

(b) Prune \mathcal{C}_{k+1} from itemsets that have infrequent subsets

(c) Test all remaining candidates from \mathcal{C}_{k+1} and remove
infrequent itemsets to obtain \mathcal{L}_{k+1}

(d) Assign $k \leftarrow k + 1$

(Or until we find
frequent itemsets of
pre-defined length)

all itemsets of length $k + 1$

\mathcal{C}_{k+1} before pruning

\mathcal{C}_{k+1} after pruning

\mathcal{L}_{k+1}

Example

TID	Bought Fruits
1	{Grapes, Apple, Pineapple}
2	{Orange, Apple, Banana}
3	{Grapes, Orange, Apple, Banana}
4	{Orange, Banana}
5	{Grapes, Apple, Banana}

 \mathcal{X} 

Itemset	Count
{Grapes}	3
{Apple}	4
{Pineapple}	1
{Orange}	3
{Banana}	4

 \mathcal{C}_1

Example

TID	Bought Fruits
1	{Grapes, Apple, Pineapple}
2	{Orange, Apple, Banana}
3	{Grapes, Orange, Apple, Banana}
4	{Orange, Banana}
5	{Grapes, Apple, Banana}

 \mathcal{X} 

Itemset	Count
{Grapes}	3
{Apple}	4
{Pineapple}	1
{Orange}	3
{Banana}	4

 \mathcal{C}_1

$\text{min_sup_count} = 2$

Itemset	Count
{Grapes}	3
{Apple}	4
{Pineapple}	1
{Orange}	3
{Banana}	4

 \mathcal{L}_1

Itemset	Count
{Grapes}	3
{Apple}	4
{Pineapple}	1
{Orange}	3
{Banana}	4

Example

TID	Bought Fruits
1	{Grapes, Apple, Pineapple}
2	{Orange, Apple, Banana}
3	{Grapes, Orange, Apple, Banana}
4	{Orange, Banana}
5	{Grapes, Apple, Banana}

 \mathcal{X}

Itemset	Count
{Grapes}	3
{Apple}	4
{Pineapple}	1
{Orange}	3
{Banana}	4

Itemset	Count
{Grapes}	3
{Apple}	4
{Pineapple}	1
{Orange}	3
{Banana}	4

min_sup_count = 2

 \mathcal{L}_1 \mathcal{C}_1

Itemset
{Grapes, Apple}
{Grapes, Orange}
{Grapes, Banana}
{Apple, Orange}
{Apple, Banana}
{Orange, Banana}

generate candidates from \mathcal{L}_1

Itemset	Count
{Grapes, Apple}	3
{Grapes, Orange}	1
{Grapes, Banana}	2
{Apple, Orange}	2
{Apple, Banana}	3
{Orange, Banana}	3

pruning based on \mathcal{L}_1 \mathcal{C}_2 \mathcal{C}_2

Example

TID	Bought Fruits
1	{Grapes, Apple, Pineapple}
2	{Orange, Apple, Banana}
3	{Grapes, Orange, Apple, Banana}
4	{Orange, Banana}
5	{Grapes, Apple, Banana}

 \mathcal{X}

Itemset	Count
{Grapes}	3
{Apple}	4
{Pineapple}	1
{Orange}	3
{Banana}	4

 \mathcal{C}_1

Itemset	Count
{Grapes}	3
{Apple}	4
{Pineapple}	1
{Orange}	3
{Banana}	4

 \mathcal{L}_1

Itemset
{Grapes, Apple}
{Grapes, Orange}
{Grapes, Banana}
{Apple, Orange}
{Apple, Banana}
{Orange, Banana}

generate candidates from \mathcal{L}_1 pruning based on \mathcal{L}_1 \mathcal{C}_2

Itemset	Count
{Grapes, Apple}	3
{Grapes, Orange}	1
{Grapes, Banana}	2
{Apple, Orange}	2
{Apple, Banana}	3
{Orange, Banana}	3

 \mathcal{C}_2 $\text{min_sup_count} = 2$ scan dataset
test candidates

Itemset	Count
{Grapes, Apple}	3
{Grapes, Orange}	1
{Grapes, Banana}	2
{Apple, Orange}	2
{Apple, Banana}	3
{Orange, Banana}	3

 \mathcal{L}_2

Example

TID	Bought Fruits
1	{Grapes, Apple, Pineapple}
2	{Orange, Apple, Banana}
3	{Grapes, Orange, Apple, Banana}
4	{Orange, Banana}
5	{Grapes, Apple, Banana}

 \mathcal{X} 

...

min_sup_count = 2

Itemset	Count
{Grapes, Apple}	3
{Grapes, Banana}	2
{Apple, Orange}	2
{Apple, Banana}	3
{Orange, Banana}	3

 \mathcal{L}_2

Example

TID	Bought Fruits
1	{Grapes, Apple, Pineapple}
2	{Orange, Apple, Banana}
3	{Grapes, Orange, Apple, Banana}
4	{Orange, Banana}
5	{Grapes, Apple, Banana}

 χ

→ ... → $\min_sup_count = 2$

Itemset	Count
{Grapes, Apple}	3
{Grapes, Banana}	2
{Apple, Orange}	2
{Apple, Banana}	3
{Orange, Banana}	3

 \mathcal{L}_2

Itemset
{Grapes, Apple, Banana}
{Grapes, Apple, Orange}
{Grapes, Banana, Orange}
{Apple, Banana, Orange}

generate candidates from \mathcal{L}_2

Itemset	Subsets of Length 2
{Grapes, Apple, Banana}	{Grapes, Apple}, {Grapes, Banana}, {Apple, Banana}
{Grapes, Apple, Orange}	{Grapes, Apple}, {Grapes, Orange}, {Apple, Orange}
{Grapes, Banana, Orange}	{Grapes, Banana}, {Grapes, Orange}, {Banana, Orange}
{Apple, Banana, Orange}	{Apple, Banana}, {Apple, Orange}, {Banana, Orange}

pruning based on \mathcal{L}_2

Itemset	Count
{Grapes, Apple, Banana}	2
{Apple, Banana, Orange}	2

$\min_sup_count = 2$
scan dataset
test candidates

 \mathcal{L}_3 \mathcal{C}_3 \mathcal{C}_3

Example

TID	Bought Fruits
1	{Grapes, Apple, Pineapple}
2	{Orange, Apple, Banana}
3	{Grapes, Orange, Apple, Banana}
4	{Orange, Banana}
5	{Grapes, Apple, Banana}

 \mathcal{X} 

...

min_sup_count = 2

Itemset	Count
{Grapes, Apple, Banana}	2
{Apple, Banana, Orange}	2

 \mathcal{L}_3

Example

TID	Bought Fruits
1	{Grapes, Apple, Pineapple}
2	{Orange, Apple, Banana}
3	{Grapes, Orange, Apple, Banana}
4	{Orange, Banana}
5	{Grapes, Apple, Banana}

 \mathcal{X} 

min_sup_count = 2

Itemset	Count
{Grapes, Apple, Banana}	2
{Apple, Banana, Orange}	2

 \mathcal{L}_3

generate candidates from \mathcal{L}_3

Itemset
{Grapes, Apple, Banana, Orange}

 \mathcal{C}_4

pruning based on \mathcal{L}_3

Itemset	Subsets of length 3
{Grapes, Apple, Banana, Orange}	{Grapes, Apple, Banana}, {Grapes, Apple, Orange}, {Grapes, Banana, Orange}, {Apple, Banana, Orange}

No more candidates of length 4 – algorithm terminates

Optimizations

Further optimizations

- Distributing the data (can be done in various ways)
- Gradually removing transactions not containing any frequent itemset of length k
- Sampling

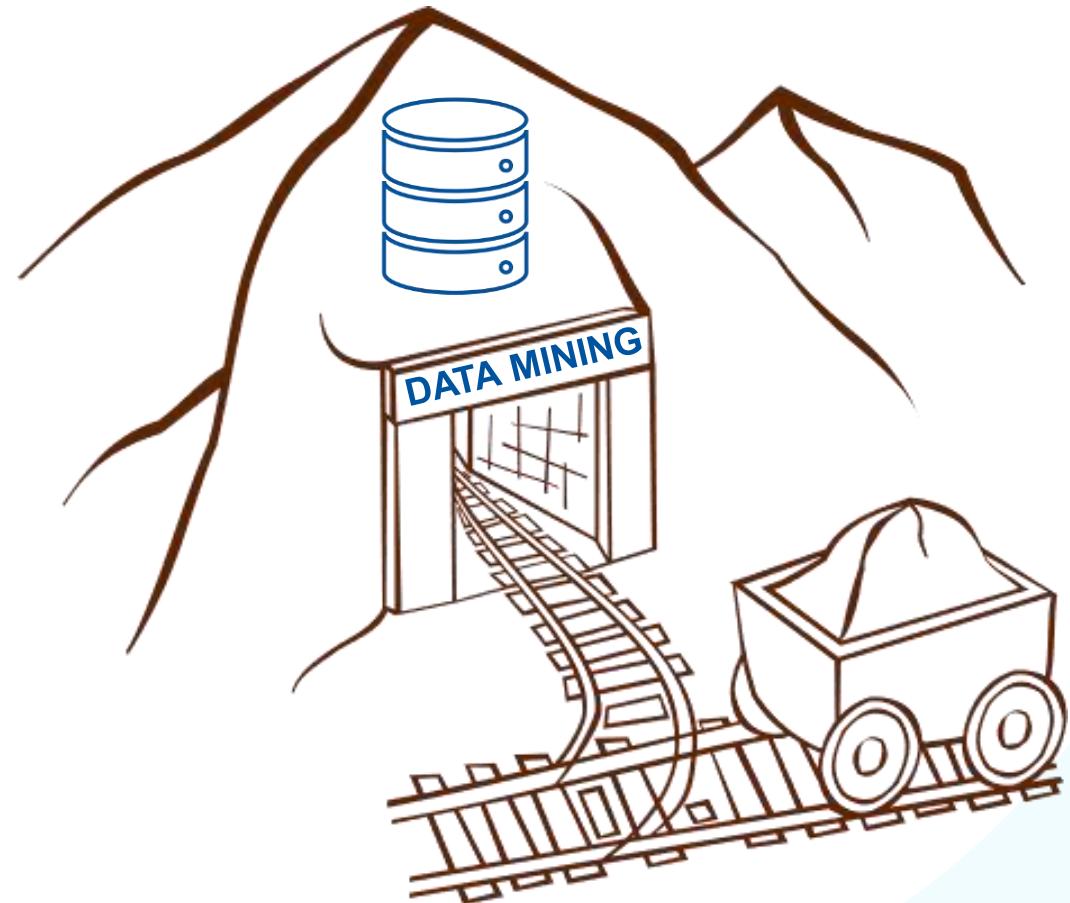
Limitations

- It may remain challenging to generate the candidate sets (may be huge)
- Each candidate needs to be tested against the whole dataset

→ FP-Growth is an approach that aims to overcome these limitations

Frequent Itemsets

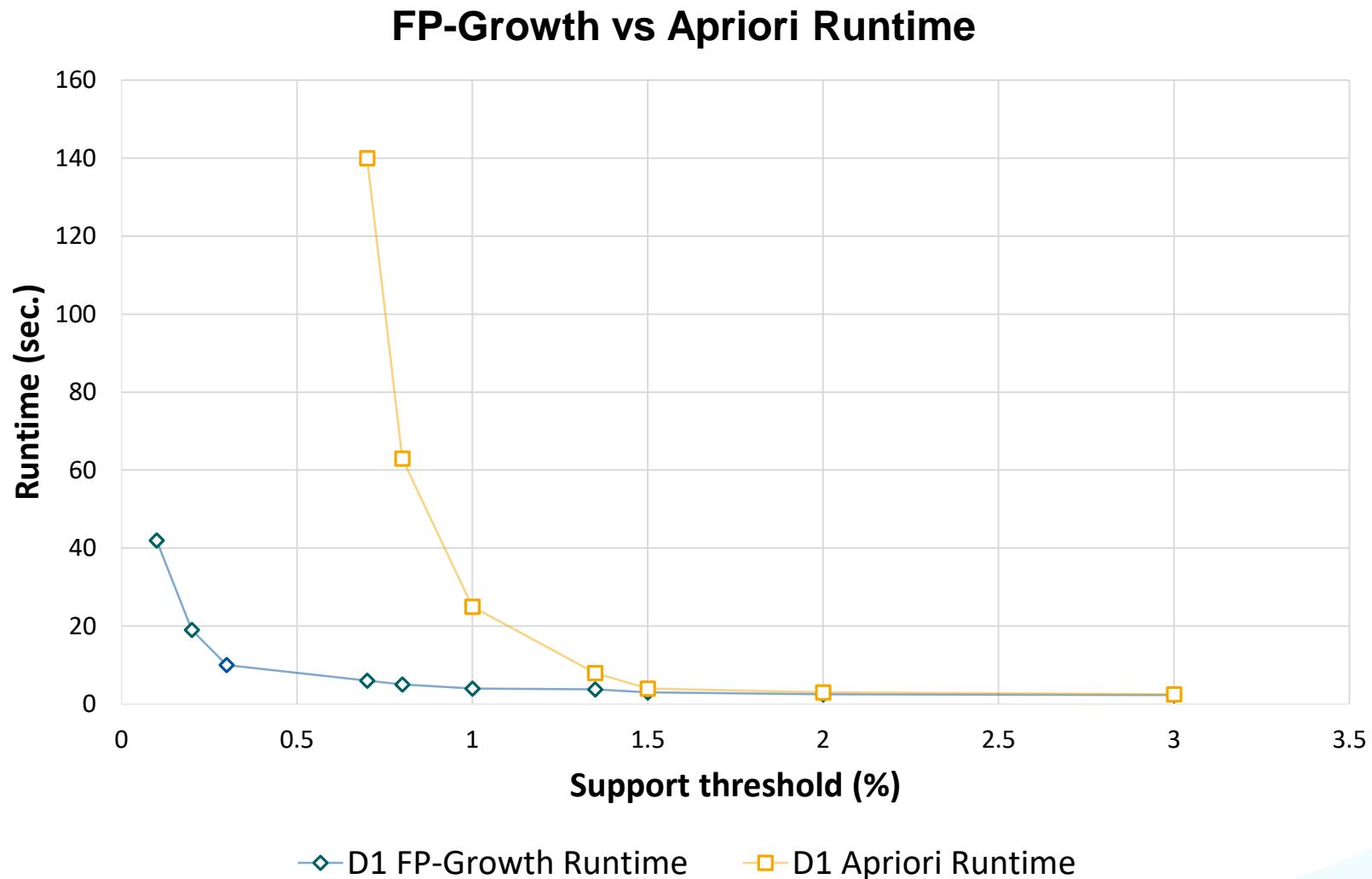
1. Introduction
2. Properties of Frequent Itemsets
3. Apriori Algorithm
4. **FP-Growth Algorithm**



Frequent Pattern Growth Algorithm

- Introduced by Jiawei Han, Jian Pei, Yiwen Yin in “Mining Frequent Patterns without Candidate Generation. SIGMOD Conference 2000: 1-12”
- Based on constructing the [Frequent Pattern Tree \(FP-Tree\)](#)
- Avoids generation of many candidates
- [Depth-first](#) rather than breadth-first
- Requires [only two passes](#) over the (potentially huge) dataset

Motivation



FP-Growth Steps

1. Determine the frequency of each item (first pass through the dataset)
2. Sort $\mathcal{I} = \{I_1, \dots, I_D\}$ based on their frequencies (I_1 is most frequent, I_D is the least frequent)
3. Remove the non-frequent items
4. The remaining items in each transactions are ordered by frequency (same as above)
5. This can be used to build a so-called prefix tree (second pass through the dataset)

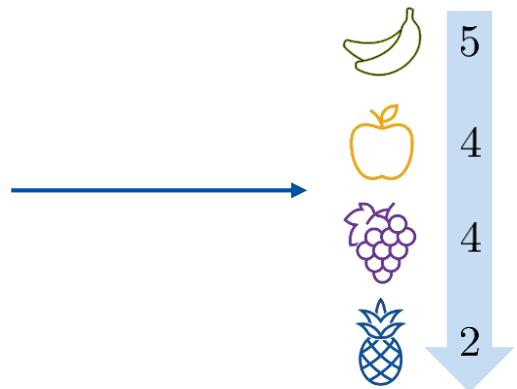
FP-Growth Steps

1. Determine the frequency of each item ([first](#) pass through the dataset)
 2. Sort $\mathcal{I} = \{I_1, \dots, I_D\}$ based on their frequencies (I_1 is most frequent, I_D is the least frequent)
 3. Remove the non-frequent items
 4. The remaining items in each transactions are [ordered by frequency](#) (same as above)
 5. This can be used to build a so-called [prefix tree](#) ([second](#) pass trough the dataset)
6. The resulting FP-tree contains all information needed to find the frequent itemsets of any length
[\(no need to traverse the dataset again\)](#)

Constructing FP-Tree – Example

TID	Bought Fruits
1	{Banana, Apple}
2	{Grapes, Banana, Pineapple}
3	{Apple, Banana}
4	{Apple, Grapes, Pineapple}
5	{Grapes, Banana}
6	{Apple, Banana, Grapes}

Dataset $\mathcal{X} \in \mathbb{M}(\mathbb{P}(\mathcal{J}))$

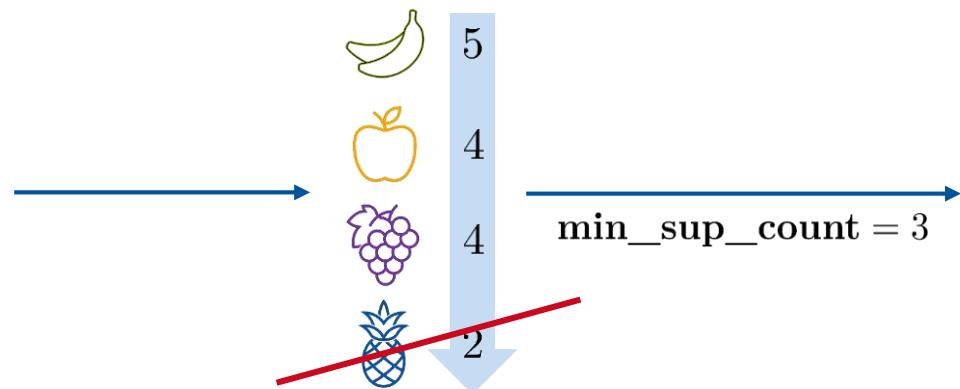


1. Determine the frequencies of items
2. Order the items based on frequency

Constructing FP-Tree – Example

TID	Bought Fruits
1	{Banana, Apple}
2	{Grapes, Banana, Pineapple}
3	{Apple, Banana}
4	{Apple, Grapes, Pineapple}
5	{Grapes, Banana}
6	{Apple, Banana, Grapes}

Dataset $\mathcal{X} \in \mathbb{M}(\mathbb{P}(\mathcal{J}))$



1. Determine the frequencies of items
2. Order the items based on frequency

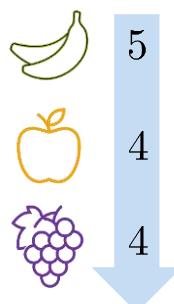
TID	Bought Fruits
1	{Banana, Apple}
2	{Banana, Grapes}
3	{Banana, Apple}
4	{Apple, Grapes}
5	{Banana, Grapes}
6	{Banana, Apple, Grapes}

3. Remove non-frequent items
4. Sort the items in the transactions

Constructing FP-Tree – Example

TID	Bought Fruits
1	{Banana, Apple}
2	{Banana, Grapes}
3	{Banana, Apple}
4	{Apple, Grapes}
5	{Banana, Grapes}
6	{Banana, Apple, Grapes}

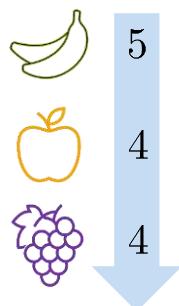
5. Build the FP-tree going through each transaction



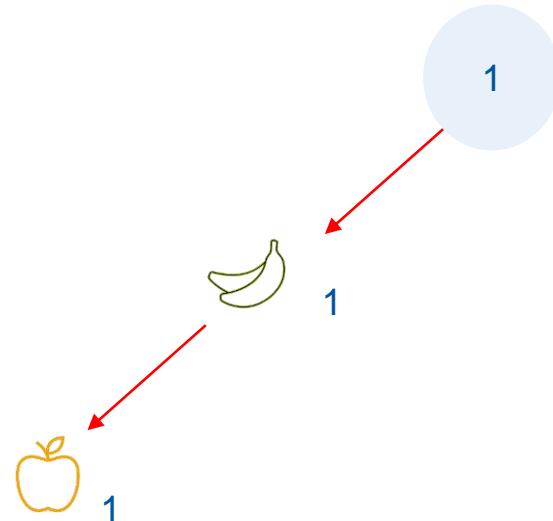
0

Constructing FP-Tree – Example

TID	Bought Fruits
1	{Banana, Apple}
2	{Banana, Grapes}
3	{Banana, Apple}
4	{Apple, Grapes}
5	{Banana, Grapes}
6	{Banana, Apple, Grapes}

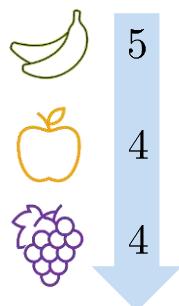


5. Build the FP-tree going through each transaction

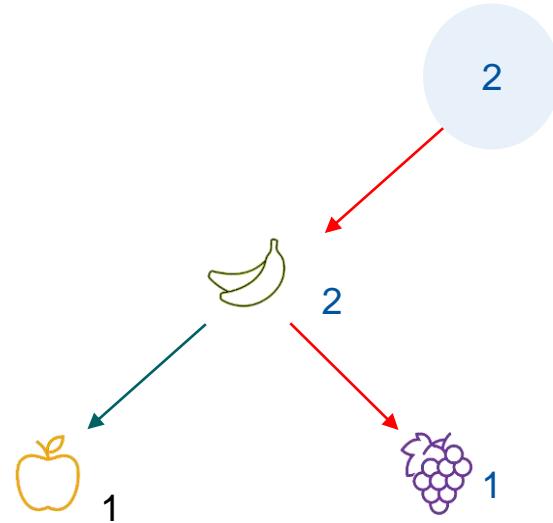


Constructing FP-Tree – Example

TID	Bought Fruits
1	{Banana, Apple}
2	Banana, Grapes
3	{Banana, Apple}
4	{Apple, Grapes}
5	{Banana, Grapes}
6	{Banana, Apple, Grapes}

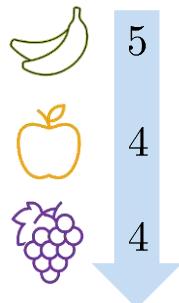


5. Build the FP-tree going through each transaction

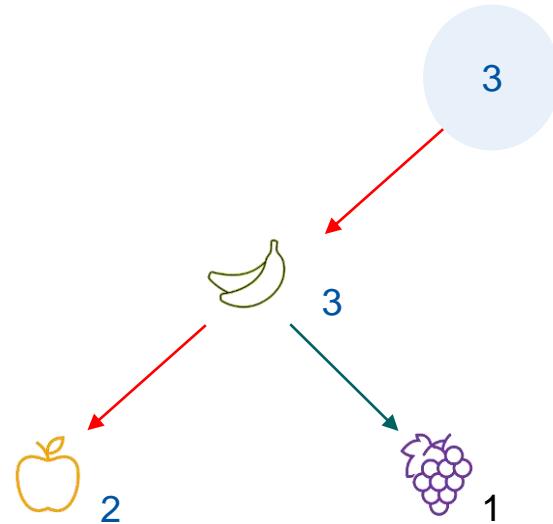


Constructing FP-Tree – Example

TID	Bought Fruits
1	{Banana, Apple}
2	{Banana, Grapes}
3	{Banana, Apple}
4	{Apple, Grapes}
5	{Banana, Grapes}
6	{Banana, Apple, Grapes}

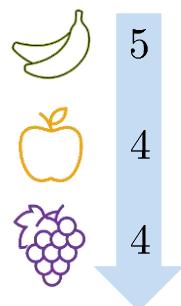


5. Build the FP-tree going through each transaction

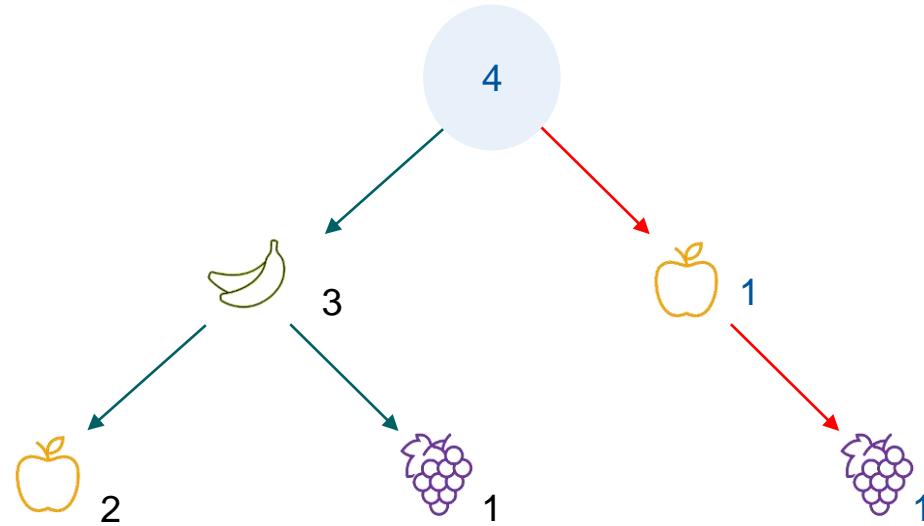


Constructing FP-Tree – Example

TID	Bought Fruits
1	{Banana, Apple}
2	{Banana, Grapes}
3	{Banana, Apple}
4	Apple, Grapes
5	{Banana, Grapes}
6	{Banana, Apple, Grapes}

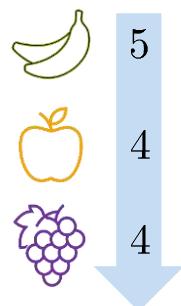


5. Build the FP-tree going through each transaction

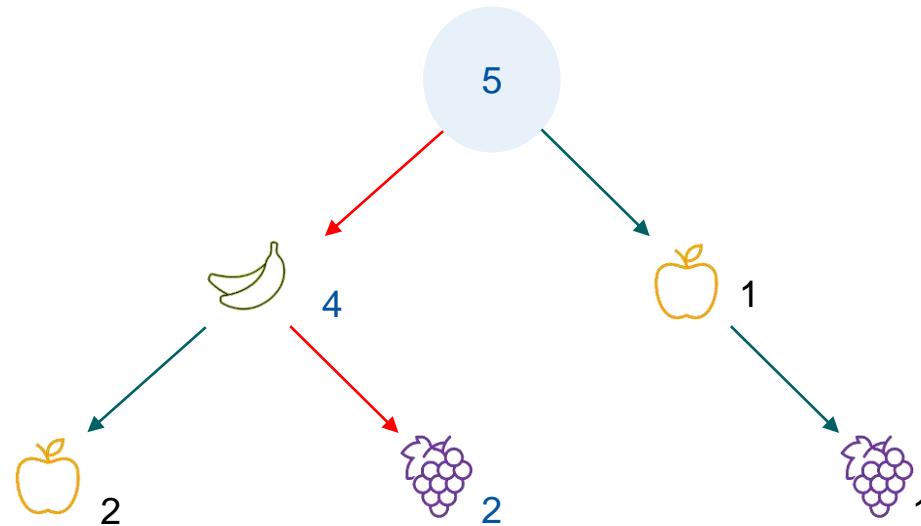


Constructing FP-Tree – Example

TID	Bought Fruits
1	{Banana, Apple}
2	{Banana, Grapes}
3	{Banana, Apple}
4	{Apple, Grapes}
5	Banana, Grapes
6	{Banana, Apple, Grapes}

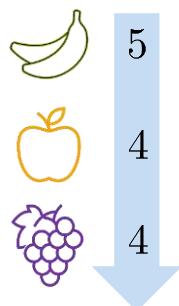


5. Build the FP-tree going through each transaction

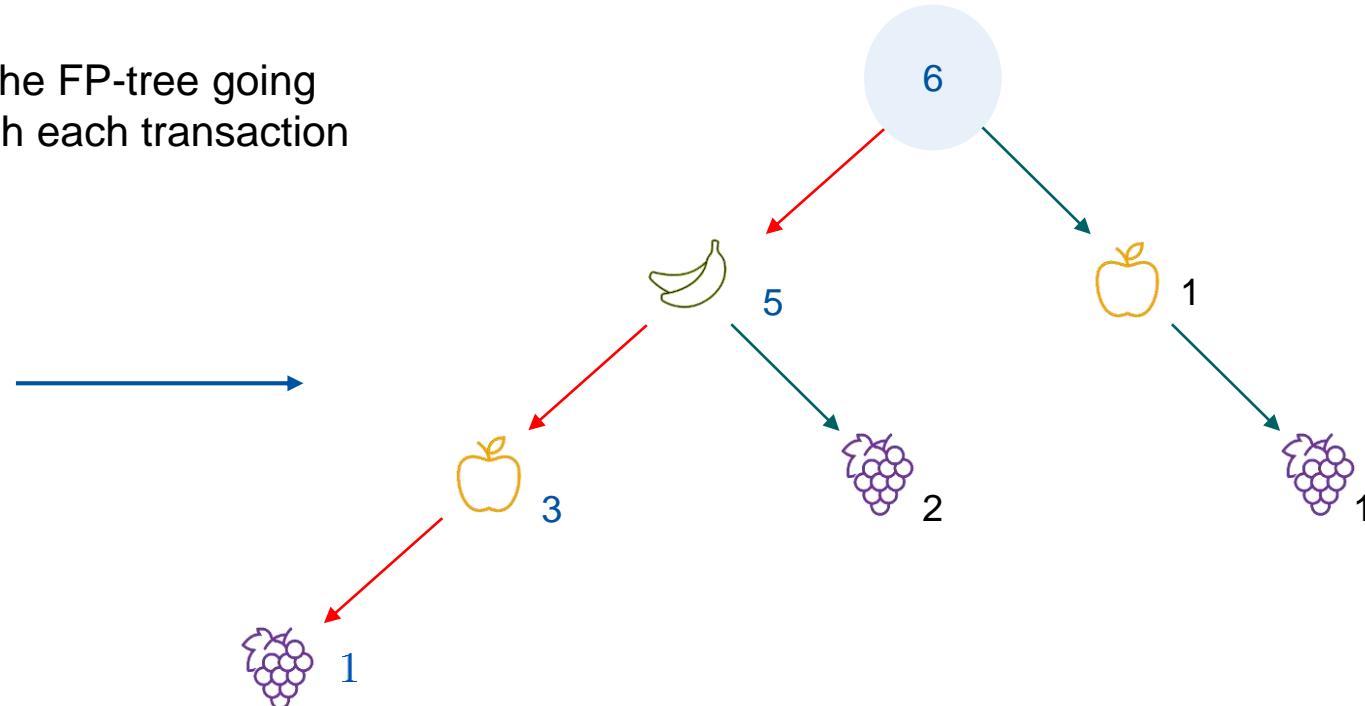


Constructing FP-Tree – Example

TID	Bought Fruits
1	{Banana, Apple}
2	{Banana, Grapes}
3	{Banana, Apple}
4	{Apple, Grapes}
5	{Banana, Grapes}
6	{Banana, Apple, Grapes}



5. Build the FP-tree going through each transaction

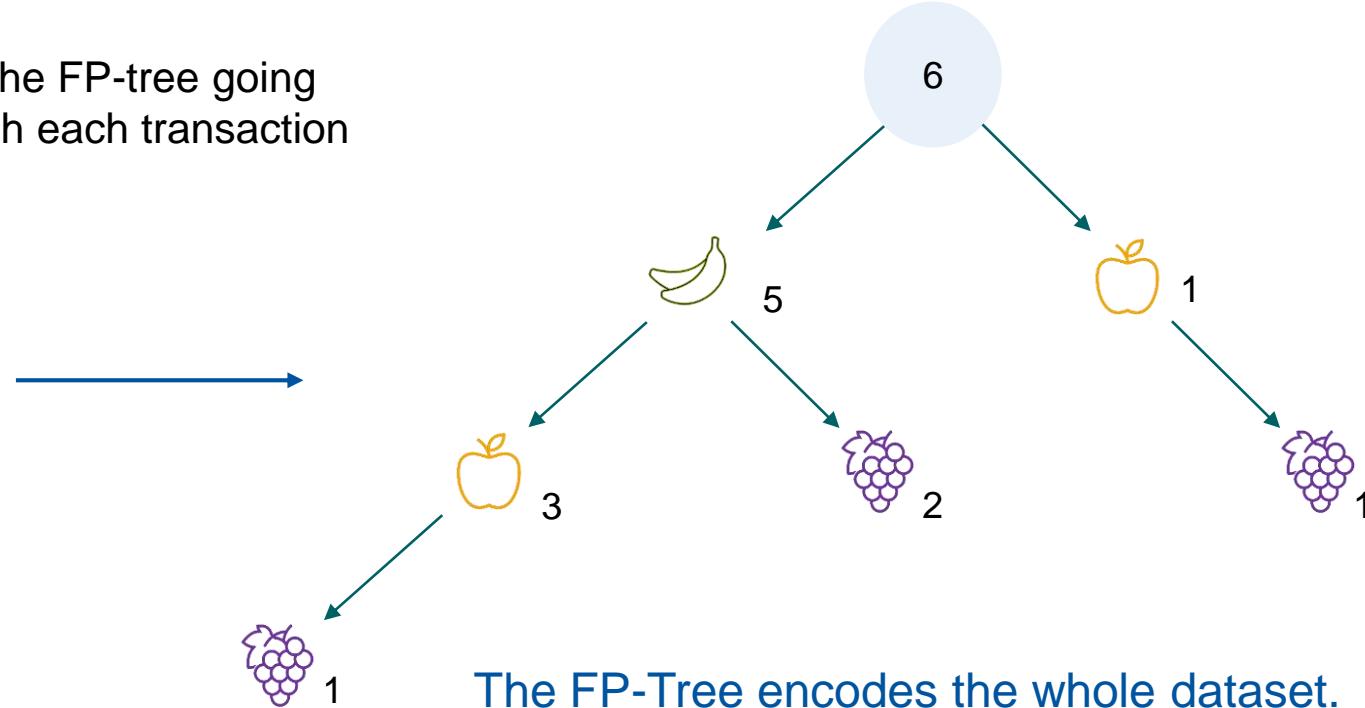


Constructing FP-Tree – Example

TID	Bought Fruits
1	{Banana, Apple}
2	{Banana, Grapes}
3	{Banana, Apple}
4	{Apple, Grapes}
5	{Banana, Grapes}
6	{Banana, Apple, Grapes}



5. Build the FP-tree going through each transaction

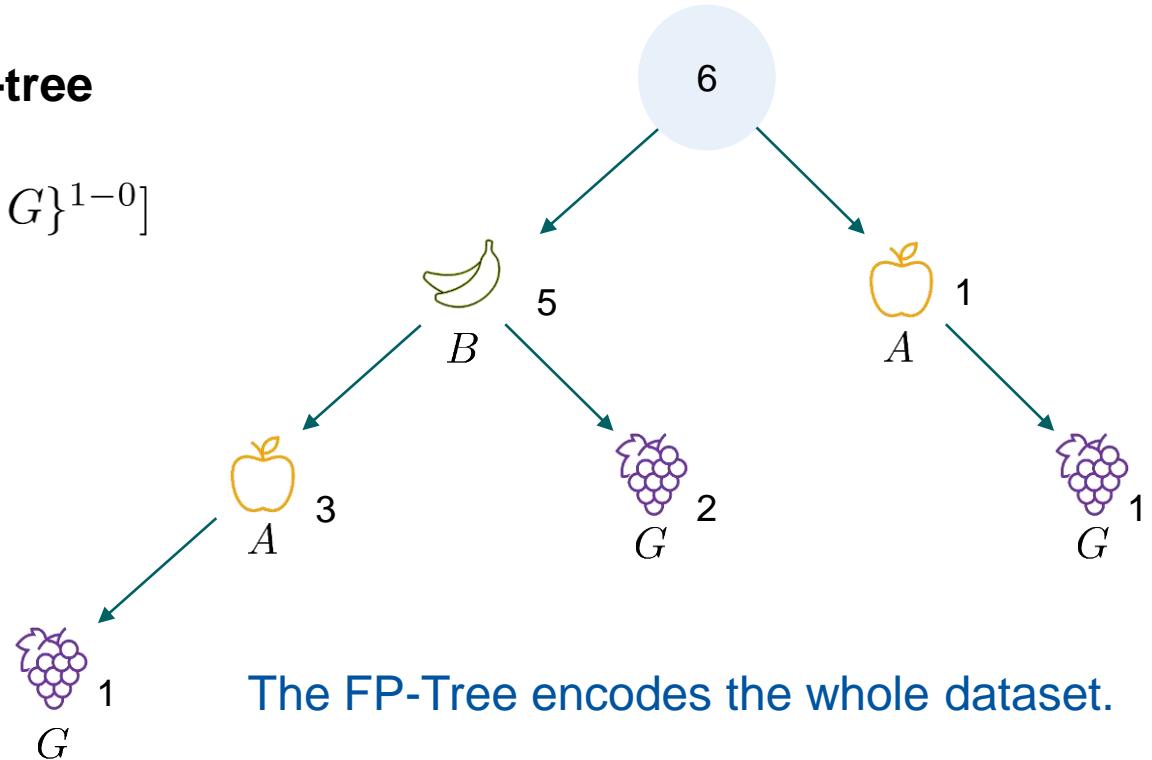


The FP-Tree encodes the whole dataset.

FP-Tree – Encodes The Dataset

We can read the transactions from the FP-tree

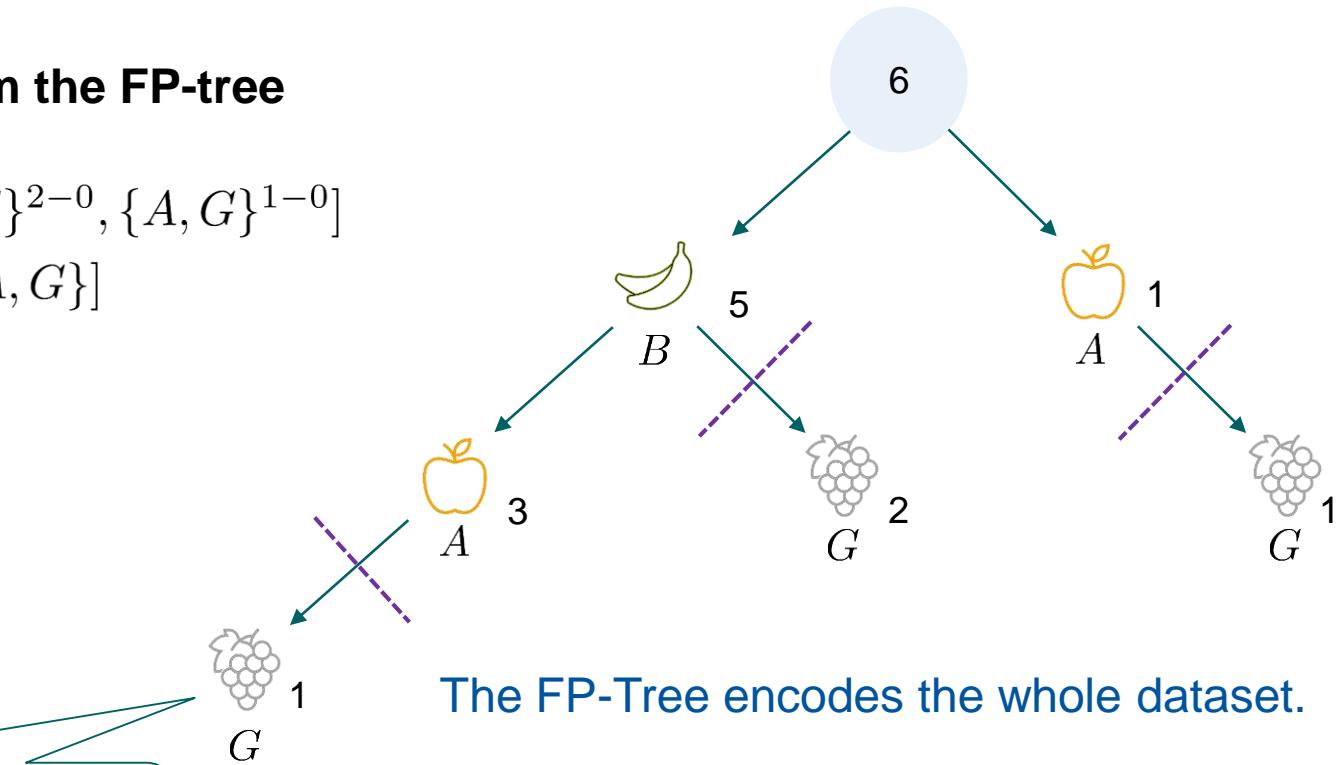
$$\begin{aligned}\mathcal{X} &= [\{B, A, G\}^{1-0}, \{B, A\}^{3-1}, \{B, G\}^{2-0}, \{A, G\}^{1-0}] \\ &= [\{B, A, G\}, \{B, A\}^2, \{B, G\}^2, \{A, G\}]\end{aligned}$$



FP-Tree – Cannot Cut Naïvely

We can read the transactions from the FP-tree

$$\begin{aligned}\mathcal{X} &= [\{B, A, G\}^{1-0}, \{B, A\}^{3-1}, \{B, G\}^{2-0}, \{A, G\}^{1-0}] \\ &= [\{B, A, G\}, \{B, A\}^2, \{B, G\}^2, \{A, G\}]\end{aligned}$$

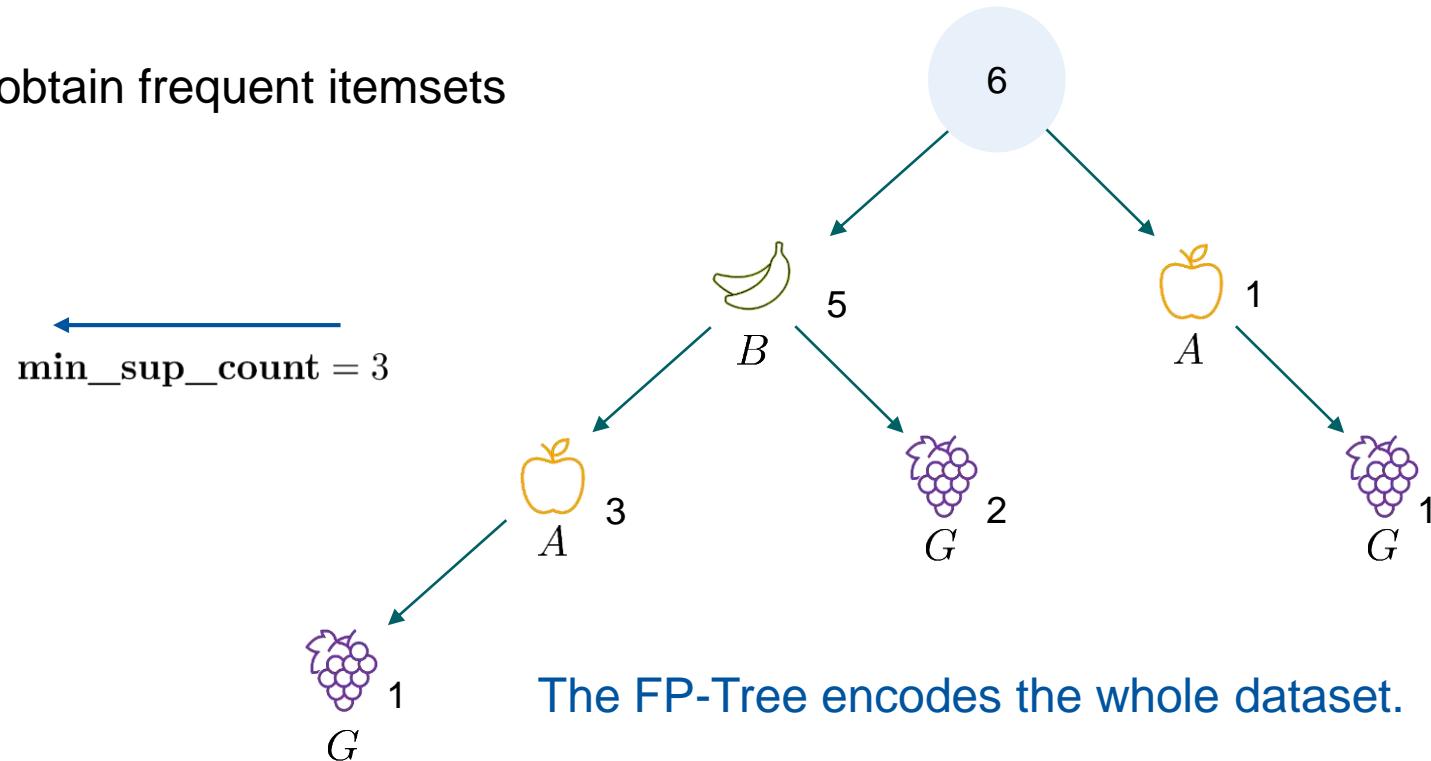


Even though G exists only once in this subtree, we can't cut it naively, because its support may be higher than the threshold (3)

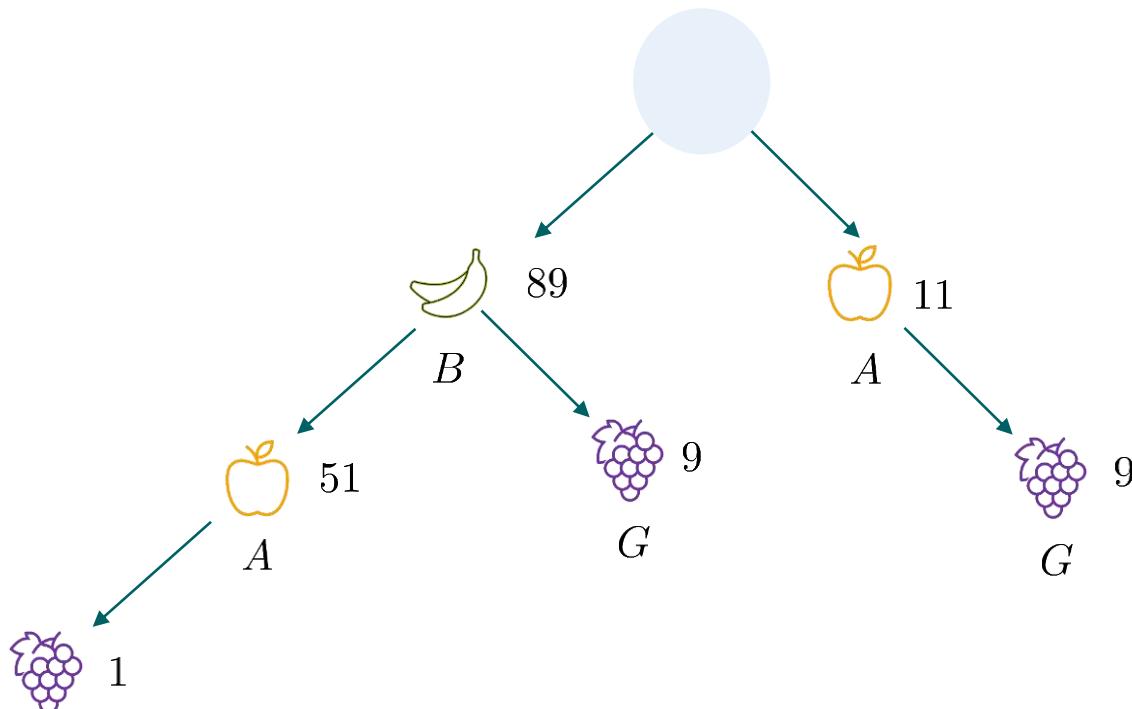
FP-Tree – Frequent Itemsets

Next: 6. Mining the FP-tree to obtain frequent itemsets

Frequent Itemsets	Support Count
{B}	5
{A}	4
{G}	4
{B, A}	3
{B, G}	3



FP-Tree Encodes Dataset – Another Example



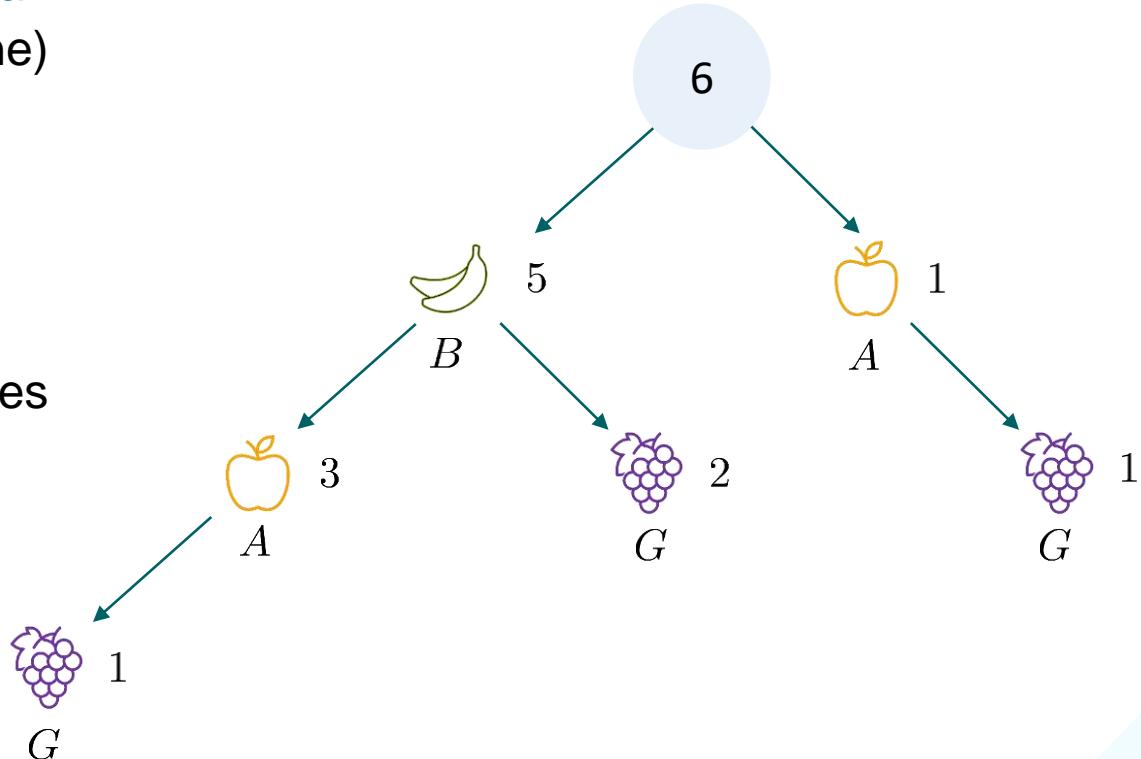
We can read the transactions from the FP-tree

$$\mathcal{X} = [\{B, A, G\}^{1-0}, \{B, A\}^{51-1}, \{B, G\}^{9-0}, \{B\}^{89-(51+9)}, \{A, G\}^{9-0}, \{A\}^{11-9}]$$

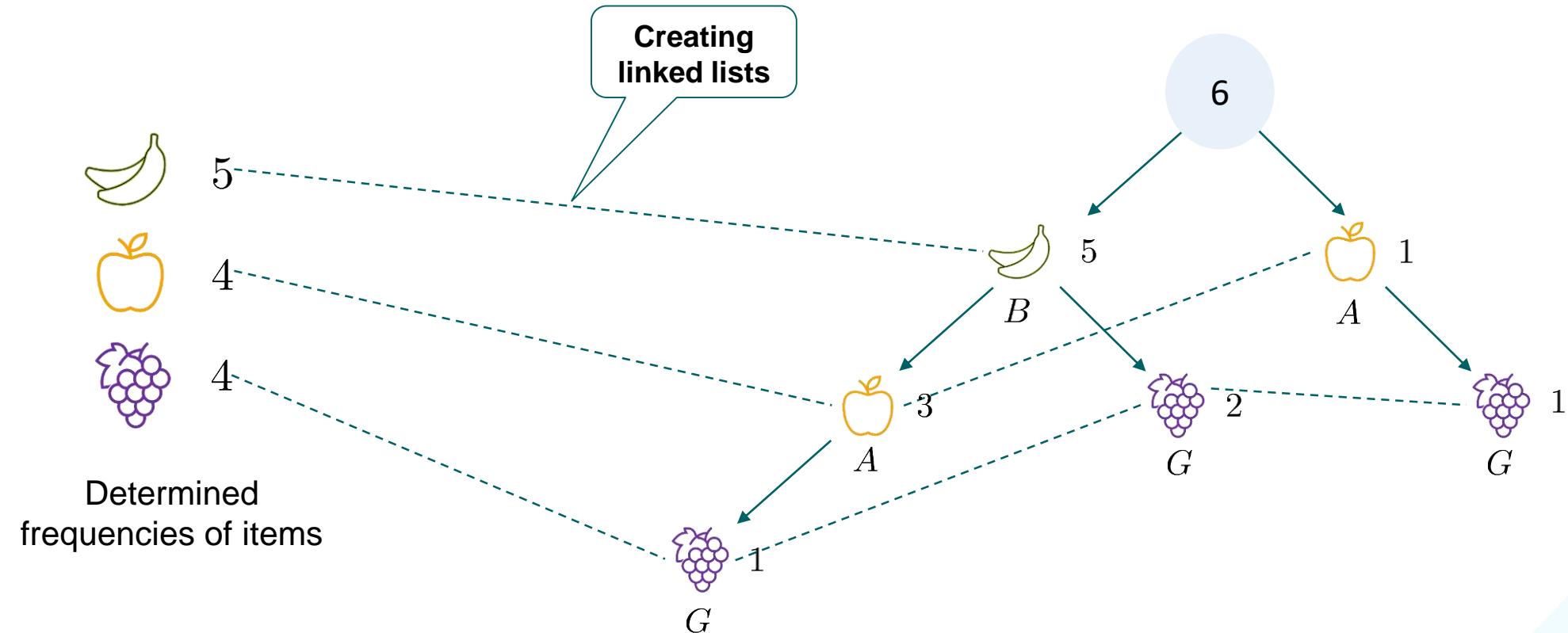
$$\mathcal{X} = [\{B, A, G\}^1, \{B, A\}^{50}, \{B, G\}^9, \{B\}^{29}, \{A, G\}^9, \{A\}^2]$$

Mining the FP-Tree – Overview

- For each frequent item, create a **conditional FP-tree** (starting with the **least** frequent one)
- The conditional FP-tree considers all transactions ending with this item
- Apply this **recursively**
- Due to recursion, we also consider postfixes that contain multiple elements
- The ordering ensures that postfixes are considered only once

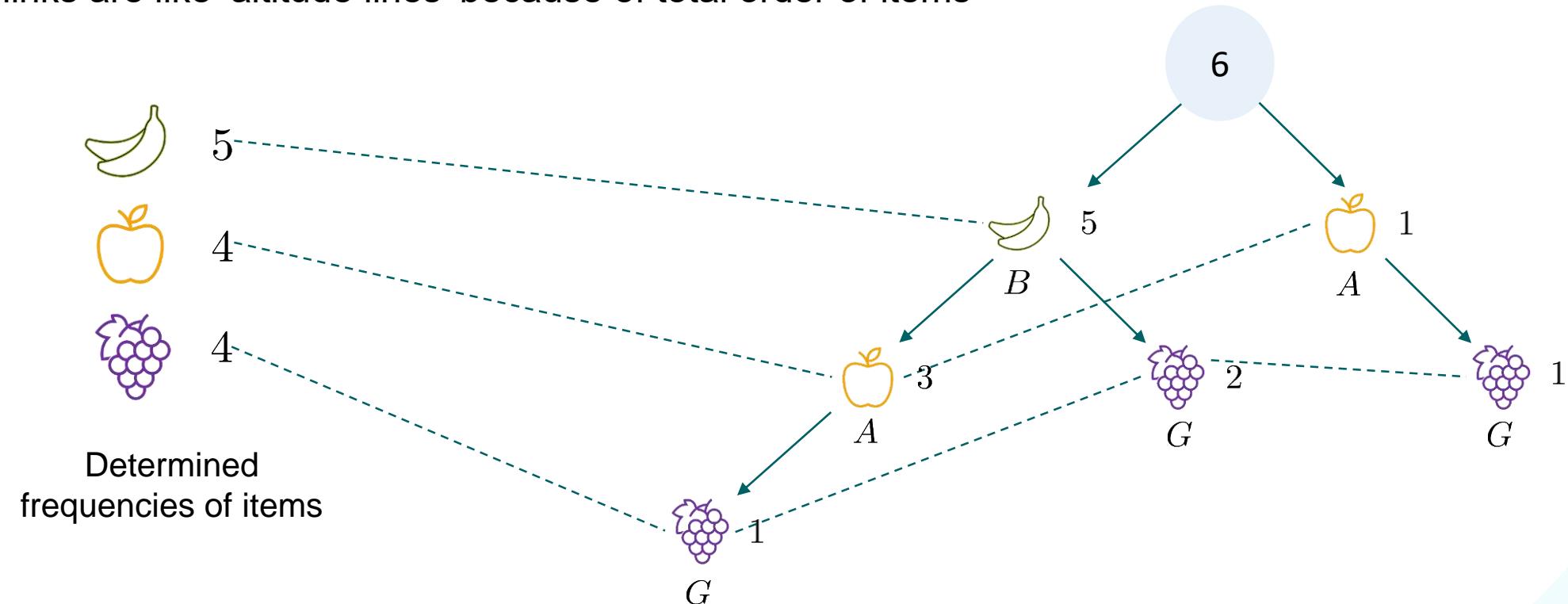


Node Links

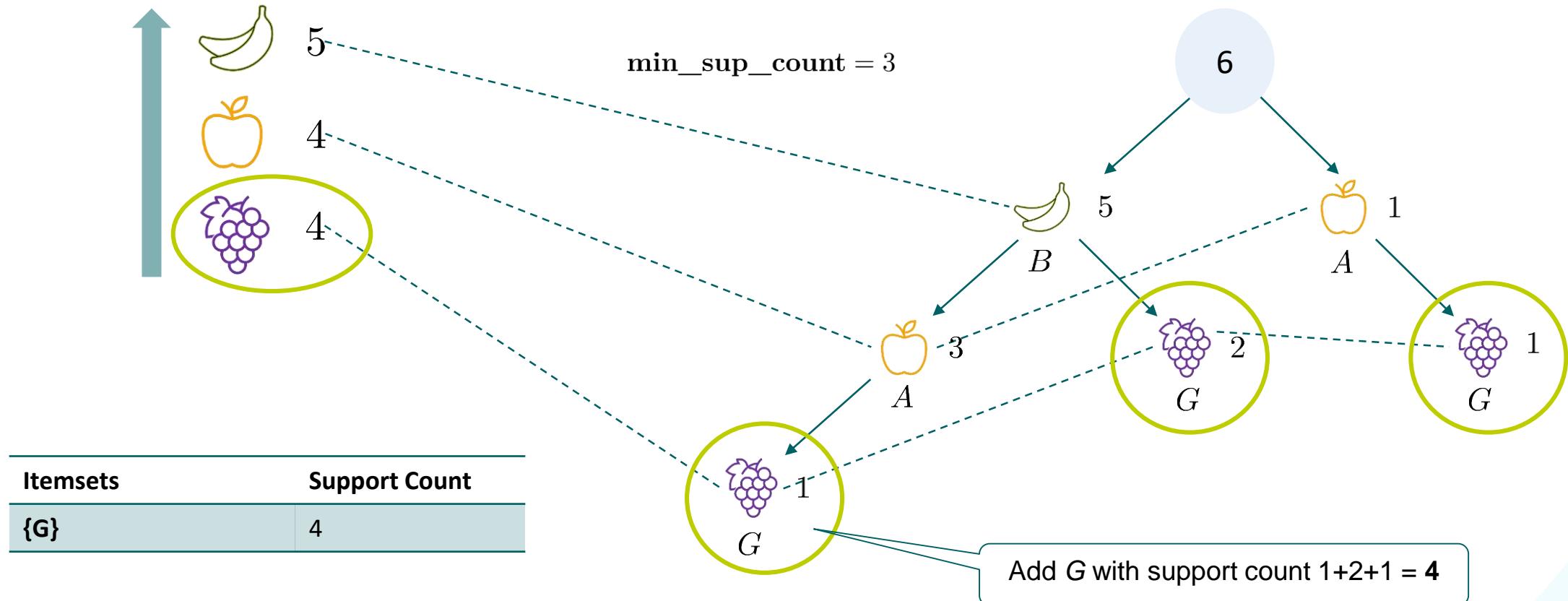


Node Links

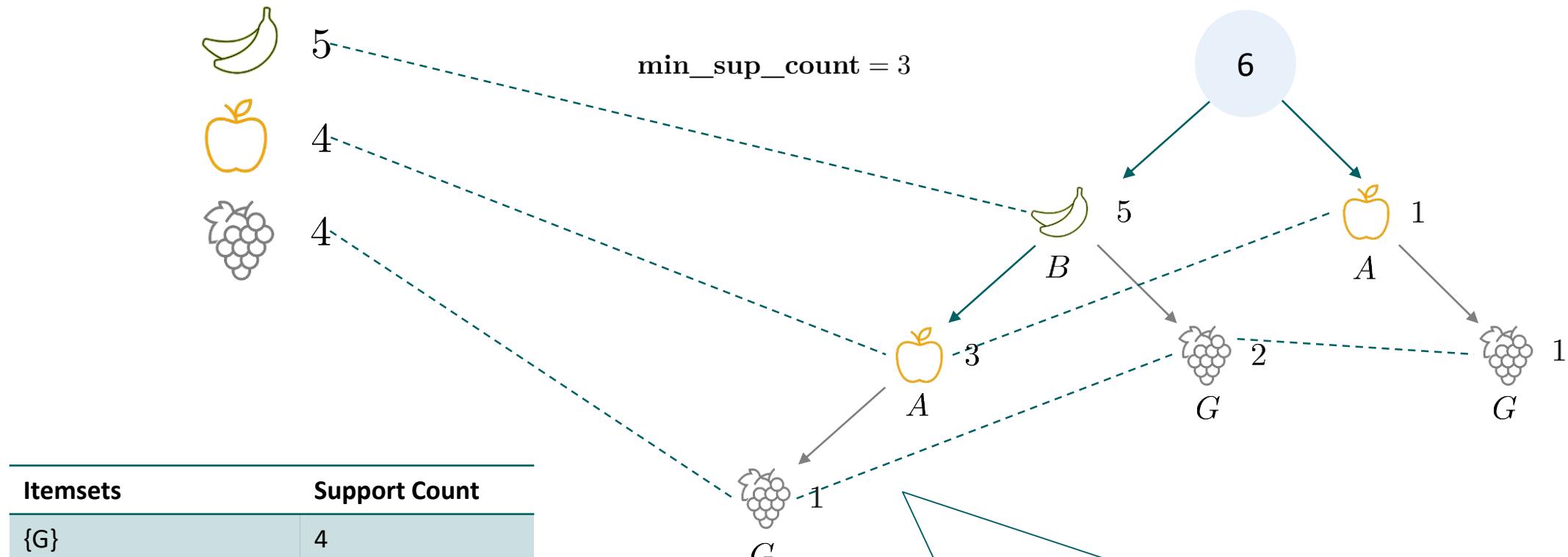
Node links are like ‘altitude lines’ because of total order of items



Consider Postfix G

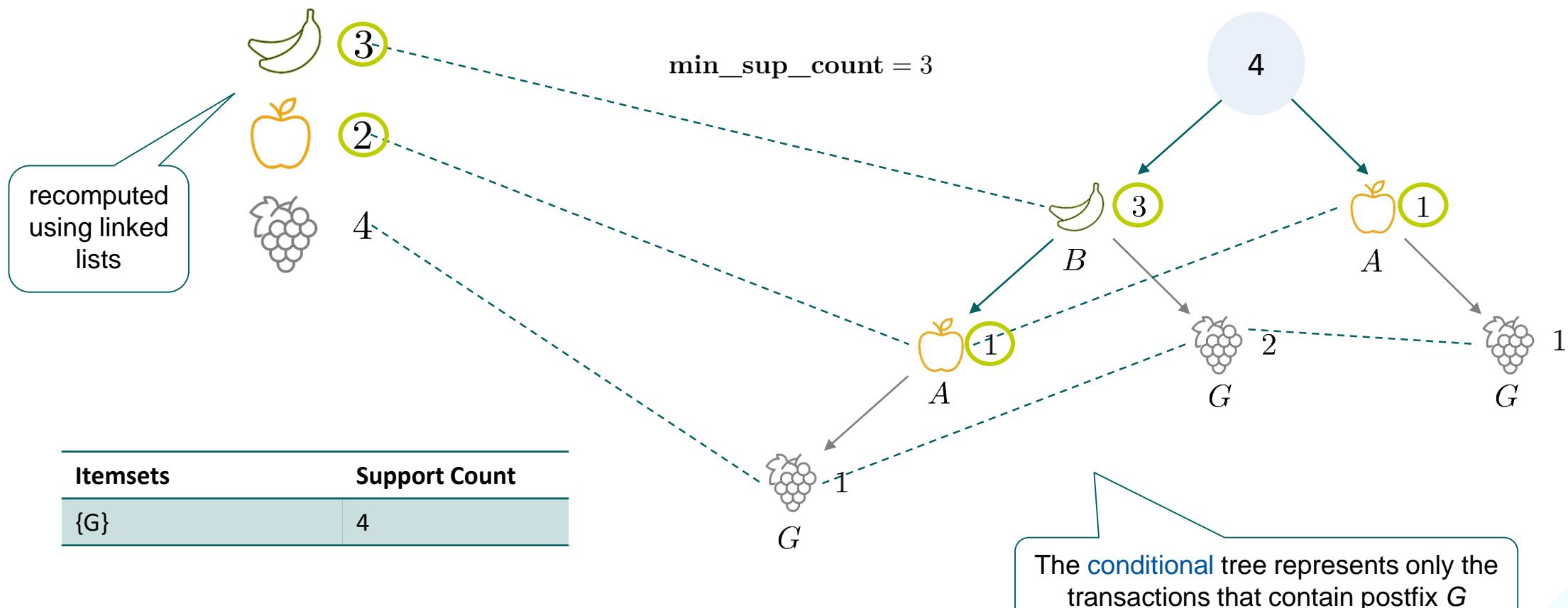


Consider Postfix G

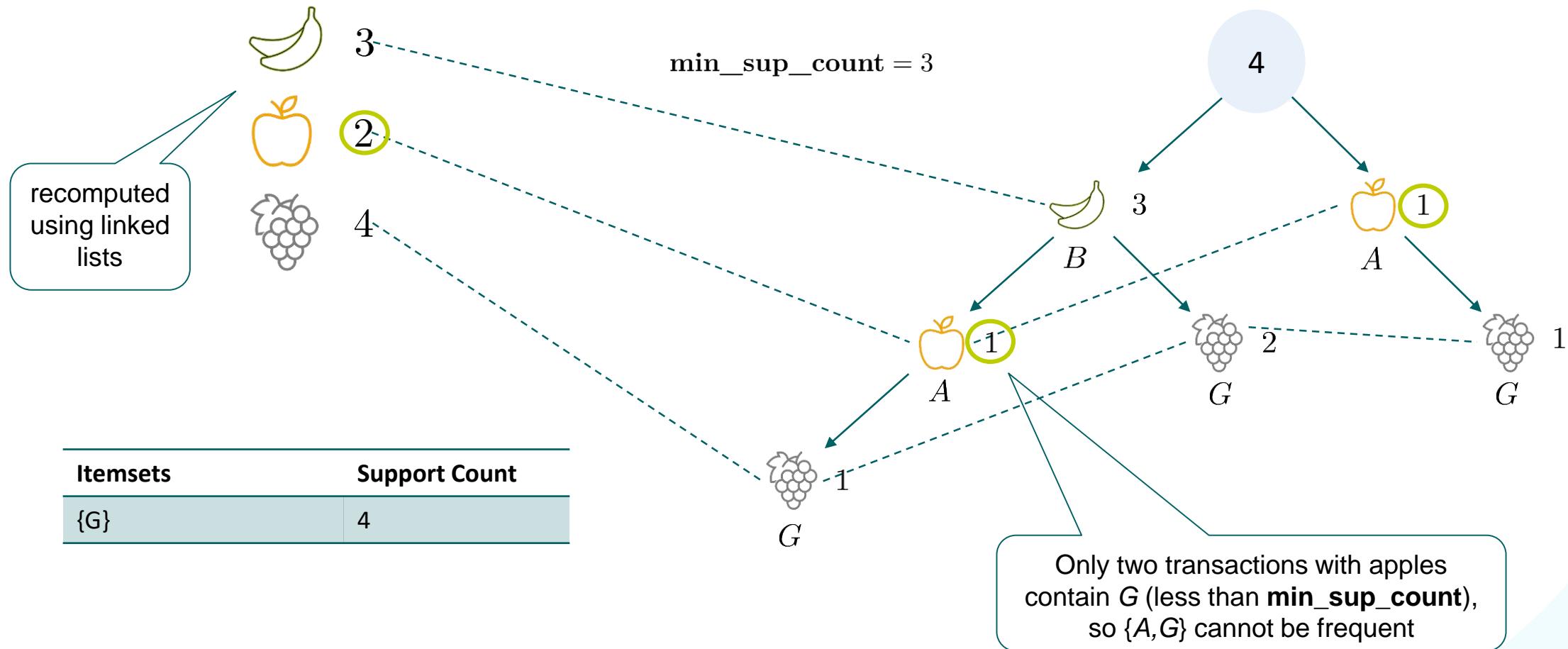


Now only consider **all the paths** leading from the root to G
(representing transactions that include G)

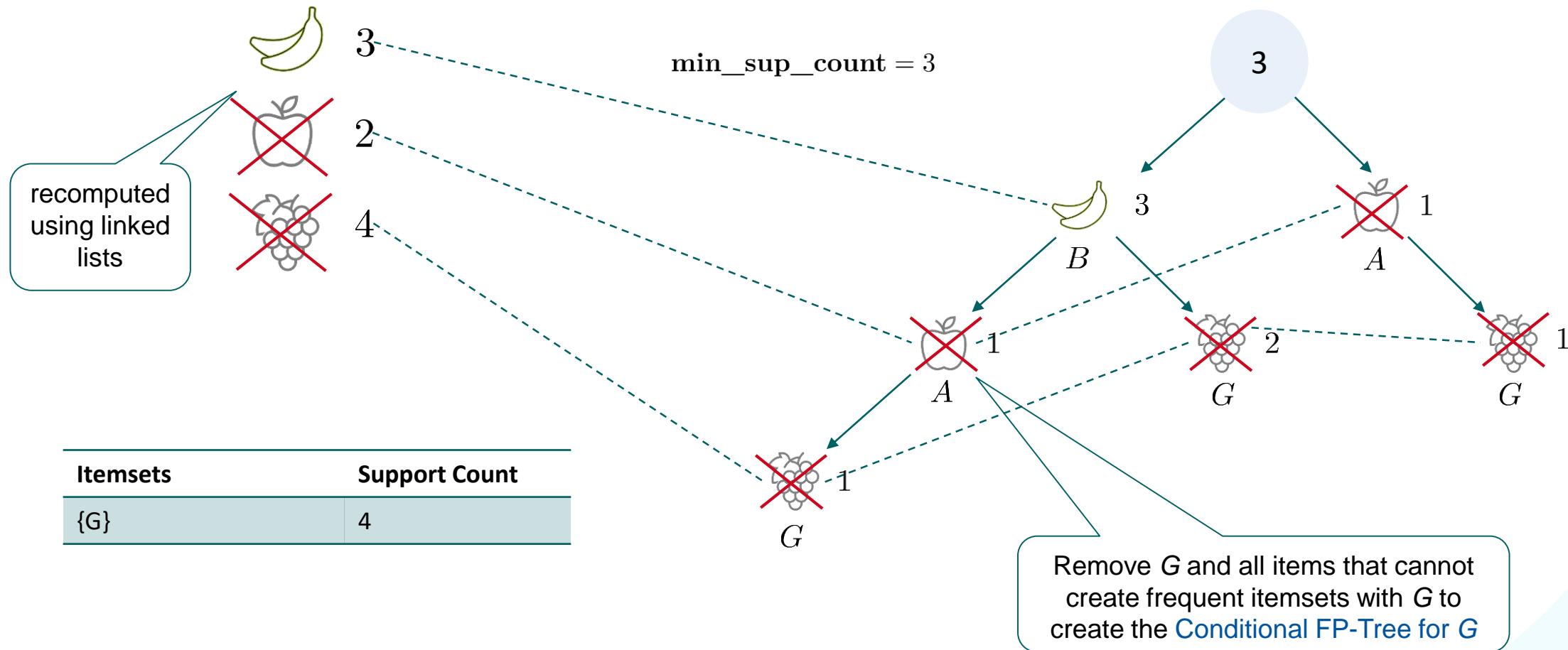
Towards the Conditional FP-Tree for Postfix G



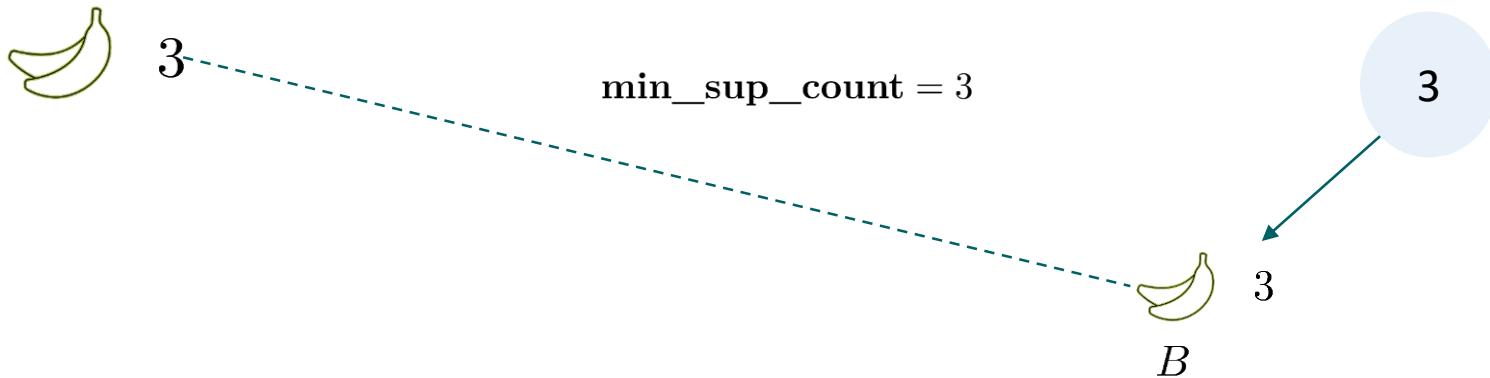
Towards the Conditional FP-Tree for Postfix G



Towards the Conditional FP-Tree for Postfix G



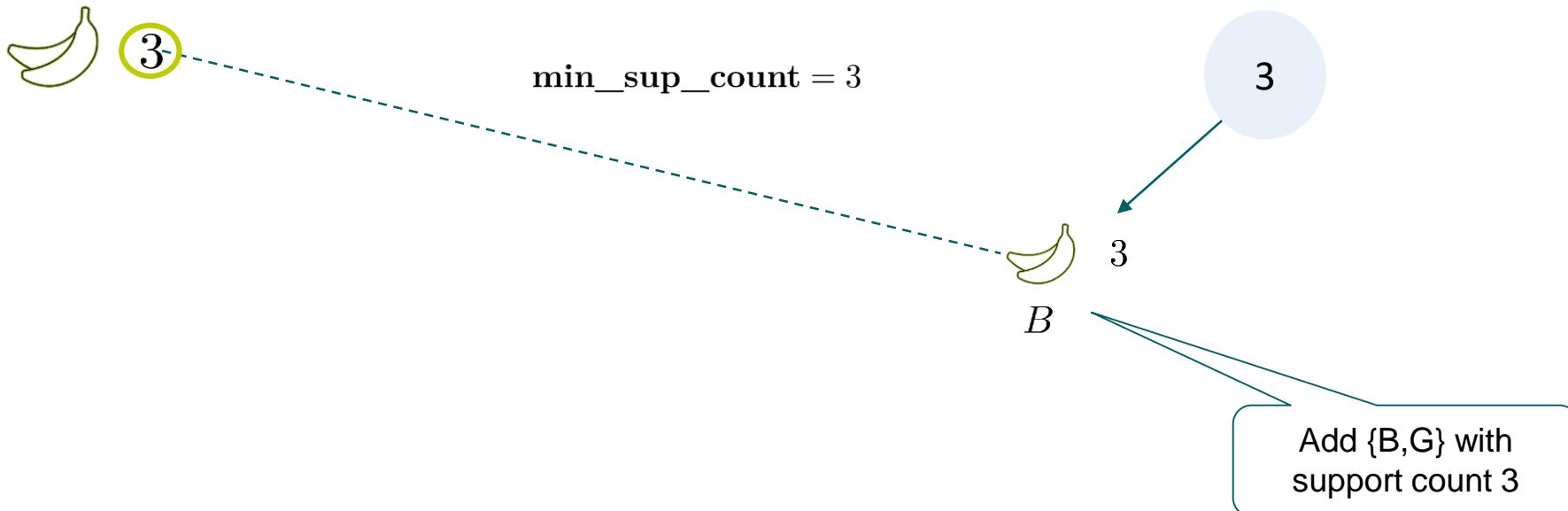
Conditional FP-Tree for Postfix G



Itemsets	Support Count
{G}	4
{..., G}	reurse

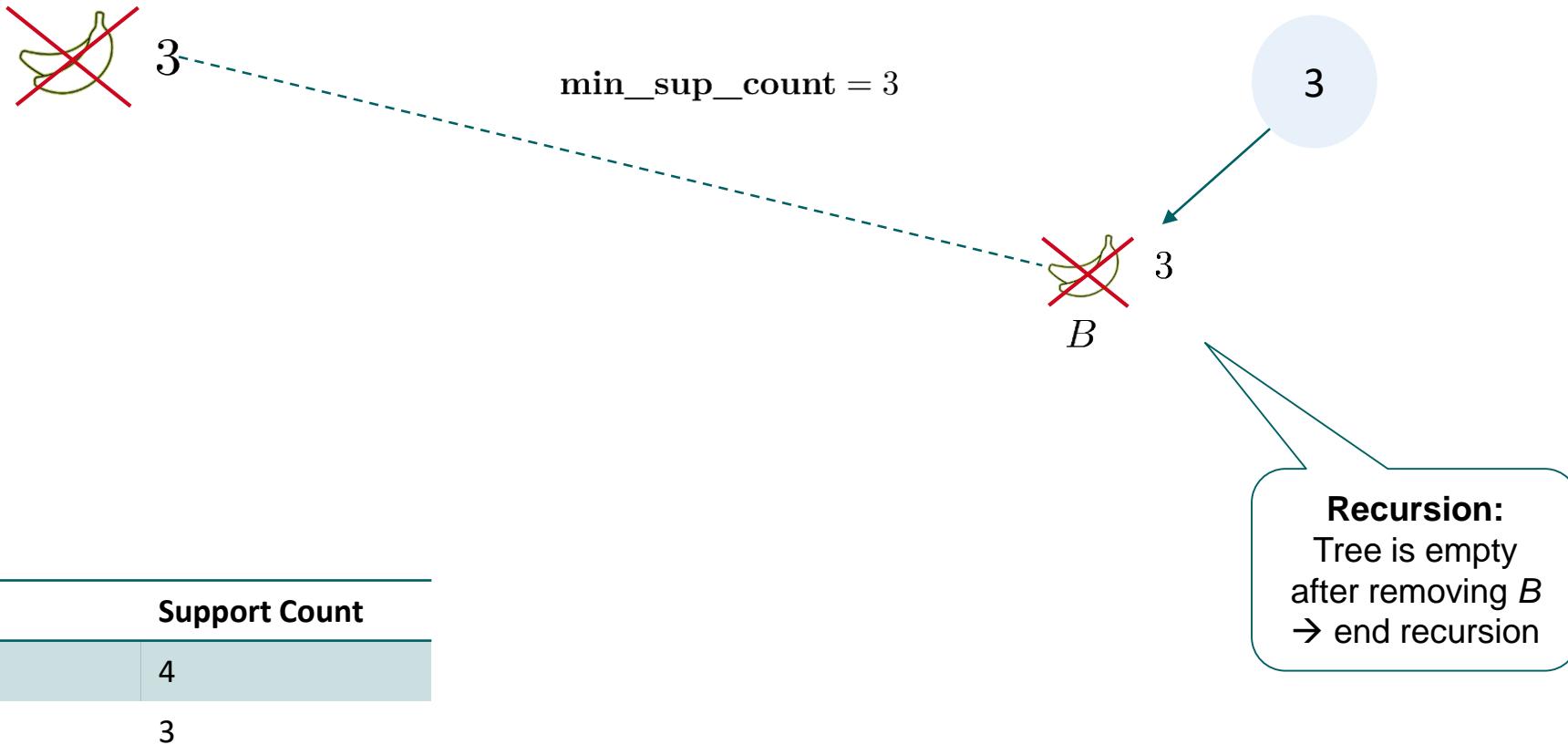
Mine the **Conditional FP-Tree for G** to
find frequent itemsets that contain G
(Recursion)

Conditional FP-Tree for Postfix G

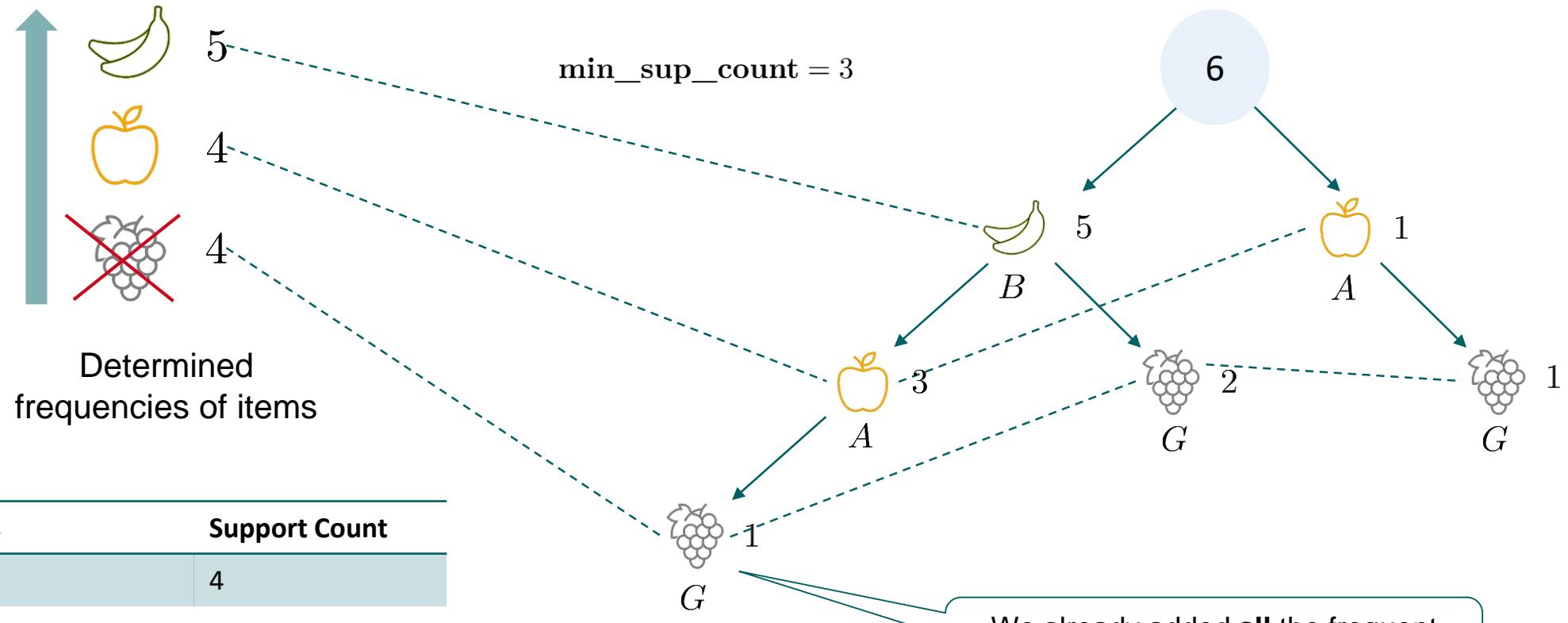


Itemsets	Support Count
$\{G\}$	4
$\{B, G\}$	3

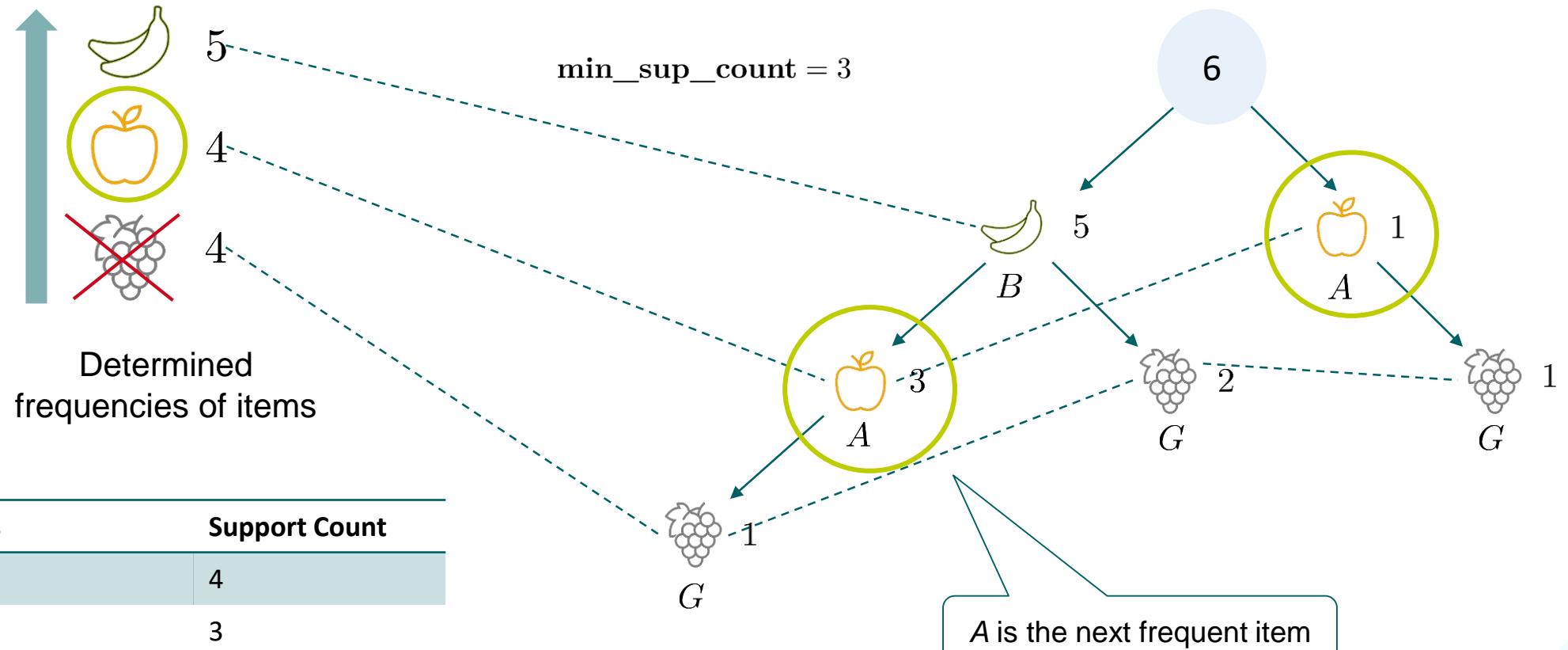
Conditional FP-Tree for Postfix G



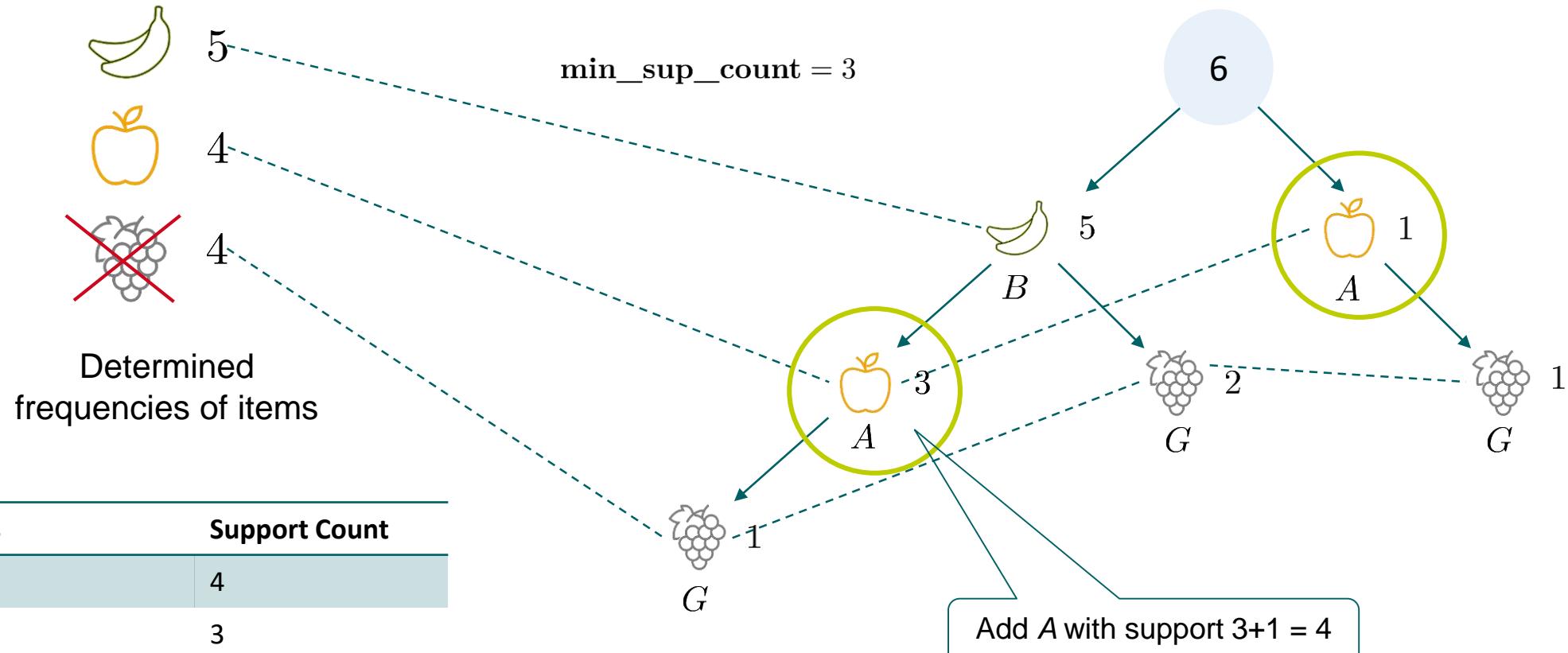
Consider Postfix A



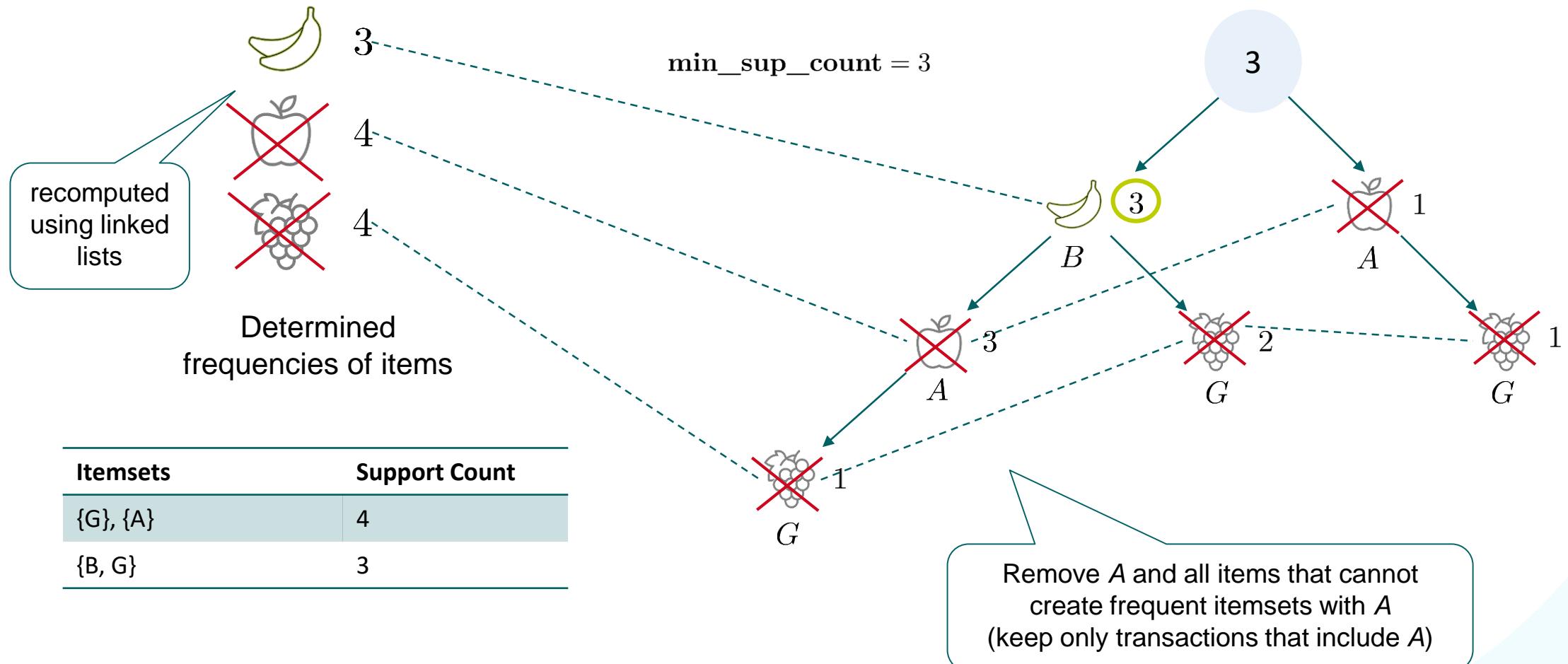
Consider Postfix A



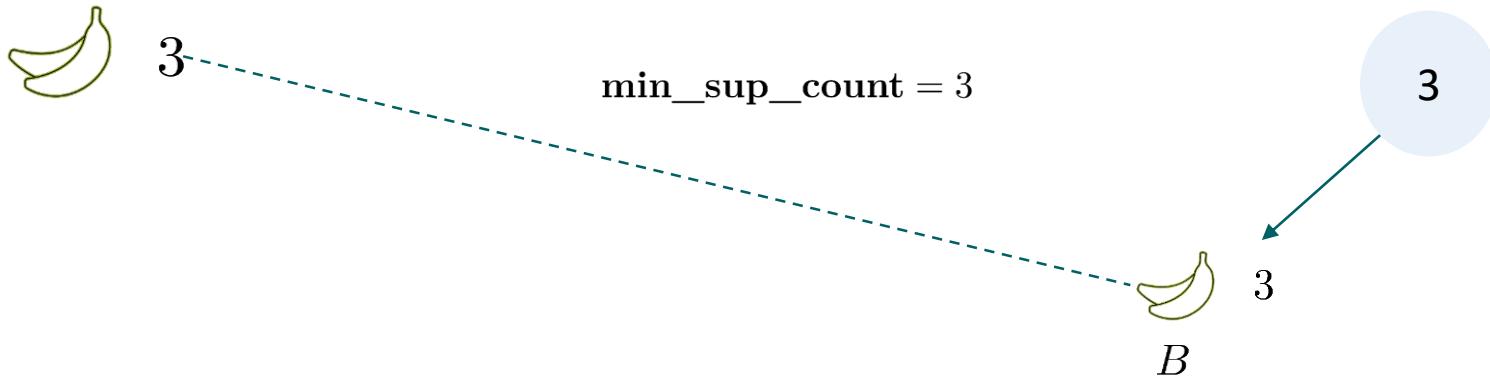
Consider Postfix A



Towards the Conditional FP-Tree for Postfix A



Towards Conditional FP-Tree for Postfix A



Determined
frequencies of items

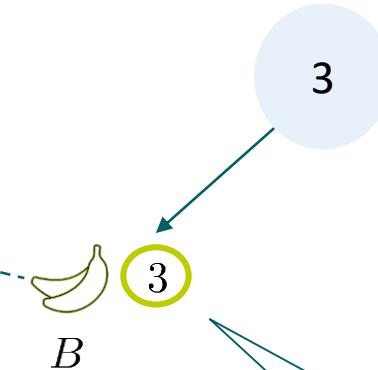
Itemsets	Support Count
{G}, {A}	4
{B, G}	3

Conditional FP-Tree for Postfix A



3

min_sup_count = 3

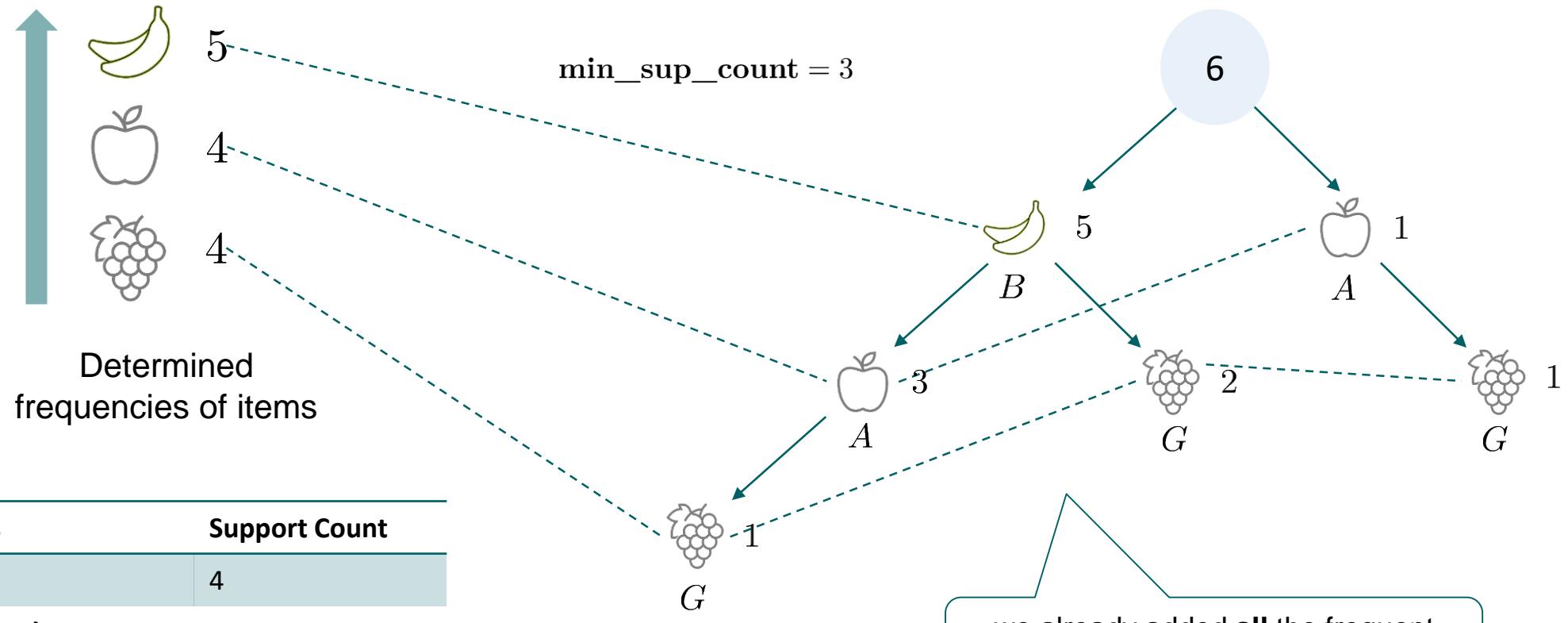


Determined
frequencies of items

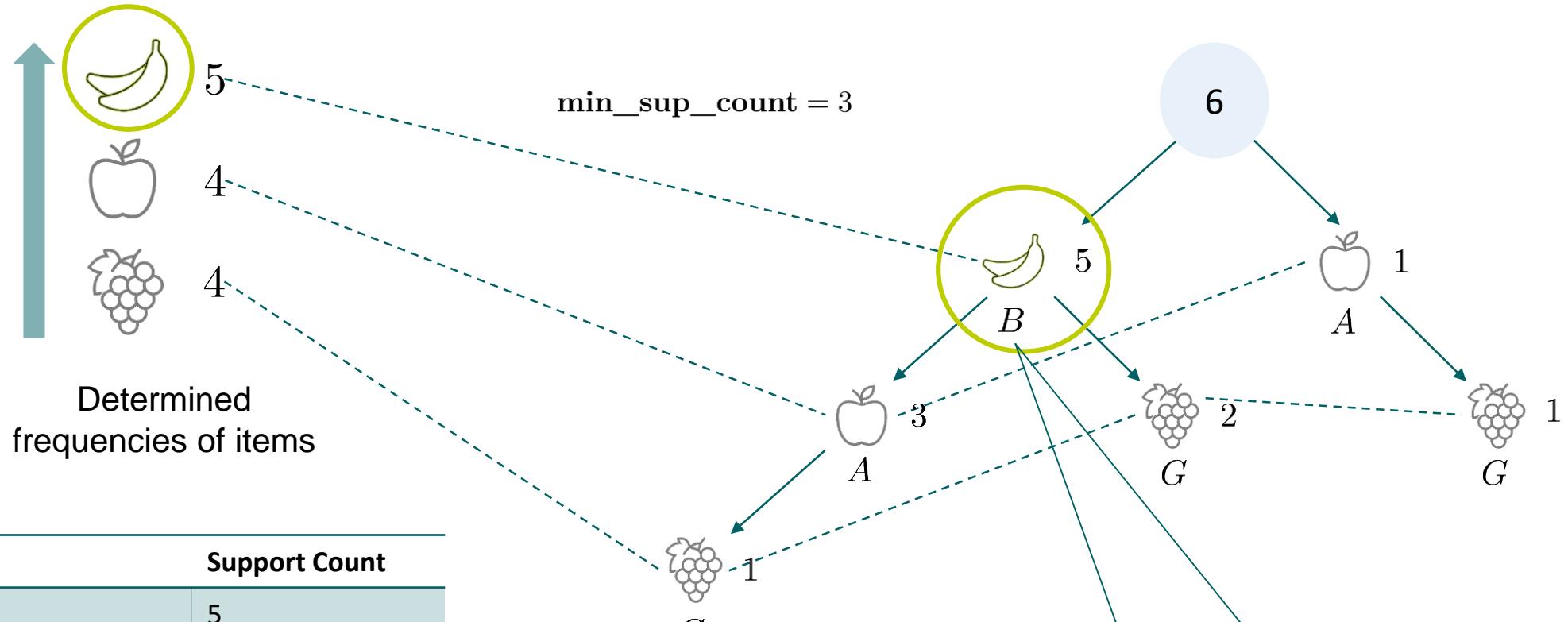
Itemsets	Support Count
{G}, {A}	4
{B, G}, {B, A}	3

Recursion:
Add {B, A} with
support 3 (then end
recursion)

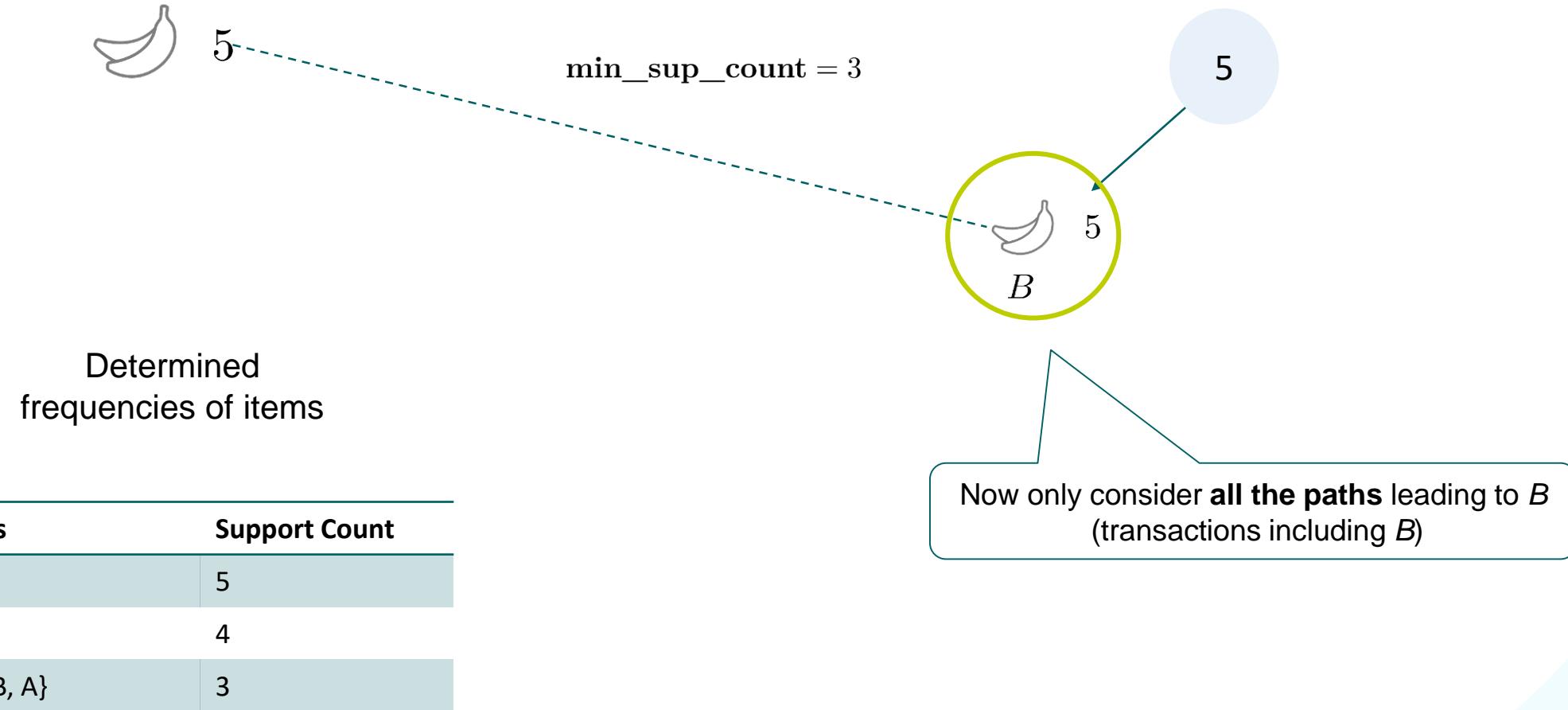
Consider Postfix B



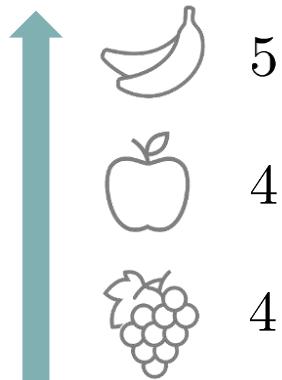
Consider Postfix B



Towards Conditional FP-Tree for Postfix B

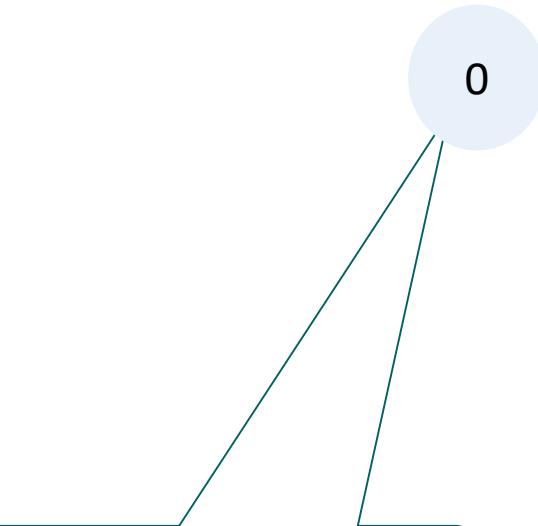


Conditional FP-Tree for Postfix B

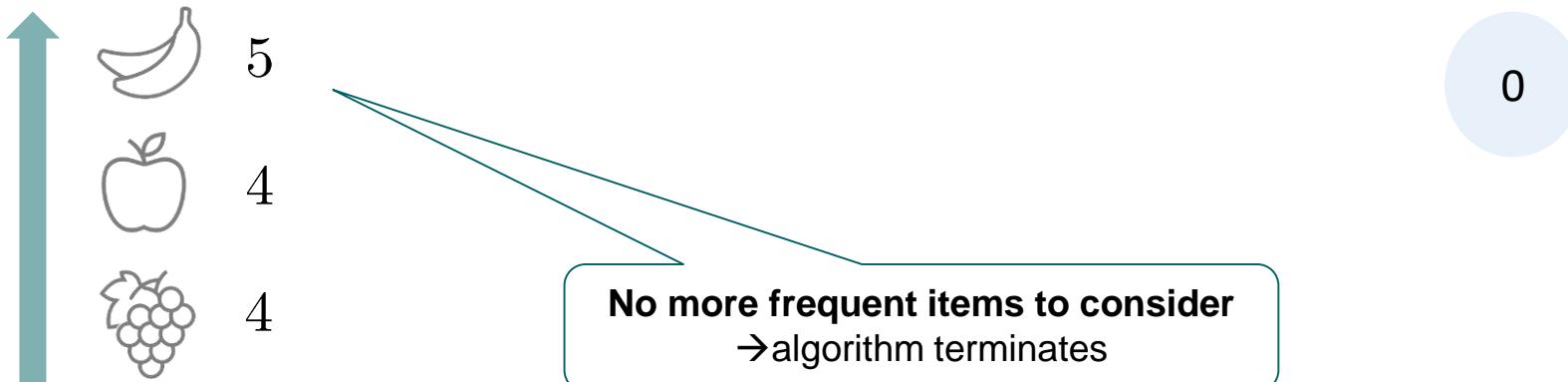


Itemsets	Support Count
{B}	5
{G}, {A}	4
{B, G}, {B, A}	3

Conditional FP-tree is empty, i.e., no additional frequent itemsets with B

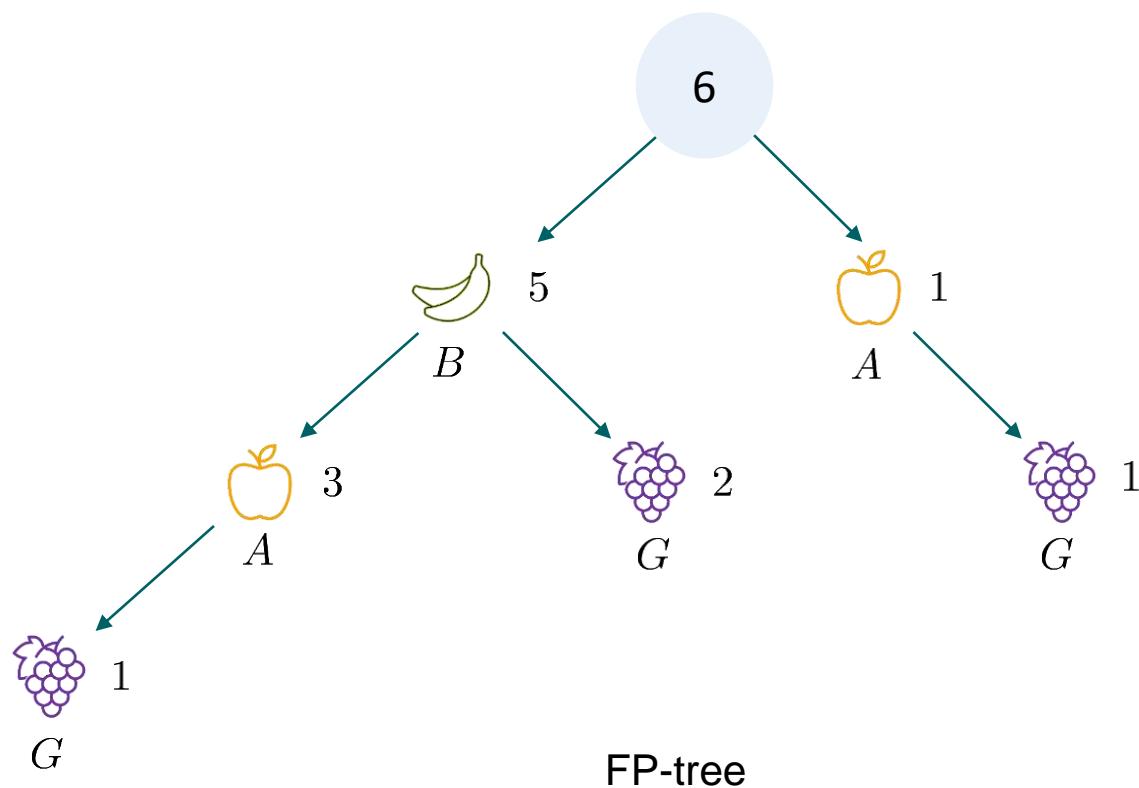


Conditional FP-Tree for Postfix B



Itemsets	Support Count
{B}	5
{G}, {A}	4
{B, G}, {B, A}	3

All Frequent Itemsets Generated



Itemsets	Support Count
{B}	5
{G}, {A}	4
{B, G}, {B, A}	3

Frequent itemsets mined

FP-Growth Algorithm – Summary

- Idea: frequent pattern growth based on FP-tree
- Method:
 - Construct the FP-tree from the dataset (previous video)
 - For each frequent item, construct its conditional pattern-base, and then its conditional FP-tree
 - Recursively repeat the process on each newly created conditional FP-tree until the tree is empty

FP-Growth Algorithm – Summary

- Advantages:
 - ✓ Only two passes through the dataset are needed (when constructing the tree)
 - ✓ Avoiding testing many hopeless candidates
 - ✓ Very fast when FP-tree fits in main memory
- However: approach has problems when FP-tree is too large to fit into memory

Frequent Itemsets – Summary

- Pattern mining is a form of unsupervised learning
- Frequent itemsets are the basis for finding patterns (ideas can be transferred to other patterns)
- Two well-known algorithms using generally applicable concepts:
 - Apriori algorithm
 - FP-growth algorithm
- Outlook
 - There may be many frequent “patterns”
 - How to determine which ones are surprising / interesting?

Association Rules – Preview

(one of the topics of the next lecture)

- $\{\text{Cheese, Bread}\} \Rightarrow \{\text{Milk}\}$

People that buy Cheese and Bread also tend to buy Milk.

- $\{\text{Track1, Track2}\} \Rightarrow \{\text{Track3}\}$

Students that take the Track 1 and Track 2 modules of BridgingAI also tend to take the Track 3 courses. (We hope you do!)

- $\{\text{Bitburger}\} \Rightarrow \{\text{Heineken, Palm}\}$

People that buy Bitburger beer tend to buy both Heineken and Palm beer.

- $\{\text{Carbonara, Margherita }\} \Rightarrow \{\text{Espresso, Tiramisu}\}$

People that buy Carbonara and Margherita also tend to buy Espresso and Tiramisu.

- $\{\text{part-245, part-345, part-456}\} \Rightarrow \{\text{part-372}\}$

When Parts 245, 345, and 456 are replaced, then often also Part 372 is replaced.

Elements of Machine Learning & Data Science

Association Rules and Sequence Mining

Lecture 10

Prof. Wil van der Aalst

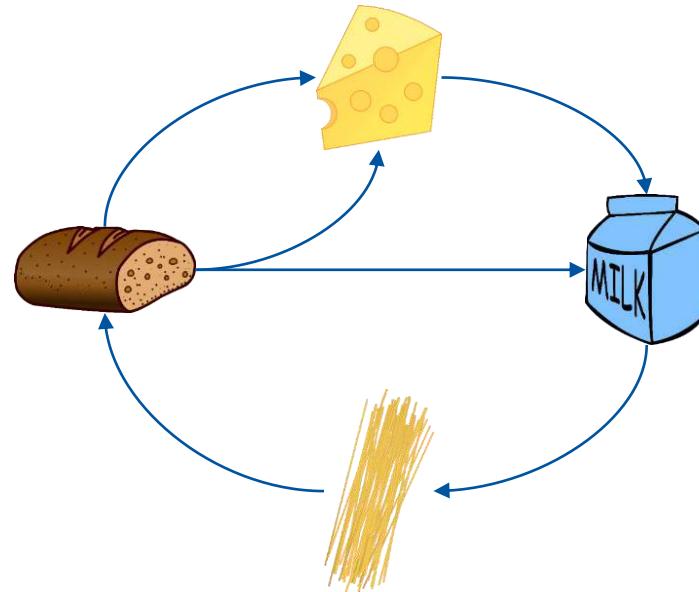
Marco Pegoraro, M.Sc.

Christopher Schwanen, M.Sc.

Tsunghao Huang, M.Sc.

Association Rules

1. Introduction
2. Generating Association Rules
3. Applications
4. Evaluation
5. Simpson's Paradox



From Frequent Itemsets to Association Rules

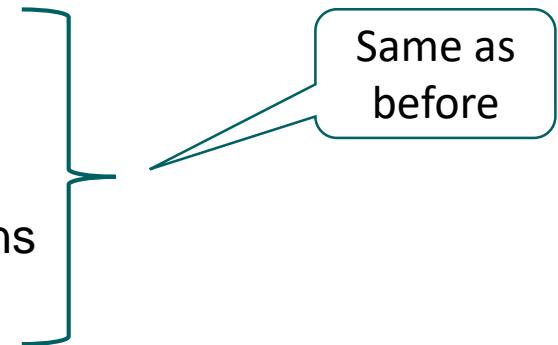
- Frequent Itemsets – a combinatorial explosion
- How to determine the interesting ones?
- How to turn itemsets into rules?



A Larger Supermarket May Have Up To 50000 Distinct Items

Association Rules - Notation

- $\mathcal{I} = \{I_1, I_2, \dots, I_D\}$ is the set of all possible items
- A transaction $\mathcal{T} \in \mathbb{P}(\mathcal{I}) \setminus \{\emptyset\}$ is a non-empty itemset
- A dataset $\mathcal{X} \in \mathbb{M}(\mathbb{P}(\mathcal{I}))$ (such that $\emptyset \notin \mathcal{X}$) is a multiset of transactions
(Here, \mathbb{M} is the multiset and \mathbb{P} is the powerset operator)
- $\mathcal{A} \Rightarrow \mathcal{B}$ with $\mathcal{A} \subseteq \mathcal{I}, \mathcal{B} \subseteq \mathcal{I}$ and $\mathcal{A} \cap \mathcal{B} = \emptyset$ is an association rule
- For example, $\{\text{Cheese, Bread}\} \Rightarrow \{\text{Milk}\}$



$$\mathcal{A} \Rightarrow \mathcal{B}$$

Association Rules - Preview

- $\{\text{Cheese, Bread}\} \Rightarrow \{\text{Milk}\}$

People that buy Cheese and Bread also tend to buy Milk.

- $\{\text{Track1, Track2}\} \Rightarrow \{\text{Track3}\}$

Students that take the Track 1 and Track 2 modules of BridgingAI also tend to take the Track 3 courses. (We hope you do!)

- $\{\text{Bitburger}\} \Rightarrow \{\text{Heineken, Palm}\}$

People that buy Bitburger beer tend to buy both Heineken and Palm beer.

- $\{\text{Carbonara, Margherita }\} \Rightarrow \{\text{Espresso, Tiramisu}\}$

People that buy Carbonara and Margherita also tend to buy Espresso and Tiramisu.

- $\{\text{part-245, part-345, part-456}\} \Rightarrow \{\text{part-372}\}$

When Parts 245, 345, and 456 are replaced, then often also Part 372 is replaced.

Support and Confidence

- **Support:** fraction of instances containing all items in $\mathcal{A} \cup \mathcal{B}$

$$\text{support}(\mathcal{A} \Rightarrow \mathcal{B}) = \text{support}(\mathcal{A} \cup \mathcal{B}) = \frac{\text{support_count}(\mathcal{A} \cup \mathcal{B})}{\text{support_count}(\emptyset)} = \frac{|[\mathcal{T} \in \mathcal{X} | \mathcal{A} \cup \mathcal{B} \subseteq \mathcal{T}]|}{|\mathcal{X}|}$$

Support and Confidence

- **Support:** fraction of instances containing all items in $\mathcal{A} \cup \mathcal{B}$

$$\text{support}(\mathcal{A} \Rightarrow \mathcal{B}) = \text{support}(\mathcal{A} \cup \mathcal{B}) = \frac{\text{support_count}(\mathcal{A} \cup \mathcal{B})}{\text{support_count}(\emptyset)} = \frac{|[\mathcal{T} \in \mathcal{X} | \mathcal{A} \cup \mathcal{B} \subseteq \mathcal{T}]|}{|\mathcal{X}|}$$

- **Confidence:** fraction of instances containing items in \mathcal{A} which contain items in $\mathcal{A} \cup \mathcal{B}$

$$\text{conf}(\mathcal{A} \Rightarrow \mathcal{B}) = \frac{\text{support}(\mathcal{A} \cup \mathcal{B})}{\text{support}(\mathcal{A})} = \frac{\text{support_count}(\mathcal{A} \cup \mathcal{B})}{\text{support_count}(\mathcal{A})} = \frac{|[\mathcal{T} \in \mathcal{X} | \mathcal{A} \cup \mathcal{B} \subseteq \mathcal{T}]|}{|[\mathcal{T} \in \mathcal{X} | \mathcal{A} \subseteq \mathcal{T}]|}$$

Support and Confidence - Example



ID	Bought Items
1	{Bread, Cheese, Milk, Pasta}
2	{Bread, Cheese, Chips}
3	{Cheese, Pasta, Milk}
4	{Bread, Cheese, Milk}
5	{Bread, Pasta}

All three items Bread, Cheese and Milk
need to be in the transaction to count

$$\text{support}(\{\text{Bread}\} \Rightarrow \{\text{Cheese}, \text{Milk}\}) = \text{support}(\{\text{Bread}, \text{Cheese}, \text{Milk}\}) = \frac{2}{5}$$

Support and Confidence - Example



ID	Bought Items
1	{Bread, Cheese, Milk, Pasta}
2	{Bread, Cheese, Chips}
3	{Cheese, Pasta, Milk}
4	{Bread, Cheese, Milk}
5	{Bread, Pasta}

$$\text{support}(\{\text{Bread}\} \Rightarrow \{\text{Cheese}, \text{Milk}\}) = \text{support}(\{\text{Bread}, \text{Cheese}, \text{Milk}\}) = \frac{2}{5}$$

$$\text{support}(\{\text{Bread}\} \Rightarrow \{\text{Cheese}, \text{Milk}\}) = \text{support}(\{\text{Cheese}, \text{Milk}\} \Rightarrow \{\text{Bread}\})$$

$$\text{support}(\{\text{Bread}\} \Rightarrow \{\text{Cheese}, \text{Milk}\}) = \text{support}(\{\text{Bread}, \text{Cheese}\} \Rightarrow \{\text{Milk}\})$$

Symmetric: moving the item does not change the value

Support and Confidence - Example



ID	Bought Items
1	{Bread, Cheese, Milk, Pasta}
2	{Bread, Cheese, Chips}
3	{Cheese, Pasta, Milk}
4	{Bread, Cheese, Milk}
5	{Bread, Pasta}

$$\text{conf}(\{\text{Bread}\} \Rightarrow \{\text{Cheese}, \text{Milk}\}) = \frac{\text{support}(\{\text{Bread}, \text{Cheese}, \text{Milk}\})}{\text{support}(\{\text{Bread}\})} = \frac{2}{4}$$

Support and Confidence - Example



ID	Bought Items
1	{Bread, Cheese, Milk, Pasta}
2	{Bread, Cheese, Chips}
3	{Cheese, Pasta, Milk}
4	{Bread, Cheese, Milk}
5	{Bread, Pasta}

$$\text{conf}(\{\text{Bread}\} \Rightarrow \{\text{Cheese}, \text{Milk}\}) = \frac{\text{support}(\{\text{Bread}, \text{Cheese}, \text{Milk}\})}{\text{support}(\{\text{Bread}\})} = \frac{2}{4}$$

$$\text{conf}(\{\text{Cheese}, \text{Milk}\} \Rightarrow \{\text{Bread}\}) = \frac{\text{support}(\{\text{Bread}, \text{Cheese}, \text{Milk}\})}{\text{support}(\{\text{Cheese}, \text{Milk}\})} = \frac{2}{3}$$

$$\text{conf}(\{\text{Bread}\} \Rightarrow \{\text{Cheese}, \text{Milk}\}) \neq \text{conf}(\{\text{Cheese}, \text{Milk}\} \Rightarrow \{\text{Bread}\})$$

Not symmetric
(equality holds only
in some rare cases)

Support and Confidence - Example



ID	Bought Items
1	{Bread, Cheese, Milk, Pasta}
2	{Bread, Cheese, Chips}
3	{Cheese, Pasta, Milk}
4	{Bread, Cheese, Milk}
5	{Bread, Pasta}

$$\text{conf}(\{\text{Bread}\} \Rightarrow \{\text{Cheese}, \text{Milk}\}) = \frac{\text{support}(\{\text{Bread}, \text{Cheese}, \text{Milk}\})}{\text{support}(\{\text{Bread}\})} = \frac{2}{4}$$

$$\text{conf}(\{\text{Bread}, \text{Cheese}\} \Rightarrow \{\text{Milk}\}) = \frac{\text{support}(\{\text{Bread}, \text{Cheese}, \text{Milk}\})}{\text{support}(\{\text{Bread}, \text{Cheese}\})} = \frac{2}{3}$$

General rule:

$$\text{conf}(\{A, B\} \Rightarrow \{C\}) \geq \text{conf}(\{A\} \Rightarrow \{B, C\})$$

Probabilistic Interpretation

- **Support:** probability that an instance contains $\mathcal{A} \cup \mathcal{B}$

$$\text{support}(\mathcal{A} \Rightarrow \mathcal{B}) = \text{support}(\mathcal{A} \cup \mathcal{B}) \approx P(\mathcal{A} \cup \mathcal{B})$$

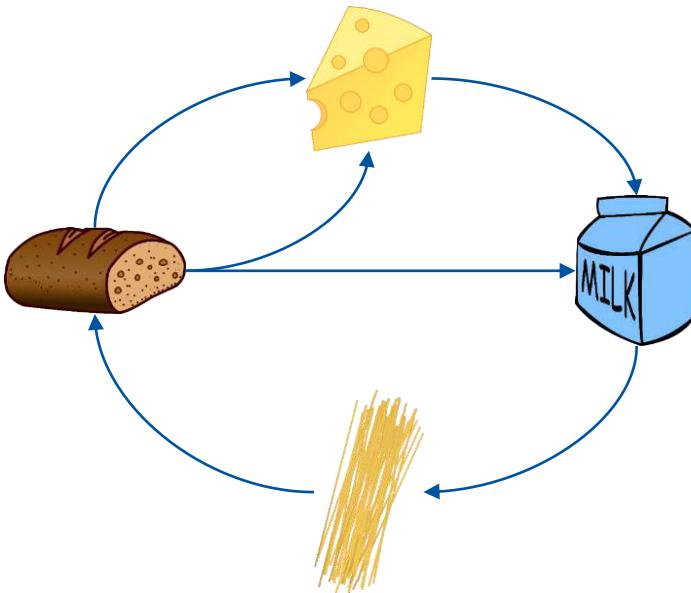
- **Confidence:** conditional probability that an instance contains items in \mathcal{B} , given that it contains items in \mathcal{A}

$$\text{conf}(\mathcal{A} \Rightarrow \mathcal{B}) = \frac{\text{support}(\mathcal{A} \cup \mathcal{B})}{\text{support}(\mathcal{A})} \approx P(\mathcal{B} | \mathcal{A})$$

Take ‘probability’ with a grain of salt - we are only considering a sample.

Association Rules

1. Introduction
2. **Generating Association Rules**
3. Applications
4. Evaluation
5. Simpson's Paradox



From Frequent Itemsets to Association Rules

Given: a dataset $\mathcal{X} \in \mathbb{M}(\mathbb{P}(\mathcal{I}))$, min_sup, min_conf

How to generate all association rules that have high support and high confidence?

$$\text{support}(\mathcal{A} \Rightarrow \mathcal{B}) = \text{support}(\mathcal{A} \cup \mathcal{B}) \geq \text{min_sup}$$

$$\text{conf}(\mathcal{A} \Rightarrow \mathcal{B}) = \frac{\text{support}(\mathcal{A} \cup \mathcal{B})}{\text{support}(\mathcal{A})} \geq \text{min_conf}$$

Ensuring $\text{support}(\mathcal{A} \Rightarrow \mathcal{B}) \geq \text{min_sup}$

✓ Easy!

- Use frequent itemsets as a basis
- Consider frequent itemsets $\mathcal{C} = \mathcal{A} \cup \mathcal{B}$ such that $|\mathcal{C}| \geq 2$ and $\mathcal{C} \geq \text{min_sup}$
(apply Apriori or FP-growth to generate such frequent itemsets)

Ensuring $\text{support}(\mathcal{A} \Rightarrow \mathcal{B}) \geq \text{min_sup}$

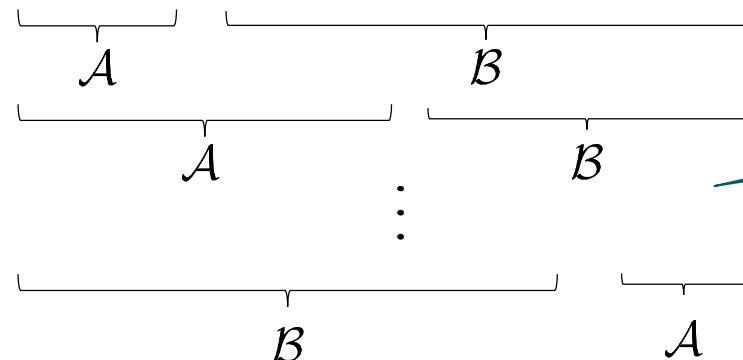
- ✓ Easy!
 - Use frequent itemsets as a basis
 - Consider frequent itemsets $\mathcal{C} = \mathcal{A} \cup \mathcal{B}$ such that $|\mathcal{C}| \geq 2$ and $\text{support}(\mathcal{C}) \geq \text{min_sup}$
(apply Apriori or FP-growth to generate such frequent itemsets)
 - Generate candidate rules $\mathcal{A} \Rightarrow \mathcal{B}$ by considering all splits of \mathcal{C} into two non-empty disjoint subsets
 - **However:** the number of such candidate rules is $2^{|\mathcal{C}|} - 2$!

Ensuring $\text{support}(\mathcal{A} \Rightarrow \mathcal{B}) \geq \text{min_sup}$

✓ Easy!

- Use frequent itemsets as a basis
- Consider frequent itemsets $\mathcal{C} = \mathcal{A} \cup \mathcal{B}$ such that $|\mathcal{C}| \geq 2$ and $\mathcal{C} \geq \text{min_sup}$ (apply Apriori or FP-growth to generate such frequent itemsets)
- Generate candidate rules $\mathcal{A} \Rightarrow \mathcal{B}$ by considering all splits of \mathcal{C} into two non-empty disjoint subsets
- **However:** the number of such candidate rules is $2^{|\mathcal{C}|} - 2$!

$$\mathcal{C} = \{\{\text{Bread}\}, \{\text{Cheese}\}, \{\text{Milk}\}, \{\text{Pasta}\}\}$$



$$|\mathcal{C}| = 4 \implies 2^4 - 2 = 14 \text{ candidate rules}$$

... and the number of candidate frequent itemsets was already exponential!

Ensuring $\text{conf}(\mathcal{A} \Rightarrow \mathcal{B}) \geq \text{min_conf}$

- Itemsets $\mathcal{A} \cup \mathcal{B}$ and \mathcal{A} are frequent
→ their supports have already been computed when using Apriori or FP-growth
- Therefore, we can simply test every candidate rule and only return the ones that satisfy the criterion:

$$\text{conf}(\mathcal{A} \Rightarrow \mathcal{B}) = \frac{\text{support}(\mathcal{A} \cup \mathcal{B})}{\text{support}(\mathcal{A})} \geq \text{min_conf}$$

No additional pass over the data needed

Ensuring $\text{conf}(\mathcal{A} \Rightarrow \mathcal{B}) \geq \text{min_conf}$

- Itemsets $\mathcal{A} \cup \mathcal{B}$ and \mathcal{A} are frequent
→ their supports have already been computed when using Apriori or FP-growth
- Therefore, we can simply test every candidate rule and only return the ones that satisfy the criterion:

$$\text{conf}(\mathcal{A} \Rightarrow \mathcal{B}) = \frac{\text{support}(\mathcal{A} \cup \mathcal{B})}{\text{support}(\mathcal{A})} \geq \text{min_conf}$$

But...

- There could be way too many association rules.
- **Most are not interesting!**

Confidence-Based Pruning

- Consider association rule $\mathcal{A} \Rightarrow \mathcal{B}$, and itemset \mathcal{C} such that $\mathcal{C} \cap (\mathcal{A} \cup \mathcal{B}) = \emptyset$
- It holds that $\text{conf}(\mathcal{A} \Rightarrow \mathcal{B} \cup \mathcal{C}) = \frac{\text{support}(\mathcal{A} \cup \mathcal{B} \cup \mathcal{C})}{\text{support}(\mathcal{A})} \leq \frac{\text{support}(\mathcal{A} \cup \mathcal{B})}{\text{support}(\mathcal{A})} = \text{conf}(\mathcal{A} \Rightarrow \mathcal{B})$

recall that the support of a superset is lower or equal

Confidence-Based Pruning

- Consider association rule $\mathcal{A} \Rightarrow \mathcal{B}$, and itemset \mathcal{C} such that $\mathcal{C} \cap (\mathcal{A} \cup \mathcal{B}) = \emptyset$
- It holds that $\text{conf}(\mathcal{A} \Rightarrow \mathcal{B} \cup \mathcal{C}) = \frac{\text{support}(\mathcal{A} \cup \mathcal{B} \cup \mathcal{C})}{\text{support}(\mathcal{A})} \leq \frac{\text{support}(\mathcal{A} \cup \mathcal{B})}{\text{support}(\mathcal{A})} = \text{conf}(\mathcal{A} \Rightarrow \mathcal{B})$
- Hence, if $\text{conf}(\mathcal{A} \Rightarrow \mathcal{B}) \leq \text{min_conf}$ then $\text{conf}(\mathcal{A} \Rightarrow \mathcal{B} \cup \mathcal{C}) \leq \text{min_conf}$
- Adding \mathcal{C} to the **right** part makes the rule **stronger**
- We can **focus on the stronger rules** meeting the confidence threshold
- This does not apply to $\text{conf}(\mathcal{A} \cup \mathcal{C} \Rightarrow \mathcal{B}) \quad ?? \quad \text{conf}(\mathcal{A} \Rightarrow \mathcal{B})$
- Additions to the left part of the rule may lead to an **increase** or **decrease**
 - $\{\text{Cheese}\} \Rightarrow \{\text{Wine}\}$ may have a confidence of 0.2
 - $\{\text{Cheese}, \text{Babyfood}\} \Rightarrow \{\text{Wine}\}$ may have a confidence of 0.1
 - $\{\text{Cheese}, \text{Chips}\} \Rightarrow \{\text{Wine}\}$ may have a confidence of 0.3

Removing Redundant Rules

- Consider two different association rules $\mathcal{A} \Rightarrow \mathcal{B}$ and $\mathcal{A}' \Rightarrow \mathcal{B}'$ with **identical** support and confidence, i.e.:
 - $\text{support}(\mathcal{A} \Rightarrow \mathcal{B}) = \text{support}(\mathcal{A}' \Rightarrow \mathcal{B}')$
 - $\text{conf}(\mathcal{A} \Rightarrow \mathcal{B}) = \text{conf}(\mathcal{A}' \Rightarrow \mathcal{B}')$
- $\mathcal{A}' \Rightarrow \mathcal{B}'$ is **redundant** if $\mathcal{A}' \subseteq \mathcal{A}$ and $\mathcal{B}' \subseteq \mathcal{B}$
- Using only **closed** frequent itemsets will avoid generating redundant rules
(Recall: An itemset is closed if there is no proper superset that has the same support)

Avoiding Generation of Redundant Rules

1. Assume $\mathcal{A}' \Rightarrow \mathcal{B}'$ is redundant, i.e., there is another rule $\mathcal{A} \Rightarrow \mathcal{B}$ such that
 - $\text{support}(\mathcal{A} \Rightarrow \mathcal{B}) = \text{support}(\mathcal{A}' \Rightarrow \mathcal{B}')$
 - $\text{conf}(\mathcal{A} \Rightarrow \mathcal{B}) = \text{conf}(\mathcal{A}' \Rightarrow \mathcal{B}')$
 - $\mathcal{A}' \subseteq \mathcal{A}$
 - $\mathcal{B}' \subseteq \mathcal{B}$
 - It holds that $\mathcal{A}' \cup \mathcal{B}' \subset \mathcal{A} \cup \mathcal{B}$ (because the rules are different)

Avoiding Generation of Redundant Rules

1. Assume $\mathcal{A}' \Rightarrow \mathcal{B}'$ is redundant, i.e., there is another rule $\mathcal{A} \Rightarrow \mathcal{B}$ such that
 - $\text{support}(\mathcal{A} \Rightarrow \mathcal{B}) = \text{support}(\mathcal{A}' \Rightarrow \mathcal{B}')$
 - $\text{conf}(\mathcal{A} \Rightarrow \mathcal{B}) = \text{conf}(\mathcal{A}' \Rightarrow \mathcal{B}')$
 - $\mathcal{A}' \subseteq \mathcal{A}$
 - $\mathcal{B}' \subseteq \mathcal{B}$
 - It holds that $\mathcal{A}' \cup \mathcal{B}' \subset \mathcal{A} \cup \mathcal{B}$ (because the rules are different)
2. Also, assume $\mathcal{A} \cup \mathcal{B}$ and $\mathcal{A}' \cup \mathcal{B}'$ are closed, i.e., there are no proper supersets with the same support
 - Hence, $\text{support}(\mathcal{A}' \Rightarrow \mathcal{B}') > \text{support}(\mathcal{A} \Rightarrow \mathcal{B})$ (cannot be equal, $\mathcal{A} \cup \mathcal{B}$ is closed)

Therefore, we find a contradiction. Closed itemsets cannot produce redundant rules.

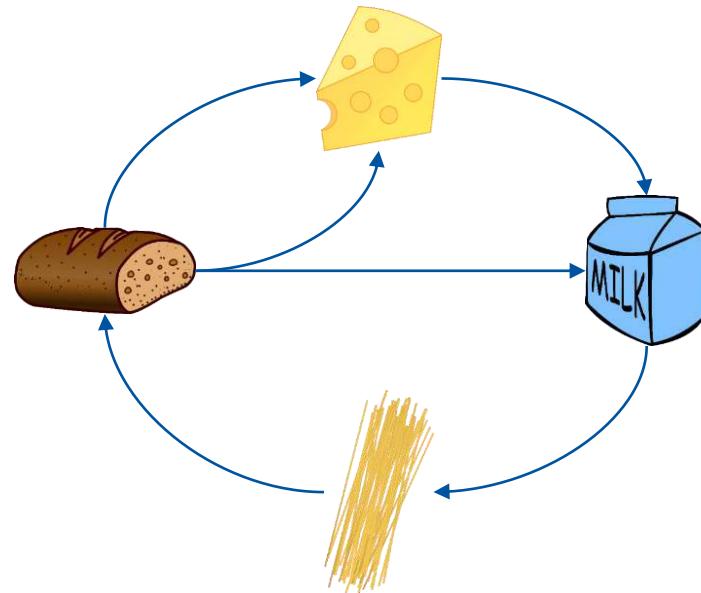
Summary

How to generate association rules that are interesting?

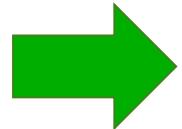
- We can generate candidate rules with high support based on frequent itemsets
- We can filter those candidates with high confidence without going back to the data
- We can prune the rules based on confidence: $\text{min_conf} \leq \text{conf}(\mathcal{A} \Rightarrow \mathcal{B} \cup \mathcal{C}) \leq \text{conf}(\mathcal{A} \Rightarrow \mathcal{B})$
- We can focus on closed frequent itemsets to avoid redundant rules
- Not enough, we need additional concepts such as “surprisingness” (lift)

Association Rules

1. Introduction
2. Generating Association Rules
3. **Applications**
4. Evaluation
5. Simpson's Paradox



Spotify



{Flowers(Miley Cyrus), Unholy(Sam Smith)} \Rightarrow {Levitating(Dua Lipa)}

{One(Metallica), Trasher(Evile)} \Rightarrow {Augen-Auf(Oomph), The Trooper(Iron Maiden)}

{Birds(Anouk), Irgendwo(Nena)} \Rightarrow {Leiser(Lea), Klavier(Lea)}

- 456 million active listeners
- 195 million premium subscribers
- Over 80 million songs

(As of January 2023)

Amazon



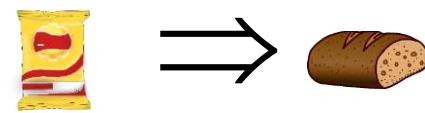
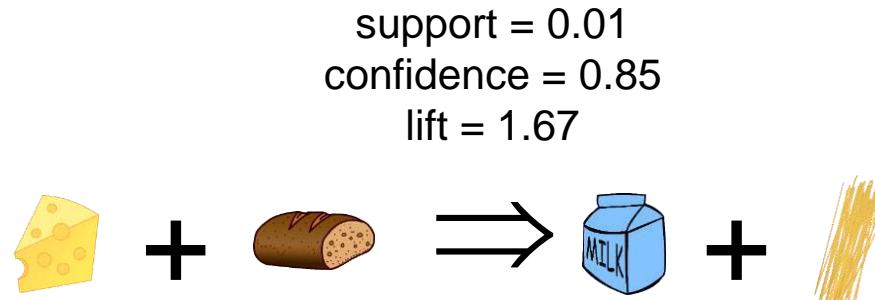
$\{ \text{Echo-Show-8}, \text{Fire-TV-Cube} \} \Rightarrow \{ \text{Kindle-Paperwhite} \}$

$\{ \text{Fire-TV-Stick-8} \} \Rightarrow \{ \text{Fire-HD-8}, \text{Blink-Mini} \}$

- 300 million active users
- Over 2 million third-party seller businesses
- Around 350 million items on the marketplace

(As of January 2023)

Supermarkets



support = 0.001
confidence = 0.15
lift = 1.2

Next to confidence and support, we will see other measures like lift

Using Features Values As Items And Instances As Itemsets

Rain	Wind	Temp	Play
Yes	Yes	15	No
No	No	34	Yes
Yes	No	23	Yes
Yes	Yes	20	Yes
No	Yes	28	No
...

- Examples consider items as products, services, etc.
- Items can also be normal features values and transactions normal instances
- This leads to itemsets of the form $\{f_1=v_1, f_2=v_2, \dots, f_n=v_n\}$ for each instance

Using Features Values As Items And Instances As Itemsets

Rain	Wind	Temp	Play
Yes	Yes	15	No
No	No	34	Yes
Yes	No	23	Yes
Yes	Yes	20	Yes
No	Yes	28	No
...

```
[{Rain=Yes, Wind=Yes, Temp=15, Play=No},  
 {Rain=No, Wind=No, Temp=34, Play=Yes},  
 {Rain=Yes, Wind=No, Temp=23, Play=Yes},  
 {Rain=Yes, Wind=Yes, Temp=20, Play=Yes},  
 {Rain=No, Wind=Yes, Temp=28, Play=No},  
 ...]
```

- Examples consider items as products, services, etc.
- Items can also be normal features values and transactions normal instances
- This leads to itemsets of the form $\{f_1=v_1, f_2=v_2, \dots, f_n=v_n\}$ for each instance

Using Features Values As Items And Instances As Itemsets

Rain	Wind	Temp	Play
Yes	Yes	15	No
No	No	34	Yes
Yes	No	23	Yes
Yes	Yes	20	Yes
No	Yes	28	No
...

[{Rain=Yes, Wind=Yes, 10≤Temp<20, Play=No},
 {Rain=No, Wind=No, 30≤Temp<40, Play=Yes},
 {Rain=Yes, Wind=No, 20≤Temp<30, Play=Yes},
 {Rain=Yes, Wind=Yes, 20≤Temp<30, Play=Yes},
 {Rain=No, Wind=Yes, 20≤Temp<30, Play=No},
 ...]

- Items can also be ranges for continuous feature values
 - $\text{Temp} \geq 25$
 - $\text{Temp} < 25$
 - $20 \leq \text{Temp} < 30$
 - Etc.
- Any dataset having instances and features can be converted into a multiset of transactions $\mathcal{X} \in \mathbb{M}(\mathbb{P}(\mathcal{I}))$

Using Features Values As Items And Instances As Itemsets

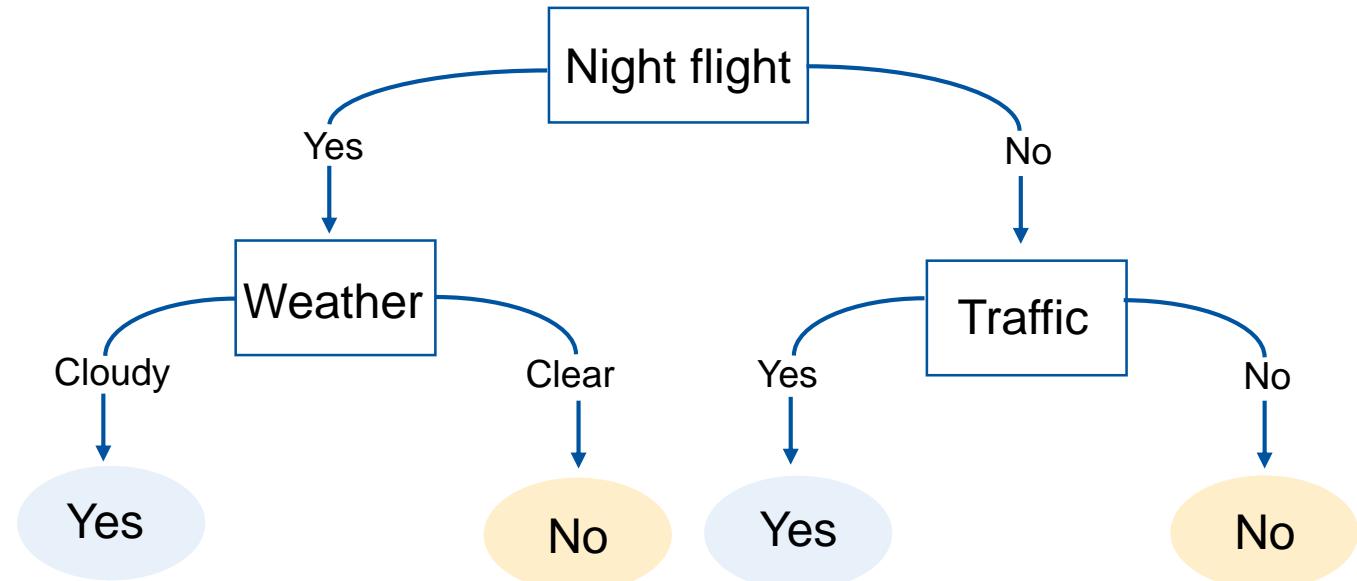
Rain	Wind	Temp	Play
Yes	Yes	15	No
No	No	34	Yes
Yes	No	23	Yes
Yes	Yes	20	Yes
No	Yes	28	No
...

- Any dataset having instances and features can be converted into a multiset of transactions $\mathcal{X} \in \mathbb{M}(\mathbb{P}(\mathcal{I}))$
- Hence, we can also have association rules of the form
 $\mathcal{A} \Rightarrow \mathcal{B}$ with $\mathcal{A} \subseteq \mathcal{I}, \mathcal{B} \subseteq \mathcal{I}$ and $\mathcal{A} \cap \mathcal{B} = \emptyset$

$\{\text{Rain=Yes, Wind=Yes}\} \Rightarrow \{\text{Play=No}\}$
 $\{\text{Temp}>30\} \Rightarrow \{\text{Rain=No, Wind=No}\}$
 $\{\text{Temp}>20, \text{Play=Yes}\} \Rightarrow \{\text{Wind=No}\}$

Link To Classification and Decision Trees

Weather	Traffic	Night flight	Flight delayed
Cloudy	No	Yes	Yes
Cloudy	Yes	No	Yes
Cloudy	Yes	No	Yes
Clear	Yes	Yes	No
Clear	No	No	No
Clear	No	No	No



{Night_flight=Yes, Weather=Cloudy} \Rightarrow {Flight_delayed=Yes}

{Night_flight=Yes, Weather=Clear} \Rightarrow {Flight_delayed=No}

{Night_flight=No, Traffic=Yes} \Rightarrow {Flight_delayed=Yes}

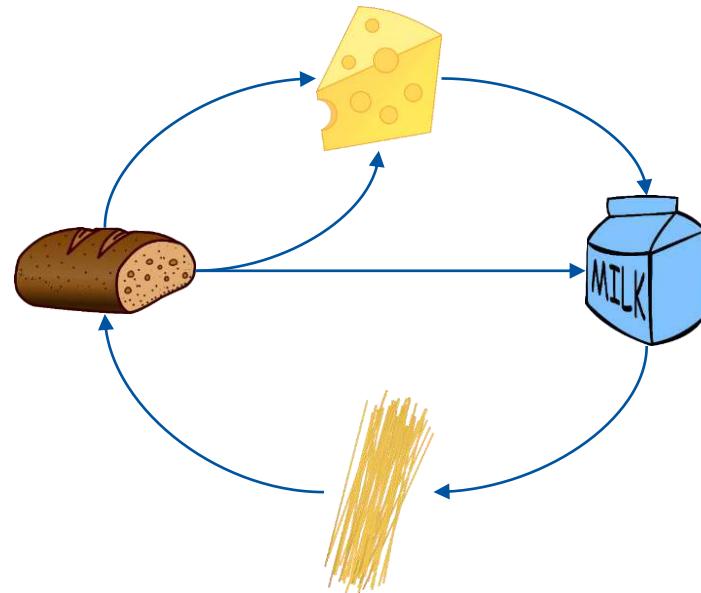
{Night_flight=No, Traffic=No} \Rightarrow {Flight_delayed=No}

Summary

- Association rules can be learned for “normal itemsets” and itemsets based on feature values
- Classification rules can be expressed as association rules
- The challenge remains that there are exponentially many candidate rules
- Confidence and support are only part of the story
 - What if many rules meet the two thresholds?
 - How to select the most interesting ones?

Association Rules

1. Introduction
2. Generating Association Rules
3. Applications
4. **Evaluation**
5. Simpson's Paradox



Association rules $\mathcal{A} \Rightarrow \mathcal{B}$

{Cheese, Chips} \Rightarrow {Wine, Beer}

{One(Metallica), Trasher(Evile)} \Rightarrow {Augen-Auf(Oomph), The Trooper(Iron Maiden)}

{Temp>20, Play=Yes} \Rightarrow {Wind=No}

{Night_flight=No, Traffic=Yes} \Rightarrow {Flight_delayed=Yes}

{Gender=Male, Sport=Football} \Rightarrow {Favorite_food=Currywurst, Age>40}

...

How to evaluate the quality of a rule?

Confusion matrix for association rules

Consider association rule $\mathcal{A} \Rightarrow \mathcal{B}$

$\mathcal{A} \Rightarrow \mathcal{B}$	\mathcal{B} is included	\mathcal{B} is not included	
\mathcal{A} is included	$\#\mathcal{AB}$	$\#\mathcal{A}\bar{\mathcal{B}}$	$\#\mathcal{A}$
\mathcal{A} is not included	$\#\bar{\mathcal{A}}\mathcal{B}$	$\#\bar{\mathcal{A}}\bar{\mathcal{B}}$	$\#\bar{\mathcal{A}}$
	$\#\mathcal{B}$	$\#\bar{\mathcal{B}}$	$\#ALL$

$$\text{support}(\mathcal{A} \Rightarrow \mathcal{B}) = \frac{\#\mathcal{AB}}{\#ALL}$$

$$\text{conf}(\mathcal{A} \Rightarrow \mathcal{B}) = \frac{\#\mathcal{AB}}{\#\mathcal{A}}$$

Confusion matrix for association rules

Consider association rule $\mathcal{A} \Rightarrow \mathcal{B}$

$\mathcal{A} \Rightarrow \mathcal{B}$	\mathcal{B} is included	\mathcal{B} is not included	
\mathcal{A} is included	$\#\mathcal{AB}$	$\#\mathcal{A}\bar{\mathcal{B}}$	$\#\mathcal{A}$
\mathcal{A} is not included	$\#\bar{\mathcal{A}}\mathcal{B}$	$\#\bar{\mathcal{A}}\bar{\mathcal{B}}$	$\#\bar{\mathcal{A}}$
	$\#\mathcal{B}$	$\#\bar{\mathcal{B}}$	$\#ALL$

$$\text{support}(\mathcal{A} \Rightarrow \mathcal{B}) = \frac{\#\mathcal{AB}}{\#ALL}$$

$$\text{conf}(\mathcal{A} \Rightarrow \mathcal{B}) = \frac{\#\mathcal{AB}}{\#\mathcal{A}}$$

The lower the better

The higher the better

Not captured in any of the metrics

High Support and High Confidence

Consider association rule $\mathcal{A} \Rightarrow \mathcal{B}$

$\mathcal{A} \Rightarrow \mathcal{B}$	\mathcal{B} is included	\mathcal{B} is not included	
\mathcal{A} is included	100	0	100
\mathcal{A} is not included	0	0	0
	100	0	100

$$\text{support}(\mathcal{A} \Rightarrow \mathcal{B}) = \frac{100}{100}$$

$$\text{conf}(\mathcal{A} \Rightarrow \mathcal{B}) = \frac{100}{100}$$

Low Support and High Confidence

Consider association rule $\mathcal{A} \Rightarrow \mathcal{B}$

$\mathcal{A} \Rightarrow \mathcal{B}$	\mathcal{B} is included	\mathcal{B} is not included	
\mathcal{A} is included	10	0	10
\mathcal{A} is not included	40	50	90
	50	50	100

$$\text{support}(\mathcal{A} \Rightarrow \mathcal{B}) = \frac{10}{100}$$

$$\text{conf}(\mathcal{A} \Rightarrow \mathcal{B}) = \frac{10}{10}$$

Low Support and Low Confidence

Consider association rule $\mathcal{A} \Rightarrow \mathcal{B}$

$\mathcal{A} \Rightarrow \mathcal{B}$	\mathcal{B} is included	\mathcal{B} is not included	
\mathcal{A} is included	10	40	50
\mathcal{A} is not included	25	25	50
	35	65	100

$$\text{support}(\mathcal{A} \Rightarrow \mathcal{B}) = \frac{10}{100}$$

$$\text{conf}(\mathcal{A} \Rightarrow \mathcal{B}) = \frac{10}{50}$$

Support and Confidence Don't Tell The Full Story

Consider association rule $\mathcal{A} \Rightarrow \mathcal{B}$

$\mathcal{A} \Rightarrow \mathcal{B}$	\mathcal{B} is included	\mathcal{B} is not included	
\mathcal{A} is included	80	10	90
\mathcal{A} is not included	0	10	10
	80	20	100

$$\text{support}(\mathcal{A} \Rightarrow \mathcal{B}) = \frac{80}{100}$$

$$\text{conf}(\mathcal{A} \Rightarrow \mathcal{B}) = \frac{80}{90}$$

Seems to be a good rule
because if \mathcal{A} is not included,
 \mathcal{B} is also never included

Support and Confidence Don't Tell The Full Story

Consider association rule $\mathcal{A} \Rightarrow \mathcal{B}$

$\mathcal{A} \Rightarrow \mathcal{B}$	\mathcal{B} is included	\mathcal{B} is not included	
\mathcal{A} is included	80	10	90
\mathcal{A} is not included	10	0	10
	90	10	100

Not captured in any of the metrics

$$\text{support}(\mathcal{A} \Rightarrow \mathcal{B}) = \frac{80}{100}$$

$$\text{conf}(\mathcal{A} \Rightarrow \mathcal{B}) = \frac{80}{90}$$

Same support and confidence,
but seems to be a poor rule
because if \mathcal{A} is not included,
 \mathcal{B} is always included

The distribution of counts in the second row does not influence support and confidence

We need Lift: How surprising?

Consider association rule $\mathcal{A} \Rightarrow \mathcal{B}$

$\mathcal{A} \Rightarrow \mathcal{B}$	\mathcal{B} is included	\mathcal{B} is not included	
\mathcal{A} is included	$\#\mathcal{AB}$	$\#\mathcal{A}\bar{\mathcal{B}}$	$\#\mathcal{A}$
\mathcal{A} is not included	$\#\bar{\mathcal{A}}\mathcal{B}$	$\#\bar{\mathcal{A}}\bar{\mathcal{B}}$	$\#\bar{\mathcal{A}}$
	$\#\mathcal{B}$	$\#\bar{\mathcal{B}}$	#ALL

$$\text{lift}(\mathcal{A} \Rightarrow \mathcal{B}) = \frac{\text{support}(\mathcal{A} \cup \mathcal{B})}{\text{support}(\mathcal{A}) \cdot \text{support}(\mathcal{B})} = \frac{P(\mathcal{A} \cup \mathcal{B})}{P(\mathcal{A}) \cdot P(\mathcal{B})} = \frac{\frac{\#\mathcal{AB}}{\#\text{ALL}}}{\frac{\#\mathcal{A}}{\#\text{ALL}} \cdot \frac{\#\mathcal{B}}{\#\text{ALL}}} = \frac{\#\mathcal{AB} \cdot \#\text{ALL}}{\#\mathcal{A} \cdot \#\mathcal{B}}$$

We need Lift

Consider association rule $\mathcal{A} \Rightarrow \mathcal{B}$

$\mathcal{A} \Rightarrow \mathcal{B}$	\mathcal{B} is included	\mathcal{B} is not included	
\mathcal{A} is included	$\#\mathcal{AB}$	$\#\mathcal{A}\bar{\mathcal{B}}$	$\#\mathcal{A}$
\mathcal{A} is not included	$\#\bar{\mathcal{A}}\mathcal{B}$	$\#\bar{\mathcal{A}}\bar{\mathcal{B}}$	$\#\bar{\mathcal{A}}$
	$\#\mathcal{B}$	$\#\bar{\mathcal{B}}$	#ALL

$$\text{lift}(\mathcal{A} \Rightarrow \mathcal{B}) = \frac{\text{support}(\mathcal{A} \cup \mathcal{B})}{\text{support}(\mathcal{A}) \cdot \text{support}(\mathcal{B})} = \frac{P(\mathcal{A} \cup \mathcal{B})}{P(\mathcal{A}) \cdot P(\mathcal{B})} = \frac{\frac{\#\mathcal{AB}}{\#\text{ALL}}}{\frac{\#\mathcal{A}}{\#\text{ALL}} \cdot \frac{\#\mathcal{B}}{\#\text{ALL}}}$$

If $\text{lift}(\mathcal{A} \Rightarrow \mathcal{B}) \approx 1$ then \mathcal{A} and \mathcal{B} are **independent** $P(\mathcal{A} \cup \mathcal{B}) \approx P(\mathcal{A}) \cdot P(\mathcal{B})$

If $\text{lift}(\mathcal{A} \Rightarrow \mathcal{B}) \ll 1$ then \mathcal{A} and \mathcal{B} are **negatively correlated** $P(\mathcal{A} \cup \mathcal{B}) \ll P(\mathcal{A}) \cdot P(\mathcal{B})$

If $\text{lift}(\mathcal{A} \Rightarrow \mathcal{B}) \gg 1$ then \mathcal{A} and \mathcal{B} are **positively correlated** $P(\mathcal{A} \cup \mathcal{B}) \gg P(\mathcal{A}) \cdot P(\mathcal{B})$

Is the Rule Surprising?

Consider association rule $\mathcal{A} \Rightarrow \mathcal{B}$

$\mathcal{A} \Rightarrow \mathcal{B}$	\mathcal{B} is included	\mathcal{B} is not included	
\mathcal{A} is included	9	1	10
\mathcal{A} is not included	81	9	90
	90	10	100

$$\text{lift}(\mathcal{A} \Rightarrow \mathcal{B}) = \frac{\text{support}(\mathcal{A} \cup \mathcal{B})}{\text{support}(\mathcal{A}) \cdot \text{support}(\mathcal{B})} = \frac{P(\mathcal{A} \cup \mathcal{B})}{P(\mathcal{A}) \cdot P(\mathcal{B})} = \frac{\frac{\# \mathcal{AB}}{\#\text{ALL}}}{\frac{\#\mathcal{A}}{\#\text{ALL}} \cdot \frac{\#\mathcal{B}}{\#\text{ALL}}}$$

$$\text{support}(\mathcal{A} \Rightarrow \mathcal{B}) = \frac{9}{100}$$

$$\text{conf}(\mathcal{A} \Rightarrow \mathcal{B}) = \frac{9}{10}$$

$$\text{lift}(\mathcal{A} \Rightarrow \mathcal{B}) = \frac{\frac{9}{100}}{\frac{10}{100} \cdot \frac{90}{100}} = 1$$

No surprise!

Is the Rule Surprising?

Consider association rule $\mathcal{A} \Rightarrow \mathcal{B}$

$\mathcal{A} \Rightarrow \mathcal{B}$	\mathcal{B} is included	\mathcal{B} is not included	
\mathcal{A} is included	9	1	10
\mathcal{A} is not included	0	90	90
	9	91	100

$$\text{lift}(\mathcal{A} \Rightarrow \mathcal{B}) = \frac{\text{support}(\mathcal{A} \cup \mathcal{B})}{\text{support}(\mathcal{A}) \cdot \text{support}(\mathcal{B})} = \frac{P(\mathcal{A} \cup \mathcal{B})}{P(\mathcal{A}) \cdot P(\mathcal{B})} = \frac{\frac{\# \mathcal{AB}}{\#\text{ALL}}}{\frac{\#\mathcal{A}}{\#\text{ALL}} \cdot \frac{\#\mathcal{B}}{\#\text{ALL}}}$$

$$\text{support}(\mathcal{A} \Rightarrow \mathcal{B}) = \frac{9}{100}$$

$$\text{conf}(\mathcal{A} \Rightarrow \mathcal{B}) = \frac{9}{10}$$

$$\text{lift}(\mathcal{A} \Rightarrow \mathcal{B}) = \frac{\frac{9}{100}}{\frac{10}{100} \cdot \frac{9}{100}} = 10$$

Surprise!

Is the Rule Surprising?

Consider association rule $\mathcal{A} \Rightarrow \mathcal{B}$

$\mathcal{A} \Rightarrow \mathcal{B}$	\mathcal{B} is included	\mathcal{B} is not included	
\mathcal{A} is included	9	1	10
\mathcal{A} is not included	90	0	90
	99	1	100

$$\text{lift}(\mathcal{A} \Rightarrow \mathcal{B}) = \frac{\text{support}(\mathcal{A} \cup \mathcal{B})}{\text{support}(\mathcal{A}) \cdot \text{support}(\mathcal{B})} = \frac{P(\mathcal{A} \cup \mathcal{B})}{P(\mathcal{A}) \cdot P(\mathcal{B})} = \frac{\frac{\# \mathcal{AB}}{\#\text{ALL}}}{\frac{\#\mathcal{A}}{\#\text{ALL}} \cdot \frac{\#\mathcal{B}}{\#\text{ALL}}}$$

$$\text{support}(\mathcal{A} \Rightarrow \mathcal{B}) = \frac{9}{100}$$

$$\text{conf}(\mathcal{A} \Rightarrow \mathcal{B}) = \frac{9}{10}$$

$$\text{lift}(\mathcal{A} \Rightarrow \mathcal{B}) = \frac{\frac{9}{100}}{\frac{10}{100} \cdot \frac{99}{100}} = \frac{10}{11}$$

a little bit ...

Selecting Association rules

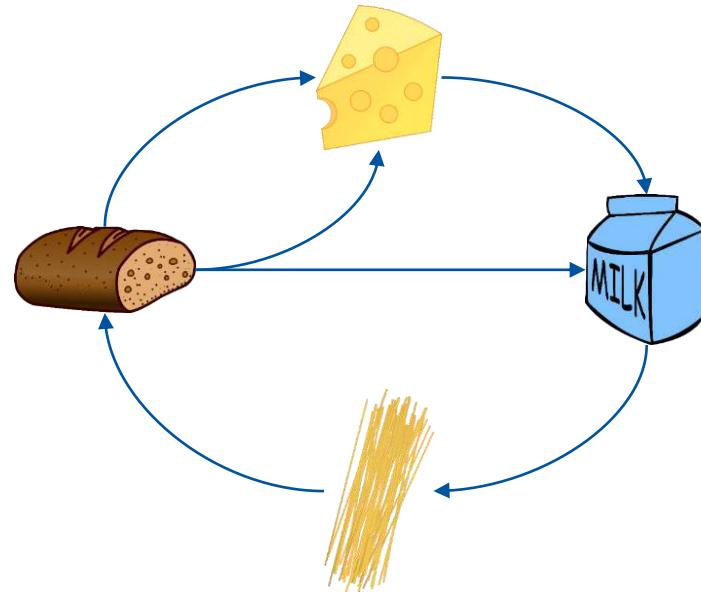
1. Set thresholds for minimal support and confidence
2. Evaluate lift and possibly other metrics for the rules remaining
3. Sort and prune based on any of the quality criteria (support, confidence, lift, etc.)

It is hard to predict the number of rules beforehand

There are many other measures of quality (conviction, leverage, collective strength, etc.)

Association Rules

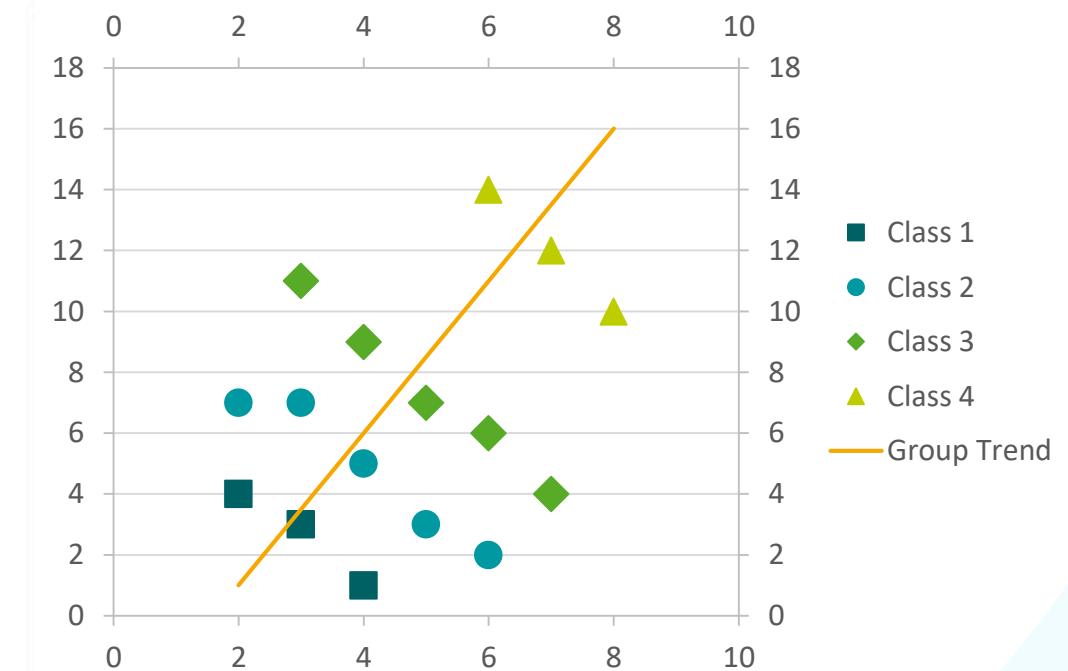
1. Introduction
2. Generating Association Rules
3. Applications
4. Evaluation
5. **Simpson's Paradox**



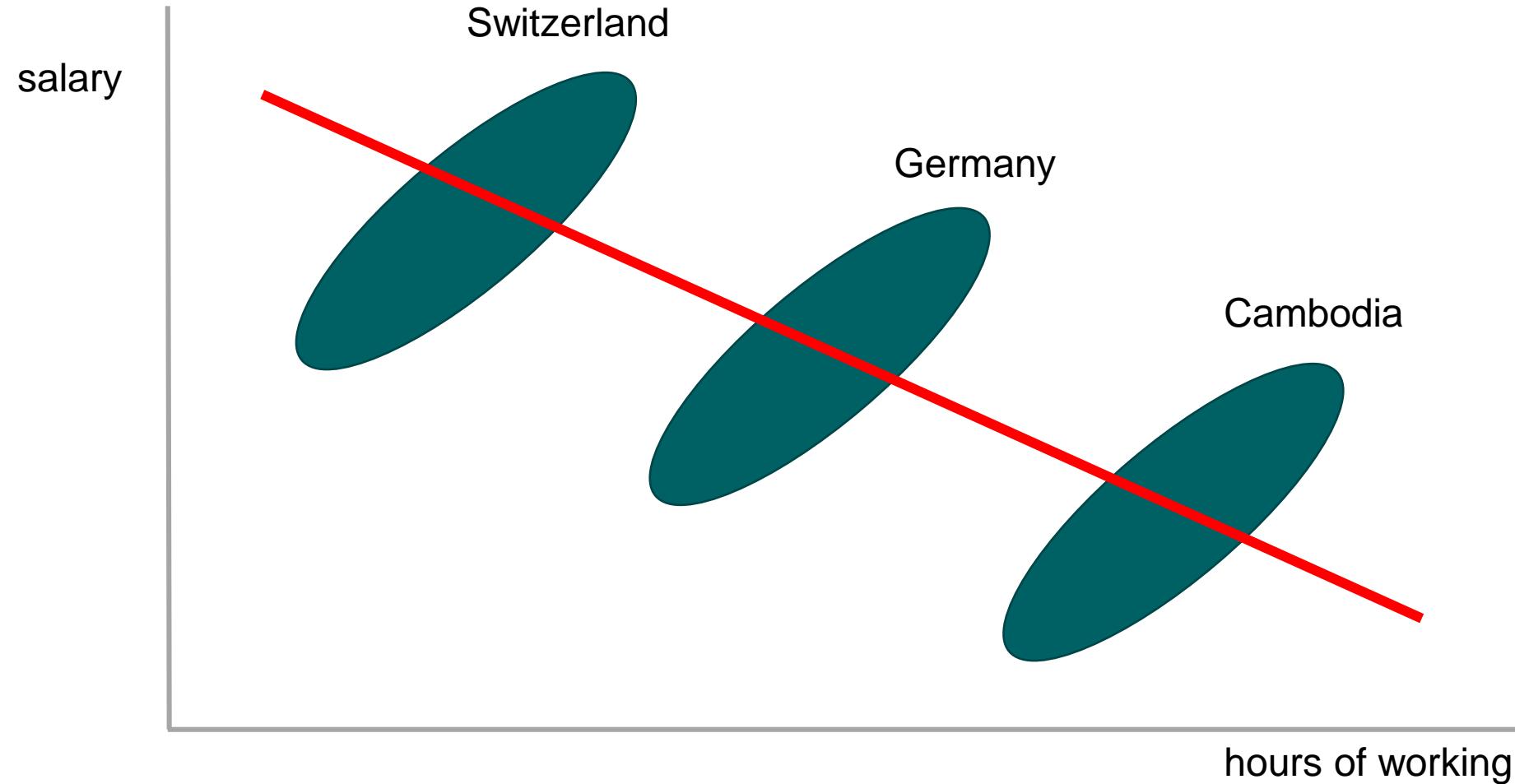
Simpson's Paradox

A trend appears in several different groups of data but **disappears** or **reverses** when these groups are combined.

- Edward Simpson in 1951 (earlier variants by Udny Yule and Karl Pearson)
- Nice example of 'How to lie with statistics?'
- The paradox is often encountered in social-science and medical-science



Simpson's Paradox When Using Regression



Simpson's Paradox in Association Rules

Consider the association rule $\mathcal{A} \Rightarrow \mathcal{B}$ and any feature which splits the instances (location, age ...)

$\mathcal{A} \Rightarrow \mathcal{B}$	\mathcal{B} is included	\mathcal{B} is not included	
\mathcal{A} is included	$a + p$	$(b - a) + (q - p)$	$b + q$
\mathcal{A} is not included	$c + r$	$(d - c) + (s - r)$	$d + s$
	$a + c + p + r$	$(b + d + q + s) - (a + c + p + r)$	$b + d + q + s$

Two classes – blue and orange
(e.g., old and young)

Simpson's Paradox in Association Rules

Consider the association rule $\mathcal{A} \Rightarrow \mathcal{B}$ and any feature which splits the instances (location, age ...)

$\mathcal{A} \Rightarrow \mathcal{B}$	\mathcal{B} is included	\mathcal{B} is not included	
\mathcal{A} is included	$a + p$	$(b - a) + (q - p)$	$b + q$
\mathcal{A} is not included	$c + r$	$(d - c) + (s - r)$	$d + s$
	$a + c + p + r$	$(b + d + q + s) - (a + c + p + r)$	$b + d + q + s$

$$\text{conf}(\mathcal{A} \Rightarrow \mathcal{B}) = \frac{a+p}{b+q}$$

$$\text{lift}(\mathcal{A} \Rightarrow \mathcal{B}) = \frac{\frac{a+p}{b+d+q+s}}{\frac{b+q}{b+d+q+s} \cdot \frac{a+c+p+r}{b+d+q+s}}$$

Simpson's Paradox in Association Rules

Consider the association rule $\mathcal{A} \Rightarrow \mathcal{B}$ and any feature which splits the instances (location, age ...)

$\mathcal{A} \Rightarrow \mathcal{B}$	\mathcal{B} is included	\mathcal{B} is not included	
\mathcal{A} is included	$a + p$	$(b - a) + (q - p)$	$b + q$
\mathcal{A} is not included	$c + r$	$(d - c) + (s - r)$	$d + s$
	$a + c + p + r$	$(b + d + q + s) - (a + c + p + r)$	$b + d + q + s$

$$\text{conf}(\mathcal{A} \Rightarrow \mathcal{B}) = \frac{a+p}{b+q}$$

$$\text{lift}(\mathcal{A} \Rightarrow \mathcal{B}) = \frac{(a+p) \cdot (b+d+q+s)}{(b+q) \cdot (a+c+p+r)}$$

Simpson's Paradox - Example

Two classes: **old** and **young**

$\text{smoke} \Rightarrow \text{cancer}$	has cancer	doesn't have cancer	
smokes	$1 + 66$	$2 + 34$	$3 + 100$
doesn't smoke	$34 + 2$	$66 + 1$	$100 + 3$
	$35 + 68$	$68 + 35$	$103 + 103$

humans $\text{conf}(\text{smoke} \Rightarrow \text{cancer}) = \frac{67}{103} = 0.65 > \text{conf}(\text{not smoke} \Rightarrow \text{cancer}) = \frac{36}{103} = 0.35$

old $\text{conf}(\text{smoke} \Rightarrow \text{cancer}) = \frac{1}{3} = 0.333 < \text{conf}(\text{not smoke} \Rightarrow \text{cancer}) = \frac{34}{100} = 0.34$

young $\text{conf}(\text{smoke} \Rightarrow \text{cancer}) = \frac{66}{100} = 0.66 < \text{conf}(\text{not smoke} \Rightarrow \text{cancer}) = \frac{2}{3} = 0.666$

Simpson's Paradox - Example

Two classes: **old** and **young**

$\text{smoke} \Rightarrow \text{cancer}$	has cancer	doesn't have cancer	
smokes	$1 + 66$	$2 + 34$	$3 + 100$
doesn't smoke	$34 + 2$	$66 + 1$	$100 + 3$
	$35 + 68$	$68 + 35$	$103 + 103$

humans $\text{conf}(\text{smoke} \Rightarrow \text{cancer}) = \frac{67}{103} = 0.65 > \text{conf}(\text{not smoke} \Rightarrow \text{cancer}) = \frac{36}{103} = 0.35$

old $\text{conf}(\text{smoke} \Rightarrow \text{cancer}) = \frac{1}{3} = 0.333 < \text{conf}(\text{not smoke} \Rightarrow \text{cancer}) = \frac{34}{100} = 0.34$

young $\text{conf}(\text{smoke} \Rightarrow \text{cancer}) = \frac{66}{100} = 0.66 < \text{conf}(\text{not smoke} \Rightarrow \text{cancer}) = \frac{2}{3} = 0.666$

Smoking is healthy for **old** and **young** people, but not for all humans!



Simpson's Paradox - Example

Two classes: old and young

smoke \Rightarrow cancer	has cancer	doesn't have cancer	
smokes	$1 + 66$	$2 + 34$	$3 + 100$
doesn't smoke	$34 + 2$	$66 + 1$	$100 + 3$
	$35 + 68$	$68 + 35$	$103 + 103$

- The presence of smoking has a strong positive effect on the occurrence of cancer in the overall set (supports the rule)
- However, the effect cannot be seen in the subsets!

Simpson's Paradox - Example

Two classes: old and young

smoke \Rightarrow cancer	has cancer	doesn't have cancer	
smokes	$1 + 66$	$2 + 34$	$3 + 100$
doesn't smoke	$34 + 2$	$66 + 1$	$100 + 3$
	$35 + 68$	$68 + 35$	$103 + 103$

humans $\text{lift}(\text{smoke} \Rightarrow \text{cancer}) = \frac{\frac{67}{206}}{\frac{103}{206} \cdot \frac{103}{206}} = \frac{67 \cdot 206}{103 \cdot 103} = 1.301$

old $\text{lift}(\text{smoke} \Rightarrow \text{cancer}) = \frac{\frac{1}{103}}{\frac{3}{103} \cdot \frac{35}{103}} = \frac{1 \cdot 103}{3 \cdot 35} = 0.9809$

young $\text{lift}(\text{smoke} \Rightarrow \text{cancer}) = \frac{\frac{66}{103}}{\frac{100}{103} \cdot \frac{68}{103}} = \frac{66 \cdot 103}{100 \cdot 68} = 0.9997$

Simpson's Paradox - Example

Two classes: old and young

smoke \Rightarrow cancer	has cancer	doesn't have cancer	
smokes	$1 + 66$	$2 + 34$	$3 + 100$
doesn't smoke	$34 + 2$	$66 + 1$	$100 + 3$
	$35 + 68$	$68 + 35$	$103 + 103$

humans $\text{lift}(\text{smoke} \Rightarrow \text{cancer}) = \frac{\frac{67}{206}}{\frac{103}{206} \cdot \frac{103}{206}} = \frac{67 \cdot 206}{103 \cdot 103} = 1.301$

Positively correlated

old $\text{lift}(\text{smoke} \Rightarrow \text{cancer}) = \frac{\frac{1}{103}}{\frac{3}{103} \cdot \frac{35}{103}} = \frac{1 \cdot 103}{3 \cdot 35} = 0.9809$

Negatively correlated

young $\text{lift}(\text{smoke} \Rightarrow \text{cancer}) = \frac{\frac{66}{103}}{\frac{100}{103} \cdot \frac{68}{103}} = \frac{66 \cdot 103}{100 \cdot 68} = 0.9997$

Simpson's Paradox – Another Example

	Computer Science		Mathematics		ALL	
	get degree	drop out	get degree	drop out	get degree	drop out
female	80 (80%)	20 (20%)	400 (40%)	600 (60%)	480 (44%)	620 (56%)
male	700 (70%)	300 (30%)	30 (30%)	70 (70%)	730 (66%)	370 (34%)

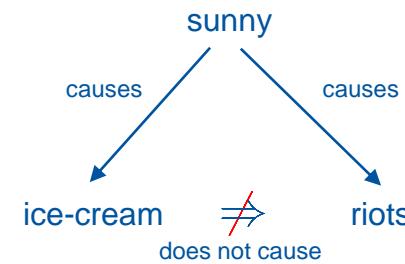
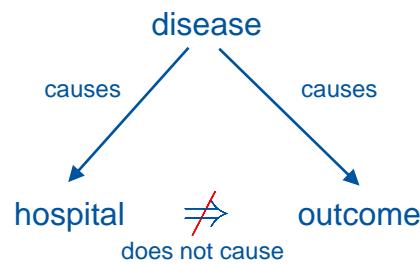
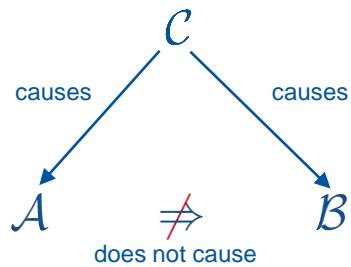
1100 females and 1100 males, 1100 CS students and 1100 math students

Simpson's Paradox – Other Examples

- The hospital in the city of Stolberg has an overall better performance (e.g., lower mortality rate) than the hospital in Aachen. However, for any specific disease, Aachen performs better. This paradox is due to different distributions of diseases (patients with more serious diseases tend to end up in Aachen and not Stolberg).
- Males have higher wages on average, but in any given profession, females earn more on average. This paradox is explained by males going for higher-paid professions.
- Low birth-weight paradox: low birth-weight children born to smoking mothers have a lower infant mortality rate than low-birth-weight children of non-smokers. Smoking is harmful and contributes to low birth weight and higher mortality than normal birth weight. However, other causes of low birth weight are generally more harmful than smoking.

Confounding

- Simpsons paradox is related to confounding, i.e., another (possibly hidden) feature that influences two other features
- A confounding feature C (also called “lurking variable”) may influence both A and B , and therefore “blur” $A \Rightarrow B$

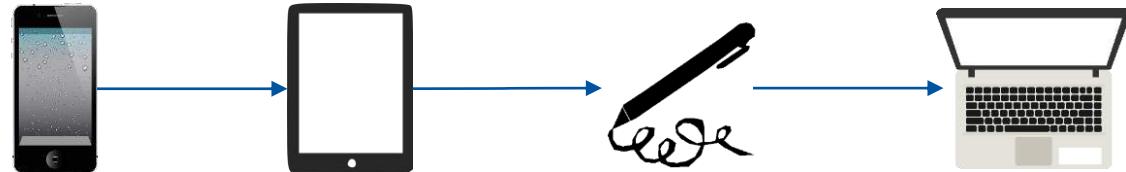


Summary

- Association rules can be discovered starting from frequent items sets $\mathcal{A} \Rightarrow \mathcal{B}$
- Any dataset with instances and feature values can be turned into a multiset of itemsets and used for association rule mining (not just “pure itemsets”)
- Support, confidence, and lift can be used to prune and sort association rules
- Rules should be interpreted carefully (Simpson’s paradox and confounders)

Sequence Mining

1. Temporal Data
2. Measuring Support
3. Apriori-All Algorithm
4. Extensions and Conclusion



Temporal Data – Discrete Timestamped Events

Time-stamp	f_1	f_2	f_3	f_4	...	f_D
t_1						
t_2						
t_3						
t_4						
t_5						
...						

Every instance happened at a specific time



Temporal Data – Discrete Timestamped Events

Event data

Time-stamp	Case ID	Activity	f_1	f_2	...	f_D
t_1	3	a				
t_2	1	a				
t_3	1	b				
t_4	2	a				
t_5	3	b				
...				

Case ID is used to group events

Activity identifies the type of event

Temporal Data – Discrete Timestamped Events

Event data

Time-stamp	Case ID	Activity	f_1	f_2	...	f_D
t_1	3	a				
t_2	1	a				
t_3	1	b				
t_4	2	a				
t_5	3	b				
...				

Case ID is used to group events

Activity identifies the type of event

Event Data

- **Timestamp** (typically **not** equal intervals)
- **Case ID** (maps events to cases)
- **Activity** (identifies the event type)
- Other features are optional (resource, location, cost, duration, ...)

Temporal Data – Discrete Timestamped Events

Event data

Time-stamp	Case ID	Activity	f_1	f_2	...	f_D
t_1	3	a				
t_2	1	a				
t_3	1	b				
t_4	2	a				
t_5	3	b				
...				

Case ID is used to group events

Activity identifies the type of event

Event Data

- **Timestamp** (typically **not** equal intervals)
- **Case ID** (maps events to cases)
- **Activity** (identifies the event type)
- Other features are optional (resource, location, cost, duration, ...)

Case 1: $\langle a, b, \dots \rangle$

Case 2: $\langle a, \dots \rangle$

Case 3: $\langle a, b, \dots \rangle$

Temporal Data – Discrete Timestamped Events

Event data

Time-stamp	Case ID	Activity	f_1	f_2	...	f_D
t_1	3	a				
t_2	1	a				
t_3	1	b				
t_4	2	a				
t_5	3	b				
...				

Case ID is used to group events

Activity identifies the type of event

Event Data

- Timestamp (typically **not** equal intervals)
- Case ID (maps events to cases)
- Activity (identifies the event type)
- Other features are optional (resource, location, cost, duration, ...)

Case 1: $\langle a, b, \dots \rangle$

Case 2: $\langle a, \dots \rangle$

Case 3: $\langle a, b, \dots \rangle$

We can **abstract** from timestamps and optional features to obtain **sequences of activities**

Temporal Data – Discrete Timestamped Events

Event data

Time-stamp	Case ID	Activity	f_1	f_2	...	f_D
t_1	3	a				
t_2	1	a				
t_3	1	b				
t_4	2	a				
t_5	3	b				
...				

Case ID is used to group events

Activity identifies the type of event

Event Data

- Timestamp (typically **not** equal intervals)
- Case ID (maps events to cases)
- Activity (identifies the event type)
- Other features are optional (resource, location, cost, duration, ...)

Case 1: $\langle a, b, \dots \rangle$

Case 2: $\langle a, \dots \rangle$

Case 3: $\langle a, b, \dots \rangle$

$\rightarrow [\langle a, b, \dots \rangle^2,$
 $\langle a, \dots \rangle]$

Event Data – Example 1

Case ID	Activity name	Timestamp	Other features	
Patient ID	Activity	Time	Doctor	Age
5611	Blood Test	12:25	Dr. Scott	45
3645	X-Ray	14:34	Dr. House	67
5611	Surgery	15:01	Dr. Scott	45
7891	Blood Test	15:03	Dr. House	24
3645	Radiation Therapy	17:25	Dr. Jenna	81
...

5611 : ⟨Blood Test, Surgery, ...⟩

3645 : ⟨X-Ray, Radiation Therapy, ...⟩

7891 : ⟨Blood Test, ...⟩

Event Data – Example 2

Case ID	Activity name	Timestamp	Other features		
Order Number	Activity	Time	Username	Product	Quantity
11152	Register Order	15.12.22 12:25	Carrie192	Iphone 14	1
52690	Ship Order	15.12.22 12:45	Johnny1	Earpods	2
11152	Check Stock	15.12.22 13:01	Carrie192	Iphone 14	1
44891	Handle Payment	30.12.22 18:01	Obelisk	USB-C Charger	3
61238	Cancel Order	11.01.23 17:25	Apex_512	MacBook Air	1
...

11152 : <Register Order, Check Stock, Cancel Order, ...>

52690 : <Ship Order, ...>

44891 : <Handle Payment, ...>

Note: 'Username' could also be our Case ID, changing the meaning of data!

Event Data – Example 2

Case ID	Activity name	Timestamp	Other features		
Order Number	Activity	Time	Username	Product	Quantity
11152	Register Order	15.12.22 12:25	Carrie192	Iphone 14	1
52690	Ship Order	15.12.22 12:45	Johnny1	Earpods	2
11152	Check Stock	15.12.22 13:01	Carrie192	Iphone 14	1
44891	Handle Payment	30.12.22 18:01	Obelisk	USB-C Charger	3
61238	Cancel Order	11.01.23 17:25	Apex_512	MacBook Air	1
...

11152 : <Register Order, Check Stock, Cancel Order, ... >

52690 : <Ship Order, ... >

88721 : <Register Order, Check Stock, Cancel Order, ... >

Note: the same sequence can occur
multiple times for different cases
(multiset of sequences)

Event data – Basis for Process Mining

Event data

Time-stamp	Case ID	Activity	f_1	f_2	...	f_D
t_1	3	a				
t_2	1	a				
t_3	1	b				
t_4	2	a				
t_5	3	b				
...				

Case ID is used to group **events**

Activity identifies the type of event

Process Mining

- Processes generate **event data**
- Every process execution is a **case**

Common Tasks

- Discover the process
- Validate the process
- Improve the process

Temporal Data – Discrete Timestamped Events

Generalized sequential data

Time-stamp	Case ID	Item
t_1	3	a
t_2	1	a
t_3	1	b
t_4	2	a
t_5	3	b
...

Item Identifier

Sequential Data

- **Timestamp** (typically **not** equal intervals)
- **Case ID** (maps events to cases)
- **Item** (identifies the item type)

Relation to **event data**:
item could be an **activity**

Temporal Data – Discrete Timestamped Events

Generalized sequential data

Timestamp	Customer ID	Purchased Item
22-07-12	1172	Razor
22-07-12	8121	Shampoo
22-07-12	1172	Shaving Cream
22-08-13	3434	Shampoo
22-09-01	1172	Shaving Cream
...

1172 :⟨Razor, Shaving Cream, Shaving Cream⟩

8121 :⟨Shampoo⟩

3434 :⟨Shampoo⟩

...

⇒ [⟨Razor, Shaving Cream, Shaving Cream⟩,
⟨Shampoo⟩², ...]

Temporal Data – Discrete Timestamped Events

Generalized sequential data

Timestamp	Customer ID	Purchased Item
22-07-12	1172	Razor
22-07-12	8121	Shampoo
22-07-12	1172	Shaving Cream
22-08-13	3434	Shampoo
22-09-01	1172	Shaving Cream
...

1172 :⟨Razor, Shaving Cream, Shaving Cream⟩

8121 :⟨Shampoo⟩

3434 :⟨Shampoo⟩

...

⇒ [⟨Razor, Shaving Cream, Shaving Cream⟩,
⟨Shampoo⟩², ...]



Timestamp	Customer ID	Purchased Itemset
22-07-12	1172	Razor, Shaving Cream
22-07-12	8121	Shampoo
22-08-13	3434	Shampoo
22-09-01	1172	Shaving Cream
...

1172 :⟨{Razor, Shaving Cream}, {Shaving Cream}⟩

8121 :⟨{Shampoo}⟩

3434 :⟨{Shampoo}⟩

...

⇒ [⟨{Razor, Shaving Cream}, {Shaving Cream}⟩,
⟨{Shampoo}⟩², ...]

Temporal Data – Discrete Timestamped Events

Generalized sequential data

Sequential Pattern Mining

- Input: a multiset of nonempty sequences of itemsets
- Main analysis question: identify frequent subsequences (recurring patterns)
- Relation to event data:
an itemset can be interpreted as activity,
an activity can be an itemset of size 1

Timestamp	Customer ID	Purchased Itemset
22-07-12	1172	Razor, Shaving Cream
22-07-12	8121	Shampoo
22-08-13	3434	Soap
22-09-01	1172	Shaving Cream
...

1172 :⟨{Razor, Shaving Cream}, {Shaving Cream}⟩

8121 :⟨{Shampoo}⟩

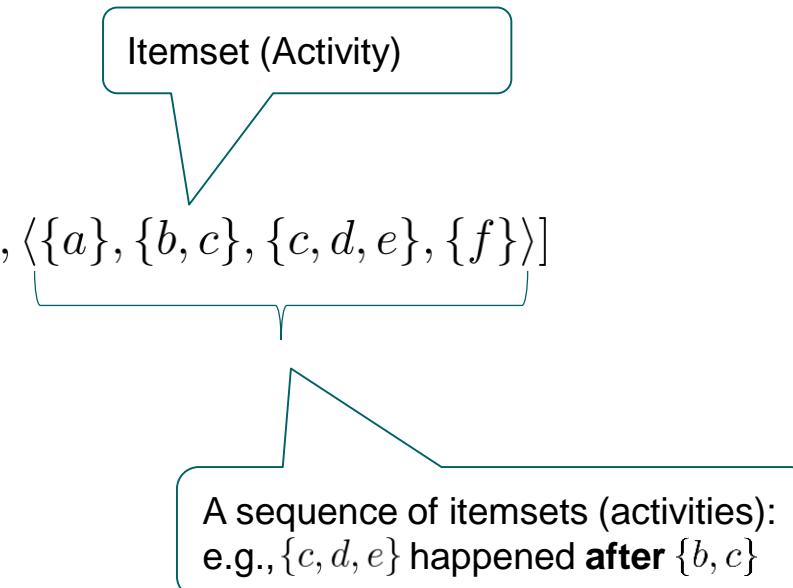
3434 :⟨{Shampoo}⟩

...

⇒ [⟨{Razor, Shaving Cream}, {Shaving Cream}⟩,
⟨{Shampoo}⟩², ...]

Sequential Pattern Mining

- Uses a specific type of (event) data as input: multiset of sequences of itemsets
- A sequence is a nonempty sequence of itemsets
- Two notations for sequence data:
 - Formal:
 $\mathcal{X} = [\langle \{a\}, \{b\}, \{c, d\}, \{e\} \rangle, \langle \{a\}, \{b\}, \{c, d\}, \{e\} \rangle, \langle \{a\}, \{b, c\}, \{c, d, e\}, \{f\} \rangle]$
 - Informal (short notation):
 $\mathcal{X} = [ab(cd)e, ab(cd)e, a(cd)e, a(bc)(cde)f]$
- Formally $\mathcal{X} \in \mathbb{M}((\mathbb{P}(\mathcal{I}))^*)$ for a set of items \mathcal{I}
(\mathbb{M} is the multiset and \mathbb{P} the powerset operator)



Sequential Pattern Mining – Input Example

Customer ID	Purchased Items	Time
1	A	15.12.22 12:25
1	A, B	15.12.22 12:45
2	B	15.12.22 13:01
3	C	30.12.22 18:01
3	A, C, D	11.01.23 17:25
4	B	31.12.22 17:32
...



Customer ID	Customer Sequence
1	$\langle \{A\}, \{A, B\} \rangle$
2	$\langle \{B\} \rangle$
3	$\langle \{C\}, \{A, C, D\} \rangle$
4	$\langle \{B\} \rangle$
...	...

Input is a **multiset of sequences of itemsets**

Sequential Pattern Mining – Input

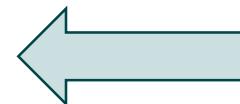
Input $\mathcal{X} \in \mathbb{M}((\mathbb{P}(\mathcal{I}))^*)$

- Formal:

$$\begin{aligned} & [\langle \{A\}, \{A, B\} \rangle, \langle \{B\} \rangle, \langle \{C\}, \{A, C, D\} \rangle, \langle \{B\} \rangle] \\ & = [\langle \{A\}, \{A, B\} \rangle, \langle \{B\} \rangle^2, \langle \{C\}, \{A, C, D\} \rangle] \end{aligned}$$

- Informal:

$$\begin{aligned} & [A(AB), B, C(ACD), B] \\ & = [A(AB), B^2, C(ACD)] \end{aligned}$$



Customer ID	Customer Sequence
1	$\langle \{A\}, \{A, B\} \rangle$
2	$\langle \{B\} \rangle$
3	$\langle \{C\}, \{A, C, D\} \rangle$
4	$\langle \{B\} \rangle$
...	...

Input is a **multiset of sequences of itemsets**



MEIN KONTO

Willkommen Wil van der Aalst

Mitglied seit 21-09-2018

Kundennummer: 3868857



Meine Bestellungen



Meine Adressen



Meine persönlichen Daten



Meine Maschinen

Benachrichtigungen &
Erinnerungen

Marketingpräferenzen



Express Checkout



Mein Kaffee Abo

Meine Bestellungen

BESTELLDATUM	STATUS	QUELLE	Liefermethode	BESTELLNUMMER	BETRAG	
01/11/2018	Geliefert	Internet	Standardlieferung - Lieferung am nächsten Werktag	25250378	97,80 €	
21/09/2018	Geliefert	Internet	Standardlieferung innerhalb von 2 Werktagen	24533528	78,60 €	

[Mehr Bestellungen anzeigen >](#)

Bestellung - 01/11/2018

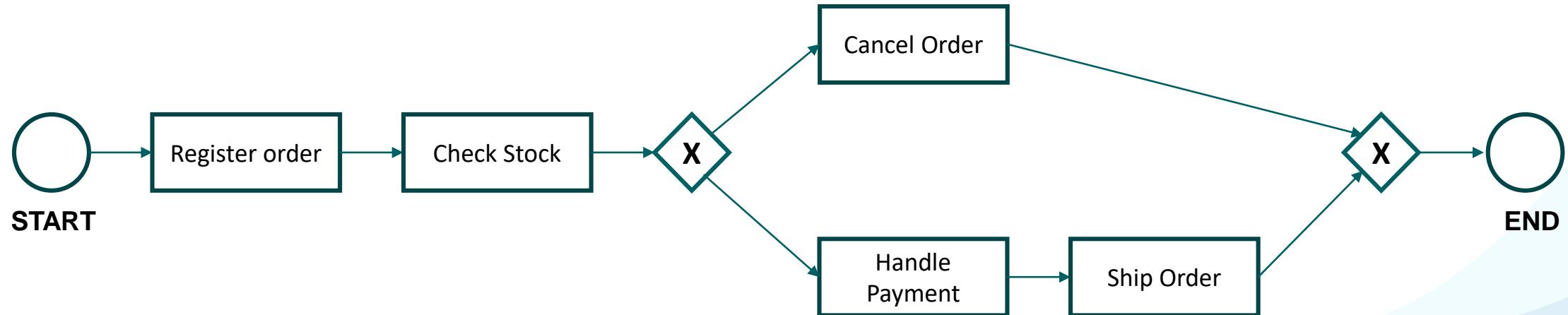
[Wieder bestellen](#)

Kapseln (250)	Stückpreis	Menge	Gesamt
Ristretto	0,38 €	x	30
Roma	0,38 €	x	80
Vivalto Lungo	0,40 €	x	50
Linizio Lungo	0,40 €	x	80
Ristretto Decaffeinato	0,40 €	x	10

 $\mathcal{X} \in \mathbb{M}((\mathbb{P}(\mathcal{I}))^*)$

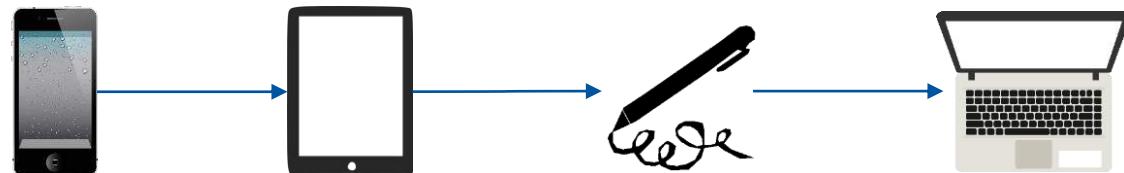
Temporal Data – Analysis Techniques

- This lecture – Sequential Pattern Mining
- Next lectures – Time Series and Process Mining:
 - Analyze and predict time series data
 - Discover, validate and improve processes



Sequence Mining

1. Temporal Data
2. **Measuring Support**
3. Apriori-All Algorithm
4. Extensions and Conclusion



Goal – Find Frequent Sequential Patterns

Customer ID	Customer Sequence
1	$\langle \{11\}, \{25\} \rangle$
2	$\langle \{31\} \rangle$
3	$\langle \{12\}, \{11\} \rangle$
...	...



Sequential Patterns with Support > Threshold (Min_Sup)
$\langle \{31\} \rangle$
$\langle \{12\}, \{11\} \rangle$
...

- Given a dataset $\mathcal{X} \in \mathbb{M}((\mathbb{P}(\mathcal{I}))^*)$ find all frequent sequential patterns
- Sequential pattern \mathcal{P} is a sequence of itemsets, i.e., $\mathcal{P} \in (\mathbb{P}(\mathcal{I}))^*$
- Support of a sequential pattern is the fraction of sequences in \mathcal{X} that contain the pattern \mathcal{P}

Containment

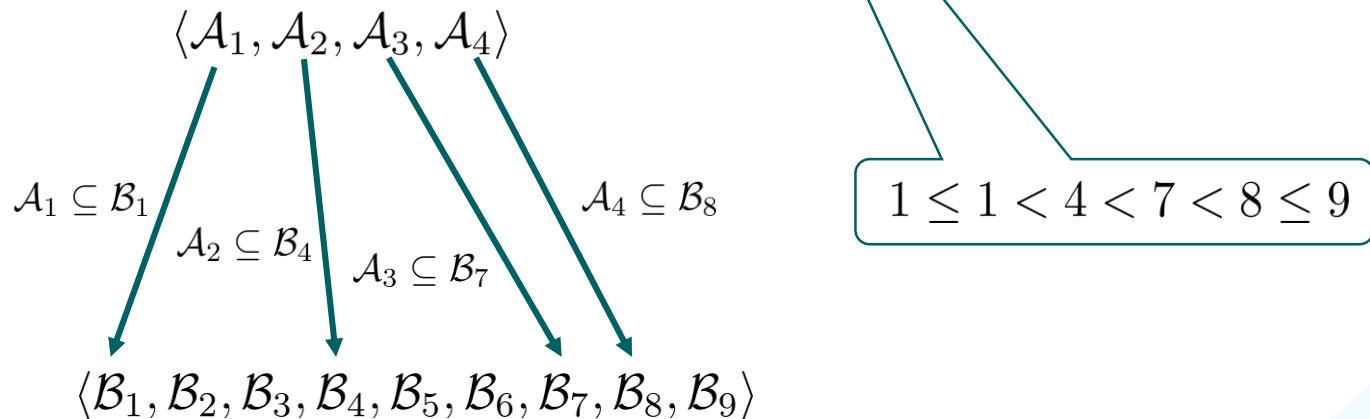
- Let $\mathcal{A} = \langle \mathcal{A}_1, \mathcal{A}_2, \dots, \mathcal{A}_n \rangle \in (\mathbb{P}(\mathcal{I}))^*$ and $\mathcal{B} = \langle \mathcal{B}_1, \mathcal{B}_2, \dots, \mathcal{B}_m \rangle \in (\mathbb{P}(\mathcal{I}))^*$ be two itemset sequences
- \mathcal{A} is contained in \mathcal{B} if there exist integers $1 \leq i_1 < i_2 < \dots < i_n \leq m$ such that

$$\mathcal{A}_1 \subseteq \mathcal{B}_{i_1}, \mathcal{A}_2 \subseteq \mathcal{B}_{i_2}, \dots, \mathcal{A}_n \subseteq \mathcal{B}_{i_n}$$

Containment

- Let $\mathcal{A} = \langle \mathcal{A}_1, \mathcal{A}_2, \dots, \mathcal{A}_n \rangle \in (\mathbb{P}(\mathcal{I}))^*$ and $\mathcal{B} = \langle \mathcal{B}_1, \mathcal{B}_2, \dots, \mathcal{B}_m \rangle \in (\mathbb{P}(\mathcal{I}))^*$ be two itemset sequences
- \mathcal{A} is contained in \mathcal{B} if there exist integers $1 \leq i_1 < i_2 < \dots < i_n \leq m$ such that

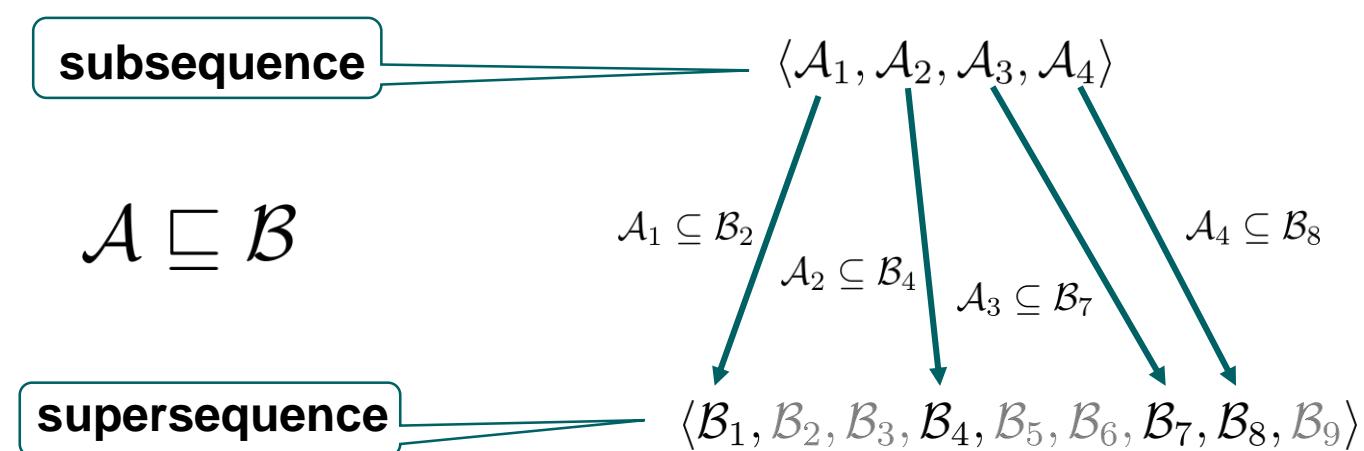
$$\mathcal{A}_1 \subseteq \mathcal{B}_{i_1}, \mathcal{A}_2 \subseteq \mathcal{B}_{i_2}, \dots, \mathcal{A}_n \subseteq \mathcal{B}_{i_n}$$



Containment

- Notation: $\mathcal{A} \sqsubseteq \mathcal{B}$ if \mathcal{A} is contained in \mathcal{B}
- If $\mathcal{A} \sqsubseteq \mathcal{B}$, then \mathcal{A} is a **subsequence** of \mathcal{B} and \mathcal{B} is a **supersequence** of \mathcal{A}

$$\mathcal{A}_1 \subseteq \mathcal{B}_{i_1}, \mathcal{A}_2 \subseteq \mathcal{B}_{i_2}, \dots, \mathcal{A}_n \subseteq \mathcal{B}_{i_n}$$



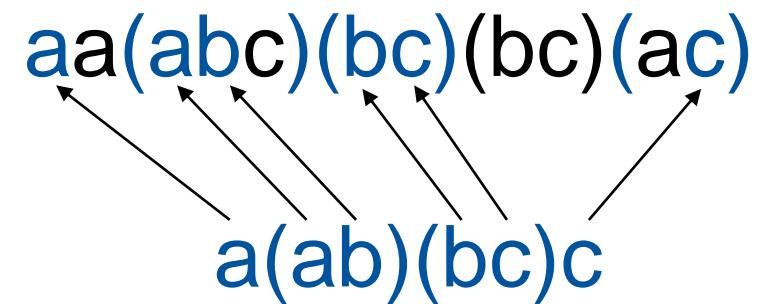
Containment - Examples

Formal notation:

- $\langle \{a\}, \{a, b\}, \{b, c\}, \{c\} \rangle \sqsubseteq \langle \{a\}, \{a\}, \{a, b, c\}, \{b, c\}, \{b, c\}, \{a, c\} \rangle$

Informal notation:

- $a(ab)(bc)c \sqsubseteq aa(abc)(bc)(bc)(ac)$



multiple mappings possible

$\langle a_1, a_2, \dots, a_n \rangle \sqsubseteq \langle b_1, b_2, \dots, b_m \rangle$ if and only if there exist integers $1 \leq i_1 < i_2 < \dots < i_n \leq m$ such that $a_1 \subseteq b_{i_1}, a_2 \subseteq b_{i_2}, \dots, a_n \subseteq b_{i_n}$

Containment - Examples

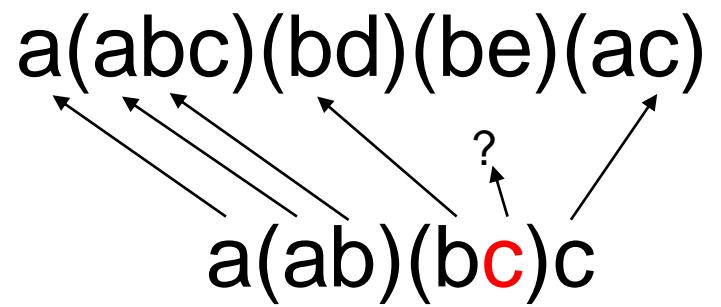
Formal notation:

- $\langle \{a\}, \{a, b\}, \{b, c\}, \{c\} \rangle \sqsubseteq \langle \{a\}, \{a\}, \{a, b, c\}, \{b, c\}, \{b, c\}, \{a, c\} \rangle$
- $\langle \{a\}, \{a, b\}, \{b, c\}, \{c\} \rangle \not\sqsubseteq \langle \{a\}, \{a, b, c\}, \{b, d\}, \{b, e\}, \{a, c\} \rangle$

Informal notation:

- $a(ab)(bc)c \sqsubseteq aa(abc)(bc)(bc)(ac)$
- $a(ab)(bc)c \not\sqsubseteq a(abc)(bd)(be)(ac)$

$\langle a_1, a_2, \dots, a_n \rangle \sqsubseteq \langle b_1, b_2, \dots, b_m \rangle$ if and only if there exist integers $1 \leq i_1 < i_2 < \dots < i_n \leq m$ such that $a_1 \sqsubseteq b_{i_1}$, $a_2 \sqsubseteq b_{i_2}$, ..., $a_n \sqsubseteq b_{i_n}$



Containment – Practice Questions

- $(ab)(bc) \sqsubseteq (bc)(ab)$?
- $ab \sqsubseteq a(ac)(bc)c$?
- $aa(ab)(bc) \sqsubseteq (ab)(ace)(bce)(ab)$?
- $(abc)ef \sqsubseteq (ab)(bc)(ef)f$?
- $(abc)ef \sqsubseteq (ab)(bc)(abcd)(ef)f$?



$\langle a_1, a_2, \dots, a_n \rangle \sqsubseteq \langle b_1, b_2, \dots, b_m \rangle$ if and only if there exist integers $1 \leq i_1 < i_2 < \dots < i_n \leq m$ such that $a_1 \sqsubseteq b_{i_1}$, $a_2 \sqsubseteq b_{i_2}$, ..., $a_n \sqsubseteq b_{i_n}$

Containment – Practice Answers

- $(ab)(bc) \not\sqsubseteq (bc)(ab)$ (incompatible order)
- $ab \sqsubseteq a(ac)(bc)c$
- $aa(ab)(bc) \not\sqsubseteq (ab)(ace)(bce)(ab)$ ((ab) cannot be mapped without also handling aa or (bc), etc.)
- $(abc)ef \not\sqsubseteq (ab)(bc)(ef)f$ (no match for (abc))
- $(abc)ef \sqsubseteq (ab)(bc)(abcd)(ef)f$

$\langle a_1, a_2, \dots, a_n \rangle \sqsubseteq \langle b_1, b_2, \dots, b_m \rangle$ if and only if there exist integers $1 \leq i_1 < i_2 < \dots < i_n \leq m$ such that $a_1 \sqsubseteq b_{i_1}$, $a_2 \sqsubseteq b_{i_2}$, ..., $a_n \sqsubseteq b_{i_n}$

Support

- The support of a sequential pattern \mathcal{P} is the fraction of sequences in \mathcal{X} that contain \mathcal{P}
- $\text{support}(\mathcal{P}) = \frac{|[\mathcal{S} \in \mathcal{X} | \mathcal{P} \sqsubseteq \mathcal{S}]|}{|\mathcal{X}|}$
- Minimum support threshold min_sup defines which sequences are frequent

Support

- The support of a sequential pattern \mathcal{P} is the fraction of sequences in \mathcal{X} that contain \mathcal{P}
- $\text{support}(\mathcal{P}) = \frac{|[\mathcal{S} \in \mathcal{X} | \mathcal{P} \sqsubseteq \mathcal{S}]|}{|\mathcal{X}|}$
- Minimum support threshold ***min_sup*** defines which sequences are frequent
- Support count is the number of sequences in \mathcal{X} that contain \mathcal{P}
- $\text{support_count}(\mathcal{P}) = |[\mathcal{S} \in \mathcal{X} | \mathcal{P} \sqsubseteq \mathcal{S}]|$

Support – Practice Questions

- $\mathcal{X} = [abcd^2, (abcd), (ab)(cd)^3, (ab)(bc)(cd)]$
- What is the support_count(\mathcal{P}) for
 - $\mathcal{P} = a$
 - $\mathcal{P} = ab$
 - $\mathcal{P} = (ab)$
 - $\mathcal{P} = (ab)c$
 - $\mathcal{P} = (ab)(bd)$
 - $\mathcal{P} = ab(cd)$



$$= [\langle \{a\}, \{b\}, \{c\}, \{d\} \rangle^2, \\ \langle \{a, b, c, d\} \rangle, \\ \langle \{a, b\} \{c, d\} \rangle^3, \\ \langle \{a, b\}, \{b, c\}, \{c, d\} \rangle]$$

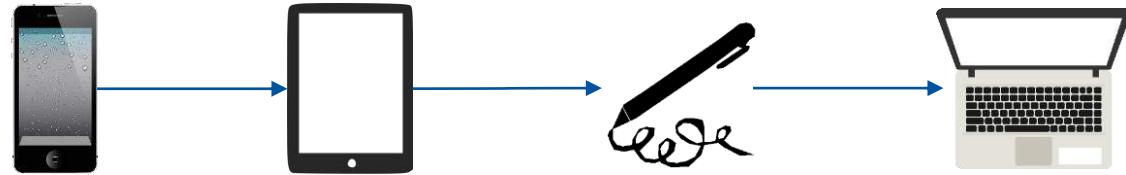
$$= [\langle \{a\}, \{b\}, \{c\}, \{d\} \rangle, \\ \langle \{a, b, c, d\} \rangle, \\ \langle \{a, b\} \{c, d\} \rangle, \\ \langle \{a, b\}, \{b, c\}, \{c, d\} \rangle, \\ \langle \{a\}, \{b\}, \{c\}, \{d\} \rangle, \\ \langle \{a, b\} \{c, d\} \rangle, \\ \langle \{a, b\} \{c, d\} \rangle]$$

Support – Practice Questions

- $\mathcal{X} = [abcd^2, (abcd), (ab)(cd)^3, (ab)(bc)(cd)]$
- What is the support_count(\mathcal{P}) for
 - $\mathcal{P} = a$ **7** : [abcd², (abcd), (ab)(cd)³, (ab)(bc)(cd)]
 - $\mathcal{P} = ab$ **3** : [abcd², (abcd), (ab)(cd)³, (ab)(bc)(cd)]
 - $\mathcal{P} = (ab)$ **5** : [abcd², (abcd), (ab)(cd)³, (ab)(bc)(cd)]
 - $\mathcal{P} = (ab)c$ **4** : [abcd², (abcd), (ab)(cd)³, (ab)(bc)(cd)]
 - $\mathcal{P} = (ab)(bd)$ **0** : [abcd², (abcd), (ab)(cd)³, (ab)(bc)(cd)]
 - $\mathcal{P} = ab(cd)$ **1** : [abcd², (abcd), (ab)(cd)³, (ab)(bc)(cd)]

Sequence Mining

1. Temporal Data
2. Measuring Support
3. **Apriori-All Algorithm**
4. Extensions and Conclusion



Brute Force Approach

Goal: find all frequent sequential patterns

- Let k be the length of the longest sequence in \mathcal{X} and q the size of the largest itemset
- Generate all sequential patterns of length $\leq k$ with itemsets of size $\leq q$ (this number is finite)
- Compute the support of each candidate pattern
- Return all that have a support higher than min_sup
- Obviously, this is very expensive!

all sequential patterns
of length k

\mathcal{L}_k

Smarter Approach Based on Apriori

- First described in Rakesh Agrawal, Ramakrishnan Srikant: Mining Sequential Patterns
- Similar to Apriori for frequent itemsets – avoid testing hopeless candidates
- If $\mathcal{A} \sqsubseteq \mathcal{B}$ (\mathcal{A} is contained in \mathcal{B}), then \mathcal{B} cannot be frequent if \mathcal{A} is not frequent
 - $\text{support}(\mathcal{A}) \geq \text{support}(\mathcal{B})$ if $\mathcal{A} \sqsubseteq \mathcal{B}$
 - if $\mathcal{A} \sqsubseteq \mathcal{B}$ and $\text{support}(\mathcal{A}) < \text{min_sup}$ then $\text{support}(\mathcal{B}) < \text{min_sup}$

Step 1 – Determine All Litemsets

- $\mathcal{L} = \{\mathcal{A} \subseteq \mathcal{I} \mid \text{support}(\langle \mathcal{A} \rangle) \geq \text{min_sup}\}$ are all itemsets that appear in a sufficient number of sequences
- These itemsets are called **litemsets** (\mathcal{L} is the set of all litemsets)

Step 1 – Determine All Litemsets

- $\mathcal{L} = \{\mathcal{A} \subseteq \mathcal{I} \mid \text{support}(\langle \mathcal{A} \rangle) \geq \text{min_sup}\}$ are all itemsets that appear in a sufficient number of sequences
- These itemsets are called **litemsets** (\mathcal{L} is the set of all litemsets)
- Consider $\mathcal{X} = [\text{abcd}, (\text{abcd}), (\text{ab})(\text{cd}), (\text{ab})(\text{bc})(\text{cd})]$ and **min_sup** = 0.7.
The following itemsets are frequent:
a (support = 4/4), b (support = 4/4), c (support = 4/4), d (support = 4/4), (ab) (support = 3/4), (cd) (support = 3/4)
- To **determine all litemsets**, we can use a variant of the original Apriori algorithm
(the only difference is that support is now counted per sequence of transactions and not per transaction)

Step 2 – Transform the Dataset

- We only need to consider the itemsets \mathcal{L}
 - There cannot be any frequent patterns that involve other itemsets
 - Frequent sequence patterns must be of the form \mathcal{L}^* !
- The set $\mathcal{L}_1 = \{\langle \mathcal{I} \rangle \mid \mathcal{I} \in \mathcal{L}\}$ is the set of all frequent sequence patterns of length 1
- $\mathcal{L}_k \subseteq \mathcal{L}^*$ is the set of all frequent sequence patterns of length exactly k
(to be ‘grown’ from shorter sequence patterns)

Step 2 – Transform the Dataset

- Transform $\mathcal{X} \in \mathbb{M}((\mathbb{P}(\mathcal{I}))^*)$ into $\mathcal{X}_T \in \mathbb{M}((\mathbb{P}(\mathcal{L}))^*)$
→ itemsets are mapped onto all itemsets they contain
- Each sequence is now described by a sequence of sets of itemsets (**extra level**)

Step 2 – Transform the Dataset

- Transform $\mathcal{X} \in \mathbb{M}((\mathbb{P}(\mathcal{I}))^*)$ into $\mathcal{X}_T \in \mathbb{M}((\mathbb{P}(\mathcal{L}))^*)$
→ itemsets are mapped onto all itemsets they contain
- Each sequence is now described by a sequence of sets of itemsets (**extra level**)

Example 1: Consider $\mathcal{L} = \{\{a\}, \{b\}, \{c\}, \{a, b\}\}$

- $\langle \{a, c\}, \{a, b, c\} \rangle$ corresponds to $\langle \{\{a\}, \{c\}\}, \{\{a\}, \{b\}, \{c\}, \{a, b\}\} \rangle$
 - because $\{a, c\}$ has frequent subsets $\{a\}, \{c\}$,
 - and $\{a, b, c\}$ has frequent subsets $\{a\}, \{b\}, \{c\}$, and $\{a, b\}$
- $\langle \{c\}, \{a, c\} \rangle$ corresponds to $\langle \{\{c\}\}, \{\{a\}, \{c\}\} \rangle$

Step 2 – Transform the Dataset

- Transform $\mathcal{X} \in \mathbb{M}((\mathbb{P}(\mathcal{I}))^*)$ into $\mathcal{X}_T \in \mathbb{M}((\mathbb{P}(\mathcal{L}))^*)$
→ itemsets are mapped onto all itemsets they contain
- Each sequence is now described by a sequence of sets of itemsets (**extra level**)

Example 2: Consider $\mathcal{L} = \{\{a\}, \{b\}, \{c\}, \{a, b\}\}$ and $\mathcal{X} = [\langle \{\textcolor{orange}{a}, \textcolor{blue}{c}\}, \{\textcolor{blue}{a}, \textcolor{red}{b}, \textcolor{blue}{c}\} \rangle, \langle \{\textcolor{red}{c}\}, \{\textcolor{green}{a}, \textcolor{green}{c}\} \rangle, \dots]$

- Then $\mathcal{X}_T = [\langle \{\{a\}\{c\}\}, \{\{a\}, \{b\}, \{c\}, \{a, b\}\} \rangle, \langle \{\{c\}\}, \{\{a\}, \{c\}\} \rangle, \dots]$

Step 2 – Transform the Dataset

- Transform $\mathcal{X} \in \mathbb{M}((\mathbb{P}(\mathcal{I}))^*)$ into $\mathcal{X}_T \in \mathbb{M}((\mathbb{P}(\mathcal{L}))^*)$
→ itemsets are mapped onto all itemsets they contain
- Each sequence is now described by a sequence of sets of itemsets (**extra level**)

This preprocessing is not essential but makes sense because the dataset is traversed many times

Example 2: Consider $\mathcal{L} = \{\{a\}, \{b\}, \{c\}, \{a, b\}\}$ and $\mathcal{X} = [\langle \{a, c\}, \{a, b, c\} \rangle, \langle \{c\}, \{a, c\} \rangle, \dots]$

- Then $\mathcal{X}_T = [\langle \{\{a\}\{c\}\}, \{\{a\}, \{b\}, \{c\}, \{a, b\}\} \rangle, \langle \{\{c\}\}, \{\{a\}, \{c\}\} \rangle, \dots]$

Testing whether a sequence pattern is supported by a sequence in the dataset is easy now!

Step 3 – Generate a Set of Candidate Sequences

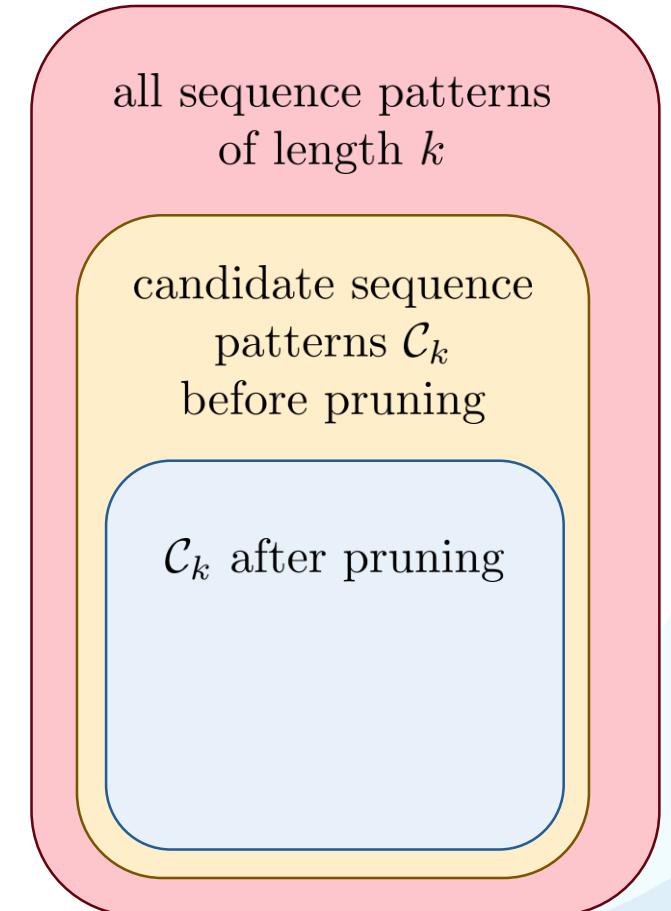
- Assume we have \mathcal{L}_{k-1} , the set of all frequent sequence patterns of length $k - 1$
(recall that $\mathcal{L}_1 = \{\langle \mathcal{I} \rangle \mid \mathcal{I} \in \mathcal{L}\}$)
- Create the set of candidate sequences \mathcal{C}_k by combining two sequences from \mathcal{L}_{k-1} where the first $k - 1$ itemsets are the same
(just like in Apriori for frequent itemsets)

all sequence patterns
of length k

candidate sequence
patterns \mathcal{C}_k
before pruning

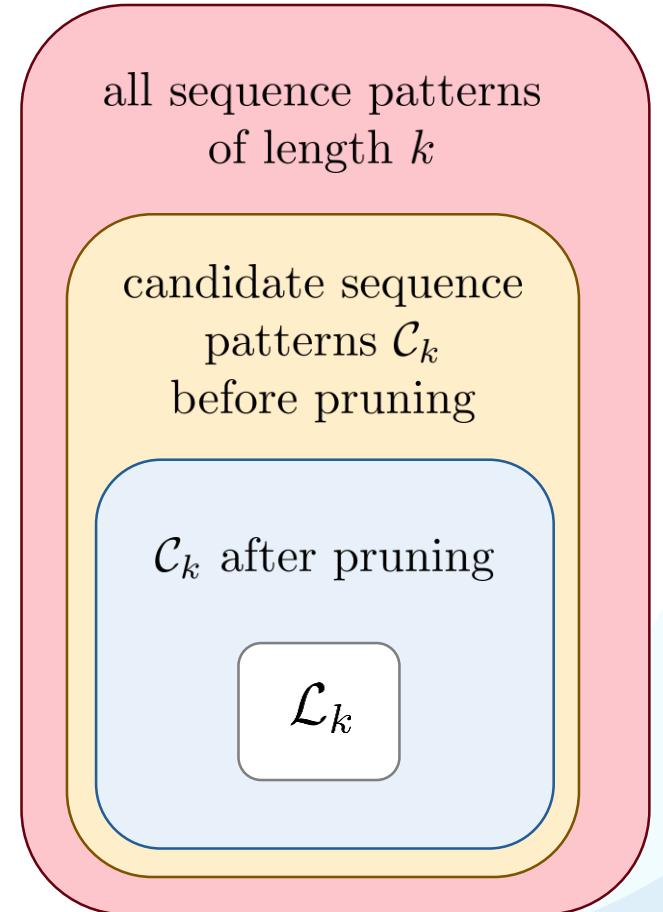
Step 4 – Prune the Set of Candidate Sequences

- For all candidate sequences $\mathcal{C} \in \mathcal{C}_k$
 - Consider all subsequences of \mathcal{C} of length $k - 1$
 - If one of these subsequences is not in \mathcal{L}_{k-1} , then remove \mathcal{C} from \mathcal{C}_k



Step 5 – Test All Candidate Sequences

- For each transformed sequence $S \in \mathcal{X}_T$: Increment the count of $\mathcal{C} \in \mathcal{C}_k$ if \mathcal{C} is contained in S
- Remove all candidates $\mathcal{C} \in \mathcal{C}_k$ that do not meet the threshold to obtain $\mathcal{L}_k = \{ \mathcal{C} \in \mathcal{C}_k \mid \text{support}(\mathcal{C}) \geq \text{min_sup} \}$
- Increment k and go to Step 3 (Candidate Generation) until $\mathcal{L}_k = \emptyset$
- $\bigcup_k \mathcal{L}_k$ is the set of all frequent sequence patterns



Step 6 (Optional) – Remove Non-Maximal Patterns

- A sequence S is a maximal frequent sequence in \mathcal{X}
 - if S is frequent,
 - and there is no real supersequence S' that is also frequent ($S \sqsubset S'$)



Step 6 (Optional) – Remove Non-Maximal Patterns

- A sequence S is a maximal frequent sequence in \mathcal{X}
 - if S is frequent,
 - and there is no real supersequence S' that is also frequent ($S \sqsubset S'$)
- It is possible to keep only the maximal sequences
- However, support information for the subsequences will be lost (subsequences may have higher supports)



Other Sequential Pattern Mining Approaches

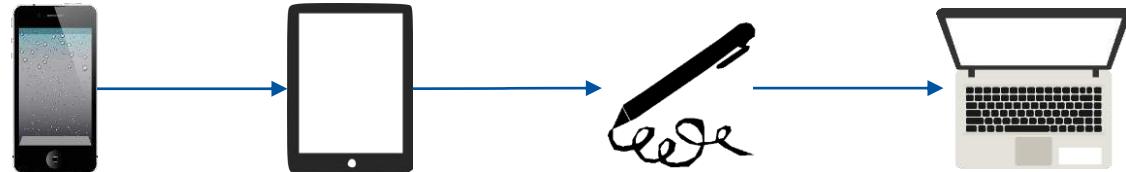
There are many other algorithms to find frequent sequential patterns:

- Maximal Frequent Sequences (MFS)
- Maximal Sequential Patterns using Sampling (MSPS)
- Indexed Bit Map (IBM)
- Sequential Pattern Mining with Length-decreasing Support (SLPMiner)
- WINEPI, MINEPI
- And many others...



Sequence Mining

1. Temporal Data
2. Measuring Support
3. Apriori-All Algorithm
4. **Extensions and Conclusion**



Association Rules Based on Frequent Sequences

- Frequent sequence patterns can be split in an ‘if’ and ‘then’ part (just like ‘normal’ association rules)
- $\langle \{beer\}, \{red, white\} \rangle \Rightarrow \langle \{beer\}, \{red, white\}, \{wodka\} \rangle$
- $\langle \{beer\}, \{beer\} \rangle \Rightarrow \langle \{beer\}, \{beer\}, \{beer\}, \{beer\}, \{beer\} \rangle$
- Many variants possible

How to Identify Interesting Frequent Sequences?

- For any technique that identifies patterns, it is important to filter out the less interesting ones
- One can look at things like correlations and base frequencies to decide how surprising sequences or sequence-based rules are (lift metric)
- It is also possible to add further constraints...

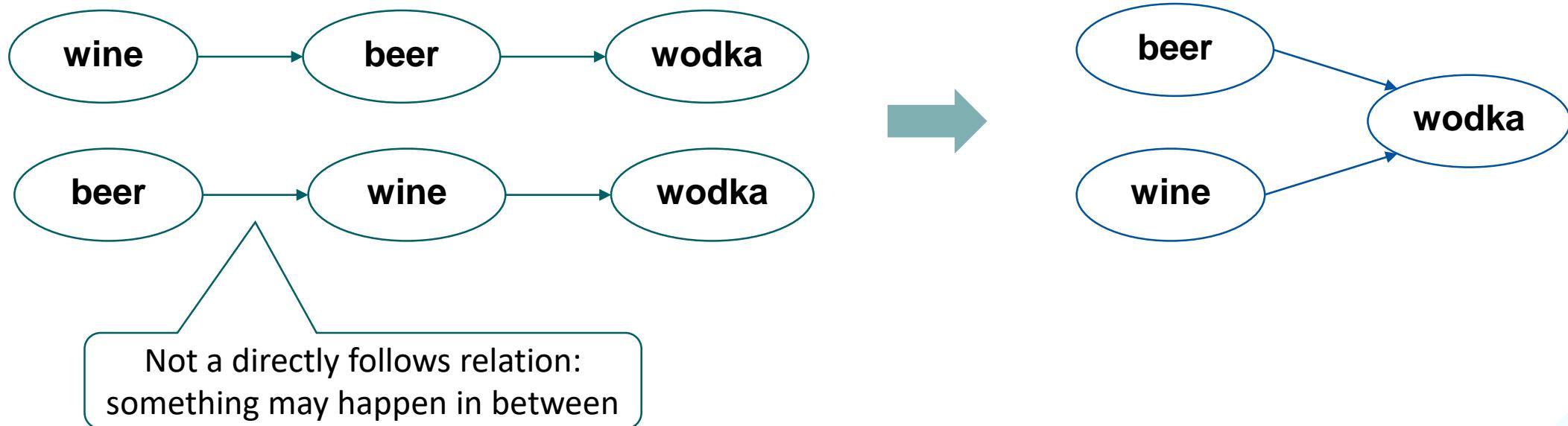
How to Identify Interesting Frequent Sequences?

Examples for further constraints:

- Item constraints: only consider sequences that include or exclude a set of items
- Length constraints: only consider patterns of a given size
- Time constraints: only consider patterns that occur in a short timeframe
(this includes gap and duration constraints)
- Regular expression constraints: only consider patterns that satisfy a regular expression or temporal constraint

Episode Mining

Extension: rather than looking for sequences we look for embedded **partial orders** (not subject of this course)



In future lectures – More Temporal Data!

Event data

Time-stamp	Case ID	Activity	f_1	f_2	...	f_D
t_1	3	a				
t_2	1	a				
t_3	1	b				
t_4	2	a				
t_5	3	b				
...				

Case ID is used to group **events**

Activity identifies the type of event

- We will see how to analyze **time series**
- **Process mining**: the analysis of event data as interplay between **events and models**

Elements of Machine Learning & Data Science

Time Series, Data Quality, and Preprocessing

Lecture 11

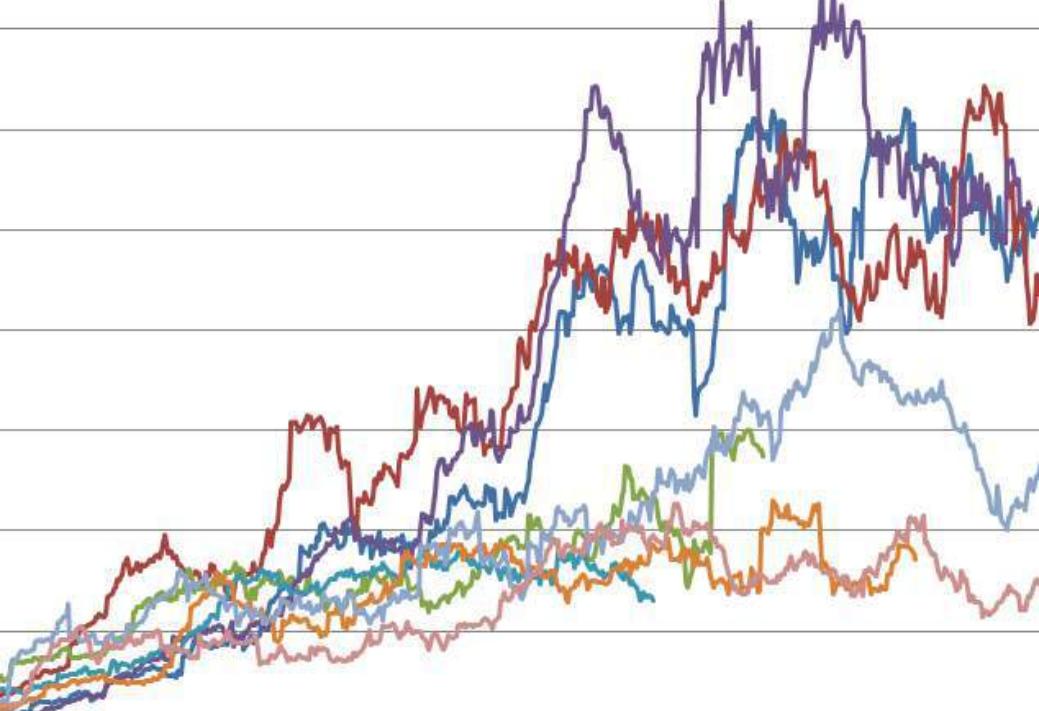
Prof. Wil van der Aalst

Marco Pegoraro, M.Sc.

Tsunghao Huang, M.Sc.

This lecture: Two Main Topics

Time series analysis



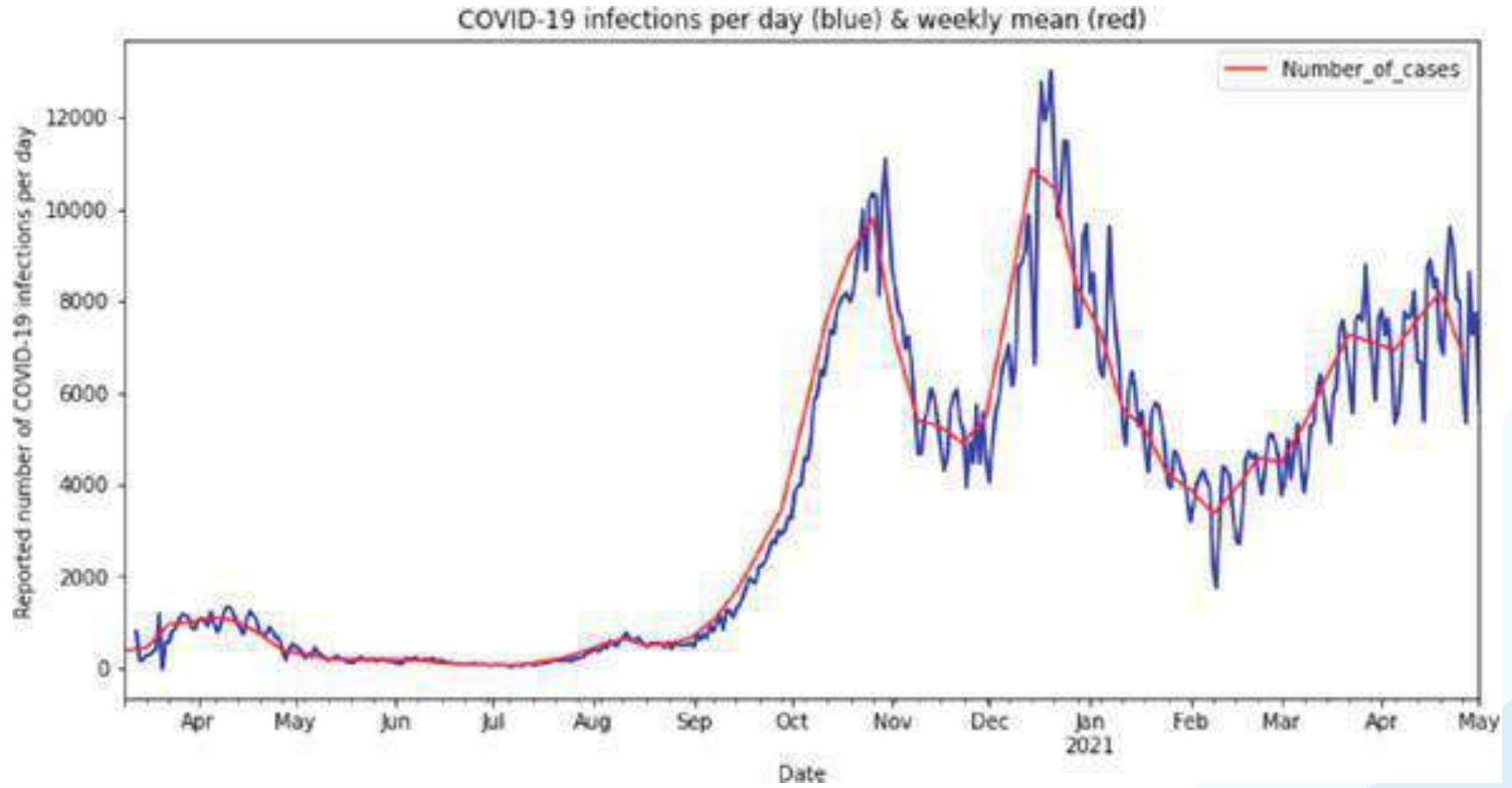
Data Quality, and Preprocessing



Generated using DALL E3

Time Series

1. Introduction
2. Analysis
3. Forecasting

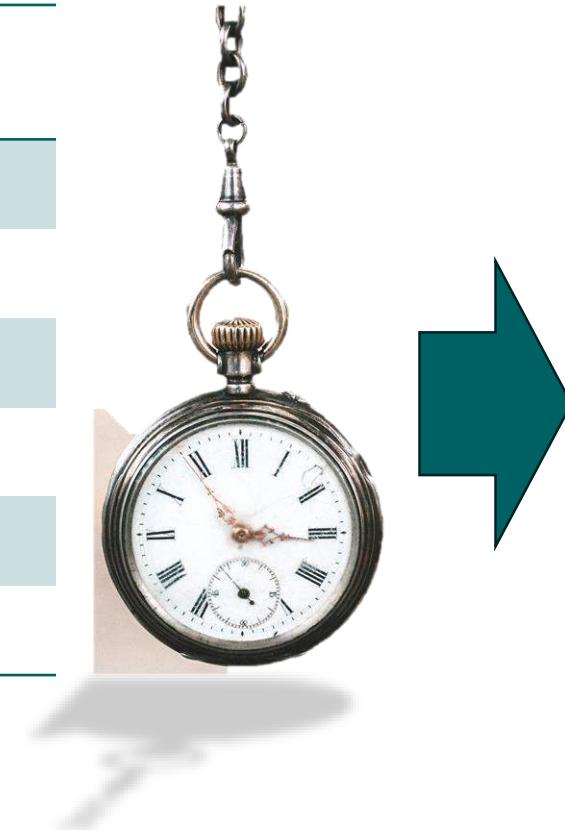


Taken from Time Series Forecasting on COVID-19 Data 10.5772/intechopen.104920

Temporal Data – Discrete Timestamped Events

Time-stamp	f_1	f_2	f_3	f_4	...	f_D
t_1						
t_2						
t_3						
t_4						
t_5						
...						

Every instance happened at a specific time



- **Sequence mining**
(seen before)
- **Process mining**
(later lecture)
- **Time series**
(this lecture)

Temporal Data – Discrete Timestamped Events

Time series data

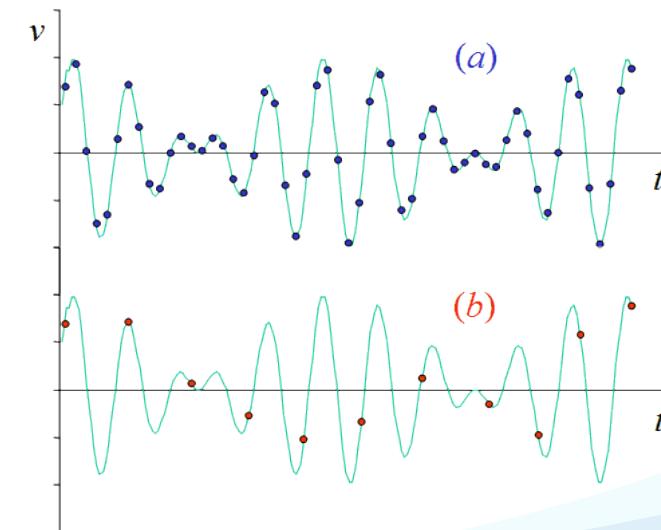
Time-stamp	f_1	f_2	f_3	f_4	...	f_D
t_1						
t_2						
t_3						
t_4						
t_5						
...						

Intervals are typically equal

numerical

Times series analysis

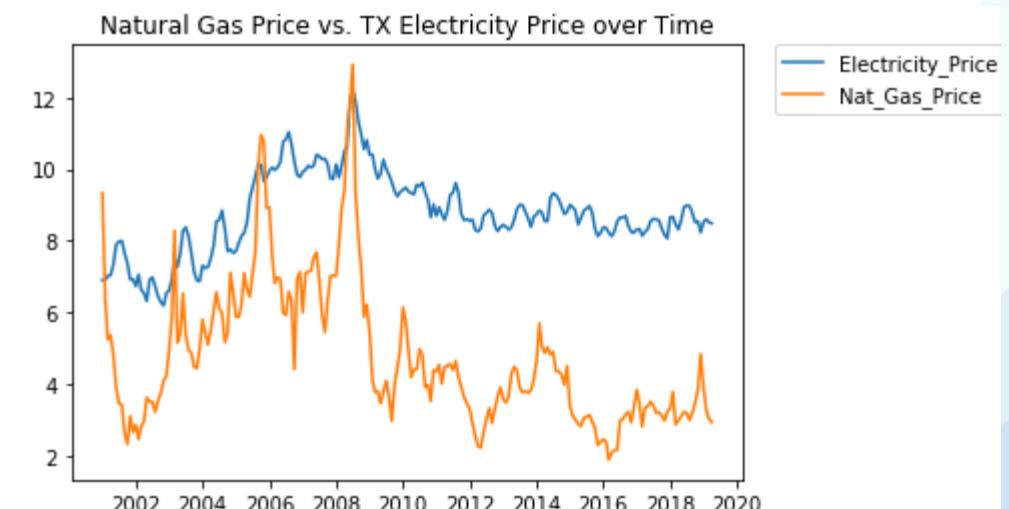
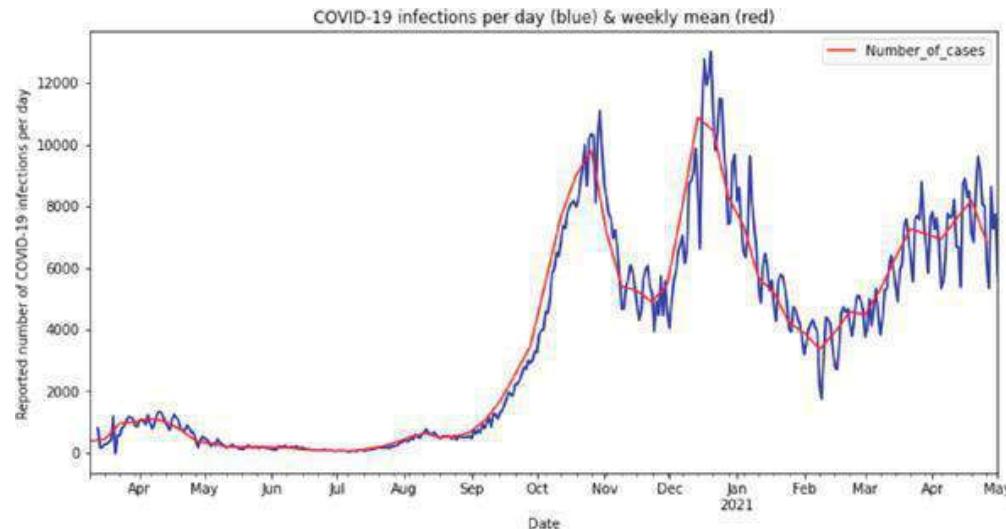
- Equidistant timestamps (determined by sampling rate)
- Features are numerical



Sampling rate
millisecond,
second, hour,
day, week, etc.

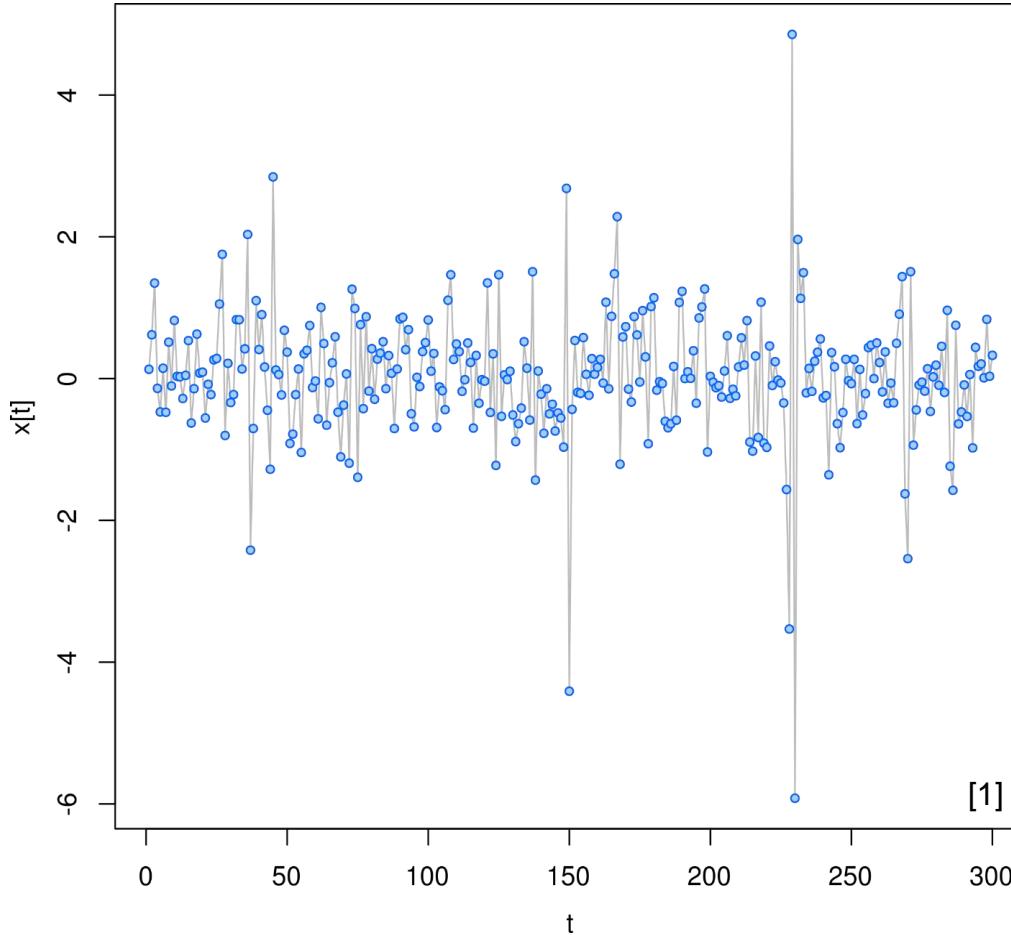
Time Series Analysis Has Numerous Applications

- Health (e.g., Covid)
- Finance (e.g., stock market)
- Inventory management
- Marketing
- Energy systems
- Climate change
- Political polls
- Etc.



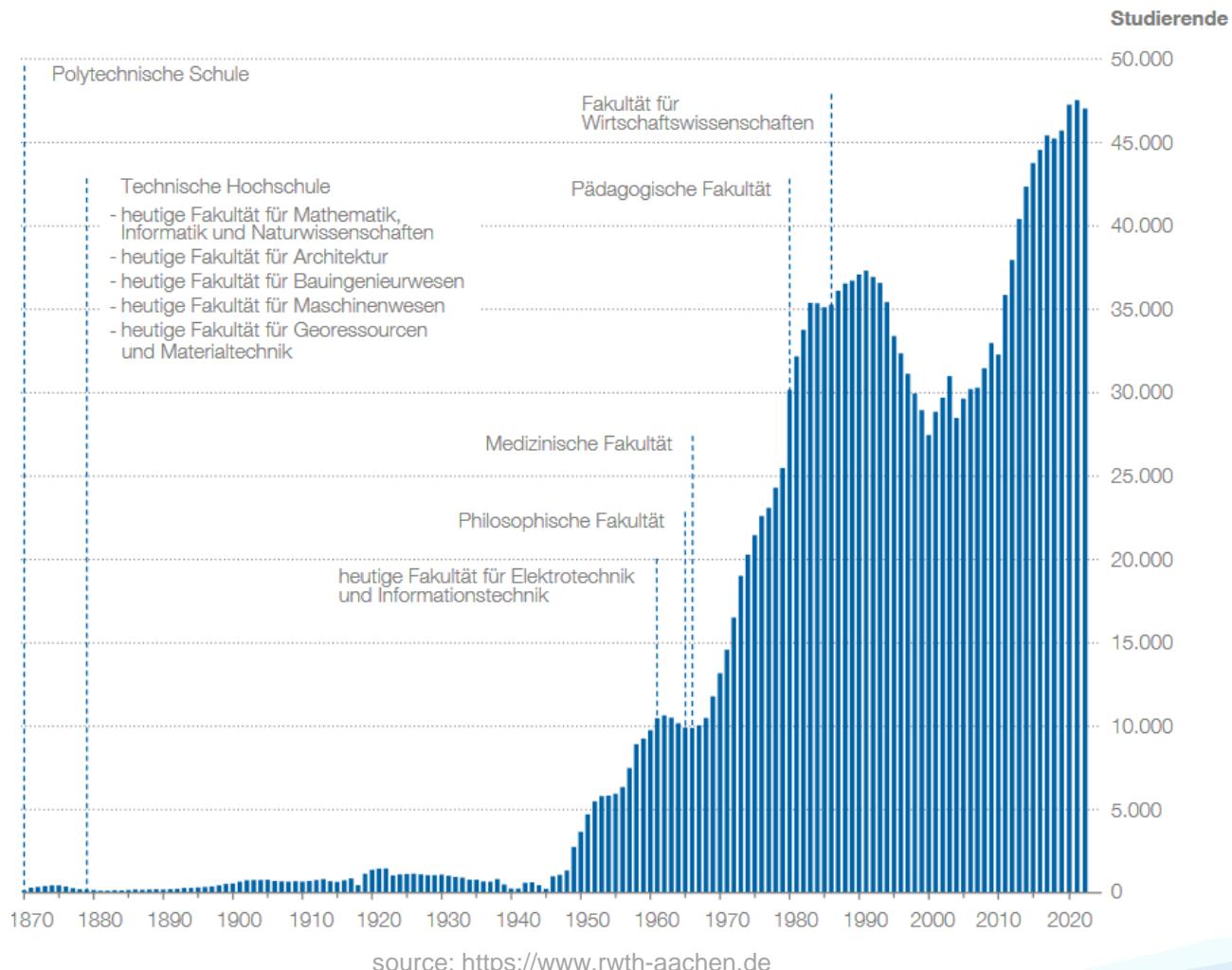
Let's Start With Univariate Time Series

Time-stamp	x
00:01	0.9
00:02	0.5
00:03	1.6
00:04	-0.8
00:05	-0.4
00:06	0.1
00:07	-0.3
00:08	-0.4
...	...



Time Series – Some More Examples

- Number of RWTH students



Time Series - Some More Examples

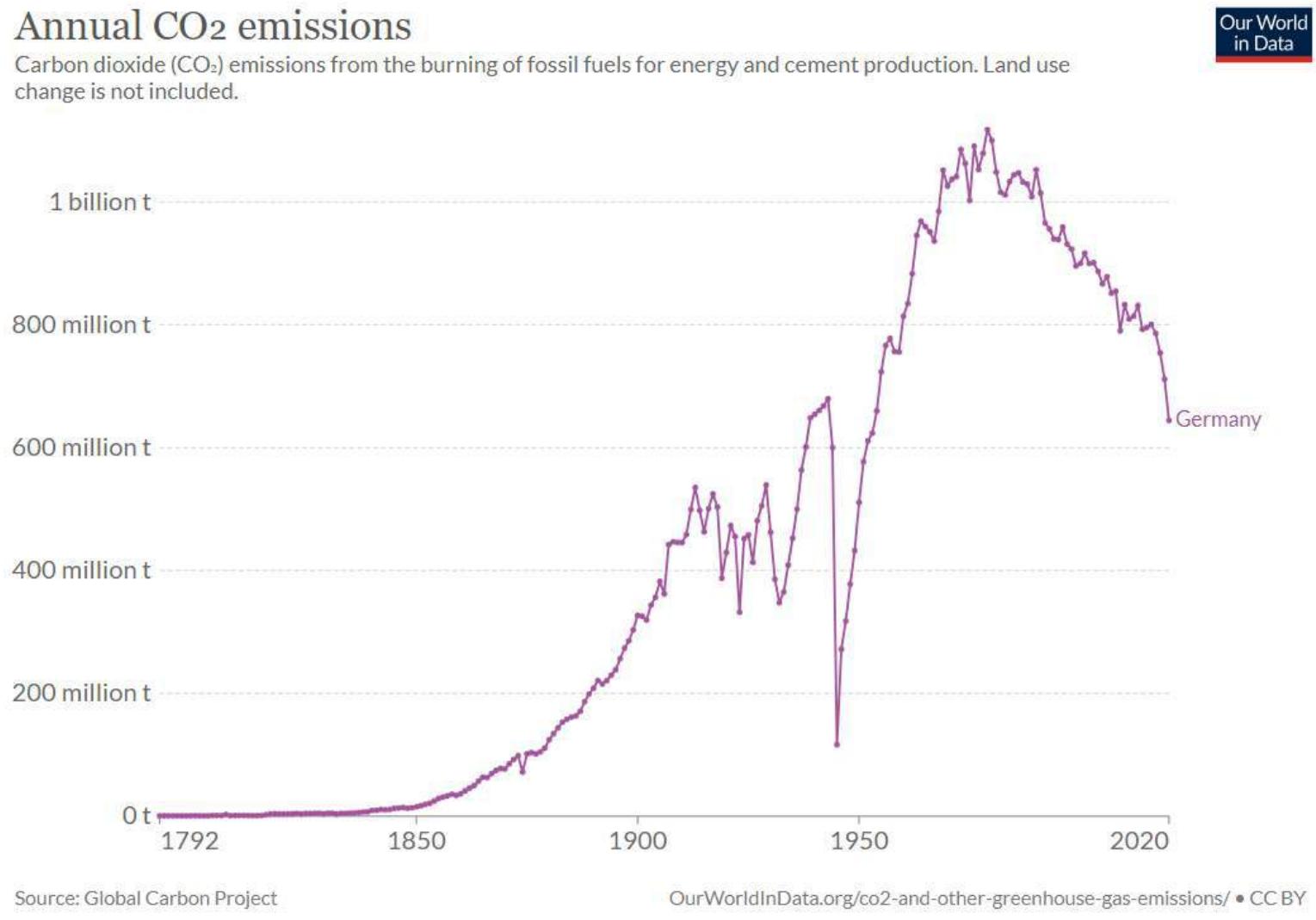
- Finance: stock price

Tesla, Inc.



Time Series - Some More Examples

- Climate: CO₂



Time Series

1. Introduction
2. **Analysis**
3. Forecasting

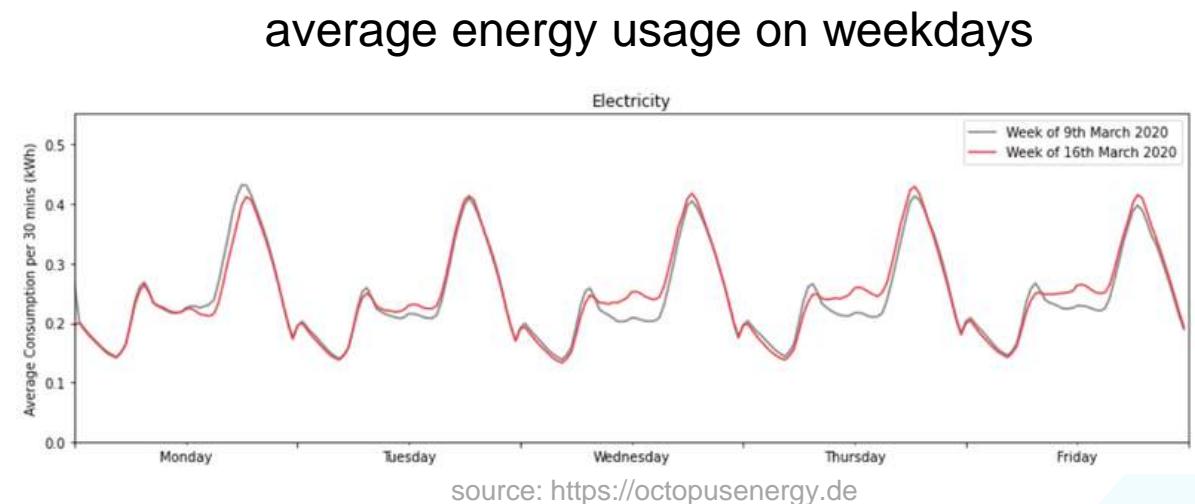


Time Series Patterns

- **Trend:** a time series exhibits a long-term increase or decrease in the data.
- **Seasonal:** a times series is affected by a fixed and known frequency (e.g., month, weekday).
- **Cyclic:** a time series exhibits rise and falls that are not of a fixed frequency (e.g., economic fluctuations).



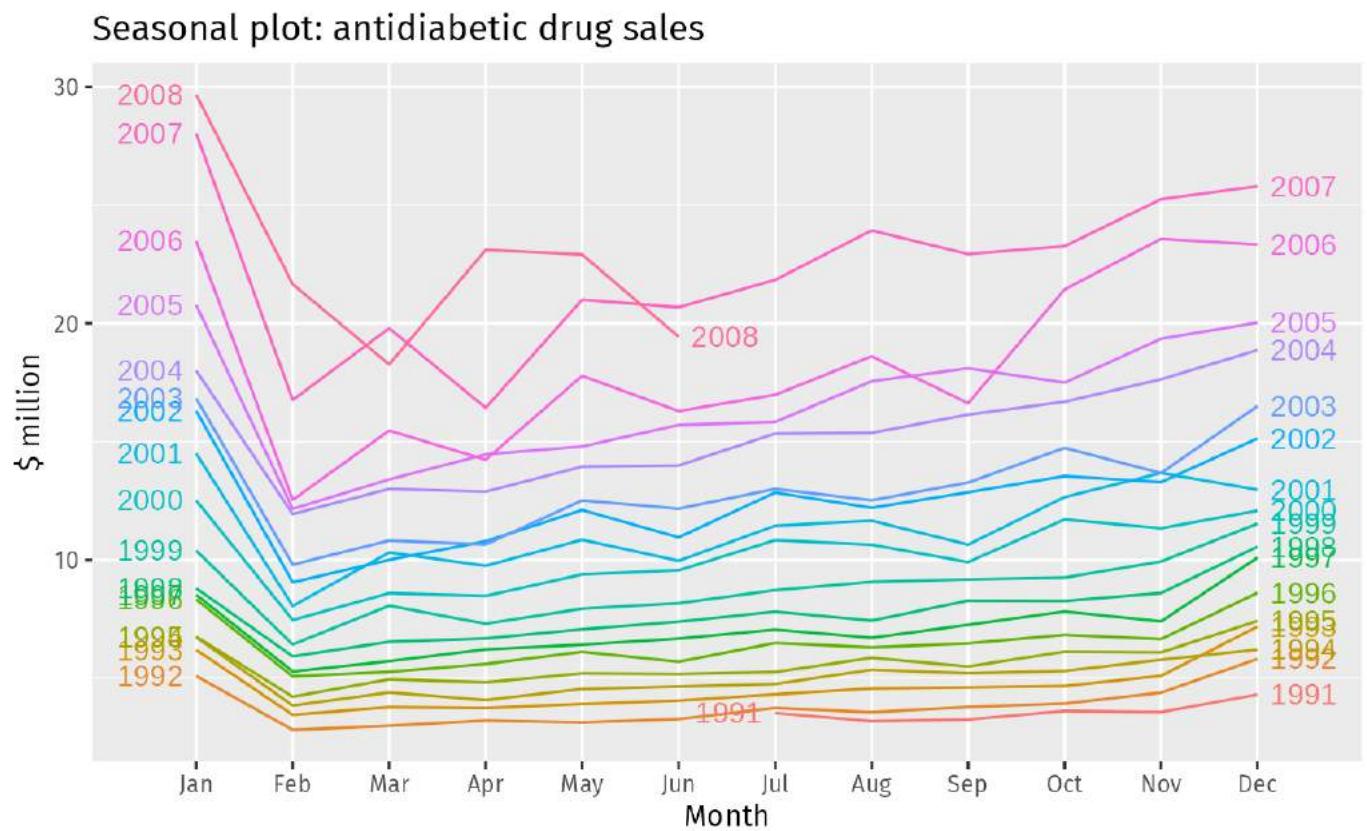
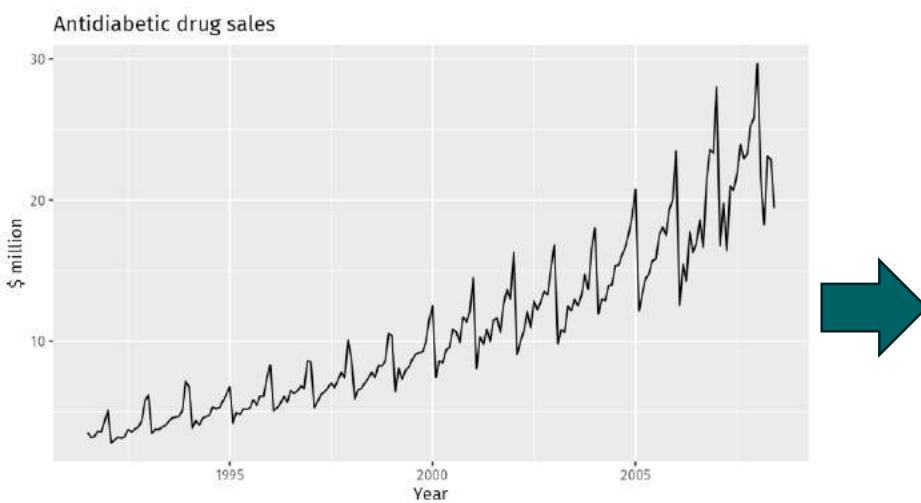
trend and cyclic patterns



seasonal patterns

Seasonal Plot

- Similar to a time plot, but the observations are plotted against a season.
- Easier to observe the underlying seasonal patterns.



Autocorrelation

- The linear relationship between lagged values of a time series.

$$r_k = \frac{\sum_{t=k+1}^T (y_t - \bar{y})(y_{t-k} - \bar{y})}{\sum_{t=1}^T (y_t - \bar{y})^2}$$

The length of the time series

The coefficient between y_t and y_{t-k}

Lag

Average of observation y

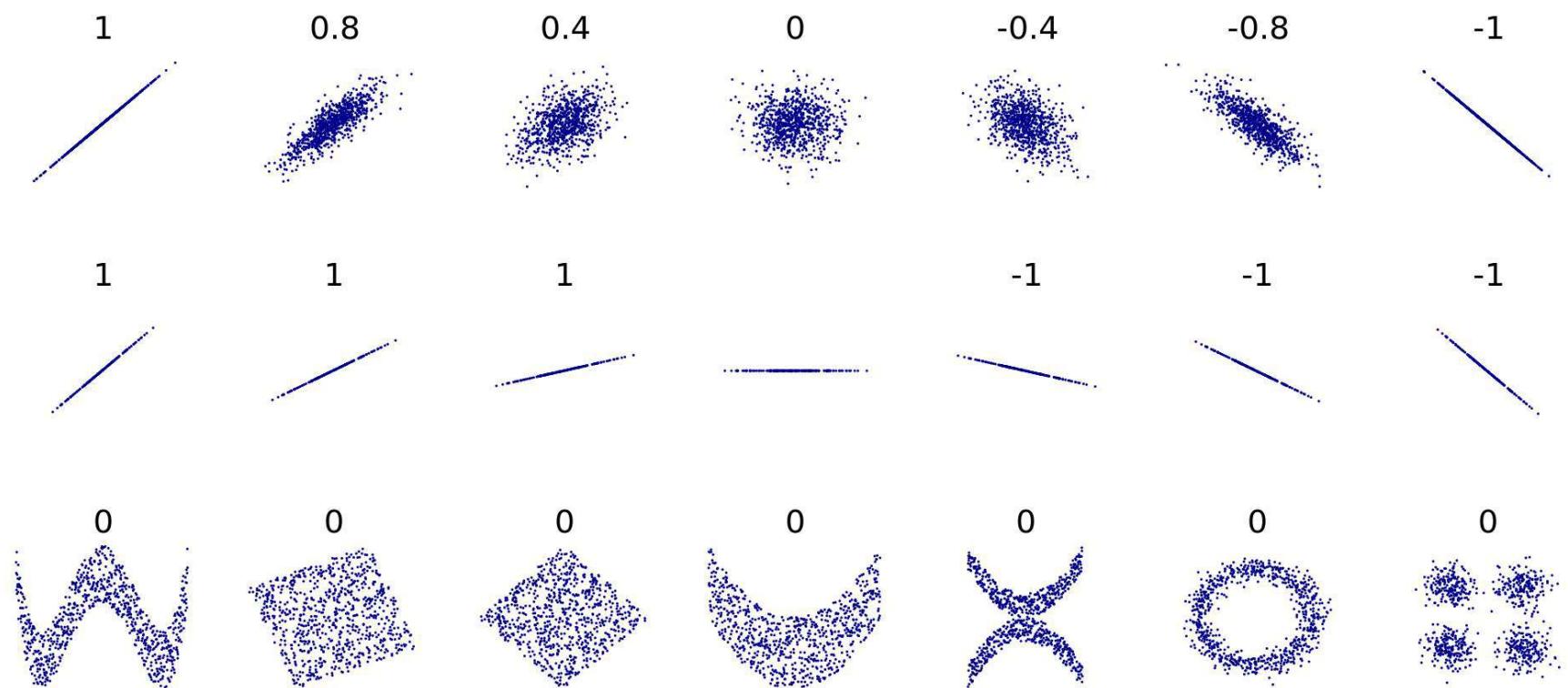
Observation at time t

Understanding Autocorrelation (1/2)

Sample correlation coefficient

$$r_{xy} = \frac{\sum_{i=1}^n (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^n (x_i - \bar{x})^2} \sqrt{\sum_{i=1}^n (y_i - \bar{y})^2}}$$

Recall: the value of a correlation coefficient ranges between -1 and $+1$.



Understanding Autocorrelation (2/2)

Sample correlation coefficient

$$r_{xy} = \frac{\sum_{i=1}^n (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^n (x_i - \bar{x})^2} \sqrt{\sum_{i=1}^n (y_i - \bar{y})^2}}$$

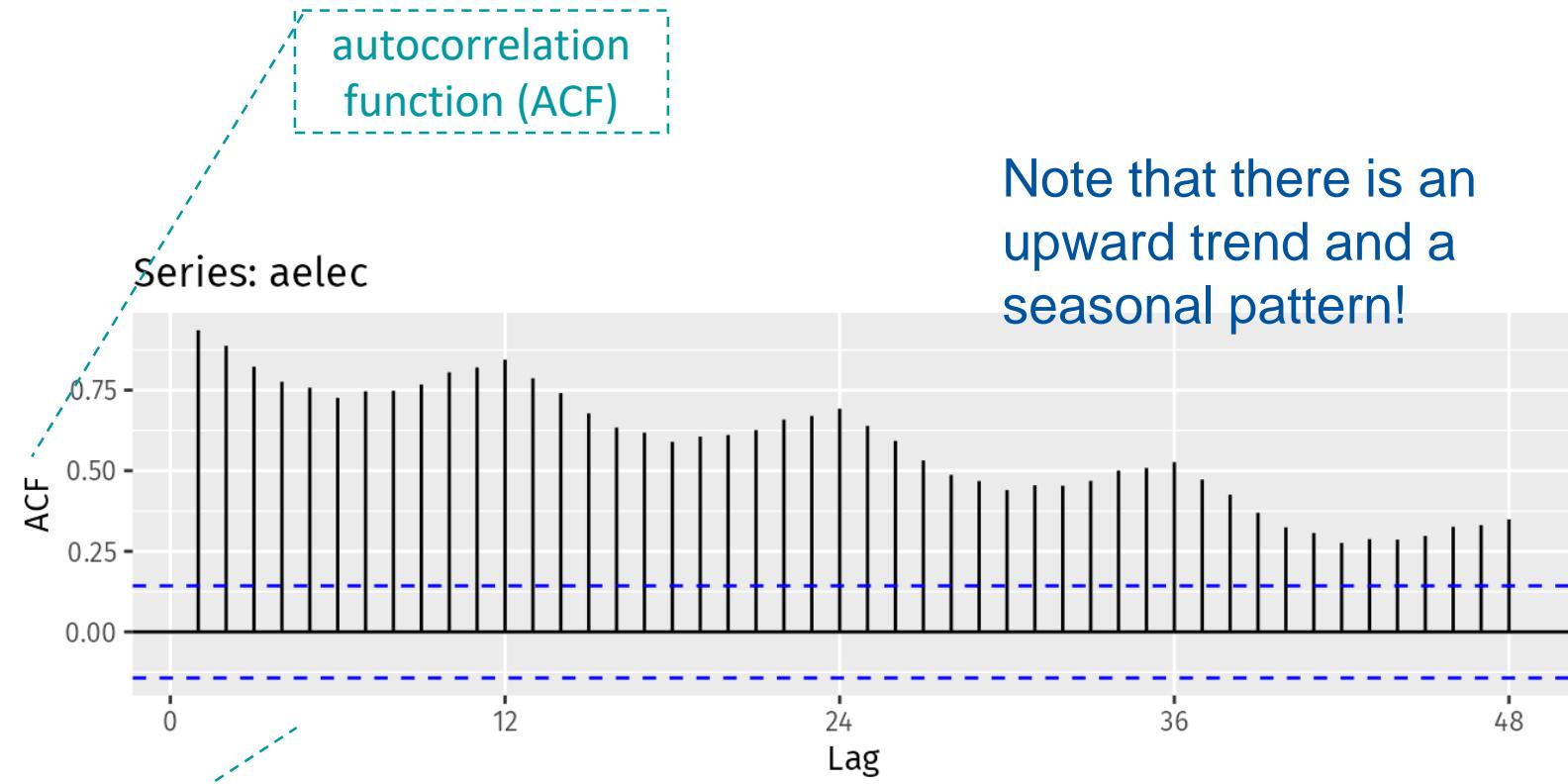
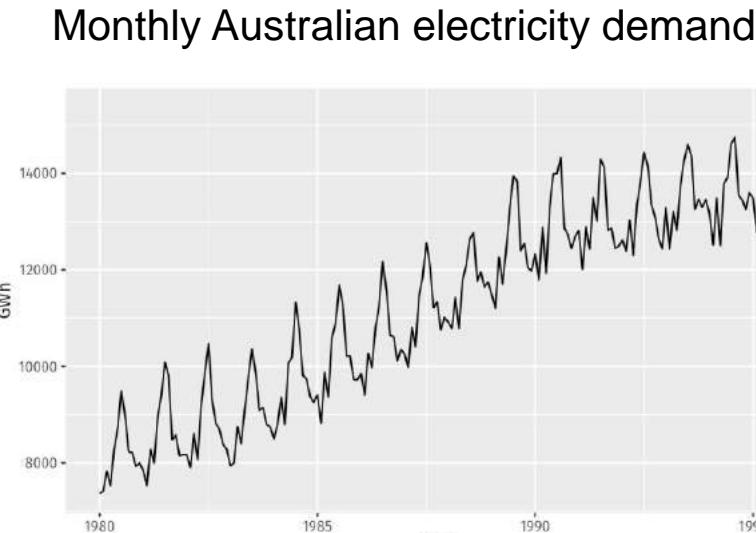
Autocorrelation

$$r_k = \frac{\sum_{t=k+1}^T (y_t - \bar{y})(y_{t-k} - \bar{y})}{\sum_{t=1}^T (y_t - \bar{y})^2}$$

There is just one sample mean \bar{y} ($\bar{x} = \bar{y}$) and typically $T \gg k$.

Correlogram

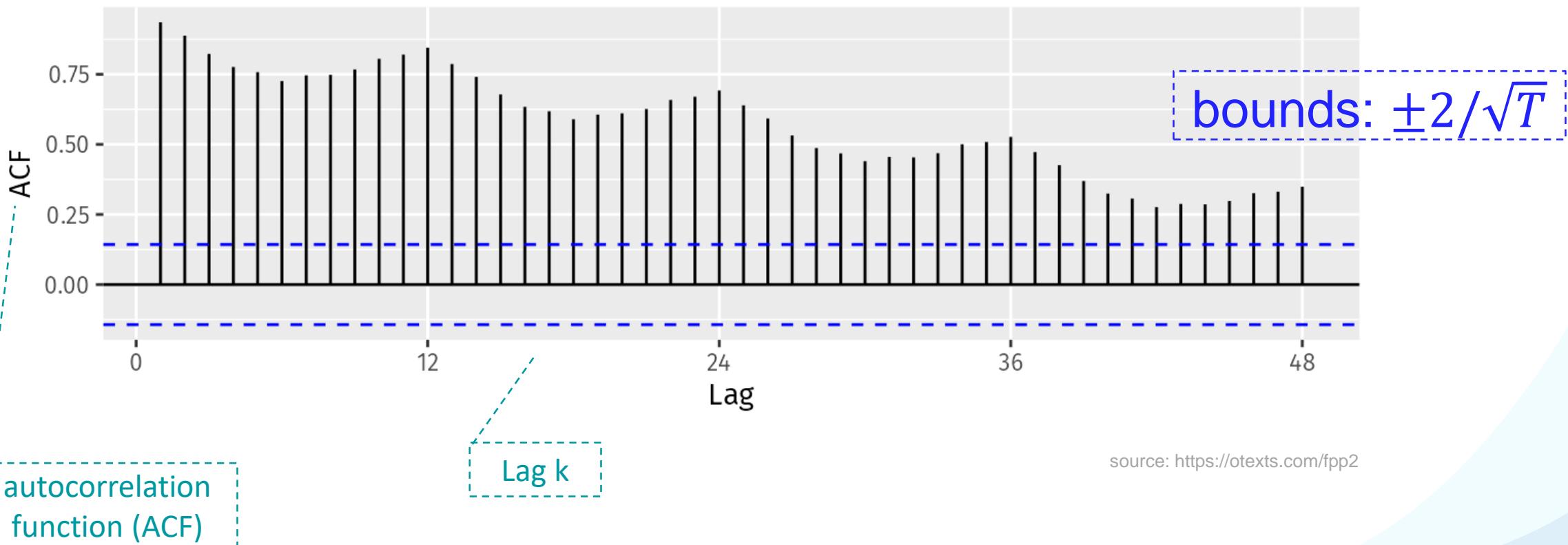
- A plot showing the autocorrelation coefficients between lagged values.



Correlogram

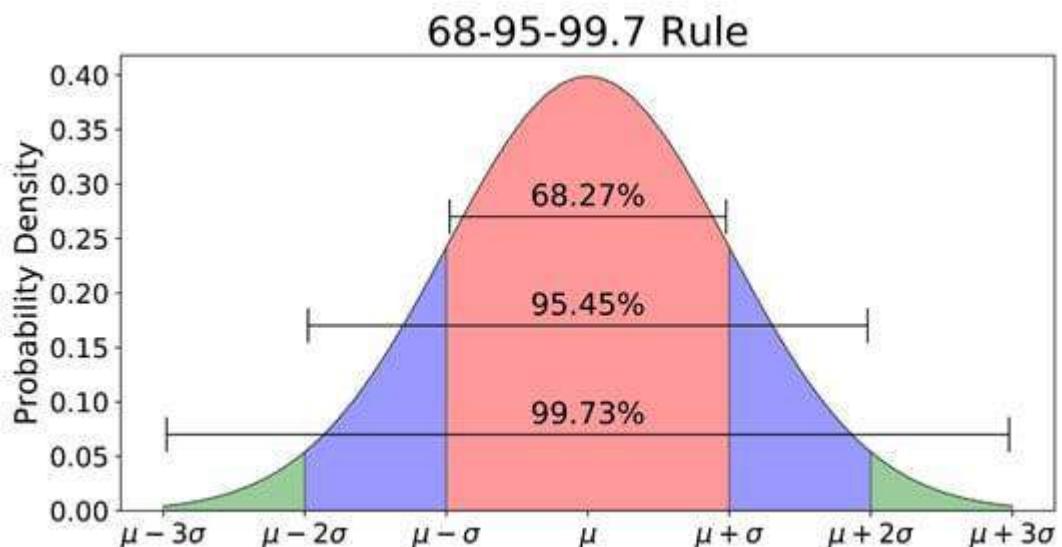
- Bounds: $\pm 2/\sqrt{T}$, where T is the number observations in the time series.

Series: aelec



Correlated or Not?

- Assume independence of subsequent observations.
- Under this assumption, the autocorrelation coefficient r_k follows (approximately!) a normal distribution with a mean of 0 and a variance of $1/T$, i.e., $\mu = 0$ and $\sigma = 1/\sqrt{T}$.
- This means that approximately 68% of the autocorrelation coefficients fall within $[-\sigma, \sigma]$, 95% of the autocorrelation coefficients fall within $[-2\sigma, 2\sigma]$, and 99.5% of the autocorrelation coefficients fall within $[-3\sigma, 3\sigma]$,



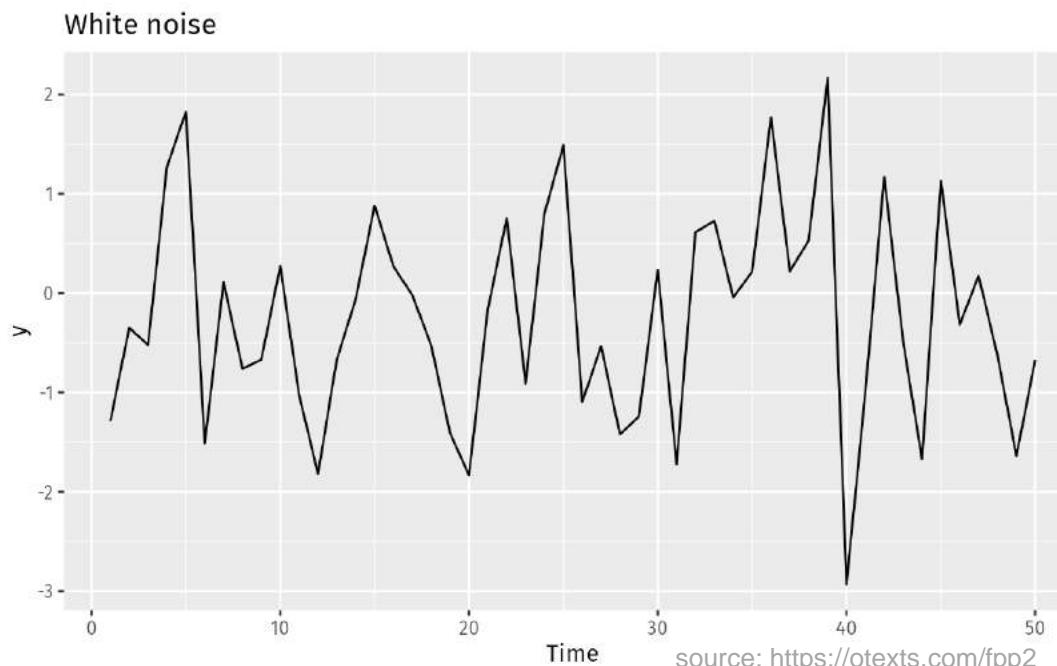
95% of the autocorrelation coefficients fall within $[-2/\sqrt{T}, 2/\sqrt{T}]$

Therefore, values outside of this interval are unlikely (less than 5%) under the independence assumption.

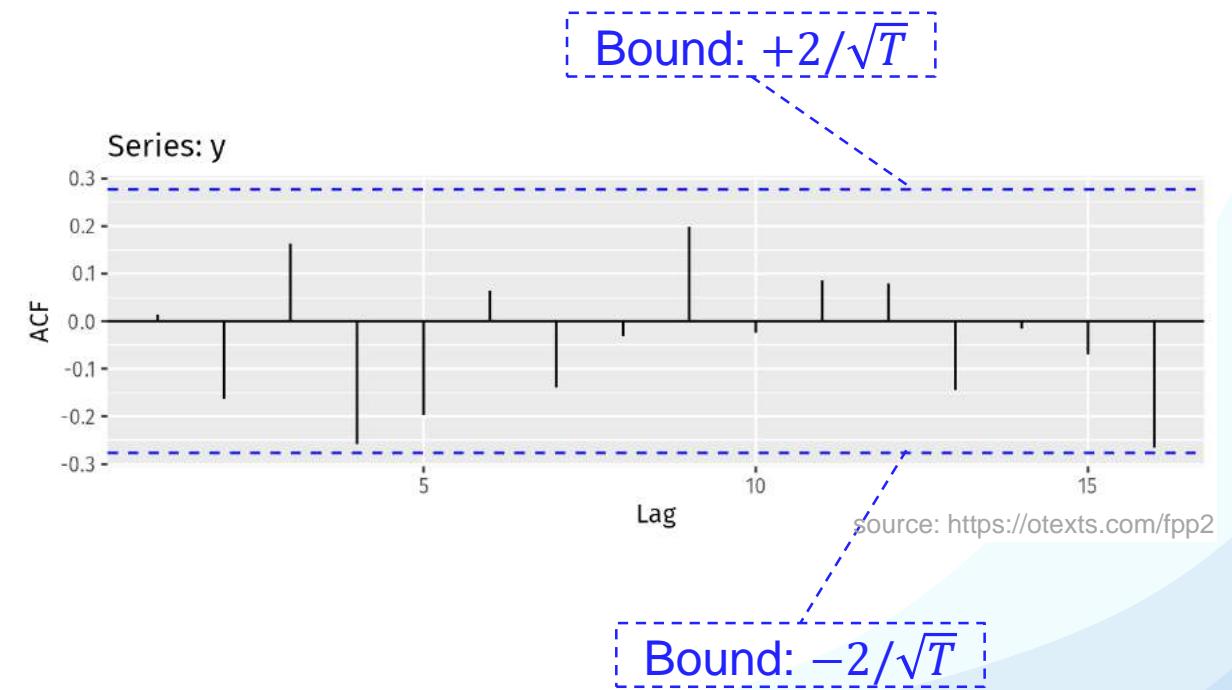
$$r_k = \frac{\sum_{t=k+1}^T (y_t - \bar{y})(y_{t-k} - \bar{y})}{\sum_{t=1}^T (y_t - \bar{y})^2}$$

White Noise

- Time series consisting of **independent** observations are called **white noise**.
- Under this assumption there is just a 5% chance that the autocorrelation coefficient will fall outside of the bounds $[-2/\sqrt{T}, 2/\sqrt{T}]$ by chance.
- Spikes outside these bounds, suggest that there may be a correlation.



white noise randomly generated



Time Series Decomposition

- Various patterns coexist
- May be helpful to split a time series into several components
- Additive decomposition

$$y_t = S_t + T_t + R_t$$

The diagram illustrates the additive decomposition of a time series. A horizontal dashed line represents the total time series y_t . Three vertical dashed lines extend downwards from this line to the components: the top dashed line is labeled "trend-cycle component", the bottom-left dashed line is labeled "seasonal component", and the bottom-right dashed line is labeled "remainder component".

- Alternative: multiplicative decomposition

$$y_t = S_t \times T_t \times R_t$$

Time Series Decomposition

Estimating the trend-cycle component T_t : moving average

- The estimate of the T_t is obtained by averaging values of the time series within k periods of t .
- A moving average of order $m = 2k + 1$ is called **m -MA**.

$$\hat{T}_t = \frac{1}{m} \sum_{j=-k}^k y_{t+j}$$

$$m = 2k + 1$$

Year	Sales (GWh)	5-MA
1989		2354.34
1990		2379.71
1991		2318.52
1992		2468.99
1993		2386.09
1994	2569.47	2424.56
1995	2575.72	2463.76
1996	2762.72	2552.60
1997	2844.50	2627.70
1998	3000.70	2750.62
1999	3108.10	2858.35
2000	3357.50	3014.70
2001	3075.70	3077.30
2002	3180.60	3144.52
2003	3221.60	3188.70
2004	3176.20	3202.32
2005	3430.60	3216.94
2006	3527.48	3307.30
2007	3637.89	3398.75
2008	3655.00	3485.43

$$k = 2$$

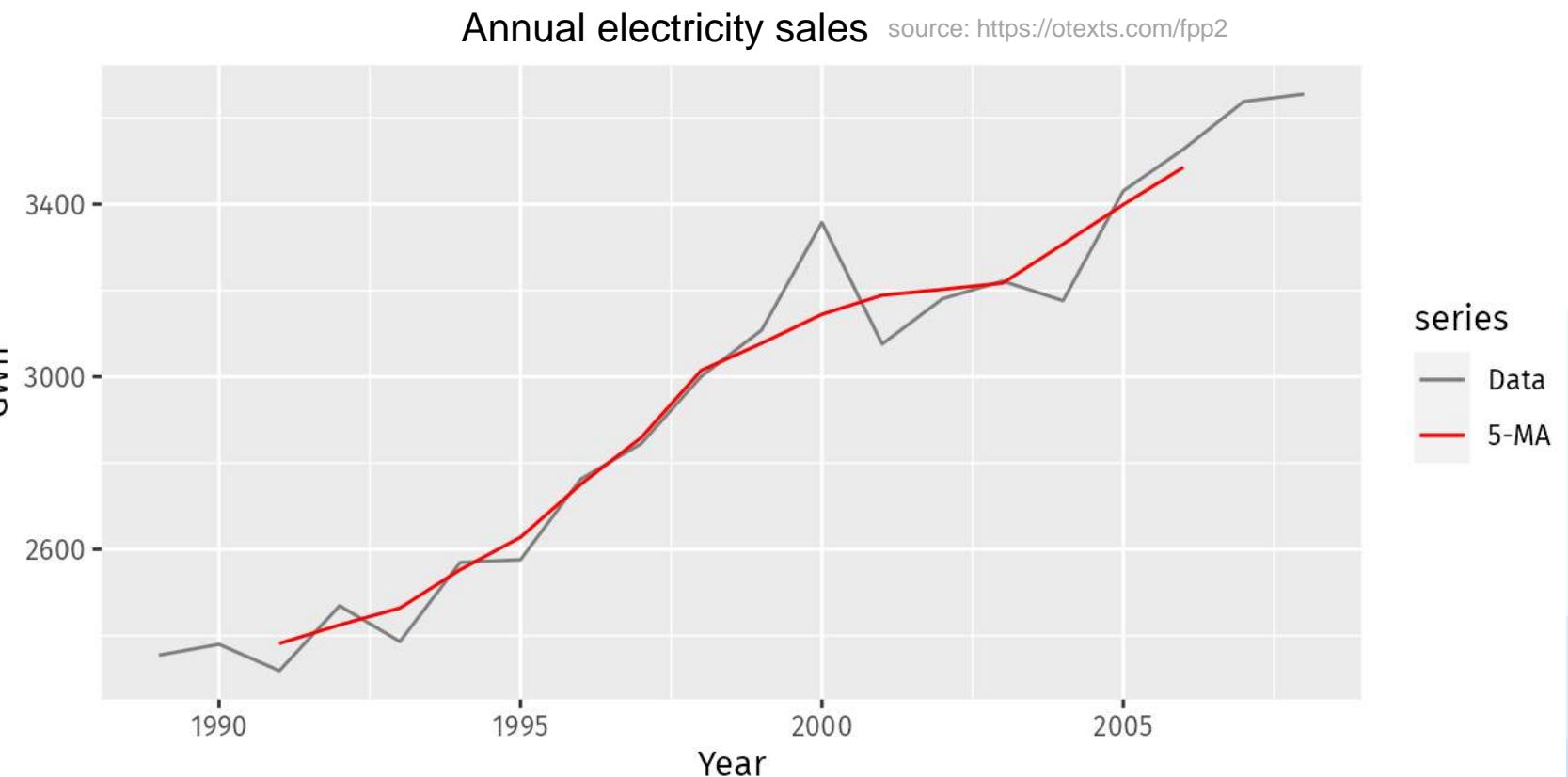
$$m = 5$$

Time Series Decomposition

Estimating the trend-cycle component T_t : moving average

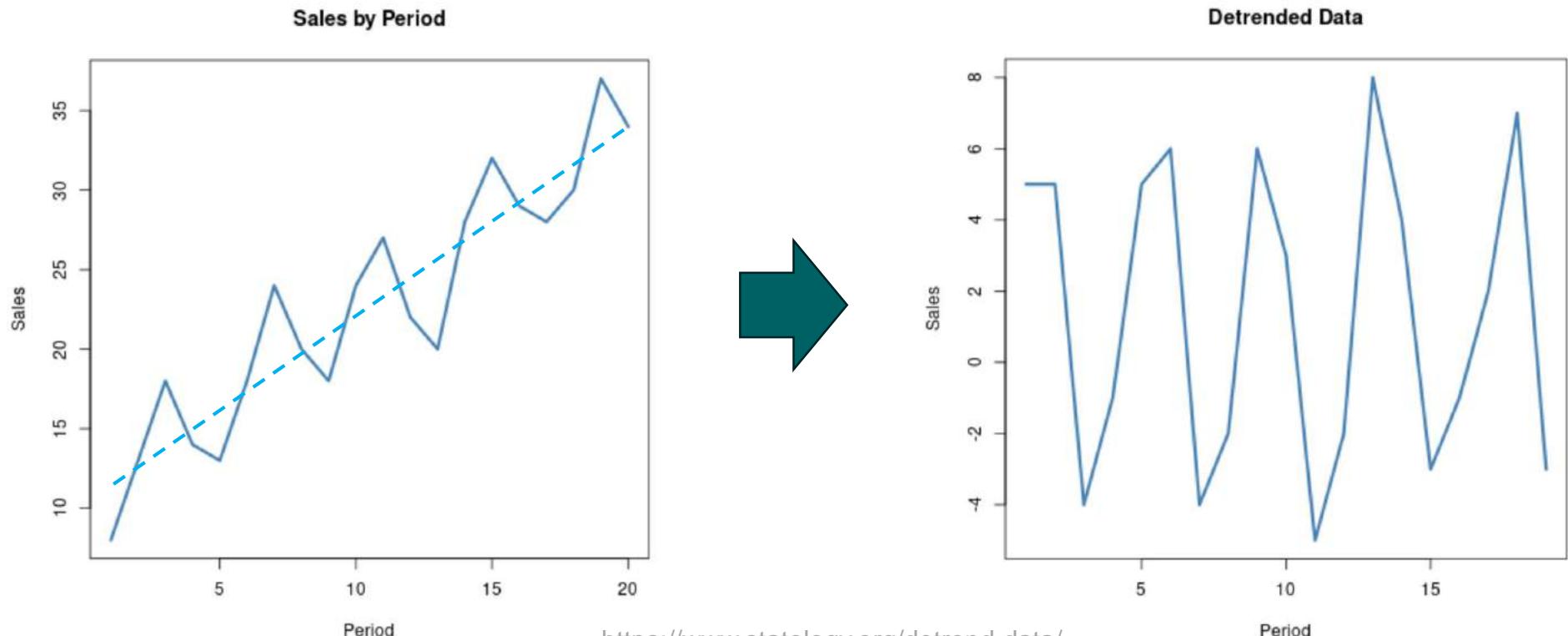
- The estimate of the T_t is obtained by averaging values of the time series within k periods of t .
- A moving average of order m is called **m -MA**

$$\hat{T}_t = \frac{1}{m} \sum_{j=-k}^k y_{t+j}$$



Time Series Decomposition

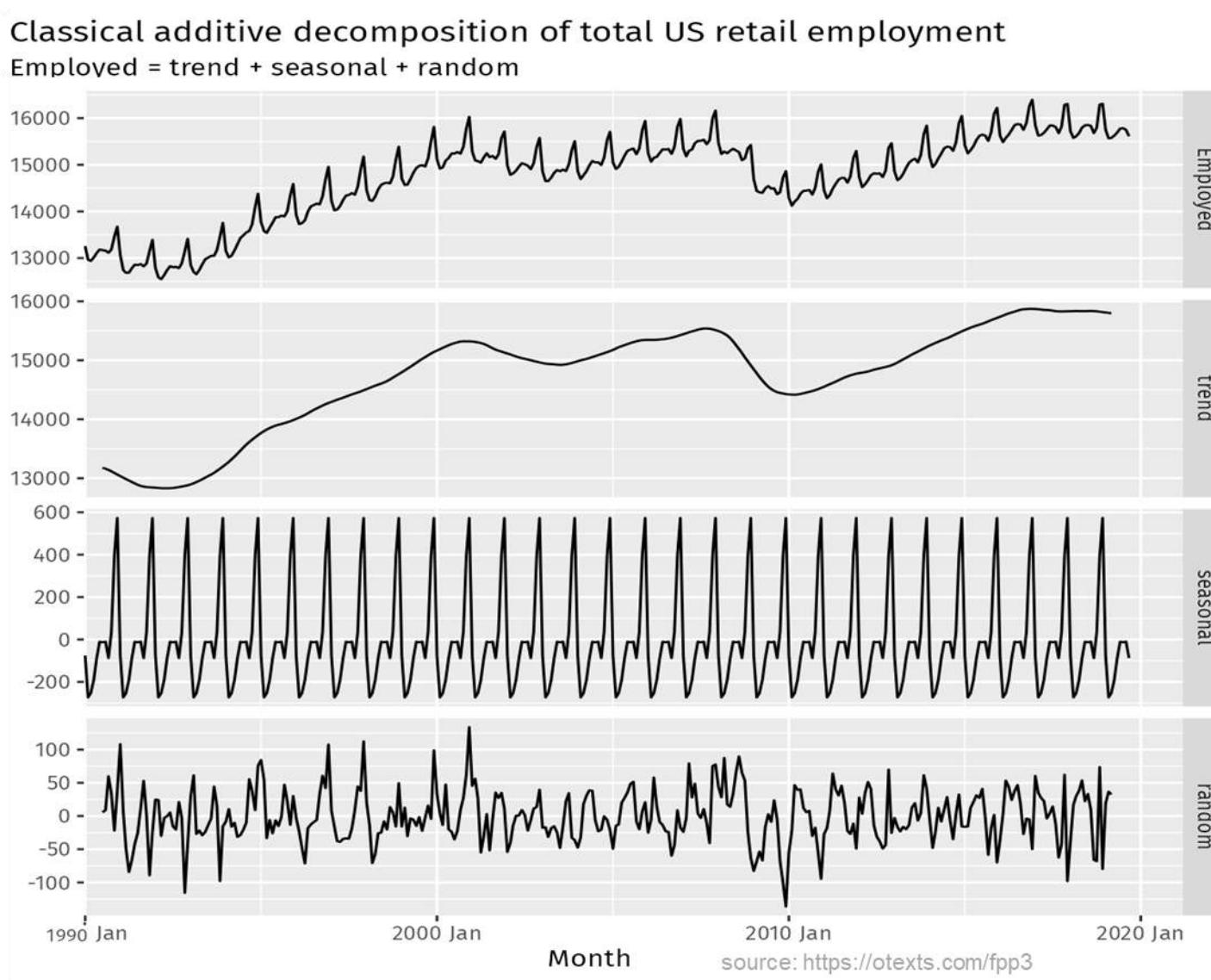
- To **detrend** time series data means to remove an underlying trend in the data.
- This way one can see subtrends in the data that are seasonal or cyclical.
- Calculate the **detrend series**: $y_t - \hat{T}_t$.
- Similarly, one can detrend for seasonal effects.



Time Series Decomposition

Classical additive decomposition of total US retail employment

Employed = trend + seasonal + random



$$y_t = S_t + T_t + R_t$$

trend-cycle

$$T_t$$

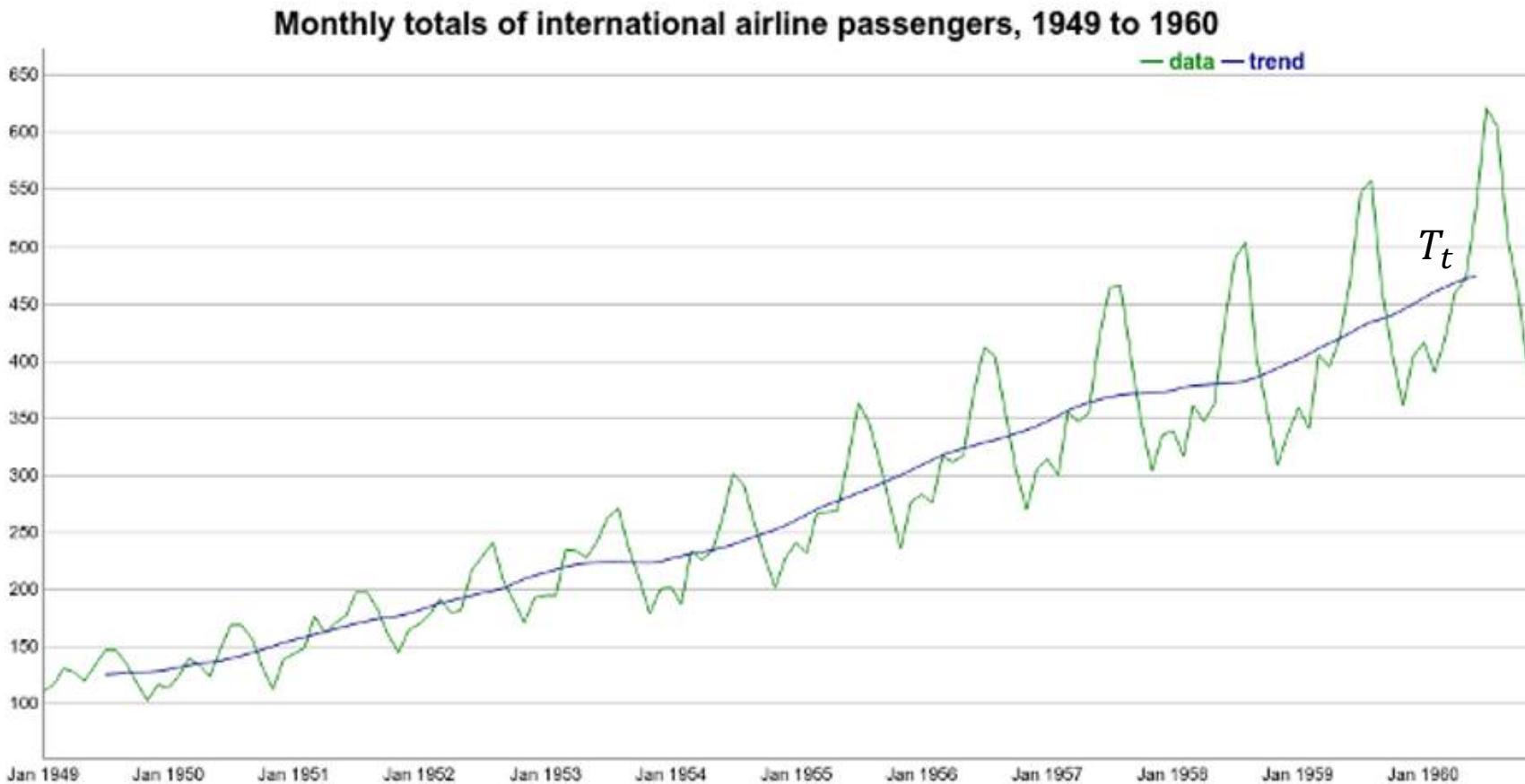
seasonal

$$S_t$$

remainder

$$R_t$$

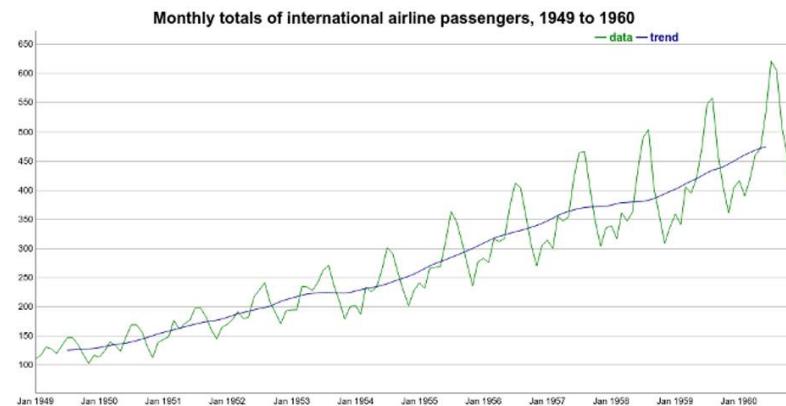
Time Series Decomposition: Example



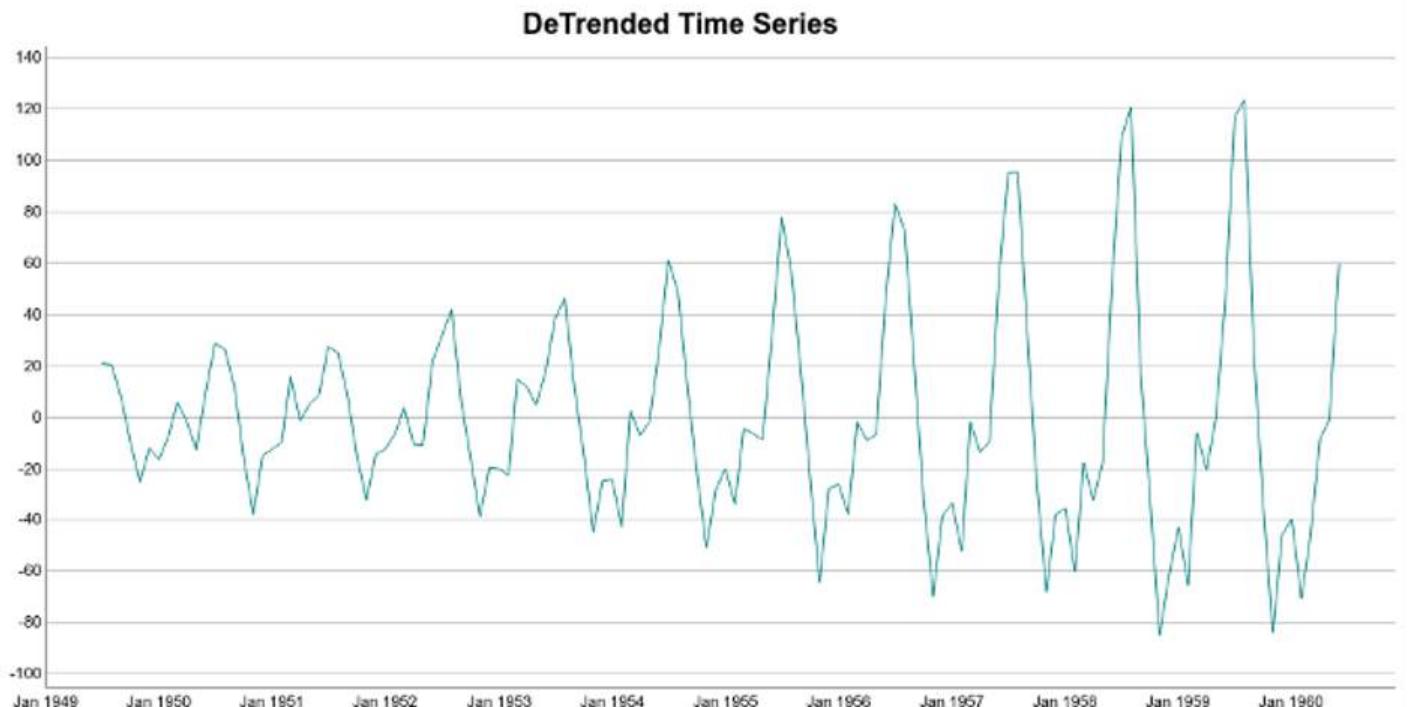
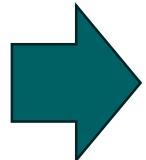
$$y_t = S_t + T_t + R_t$$

Example taken from <https://www.encora.com/insights/a-visual-guide-to-time-series-decomposition-analysis>

Time Series Decomposition: Example

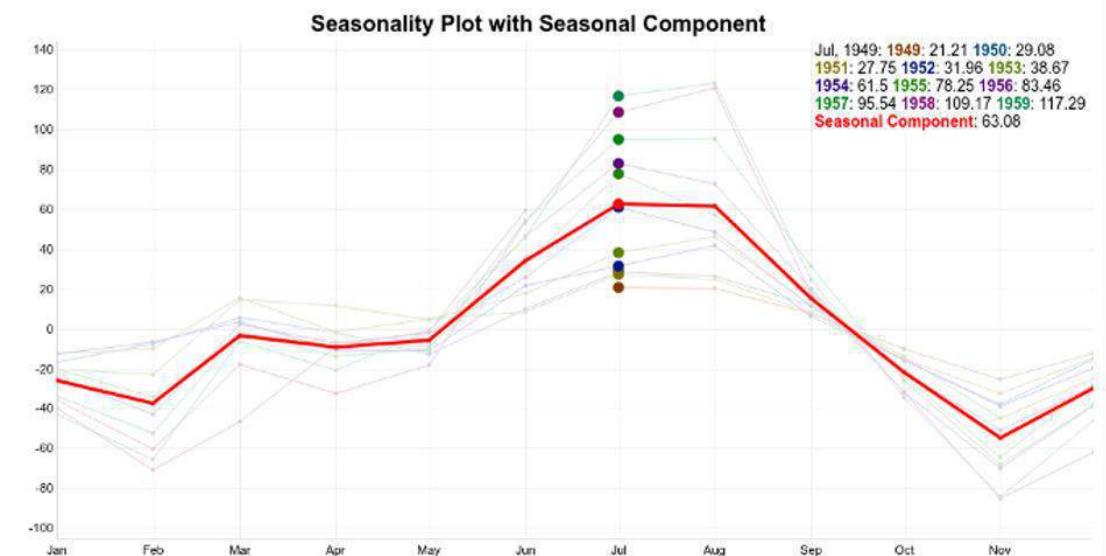
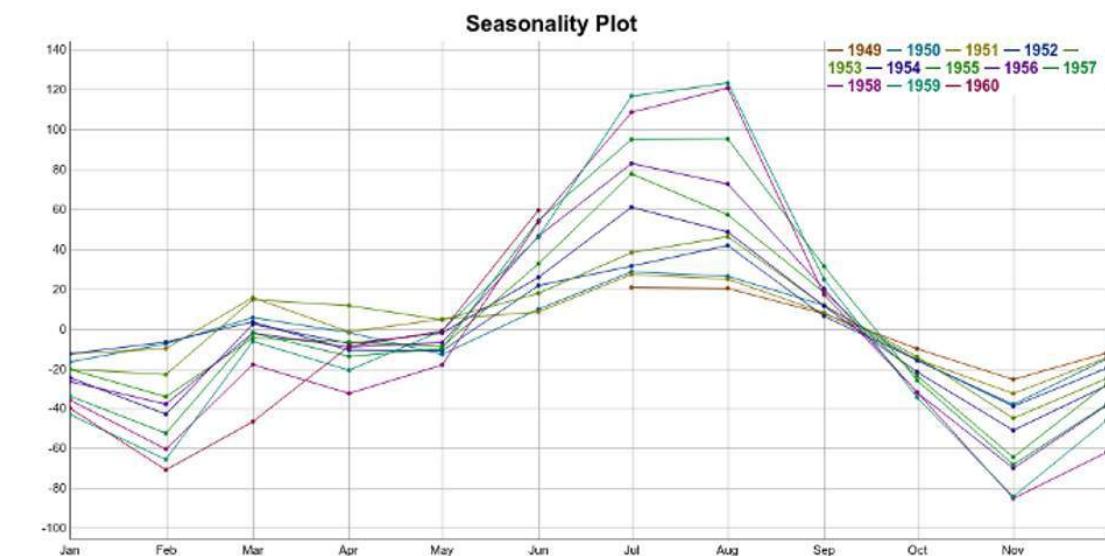


$$y_t = S_t + T_t + R_t$$

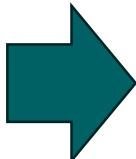
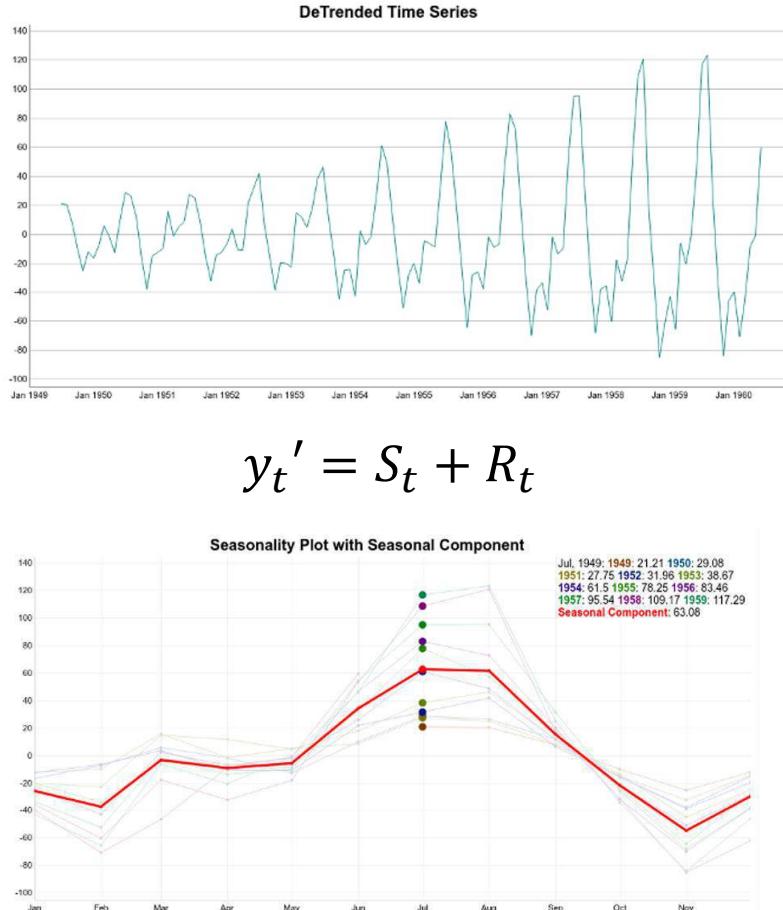
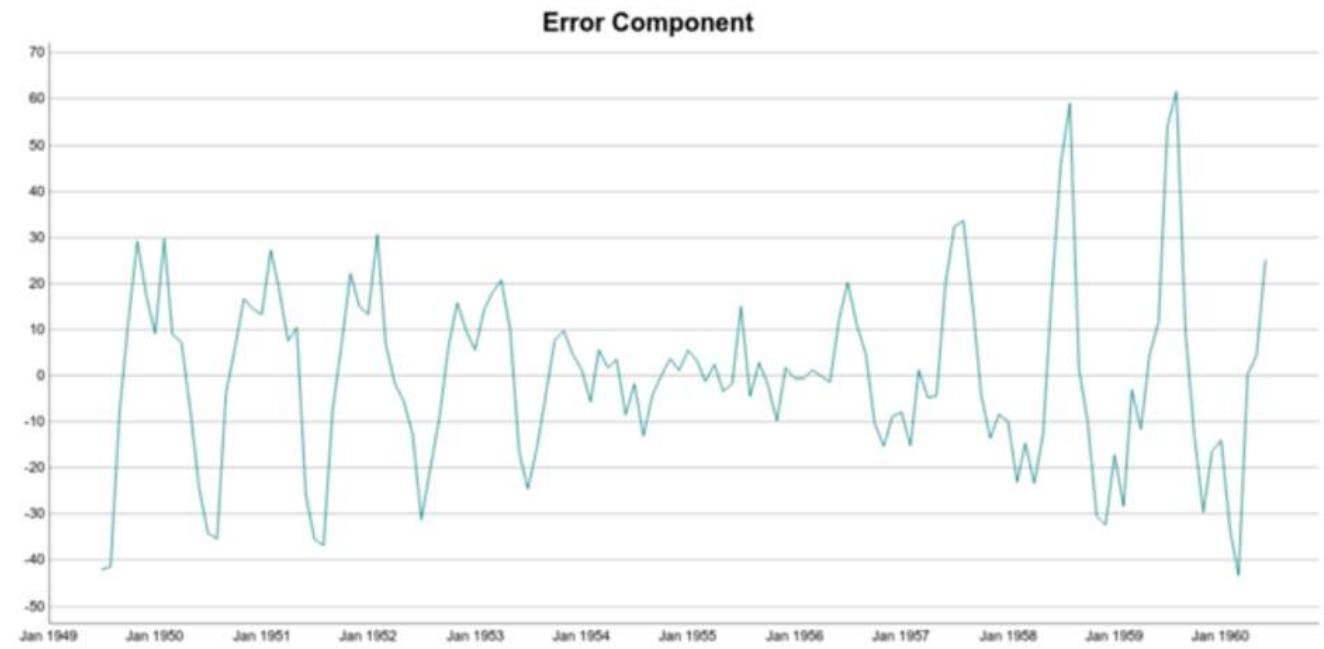


$$y'_t = S_t + R_t = y_t - T_t$$

Time Series Decomposition: Example

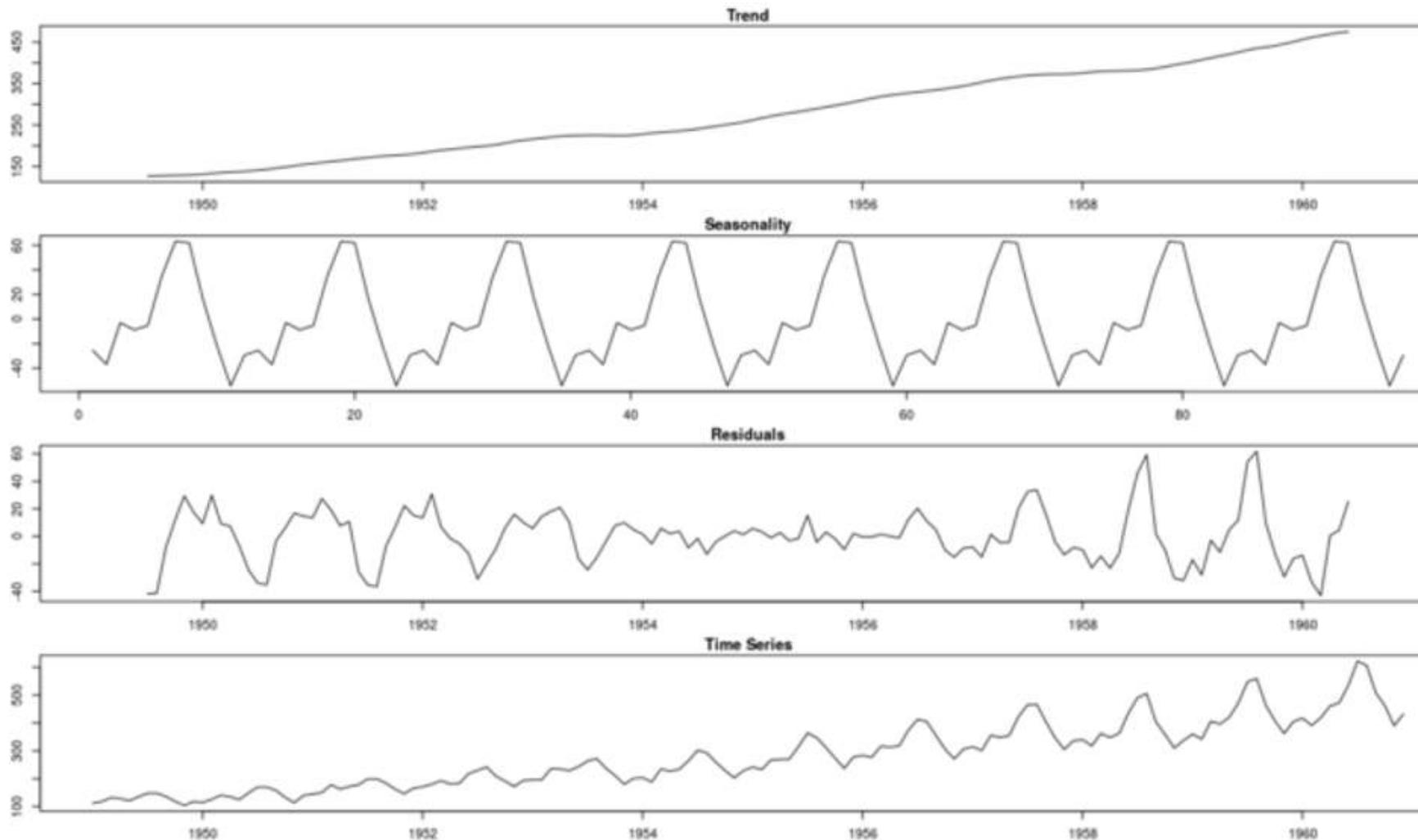
 S_t

Time Series Decomposition: Example

 S_t 

$$y_t'' = R_t = y_t - (S_t + T_t)$$

Time Series Decomposition: Example

 T_t S_t R_t

$$y_t = S_t + T_t + R_t$$

Time Series

1. Introduction
2. Analysis
3. **Forecasting**



Autoregressive (AR) Models

- An Autoregressive (AR) model is a regression of the variable against itself.
- The variable of interest is forecasted using a linear combination of past values of the variable.

$$y_t = c + \phi_1 y_{t-1} + \phi_2 y_{t-2} + \dots + \phi_p y_{t-p} + \varepsilon_t$$

Diagram illustrating the components of the AR(p) model:

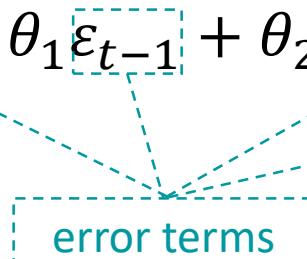
- coefficient**: Labels the term ϕ_1 .
- past value of y at time $t - 1$** : Labels the term y_{t-1} .
- error term**: Labels the term ε_t .

- It is referred to as a $AR(p)$ model, an Autoregressive (AR) model of order p .

ϕ is “phi”

Moving Average (MA) Models

- A Moving Average (MA) model is a regression of the past errors.
- The variable of interest is forecasted using a linear combination of past forecast errors.

$$y_t = c + \varepsilon_t + \theta_1 \varepsilon_{t-1} + \theta_2 \varepsilon_{t-2} + \cdots + \theta_q \varepsilon_{t-q}$$


The diagram shows the mathematical formula for a Moving Average (MA) model. The equation is $y_t = c + \varepsilon_t + \theta_1 \varepsilon_{t-1} + \theta_2 \varepsilon_{t-2} + \cdots + \theta_q \varepsilon_{t-q}$. Above the equation, each term $\varepsilon_t, \varepsilon_{t-1}, \varepsilon_{t-2}, \dots, \varepsilon_{t-q}$ is enclosed in a small blue dashed box. Below the equation, a larger dashed box also encloses these terms, with the label "error terms" centered inside it.

- Note that given $c, \theta_1, \theta_2, \dots, \theta_q$, it is possible to compute $\varepsilon_1, \varepsilon_2, \dots, \varepsilon_T$.
- It is referred to as a MA(q) model, a Moving Average (MA) model of order q .

θ is “theta”

Moving Average (MA) Models: Compute Errors

An example of an MA(1) Model

$$y_t = c + \varepsilon_t + \theta_1 \varepsilon_{t-1}$$

10

0.5

assumed to be given, e.g., through EM

Time	Forecasted Values (\hat{y}_t)	Error at time t (ε_t)	Actual Values (y_t)
1	10		9
2	$9.5 = 10 + 0.5(-1)$		11.5
3	$11 = 10 + 0.5(2)$		10
4	$9.5 = 10 + 0.5(-1)$		10.5
5	$11 = 10 + 0.5(2)$		10
6	$9 = 10 + 0.5(-1)$		9

can be derived

Moving Average (MA) Models: Compute Errors

An example of an MA(1) Model

$$y_t = c + \varepsilon_t + \theta_1 \varepsilon_{t-1}$$

10

0.5

assumed to be given, e.g., through EM

Time	Forecasted Values (\hat{y}_t)	Error at time t (ε_t)	Actual Values (y_t)
1	10	-1	9
2	$9.5 = 10 + 0.5(-1)$	2	11.5
3	$11 = 10 + 0.5(2)$	-1	10
4	$9.5 = 10 + 0.5(-1)$	2	10.5
5	$11 = 10 + 0.5(2)$	-1	10
6	$9 = 10 + 0.5(-1)$	0	9

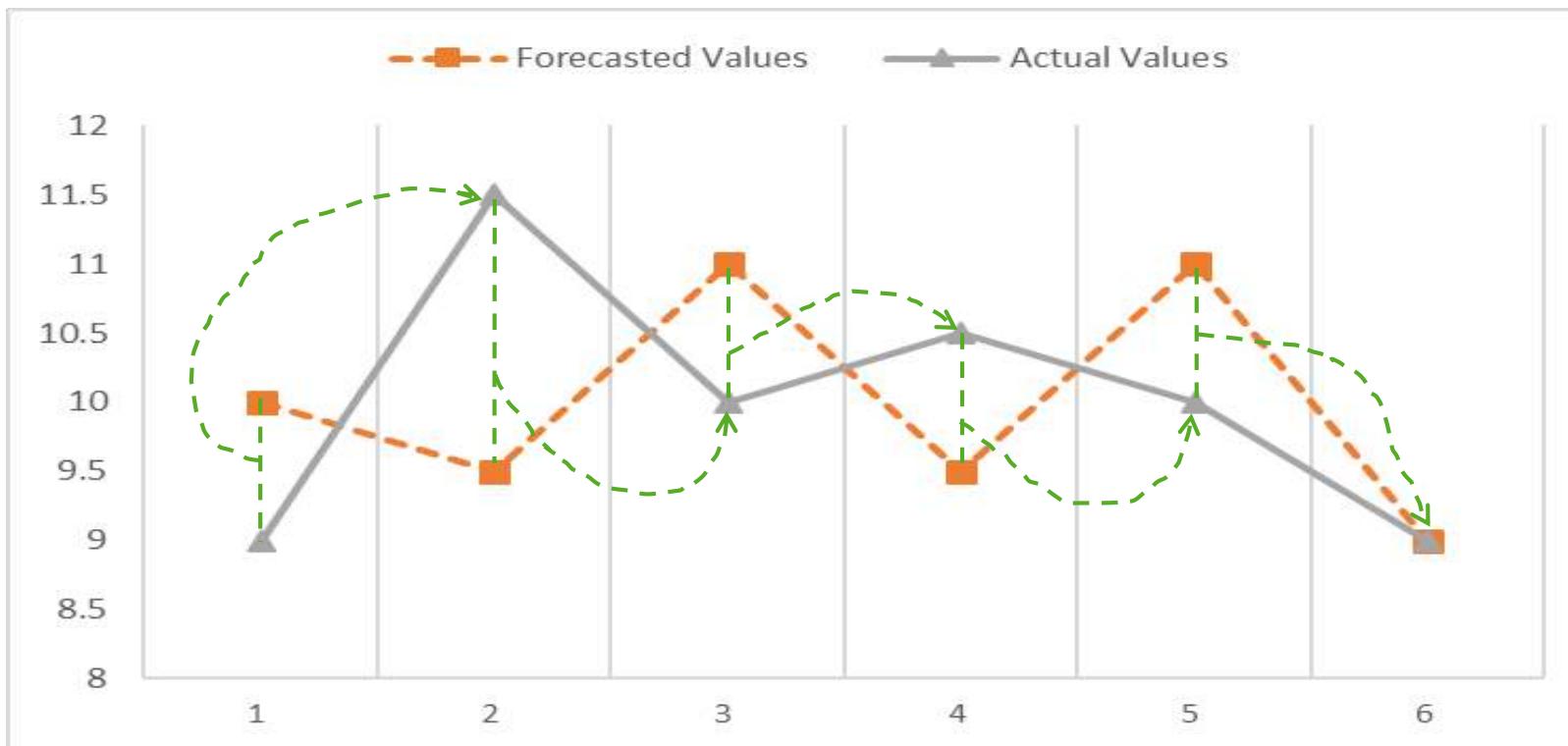
can be derived

Moving Average (MA) Models: Interpret in Reverse

An example of an MA(1) Model

$$y_t = c + \varepsilon_t + \theta_1 \varepsilon_{t-1}$$

10 0.5



ARMA: Combine Autoregressive (AR) and Moving Average (MA) Models

$$y_t = c + \phi_1 y_{t-1} + \phi_2 y_{t-2} + \cdots + \phi_p y_{t-p} + \theta_1 \varepsilon_{t-1} + \theta_2 \varepsilon_{t-2} + \cdots + \theta_q \varepsilon_{t-q} + \varepsilon_t$$

Autoregressive (AR) **Moving Average (MA)**

- Note that given $c, \phi_1, \phi_2, \dots, \phi_p, \theta_1, \theta_2, \dots, \theta_q$, it is possible to compute $\varepsilon_1, \varepsilon_2, \dots, \varepsilon_T$.
- Use for example Expectation-Maximization (EM) algorithms

ϕ is “phi”

θ is “theta”

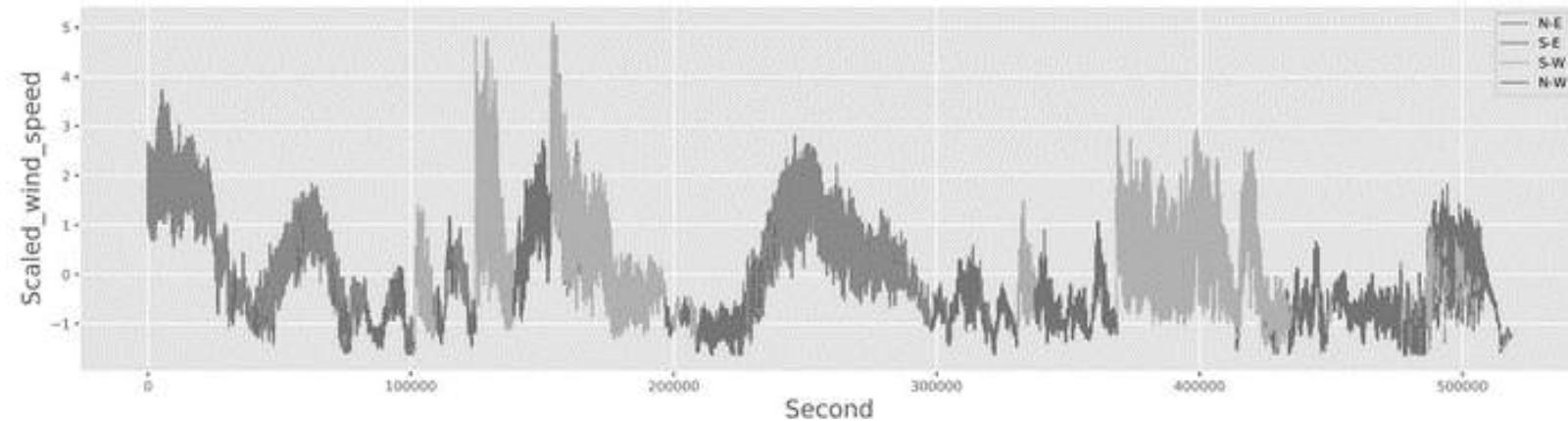
Stationarity & Differencing

The observation y_t of a **stationary time series** does not depend on the time t .

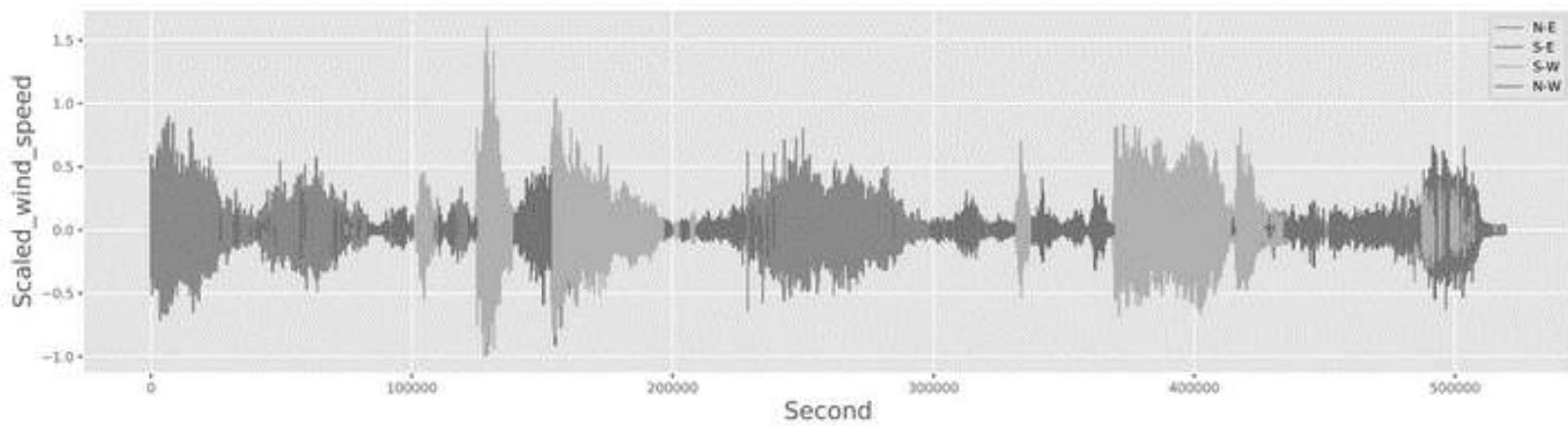
- Obviously, time series with trends or seasonality are not stationary.
- Therefore, one may use **differencing** to focus on differences rather than absolute values (**first-order differencing**)
- The new series represents the change between consecutive observations: $y'_t = y_t - y_{t-1}$. The result is a new time series and then it is “business as usual”
- Sometimes, it may be necessary to difference the series a second time (**second-order differencing**) to make it stationary $y''_t = y'_t - y'_{t-1} = (y_t - y_{t-1}) - (y_{t-1} - y_{t-2}) = y_t - 2y_{t-1} + y_{t-2}$.

Stationarity & Differencing

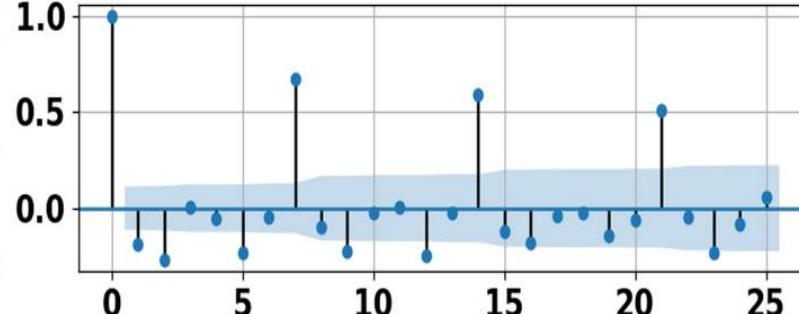
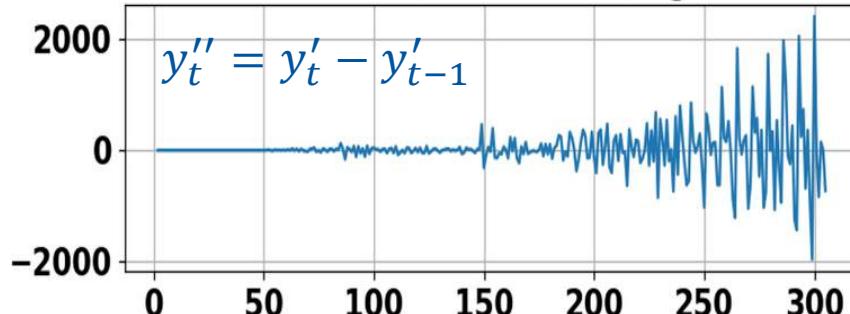
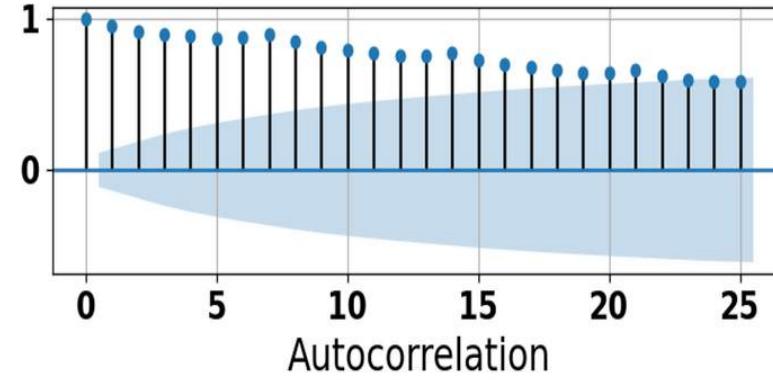
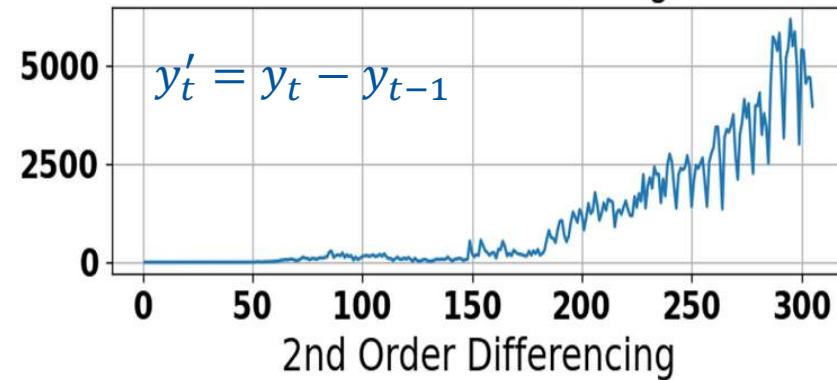
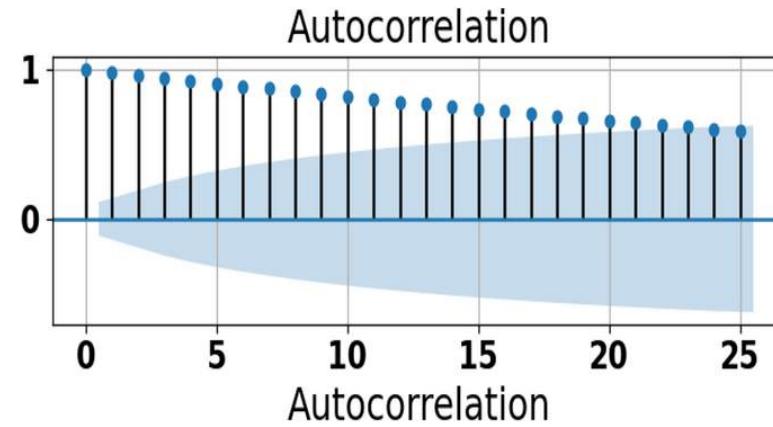
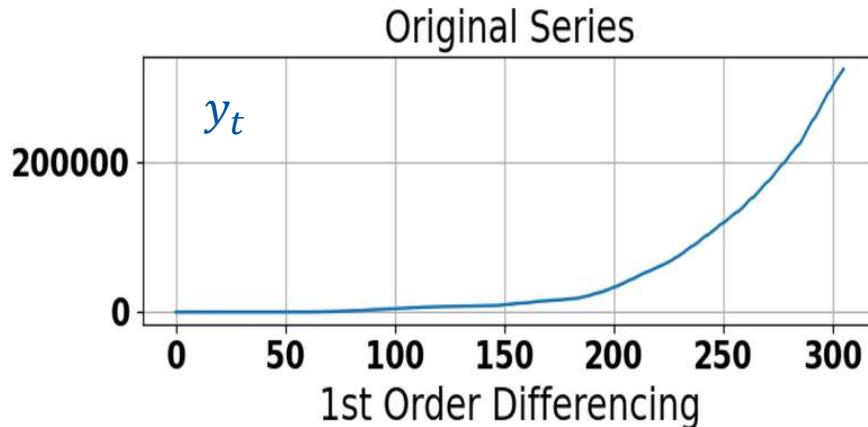
y_t



$y'_t = y_t - y_{t-1}$



Stationarity & Differencing

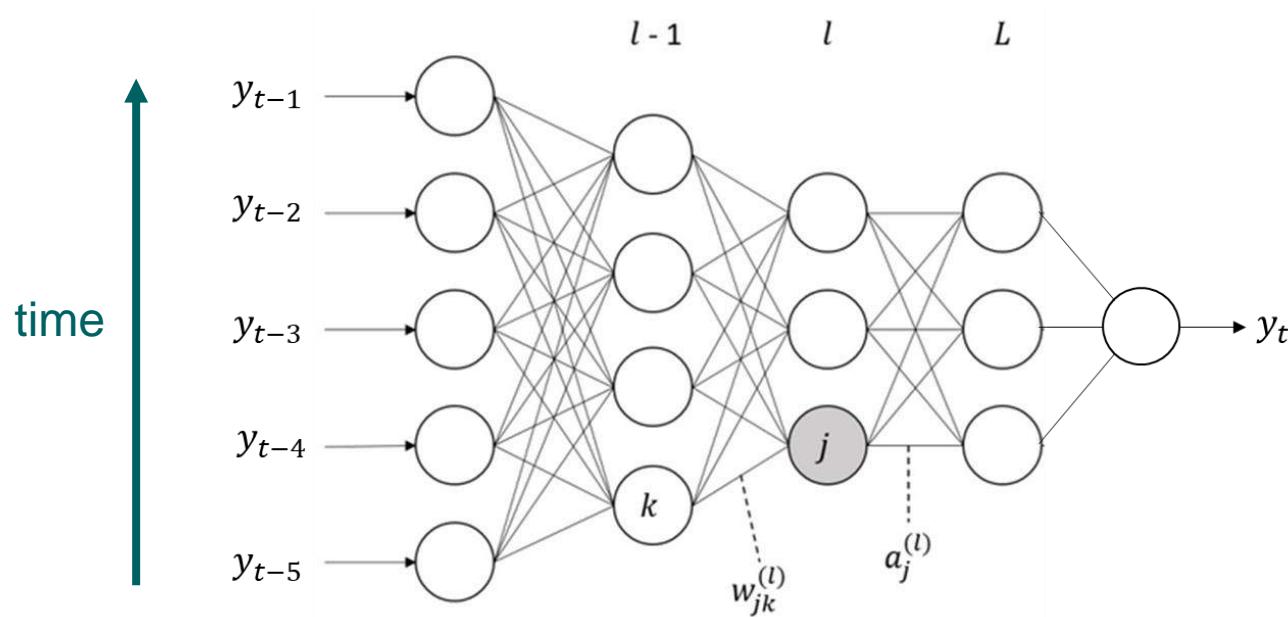


ARIMA Models

- Combination of the elements present before
 - **AR**: Autoregressive (lagged values)
 - **I**: Integrated (differencing to make series stationary)
 - **MA**: Moving Average (lagged errors)
- Hence **ARIMA** is **ARMA** with differencing
- Parameters: $ARIMA(p, d, q)$
 - p = order of the autoregressive part
 - d = number of times for differencing
 - q = order of the moving average part

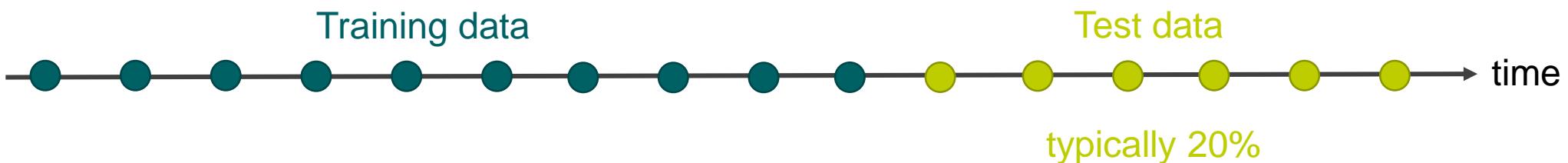
Machine Learning Models

- Feed-forward Neural Networks
- Recurrent Neural Networks, e.g, long short-term memory
- ...



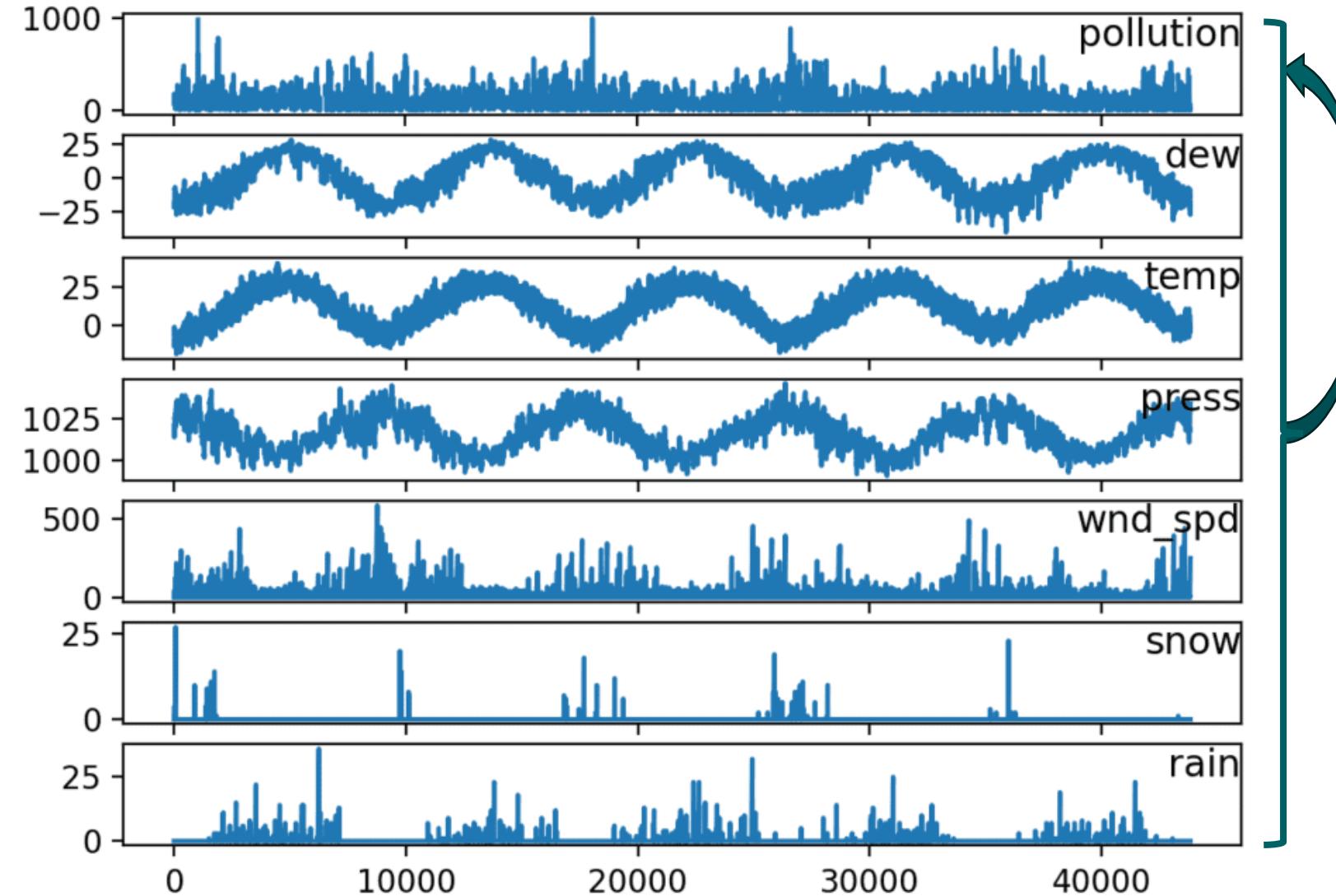
Evaluation

- A model that fits the training data well will not necessarily forecast well.
- We should split the data into training and test set.



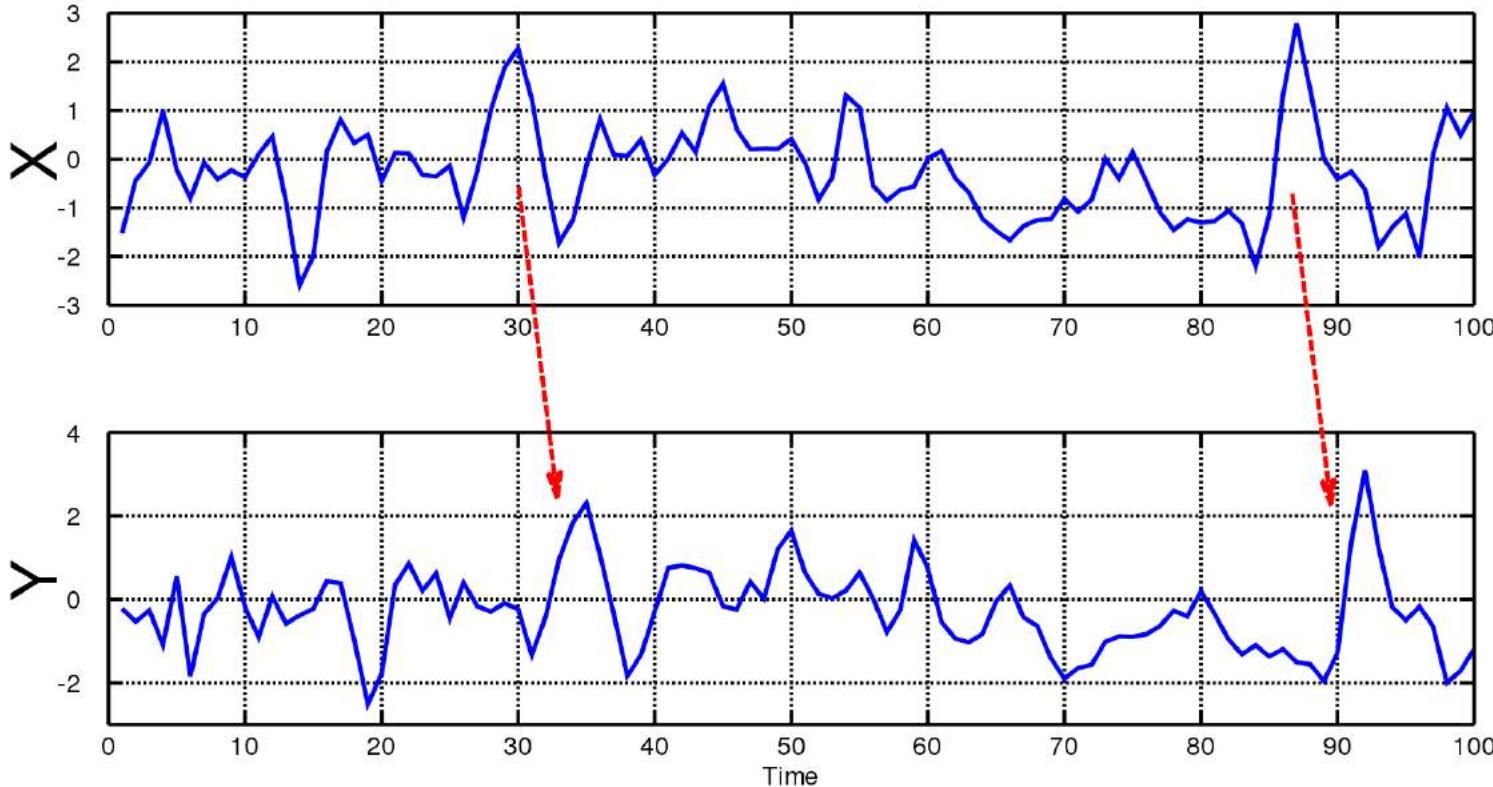
- Forecast errors: the difference between an observed value and its forecast
- Examples:
 - Mean Absolute Error (MAE): $\frac{\sum_{i=1}^n |\epsilon_i|}{n}$
 - Mean Squared Error (MSE): $\frac{\sum_{i=1}^n \epsilon_i^2}{n}$

From Univariate To Multivariate Time Series



- Next to time as a feature, multiple numerical features.
- One feature may depend on itself and other features.
- Assumption: future values do not cause current values.

Granger Causality



- The Granger causality test is a statistical hypothesis test for determining whether one time series is useful in forecasting another.
- Are predictions of the value of Y based on its own past values and on the past values of X better than predictions of Y based only on Y's own past values?
- Not really causality ...

https://en.wikipedia.org/wiki/Granger_causality

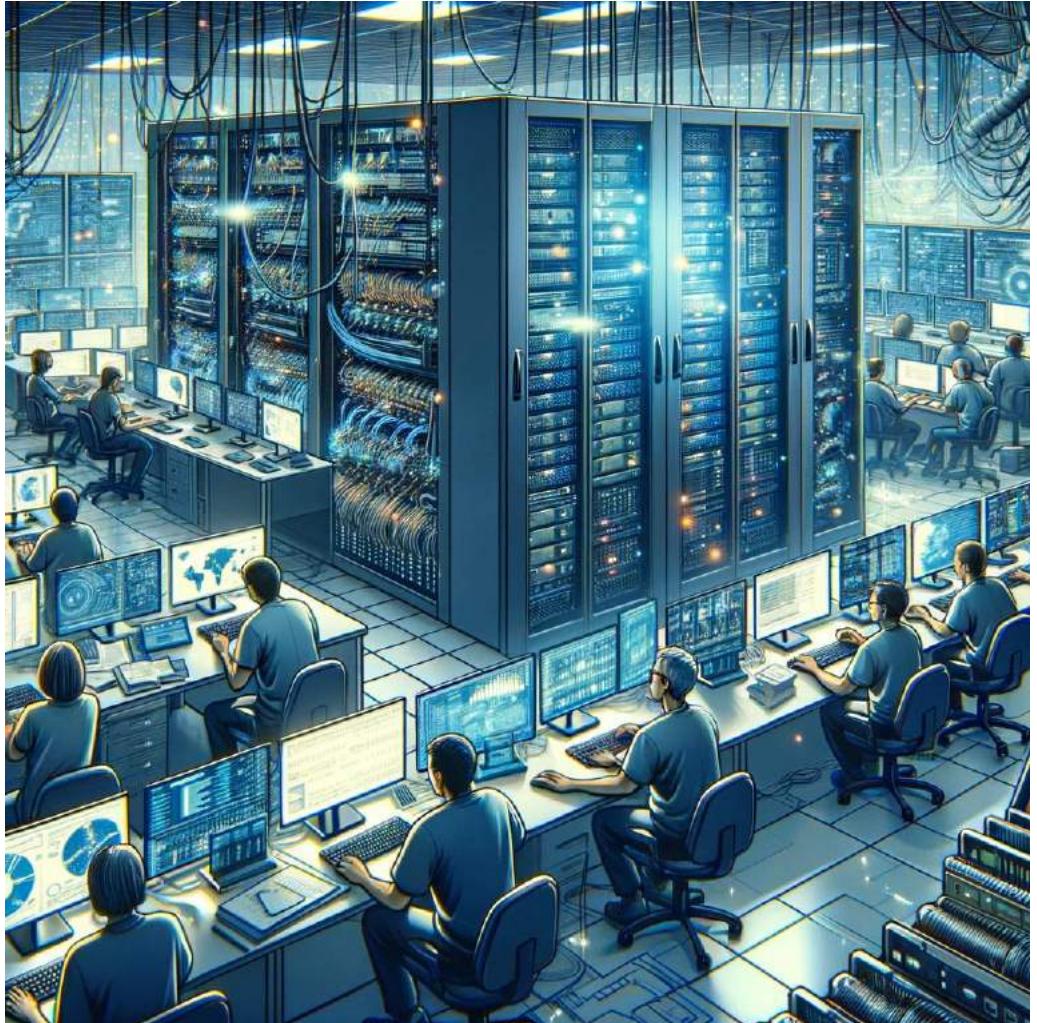
Let's Take A Step Back: How to Get the Data?



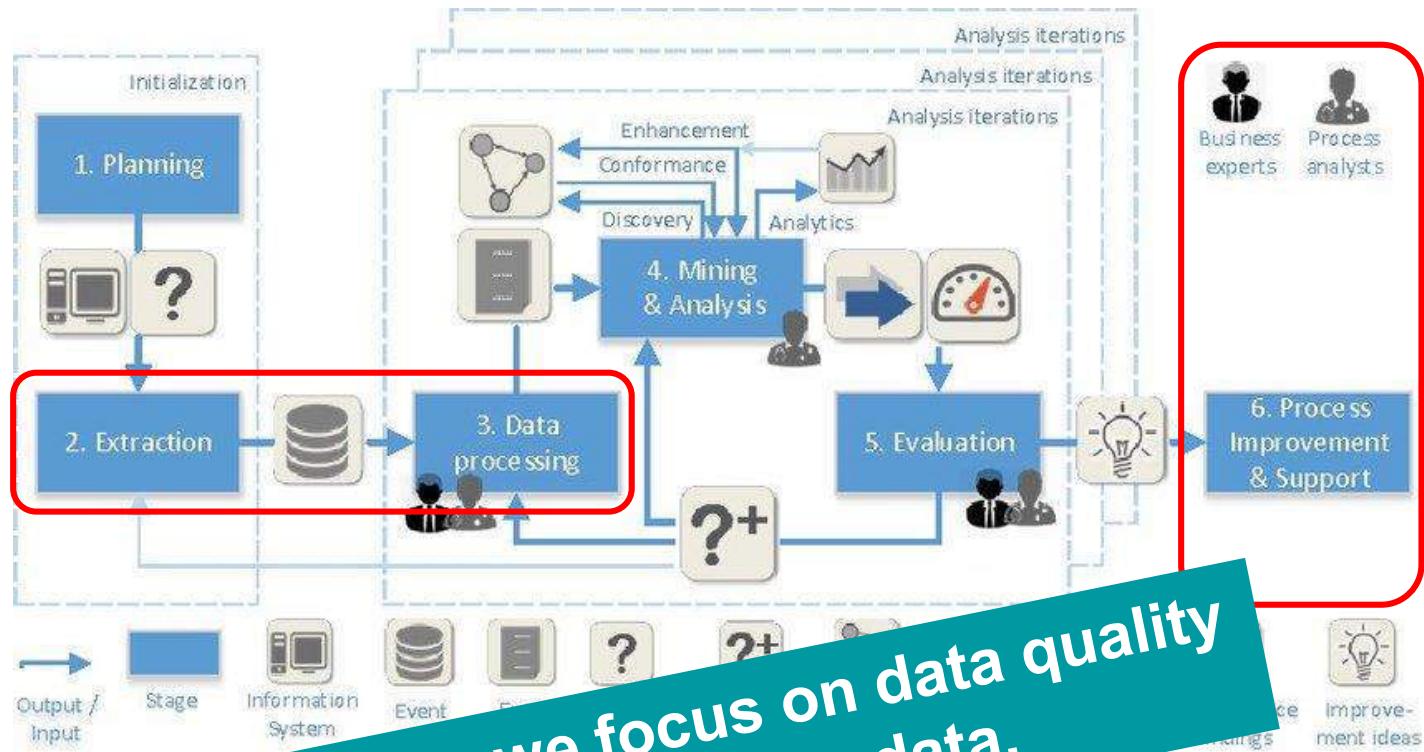
80/20

It is not uncommon that 80% of the effort/time in a data science project is devoted to finding, extracting, cleaning, and transforming the data. Only 20% is concerned with analysis.

The Two Biggest Hurdles in Practice: Getting the Data and Implementing Changes



Example: Process Mining



In the remainder, we focus on data quality and preprocessing of tabular data.

EKPO – Purchasing Document Item
#1 MANDT – Client
#2 EBELN – Purchasing Document Number
#3 EBELP – Item Number of Purchasing Document
#4 LOEKZ – Deletion indicator in purchasing document
#5 STATU – RFQ status
...
#299 POL_ID – Order List Item Number
#300 CONS_ORDER – Purchase Order for Consignment

CIO of a US bank: "We reduced the number of applications from 12.000 to 8.000" : -)

An SAP installation has hundreds of thousands of tables.

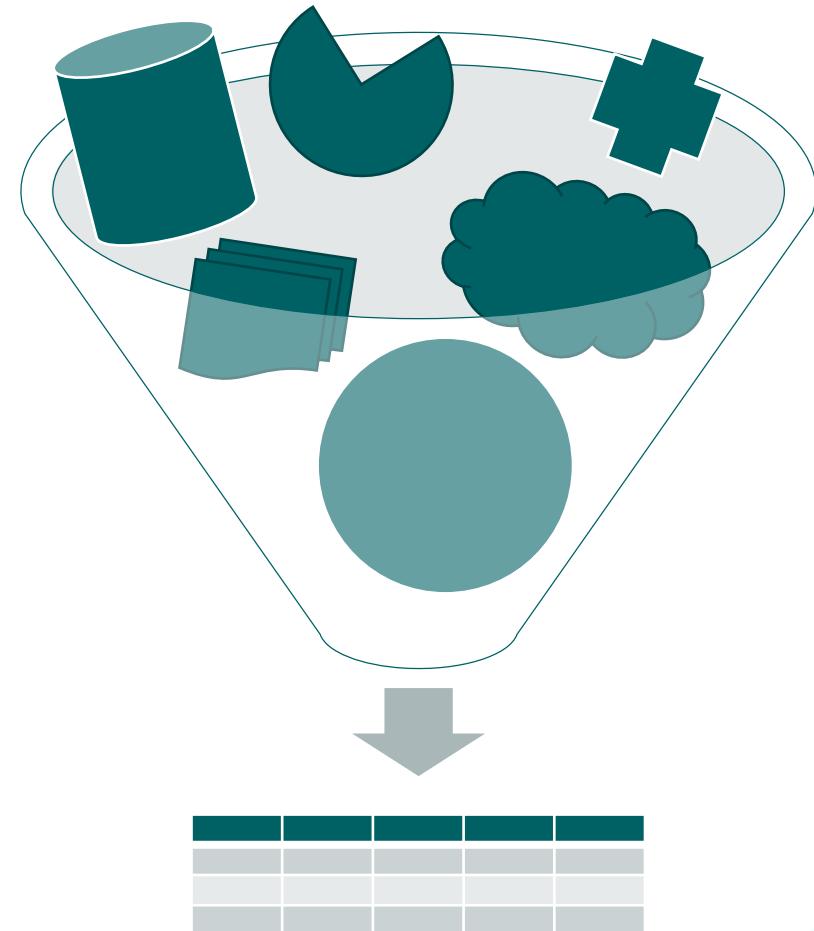
Tables may have hundreds of columns (e.g. EKPO has > 300 fields).

Organizations such as Siemens have 70 SAP installations.



Data Quality & Preprocessing

1. Introduction
2. Missing Values
3. Outliers
4. Semantic Problems
5. Transformation & Normalization
6. Data Reduction
7. Conclusion



Data Science Pipeline

- Garbage in, garbage out
- Possible **problems** (big data, security), **errors** (data quality), **biases** (e.g., survivorship bias) everywhere
- Problems, errors and biases **propagate**

Goal: increase data quality and modify the data to suit the analysis question and applied techniques



Data Quality Aspects

- Accuracy
- Completeness
- Consistency
- Timeliness
- Believability
- Interpretability

- Consider when setting up databases etc.
- Use to gain overview of quality of provided data

Name	Age	Siblings	Date of Admission
Sara Johnson	55	0	30.09.2022
NAME	17		23-11-22
Smith	28	2	8/24/22
Emma Miller	2	56	May 10 th , 22
Jones	87	3	220701
...	

Example Data

Data Quality Aspects

- Accuracy
- Completeness
- Consistency
- Timeliness
- Believability
- Interpretability

Are the values correct? Is it possible to identify errors in the data?

Name	Age	Siblings	Date of Admission
Sara Johnson	55	0	30.09.2022
NAME	17		23-11-22
Smith	28	2	8/24/22
Emma Miller	2	56	May 10 th , 22
Jones	87	3	220701
...	

Emma seems to have an improbable number of siblings. Was the value entered incorrectly?

Data Quality Aspects

- Accuracy
- **Completeness**
- Consistency
- Timeliness
- Believability
- Interpretability

Are values missing?
Is there 'disguised' missing data?
(e.g., default or pre-selected values)

Name	Age	Siblings	Date of Admission
Sara Johnson	55	0	30.09.2022
NAME	17		23-11-22
Smith	28	2	8/24/22
Emma Miller	2	56	May 10 th , 22
Jones	87	3	220701
...	

NAME is the default placeholder where the name should have been entered.

Data Quality Aspects

- Accuracy
- Completeness
- **Consistency**
- Timeliness
- Believability
- Interpretability

Does the data adhere to common naming conventions and formats?

Are these conventions and formats used consistently throughout the data?

Name	Age	Siblings	Date of Admission
Sara Johnson	55	0	30.09.2022
NAME	17		23-11-22
Smith	28	2	8/24/22
Emma Miller	2	56	May 10 th , 22
Jones	87	3	220701
...	

The first name is not always included. The admission date format varies.

Data Quality Aspects

- Accuracy
- Completeness
- Consistency
- **Timeliness**
- Believability
- Interpretability

Data may be missing for some time periods
(aging, lost updates, etc.).

Some values may be up-to-date while others are outdated.

Name	Age	Siblings	Date of Admission
Sara Johnson	55	0	30.09.2022
NAME	17		23-11-22
Smith	28	2	8/24/22
Emma Miller	2	56	May 10 th , 22
Jones	87	3	220701
...	

People gain years (i.e., celebrate birthdays) all year, but age is updated only occasionally.

Data Quality Aspects

- Accuracy
- Completeness
- Consistency
- Timeliness
- **Believability**
- Interpretability

Does the user trust the data, i.e., does the user believe the data to be true, real, credible?

Depends on the data source and processing history.

Name	Age	Siblings	Date of Admission
Sara Johnson	55	0	30.09.2022
NAME	17		23-11-22
Smith	28	2	8/24/22
Emma Miller	2	56	May 10 th , 22
Jones	87	3	220701
...	

Previous errors and inconsistencies have decreased the trust in the system.

Data Quality Aspects

- Accuracy
- Completeness
- Consistency
- Timeliness
- Believability
- **Interpretability**

Are the data understandable without much explanation?

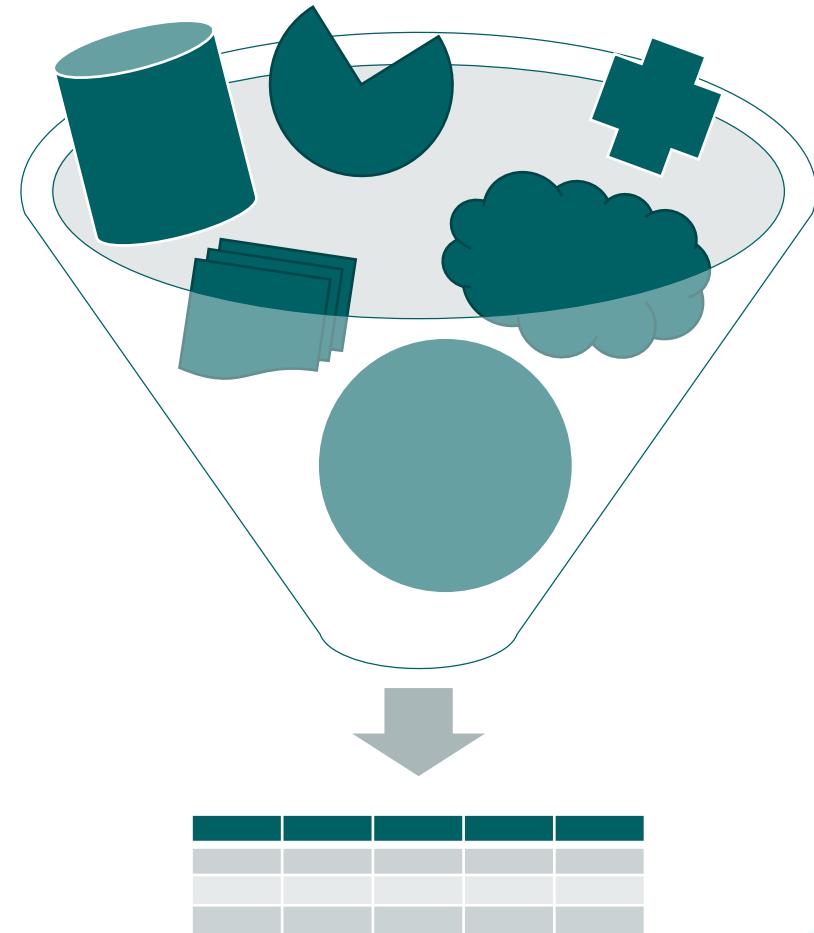
Does it leave room for ambiguity?

Name	Age	Siblings	Date of Admission
Sara Johnson	55	0	30.09.2022
NAME	17		23-11-22
Smith	28	2	8/24/22
Emma Miller	2	56	May 10 th , 22
Jones	87	3	220701
...	

Is the age given in years or in months? Does the number of siblings include half-siblings?

Data Quality & Preprocessing

1. Introduction
2. **Missing Values**
3. Outliers
4. Semantic Problems
5. Transformation & Normalization
6. Data Reduction
7. Conclusion



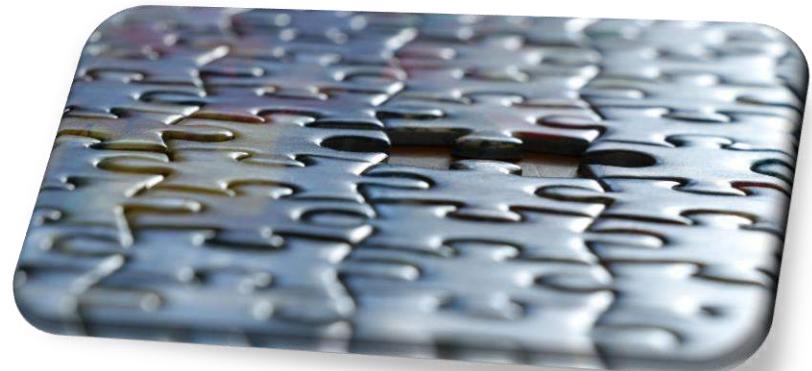
Detecting Missing Values

Missing values may be obvious...

- Empty value
- NaN / NA

... or may be disguised!

- Default value
- Invalid value



Handling Missing Values

- 1) Fill in manually
- 2) Ignore
- 3) Fill in a derived value



1



3



2

Handling Missing Values: Ignore



Discard the feature

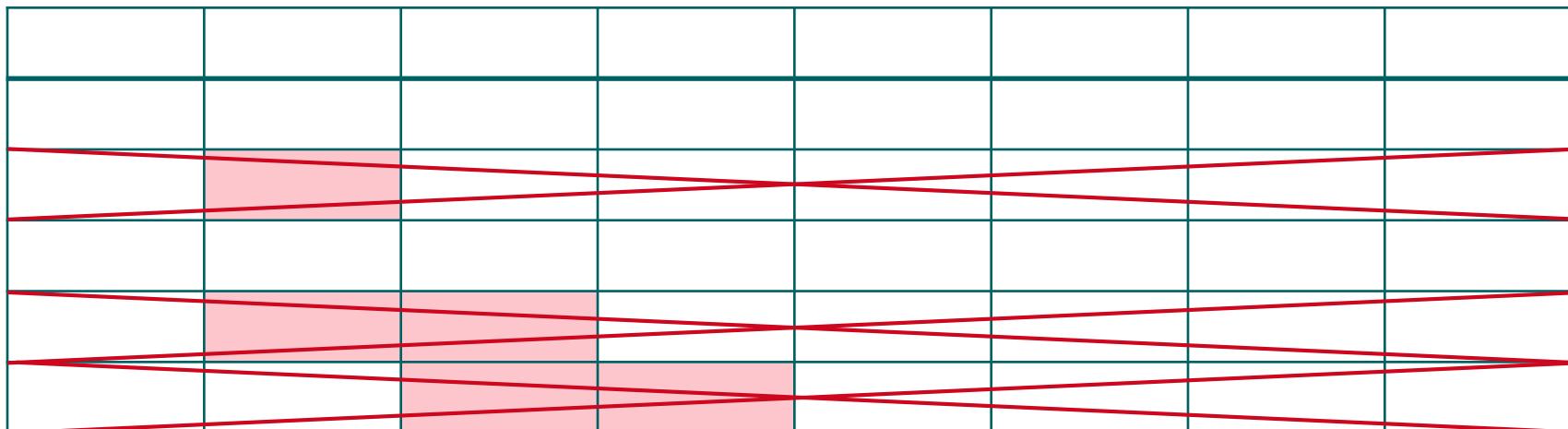
- The whole feature is removed from the data
 - Usually done if the number of missing values is too large to allow meaningful analysis
(as a rule of thumb, if more than 60 % of the feature values are missing)

A 5x5 grid with red lines forming an X. The intersections at (1,1), (2,3), (3,1), and (3,5) are shaded pink.

Handling Missing Values : Ignore

Discard the instance

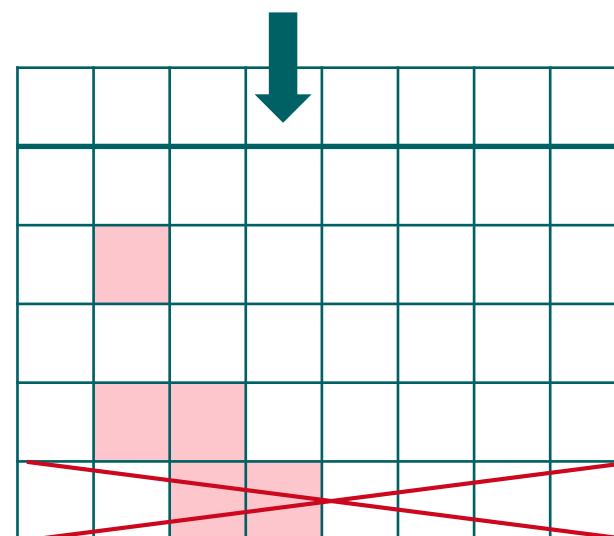
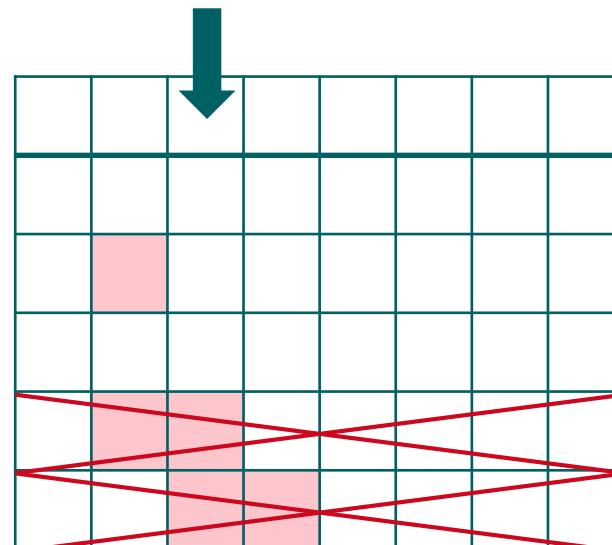
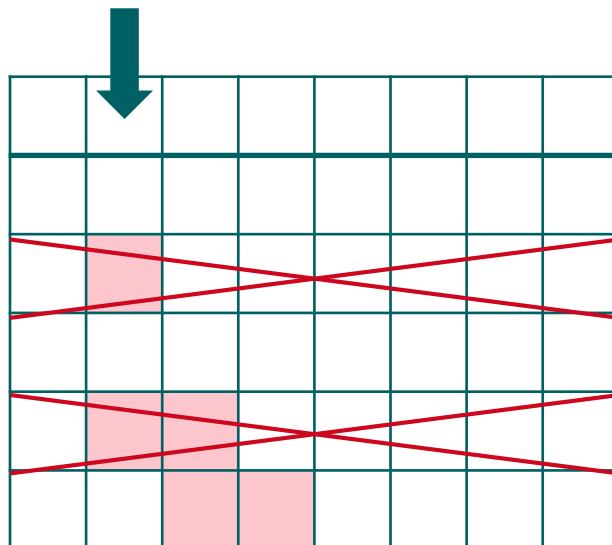
- The entire instance is simply **discarded**
- Usually done when the whole instance becomes unusable (e.g., labeling attribute for classification is missing)
- If the data set misses a lot of values, this technique may make the whole data set **unusable** or introduce a **bias**



Handling Missing Values: Ignore

Ignore the instance only for features where the value is missing

- The instance is ignored when analyzing features where it misses a value
- Information for other features remains usable



Handling Missing Values: Create



Mean/median/mode of the whole feature

- Compute mean/median/mode and fill the gaps accordingly
- May introduce values far away from the real value
- Example: compute yearly income

Handling Missing Values: Create

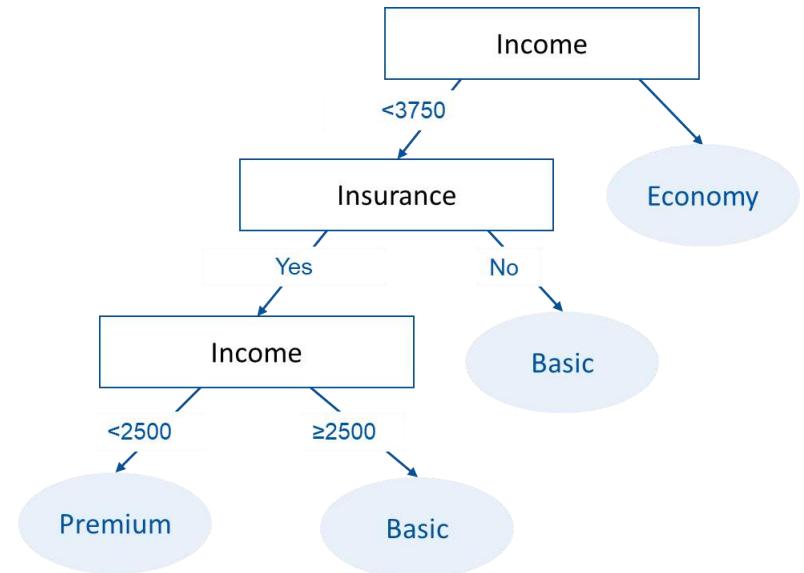
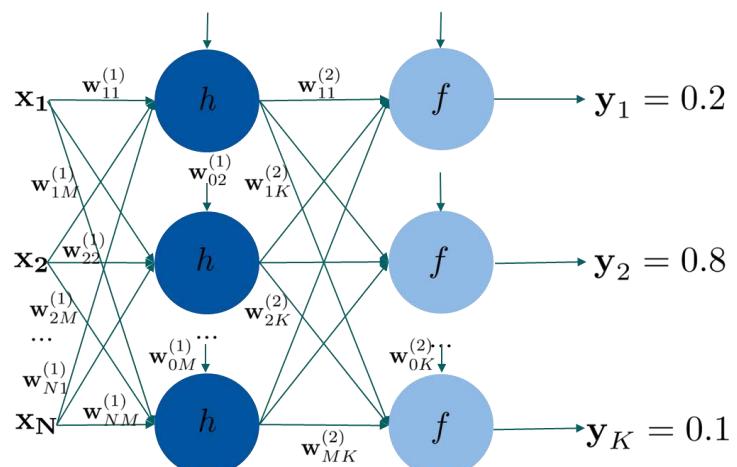
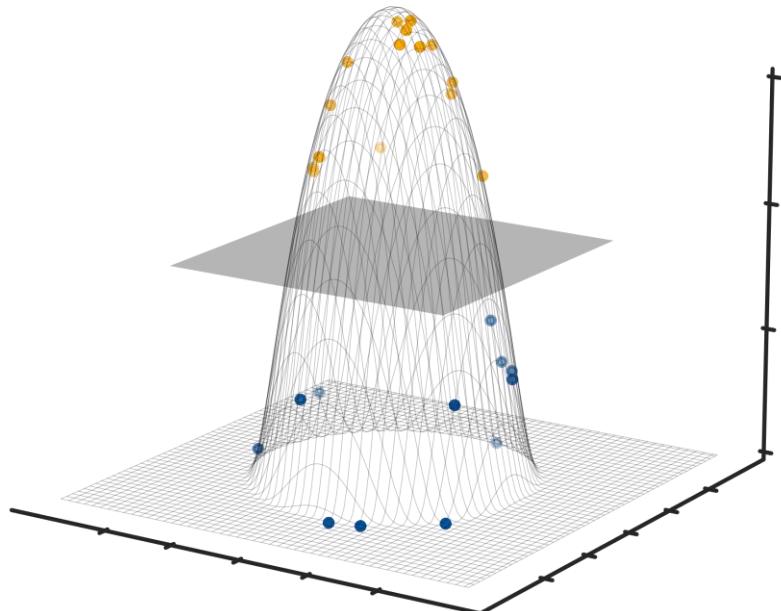
Mean/median/mode of all instance belonging to the same class

- Compute mean/median/mode only based on instances with the same class label
 - Higher chances to be accurate compared to the overall mean/median/mode
 - Only valuable if we have meaningful groups in the data
-
- Example: compute income for a 20-year-old Student living in Aachen, Germany

Handling Missing Values: Create

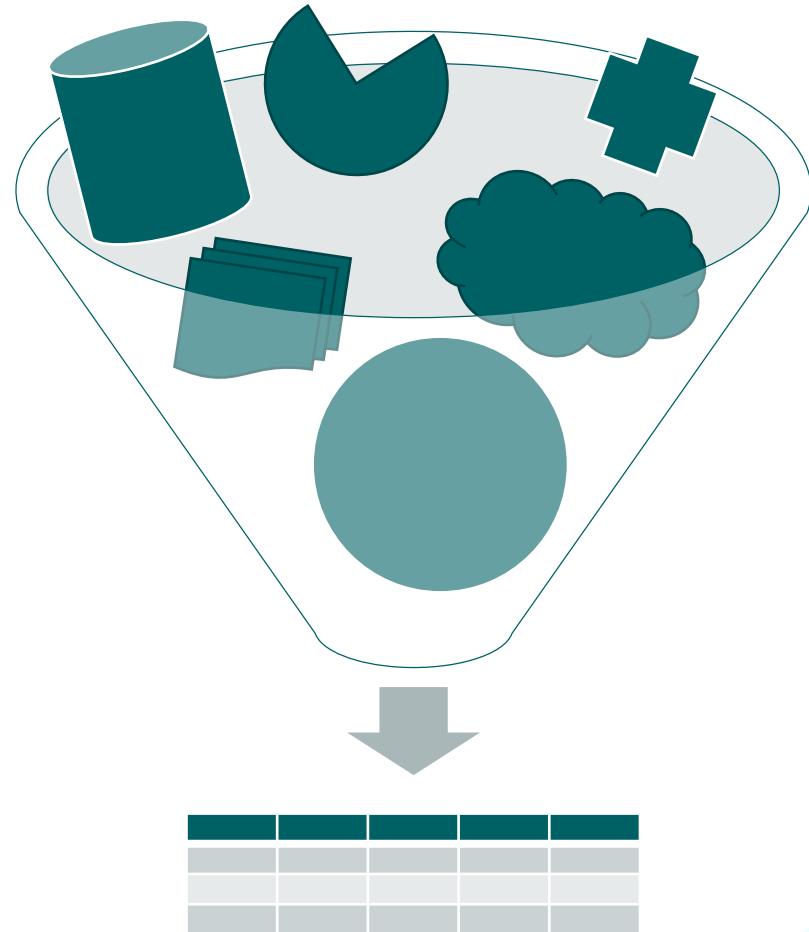
Complex derived value (use a predictor model)

- Fill in the value given by a suitable prediction model
- E.g., decision trees, regression, NNs, SVMs...

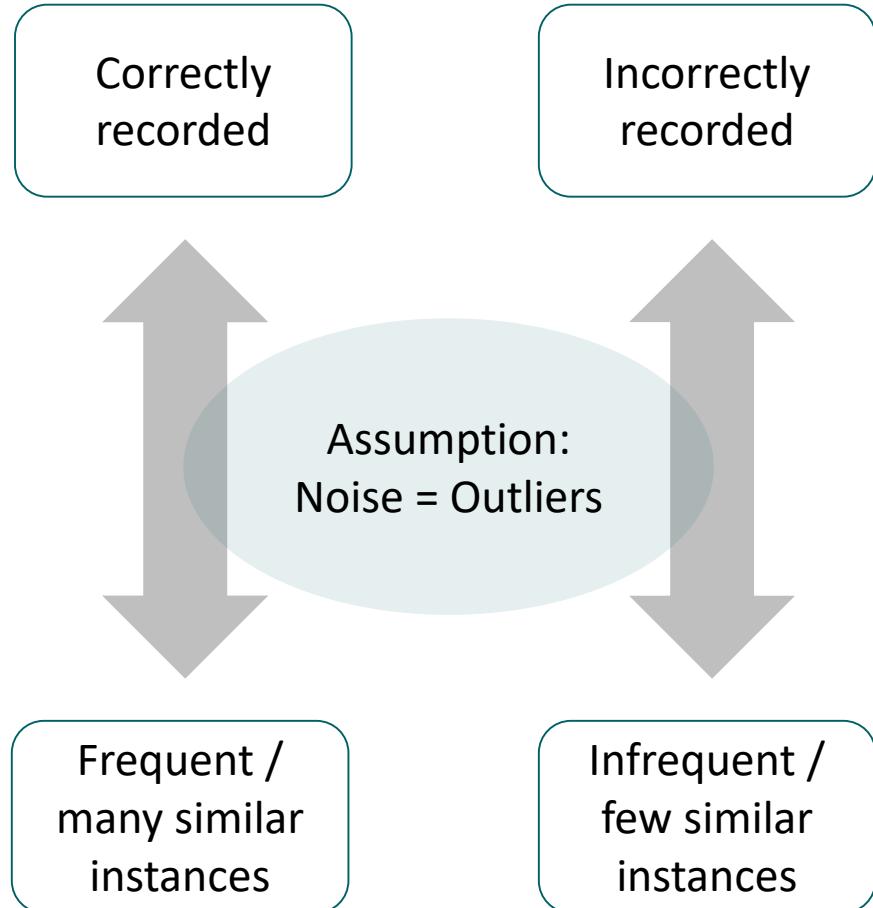


Data Quality & Preprocessing

1. Introduction
2. Missing Values
- 3. Outliers**
4. Semantic Problems
5. Transformation & Normalization
6. Data Reduction
7. Conclusion



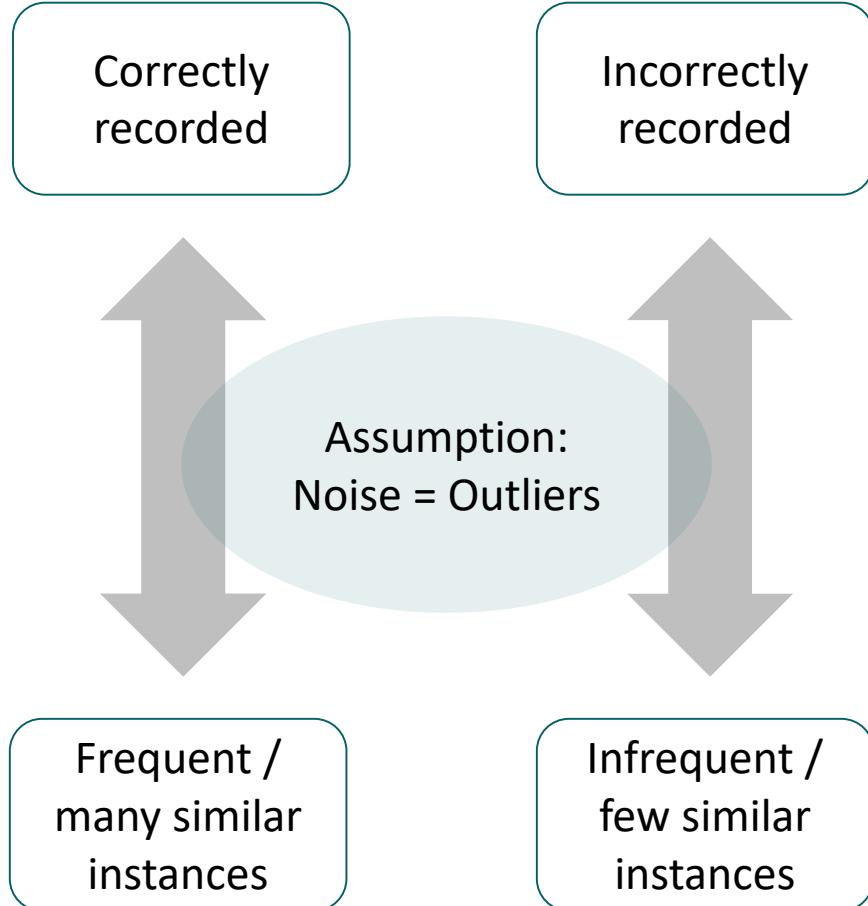
Introduction



What is noise?

- We assume that noise causes outliers
- Thus, **outliers** indicate noise

Outlier Detection

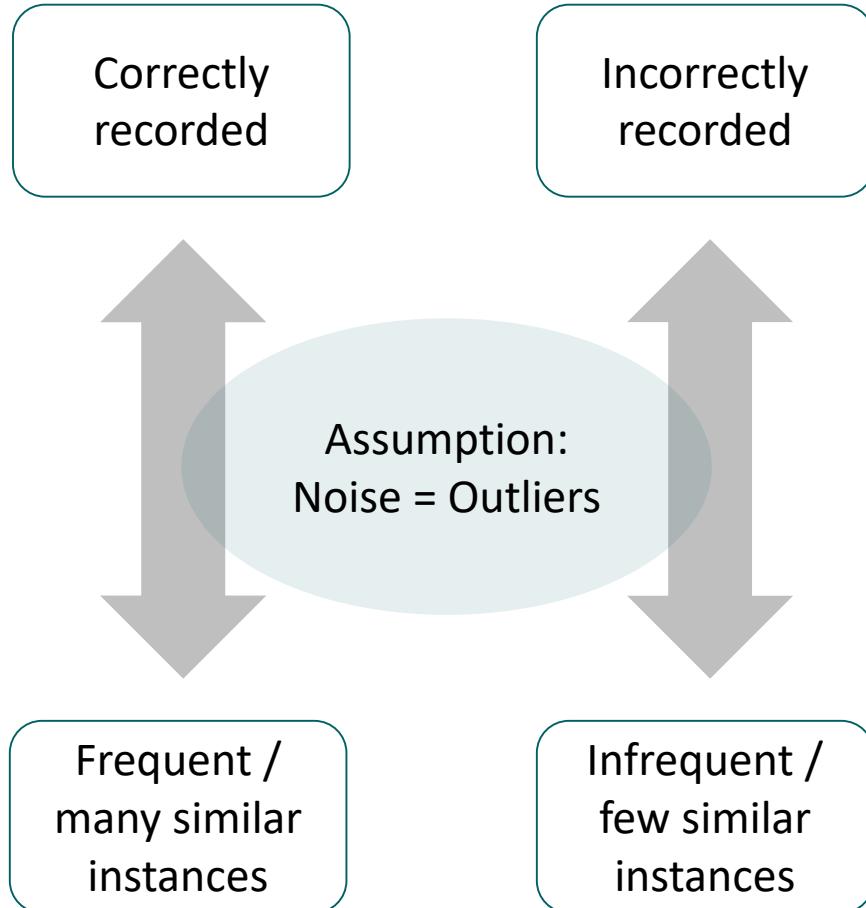


How to detect outliers?

- Boxplots
- Decision trees
- Regression
- SVMs
- Clustering
- ...

→ Predictor models can be used to **define** outliers

Outlier Handling



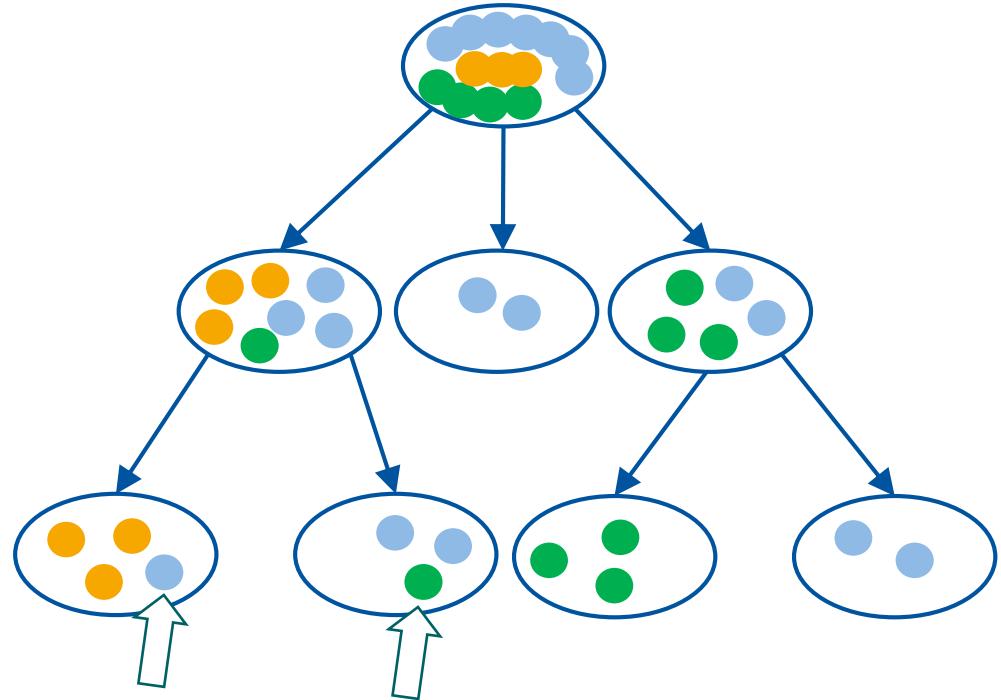
How to handle outliers?

Outliers can be [handled as missing values](#):

- Fill in a correct value manually
- Ignore the feature/instance
- Replace with a derived value

→ Predictor models can be used to [replace outliers](#)

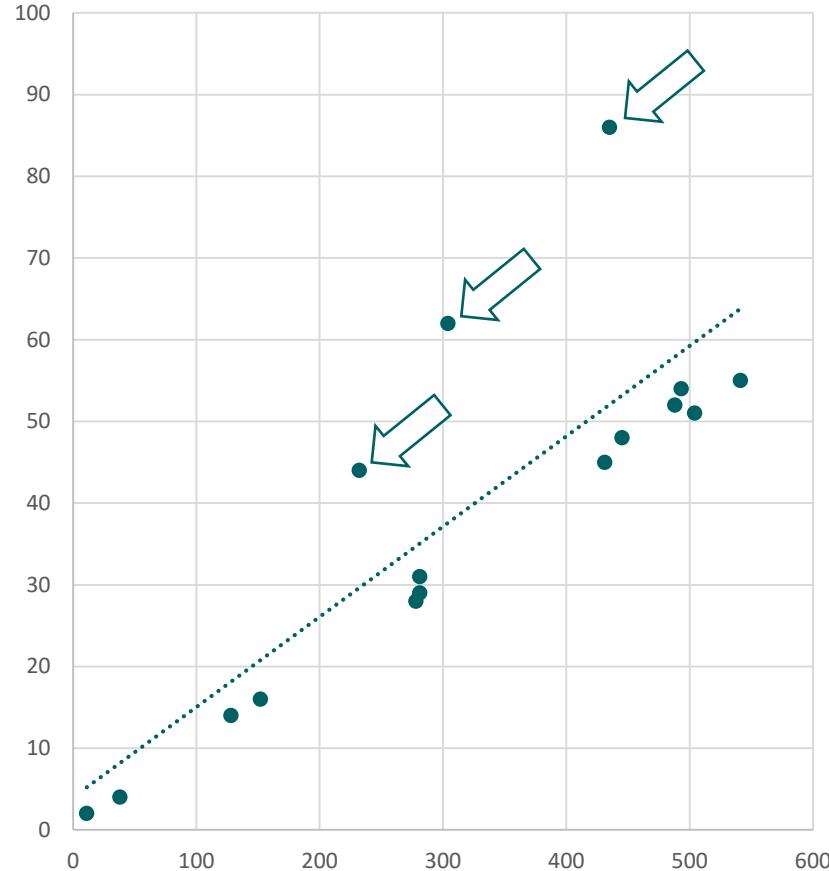
Outlier Detection - Decision Trees



How to detect outliers?

- Every leaf node is assigned a class label
- Instances in that leaf node with a **non-matching class label** can be considered outliers

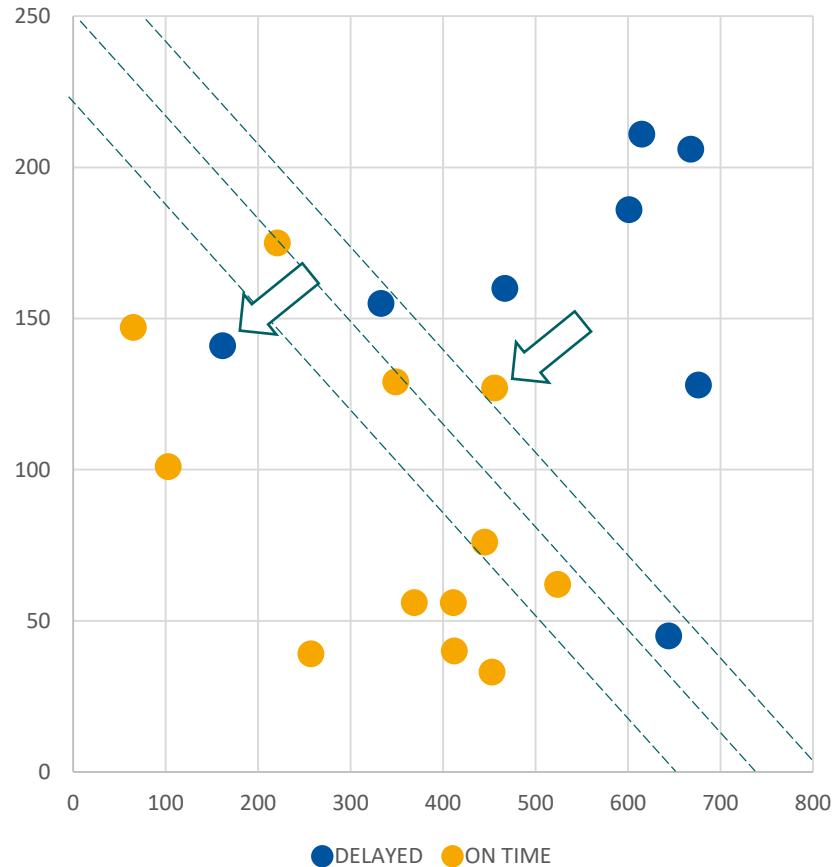
Outlier Detection - Regression



How to detect outliers?

- Instances which are **far away** from the predicted value are considered outliers
- The definition of 'far away' depends on an **error function** and **threshold**

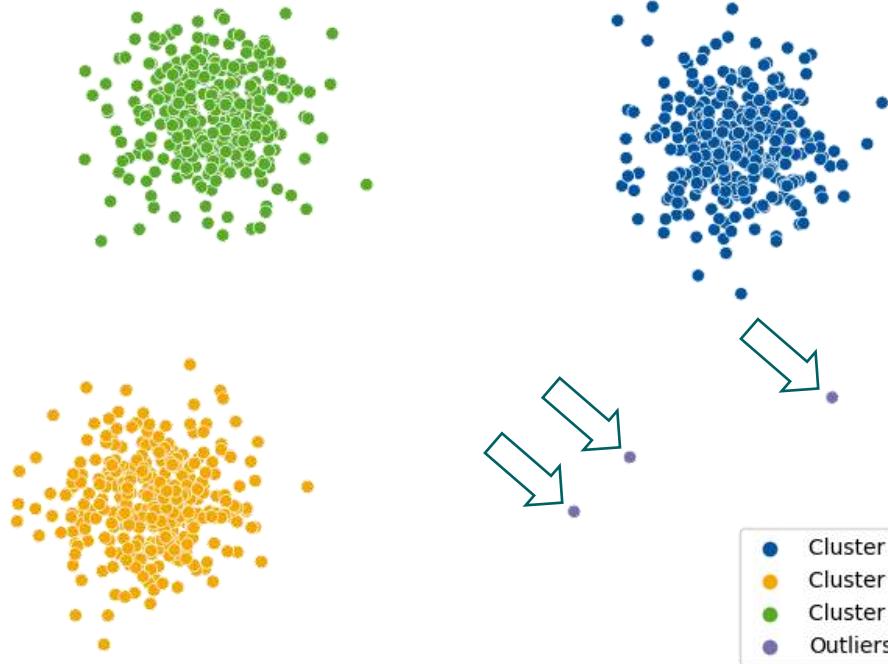
Outlier Detection – SVM



How to detect outliers?

- Instances which are (too far) on the [wrong side](#) of the hyperplane are considered outliers
- Soft margin may be used to define how far

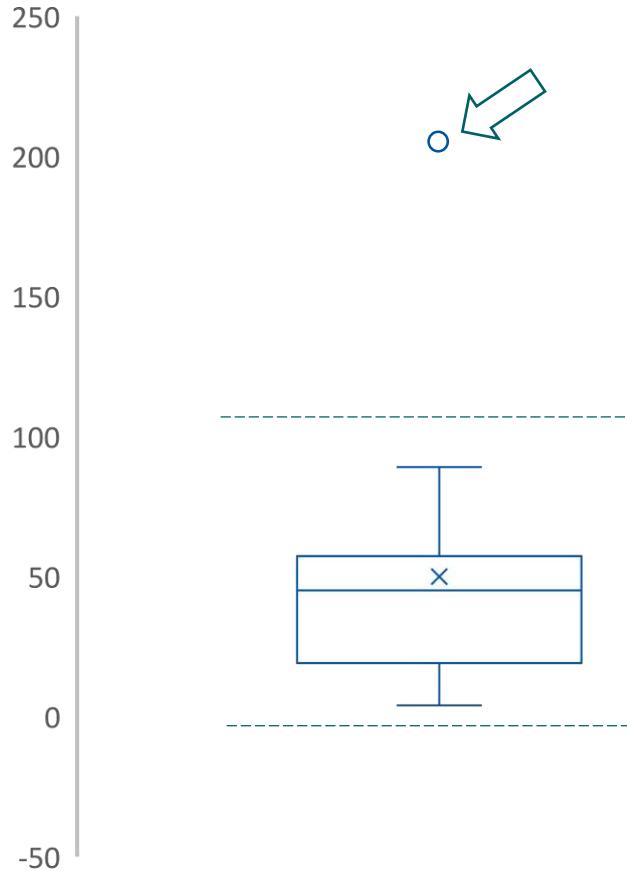
Outlier Detection - Clustering



How to detect outliers?

- Instances **outside of any cluster** can be considered outliers

Outlier Detection - Boxplots



How to detect outliers?

- Instances **above** the upper fence
- Instances **below** the lower fence

→ Outlier handling option:
Clamp values to the nearest fence

Outlier Handling



[1]

How to handle outliers?

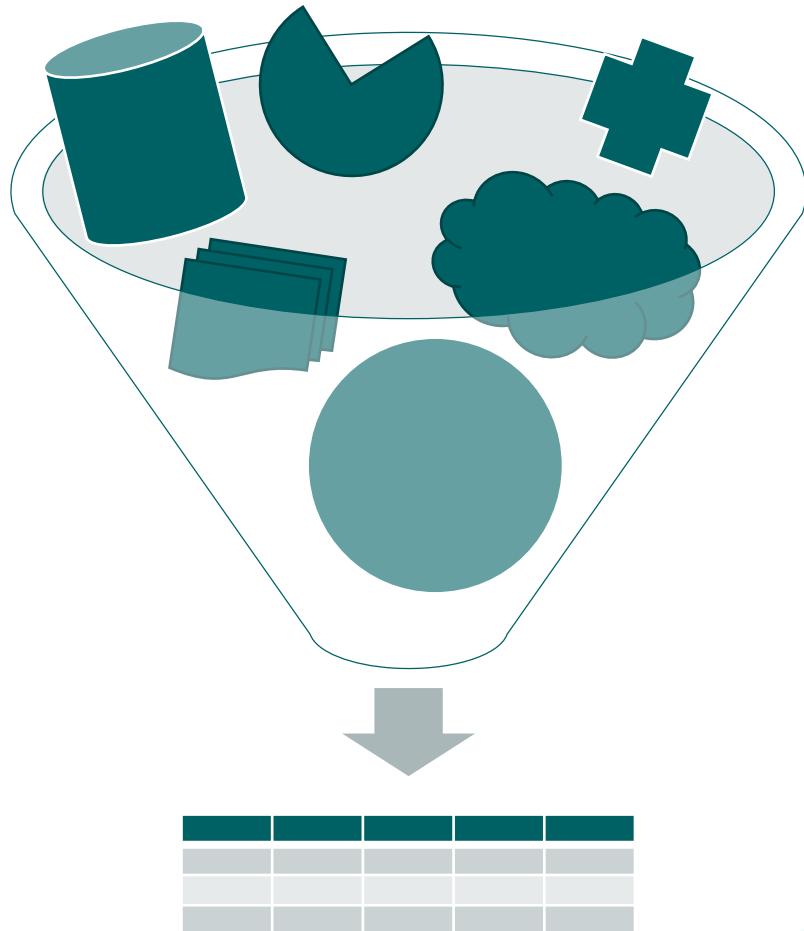
Outliers can be [handled as missing values](#):

- 1) Fill in a correct value manually
- 2) Ignore the feature/instance
- 3) Replace with a derived value

Again: the appropriate method depends on the data and purpose

Data Quality & Preprocessing

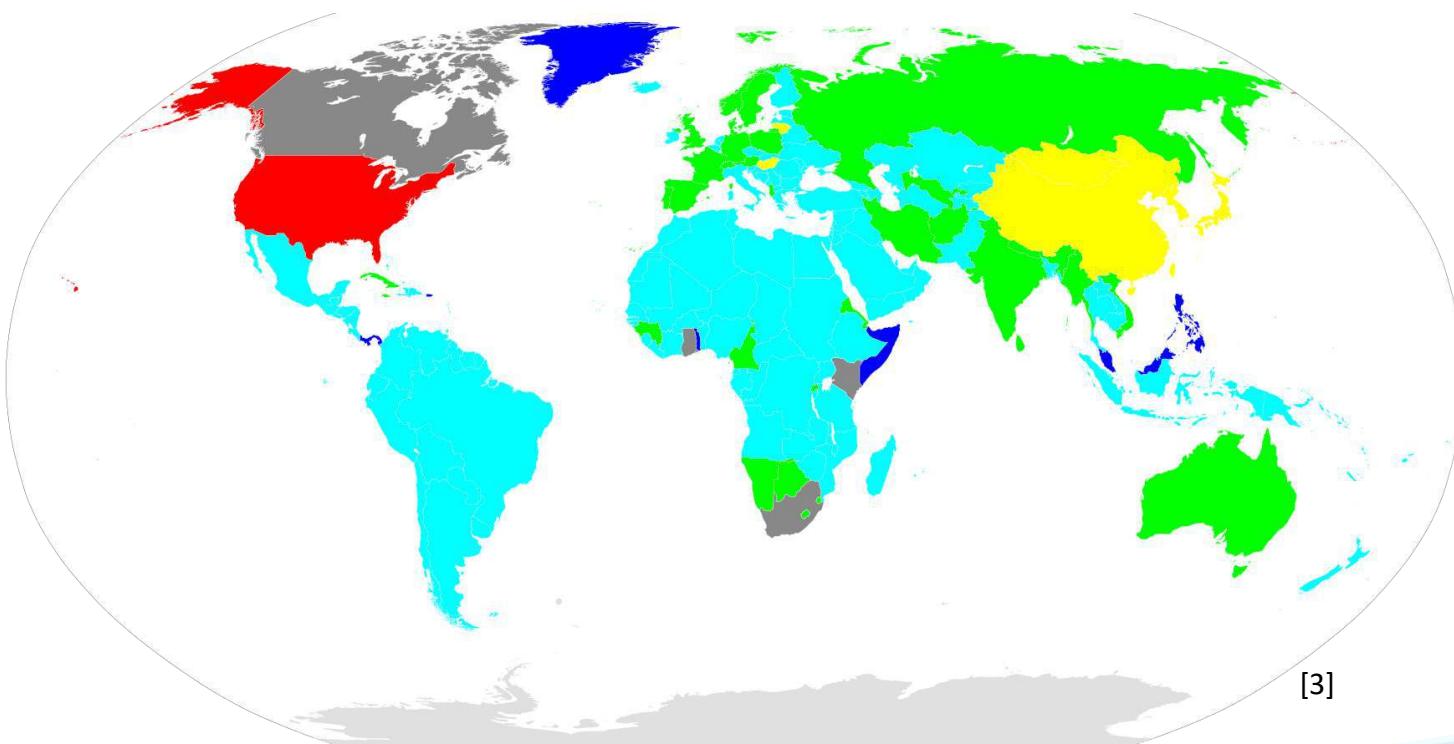
1. Introduction
2. Missing Values
3. Outliers
- 4. Semantic Problems**
5. Transformation & Normalization
6. Data Reduction
7. Conclusion



Semantic Problems - Examples

Data integration

- Merging systems
- Merging data sources



[3]

Region	Common Format
--------	---------------

	dd-mm-yyyy mm-dd-yyyy
	dd-mm-yyyy yyyy-mm-dd
	mm-dd-yyyy yyyy-mm-dd dd-mm-yyyy
	yyyy-mm-dd
	mm-dd-yyyy yyyy-mm-dd
	dd-mm-yyyy

06/07/2023 vs 07/06/2023

Data Integration

Data is collected by different systems and stored separately → merging can introduce problems

Redundancies:

- multiple entries referring to the same instance (often caused by maintenance inconsistencies)
- e.g. two entries for the same purchase but with different addresses (duplication instead of update)
- ‘Wil van der Aalst’, ‘van der Aalst, Wil’, ‘W. Aalst’, ‘Willibrordus Van’, ‘W.M.P.’…

Name	Age	Date of Admission
Sara Johnson	55	30.09.2022
Sara Johnson	56	30.09.2022
Bob Smith	28	8/24/22
...	...	

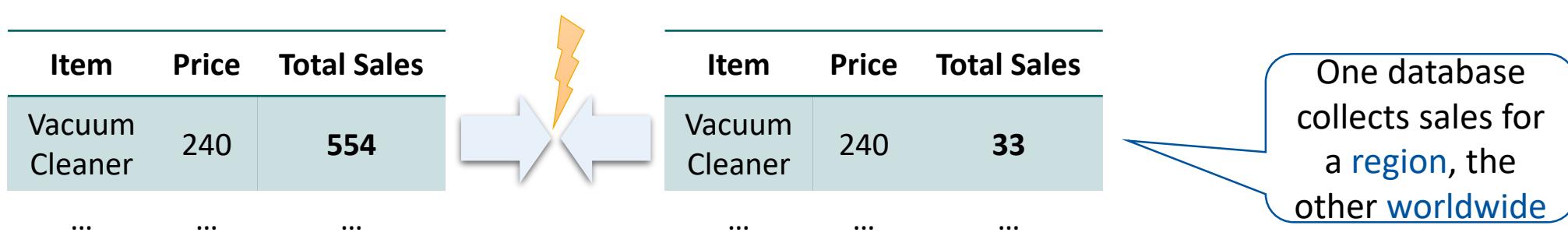
Different instances
or duplication
instead of update?

Data Integration

Data is collected by different systems and stored separately → merging can introduce problems

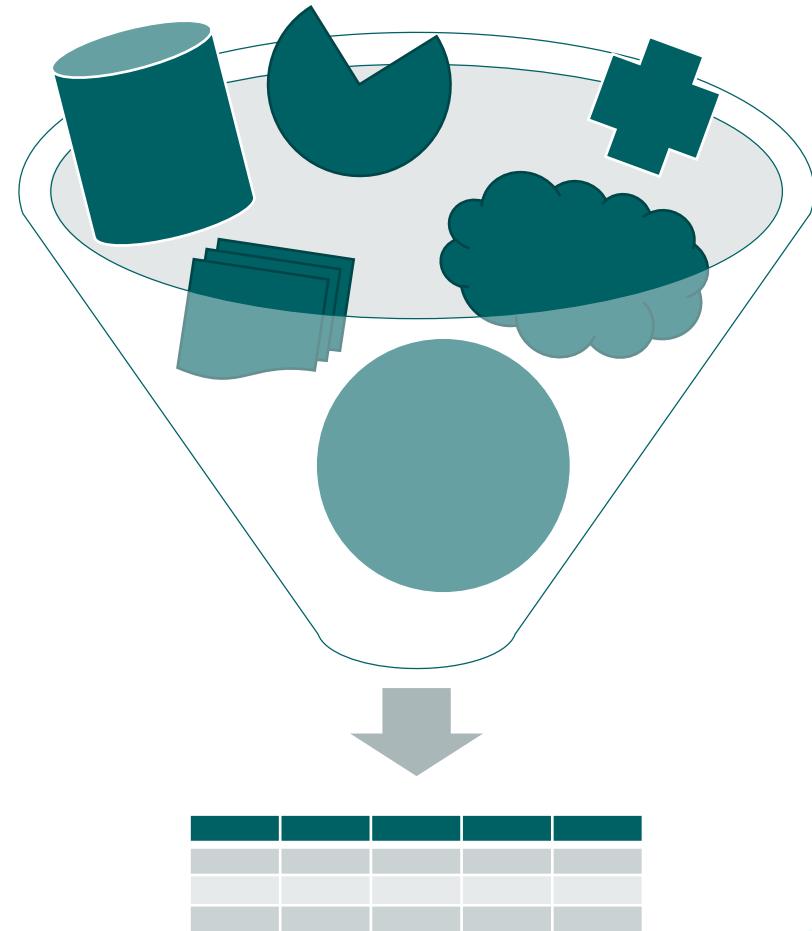
Inconsistencies & Semantic problems:

- data sources using different scales, scopes, encoding, representation, abstraction levels ...
- e.g., prices use different currencies, and may be represented with or without VAT
- e.g., 'total sales' might refer to company wide sales, or to one specific region/country



Data Quality & Preprocessing

1. Introduction
2. Missing Values
3. Outliers
4. Semantic Problems
- 5. Transformation & Normalization**
6. Data Reduction
7. Conclusion



Preprocessing – Preparing the Data for Analysis

- Transformation: change the data to the right data type
- Normalization: adjust the influence of features
- Reduction: make the data smaller for analysis



[1]

Preprocessing – Transformation

- One-hot encoding: categorical to numerical
- Binning: numerical to categorical

The diagram illustrates the transformation of a dataset from raw features to one-hot encoded features. On the left, a table shows raw data with columns f_1 , f_2 , and class. The f_1 column has categories high, high, medium, low, high, and medium. The f_2 column has values 88, 76, 32, 89, 21, and 45. The class column has values A, B, B, C, C, and A. A blue arrow points from this table to a second table on the right, which represents the one-hot encoding of the f_1 column.

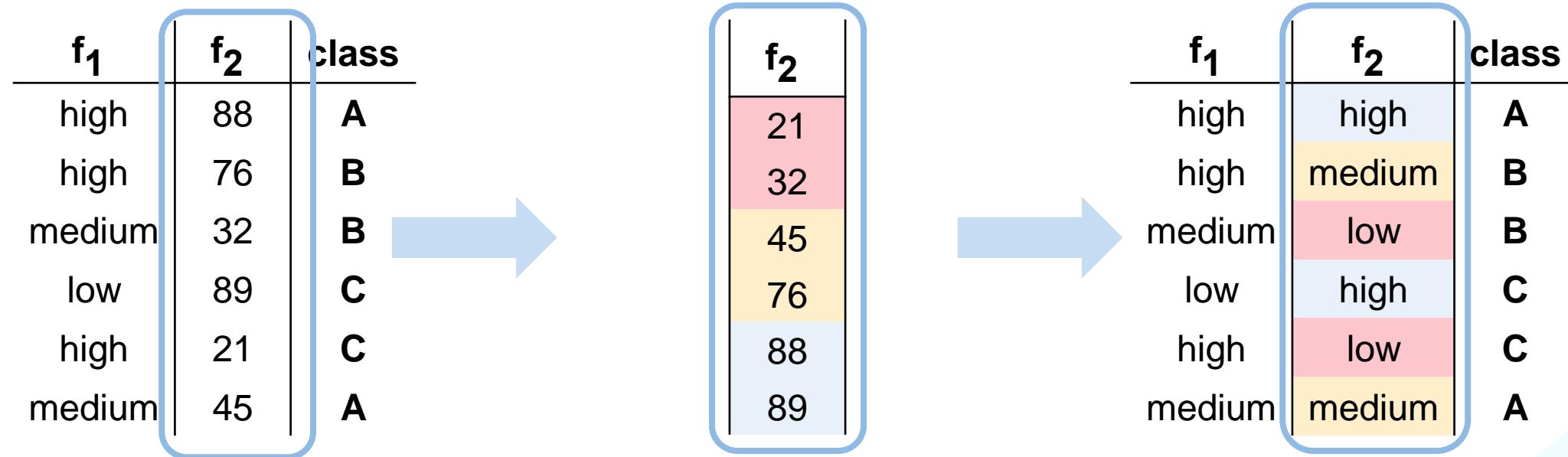
f_1	f_2	class
high	88	A
high	76	B
medium	32	B
low	89	C
high	21	C
medium	45	A

→

f_1 - high	f_1 - medium	f_1 - low	f_2	class
1	0	0	88	A
1	0	0	76	B
0	1	0	32	B
0	0	1	89	C
1	0	0	21	C
0	1	0	45	A

Preprocessing – Transformation

- One-hot encoding: categorical to numerical
- Binning: numerical to categorical

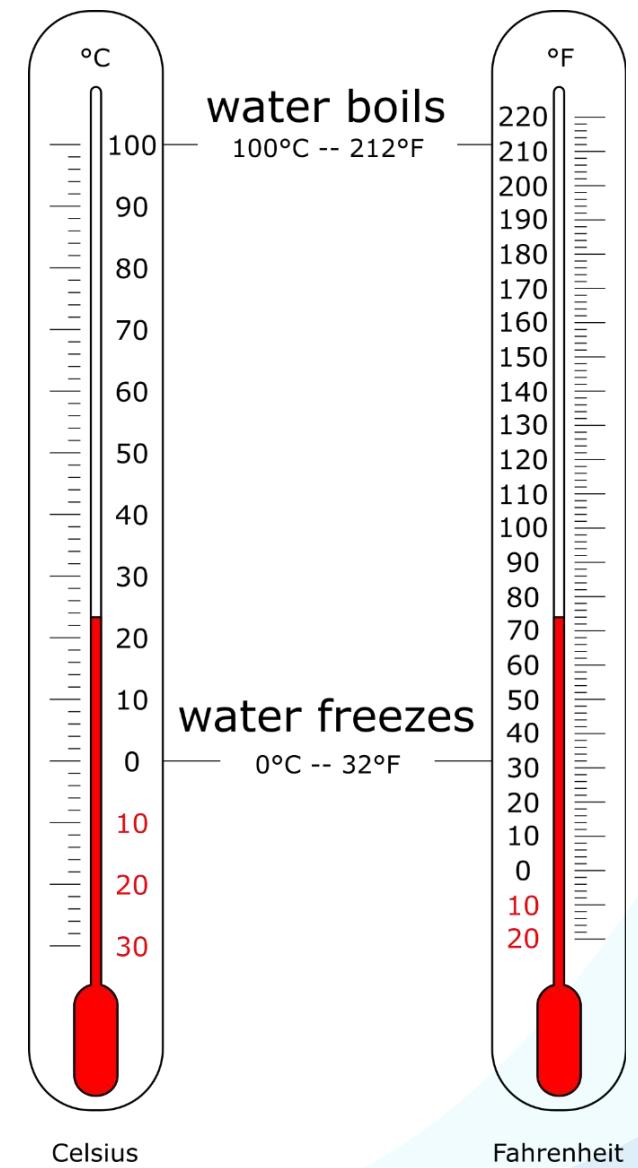


Preprocessing – Normalization

Adjusting the influence of features

- Feature weight and range often depends on the chosen unit (km, mm, miles, ...)
 - Algorithms tend to give more weight to features with a large range
- May introduce an unwanted bias
- May hinder interpretability
- Scales may be non-linear (e.g. logarithmic)

$$\text{Sum of squared errors: } \frac{1}{2} \sum_{i=1}^N (t_i - \mathbb{M}(\mathbf{x}_i))^2$$



Preprocessing – Normalization

Min-max normalization

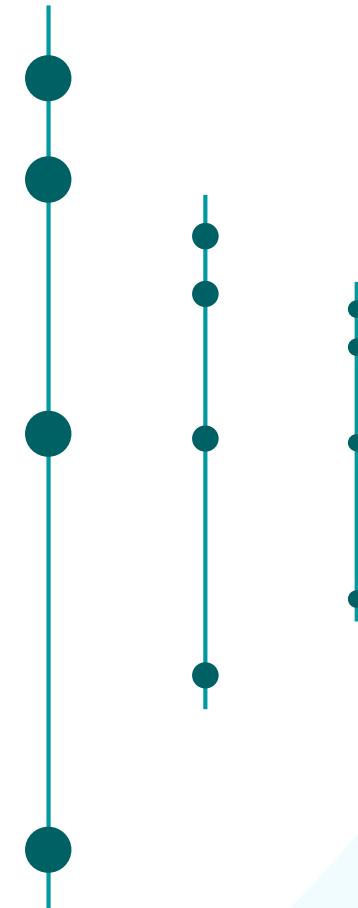
- Maps the values onto a **predefined range** [low, high]
- Preserves **relative differences**, i.e., relations between the data values

We normalize feature d by replacing its value for each instance i as follows:

$$\text{norm}(\mathbf{x}_i[d]) = \frac{\mathbf{x}_i[d] - d_{\min}}{d_{\max} - d_{\min}} \cdot (\text{high} - \text{low}) + \text{low}$$

maximal value
of feature d

minimal value
of feature d



Preprocessing – Normalization

Min-max normalization

d
11
82
33
12
76

$$\begin{aligned}d_{min} &= 11 \\d_{max} &= 82\end{aligned}$$



Consider
high = 100
low = 5

$$\text{norm}(\mathbf{x_i}[d]) = \frac{\mathbf{x_i}[d] - d_{\min}}{d_{\max} - d_{\min}} \cdot (\text{high} - \text{low}) + \text{low}$$

Preprocessing – Normalization

Min-max normalization

d	norm(d)	norm(d)
11	$(11 - 11)/(82 - 11) \cdot (100 - 5) + 5$	5
82	$(82 - 11)/(82 - 11) \cdot (100 - 5) + 5$	100
33	$(33 - 11)/(82 - 11) \cdot (100 - 5) + 5$	34.44
12	$(12 - 11)/(82 - 11) \cdot (100 - 5) + 5$	6.34
76	$(76 - 11)/(82 - 11) \cdot (100 - 5) + 5$	91.97

≈

$$\text{norm}(\mathbf{x_i}[d]) = \frac{\mathbf{x_i}[d] - d_{\min}}{d_{\max} - d_{\min}} \cdot (\text{high} - \text{low}) + \text{low}$$

Preprocessing – Normalization

Standard score (Z-score) normalization

- Uses the standard deviation to quantify the significance of the difference between a value and the overall mean
- Range is $[-\infty, \infty]$, but 0 has a clear meaning
- Useful when actual minimum and maximum of the attribute are unknown
- Useful when outliers may impact min-max normalization

For each i :

$$\text{norm}(\mathbf{x}_i[d]) = \frac{\mathbf{x}_i[d] - \bar{d}}{\text{sd}(d)}$$

\bar{d} is the mean of all values of feature d :

$$\frac{1}{N} \sum_{i=1}^N \mathbf{x}_i[d]$$

$\text{sd}(d)$ is the standard deviation of feature d :

$$\sqrt{\left(\frac{\sum_{i=1}^N (\mathbf{x}_i[d] - \bar{d})^2}{N-1} \right)}$$

Preprocessing – Normalization

Standard score (Z-score) normalization

d
11
82
33
12
76



$$\bar{d} = 42.8$$
$$\text{sd}(d) = 34.259$$

$$\text{norm}(\mathbf{x_i}[d]) = \frac{\mathbf{x_i}[d] - \bar{d}}{\text{sd}(d)}$$

Preprocessing – Normalization

Standard score (Z-score) normalization

d		norm(d)	norm(d)
11		(11 - 42.8)/34.259	-0.93
82	→	(82 - 42.8)/34.259	1.14
33		(33 - 42.8)/34.259	-0.29
12		(12 - 42.8)/34.259	-0.90
76		(76 - 42.8)/34.259	0.97

\approx

$$\text{norm}(\mathbf{x_i}[d]) = \frac{\mathbf{x_i}[d] - \bar{d}}{\text{sd}(d)}$$

Preprocessing – Normalization

Decimal scaling

- Moves the decimal point of the values based on the maximum value
- Scales all values to the interval [-1,1]

For each i :

$$\text{norm}(\mathbf{x}_i[d]) = \frac{\mathbf{x}_i[d]}{10^j}$$

j is chosen such that
the $|d_{max}|$ is of the
form $0.X$ (where X
does not start with 0)



Preprocessing – Normalization

Decimal scaling

- Moves the decimal point of the values based on the maximum value
- Scales all values to the interval [-1,1]

For each i :

$$\text{norm}(\mathbf{x}_i[d]) = \frac{\mathbf{x}_i[d]}{10^j}$$

j is chosen such that
the $|d_{max}|$ is of the
form $0.X$ (where X
does not start with 0)

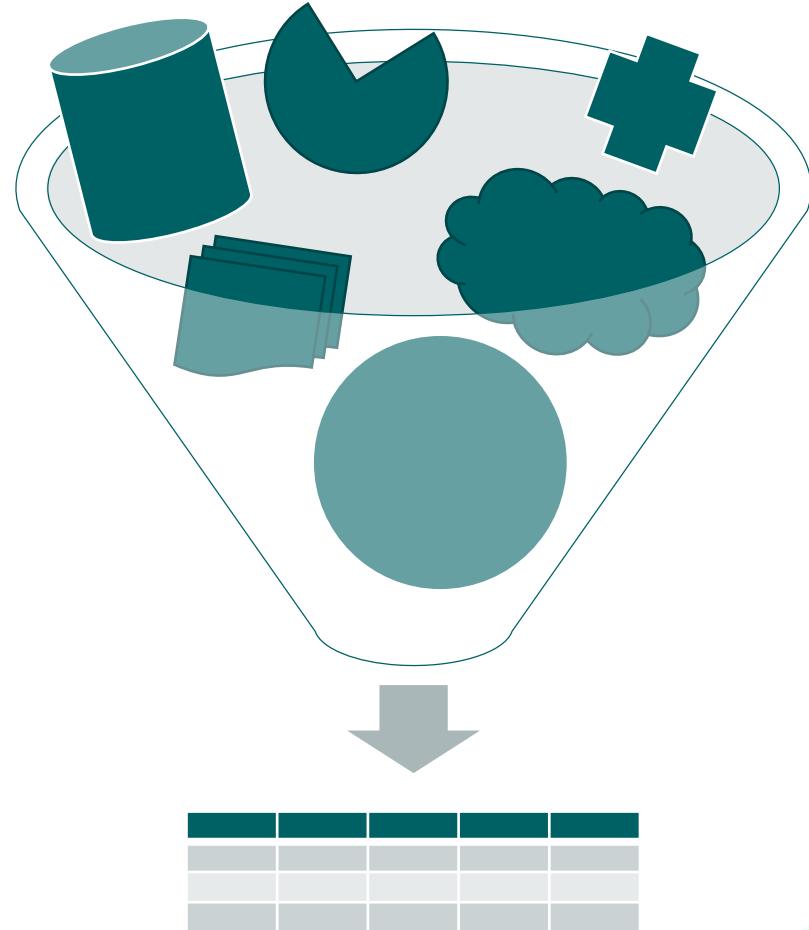
Examples:

- Values in [-877.0, 4.0] are normalized to the range [-0.877, 0.004]
- Values in [0.0003, 0.08] are normalized to the range [0.003, 0.8]



Data Quality & Preprocessing

1. Introduction
2. Missing Values
3. Outliers
4. Semantic Problems
5. Transformation & Normalization
6. **Data Reduction**
7. Conclusion



Preprocessing – Data Reduction

- Analysis may become unfeasible due to size of data
- Goal: reduce the data size but maintain same (or similar) analysis results
- Feature reduction: remove or replace some features
- Instance reduction: remove, replace or aggregate some instances

Feature reduction

ID	f_1	f_2	...	f_D
1				
2				
...				
N				

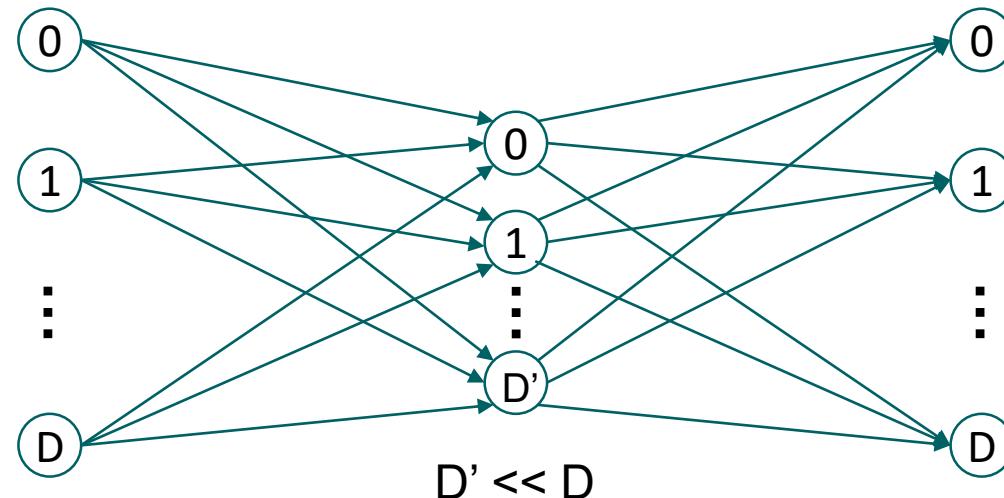
Instance reduction



Preprocessing – Feature Reduction

Projecting data on **fewer** dimensions

- **Autoencoders** (compare text mining): a special type of NN which transforms the input data into a representation with less dimensions (encoding)
- **Principal Component Analysis (PCA)**: represent original features by a few orthogonal (uncorrelated) variables that capture most of the variability



Preprocessing – Feature Reduction

Projecting data on **fewer dimensions**

- [Autoencoders](#) (compare text mining): a special type of NN which transforms the input data into a representation with less dimensions (encoding)
- [Principal Component Analysis \(PCA\)](#): represent original features by a few orthogonal (uncorrelated) variables that capture most of the variability

Feature subset selection: detect and remove **irrelevant/redundant features**

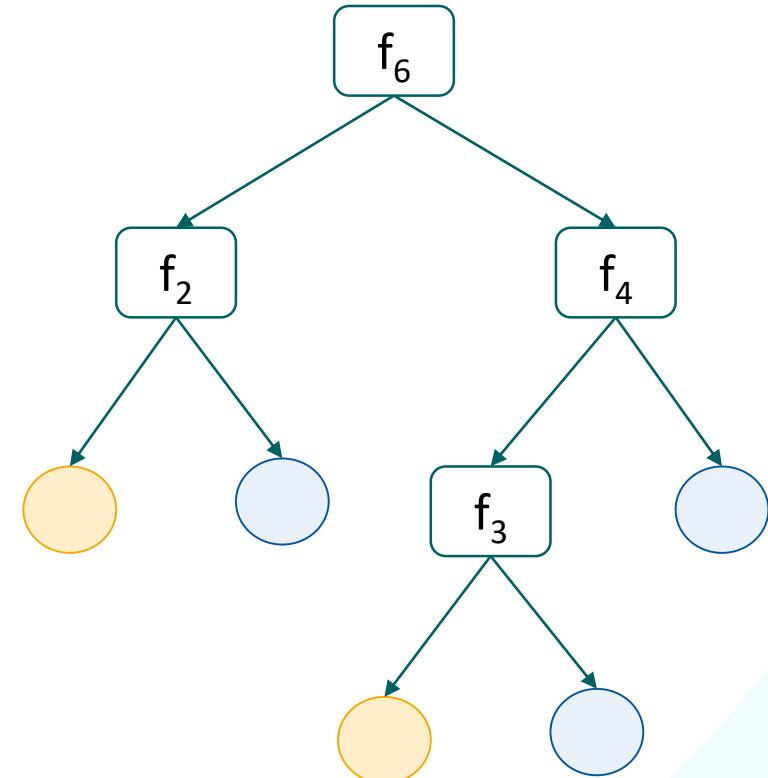
- Use domain knowledge (e.g., remove identifiers)
- Exploit dependencies (e.g., delete features that can be estimated from others using regression)
- Model-driven (e.g. delete features that are not used in a constructed decision tree or, more general, features that can be left out without reducing the quality of the model much)

Preprocessing – Feature Subset Selection

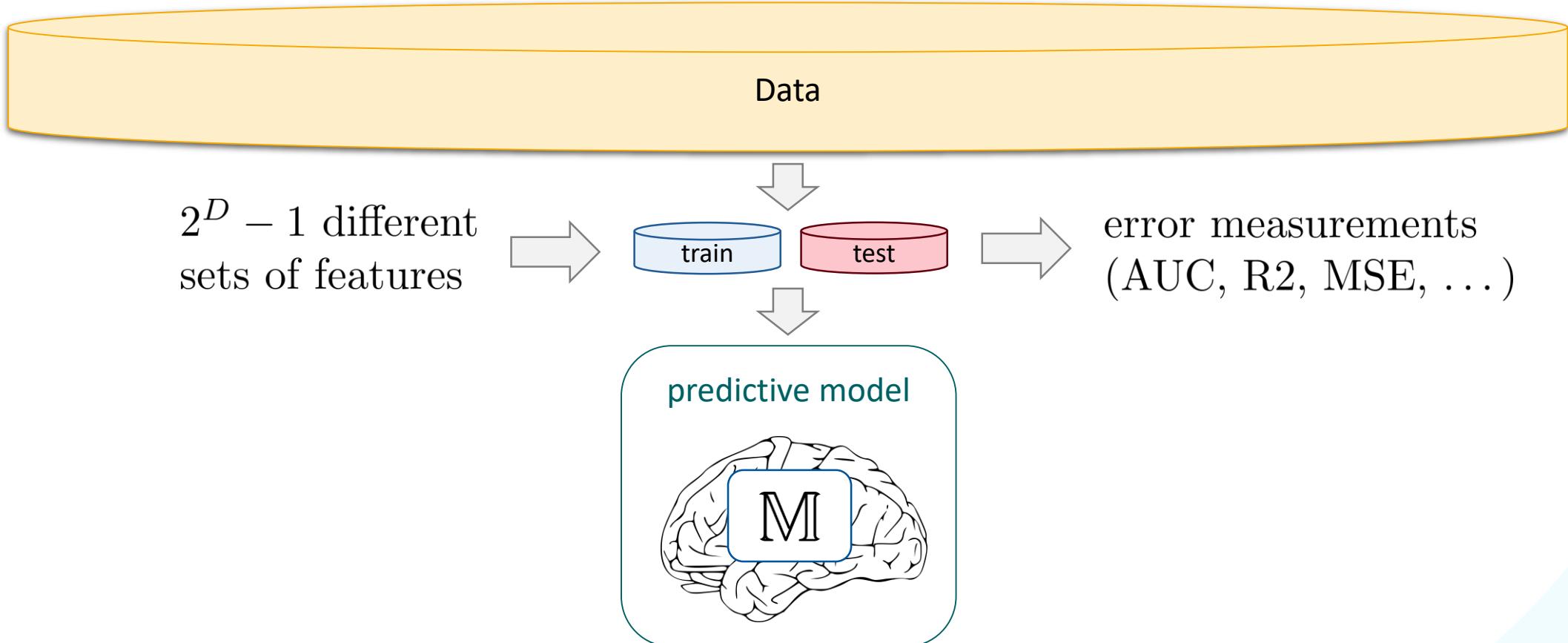
Example

1. Initial features: $f_1, f_2, f_3, f_4, f_5, f_6, f_7, f_8, f_9$
2. Construct a tree
3. f_2, f_3, f_4, f_6 are relevant (according to the tree)

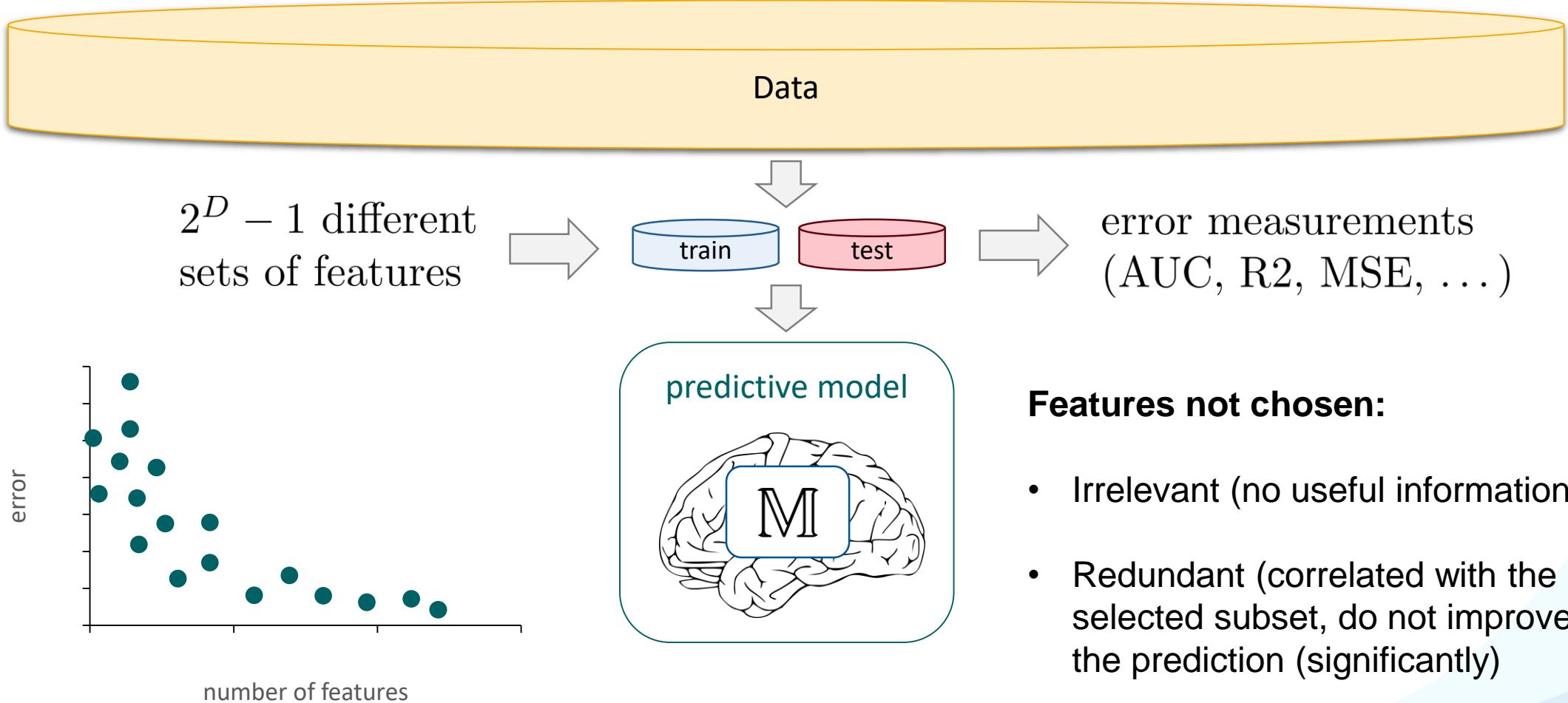
Idea: features correlated with the chosen subset would not improve the classification (significantly) and are therefore not part of the tree.



Preprocessing – Feature Subset Selection

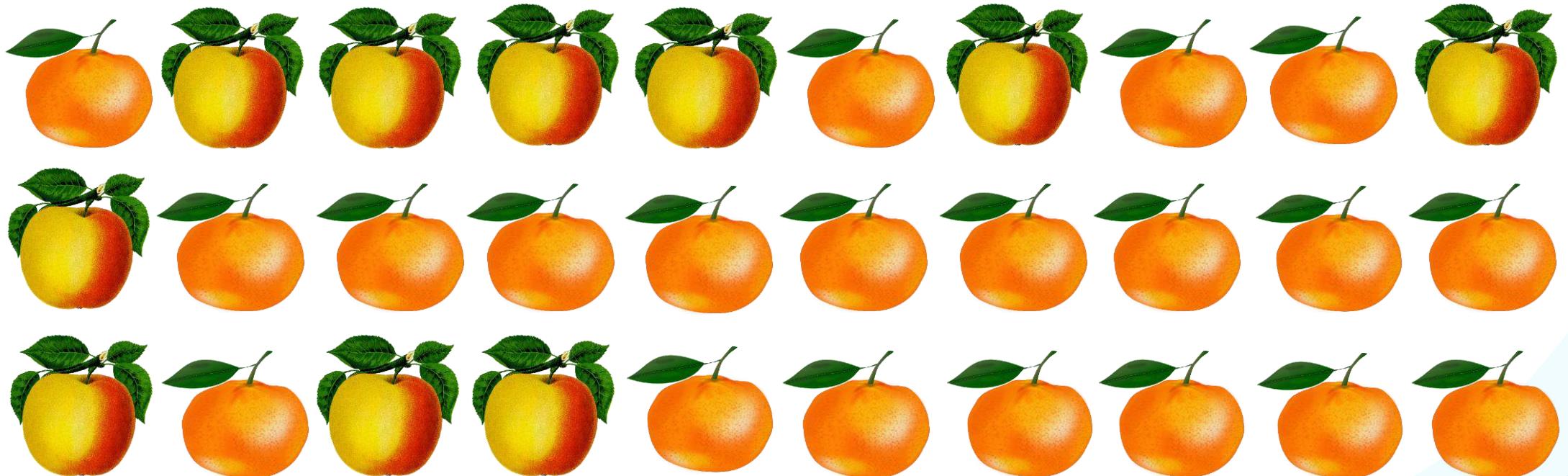


Preprocessing – Feature Subset Selection



Preprocessing – Sampling

Goals: make the data smaller, remove or introduce biases

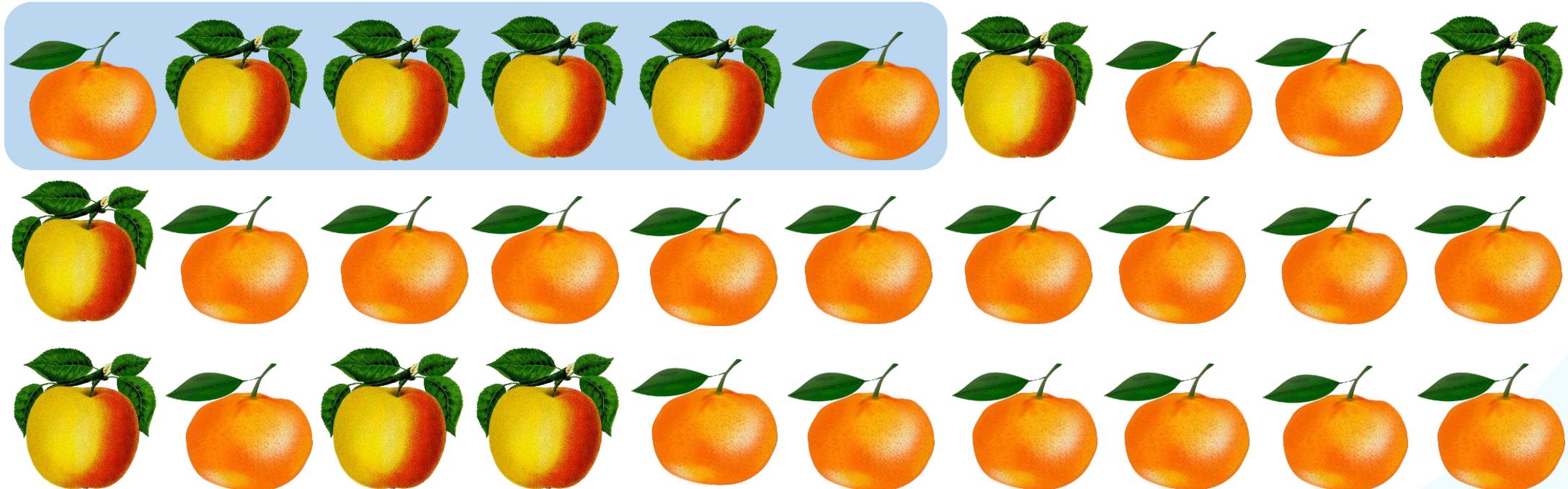


Preprocessing – Sampling



Top sampling:

take the first N instances

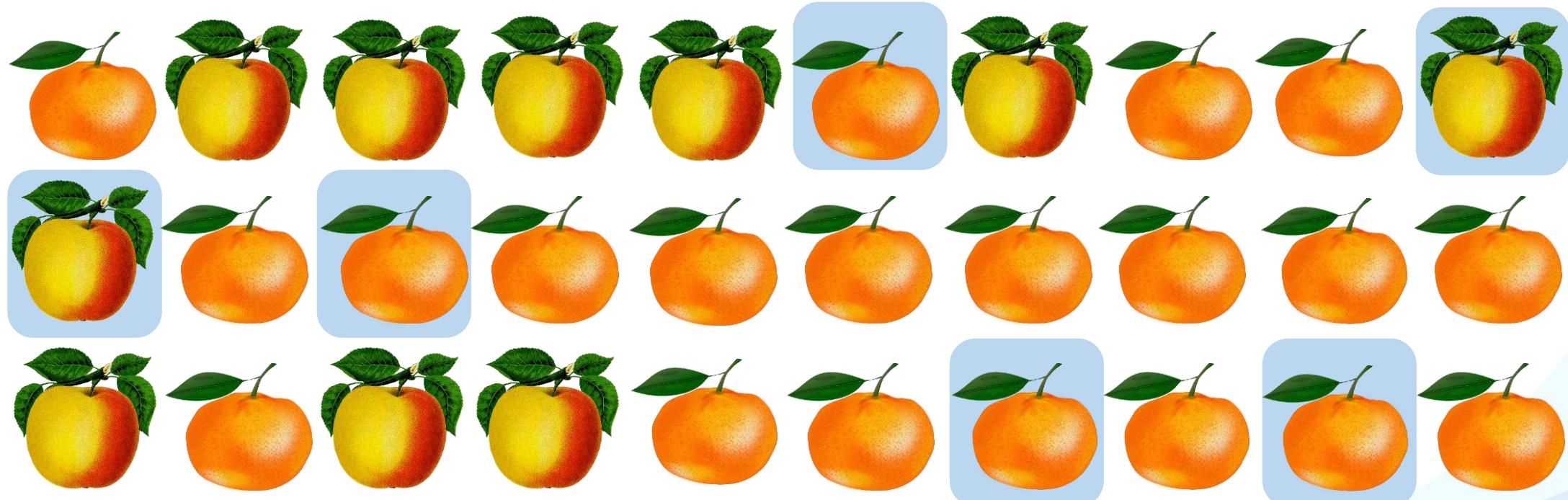


Preprocessing – Sampling

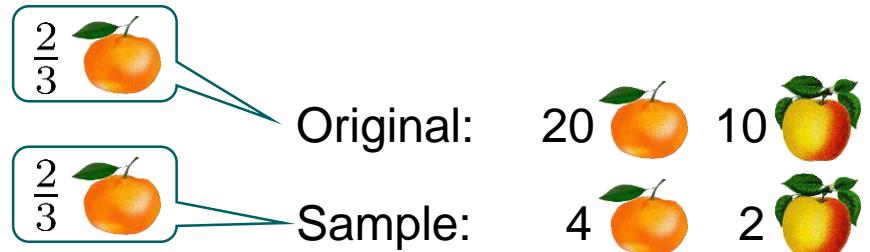


Random sampling:

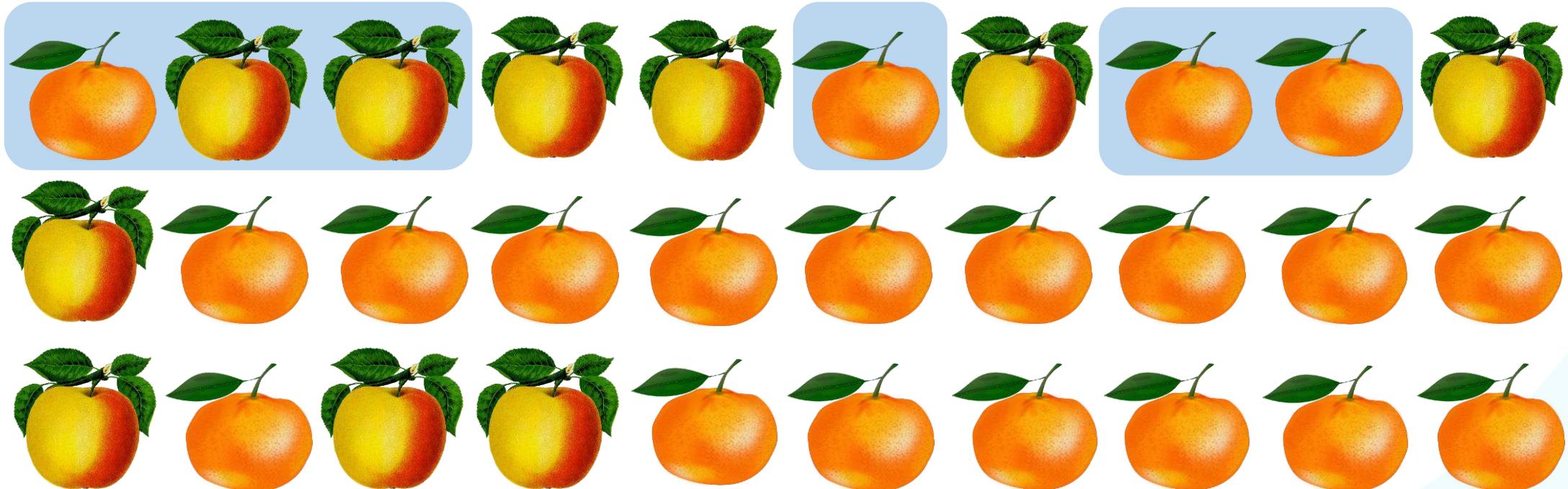
take N arbitrary instances (based on random generator)



Preprocessing – Sampling



Stratified sampling: ensure that relative frequencies are maintained (e.g., take the same percentage from every group)

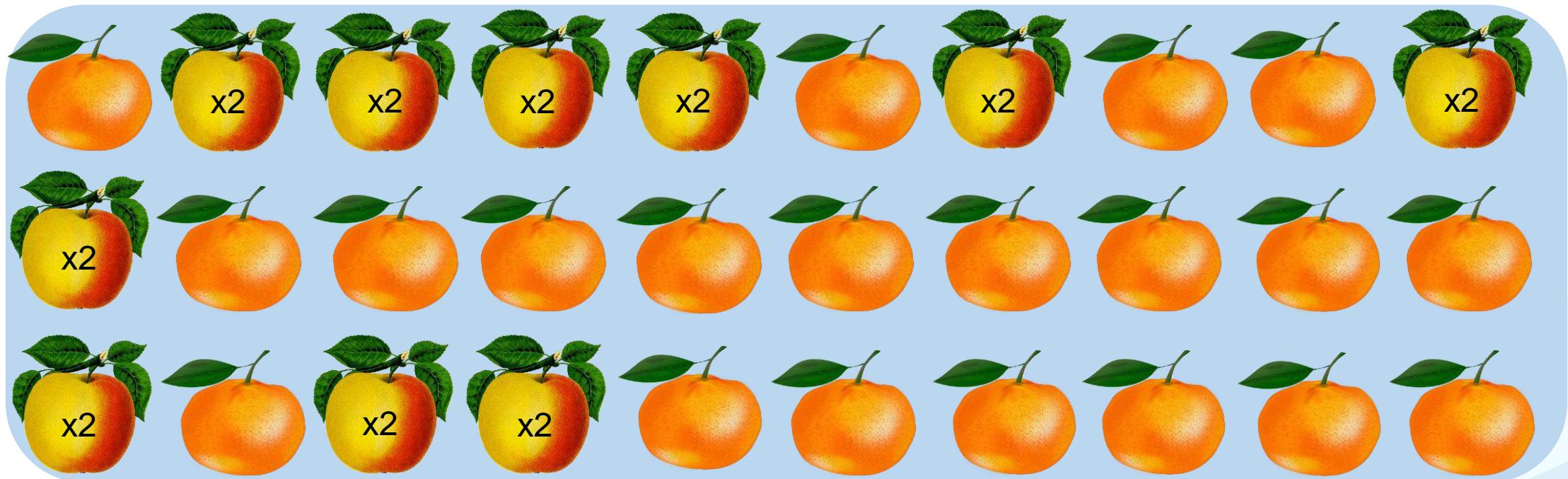


Preprocessing – Sampling



Over-sampling: ensure a certain distribution

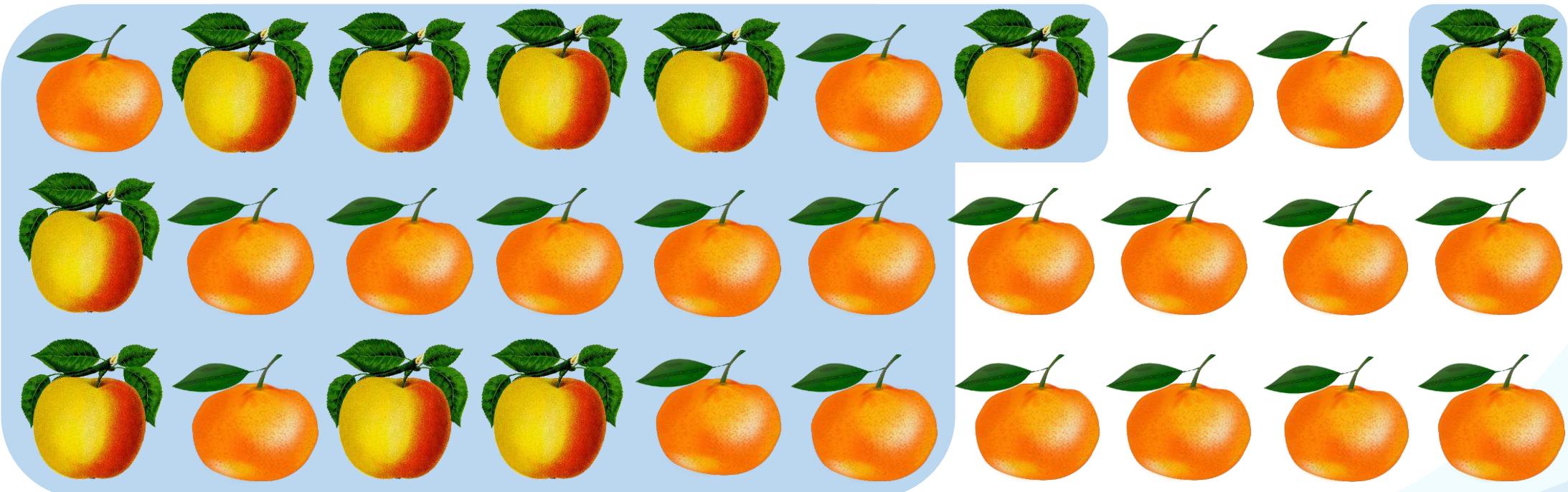
(e.g. equal frequency for each group) by duplicating under-represented instances



Preprocessing – Sampling



Under-sampling: ensure a certain distribution
(e.g. equal frequency for each group) by leaving out over-represented instances



To Conclude



Goal: increase data quality and modify the data to suit the analysis question and applied techniques

Best strategy/solution: depends on the data, context and goal of the analysis

Data quality aspects

- Missing data
- Noise/outliers
- Semantic problems

Garbage in, Garbage out (GIGO)

Data preprocessing

- Transformation
- Normalization
- Data reduction



Elements of Machine Learning & Data Science

Winter semester 2023/24

Evaluation of Supervised Learning (1)

Prof. Holger Hoos (partially based on material from Wil van der Aalst)

Learning Goals

At the end of this module, students should be able to

- assess the quality of a model obtained from a supervised machine learning method using widely accepted methods, including standard performance metrics, confusion matrices, ROC curves
- demonstrate understanding and working knowledge of the problems that can occur when using supervised learning procedures and the models obtained from them
- explain when and why it is important to distinguish between training, validation and testing data
- explain standard validation techniques, including k -fold and leave-one-out cross-validation
- assess performance differences using appropriate statistical techniques
- explain the problems that can arise from unbalanced data sets and demonstrate understanding as well as working knowledge of methods for addressing these problems

Key questions:

- How good is an ML model?
- How good could an ML model be?



Key questions:

- How good is an ML model?
- How good could an ML model be?



You have used supervised ML to train a predictive model.



Question: How do you assess the quality of the model?

Motivation: Predicting delayed flights



ID	Origin	Destination	Precipitation	...	Traffic	Target
1	Frankfurt	Cologne	139	...	152	On Time
2	Madrid	Paris	349	...	55	On Time
3	La Paz	Madrid	702	...	76	Delayed
4	Hanoi	Singapore	251	...	169	On Time
5	Dubai	Frankfurt	615	...	117	Delayed
6	Cologne	Madrid	400	...	89	On Time
7	Bergen	Paris	698	...	28	Delayed
8	Rome	Barcelona	322	...	9	On Time
9	Berlin	Rome	221	...	5	On Time
10	Paris	Paris	132	...	165	On Time
11	Toronto	Frankfurt	730	...	220	Delayed
...

You have used supervised ML to train a predictive model.



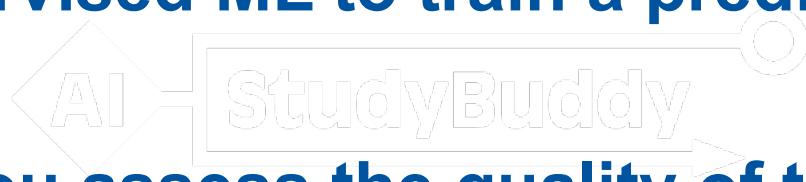
Question: How do you assess the quality of the model?

TPS = Think, Pair, Share Exercises

1. Be ready to take some notes.
 2. Think about the problem/question.
 3. Jot down your answers (bullet points).
- 
4. Pair up with your neighbour, explain/discuss your answers.
 5. Modify your answers based on your discussion.
6. Volunteer to give/explain your answer to everyone.

TPS Exercise:

You have used supervised ML to train a predictive model.

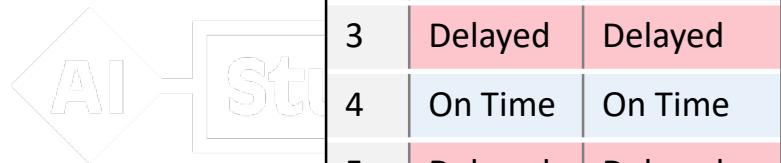


Question: How do you assess the quality of the model?

Running Example

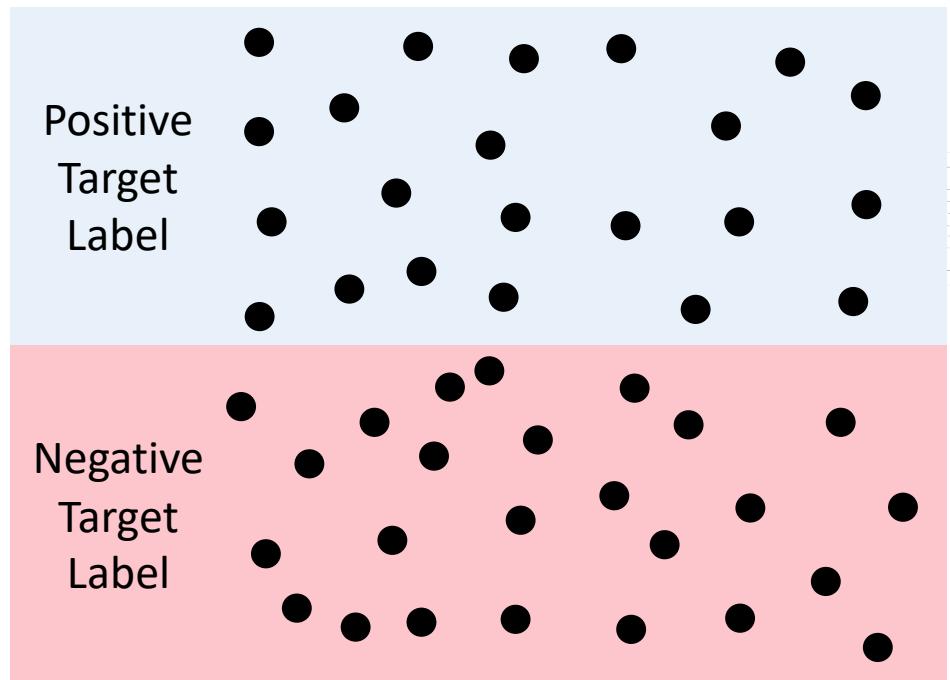
Predicting delayed flights (set of 20 instances)

- Target Feature:
On Time (positive),
Delayed (negative)

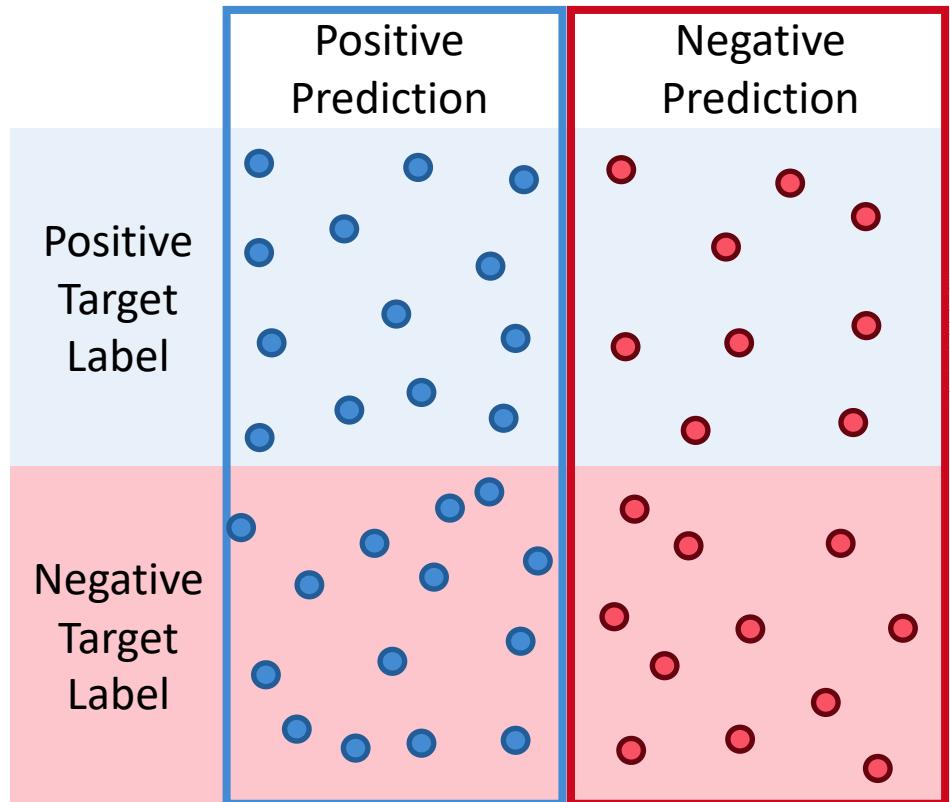


ID	Target Label	Prediction
1	On Time	Delayed
2	On Time	Delayed
3	Delayed	Delayed
4	On Time	On Time
5	Delayed	Delayed
6	On Time	On Time
7	Delayed	Delayed
8	On Time	On Time
9	On Time	On Time
10	On Time	On Time

ID	Target Label	Prediction
11	Delayed	Delayed
12	On Time	Delayed
13	Delayed	Delayed
14	Delayed	Delayed
15	Delayed	Delayed
16	Delayed	Delayed
17	Delayed	On Time
18	On Time	On Time
19	Delayed	Delayed
20	Delayed	On Time

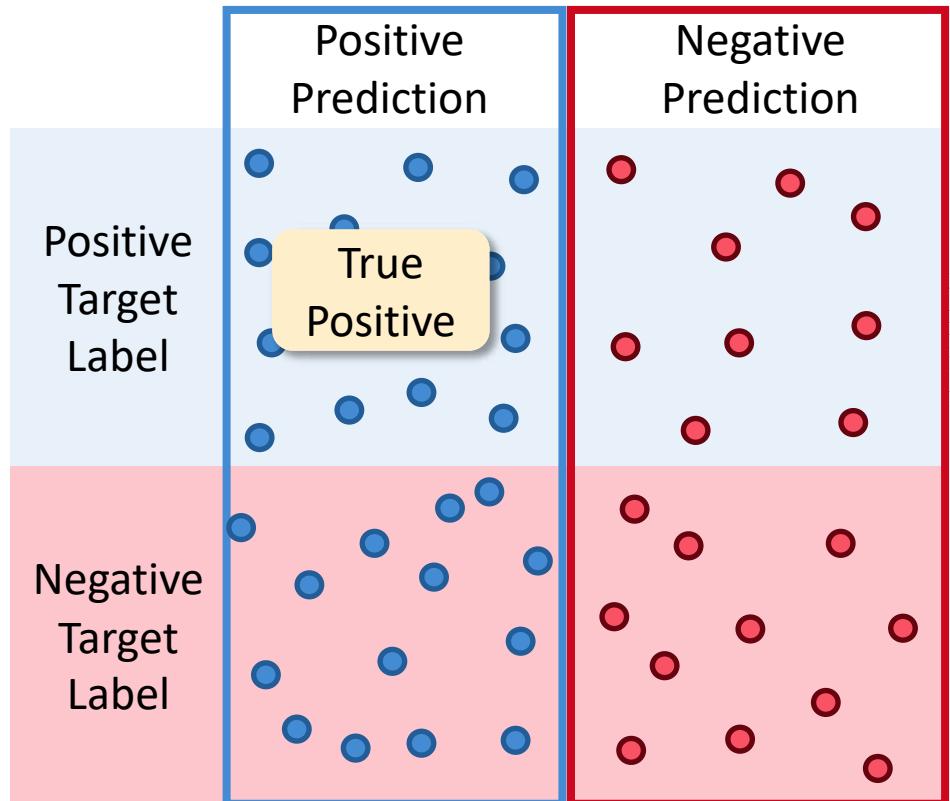


ID	Target Label	Prediction	
1	On Time	Delayed	
2	On Time	Delayed	
3	Delayed	Delayed	
4	On Time	On Time	
5	Delayed	Delayed	
6	On Time	On Time	
7	Delayed	Delayed	
8	On Time	On Time	
9	On Time	On Time	
10	On Time	On Time	
11	Delayed	Delayed	
12	On Time	Delayed	
13	Delayed	Delayed	
14	Delayed	Delayed	
15	Delayed	Delayed	
16	Delayed	Delayed	
17	Delayed	On Time	
18	On Time	On Time	
19	Delayed	Delayed	
20	Delayed	On Time	



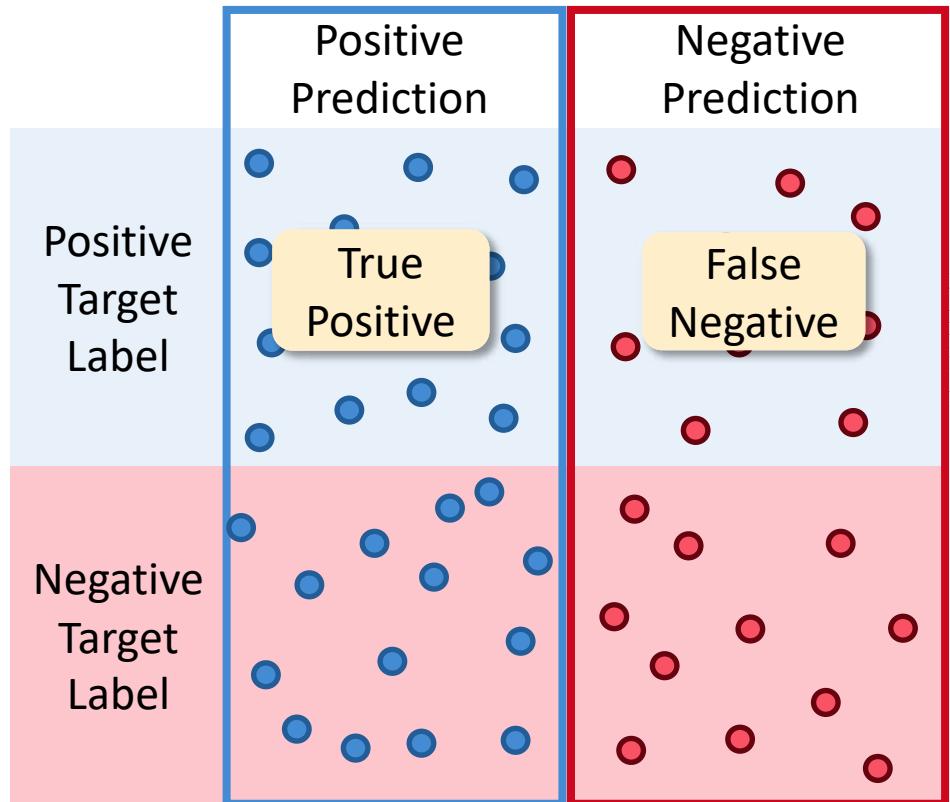
ID	Target Label	Prediction	
1	On Time	Delayed	
2	On Time	Delayed	
3	Delayed	Delayed	
4	On Time	On Time	
5	Delayed	Delayed	
6	On Time	On Time	
7	Delayed	Delayed	
8	On Time	On Time	
9	On Time	On Time	
10	On Time	On Time	

ID	Target Label	Prediction	
11	Delayed	Delayed	
12	On Time	Delayed	
13	Delayed	Delayed	
14	Delayed	Delayed	
15	Delayed	Delayed	
16	Delayed	Delayed	
17	Delayed	On Time	
18	On Time	On Time	
19	Delayed	Delayed	
20	Delayed	On Time	



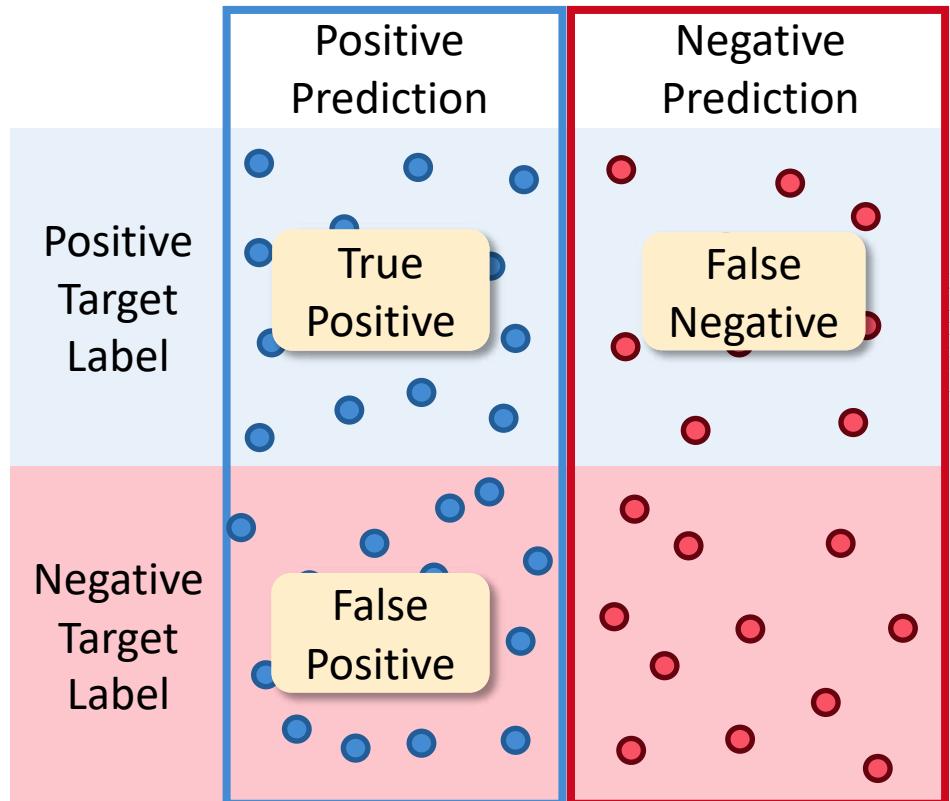
ID	Target Label	Prediction	
1	On Time	Delayed	
2	On Time	Delayed	
3	Delayed	Delayed	
4	On Time	On Time	TP
5	Delayed	Delayed	
6	On Time	On Time	TP
7	Delayed	Delayed	
8	On Time	On Time	TP
9	On Time	On Time	TP
10	On Time	On Time	TP

ID	Target Label	Prediction	
11	Delayed	Delayed	
12	On Time	Delayed	
13	Delayed	Delayed	
14	Delayed	Delayed	
15	Delayed	Delayed	
16	Delayed	Delayed	
17	Delayed	On Time	
18	On Time	On Time	TP
19	Delayed	Delayed	
20	Delayed	On Time	



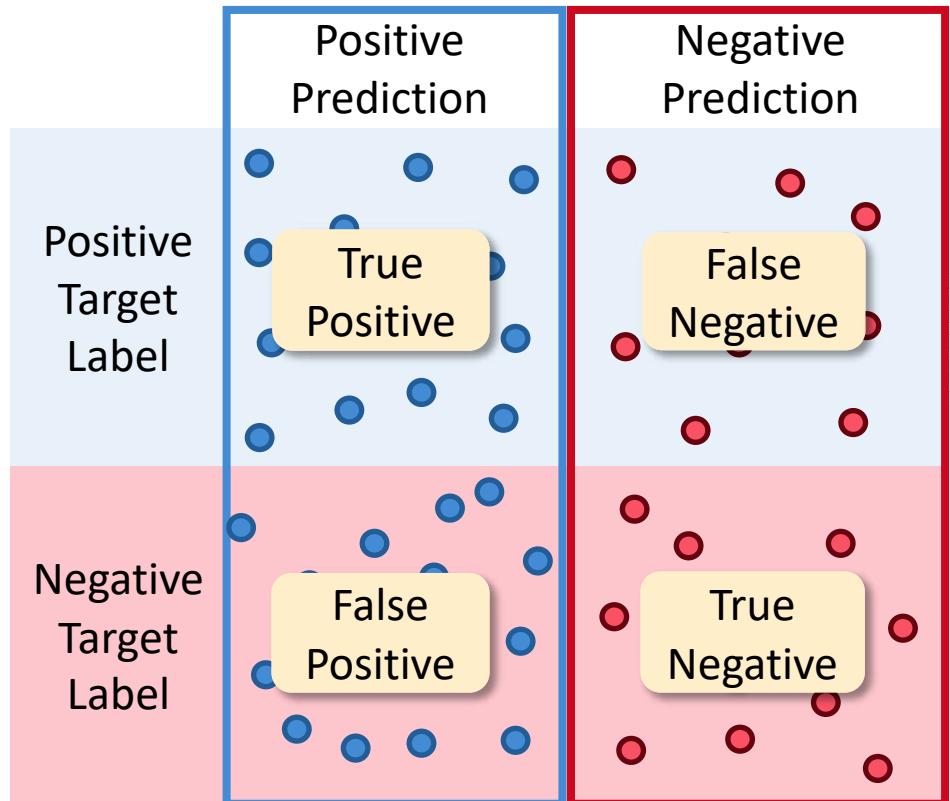
ID	Target Label	Prediction	
1	On Time	Delayed	FN
2	On Time	Delayed	FN
3	Delayed	Delayed	
4	On Time	On Time	TP
5	Delayed	Delayed	
6	On Time	On Time	TP
7	Delayed	Delayed	
8	On Time	On Time	TP
9	On Time	On Time	TP
10	On Time	On Time	TP

ID	Target Label	Prediction	
11	Delayed	Delayed	
12	On Time	Delayed	FN
13	Delayed	Delayed	
14	Delayed	Delayed	
15	Delayed	Delayed	
16	Delayed	Delayed	
17	Delayed	On Time	
18	On Time	On Time	TP
19	Delayed	Delayed	
20	Delayed	On Time	



ID	Target Label	Prediction	
1	On Time	Delayed	FN
2	On Time	Delayed	FN
3	Delayed	Delayed	
4	On Time	On Time	TP
5	Delayed	Delayed	
6	On Time	On Time	TP
7	Delayed	Delayed	
8	On Time	On Time	TP
9	On Time	On Time	TP
10	On Time	On Time	TP

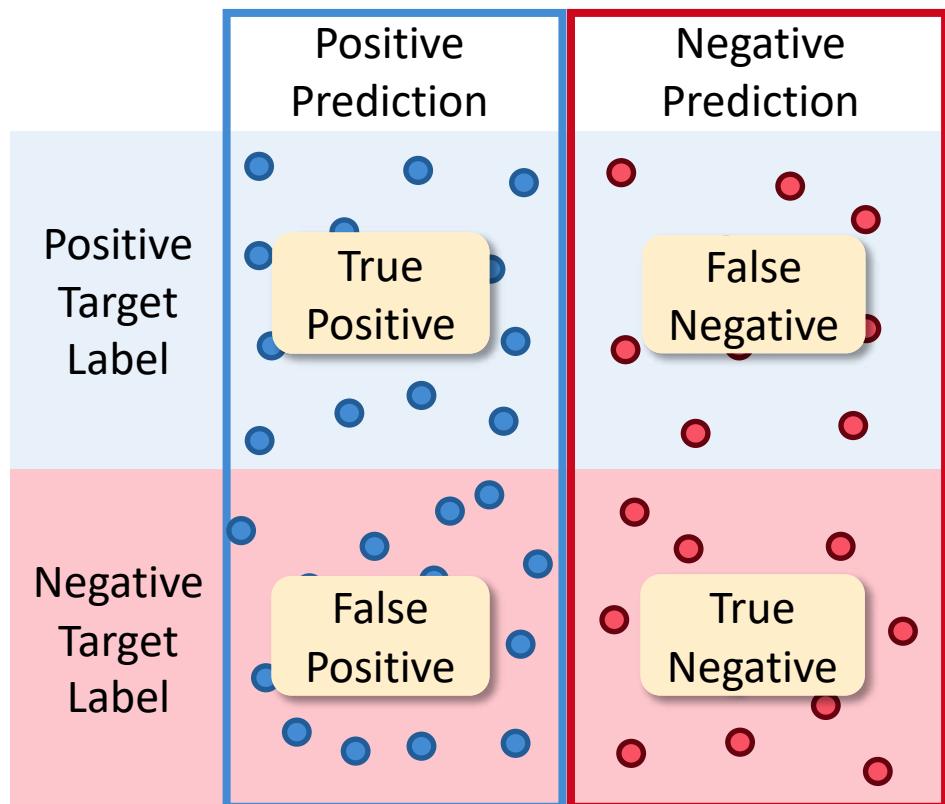
ID	Target Label	Prediction	
11	Delayed	Delayed	
12	On Time	Delayed	FN
13	Delayed	Delayed	
14	Delayed	Delayed	
15	Delayed	Delayed	
16	Delayed	Delayed	
17	Delayed	On Time	FP
18	On Time	On Time	TP
19	Delayed	Delayed	
20	Delayed	On Time	FP



ID	Target Label	Prediction	
1	On Time	Delayed	FN
2	On Time	Delayed	FN
3	Delayed	Delayed	TN
4	On Time	On Time	TP
5	Delayed	Delayed	TN
6	On Time	On Time	TP
7	Delayed	Delayed	TN
8	On Time	On Time	TP
9	On Time	On Time	TP
10	On Time	On Time	TP

ID	Target Label	Prediction	
11	Delayed	Delayed	TN
12	On Time	Delayed	FN
13	Delayed	Delayed	TN
14	Delayed	Delayed	TN
15	Delayed	Delayed	TN
16	Delayed	Delayed	TN
17	Delayed	On Time	FP
18	On Time	On Time	TP
19	Delayed	Delayed	TN
20	Delayed	On Time	FP

Confusion Matrix



ID	Target Label	Prediction	
1	On Time	Delayed	FN
2	On Time	Delayed	FN
3	Delayed	Delayed	TN
4	On Time	On Time	TP
5	Delayed	Delayed	TN
6	On Time	On Time	TP
7	Delayed	Delayed	TN
8	On Time	On Time	TP
9	On Time	On Time	TP
10	On Time	On Time	TP

ID	Target Label	Prediction	
11	Delayed	Delayed	TN
12	On Time	Delayed	FN
13	Delayed	Delayed	TN
14	Delayed	Delayed	TN
15	Delayed	Delayed	TN
16	Delayed	Delayed	TN
17	Delayed	On Time	FP
18	On Time	On Time	TP
19	Delayed	Delayed	TN
20	Delayed	On Time	FP

Confusion Matrix

	Positive Prediction	Negative Prediction
Positive Target Label	TP (number of true positives)	FN (number of false negatives)
Negative Target Label	FP (number of false positives)	TN (number of true negatives)



ID	Target Label	Prediction		ID	Target Label	Prediction	
1	On Time	Delayed	FN	11	Delayed	Delayed	TN
2	On Time	Delayed	FN	12	On Time	Delayed	FN
3	Delayed	Delayed	TN	13	Delayed	Delayed	TN
4	On Time	On Time	TP	14	Delayed	Delayed	TN
5	Delayed	Delayed	TN	15	Delayed	Delayed	TN
6	On Time	On Time	TP	16	Delayed	Delayed	TN
7	Delayed	Delayed	TN	17	Delayed	On Time	FP
8	On Time	On Time	TP	18	On Time	On Time	TP
9	On Time	On Time	TP	19	Delayed	Delayed	TN
10	On Time	On Time	TP	20	Delayed	On Time	FP

Confusion Matrix

	Positive Prediction	Negative Prediction
Positive Target Label	6	3
Negative Target Label	2	9



ID	Target Label	Prediction	
1	On Time	Delayed	FN
2	On Time	Delayed	FN
3	Delayed	Delayed	TN
4	On Time	On Time	TP
5	Delayed	Delayed	TN
6	On Time	On Time	TP
7	Delayed	Delayed	TN
8	On Time	On Time	TP
9	On Time	On Time	TP
10	On Time	On Time	TP
ID	Target Label	Prediction	
11	Delayed	Delayed	TN
12	On Time	Delayed	FN
13	Delayed	Delayed	TN
14	Delayed	Delayed	TN
15	Delayed	Delayed	TN
16	Delayed	Delayed	TN
17	Delayed	On Time	FP
18	On Time	On Time	TP
19	Delayed	Delayed	TN
20	Delayed	On Time	FP

TPS Exercise:

We have trained a predictive model using supervised learning and computed a confusion matrix based on predictions on a given set of data.



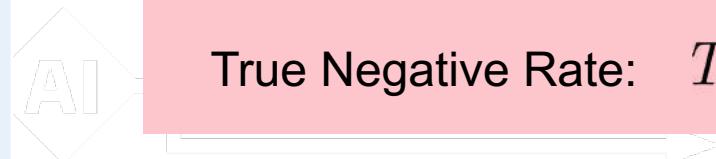
Question:

How can we assess performance with a single number?

Confusion Matrix

Performance Measures

	Positive Prediction	Negative Prediction
Positive Target Label	TP=6	FN=3
Negative Target Label	FP=2	TN=9



$$\text{True Positive Rate: } TPR = \frac{TP}{TP+FN}$$

$$\text{False Negative Rate: } FNR = \frac{FN}{TP+FN}$$

$$\text{False Positive Rate: } FPR = \frac{FP}{FP+TN}$$

$$\text{True Negative Rate: } TNR = \frac{TN}{FP+TN}$$

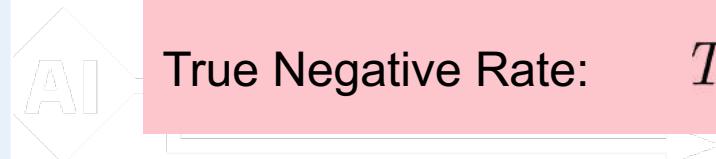
$$\text{Classification Accuracy: } \frac{TP+TN}{TP+TN+FP+FN}$$

$$\text{Misclassification Rate: } \frac{FP+FN}{TP+TN+FP+FN}$$

Confusion Matrix

Performance Measures

	Positive Prediction	Negative Prediction
Positive Target Label	TP=6	FN=3
Negative Target Label	FP=2	TN=9



True Positive Rate: $TPR = \frac{TP}{TP+FN}$

False Negative Rate: $FNR = \frac{FN}{TP+FN}$

False Positive Rate: $FPR = \frac{FP}{FP+TN}$

True Negative Rate: $TNR = \frac{TN}{FP+TN}$

Recall: $recall = \frac{TP}{TP+FN} = TPR$

Precision: $precision = \frac{TP}{TP+FP}$

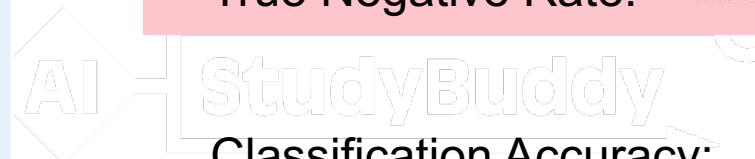
F_1 : $F_1 = 2 \cdot \frac{precision \cdot recall}{precision + recall}$

Confusion Matrix



Performance Measures

	Positive Prediction	Negative Prediction
Positive Target Label	TP=6	FN=3
Negative Target Label	FP=2	TN=9



True Positive Rate: $TPR = \frac{TP}{TP+FN} = \frac{6}{6+3} = \frac{2}{3}$

False Negative Rate: $FNR = \frac{FN}{TP+FN} = \frac{3}{6+3} = \frac{1}{3}$

False Positive Rate: $FPR = \frac{FP}{FP+TN} = \frac{2}{2+9} = \frac{2}{11}$

True Negative Rate: $TNR = \frac{TN}{FP+TN} = \frac{9}{2+9} = \frac{9}{11}$

$TPR + FNR = 1$

Classification Accuracy:

Misclassification Rate:

Recall:

Precision:

F_1 :

$FPR + TNR = 1$

Confusion Matrix



Performance Measures

	Positive Prediction	Negative Prediction
Positive Target Label	TP=6	FN=3
Negative Target Label	FP=2	TN=9



True Positive Rate: $TPR = \frac{TP}{TP+FN} = \frac{6}{6+3} = \frac{2}{3}$

False Negative Rate: $FNR = \frac{FN}{TP+FN} = \frac{3}{6+3} = \frac{1}{3}$

False Positive Rate: $FPR = \frac{FP}{FP+TN} = \frac{2}{2+9} = \frac{2}{11}$

True Negative Rate: $TNR = \frac{TN}{FP+TN} = \frac{9}{2+9} = \frac{9}{11}$

StudyBuddy

Classification Accuracy: $\frac{TP+TN}{TP+TN+FP+FN} = \frac{6+9}{6+9+2+3} = \frac{15}{20}$

Misclassification Rate: $\frac{FP+FN}{TP+TN+FP+FN} = \frac{2+3}{6+9+2+3} = \frac{5}{20}$

Recall:

Precision:

F_1 :

Classification Accuracy
+ Misclassification Rate = 1

Confusion Matrix



Performance Measures

	Positive Prediction	Negative Prediction
Positive Target Label	TP=6	FN=3
Negative Target Label	FP=2	TN=9



True Positive Rate: $TPR = \frac{TP}{TP+FN} = \frac{6}{6+3} = \frac{2}{3}$

False Negative Rate: $FNR = \frac{FN}{TP+FN} = \frac{3}{6+3} = \frac{1}{3}$

False Positive Rate: $FPR = \frac{FP}{FP+TN} = \frac{2}{2+9} = \frac{2}{11}$

True Negative Rate: $TNR = \frac{TN}{FP+TN} = \frac{9}{2+9} = \frac{9}{11}$

Classification Accuracy: $\frac{TP+TN}{TP+TN+FP+FN} = \frac{6+9}{6+9+2+3} = \frac{15}{20}$

Misclassification Rate: $\frac{FP+FN}{TP+TN+FP+FN} = \frac{2+3}{6+9+2+3} = \frac{5}{20}$

Recall: $recall = \frac{TP}{TP+FN} = TPR = \frac{2}{3} \approx 0.67$

Precision: $precision = \frac{TP}{TP+FP} = \frac{6}{6+2} = \frac{3}{4} = 0.75$

F_1 : $F_1 = 2 \cdot \frac{precision \cdot recall}{precision + recall} = \frac{2 \cdot \frac{3}{4} \cdot \frac{2}{3}}{\frac{3}{4} + \frac{2}{3}} = \frac{12}{17} \approx 0.71$

TPS Exercise:

We have trained a predictive model using supervised learning and computed a confusion matrix based on predictions on a given set of data.



Question:

Which measure should we use to assess performance?

TPS Exercise:

We have trained a predictive model using supervised learning and computed a confusion matrix based on predictions on a given set of data.



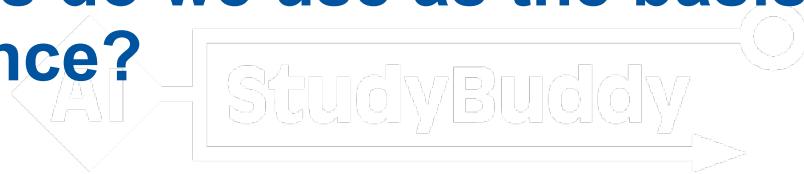
Question:

Which measure should we use to assess performance?

It depends ...

Often, a single measure is not enough.

What set of instances do we use as the basis for assessing performance?

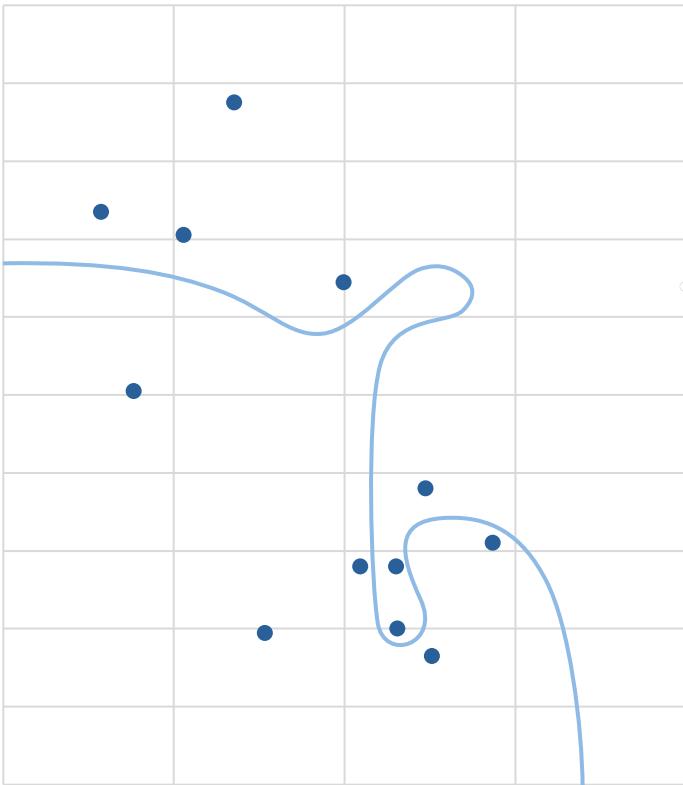


What set of instances do we use as the basis for assessing performance?

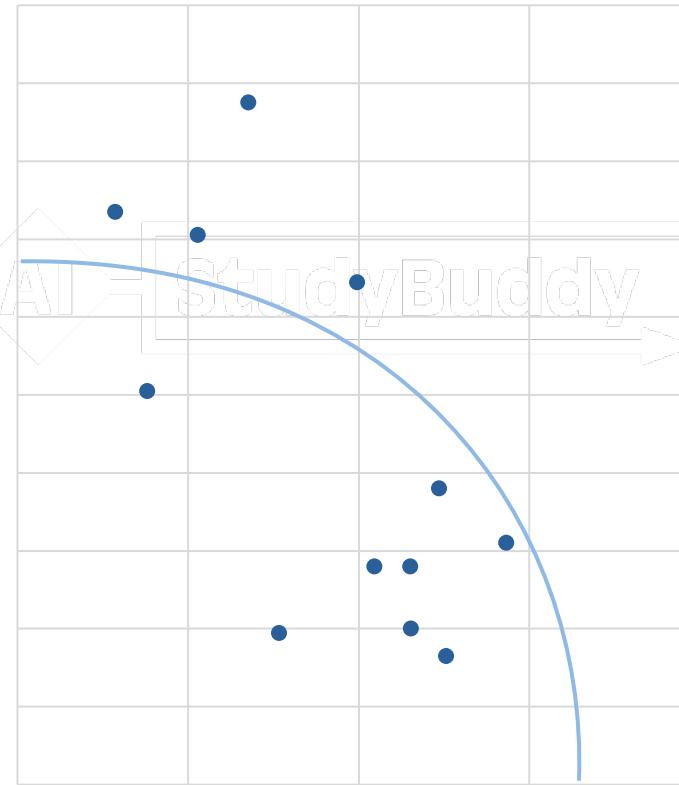


Let's use training instances. What could go wrong?

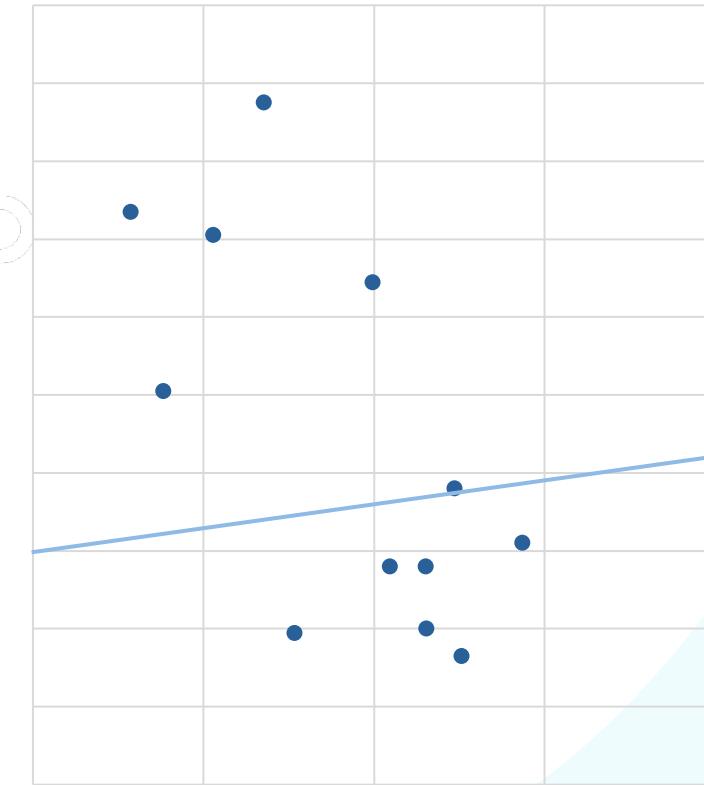
Overfitting



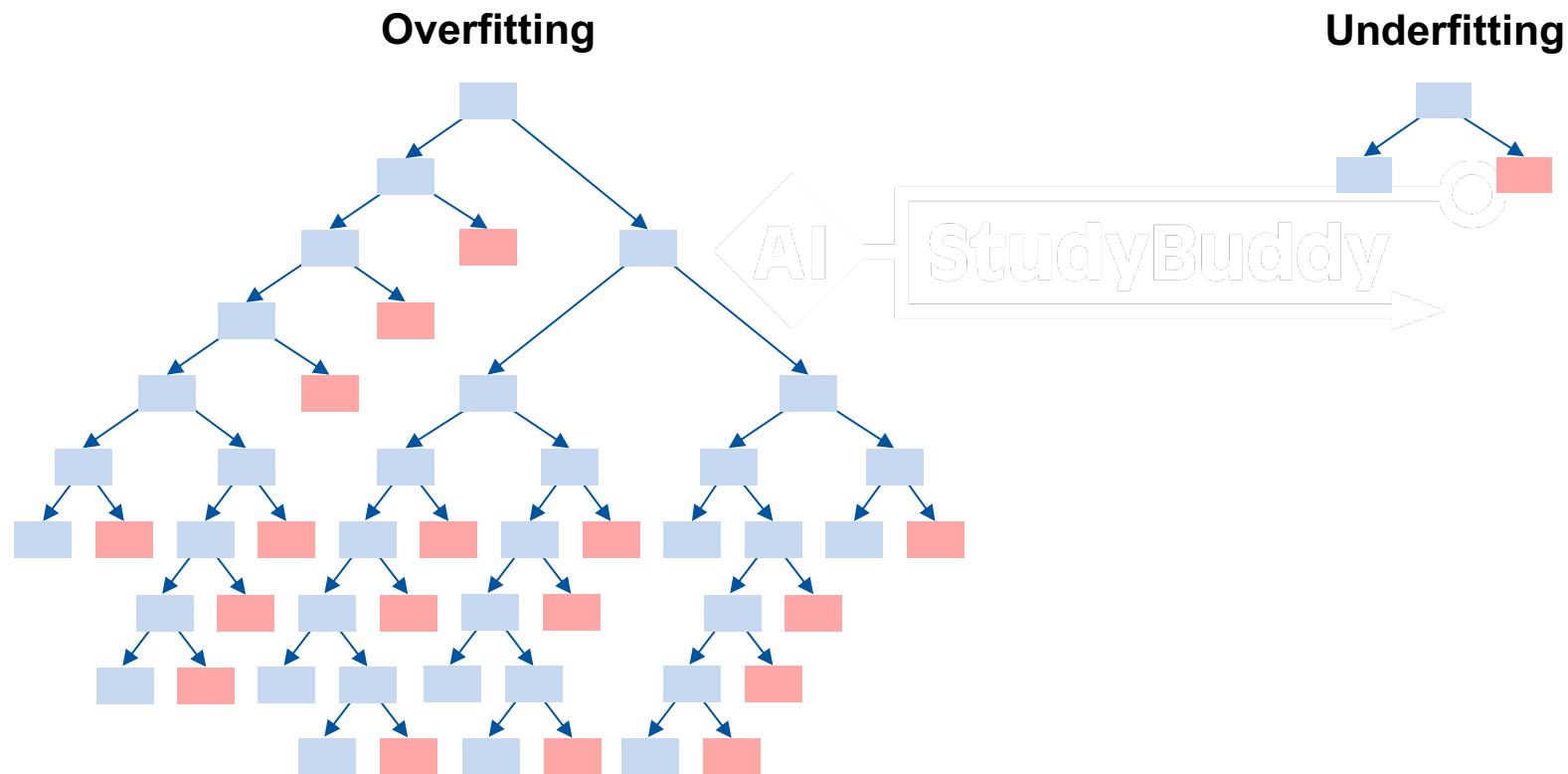
Good



Underfitting



Flight Classification (Running Example)



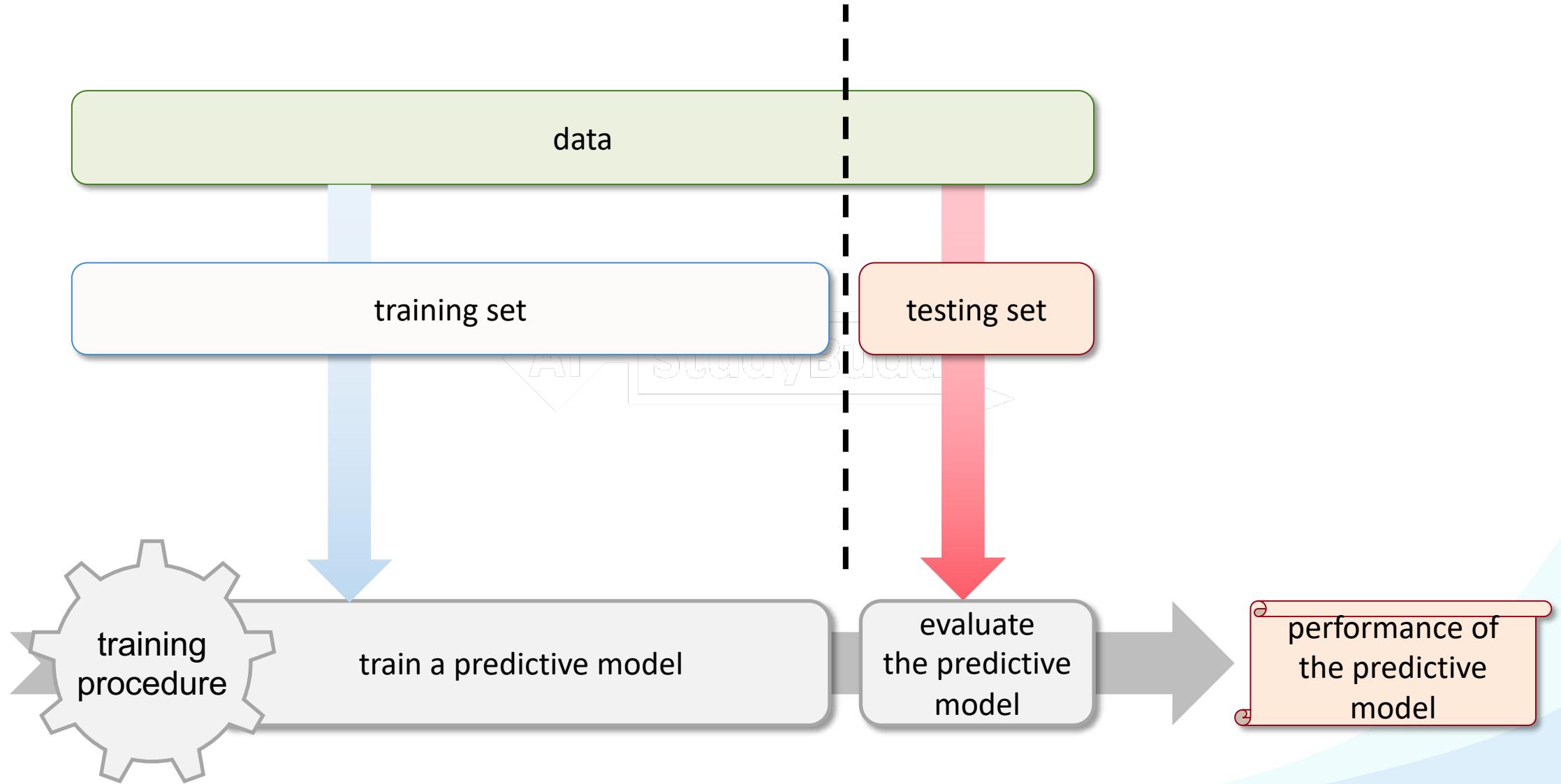
ID	Target
1	On Time
2	On Time
3	Delayed
4	On Time
5	Delayed
6	On Time
7	Delayed
8	On Time
9	On Time
10	On Time
11	Delayed
...	...

What set of instances do we use as the basis for assessing performance?

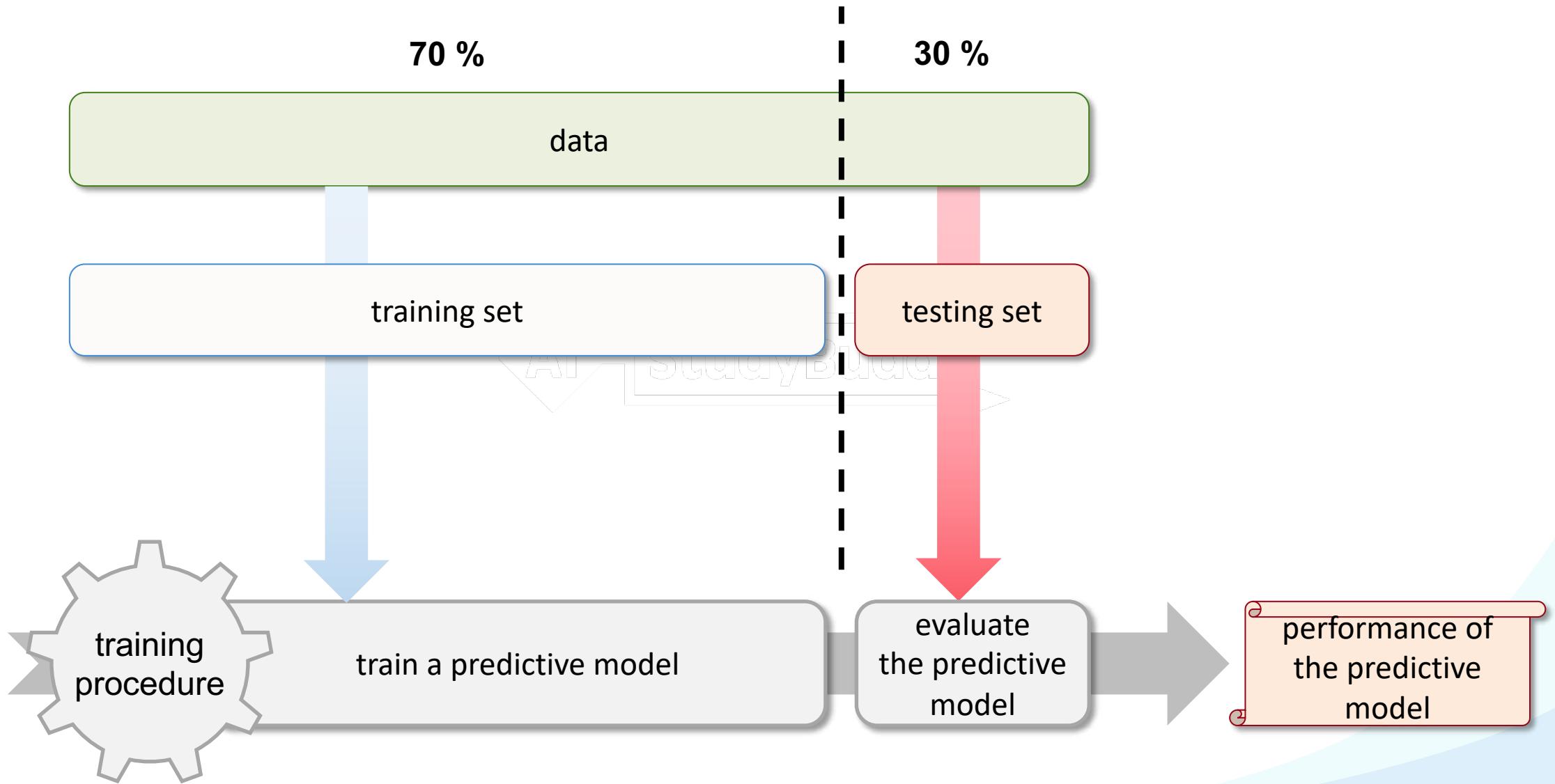


Key issue: generalisation to new data
→ **don't assess performance based on training data!**

Training & Testing Data

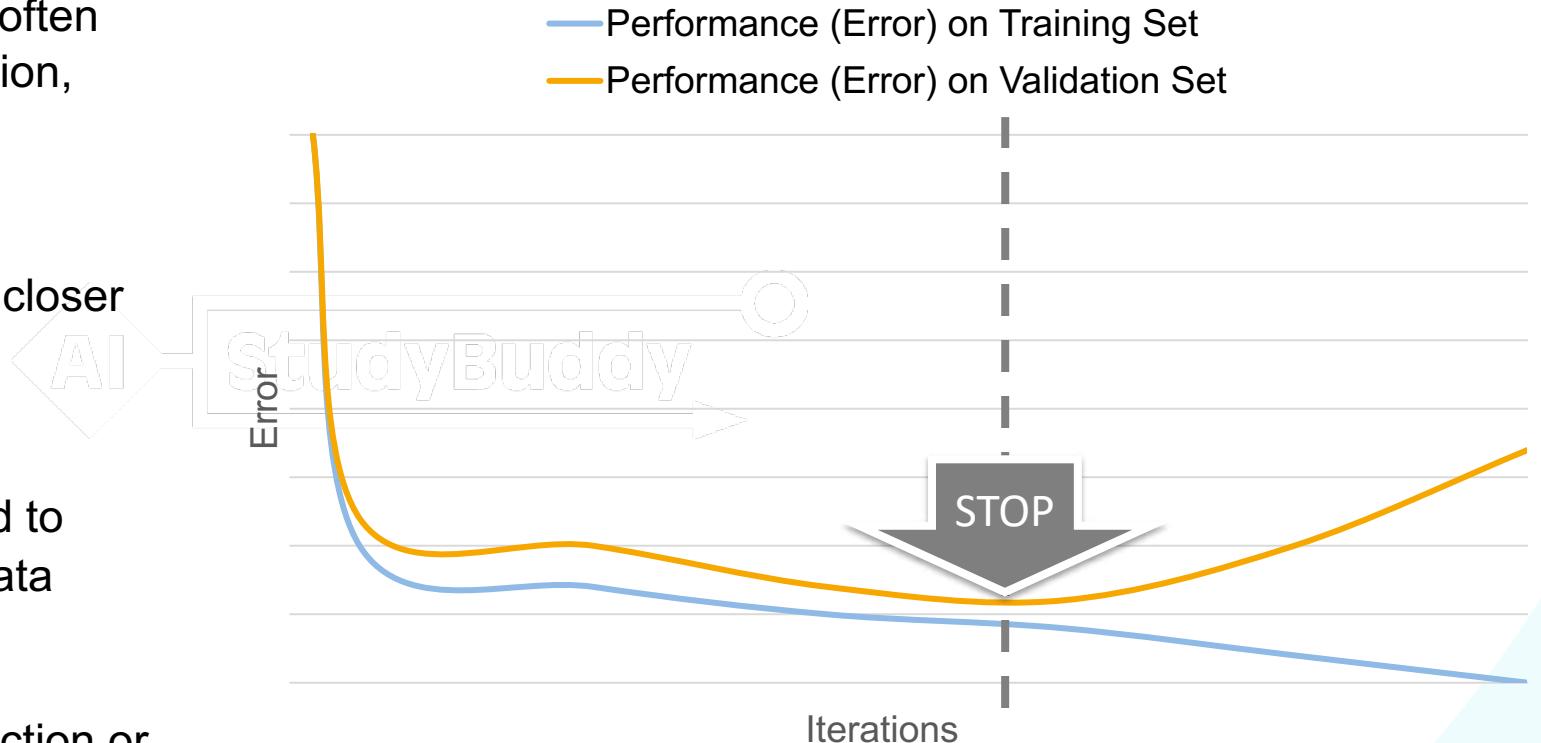


Training & Testing Data

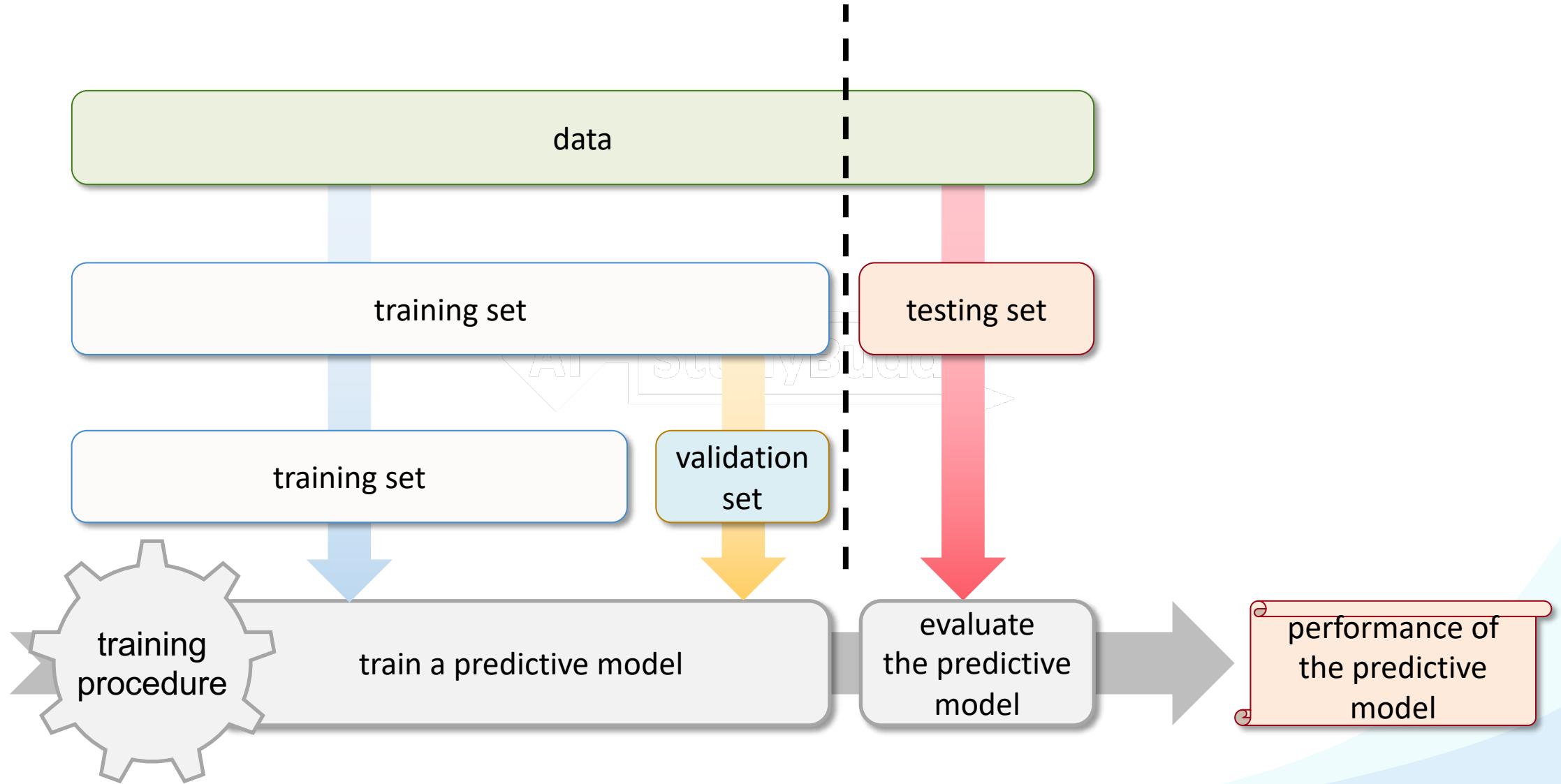


Validation Set

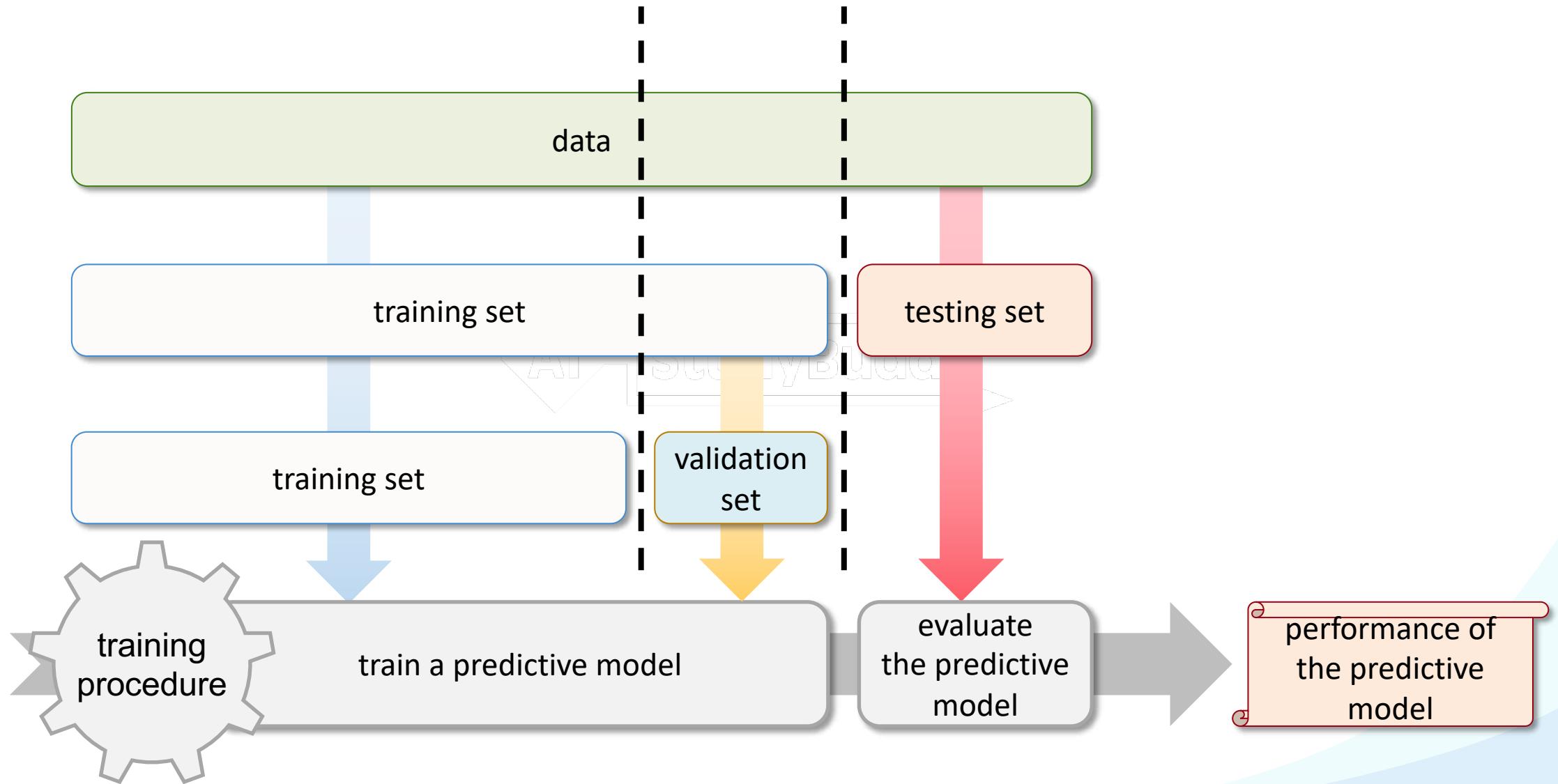
- Training a predictive model is often done iteratively (e.g., Regression, Neural Networks)
- The model is fitted closer and closer to the training data
- The validation set can be used to avoid overfitting the training data
- Often used for parameter selection or hyperparameter tuning



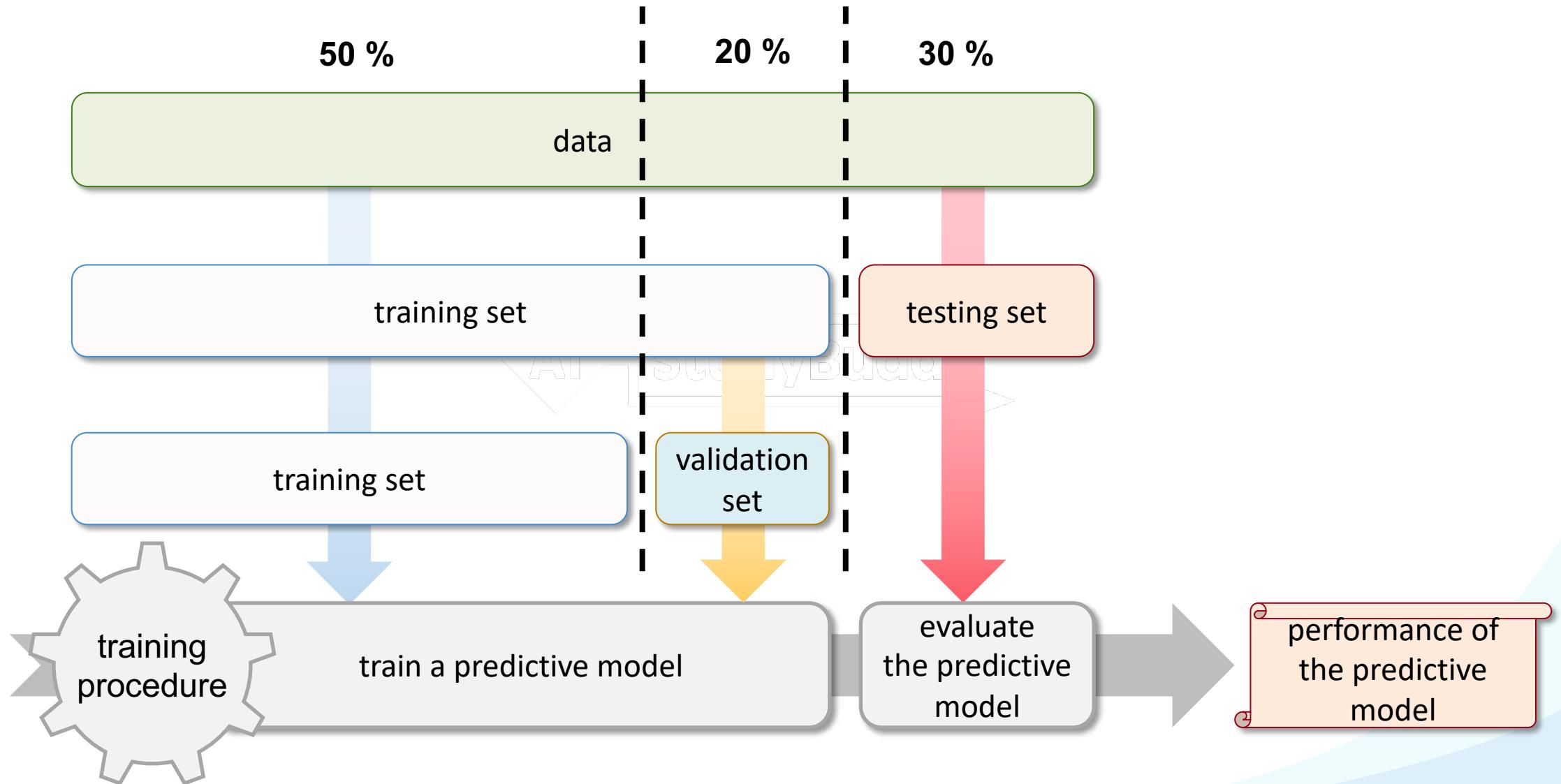
Training & Testing Data



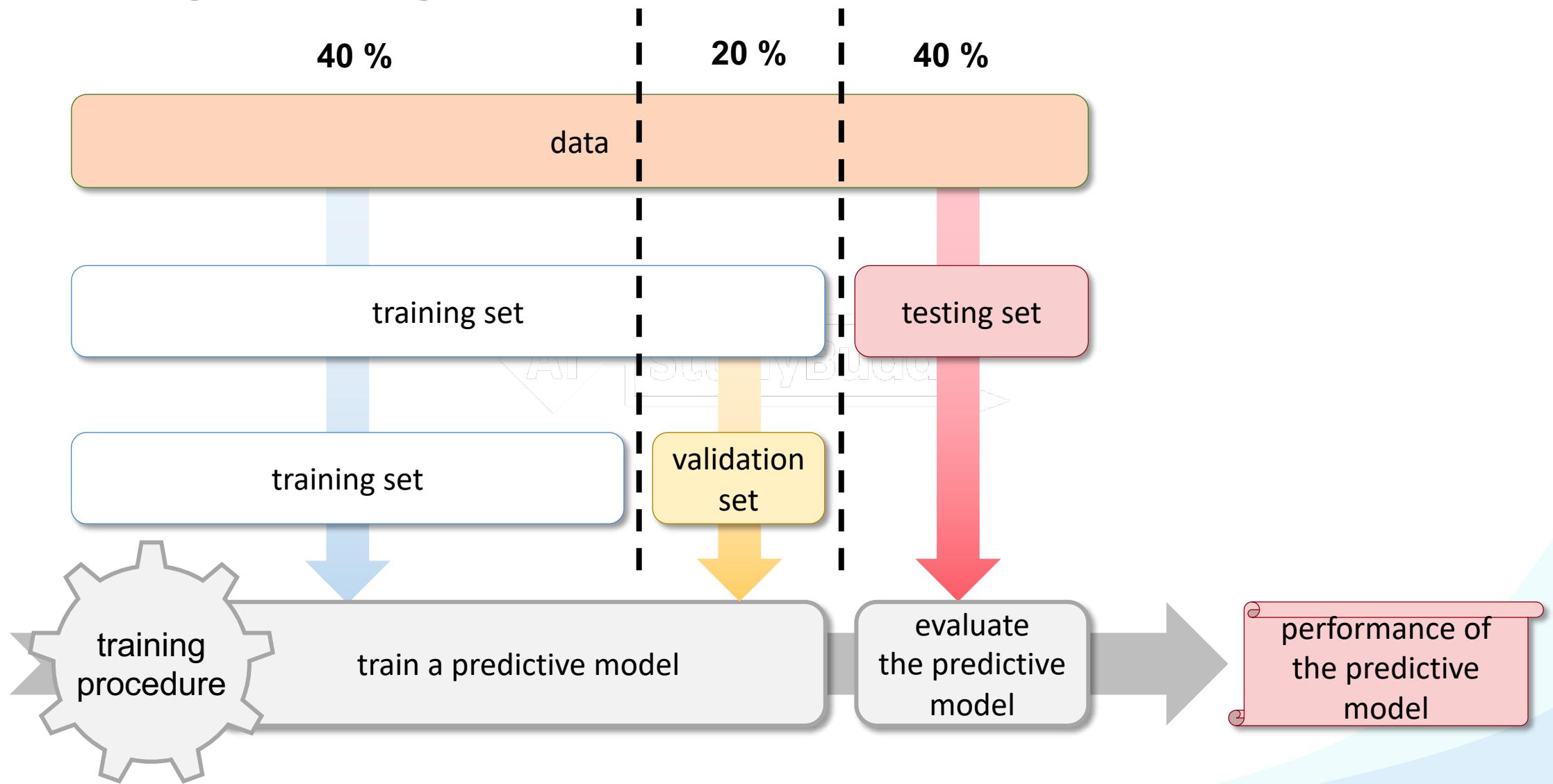
Training & Testing Data



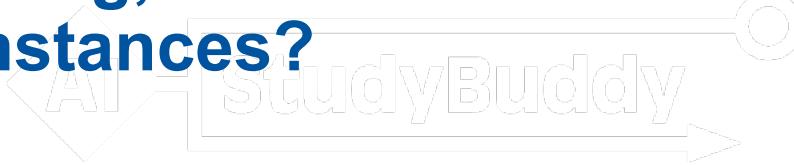
Training & Testing Data



Training & Testing Data



**How to split into training, validation and testing sets
if there are only 20 instances?**



TPS Exercise:

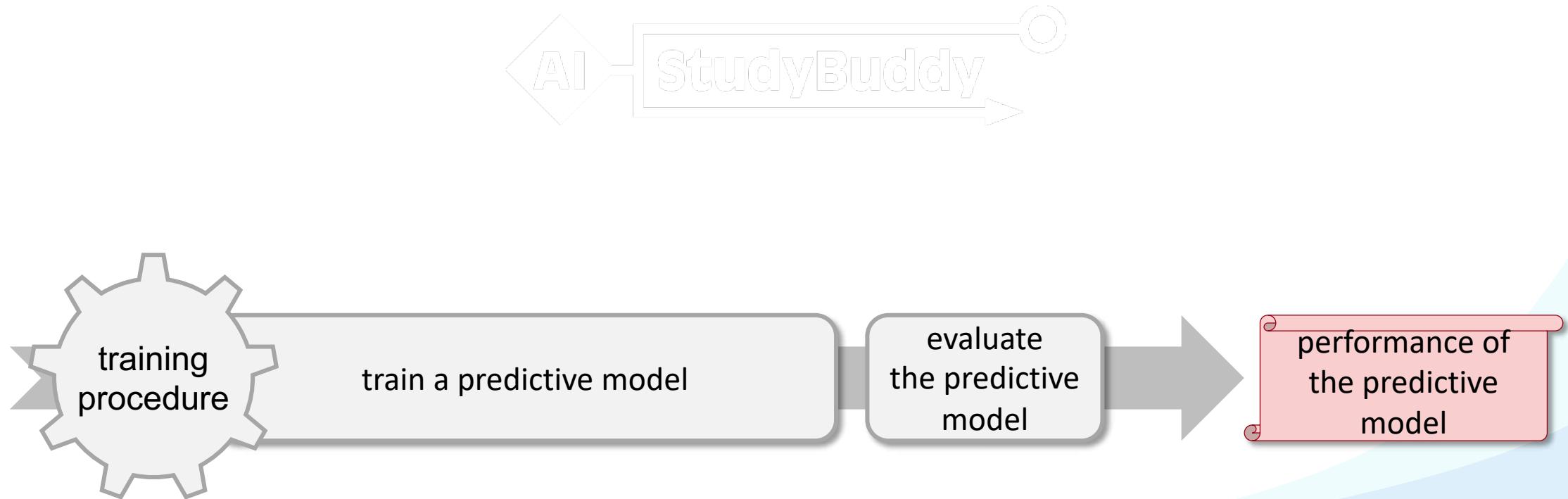
You are using supervised learning to obtain a predictive model, training on a dataset with only 20 instances.



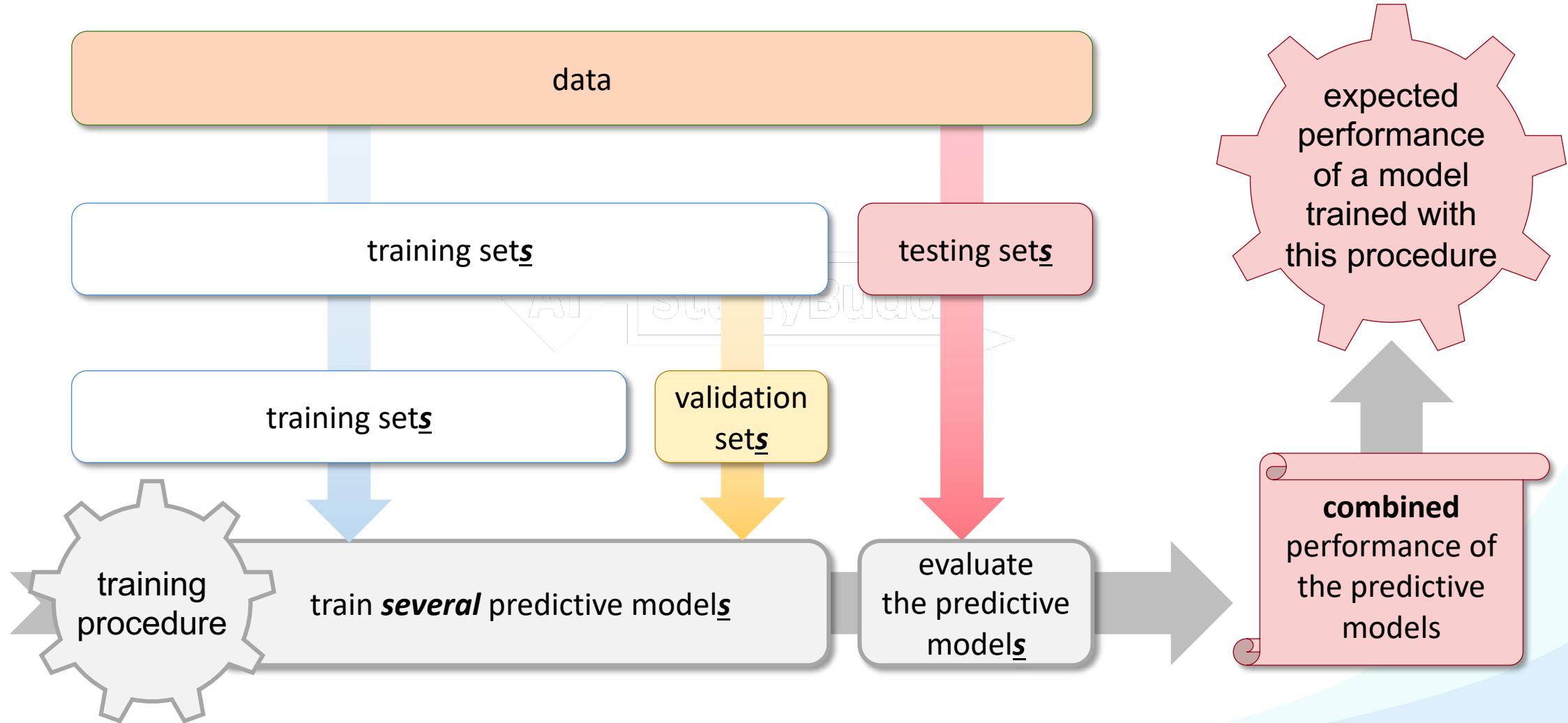
Question: How do you assess the quality of the model?

Dealing with small datasets:

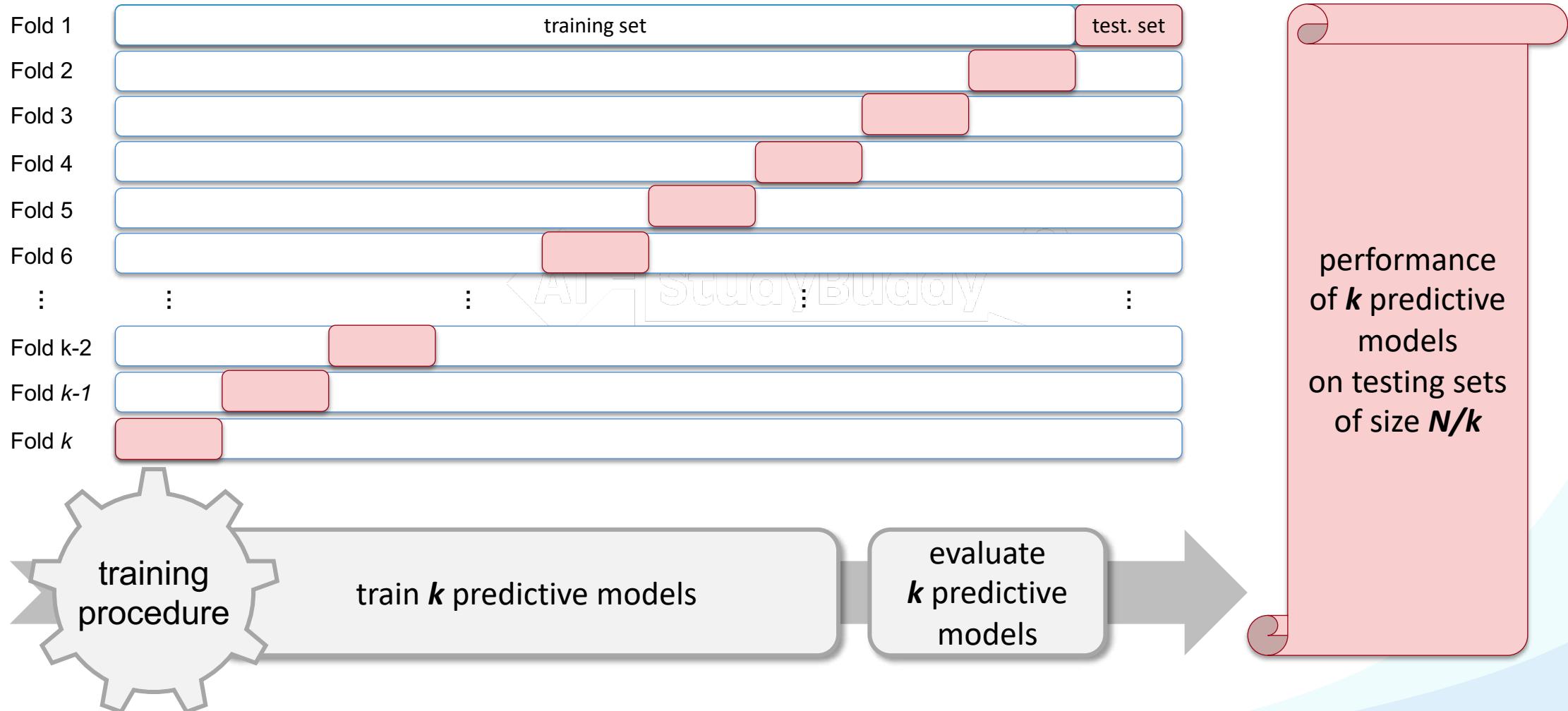
- Splitting into **one** training and **one** testing set is reliable only for sufficiently large data sets
- On small data sets the training, validation or testing set become too small
- Small data set increases danger of a ‘lucky split’ (with most easy instances in the testing set)



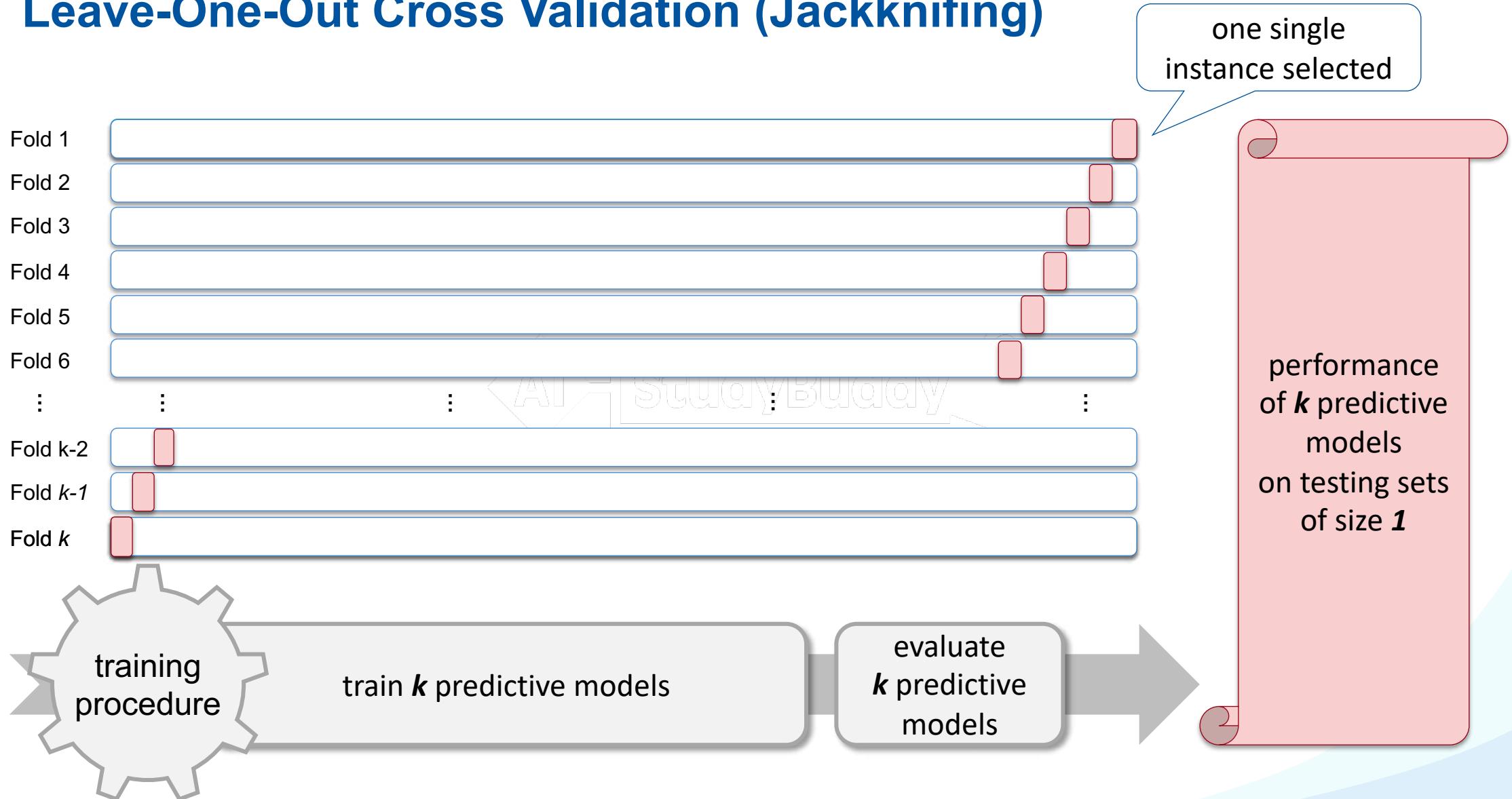
Motivation



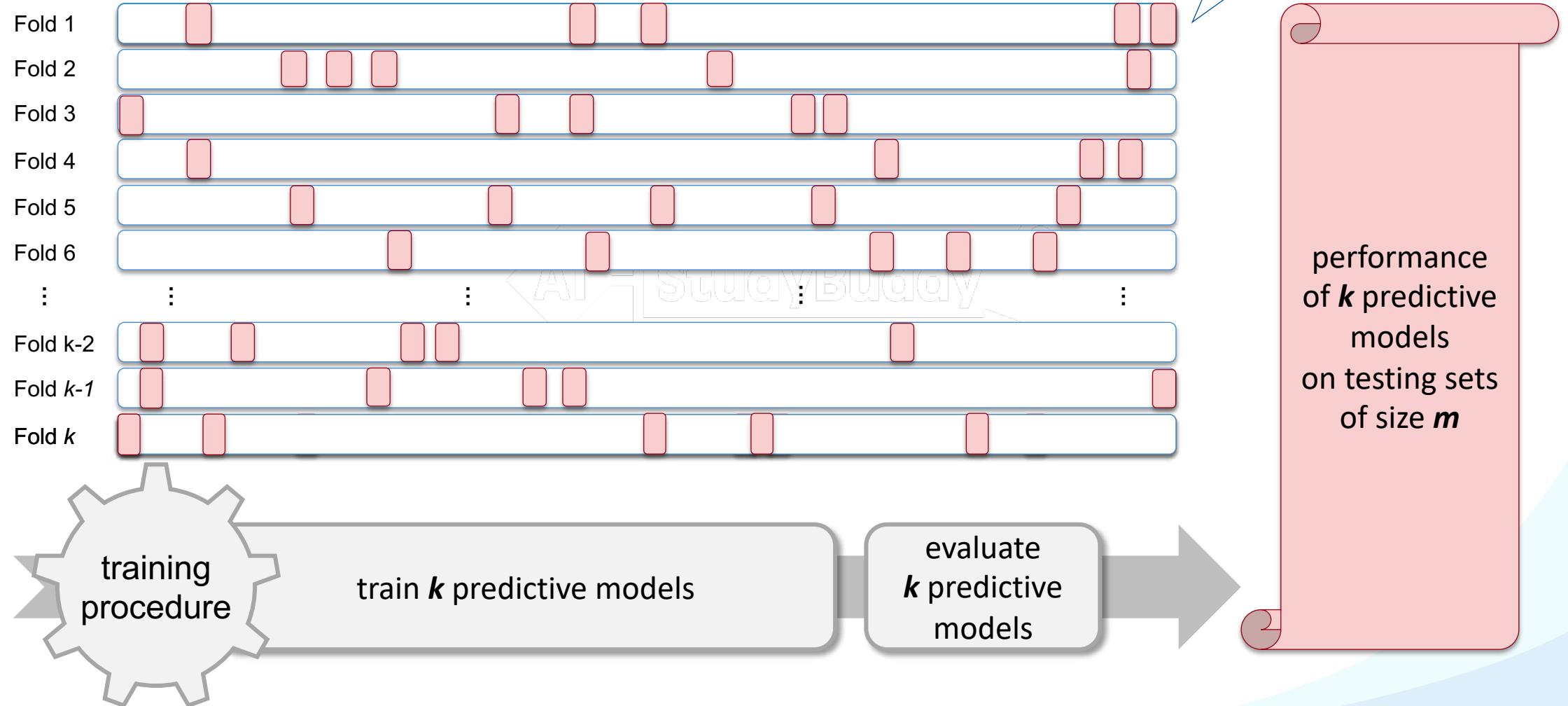
***k*-Fold Cross Validation**



Leave-One-Out Cross Validation (Jackknifing)



Bootstrapping



What problem could arise?

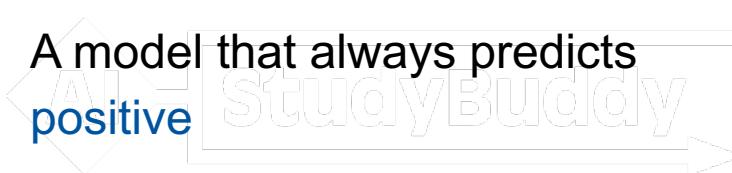
ID	Target Label	Prediction	ID	Target Label	Prediction
1	On Time	On Time	11	On Time	On Time
2	On Time	On Time	12	On Time	On Time
3	On Time	On Time	13	On Time	On Time
4	On Time	On Time	14	On Time	On Time
5	On Time	On Time	15	On Time	On Time
6	On Time	On Time	16	On Time	On Time
7	On Time	On Time	17	On Time	On Time
8	On Time	On Time	18	On Time	On Time
9	On Time	On Time	19	Delayed	On Time
10	On Time	On Time	20	Delayed	On Time

Imbalanced Data

	On Time Prediction	Delayed Prediction
On Time Target Label	18	0
Delayed Target Label	2	0

Motivational Example

- A test set with many (18) **positive** instances and few (2) **negative** instances
- A model that always predicts **positive**



ID	Target Label	Prediction	ID	Target Label	Prediction
1	On Time	On Time	11	On Time	On Time
2	On Time	On Time	12	On Time	On Time
3	On Time	On Time	13	On Time	On Time
4	On Time	On Time	14	On Time	On Time
5	On Time	On Time	15	On Time	On Time
6	On Time	On Time	16	On Time	On Time
7	On Time	On Time	17	On Time	On Time
8	On Time	On Time	18	On Time	On Time
9	On Time	On Time	19	Delayed	On Time
10	On Time	On Time	20	Delayed	On Time

Imbalanced Data

	On Time Prediction	Delayed Prediction
On Time Target Label	18	0
Delayed Target Label	2	0

Motivational Example

- A test set with many (18) **positive** instances and few (2) **negative** instances
- A model that always predicts **positive (= On Time)**

Recall:

$$\text{recall} = \frac{TP}{TP+FN} = \frac{18}{18+0} = 1.0$$

Precision:

$$\text{precision} = \frac{TP}{TP+FP} = \frac{18}{18+2} = \frac{18}{20} = 0.9$$

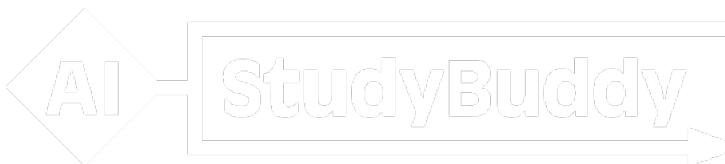
ID	Target Label	Prediction	ID	Target Label	Prediction
1	On Time	On Time	11	On Time	On Time
2	On Time	On Time	12	On Time	On Time
3	On Time	On Time	13	On Time	On Time
4	On Time	On Time	14	On Time	On Time
5	On Time	On Time	15	On Time	On Time
6	On Time	On Time	16	On Time	On Time
7	On Time	On Time	17	On Time	On Time
8	On Time	On Time	18	On Time	On Time
9	On Time	On Time	19	Delayed	On Time
10	On Time	On Time	20	Delayed	On Time

Imbalanced Data

	On Time Prediction	Delayed Prediction
On Time Target Label	18	0
Delayed Target Label	2	0

Average Class Accuracy

Average recall over the elements in the set of possible target feature values $C = \{\text{Delayed}, \text{On Time}\}$



ID	Target Label	Prediction	ID	Target Label	Prediction
1	On Time	On Time	11	On Time	On Time
2	On Time	On Time	12	On Time	On Time
3	On Time	On Time	13	On Time	On Time
4	On Time	On Time	14	On Time	On Time
5	On Time	On Time	15	On Time	On Time
6	On Time	On Time	16	On Time	On Time
7	On Time	On Time	17	On Time	On Time
8	On Time	On Time	18	On Time	On Time
9	On Time	On Time	19	Delayed	On Time
10	On Time	On Time	20	Delayed	On Time

Imbalanced Data

	On Time Prediction	Delayed Prediction
On Time Target Label	18	0
Delayed Target Label	2	0
$recall = \frac{TP}{TP+FN} = \frac{18}{18+0} = 1.0$		
	Delayed Prediction	On Time Prediction
Delayed Target Label	0	2
On Time Target Label	0	18

$$recall = \frac{TP}{TP+FN} = \frac{0}{0+2} = 0.0$$

Average Class Accuracy

Average recall over the elements in the set of possible target feature values $C = \{\text{Delayed}, \text{On Time}\}$



ID	Target Label	Prediction	ID	Target Label	Prediction
1	On Time	On Time	11	On Time	On Time
2	On Time	On Time	12	On Time	On Time
3	On Time	On Time	13	On Time	On Time
4	On Time	On Time	14	On Time	On Time
5	On Time	On Time	15	On Time	On Time
6	On Time	On Time	16	On Time	On Time
7	On Time	On Time	17	On Time	On Time
8	On Time	On Time	18	On Time	On Time
9	On Time	On Time	19	Delayed	On Time
10	On Time	On Time	20	Delayed	On Time

Imbalanced Data

	On Time Prediction	Delayed Prediction
On Time Target Label	18	0
Delayed Target Label	2	0
$recall = \frac{TP}{TP+FN} = \frac{18}{18+0} = 1.0$		
	Delayed Prediction	On Time Prediction
Delayed Target Label	0	2
On Time Target Label	0	18

$$recall = \frac{TP}{TP+FN} = \frac{0}{0+2} = 0.0$$

Average Class Accuracy

Average recall over the elements in the set of possible target feature values $C = \{\text{Delayed}, \text{On Time}\}$



- arithmetic mean:

$$\frac{1}{|C|} \sum_{c \in C} recall_c$$

- harmonic mean:

$$\frac{1}{\frac{1}{|C|} \sum_{c \in C} \frac{1}{recall_c}}$$

ID	Target Label	Prediction	ID	Target Label	Prediction
1	On Time	On Time	11	On Time	On Time
2	On Time	On Time	12	On Time	On Time
3	On Time	On Time	13	On Time	On Time
4	On Time	On Time	14	On Time	On Time
5	On Time	On Time	15	On Time	On Time
6	On Time	On Time	16	On Time	On Time
7	On Time	On Time	17	On Time	On Time
8	On Time	On Time	18	On Time	On Time
9	On Time	On Time	19	Delayed	On Time
10	On Time	On Time	20	Delayed	On Time

Imbalanced Data

	On Time Prediction	Delayed Prediction
On Time Target Label	18	0
Delayed Target Label	2	0
$recall = \frac{TP}{TP+FN} = \frac{18}{18+0} = 1.0$		
	Delayed Prediction	On Time Prediction
Delayed Target Label	0	2
On Time Target Label	0	18

$$recall = \frac{TP}{TP+FN} = \frac{0}{0+2} = 0.0$$

Average Class Accuracy

Average recall over the elements in the set of possible target feature values

$$C = \{\text{Delayed, On Time}\}$$

- arithmetic mean:

$$\frac{1}{|C|} \sum_{c \in C} recall_c = \frac{1}{2}(1 + 0) = 0.5$$

- harmonic mean:

$$\frac{1}{\frac{1}{|C|} \sum_{c \in C} \frac{1}{recall_c}} = \frac{1}{\frac{1}{2}(\frac{1}{1} + \frac{1}{0})} = 0.0$$

$\frac{1}{0} = \infty$ in the limit

ID	Target Label	Prediction	ID	Target Label	Prediction
1	On Time	On Time	11	On Time	On Time
2	On Time	On Time	12	On Time	On Time
3	On Time	On Time	13	On Time	On Time
4	On Time	On Time	14	On Time	On Time
5	On Time	On Time	15	On Time	On Time
6	On Time	On Time	16	On Time	On Time
7	On Time	On Time	17	On Time	On Time
8	On Time	On Time	18	On Time	On Time
9	On Time	On Time	19	Delayed	On Time
10	On Time	On Time	20	Delayed	On Time

What's worse:

**Predicting a flight to be delayed and having it arrive on time,
or predicting it to be on time and find it to be delayed?**



- Does the self-driving car need to stop?
- Should the patient be tested for a severe disease?

➡ FPs and FNs can have (very) different cost!



[2]

Profit Matrix

Example Flight Classification

- Correctly inform customers about a delay:
 - Customers can plan to arrive later
 - **A little** ‘profit’ from less unhappy customers
- Incorrectly inform customers about a delay:
 - Customers arrive too late
 - **Huge** loss of ‘profit’ by unnecessarily delayed flight
- Incorrectly predicting ‘Delayed’ (FN) costs more than incorrectly predicting ‘On Time’ (FP)

		Prediction	
		On Time	Delay
Target Label	On Time	0	-80
	Delay	-10	20

Profit (Utility) Matrix

		Prediction	
		On Time	Delay
		M ₁	
Target Label	On Time	6	3
	Delay	2	9

		Prediction	
		On Time	Delay
		M ₂	
Target Label	On Time	5	0
	Delay	9	6

		Prediction	
		On Time	Delay
		Profit Matrix	
Target Label	On Time	0	-80
	Delay	-10	20

Profit Matrix

		Prediction	
		On Time	Delay
		M ₁	
Target Label	On Time	6	3
	Delay	2	9

		Prediction	
		On Time	Delay
		M ₂	
Target Label	On Time	5	0
	Delay	9	6

		Prediction	
		On Time	Delay
		M ₁	
Target Label	On Time	0	-240
	Delay	-20	180
Profit		-80	

		Prediction	
		On Time	Delay
		M ₂	
Target Label	On Time	0	0
	Delay	-90	120
Profit		30	

Profit Matrix

		Prediction	
		On Time	Delay
		On Time	
Target Label	On Time	0	-80
	Delay	-10	20

Profit Matrix

		Prediction	
		On Time	Delay
		M ₁	
Target Label	On Time	6	3
	Delay	2	9

		Prediction	
		On Time	Delay
		M ₂	
Target Label	On Time	5	0
	Delay	9	6

		Prediction	
		On Time	Delay
		M ₁	
Target Label	On Time	0	-240
	Delay	-20	180
Profit		-80	

		Prediction	
		On Time	Delay
		M ₂	
Target Label	On Time	0	0
	Delay	-90	120
Profit		30	

		Prediction	
		On Time	Delay
		Profit Matrix	
Target Label	On Time	0	-80
	Delay	-10	20

$$\text{profit} = \mathbf{FP} \cdot \mathbf{FP}_{\text{profit}} + \mathbf{TP} \cdot \mathbf{TP}_{\text{profit}} \\ + \mathbf{FN} \cdot \mathbf{FN}_{\text{profit}} + \mathbf{TN} \cdot \mathbf{TN}_{\text{profit}}$$

Key concepts covered today:

- confusion matrix
- performance measures for binary classification
- training, testing and validation sets
- *k*-fold cross validation
- leave-one-out cross validation (jackknife)
- bootstrap sampling validation
- imbalanced data, average class accuracy
- profit (utility) matrix

Preparation for Tuesday:

Investigate the following questions:

- **How to assess predictive models for multi-class classification?**
(> 2 target classes, e.g., on time, mildly delayed, severely delayed)
- **How to assess predictive models for regression tasks?**
(predictions = numbers, e.g., minutes of delay)

(We will use this for TPS exercises with the T part done before class.)

Sources

- [1] Erik Heddema on Unsplash, Unsplash License,
(https://unsplash.com/de/fotos/k_kz0jmyOmE)
- [2] Matt C on Unsplash, Unsplash License,
(<https://unsplash.com/de/fotos/ubHRHM37ddE>)

Elements of Machine Learning & Data Science

Winter semester 2023/24

Evaluation of Supervised Learning (2)

Prof. Holger Hoos (partially based on material from Wil van der Aalst)

Key questions:

- How good is an ML model?
- How good could an ML model be?



You have used supervised ML to train a predictive model.



Question: How do you assess the quality of the model?

NB: So far, focus on binary classification problems.

Key concepts covered last class:

- confusion matrix
- performance measures for binary classification
- training, testing and validation sets
- *k*-fold cross validation
- leave-one-out cross validation (jackknife)
- bootstrap sampling validation
- imbalanced data, average class accuracy
- profit (utility) matrix

Preparation for today:

Investigate the following questions:

- How to assess predictive models for multi-class classification?**
(> 2 target classes, e.g., on time, mildly delayed, severely delayed)
- How to assess predictive models for regression tasks?**
(predictions = numbers, e.g., minutes of delay)

(We will use this for TPS exercises with the T part done before class.)

TPS Exercise (T part = done as homework)

Question:

**How to assess predictive models for multi-class classification?
(> 2 target classes, e.g., on time, mildly delayed, severely delayed)**

Multinomial Targets

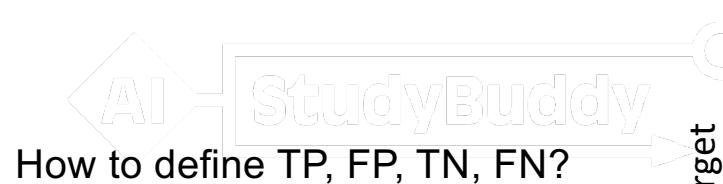
ID	Target Label	Prediction
1	On Time	Delayed
2	On Time	Delayed
3	Delayed	Canceled
4	Canceled	On Time
5	Delayed	Delayed
6	On Time	On Time
7	Delayed	Delayed
8	Canceled	Canceled
9	On Time	On Time
10	On Time	On Time

- More than two possible values for the target feature
- How to compute confusion matrix-based performance measures?



Multinomial Targets

ID	Target Label	Prediction
1	On Time	Delayed
2	On Time	Delayed
3	Delayed	Canceled
4	Canceled	On Time
5	Delayed	Delayed
6	On Time	On Time
7	Delayed	Delayed
8	Canceled	Canceled
9	On Time	On Time
10	On Time	On Time



		Prediction		
		On Time	Delayed	Canceled
Target	On Time	3	2	0
	Delayed	0	2	1
Canceled	1	0	1	

Multinomial Targets

For each possible target label value:

- Consider this label as **positive**, all others as **negative**
- Compute TP, TN, FP, FN as before
- Compute performance measures as before

		Prediction		
		On Time	Delayed	Canceled
Target	On Time	3	2	0
	Delayed	0	2	1
Canceled	On Time	1	0	1
	Delayed	0	1	0

Multinomial Targets

For each possible target label value:

- Consider this label as **positive**, all others as **negative**
- Compute TP, TN, FP, FN as before
- Compute performance measures as before

On Time → Positive

Delayed, Canceled → Negative

		Prediction		
		On Time	Delayed	Canceled
Target	On Time	3	2	0
	Delayed	0	2	1
Canceled	Canceled	1	0	1

Multinomial Targets

For each possible target label value:

- Consider this label as **positive**, all others as **negative**
- Compute TP, TN, FP, FN as before
- Compute performance measures as before

On Time → Positive

Delayed, Canceled → Negative

$$TP=3, FN=2+0=2, FP=0+1=1, TN=2+1+0+1=4$$

		Prediction		
		On Time	Delayed	Canceled
Target	On Time	3	2	0
	Delayed	0	2	1
Canceled	On Time	1	4	1
	Delayed	1	0	1

Multinomial Targets

For each possible target label value:

- Consider this label as **positive**, all others as **negative**
- Compute TP, TN, FP, FN as before
- Compute performance measures as before

On Time → Positive

Delayed, Canceled → Negative

$$precision_{\text{on time}} = \frac{TP_{\text{on time}}}{TP_{\text{on time}} + FP_{\text{on time}}} = \frac{3}{3 + (0 + 1)} = \frac{3}{4}$$

$$recall_{\text{on time}} = \frac{TP_{\text{on time}}}{TP_{\text{on time}} + FN_{\text{on time}}} = \frac{3}{3 + (2 + 0)} = \frac{3}{5}$$

		Prediction		
		On Time	Delayed	Canceled
Target	On Time	3	2	0
	Delayed	0	2	1
	Canceled	1	0	1

Multinomial Targets

For each possible target label value:

- Consider this label as **positive**, all others as **negative**
- Compute TP, TN, FP, FN as before
- Compute performance measures as before



Delayed → Positive

On Time, Canceled → Negative

$$precision_{\text{delayed}} = \frac{TP_{\text{delayed}}}{TP_{\text{delayed}} + FP_{\text{delayed}}} = \frac{2}{2 + (2 + 0)} = \frac{1}{2}$$

$$recall_{\text{delayed}} = \frac{TP_{\text{delayed}}}{TP_{\text{delayed}} + FN_{\text{delayed}}} = \frac{2}{2 + (0 + 1)} = \frac{2}{3}$$

		Prediction		
		On Time	Delayed	Canceled
Target	On Time	3	2	0
	Delayed	0	2	1
Canceled	On Time	1	0	1
	Delayed	0	1	0

Multinomial Targets

For each possible target label value:

- Consider this label as **positive**, all others as **negative**
- Compute TP, TN, FP, FN as before
- Compute performance measures as before



Canceled → Positive

On Time, Delayed → Negative

$$precision_{canceled} = \frac{TP_{canceled}}{TP_{canceled} + FP_{canceled}} = \frac{1}{1 + (0 + 1)} = \frac{1}{2}$$

$$recall_{canceled} = \frac{TP_{canceled}}{TP_{canceled} + FN_{canceled}} = \frac{1}{1 + (1 + 0)} = \frac{1}{2}$$

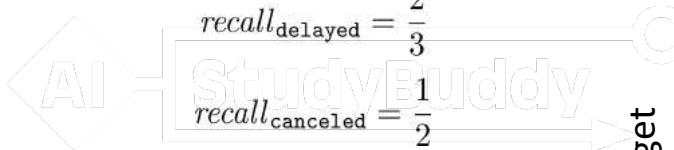
		Prediction		
		On Time	Delayed	Canceled
Target	On Time	3	2	0
	Delayed	0	2	1
Canceled	On Time	1	0	1
	Delayed	0	1	0

Multinomial Targets

Individual recalls can be combined using **average class accuracy** (harmonic mean):

K is the number of label values

$$\begin{aligned} & \frac{1}{K} \cdot \left(\sum_{k=1}^K \left(\frac{1}{recall_k} \right) \right)^{-1} \\ \Rightarrow & \frac{1}{3} \cdot \left(\frac{1}{recall_{\text{on time}}} + \frac{1}{recall_{\text{delayed}}} + \frac{1}{recall_{\text{canceled}}} \right)^{-1} \\ = & \frac{18}{31} \approx 0.58 \end{aligned}$$



		Prediction		
		On Time	Delayed	Canceled
On Time	On Time	3	2	0
	Delayed	0	2	1
Canceled	Canceled	1	0	1

TPS Exercise (T part = done as homework)

Question:



How to assess predictive models for regression tasks?

(predictions = numbers, e.g., minutes of delay)

Reminder: Error Functions

Sum of squared errors $\frac{1}{2} \sum_{i=1}^N ((t_i - \mathbb{M}(\mathbf{x}_i))^2)$

Mean squared error $\frac{1}{N} \sum_{i=1}^N ((t_i - \mathbb{M}(\mathbf{x}_i))^2)$

Root mean squared error $\sqrt{\frac{1}{N} \sum_{i=1}^N ((t_i - \mathbb{M}(\mathbf{x}_i))^2)}$

Mean absolute error $\frac{1}{N} \sum_{i=1}^N |t_i - \mathbb{M}(\mathbf{x}_i)|$

For the i th instance,

t_i is the true target value and
 $\mathbb{M}(\mathbf{x}_i)$ is the predicted value.

Coefficient of Determination (R^2)

- Compare model performance with the model that always guesses the average (baseline)
- Close to 0 → no better than guessing the average
- Close to 1 → all predictions are perfect
- Cross validation as before



$$R^2 = 1 - \frac{\text{sum of squared errors}}{\text{total sum of squares}}$$

$$\text{sum of squared errors} = \sum_{i=1}^N ((t_i - \mathbb{M}(\mathbf{x}_i))^2)$$

$$\text{total sum of squares} = \sum_{i=1}^N (t_i - \bar{t})^2$$

\bar{t} is the mean of all target values:
 $\frac{1}{N} \sum_{j=1}^N t_j$

Coefficient of Determination (R^2) – Example

$$R^2 = 1 - \frac{\text{sum of squared errors}}{\text{total sum of squares}}$$

$$\text{sum of squared errors} = \sum_{i=1}^N ((t_i - \mathbb{M}(\mathbf{x}_i))^2)$$

$$\text{total sum of squares} = \sum_{i=1}^N (t_i - \bar{t})^2$$

ID	Delay [min]	Predicted Delay [min]	$t_i - \mathbb{M}(\mathbf{x}_i)$	$(t_i - \mathbb{M}(\mathbf{x}_i))^2$	$t_i - \bar{t}$	$(t_i - \bar{t})^2$
1	34	15				
2	-6	-9				
3	3	2				
4	9	8				

Coefficient of Determination (R^2) – Example

$$R^2 = 1 - \frac{\text{sum of squared errors}}{\text{total sum of squares}}$$

$$\text{sum of squared errors} = \sum_{i=1}^N ((t_i - \mathbb{M}(\mathbf{x}_i))^2)$$

$$\text{total sum of squares} = \sum_{i=1}^N (t_i - \bar{t})^2$$

ID	Delay [min]	Predicted Delay [min]	$t_i - \mathbb{M}(\mathbf{x}_i)$	$(t_i - \mathbb{M}(\mathbf{x}_i))^2$	$t_i - \bar{t}$	$(t_i - \bar{t})^2$
1	34	15	19	361	24	576
2	-6	-9	3	9	-16	256
3	3	2	1	1	-7	49
4	9	8	1	1	-1	1
Mean:	10			Sum: 372	Sum: 882	

Coefficient of Determination (R^2) – Example

$$R^2 = 1 - \frac{\text{sum of squared errors}}{\text{total sum of squares}}$$

$$\text{sum of squared errors} = \sum_{i=1}^N ((t_i - \mathbb{M}(\mathbf{x}_i))^2) = \frac{1}{2} \cdot 372 = 186$$

$$\text{total sum of squares} = \sum_{i=1}^N (t_i - \bar{t})^2 = \frac{1}{2} \cdot 882 = 441$$

ID	Delay [min]	Predicted Delay [min]	$t_i - \mathbb{M}(\mathbf{x}_i)$	$(t_i - \mathbb{M}(\mathbf{x}_i))^2$	$t_i - \bar{t}$	$(t_i - \bar{t})^2$
1	34	15	19	361	24	576
2	-6	-9	3	9	-16	256
3	3	2	1	1	-7	49
4	9	8	1	1	-1	1
Mean:	10			Sum: 372	Sum: 882	

Coefficient of Determination (R^2) – Example

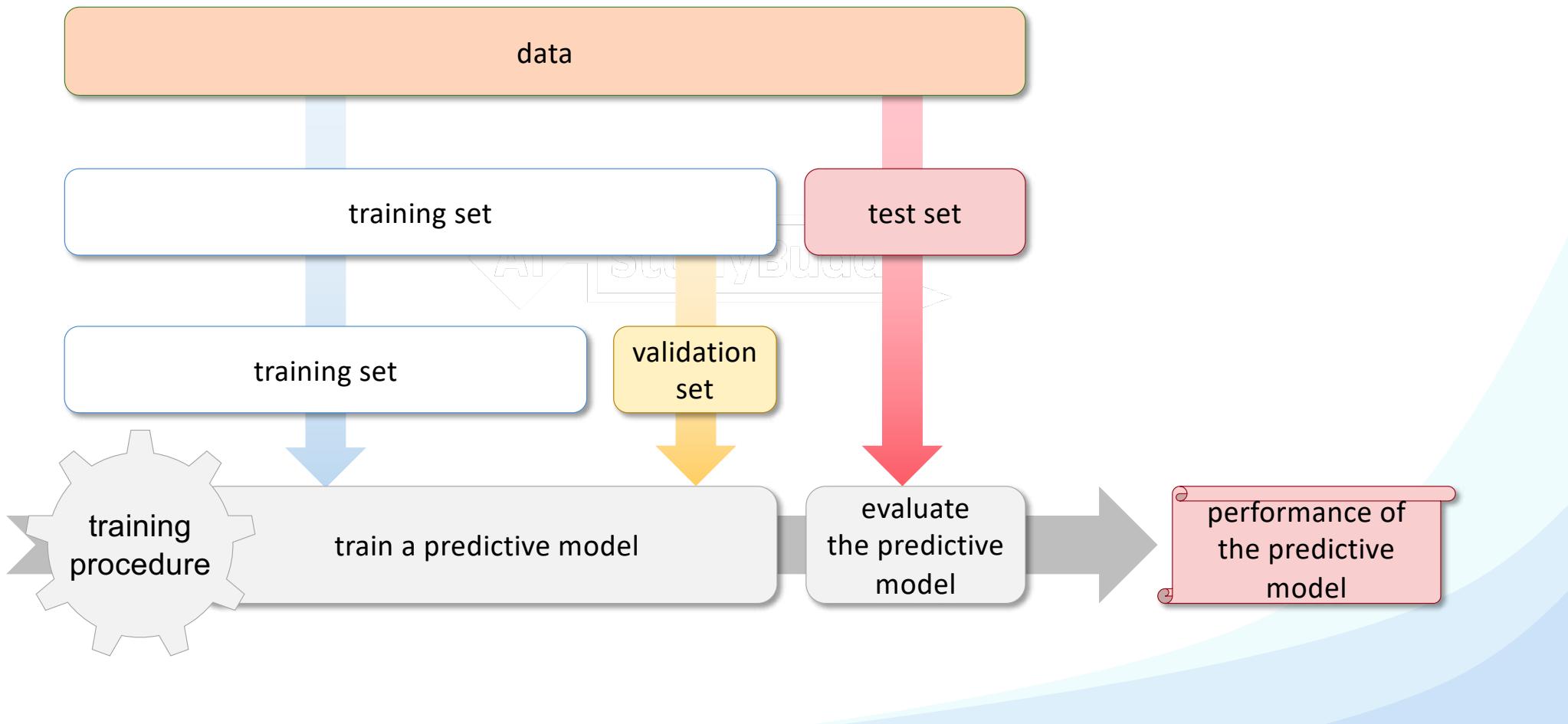
$$R^2 = 1 - \frac{\text{sum of squared errors}}{\text{total sum of squares}} = 1 - \frac{186}{441} \approx 0.42$$

$$\text{sum of squared errors} = \sum_{i=1}^N ((t_i - \mathbb{M}(\mathbf{x}_i))^2) = \frac{1}{2} \cdot 372 = 186$$

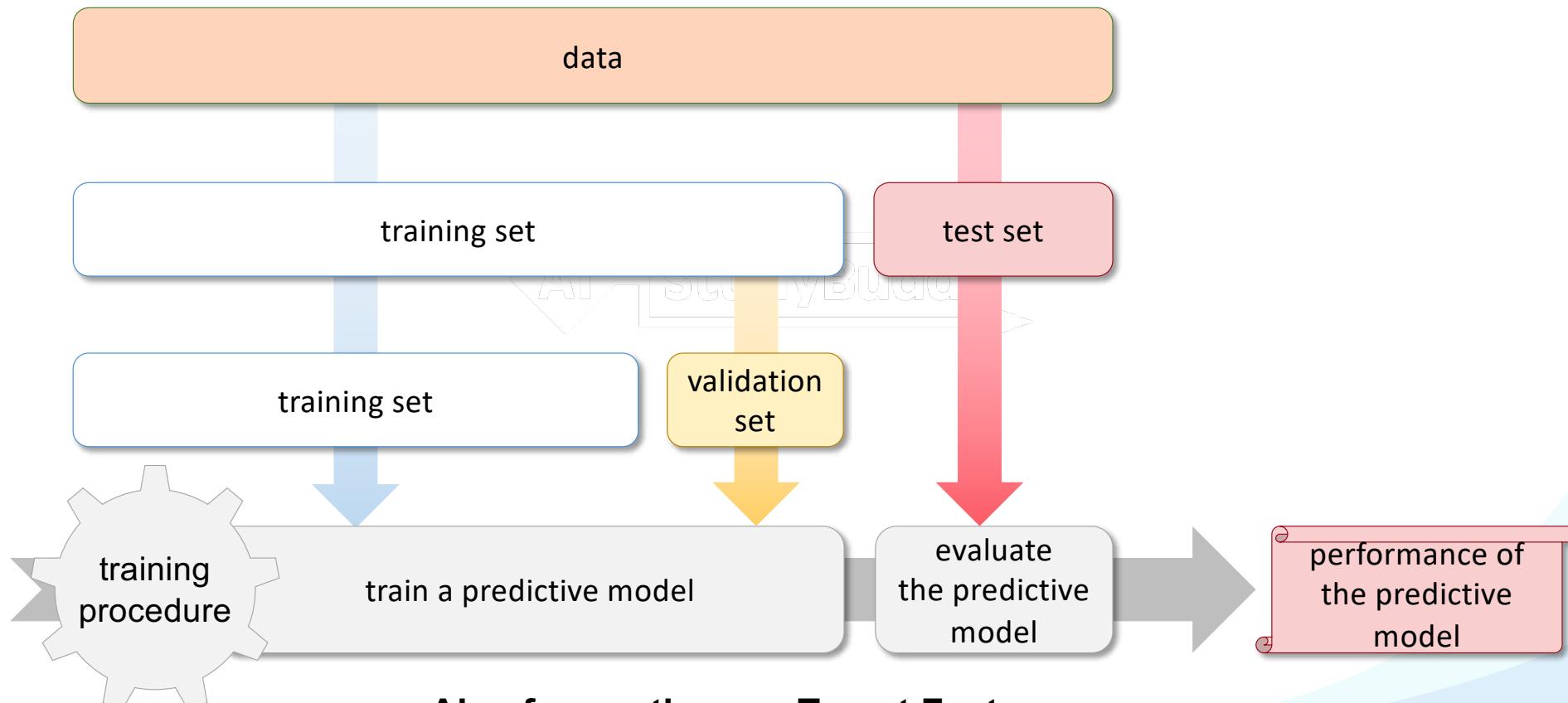
$$\text{total sum of squares} = \sum_{i=1}^N (t_i - \bar{t})^2 = \frac{1}{2} \cdot 882 = 441$$

ID	Delay [min]	Predicted Delay [min]	$t_i - \mathbb{M}(\mathbf{x}_i)$	$(t_i - \mathbb{M}(\mathbf{x}_i))^2$	$t_i - \bar{t}$	$(t_i - \bar{t})^2$
1	34	15	19	361	24	576
2	-6	-9	3	9	-16	256
3	3	2	1	1	-7	49
4	9	8	1	1	-1	1
Mean:	10			Sum: 372	Sum: 882	

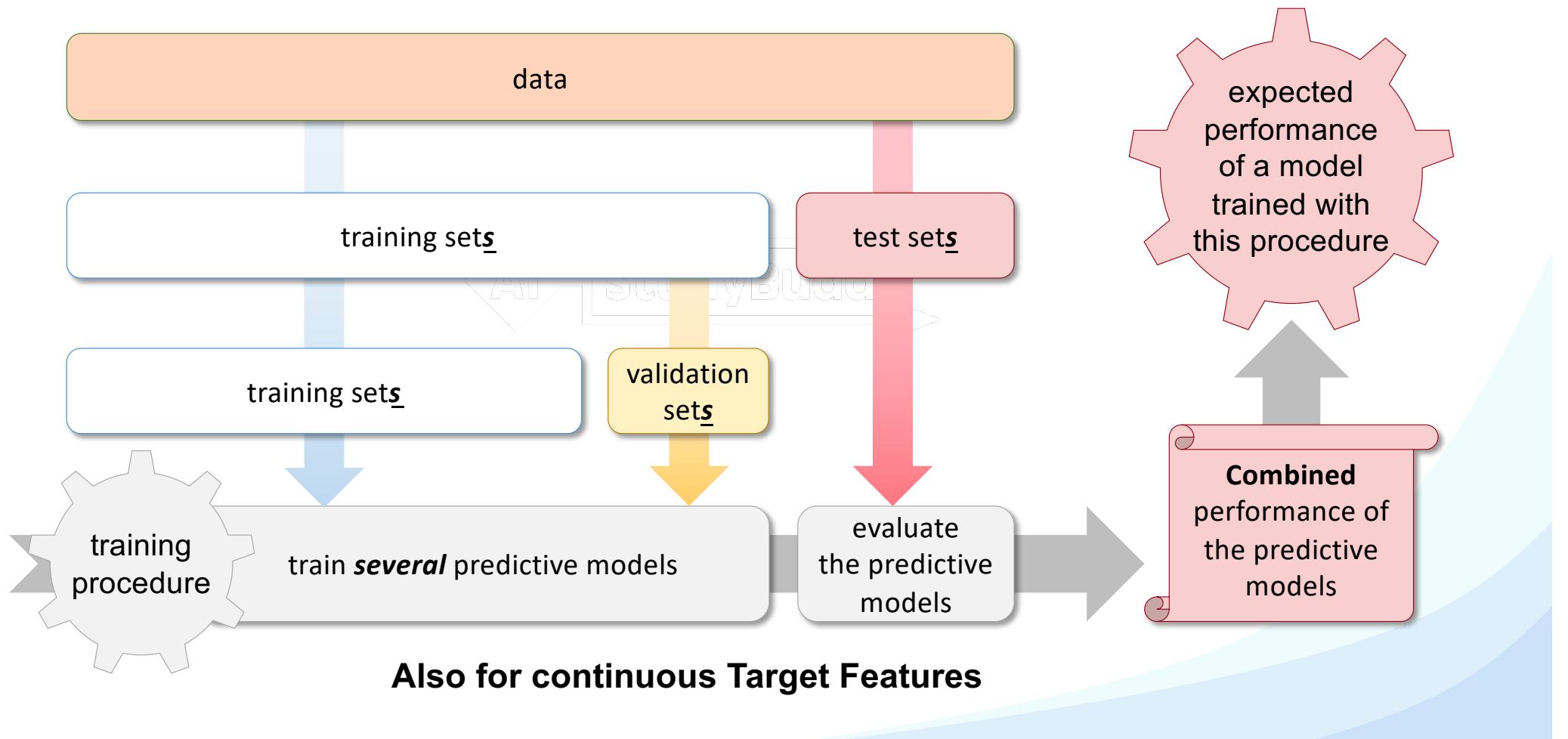
Reminder



Reminder



Reminder (2)



TPS Exercise

You have used supervised ML to train a predictive model for a binary classification problem. The model gives you a numerical prediction score between 0 and 1.

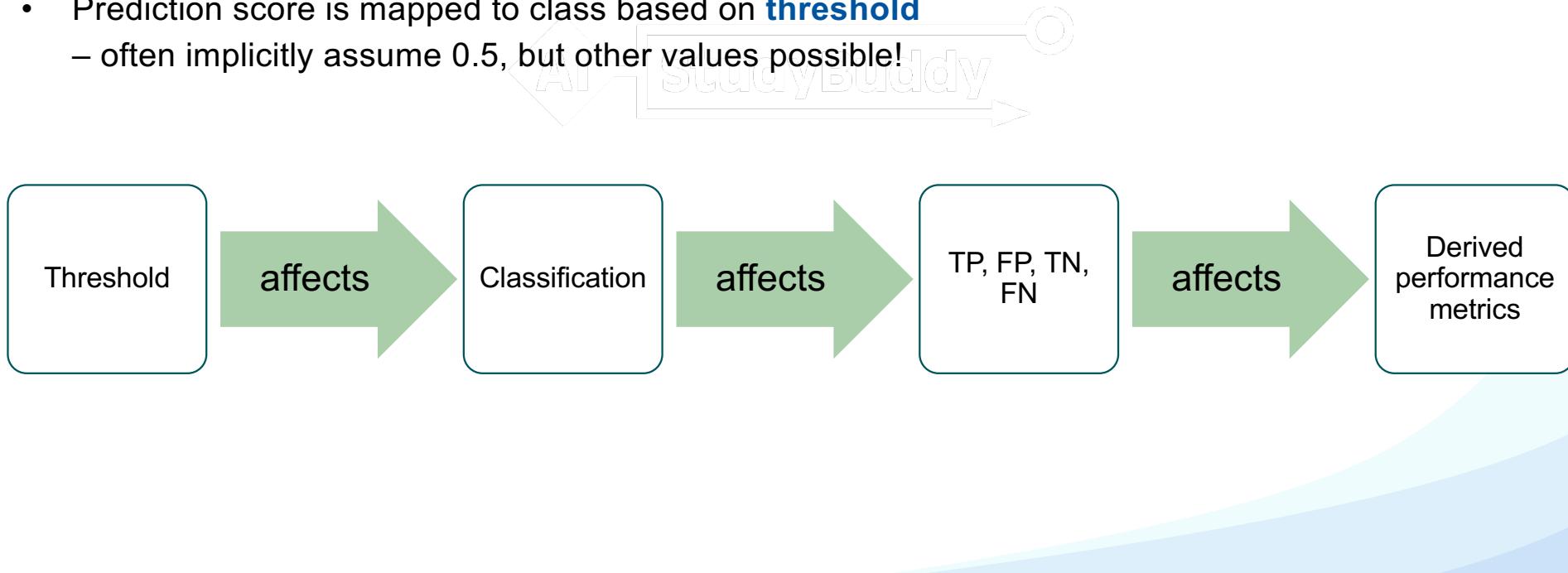


Question:

How to assess the quality of the model?

Motivation

- Models often return **prediction score** representing how ‘sure’ they are about the target feature (e.g., logistic regression, decision trees, Bayes, NNs)
- Assume prediction score $\in [0,1]$
- Prediction score is mapped to class based on **threshold**
 - often implicitly assume 0.5, but other values possible!



Changing the Threshold - Example

Prediction			
0.25	On Time	Delayed	TPR = 1
Target	On Time	Delayed	FPR = 0.8
Delayed	5	0	TNR = 1 - FPR
Misclassification Rate:	4	1	FNR = 1 - TPR



ID	Target Label	Prediction Score	Prediction for various thresholds		
			0.25	0.5	0.75
1	Delayed	0.12	Delayed		
2	Delayed	0.28	On Time		
3	Delayed	0.30	On Time		
4	Delayed	0.29	On Time		
5	On Time	0.43	On Time		
6	Delayed	0.54	On Time		
7	On Time	0.63	On Time		
8	On Time	0.72	On Time		
9	On Time	0.84	On Time		
10	On Time	0.99	On Time		

Changing the Threshold - Example

Prediction				
Target	0.25	On Time	Delayed	TPR = 1
On Time	5	0		FPR = 0.8
Delayed	4	1		TNR = 1 - FPR
Misclassification Rate:		0.4		FNR = 1 - TPR

Prediction				
Target	0.5	On Time	Delayed	TPR = 0.8
On Time	4	1		FPR = 0.2
Delayed	1	4		
Misclassification Rate:		0.2		

ID	Target Label	Prediction Score	Prediction for various thresholds		
			0.25	0.5	0.75
1	Delayed	0.12	Delayed	Delayed	
2	Delayed	0.28	On Time	Delayed	
3	Delayed	0.30	On Time	Delayed	
4	Delayed	0.29	On Time	Delayed	
5	On Time	0.43	On Time	Delayed	
6	Delayed	0.54	On Time	On Time	
7	On Time	0.63	On Time	On Time	
8	On Time	0.72	On Time	On Time	
9	On Time	0.84	On Time	On Time	
10	On Time	0.99	On Time	On Time	

Changing the Threshold - Example

Prediction				
Target	0.25	On Time	Delayed	TPR = 1
On Time	5	0		FPR = 0.8
Delayed	4	1		TNR = 1 - FPR
Misclassification Rate:		0.4		FNR = 1 - TPR
Prediction				
Target	0.5	On Time	Delayed	TPR = 0.8
On Time	4	1		FPR = 0.2
Delayed	1	4		
Misclassification Rate:		0.2		
Prediction				
Target	0.75	On Time	Delayed	TPR = 0.4
On Time	2	3		FPR = 0
Delayed	0	5		
Misclassification Rate:		0.3		

ID	Target Label	Prediction Score	Prediction for various thresholds		
			0.25	0.5	0.75
1	Delayed	0.12	Delayed	Delayed	Delayed
2	Delayed	0.28	On Time	Delayed	Delayed
3	Delayed	0.30	On Time	Delayed	Delayed
4	Delayed	0.29	On Time	Delayed	Delayed
5	On Time	0.43	On Time	Delayed	Delayed
6	Delayed	0.54	On Time	On Time	Delayed
7	On Time	0.63	On Time	On Time	Delayed
8	On Time	0.72	On Time	On Time	Delayed
9	On Time	0.84	On Time	On Time	On Time
10	On Time	0.99	On Time	On Time	On Time

Receiver Operating Characteristic (ROC) Curve – Example

Prediction		
Target	On Time	Delayed
0.25	5	0
On Time	4	1
Misclassification Rate:	0.4	

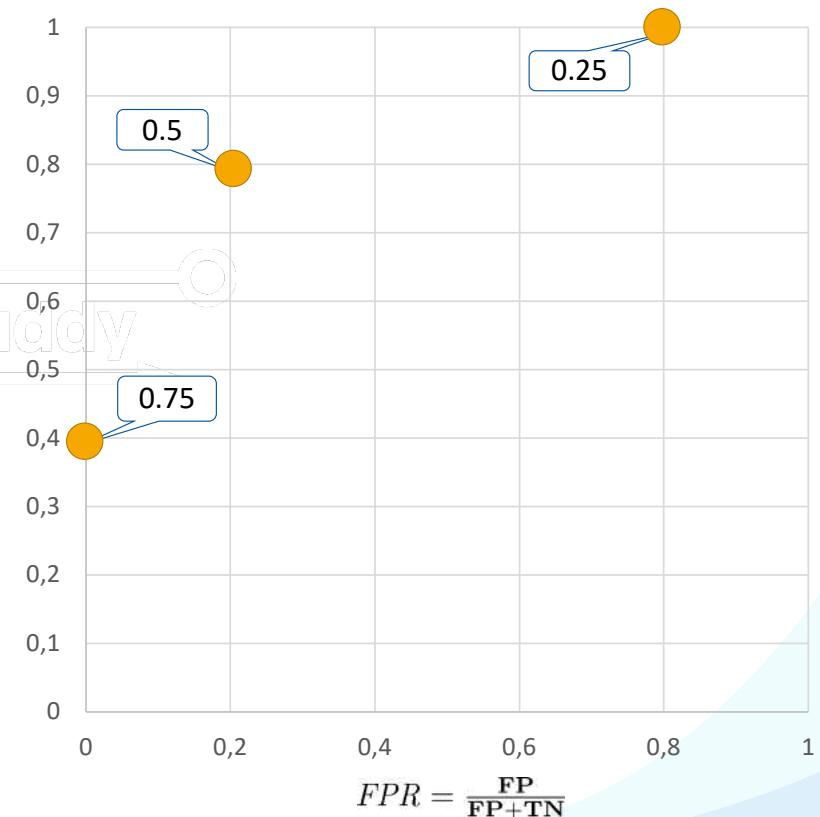
$TPR = 1$
 $FPR = 0.8$
 $TNR = 1 - FPR$
 $FNR = 1 - TPR$

Prediction		
Target	On Time	Delayed
0.5	4	1
On Time	1	4
Misclassification Rate:	0.2	

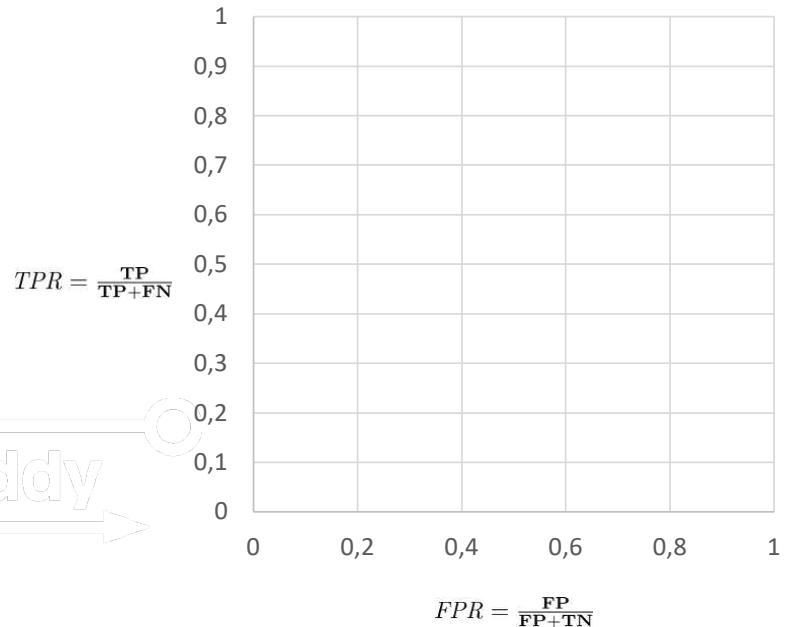
$TPR = 0.8$
 $FPR = 0.2$

Prediction		
Target	On Time	Delayed
0.75	2	3
On Time	0	5
Misclassification Rate:	0.3	

$TPR = 0.4$
 $FPR = 0$



TPS Exercise



Questions:

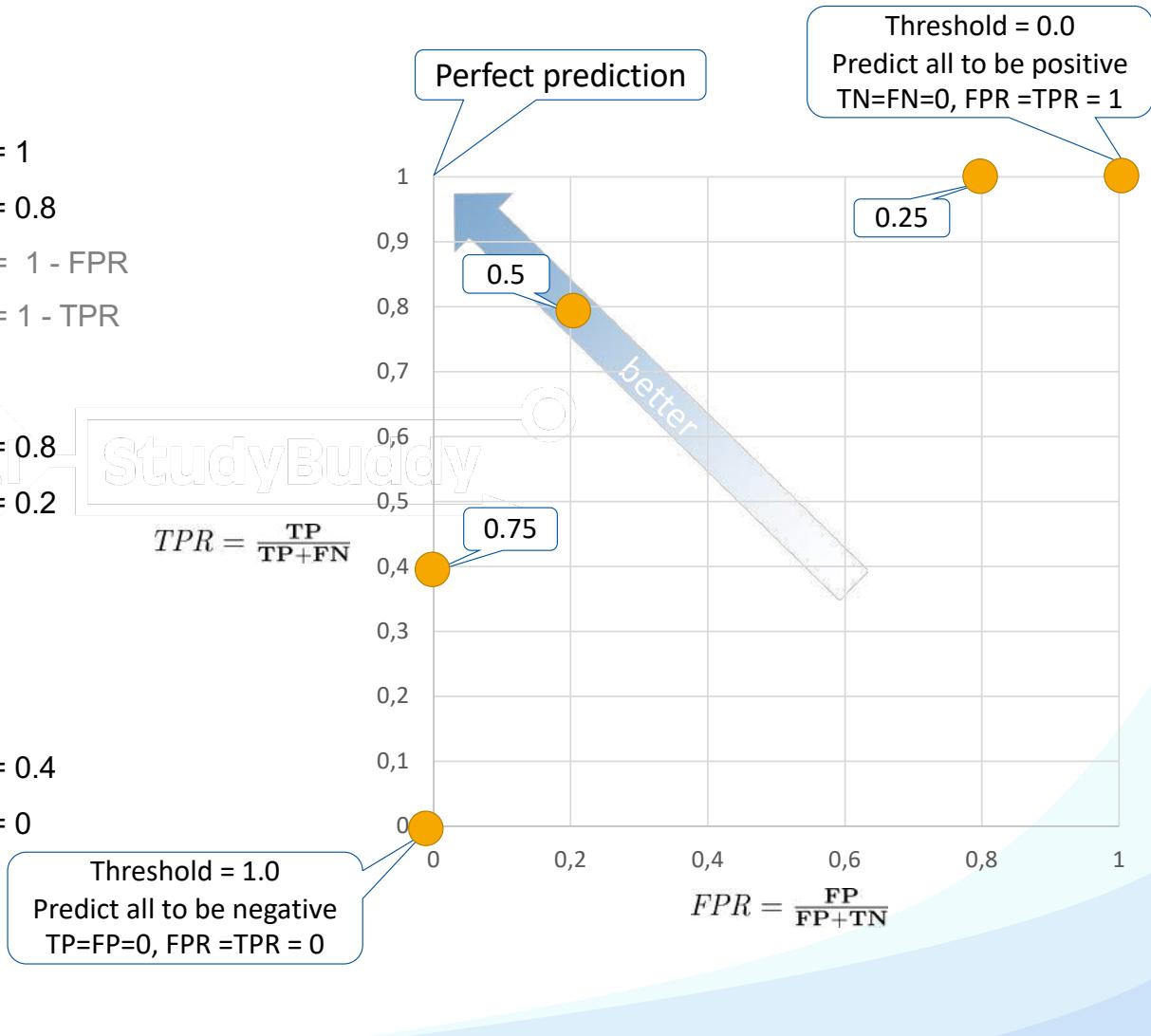
- 1) What does an ideal ROC Curve look like?
- 2) What about worst-case ROC Curve?

ROC Curve – Example

Prediction			
Target	On Time	Delayed	
	0.25	5	0
Misclassification Rate:		0.4	

Prediction			
Target	On Time	Delayed	
	0.5	4	1
Misclassification Rate:		0.2	

Prediction			
Target	On Time	Delayed	
	0.75	2	3
Misclassification Rate:		0.3	

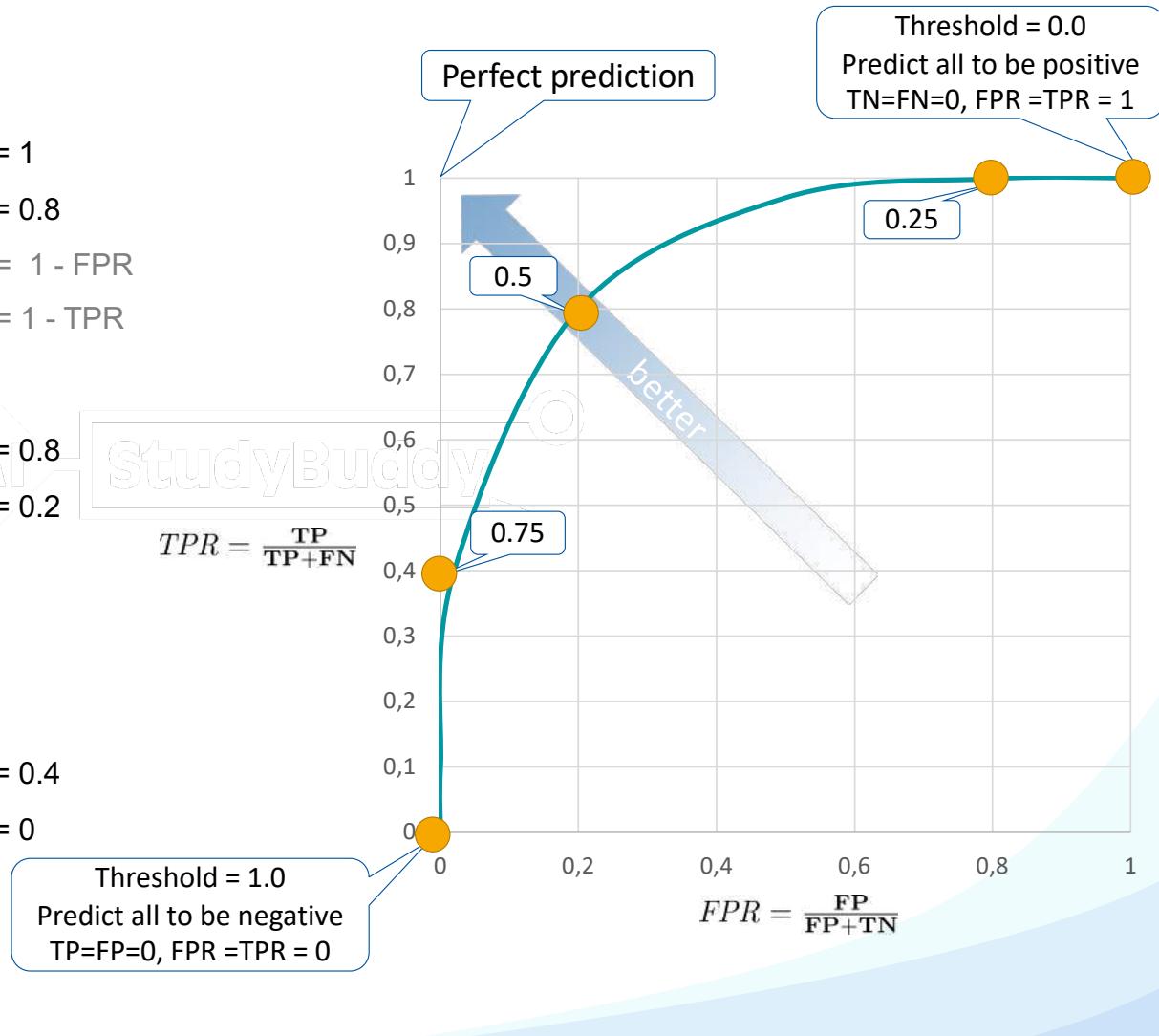


ROC Curve – Example

Prediction			
Target	On Time	Delayed	
	0.25	0	TPR = 1 FPR = 0.8 TNR = 1 - FPR FNR = 1 - TPR
On Time	5	0	
Delayed	4	1	
Misclassification Rate:	0.4		

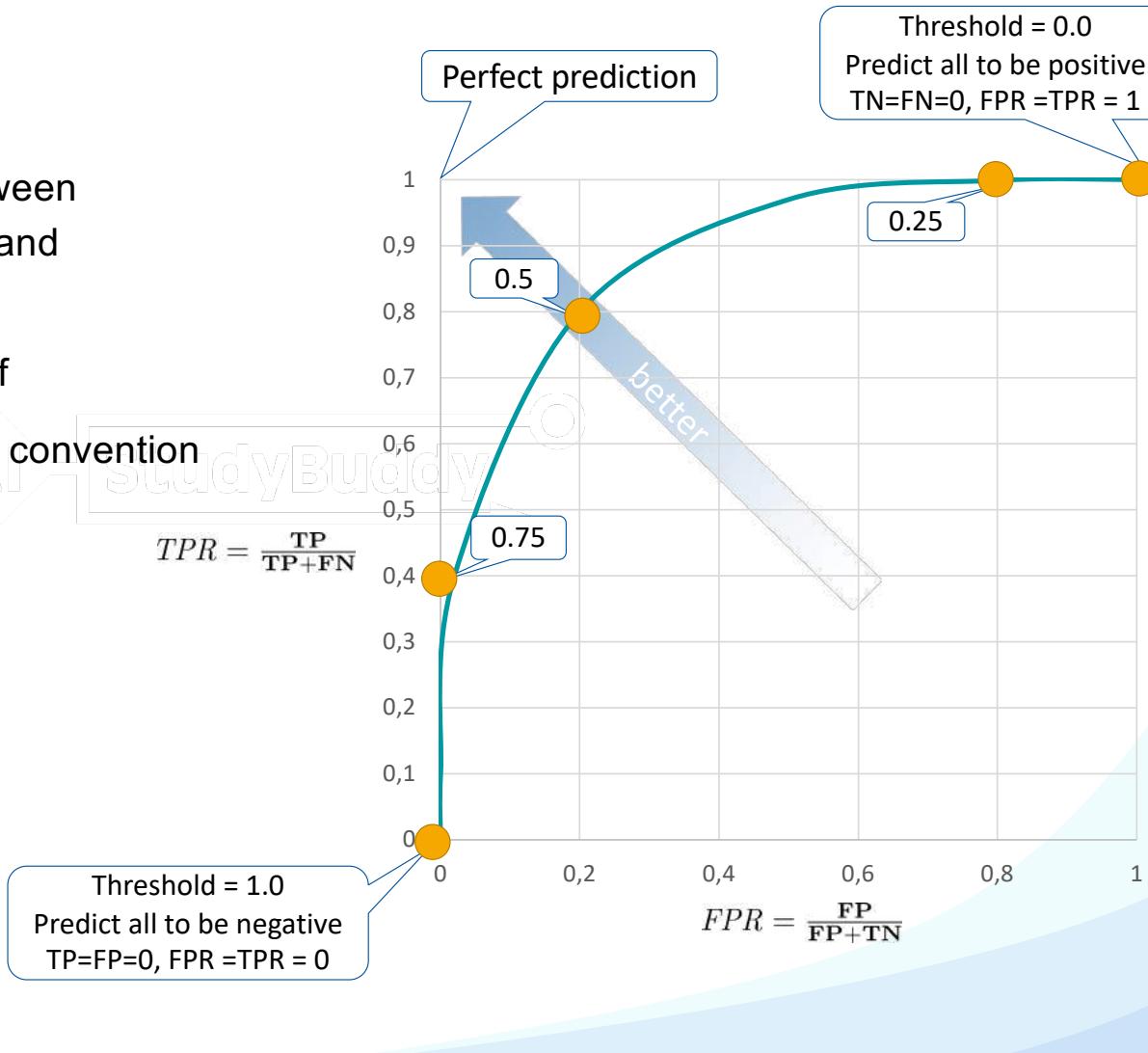
Prediction			
Target	On Time	Delayed	
	0.5	1	TPR = 0.8 FPR = 0.2
On Time	4	1	
Delayed	1	4	
Misclassification Rate:	0.2		

Prediction			
Target	On Time	Delayed	
	0.75	3	TPR = 0.4 FPR = 0
On Time	2	3	
Delayed	0	5	
Misclassification Rate:	0.3		



ROC Curve – Example

- Threshold controls **trade-off** between accuracy for positive predictions and accuracy for negative predictions
- ROC curve captures this trade-off
- Focus on positive (TPR, FPR) by convention



ROC Curve – Beating Random Guessing

Data set with N instances:

Fraction of q **positive** instances,
fraction of $1-q$ **negative** instances

Prediction Model:

Guess **positive** with probability p and
negative with probability $1-p$



ROC Curve – Beating Random Guessing

Data set with N instances:

Fraction of q instances is **positive**,
fraction of $1-q$ instances is **negative**

Prediction Model:

Guess **positive** with probability p and
negative with probability $1-p$



Expected Performance:

$$\mathbf{TP} = p \cdot q \cdot N$$

$$\mathbf{TN} = (1 - p) \cdot (1 - q) \cdot N$$

$$\mathbf{FP} = p \cdot (1 - q) \cdot N$$

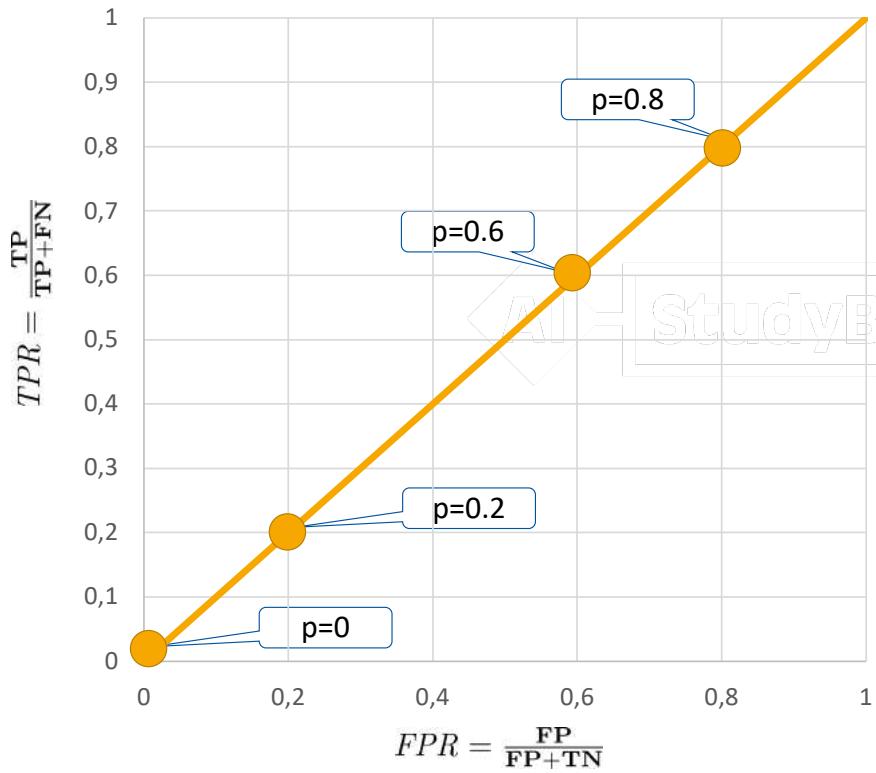
$$\mathbf{FN} = (1 - p) \cdot q \cdot N$$

$$\mathbf{TPR} = \frac{\mathbf{TP}}{\mathbf{TP} + \mathbf{FN}} = \frac{p \cdot q \cdot N}{p \cdot q \cdot N + (1-p) \cdot q \cdot N} = p$$

$$\mathbf{FPR} = \frac{\mathbf{FP}}{\mathbf{TN} + \mathbf{FP}} = \frac{p \cdot (1-q) \cdot N}{(1-p) \cdot (1-q) \cdot N + p \cdot (1-q) \cdot N} = p$$

→ Performance is independent of q , N !

ROC Curve – Beating Random Guessing



Expected Performance:

$$\mathbf{TP} = p \cdot q \cdot N$$

$$\mathbf{TN} = (1 - p) \cdot (1 - q) \cdot N$$

$$\mathbf{FP} = p \cdot (1 - q) \cdot N$$

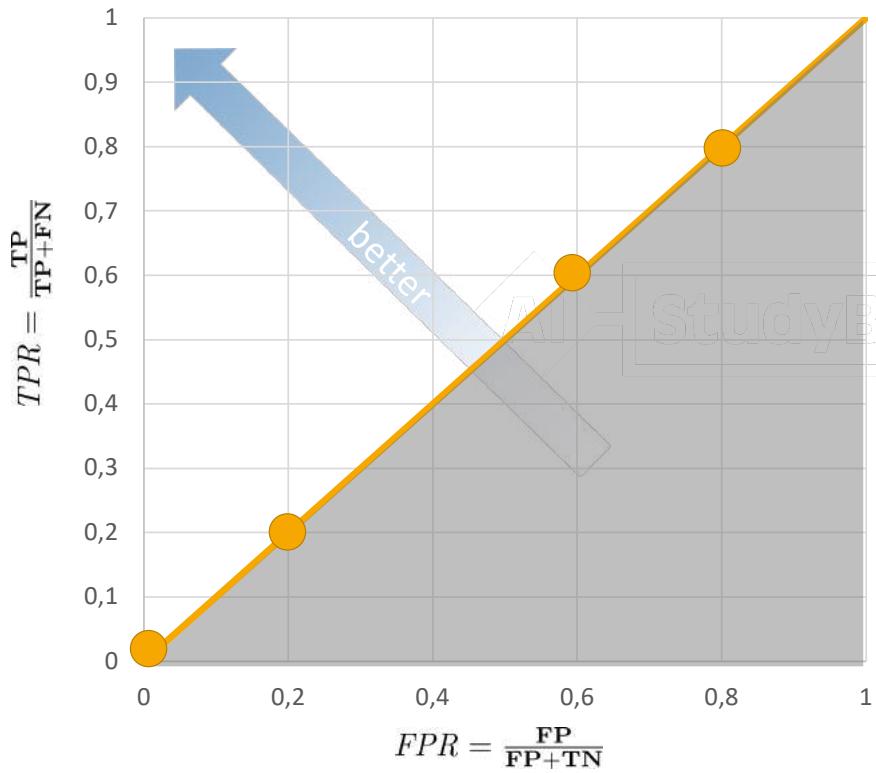
$$\mathbf{FN} = (1 - p) \cdot q \cdot N$$

$$\mathbf{TPR} = \frac{\mathbf{TP}}{\mathbf{TP} + \mathbf{FN}} = \frac{p \cdot q \cdot N}{p \cdot q \cdot N + (1-p) \cdot q \cdot N} = p$$

$$\mathbf{FPR} = \frac{\mathbf{FP}}{\mathbf{TN} + \mathbf{FP}} = \frac{p \cdot (1-q) \cdot N}{(1-p) \cdot (1-q) \cdot N + p \cdot (1-q) \cdot N} = p$$

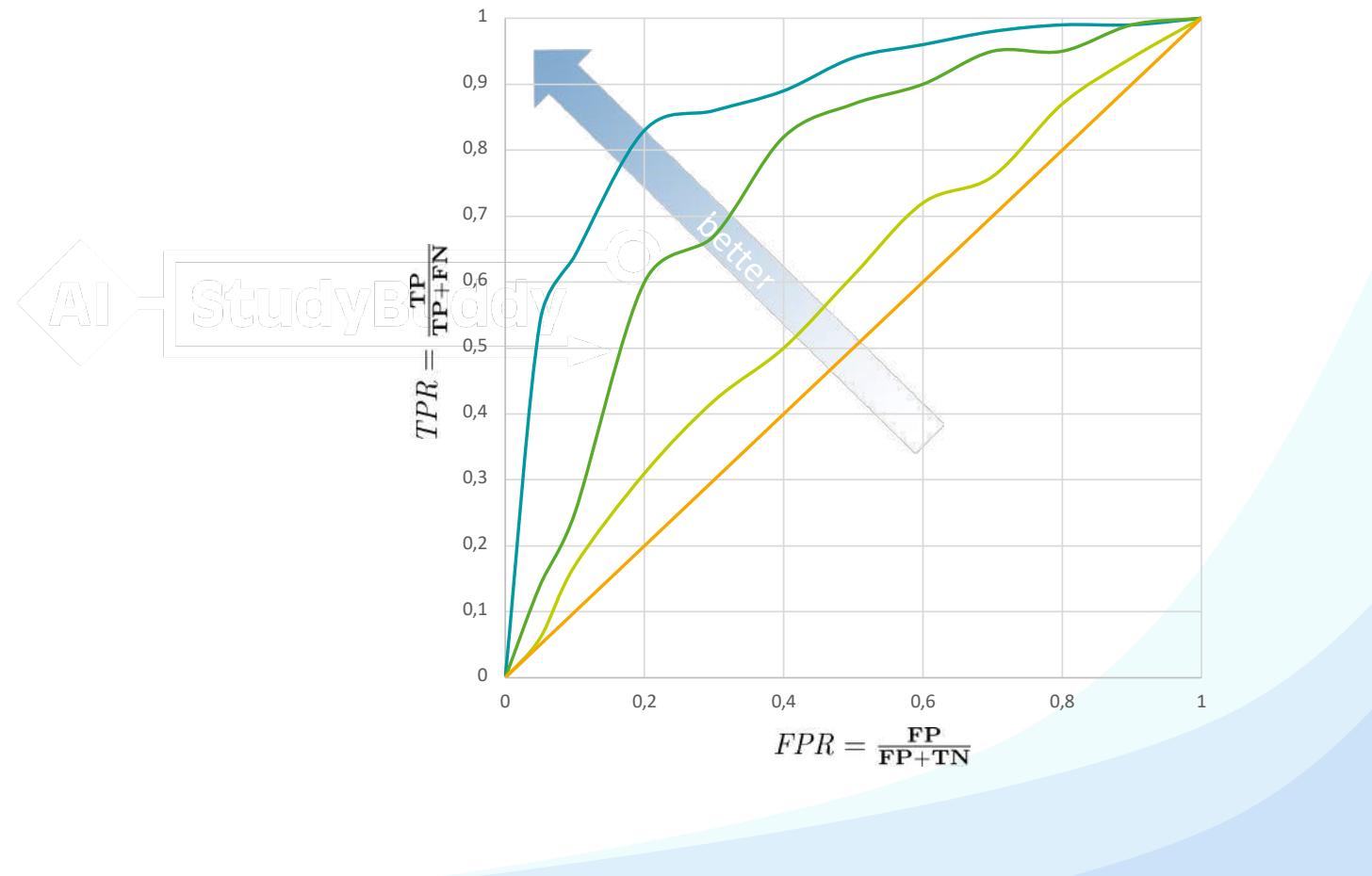
→ Performance is independent of q , N !

ROC Curve – Beating Random Guessing



- Every prediction model is at least as good as random guessing (if not, just invert the predictions)
- Thus, area under diagonal is uninteresting

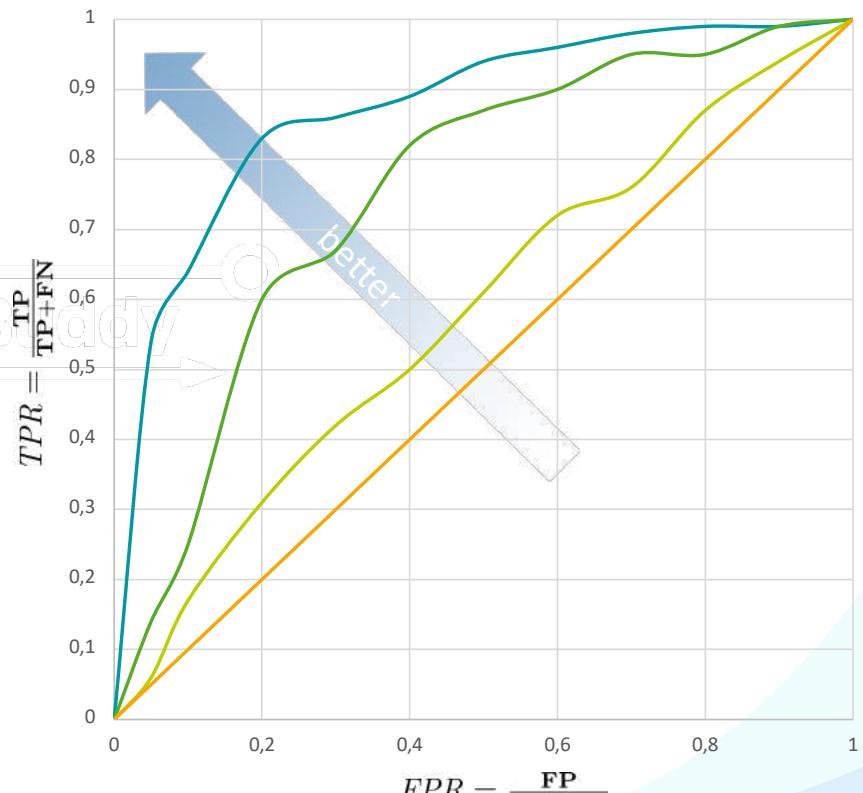
Example ROC Curves



ROC Index / AUC (Area Under the Curve)

Which model has best performance?

- ROC Index / AUC (Area Under the Curve)
- Larger area → closer to optimum
- Computable as integral of curve



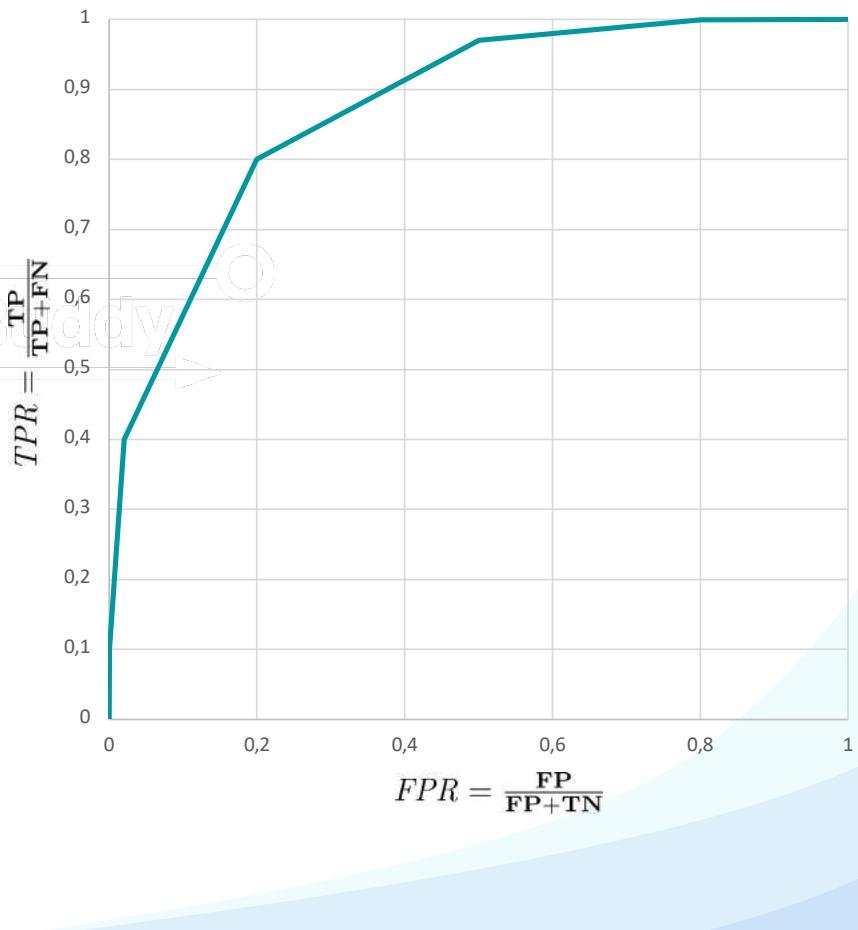
ROC Index / AUC (Area Under the Curve)

Which model has best performance?

- ROC Index / AUC (Area Under the Curve)
- Larger area → closer to optimum
- Computable as integral of curve

$$\sum_{i=2}^{|T|} ((\text{FPR}_i - \text{FPR}_{i-1}) \cdot \frac{(\text{TPR}_i + \text{TPR}_{i-1})}{2})$$

T is the set of thresholds
 FPR for the i th threshold
 TPR for the $(i-1)$ th threshold



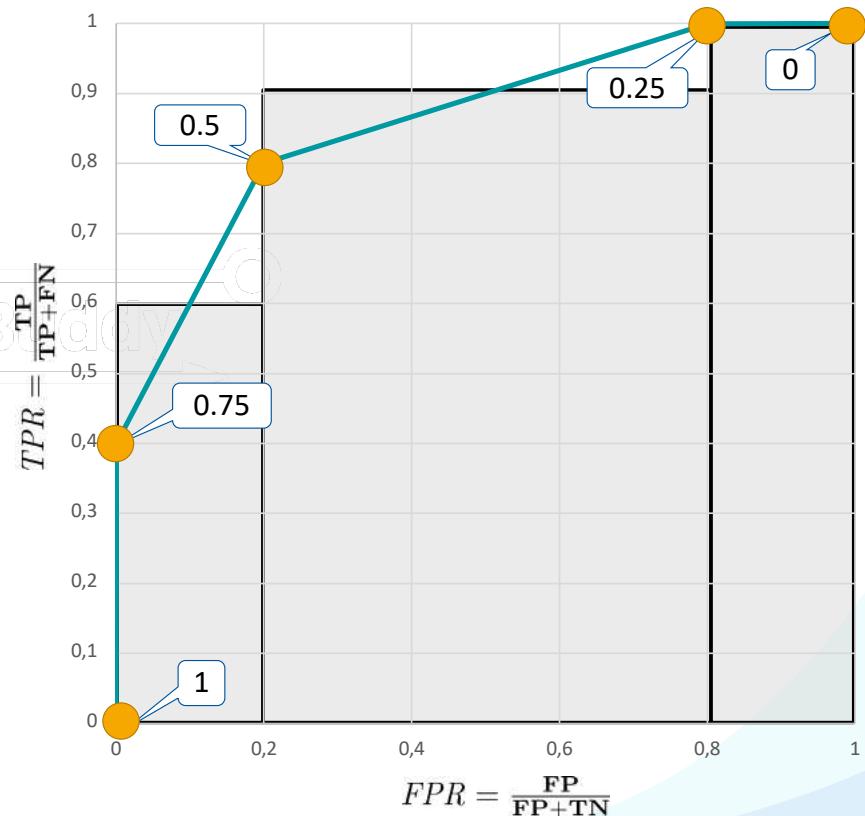
ROC Index / AUC (Area Under the Curve)

Example

$$\sum_{i=2}^{|T|} ((\text{FPR}_i - \text{FPR}_{i-1}) \cdot \frac{(\text{TPR}_i + \text{TPR}_{i-1})}{2})$$

$$T = \{1.0, 0.75, 0.5, 0.25, 0.0\}$$

$$\begin{aligned} & (0.0 - 0.0) \cdot \frac{(0.4 + 0.0)}{2} \quad \text{1.0 to 0.75} \\ & + (0.2 - 0.0) \cdot \frac{(0.8 + 0.4)}{2} \quad \text{0.75 to 0.5} \\ & + (0.8 - 0.2) \cdot \frac{(1.0 + 0.8)}{2} \quad \text{0.5 to 0.25} \\ & + (1.0 - 0.8) \cdot \frac{(1.0 + 1.0)}{2} \quad \text{0.25 to 0.0} \\ & = 0.0 + 0.04 + 0.54 + 0.2 = 0.78 \end{aligned}$$



TPS Exercise

You are comparing two predictive models (e.g., obtained from two different supervised learning methods).

Question:



- 1) How to assess performance differences?
- 2) What could go wrong?

Which is better?

		Prediction	
		On Time	Delay
		M ₁	M ₂
Target Label	On Time	7	3
	Delay	4	6

		Prediction	
		On Time	Delay
		M ₂	M ₁
Target Label	On Time	5	5
	Delay	4	6

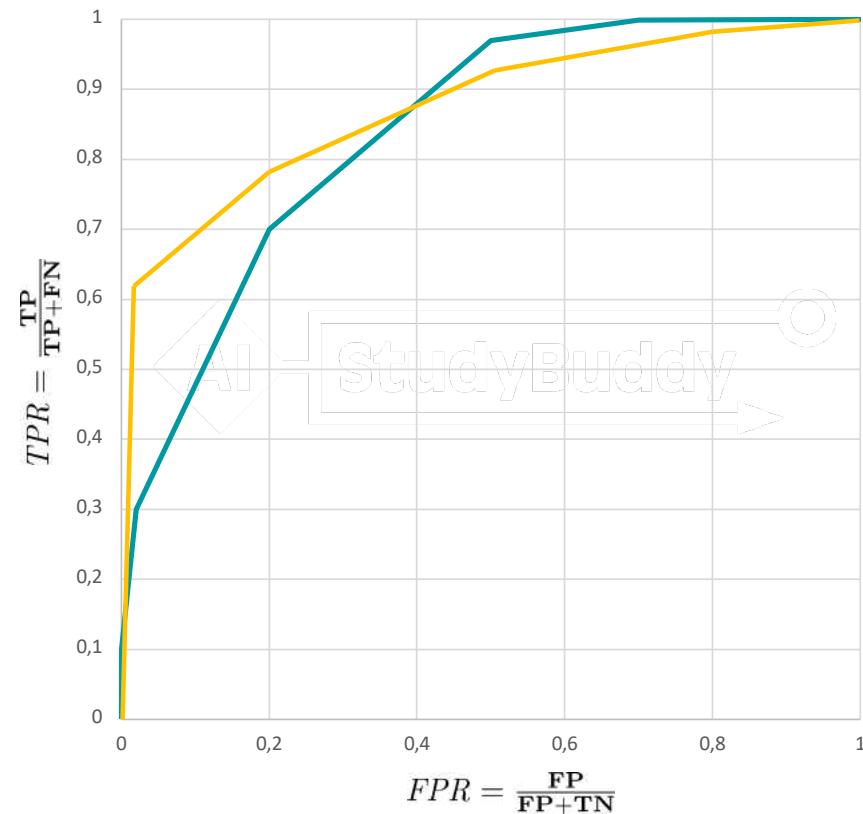
Which is better?

		Prediction	
		On Time	Delay
Target Label	M ₁	On Time	5
	M ₂	On Time	5
Target Label	M ₁	Delay	4
	M ₂	Delay	5
		On Time	4
		Delay	6

Which is better?

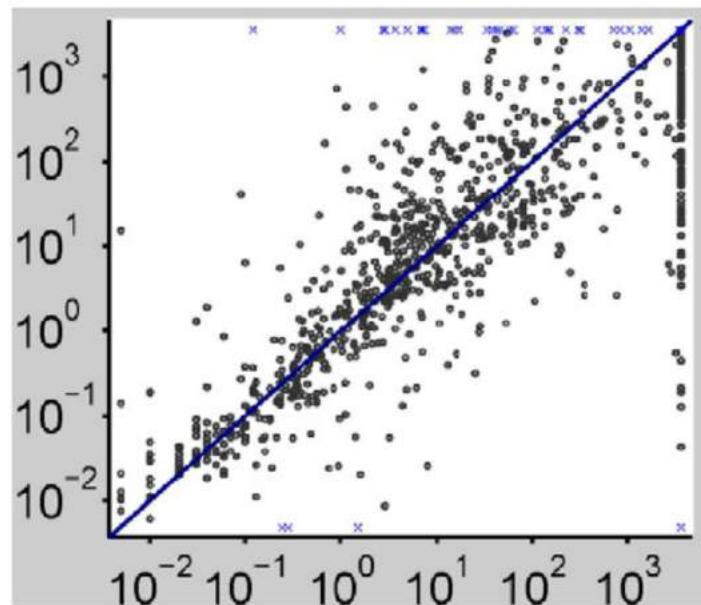
M_1

M_2

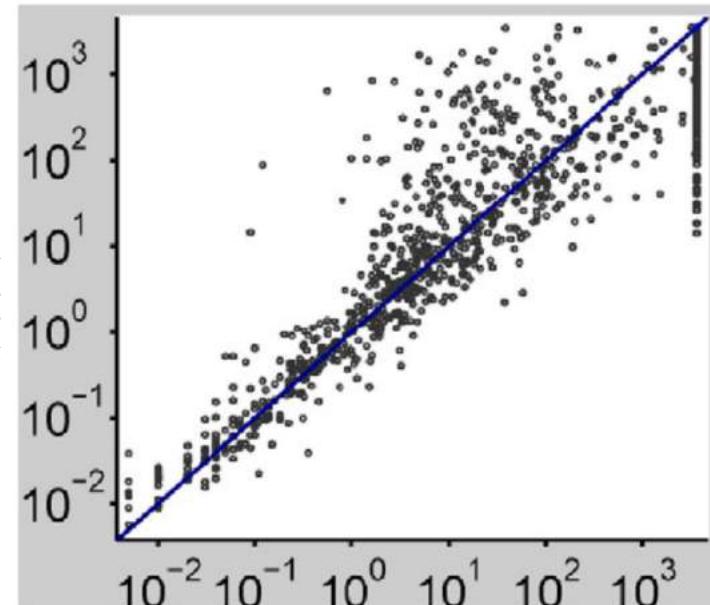


Which is better?

M_1 (Neural Network)



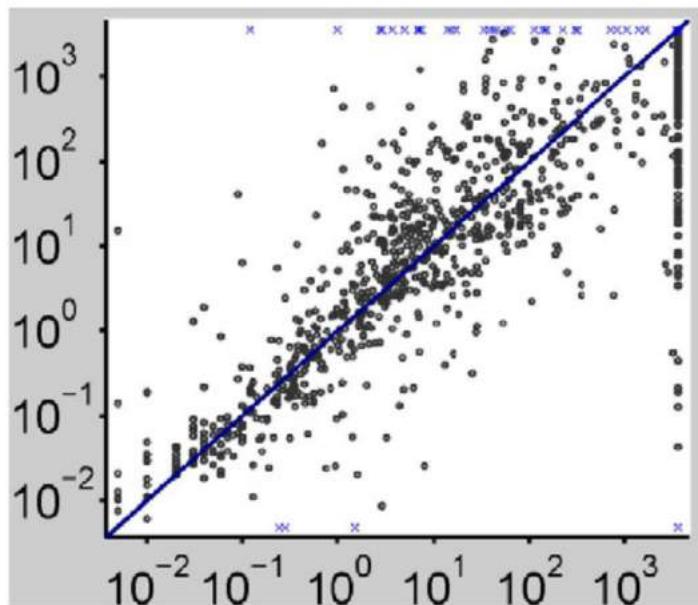
M_2 (Random Forest)



(Source: F. Hutter, L. Xu, H. Hoos, Kevin Leyton-Brown: Algorithm runtime prediction: Methods & evaluation, Artificial Intelligence 206 (2014) 79–111)

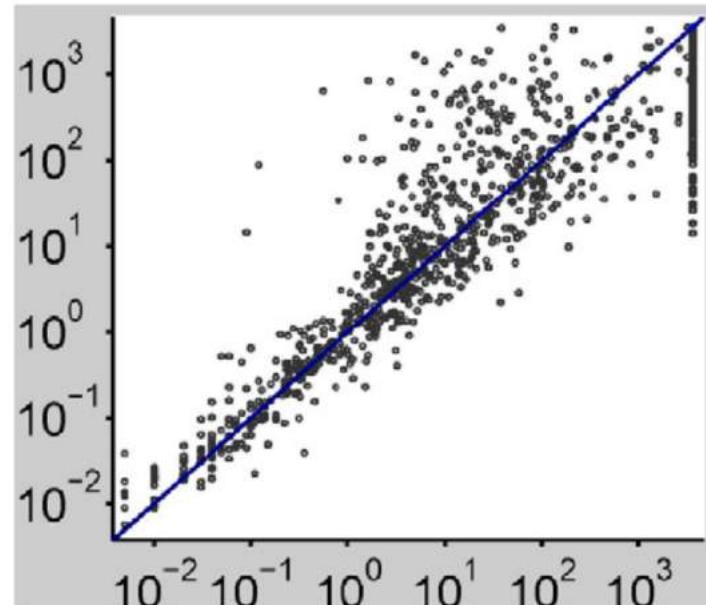
Which is better?

M_1 (Neural Network)



RMSE = 1.1

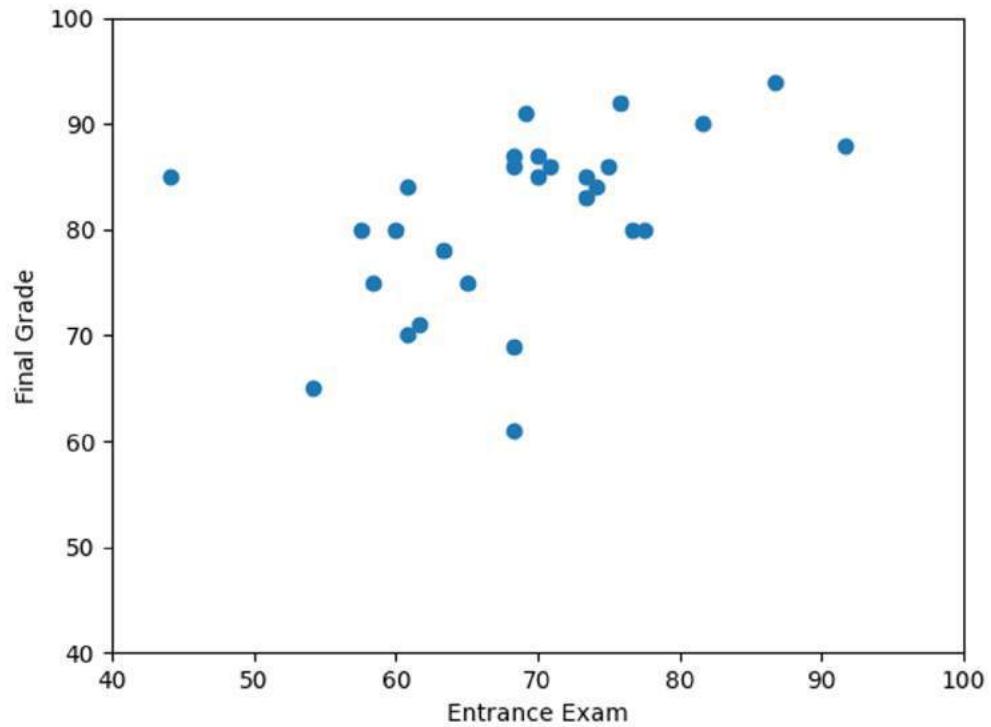
M_2 (Random Forest)



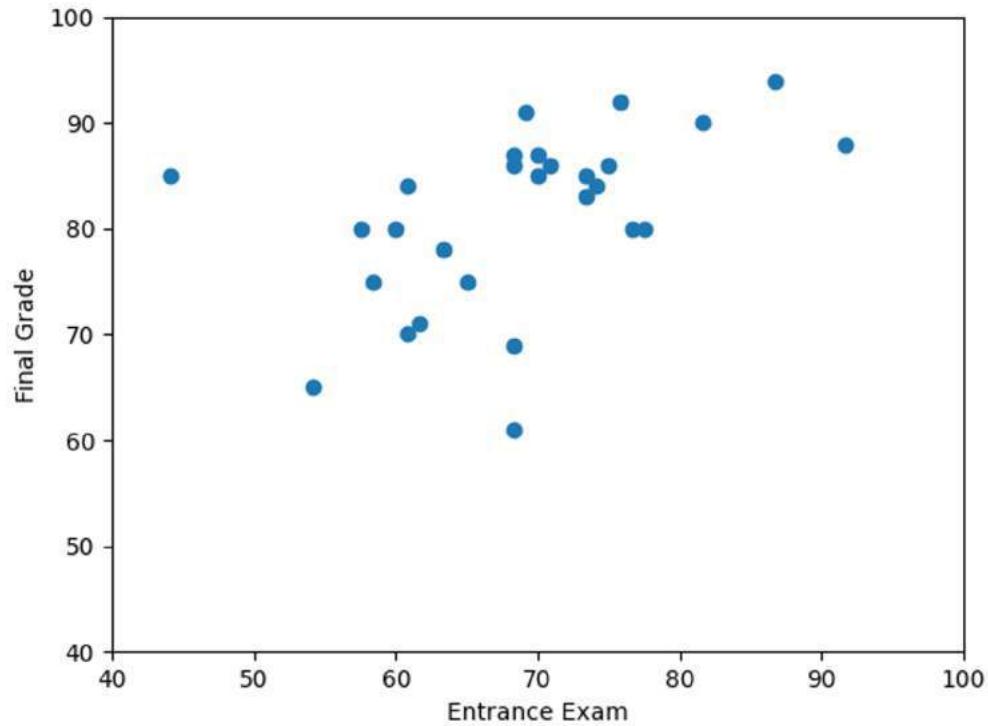
RMSE = 0.72

(Source: F. Hutter, L. Xu, H. Hoos, Kevin Leyton-Brown: Algorithm runtime prediction: Methods & evaluation, Artificial Intelligence 206 (2014) 79–111)

Assessing performance correlation

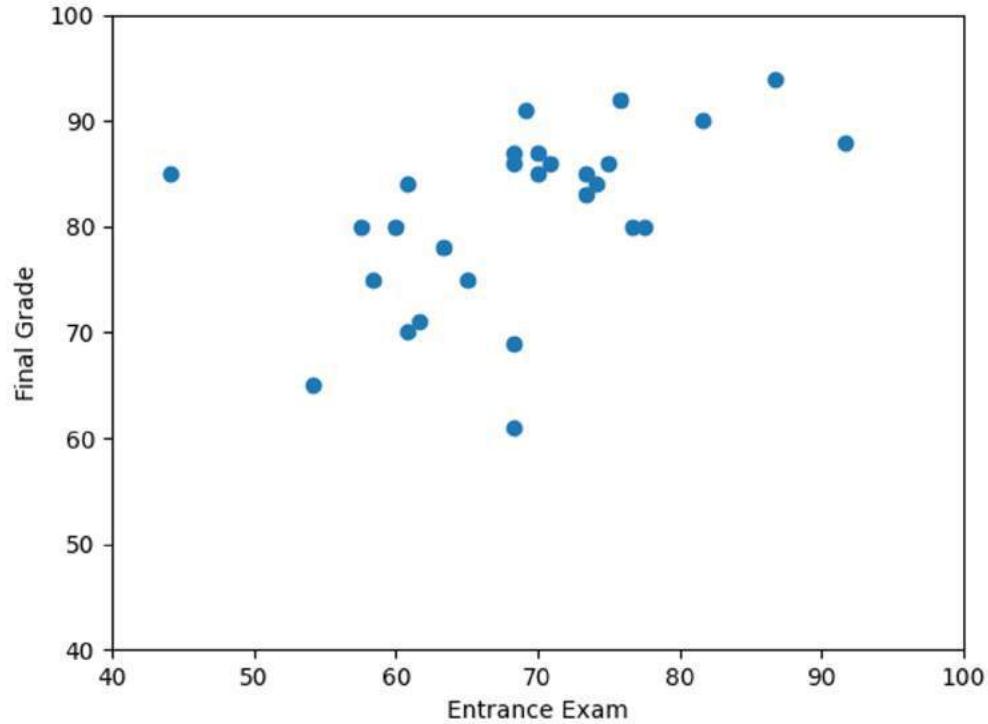


Assessing performance correlation



Pearson correlation coefficient = 0.41 (barely moderate association)

Assessing performance correlation



Pearson correlation coefficient = **0.41** (barely moderate association)
Spearman rank correlation coefficient = **0.58** (borderline strong association)

Pearson correlation coefficient:

- measures linear relationship between two sets of data
- both sets of data follow normal distribution (no outliers)



Pearson correlation coefficient:

- measures linear relationship between two sets of data
- both sets of data follow normal distribution (no outliers)

Spearman rank correlation coefficient:

- sort the data and assign ranks (1, 2, ...) = rank transformation
- compute Pearson CC → Spearman CC
- assumes monotonic relationship between two sets of data
- does not require normality assumption (non-parametric)

Which is better?

M_1 : accuracy from k -fold cross-validation = 0.712

M_2 : accuracy from k -fold cross-validation = 0.721

Which is better?

M_1 : accuracy from k -fold cross-validation = 0.712

M_2 : accuracy from k -fold cross-validation = 0.721

- performance differences may be due to random effects
- assess statistical significance using statistical hypothesis testing

Quick refresher on statistical hypothesis testing

H_0 : null-hypothesis, typically a statement of no significant effect
here: no significant performance difference between M_1, M_2

α : significance threshold = max. probability of incorrectly rejecting H_0
(incorrectly claiming significant differences = false positive = type I error)

NB: false negatives can also occur = failure to reject correct H_0
= type II error = incorrectly claiming ‘equal’ performance
(determined by power of the test)

p -value : (estimate) of the probability of committing a type I error

$p < \alpha \rightarrow \text{reject } H_0$

→ **NB: tests rely on assumptions to work correctly**

Testing for significance of performance differences

- consider performance values (e.g., accuracy) over folds (= empirical distribution) for M_1, M_2

→ $(m_{1,1}, m_{1,2}, \dots m_{1,k}),$
 $(m_{2,1}, m_{2,2}, \dots m_{2,k}),$

- consider pairs $(m_{1,i}, m_{2,i})$ for each fold
(NB: these correspond to the points in a scatter plot, one point per fold)

- use a paired t-test to assess statistical significance of performance differences between M_1, M_2 on the given test set based on the given fold, using standard significance level $\alpha = 0.05$

Caution: paired t -test requires normality assumption!

How can we know whether performance data over folds follows a normal distribution?

What to do if it doesn't?



Caution: paired *t*-test requires normality assumption!

How can we know whether performance data over folds follows a normal distribution?

→ check QQ plot or use normality test (e.g., Shapiro–Wilk)

What to do if it doesn't?



→ use a non-parametric test, e.g., Wilcoxon Signed-Ranks Test

Comparing two predictive models

- assess performance of each model individually
- analyse performance correlation
 - classification: overlap/differences in FP, FN, misclassifications
 - regression: scatter plot, correlation coefficient
- use appropriate statistical tests



Don't...

- limit analysis to single performance metric
- limit correlation to single number
(in particular: standard = Pearson correlation coefficient)

TPS Exercise

You are using a randomised supervised ML procedure to train a predictive model.

Questions:



- 1) How to assess the training procedure?
- 2) What could go wrong?

Evaluating randomised supervised ML procedures

- perform p independent runs ($p \geq 2$)
→ p models
- assess & compare performance of all p models
- inspect / analyse distribution of performance metrics,
multiple performance metrics

Don't...

- just aggregate performance over all p models
- limit analysis to single performance metric
- report only the best result! (No cherry picking!)

TPS Exercise

You have trained a predictive model using supervised ML,
you've carefully assessed its performance and
deployed it in practice.

Questions:



What could happen to invalidate earlier
performance assessment?

TPS Exercise

You have trained a predictive model using supervised ML,
you've carefully assessed its performance and
deployed it in practice.

Questions:



What could happen to invalidate earlier
performance assessment?

→ Performance degradation due to concept drift
(violation of supervised learning assumption)

Key concepts covered today:

- performance measures for multi-class classification
(multinomial prediction targets)
- performance measures for regression models
(numerical prediction targets)
- ROC curves, AUC
- randomness in the training procedure
- comparative performance analysis
- Spearman's rank correlation coefficient
- statistical significance tests

Learning Goals

At the end of this module, students should be able to

- assess the quality of a model obtained from a supervised machine learning method using widely accepted methods, including standard performance metrics, confusion matrices, ROC curves
- demonstrate understanding and working knowledge of the problems that can occur when using supervised learning procedures and the models obtained from them
- explain when and why it is important to distinguish between training, validation and testing data
- explain standard validation techniques, including k -fold and leave-one-out cross-validation
- assess performance differences using appropriate statistical techniques
- explain the problems that can arise from unbalanced data sets and demonstrate understanding as well as working knowledge of methods for addressing these problems

Elements of Machine Learning & Data Science

Winter semester 2023/24

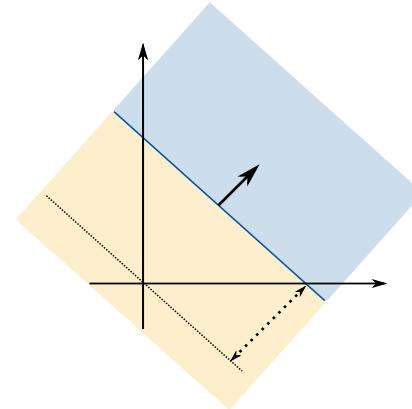
Lecture 14 – Linear Discriminants

01.12.2023

Prof. Bastian Leibe

Machine Learning Topics

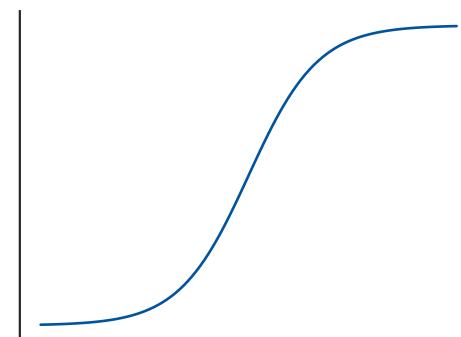
1. Introduction to ML
2. Probability Density Estimation
- 3. Linear Discriminants**
4. Linear Regression
5. Logistic Regression
6. Support Vector Machines
7. AdaBoost
8. Neural Network Basics



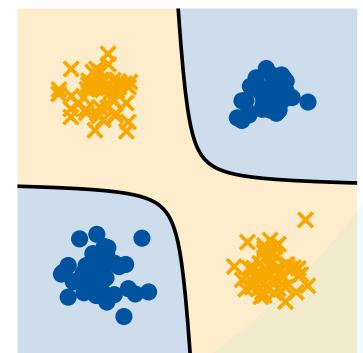
Linear Discriminant Functions

$$E(\mathbf{w}) = \frac{1}{2} \sum_n (y(\mathbf{x}_n; \mathbf{w}) - t_n)^2$$

Least-Squares Classification



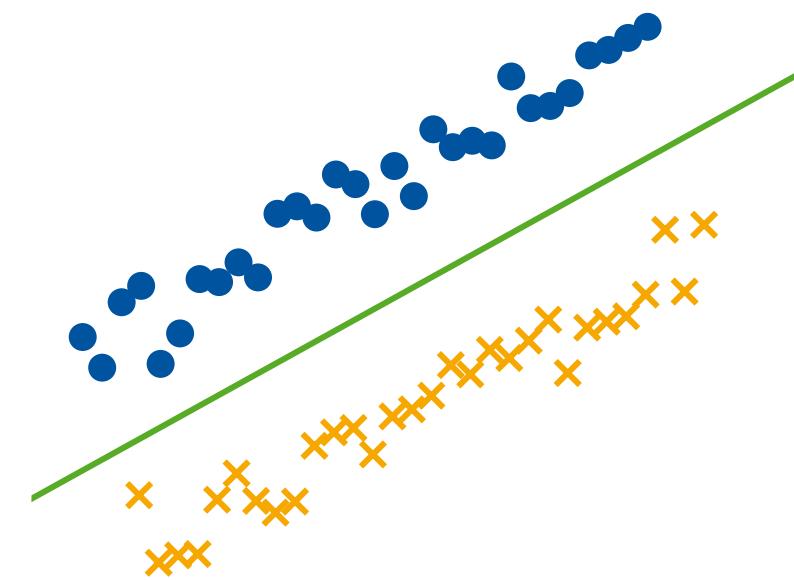
Activation Functions



Basis Functions

Linear Discriminants

1. Motivation: Discriminant Functions
2. Linear Discriminant Functions
3. Least-Squares Classification
4. Generalized Linear Discriminants
5. Basis Functions



Discriminant Functions

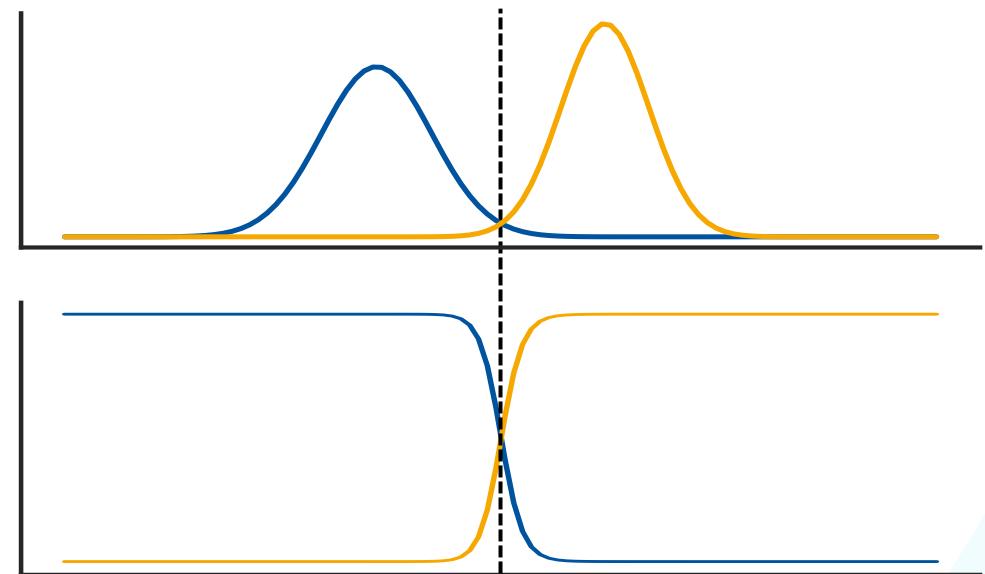
- Remember Bayes Decision Theory
 - Model the likelihood $p(\mathbf{x}|\mathcal{C}_k)$ and the prior $p(\mathcal{C}_k)$
 - From this, we can compute the posterior $p(\mathcal{C}_k|\mathbf{x})$
 - Bayes optimal decision: Decide for class \mathcal{C}_1 if

$$p(\mathcal{C}_1|\mathbf{x}) > p(\mathcal{C}_2|\mathbf{x}) \Leftrightarrow$$

$$\frac{p(\mathbf{x}|\mathcal{C}_1)}{p(\mathbf{x}|\mathcal{C}_2)} > \frac{p(\mathcal{C}_2)}{p(\mathcal{C}_1)}$$

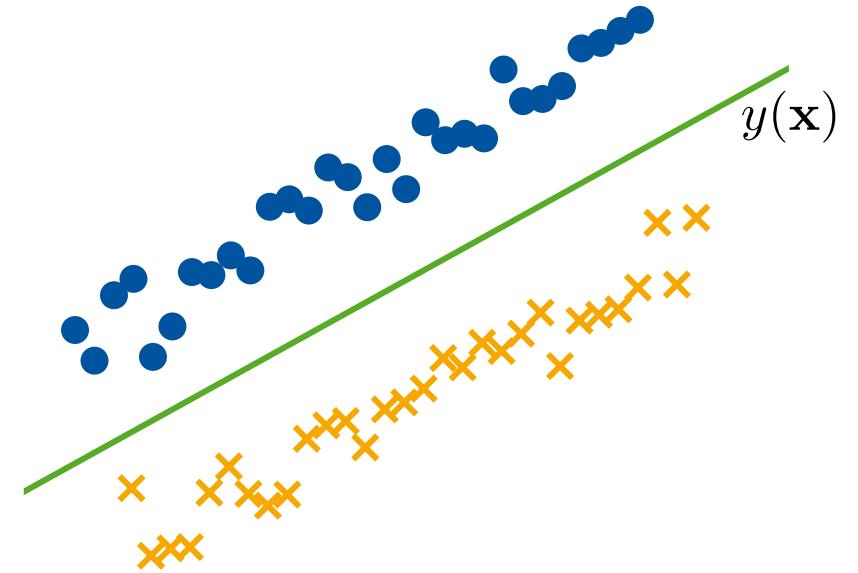
by comparing posteriors

by comparing likelihoods and priors



Discriminant Functions

- This chapter: Different approach
 - Directly represent the decision boundary with a **discriminant function** $y(\mathbf{x})$
 - Without explicit modeling of probability densities!
 - *We will learn ways to define $y(\mathbf{x})$ such that we still make a decision based on posteriors...*
 - *...but we don't have to. This framework gives us more flexibility.*



Idea

- Formulate classification in terms of comparisons

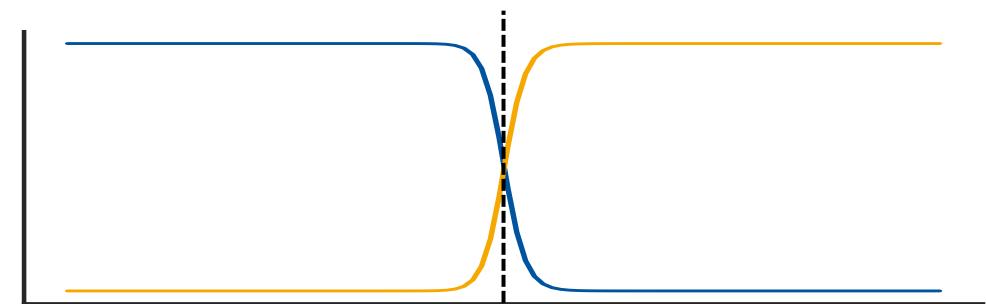
- Bayes Decision Theory:

$$\begin{aligned} p(\mathcal{C}_1|\mathbf{x}) &> p(\mathcal{C}_2|\mathbf{x}) \\ \iff p(\mathcal{C}_1|\mathbf{x}) - p(\mathcal{C}_2|\mathbf{x}) &> 0 \\ \iff y(\mathbf{x}) &> 0 \end{aligned}$$

- More general:

- Define a **discriminant function** $y(\mathbf{x})$
 - Classify \mathbf{x} as class \mathcal{C}_1 if $y(\mathbf{x}) > 0$

- Advantage: more flexibility



E.g., we could now define

$$y(\mathbf{x}) = p(\mathcal{C}_1|\mathbf{x}) - p(\mathcal{C}_2|\mathbf{x})$$

$$y(\mathbf{x}) = \ln \frac{p(\mathbf{x}|\mathcal{C}_1)}{p(\mathbf{x}|\mathcal{C}_2)} + \ln \frac{p(\mathcal{C}_1)}{p(\mathcal{C}_2)}$$

Idea

- **Multi-class case**
 - Define **discriminant functions**
$$y_1(\mathbf{x}), \dots, y_K(\mathbf{x})$$
 - Classify \mathbf{x} as class \mathcal{C}_k if
$$y_k(\mathbf{x}) > y_j(\mathbf{x}) \quad \forall j \neq k$$
- Again, this is compatible with Bayes Decision Theory:

- E.g., $y_k(\mathbf{x}) = p(\mathcal{C}_k | \mathbf{x})$

$$y_k(\mathbf{x}) = p(\mathbf{x} | \mathcal{C}_k) p(\mathcal{C}_k)$$

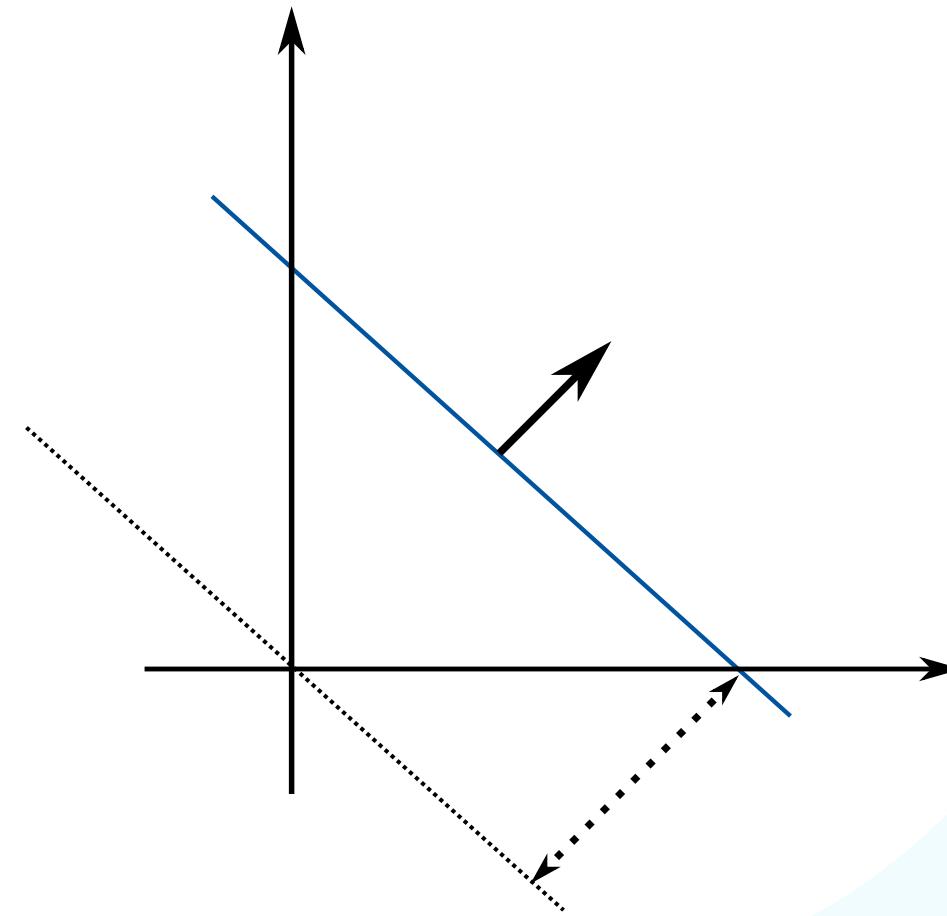
$$y_k(\mathbf{x}) = \log p(\mathbf{x} | \mathcal{C}_k) + \log p(\mathcal{C}_k)$$

Problem Formulation

- **General classification problem**
 - Goal: take a new input \mathbf{x} and assign it to one of K classes \mathcal{C}_k
 - Given training set $\mathcal{X} = \{\mathbf{x}_1, \dots, \mathbf{x}_N\}$ with target values $\mathcal{T} = \{\mathbf{t}_1, \dots, \mathbf{t}_N\}$
⇒ Learn a discriminant function $y(\mathbf{x})$ to perform the classification
- **2-class problem**
 - **Binary target values** $t_n \in \{-1, 1\}$ or $t_n \in \{0, 1\}$
- **K -class problem**
 - **1-of- K coding scheme**, e.g. $\mathbf{t}_n = (0, 1, 0, 0, 0)^T$

Linear Discriminants

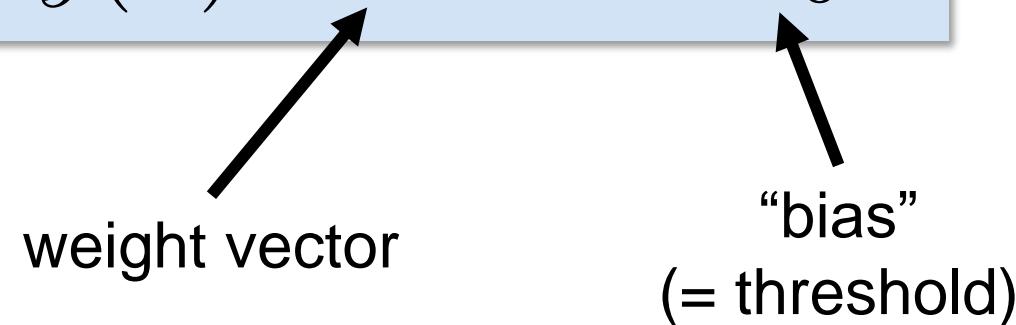
1. Motivation: Discriminant Functions
2. **Linear Discriminant Functions**
3. Least-Squares Classification
4. Generalized Linear Discriminants
5. Basis Functions



Linear Discriminant Functions

- 2-class problem
 - $y(\mathbf{x}) > 0$: Decide for class \mathcal{C}_1 , else for \mathcal{C}_2
- In the following, we focus on [linear discriminant functions](#):

$$y(\mathbf{x}) = \mathbf{w}^\top \mathbf{x} + w_0$$



weight vector

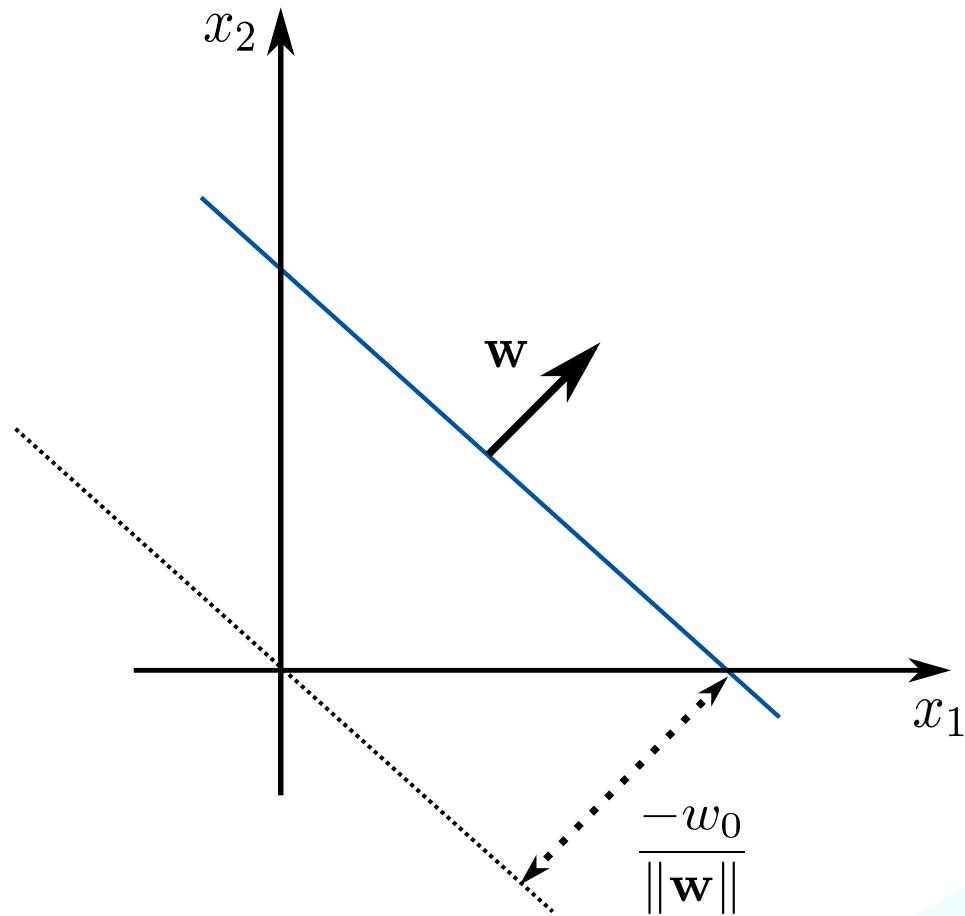
“bias”
(= threshold)

- If a dataset can be perfectly classified by a linear discriminant, we call it [linearly separable](#).

Intuition

$$y(\mathbf{x}) = \mathbf{w}^\top \mathbf{x} + w_0$$

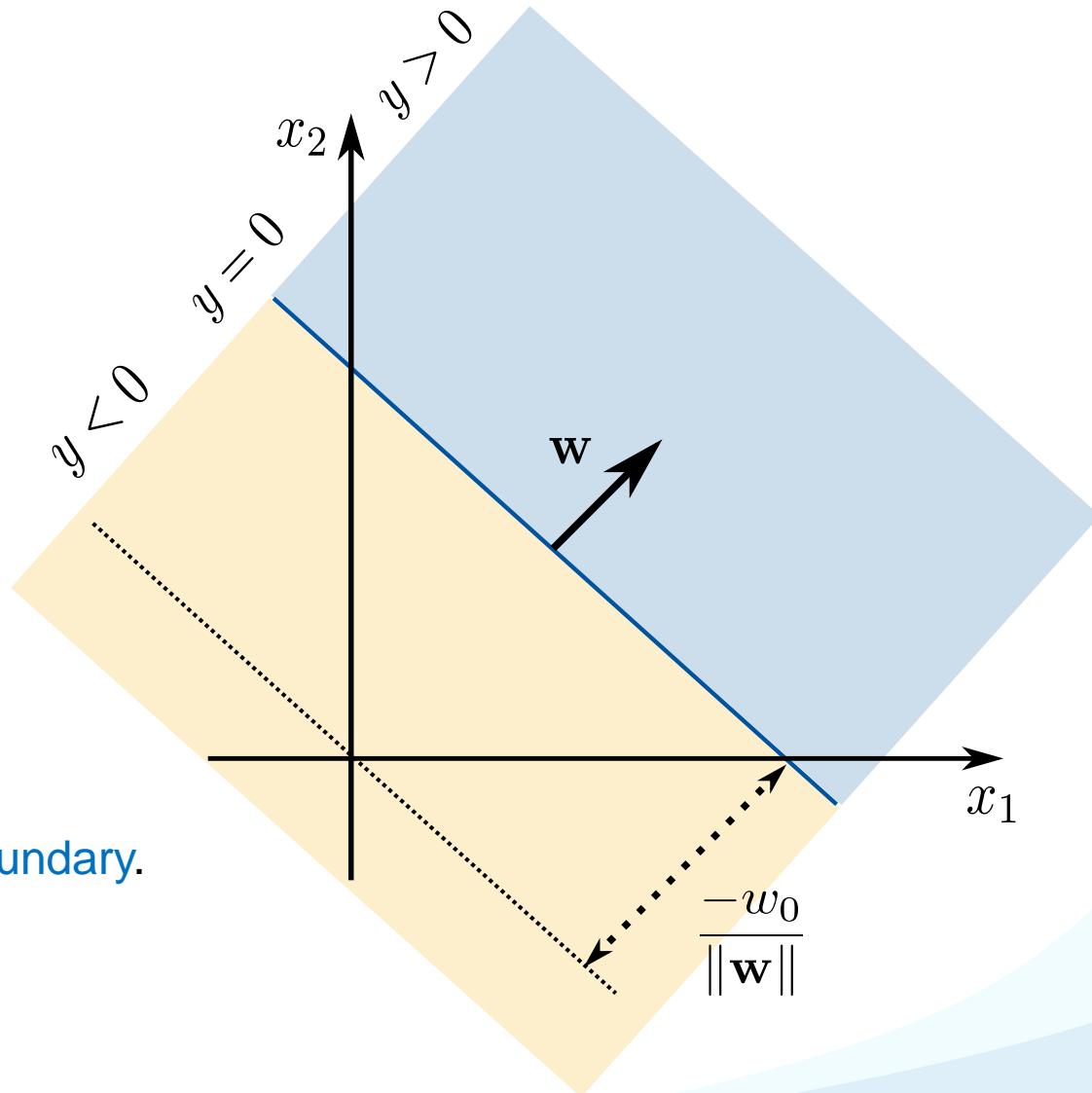
- Graphical interpretation:
 - Normal equation of a hyperplane
 - Normal vector: \mathbf{w}
 - Offset: $\frac{-w_0}{\|\mathbf{w}\|}$



Intuition

$$y(\mathbf{x}) = \mathbf{w}^T \mathbf{x} + w_0$$

- Graphical interpretation:
 - Normal equation of a hyperplane
- The hyperplane is given by $y(\mathbf{x}) = 0$
 - One side: $y(\mathbf{x}) > 0$
 - Other side: $y(\mathbf{x}) < 0$
- This hyperplane defines the decision boundary.



Notation

- Let's look at the equation in detail

$$\begin{aligned}y(\mathbf{x}) &= \mathbf{w}^T \mathbf{x} + w_0 \\&= \sum_{i=1}^D w_i x_i + w_0\end{aligned}$$

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_D \end{bmatrix} \quad \mathbf{w} = \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_D \end{bmatrix}$$

- Alternative notation with extended vector

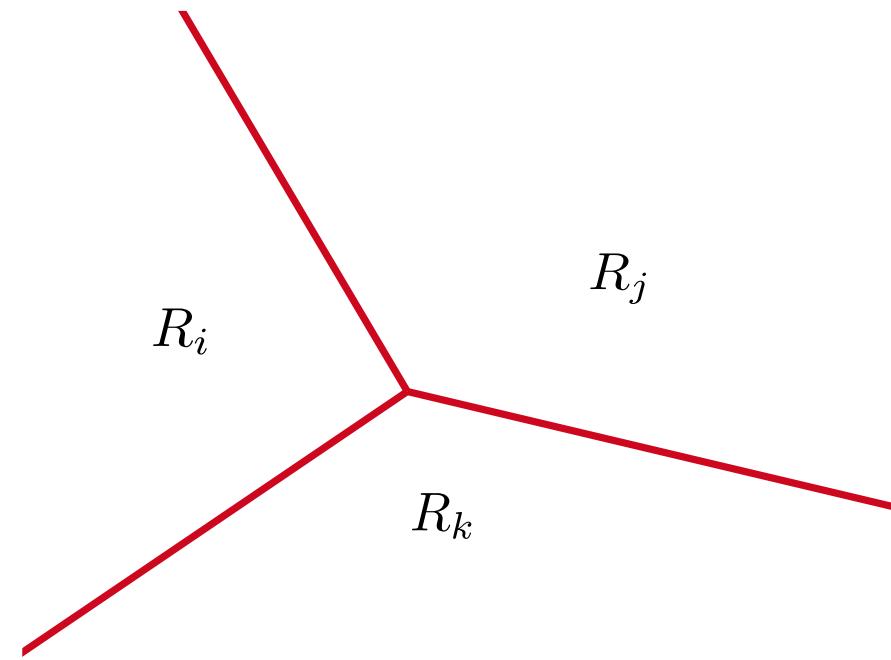
$$\begin{aligned}y(\mathbf{x}) &= \sum_{i=0}^D w_i x_i \quad \text{with } x_0 = 1 \\&= \tilde{\mathbf{w}}^T \tilde{\mathbf{x}}\end{aligned}$$

$$\tilde{\mathbf{x}} = \begin{bmatrix} 1 \\ x_1 \\ x_2 \\ \vdots \\ x_D \end{bmatrix} \quad \tilde{\mathbf{w}} = \begin{bmatrix} w_0 \\ w_1 \\ w_2 \\ \vdots \\ w_D \end{bmatrix}$$

D : Number of Dimensions

Extension to Multiple Classes

- **K-class discriminant**
 - Combination of K linear functions
$$y_k(\mathbf{x}) = \mathbf{w}_k^T \mathbf{x} + w_{k0}, \quad k = 1, \dots, K$$
- Interpretation
 - Decide for \mathcal{C}_k iff $y_k(\mathbf{x}) > y_j(\mathbf{x}) \quad \forall j \neq k$
 - Resulting decision hyperplanes:
$$(\mathbf{w}_k - \mathbf{w}_j)^T \mathbf{x} + (w_{k0} - w_{j0}) = 0$$



Extension to Multiple Classes

- **K -class discriminant**

- Combination of K linear functions

$$y_k(\mathbf{x}) = \mathbf{w}_k^T \mathbf{x} + w_{k0}, \quad k = 1, \dots, K$$

- Interpretation

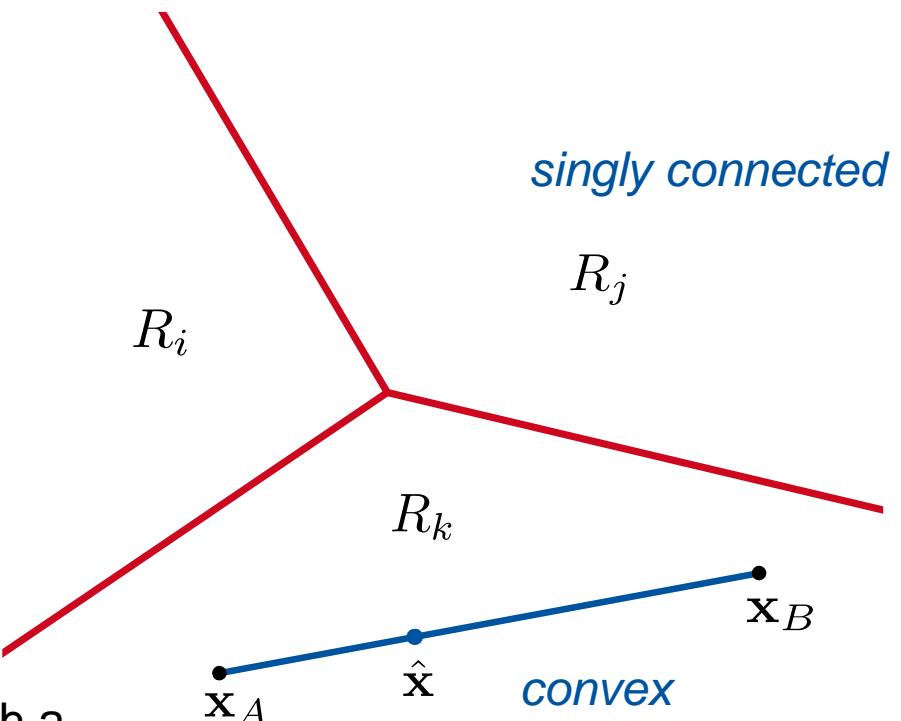
- Decide for \mathcal{C}_k iff $y_k(\mathbf{x}) > y_j(\mathbf{x}) \quad \forall j \neq k$

- Resulting decision hyperplanes:

$$(\mathbf{w}_k - \mathbf{w}_j)^T \mathbf{x} + (w_{k0} - w_{j0}) = 0$$

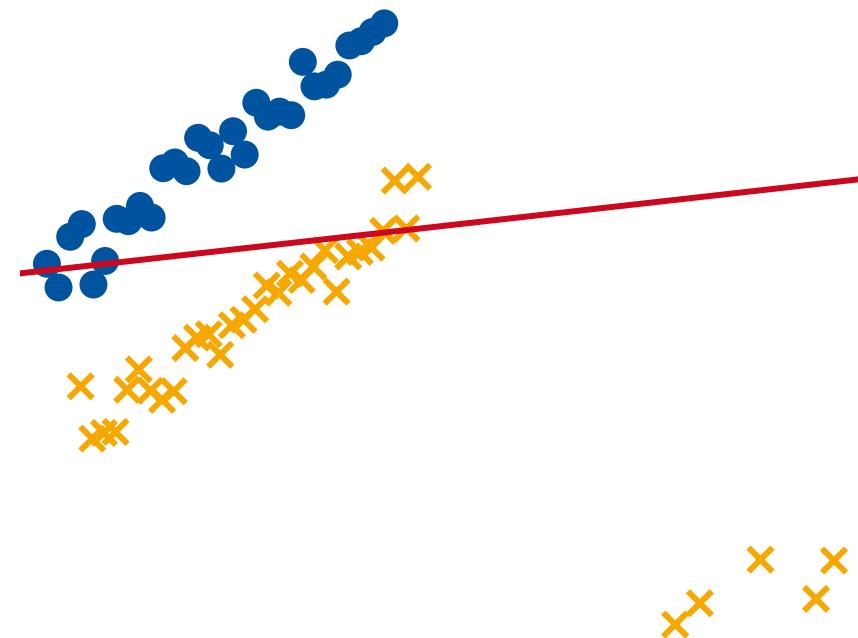
- It can be shown that the decision regions of such a discriminant are always **singly connected** and **convex**.

⇒ Particularly suitable for problems with unimodal conditional densities $p(\mathbf{x}|\mathbf{w}_i)$



Linear Discriminants

1. Motivation: Discriminant Functions
2. Linear Discriminant Functions
3. **Least-Squares Classification**
4. Generalized Linear Discriminants
5. Basis Functions



General Classification Problem

- K Classes described by linear discriminant models:

$$y_k(\mathbf{x}) = \mathbf{w}_k^\top \mathbf{x} + w_{k0}, \quad k = 1, \dots, K$$

- Group them together using vector notation:

$$\mathbf{y}(\mathbf{x}) = \widetilde{\mathbf{W}}^\top \widetilde{\mathbf{x}} = \begin{bmatrix} y_1(\mathbf{x}) \\ y_2(\mathbf{x}) \\ \vdots \\ y_K(\mathbf{x}) \end{bmatrix} \quad \text{where}$$

$$\widetilde{\mathbf{W}} = [\widetilde{\mathbf{w}}_1, \dots, \widetilde{\mathbf{w}}_K] = \begin{bmatrix} w_{10} & \cdots & w_{K0} \\ \vdots & \ddots & \vdots \\ w_{1D} & \cdots & w_{KD} \end{bmatrix}$$
$$\widetilde{\mathbf{x}} = [1, x_1, \dots, x_D]^\top$$

- The output will be in **1-of- K** notation.
⇒ We can directly compare it to the target value

$$\mathbf{t} = [t_1, \dots, t_k]^\top$$

Classification Problem for the Entire Dataset

- Write the classification output for the whole dataset:

$$\mathbf{Y}(\tilde{\mathbf{X}}) = \tilde{\mathbf{X}}\tilde{\mathbf{W}}$$

where

$$\tilde{\mathbf{W}} = [\tilde{\mathbf{w}}_1, \dots, \tilde{\mathbf{w}}_K] = \begin{bmatrix} w_{01} & \cdots & w_{0K} \\ \vdots & \ddots & \vdots \\ w_{D1} & \cdots & w_{DK} \end{bmatrix}$$

- Using the data matrix

$$\tilde{\mathbf{X}} = \begin{bmatrix} \tilde{\mathbf{x}}_1^T \\ \vdots \\ \tilde{\mathbf{x}}_N^T \end{bmatrix} = \begin{bmatrix} x_{10} & \cdots & x_{1D} \\ \vdots & \ddots & \vdots \\ x_{N0} & \cdots & x_{ND} \end{bmatrix}$$

- Similarly, we group the target vectors in a matrix

$$\mathbf{T} = \begin{bmatrix} \mathbf{t}_1^T \\ \vdots \\ \mathbf{t}_N^T \end{bmatrix} = \begin{bmatrix} t_{11} & \cdots & t_{1K} \\ \vdots & \ddots & \vdots \\ t_{N1} & \cdots & t_{NK} \end{bmatrix}$$

- We can now compare the classifier output with the target matrix: $\tilde{\mathbf{X}}\tilde{\mathbf{W}} - \mathbf{T}$

Defining the Classification Problem

- Comparing the classifier output with the target matrix:

$$\widetilde{\mathbf{X}} \widetilde{\mathbf{W}} - \mathbf{T}$$

- Goal: Choose $\widetilde{\mathbf{W}}$ such that this difference becomes minimal
 - What does *minimal* mean here?
 - How strongly do we want to penalize deviations from the ideal target value?*
- Idea: define an **error function** that specifies the **loss** for each deviation

$$E(\widetilde{\mathbf{W}}) = \sum_{n=1}^N \sum_{k=1}^K L(y_k(\mathbf{x}_n; \mathbf{w}_k), t_{nk})$$

Least-Squares Classification

- Simplest approach: minimize Sum-of-squares error

$$\begin{aligned} E(\mathbf{W}) &= \frac{1}{2} \sum_{n=1}^N \sum_{k=1}^K (y_k(\mathbf{x}_n; \mathbf{w}_k) - t_{nk})^2 \\ &= \frac{1}{2} \sum_{n=1}^N \sum_{k=1}^K (\mathbf{w}_k^\top \mathbf{x}_n - t_{nk})^2 \end{aligned}$$

*Simplified notation:
Leaving out the $\tilde{\mathbf{x}}$...*

- How do we minimize this function?
 - *Take the derivative and set it to zero...*

Derivation

- Let's concentrate on the two-class case first:

$$E(\mathbf{w}) = \frac{1}{2} \sum_{n=1}^N (\mathbf{w}^\top \mathbf{x}_n - t_n)^2 \quad t_n \in \{-1, 1\}$$

- Taking the derivative:

$$\begin{aligned} \frac{\partial E(\mathbf{w})}{\partial \mathbf{w}} &= \sum_{n=1}^N (\mathbf{w}^\top \mathbf{x}_n - t_n) \mathbf{x}_n \\ &= \mathbf{X}^\top (\mathbf{X}\mathbf{w} - \mathbf{t}) \end{aligned}$$

with $\mathbf{X} = \begin{pmatrix} \mathbf{x}_1^\top \\ \vdots \\ \mathbf{x}_N^\top \end{pmatrix} \quad \mathbf{t} = (t_1, \dots, t_N)^\top$

Linear Algebra textbook:

$$\boxed{\frac{\partial \mathbf{a}^\top \mathbf{b}}{\partial \mathbf{a}} = \mathbf{b}}$$

- Setting the derivative to zero:

$$\frac{\partial E(\mathbf{w})}{\partial \mathbf{w}} = \mathbf{X}^T (\mathbf{X}\mathbf{w} - \mathbf{t}) \stackrel{!}{=} 0 \quad \text{"pseudo-inverse"}$$
$$\mathbf{X}^T \mathbf{X} \mathbf{w} = \mathbf{X}^T \mathbf{t}$$
$$\mathbf{w} = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{t}$$
$$= \mathbf{X}^\dagger \mathbf{t}$$

- We then obtain the discriminant function as

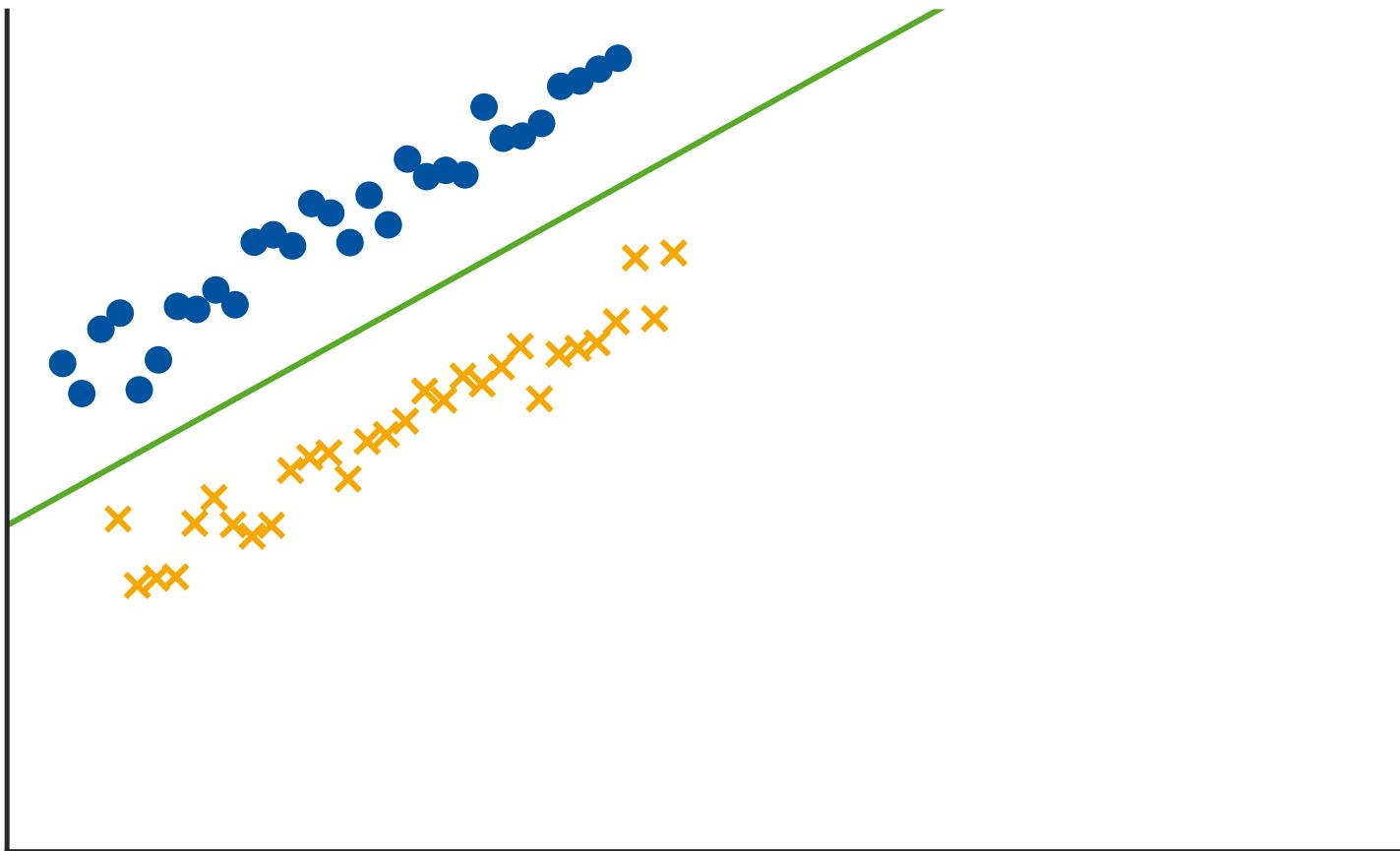
$$y(\mathbf{x}; \mathbf{w}) = \mathbf{w}^T \mathbf{x} = \mathbf{t}^T (\mathbf{X}^\dagger)^T \mathbf{x}$$

Exact, closed-form solution!

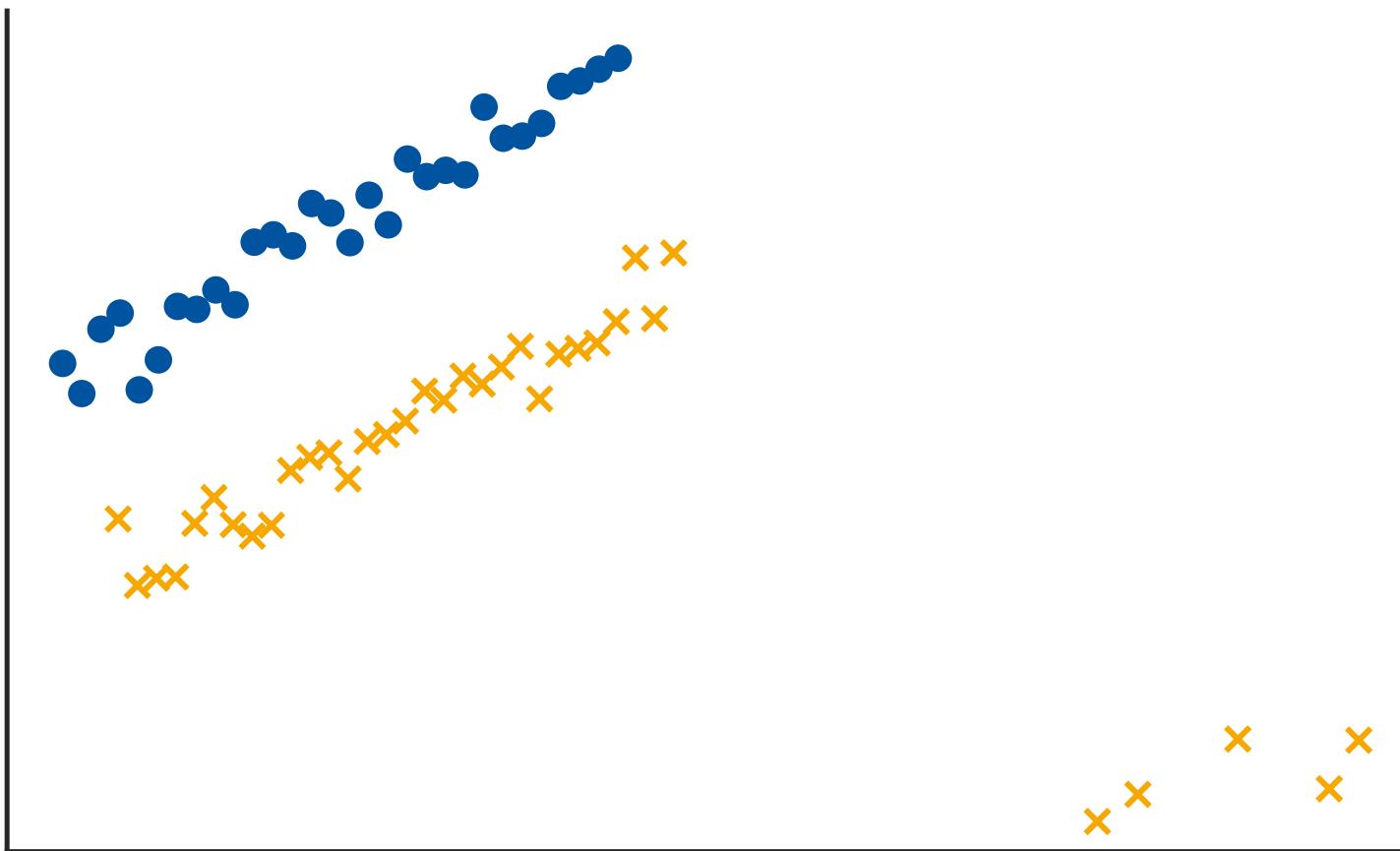
Example: Two Classes



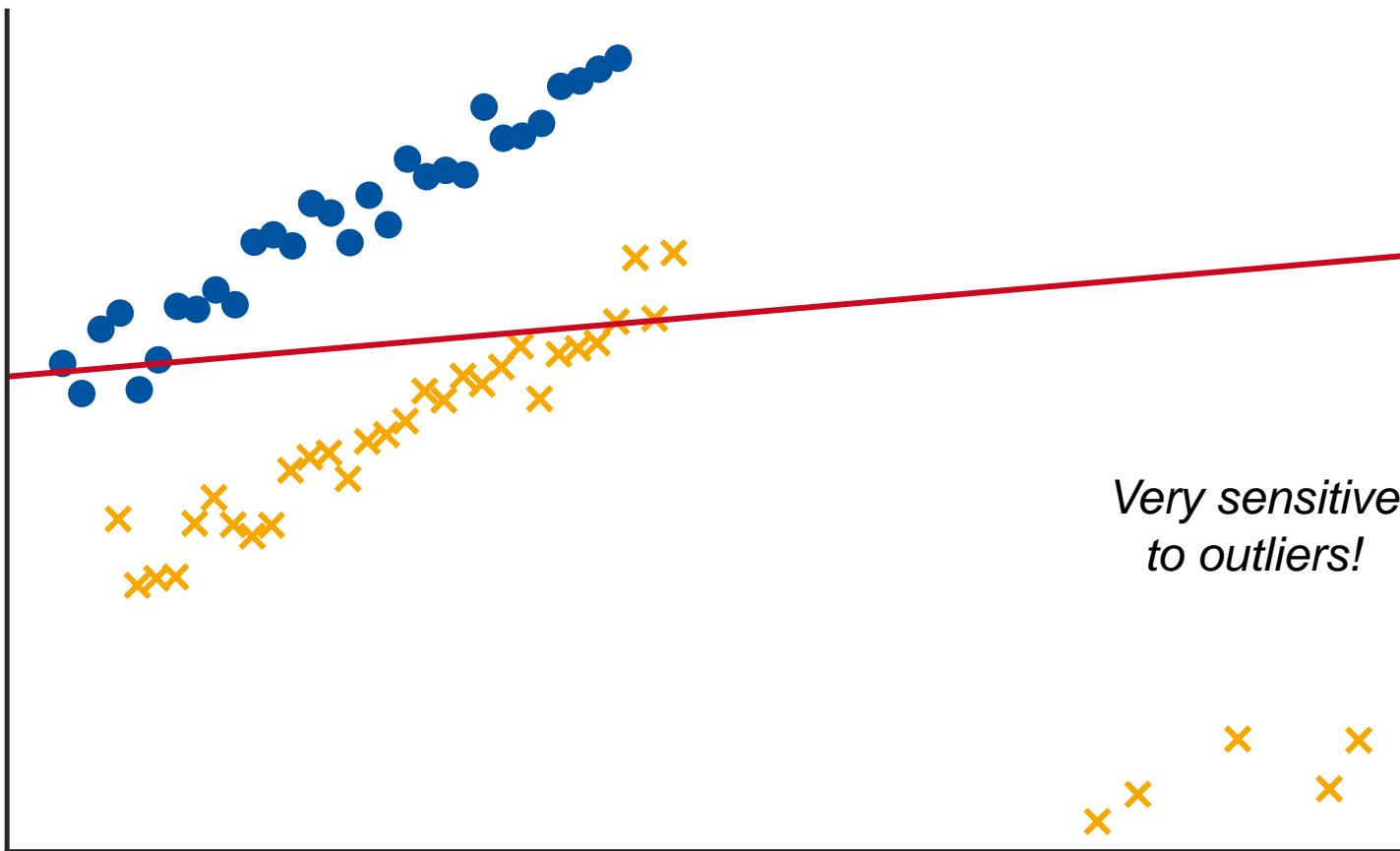
Example: Two Classes



Example: Two Classes



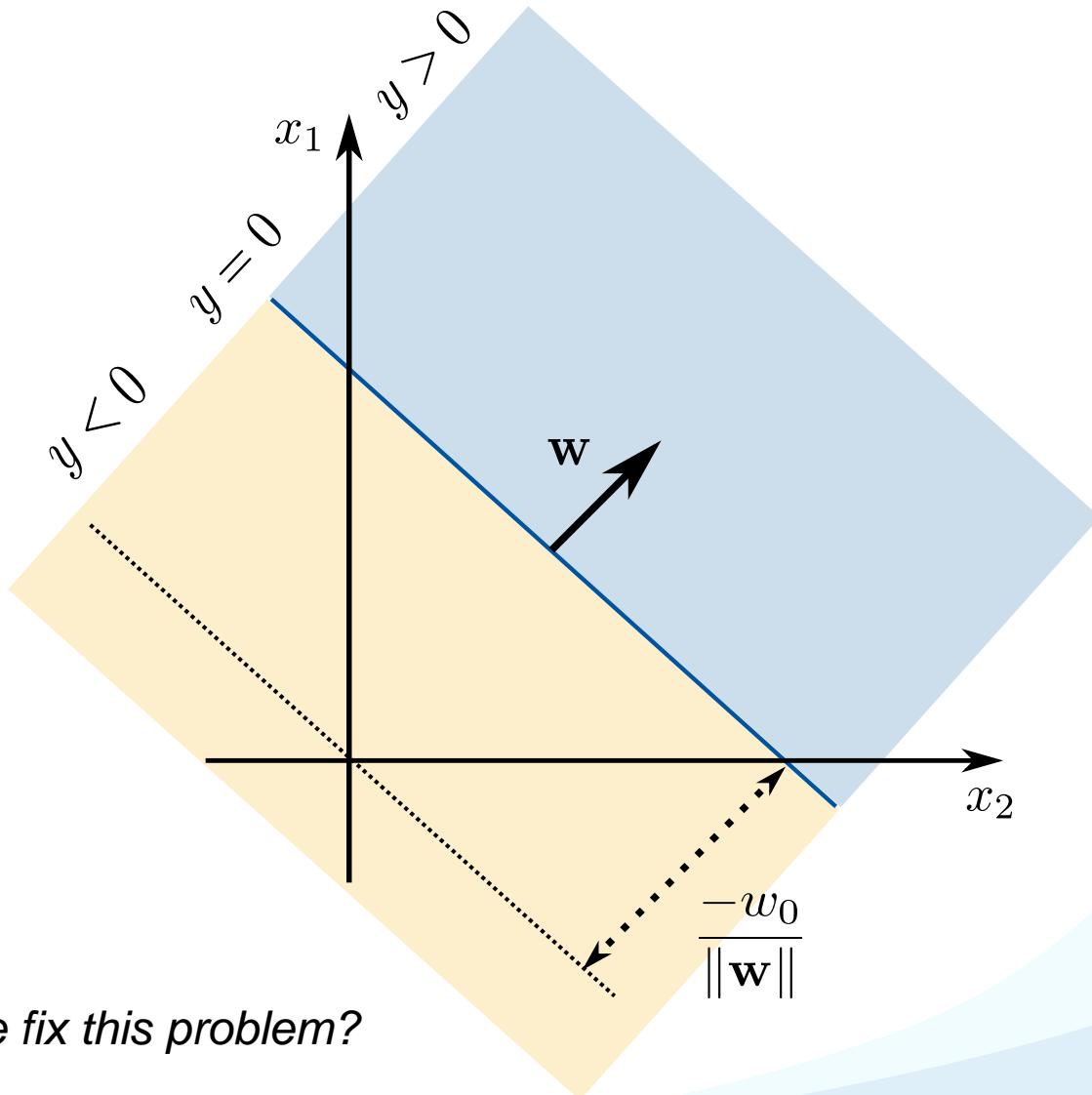
Example: Two Classes



Why Does This Happen?

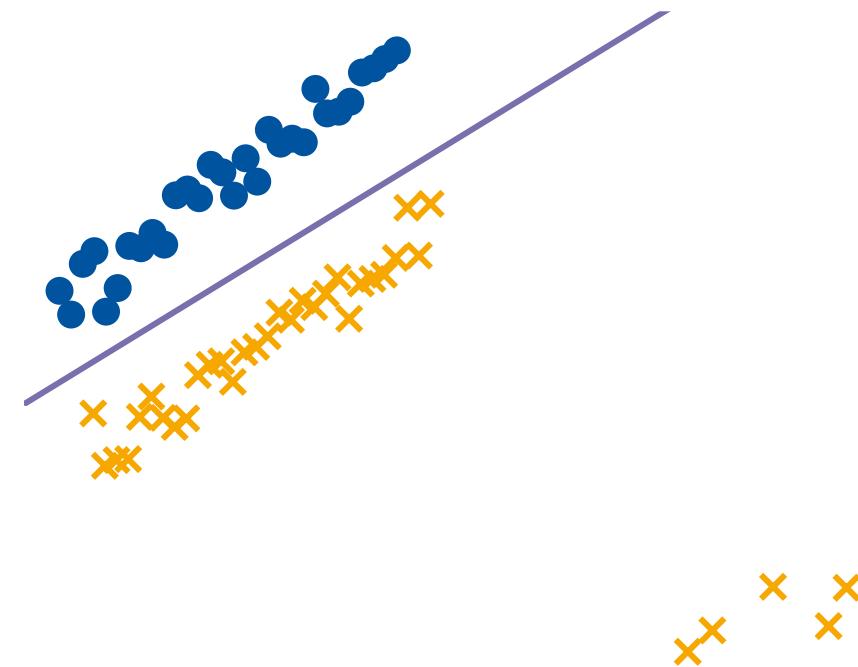
$$y(\mathbf{x}) = \mathbf{w}^T \mathbf{x} + w_0$$

- Remember the interpretation of $y(\mathbf{x})$
 - Normal equation of a hyperplane
- $y(\mathbf{x})$ measures the (signed) distance of the point \mathbf{x} from the hyperplane.
- However, we now compare it to a target value of $t_n \in \{-1, 1\}$...



Linear Discriminants

1. Motivation: Discriminant Functions
2. Linear Discriminant Functions
3. Least-Squares Classification
4. **Generalized Linear Discriminants**
5. Basis Functions



Generalized Linear Models

- So far: model classification by linear discriminant function

$$y(\mathbf{x}) = \mathbf{w}^T \mathbf{x} + w_0$$

- Generalize this with an activation function $g(\cdot)$:

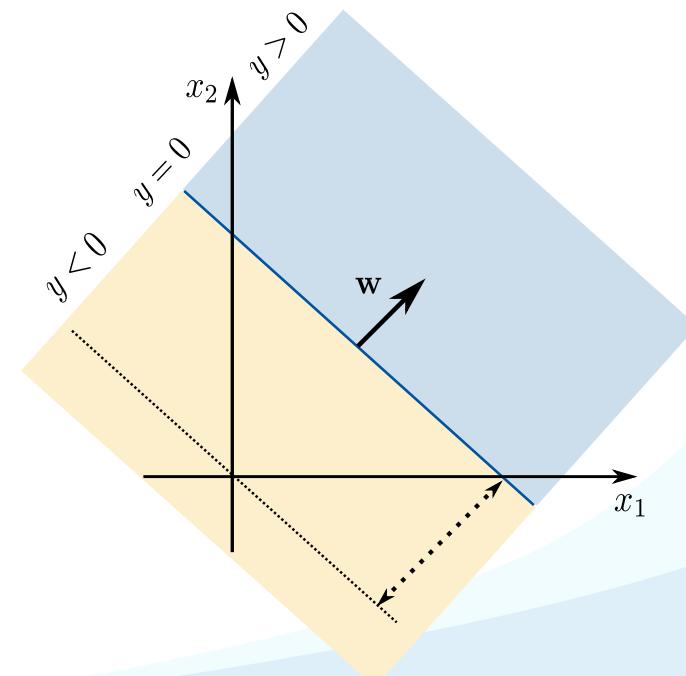
$$y(\mathbf{x}) = g(\mathbf{w}^T \mathbf{x} + w_0)$$

- Remarks
 - $g(\cdot)$ may be non-linear.
 - Decision surfaces correspond to

$$y(\mathbf{x}) = \text{const} \iff \mathbf{w}^T \mathbf{x} + w_0 = \text{const}$$

\Rightarrow If $g(\cdot)$ is monotonous (which is typically the case),
the decision boundaries are still linear functions of \mathbf{x} .

Generalized Linear Model



Activation Functions

- Recall least-squares classification:
 - Outliers have strong influence

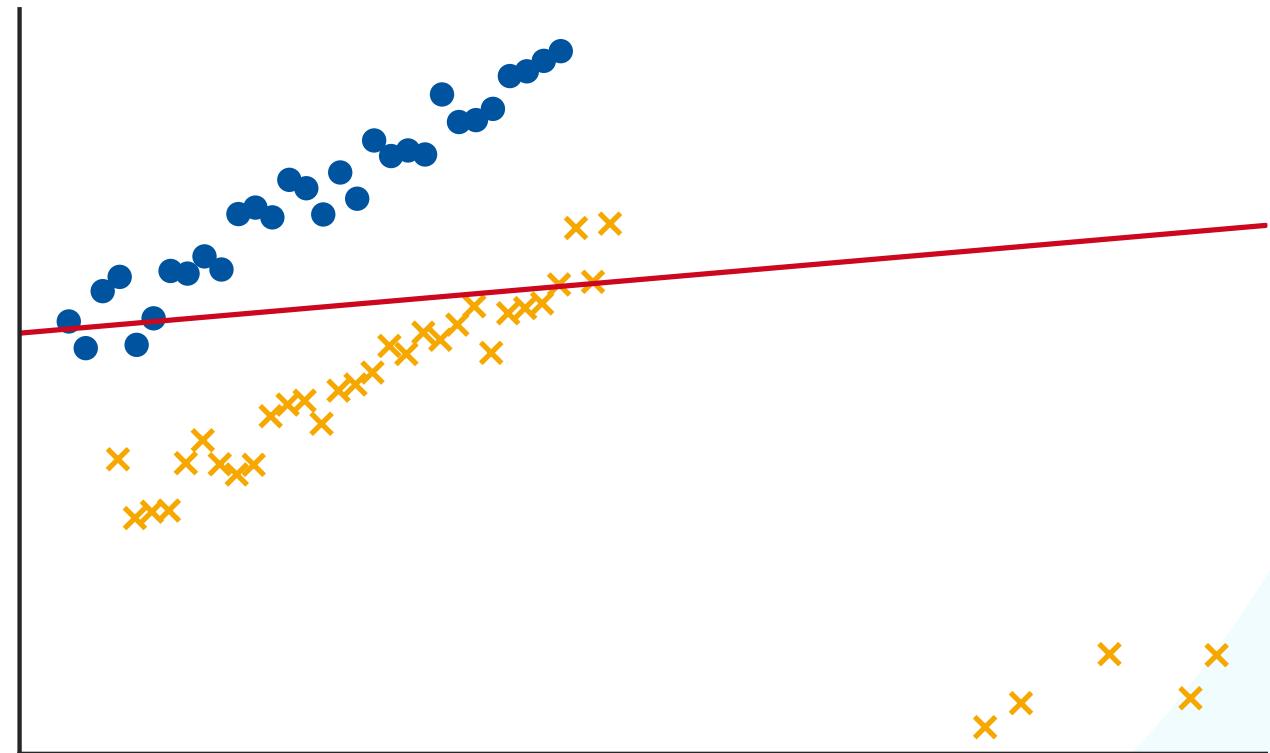
$$E(\mathbf{w}) = \sum_{n=1}^N (y(\mathbf{x}_n; \mathbf{w}) - t_n)^2$$

- This is because the output $y(\mathbf{x}; \mathbf{w})$ can grow arbitrarily large:

$$y(\mathbf{x}; \mathbf{w}) = \mathbf{w}^\top \mathbf{x} + w_0$$

- Choosing a suitable nonlinearity can limit those influences:

$$y(\mathbf{x}; \mathbf{w}) = g(\mathbf{w}^\top \mathbf{x} + w_0)$$



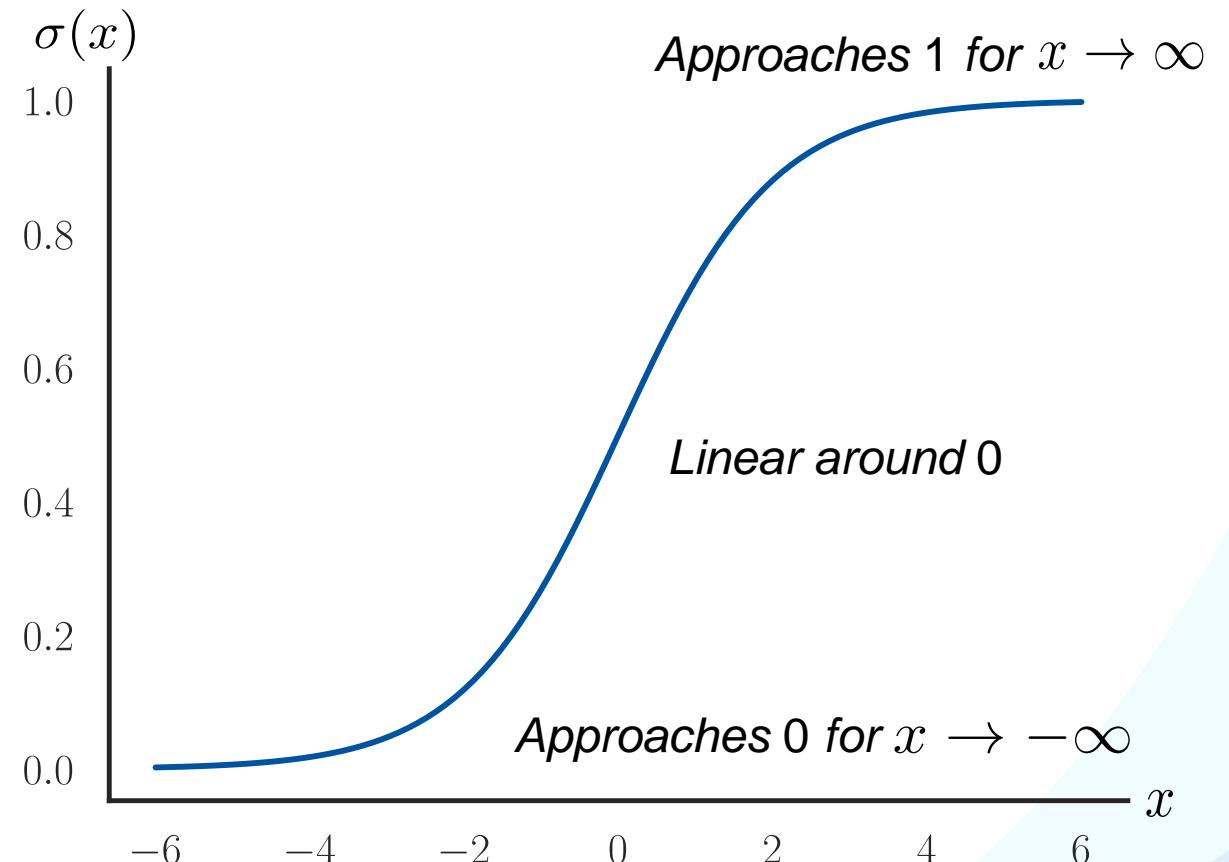
The Logistic Sigmoid

- To limit the influence of outliers, we can use the **logistic sigmoid** function:

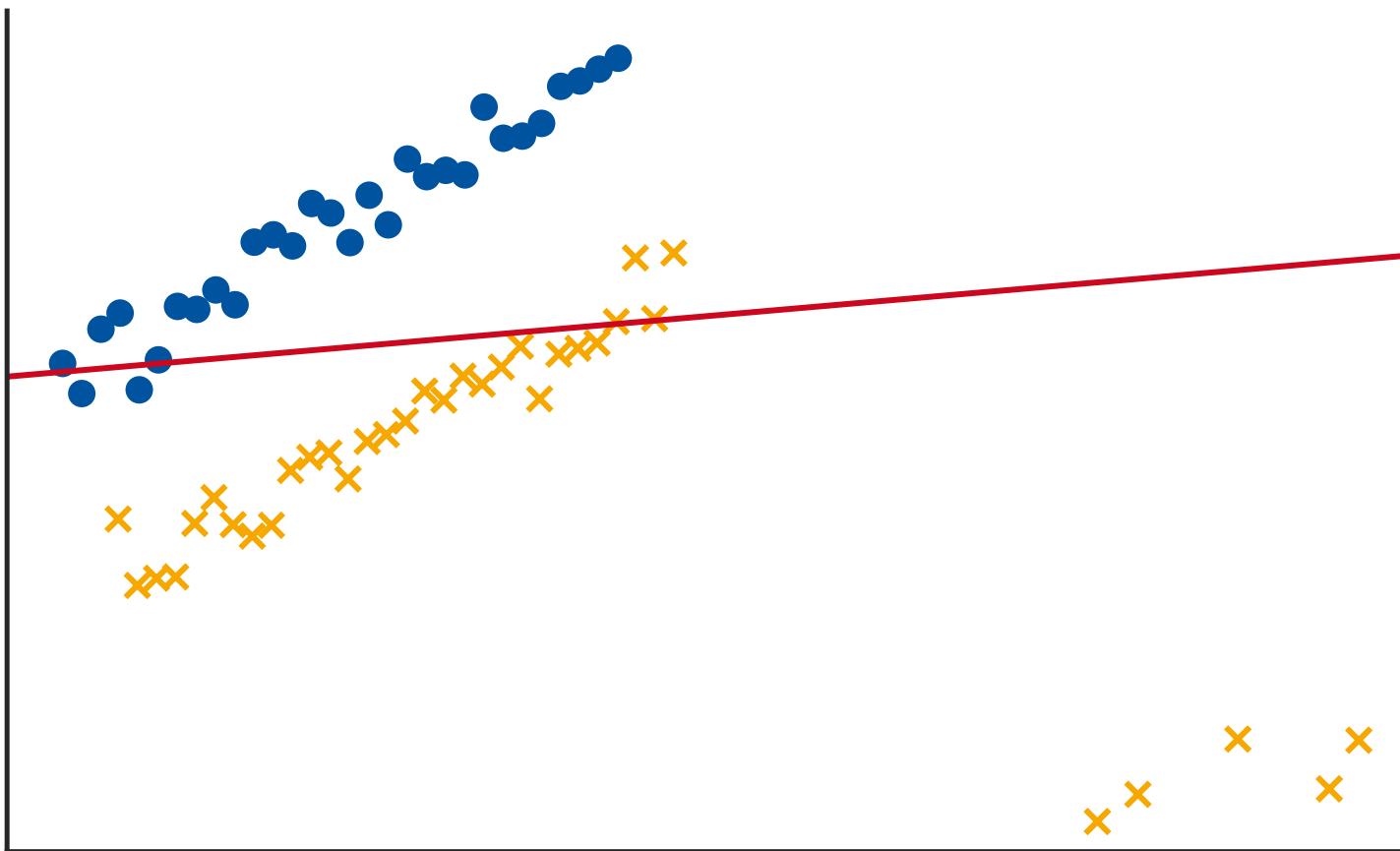
$$\sigma(x) = \frac{1}{1 + \exp(-x)}$$

- For 2-class problems, we scale it to the output range (-1,1) (known as **tangens hyperbolicus**):

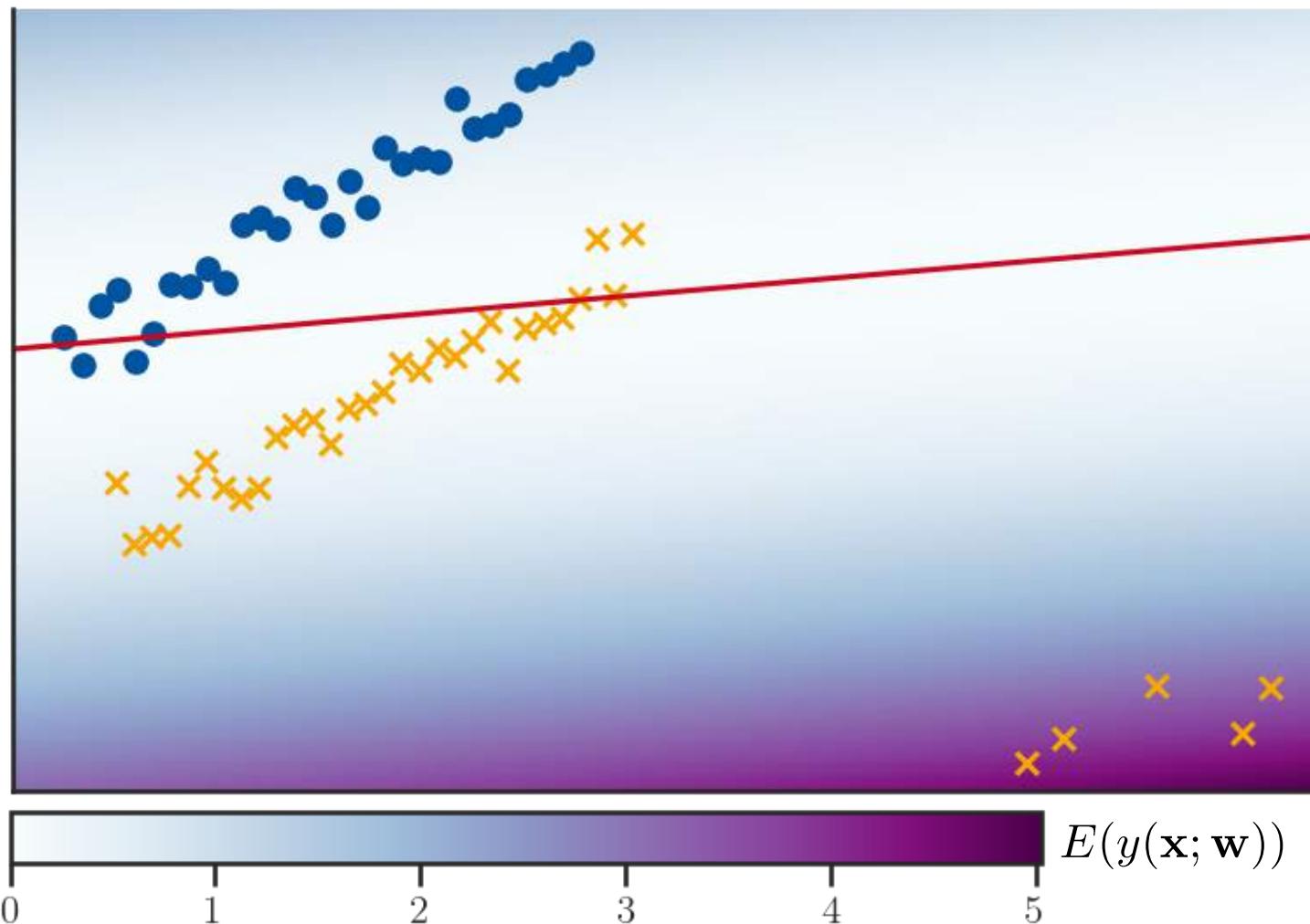
$$\tanh(x) = 2\sigma(x) - 1$$



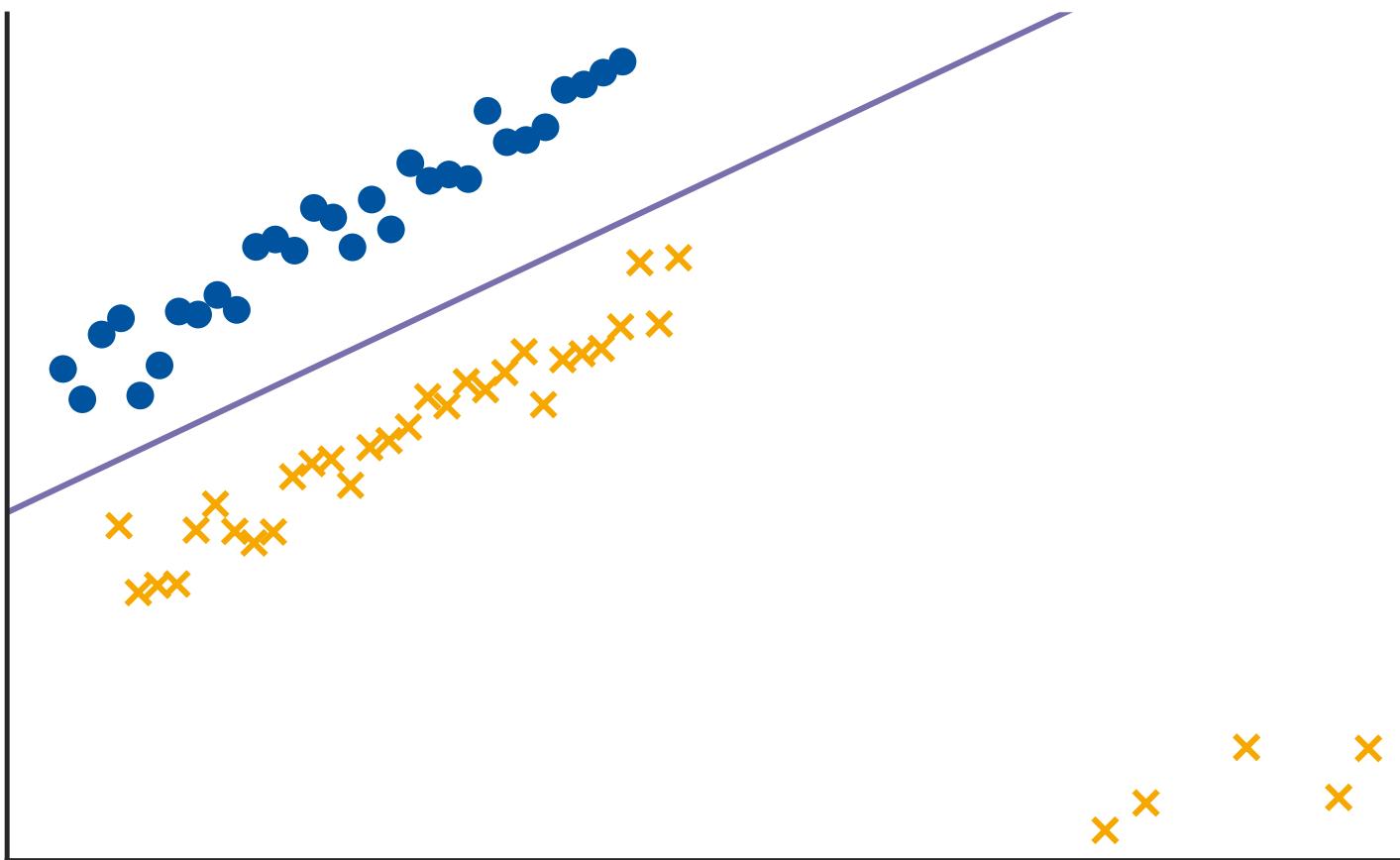
Example



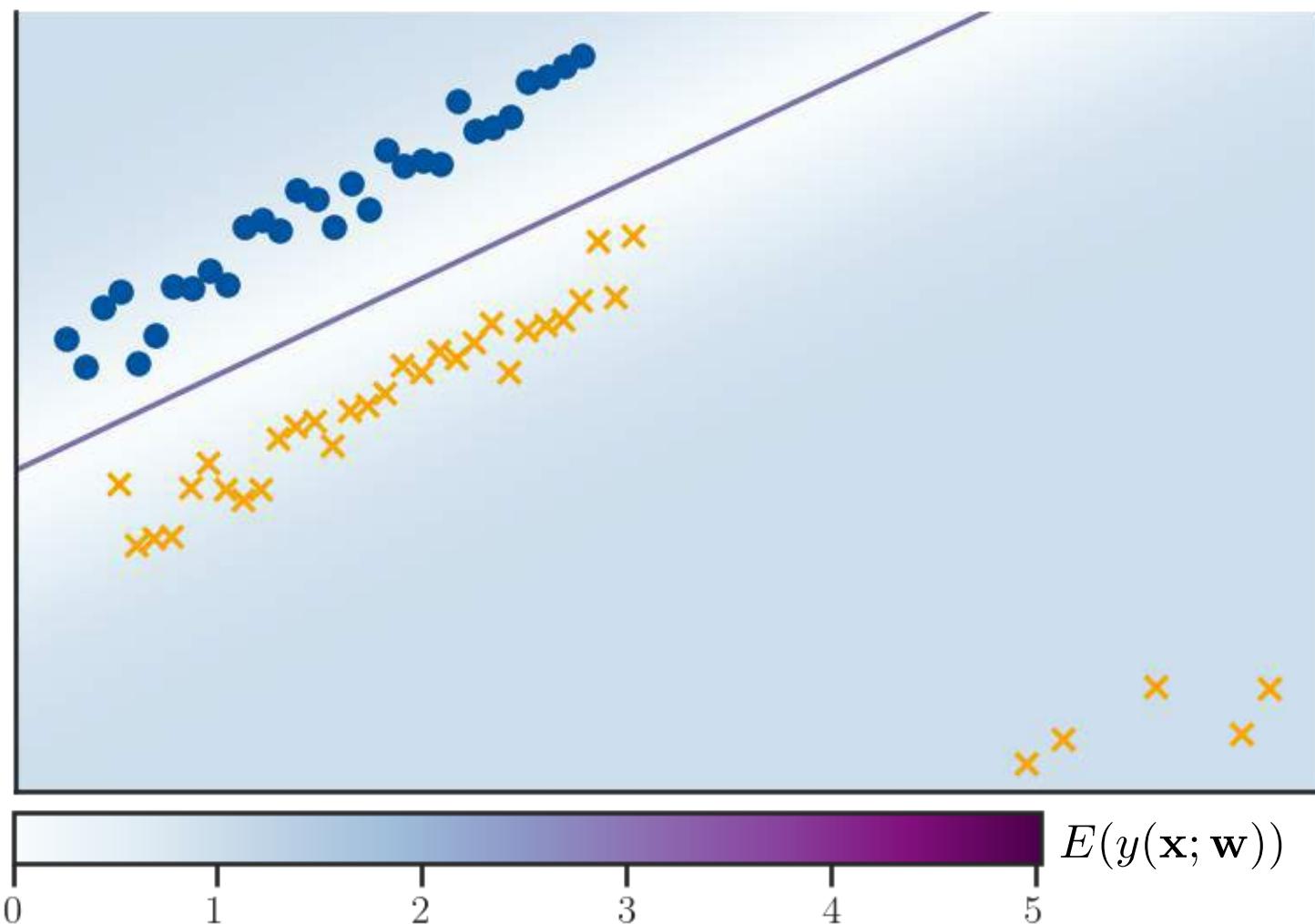
Example



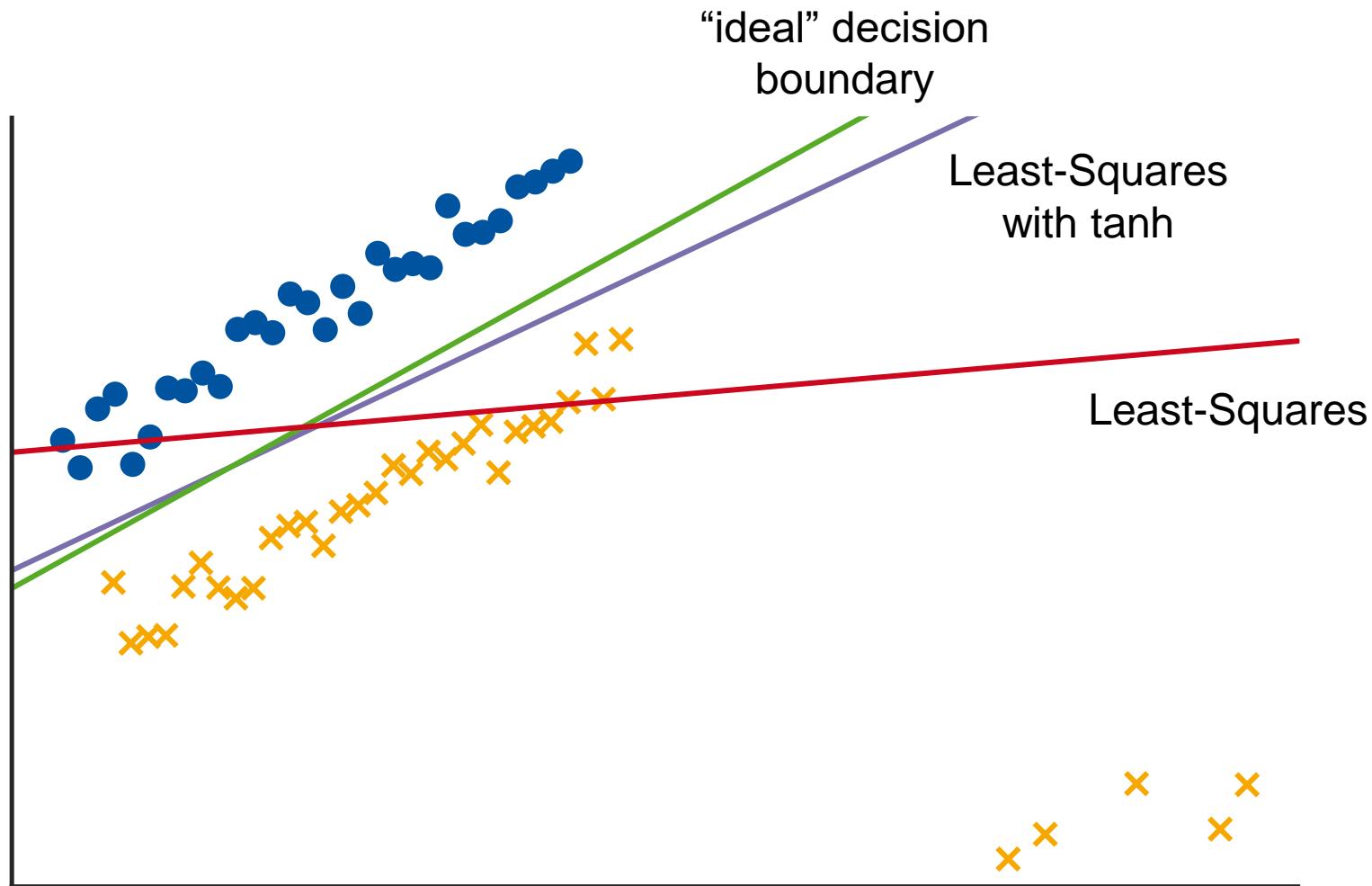
Example



Example



Example



Discussion: Activation Functions

Advantages

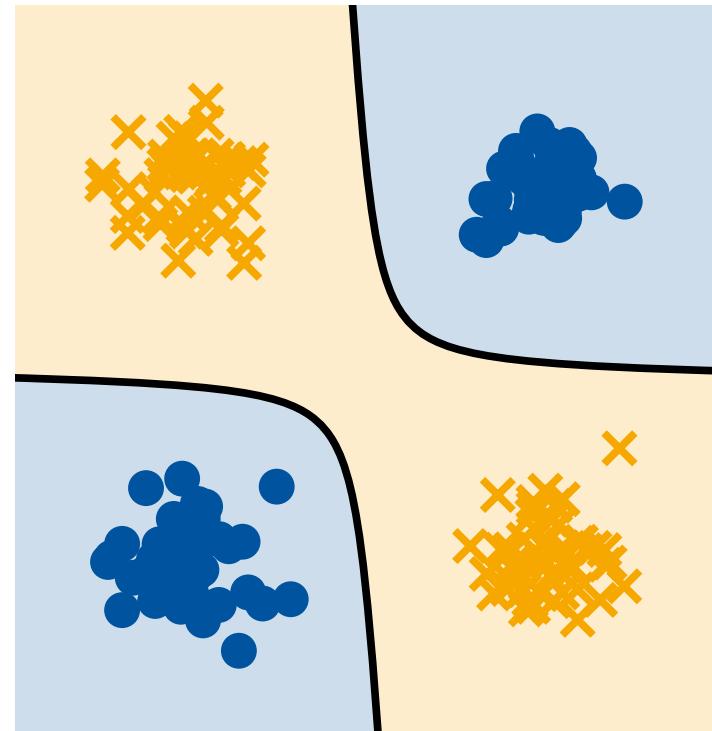
- Nonlinearity gives more flexibility.
- Can be used to limit the effect of outliers.
- Choice of sigmoid actually has a nice probabilistic interpretation.

Limitations

- Least-squares minimization in general no longer leads to a closed-form analytical solution.
⇒ Need to apply iterative methods.

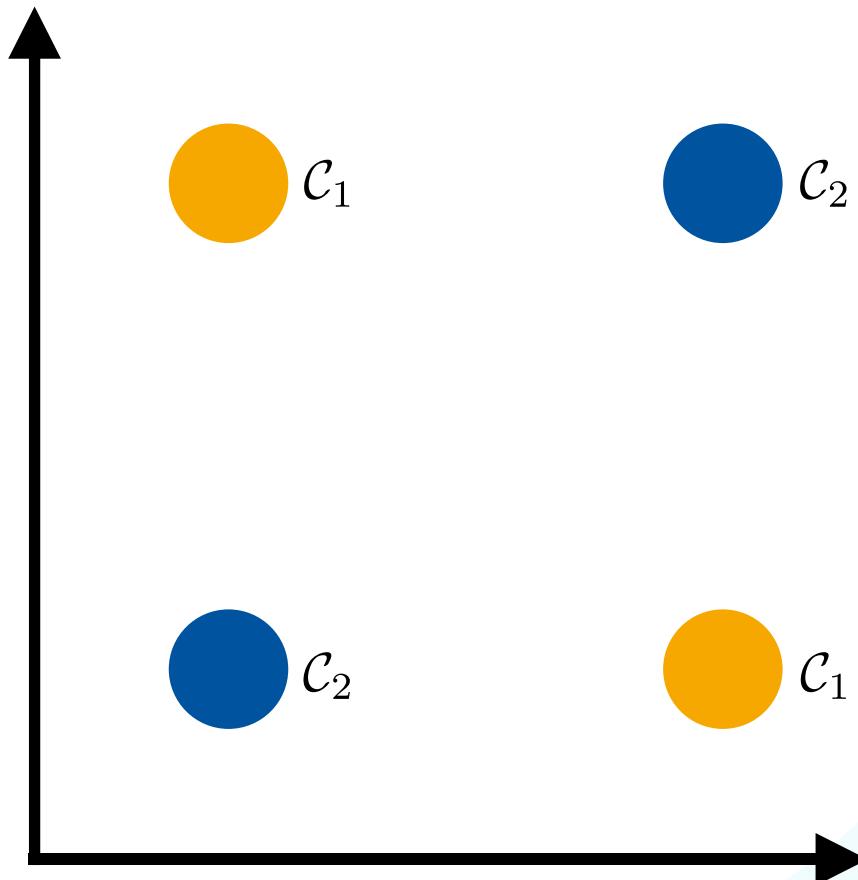
Linear Discriminants

1. Motivation: Discriminant Functions
2. Linear Discriminant Functions
3. Least-Squares Classification
4. Generalized Linear Discriminants
5. **Basis Functions**



Basis Functions

- So far: assumed linear separability
 - Very restrictive assumption, classical counterexample: XOR
 - We need non-linear decision boundaries...
- Solution: use non-linear **basis functions** $\phi_j(\mathbf{x})$:
$$y(\mathbf{x}) = \sum_{j=1}^M w_j \phi_j(\mathbf{x}) + w_0$$
- By choosing the right ϕ , every continuous function can (in principle) be approximated with arbitrary accuracy



Intuition

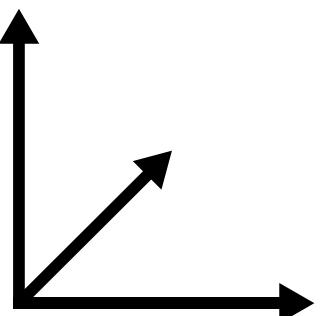
$$y_k(\mathbf{x}) = \sum_{j=0}^M w_{kj} \phi_j(\mathbf{x}) = \mathbf{w}_k^\top \phi(\mathbf{x})$$

This is still a **linear problem** in $\phi(\mathbf{x})$.

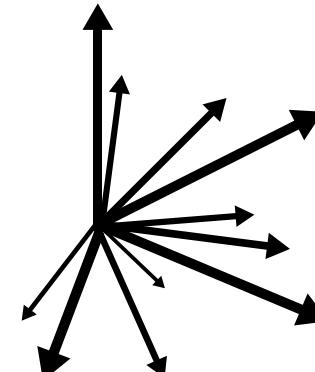
$\phi_j(\mathbf{x})$ are called **basis functions**.

But, depending on $\phi(\cdot)$, it may now be a nonlinear problem in \mathbf{x} .

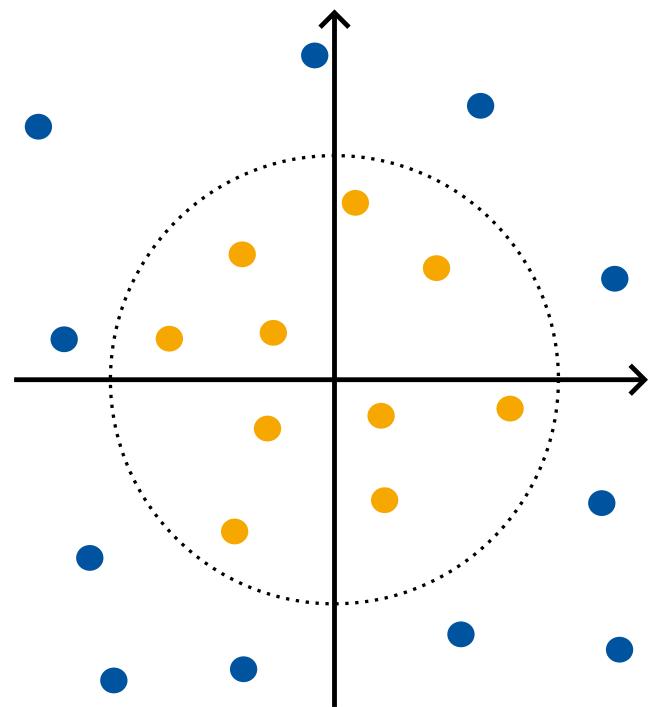
Typically, $\phi_0(\mathbf{x}) = 1$ so that w_0 acts as a bias.



$$\phi : \mathbb{R}^D \mapsto \mathbb{R}^M$$

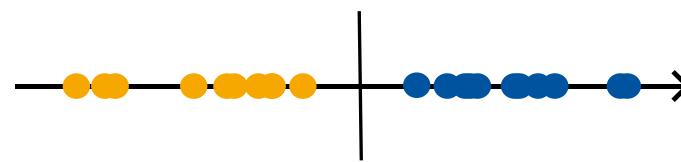


Usually, ϕ maps into a higher-dimensional space.



Not linearly separable

$$\phi(\mathbf{x}) = x_1^2 + x_2^2$$



Linearly separable

Example: Polynomial Basis Functions

- Polynomial basis functions map x to powers of x :

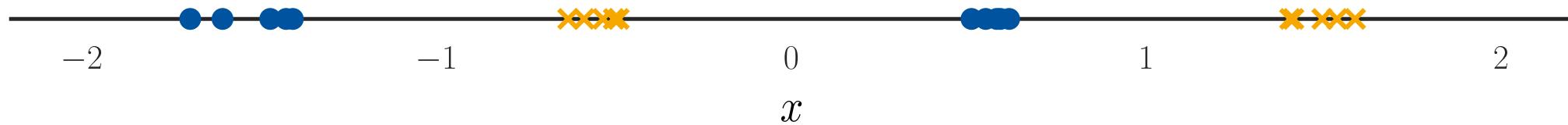
$$\phi(x) = (x^m, x^{m-1}, \dots, x, 1)^T$$

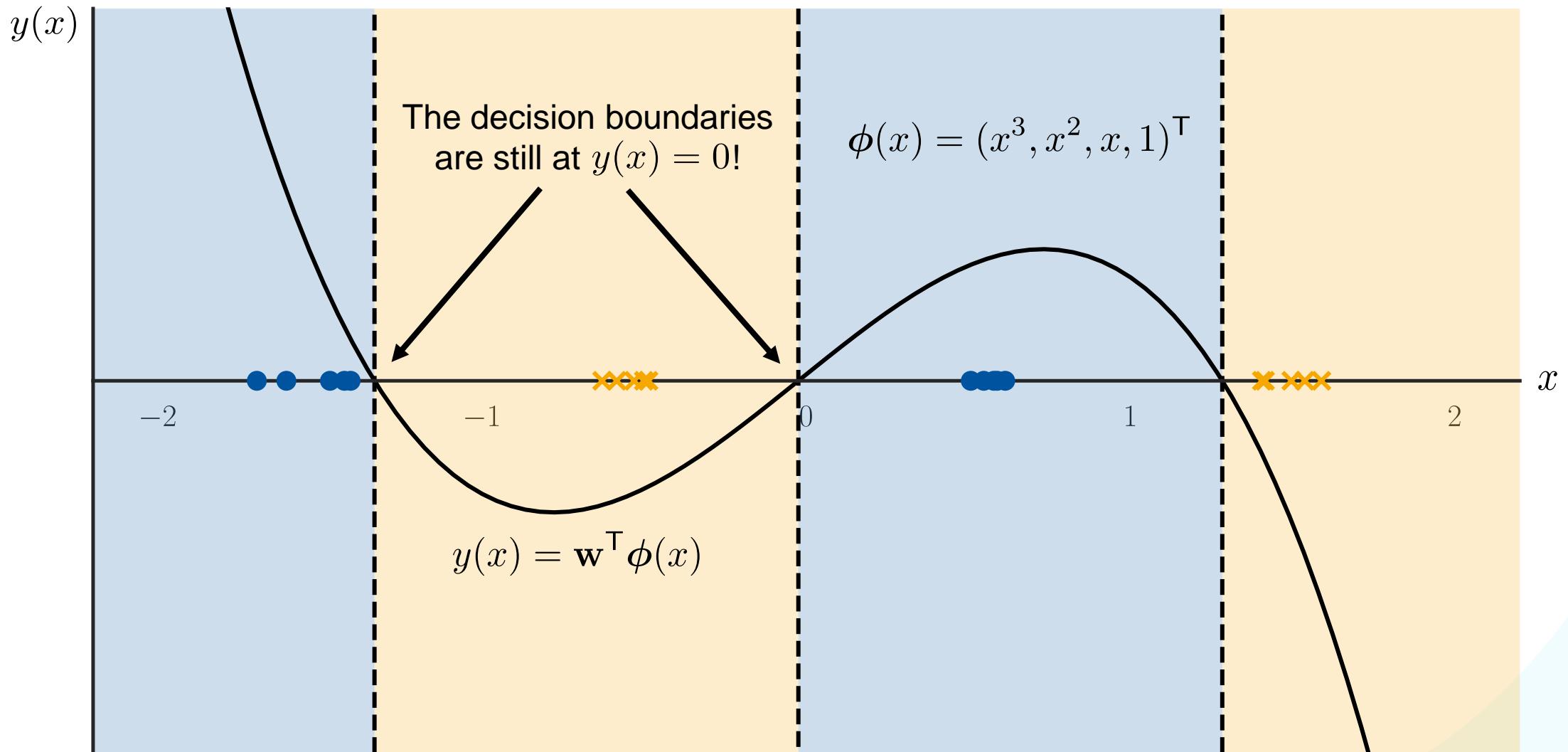
- When we optimize $\mathbf{w}^T \phi(x)$ with polynomial basis functions, we implicitly optimize the coefficients of a polynomial in x :

$$\begin{aligned}y(x) &= \mathbf{w}^T \phi(x) \\&= w_m x^m + w_{m-1} x^{m-1} + \dots + w_1 x + w_0\end{aligned}$$

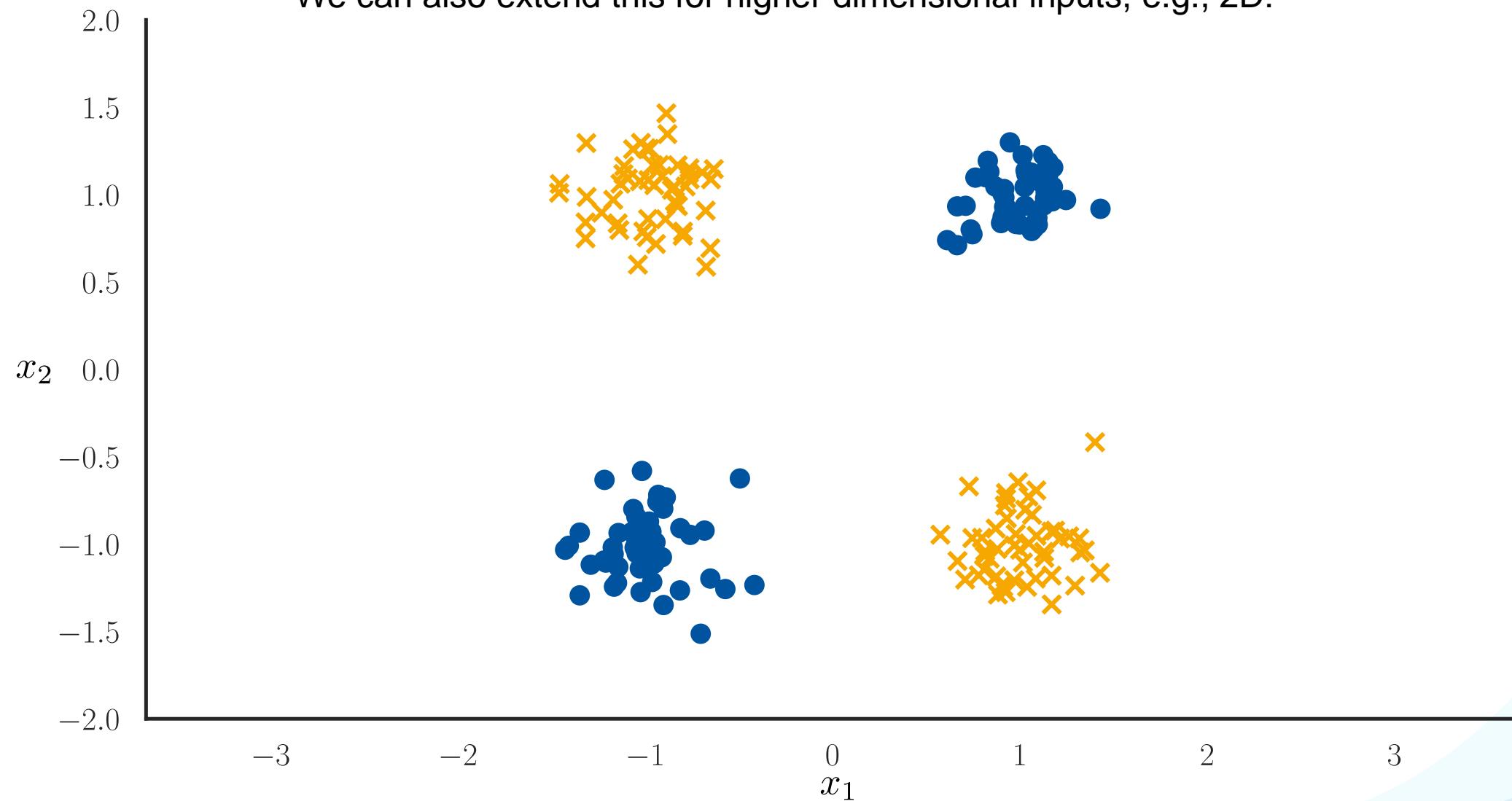
- As before, we decide for \mathcal{C}_1 if $y(x) > 0$.

Let's use a third-degree polynomial: $\phi(x) = (x^3, x^2, x, 1)^\top$

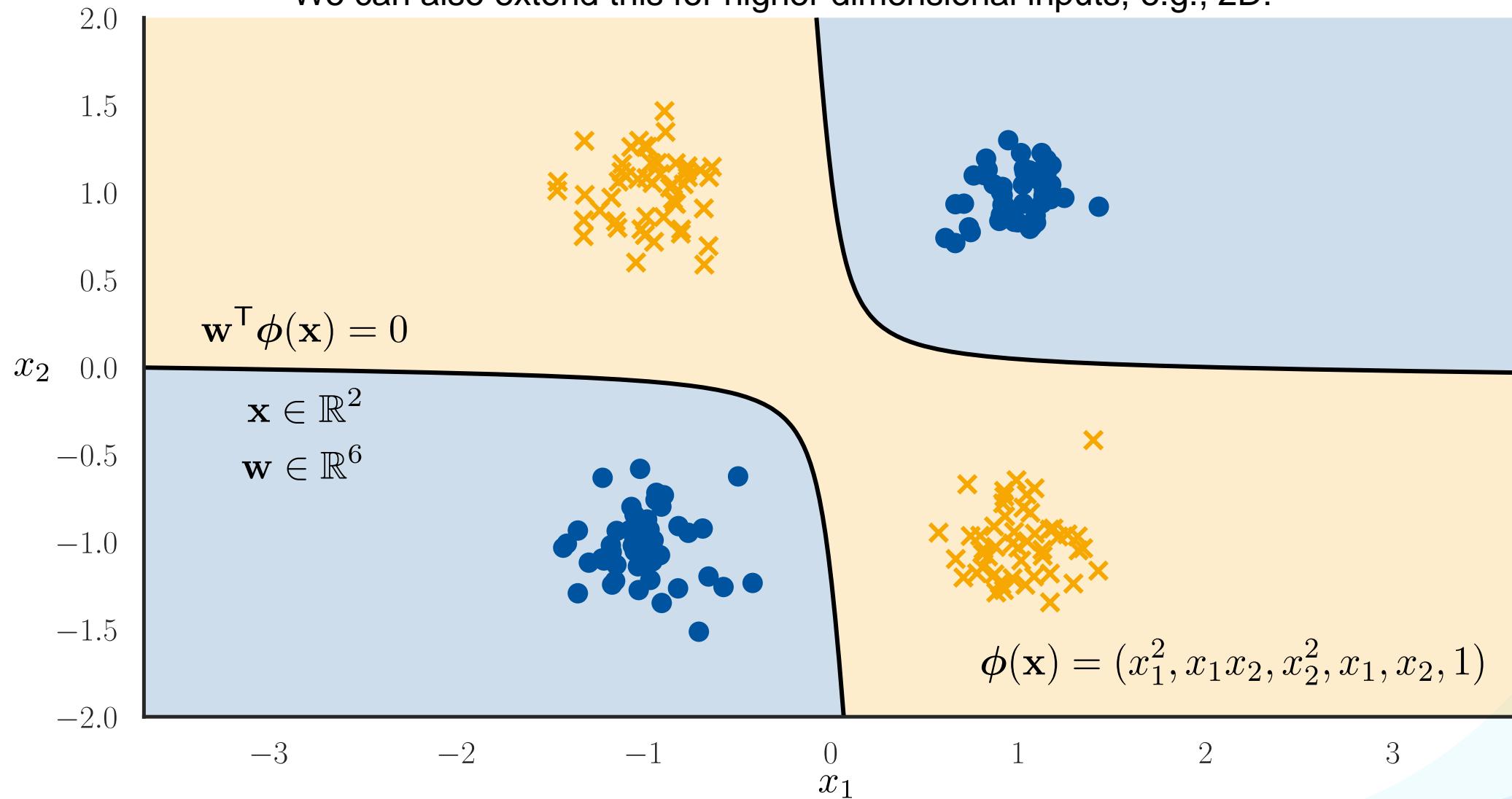




We can also extend this for higher dimensional inputs, e.g., 2D:



We can also extend this for higher dimensional inputs, e.g., 2D:



Discussion: Basis Functions

Advantages

- Basis functions allow us to address linearly non-separable problems
- The problem is still linear in $\phi(\mathbf{x})$ (but may be nonlinear in \mathbf{x}).
- We can think of $\phi(\mathbf{x})$ as transforming the data into a feature space in which the problem is easier to solve.
- In general, it is easier to find a separating hyperplane in higher-dimensional spaces.

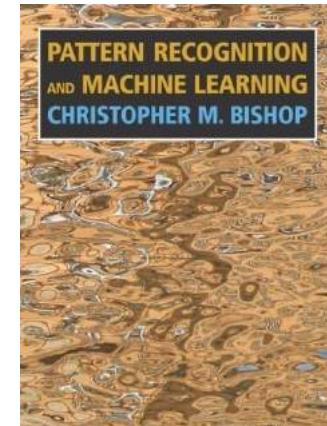
Limitations

- The right choice of $\phi(\mathbf{x})$ depends on the problem and is another hyperparameter to optimize.
- Flexibility is limited by the curse of dimensionality. Evaluating $\mathbf{w}^T \phi(\mathbf{x})$ can be expensive in high-dimensional spaces.
- Choosing a higher-dimensional feature space $\phi(\mathbf{x})$ increases the capacity of the classifier and may lead to overfitting.

References and Further Reading

- More information about Linear Discriminants is available in Chapter 4.1 of Bishop's book.

Christopher M. Bishop
Pattern Recognition and Machine Learning
Springer, 2006



Elements of Machine Learning & Data Science

Winter semester 2023/24

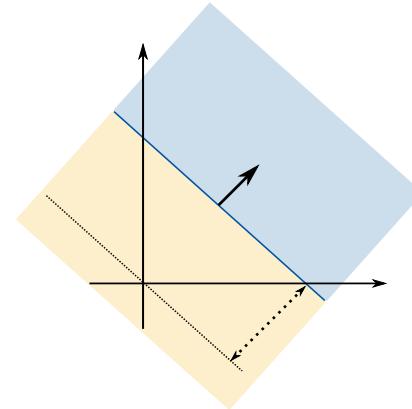
Lecture 15 – Linear Regression

05.12.2023

Prof. Bastian Leibe

Machine Learning Topics

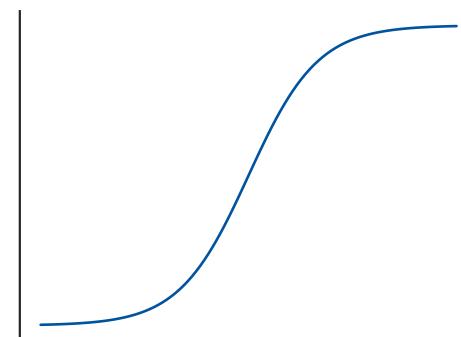
1. Introduction to ML
2. Probability Density Estimation
- 3. Linear Discriminants**
4. Linear Regression
5. Logistic Regression
6. Support Vector Machines
7. AdaBoost
8. Neural Network Basics



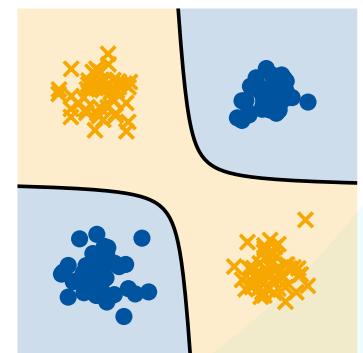
Linear Discriminant Functions

$$E(\mathbf{w}) = \frac{1}{2} \sum_n (y(\mathbf{x}_n; \mathbf{w}) - t_n)^2$$

Least-Squares Classification



Activation Functions



Basis Functions

Recap: Generalized Linear Models

- So far: model classification by linear discriminant function

$$y(\mathbf{x}) = \mathbf{w}^T \mathbf{x} + w_0$$

- Generalize this with an activation function $g(\cdot)$:

$$y(\mathbf{x}) = g(\mathbf{w}^T \mathbf{x} + w_0)$$

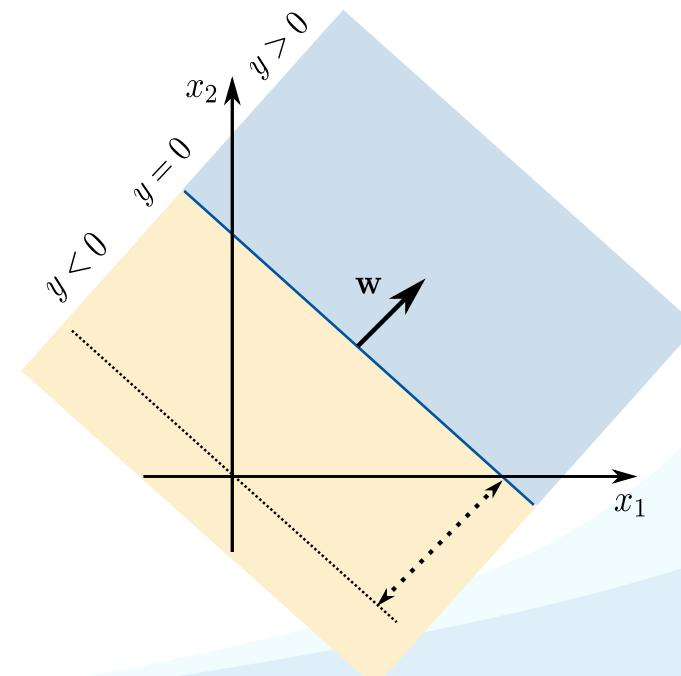
- Remarks

- $g(\cdot)$ may be non-linear.
 - Decision surfaces correspond to

$$y(\mathbf{x}) = \text{const} \iff \mathbf{w}^T \mathbf{x} + w_0 = \text{const}$$

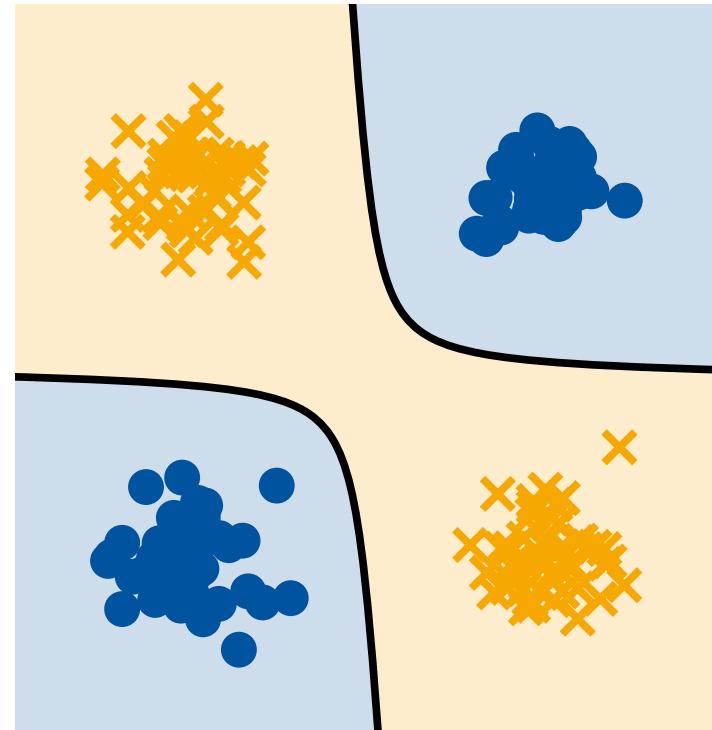
\Rightarrow If $g(\cdot)$ is monotonous (which is typically the case),
the decision boundaries are still linear functions of \mathbf{x} .

Generalized Linear Model



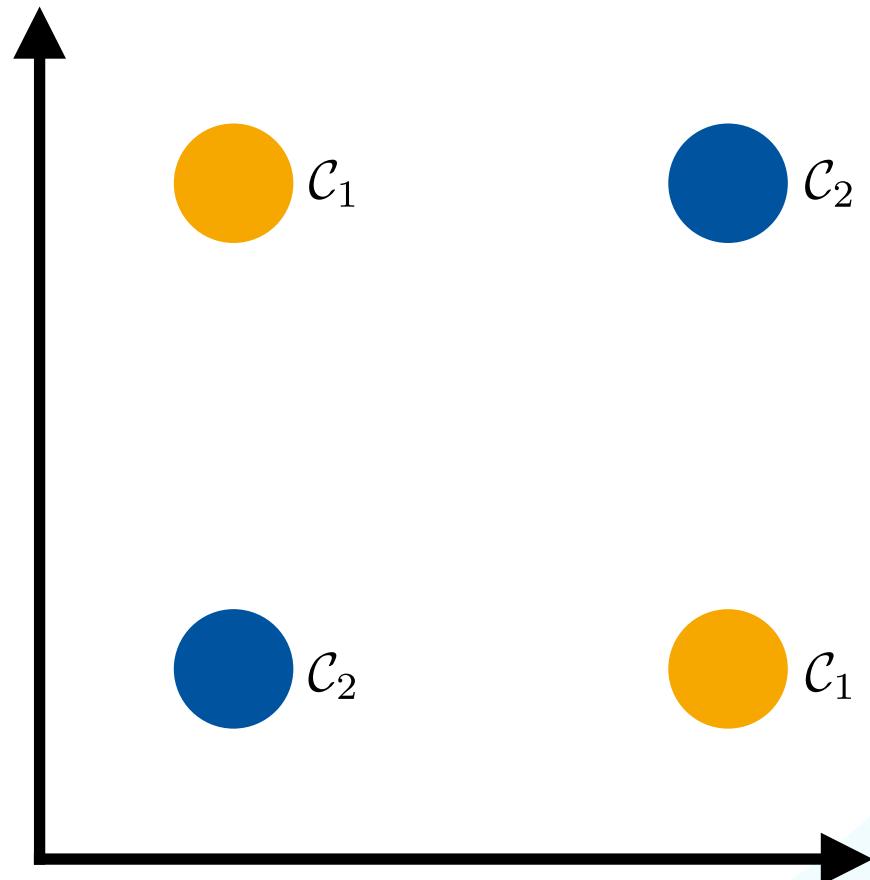
Linear Discriminants

1. Motivation: Discriminant Functions
2. Linear Discriminant Functions
3. Least-Squares Classification
4. Generalized Linear Discriminants
5. **Basis Functions**
6. Error Function Analysis



Basis Functions

- So far: assumed linear separability
 - Very restrictive assumption, classical counterexample: XOR
 - We need non-linear decision boundaries...
- Solution: use non-linear **basis functions** $\phi_j(\mathbf{x})$:
$$y(\mathbf{x}) = \sum_{j=1}^M w_j \phi_j(\mathbf{x}) + w_0$$
- By choosing the right ϕ , every continuous function can (in principle) be approximated with arbitrary accuracy



Intuition

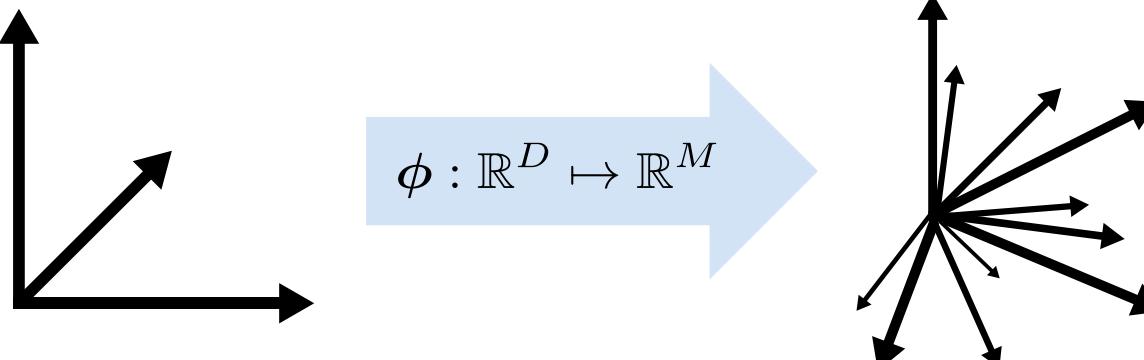
$$y_k(\mathbf{x}) = \sum_{j=0}^M w_{kj} \phi_j(\mathbf{x}) = \mathbf{w}_k^\top \phi(\mathbf{x})$$

This is still a **linear problem** in $\phi(\mathbf{x})$.

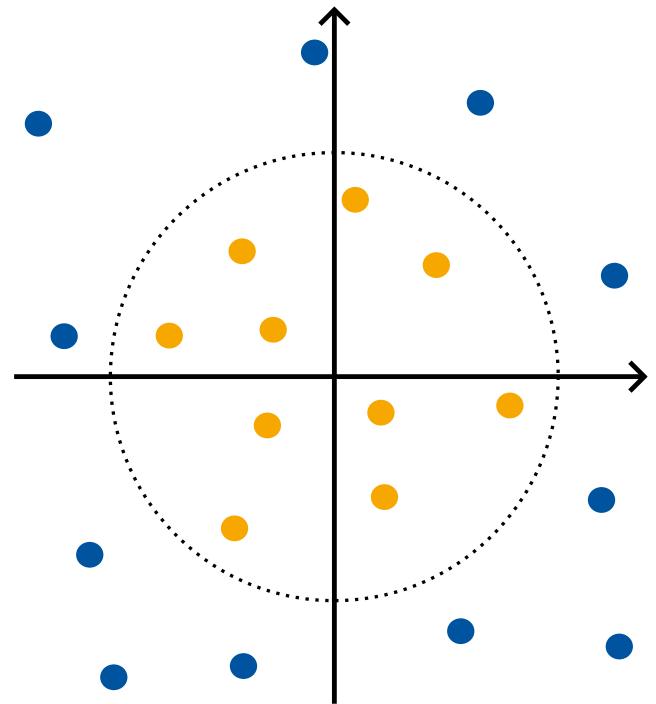
$\phi_j(\mathbf{x})$ are called **basis functions**.

But, depending on $\phi(\cdot)$, it may now be a nonlinear problem in \mathbf{x} .

Typically, $\phi_0(\mathbf{x}) = 1$ so that w_0 acts as a bias.

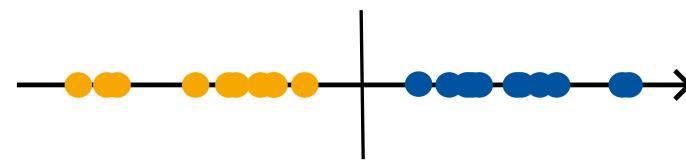


Usually, ϕ maps into a higher-dimensional space.



Not linearly separable

$$\phi(\mathbf{x}) = x_1^2 + x_2^2$$



Linearly separable

Example: Polynomial Basis Functions

- Polynomial basis functions map x to powers of x :

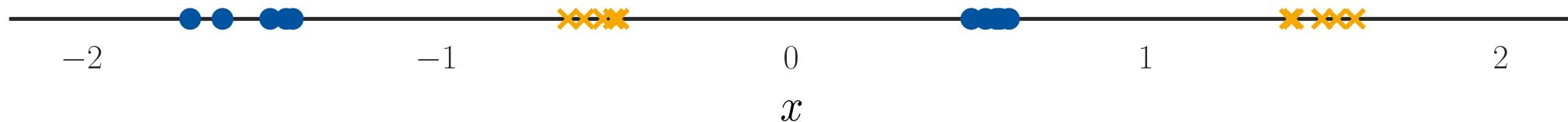
$$\phi(x) = (x^m, x^{m-1}, \dots, x, 1)^T$$

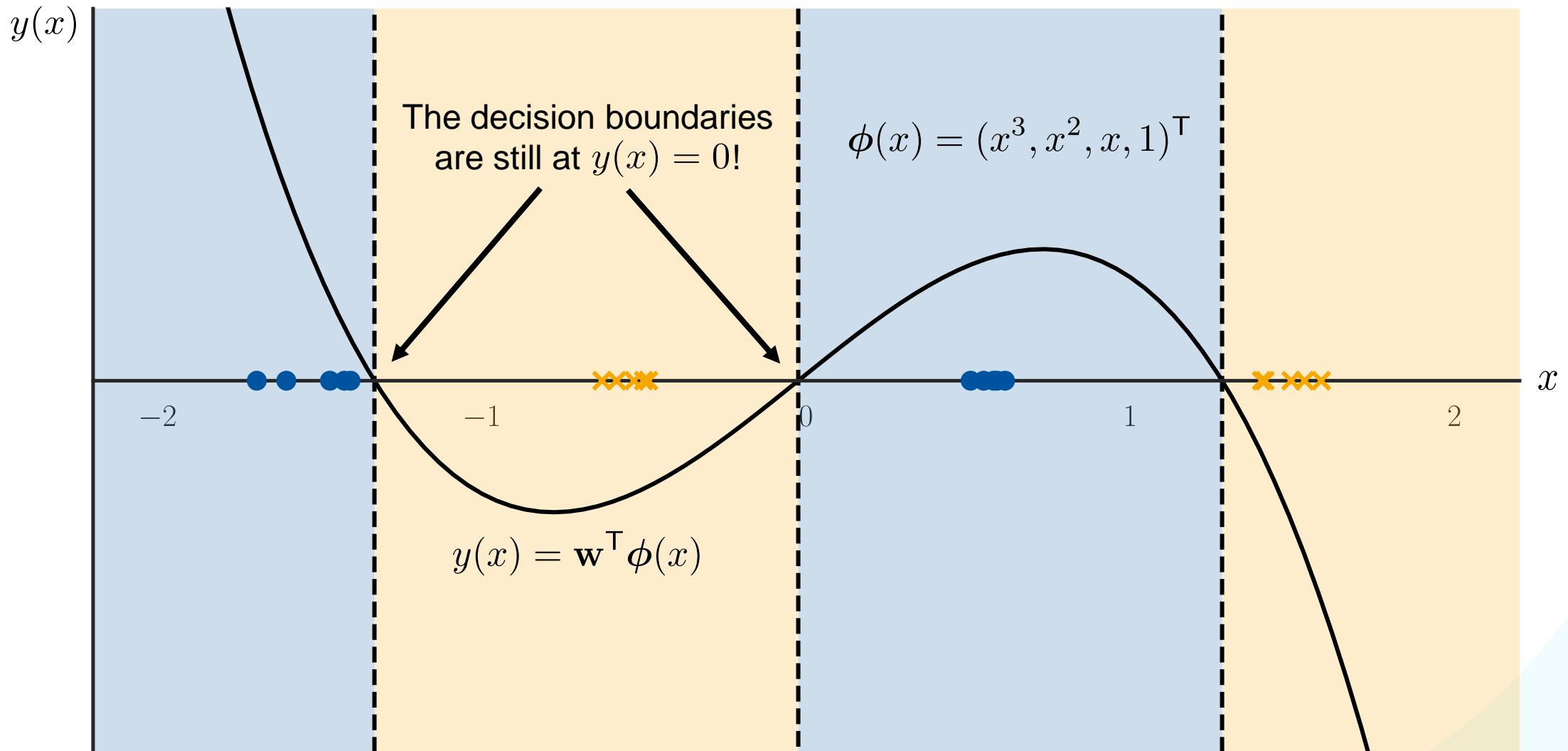
- When we optimize $\mathbf{w}^T \phi(x)$ with polynomial basis functions, we implicitly optimize the coefficients of a polynomial in x :

$$\begin{aligned}y(x) &= \mathbf{w}^T \phi(x) \\&= w_m x^m + w_{m-1} x^{m-1} + \dots + w_1 x + w_0\end{aligned}$$

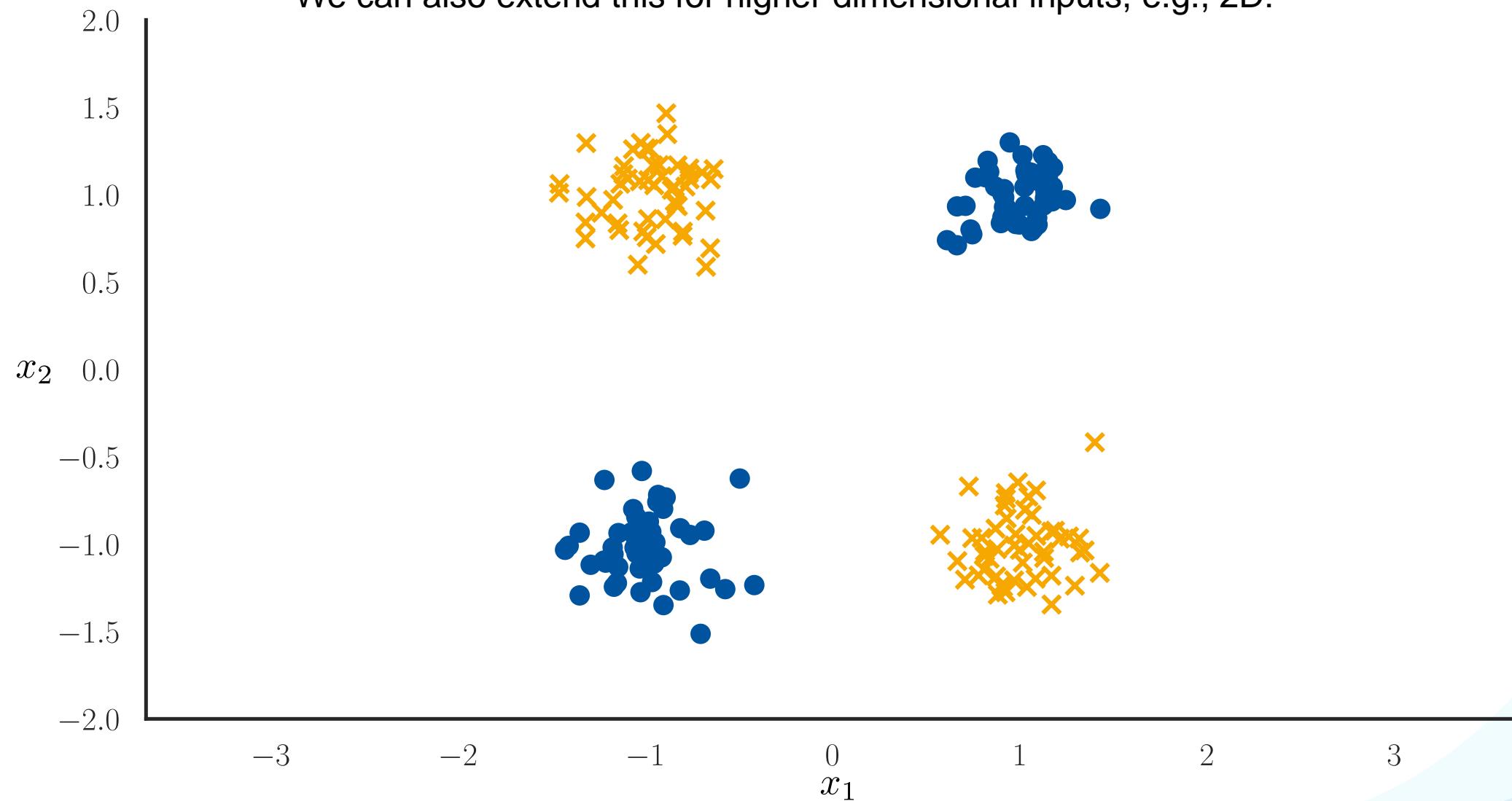
- As before, we decide for \mathcal{C}_1 if $y(x) > 0$.

Let's use a third-degree polynomial: $\phi(x) = (x^3, x^2, x, 1)^\top$

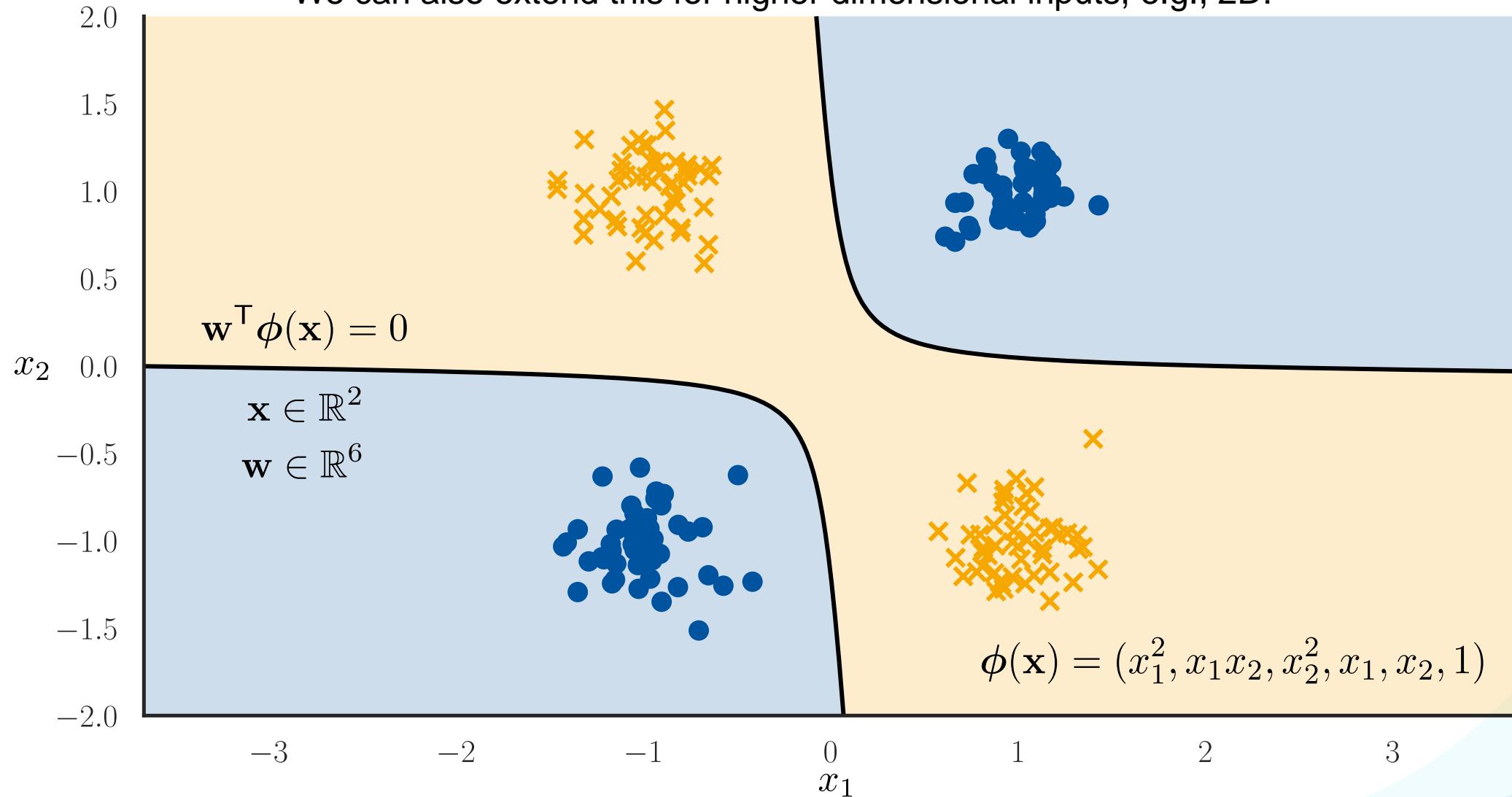




We can also extend this for higher dimensional inputs, e.g., 2D:



We can also extend this for higher dimensional inputs, e.g., 2D:



Discussion: Basis Functions

Advantages

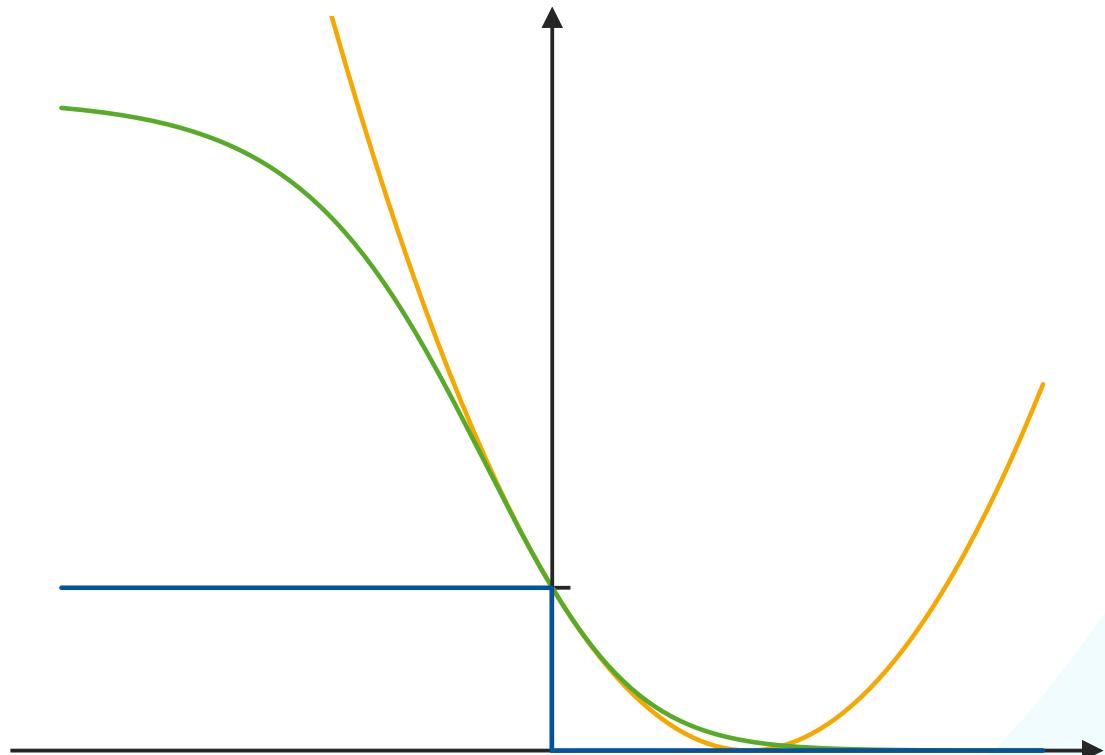
- Basis functions allow us to address linearly non-separable problems
- The problem is still linear in $\phi(\mathbf{x})$ (but may be nonlinear in \mathbf{x}).
- We can think of $\phi(\mathbf{x})$ as transforming the data into a feature space in which the problem is easier to solve.
- In general, it is easier to find a separating hyperplane in higher-dimensional spaces.

Limitations

- The right choice of $\phi(\mathbf{x})$ depends on the problem and is another hyperparameter to optimize.
- Flexibility is limited by the curse of dimensionality. Evaluating $\mathbf{w}^T \phi(\mathbf{x})$ can be expensive in high-dimensional spaces.
- Choosing a higher-dimensional feature space $\phi(\mathbf{x})$ increases the capacity of the classifier and may lead to overfitting.

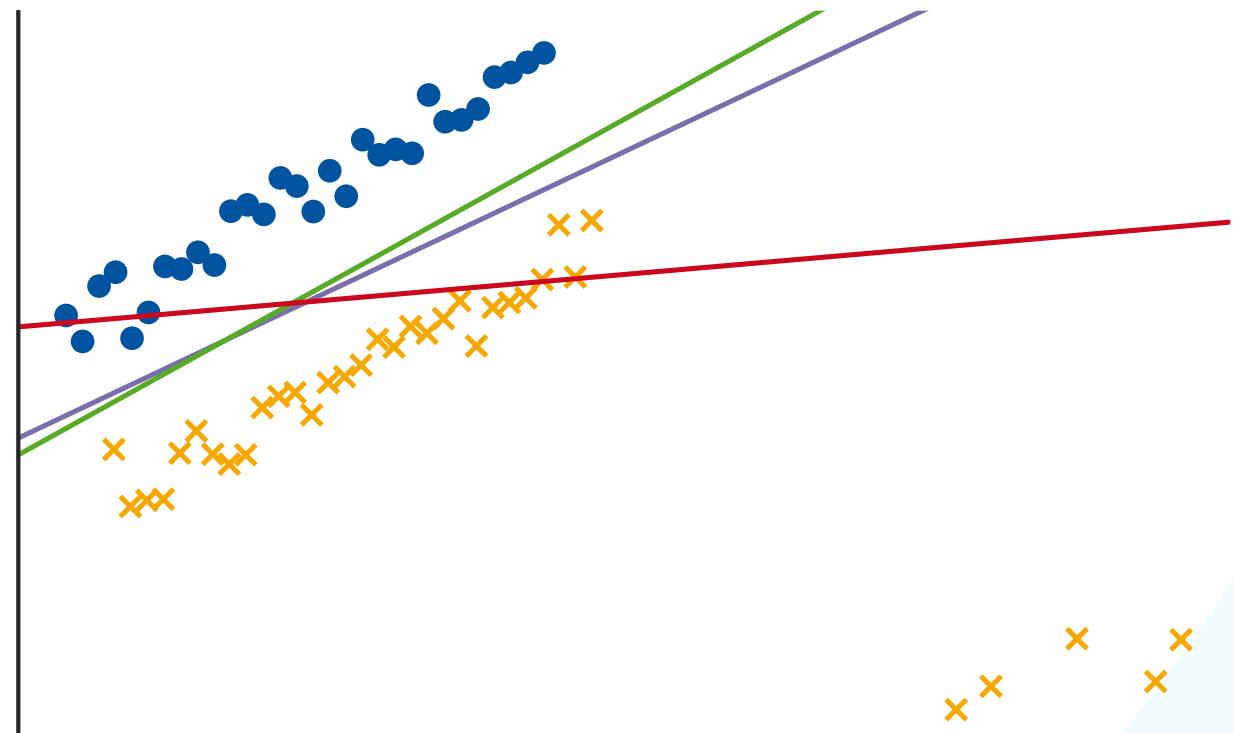
Linear Discriminants

1. Motivation: Discriminant Functions
2. Linear Discriminant Functions
3. Least-Squares Classification
4. Generalized Linear Discriminants
5. Basis Functions
6. **Error Function Analysis**

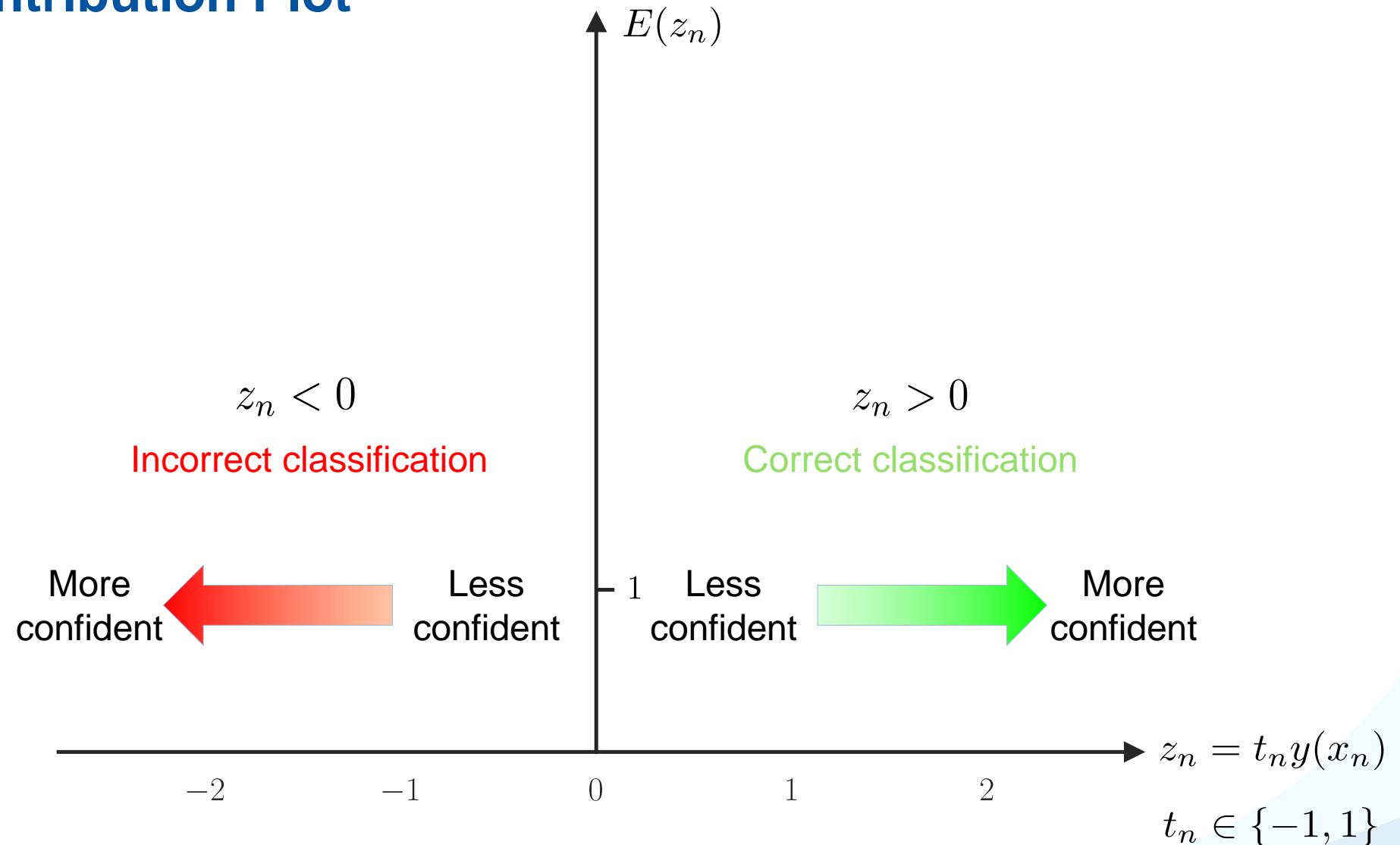


Error Function Analysis

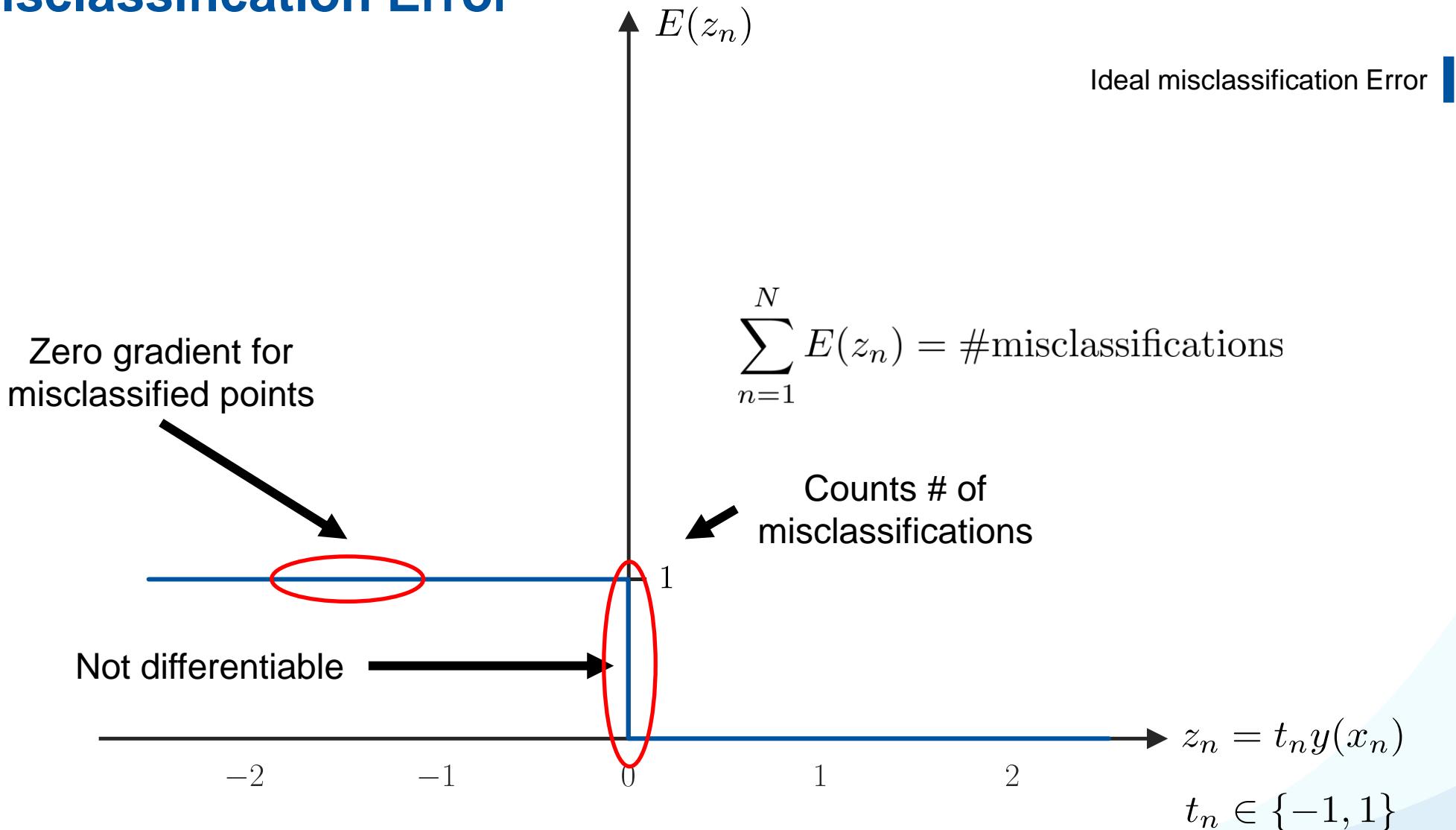
- We have seen how to learn **generalized linear discriminant** models by optimizing an error function.
 - We observed problems with **Least-Squares Classification** based on the squared error function.
 - In particular, sensitivity to outliers
 - Can we predict when such problems will occur?
- *Let's analyze the behavior of error functions in more detail...*



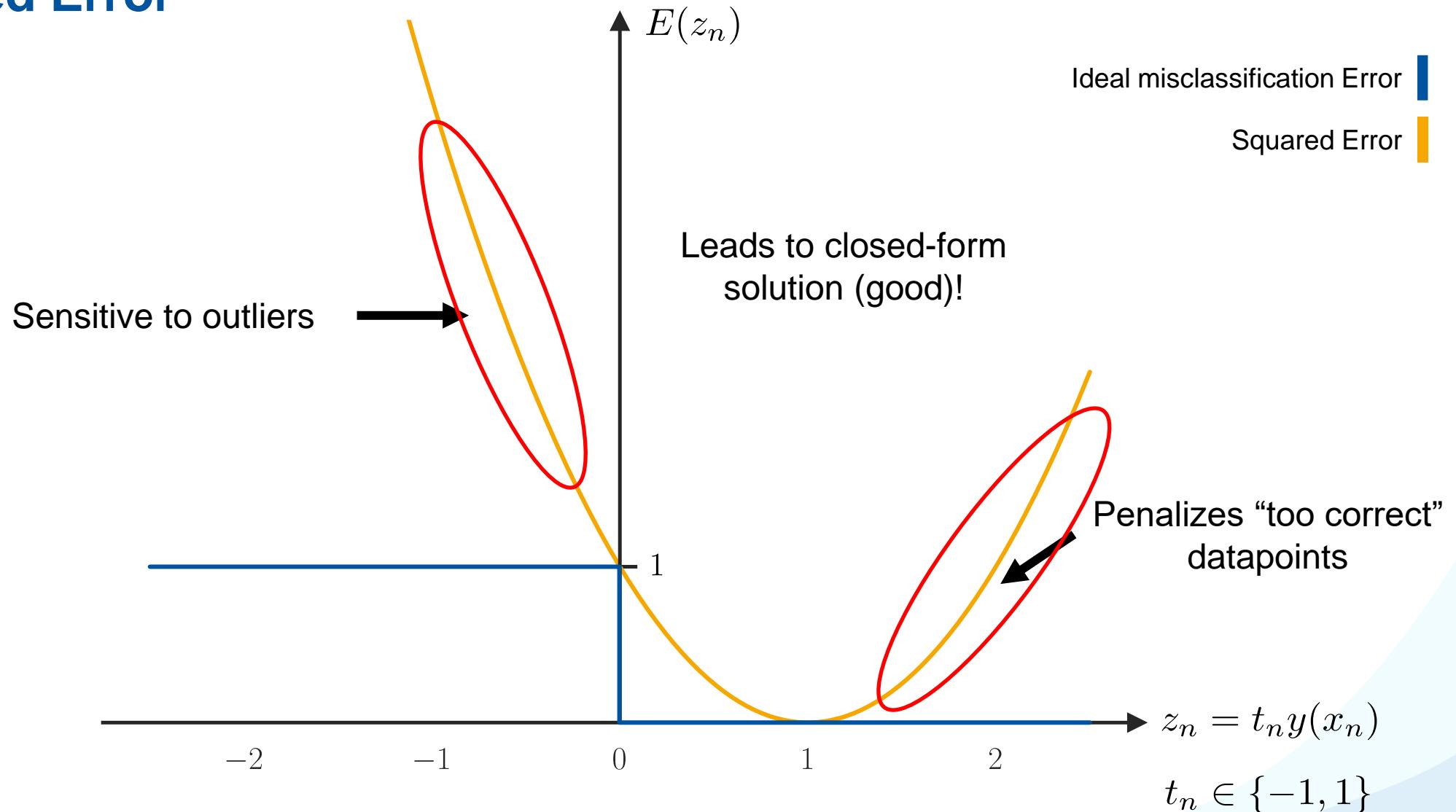
Error Contribution Plot



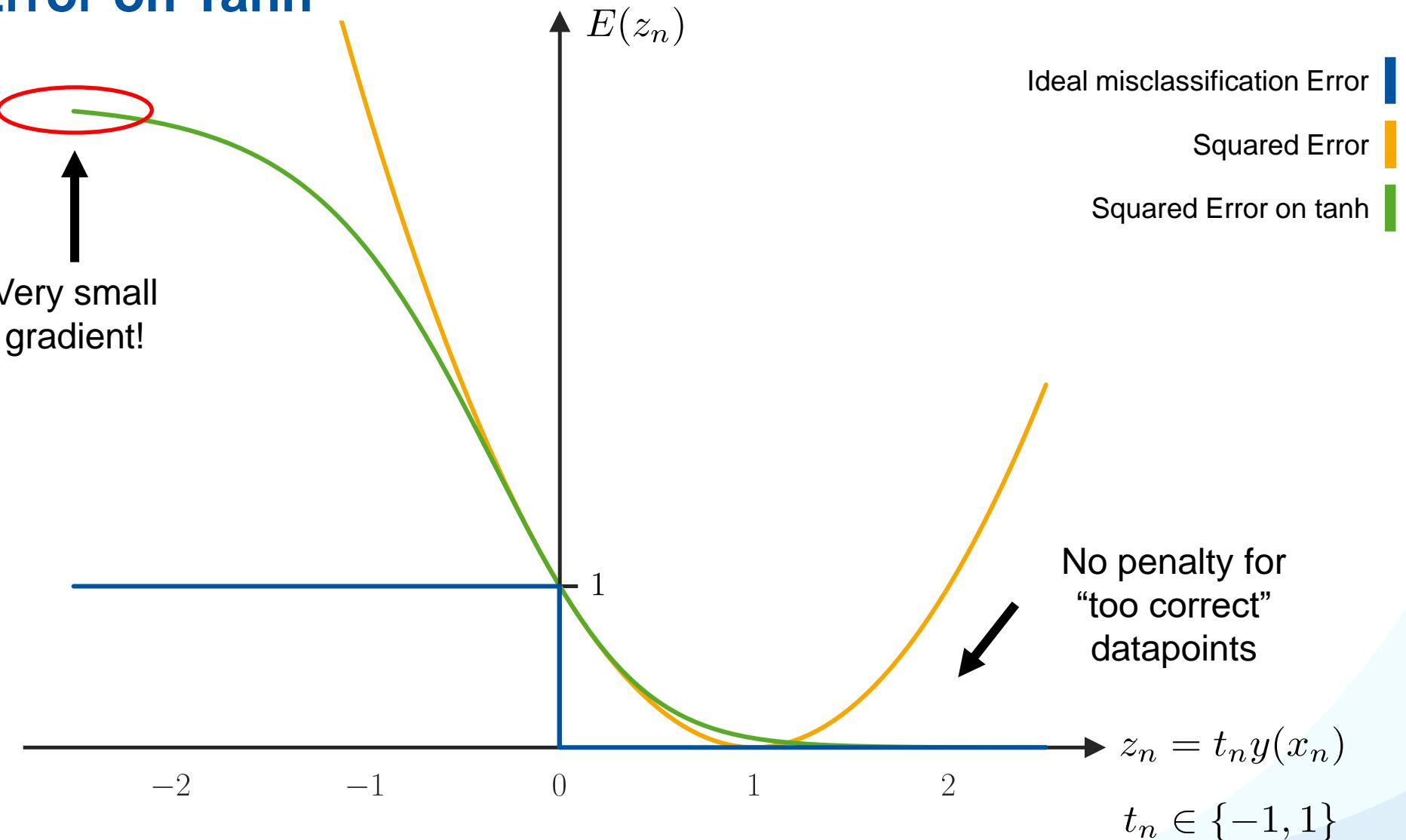
Ideal Misclassification Error



Squared Error

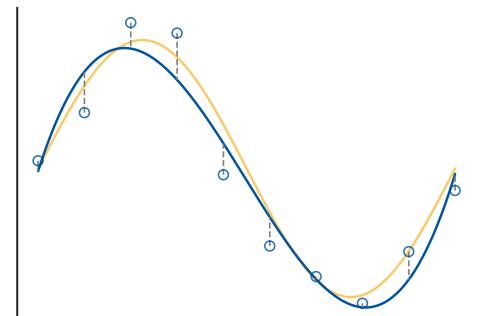


Squared Error on Tanh

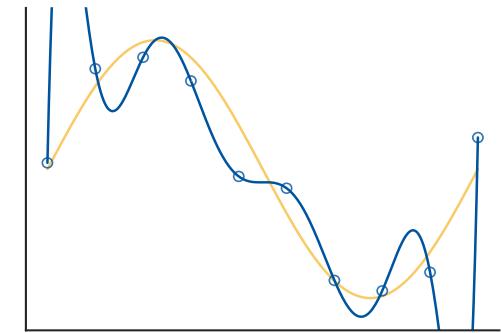


Machine Learning Topics

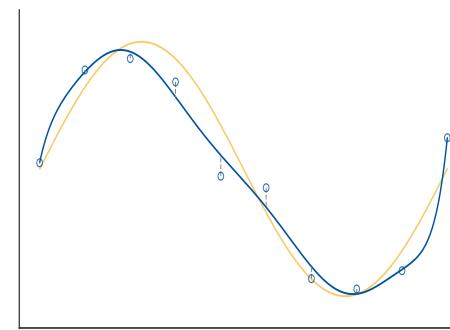
1. Introduction to ML
2. Probability Density Estimation
3. Linear Discriminants
- 4. Linear Regression**
5. Logistic Regression
6. Support Vector Machines
7. AdaBoost
8. Neural Network Basics



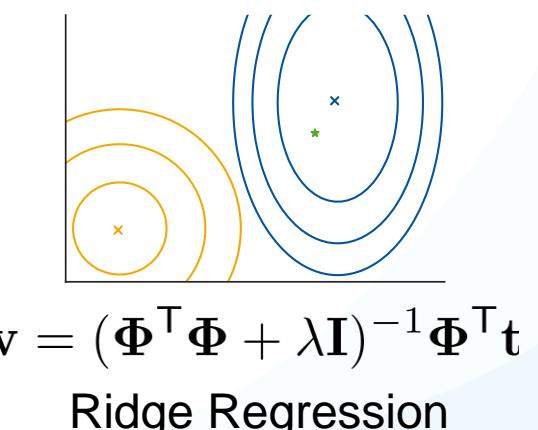
$y(\mathbf{x}) = \mathbf{w}^T \phi(\mathbf{x}) + w_0$
Linear Regression Functions



Overfitting



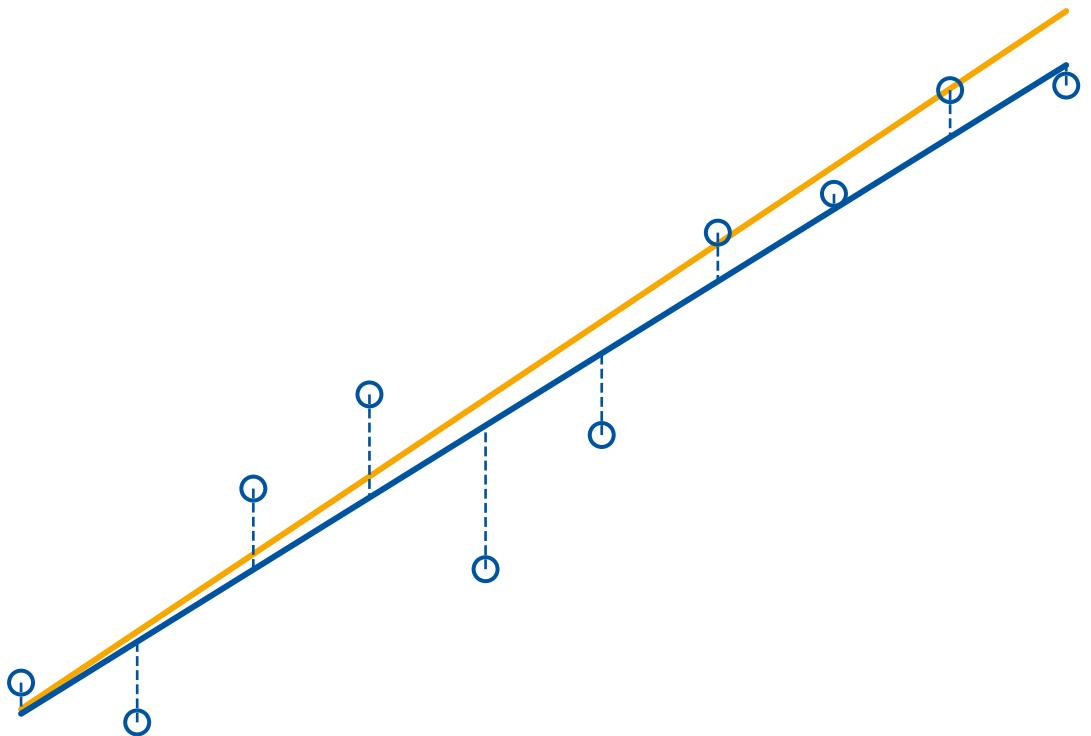
$E(\mathbf{w}) = L(\mathbf{w}) + \lambda \Omega(\mathbf{w})$
Regularization



$\mathbf{w} = (\Phi^T \Phi + \lambda \mathbf{I})^{-1} \Phi^T \mathbf{t}$
Ridge Regression

Linear Regression

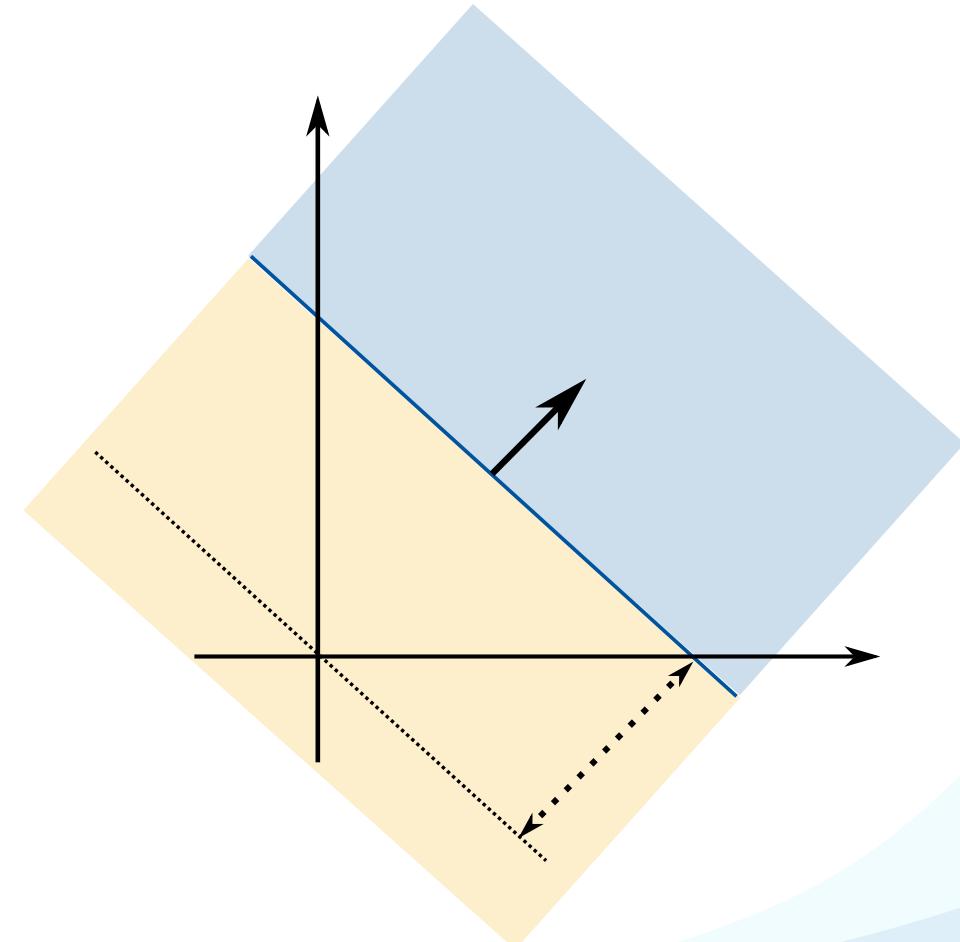
1. Motivation
2. Least-Squares Regression
3. Regularization
4. Ridge Regression
5. The Bias-Variance Tradeoff



Motivation: Linear Regression

- We have seen how to build classifiers with linear functions:

$$y(\mathbf{x}) = \mathbf{w}^T \phi(\mathbf{x}) + w_0$$

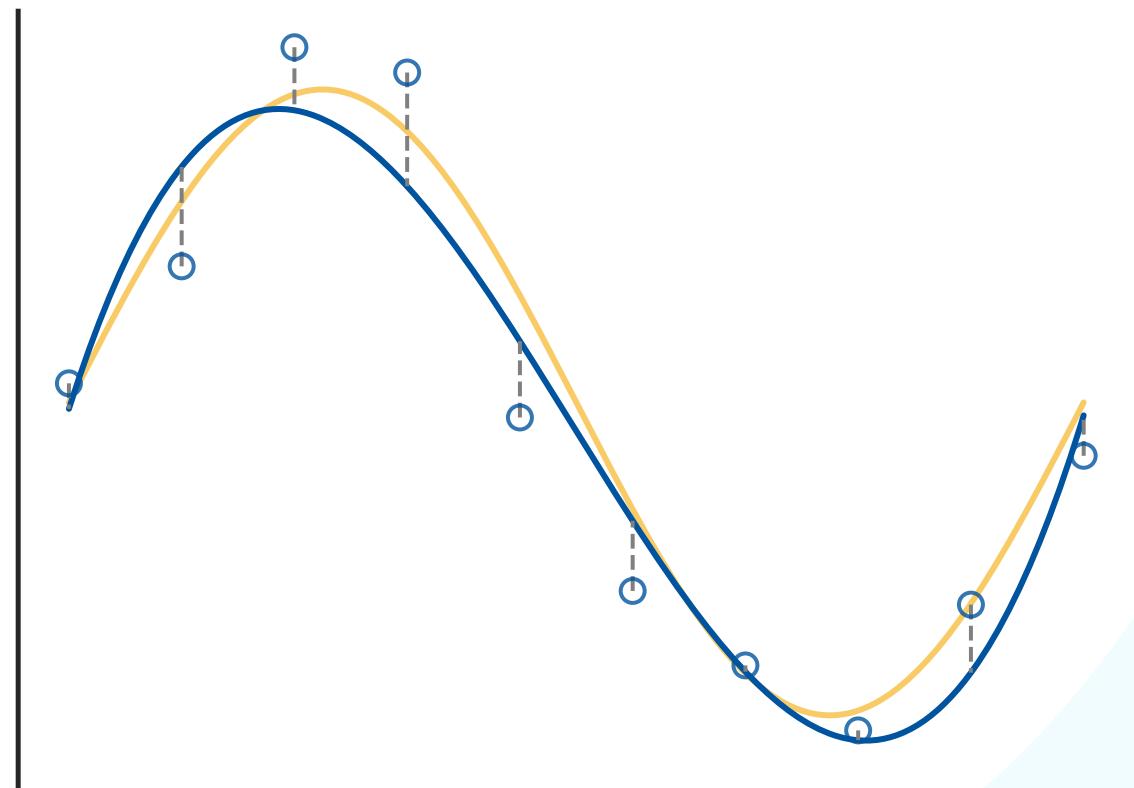


Motivation: Linear Regression

- We have seen how to build classifiers with linear functions:

$$y(\mathbf{x}) = \mathbf{w}^T \phi(\mathbf{x}) + w_0$$

- Now, we will use this model to estimate arbitrary functions using real-valued labels $t_n \in \mathbb{R}$.
- Key assumption: data is generated by some function $h(\mathbf{x})$ with Gaussian noise:
$$t_n = h(\mathbf{x}) + \epsilon$$
- This model is called [linear regression](#).



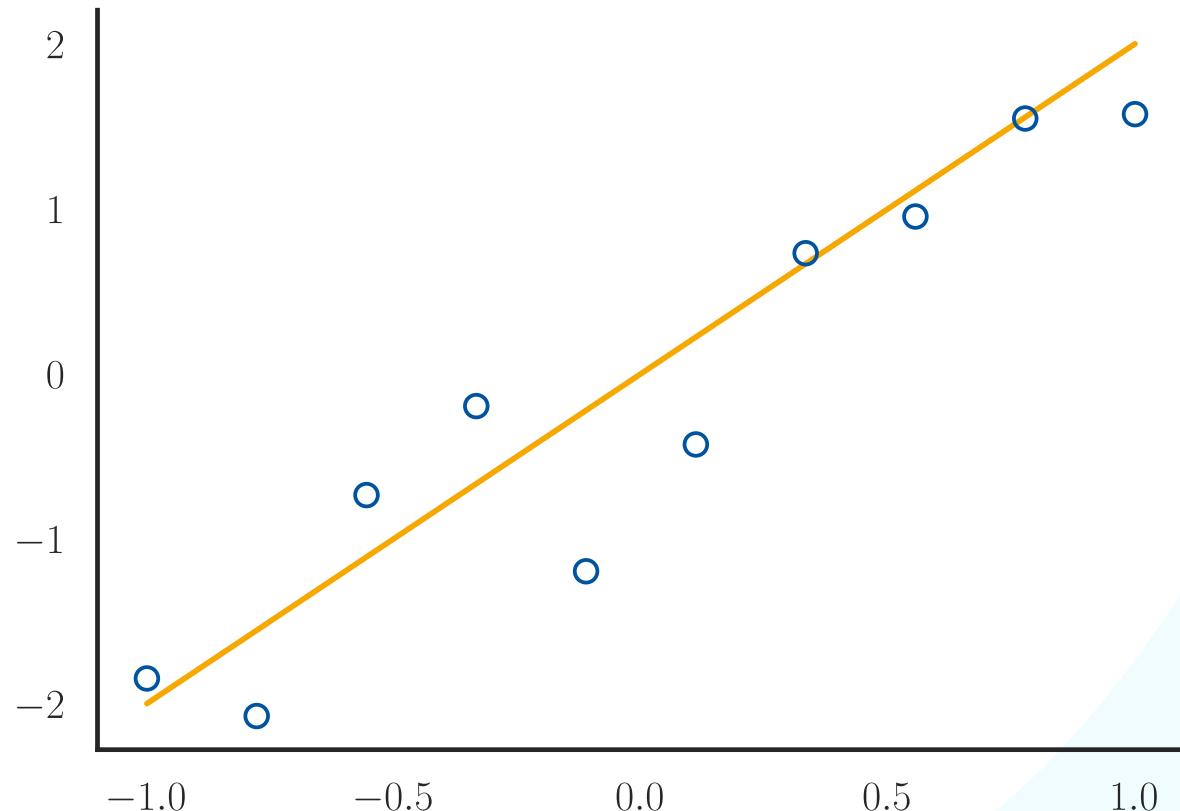
Example: Linear target functions

$$y(x) = w_1x + w_0$$

- We assume ground truth is a linear function:

$$f(x) = mx + b$$

- We try to find a line that minimizes the distance to the samples.



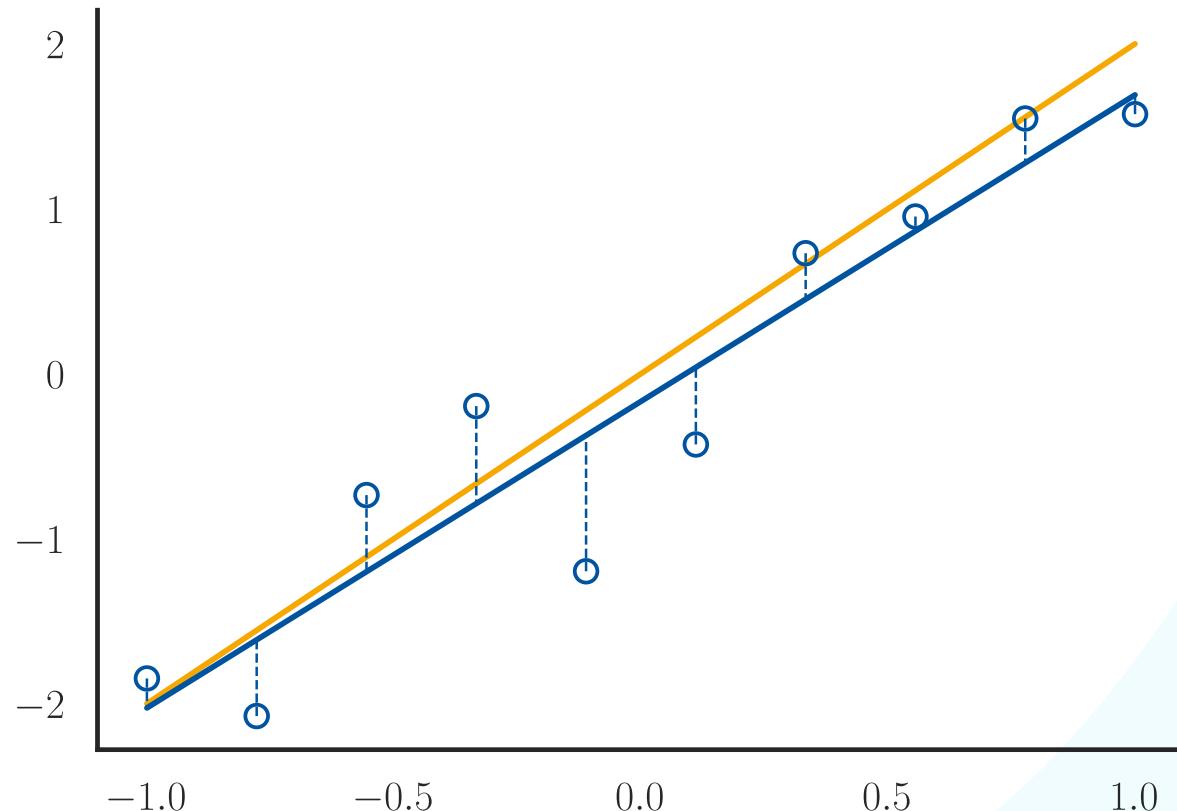
Example: Linear target functions

$$y(x) = w_1x + w_0$$

- We assume ground truth is a linear function:

$$f(x) = mx + b$$

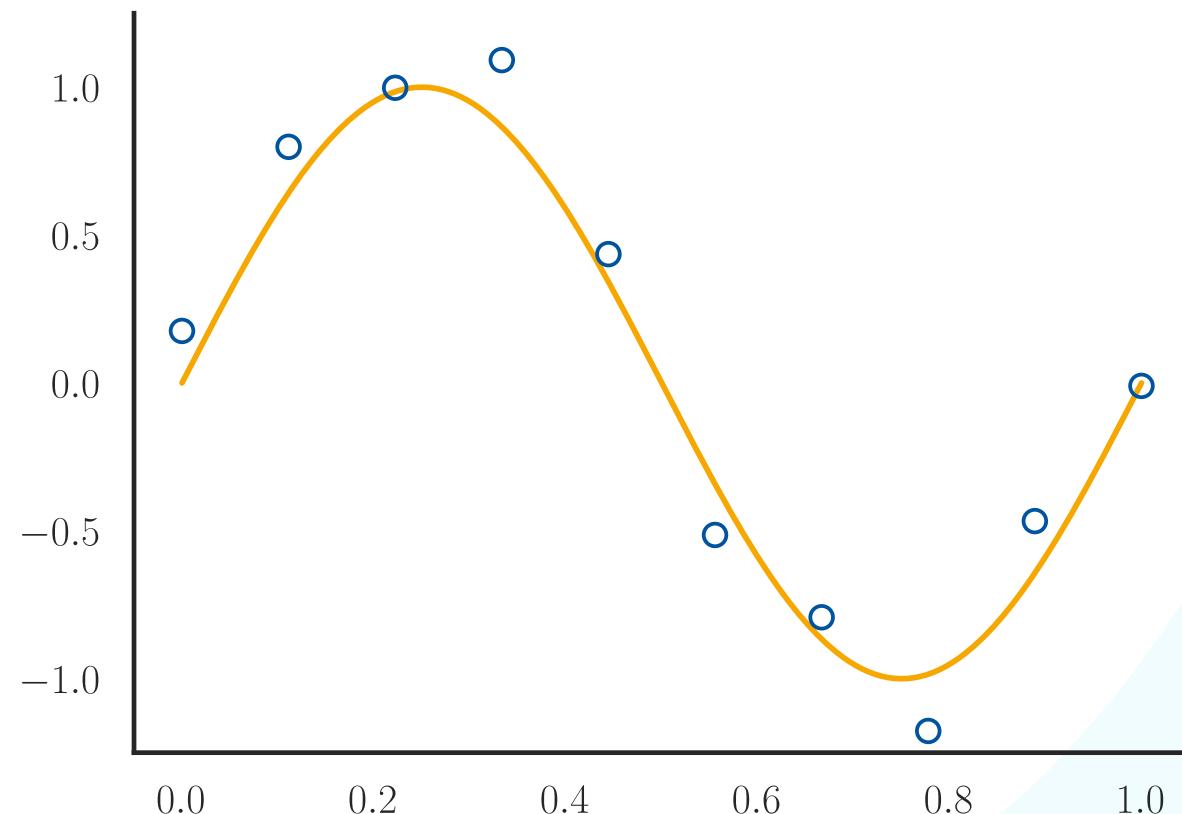
- We try to find a line that minimizes the distance to the samples.



Example: Non-linear target functions

$$y(x) = \mathbf{w}^T \phi(x) + w_0$$

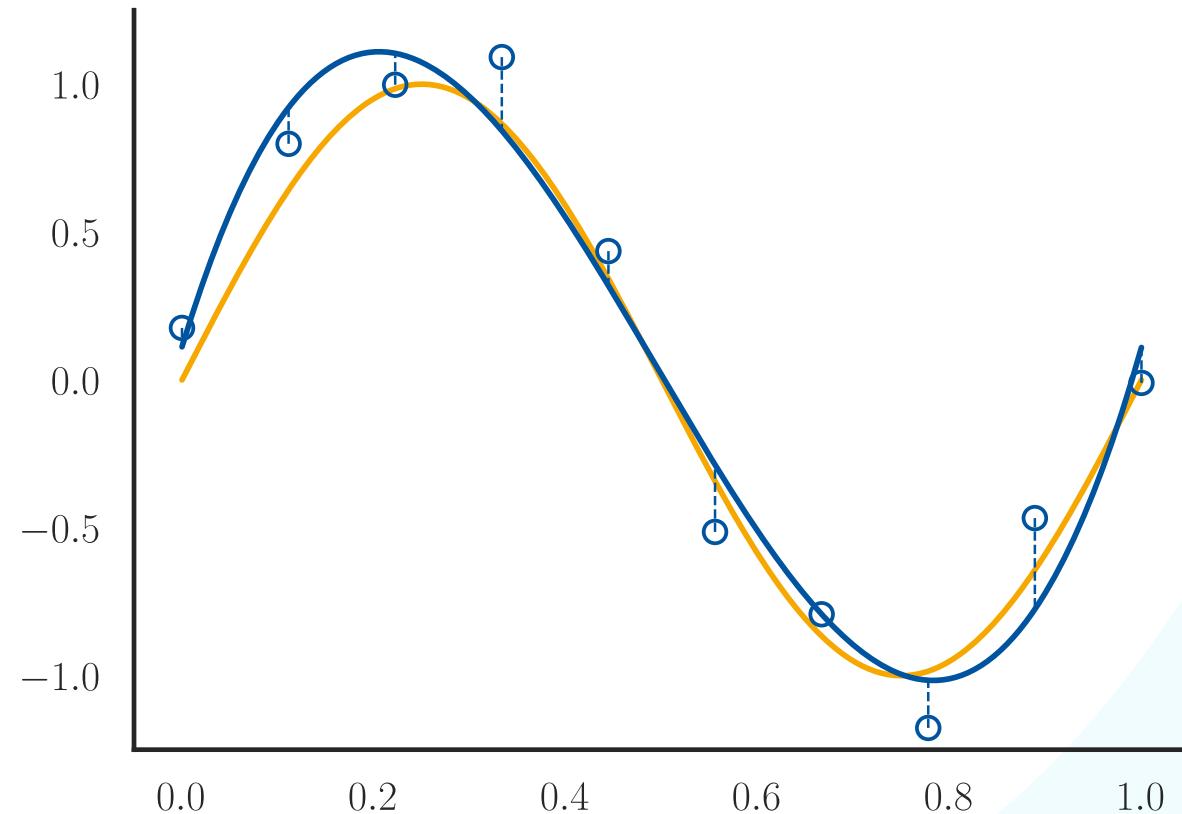
- We can use basis functions to fit arbitrary functions $f(\mathbf{x})$.
- For example, polynomial basis functions.



Example: Non-linear target functions

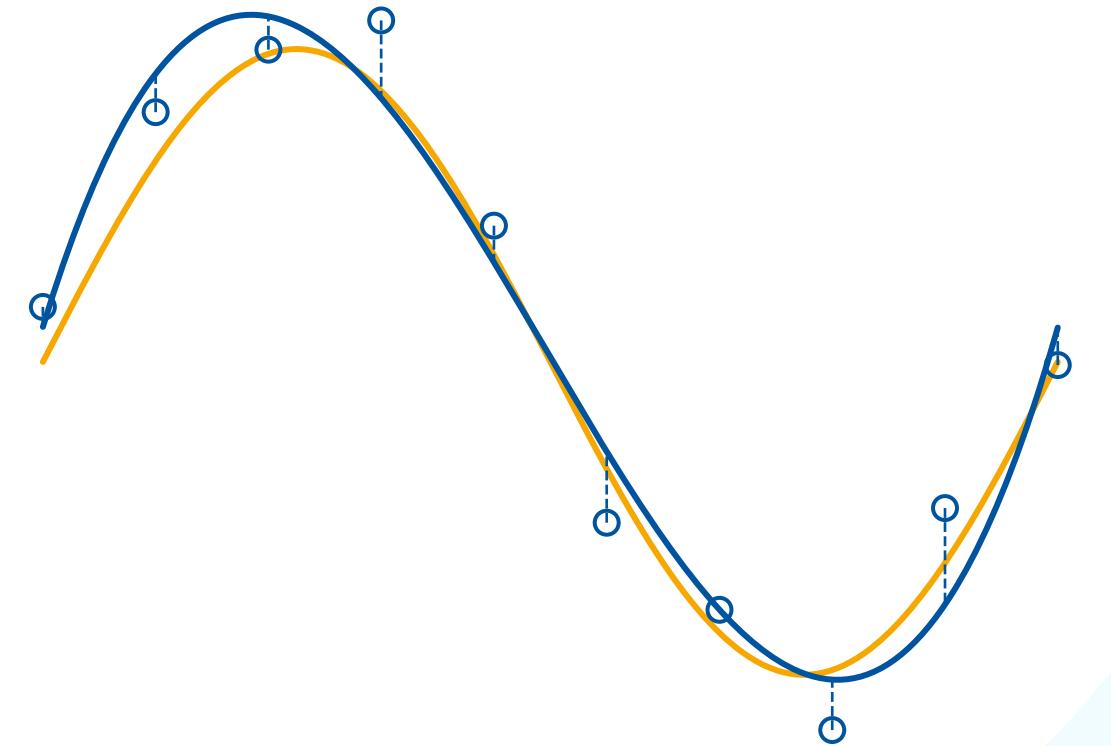
$$y(x) = \mathbf{w}^T \phi(x) + w_0$$

- We can use basis functions to fit arbitrary functions $f(\mathbf{x})$.
- For example, polynomial basis functions.
- Finding a good set of basis functions usually requires some insight into the data...



Linear Regression

1. Motivation
2. **Least-Squares Regression**
3. Regularization
4. Ridge Regression
5. The Bias-Variance Tradeoff



Least-Squares Regression

- We want to optimize the difference between our predictor $y(\mathbf{x}_n; \mathbf{w})$ and the targets t_n .
- The only difference is that our targets t_n are now continuous values.
- Again, use the familiar **squared error** objective:

$$E(\mathbf{w}) = \frac{1}{2} \sum_{n=1}^N (y(\mathbf{x}_n; \mathbf{w}) - t_n)^2$$

- This has the same solution as for classification (normal equations).

$$y(\mathbf{x}_n; \mathbf{w}) = \mathbf{w}^\top \phi(\mathbf{x}_n)$$

$$\begin{aligned}\frac{\partial E(\mathbf{w})}{\partial \mathbf{w}} &= \sum_{n=1}^N (\mathbf{w}^\top \phi_n - t_n) \phi_n \\ &= \Phi^\top (\Phi \mathbf{w} - \mathbf{t}) \stackrel{!}{=} 0\end{aligned}$$

$$\Rightarrow \mathbf{w} = (\Phi^\top \Phi)^{-1} \Phi^\top \mathbf{t}$$

Example: Fitting a polynomial

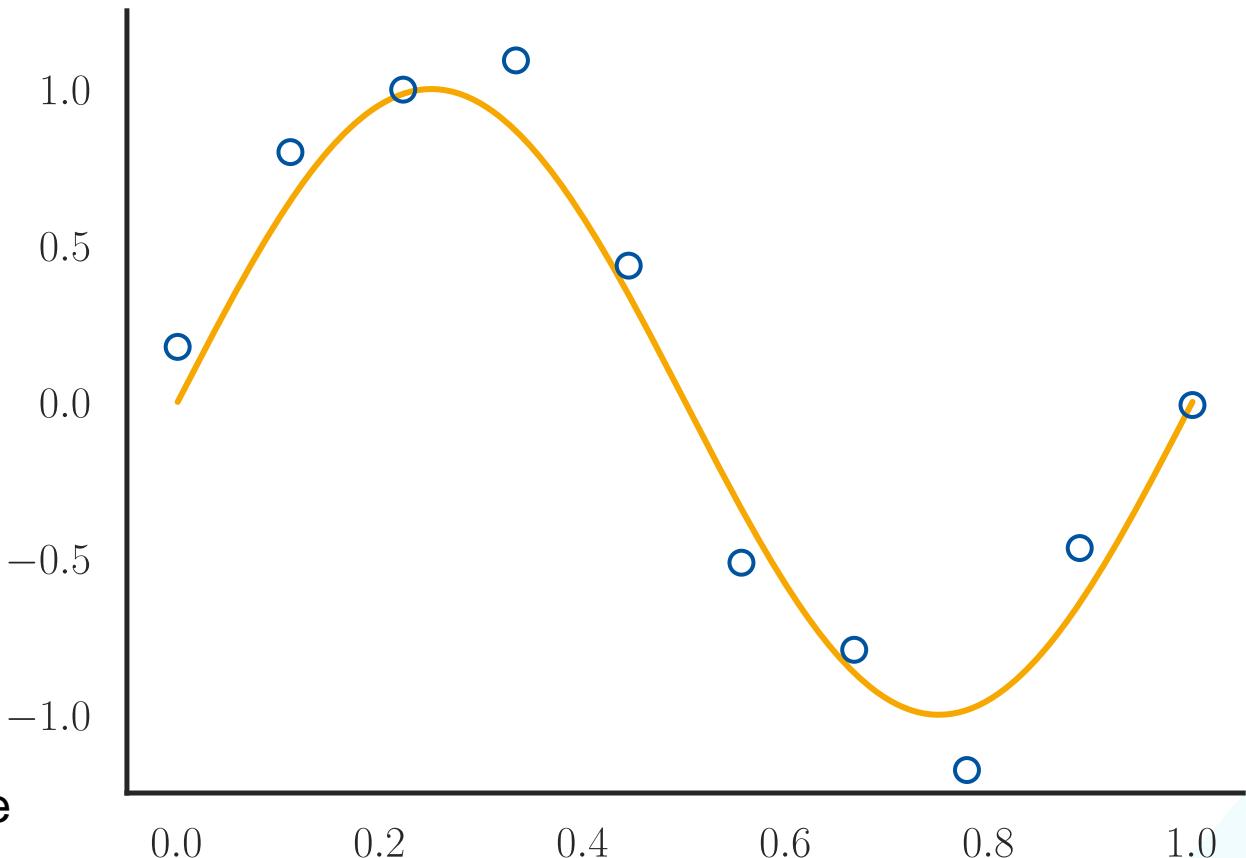
- This is clearly not a linear function.
- Let's use polynomial basis functions:

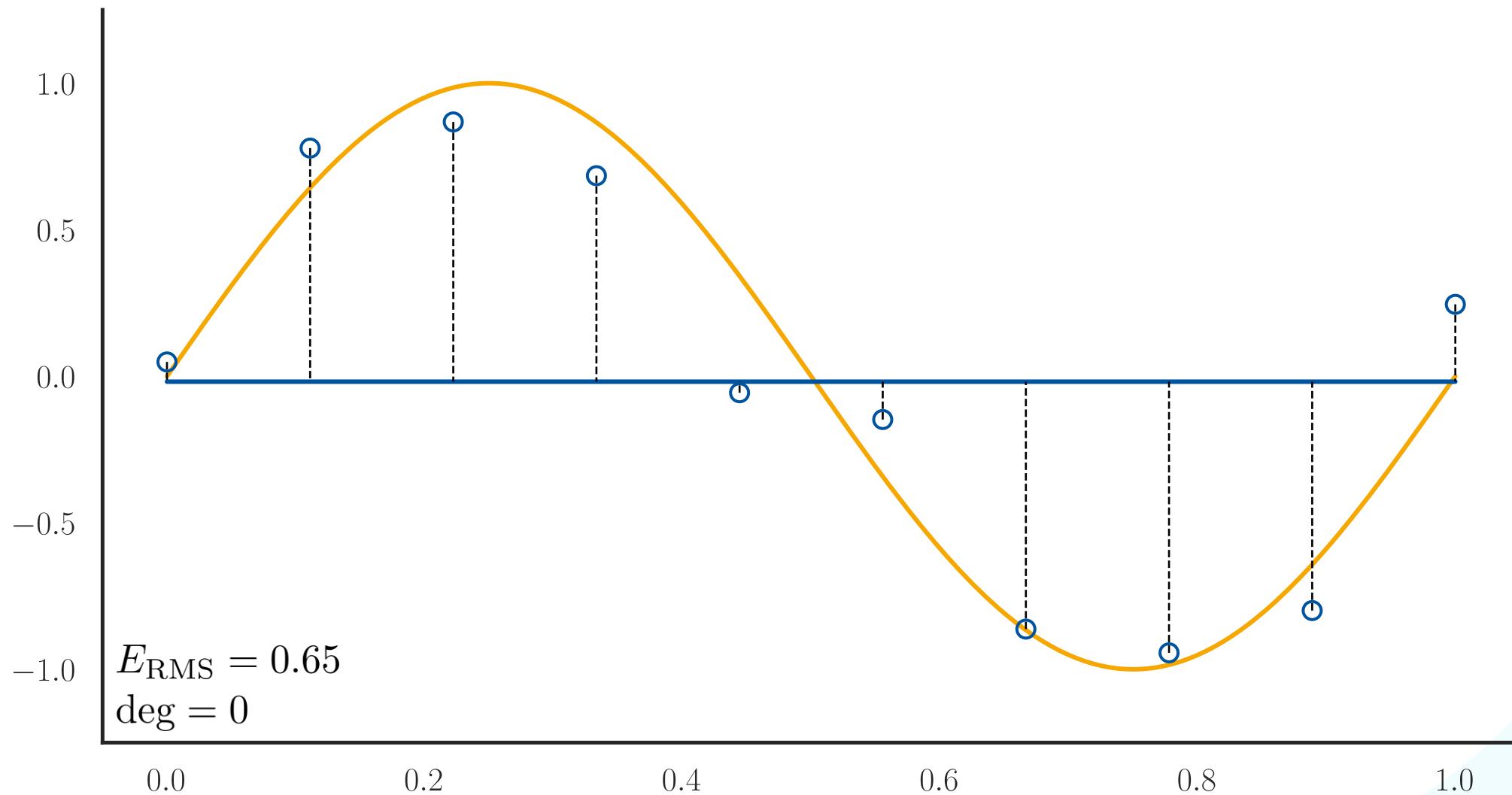
$$\phi_j(x) = (x^j)$$

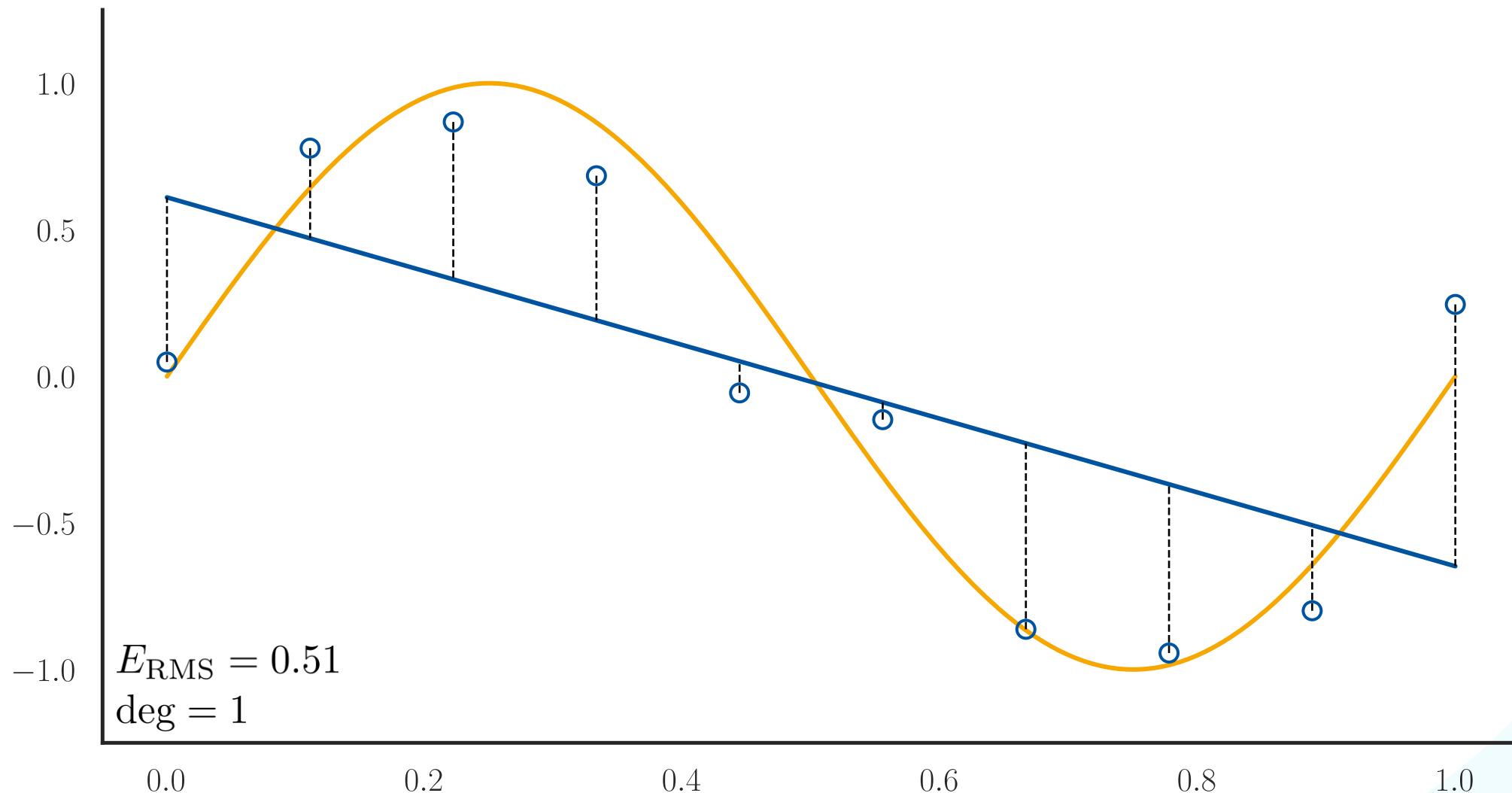
- Which degree should we use?
- Compare different models by their Root Mean Square Error:

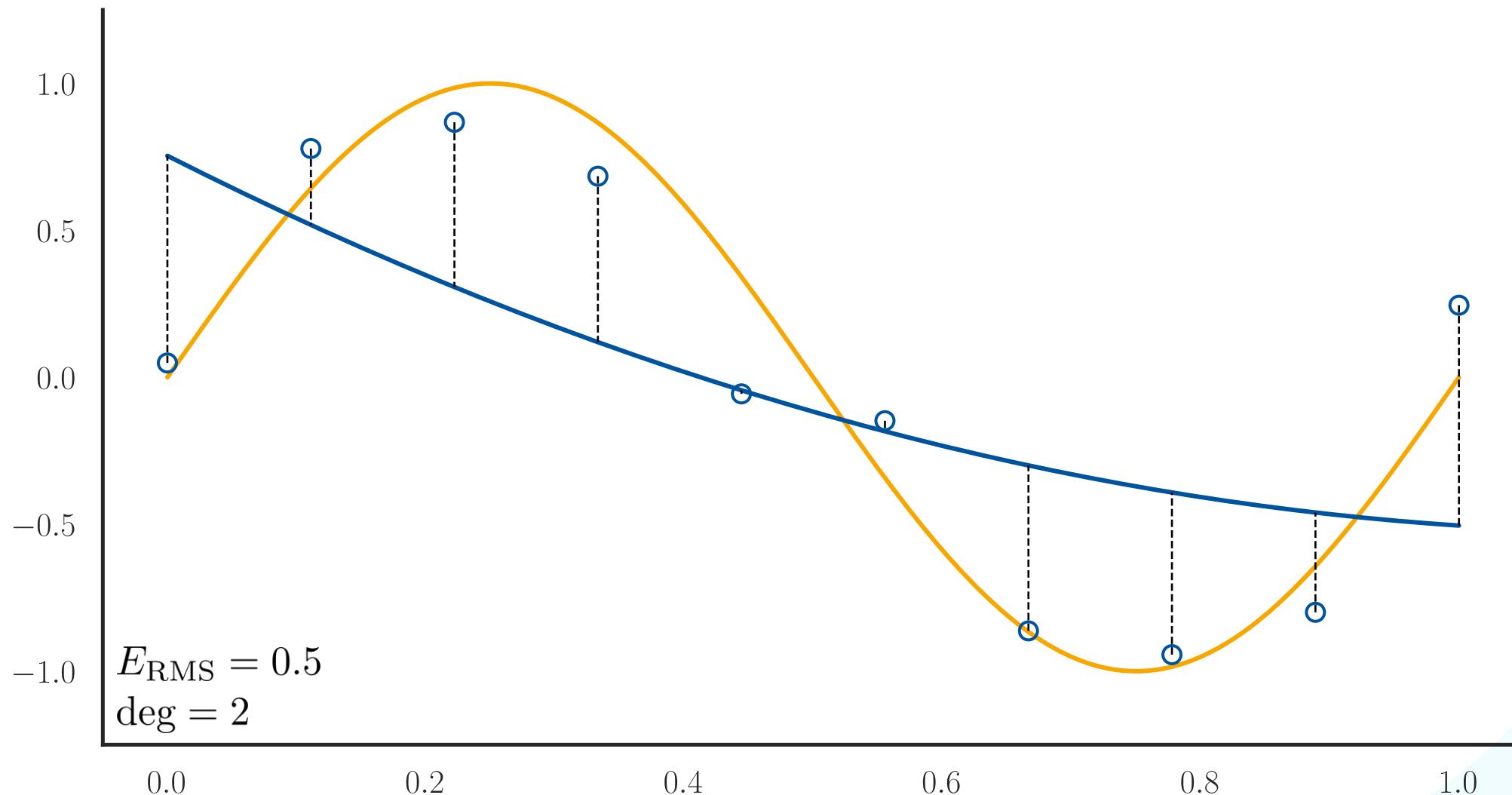
$$E_{\text{RMS}} = \sqrt{\frac{2E(\mathbf{w}^*)}{N}}$$

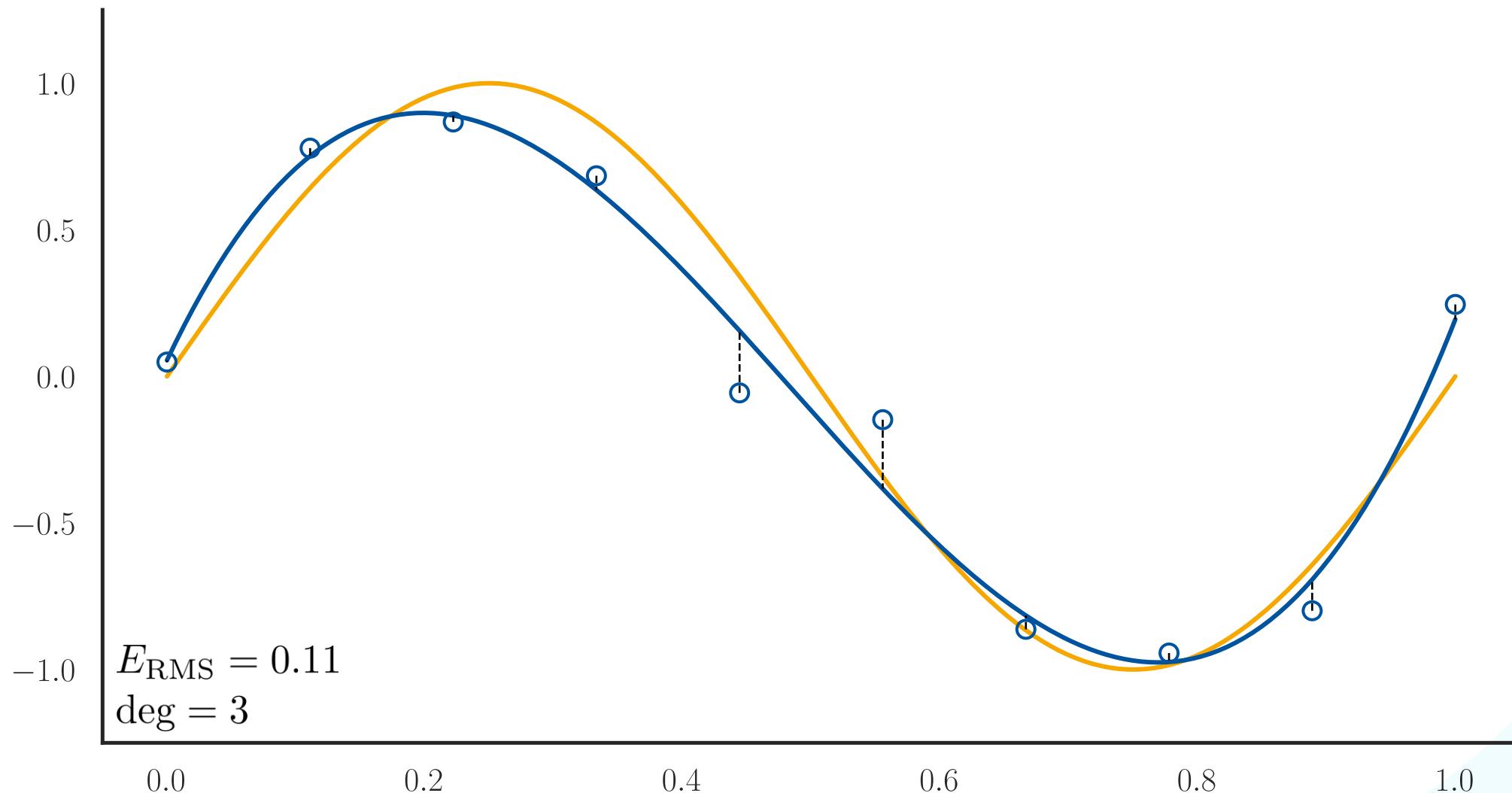
- RMS is independent of training set size and has the same scale as the data.

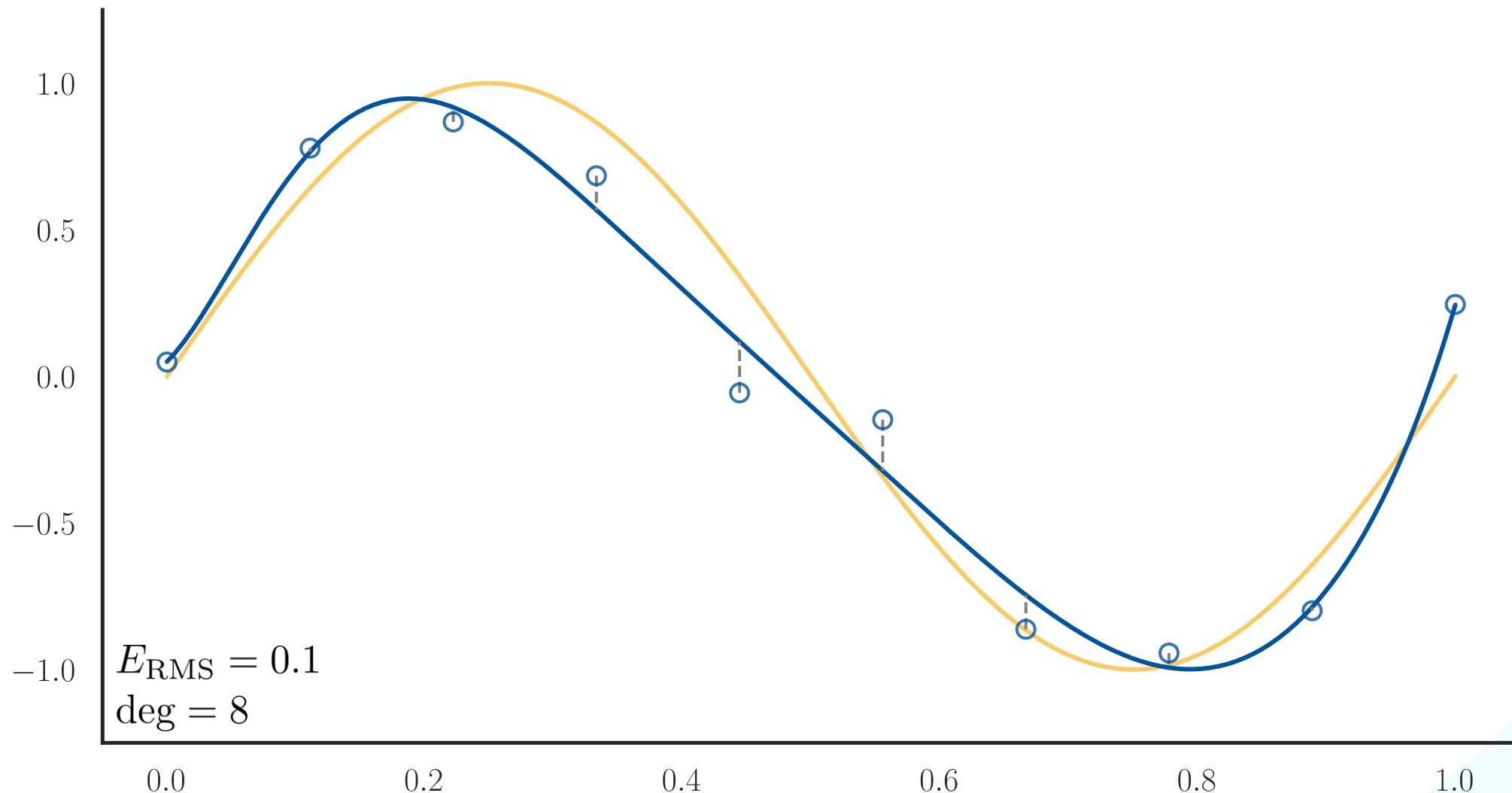


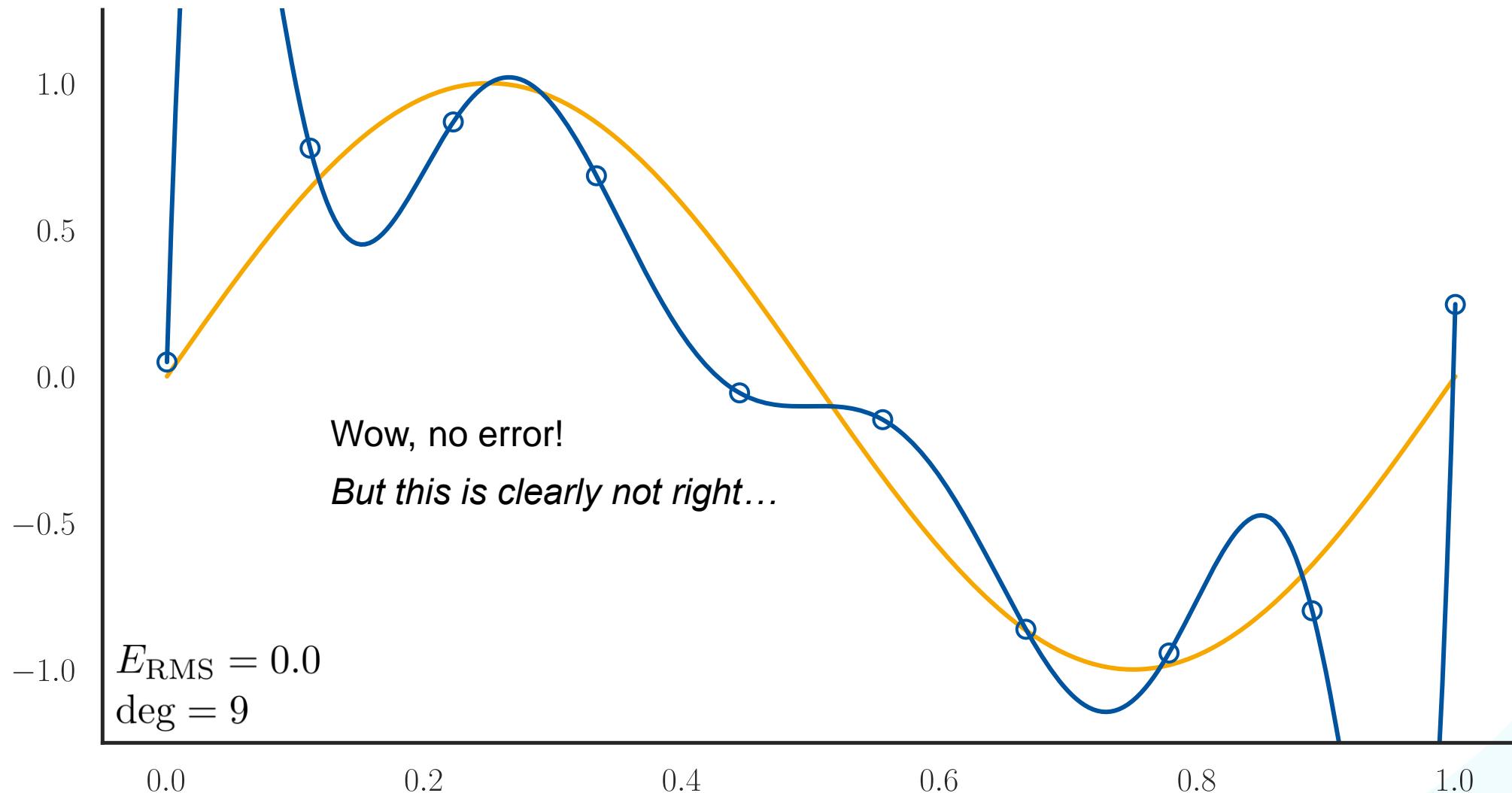


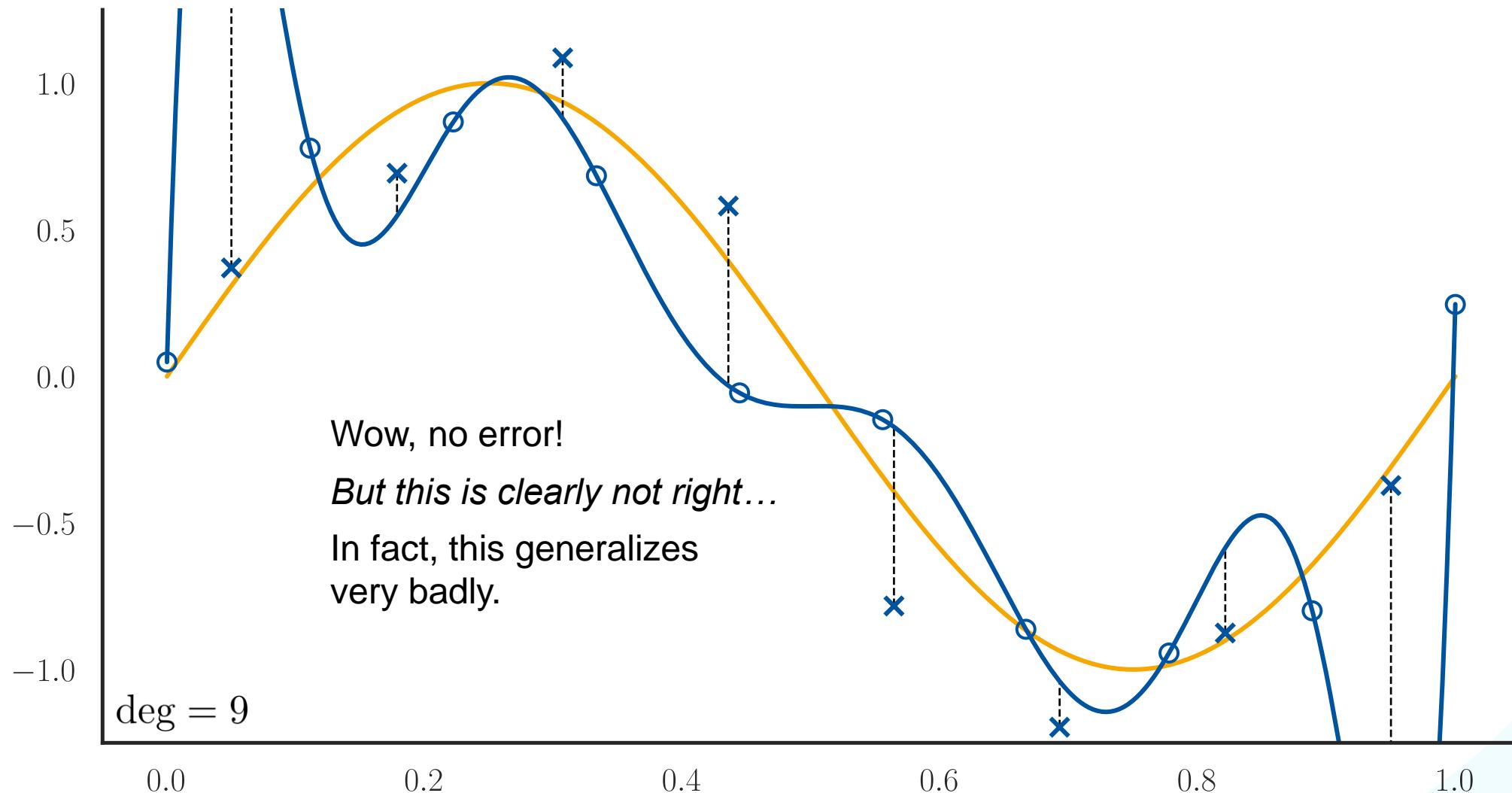






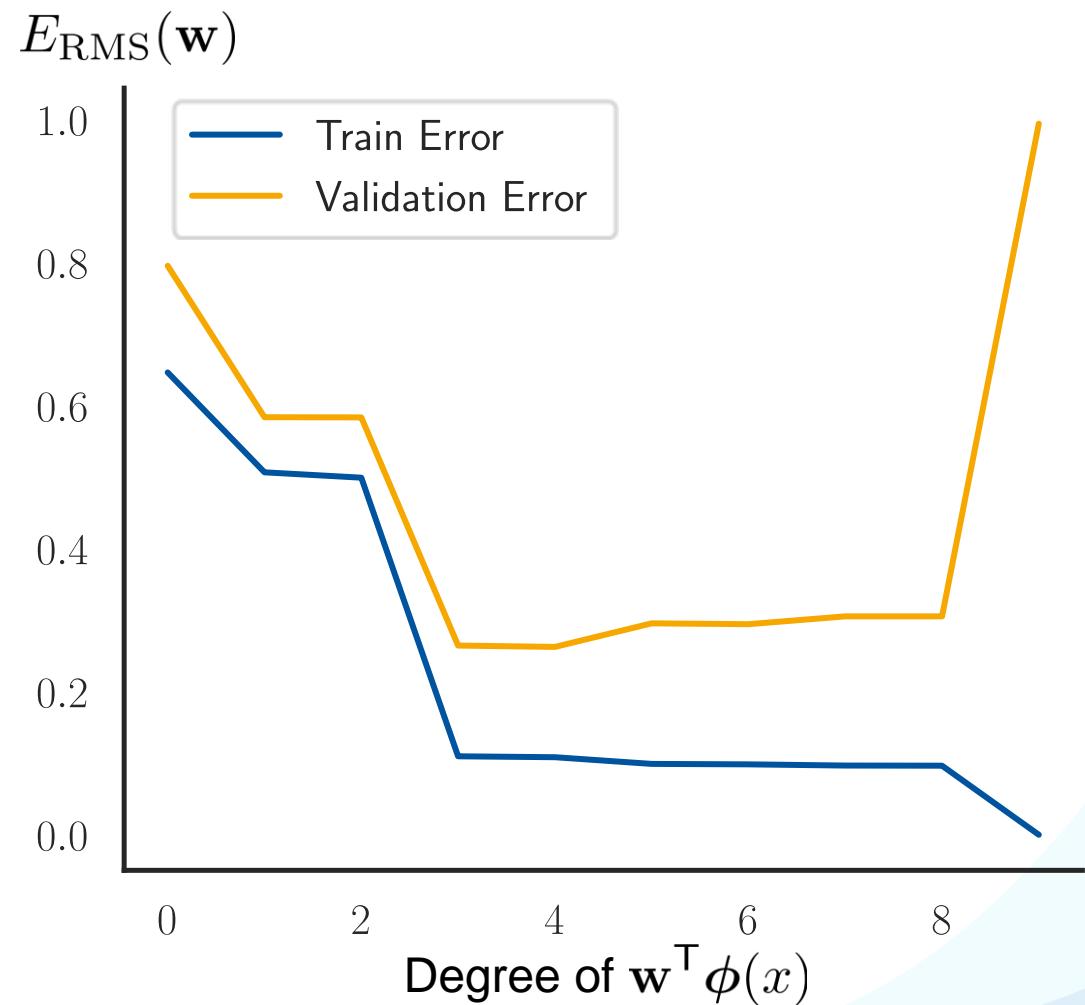


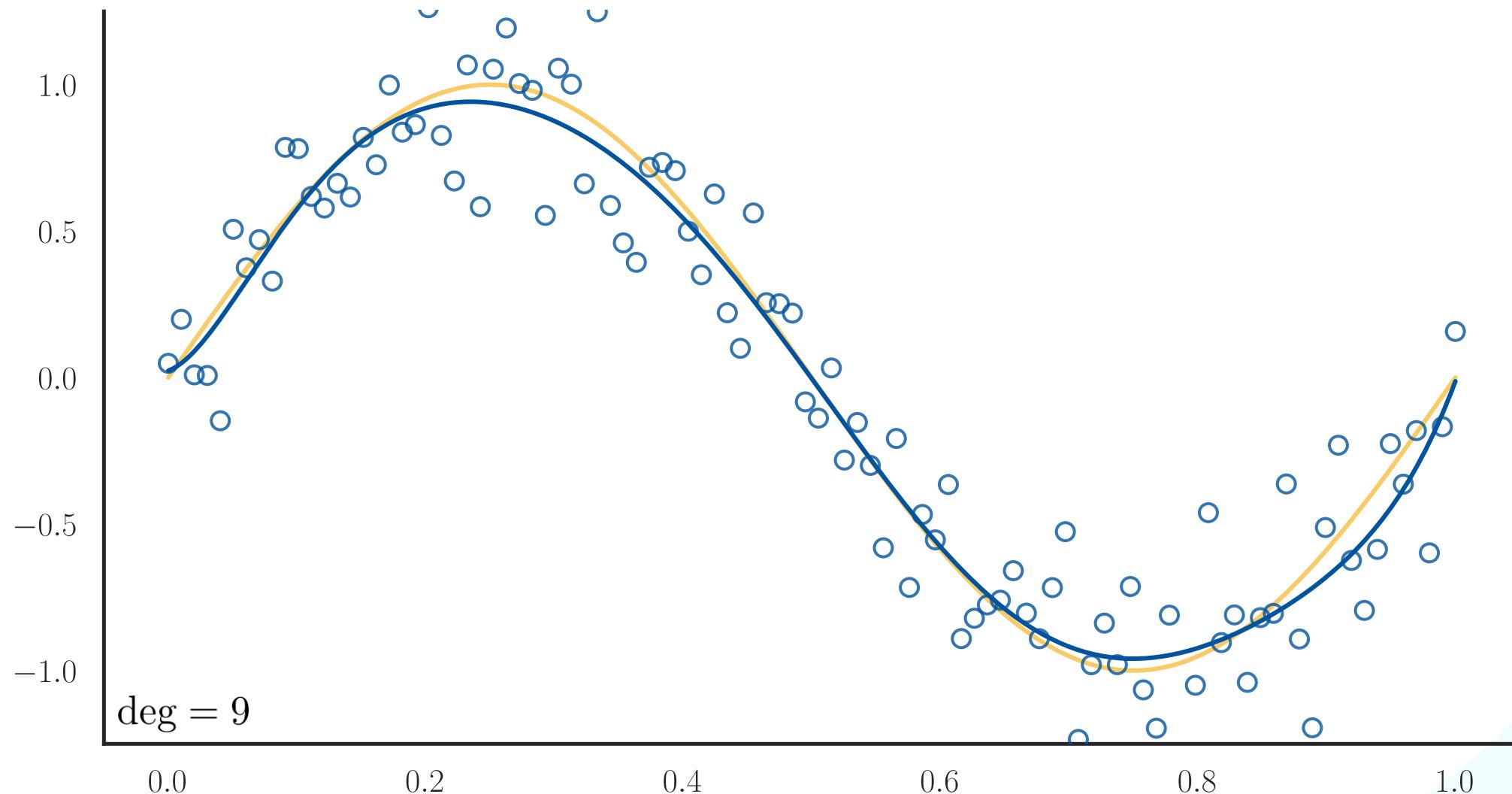




Overfitting

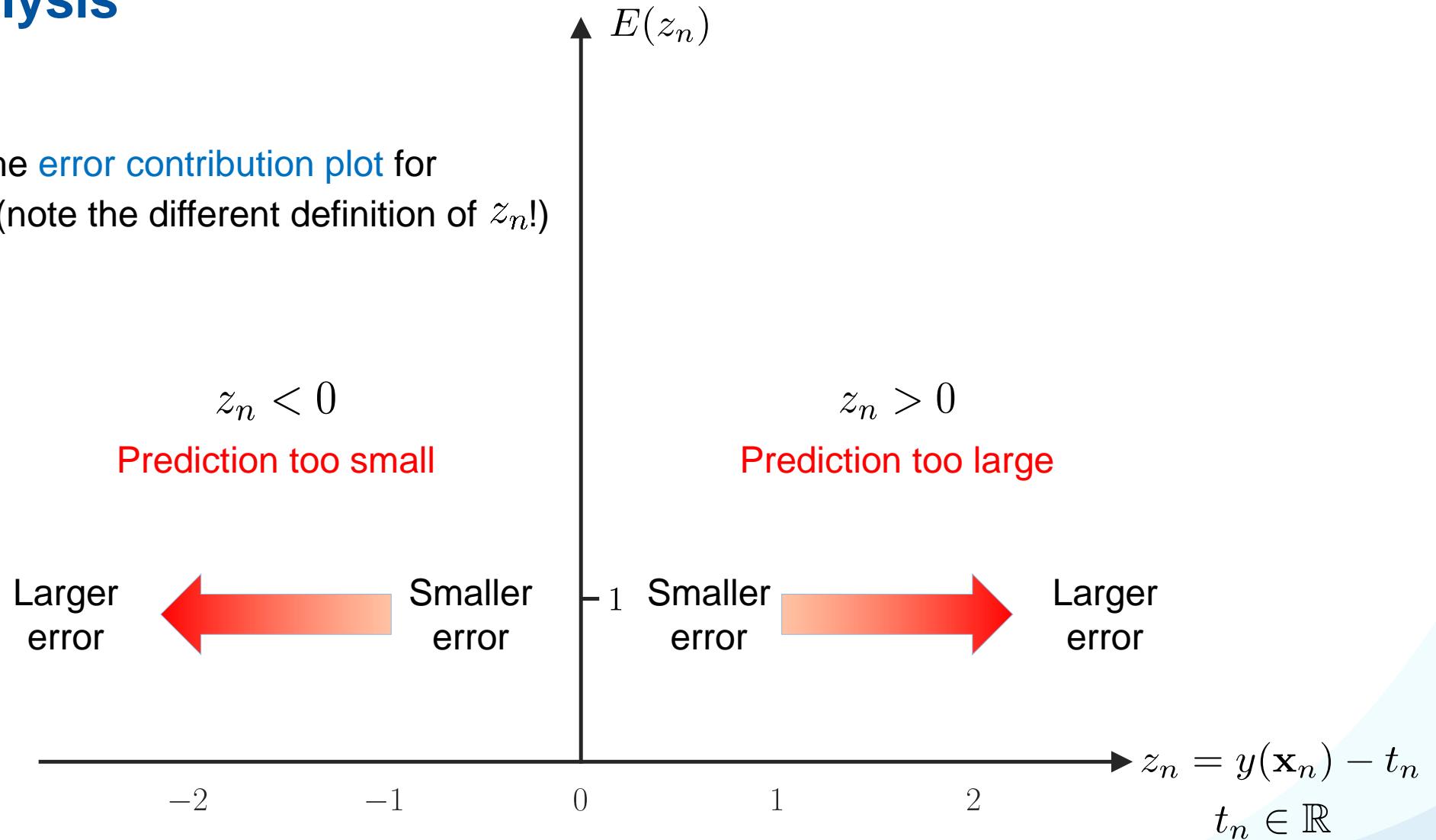
- We fit the dataset perfectly, but the resulting function is clearly not what we want.
- This phenomenon is called **overfitting**.
- Remember: we assume $t_n = h(\mathbf{x}_n) + \epsilon$.
- Our model is “too” powerful and models the noise instead of the underlying function!
- *What can we do to avoid overfitting?*
 - One solution: More data!



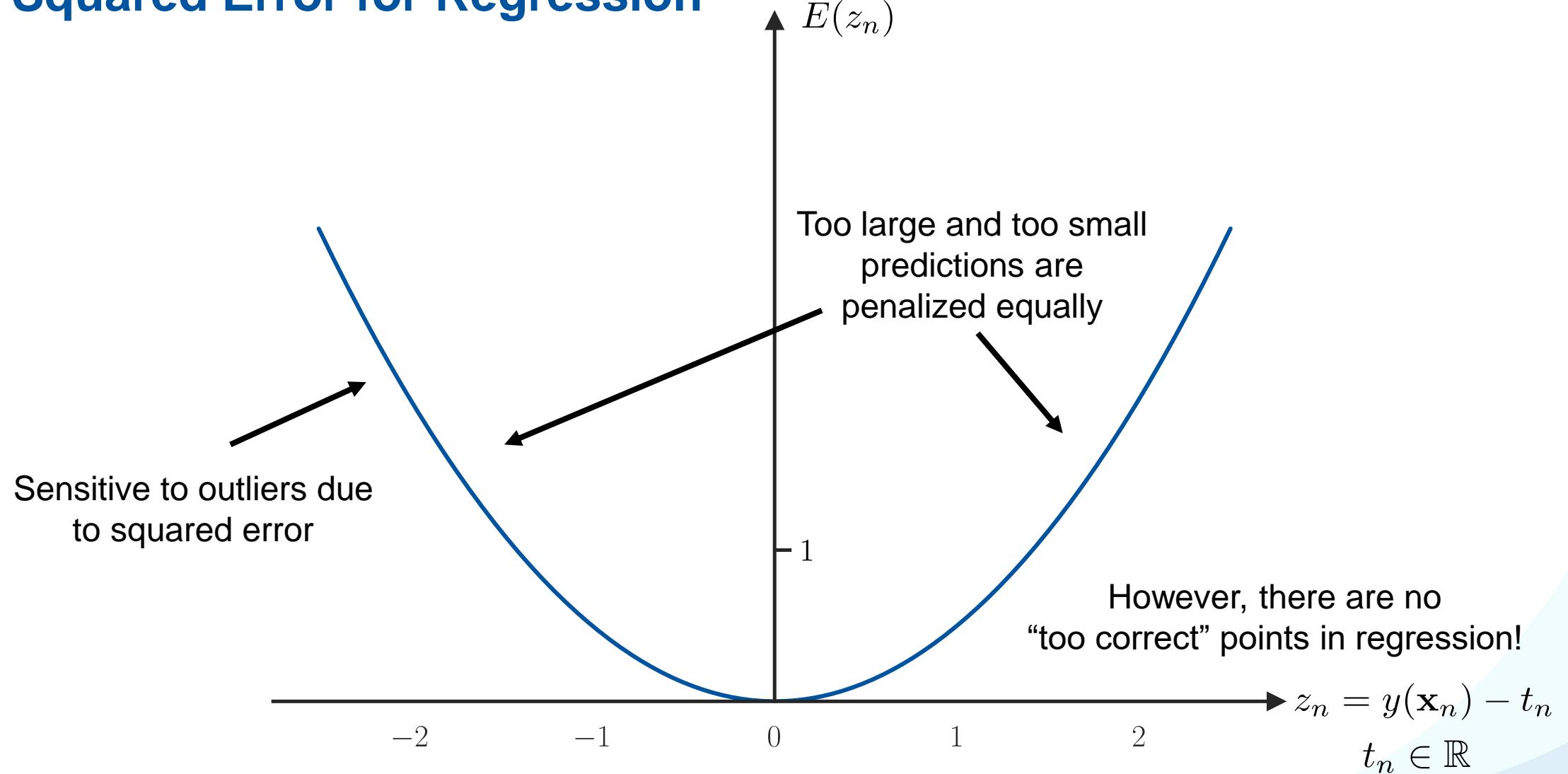


Error Analysis

- Variant of the error contribution plot for regression (note the different definition of z_n !)



Squared Error for Regression



Discussion: Least-Squares Regression

Advantages

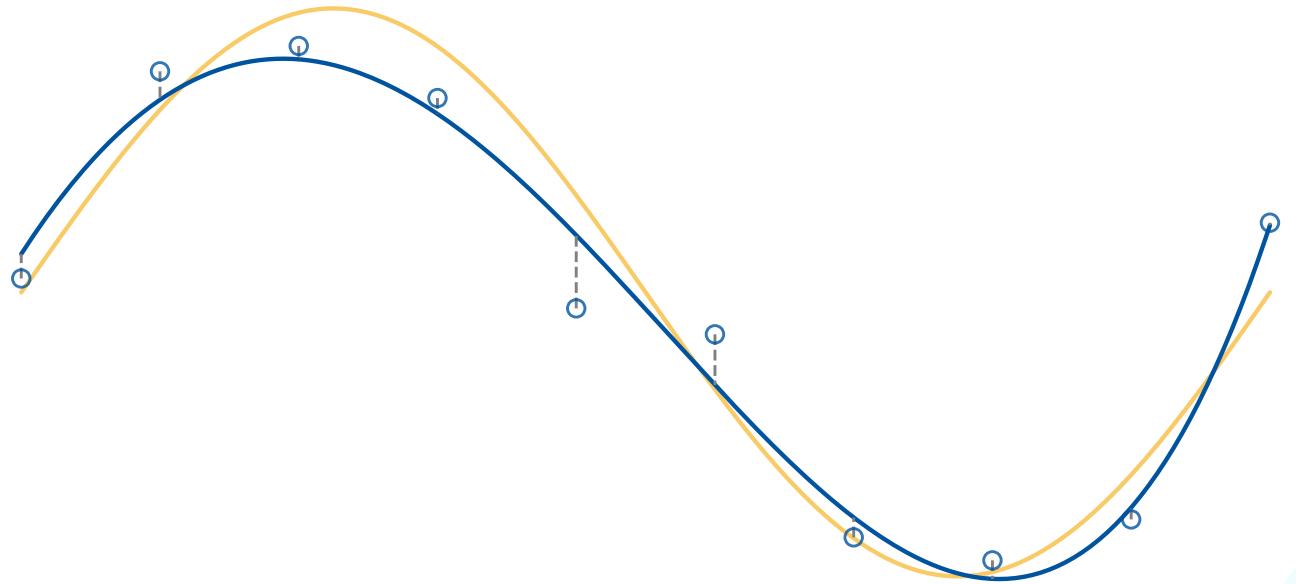
- Squared error leads to closed-form solution of the regression problem.
- We can use basis functions to fit non-linear functions while staying within the framework of linear regression.
- Polynomial basis functions with different degrees of the polynomial result in regression functions with different capacities to approximate the target function.
- We can compare their results using the RMS error.

Limitations

- The squared error for regression is not robust to outliers, but it does not exhibit the systematic problems of least-squares classification.
- Overfitting when the degree of the polynomial becomes too large.
- Overfitting is a function of the available amount of data (and is more likely to occur with small training sets)

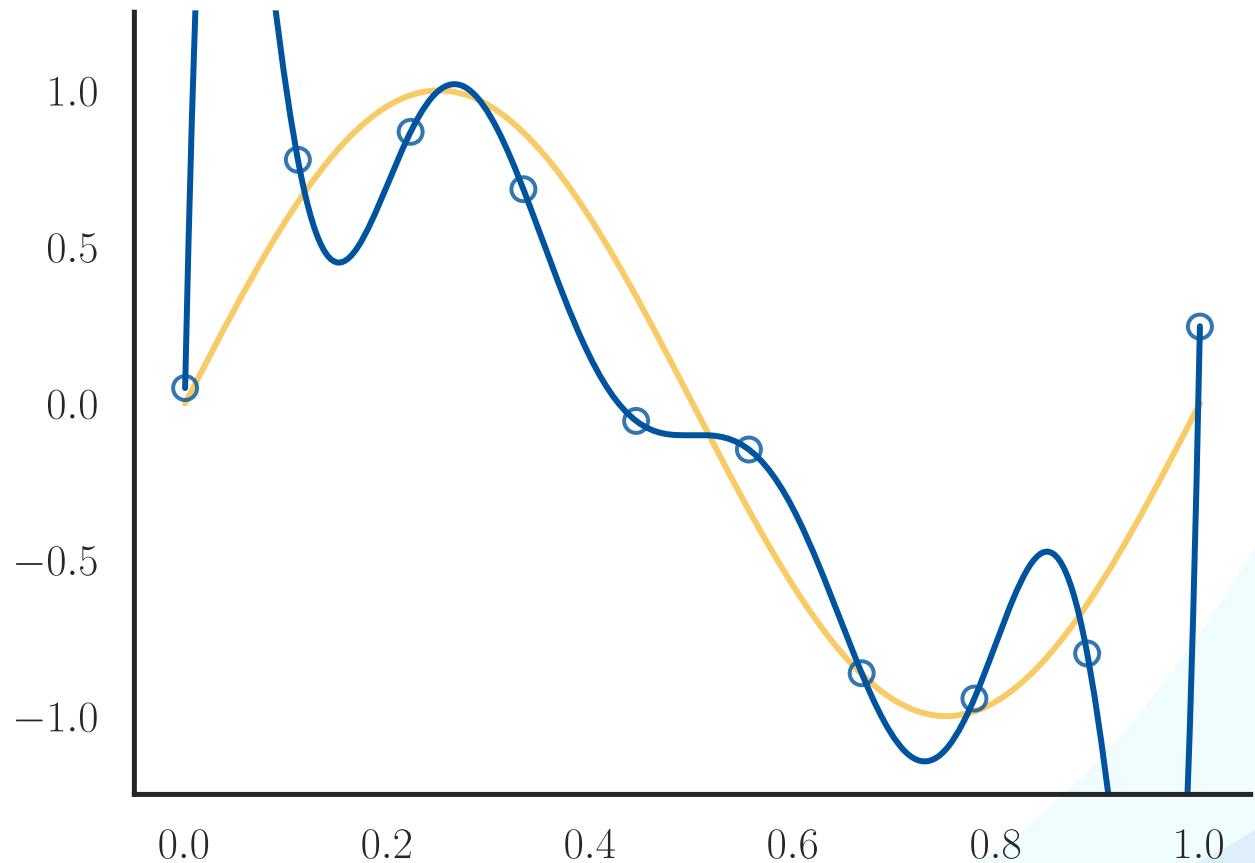
Linear Regression

1. Linear Regression
2. Least-Squares Regression
- 3. Regularization**
4. Ridge Regression
5. The Bias-Variance Tradeoff



Regularization

- With enough parameters, our model will overfit to the training set.



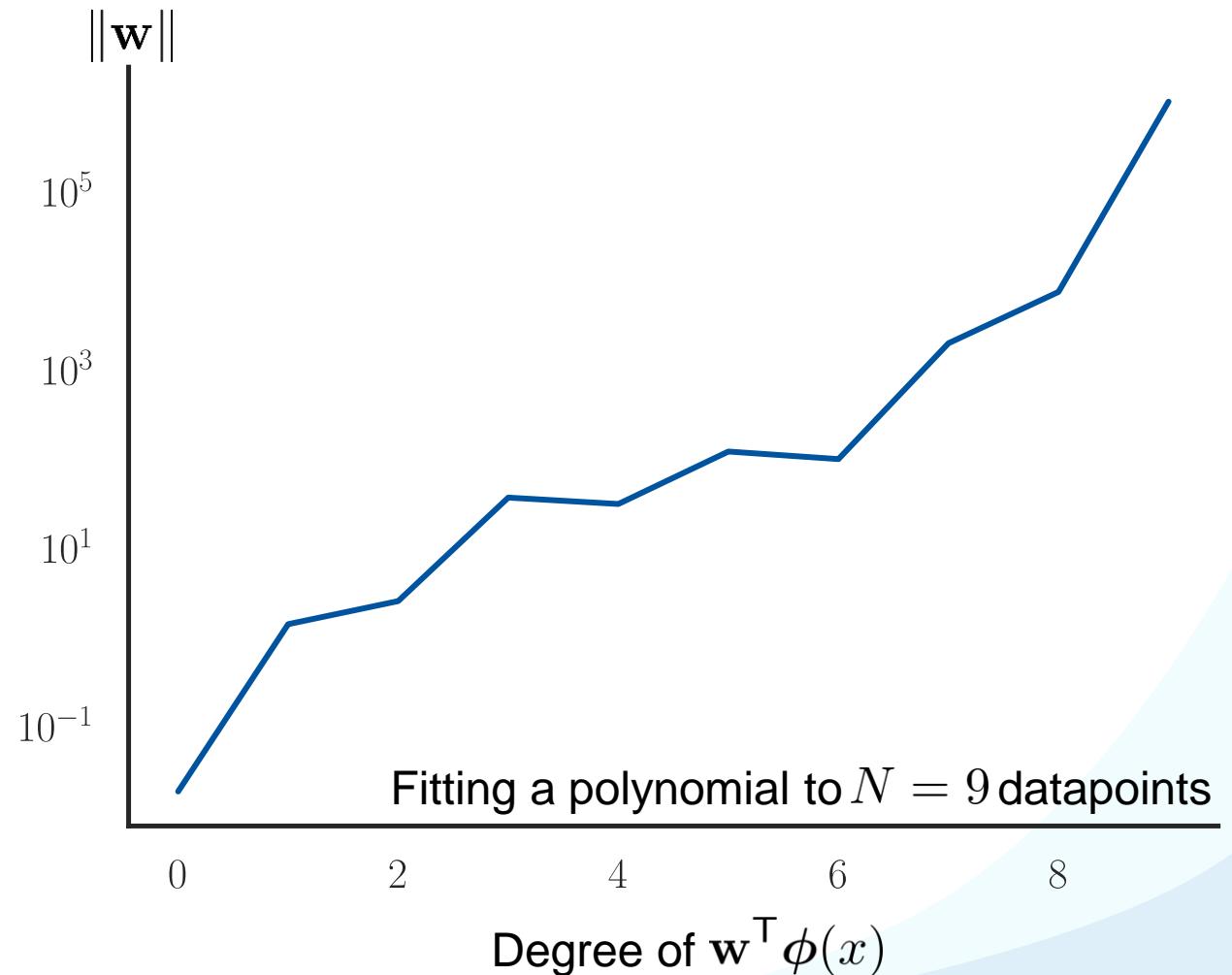
Regularization

- With enough parameters, our model will overfit to the training set.
- This leads to very large coefficient values w_i and thus to a large $\|\mathbf{w}\|$.
- Solution: penalize large parameters.

$$E(\mathbf{w}) = L(\mathbf{w}) + \lambda\Omega(\mathbf{w})$$

$$\Omega(\mathbf{w}) = \frac{1}{2}\|\mathbf{w}\|^2$$

- $L(\mathbf{w})$ is called the **loss** term. Here, we can use the familiar squared loss.
- $\Omega(\mathbf{w})$ is called the **regularizer**. Here, we use a squared regularizer.



Note: Excluding the Bias

- The bias w_0 is usually not regularized, since it does not change the functions' complexity.
- Therefore, we do not include it in $\Omega(\mathbf{w})$ here.
- We can fit the model without a bias by estimating \mathbf{w} on **centered** data:

$$t_n^c = t_n - \bar{t} \quad \mathbf{x}_n^c = \mathbf{x}_n - \bar{\mathbf{x}}$$

$$\bar{t} = \frac{1}{N} \sum_{n=1}^N t_n \quad \bar{\mathbf{x}} = \frac{1}{N} \sum_{n=1}^N \mathbf{x}_n$$

- And computing w_0 afterwards:

$$w_0 = \bar{t} - \mathbf{w}^\top \bar{\mathbf{x}}$$

$$y(\mathbf{x}; \mathbf{w}) = \mathbf{w}^\top \phi(\mathbf{x}) + w_0$$

$$L(\mathbf{w}) = \frac{1}{2} \sum_{n=1}^N ((\mathbf{w}^\top \phi(\mathbf{x}_n) + w_0) - t_n)^2$$

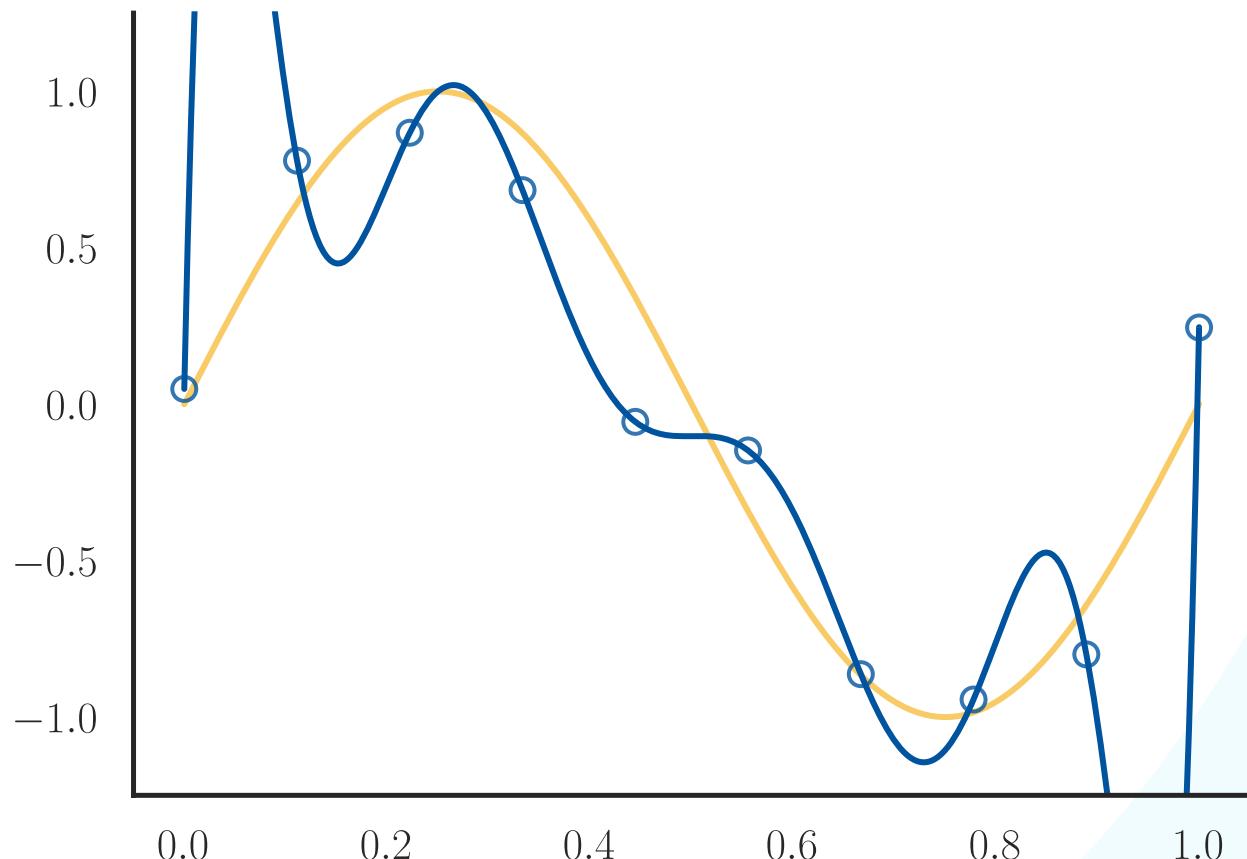
$$\Omega(\mathbf{w}) = \frac{1}{2} \|\mathbf{w}\|^2 = \frac{1}{2} \sum_{j=1}^M w_j^2$$

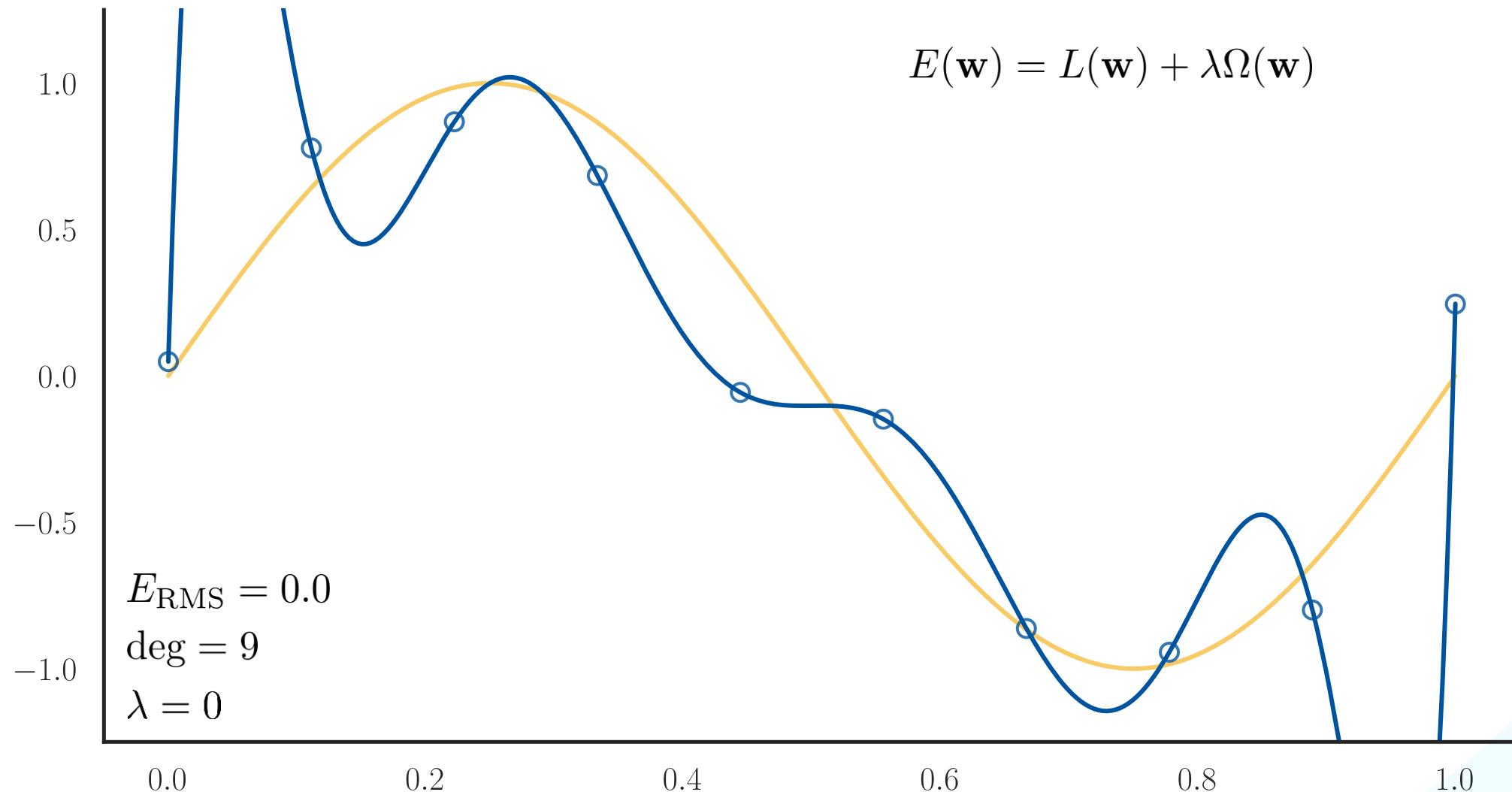
Example: Regularizing a Polynomial

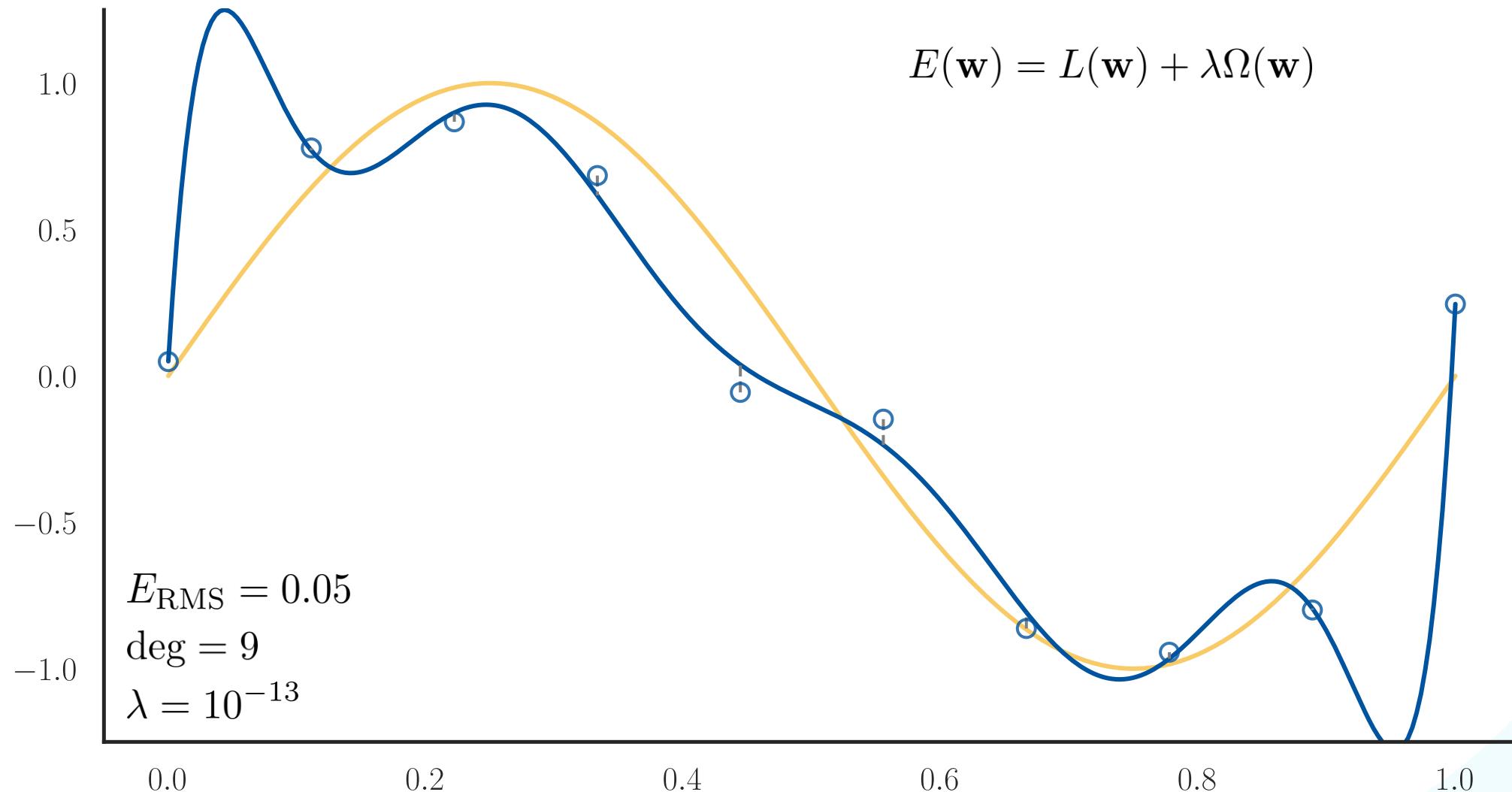
- Again, use polynomial basis functions:

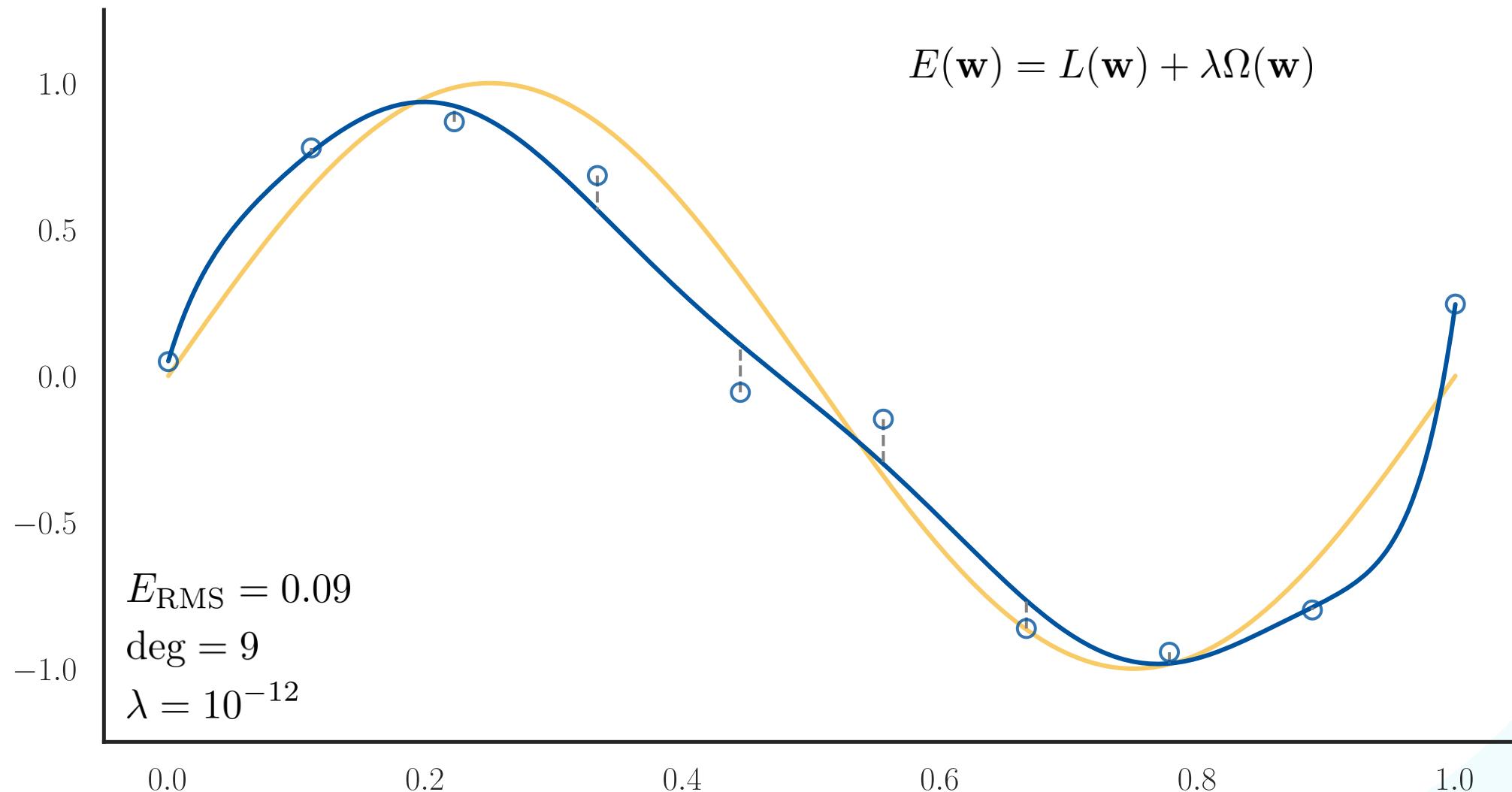
$$\phi_j(x) = (x^j)$$

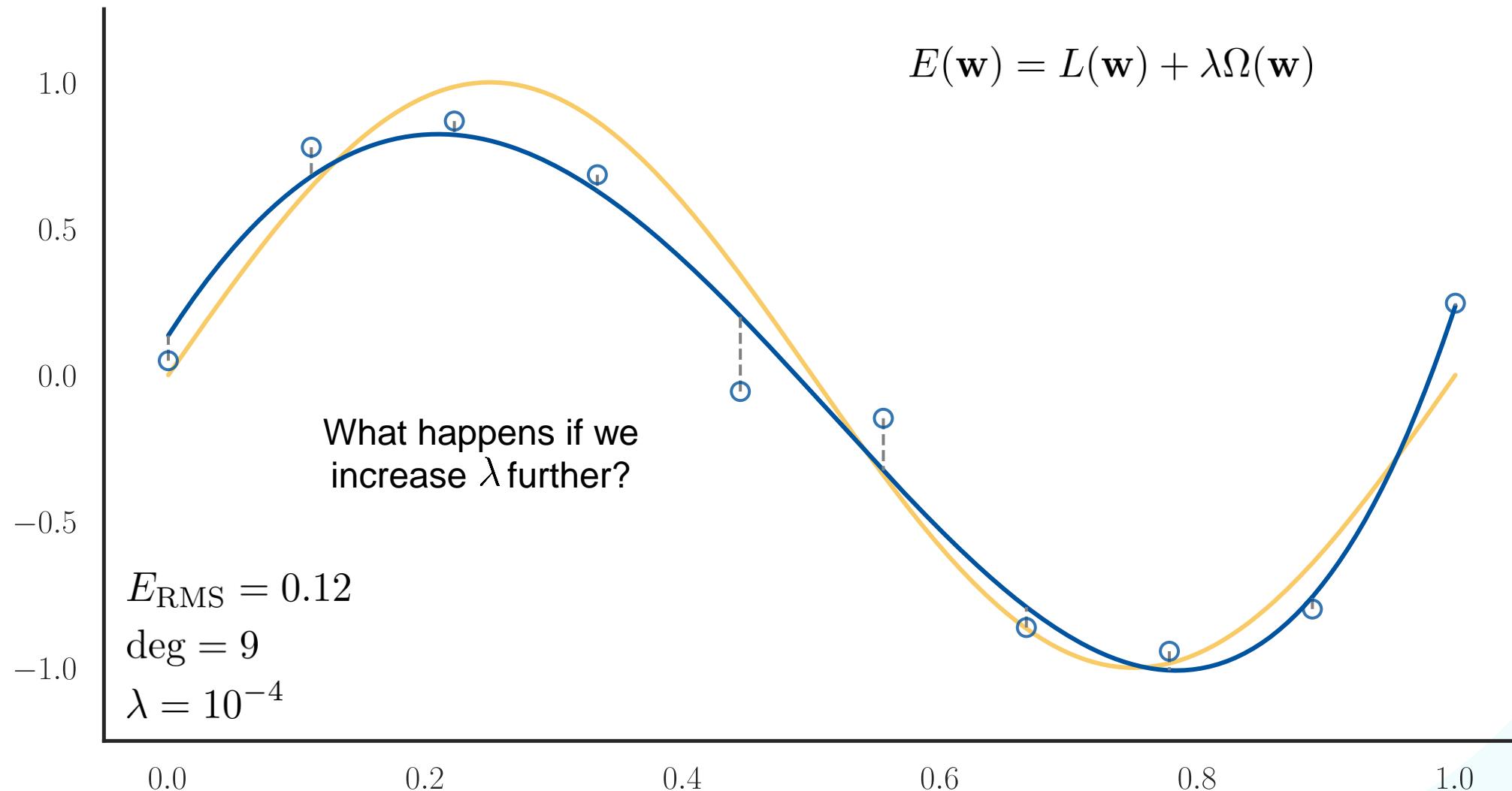
- Start off with an overfitting model.
- How much should we regularize?*

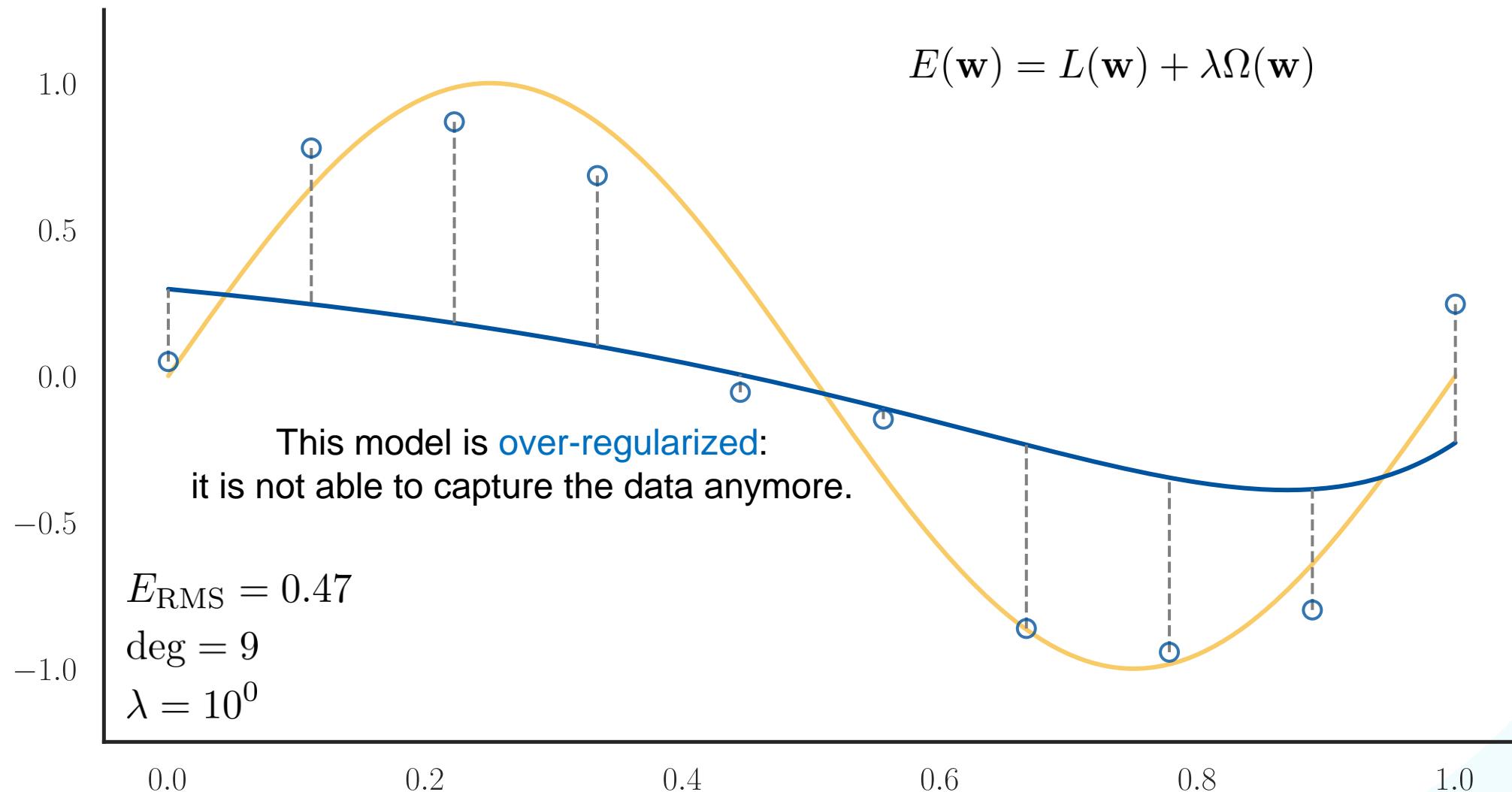






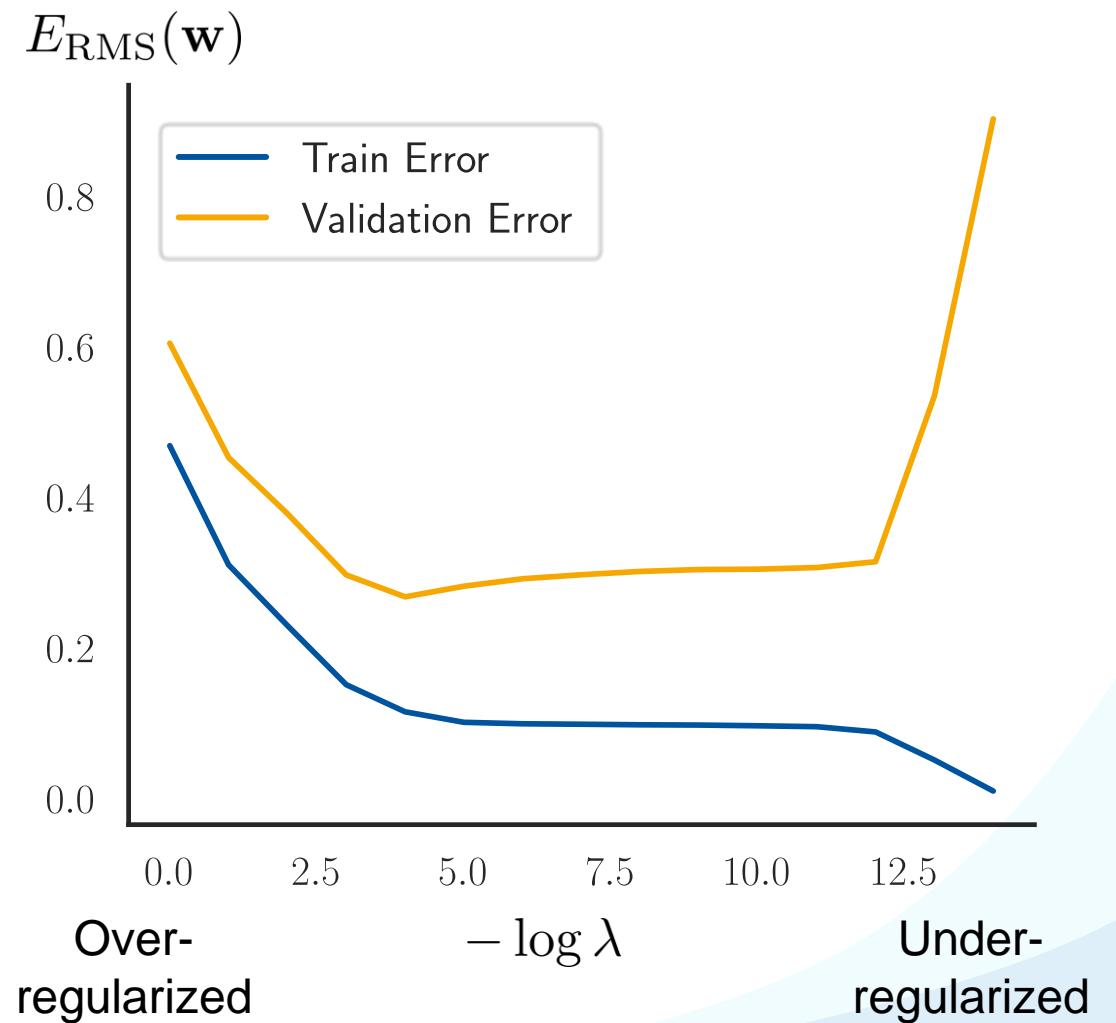






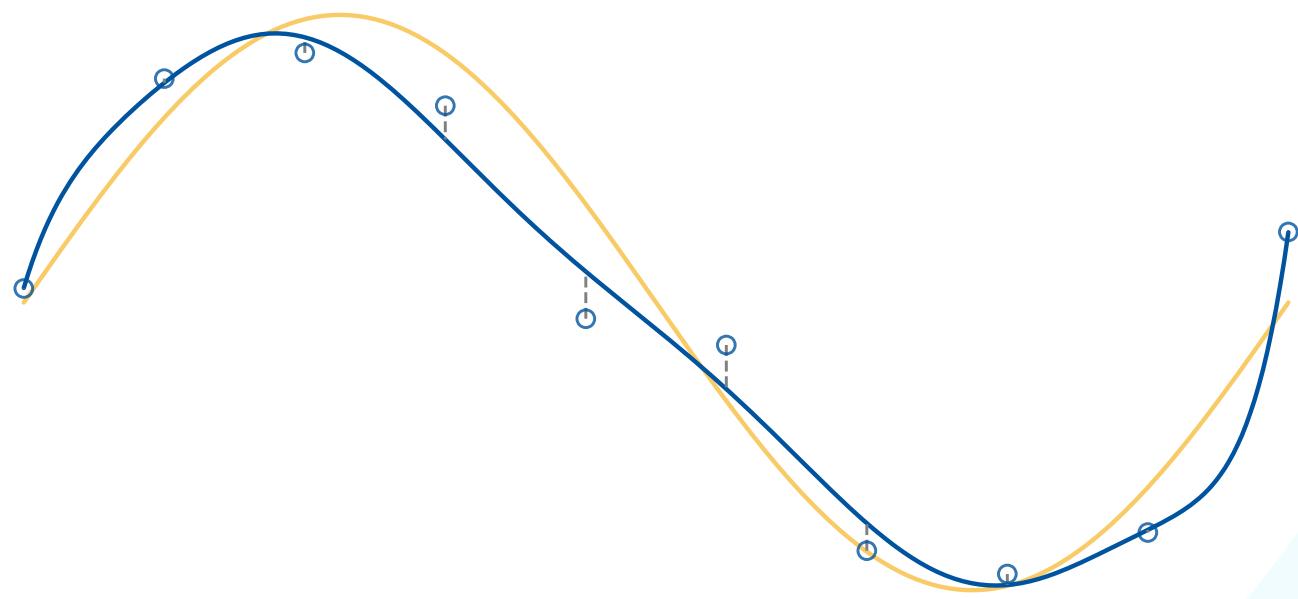
Choosing the right Regularization

- Regularization allows us to apply complex model on small datasets.
- However, we shifted the problem from selecting a suitable model to selecting a suitable regularization.
- The regularization factor λ becomes a [hyperparameter](#).



Linear Regression

1. Motivation
2. Least-Squares Regression
3. Regularization
4. **Ridge Regression**
5. The Bias-Variance Tradeoff



Ridge Regression

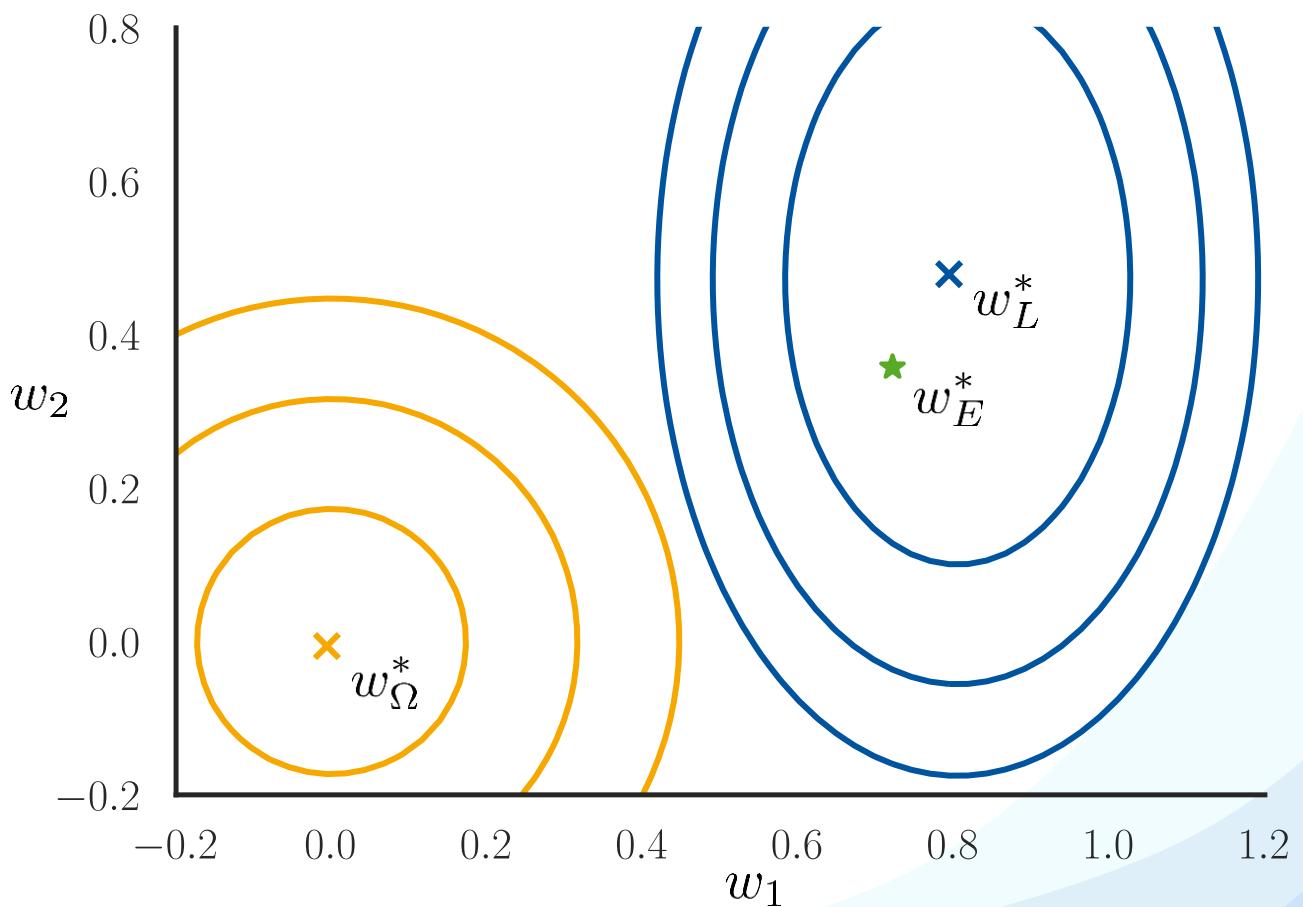
- We want to jointly minimize the squared error and the regularization term:

$$E(\mathbf{w}) = L(\mathbf{w}) + \lambda\Omega(\mathbf{w})$$

$$L(\mathbf{w}) = \frac{1}{2} \sum_{n=1}^N (y(\mathbf{x}_n; \mathbf{w}) - t_n)^2$$

$$\Omega(\mathbf{w}) = \frac{1}{2} \|\mathbf{w}\|^2$$

- This model is called ridge regression.



Derivation

$$E(\mathbf{w}) = \frac{1}{2} \sum_{n=1}^N (y(\mathbf{x}_n; \mathbf{w}) - t_n)^2 + \frac{\lambda}{2} \|\mathbf{w}\|^2$$

$$\begin{aligned} \nabla E(\mathbf{w}) &= \sum_{n=1}^N (\mathbf{w}^\top \boldsymbol{\phi}(\mathbf{x}_n) - t_n) \boldsymbol{\phi}(\mathbf{x}_n) + \lambda \mathbf{w} \stackrel{!}{=} 0 \\ &\quad \boldsymbol{\Phi}^\top (\boldsymbol{\Phi} \mathbf{w} - \mathbf{t}) + \lambda \mathbf{w} = 0 \end{aligned}$$

$$(\boldsymbol{\Phi}^\top \boldsymbol{\Phi} + \lambda \mathbf{I}) \mathbf{w} = \boldsymbol{\Phi}^\top \mathbf{t}$$

$$\mathbf{w} = (\boldsymbol{\Phi}^\top \boldsymbol{\Phi} + \lambda \mathbf{I})^{-1} \boldsymbol{\Phi}^\top \mathbf{t}$$

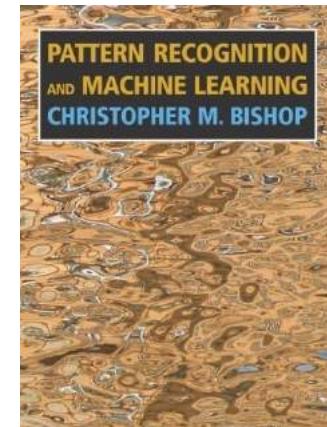


Effect of **regularization**: keeps the inverse well-conditioned.

References and Further Reading

- More information about [Linear Discriminants](#) is available in Chapter 4.1 of Bishop's book. For more information about [Linear Regression](#), read Chapter 3.1.

Christopher M. Bishop
Pattern Recognition and Machine Learning
Springer, 2006



Elements of Machine Learning & Data Science

Winter semester 2023/24

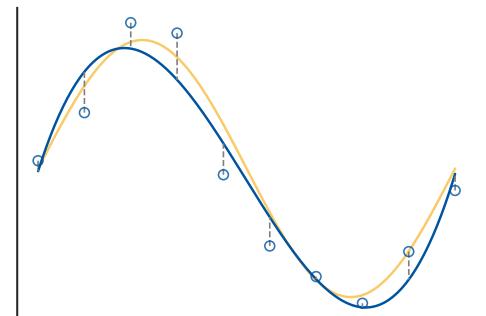
Lecture 16 – Logistic Regression

08.12.2023

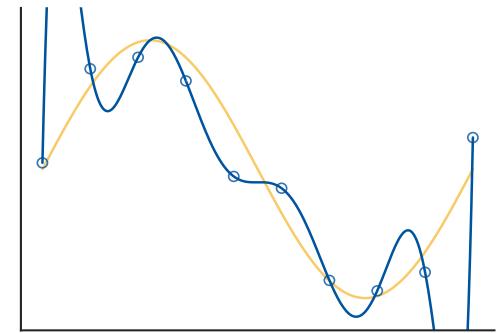
Prof. Bastian Leibe

Machine Learning Topics

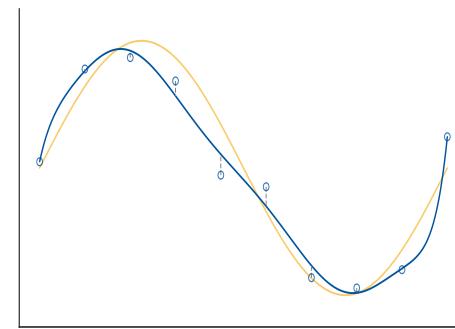
1. Introduction to ML
2. Probability Density Estimation
3. Linear Discriminants
- 4. Linear Regression**
5. Logistic Regression
6. Support Vector Machines
7. AdaBoost
8. Neural Network Basics



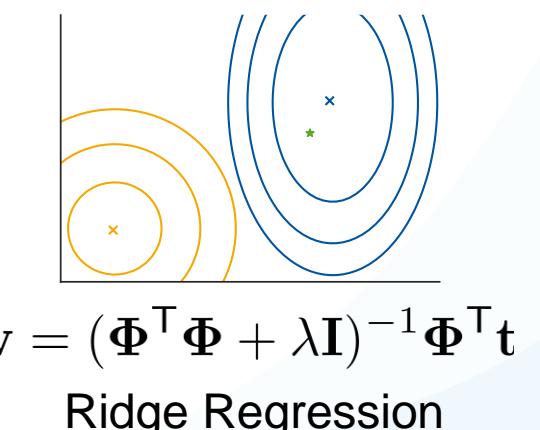
$y(\mathbf{x}) = \mathbf{w}^T \phi(\mathbf{x}) + w_0$
Linear Regression Functions



Overfitting



$E(\mathbf{w}) = L(\mathbf{w}) + \lambda \Omega(\mathbf{w})$
Regularization



$\mathbf{w} = (\Phi^T \Phi + \lambda \mathbf{I})^{-1} \Phi^T \mathbf{t}$
Ridge Regression

Recap: Least-Squares Regression

- We want to optimize the difference between our predictor $y(\mathbf{x}_n; \mathbf{w})$ and the targets t_n .
- The only difference is that our targets t_n are now continuous values.
- Again, use the familiar **squared error** objective:

$$E(\mathbf{w}) = \frac{1}{2} \sum_{n=1}^N (y(\mathbf{x}_n; \mathbf{w}) - t_n)^2$$

- This has the same solution as for classification (normal equations).

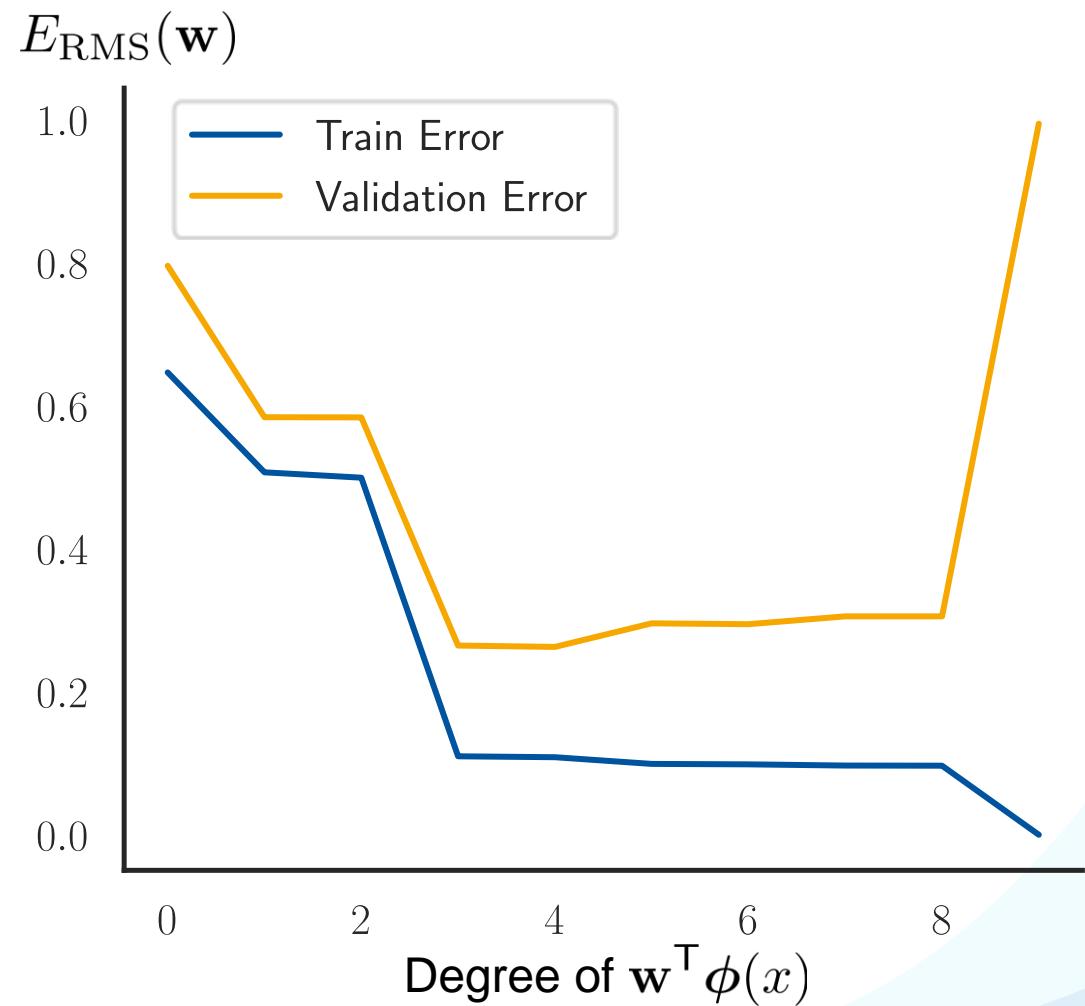
$$y(\mathbf{x}_n; \mathbf{w}) = \mathbf{w}^\top \phi(\mathbf{x}_n)$$

$$\begin{aligned}\frac{\partial E(\mathbf{w})}{\partial \mathbf{w}} &= \sum_{n=1}^N (\mathbf{w}^\top \phi_n - t_n) \phi_n \\ &= \Phi^\top (\Phi \mathbf{w} - \mathbf{t}) \stackrel{!}{=} 0\end{aligned}$$

$$\Rightarrow \mathbf{w} = (\Phi^\top \Phi)^{-1} \Phi^\top \mathbf{t}$$

Recap: Overfitting

- We fit the dataset perfectly, but the resulting function is clearly not what we want.
- This phenomenon is called **overfitting**.
- Remember: we assume $t_n = h(\mathbf{x}_n) + \epsilon$.
- Our model is “too” powerful and models the noise instead of the underlying function!
- *What can we do to avoid overfitting?*



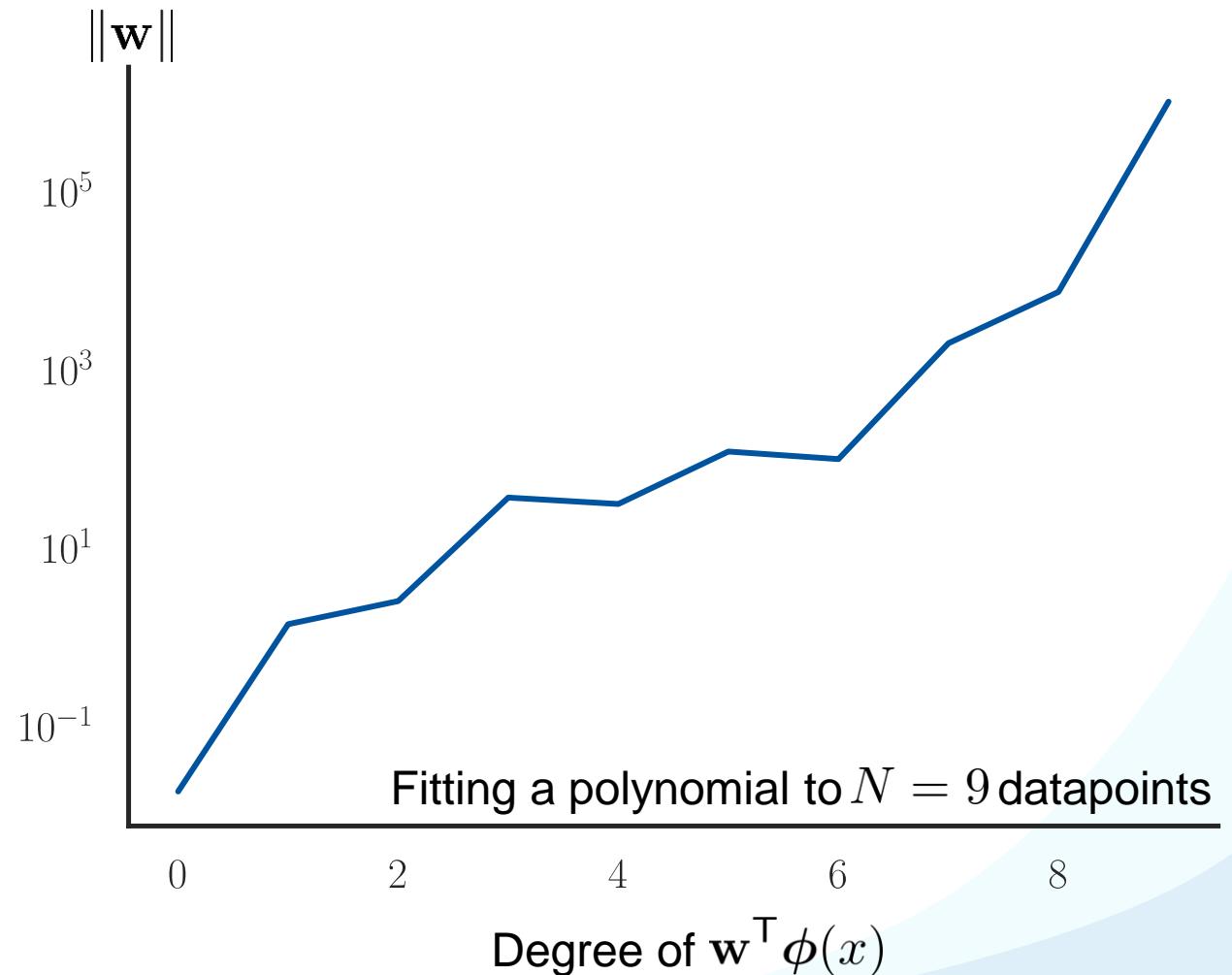
Recap: Regularization

- With enough parameters, our model will overfit to the training set.
- This leads to very large coefficient values w_i and thus to a large $\|\mathbf{w}\|$.
- Solution: penalize large parameters.

$$E(\mathbf{w}) = L(\mathbf{w}) + \lambda \Omega(\mathbf{w})$$

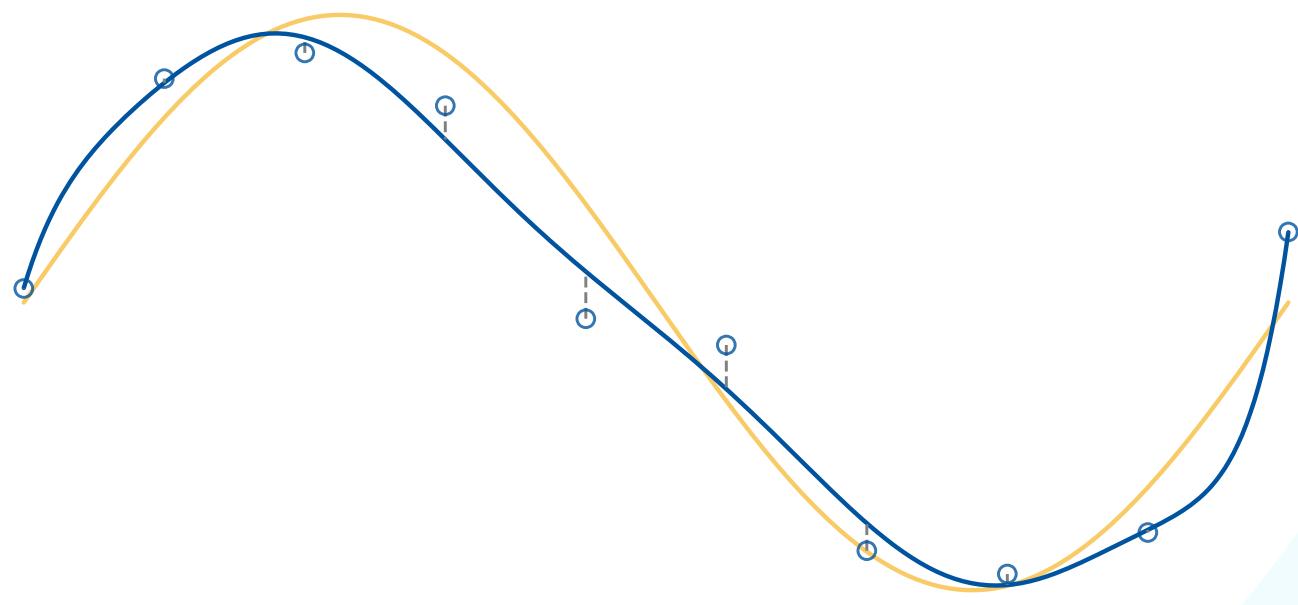
$$\Omega(\mathbf{w}) = \frac{1}{2} \|\mathbf{w}\|^2$$

- $L(\mathbf{w})$ is called the **loss** term. Here, we can use the familiar squared loss.
- $\Omega(\mathbf{w})$ is called the **regularizer**. Here, we use a squared regularizer.



Linear Regression

1. Motivation
2. Least-Squares Regression
3. Regularization
4. **Ridge Regression**



Ridge Regression

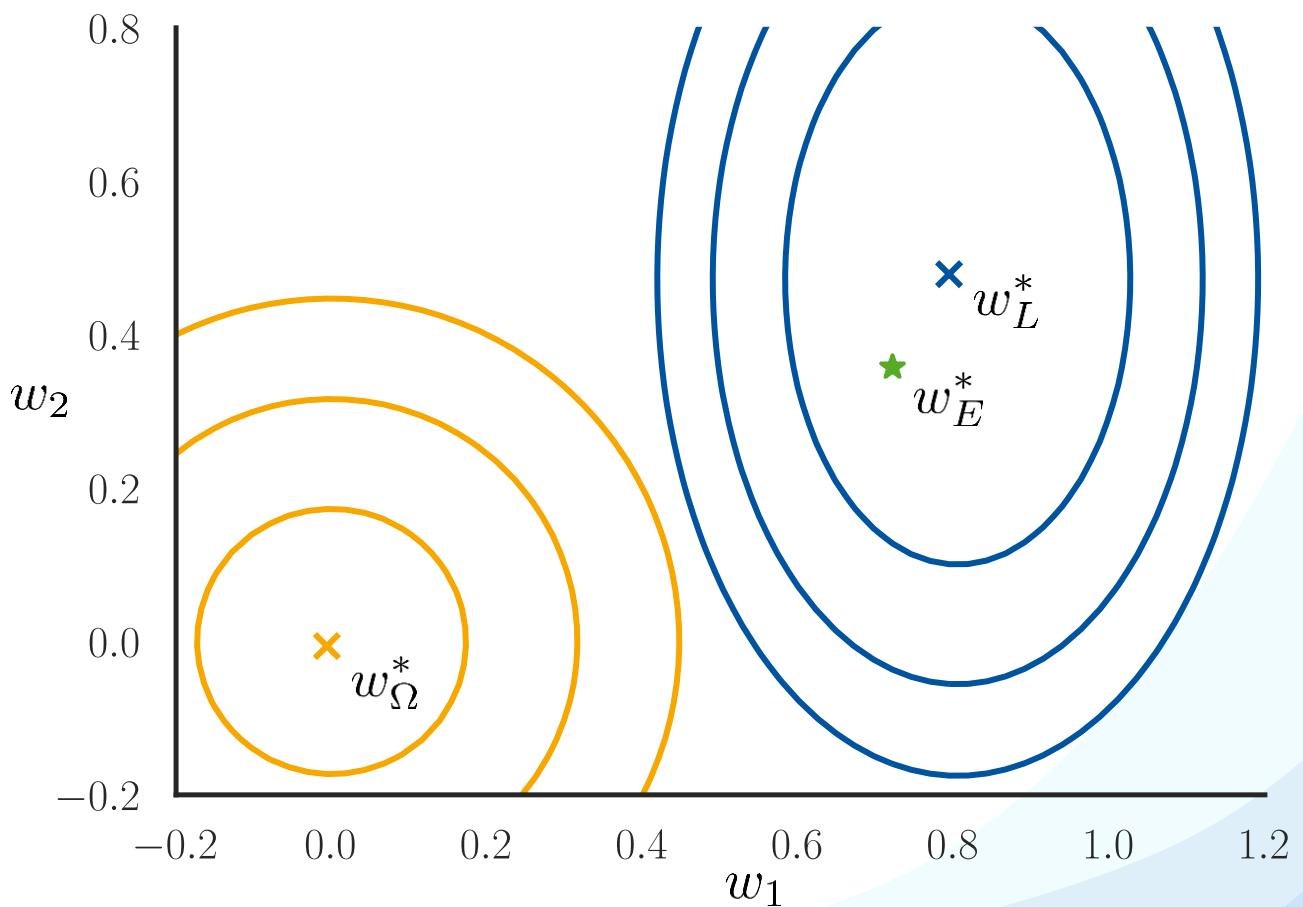
- We want to jointly minimize the squared error and the regularization term:

$$E(\mathbf{w}) = L(\mathbf{w}) + \lambda\Omega(\mathbf{w})$$

$$L(\mathbf{w}) = \frac{1}{2} \sum_{n=1}^N (y(\mathbf{x}_n; \mathbf{w}) - t_n)^2$$

$$\Omega(\mathbf{w}) = \frac{1}{2} \|\mathbf{w}\|^2$$

- This model is called ridge regression.



Derivation

$$E(\mathbf{w}) = \frac{1}{2} \sum_{n=1}^N (y(\mathbf{x}_n; \mathbf{w}) - t_n)^2 + \frac{\lambda}{2} \|\mathbf{w}\|^2$$

$$\begin{aligned}\nabla E(\mathbf{w}) &= \sum_{n=1}^N (\mathbf{w}^\top \boldsymbol{\phi}(\mathbf{x}_n) - t_n) \boldsymbol{\phi}(\mathbf{x}_n) + \lambda \mathbf{w} \stackrel{!}{=} 0 \\ \boldsymbol{\Phi}^\top (\boldsymbol{\Phi} \mathbf{w} - \mathbf{t}) + \lambda \mathbf{w} &= 0\end{aligned}$$

$$(\boldsymbol{\Phi}^\top \boldsymbol{\Phi} + \lambda \mathbf{I}) \mathbf{w} = \boldsymbol{\Phi}^\top \mathbf{t}$$

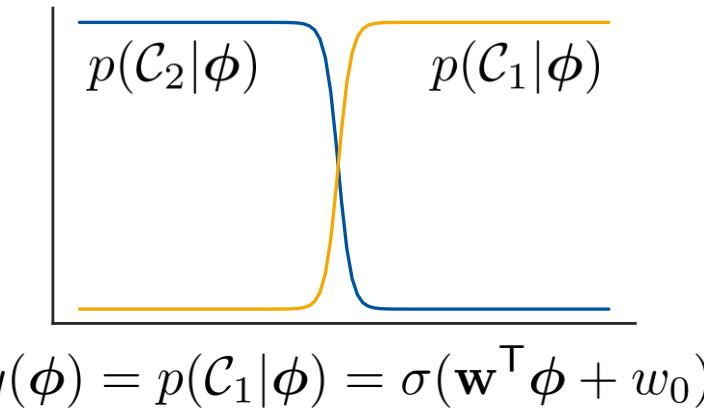
$$\mathbf{w} = (\boldsymbol{\Phi}^\top \boldsymbol{\Phi} + \lambda \mathbf{I})^{-1} \boldsymbol{\Phi}^\top \mathbf{t}$$



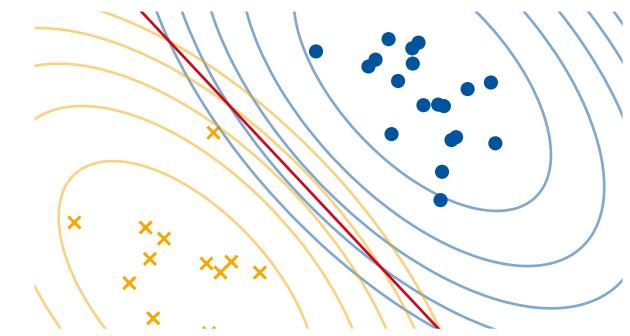
Effect of **regularization**: keeps the inverse well-conditioned.

Machine Learning Topics

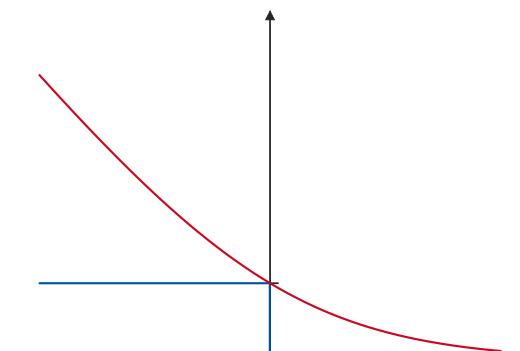
1. Introduction to ML
2. Probability Density Estimation
3. Linear Discriminants
4. Linear Regression
5. **Logistic Regression**
6. Support Vector Machines
7. AdaBoost
8. Neural Network Basics



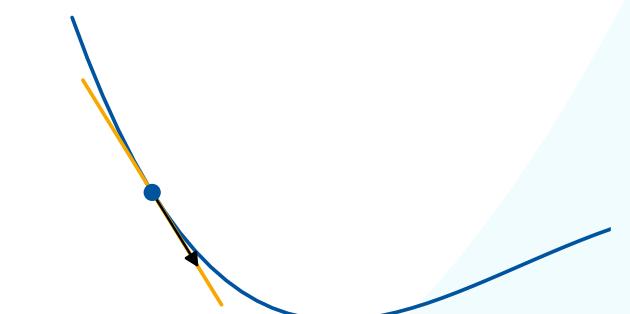
Logistic Regression
Formulation



Parameter Efficiency



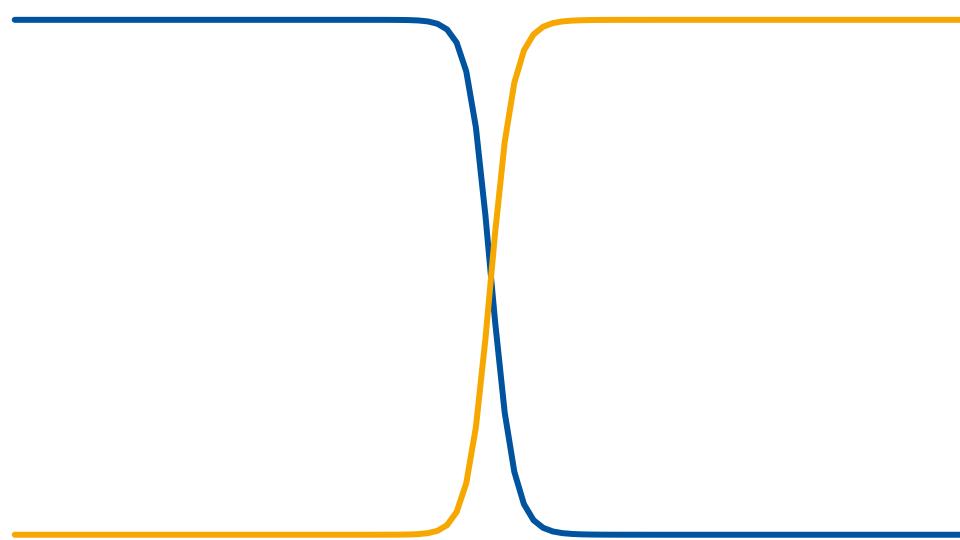
$E(\mathbf{w}) = -\ln p(\mathbf{t}|\mathbf{w})$
Cross-Entropy Error



$\mathbf{w}^{(\tau+1)} = \mathbf{w}^{(\tau)} - \eta \nabla E(\mathbf{w})$
Iterative Optimization

Logistic Regression

- 1. Logistic Regression Formulation**
2. Motivation and Background
3. Iterative Optimization
4. First-Order Gradient Descent
5. Second-Order Gradient Descent
6. Error Function Analysis



Motivation

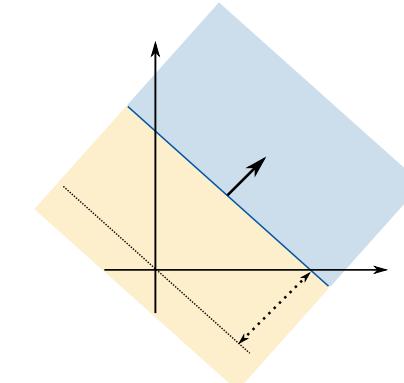
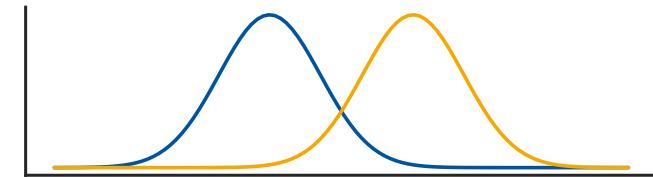
- We have seen how to build probabilistic classifiers using Bayes' Theorem:

$$y_k(\mathbf{x}) = p(\mathcal{C}_k|\mathbf{x}) \propto p(\mathbf{x}|\mathcal{C}_k)p(\mathcal{C}_k)$$

- We have directly modeled the decision boundary with linear discriminants:

$$y(\mathbf{x}) = \mathbf{w}^T \mathbf{x} + w_0$$

- In the following, we will combine those two ideas
 - We will model the posterior $p(\mathcal{C}_k|\mathbf{x})$
 - But we will do that using a linear discriminant function $y(\mathbf{x}) = \mathbf{w}^T \mathbf{x} + w_0$
- The resulting model will be called **logistic regression**.



Reminder: Probabilistic Classification

- Remember what we did in probabilistic classification
 - We modeled the likelihood of each class

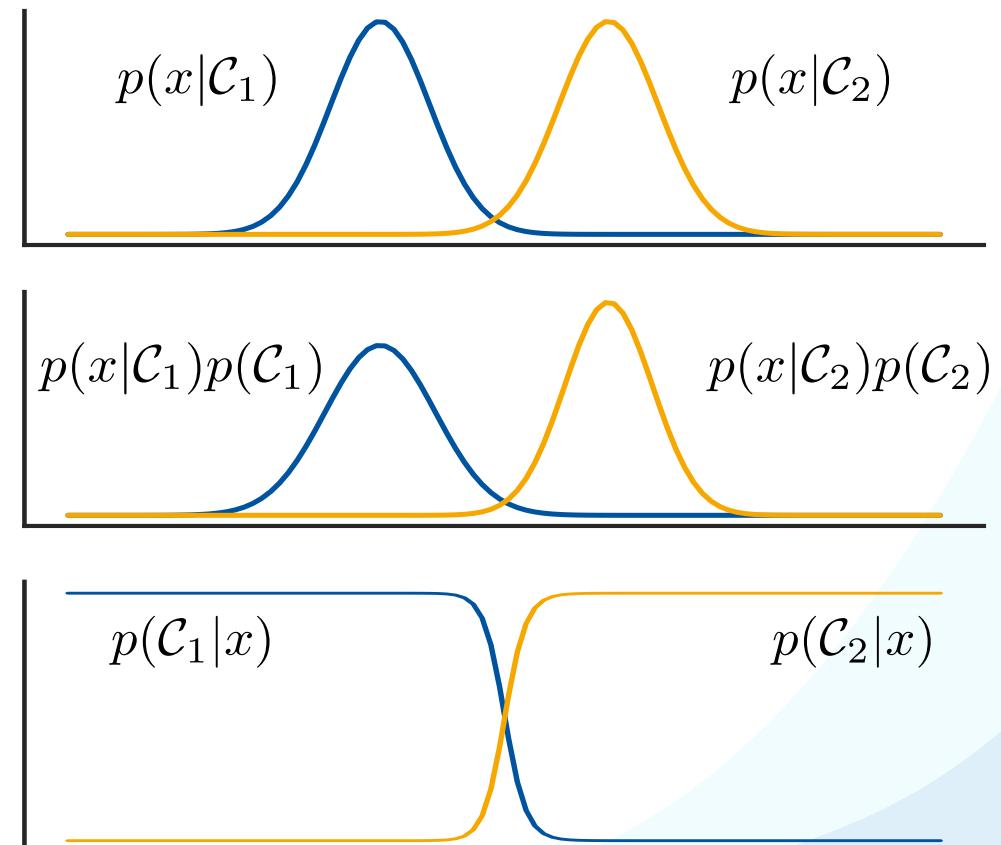
$$p(\mathbf{x}|\mathcal{C}_k)$$

- We scaled the likelihoods with the priors

$$p(\mathbf{x}|\mathcal{C}_k)p(\mathcal{C}_k)$$

- We normalized to compute the posterior

$$p(\mathcal{C}_1|\mathbf{x}) = \frac{p(\mathbf{x}|\mathcal{C}_1)p(\mathcal{C}_1)}{p(\mathbf{x}|\mathcal{C}_1)p(\mathcal{C}_1) + p(\mathbf{x}|\mathcal{C}_2)p(\mathcal{C}_2)}$$



- Let's now start with the posterior and rewrite it

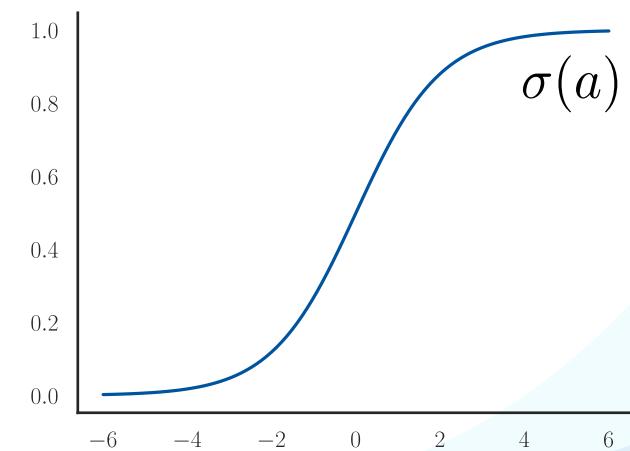
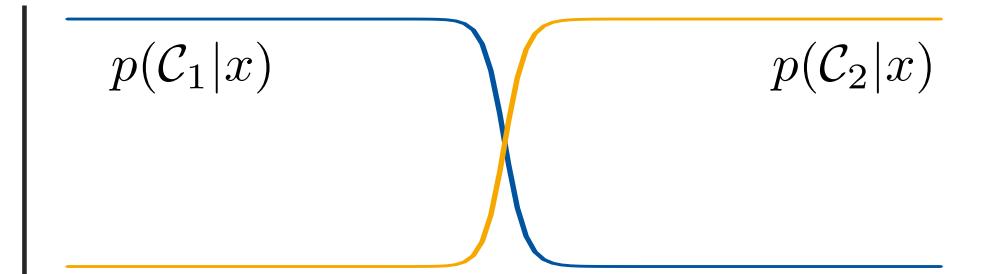
$$\begin{aligned}
 p(\mathcal{C}_1|\mathbf{x}) &= \frac{p(\mathbf{x}|\mathcal{C}_1)p(\mathcal{C}_1)}{p(\mathbf{x}|\mathcal{C}_1)p(\mathcal{C}_1) + p(\mathbf{x}|\mathcal{C}_2)p(\mathcal{C}_2)} \\
 &= \frac{1}{1 + \frac{p(\mathbf{x}|\mathcal{C}_2)p(\mathcal{C}_2)}{p(\mathbf{x}|\mathcal{C}_1)p(\mathcal{C}_1)}} \\
 &= \frac{1}{1 + \exp(-a)} \quad =: \sigma(a)
 \end{aligned}$$

- This is the equation for the **logistic sigmoid** function $\sigma(a)$

⇒ If we set

$$a = \ln \frac{p(\mathbf{x}|\mathcal{C}_1)p(\mathcal{C}_1)}{p(\mathbf{x}|\mathcal{C}_2)p(\mathcal{C}_2)}$$

the logistic sigmoid expresses a posterior probability!



Properties of the Logistic Sigmoid

- Definition:

$$\sigma(a) = \frac{1}{1 + \exp(-a)}$$

- Inverse (also known as **logit** function):

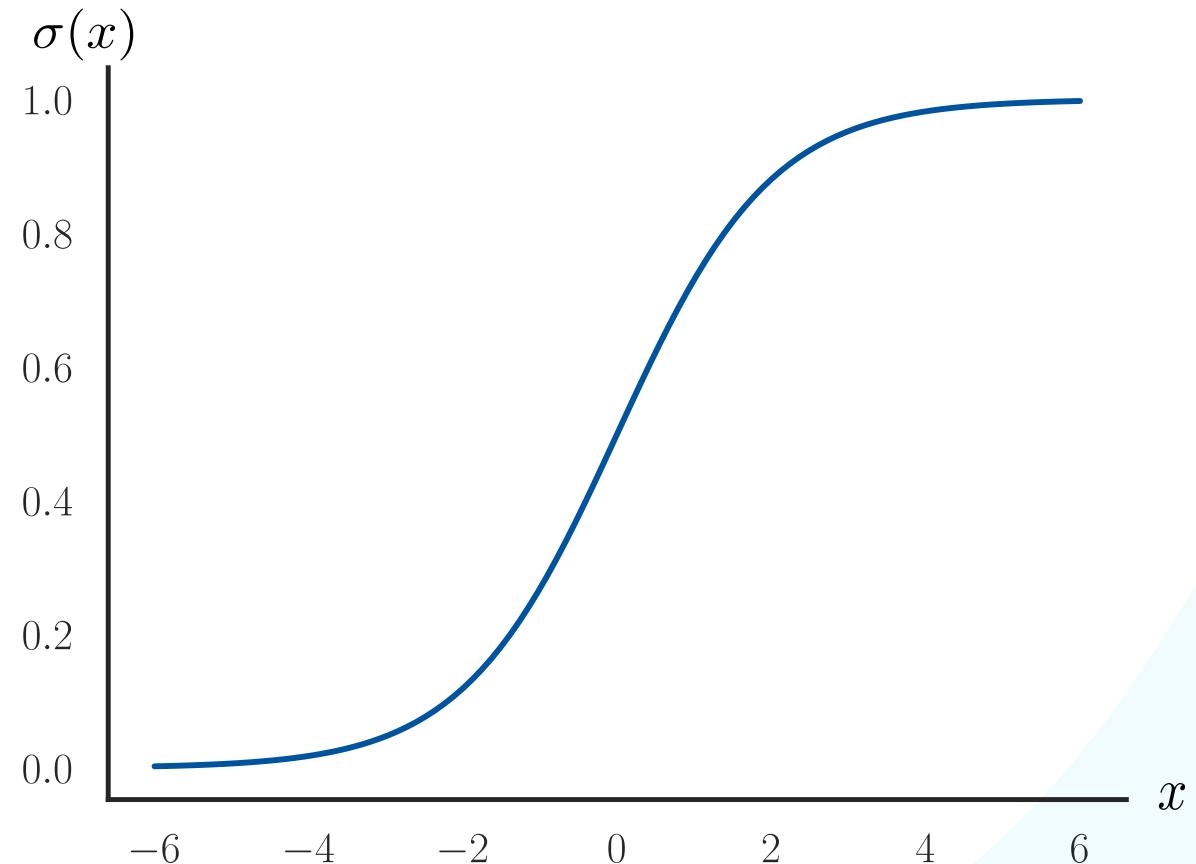
$$a = \ln\left(\frac{\sigma(a)}{1 - \sigma(a)}\right)$$

- Symmetry:

$$\sigma(-a) = 1 - \sigma(a)$$

- Derivative:

$$\frac{\partial \sigma(a)}{\partial a} = \sigma(a)(1 - \sigma(a))$$



Logistic Regression

- We now define the **logistic regression** model
 - For the start, let us assume two classes $\mathcal{C}_1, \mathcal{C}_2$.
 - We model the class posteriors $p(\mathcal{C}_k|\mathbf{x})$ as

$$p(\mathcal{C}_1|\mathbf{x}) = y(\mathbf{x}) = \sigma(\mathbf{w}^T \mathbf{x})$$



$$p(\mathcal{C}_2|\mathbf{x}) = 1 - p(\mathcal{C}_1|\mathbf{x})$$

Logistic sigmoid
activation function:

$$\sigma(a) = \frac{1}{1 + \exp(-a)}$$

- I.e., we define a linear discriminant model

$$y(\mathbf{x}) = \sigma(\mathbf{w}^T \mathbf{x})$$

that is meant to represent the class posterior with
the help of a logistic sigmoid activation function $\sigma(a)$.

- Our target labels are now $t_n \in \{0, 1\}$.

Error Function

- Consider a data set $\mathcal{D} = \{(\phi_1, t_1), \dots, (\phi_N, t_N)\}$
 - with data points $\Phi = [\phi_1, \dots, \phi_N]^\top$, $\phi_n = \phi(\mathbf{x}_n)$
 - And target labels $\mathbf{t} = [t_1, \dots, t_N]^\top$, $t_n \in \{0, 1\}$
- Maximum likelihood approach**
 - With $y_n = p(\mathcal{C}_1 | \phi_n) = \sigma(\mathbf{w}^\top \phi_n)$
 - We model the probability of the target labels \mathbf{t} given our model parameters \mathbf{w} as

Trick: use t_n as an indicator variable

$$p(\mathbf{t} | \mathbf{w}) = \prod_{n=1}^N \begin{cases} p(\mathcal{C}_1 | \phi_n) & , t_n = 1 \\ p(\mathcal{C}_2 | \phi_n) & , t_n = 0 \end{cases} = \prod_{n=1}^N \begin{cases} y_n & , t_n = 1 \\ (1 - y_n) & , t_n = 0 \end{cases} = \prod_{n=1}^N y_n^{t_n} (1 - y_n)^{1-t_n}$$

- Maximum likelihood approach

$$p(\mathbf{t}|\mathbf{w}) = \prod_{n=1}^N y_n^{t_n} (1 - y_n)^{1-t_n}$$

- Define the error function as the negative log-likelihood

$$E(\mathbf{w}) = -\ln p(\mathbf{t}|\mathbf{w})$$

$$= -\sum_{n=1}^N (t_n \ln y_n + (1 - t_n) \ln(1 - y_n))$$

- This function is known as the binary cross-entropy error.

Softmax Regression

- Multi-class extension of logistic regression
 - Generalization to K classes with target labels in 1-of- K notation $\mathbf{t}_n = [0, 1, \dots, 0]^T$
 - Again, we define a linear discriminant function that models the class posteriors

$$\mathbf{y}(\mathbf{x}; \mathbf{w}) = \begin{bmatrix} p(\mathcal{C}_1 | \mathbf{x}; \mathbf{w}) \\ p(\mathcal{C}_2 | \mathbf{x}; \mathbf{w}) \\ \vdots \\ p(\mathcal{C}_K | \mathbf{x}; \mathbf{w}) \end{bmatrix} = \frac{1}{\sum_{j=1}^K \exp(\mathbf{w}_j^\top \mathbf{x})} \begin{bmatrix} \exp(\mathbf{w}_1^\top \mathbf{x}) \\ \exp(\mathbf{w}_2^\top \mathbf{x}) \\ \vdots \\ \exp(\mathbf{w}_K^\top \mathbf{x}) \end{bmatrix}$$

- This makes use of the **softmax** function as a multi-class extension of the logistic sigmoid

$$\text{softmax}(\mathbf{a}) = \frac{\exp(a_k)}{\sum_{j=1}^K \exp(a_j)}$$

- We can write the **binary cross-entropy error** as

$$\begin{aligned} E(\mathbf{w}) &= - \sum_{n=1}^N (t_n \ln y_n + (1 - t_n) \ln(1 - y_n)) \\ &= - \sum_{n=1}^N \sum_{k=0}^1 (\mathbb{I}(t_n = k) \ln p(\mathcal{C}_k | \mathbf{x}_n; \mathbf{w})) \end{aligned}$$

indicator function

$$\mathbb{I}(\varphi) = \begin{cases} 1 & \text{if } \varphi \text{ is true} \\ 0 & \text{otherwise} \end{cases}$$

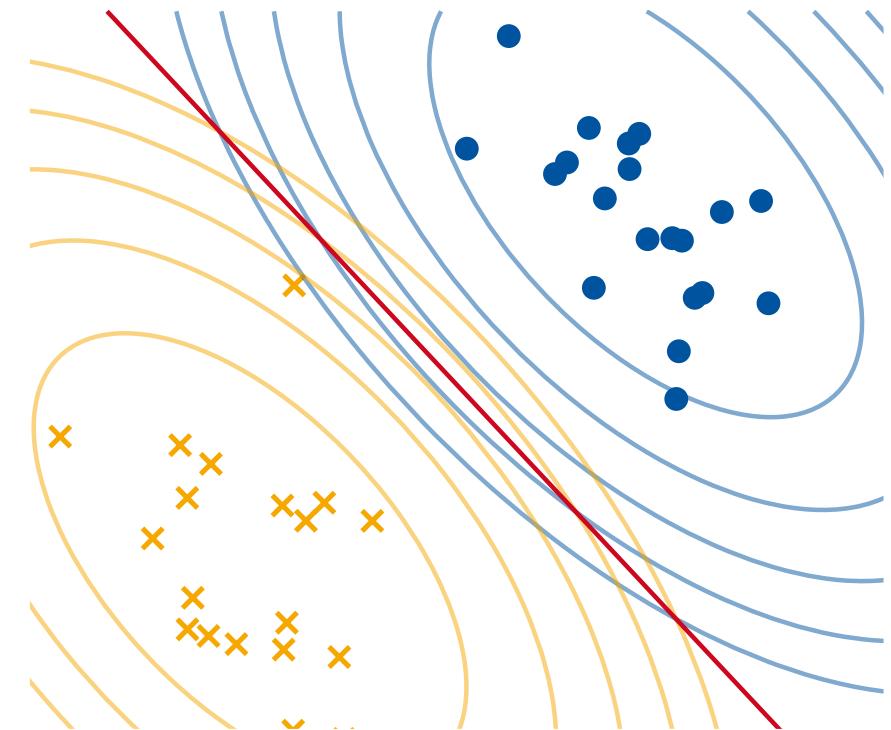
- Using one-hot labels \mathbf{t}_n , the generalization to K classes is:

$$E(\mathbf{w}) = - \sum_{n=1}^N \sum_{k=1}^K \left(\mathbb{I}(t_{kn} = 1) \ln \frac{\exp(\mathbf{w}_k^\top \mathbf{x})}{\sum_{j=1}^K \exp(\mathbf{w}_j^\top \mathbf{x})} \right)$$

- This function is known as the **multi-class cross-entropy error** or **softmax cross-entropy error**.

Logistic Regression

1. Logistic Regression Formulation
2. **Motivation and Background**
3. Iterative Optimization
4. First-Order Gradient Descent
5. Second-Order Gradient Descent
6. Error Function Analysis

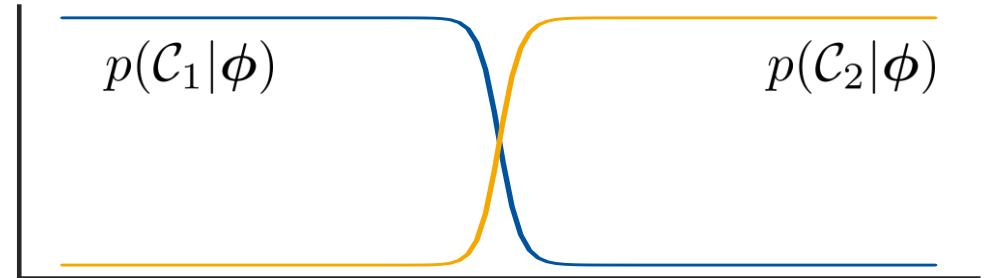


Motivation: Why Logistic Regression?

- Logistic Regression uses models of the form

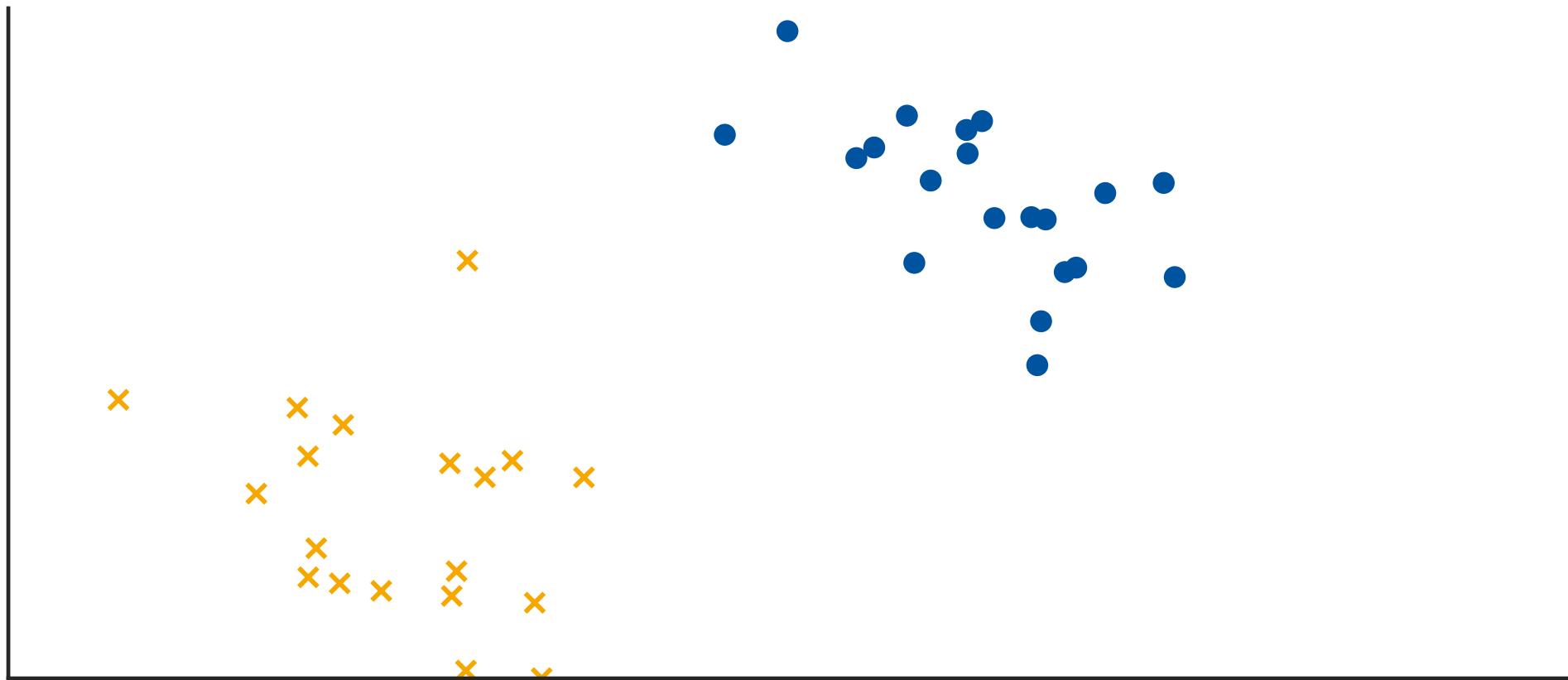
$$p(\mathcal{C}_1|\phi) = y(\phi) = \sigma(\mathbf{w}^T \phi)$$

$$p(\mathcal{C}_2|\phi) = 1 - p(\mathcal{C}_1|\phi)$$



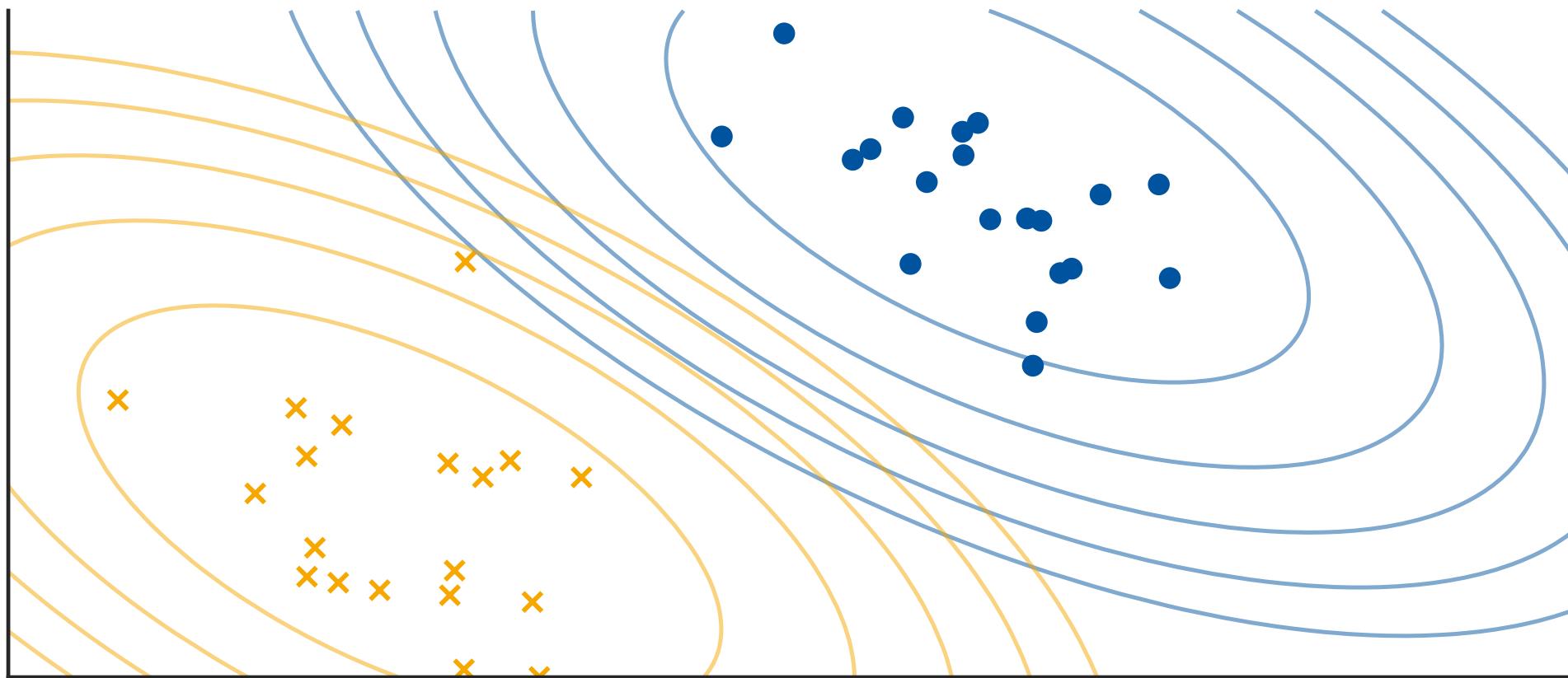
- Interpretation
 - We model the **class posteriors** $p(\mathcal{C}_k|\phi)$, as required to make Bayes optimal decisions.
 - We have seen previously that we can obtain $p(\mathcal{C}_k|\phi) = p(\phi|\mathcal{C}_k)p(\mathcal{C}_k)$.
 - However, here we model $p(\mathcal{C}_k|\phi)$ as a **linear discriminant function** $y(\phi) = \sigma(\mathbf{w}^T \phi)$ instead.
- *Why should we do this?*
 - *What advantage does such a model have compared to direct modeling of the probabilities?*

Example



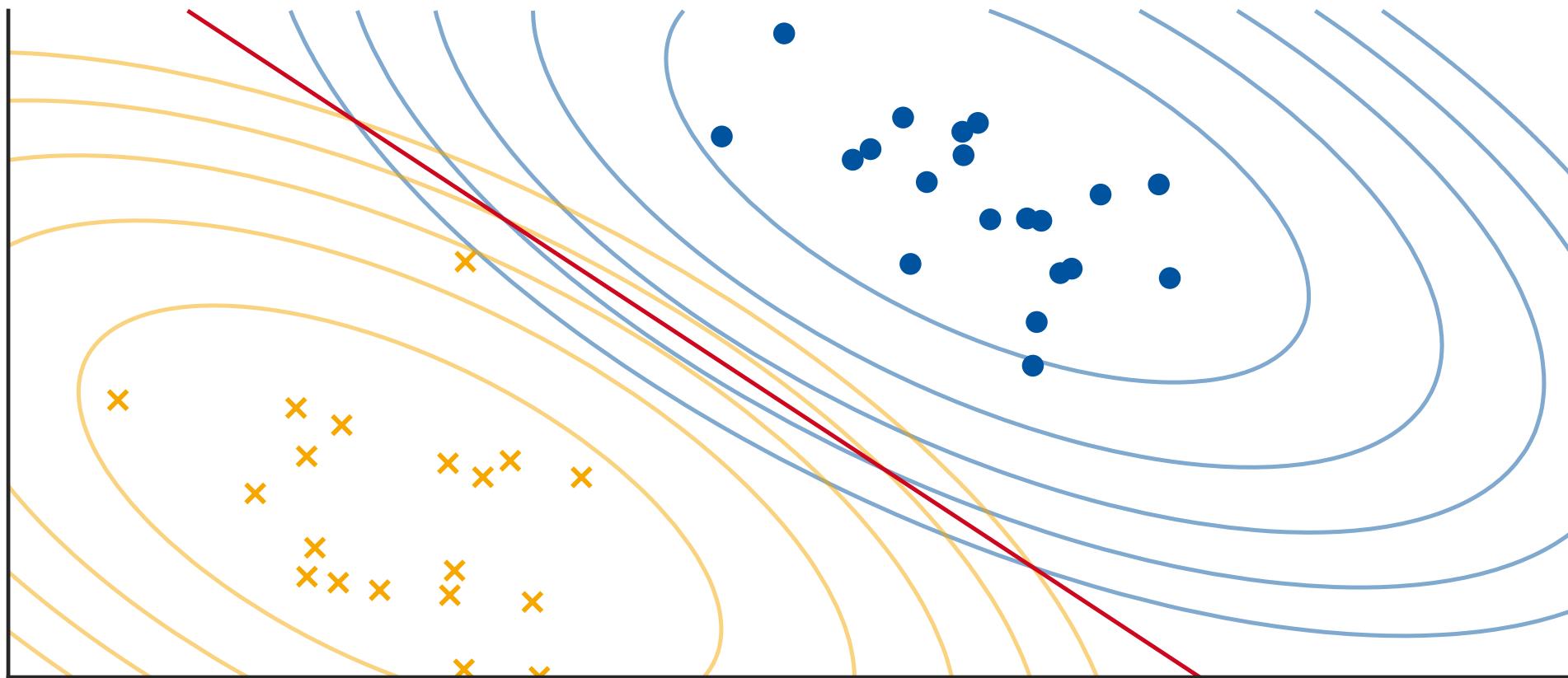
Let's assume the $p(\phi|\mathcal{C}_k)$ are modeled using Gaussians with equal covariances.

Example



Let's assume the $p(\phi|\mathcal{C}_k)$ are modeled using Gaussians with equal covariances.

Example



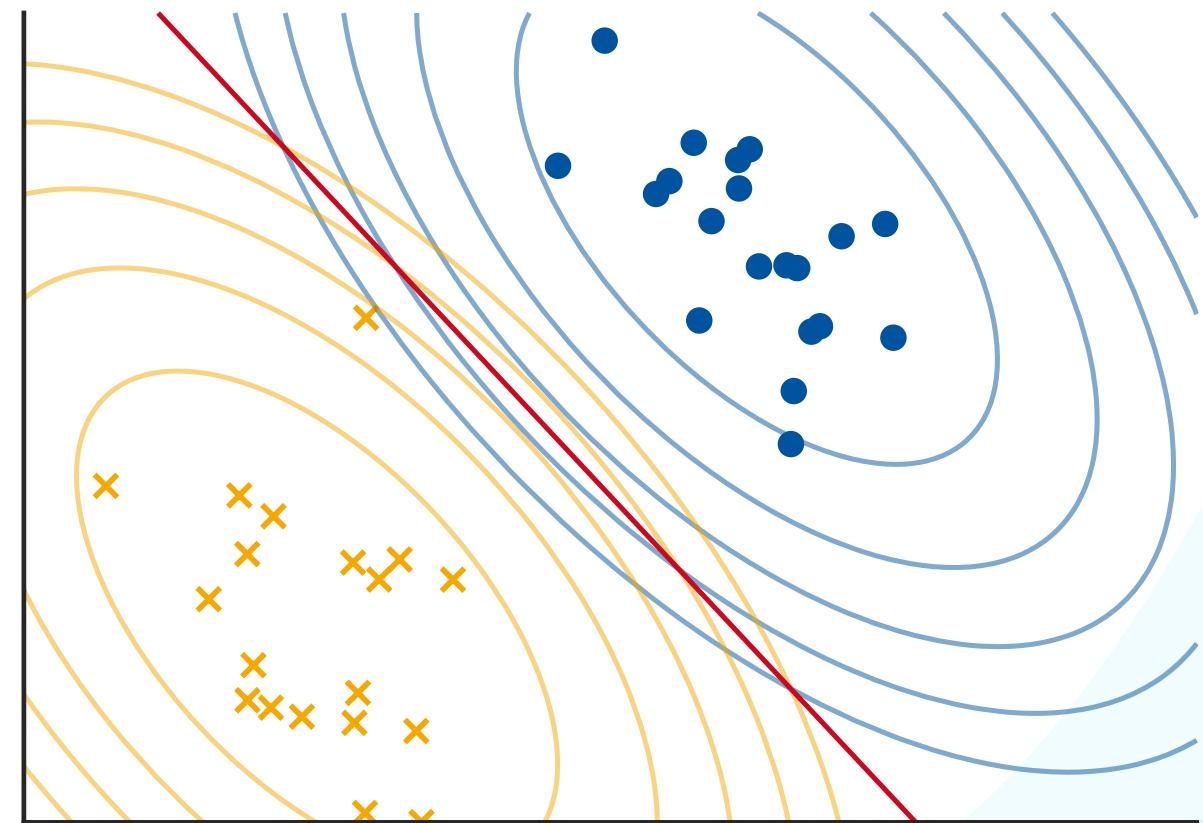
Let's assume the $p(\phi|\mathcal{C}_k)$ are modeled using Gaussians with equal covariances.

⇒ The decision boundary between them will be linear!

Parameter Efficiency

- #Parameters needed for generative models:
 - Assuming an M -dimensional feature space
 - Prior $p(\mathcal{C}_1)$ 1
 - Means μ_1, μ_2 $2M$
 - Covariances Σ $M(M + 1)/2$
- $$\Rightarrow \text{Total} \quad M(M + 5)/2 + 1$$
-
- #Parameters needed for logistic regression:
 - Weights w M

\Rightarrow For large M , logistic regression has clear advantages!



Discussion: Logistic Regression

Advantages

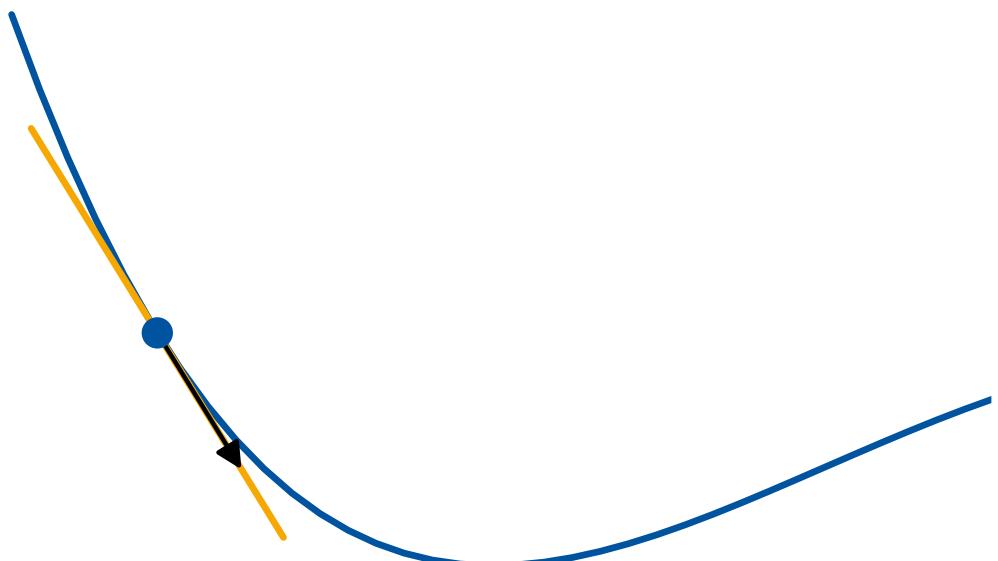
- Nice probabilistic interpretation, directly represents the posterior.
- Requires fewer parameters than modeling the likelihood + prior.
- Cross-Entropy error is convex: unique minimum exists.
- More robust than least-squares.

Limitations

- No closed-form solution, requires iterative optimization approach.

Logistic Regression

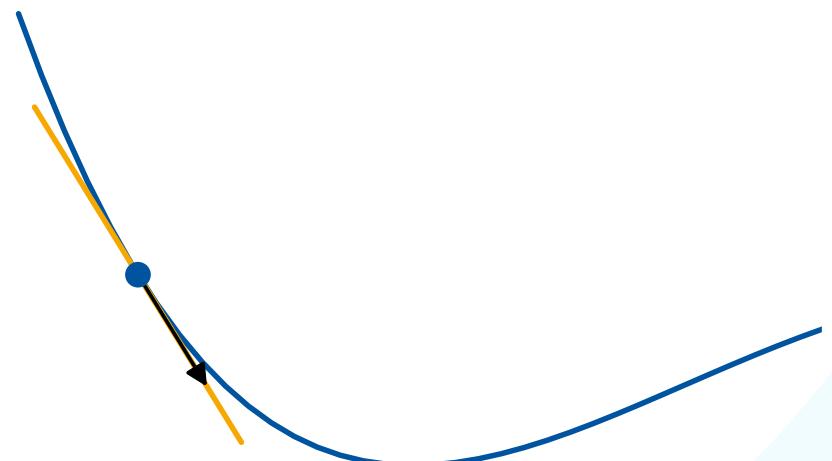
1. Logistic Regression Formulation
2. Motivation and Background
- 3. Iterative Optimization**
4. First-Order Gradient Descent
5. Second-Order Gradient Descent
6. Error Function Analysis



Iterative Optimization

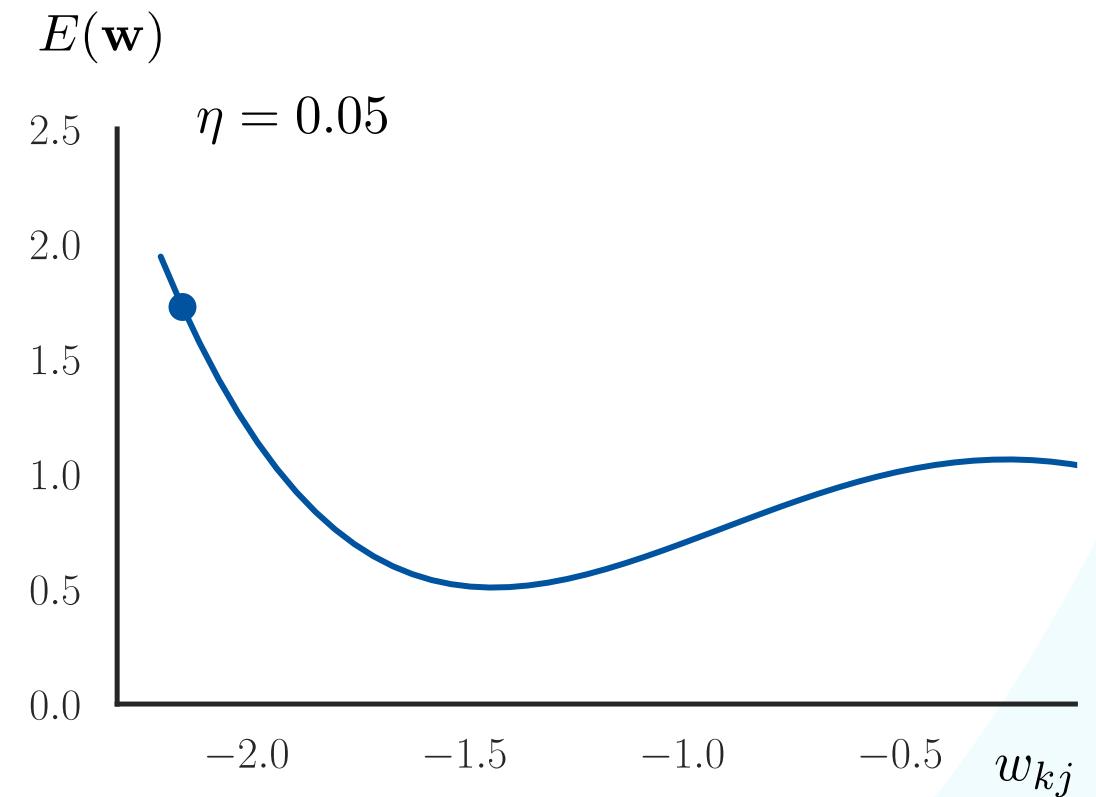
- In general, generalized linear discriminants with nonlinear activation and/or basis functions can no longer be optimized in closed form.
- Instead, we use iterative optimization schemes.
- Here: **Gradient Descent**.
 - Start with initial guess for parameter values.
 - Move towards a minimum of the error function by following the direction of steepest descent.
 - Iterate until convergence

$$y_k(\mathbf{x}) = g \left(\sum_{j=0}^M w_{kj} \phi_j(\mathbf{x}) \right) = g(\mathbf{w}^\top \phi(\mathbf{x}))$$



Idea: Gradient Descent

- Start with an initial guess of parameter values $w_{kj}^{(0)}$.

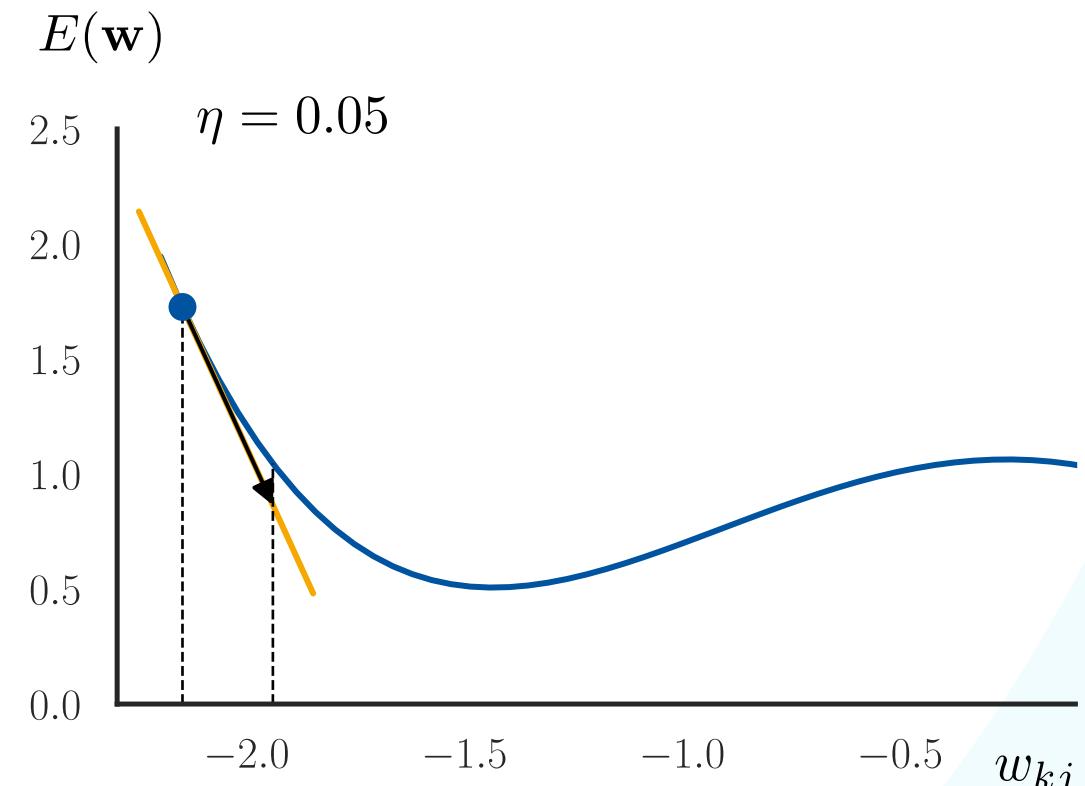


Idea: Gradient Descent

- Start with an initial guess of parameter values $w_{kj}^{(0)}$.
- Follow the gradient to move to a (local) minimum:

$$w_{kj}^{(\tau+1)} = w_{kj}^{(\tau)} - \eta \frac{\partial E(\mathbf{w})}{\partial w_{kj}} \Big|_{\mathbf{w}^{(\tau)}}$$

- η is called the **learning rate**.
- This corresponds to a 1st-order Taylor expansion.
 - I.e., we approximate the error function by its tangent plane around the current point $\mathbf{w}^{(\tau)}$.
- Repeat this procedure for a number of steps.

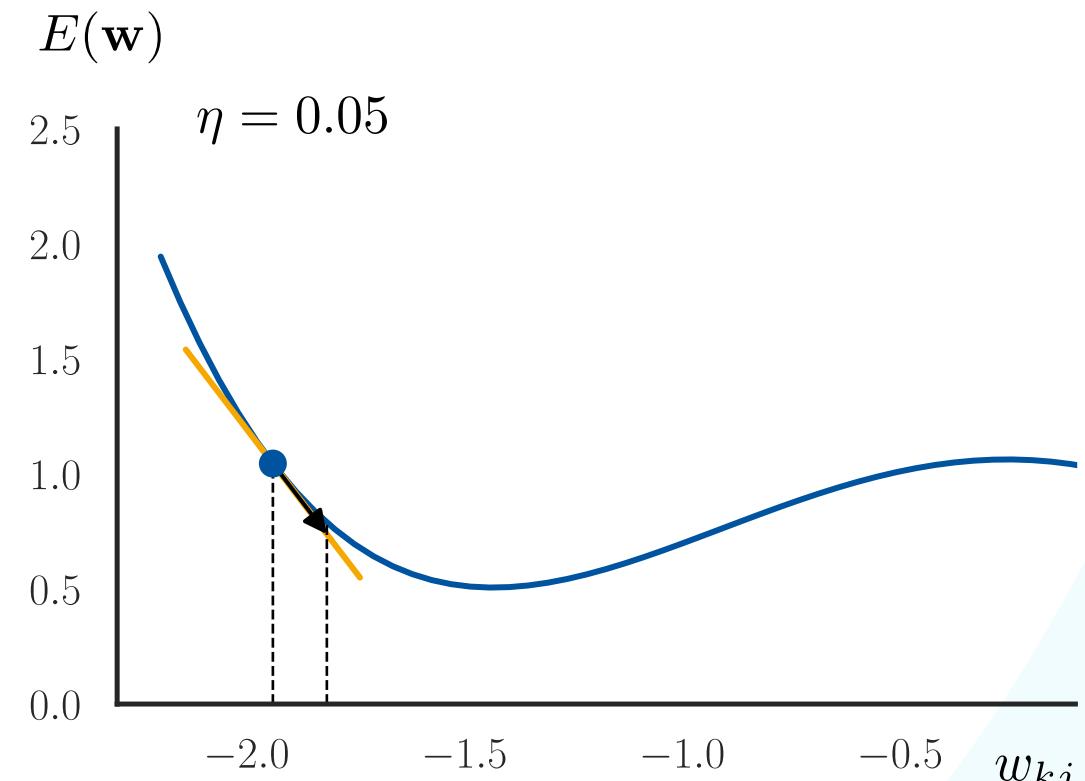


Idea: Gradient Descent

- Start with an initial guess of parameter values $w_{kj}^{(0)}$.
- Follow the gradient to move to a (local) minimum:

$$w_{kj}^{(\tau+1)} = w_{kj}^{(\tau)} - \eta \frac{\partial E(\mathbf{w})}{\partial w_{kj}} \Big|_{\mathbf{w}^{(\tau)}}$$

- η is called the **learning rate**.
- This corresponds to a 1st-order Taylor expansion.
 - I.e., we approximate the error function by its tangent plane around the current point $\mathbf{w}^{(\tau)}$.
- Repeat this procedure for a number of steps.

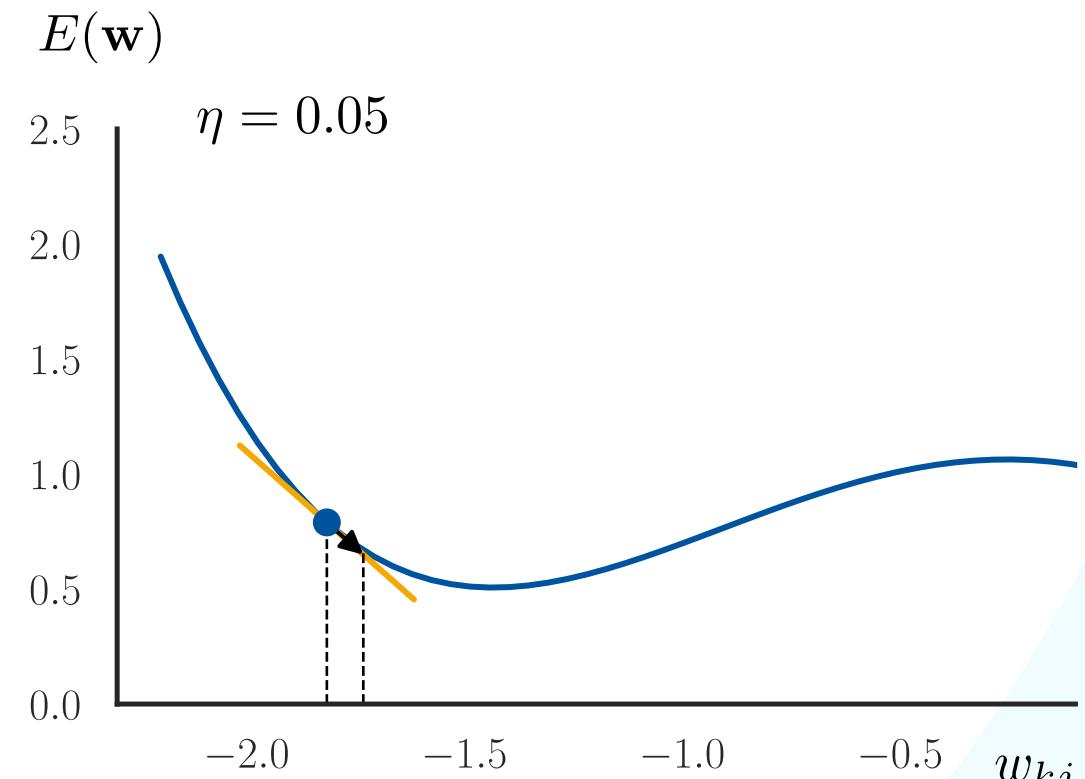


Idea: Gradient Descent

- Start with an initial guess of parameter values $w_{kj}^{(0)}$.
- Follow the gradient to move to a (local) minimum:

$$w_{kj}^{(\tau+1)} = w_{kj}^{(\tau)} - \eta \frac{\partial E(\mathbf{w})}{\partial w_{kj}} \Big|_{\mathbf{w}^{(\tau)}}$$

- η is called the **learning rate**.
- This corresponds to a 1st-order Taylor expansion.
 - I.e., we approximate the error function by its tangent plane around the current point $\mathbf{w}^{(\tau)}$.
- Repeat this procedure for a number of steps.

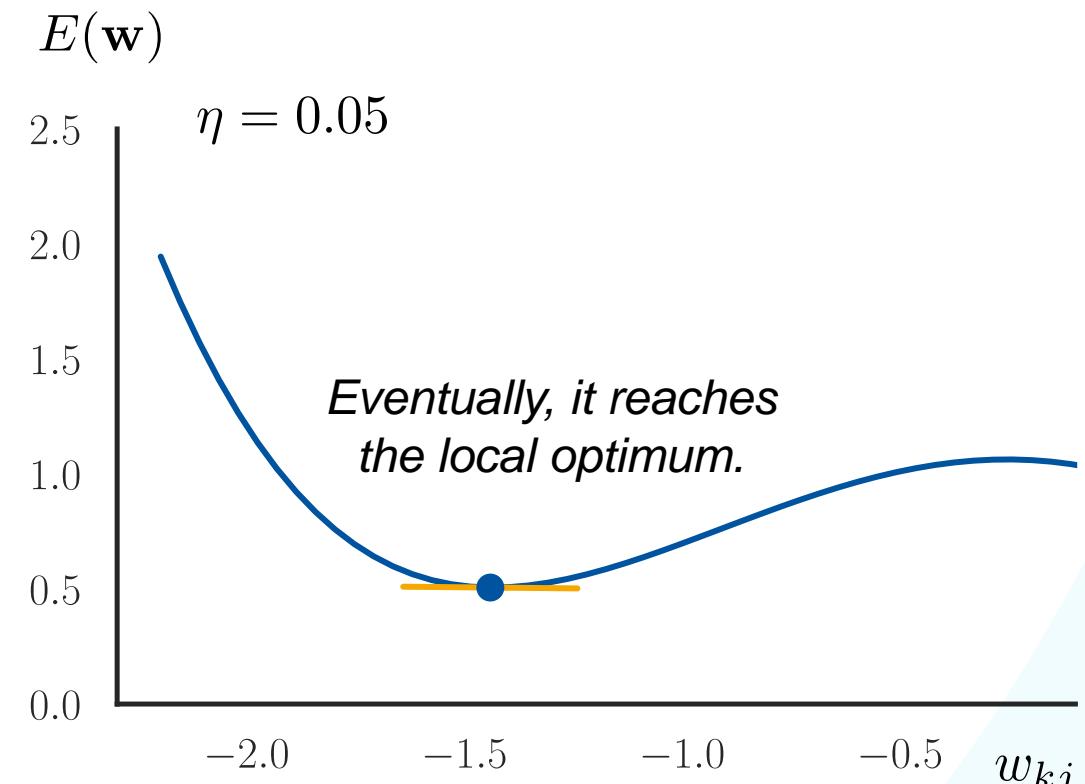


Idea: Gradient Descent

- Start with an initial guess of parameter values $w_{kj}^{(0)}$.
- Follow the gradient to move to a (local) minimum:

$$w_{kj}^{(\tau+1)} = w_{kj}^{(\tau)} - \eta \frac{\partial E(\mathbf{w})}{\partial w_{kj}} \Big|_{\mathbf{w}^{(\tau)}}$$

- η is called the **learning rate**.
- This corresponds to a 1st-order Taylor expansion.
 - I.e., we approximate the error function by its tangent plane around the current point $\mathbf{w}^{(\tau)}$.
- Repeat this procedure for a number of steps.



Discussion: Gradient Descent

Advantages

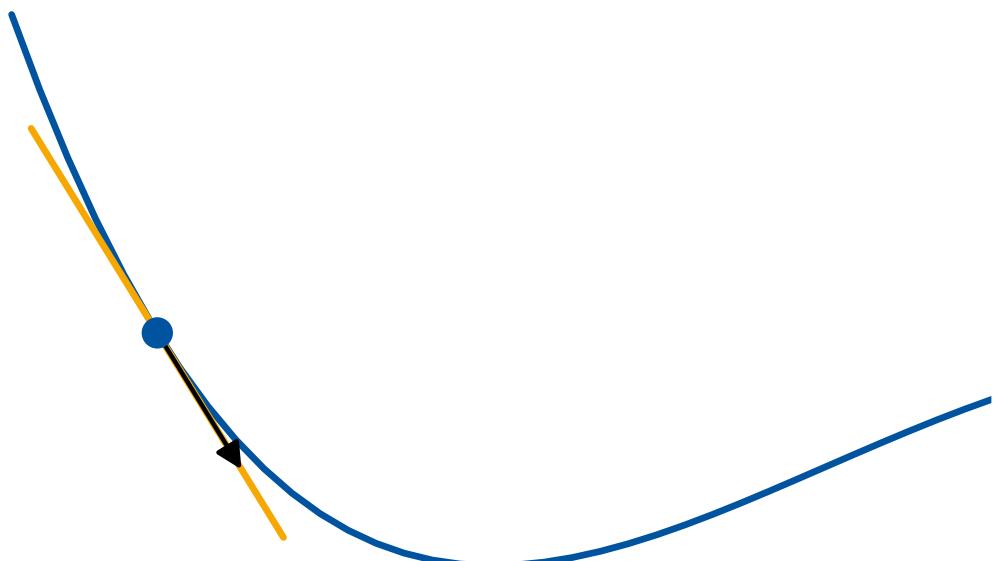
- Simple approach for iterative optimization.
- Approximates the error function by its tangent plane around the current point in order to find the direction of steepest descent.

Limitations

- Local optimization. Unless the error function is convex, will only converge to a local optimum.
- Relatively slow convergence (can be improved by second-order approaches).
- In practice, finding a good step size ([learning rate](#)) is important for fast convergence.

Logistic Regression

1. Logistic Regression Formulation
2. Motivation and Background
3. Iterative Optimization
4. **First-Order Gradient Descent**
5. Second-Order Gradient Descent
6. Error Function Analysis



First-order Optimization

- Logistic regression uses the **binary cross-entropy error**:

$$E(\mathbf{w}) = - \sum_{n=1}^N (t_n \ln y(\mathbf{x}_n; \mathbf{w}) + (1 - t_n) \ln(1 - y(\mathbf{x}_n; \mathbf{w})))$$

- Properties
 - Convex function, so it has a unique minimum
 - But no closed-form solution
- We need to use iterative methods for optimization
 - Let's try (first-order) gradient descent:

$$\mathbf{w}^{(\tau+1)} = \mathbf{w}^{(\tau)} - \eta \nabla E(\mathbf{w})$$

Gradient of the Cross-Entropy Error

$$E(\mathbf{w}) = - \sum_{n=1}^N (t_n \ln y_n + (1 - t_n) \ln(1 - y_n))$$

$$\begin{aligned}\nabla E(\mathbf{w}) &= - \sum_{n=1}^N \left(t_n \frac{\partial}{\partial \mathbf{w}} y_n + (1 - t_n) \frac{\partial}{\partial \mathbf{w}} (1 - y_n) \right) \\ &= - \sum_{n=1}^N \left(t_n \frac{y_n(1 - y_n)}{y_n} \phi_n + (1 - t_n) \frac{y_n(1 - y_n)}{(1 - y_n)} \phi_n \right) \\ &= - \sum_{n=1}^N ((t_n - t_n y_n - y_n + t_n y_n) \phi_n) \\ &= \sum_{n=1}^N (y_n - t_n) \phi_n\end{aligned}$$

$$y_n = y(\mathbf{x}_n; \mathbf{w})$$

$$\begin{aligned}\sigma'(a) &= \sigma(a)(1 - \sigma(a)) \\ \frac{\partial y_n}{\partial \mathbf{w}} &= y_n(1 - y_n) \phi_n\end{aligned}$$

$$\phi_n = \phi(\mathbf{x}_n)$$

- The gradient for logistic regression is

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

- We can plug this into gradient descent:

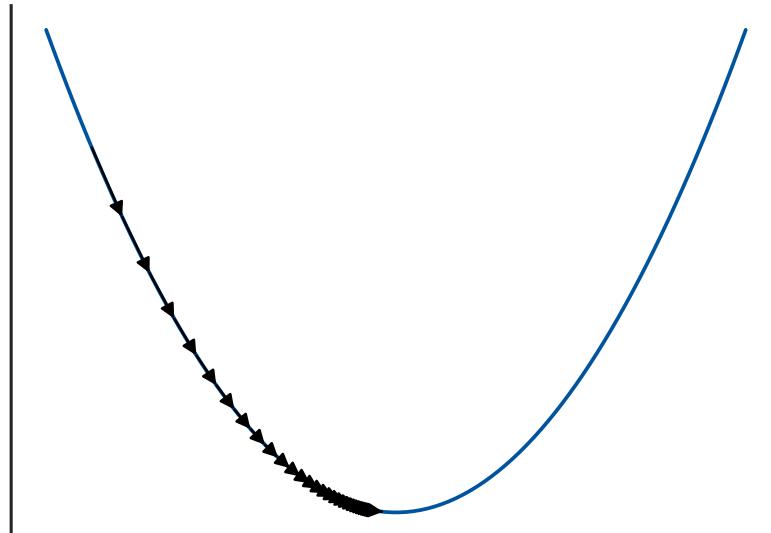
$$\begin{aligned}\mathbf{w}^{(\tau+1)} &= \mathbf{w}^{(\tau)} - \eta \nabla E(\mathbf{w}) \\ &= \mathbf{w}^{(\tau)} - \eta \sum_{n=1}^N (y_n - t_n) \phi_n\end{aligned}$$

*How should we choose
the learning rate?*

- This update rule is known as the **Delta rule** (= LMS rule)
 - Simply feed back the input data points, weighted by the classification error.*

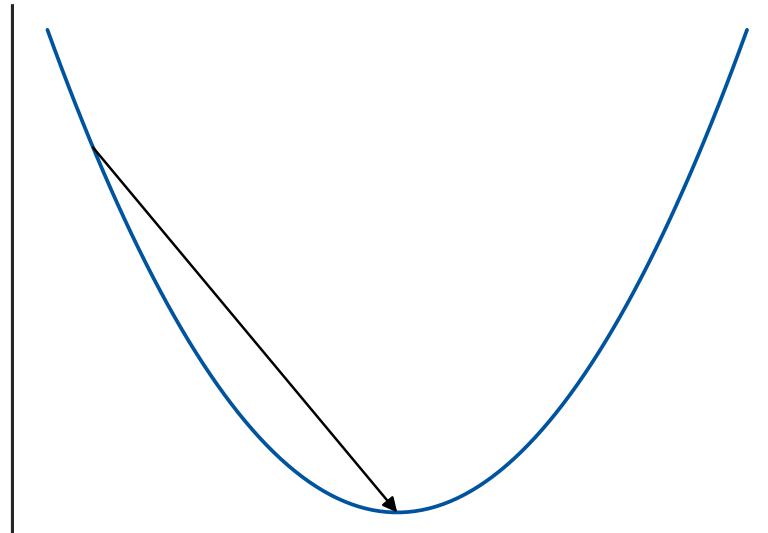
Effects of the learning rate

η too small



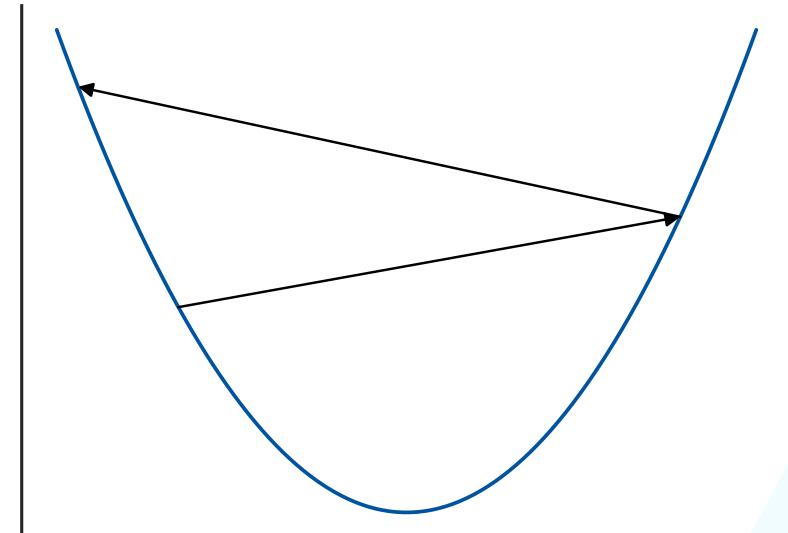
Convergence is slow

η_{opt}



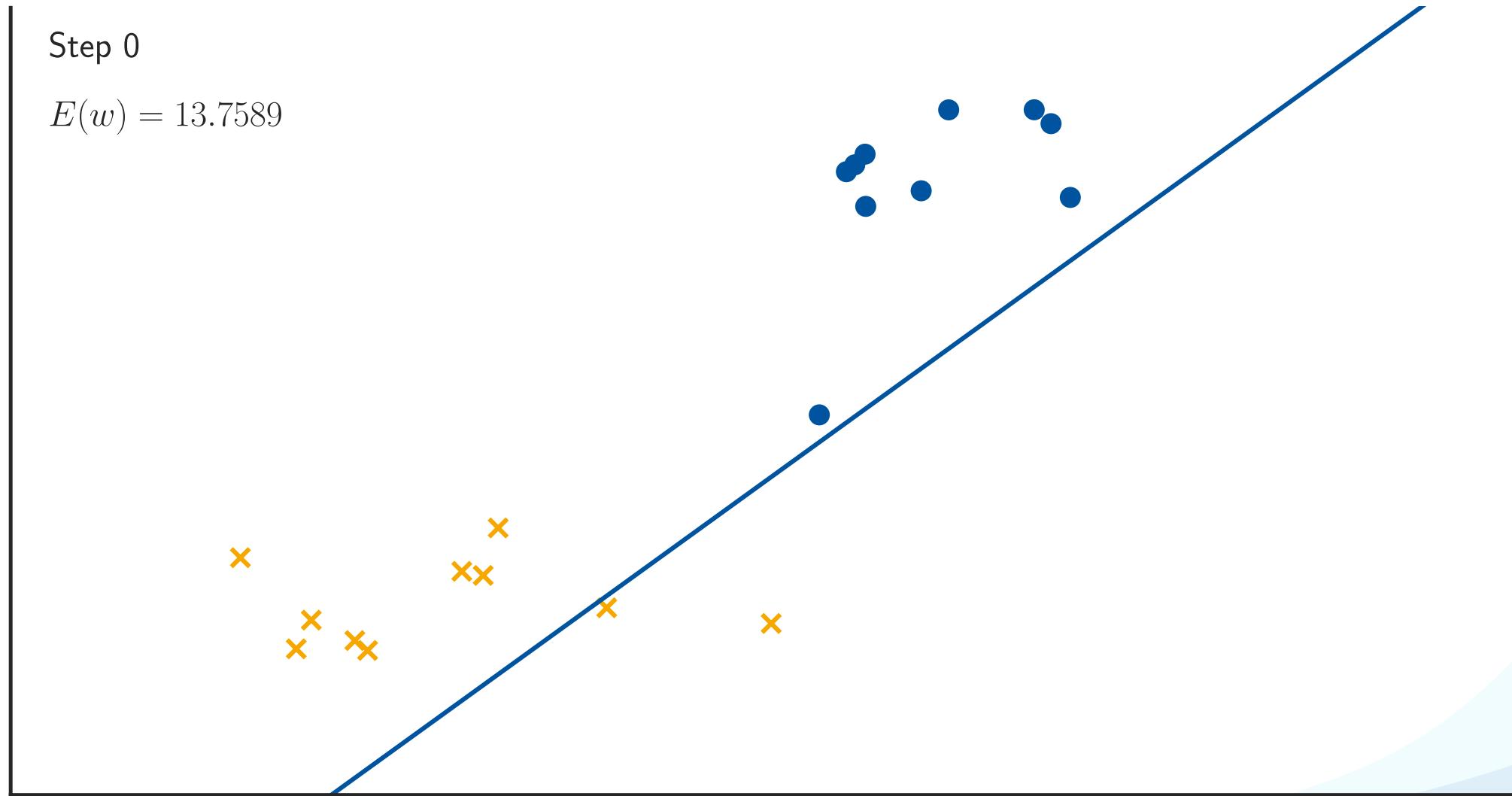
Converges ideally in
a single step

η too large



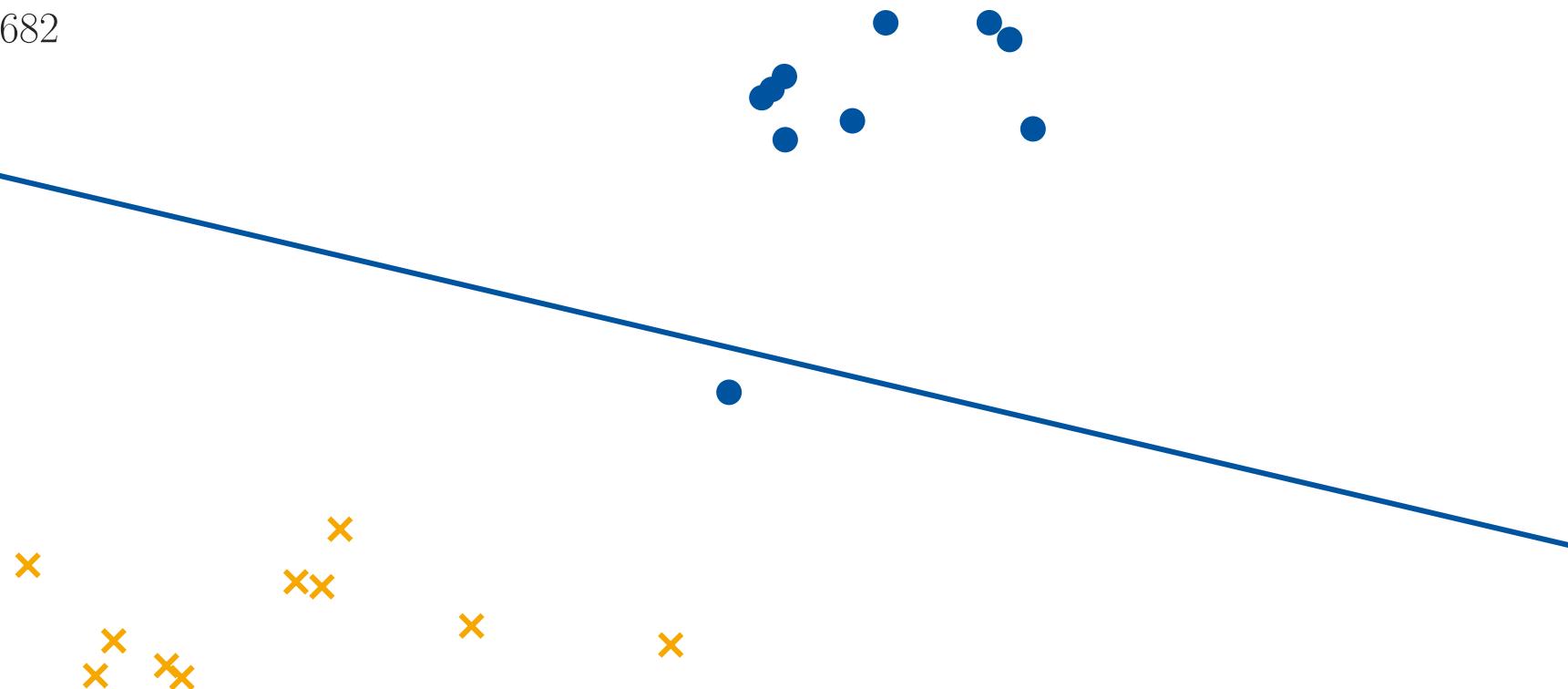
Might not converge

Example: Logistic Regression with Gradient Descent



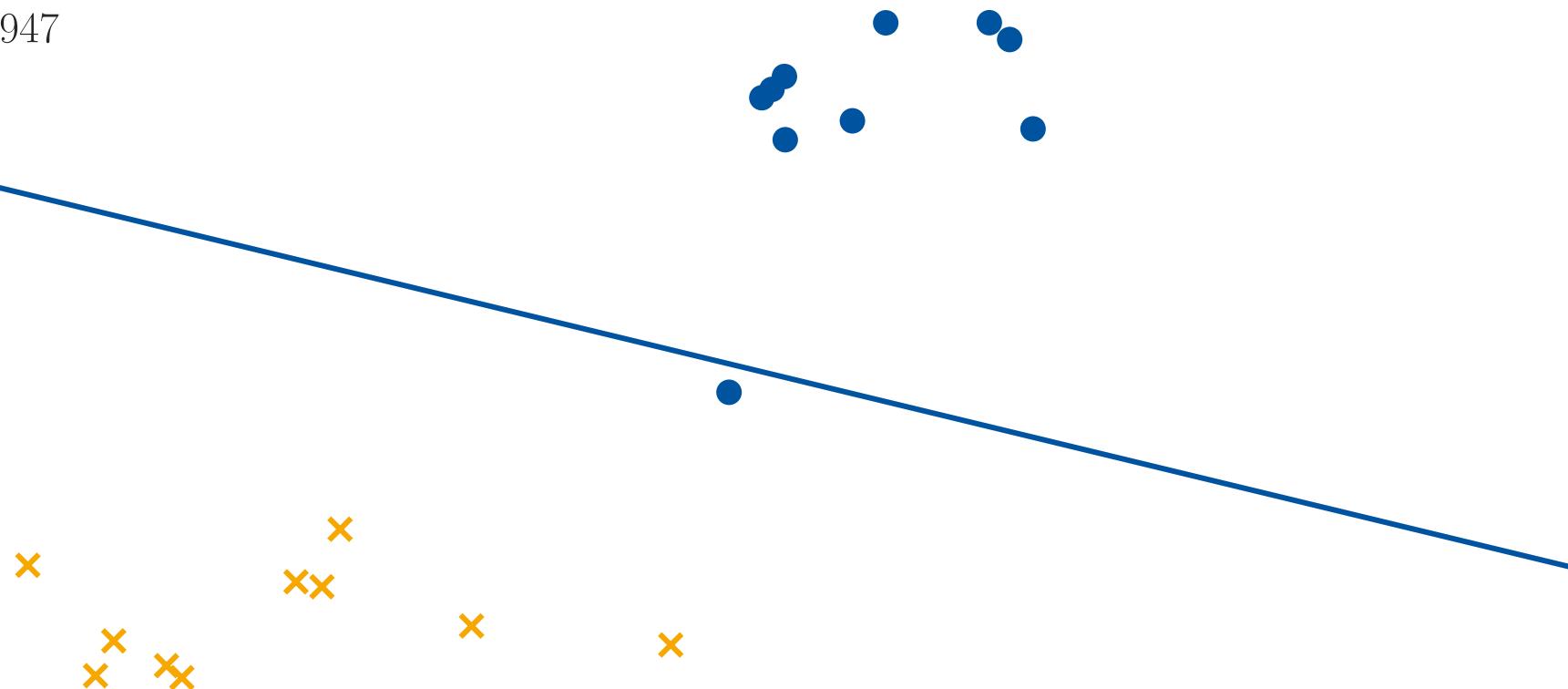
Step 1

$$E(w) = 1.3682$$



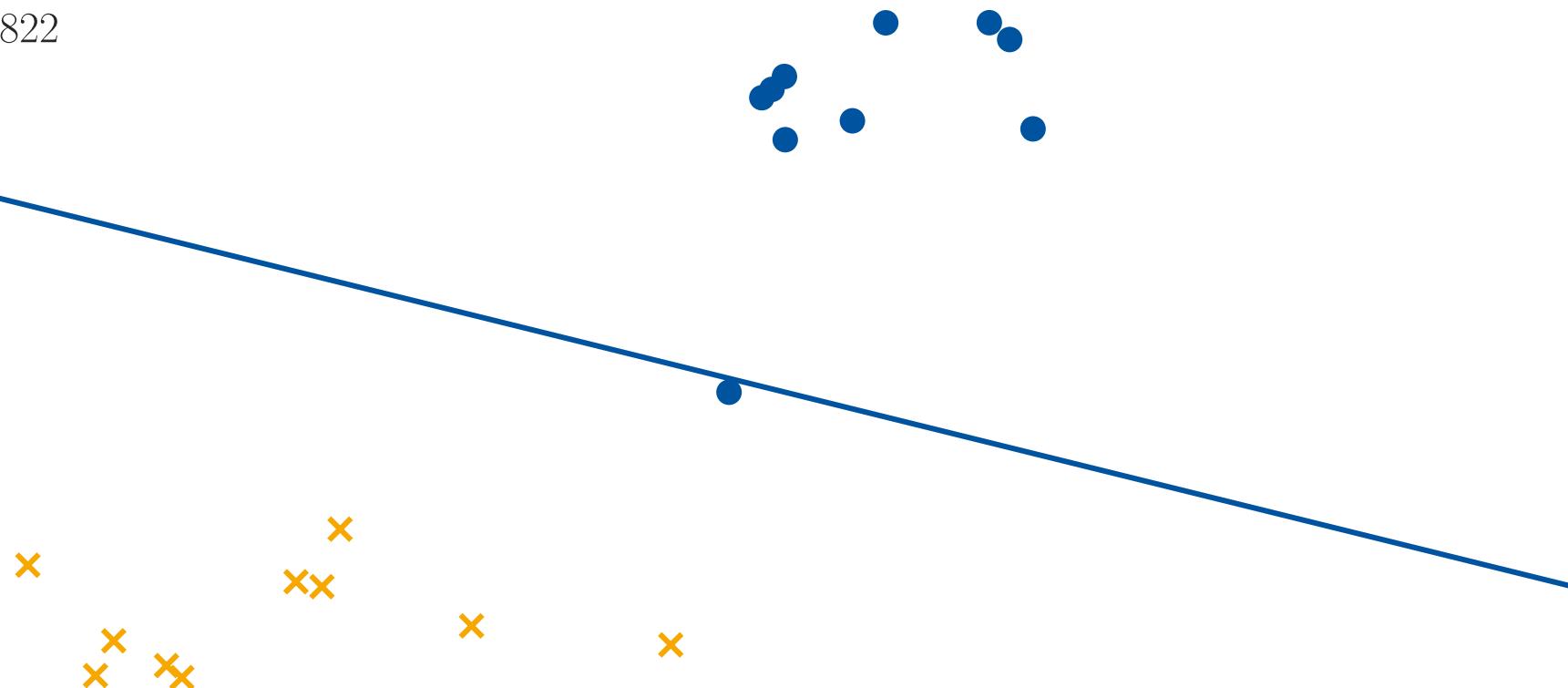
Step 2

$$E(w) = 1.0947$$



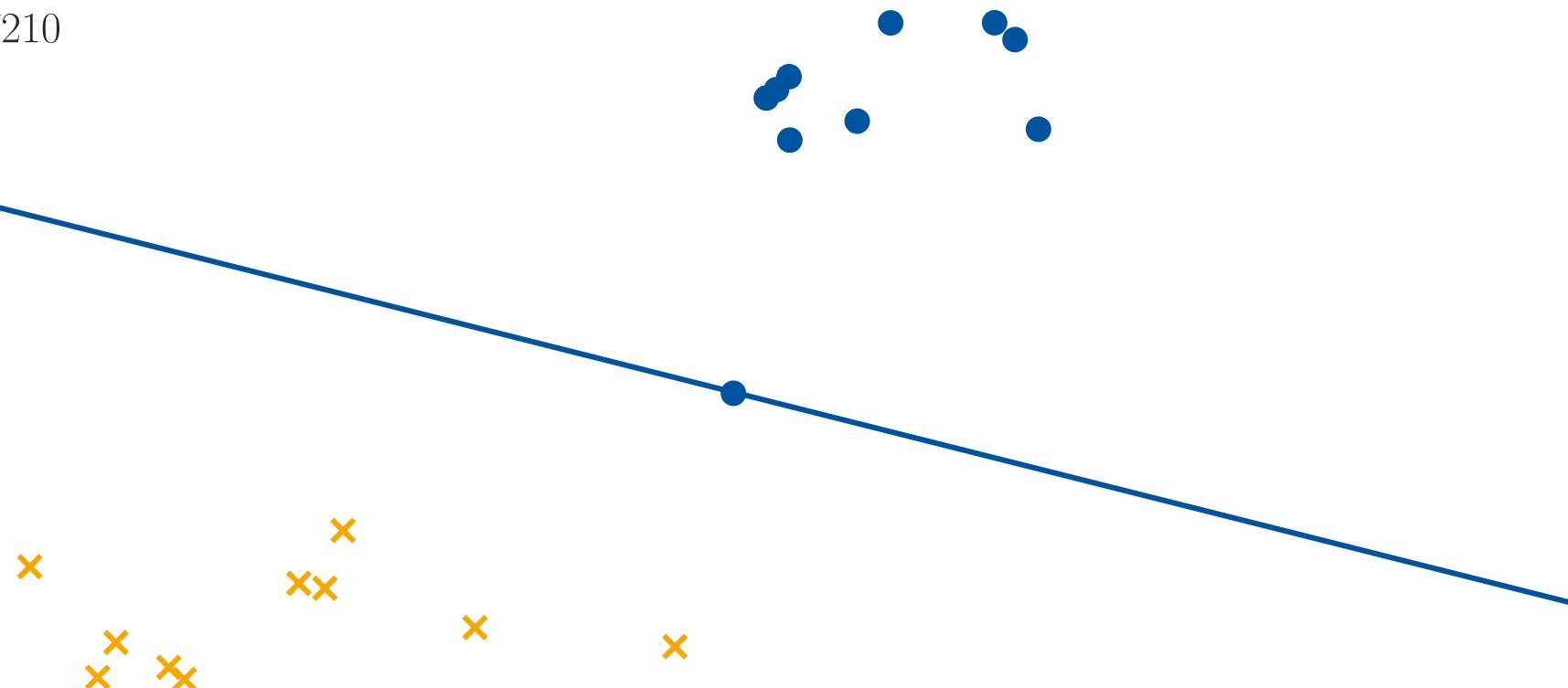
Step 3

$$E(w) = 0.8822$$



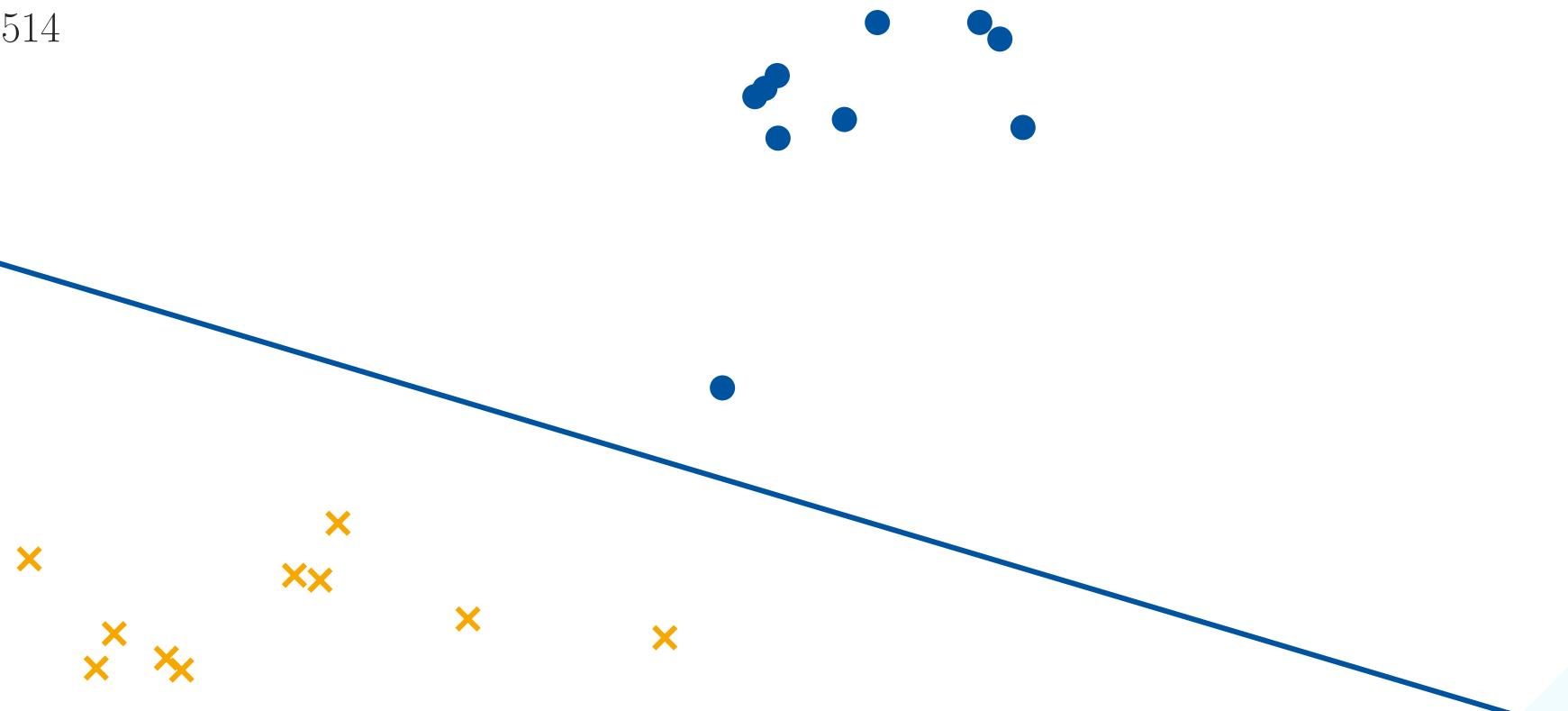
Step 4

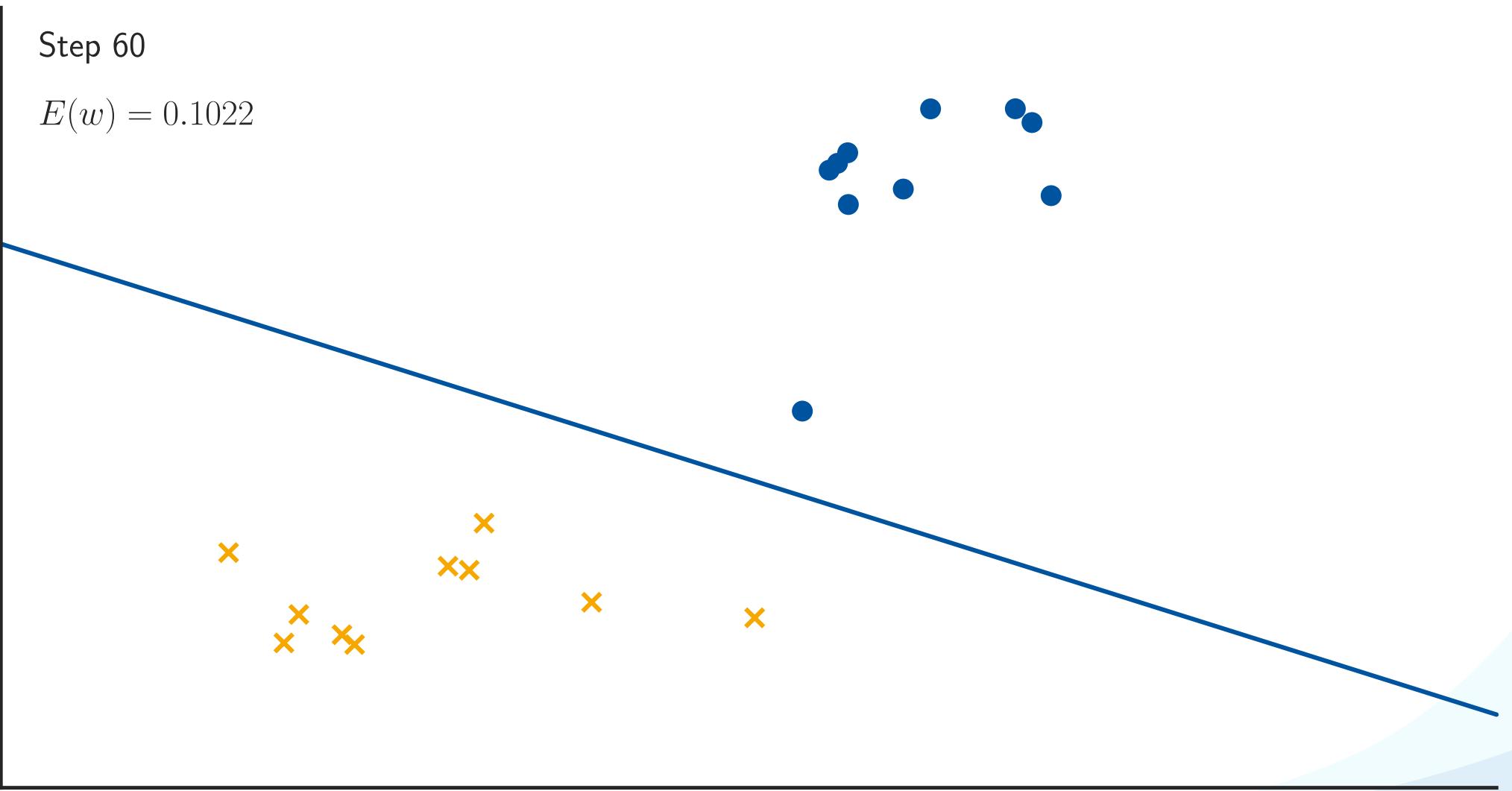
$$E(w) = 0.7210$$



Step 30

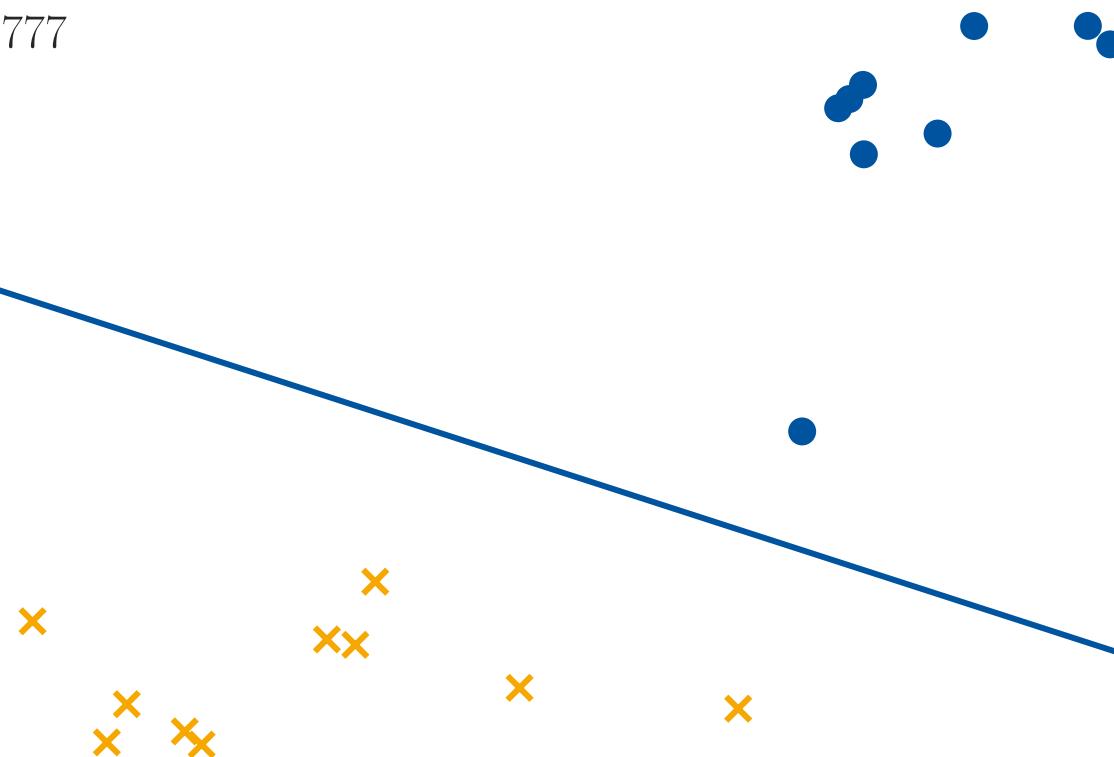
$$E(w) = 0.1514$$





Step 90

$$E(w) = 0.0777$$



Discussion: Logistic Regression with Gradient Descent

Advantages

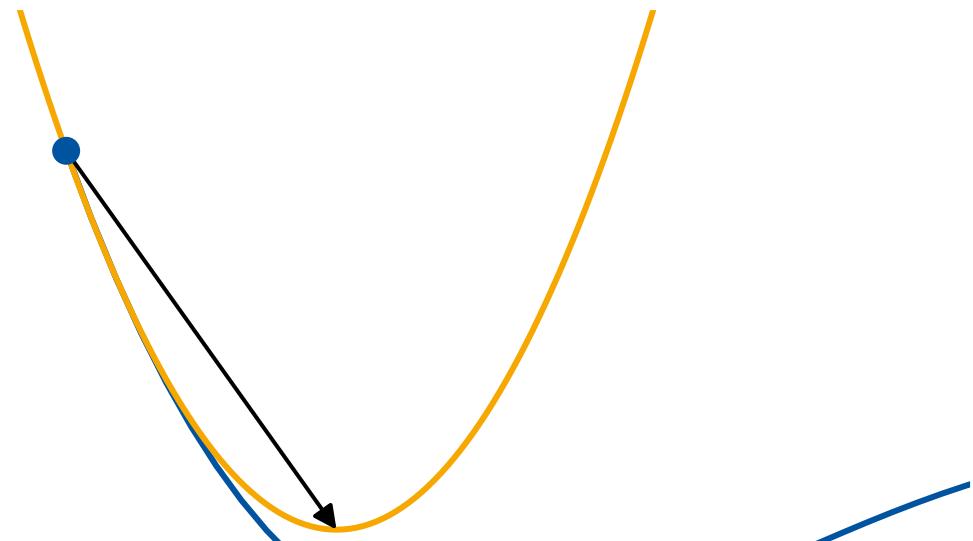
- Simple iterative optimization scheme with a familiar update rule ([Delta rule](#)).

Limitations

- Slow convergence
- Need to choose a suitable learning rate.

Logistic Regression

1. Logistic Regression Formulation
2. Motivation and Background
3. Iterative Optimization
4. First-Order Gradient Descent
- 5. Second-Order Gradient Descent**
6. Error Function Analysis



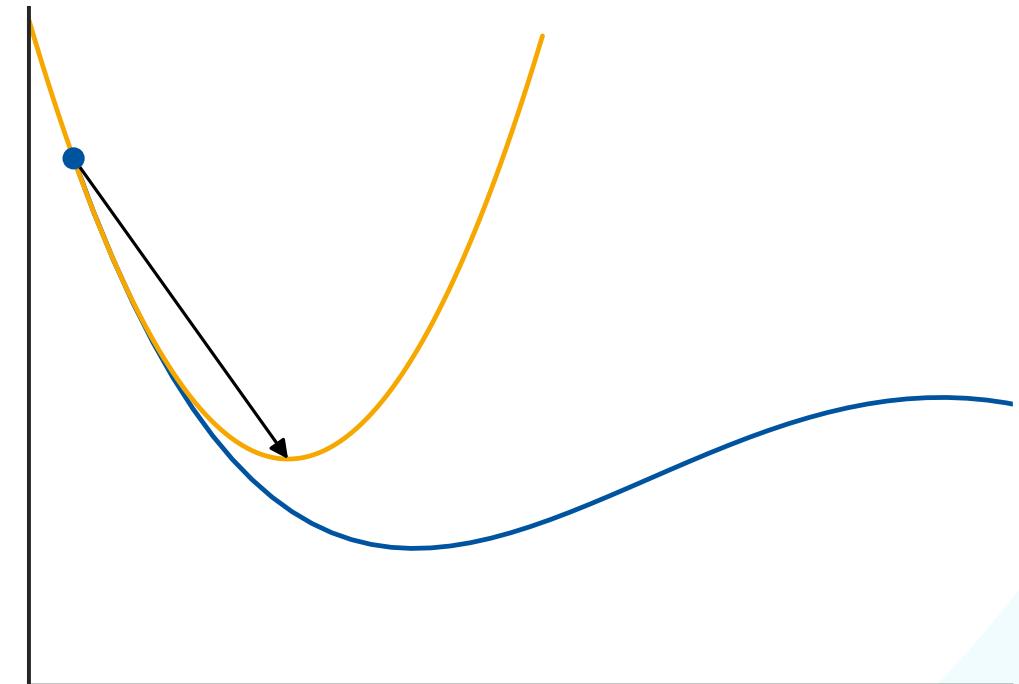
Second-Order Optimization

- So far, we have optimized the cross-entropy error with gradient descent:

$$\mathbf{w}^{(\tau+1)} = \mathbf{w}^{(\tau)} - \eta \nabla E(\mathbf{w})$$

- This is a first-order approximation, and it heavily depends on the learning rate η .

- Instead, we can apply a second-order optimization scheme that converges faster and is independent of the learning rate.



Newton-Raphson Gradient Descent

- Second-order Newton-Raphson update scheme:

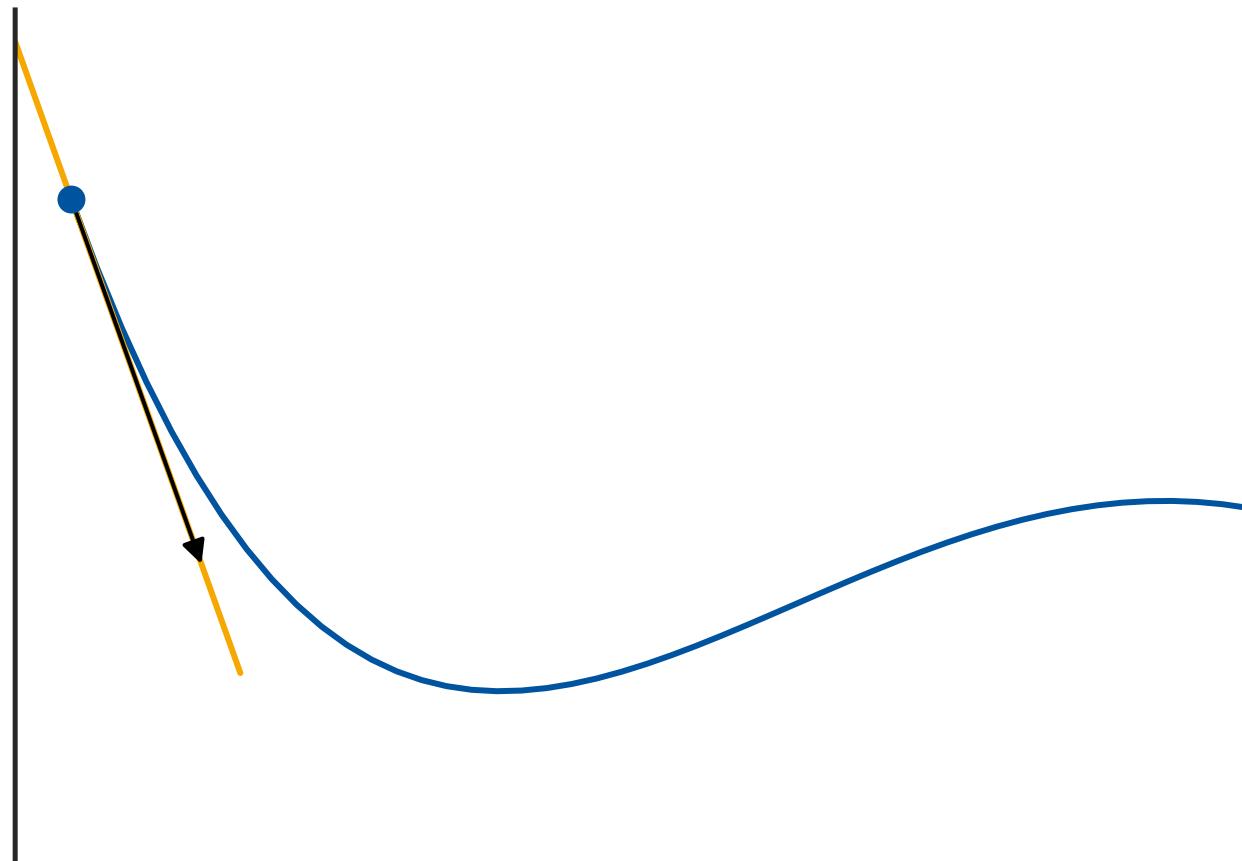
$$\mathbf{w}^{(\tau+1)} = \mathbf{w}^{(\tau)} - \mathbf{H}^{-1} \nabla E(\mathbf{w})$$

- Here, $\mathbf{H} = \nabla \nabla E(\mathbf{w})$ is the Hessian matrix, i.e., the matrix of second derivatives:

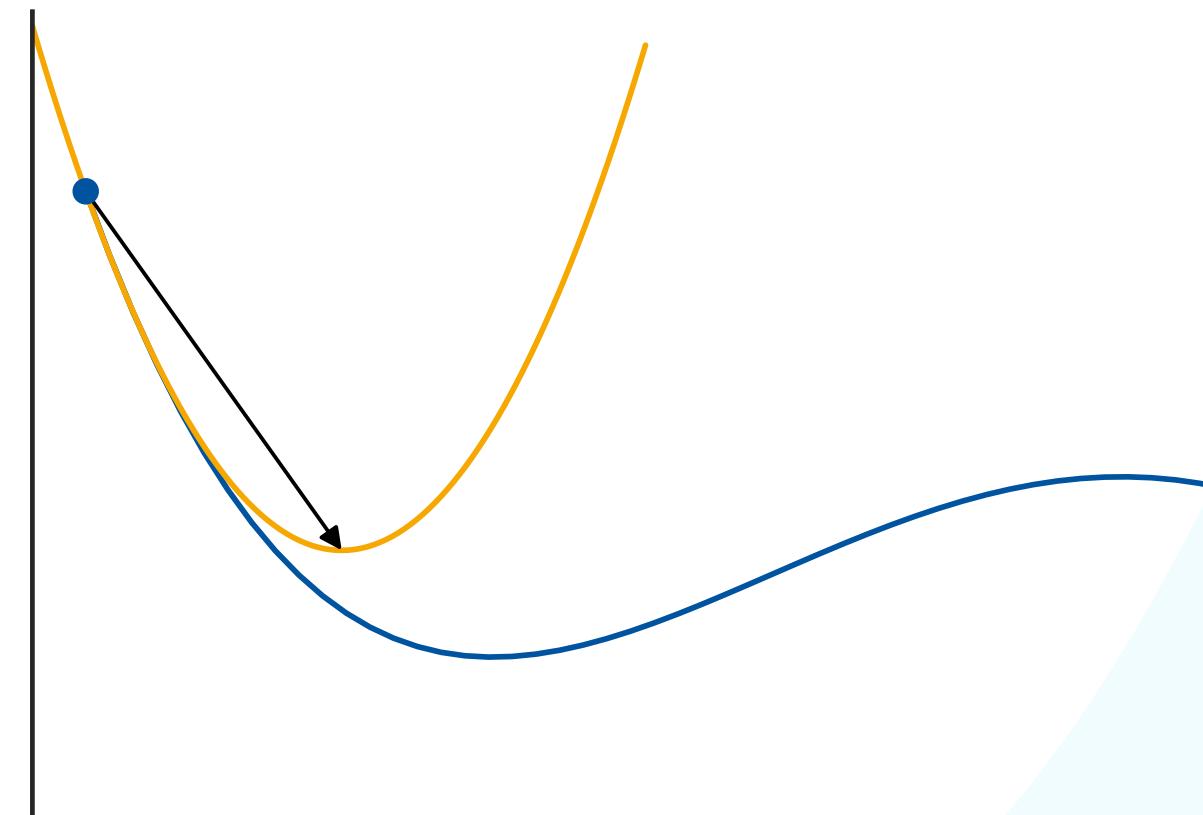
$$\mathbf{H}_{ij} = \frac{\partial^2 E(\mathbf{w})}{\partial w_i \partial w_j}$$

- Properties
 - Local quadratic approximation
 - Much faster convergence by taking into account the curvature of the error function.

Intuition

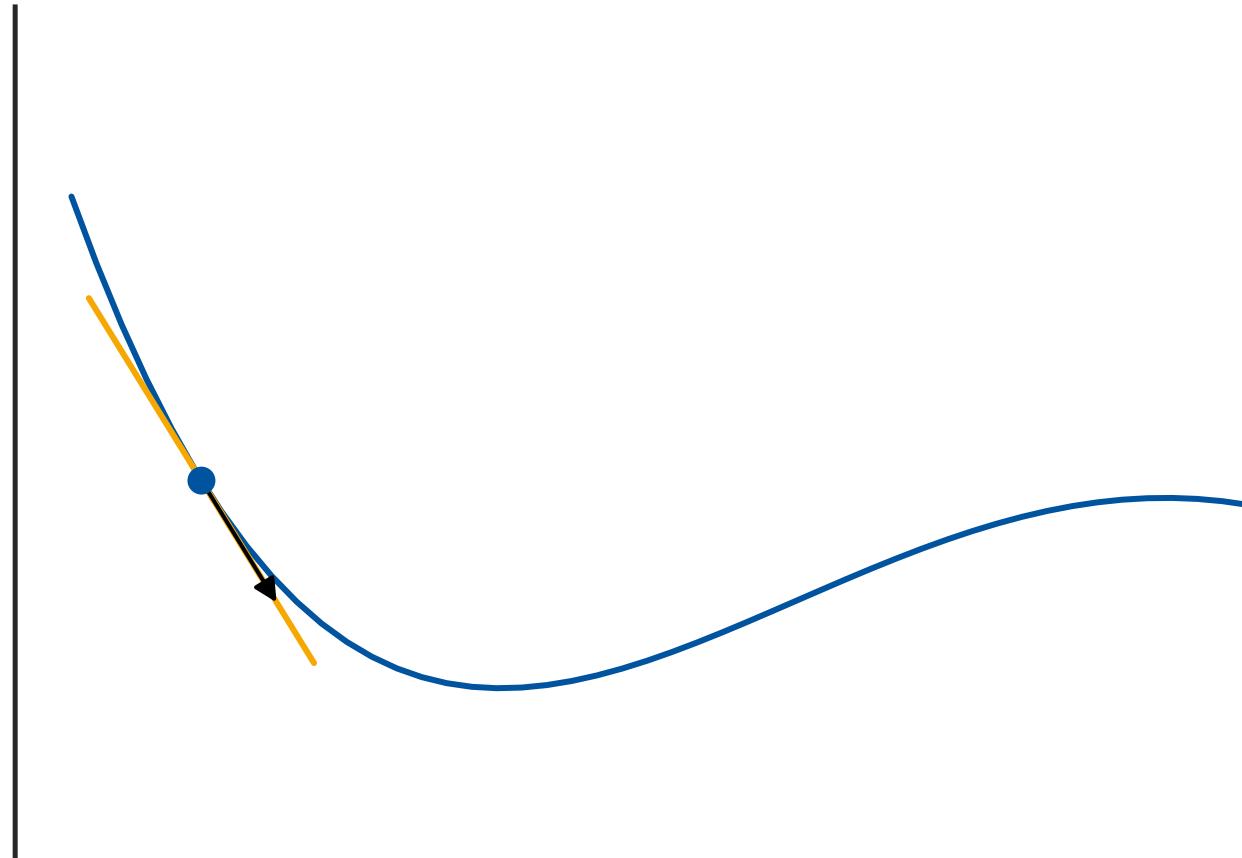


First-Order
 $\eta = 0.005$

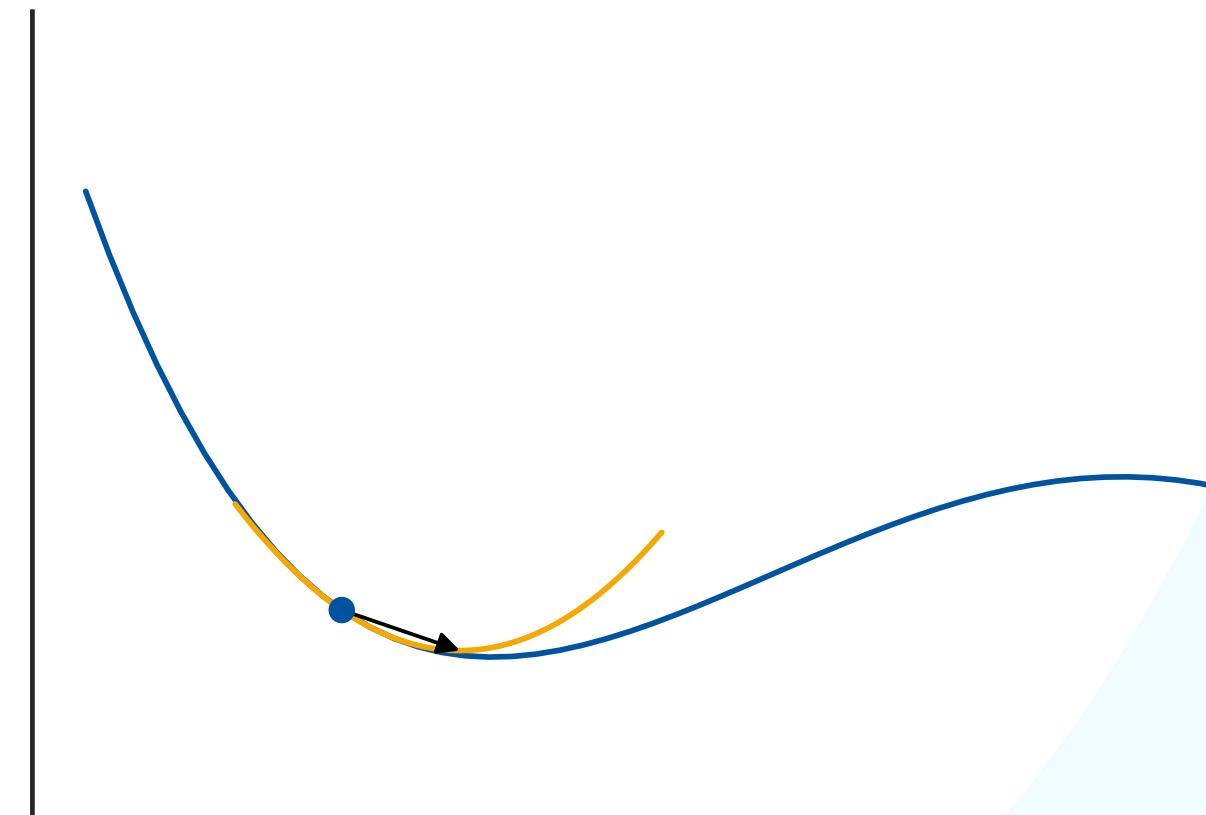


Second-Order

Intuition

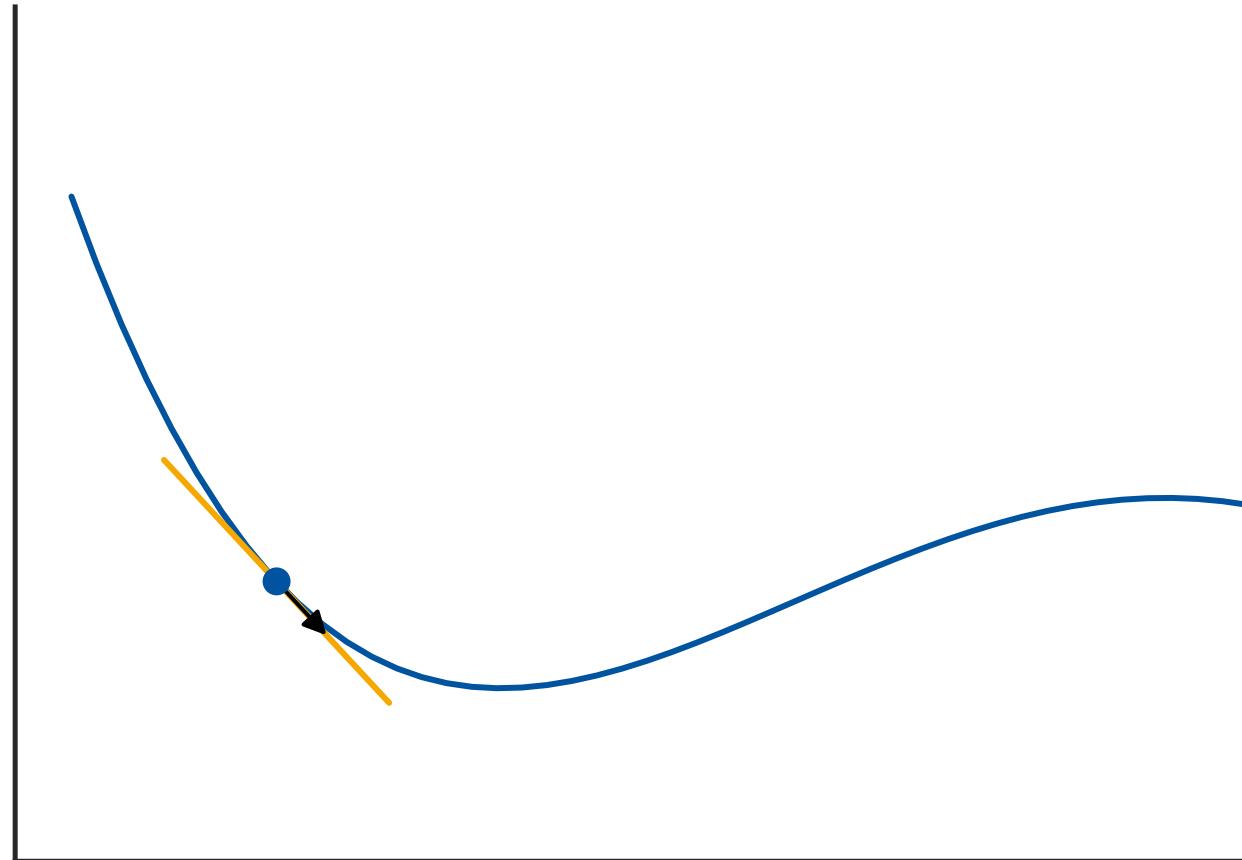


First-Order
 $\eta = 0.005$

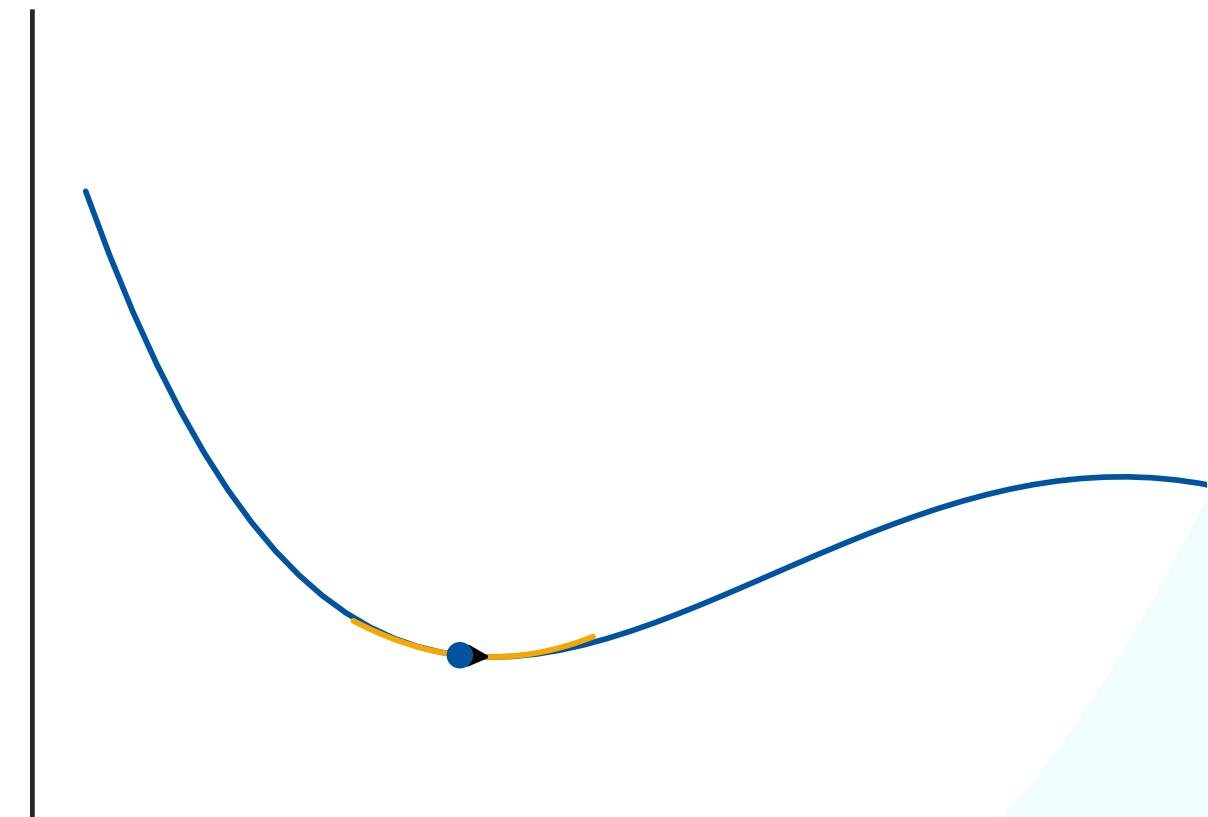


Second-Order

Intuition

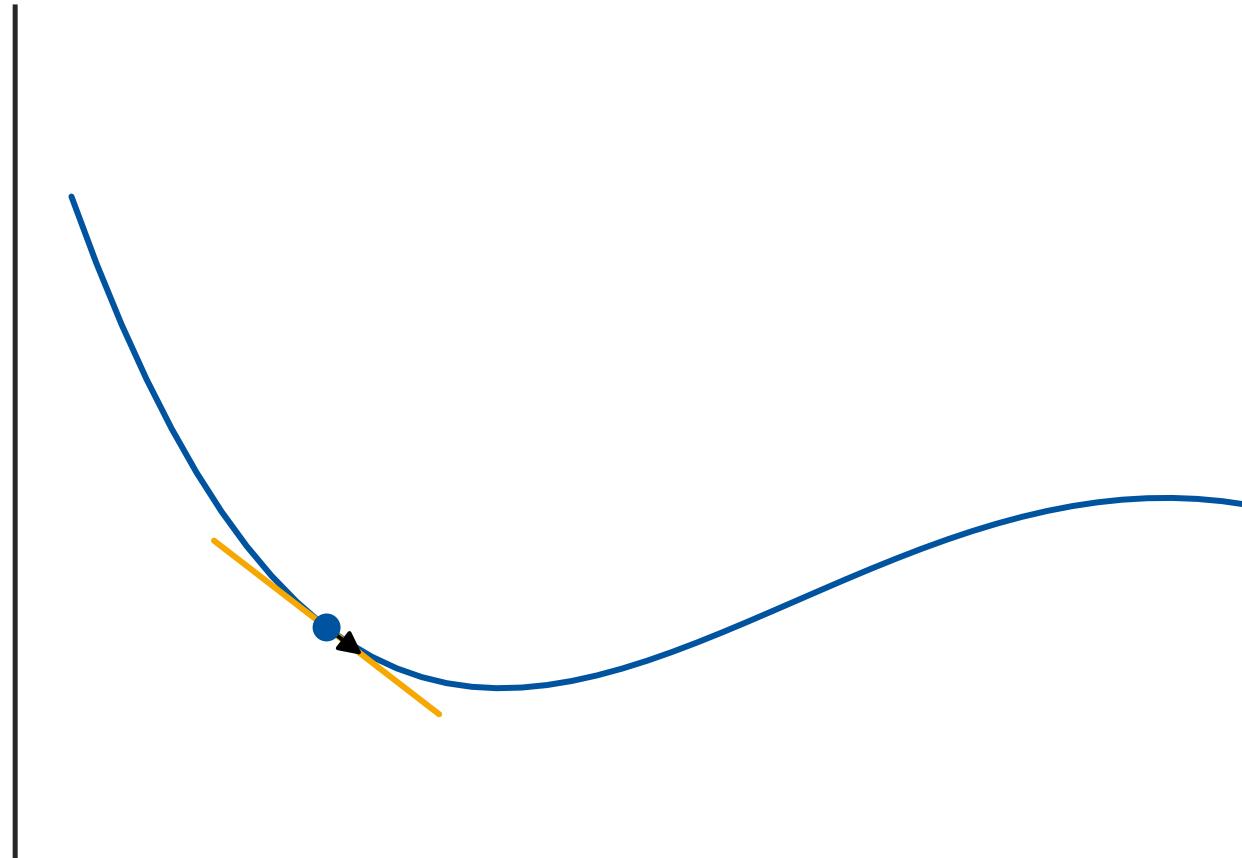


First-Order
 $\eta = 0.005$

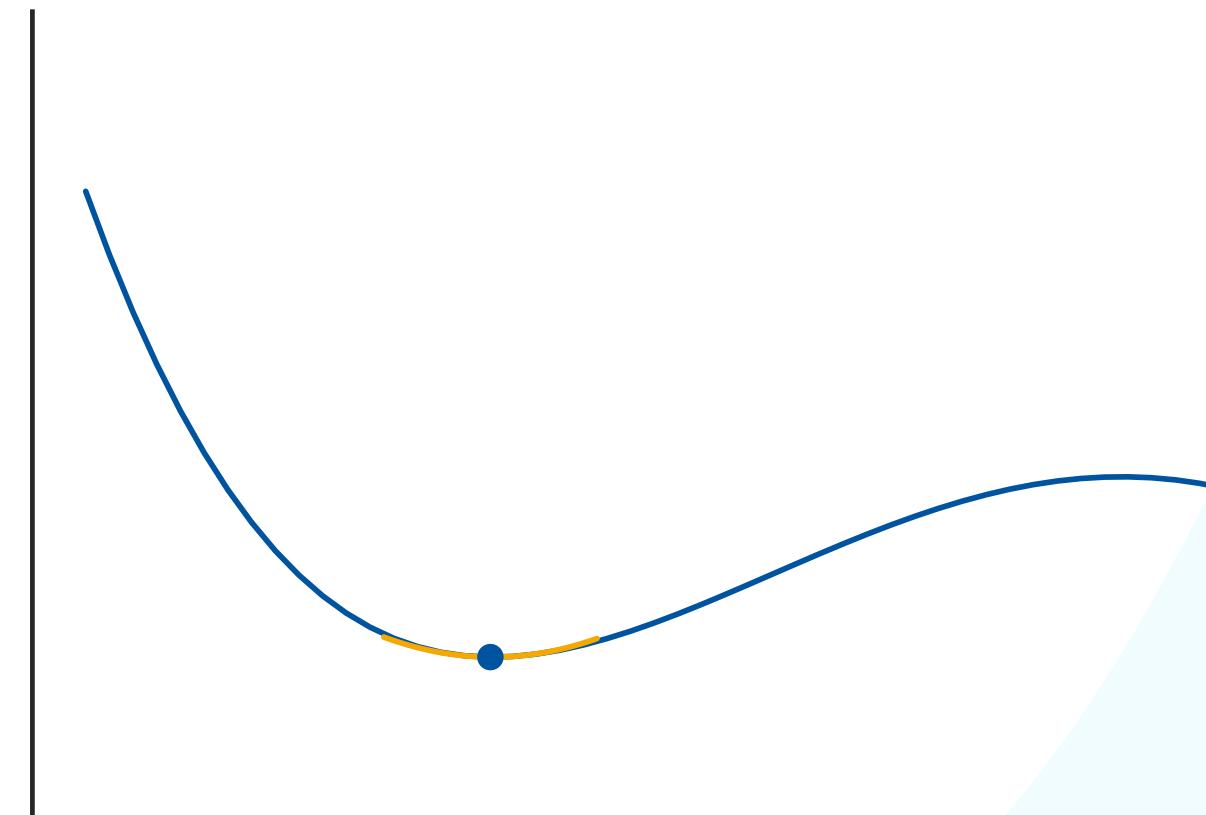


Second-Order

Intuition



First-Order
 $\eta = 0.005$



Second-Order

Intuition

*First-Order needs
another 16 steps...*

First-Order
 $\eta = 0.005$

Second-Order

Newton-Raphson for Least-Squares

- First, we apply it to least-squares:

$$E(\mathbf{w}) = \frac{1}{2} \sum_{n=1}^N (\mathbf{w}^\top \boldsymbol{\phi}_n - t_n)^2$$

$$\nabla E(\mathbf{w}) = \sum_{n=1}^N (\mathbf{w}^\top \boldsymbol{\phi}_n - t_n) \boldsymbol{\phi}_n = \boldsymbol{\Phi}^\top \boldsymbol{\Phi} \mathbf{w} - \boldsymbol{\Phi}^\top \mathbf{t}$$

$$\mathbf{H} = \nabla \nabla E(\mathbf{w}) = \sum_{n=1}^N \boldsymbol{\phi}_n \boldsymbol{\phi}_n^\top = \boldsymbol{\Phi}^\top \boldsymbol{\Phi}$$

$$\boldsymbol{\Phi} = \begin{pmatrix} \boldsymbol{\phi}_1^\top \\ \vdots \\ \boldsymbol{\phi}_N^\top \end{pmatrix}$$

- Resulting update scheme ([normal equations](#)):

$$\begin{aligned} \mathbf{w}^{(\tau+1)} &= \mathbf{w}^{(\tau)} - (\boldsymbol{\Phi}^\top \boldsymbol{\Phi})^{-1} (\boldsymbol{\Phi}^\top \boldsymbol{\Phi} \mathbf{w}^{(\tau)} - \boldsymbol{\Phi}^\top \mathbf{t}) \\ &= (\boldsymbol{\Phi}^\top \boldsymbol{\Phi})^{-1} \boldsymbol{\Phi}^\top \mathbf{t} \end{aligned}$$

This is the closed-form solution of the least-squares objective!

Newton-Raphson for the Cross-Entropy Error

- Now, let's try Newton-Raphson on the cross-entropy error function:

$$E(\mathbf{w}) = \frac{1}{2} \sum_{n=1}^N (t_n \ln y_n + (1 - t_n) \ln(1 - y_n))$$

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

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

$$\boxed{\begin{aligned}\sigma'(a) &= \sigma(a)(1 - \sigma(a)) \\ \frac{\partial y_n}{\partial \mathbf{w}} &= y_n(1 - y_n) \boldsymbol{\phi}_n\end{aligned}}$$

- where $\mathbf{R} \in \mathbb{R}^{N \times N}$ is an $N \times N$ diagonal matrix with $R_{nn} = y_n(1 - y_n)$.
- The Hessian now depends on \mathbf{w} through the weighting matrix \mathbf{R} .

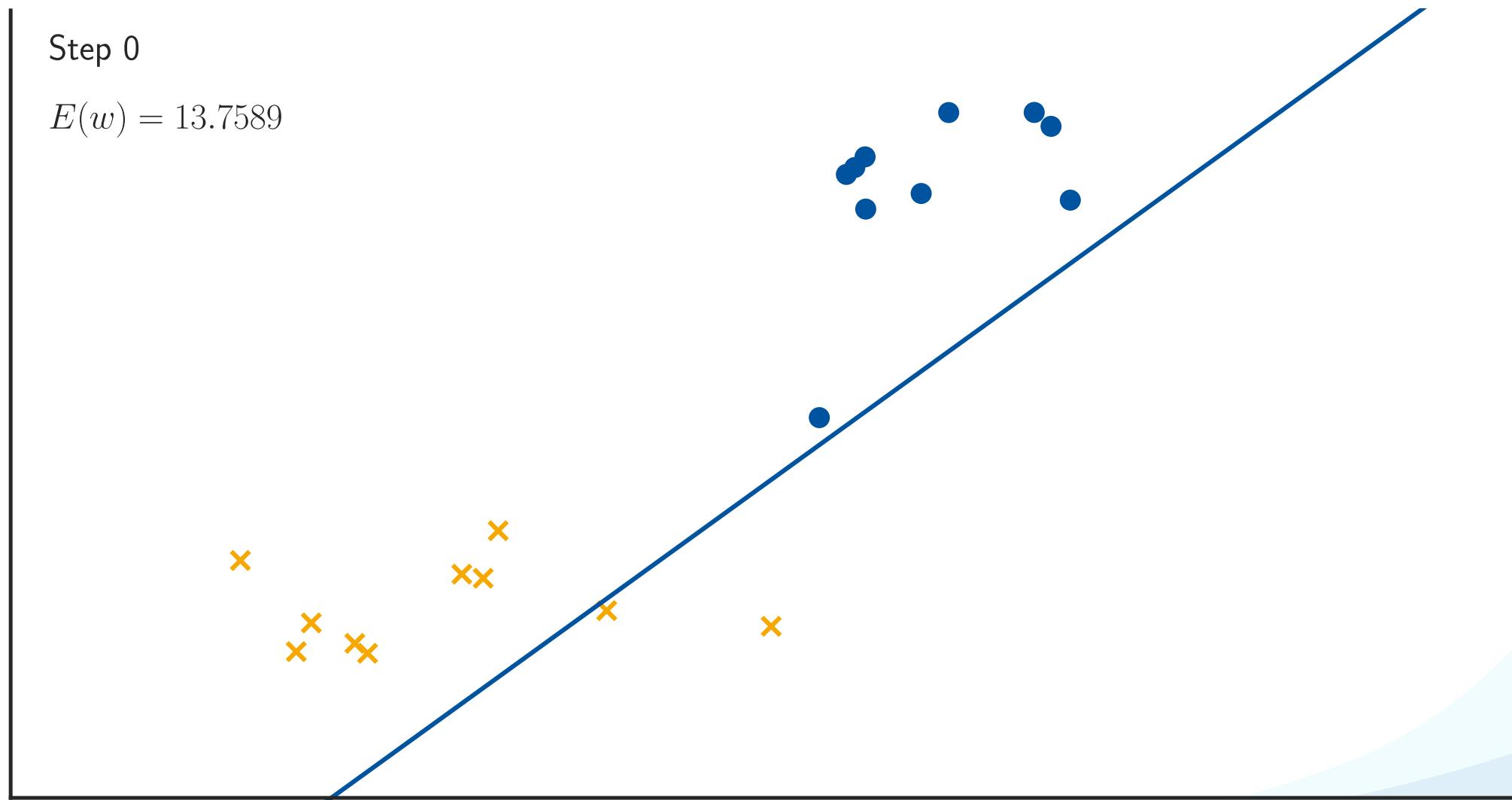
Iteratively Reweighted Least Squares (IRLS)

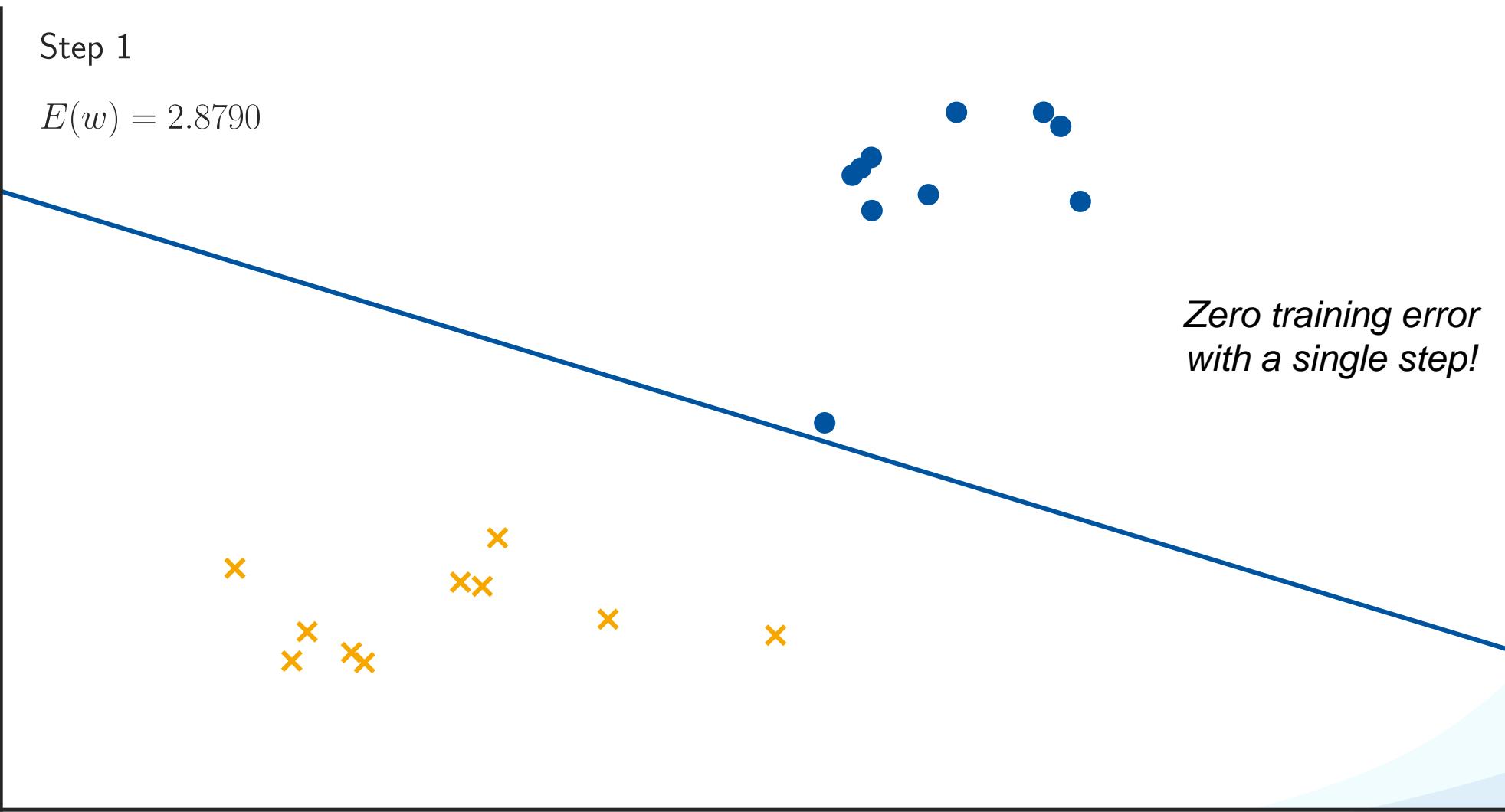
- Update equations:

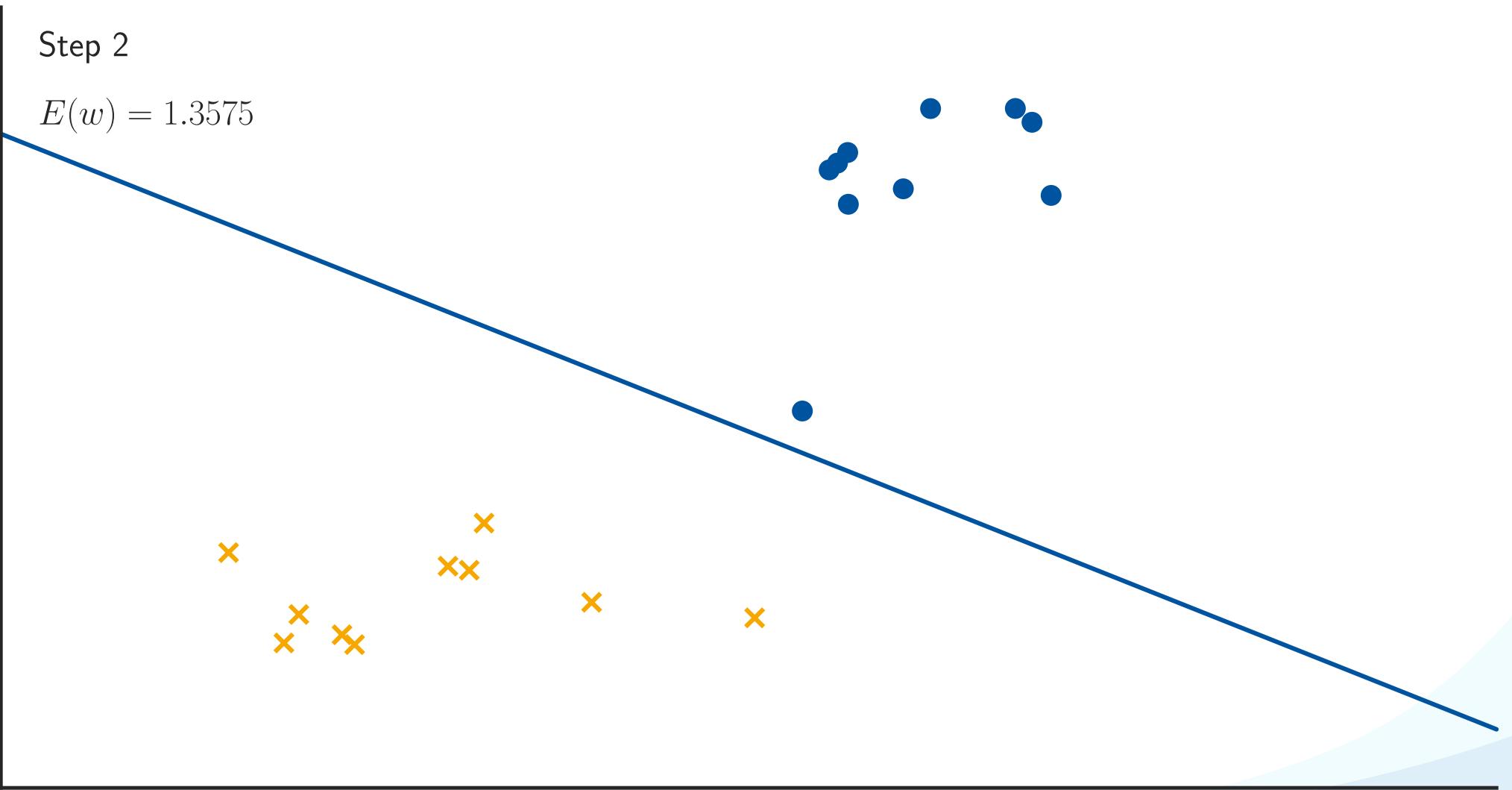
$$\begin{aligned}\mathbf{w}^{(\tau+1)} &= \mathbf{w}^{(\tau)} - (\Phi^T \mathbf{R} \Phi)^{-1} \Phi^T (\mathbf{y} - \mathbf{t}) \\ &= (\Phi^T \mathbf{R} \Phi)^{-1} \left(\Phi^T \mathbf{R} \Phi \mathbf{w}^{(\tau)} - \Phi^T (\mathbf{y} - \mathbf{t}) \right) \\ &= (\Phi^T \mathbf{R} \Phi)^{-1} \Phi^T \mathbf{R} \mathbf{z} \\ \text{with } \mathbf{z} &= \Phi \mathbf{w}^{(\tau)} - \mathbf{R}^{-1} (\mathbf{y} - \mathbf{t})\end{aligned}$$

- Very similar form (normal equations).
 - But now with non-constant weighting matrix \mathbf{R} (depends on \mathbf{w}).
 - Need to apply normal equations iteratively.
 - This is called [Iteratively Reweighted Least-Squares \(IRLS\)](#).

Example: Logistic Regression with IRLS

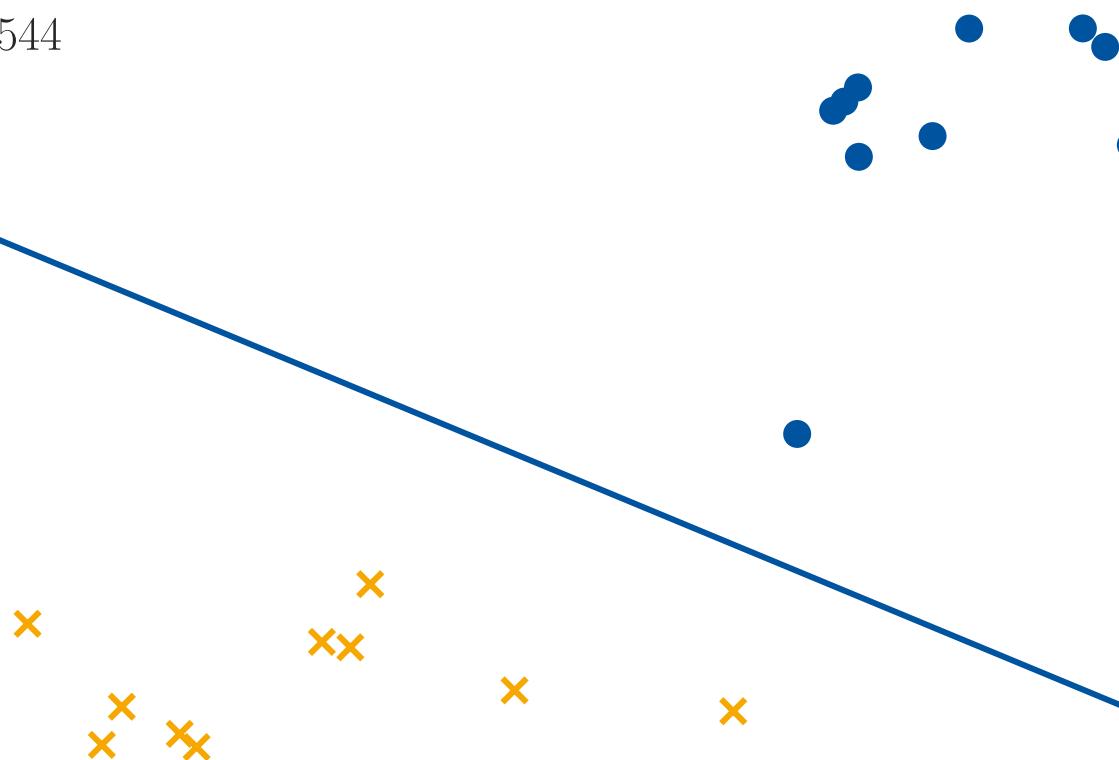






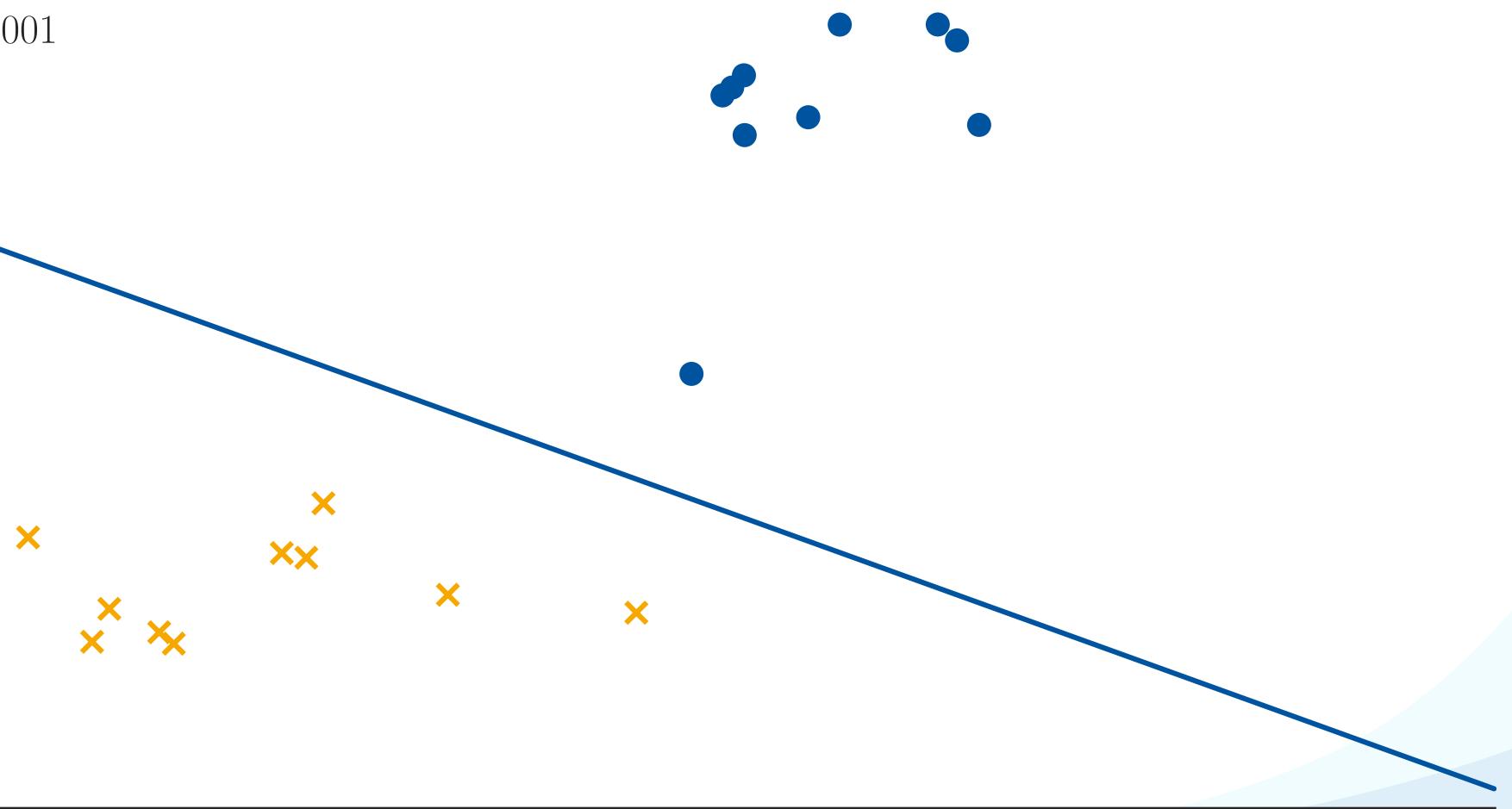
Step 6

$$E(w) = 0.0544$$



Step 12

$$E(w) = 0.0001$$



Discussion: Second-Order Optimization

Advantages

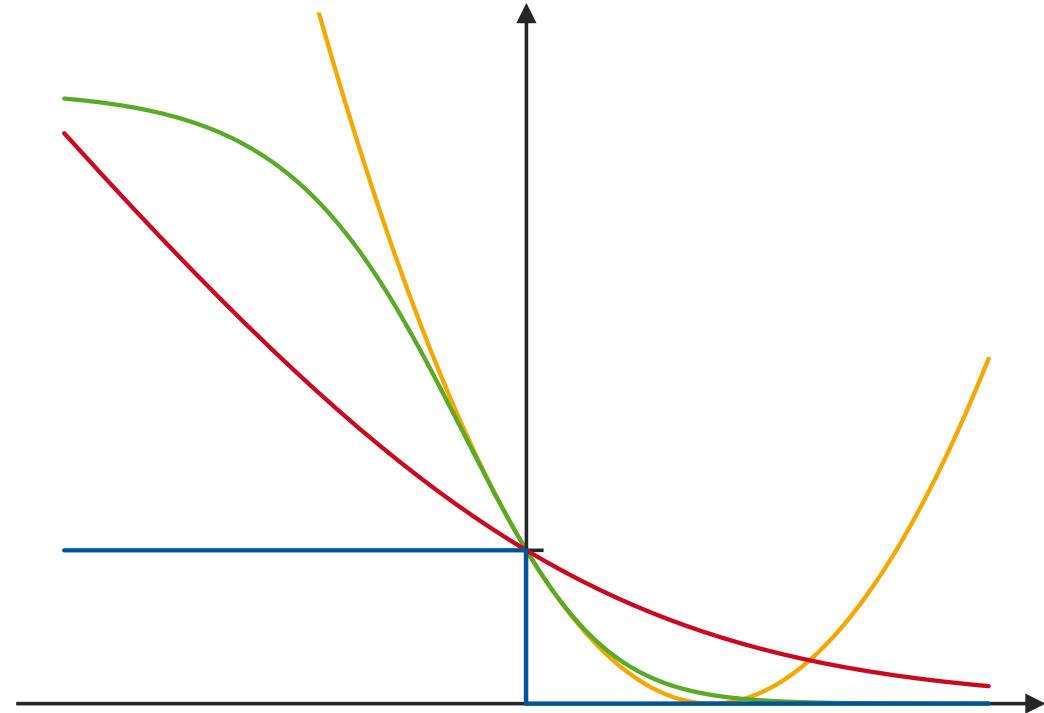
- Faster convergence than first-order methods

Limitations

- Second-order approach, relies on computing second derivatives.
- Computing (and inverting) the Hessian matrix is expensive for problems with many parameters.

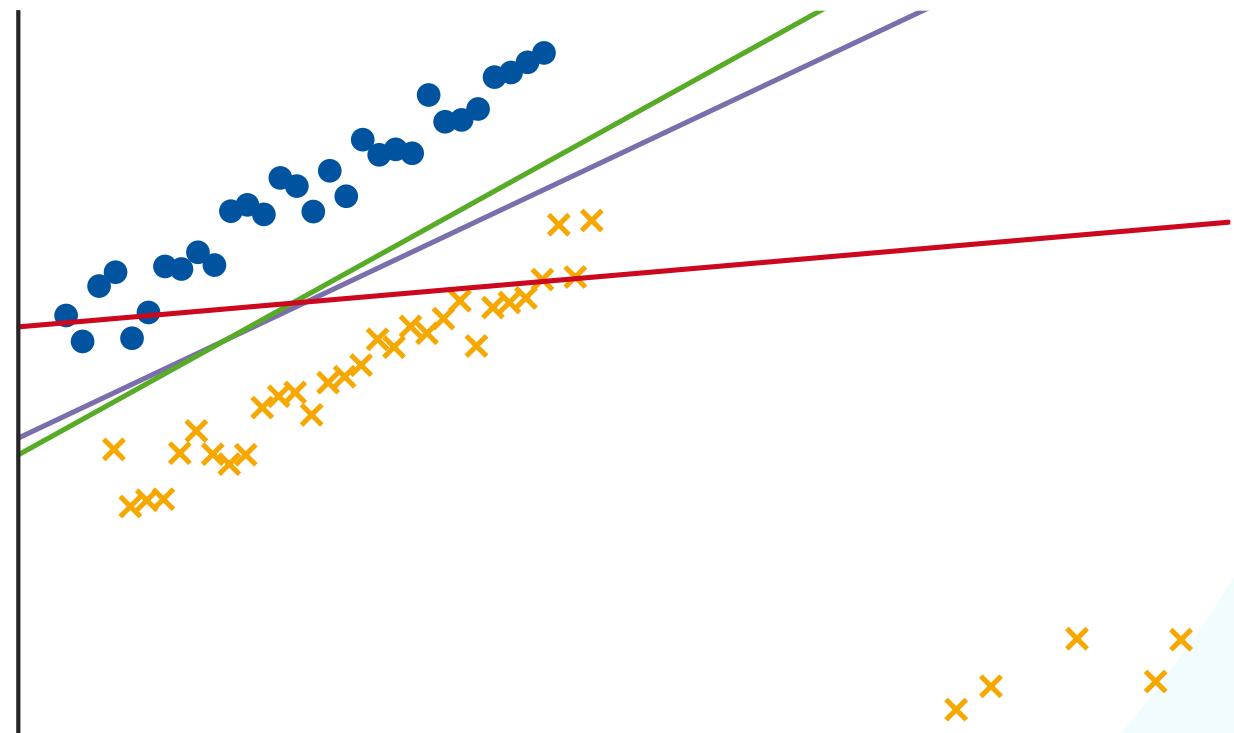
Logistic Regression

1. Logistic Regression Formulation
2. Motivation and Background
3. Iterative Estimation
4. First-Order Gradient Descent
5. Second-Order Gradient Descent
6. **Error Function Analysis**

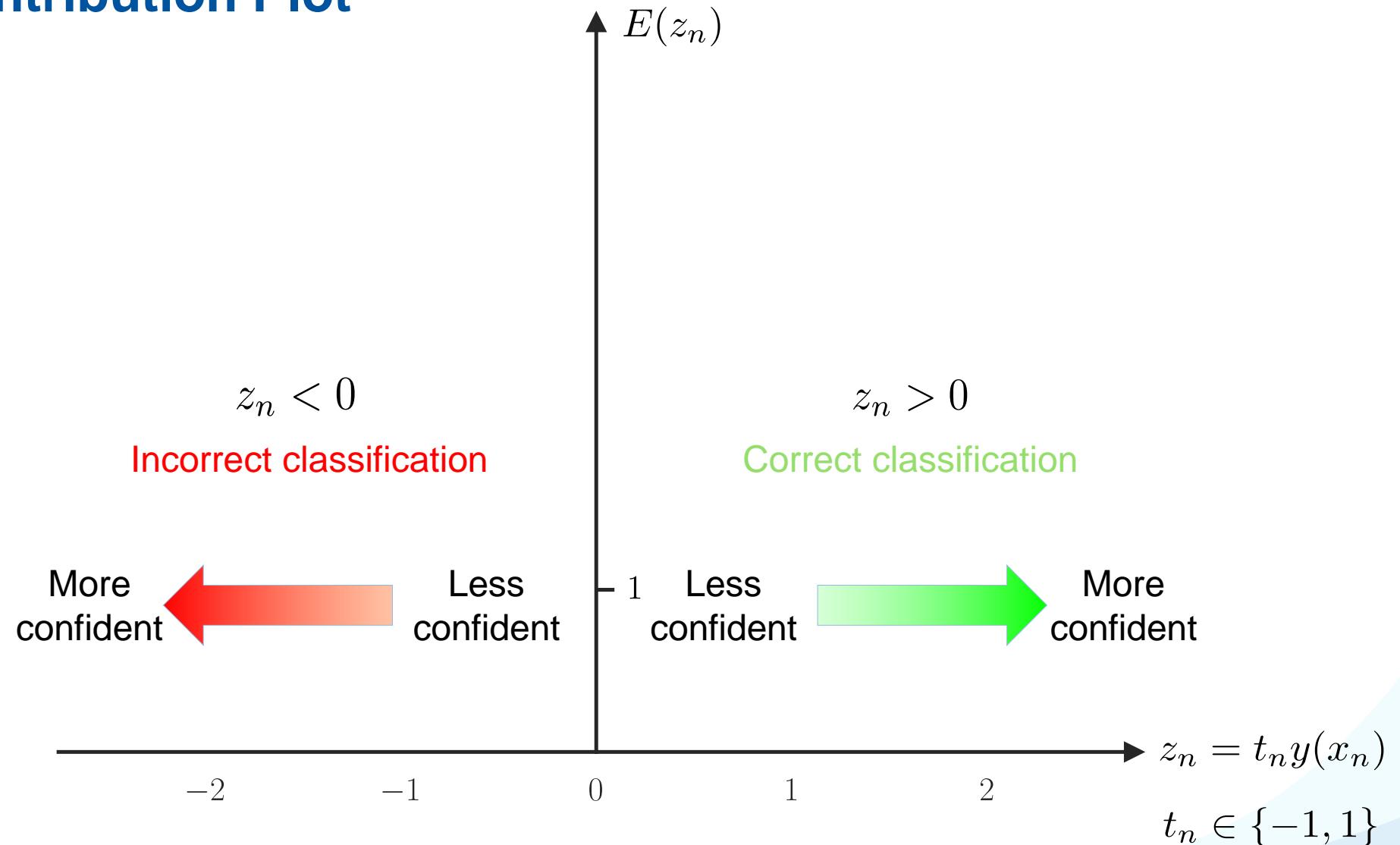


Error Function Analysis

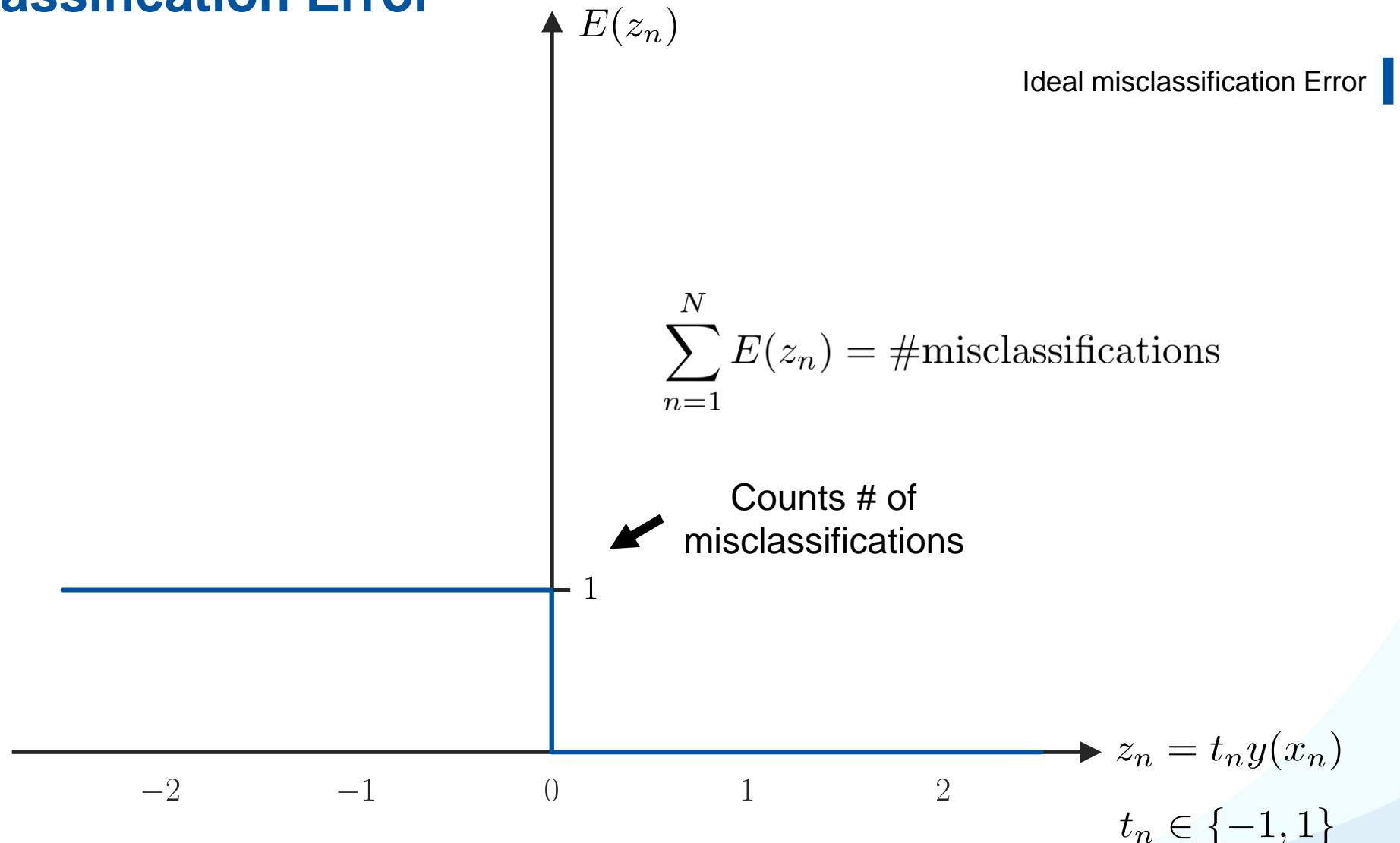
- We have seen how to learn **generalized linear discriminant** models by optimizing an error function.
 - We observed problems with **least-squares classification** based on the squared error function.
 - We have seen that **logistic regression** behaves more robustly.
- *Let's analyze the cross-entropy error in more detail...*



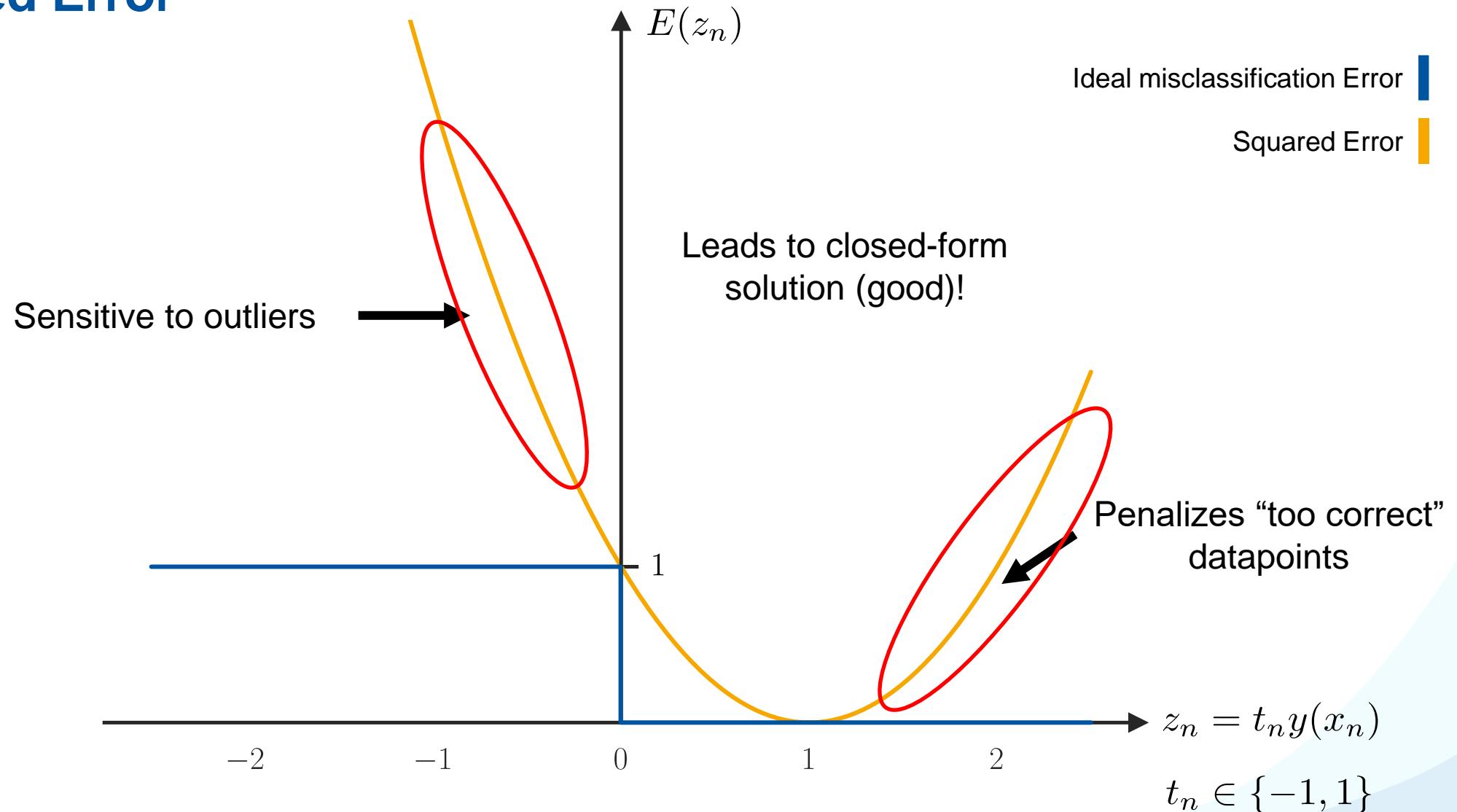
Error Contribution Plot



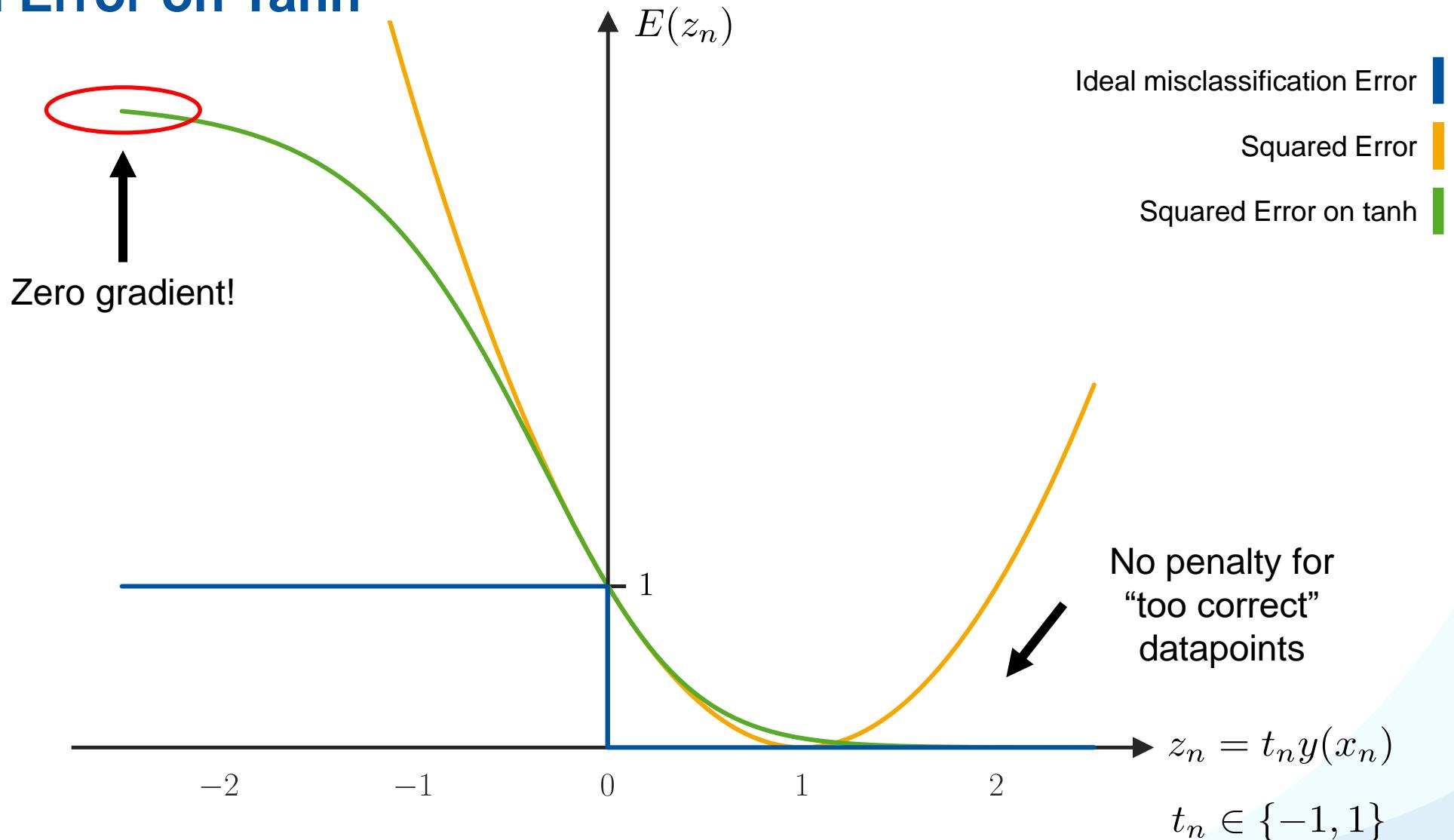
Ideal Misclassification Error



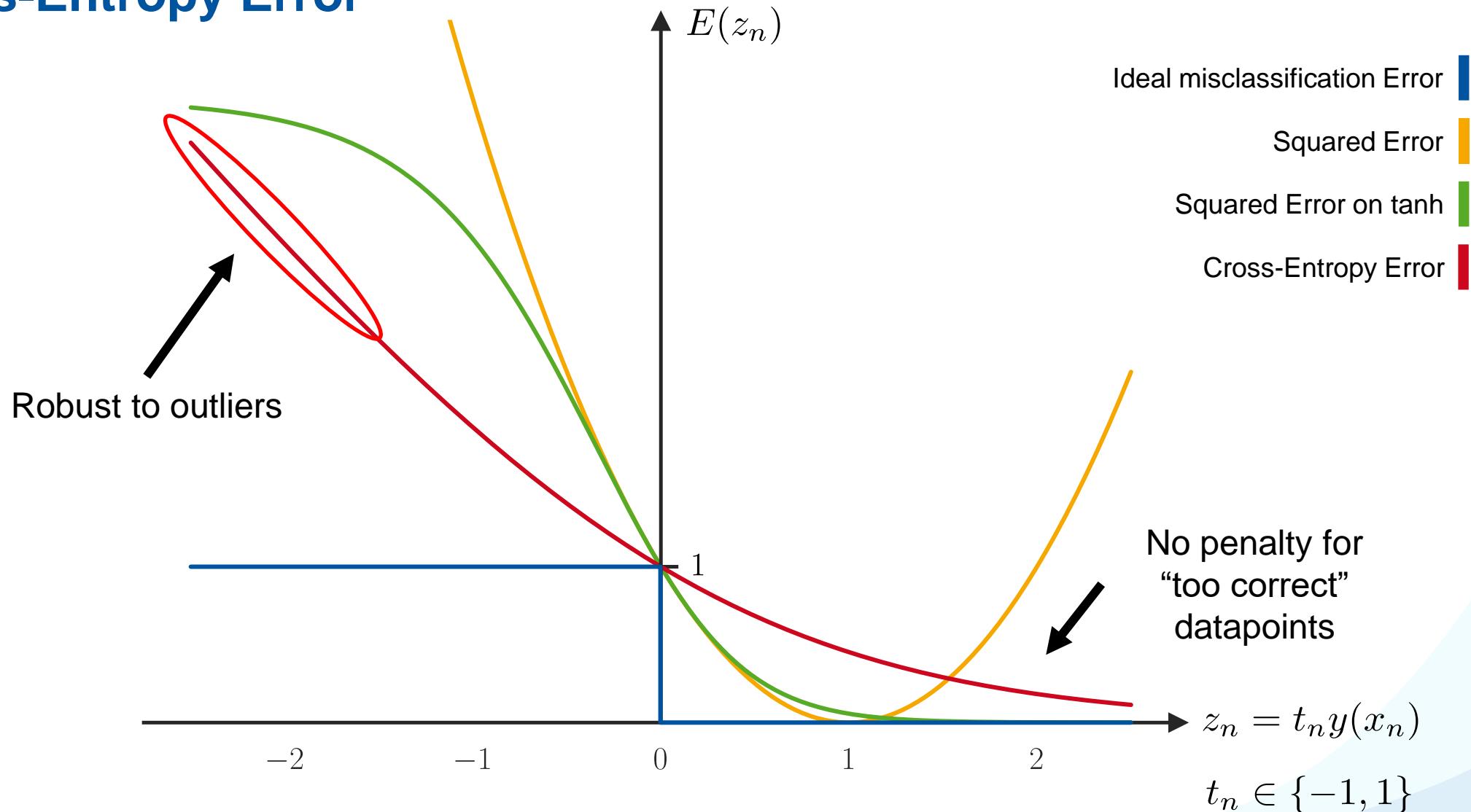
Squared Error



Squared Error on Tanh



Cross-Entropy Error



Discussion: Cross-Entropy Error

Advantages

- Minimizer of this error corresponds to class posteriors
- Convex function, unique minimum exists
- Robust to outliers

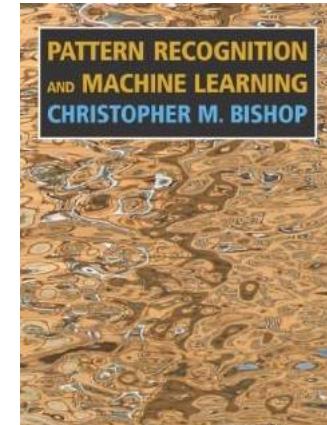
Limitations

- No closed-form solution, requires iterative estimation

References and Further Reading

- More information about [Logistic Regression](#) is available in Chapter 4.3 of Bishop's book.

Christopher M. Bishop
Pattern Recognition and Machine Learning
Springer, 2006



Elements of Machine Learning & Data Science

Winter semester 2023/24

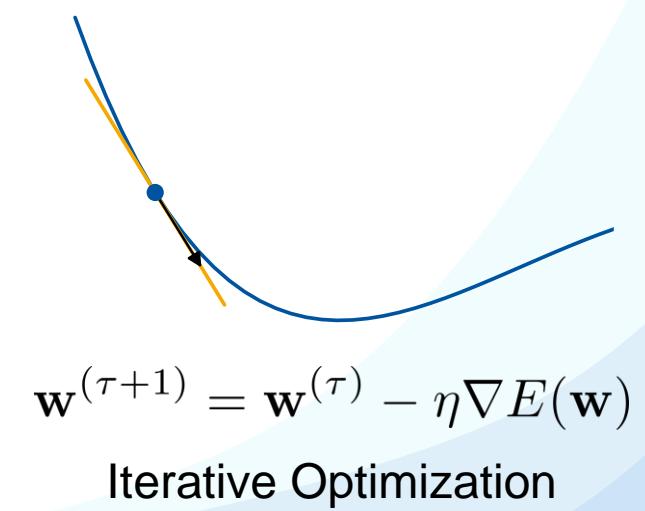
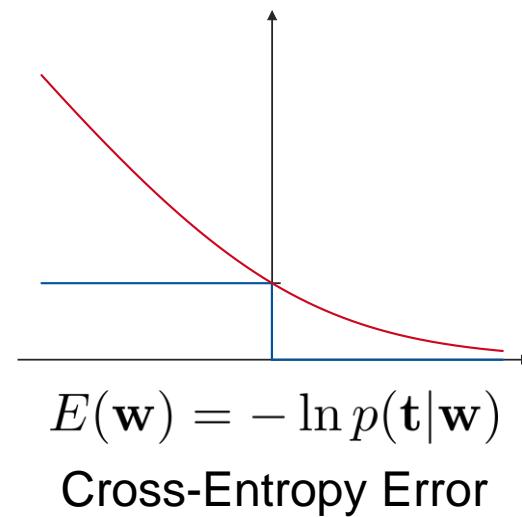
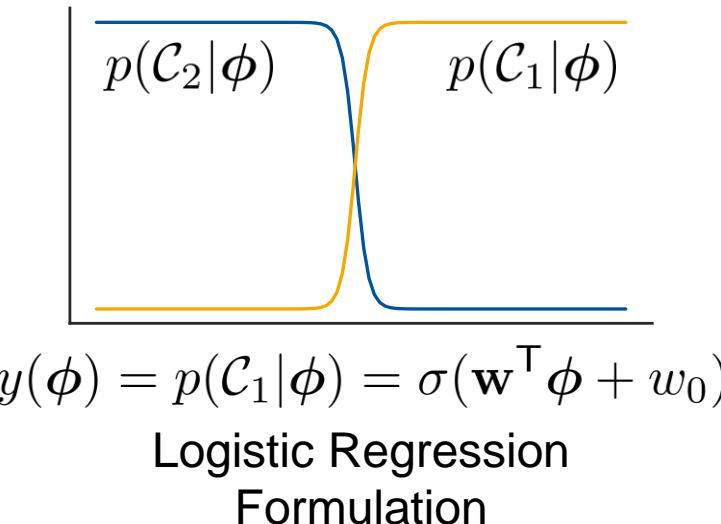
Lecture 17 – Support Vector Machines I

12.12.2023

Prof. Bastian Leibe

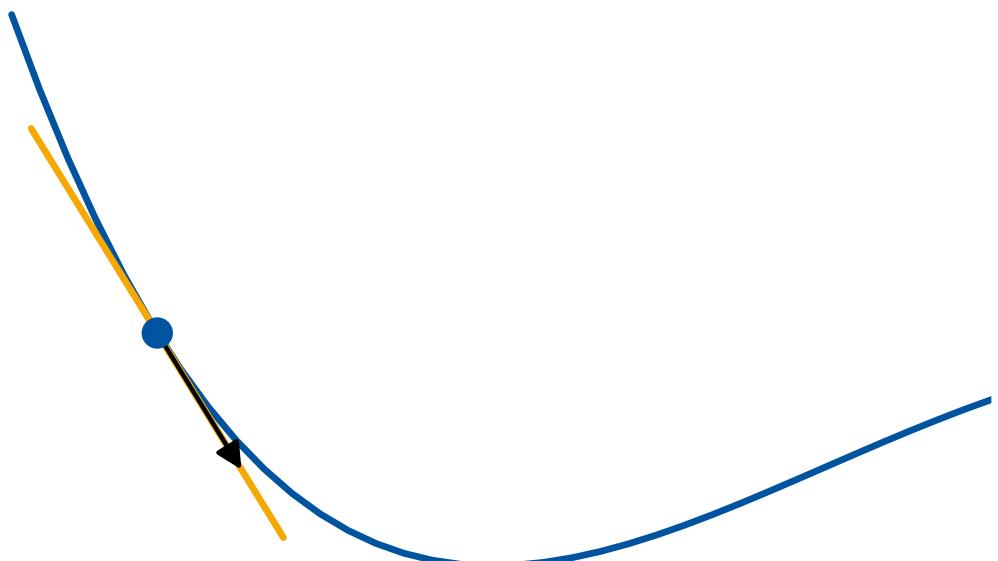
Machine Learning Topics

1. Introduction to ML
2. Probability Density Estimation
3. Linear Discriminants
4. Linear Regression
5. **Logistic Regression**
6. Support Vector Machines
7. AdaBoost
8. Neural Network Basics



Logistic Regression

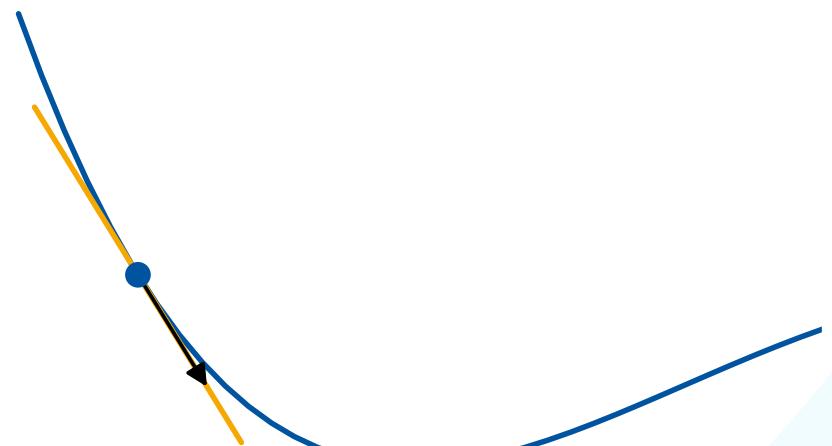
1. Logistic Regression Formulation
2. Motivation and Background
- 3. Iterative Optimization**
4. First-Order Gradient Descent
5. Second-Order Gradient Descent
6. Error Function Analysis



Recap: Iterative Optimization

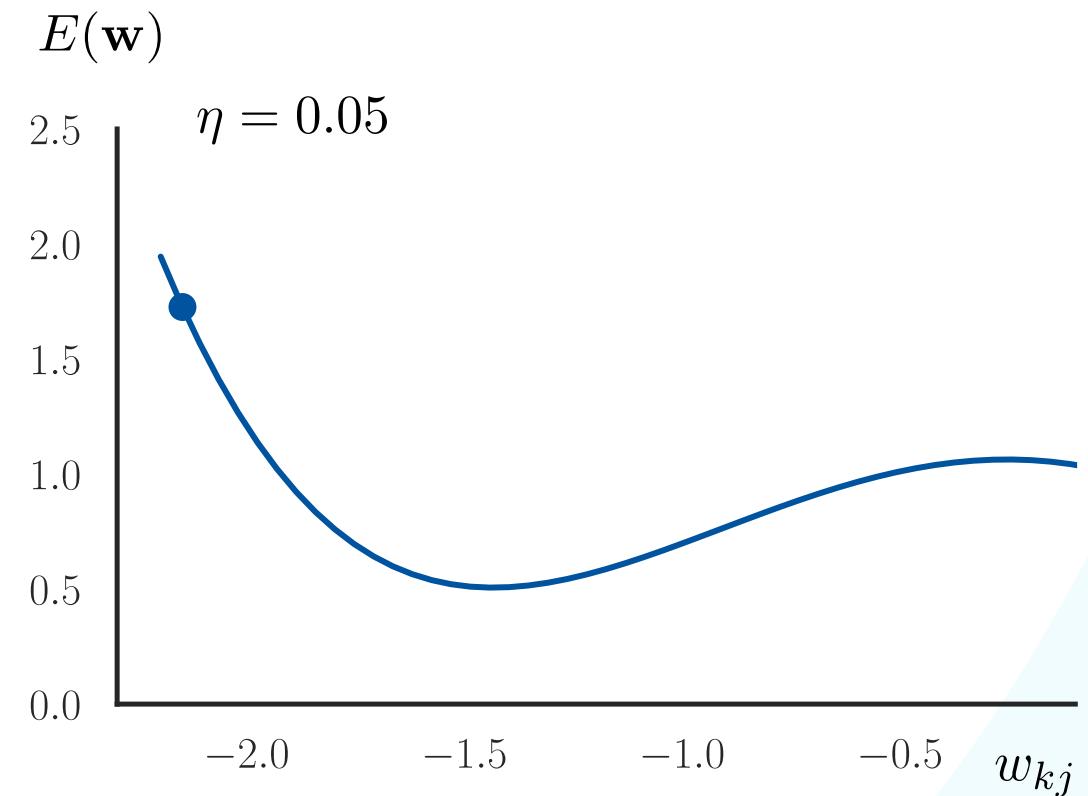
- In general, generalized linear discriminants with nonlinear activation and/or basis functions can no longer be optimized in closed form.
- Instead, we use iterative optimization schemes.
- Here: **Gradient Descent**.
 - Start with initial guess for parameter values.
 - Move towards a minimum of the error function by following the direction of steepest descent.
 - Iterate until convergence

$$y_k(\mathbf{x}) = g \left(\sum_{j=0}^M w_{kj} \phi_j(\mathbf{x}) \right) = g(\mathbf{w}^\top \phi(\mathbf{x}))$$



Idea: Gradient Descent

- Start with an initial guess of parameter values $w_{kj}^{(0)}$.

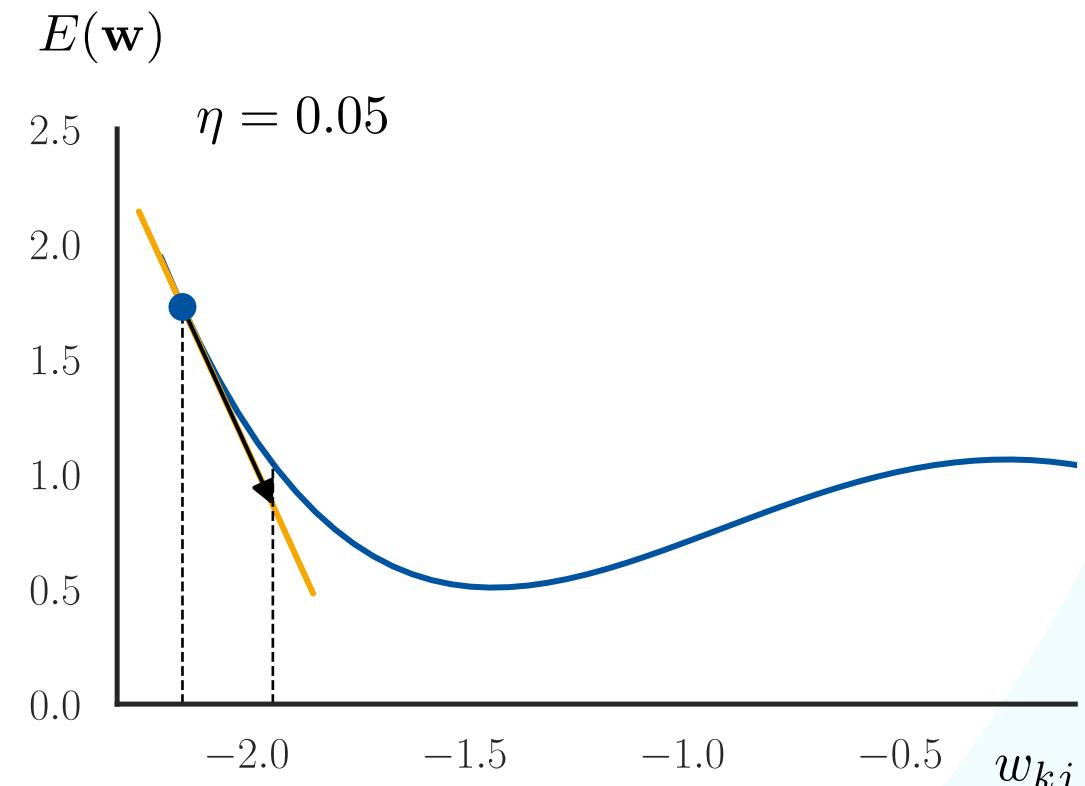


Idea: Gradient Descent

- Start with an initial guess of parameter values $w_{kj}^{(0)}$.
- Follow the gradient to move to a (local) minimum:

$$w_{kj}^{(\tau+1)} = w_{kj}^{(\tau)} - \eta \frac{\partial E(\mathbf{w})}{\partial w_{kj}} \Big|_{\mathbf{w}^{(\tau)}}$$

- η is called the **learning rate**.
- This corresponds to a 1st-order Taylor expansion.
 - I.e., we approximate the error function by its tangent plane around the current point $\mathbf{w}^{(\tau)}$.
- Repeat this procedure for a number of steps.

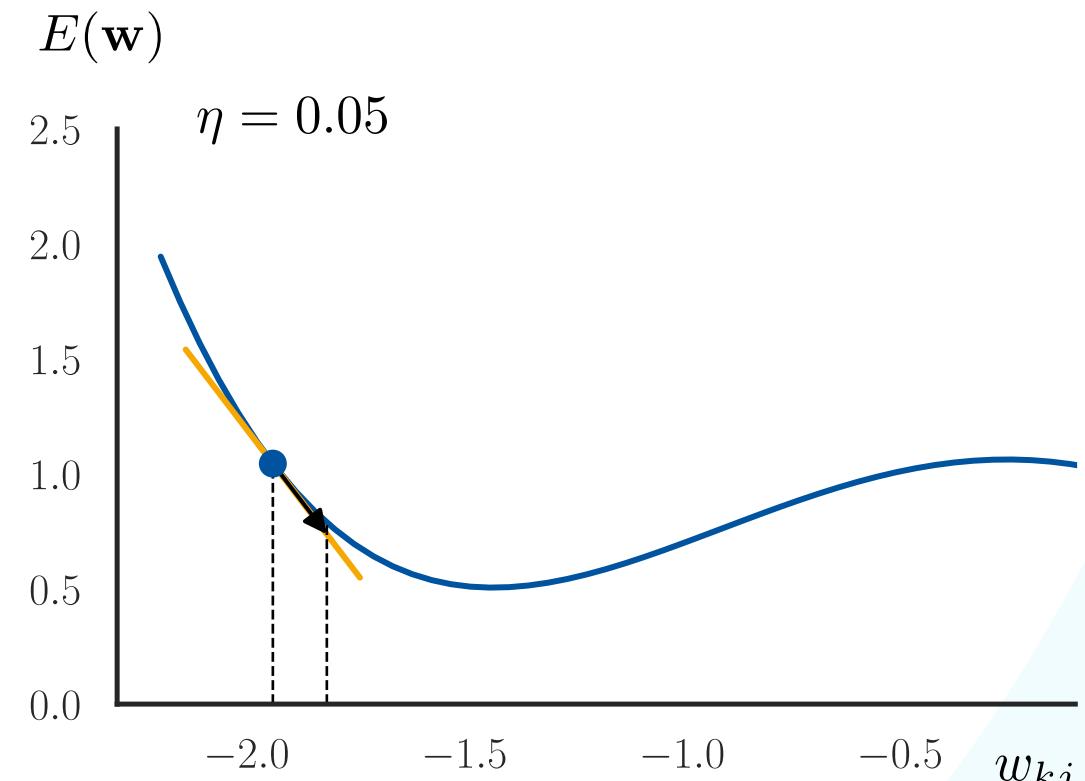


Idea: Gradient Descent

- Start with an initial guess of parameter values $w_{kj}^{(0)}$.
- Follow the gradient to move to a (local) minimum:

$$w_{kj}^{(\tau+1)} = w_{kj}^{(\tau)} - \eta \frac{\partial E(\mathbf{w})}{\partial w_{kj}} \Big|_{\mathbf{w}^{(\tau)}}$$

- η is called the **learning rate**.
- This corresponds to a 1st-order Taylor expansion.
 - I.e., we approximate the error function by its tangent plane around the current point $\mathbf{w}^{(\tau)}$.
- Repeat this procedure for a number of steps.

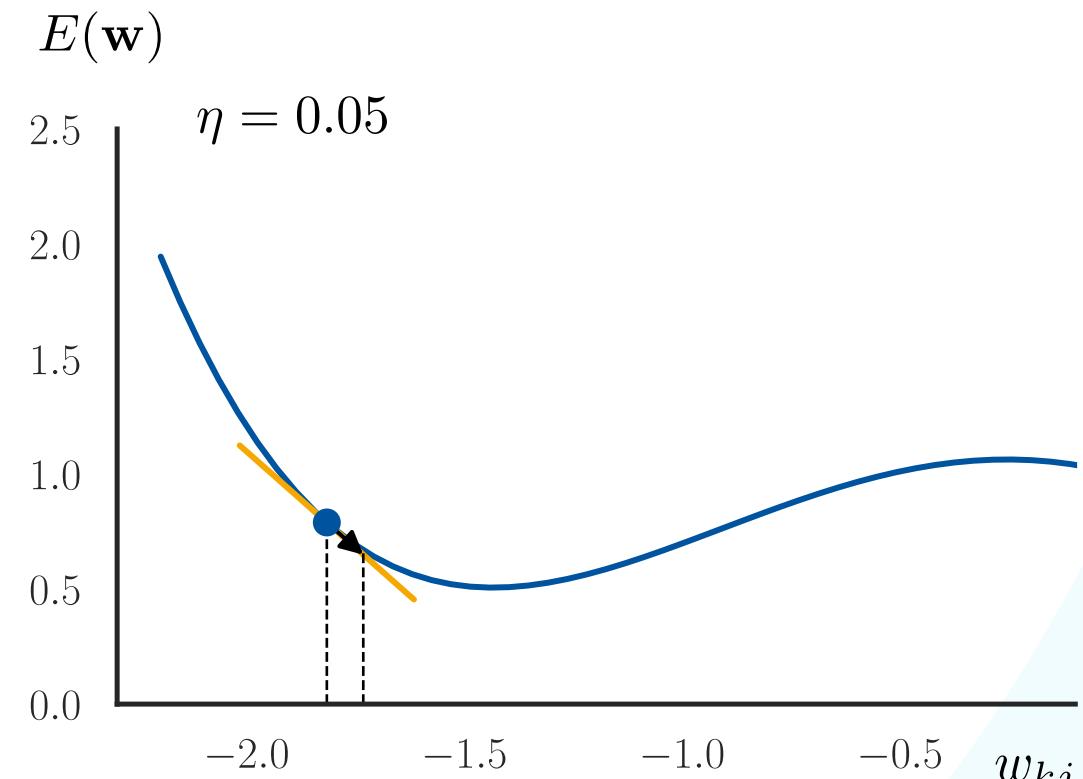


Idea: Gradient Descent

- Start with an initial guess of parameter values $w_{kj}^{(0)}$.
- Follow the gradient to move to a (local) minimum:

$$w_{kj}^{(\tau+1)} = w_{kj}^{(\tau)} - \eta \frac{\partial E(\mathbf{w})}{\partial w_{kj}} \Big|_{\mathbf{w}^{(\tau)}}$$

- η is called the **learning rate**.
- This corresponds to a 1st-order Taylor expansion.
 - I.e., we approximate the error function by its tangent plane around the current point $\mathbf{w}^{(\tau)}$.
- Repeat this procedure for a number of steps.

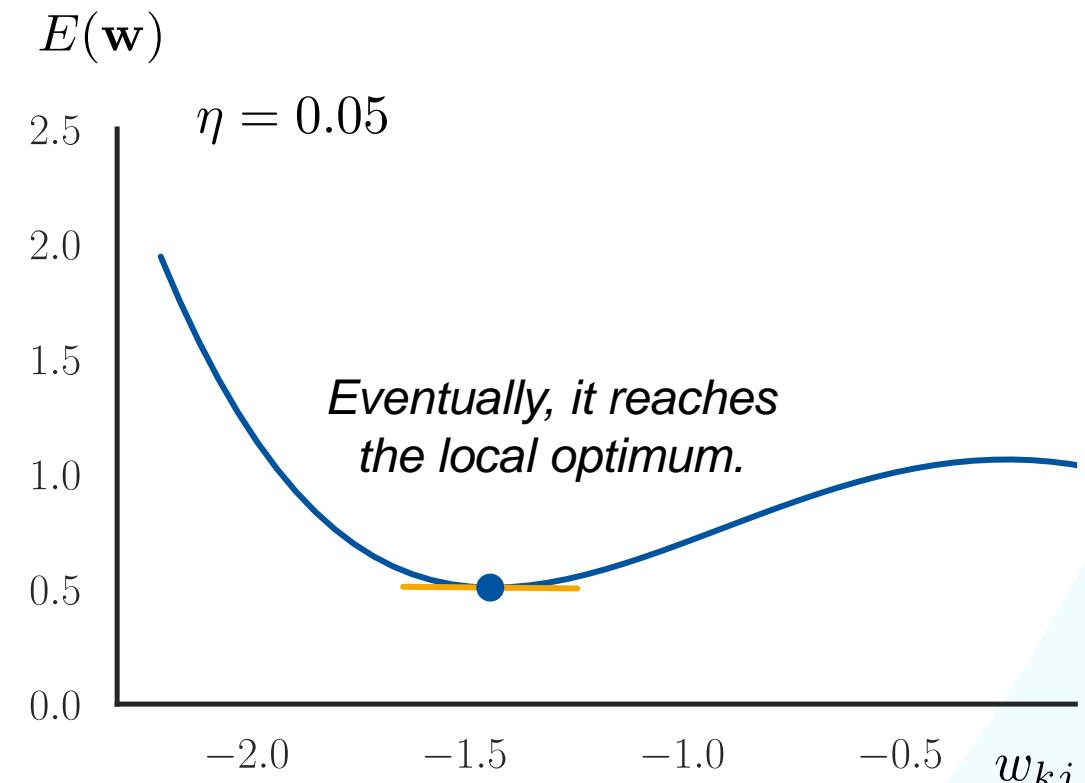


Idea: Gradient Descent

- Start with an initial guess of parameter values $w_{kj}^{(0)}$.
- Follow the gradient to move to a (local) minimum:

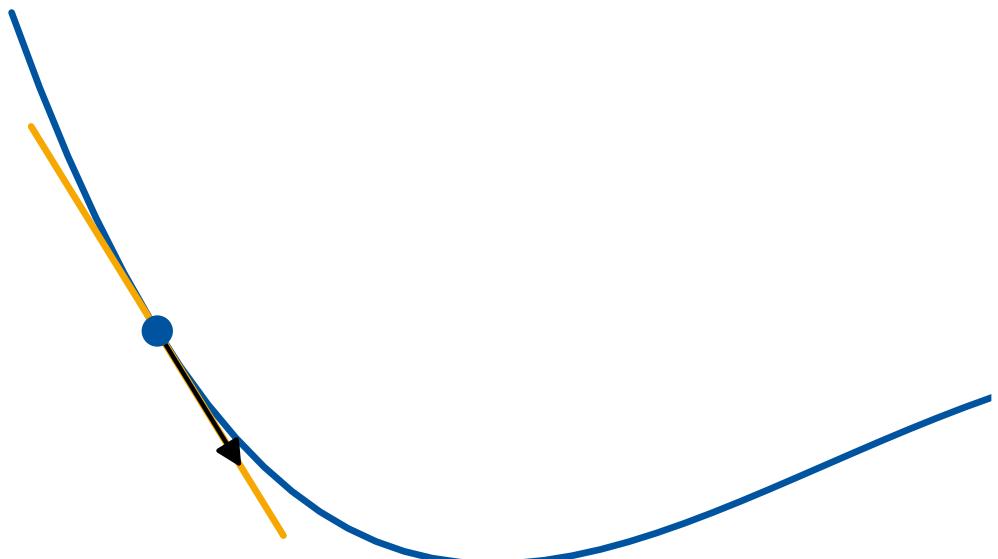
$$w_{kj}^{(\tau+1)} = w_{kj}^{(\tau)} - \eta \frac{\partial E(\mathbf{w})}{\partial w_{kj}} \Big|_{\mathbf{w}^{(\tau)}}$$

- η is called the **learning rate**.
- This corresponds to a 1st-order Taylor expansion.
 - I.e., we approximate the error function by its tangent plane around the current point $\mathbf{w}^{(\tau)}$.
- Repeat this procedure for a number of steps.



Logistic Regression

1. Logistic Regression Formulation
2. Motivation and Background
3. Iterative Optimization
4. **First-Order Gradient Descent**
5. Second-Order Gradient Descent
6. Error Function Analysis



First-order Optimization

- Logistic regression uses the **binary cross-entropy error**:

$$E(\mathbf{w}) = - \sum_{n=1}^N (t_n \ln y(\mathbf{x}_n; \mathbf{w}) + (1 - t_n) \ln(1 - y(\mathbf{x}_n; \mathbf{w})))$$

- Properties
 - Convex function, so it has a unique minimum
 - But no closed-form solution
- We need to use iterative methods for optimization
 - Let's try (first-order) gradient descent:

$$\mathbf{w}^{(\tau+1)} = \mathbf{w}^{(\tau)} - \eta \nabla E(\mathbf{w})$$

Gradient of the Cross-Entropy Error

$$E(\mathbf{w}) = - \sum_{n=1}^N (t_n \ln y_n + (1 - t_n) \ln(1 - y_n))$$

$$\begin{aligned}\nabla E(\mathbf{w}) &= - \sum_{n=1}^N \left(t_n \frac{\partial}{\partial \mathbf{w}} y_n + (1 - t_n) \frac{\partial}{\partial \mathbf{w}} (1 - y_n) \right) \\ &= - \sum_{n=1}^N \left(t_n \frac{y_n(1 - y_n)}{y_n} \phi_n + (1 - t_n) \frac{y_n(1 - y_n)}{(1 - y_n)} \phi_n \right) \\ &= - \sum_{n=1}^N ((t_n - t_n y_n - y_n + t_n y_n) \phi_n) \\ &= \sum_{n=1}^N (y_n - t_n) \phi_n\end{aligned}$$

$$y_n = y(\mathbf{x}_n; \mathbf{w})$$

$$\begin{aligned}\sigma'(a) &= \sigma(a)(1 - \sigma(a)) \\ \frac{\partial y_n}{\partial \mathbf{w}} &= y_n(1 - y_n) \phi_n\end{aligned}$$

$$\phi_n = \phi(\mathbf{x}_n)$$

- The gradient for logistic regression is

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

- We can plug this into gradient descent:

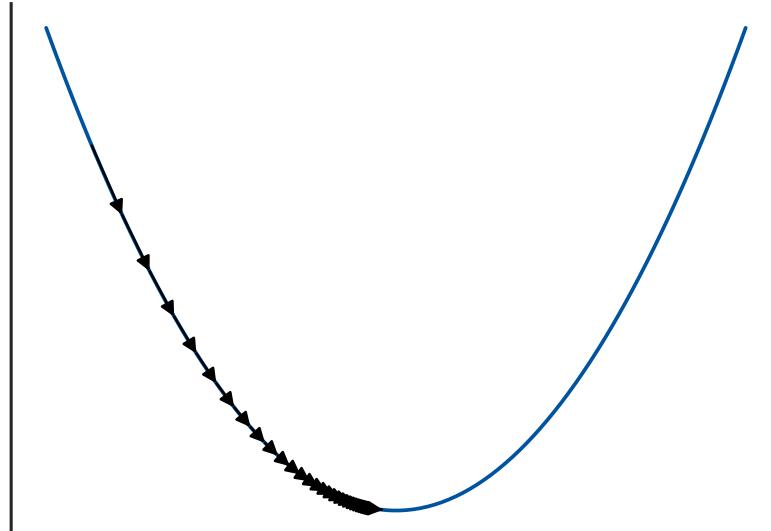
$$\begin{aligned}\mathbf{w}^{(\tau+1)} &= \mathbf{w}^{(\tau)} - \eta \nabla E(\mathbf{w}) \\ &= \mathbf{w}^{(\tau)} - \eta \sum_{n=1}^N (y_n - t_n) \phi_n\end{aligned}$$

*How should we choose
the learning rate?*

- This update rule is known as the **Delta rule** (= LMS rule)
 - Simply feed back the input data points, weighted by the classification error.*

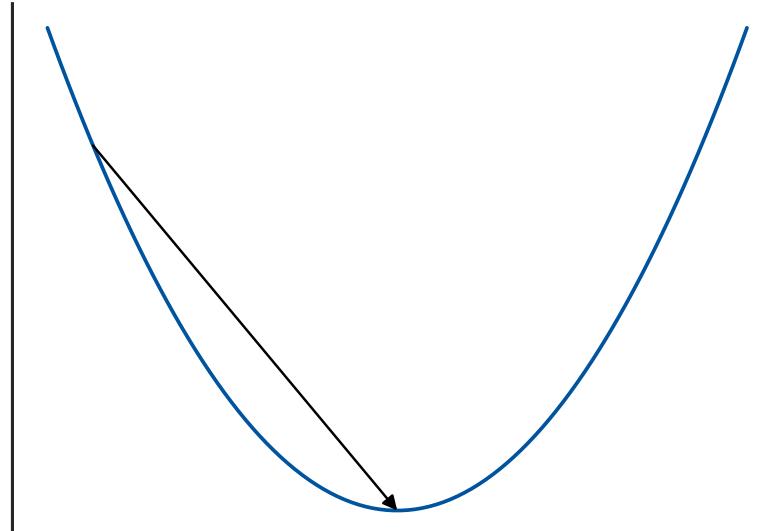
Effects of the learning rate

η too small



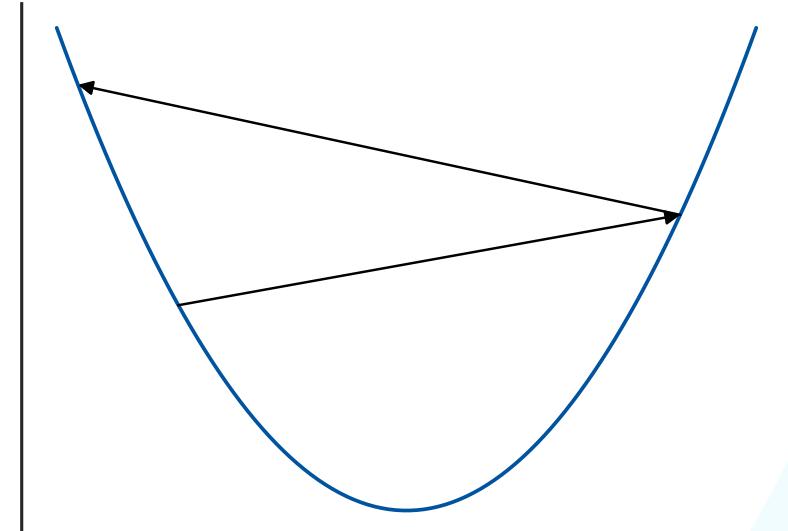
Convergence is slow

η_{opt}



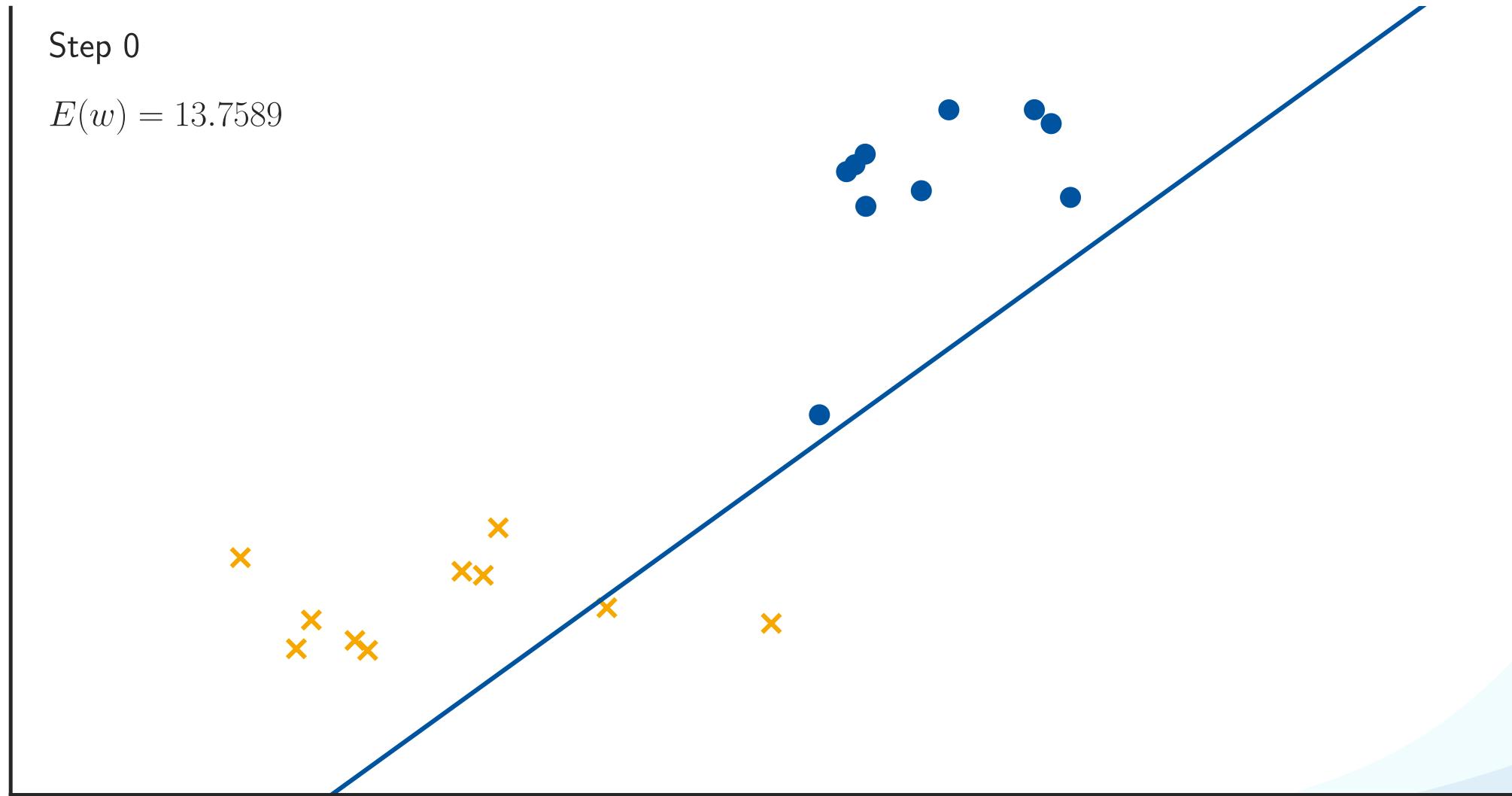
Converges ideally in
a single step

η too large



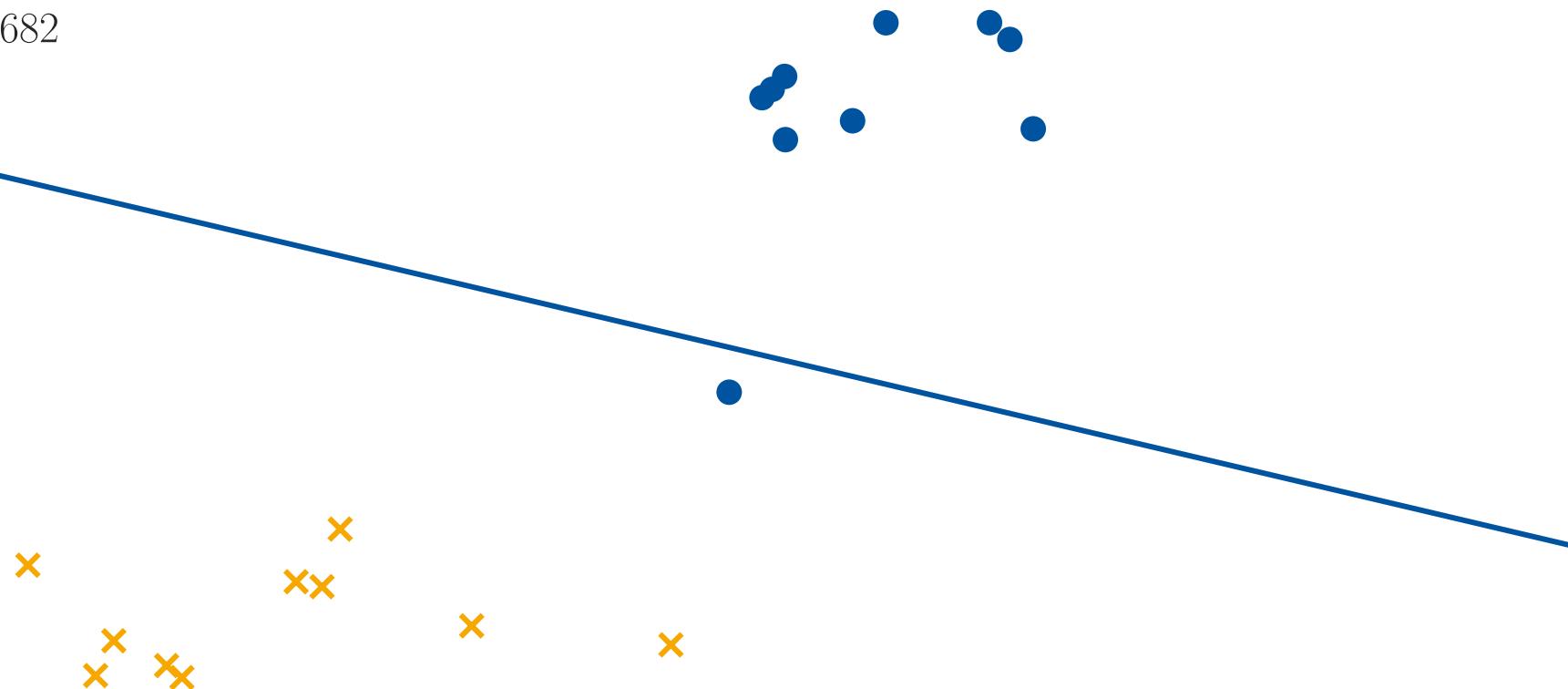
Might not converge

Example: Logistic Regression with Gradient Descent



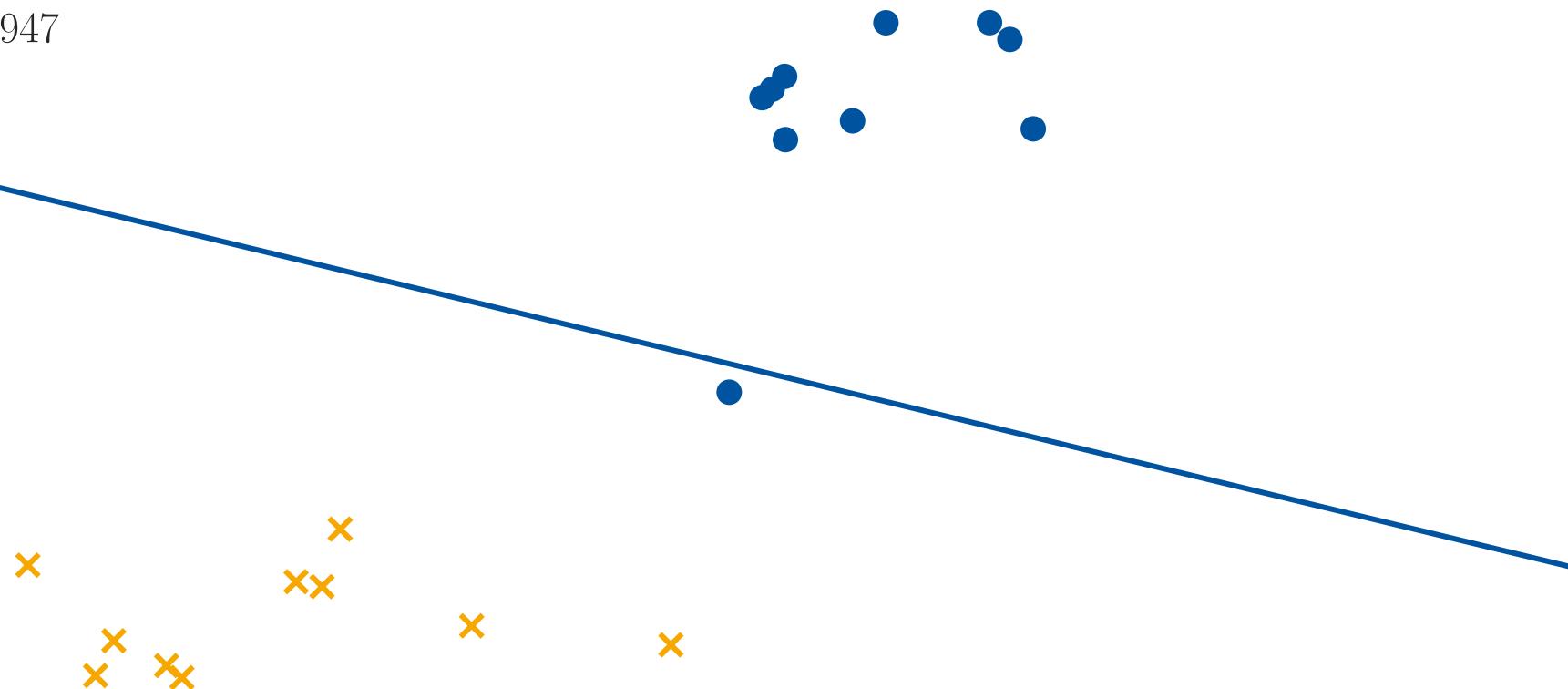
Step 1

$$E(w) = 1.3682$$



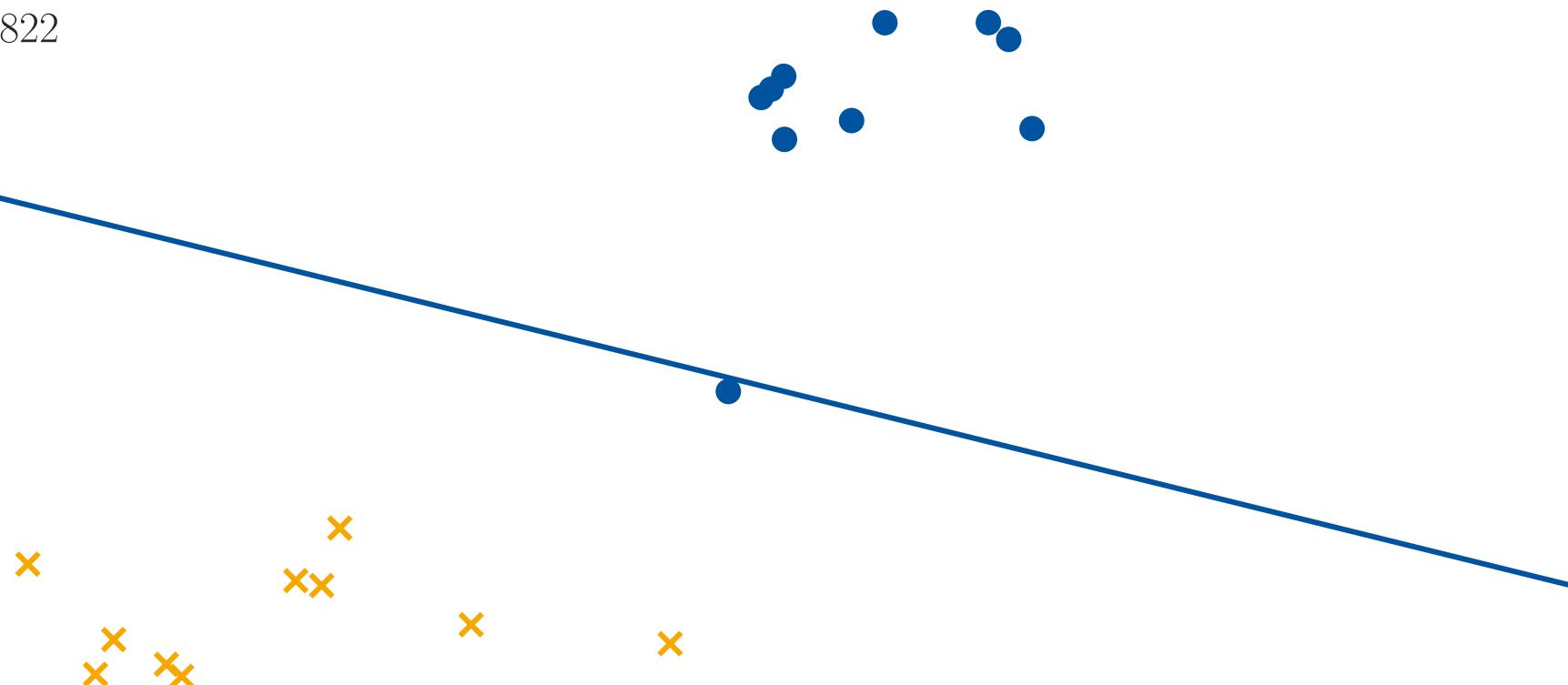
Step 2

$$E(w) = 1.0947$$



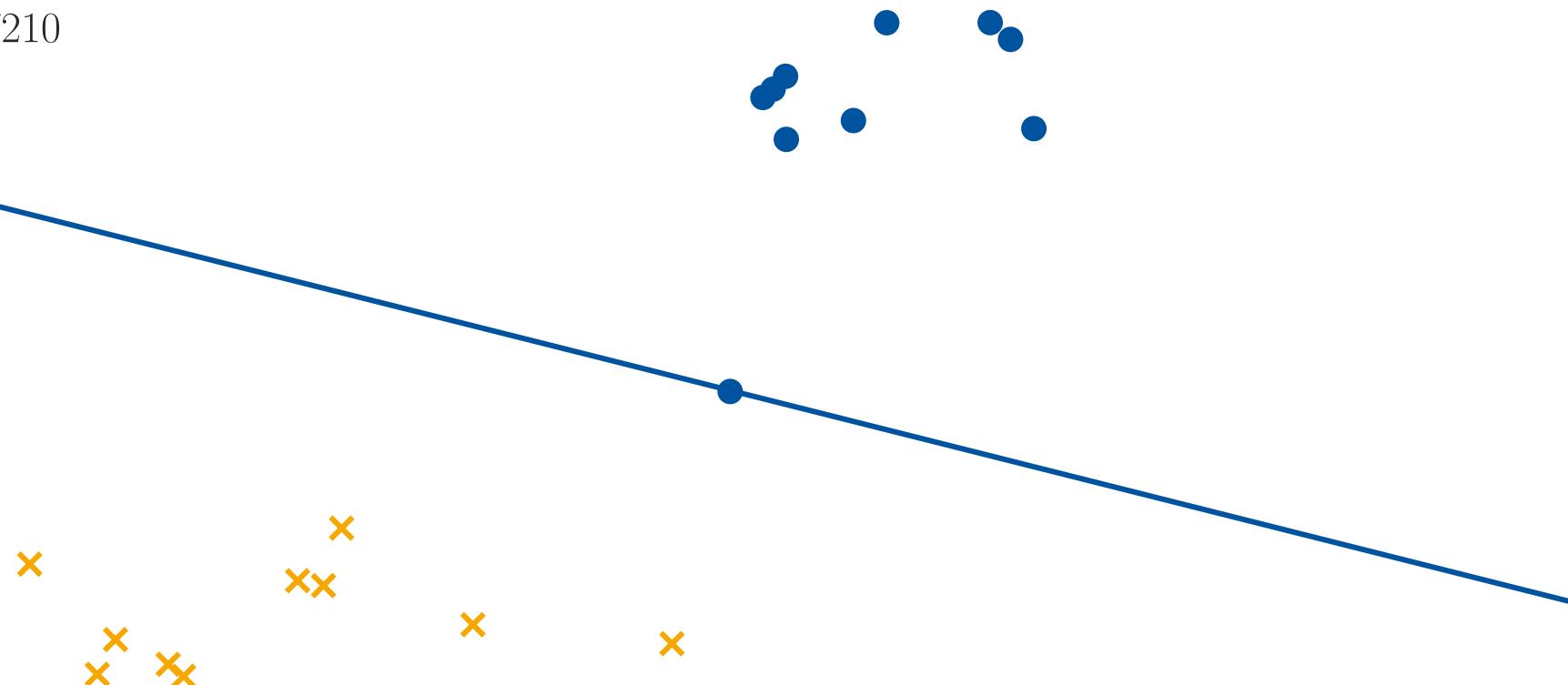
Step 3

$$E(w) = 0.8822$$



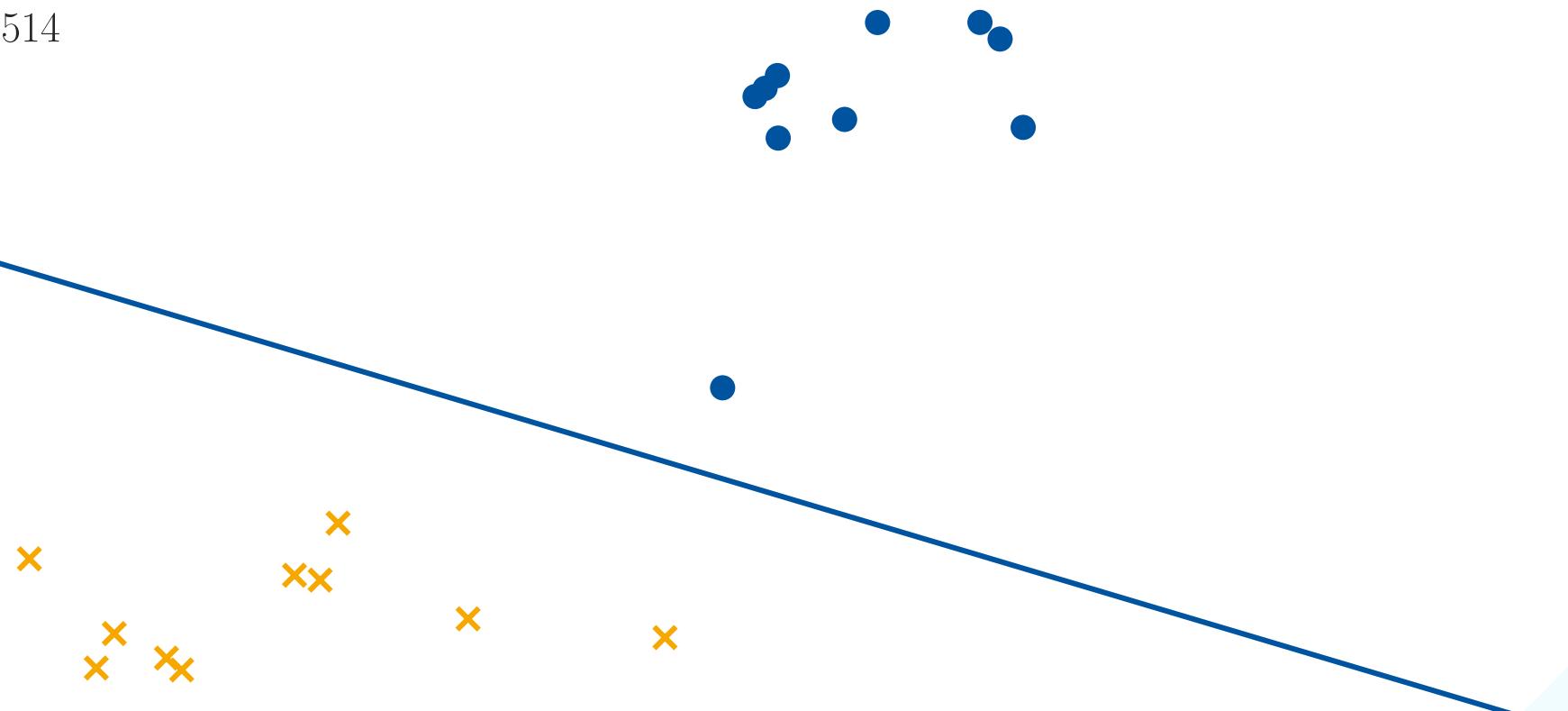
Step 4

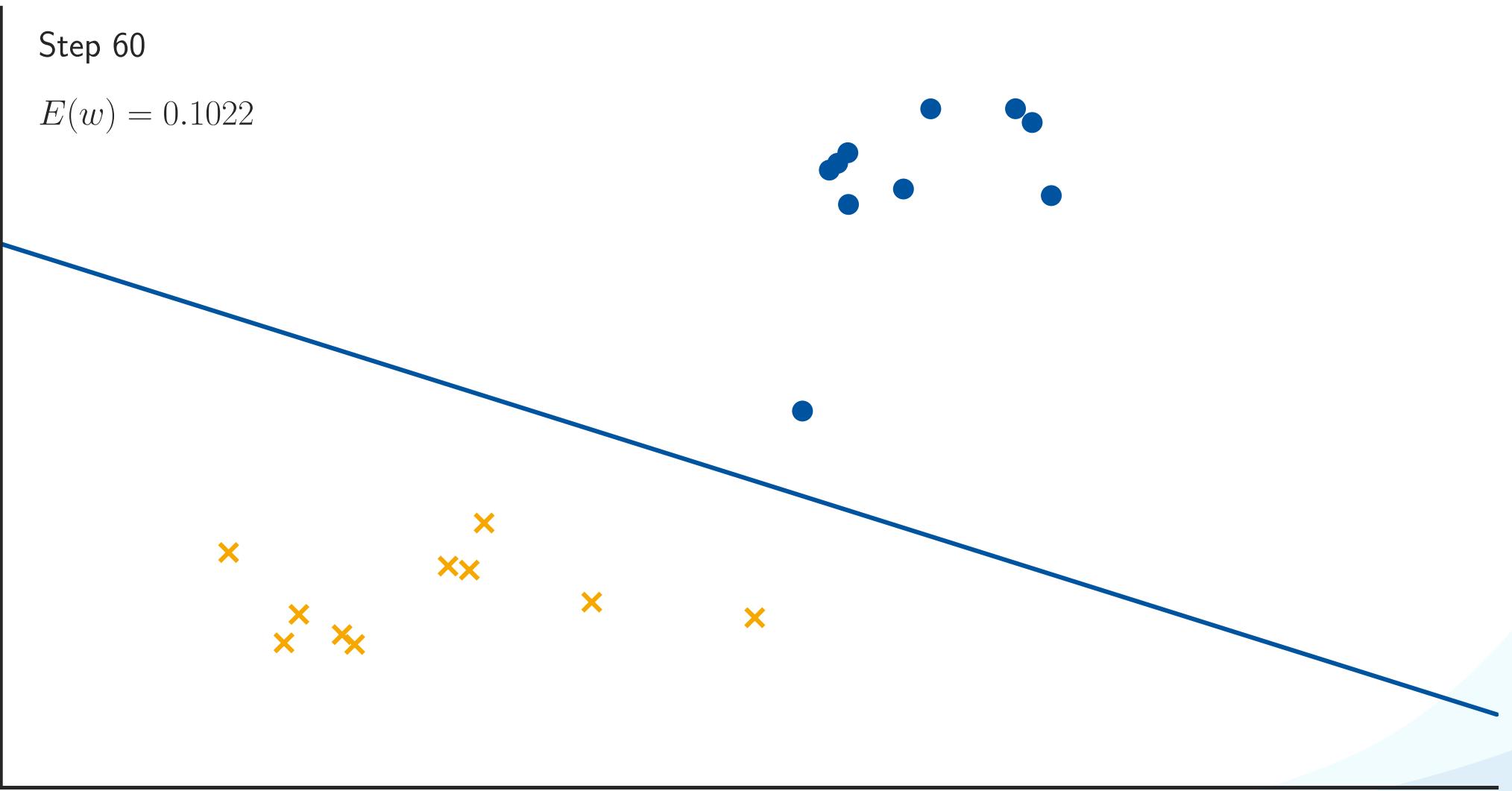
$$E(w) = 0.7210$$

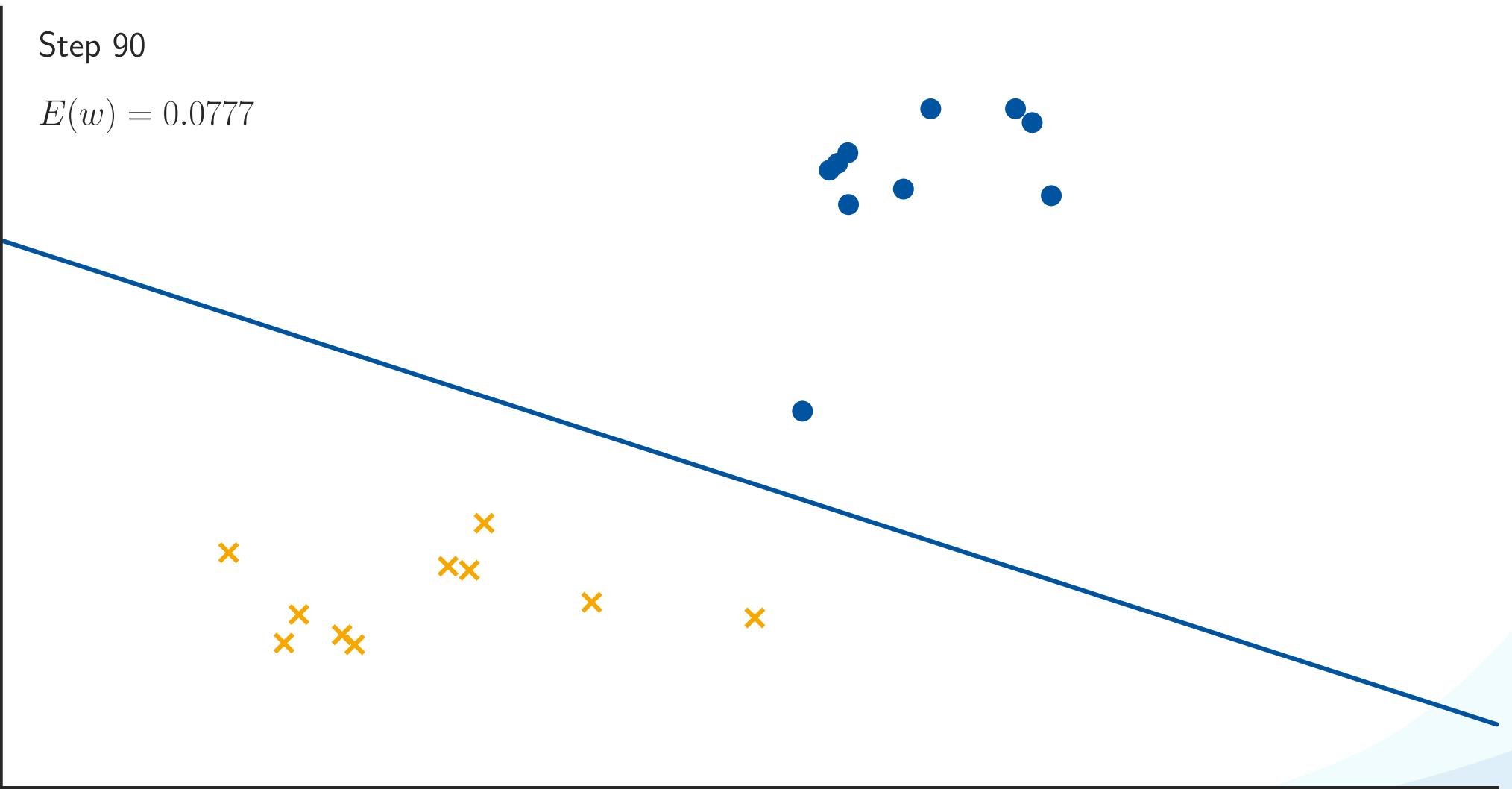


Step 30

$$E(w) = 0.1514$$







Discussion: Logistic Regression with Gradient Descent

Advantages

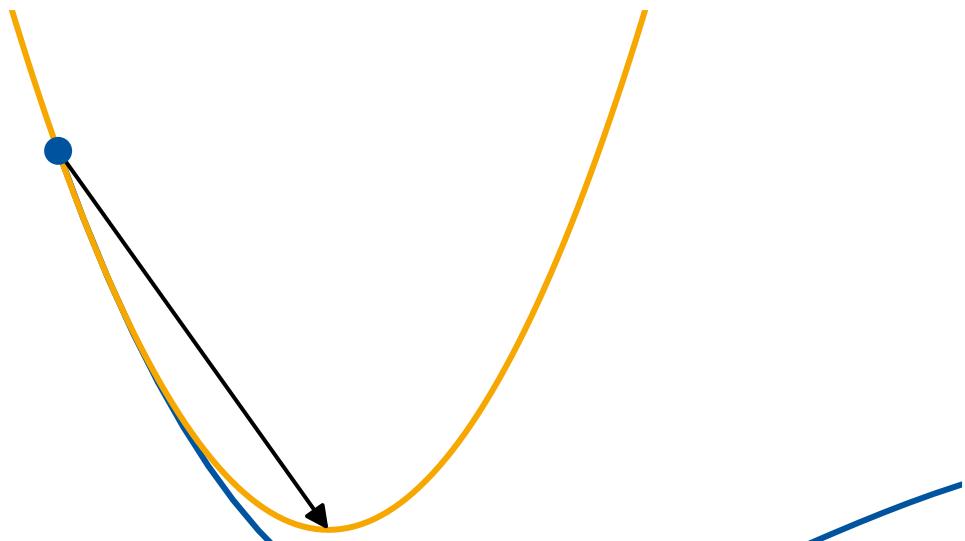
- Simple iterative optimization scheme with a familiar update rule ([Delta rule](#)).

Limitations

- Slow convergence
- Need to choose a suitable learning rate.

Logistic Regression

1. Logistic Regression Formulation
2. Motivation and Background
3. Iterative Optimization
4. First-Order Gradient Descent
- 5. Second-Order Gradient Descent**
6. Error Function Analysis



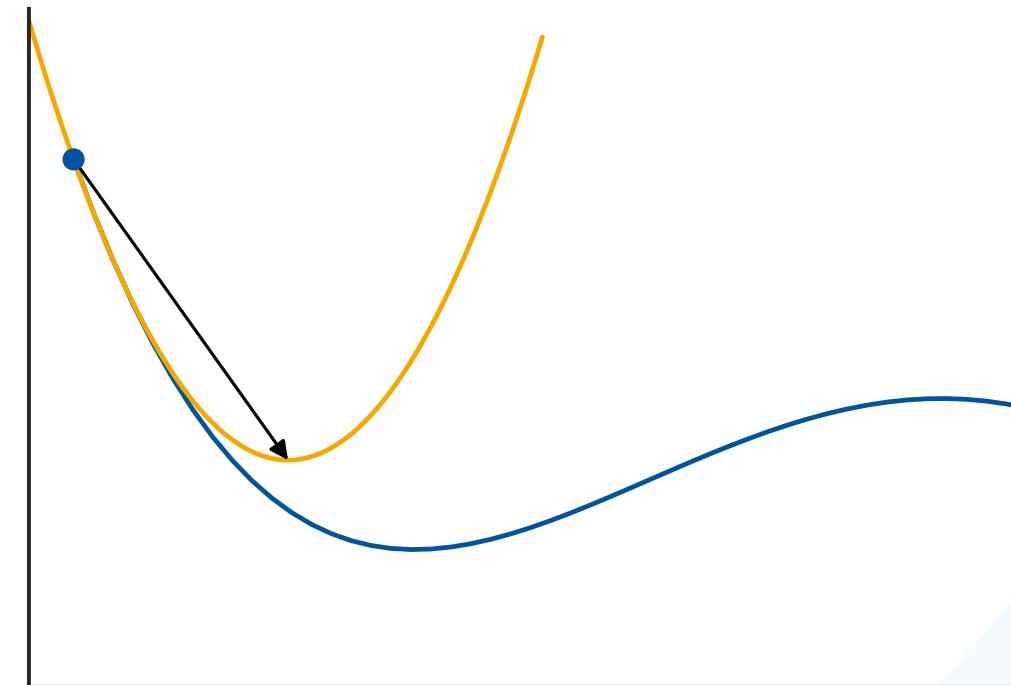
Second-Order Optimization

- So far, we have optimized the cross-entropy error with gradient descent:

$$\mathbf{w}^{(\tau+1)} = \mathbf{w}^{(\tau)} - \eta \nabla E(\mathbf{w})$$

- This is a first-order approximation, and it heavily depends on the learning rate η .

- Instead, we can apply a second-order optimization scheme that converges faster and is independent of the learning rate.



Newton-Raphson Gradient Descent

- Second-order Newton-Raphson update scheme:

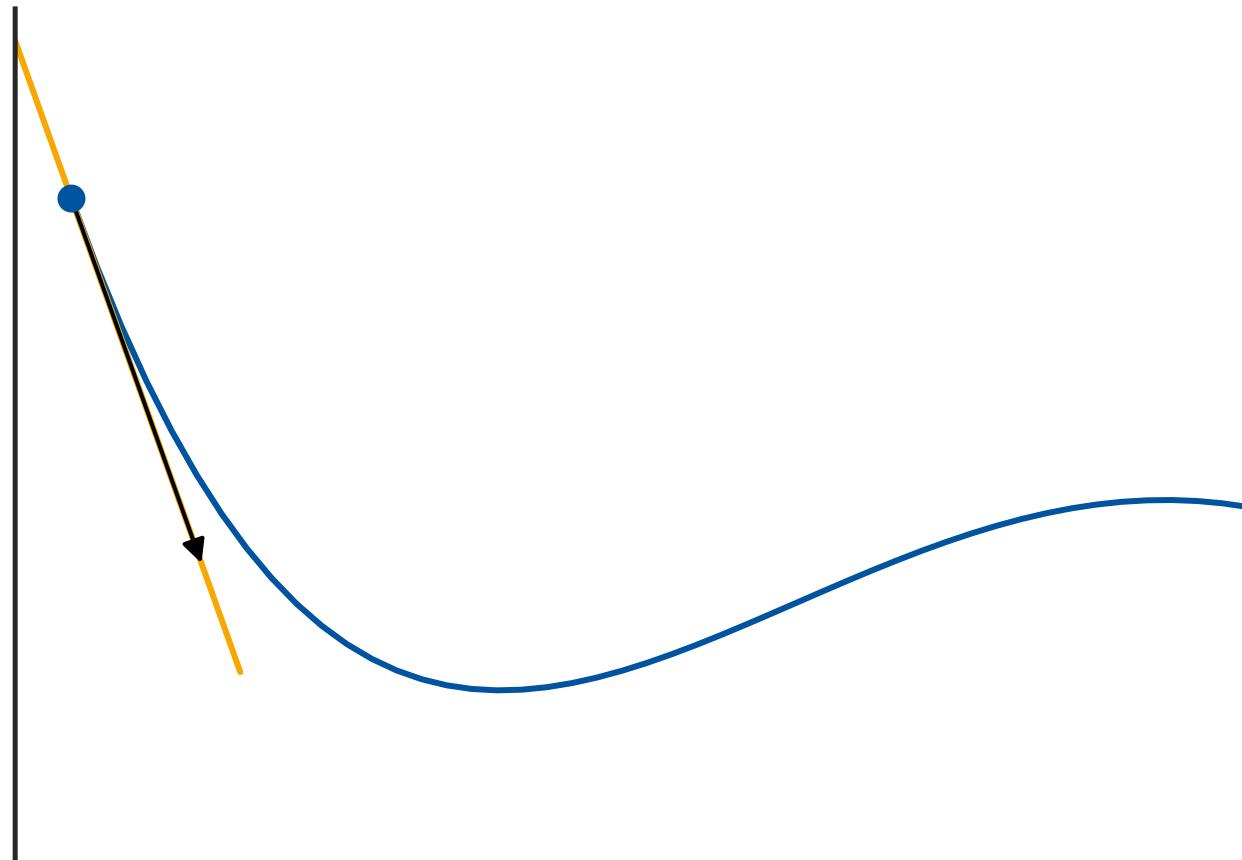
$$\mathbf{w}^{(\tau+1)} = \mathbf{w}^{(\tau)} - \mathbf{H}^{-1} \nabla E(\mathbf{w})$$

- Here, $\mathbf{H} = \nabla \nabla E(\mathbf{w})$ is the Hessian matrix, i.e., the matrix of second derivatives:

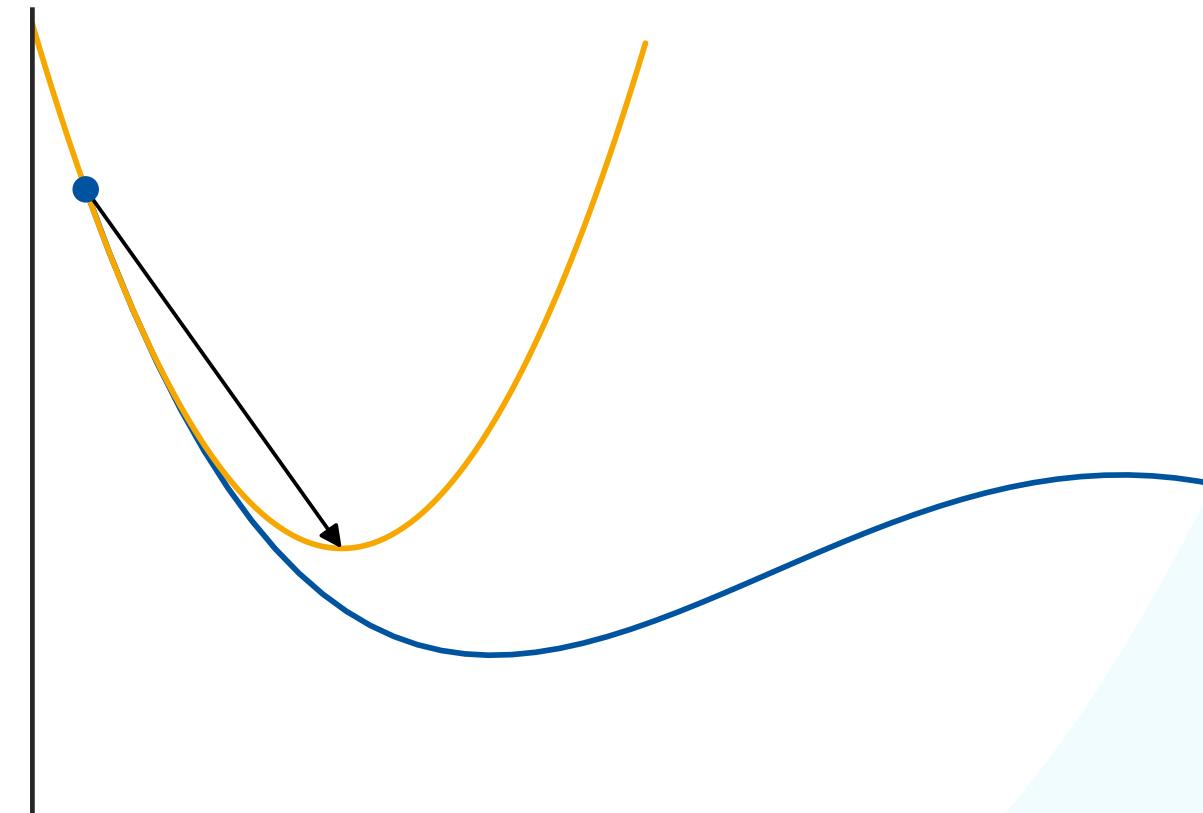
$$\mathbf{H}_{ij} = \frac{\partial^2 E(\mathbf{w})}{\partial w_i \partial w_j}$$

- Properties
 - Local quadratic approximation
 - Much faster convergence by taking into account the curvature of the error function.

Intuition

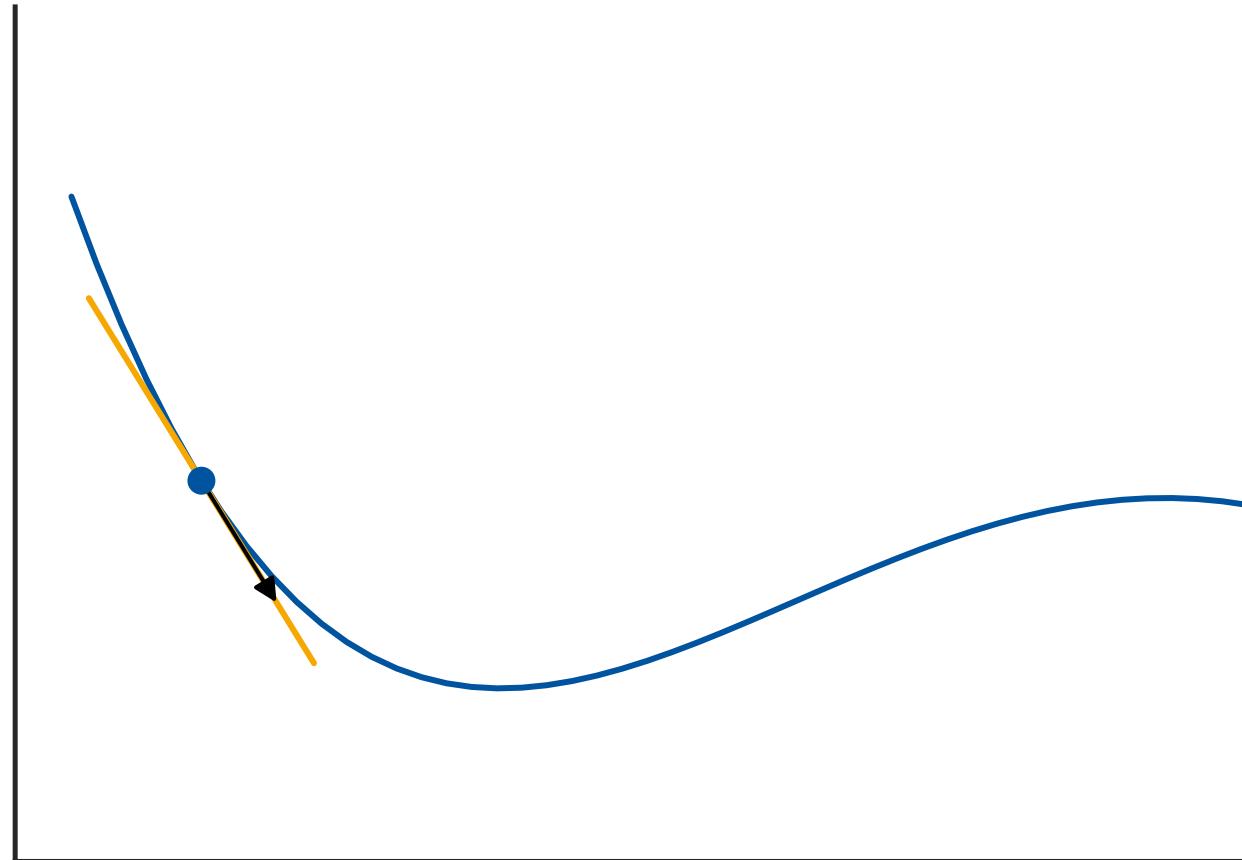


First-Order
 $\eta = 0.005$

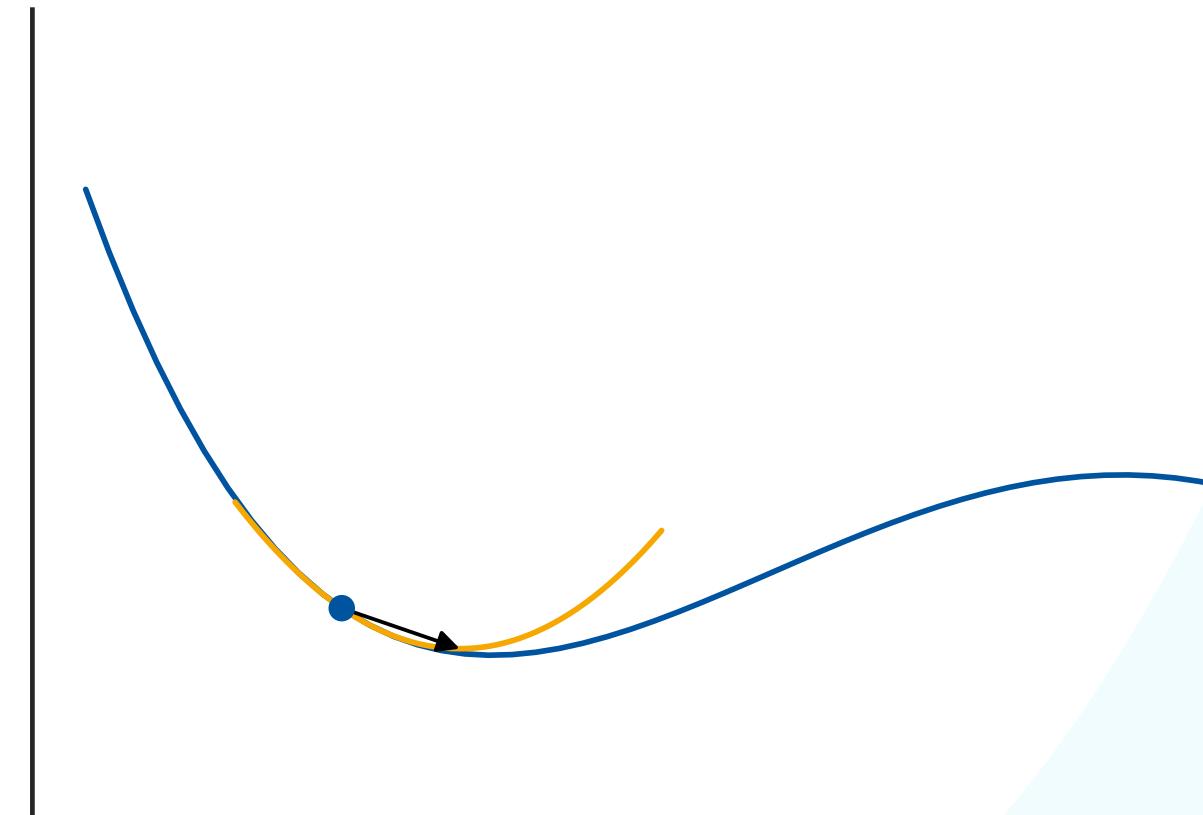


Second-Order

Intuition

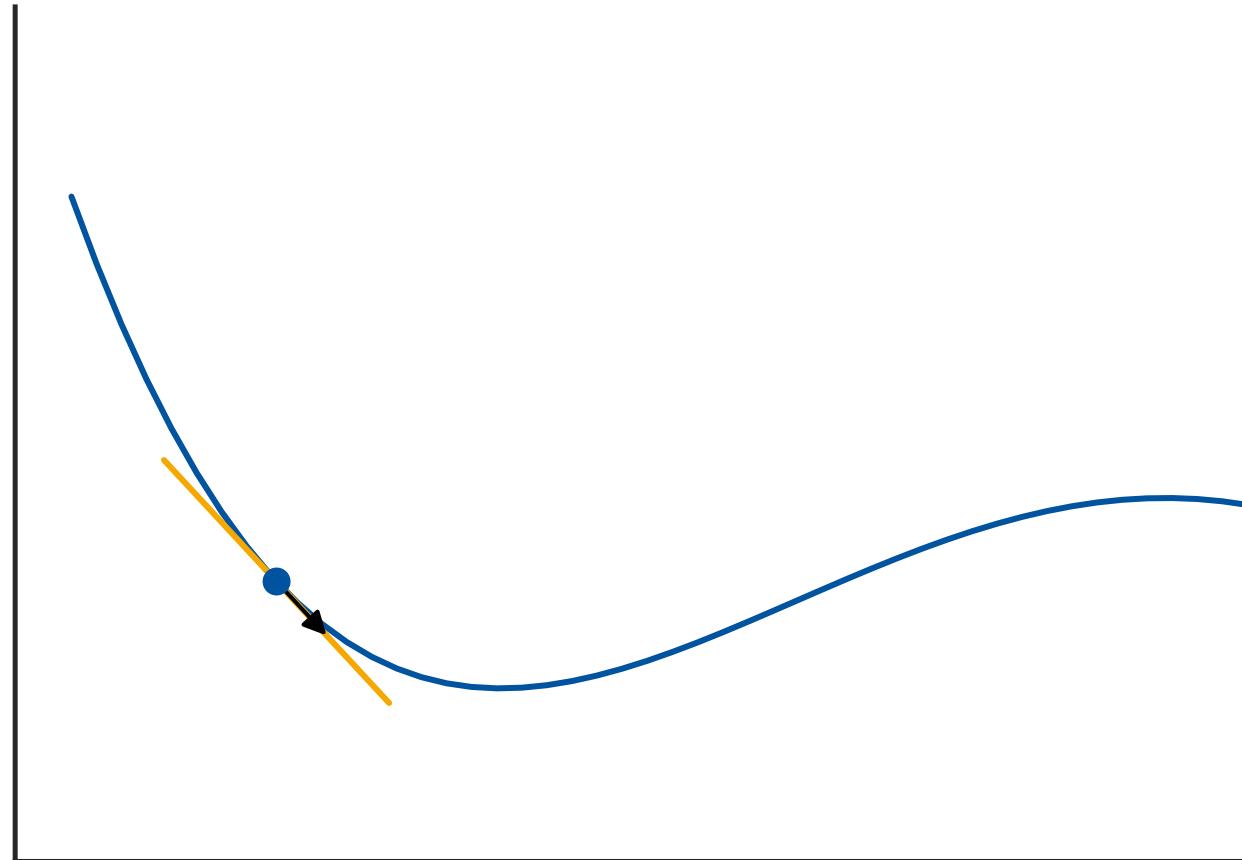


First-Order
 $\eta = 0.005$

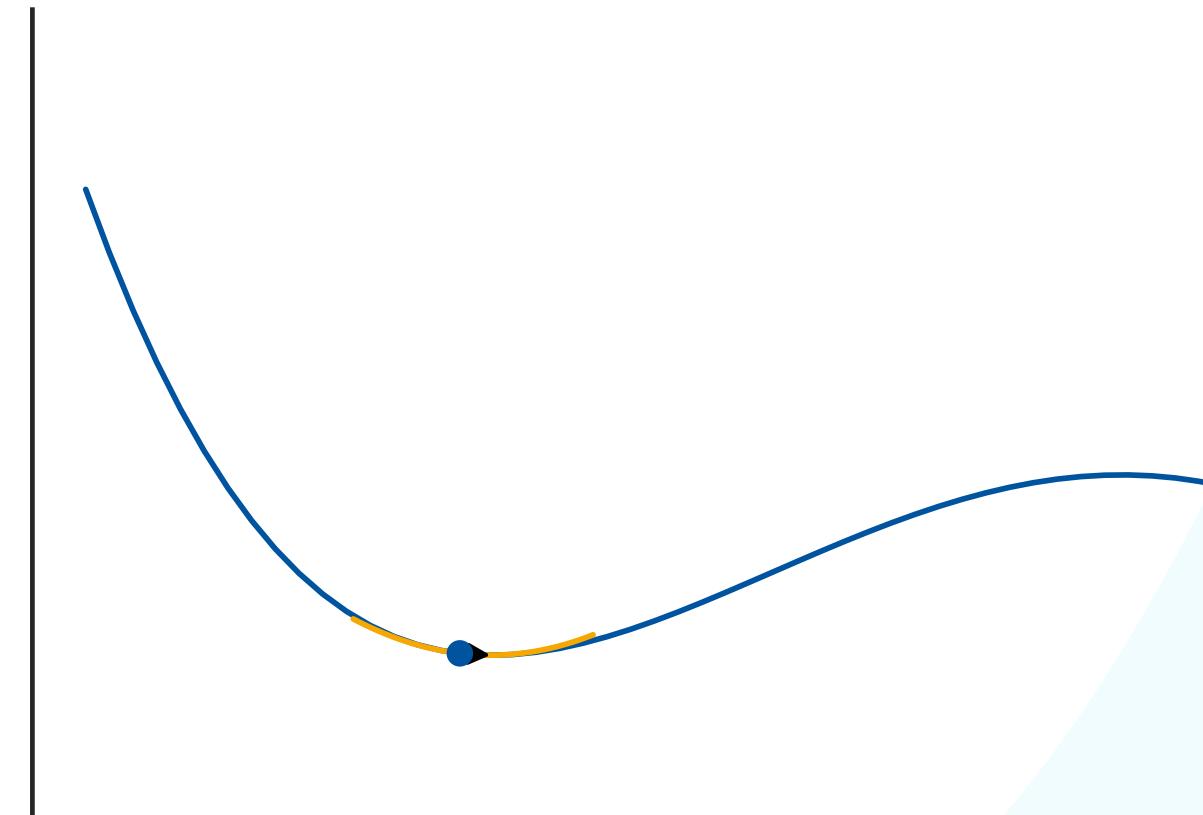


Second-Order

Intuition

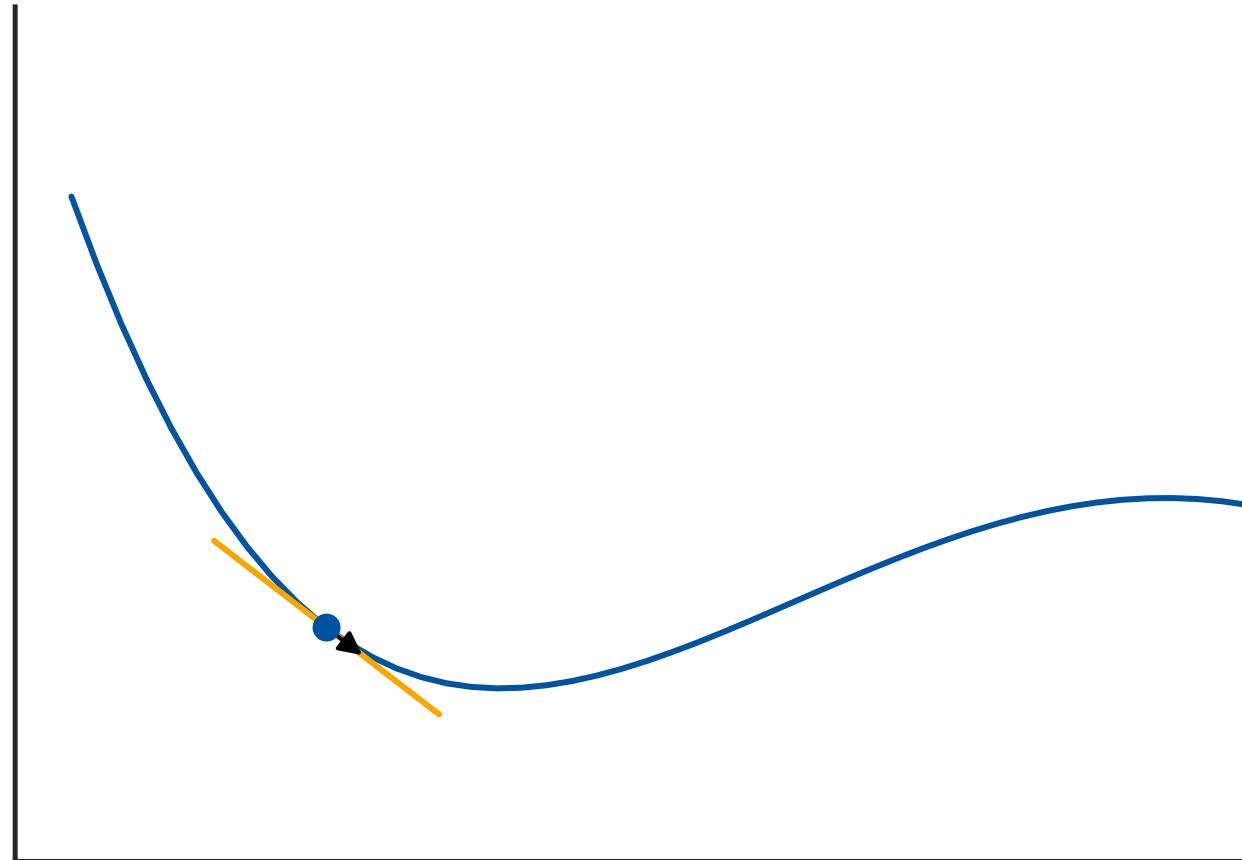


First-Order
 $\eta = 0.005$

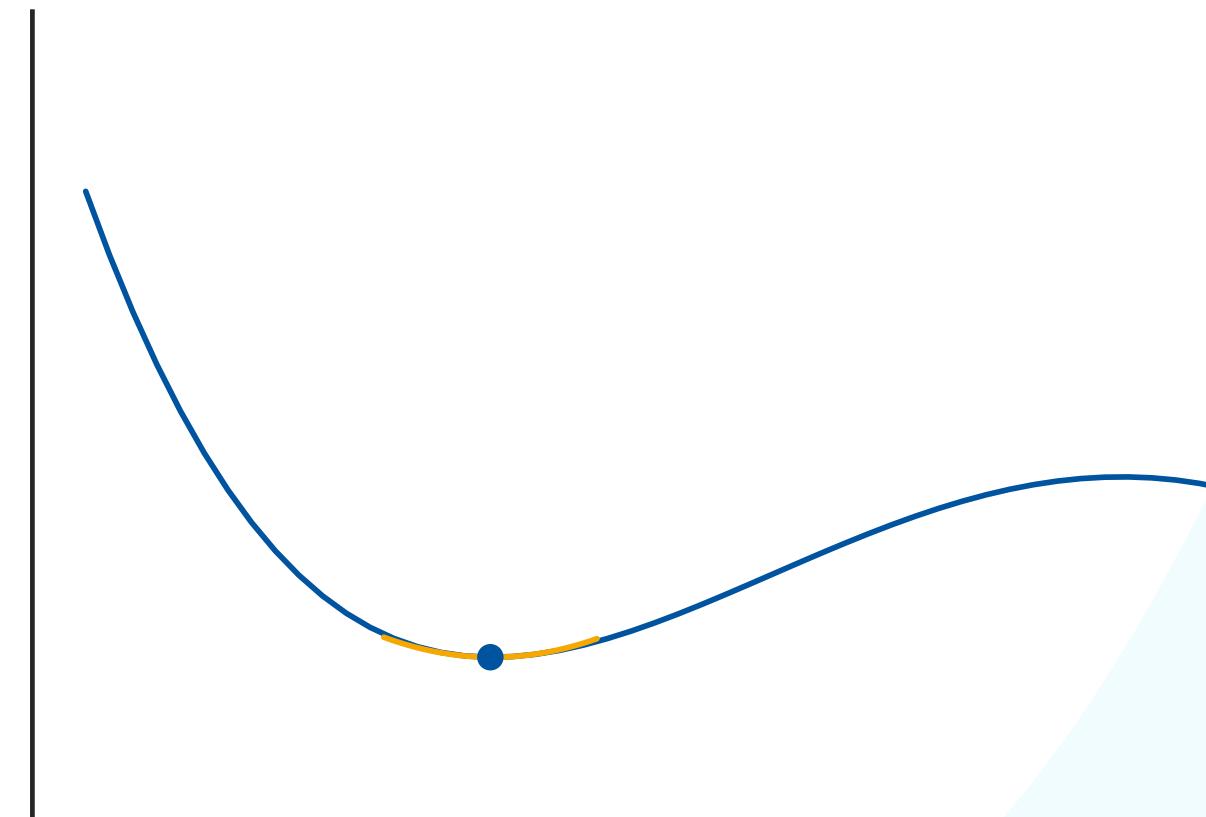


Second-Order

Intuition



First-Order
 $\eta = 0.005$



Second-Order

Intuition

*First-Order needs
another 16 steps...*

First-Order
 $\eta = 0.005$

Second-Order

Newton-Raphson for Least-Squares

- First, we apply it to least-squares:

$$E(\mathbf{w}) = \frac{1}{2} \sum_{n=1}^N (\mathbf{w}^\top \boldsymbol{\phi}_n - t_n)^2$$

$$\nabla E(\mathbf{w}) = \sum_{n=1}^N (\mathbf{w}^\top \boldsymbol{\phi}_n - t_n) \boldsymbol{\phi}_n = \boldsymbol{\Phi}^\top \boldsymbol{\Phi} \mathbf{w} - \boldsymbol{\Phi}^\top \mathbf{t}$$

$$\mathbf{H} = \nabla \nabla E(\mathbf{w}) = \sum_{n=1}^N \boldsymbol{\phi}_n \boldsymbol{\phi}_n^\top = \boldsymbol{\Phi}^\top \boldsymbol{\Phi}$$

$$\boldsymbol{\Phi} = \begin{pmatrix} \boldsymbol{\phi}_1^\top \\ \vdots \\ \boldsymbol{\phi}_N^\top \end{pmatrix}$$

- Resulting update scheme ([normal equations](#)):

$$\begin{aligned} \mathbf{w}^{(\tau+1)} &= \mathbf{w}^{(\tau)} - (\boldsymbol{\Phi}^\top \boldsymbol{\Phi})^{-1} (\boldsymbol{\Phi}^\top \boldsymbol{\Phi} \mathbf{w}^{(\tau)} - \boldsymbol{\Phi}^\top \mathbf{t}) \\ &= (\boldsymbol{\Phi}^\top \boldsymbol{\Phi})^{-1} \boldsymbol{\Phi}^\top \mathbf{t} \end{aligned}$$

This is the closed-form solution of the least-squares objective!

Newton-Raphson for the Cross-Entropy Error

- Now, let's try Newton-Raphson on the cross-entropy error function:

$$E(\mathbf{w}) = \frac{1}{2} \sum_{n=1}^N (t_n \ln y_n + (1 - t_n) \ln(1 - y_n))$$

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

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

$$\sigma'(a) = \sigma(a)(1 - \sigma(a))$$

$$\frac{\partial y_n}{\partial \mathbf{w}} = y_n(1 - y_n) \phi_n$$

- where $\mathbf{R} \in \mathbb{R}^{N \times N}$ is an $N \times N$ diagonal matrix with $R_{nn} = y_n(1 - y_n)$.
- The Hessian now depends on \mathbf{w} through the weighting matrix \mathbf{R} .

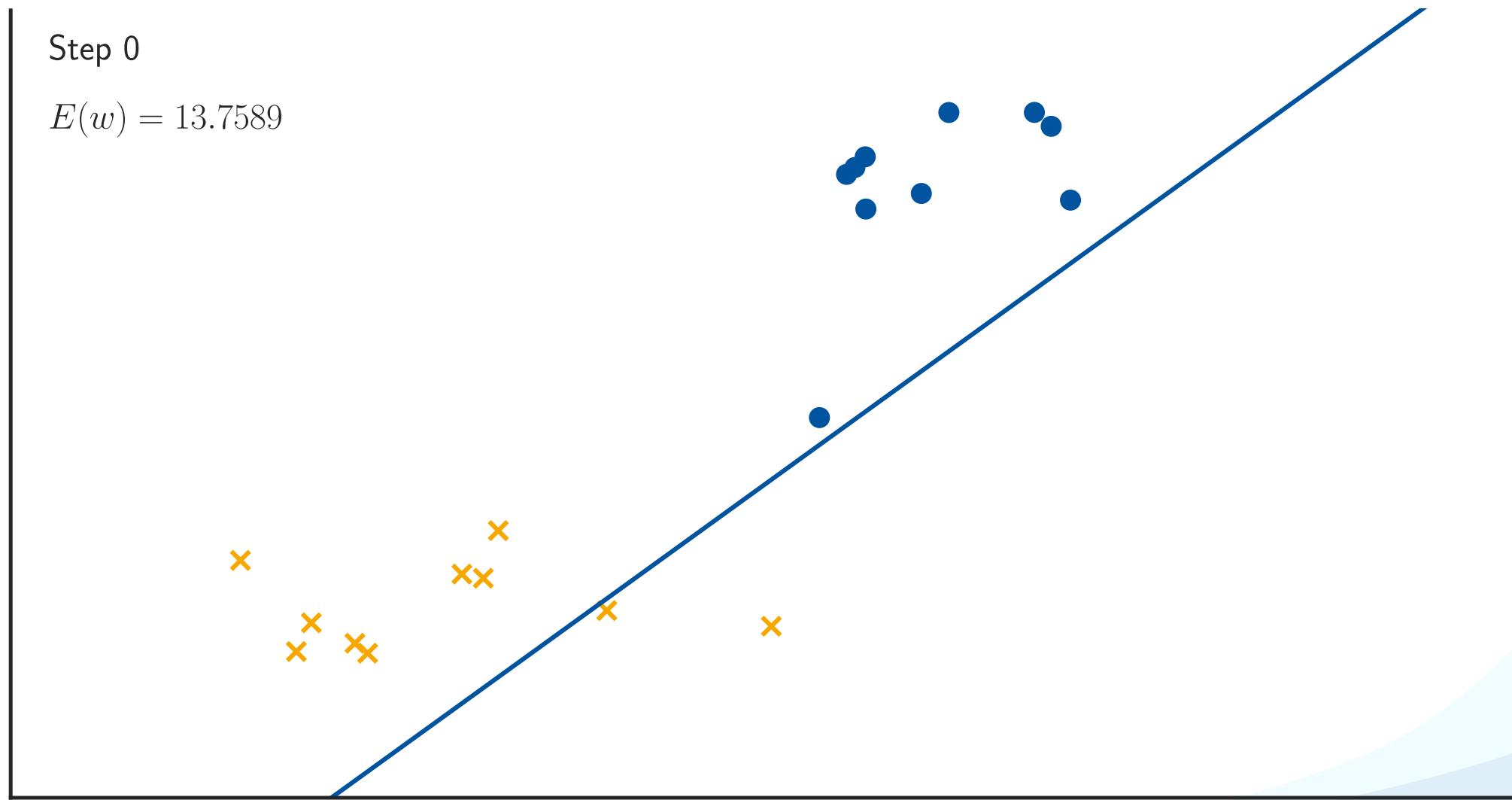
Iteratively Reweighted Least Squares (IRLS)

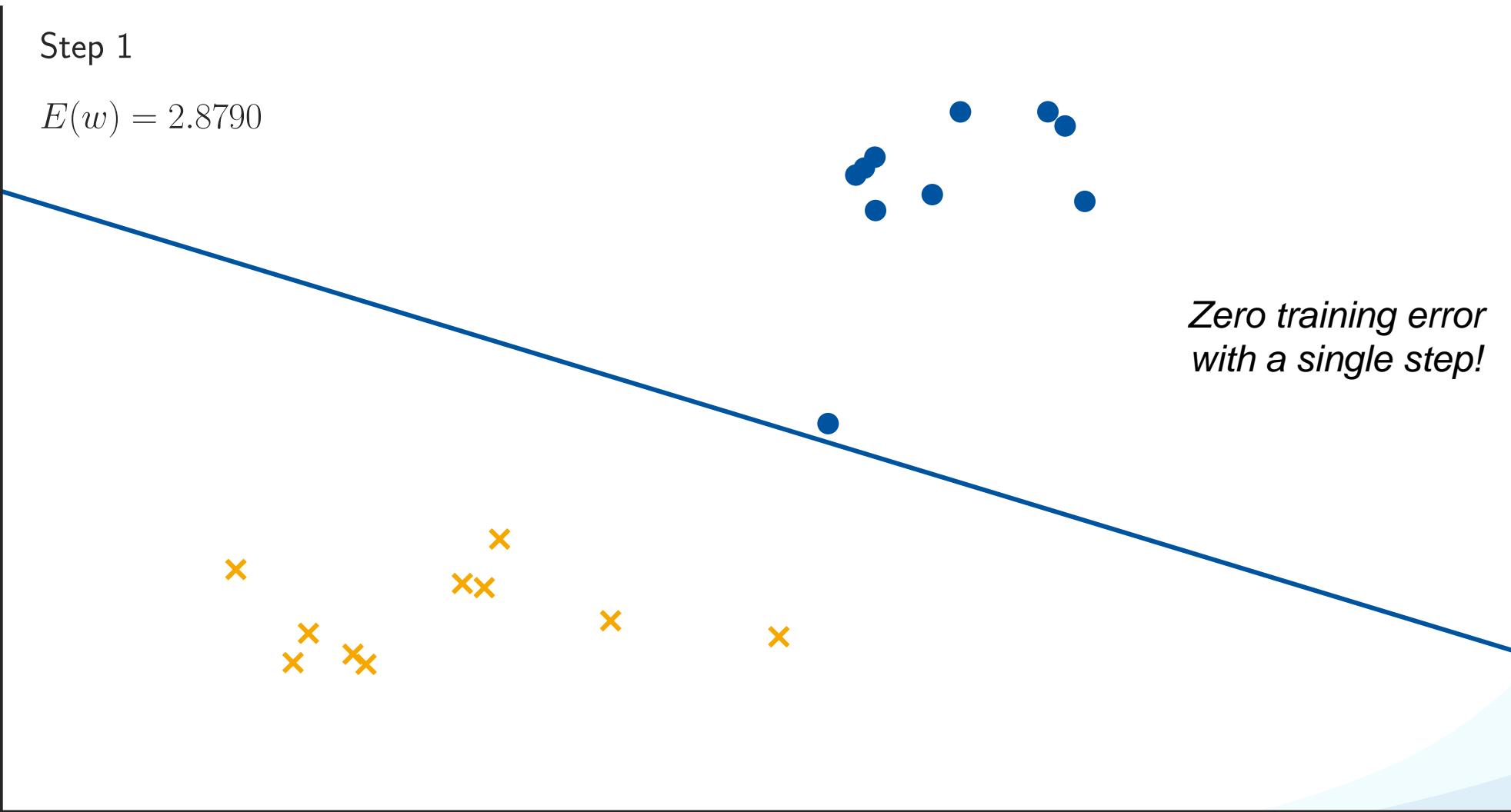
- Update equations:

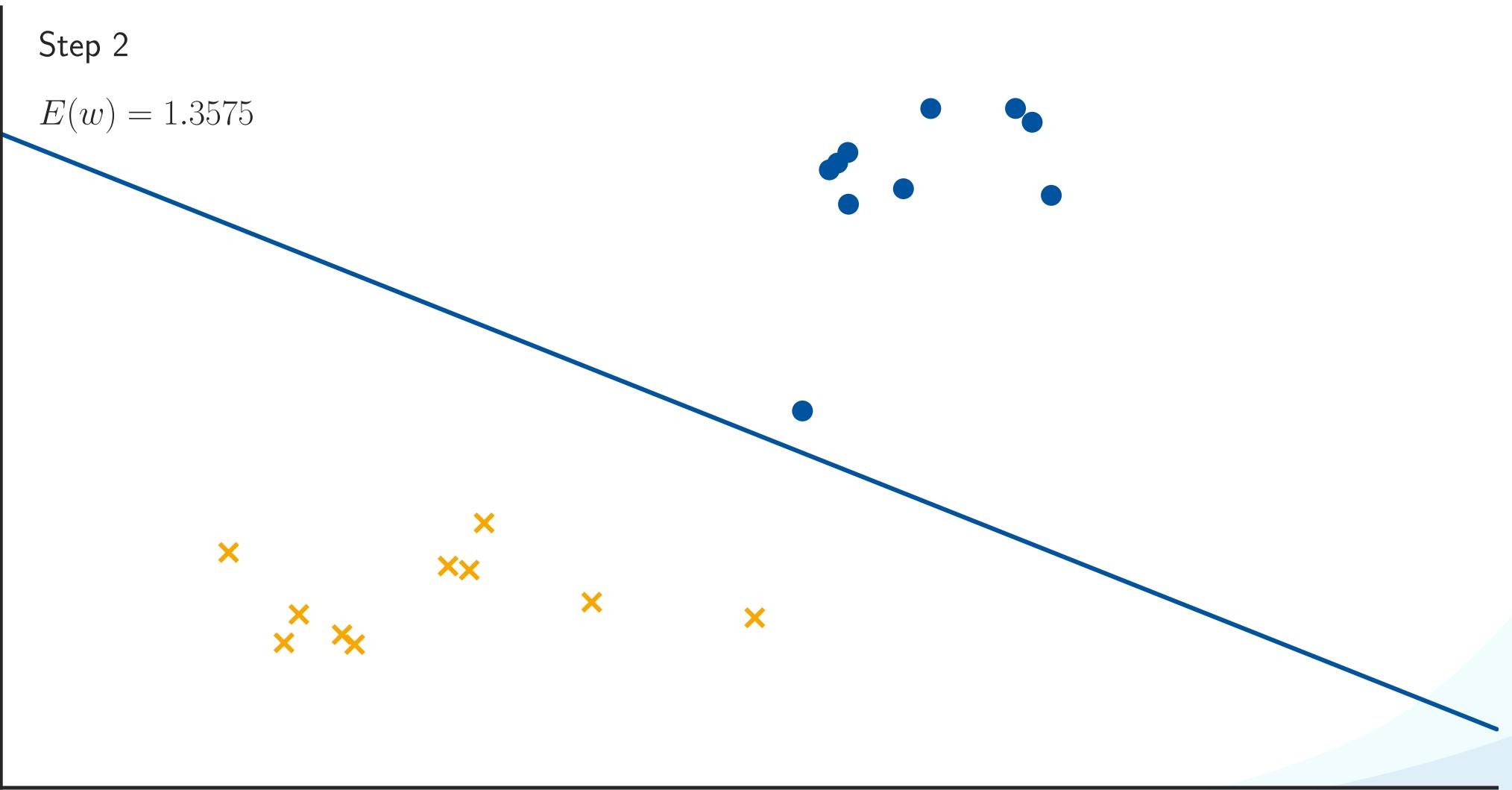
$$\begin{aligned}\mathbf{w}^{(\tau+1)} &= \mathbf{w}^{(\tau)} - (\Phi^T \mathbf{R} \Phi)^{-1} \Phi^T (\mathbf{y} - \mathbf{t}) \\ &= (\Phi^T \mathbf{R} \Phi)^{-1} \left(\Phi^T \mathbf{R} \Phi \mathbf{w}^{(\tau)} - \Phi^T (\mathbf{y} - \mathbf{t}) \right) \\ &= (\Phi^T \mathbf{R} \Phi)^{-1} \Phi^T \mathbf{R} \mathbf{z} \\ \text{with } \mathbf{z} &= \Phi \mathbf{w}^{(\tau)} - \mathbf{R}^{-1} (\mathbf{y} - \mathbf{t})\end{aligned}$$

- Very similar form (normal equations).
 - But now with non-constant weighting matrix \mathbf{R} (depends on \mathbf{w}).
 - Need to apply normal equations iteratively.
 - This is called Iteratively Reweighted Least-Squares (IRLS).

Example: Logistic Regression with IRLS

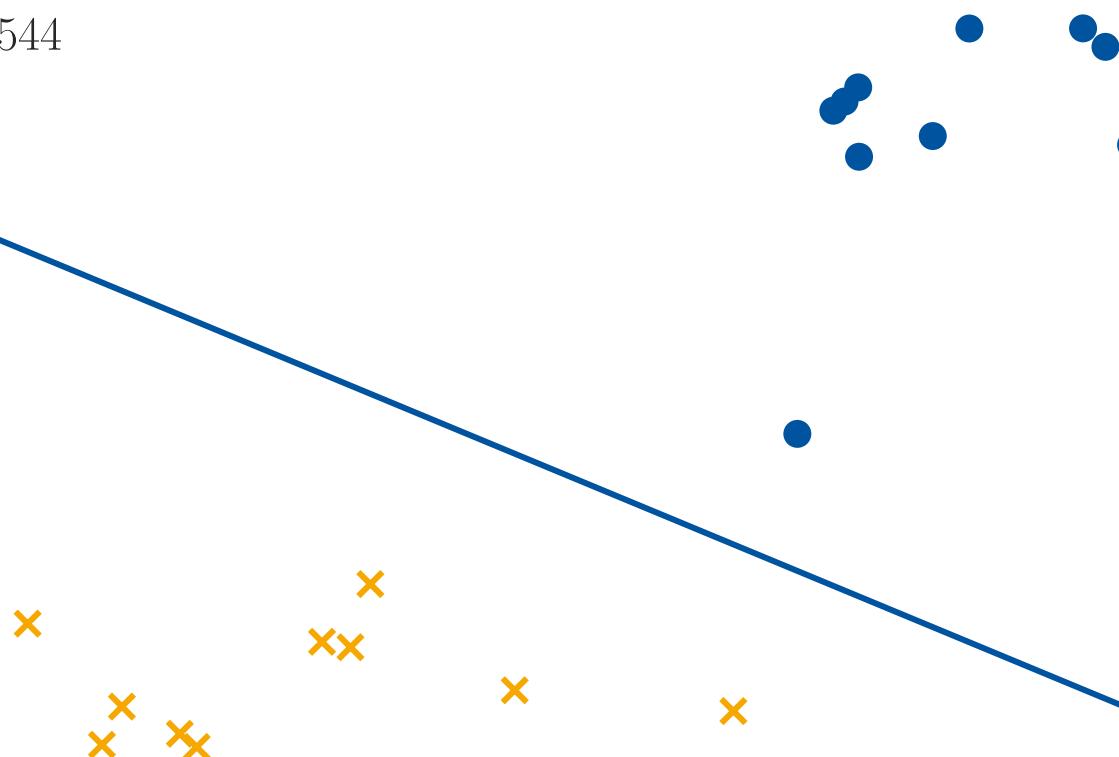






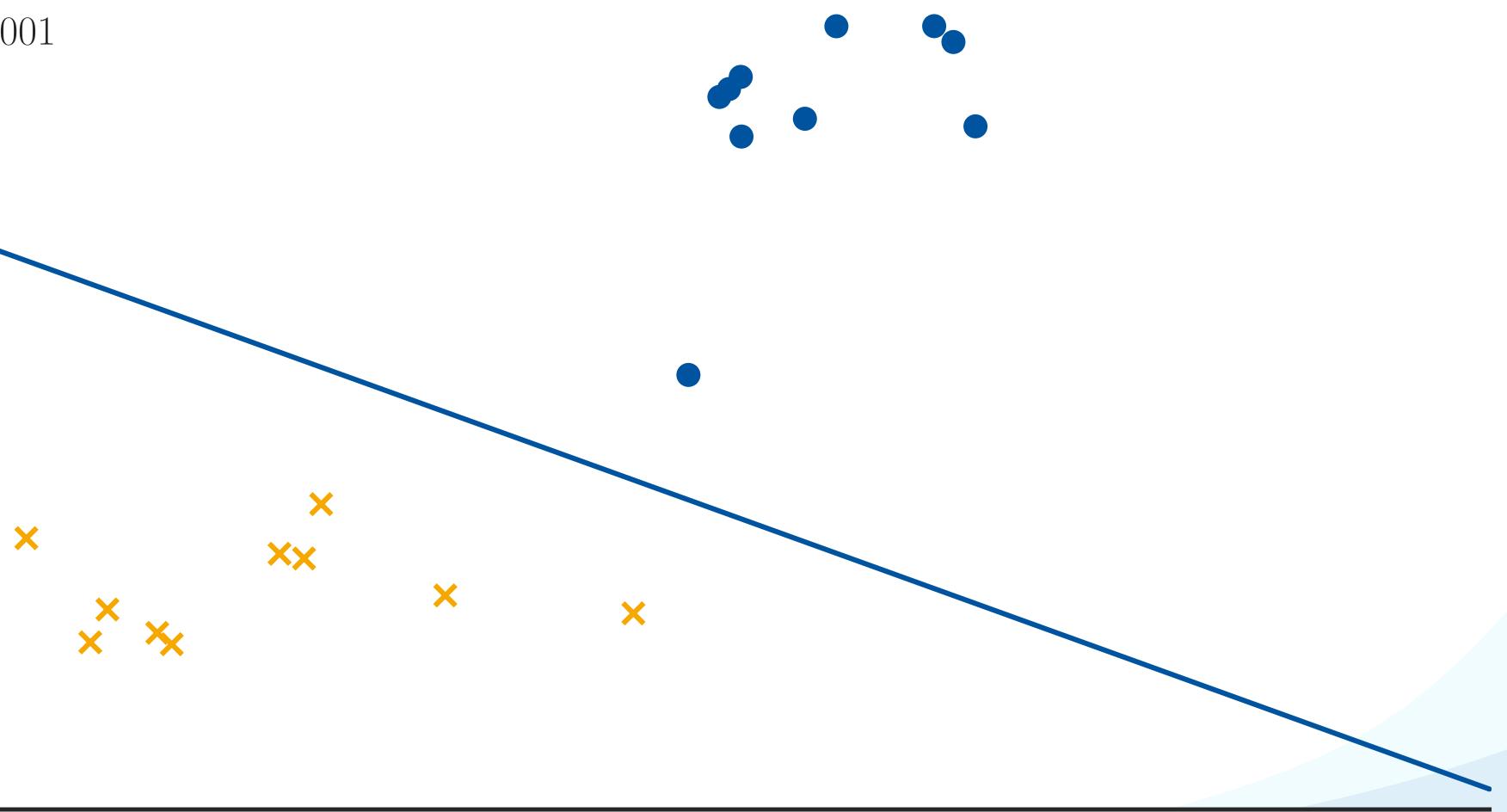
Step 6

$$E(w) = 0.0544$$



Step 12

$$E(w) = 0.0001$$



Discussion: Second-Order Optimization

Advantages

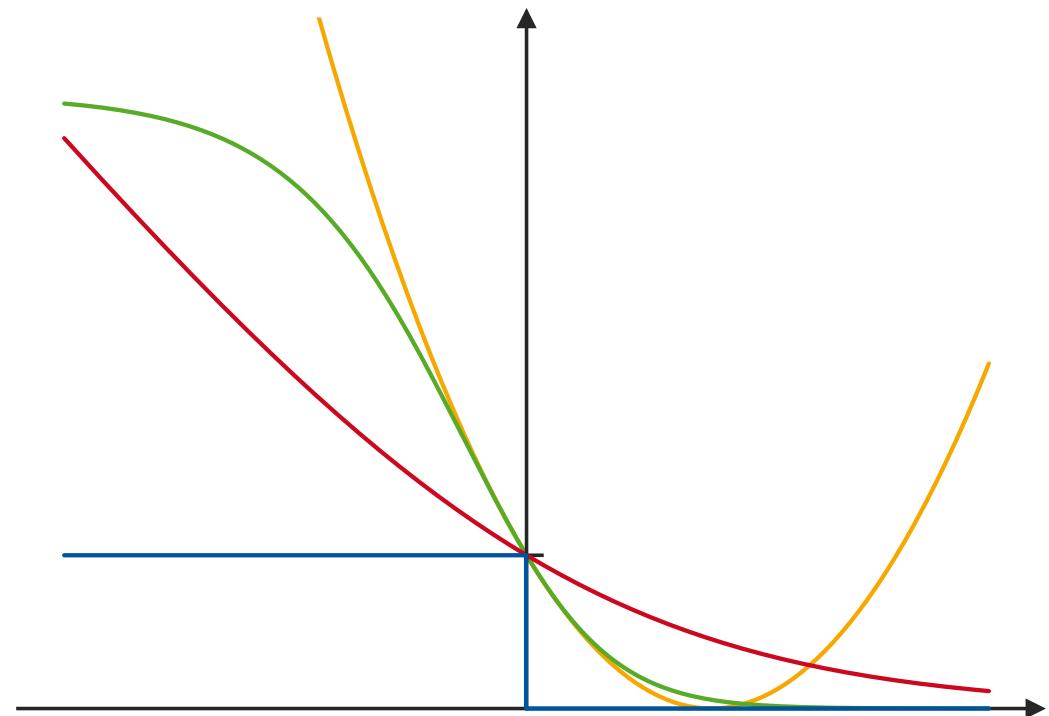
- Faster convergence than first-order methods

Limitations

- Second-order approach, relies on computing second derivatives.
- Computing (and inverting) the Hessian matrix is expensive for problems with many parameters.

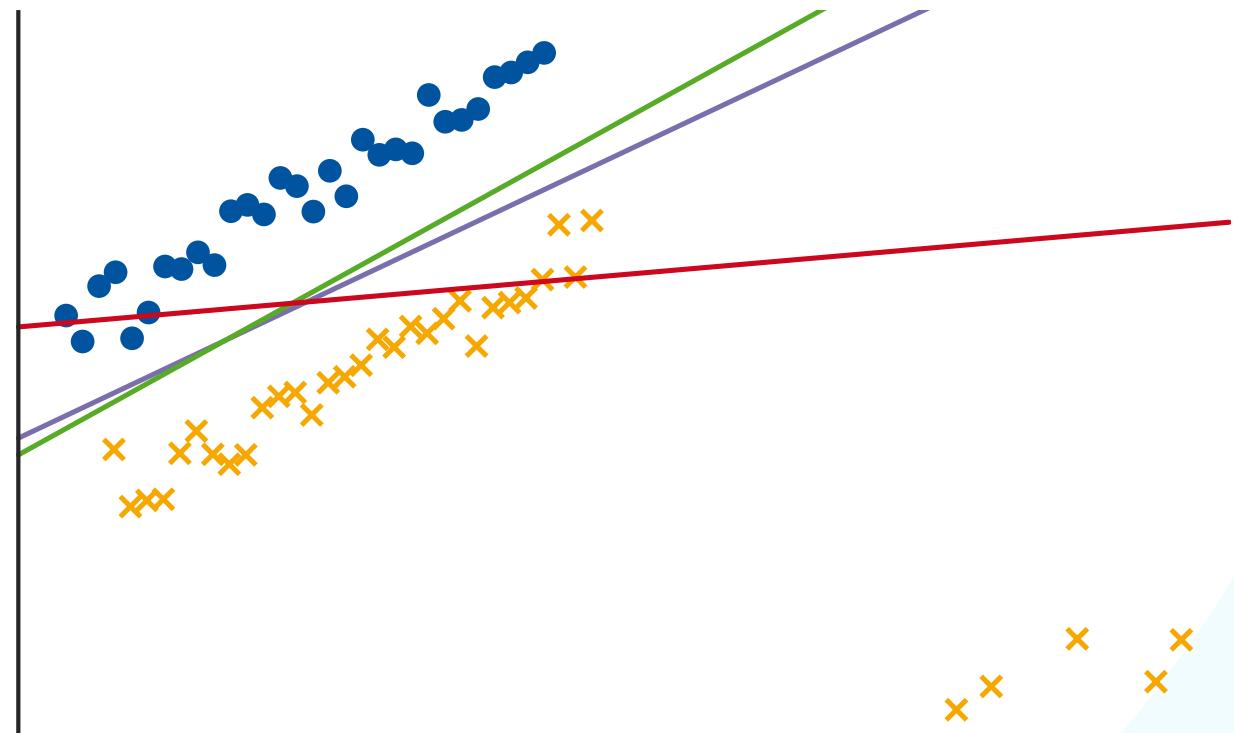
Logistic Regression

1. Logistic Regression Formulation
2. Motivation and Background
3. Iterative Estimation
4. First-Order Gradient Descent
5. Second-Order Gradient Descent
6. **Error Function Analysis**

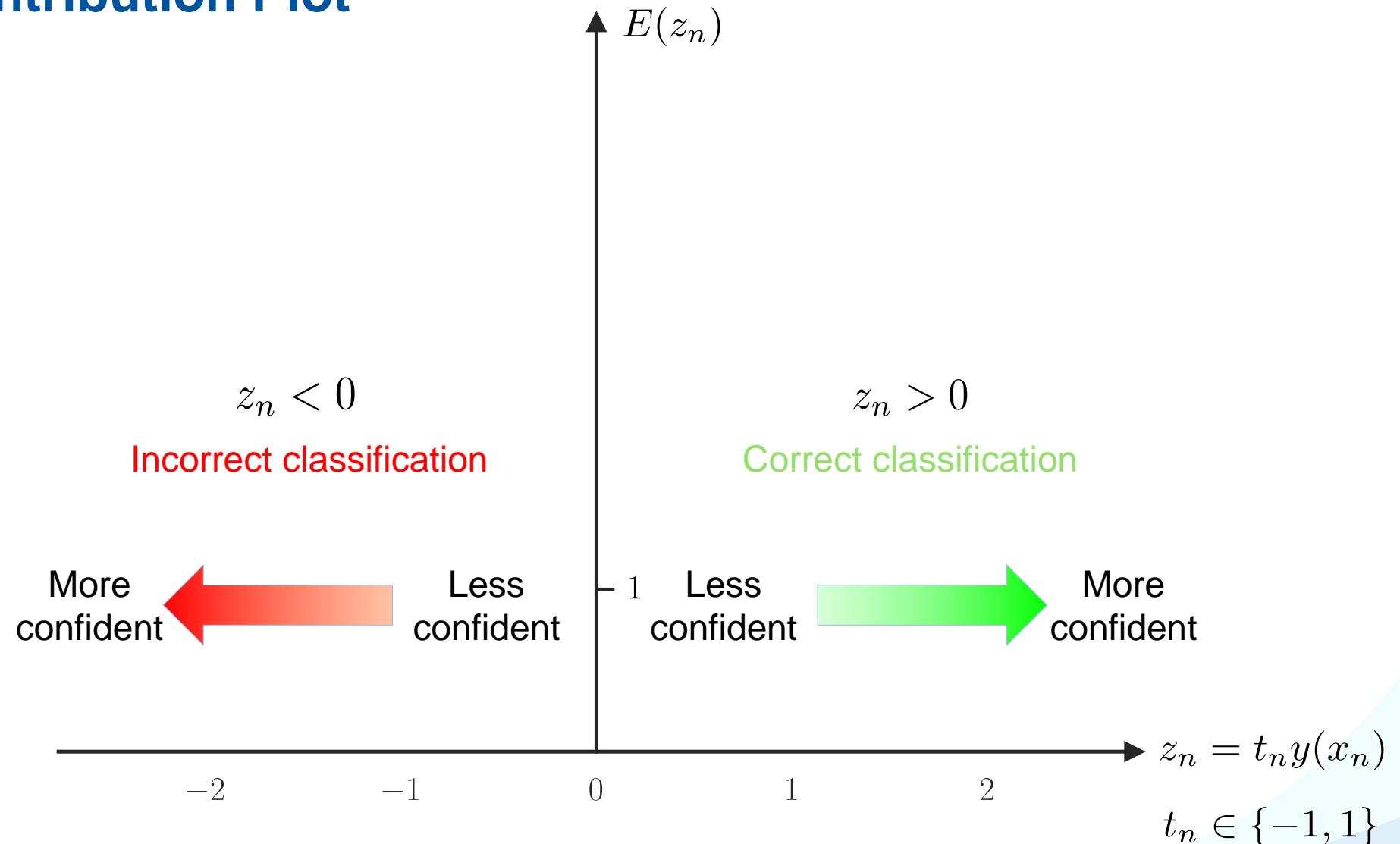


Error Function Analysis

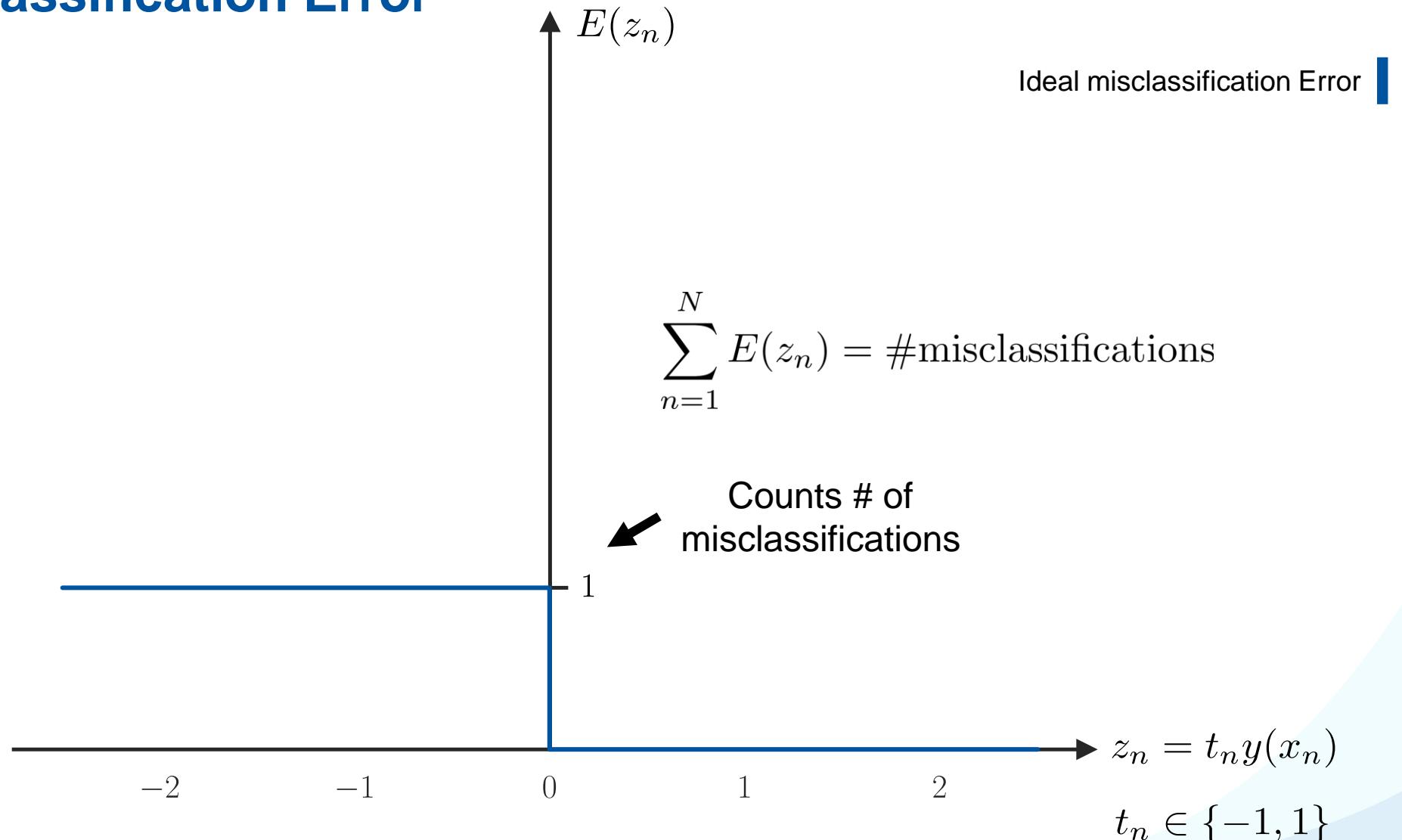
- We have seen how to learn **generalized linear discriminant** models by optimizing an error function.
 - We observed problems with **least-squares classification** based on the squared error function.
 - We have seen that **logistic regression** behaves more robustly.
- *Let's analyze the cross-entropy error in more detail...*



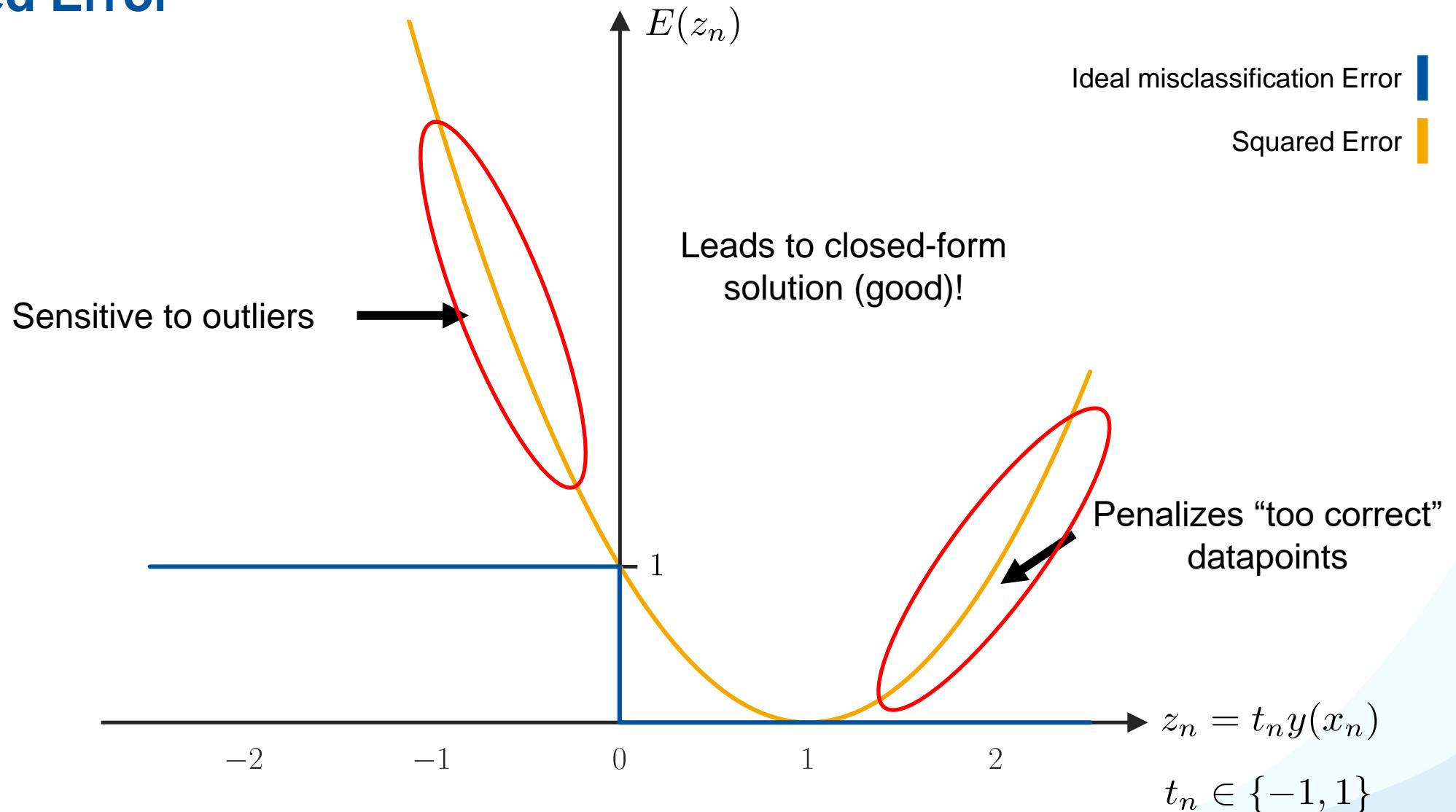
Error Contribution Plot



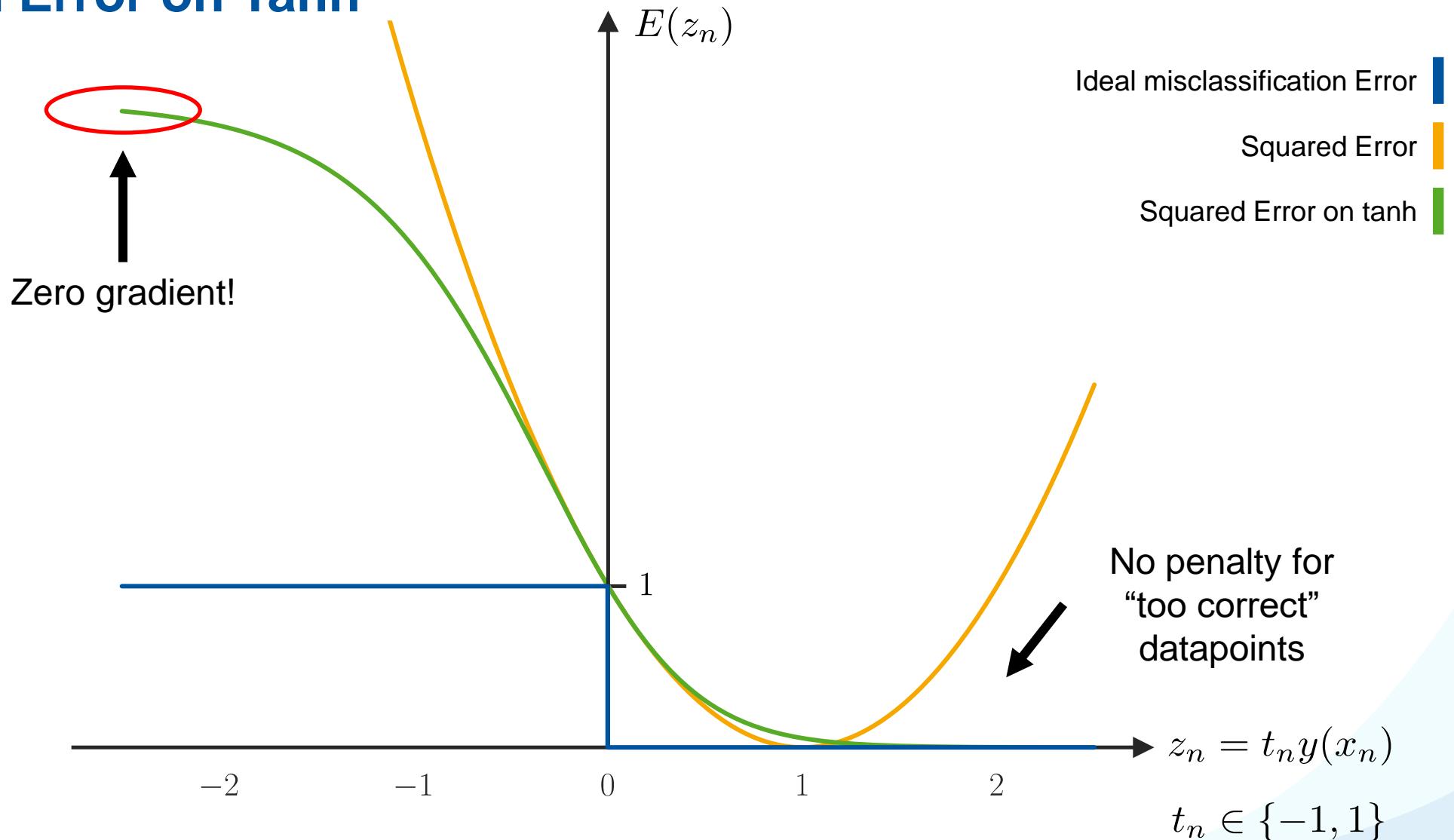
Ideal Misclassification Error



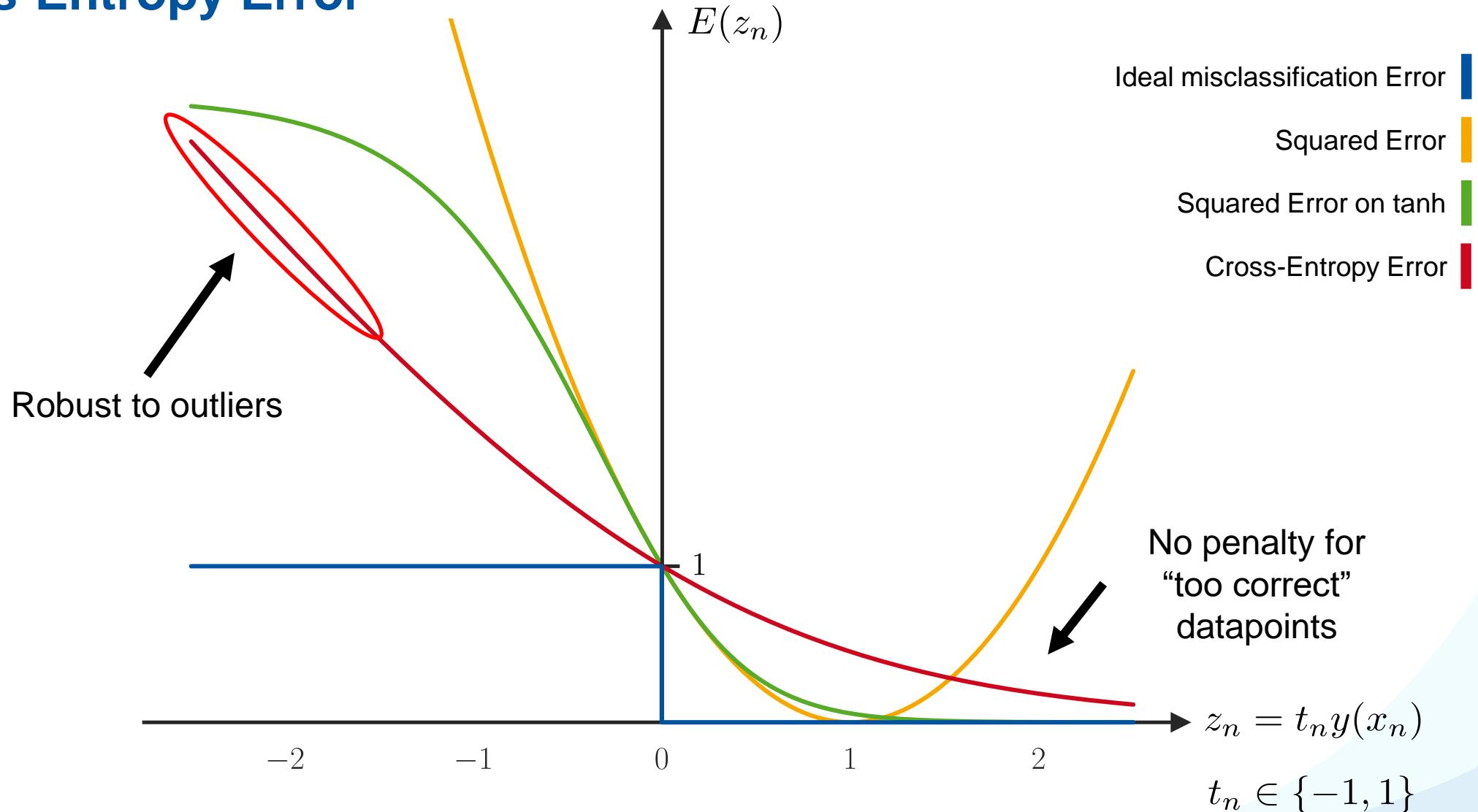
Squared Error



Squared Error on Tanh



Cross-Entropy Error



Discussion: Cross-Entropy Error

Advantages

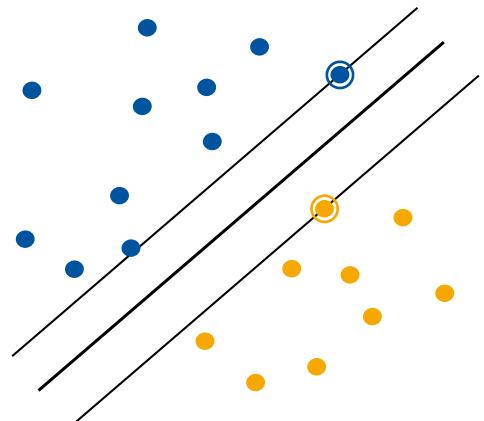
- Minimizer of this error corresponds to class posteriors
- Convex function, unique minimum exists
- Robust to outliers

Limitations

- No closed-form solution, requires iterative estimation

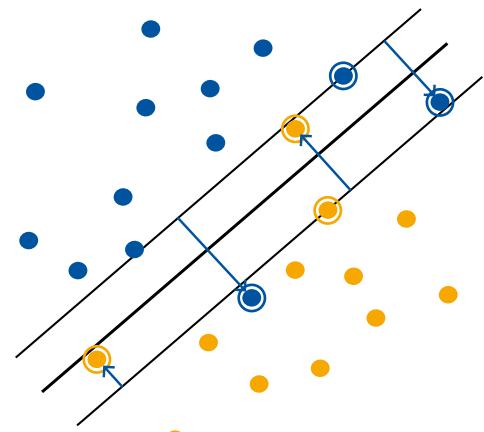
Machine Learning Topics

1. Introduction to ML
2. Probability Density Estimation
3. Linear Discriminants
4. Linear Regression
5. Logistic Regression
- 6. Support Vector Machines**
7. AdaBoost
8. Neural Network Basics

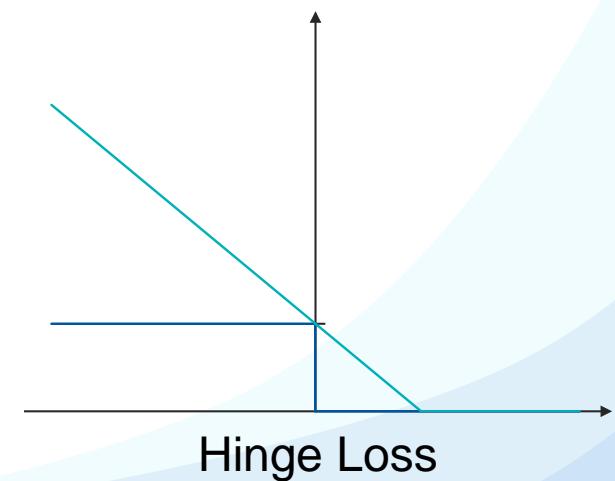


$$L_p(\mathbf{w}, b, \mathbf{a})$$
$$L_d(\mathbf{a})$$

Maximum Margin Classification



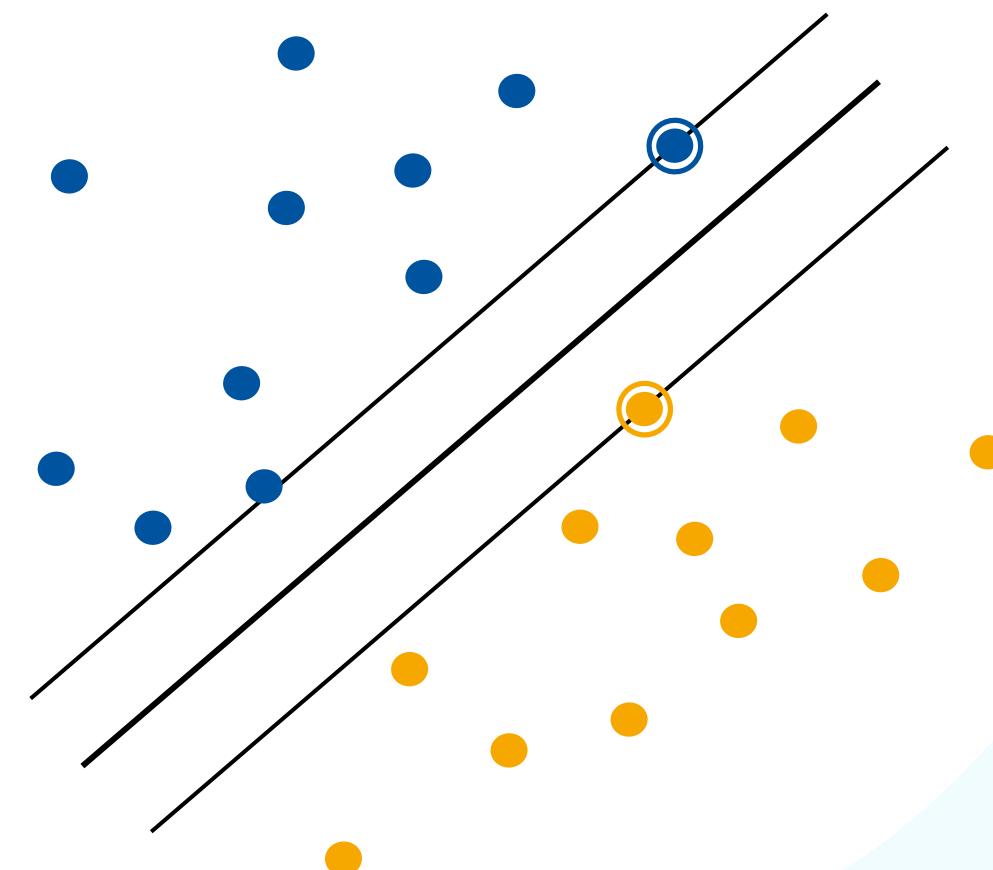
Primal & Dual Form



Soft-Margin SVM

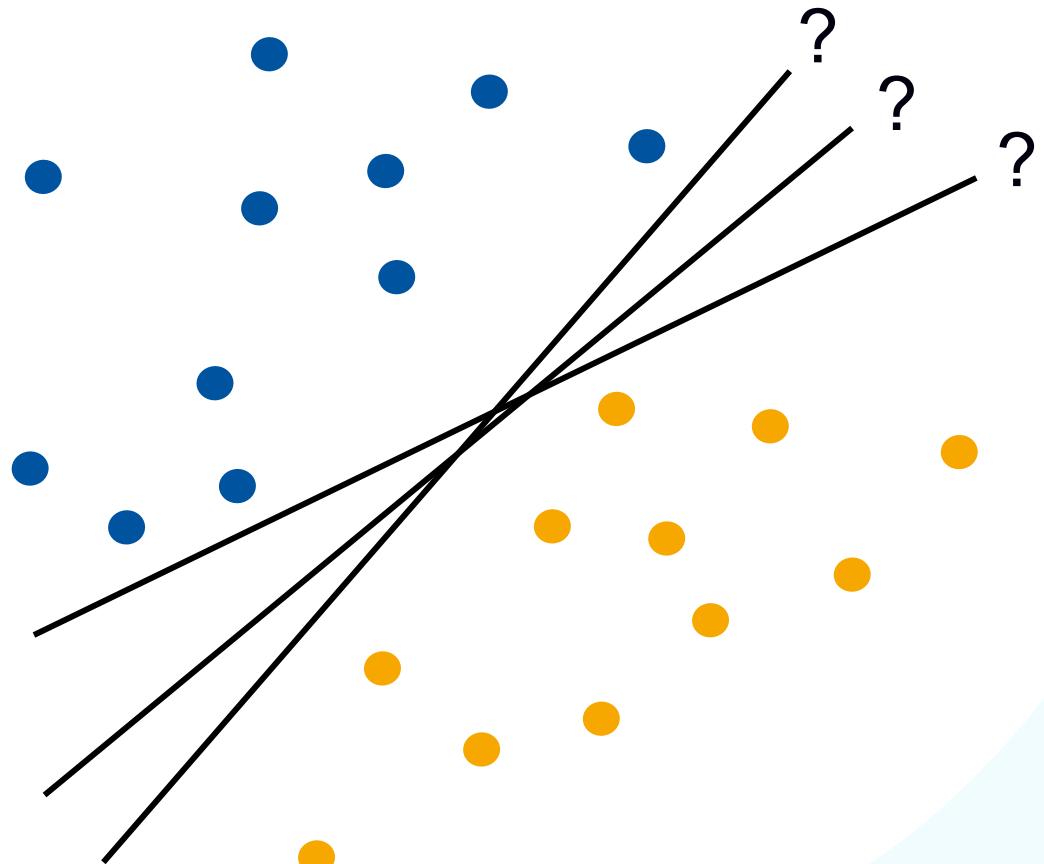
Support Vector Machines

1. Maximum Margin Classification
2. Primal Formulation
3. Dual Formulation
4. Soft-Margin SVMs
5. Non-linear SVMs
6. Error Function Analysis



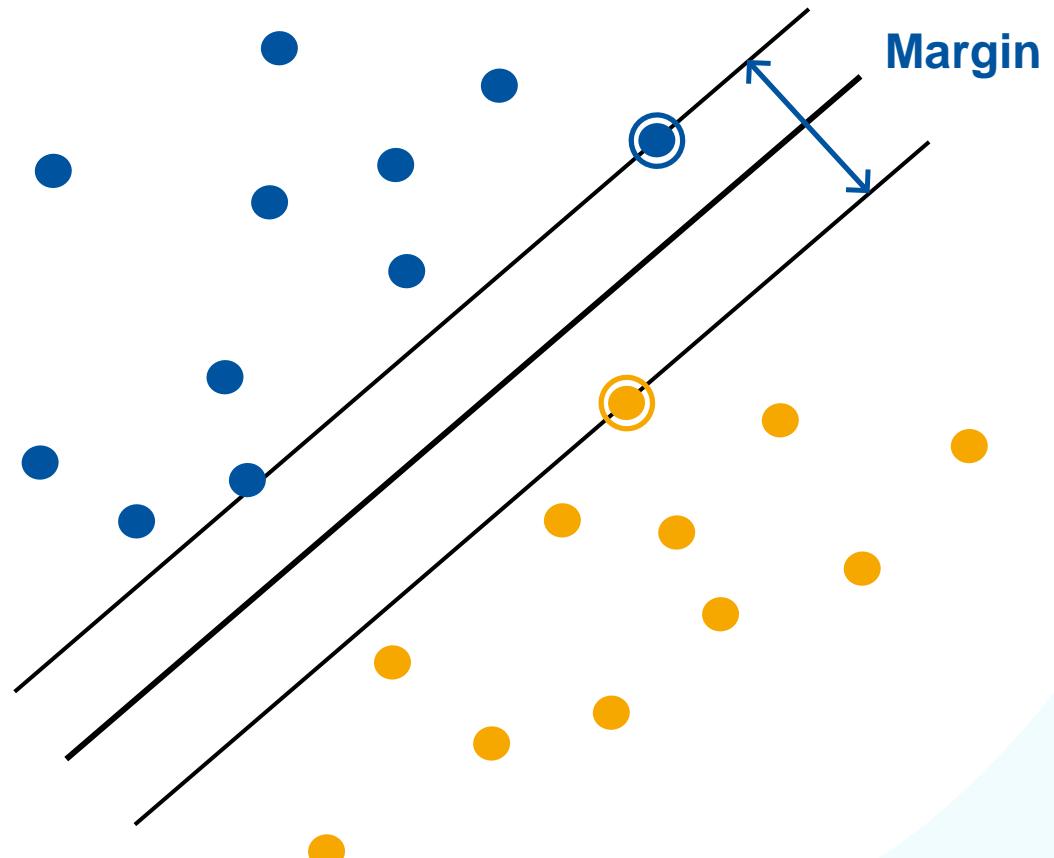
Maximum Margin Classification

- Overfitting is often a problem with linearly separable data
 - Which of the many possible decision boundaries is correct?
 - All of them have zero error on the training set...
 - However, they will perform differently on novel test data.
- *How can we select the classifier with the best generalization performance?*



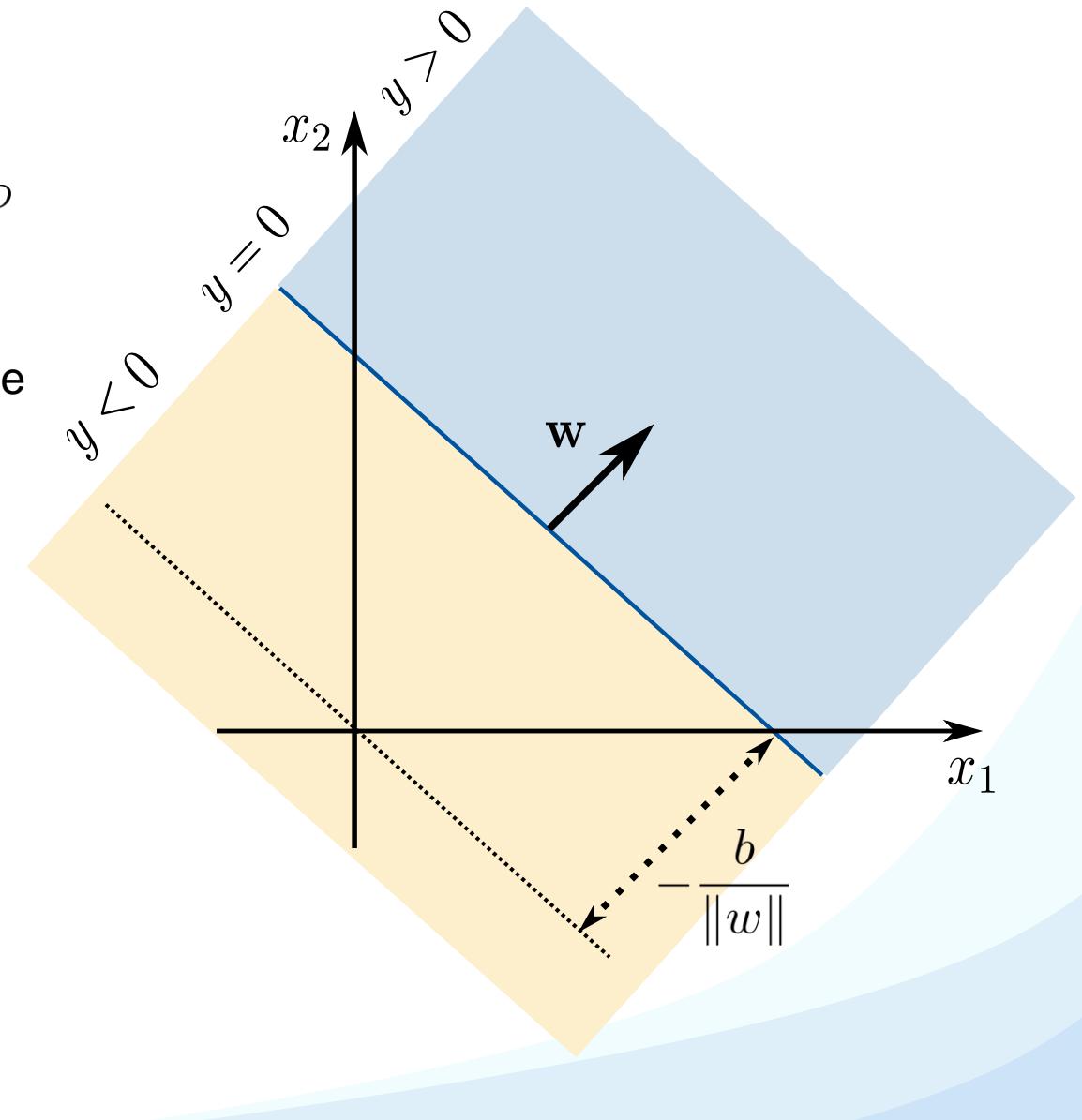
Maximum Margin Classification

- Intuitively, we want to choose the classifier which leaves maximal “safety room” for future data points.
- This classifier has the largest **margin** between positive and negative points.
- It can be shown: The larger the margin, the lower the capacity for overfitting.



Intuition

- Let's first consider linearly separable data:
 - N training data points $\{(\mathbf{x}_i, t_i)\}_{i=1}^N$, $\mathbf{x}_i \in \mathbb{R}^D$
 - Binary labels $t_i \in \{-1, 1\}$
- A linear discriminant function models a hyperplane separating the data:
$$y(\mathbf{x}) = \mathbf{w}^\top \mathbf{x} + b$$
- Note that we denote the bias explicitly with b .
- Decision rule
 - Decide for \mathcal{C}_1 if $y(\mathbf{x}) > 0$, else for \mathcal{C}_2 .



Support Vector Machines

- Assuming linearly separable data, we can always find a hyperplane with

$$\mathbf{w}^T \mathbf{x}_n + b \geq +1 \text{ for } t_n = +1$$

$$\mathbf{w}^T \mathbf{x}_n + b \leq -1 \text{ for } t_n = -1$$

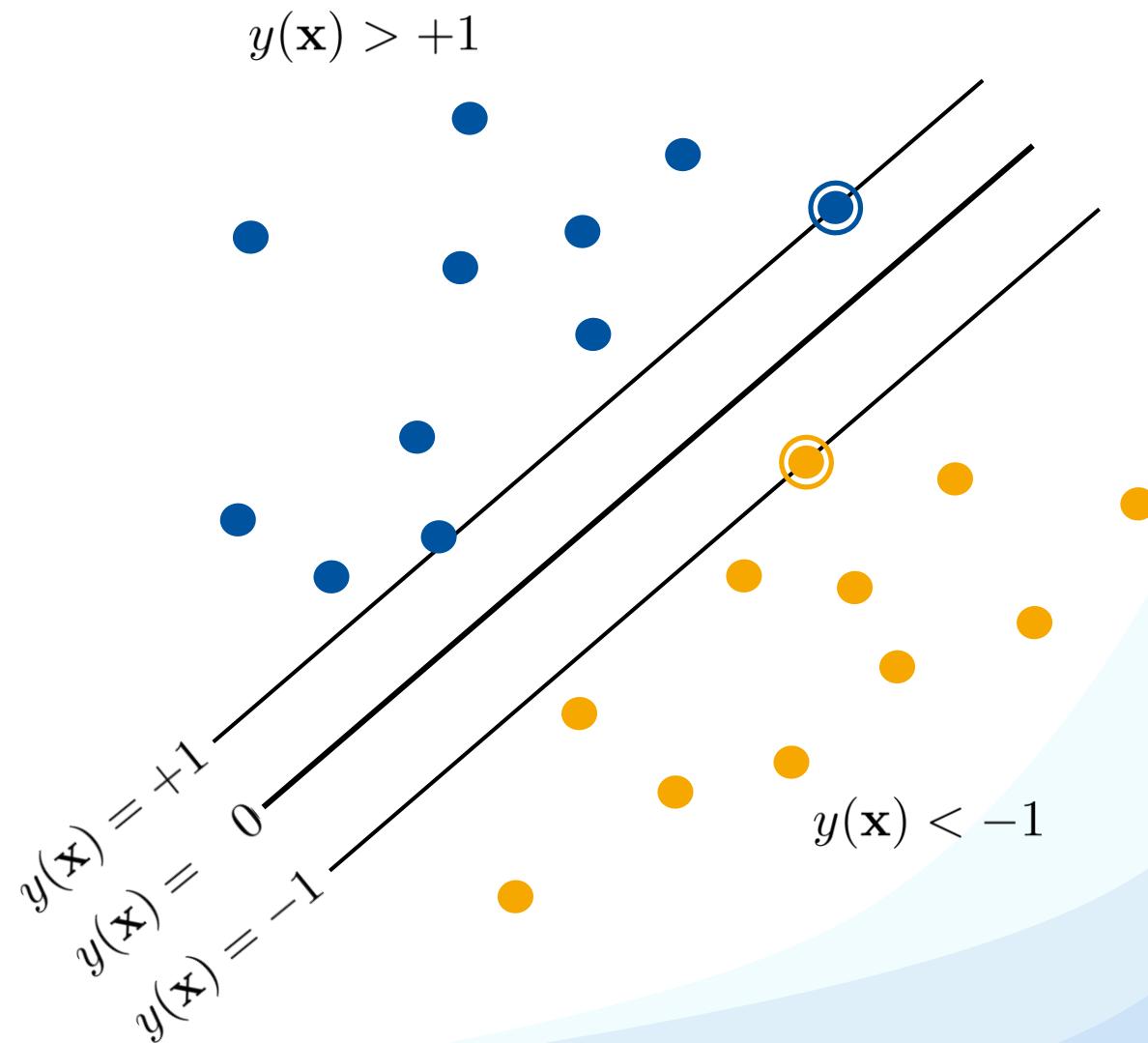
- In short:

$$t_n(\mathbf{w}^T \mathbf{x}_n + b) \geq 1 \quad \forall n$$

- We can rescale \mathbf{w} such that the equation holds exactly for the points on the margin:

$$t_n(\mathbf{w}^T \mathbf{x}_n + b) = 1$$

- There will be at least one such point on either side.



- We can choose \mathbf{w} such that

$$\mathbf{w}^T \mathbf{x}_n + b = +1 \text{ for one } t_n = +1$$

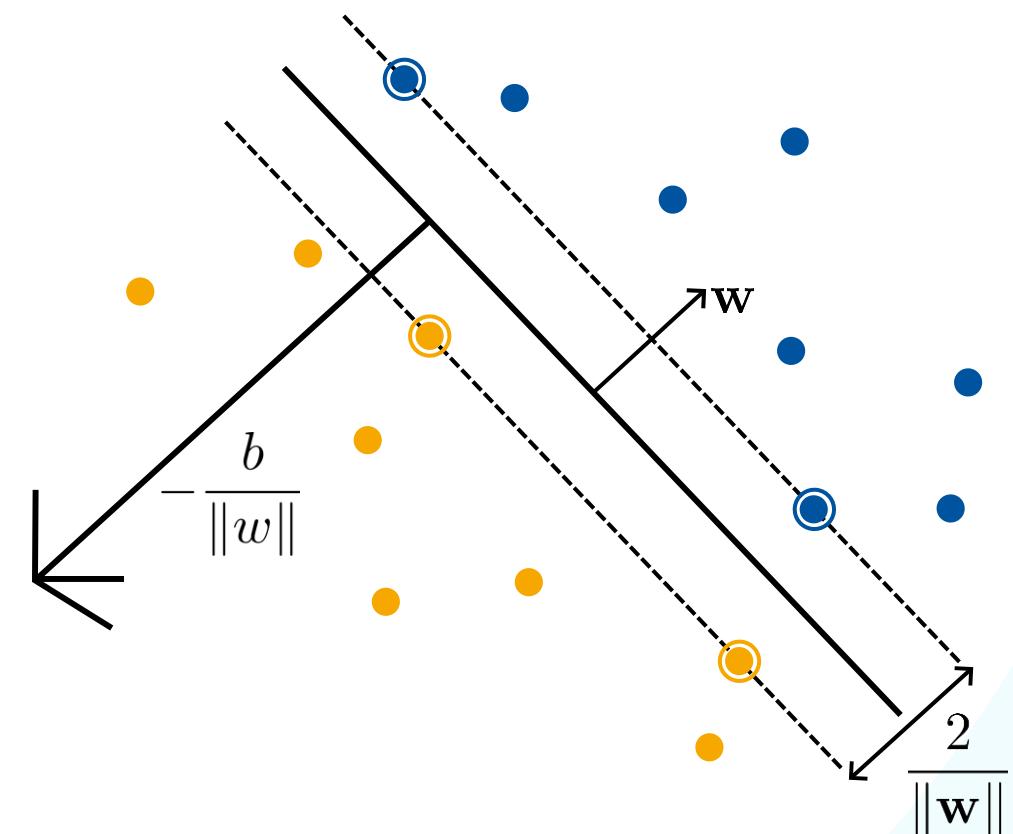
$$\mathbf{w}^T \mathbf{x}_n + b = -1 \text{ for one } t_n = -1$$

- The distance between those hyperplanes is then the margin:

$$d_- = d_+ = \frac{1}{\|\mathbf{w}\|}$$

$$d_- + d_+ = \frac{2}{\|\mathbf{w}\|}$$

\Rightarrow Maximize the margin by minimizing $\|\mathbf{w}\|^2$



- Optimization problem
 - Find the hyperplane with maximum margin by optimizing:

$$\arg \min_{\mathbf{w}, b} \frac{1}{2} \|\mathbf{w}\|^2$$

“Maximize the margin”

such that

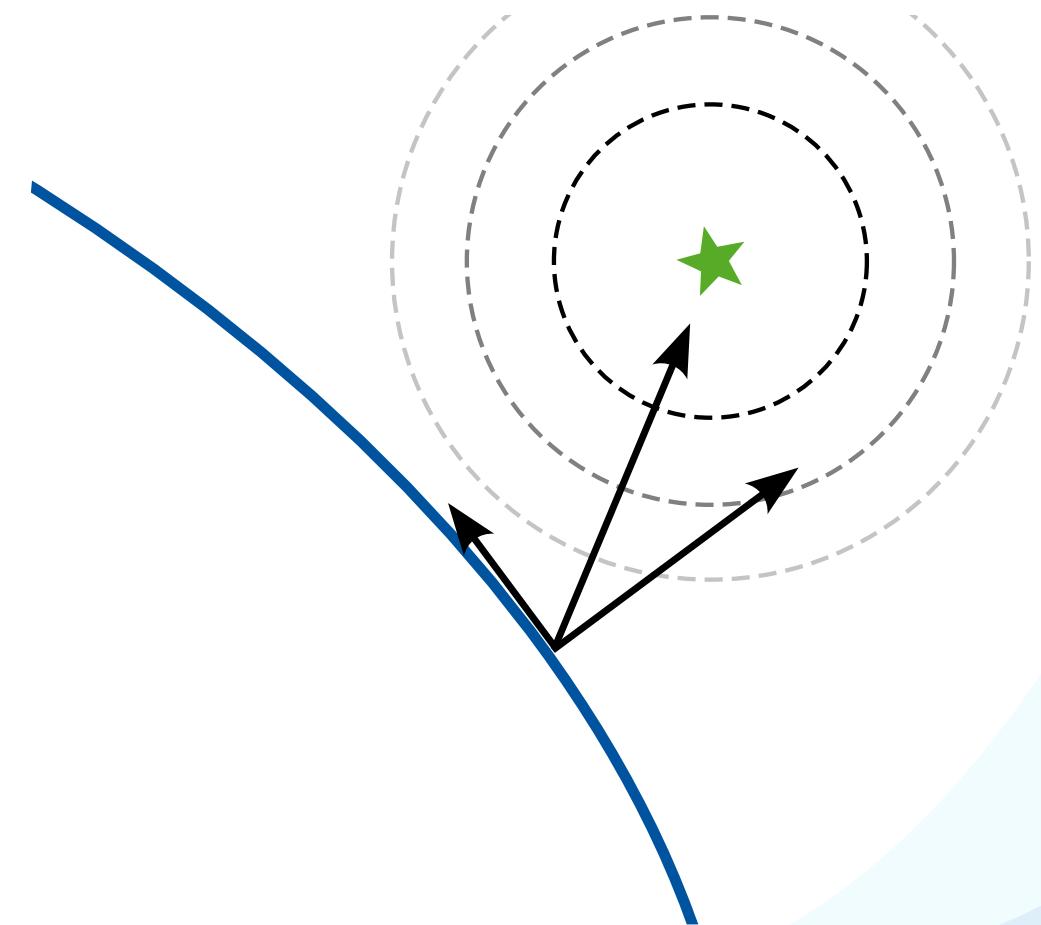
$$t_n(\mathbf{w}^\top \mathbf{x}_n + b) \geq 1 \quad \forall n$$

*“such that each point
is on the correct side
of the margin”*

- This is a **quadratic programming problem** with linear constraints.

Support Vector Machines

1. Maximum Margin Classification
 - a) **Constrained Optimization**
2. Primal Formulation
3. Dual Formulation
4. Soft-Margin SVMs
5. Non-linear SVMs
6. Error Function Analysis



Constrained Optimization

- Recall the SVM objective:

$$\arg \min_{\mathbf{w}, b} \frac{1}{2} \|\mathbf{w}\|^2, \quad \text{such that} \quad t_n(\mathbf{w}^\top \mathbf{x}_n + b) \geq 1 \quad \forall n$$

- This is a **constrained optimization problem**.
 - We want to optimize an objective $K(\mathbf{x})$ subject to constraints $f(\mathbf{x})$:

$\underset{\mathbf{x}}{\text{opt}} K(\mathbf{x})$	$\min \text{ or } \max$
such that	$f(\mathbf{x}) = 0$
	<i>equality constraints</i>
	$f(\mathbf{x}) \geq 0$
	<i>inequality constraints</i>

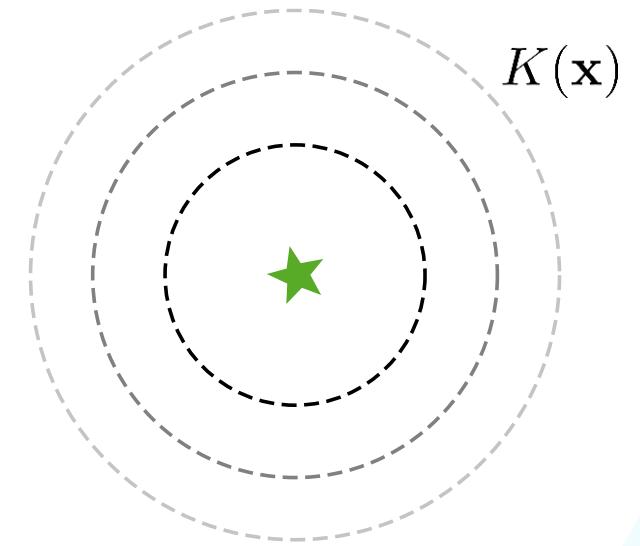
SVM

$$K(\mathbf{w}) = \frac{1}{2} \|\mathbf{w}\|^2$$
$$f_n(\mathbf{w}) = t_n(\mathbf{w}^\top \mathbf{x}_n + b) - 1 \geq 0 \quad \forall n$$

- We can solve such constrained optimization problems using the technique of **Lagrange multipliers**.

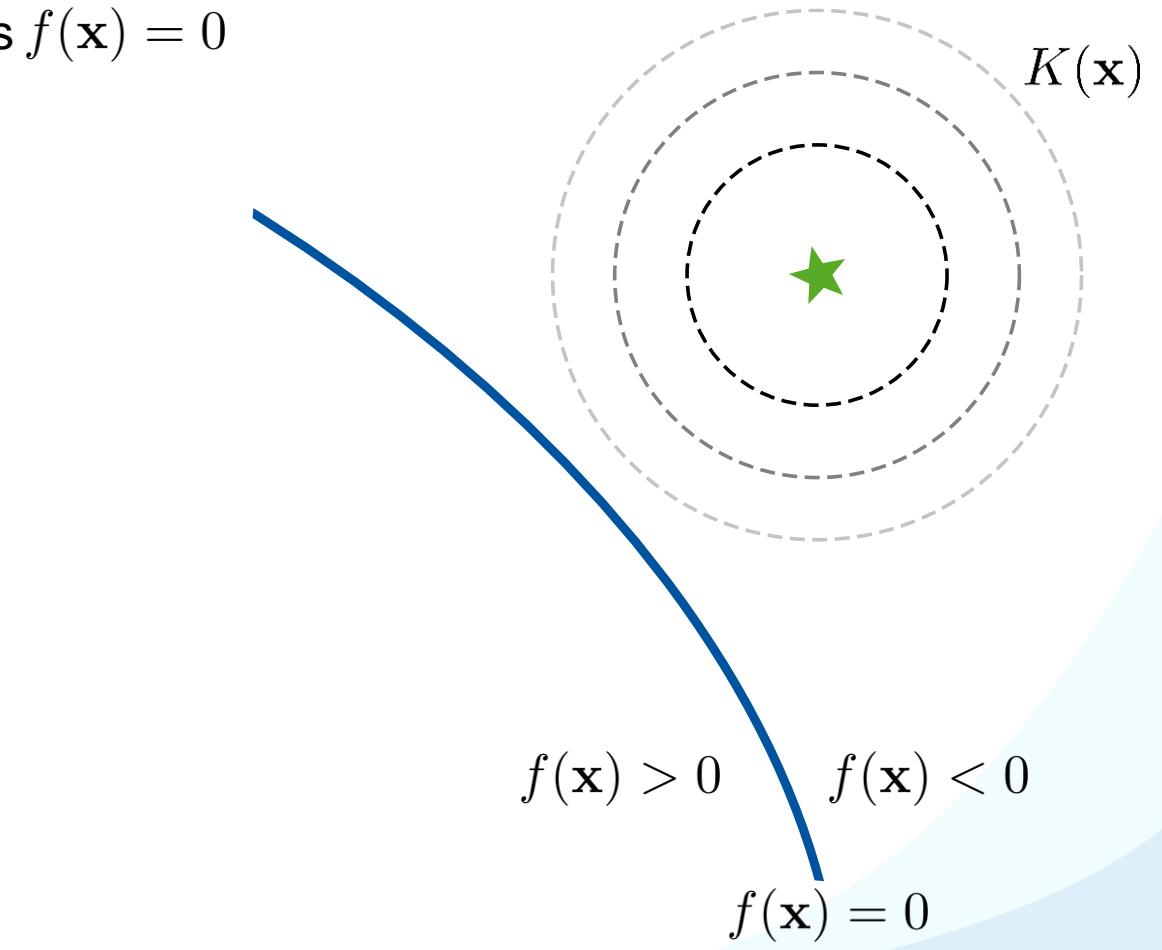
Lagrange Multipliers

- We want to maximize $K(\mathbf{x})$



Lagrange Multipliers

- We want to maximize $K(\mathbf{x})$ subject to constraints $f(\mathbf{x}) = 0$



Lagrange Multipliers

- We want to maximize $K(\mathbf{x})$ subject to constraints $f(\mathbf{x}) = 0$
- We can only move along $\nabla_{||} K = \nabla K + \lambda \nabla f$, with $\lambda \neq 0$.
- Add the constraints to the objective by introducing auxiliary variables λ :

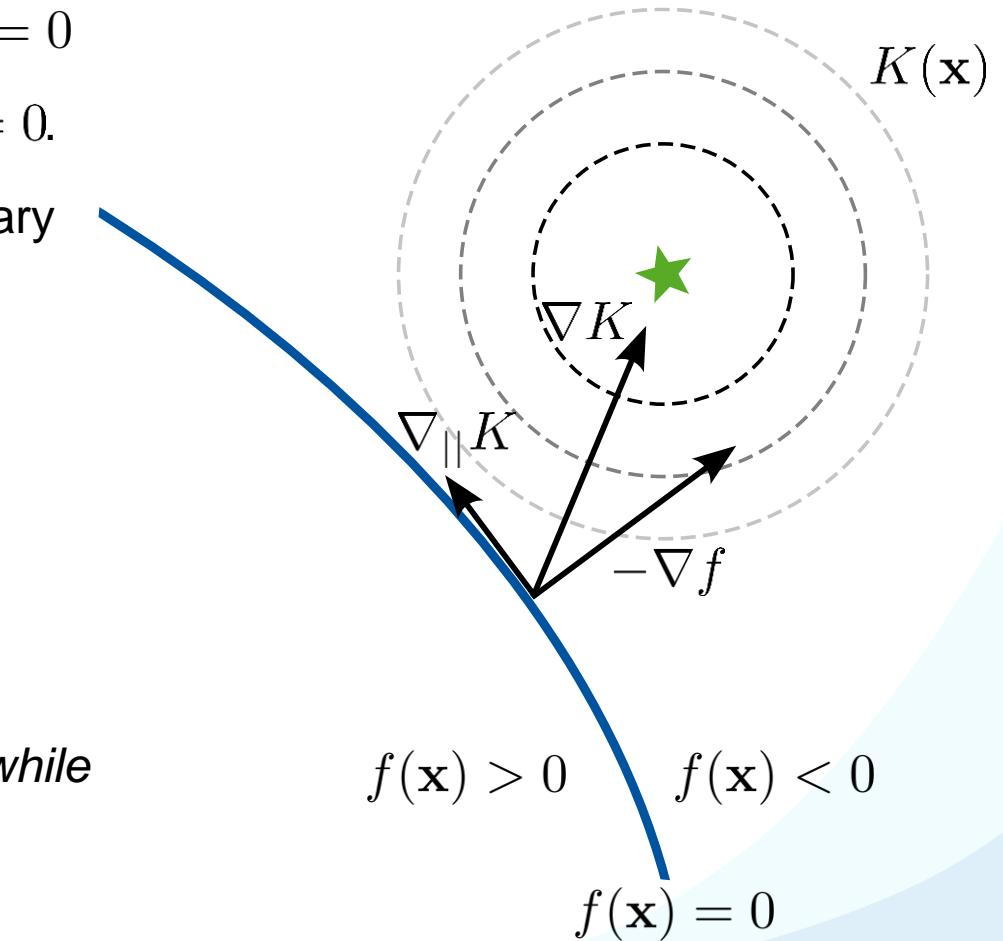
$$\mathcal{L}(\mathbf{x}, \lambda) = K(\mathbf{x}) + \lambda f(\mathbf{x})$$

- \mathcal{L} is called the **Lagrangian** form of the optimization problem, and λ is referred to as a **Lagrange multiplier**.
- Optimize \mathcal{L} :

$$\frac{\partial \mathcal{L}}{\partial \mathbf{x}} = \nabla_{||} K \stackrel{!}{=} 0$$

$$\frac{\partial \mathcal{L}}{\partial \lambda} = f(\mathbf{x}) \stackrel{!}{=} 0$$

The objective is maximized while satisfying the constraints.



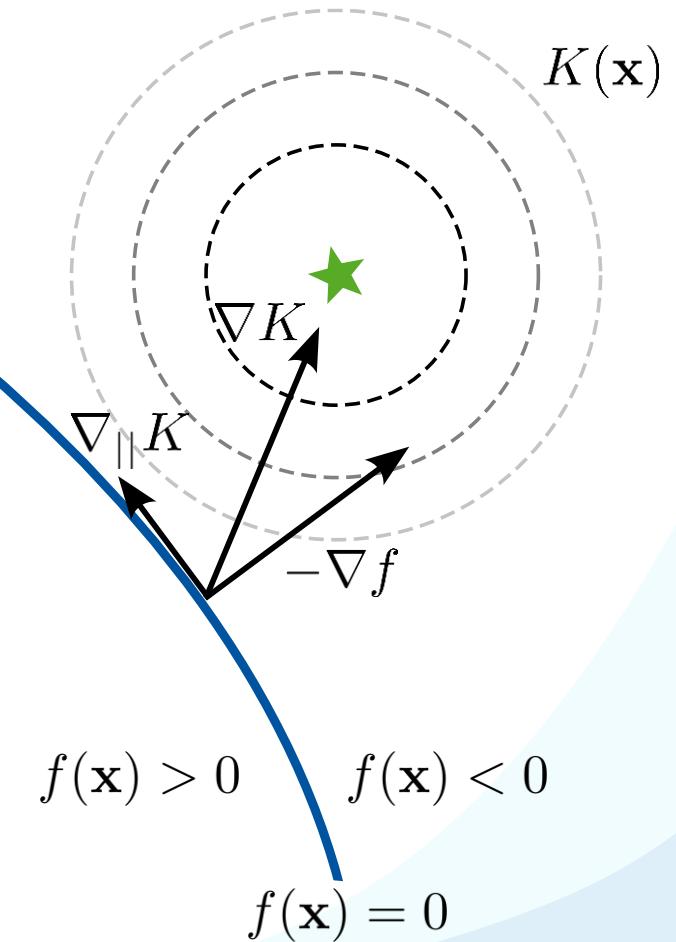
Inequality Constraints

- Now let's use inequality constraints $f(\mathbf{x}) \geq 0$.
- Optimize $\mathcal{L}(\mathbf{x}, \lambda) = K(\mathbf{x}) + \lambda f(\mathbf{x})$
 - Two cases
 - Solution lies on boundary:
 $\Rightarrow f(\mathbf{x}) = 0$ for some $\lambda > 0$
 - Solution lies inside $f(\mathbf{x}) > 0$:
 \Rightarrow Constraint inactive: $\lambda = 0$
- Karush-Kuhn-Tucker (KKT) conditions:

$$\begin{aligned}\lambda &\geq 0 \\ f(\mathbf{x}) &\geq 0 \\ \lambda f(\mathbf{x}) &= 0\end{aligned}$$

} In both cases:
 $\lambda f(\mathbf{x}) = 0$

*All valid solutions
need to fulfill the
KKT conditions.*



Maximization vs. Minimization

- Note: differences for maximization vs. minimization.
- If we want to **maximize** $K(\mathbf{x})$ subject to $f(\mathbf{x}) \geq 0$, we optimize the Lagrangian form

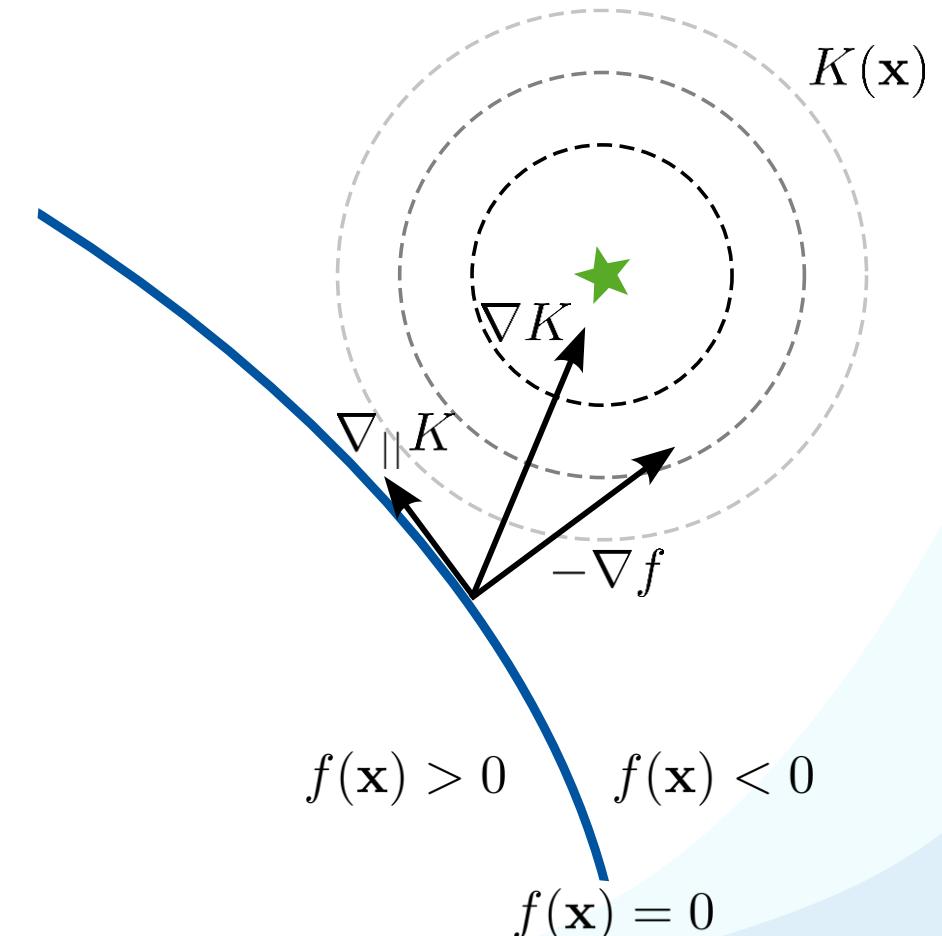
$$\mathcal{L}(\mathbf{x}, \lambda) = K(\mathbf{x}) + \lambda f(\mathbf{x})$$

- **maximize** w.r.t. \mathbf{x}
- **minimize** w.r.t. λ

- If we want to **minimize** $K(\mathbf{x})$ subject to $f(\mathbf{x}) \geq 0$, we optimize the Lagrangian form

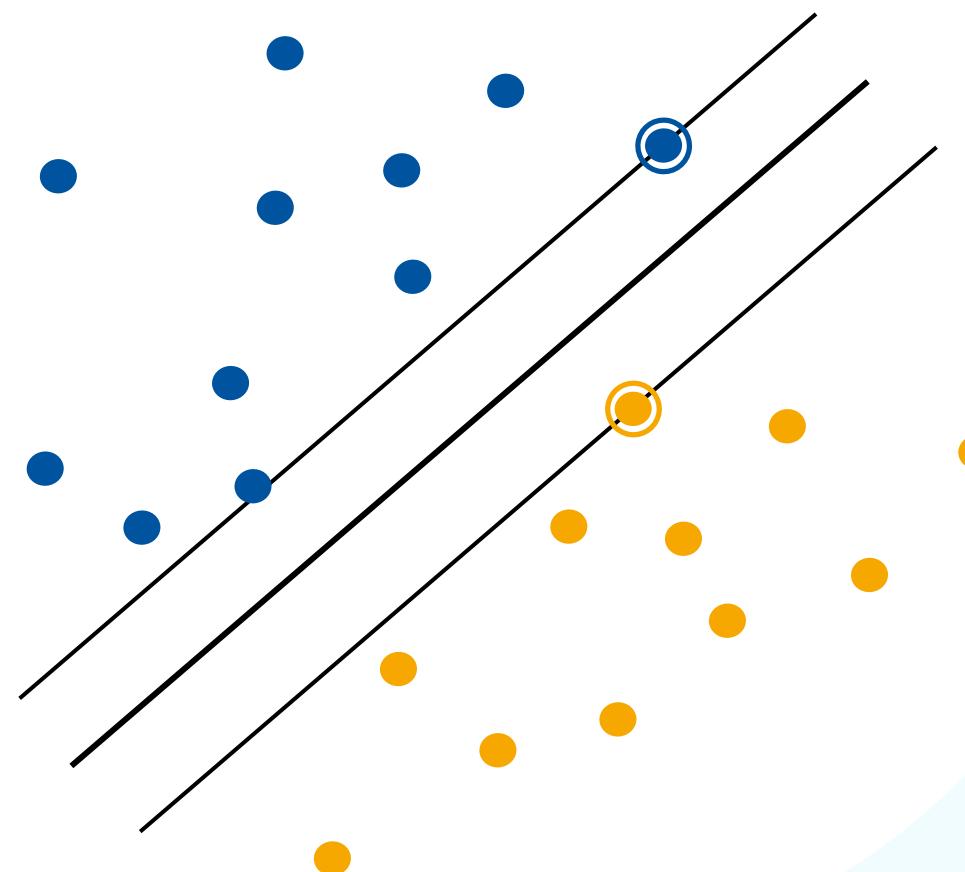
$$\mathcal{L}(\mathbf{x}, \lambda) = K(\mathbf{x}) - \lambda f(\mathbf{x})$$

- **minimize** w.r.t. \mathbf{x}
- **maximize** w.r.t. λ



Support Vector Machines

1. Maximum Margin Classification
2. **Primal Formulation**
3. Dual Formulation
4. Soft-Margin SVMs
5. Non-linear SVMs
6. Error Function Analysis



Primal SVM Formulation

- Recall the SVM objective:

$$\arg \min_{\mathbf{w}, b} \frac{1}{2} \|\mathbf{w}\|^2, \quad \text{such that} \quad t_n(\mathbf{w}^\top \mathbf{x}_n + b) \geq 1 \quad \forall n$$

- We introduce positive Lagrange multipliers $a_n \geq 0$ and get the [primal form](#) of SVMs:

$$L(\mathbf{w}, b, \mathbf{a}) = \frac{1}{2} \|\mathbf{w}\|^2 - \sum_{n=1}^N a_n [t_n(\mathbf{w}^\top \mathbf{x}_n + b) - 1]$$

Necessary and sufficient conditions:

$$\begin{aligned} a_n &\geq 0 \\ t_n(\mathbf{w}^\top \mathbf{x}_n + b) - 1 &\geq 0 \\ a_n[t_n(\mathbf{w}^\top \mathbf{x}_n + b) - 1] &= 0 \end{aligned}$$

KKT conditions:

$$\begin{aligned} \lambda &\geq 0 \\ f(\mathbf{x}) &\geq 0 \\ \lambda f(\mathbf{x}) &= 0 \end{aligned}$$

Lagrangian Formulation

- We want to minimize the primal form:

$$L(\mathbf{w}, b, \mathbf{a}) = \frac{1}{2} \|\mathbf{w}\|^2 - \sum_{n=1}^N a_n [t_n (\mathbf{w}^\top \mathbf{x}_n + b) - 1]$$

$$\frac{\partial L(\mathbf{w}, b, \mathbf{a})}{\partial b} = \sum_{n=1}^N a_n t_n$$

$$\frac{\partial L(\mathbf{w}, b, \mathbf{a})}{\partial \mathbf{w}} = \mathbf{w} - \sum_{n=1}^N a_n t_n \mathbf{x}_n$$

- Setting the gradients for \mathbf{w}, b to zero, we get:

$$\frac{\partial L}{\partial b} = 0 \Rightarrow \sum_{n=1}^N a_n t_n = 0$$

$$\frac{\partial L}{\partial \mathbf{w}} = 0 \Rightarrow \mathbf{w} = \sum_{n=1}^N a_n t_n \mathbf{x}_n$$

- The hyperplane is computed as a linear combination of training examples:

$$\mathbf{w} = \sum_{n=1}^N a_n t_n \mathbf{x}_n$$

$$\frac{\partial L}{\partial \mathbf{w}} = 0 \Rightarrow \mathbf{w} = \sum_{n=1}^N a_n t_n \mathbf{x}_n$$

- Additionally, the solution needs to fulfill

$$a_n [t_n(\mathbf{w}^\top \mathbf{x}_n + b) - 1] = 0$$

- This implies $a_n > 0$ only for those points for which

$$[t_n(\mathbf{w}^\top \mathbf{x}_n + b) - 1] = 0$$

KKT conditions:

$$a_n \geq 0$$

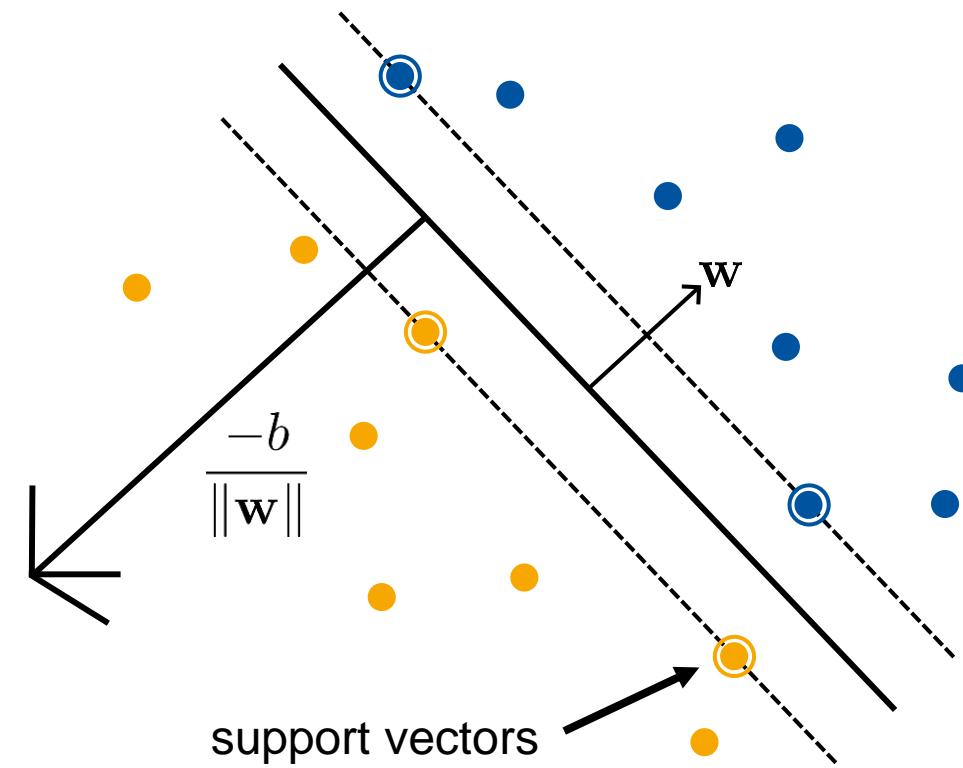
$$t_n(\mathbf{w}^\top \mathbf{x}_n + b) - 1 \geq 0$$

$$a_n [t_n(\mathbf{w}^\top \mathbf{x}_n + b) - 1] = 0$$

Only some data points influence the decision boundary!

Intuition

- The training points with $a_n > 0$ are called **support vectors**.
- They are the points on the margin.
- This makes the SVM robust to “too correct” points!



- We still need to find b .
- Observation: Any support vector \mathbf{x}_n satisfies

$$t_n y(\mathbf{x}_n) = t_n \left(\sum_{m \in \mathcal{S}} a_m t_m \mathbf{x}_m^\top \mathbf{x}_n + b \right) = 1$$

- Using $t_n^2 = 1$, we can derive

$$b = t_n - \sum_{m \in \mathcal{S}} a_m t_m \mathbf{x}_m^\top \mathbf{x}_n$$

- In practice, it is more robust to average over all support vectors:

$$b = \frac{1}{N_{\mathcal{S}}} \sum_{n \in \mathcal{S}} \left(t_n - \sum_{m \in \mathcal{S}} a_m t_m \mathbf{x}_m^\top \mathbf{x}_n \right)$$

Advantages

- SVMs yield a linear classifier with “guaranteed” generalization capability.
- Convex optimization, yields globally optimal solution.
- Solution depends only on a subset of the input data points, the **support vectors**.
- Automatic robustness against “too correct” data points.

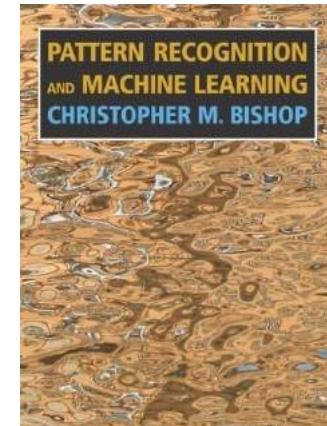
Limitations

- Need to solve **quadratic programming** problem: time complexity for that is cubic in the number of variables.
- Here: Time complexity is in $\mathcal{O}(D^3)$.
- Scaling to high-dimensional data is difficult.

References and Further Reading

- More information about SVMs is available in Chapter 7.1 of Bishop's book.

Christopher M. Bishop
Pattern Recognition and Machine Learning
Springer, 2006



Elements of Machine Learning & Data Science

Winter semester 2023/24

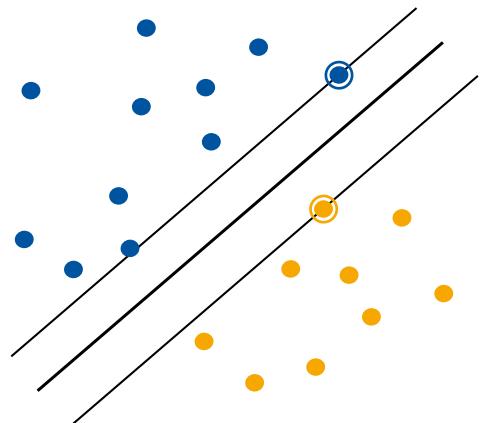
Lecture 17 – Support Vector Machines I

12.12.2023

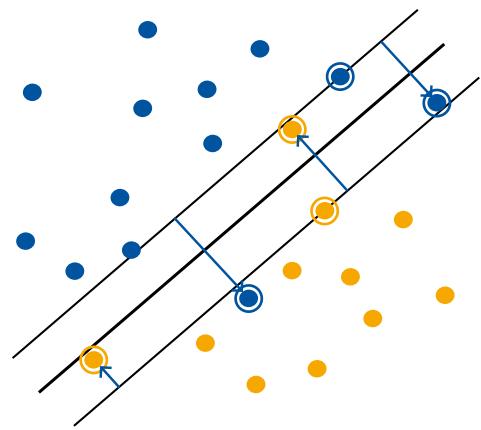
Prof. Bastian Leibe

Machine Learning Topics

1. Introduction to ML
2. Probability Density Estimation
3. Linear Discriminants
4. Linear Regression
5. Logistic Regression
- 6. Support Vector Machines**
7. (AdaBoost)
8. Neural Network Basics



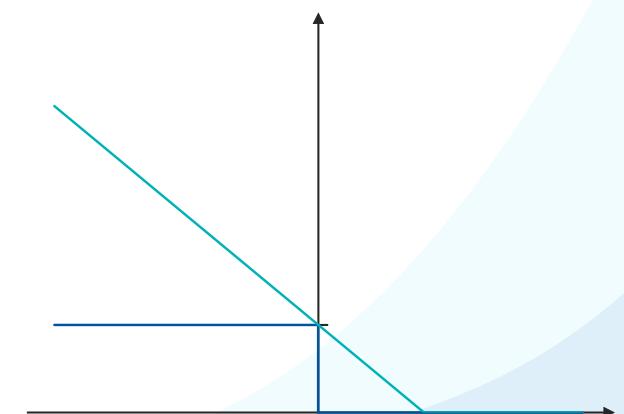
Maximum Margin Classification



Soft-Margin SVM

$$L_p(\mathbf{w}, b, \mathbf{a})$$
$$L_d(\mathbf{a})$$

Primal & Dual Form



Hinge Loss

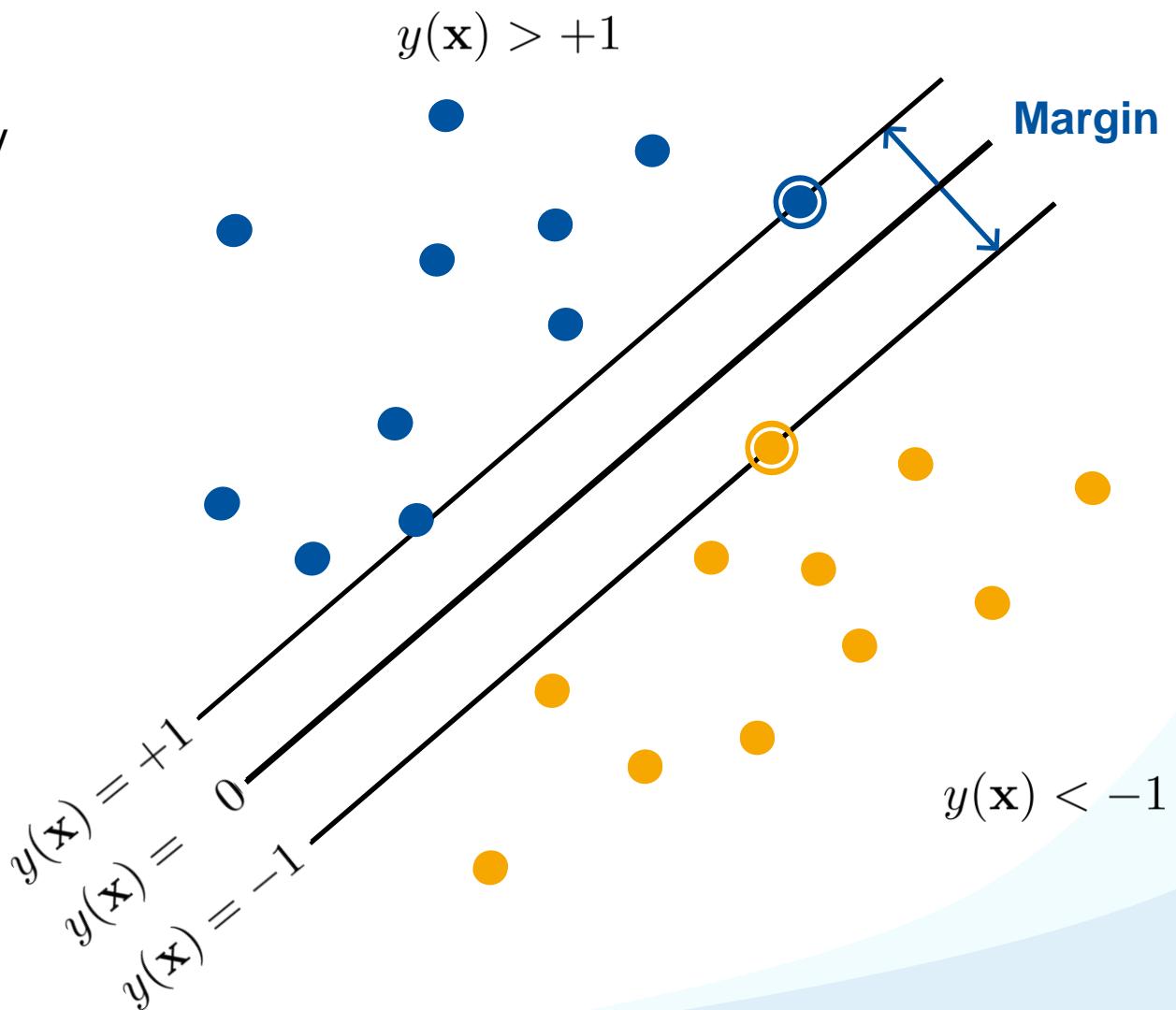
Recap: Maximum Margin Classification

- Intuitively, we want to choose the classifier which leaves maximal “safety room” for future data points.
- This classifier has the largest **margin** between positive and negative points.
- We can rescale \mathbf{w} such that the distance of the points on the margin to the decision boundary is exactly 1.

$$t_n(\mathbf{w}^T \mathbf{x}_n + b) = 1$$

- If the data is linearly separable, then for all points, the following must hold:

$$t_n(\mathbf{w}^T \mathbf{x}_n + b) \geq 1 \quad \forall n$$



- Optimization problem
 - Find the hyperplane with maximum margin by optimizing:

$$\arg \min_{\mathbf{w}, b} \frac{1}{2} \|\mathbf{w}\|^2$$

“Maximize the margin”

such that

$$t_n(\mathbf{w}^\top \mathbf{x}_n + b) \geq 1 \quad \forall n$$

*“such that each point
is on the correct side
of the margin”*

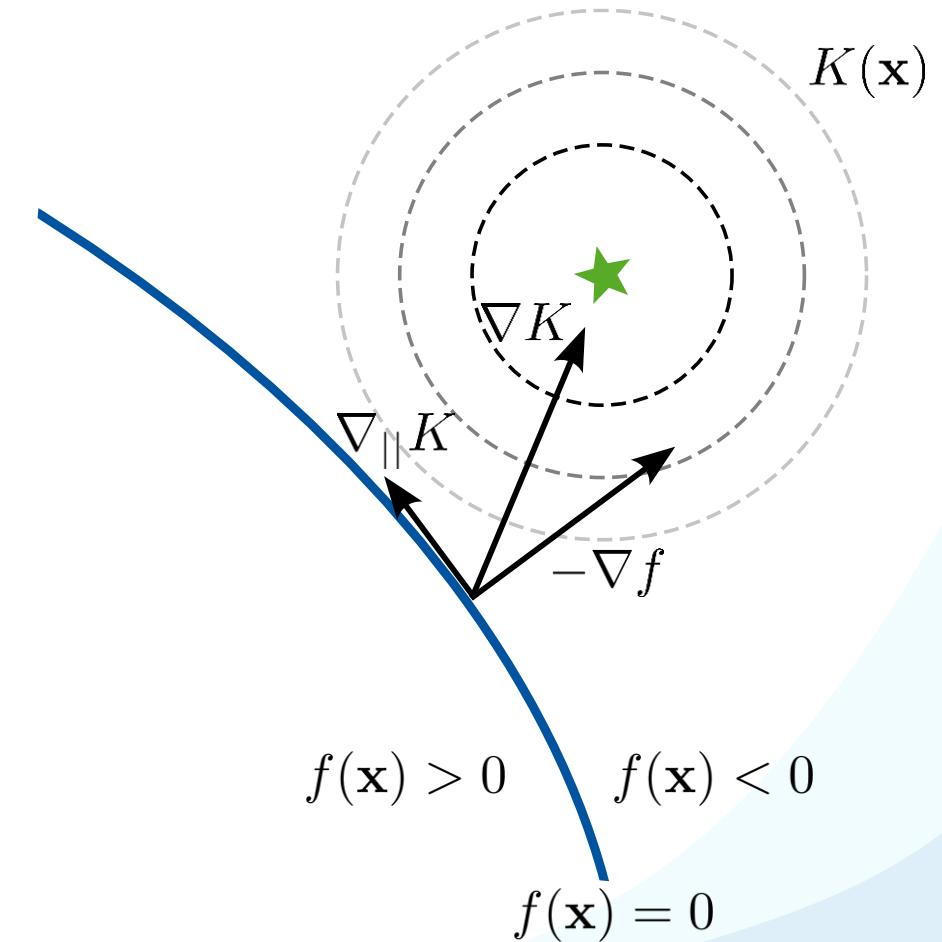
- This is a **quadratic programming problem** with linear constraints.

Recap: Constrained Optimization with Lagrange Multipliers

- If we want to minimize $K(\mathbf{x})$ subject to $f(\mathbf{x}) \geq 0$, we optimize the Lagrangian form

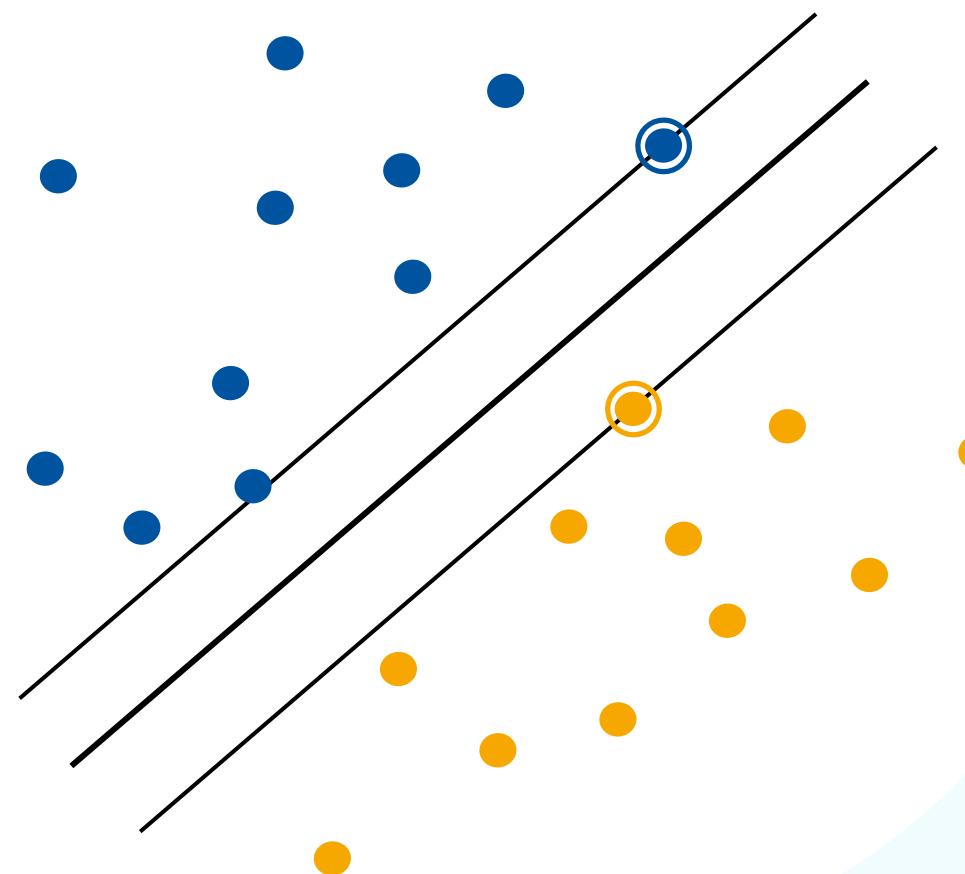
$$\mathcal{L}(\mathbf{x}, \lambda) = K(\mathbf{x}) - \lambda f(\mathbf{x})$$
 - minimize w.r.t. \mathbf{x}
 - maximize w.r.t. λ
- I.e., we introduce an auxiliary variable λ for every constraint. λ is called a **Lagrange multiplier**.
- All valid solution need to fulfill the **Karush-Kuhn-Tucker (KKT)** conditions

$$\begin{aligned}\lambda &\geq 0 \\ f(\mathbf{x}) &\geq 0 \\ \lambda f(\mathbf{x}) &= 0\end{aligned}$$



Support Vector Machines

1. Maximum Margin Classification
2. **Primal Formulation**
3. Dual Formulation
4. Soft-Margin SVMs
5. Non-linear SVMs
6. Error Function Analysis



Primal SVM Formulation

- Recall the SVM objective:

$$\arg \min_{\mathbf{w}, b} \frac{1}{2} \|\mathbf{w}\|^2, \quad \text{such that} \quad t_n(\mathbf{w}^\top \mathbf{x}_n + b) \geq 1 \quad \forall n$$

- We introduce positive Lagrange multipliers $a_n \geq 0$ and get the **primal form** of SVMs:

$$L(\mathbf{w}, b, \mathbf{a}) = \frac{1}{2} \|\mathbf{w}\|^2 - \sum_{n=1}^N a_n [t_n(\mathbf{w}^\top \mathbf{x}_n + b) - 1]$$

Necessary and sufficient conditions:

$$\begin{aligned} a_n &\geq 0 \\ t_n(\mathbf{w}^\top \mathbf{x}_n + b) - 1 &\geq 0 \\ a_n[t_n(\mathbf{w}^\top \mathbf{x}_n + b) - 1] &= 0 \end{aligned}$$

KKT conditions:

$$\begin{aligned} \lambda &\geq 0 \\ f(\mathbf{x}) &\geq 0 \\ \lambda f(\mathbf{x}) &= 0 \end{aligned}$$

Lagrangian Formulation

- We want to minimize the primal form:

$$L(\mathbf{w}, b, \mathbf{a}) = \frac{1}{2} \|\mathbf{w}\|^2 - \sum_{n=1}^N a_n [t_n (\mathbf{w}^\top \mathbf{x}_n + b) - 1]$$

$$\frac{\partial L(\mathbf{w}, b, \mathbf{a})}{\partial b} = \sum_{n=1}^N a_n t_n$$

$$\frac{\partial L(\mathbf{w}, b, \mathbf{a})}{\partial \mathbf{w}} = \mathbf{w} - \sum_{n=1}^N a_n t_n \mathbf{x}_n$$

- Setting the gradients for \mathbf{w}, b to zero, we get:

$$\frac{\partial L}{\partial b} = 0 \Rightarrow \sum_{n=1}^N a_n t_n = 0$$

$$\frac{\partial L}{\partial \mathbf{w}} = 0 \Rightarrow \mathbf{w} = \sum_{n=1}^N a_n t_n \mathbf{x}_n$$

- The hyperplane is computed as a linear combination of training examples:

$$\mathbf{w} = \sum_{n=1}^N a_n t_n \mathbf{x}_n$$

$$\frac{\partial L}{\partial \mathbf{w}} = 0 \Rightarrow \mathbf{w} = \sum_{n=1}^N a_n t_n \mathbf{x}_n$$

- Additionally, the solution needs to fulfill

$$a_n [t_n(\mathbf{w}^\top \mathbf{x}_n + b) - 1] = 0$$

- This implies $a_n > 0$ only for those points for which

$$[t_n(\mathbf{w}^\top \mathbf{x}_n + b) - 1] = 0$$

KKT conditions:

$$a_n \geq 0$$

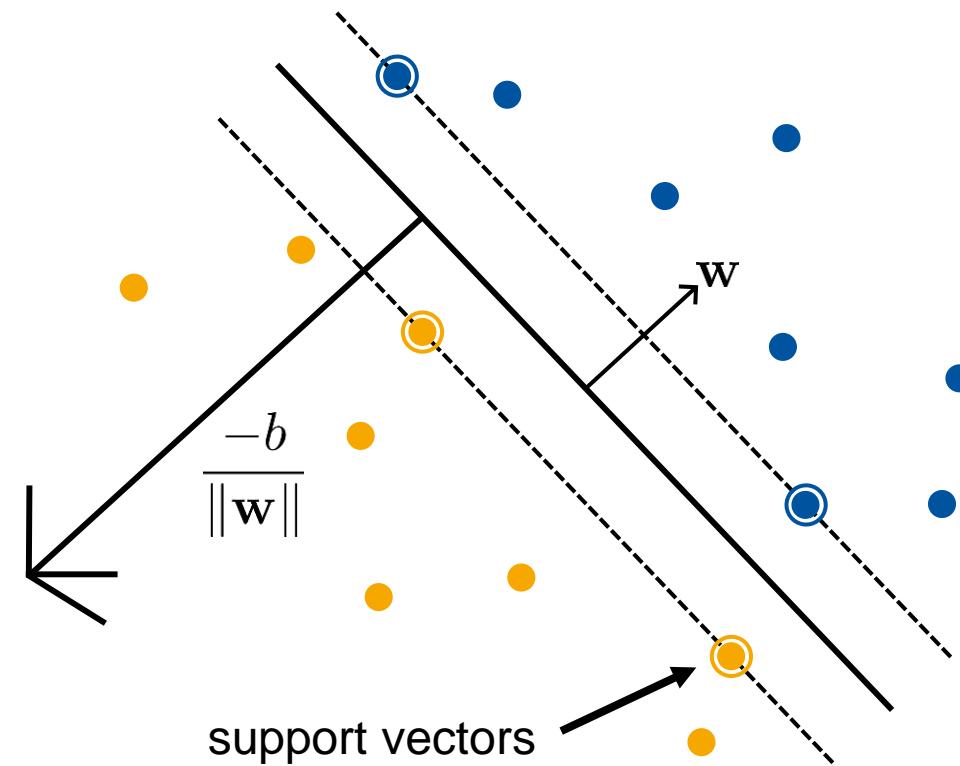
$$t_n(\mathbf{w}^\top \mathbf{x}_n + b) - 1 \geq 0$$

$$a_n [t_n(\mathbf{w}^\top \mathbf{x}_n + b) - 1] = 0$$

Only some data points influence the decision boundary!

Intuition

- The training points with $a_n > 0$ are called **support vectors**.
- They are the points on the margin.
- This makes the SVM robust to “too correct” points!



- We still need to find b .
- Observation: Any support vector \mathbf{x}_n satisfies

$$t_n y(\mathbf{x}_n) = t_n \left(\sum_{m \in \mathcal{S}} a_m t_m \mathbf{x}_m^\top \mathbf{x}_n + b \right) = 1$$

- Using $t_n^2 = 1$, we can derive

$$b = t_n - \sum_{m \in \mathcal{S}} a_m t_m \mathbf{x}_m^\top \mathbf{x}_n$$

- In practice, it is more robust to average over all support vectors:

$$b = \frac{1}{N_{\mathcal{S}}} \sum_{n \in \mathcal{S}} \left(t_n - \sum_{m \in \mathcal{S}} a_m t_m \mathbf{x}_m^\top \mathbf{x}_n \right)$$

Advantages

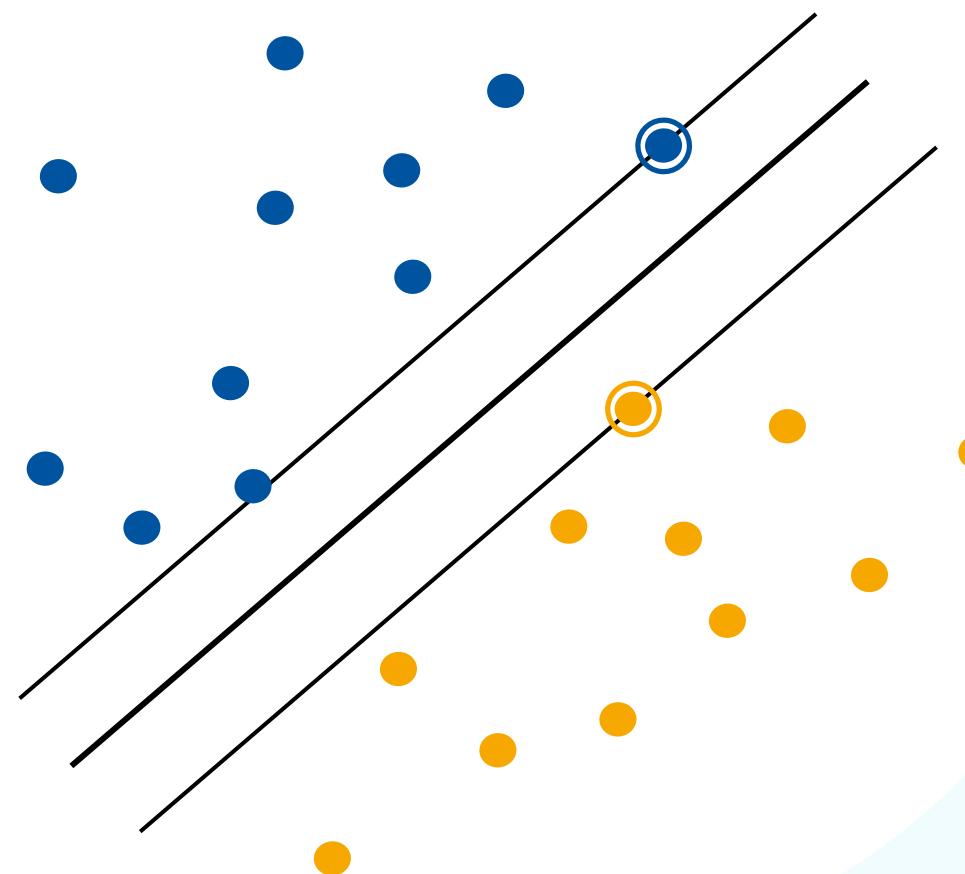
- SVMs yield a linear classifier with “guaranteed” generalization capability.
- Convex optimization, yields globally optimal solution.
- Solution depends only on a subset of the input data points, the **support vectors**.
- Automatic robustness against “too correct” data points.

Limitations

- Need to solve **quadratic programming** problem: time complexity for that is cubic in the number of variables.
- Here: Time complexity is in $\mathcal{O}(D^3)$.
- Scaling to high-dimensional data is difficult.

Support Vector Machines

1. Maximum Margin Classification
2. Primal Formulation
3. **Dual Formulation**
4. Soft-Margin SVMs
5. Non-linear SVMs
6. Error Function Analysis



Reminder: Primal SVM Formulation

- SVM objective:

$$\arg \min_{\mathbf{w}, b} \frac{1}{2} \|\mathbf{w}\|^2, \quad \text{such that} \quad t_n(\mathbf{w}^\top \mathbf{x}_n + b) \geq 1 \quad \forall n$$

- This is a **Quadratic Programming (QP)** problem with **linear inequality constraints**.
 - In order to solve it, we have derived the **Lagrangian primal form**

$$L_p(\mathbf{w}, b, \mathbf{a}) = \frac{1}{2} \|\mathbf{w}\|^2 - \sum_{n=1}^N a_n [t_n(\mathbf{w}^\top \mathbf{x}_n + b) - 1]$$

- We are *minimizing* this objective with respect to \mathbf{w} and b , and *maximizing* with respect to \mathbf{a} .

Solving a QP

- SVM objective:

$$\arg \min_{\mathbf{w}, b} \frac{1}{2} \|\mathbf{w}\|^2, \quad \text{such that} \quad t_n(\mathbf{w}^\top \mathbf{x}_n + b) \geq 1 \quad \forall n$$

- Solving QPs is a well-understood problem
 - Typically done with the help of a [QP solver](#).
 - Solving a QP in K variables can be done in runtime $\mathcal{O}(K^3)$.
- In our case: $\mathbf{x}, \mathbf{w} \in \mathbb{R}^D$
 - #Variables: $D + 1$
 \Rightarrow Complexity: $\mathcal{O}(D^3)$

Solving a QP

- SVM objective:

$$\arg \min_{\mathbf{w}, b} \frac{1}{2} \|\mathbf{w}\|^2, \quad \text{such that} \quad t_n(\mathbf{w}^\top \phi(\mathbf{x}_n) + b) \geq 1 \quad \forall n$$

- Solving QPs is a well-understood problem
 - Typically done with the help of a [QP solver](#).
 - Solving a QP in K variables can be done in runtime $\mathcal{O}(K^3)$.
 - In our case: $\mathbf{x}, \mathbf{w} \in \mathbb{R}^D$
 - #Variables: $D + 1$
 - ⇒ Complexity: $\mathcal{O}(D^3)$
 - With basis functions: $\phi(\mathbf{x}), \mathbf{w} \in \mathbb{R}^M$, $M \gg D$
 - #Variables: $M + 1$
 - ⇒ Complexity: $\mathcal{O}(M^3)$
- ⇒ [Curse of dimensionality](#), the SVM Primal Form does not scale well!

Dual Form of the SVM Objective

- Maximize

$$L_d(\mathbf{a}) = \sum_{n=1}^N a_n - \frac{1}{2} \sum_{n=1}^N \sum_{m=1}^N a_n a_m t_n t_m (\mathbf{x}_m^\top \mathbf{x}_n)$$

under the conditions

$$a_n \geq 0 \quad \forall n$$

$$\sum_{n=1}^N a_n t_n = 0$$

- We now have an optimization problem in N variables.
 \Rightarrow Complexity: $\mathcal{O}(N^3)$

For the derivation, please watch the video



Discussion

- What have we gained?
 - Previous complexity was $\mathcal{O}(D^3)$, now it is $\mathcal{O}(N^3)$.
 - *Isn't this much worse for large training sets???*
- However, the dual form has several advantages
 1. SVMs have sparse solutions: $a_n \neq 0$ only for support vectors.
 - This makes very efficient algorithms possible.
 - E.g., Sequential Minimal Optimization (**SMO**)
 - Effective runtime between $\mathcal{O}(N)$ and $\mathcal{O}(N^2)$.
 2. No dependency on the dimensionality anymore.
 - We can work with high-dimensional feature spaces!

Advantages

- Optimization problem only depends on the Lagrange multipliers a_n resulting in a worst-case runtime complexity of $\mathcal{O}(N^3)$.
- Since SVMs have sparse solutions and only few $a_n \neq 0$, specialized algorithms can solve the dual form very efficiently.
- The complexity of QP optimization no longer depends on the dimensionality of the feature space. This makes it possible to use very high-dimensional feature spaces.

Limitations

- Evaluating the SVM decision function

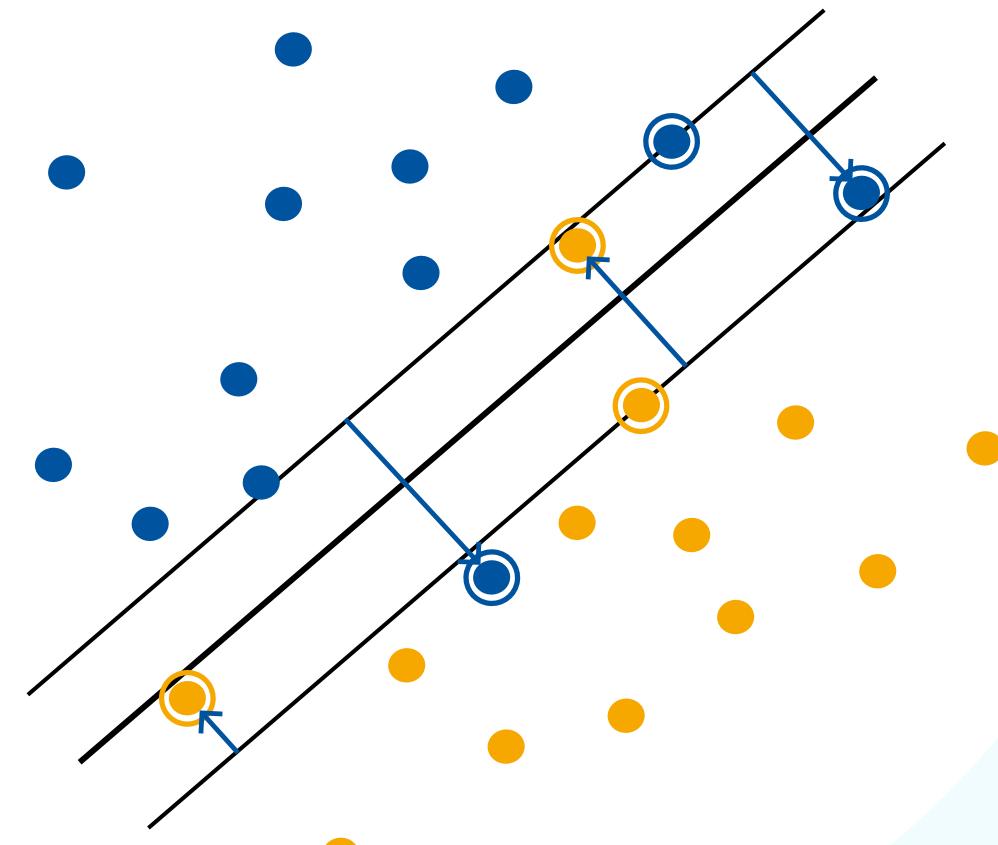
$$y(\mathbf{x}) = \mathbf{w}^\top \phi(\mathbf{x}) + b$$

$$\text{with } \mathbf{w} = \sum_{n=1}^N a_n t_n \phi(\mathbf{x}_n)$$

is still costly for high-dimensional feature spaces $\phi(\mathbf{x})$.

Support Vector Machines

1. Maximum Margin Classification
2. Primal Formulation
3. Dual Formulation
4. **Soft-Margin SVMs**
5. Non-linear SVMs
6. Error Function Analysis



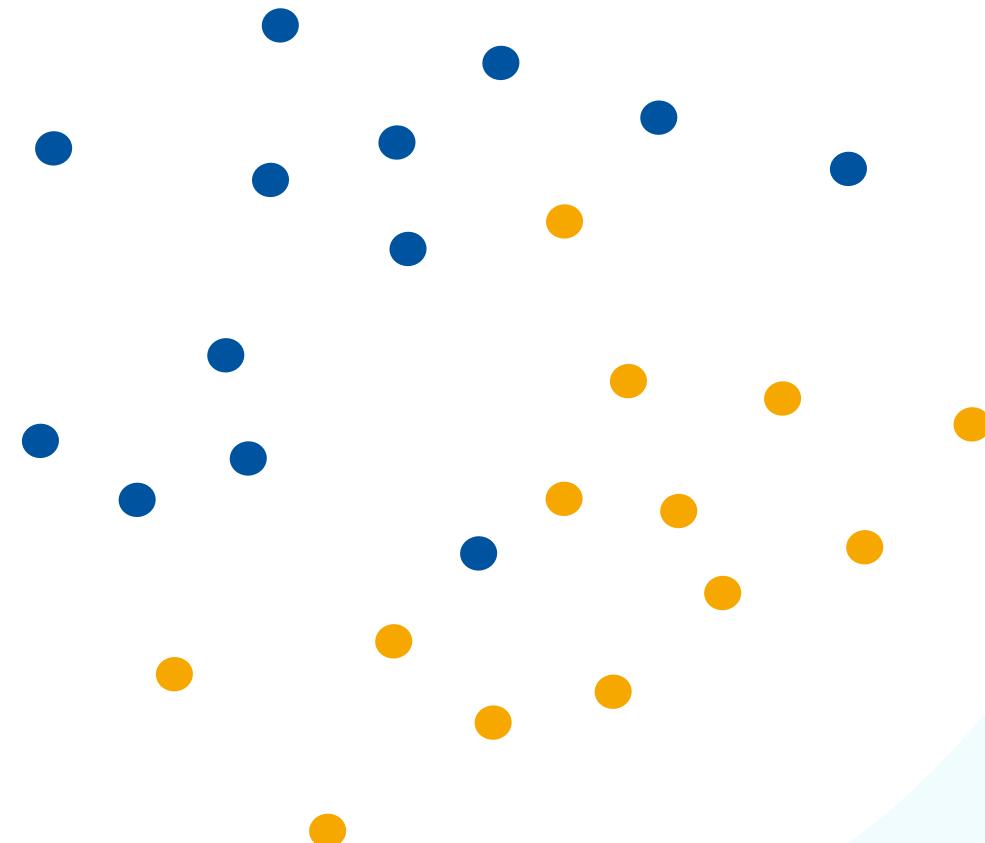
Soft-Margin SVM

- So far, we assumed linearly separable data.
 - Our current formulation has no solution if the data are not linearly separable!

$$\arg \min_{\mathbf{w}, b} \frac{1}{2} \|\mathbf{w}\|^2,$$

such that $t_n(\mathbf{w}^\top \mathbf{x}_n + b) \geq 1 \quad \forall n$

- Need to introduce tolerance to outlier data points.
 - The resulting model is called **soft-margin SVM**.



Slack Variables

- For non-linearly separable data, not all constraints can be satisfied:

$$\mathbf{w}^T \mathbf{x}_n + b \geq +1 \quad \text{for } t_n = +1$$

$$\mathbf{w}^T \mathbf{x}_n + b \leq -1 \quad \text{for } t_n = -1$$

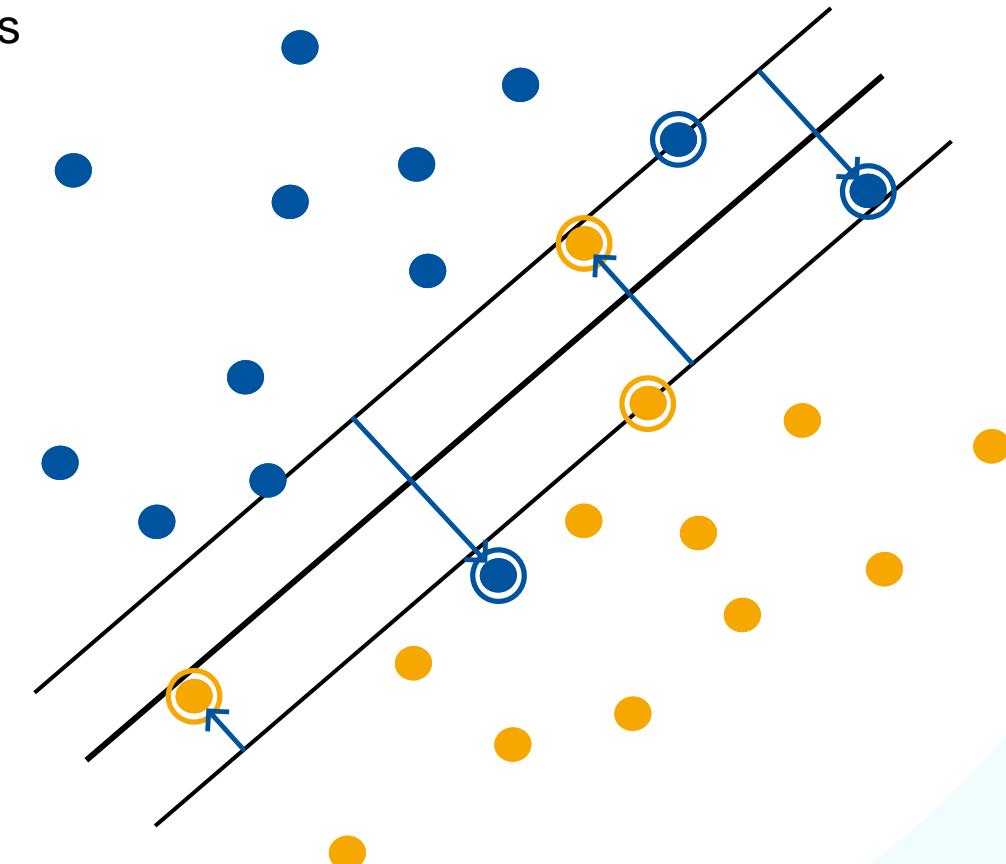
- Idea: Introduce **slack variables** $\xi_n \geq 0$:

$$\mathbf{w}^T \mathbf{x}_n + b \geq +1 - \xi_n \quad \text{for } t_n = +1$$

$$\mathbf{w}^T \mathbf{x}_n + b \leq -1 + \xi_n \quad \text{for } t_n = -1$$

⇒ We allow some datapoints to violate the constraint.

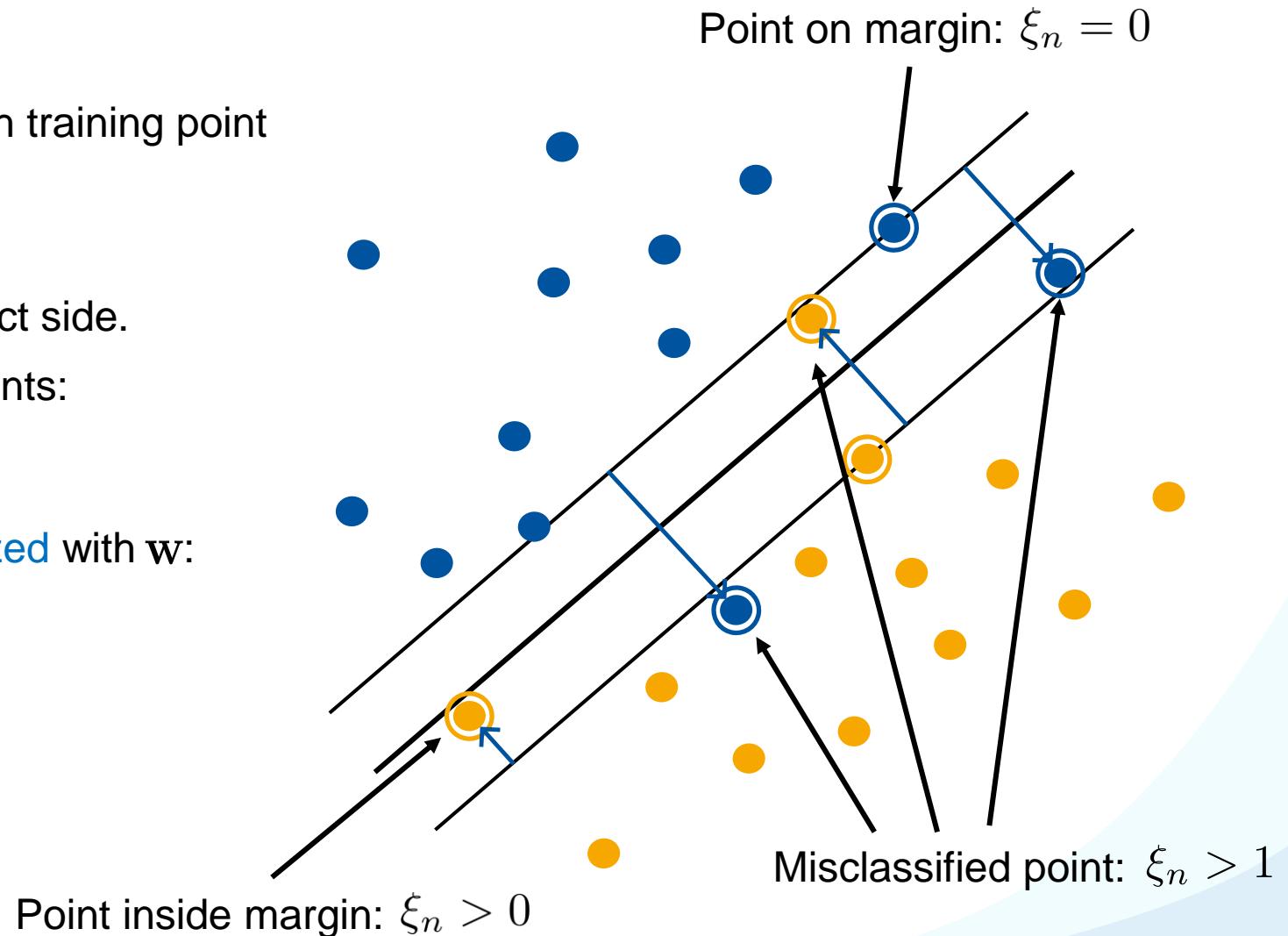
- For those points, the slack ξ_n makes up for the difference.



- Slack variables
 - One slack variable ξ_n for each training point
- Effect
 - $\xi_n = 0$ for points on the correct side.
 - **Linear penalty** for all other points:
$$\xi_n = |t_n - y(\mathbf{x}_n)|$$
- Slack variables are **jointly optimized** with \mathbf{w} :

$$\min \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{n=1}^N \xi_n$$

where C is a tradeoff parameter.



New Primal Formulation

- Minimize

$$L_p = \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{n=1}^N \xi_n - \underbrace{\sum_{n=1}^N a_n [t_n(y(\mathbf{x}_n) - 1 + \xi_n)]}_{\text{Constraint}} - \underbrace{\sum_{n=1}^N \mu_n \xi_n}_{\text{Constraint}}$$

$$t_n y(\mathbf{x}_n) \geq 1 - \xi_n \quad \xi_n \geq 0$$

- KKT conditions

$$\begin{array}{ll} a_n \geq 0 & \mu_n \geq 0 \\ t_n y(\mathbf{x}_n) - 1 + \xi_n \geq 0 & \xi_n \geq 0 \\ a_n [t_n y(\mathbf{x}_n) - 1 + \xi_n] = 0 & \mu_n \xi_n = 0 \end{array}$$

New Dual Formulation

- Maximize

$$L_d(\mathbf{a}) = \sum_{n=1}^N a_n - \frac{1}{2} \sum_{n=1}^N \sum_{m=1}^N a_n a_m t_n t_m (\mathbf{x}_m^\top \mathbf{x}_n)$$

- Under the side conditions

$$0 \leq a_n \leq C \quad \forall n$$

$$\sum_{n=1}^N a_n t_n = 0$$

This is the only difference to before.

New Solution

- The decision hyperplane is again a linear combination of training samples:

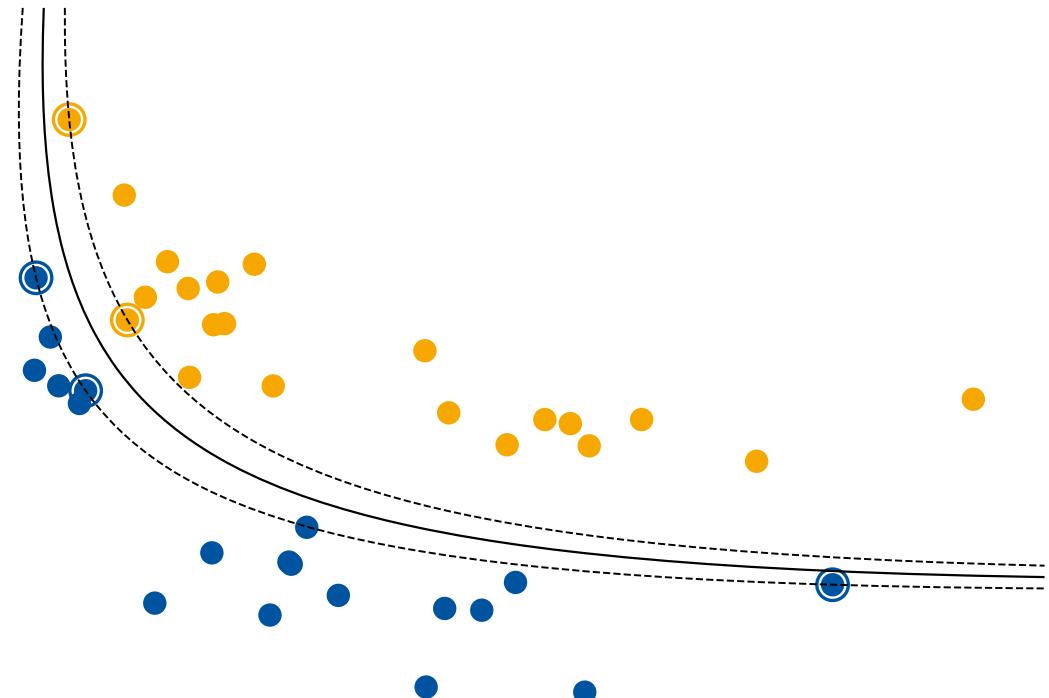
$$\mathbf{w} = \sum_{n=1}^N a_n t_n \mathbf{x}_n$$

- This is still a sparse solution:
 - $a_n = 0$ for points on the correct side of the margin
 - Slack points with $\xi_n > 0$ are now also support vectors!
- Compute b by averaging over support vectors (points with $0 < a_n < C$):

$$b = \frac{1}{N_S} \sum_{n \in S} \left(t_n - \sum_{m \in S} a_m t_m \mathbf{x}_m^\top \mathbf{x}_n \right)$$

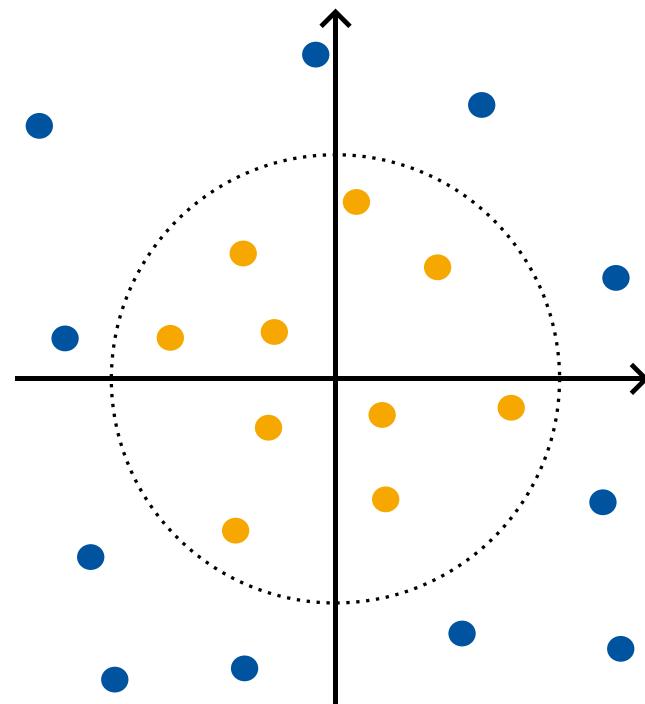
Support Vector Machines

1. Maximum Margin Classification
2. Primal Formulation
3. Dual Formulation
4. Soft-Margin SVMs
5. **Non-Linear SVMs**
6. Error Function Analysis



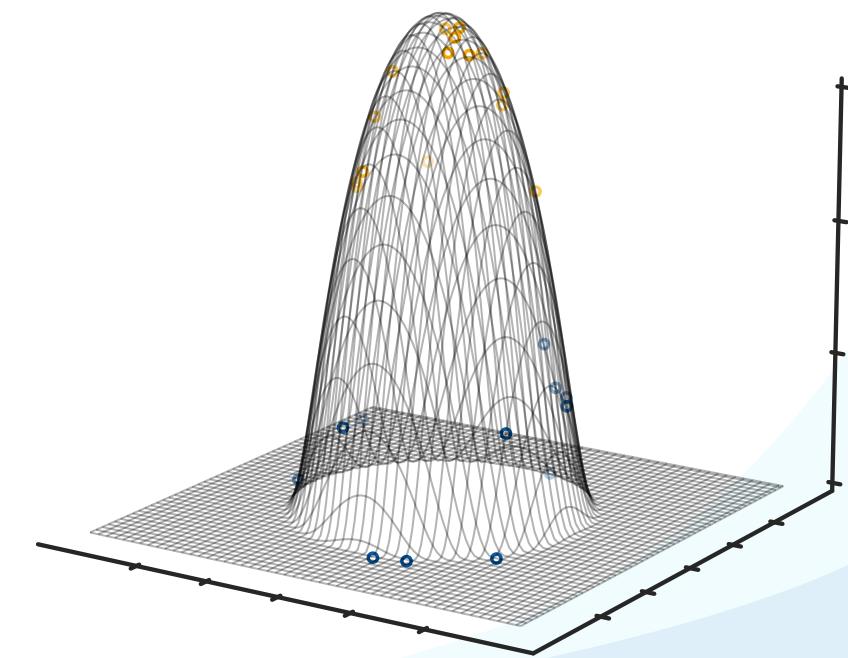
Non-Linear SVMs

- So far, we have only considered linear decision boundaries.
- We now combine non-linear basis functions with SVMs.



$$\phi : \mathbb{R}^D \rightarrow \mathbb{R}^M$$

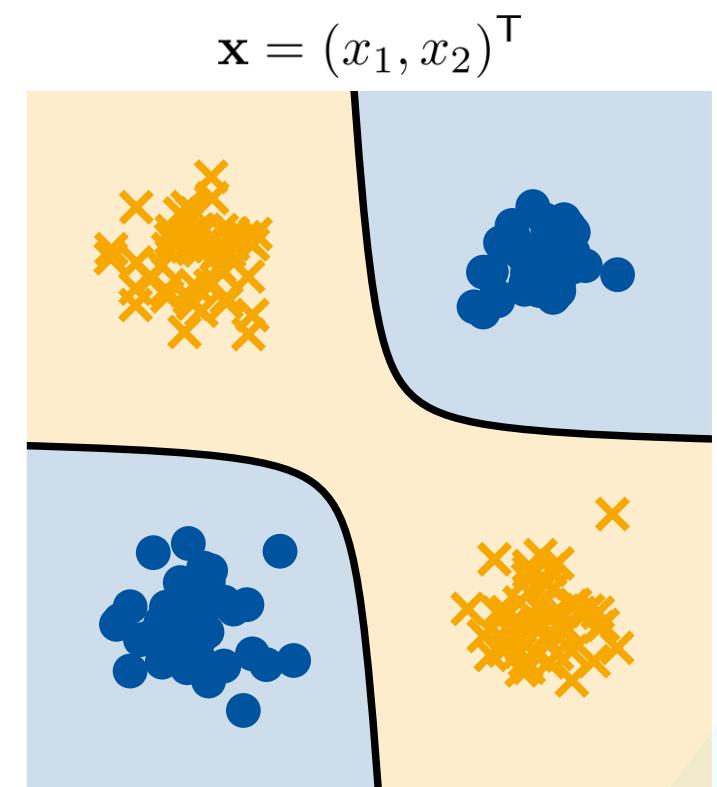
$$M \gg D$$



Feature Spaces

- We have already seen non-linear basis functions:
 - Apply a nonlinear transformation ϕ to the data points \mathbf{x}_n :
$$\mathbf{x} \in \mathbb{R}^D, \quad \phi : \mathbb{R}^D \rightarrow \mathbb{R}^M$$
 - Classify with a hyperplane in higher-dim. space \mathbb{R}^M :
$$\mathbf{w}^\top \phi(\mathbf{x}) + b = 0$$

\Rightarrow Linear classifier in \mathbb{R}^M , nonlinear classifier in \mathbb{R}^D .
- Let us now apply this to SVMs...
 - We can train our SVM on the transformed features $\phi(\mathbf{x})$ to get non-linear decision boundaries.
 - Usually, $M \gg D$: evaluating $\mathbf{w}^\top \phi(\mathbf{x})$ can be quite expensive!



$$\phi(\mathbf{x}) = (x_1, x_2, x_1x_2, x_1^2, x_2^2)^\top$$

The Kernel Trick

- On a closer look, $\phi(\mathbf{x})$ only appears in the form of dot products:

$$L_d(\mathbf{a}) = \sum_{n=1}^N a_n - \frac{1}{2} \sum_{n=1}^N \sum_{m=1}^N a_n a_m t_n t_m (\phi(\mathbf{x}_m)^\top \phi(\mathbf{x}_n))$$

$$\begin{aligned} y(\mathbf{x}) &= \mathbf{w}^\top \phi(\mathbf{x}) + b \\ &= \sum_{n=1}^N a_n t_n \phi(\mathbf{x}_n)^\top \phi(\mathbf{x}) + b \end{aligned}$$

$$\boxed{\mathbf{w} = \sum_{n=1}^N a_n t_n \phi(\mathbf{x}_n)}$$

The Kernel Trick

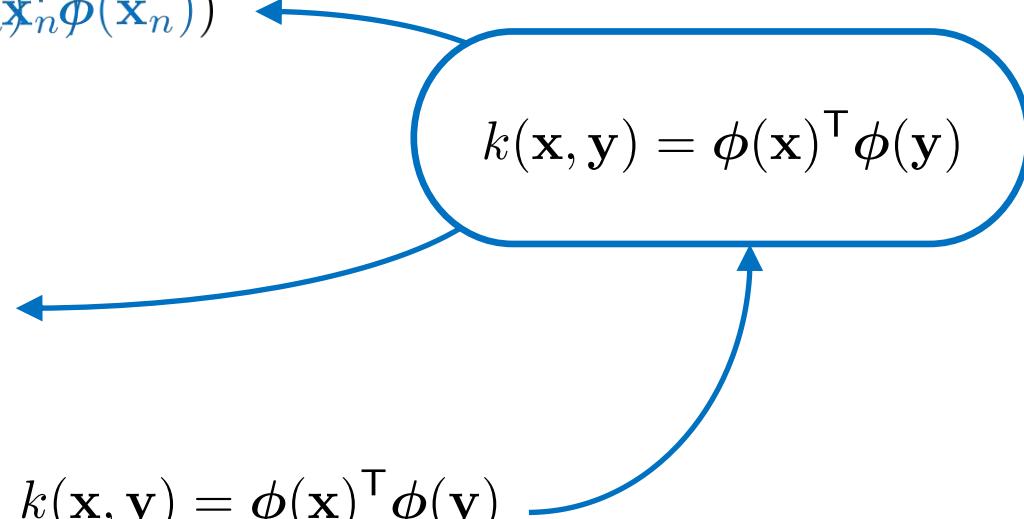
- On a closer look, $\phi(\mathbf{x})$ only appears in the form of dot products:

$$L_d(\mathbf{a}) = \sum_{n=1}^N a_n - \frac{1}{2} \sum_{n=1}^N \sum_{m=1}^N a_n a_m t_n t_m k(\phi(\mathbf{x}_m)^\top \phi(\mathbf{x}_n))$$

$$y(\mathbf{x}) = \mathbf{w}^\top \phi(\mathbf{x}) + b$$

$$= \sum_{n=1}^N a_n t_n k(\phi(\mathbf{x}_n)^\top \phi(\mathbf{x})) + b$$

Define a kernel function $k(\mathbf{x}, \mathbf{y}) = \phi(\mathbf{x})^\top \phi(\mathbf{y})$
 \Rightarrow Use the kernel instead of the dot product.

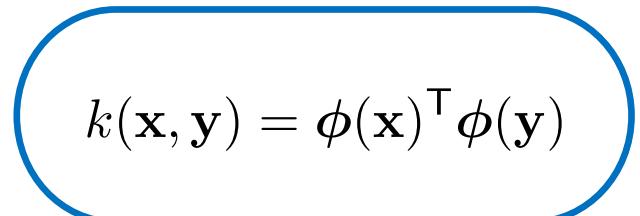


The Kernel Trick

- On a closer look, $\phi(\mathbf{x})$ only appears in the form of dot products:

$$L_d(\mathbf{a}) = \sum_{n=1}^N a_n - \frac{1}{2} \sum_{n=1}^N \sum_{m=1}^N a_n a_m t_n t_m k(\mathbf{x}_m, \mathbf{x}_n)$$

$$\begin{aligned} y(\mathbf{x}) &= \mathbf{w}^\top \phi(\mathbf{x}) + b \\ &= \sum_{n=1}^N a_n t_n k(\mathbf{x}_n, \mathbf{x}) + b \end{aligned}$$



$$k(\mathbf{x}, \mathbf{y}) = \phi(\mathbf{x})^\top \phi(\mathbf{y})$$

Define a kernel function $k(\mathbf{x}, \mathbf{y}) = \phi(\mathbf{x})^\top \phi(\mathbf{y})$
 \Rightarrow Use the kernel instead of the dot product.

- $k(\cdot, \cdot)$ implicitly maps the data to some higher-dimensional space, without having to compute $\phi(\mathbf{x})$.

When Can We Apply the Kernel Trick?

- In order for this to work, $k(\cdot, \cdot)$ needs to define an implicit mapping.
- Formally
 - A function $k(\mathbf{x}_1, \mathbf{x}_2) : \mathbb{R}^D \times \mathbb{R}^D \rightarrow \mathbb{R}$ is a **kernel function**, iff
 - There is a mapping $\phi(\mathbf{x}) : \mathbb{R}^D \rightarrow \mathcal{H}$ such that
- *When will this be the case?*

$$k(\mathbf{x}_1, \mathbf{x}_2) = \phi(\mathbf{x}_1)^T \phi(\mathbf{x}_2) \quad \forall \mathbf{x}_1, \mathbf{x}_2$$

- When is a function $k(\mathbf{x}_1, \mathbf{x}_2)$ a valid kernel function? Two ways to check:

1. Every **Gram matrix** K of k is symmetric positive definite:

$$K = \begin{bmatrix} k(\mathbf{x}_1, \mathbf{x}_1) & \cdots & k(\mathbf{x}_1, \mathbf{x}_N) \\ \vdots & \ddots & \vdots \\ k(\mathbf{x}_N, \mathbf{x}_1) & \cdots & k(\mathbf{x}_N, \mathbf{x}_N) \end{bmatrix}$$

A matrix M is **positive definite** if all eigenvalues of K are positive.

- This is easy to verify for a given training set $\{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N\}$.
- Unfortunately, it has to hold for *every possible* such set.

\Rightarrow Very hard to prove in practice.

- When is a function $k(\mathbf{x}_1, \mathbf{x}_2)$ a valid kernel function? Two ways to check:

2. We can construct valid kernels from other valid kernels:

- Given valid kernels $k_1(\mathbf{x}, \mathbf{x}')$ and $k_2(\mathbf{x}, \mathbf{x}')$, the following combinations will also be valid

$$k(\mathbf{x}, \mathbf{x}') = c \cdot k_1(\mathbf{x}, \mathbf{x}')$$

$$k(\mathbf{x}, \mathbf{x}') = f(\mathbf{x})k_1(\mathbf{x}, \mathbf{x}')f(\mathbf{x}')$$

$$k(\mathbf{x}, \mathbf{x}') = \text{polynomial}(k_1(\mathbf{x}, \mathbf{x}')) \quad (\text{with nonnegative coefficients})$$

$$k(\mathbf{x}, \mathbf{x}') = \exp(k_1(\mathbf{x}, \mathbf{x}'))$$

$$k(\mathbf{x}, \mathbf{x}') = k_1(\mathbf{x}, \mathbf{x}') + k_2(\mathbf{x}, \mathbf{x}')$$

$$k(\mathbf{x}, \mathbf{x}') = k_1(\mathbf{x}, \mathbf{x}')k_2(\mathbf{x}, \mathbf{x}')$$

$$k(\mathbf{x}, \mathbf{x}') = k_3(\phi(\mathbf{x}), \phi(\mathbf{x}'))$$

$$k(\mathbf{x}, \mathbf{x}') = \mathbf{x}^\top \mathbf{A} \mathbf{x}'$$

\Rightarrow Much easier to apply in practice.

New SVM Formulation

- Maximize

$$L_d(\mathbf{a}) = \sum_{n=1}^N a_n - \frac{1}{2} \sum_{n=1}^N \sum_{m=1}^N a_n a_m t_n t_m k(\mathbf{x}_m, \mathbf{x}_n)$$

under the constraints $0 \leq a_n \leq C \quad \forall n$

$$\sum_{n=1}^N a_n t_n = 0$$

- Classify new data points using

$$y(\mathbf{x}) = \sum_{n=1}^N a_n t_n k(\mathbf{x}_n, \mathbf{x}) + b$$

Example: Polynomial Kernel

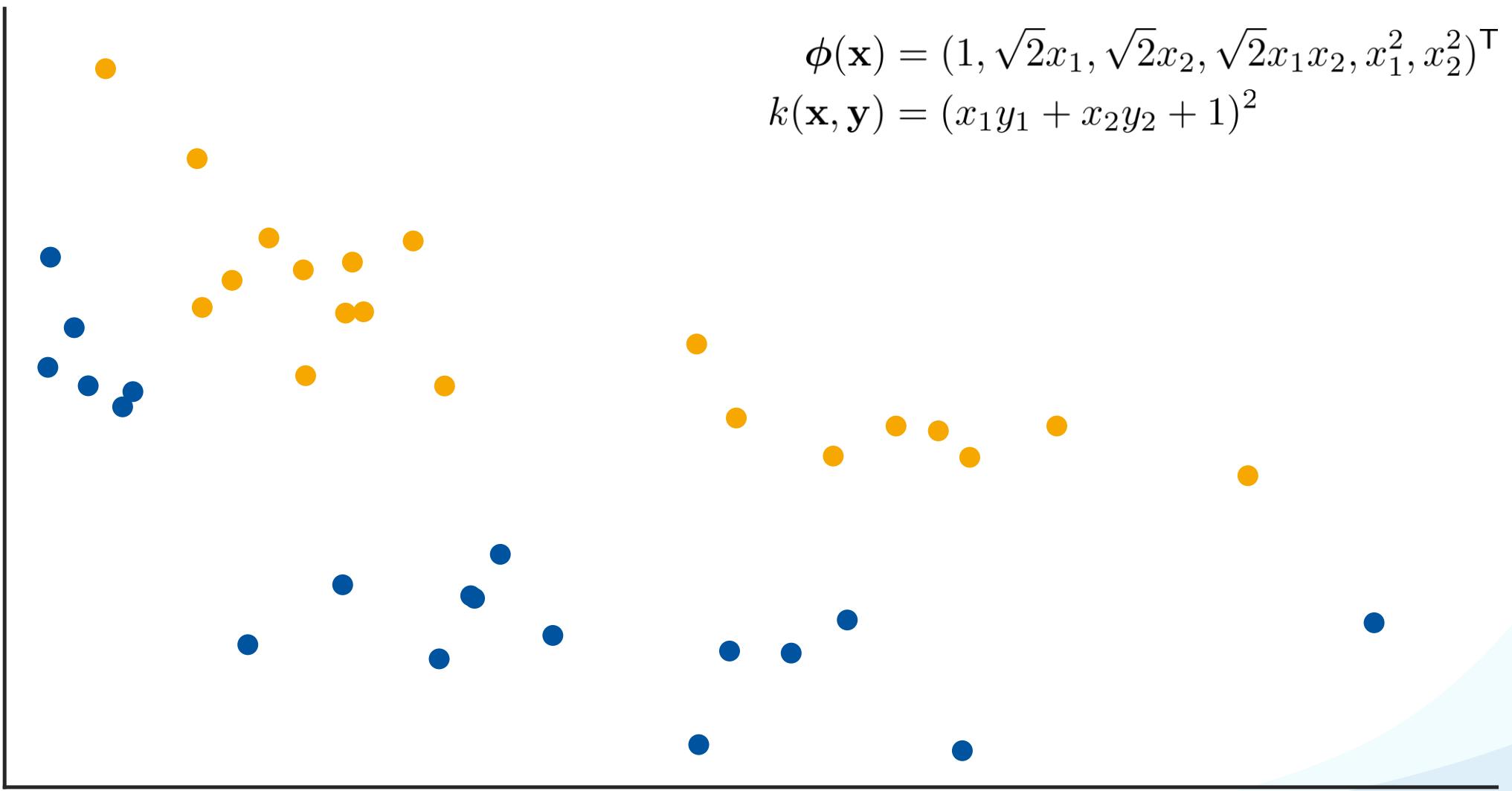
- We slightly adjust the polynomial basis function that we know:

$$\phi(\mathbf{x}) = (1, \sqrt{2}x_1, \sqrt{2}x_2, \sqrt{2}x_1x_2, x_1^2, x_2^2)^T$$

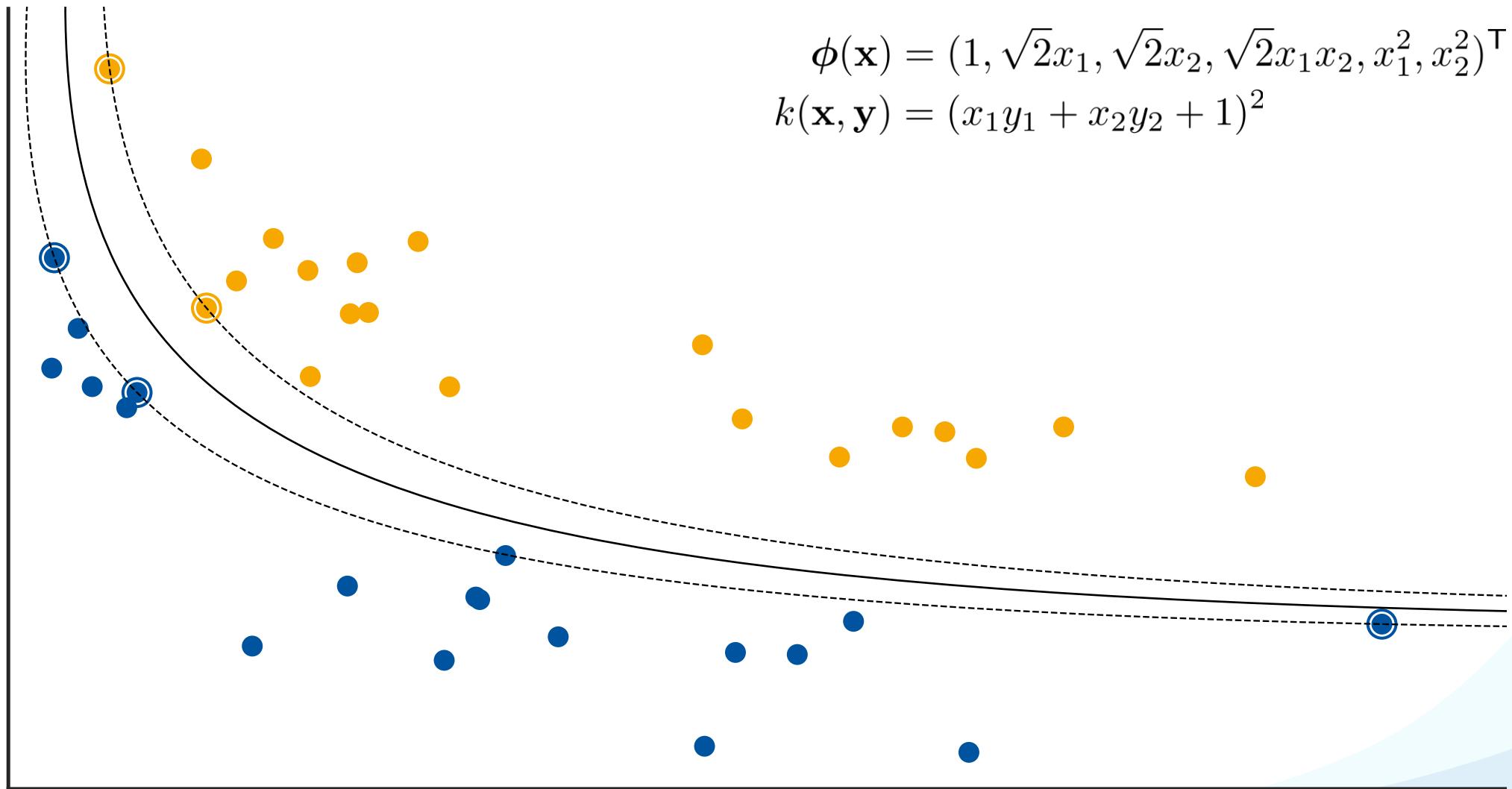
$$k(\mathbf{x}, \mathbf{y}) = (\mathbf{x}^T \mathbf{y} + 1)^2 = \phi(\mathbf{x})^T \phi(\mathbf{y})$$

- In fact, $(\mathbf{x}^T \mathbf{y} + 1)^p$ is the kernel function for a polynomial of degree p .

Example



Example



Advantages

- We can use high-dimensional or even infinite dimensional feature spaces
 - Since $\phi(\mathbf{x})$ is never computed explicitly.
- We can work with non-vector space data
 - We can define kernel functions for arbitrary data types!
 - Graphs, Sets, Sequences, Histograms,
 - ...
- Simple to use and work very well in most cases

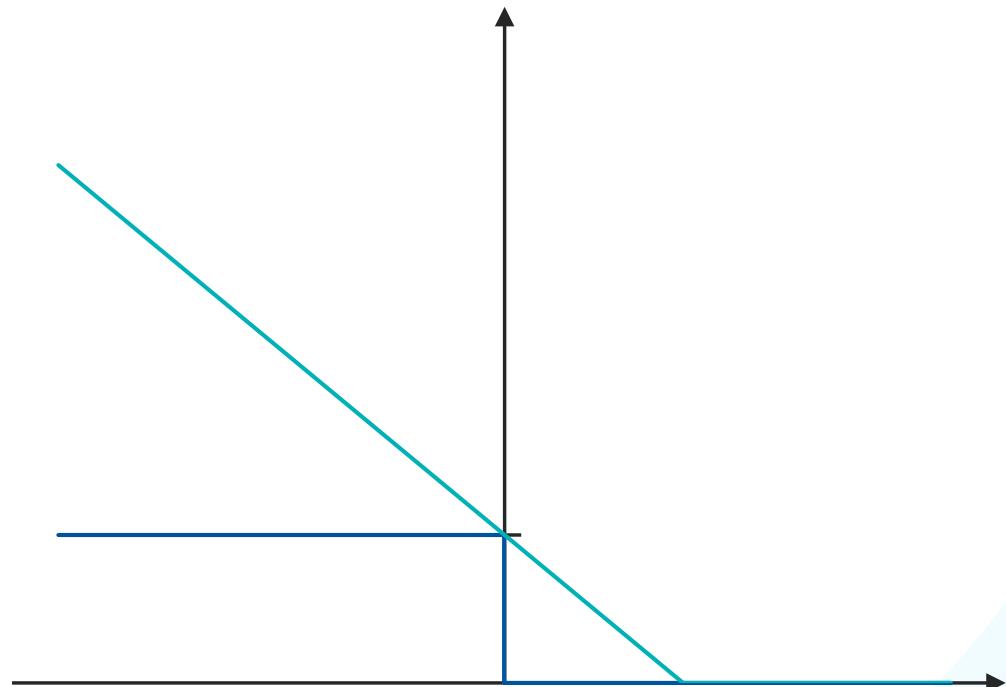
Limitations

- Which kernel to choose?
 - **Model selection** problem
- How to choose kernel parameters?
 - **Hyperparameter optimization** problem, usually solved by performing a **grid search** over the validation set
- Evaluation speed scales linearly with number of support vectors

$$y(\mathbf{x}) = \sum_{n=1}^N a_n t_n k(\mathbf{x}_n, \mathbf{x}) + b$$

Support Vector Machines

1. Maximum Margin Classification
2. Primal Formulation
3. Dual Formulation
4. Soft-Margin SVMs
5. Non-linear SVMs
6. **Error Function Analysis**



Error Function Analysis

- We know how to formulate and optimize an SVM as a convex optimization problem:

$$\arg \min_{\mathbf{w}, b, \xi_n \in \mathbb{R}^+} \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{n=1}^N \xi_n$$

subject to the constraints

$$t_n y(\mathbf{x}_n) \geq 1 - \xi_n$$

Error Function Analysis

- We know how to formulate and optimize an SVM as a convex optimization problem:

$$\arg \min_{\mathbf{w}, b, \xi_n \in \mathbb{R}^+} \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{n=1}^N \xi_n$$

subject to the constraints

$$t_n y(\mathbf{x}_n) \geq 1 - \xi_n$$

But what error function does this correspond to?

- Integrate the constraints into the objective function:
 - Rewrite as $\xi_n \geq 1 - t_n y(\mathbf{x}_n)$
 - Thus, we obtain

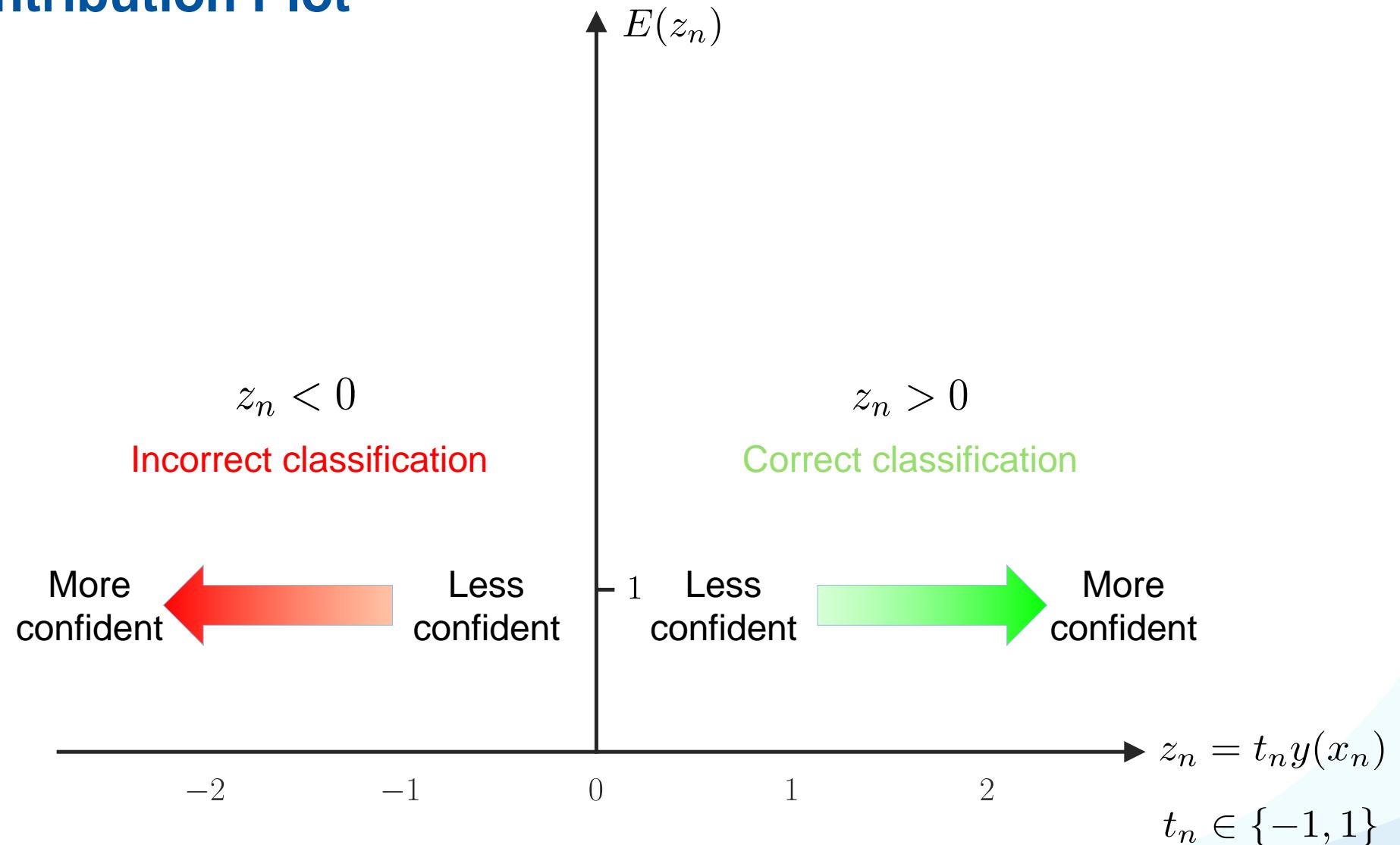
$$\min_{\mathbf{w}, b} E(\mathbf{w}) = \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{n=1}^N [1 - t_n y(\mathbf{x}_n)]_+$$
$$[x]_+ \equiv \max\{x, 0\}$$

The Hinge Loss

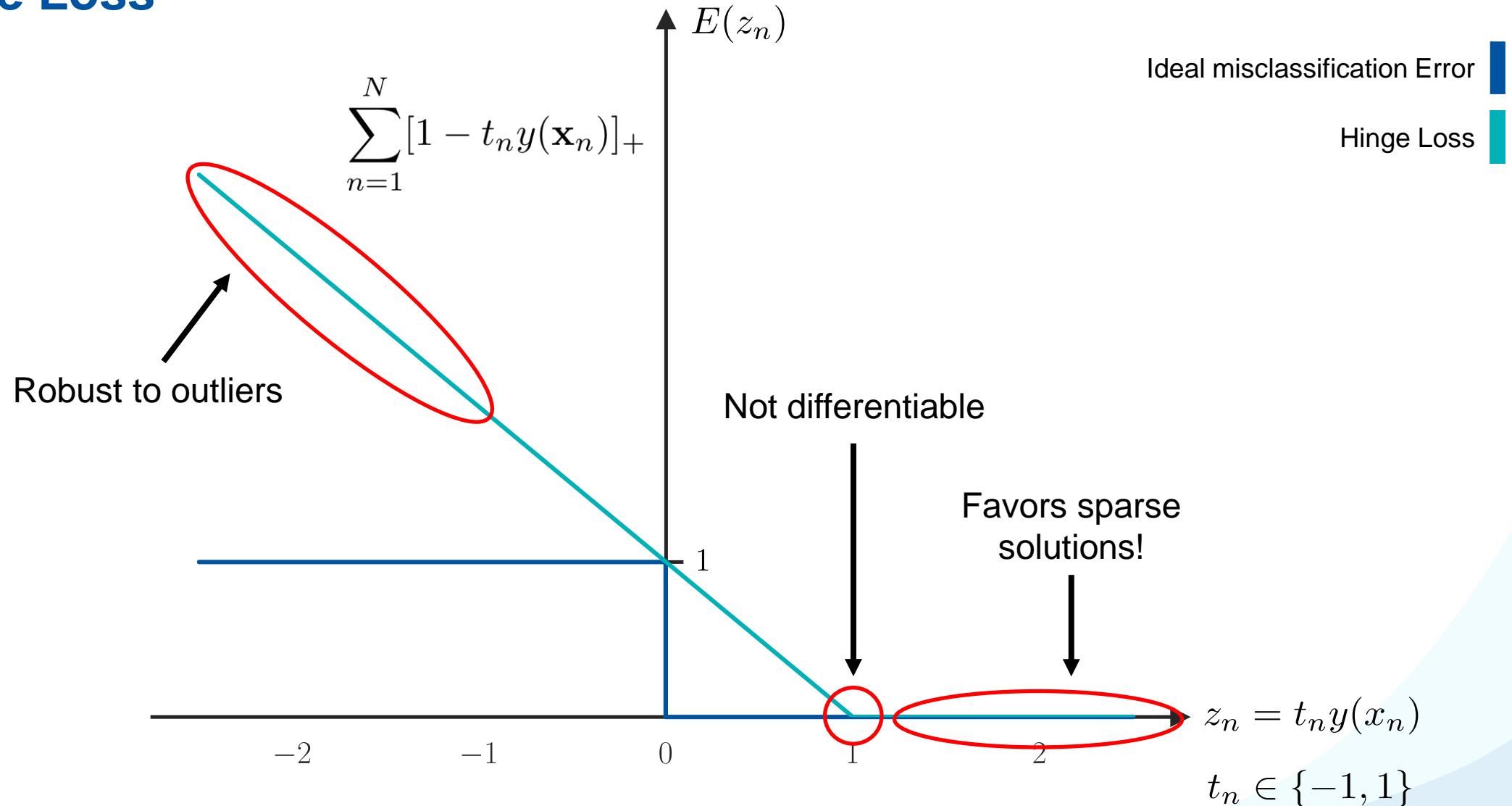
$$E(\mathbf{w}) = \underbrace{\frac{1}{2} \|\mathbf{w}\|^2}_{L_2 \text{ regularization}} + C \underbrace{\sum_{n=1}^N [1 - t_n y(\mathbf{x}_n)]_+}_{\text{Hinge loss}}$$

- Regularization bounds parameter size.
- Hinge Loss enforces sparsity:
 - Only a **subset of training data points** actually influences the decision boundary.
 - Still, all input dimensions are used.
- This formulation corresponds to an unconstrained optimization of a non-differentiable function.
 - Very efficient: stochastic (sub-)gradient descent.

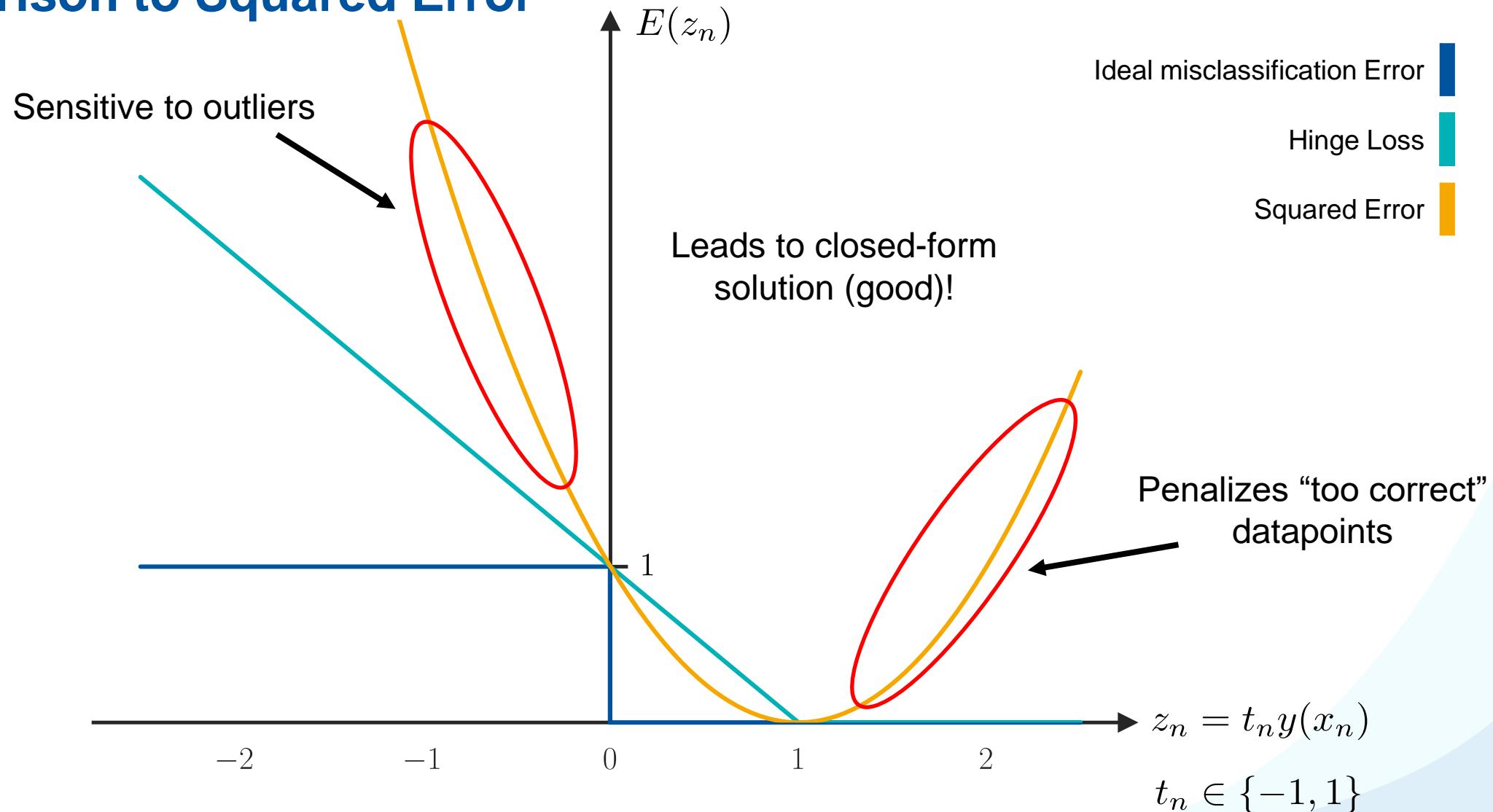
Error Contribution Plot



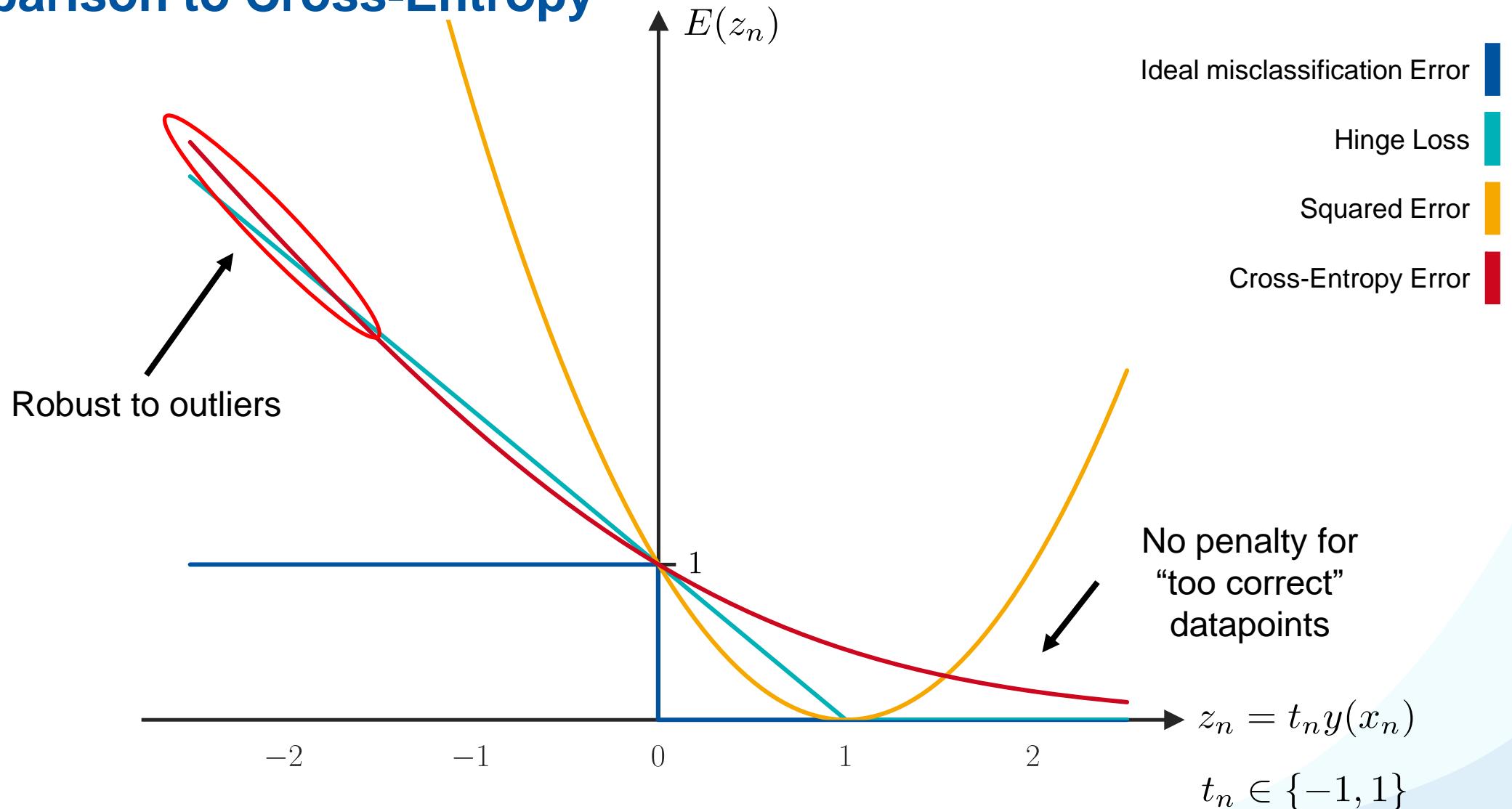
Hinge Loss



Comparison to Squared Error



Comparison to Cross-Entropy



Discussion: Hinge Loss

Advantages

- Favors sparse solutions that only depend on a subset of training data points.
- Robust to outliers (only a linear penalty for misclassified points).
- Convex function, unique minimum exists.

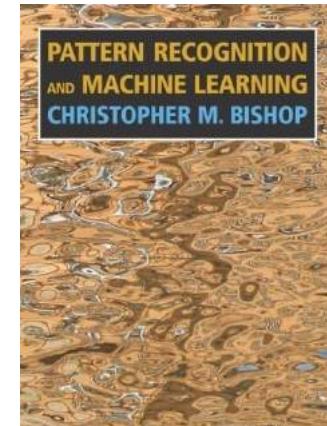
Limitations

- Not differentiable (cannot minimize this loss using standard gradient descent, but need to use [subgradient descent](#)).

References and Further Reading

- More information about SVMs is available in Chapter 7.1 of Bishop's book.

Christopher M. Bishop
Pattern Recognition and Machine Learning
Springer, 2006



Elements of Machine Learning & Data Science

Winter semester 2023/24

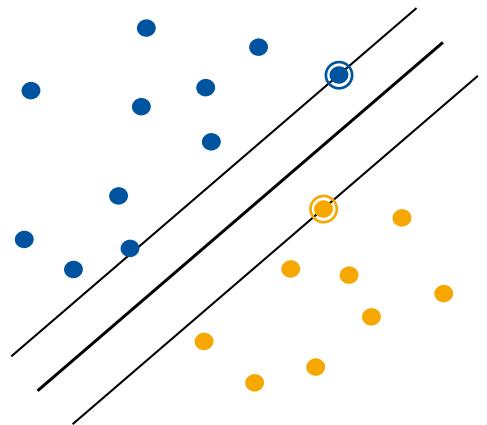
Lecture 19 – Neural Networks Basics

19.12.2023

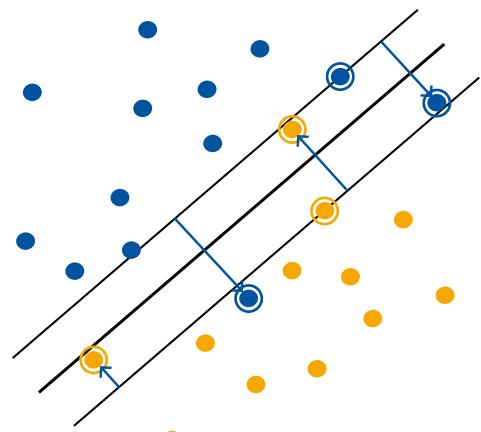
Prof. Bastian Leibe

Machine Learning Topics

1. Introduction to ML
2. Probability Density Estimation
3. Linear Discriminants
4. Linear Regression
5. Logistic Regression
6. **Support Vector Machines**
7. (AdaBoost)
8. Neural Network Basics



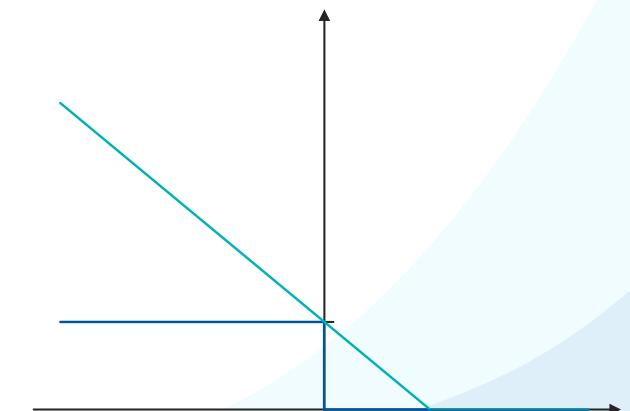
Maximum Margin Classification



Soft-Margin SVM

$$L_p(\mathbf{w}, b, \mathbf{a})$$
$$L_d(\mathbf{a})$$

Primal & Dual Form



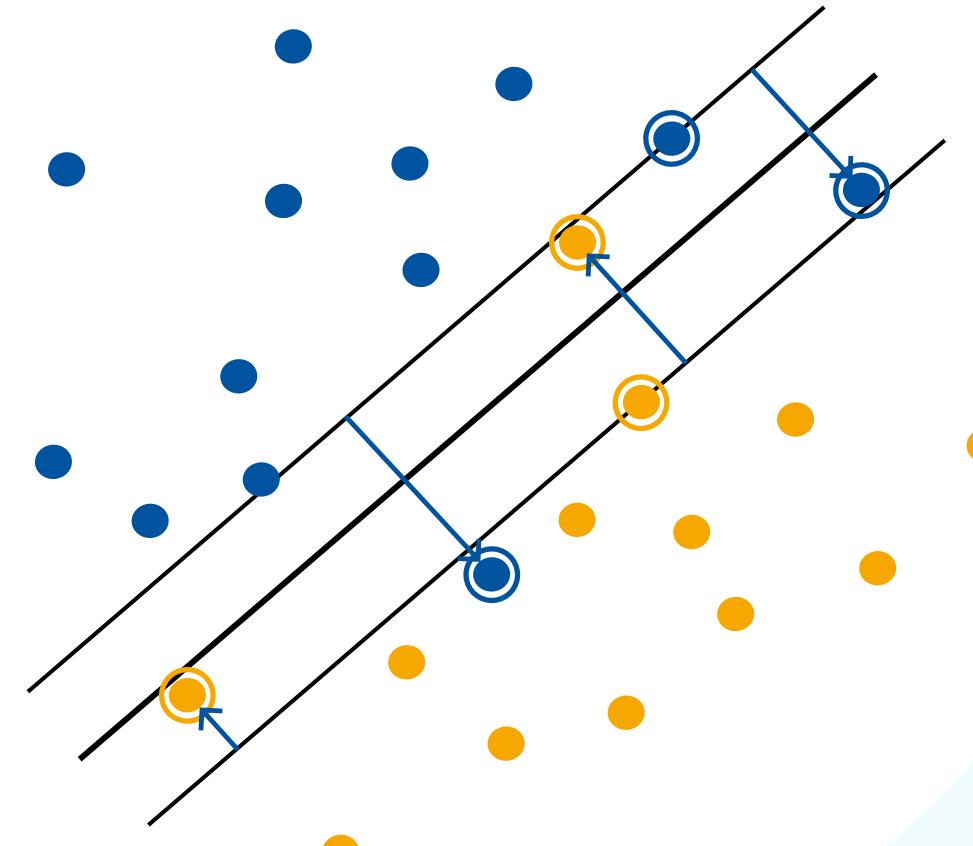
Hinge Loss

Recap: Soft-Margin SVM

- Idea: Introduce **slack variables** $\xi_n \geq 0$
 - One slack variable ξ_n for each training point
- Effect
 - $\xi_n = 0$ for points on the correct side.
 - **Linear penalty** for all other points:
$$\xi_n = |t_n - y(\mathbf{x}_n)|$$
- Slack variables are **jointly optimized** with \mathbf{w} :

$$\min \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{n=1}^N \xi_n$$

where C is a tradeoff parameter.



Recap: Soft-Margin SVM Primal Form

- Minimize

$$L_p = \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{n=1}^N \xi_n - \underbrace{\sum_{n=1}^N a_n [t_n(y(\mathbf{x}_n) - 1 + \xi_n)]}_{\text{Constraint}} - \underbrace{\sum_{n=1}^N \mu_n \xi_n}_{\text{Constraint}}$$
$$t_n y(\mathbf{x}_n) \geq 1 - \xi_n \quad \xi_n \geq 0$$

- KKT conditions

$$\begin{array}{ll} a_n \geq 0 & \mu_n \geq 0 \\ t_n y(\mathbf{x}_n) - 1 + \xi_n \geq 0 & \xi_n \geq 0 \\ a_n [t_n y(\mathbf{x}_n) - 1 + \xi_n] = 0 & \mu_n \xi_n = 0 \end{array}$$

Recap: Soft-Margin SVM Dual Form

- Maximize

$$L_d(\mathbf{a}) = \sum_{n=1}^N a_n - \frac{1}{2} \sum_{n=1}^N \sum_{m=1}^N a_n a_m t_n t_m (\mathbf{x}_m^\top \mathbf{x}_n)$$

- Under the side conditions

$$0 \leq a_n \leq C \quad \forall n$$

$$\sum_{n=1}^N a_n t_n = 0$$

This is the only difference to before.

Recap: Non-linear SVM Formulation

- Maximize

$$L_d(\mathbf{a}) = \sum_{n=1}^N a_n - \frac{1}{2} \sum_{n=1}^N \sum_{m=1}^N a_n a_m t_n t_m k(\mathbf{x}_m, \mathbf{x}_n)$$

under the constraints $0 \leq a_n \leq C \quad \forall n$

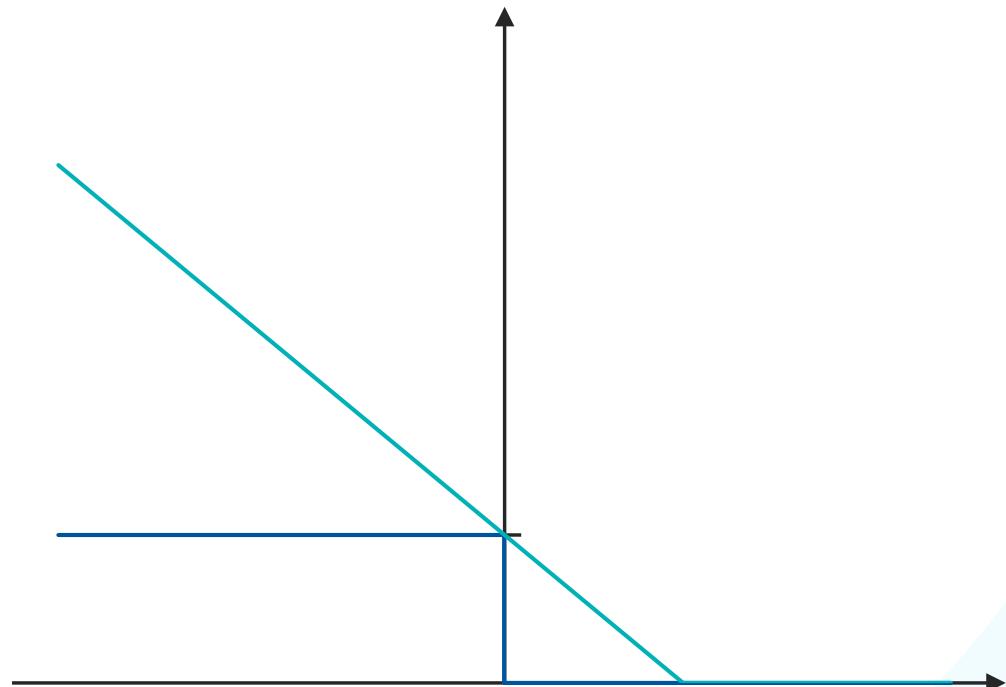
$$\sum_{n=1}^N a_n t_n = 0$$

- Classify new data points using

$$y(\mathbf{x}) = \sum_{n=1}^N a_n t_n k(\mathbf{x}_n, \mathbf{x}) + b$$

Support Vector Machines

1. Maximum Margin Classification
2. Primal Formulation
3. Dual Formulation
4. Soft-Margin SVMs
5. Non-linear SVMs
6. **Error Function Analysis**



Error Function Analysis

- We know how to formulate and optimize an SVM as a convex optimization problem:

$$\arg \min_{\mathbf{w}, b, \xi_n \in \mathbb{R}^+} \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{n=1}^N \xi_n$$

subject to the constraints

$$t_n y(\mathbf{x}_n) \geq 1 - \xi_n$$

Error Function Analysis

- We know how to formulate and optimize an SVM as a convex optimization problem:

$$\arg \min_{\mathbf{w}, b, \xi_n \in \mathbb{R}^+} \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{n=1}^N \xi_n$$

But what error function does this correspond to?

subject to the constraints

$$t_n y(\mathbf{x}_n) \geq 1 - \xi_n$$

- Integrate the constraints into the objective function:
 - Rewrite as $\xi_n \geq 1 - t_n y(\mathbf{x}_n)$
 - Thus, we obtain

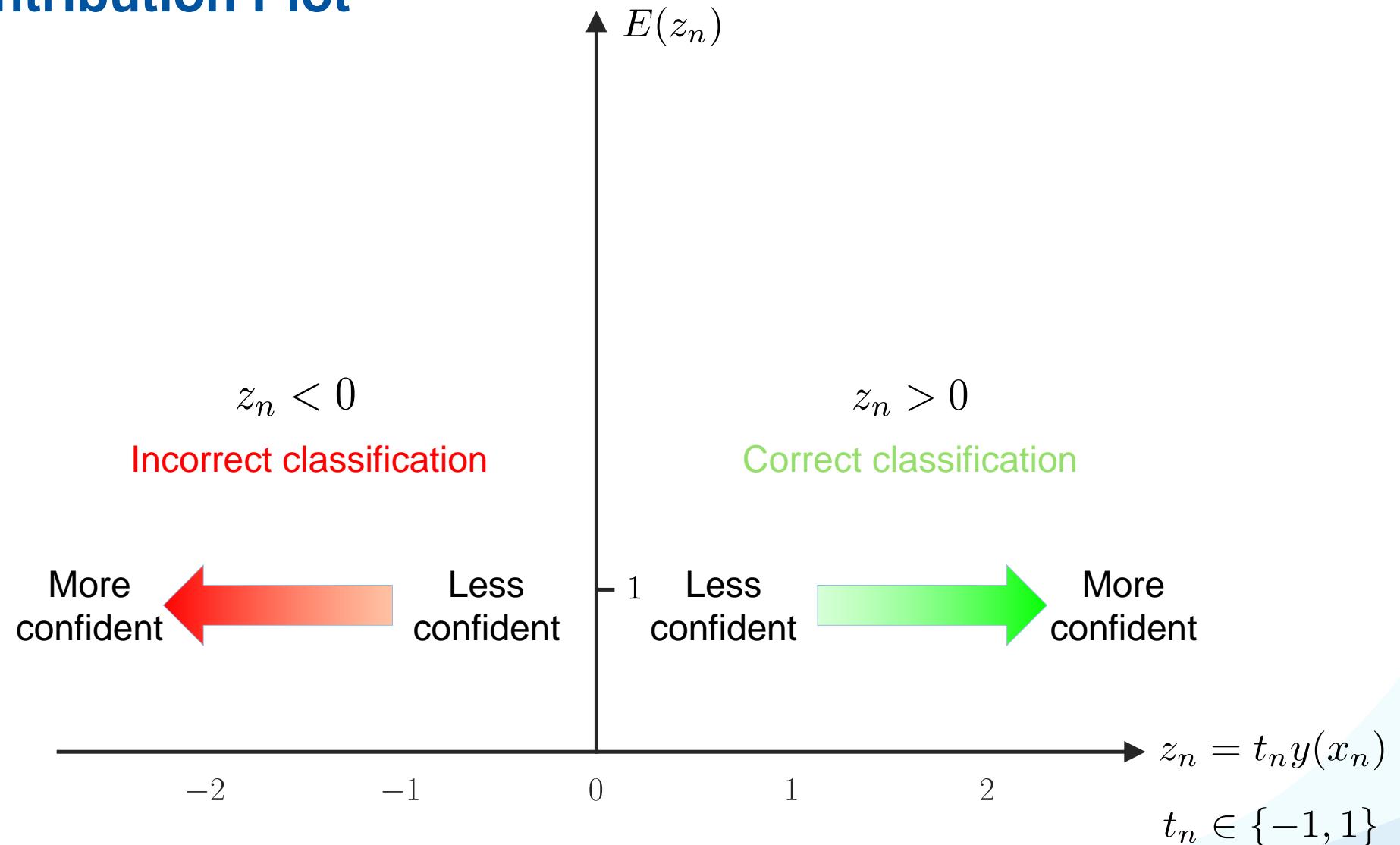
$$\min_{\mathbf{w}, b} E(\mathbf{w}) = \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{n=1}^N [1 - t_n y(\mathbf{x}_n)]_+$$
$$[x]_+ \equiv \max\{x, 0\}$$

The Hinge Loss

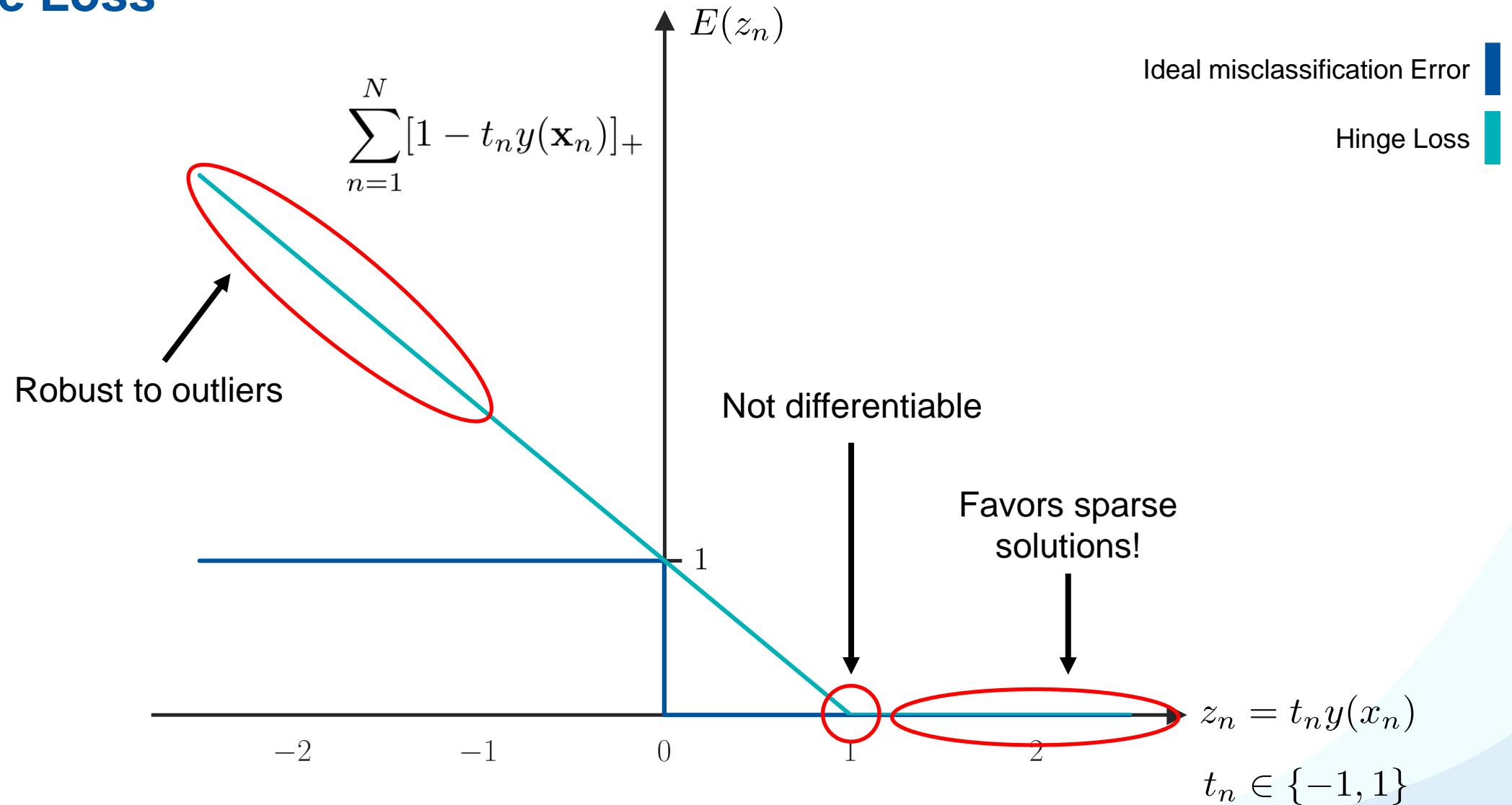
$$E(\mathbf{w}) = \underbrace{\frac{1}{2} \|\mathbf{w}\|^2}_{L_2 \text{ regularization}} + C \underbrace{\sum_{n=1}^N [1 - t_n y(\mathbf{x}_n)]_+}_{\text{Hinge loss}}$$

- Regularization bounds parameter size.
- Hinge Loss enforces sparsity:
 - Only a **subset of training data points** actually influences the decision boundary.
 - Still, all input dimensions are used.
- This formulation corresponds to an unconstrained optimization of a non-differentiable function.
 - Very efficient: stochastic (sub-)gradient descent.

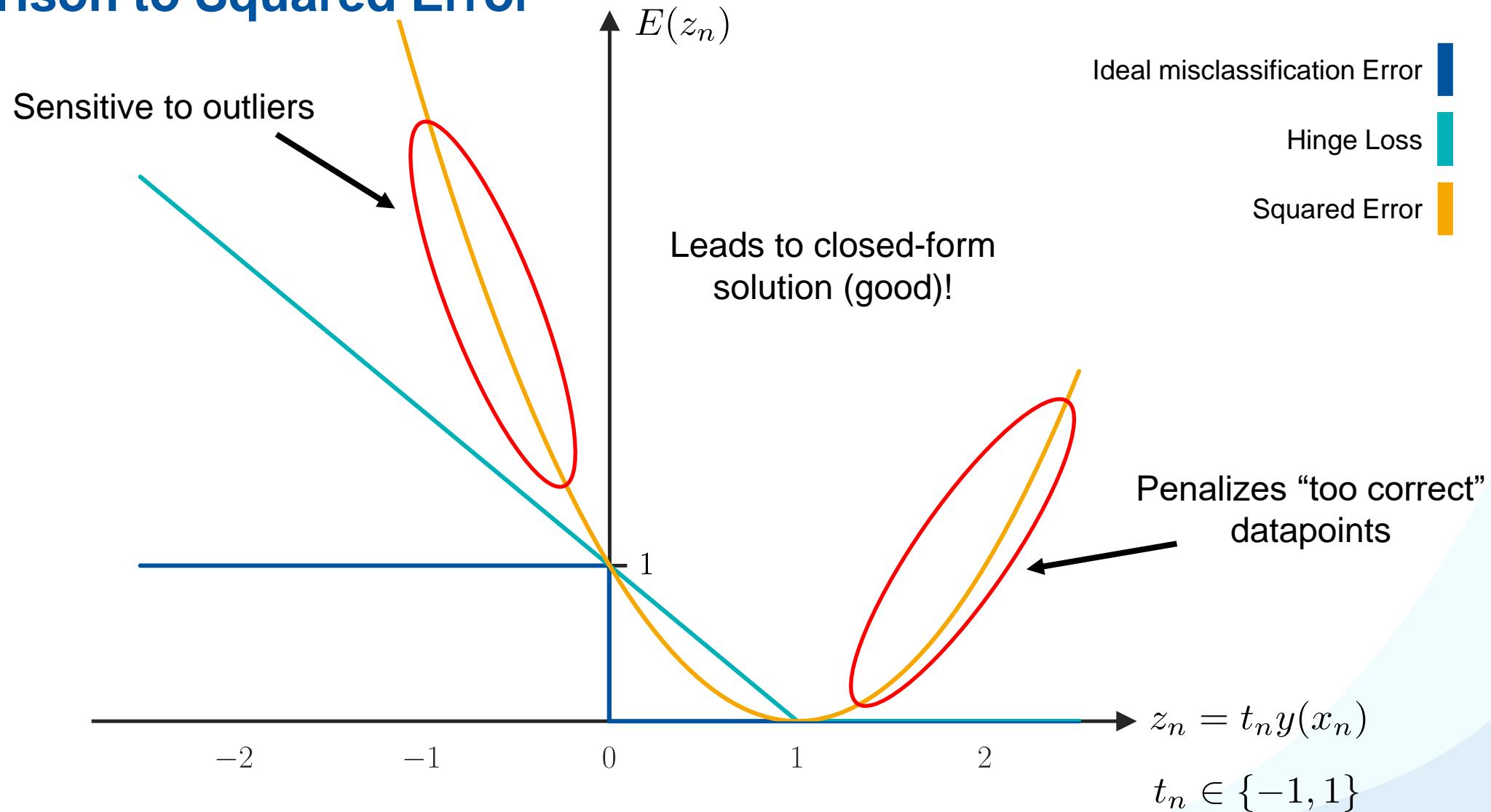
Error Contribution Plot



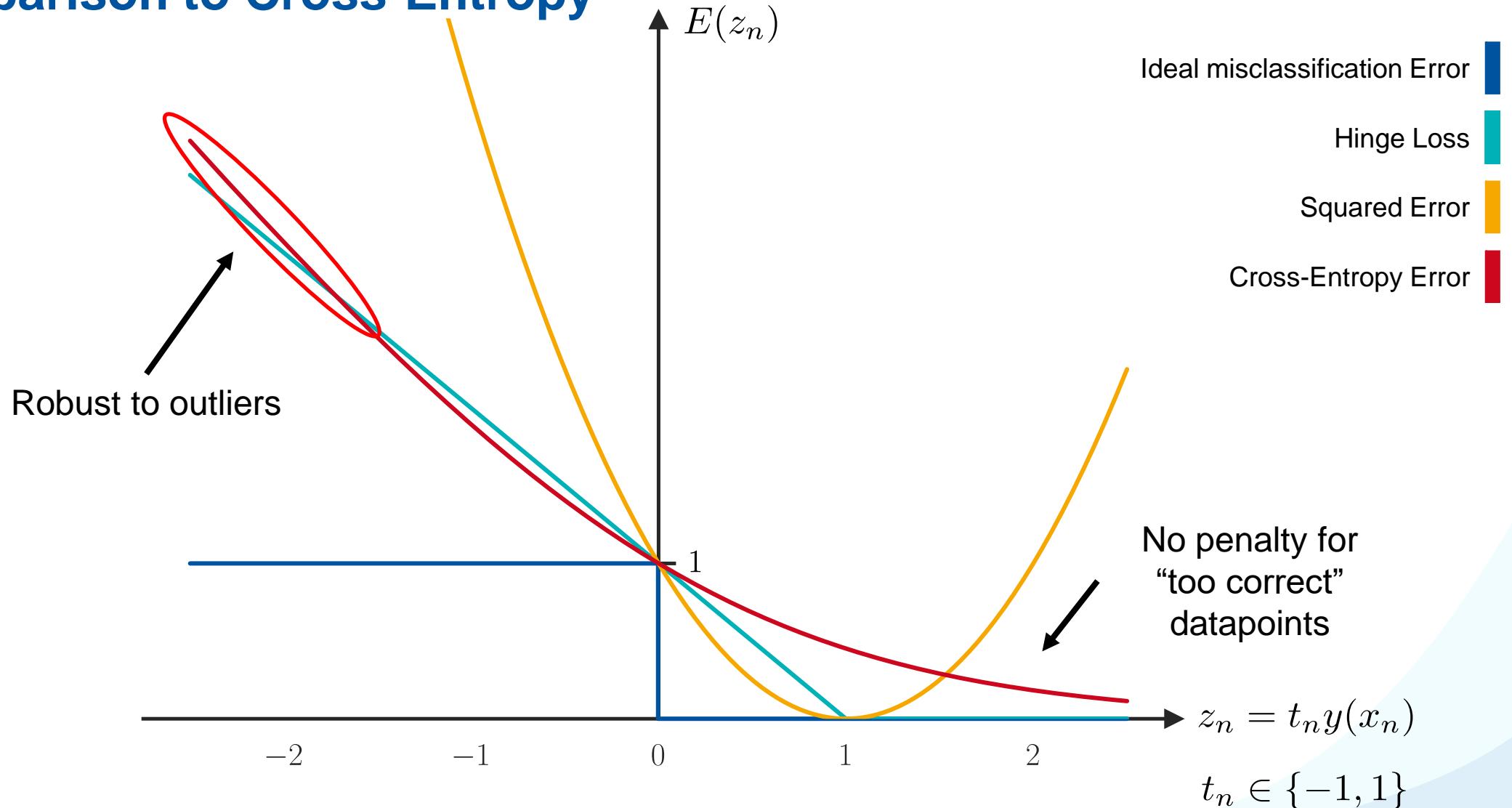
Hinge Loss



Comparison to Squared Error



Comparison to Cross-Entropy



Discussion: Hinge Loss

Advantages

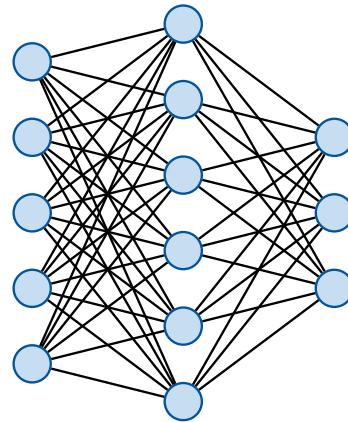
- Favors sparse solutions that only depend on a subset of training data points.
- Robust to outliers (only a linear penalty for misclassified points).
- Convex function, unique minimum exists.

Limitations

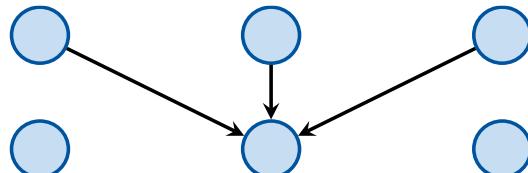
- Not differentiable (cannot minimize this loss using standard gradient descent, but need to use [subgradient descent](#)).

Machine Learning Topics

1. Introduction to ML
2. Probability Density Estimation
3. Linear Discriminants
4. Linear Regression
5. Logistic Regression
6. Support Vector Machines
7. (AdaBoost)
8. **Neural Network Basics**



Multi-Layer Perceptrons

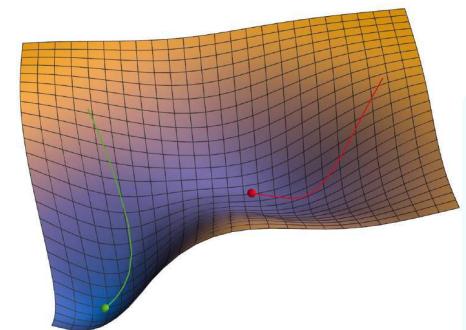


Backpropagation

$$\frac{1}{2}(y(\mathbf{x}) - t)^2$$

$$[1 - ty(\mathbf{x})]_+ - \sum_k \left(\mathbb{I}(t = k) \ln \frac{\exp(y_k(\mathbf{x}))}{\sum_j \exp(y_j(\mathbf{x}))} \right) \\ \frac{1}{2} \|\mathbf{w}\|^2$$

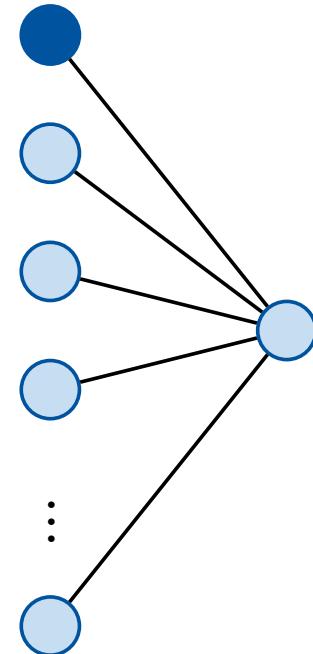
Losses & Regularizers



Stochastic Gradient Descent

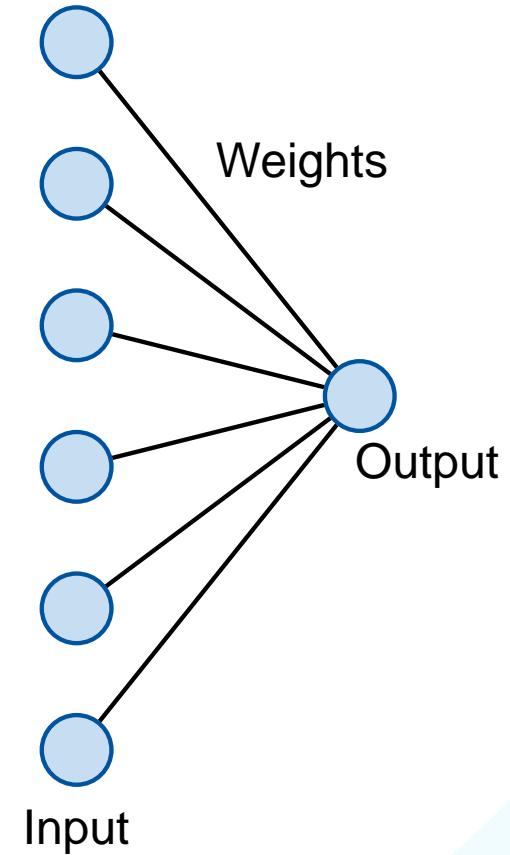
Neural Network Basics

1. Perceptrons
2. Multi-Layer Perceptrons
3. Loss Functions
4. Backpropagation
5. Stochastic Gradient Descent



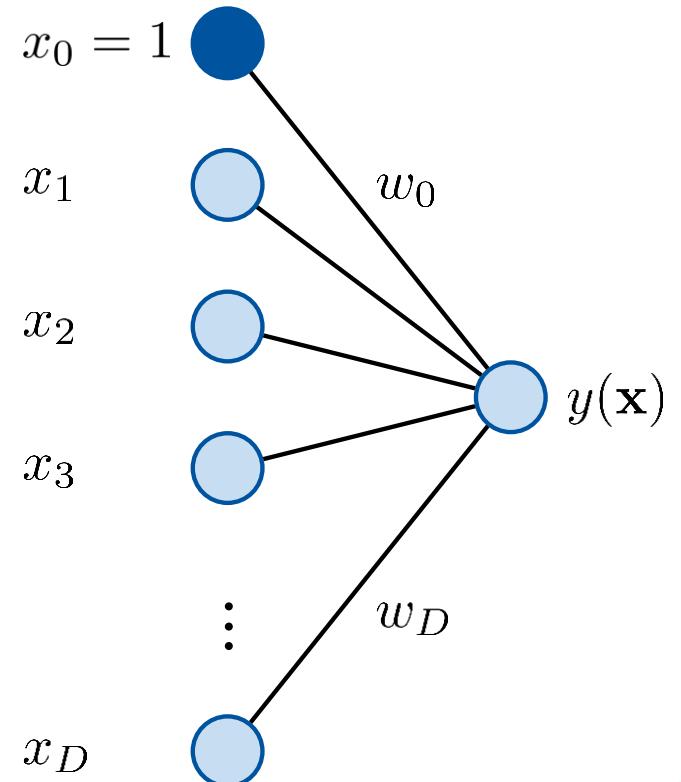
Perceptrons

- Inspired by biological neurons.
- The output is determined by the activation of the input nodes and a set of weights connecting input and output layers.



Basic Perceptron

- Input Layer:
 - Hand-designed features
- Outputs:
 - Linear outputs
$$y(\mathbf{x}) = \mathbf{w}^T \mathbf{x} + w_0$$
 - Logistic outputs
$$y(\mathbf{x}) = \sigma(\mathbf{w}^T \mathbf{x} + w_0)$$
- Learning: finding optimal weights \mathbf{w} .



Multi-Class Networks

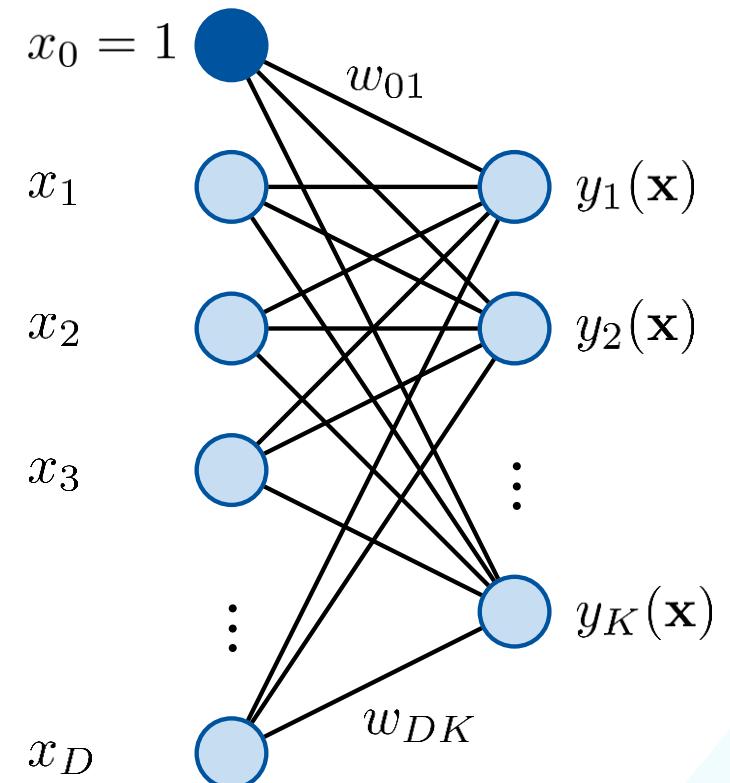
- One output node per class:
 - Linear outputs

$$y_k(\mathbf{x}) = \sum_{i=0}^D w_{ki} \mathbf{x}_i$$

- Logistic outputs

$$y_k(\mathbf{x}) = \sigma \left(\sum_{i=0}^D w_{ki} \mathbf{x}_i \right)$$

- We can do **multidimensional linear regression** or **multiclass classification** this way.



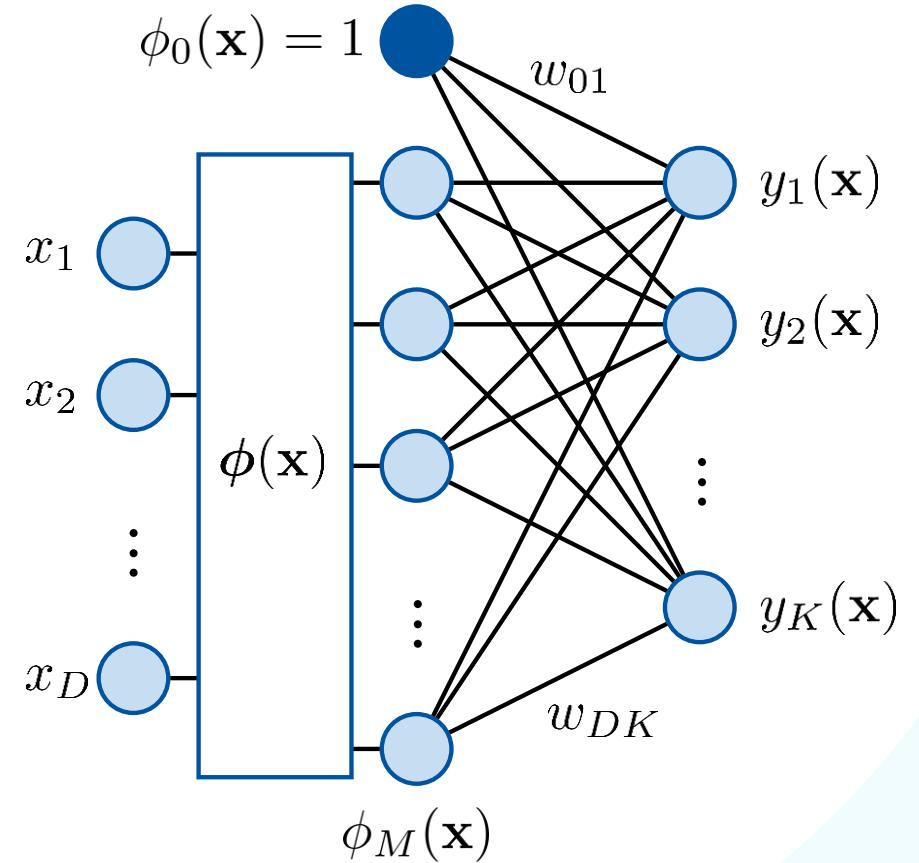
Non-Linear Basis Functions

- Apply a (fixed) mapping $\phi(\mathbf{x})$ to inputs:
 - Linear outputs

$$y_k(\mathbf{x}) = \sum_{j=0}^M w_{kj} \phi_j(\mathbf{x})$$

- Logistic outputs

$$y_k(\mathbf{x}) = \sigma \left(\sum_{j=0}^M w_{kj} \phi_j(\mathbf{x}) \right)$$



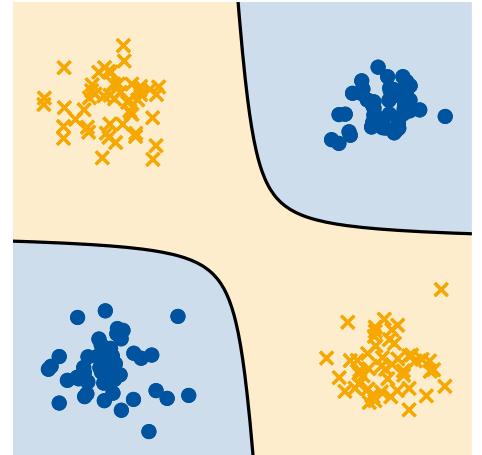
Connections to linear discriminants

- All of this should feel very familiar.
- Perceptrons are generalized linear discriminants!
- What does that mean?
 - We have the same limitations as before.
 - Can model any separable function perfectly, given the right input features.
 - For some tasks, this may require an exponential number of input features.

⇒ *It is the feature design that solves the task!*

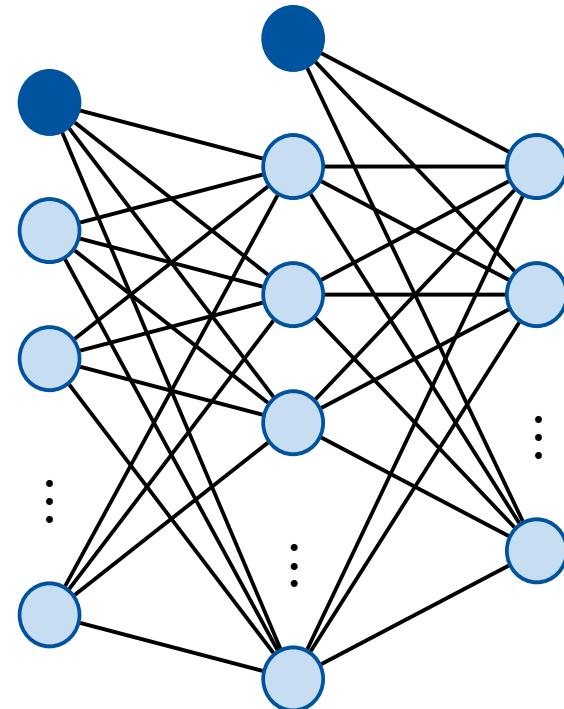
Limitations so far

- Generalized linear discriminants (including perceptrons) are very limited.
 - A linear classifier cannot solve certain problems (e.g., XOR).
 - However, with a non-linear classifier based on suitable features, the problem becomes solvable.
 - So far, we have designed the features and kernels by hand.
- ⇒ How can we *learn* good feature representations?



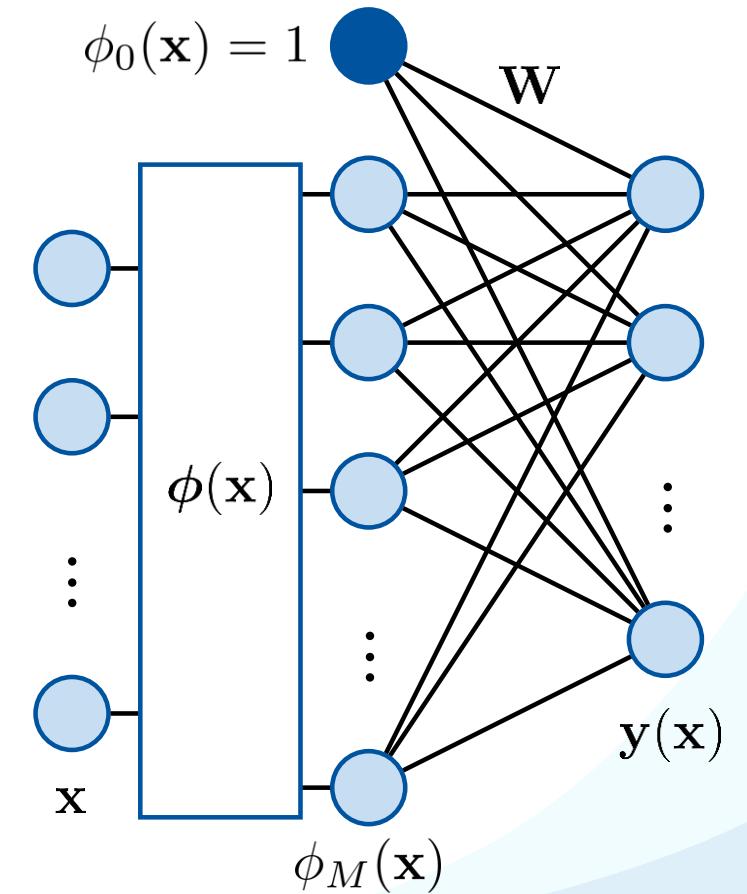
Neural Network Basics

1. Perceptrons
2. **Multi-Layer Perceptrons**
3. Loss Functions
4. Backpropagation
5. Stochastic Gradient Descent



Multi-Layer Perceptrons

- Perceptrons are limited by having a fixed input mapping.



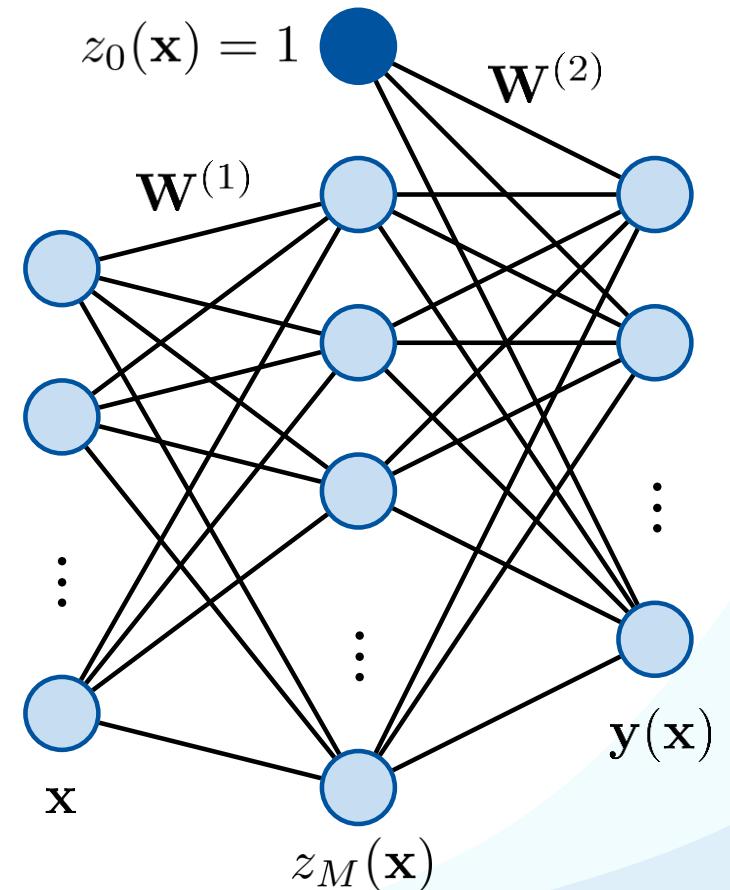
Multi-Layer Perceptrons

- Perceptrons are limited by having a fixed input mapping.
- Replace it with a **hidden layer** that learns suitable features.
- Output of hidden layer is input to next layer.
- Each layer computes a matrix multiplication and applies an elementwise **activation function** $g(\cdot)$:

$$\mathbf{z}(\mathbf{x}) = g^{(1)} \left(\mathbf{W}^{(1)} \mathbf{x} \right)$$

$$\mathbf{y}(\mathbf{x}) = g^{(2)} \left(\mathbf{W}^{(2)} \mathbf{z}(\mathbf{x}) \right)$$

- Key step: Now we also make the earlier layer learnable!
- This is known as a **Multi-Layer Perceptron (MLP)**.

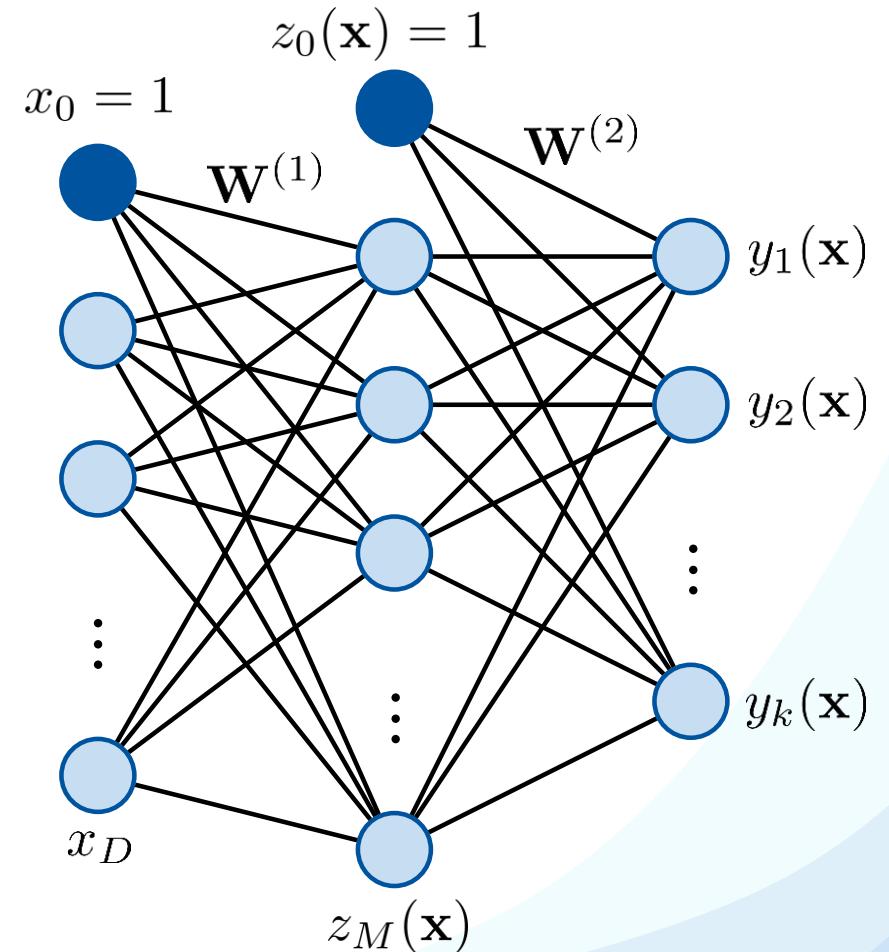


General Network Structure

- Multi-Layer Perceptron model:

$$y_k(\mathbf{x}) = g^{(2)} \left(\sum_{i=0}^M w_{ki}^{(2)} g^{(1)} \left(\sum_{j=0}^D w_{ij}^{(1)} x_j \right) \right)$$

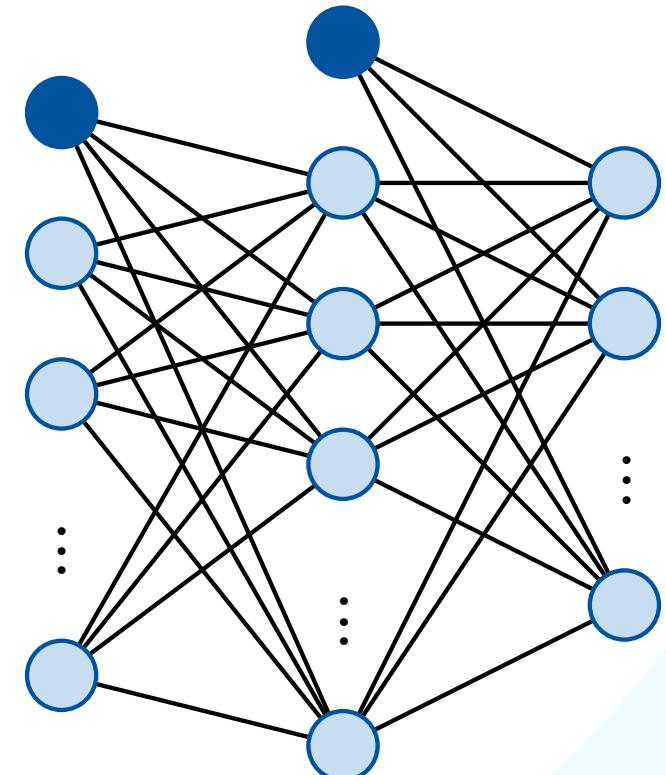
- Usually, each layer adds a bias term.
- Activation functions between layers should be non-linear.
 - For example: $g^{(2)}(a) = \sigma(a)$, $g^{(1)}(a) = \max\{a, 0\}$
 - With linear activations, successive layers would still compute a linear function.
- The hidden layer can have an arbitrary number of nodes.
- There can also be multiple hidden layers.



MLPs are Universal Approximators

$$y_k(\mathbf{x}) = g^{(2)} \left(\sum_{i=0}^M w_{ki}^{(2)} g^{(1)} \left(\sum_{j=0}^D w_{ij}^{(1)} x_j \right) \right)$$

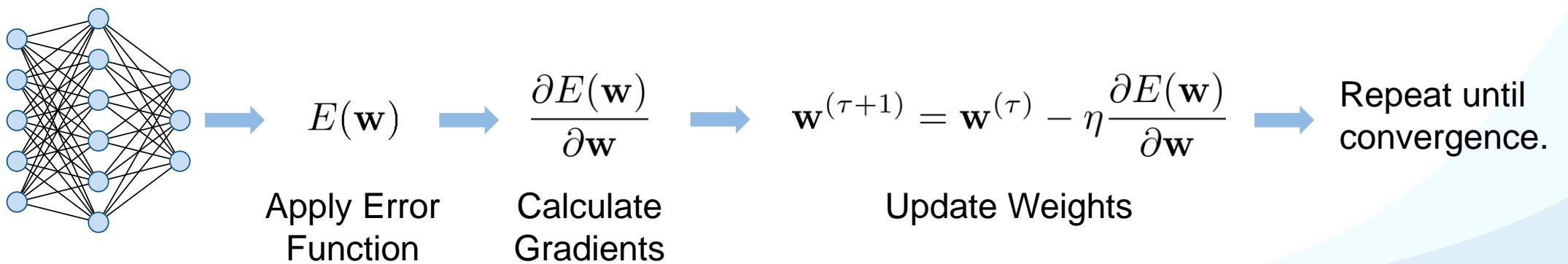
- **Universal Approximator Theorem:**
 - A network with one hidden layer can approximate any continuous function of a compact domain arbitrarily well (assuming sufficient hidden nodes).
- ⇒ *Way more powerful than linear models!*



Learning with Hidden Units

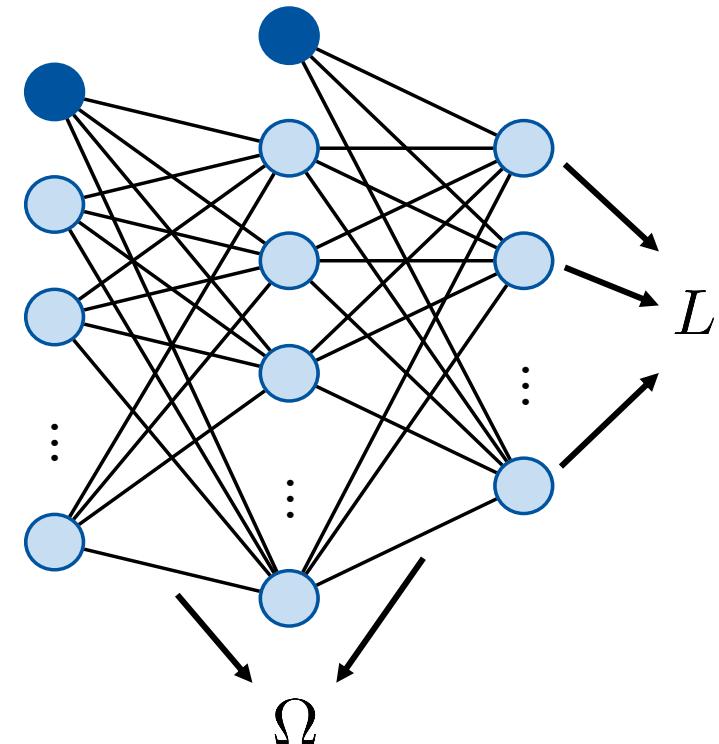
- We now have a model that contains multiple layers of adaptive non-linear hidden units.
- How can we train such models?
 - Need to train *all* weights, not just last layer.
 - Learning the weights to the hidden units = learning features.
 - We don't know what the hidden units should do.
- Basic Idea: **Gradient Descent**.

This is the main challenge of deep learning!



Neural Network Basics

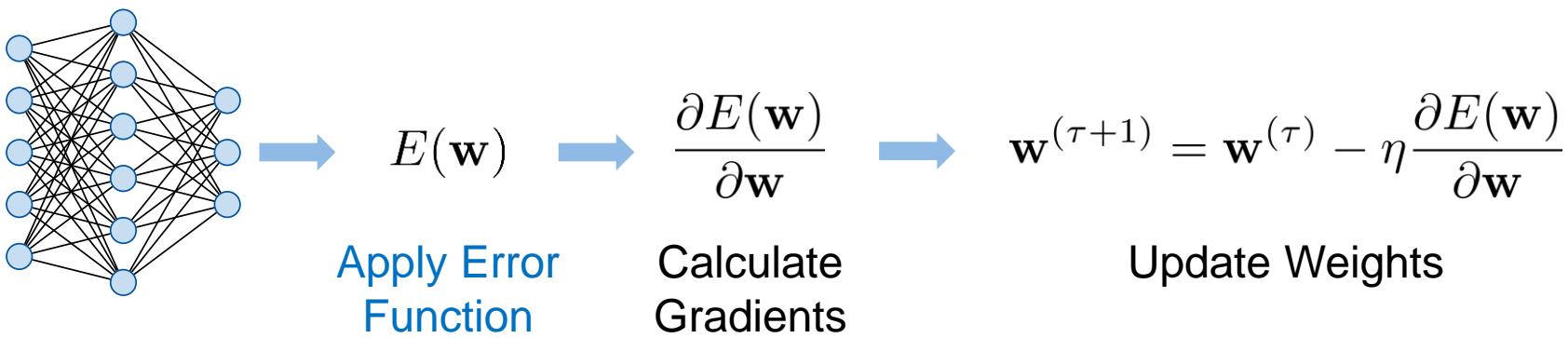
1. Perceptrons
2. Multi-Layer Perceptrons
3. **Loss Functions**
4. Backpropagation
5. Stochastic Gradient Descent



Loss Functions

- We train Neural Networks by minimizing an **error function**
 - In principle, any differentiable objective function can be used here.
 - Typically, we use a combination of a **loss function** $L(t, y(\mathbf{x}))$ and a **regularizer** $\Omega(\mathbf{w})$:

$$E(\mathbf{w}) = \sum_{n=1}^N L(t_n, y(\mathbf{x}_n; \mathbf{w})) + \lambda\Omega(\mathbf{w})$$



Examples of Loss Functions

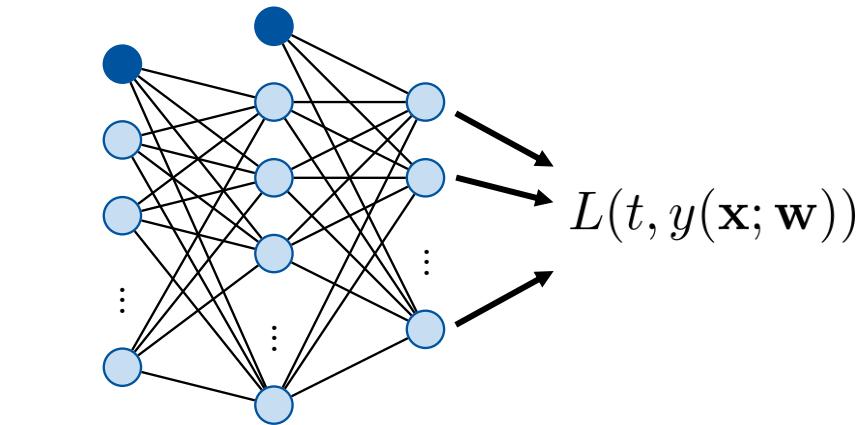
- We can use any of the loss functions we have seen so far to achieve different effects:

- L_2 loss (Squared Error)

$$L(t, y(\mathbf{x})) = \frac{1}{2}(y(\mathbf{x}) - t)^2$$

- Binary Cross-Entropy loss

$$L(t, y(\mathbf{x})) = -(t \ln \sigma(y(\mathbf{x})) + (1 - t) \ln(1 - \sigma(y(\mathbf{x}))))$$



\Rightarrow Least-squares regression / classif.

- Hinge loss

$$L(t, y(\mathbf{x})) = [1 - ty(\mathbf{x})]_+$$

\Rightarrow Logistic regression

- Multi-Class Cross-Entropy loss

$$L(t, \mathbf{y}(\mathbf{x})) = - \sum_k \left(\mathbb{I}(t = k) \ln \frac{\exp(y_k(\mathbf{x}))}{\sum_j \exp(y_j(\mathbf{x}))} \right)$$

\Rightarrow SVM classification

\Rightarrow Multi-class probabilistic classification

Examples of Regularization Terms

- Similarly, we can use any of the regularization terms we have seen so far:

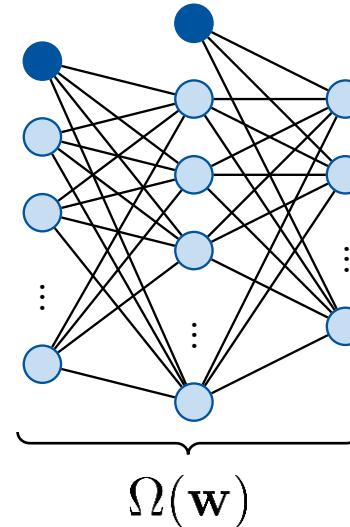
- L_2 regularizer (“Weight Decay”)

$$\Omega(\mathbf{w}) = \frac{1}{2} \|\mathbf{w}\|_2^2$$

- L_1 regularizer

$$\Omega(\mathbf{w}) = \frac{1}{2} \|\mathbf{w}\|_1$$

- Since Neural Networks have many parameters, regularization becomes an important consideration.
 - Many of the more advanced NN “training tricks” can also be understood as a form of regularization

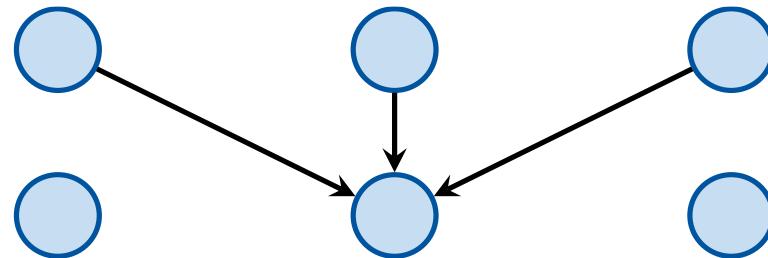


\Rightarrow Prevents overfitting

\Rightarrow Enforces sparsity (feature selection)

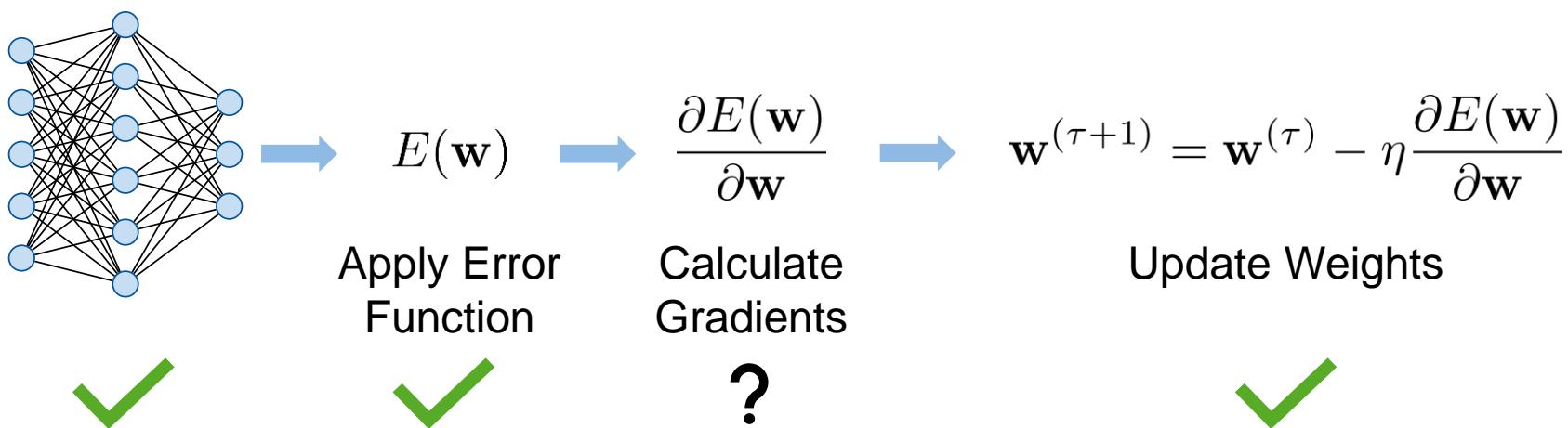
Neural Network Basics

1. Perceptrons
2. Multi-Layer Perceptrons
3. Loss Functions
4. **Backpropagation**
5. Stochastic Gradient Descent



Backpropagation

- We know a flexible model that is able to learn features.
- We also know how to compute an error estimate.
- Now we need to compute the gradients with respect to our parameters.



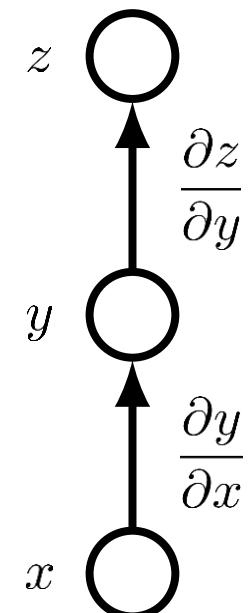
Approach 1: Naïve Analytical Differentiation

- Compute the gradients of each variable analytically.
- Scalar case is straightforward:

$$\Delta z = \frac{\partial z}{\partial y} \Delta y \quad \Delta y = \frac{\partial y}{\partial x} \Delta x$$

$$\Delta z = \frac{\partial z}{\partial y} \frac{\partial y}{\partial x} \Delta x$$

$$\frac{\partial z}{\partial x} = \frac{\partial z}{\partial y} \frac{\partial y}{\partial x}$$

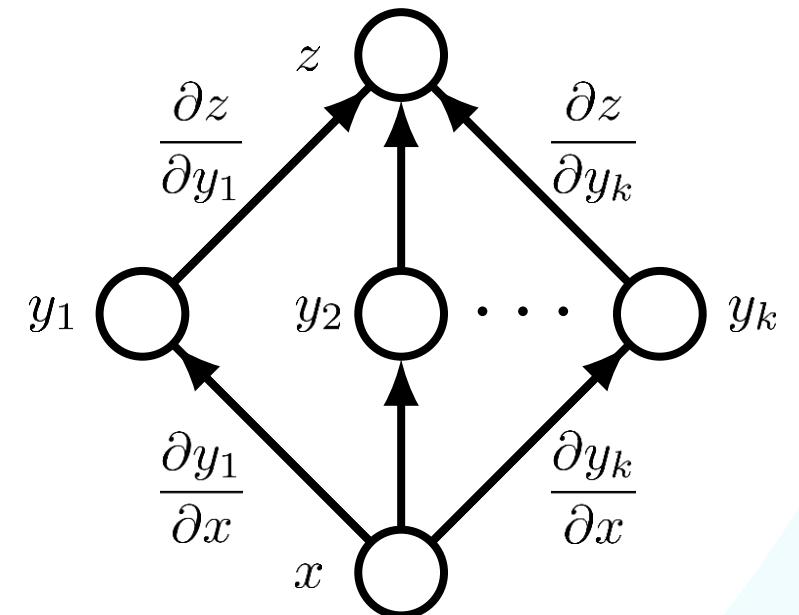


Approach 1: Naïve Analytical Differentiation

- Compute the gradients of each variable analytically.
- Scalar case is straightforward.
- Multi-dimensional case: **Total derivative**
 - Need to sum over all paths to target variable:

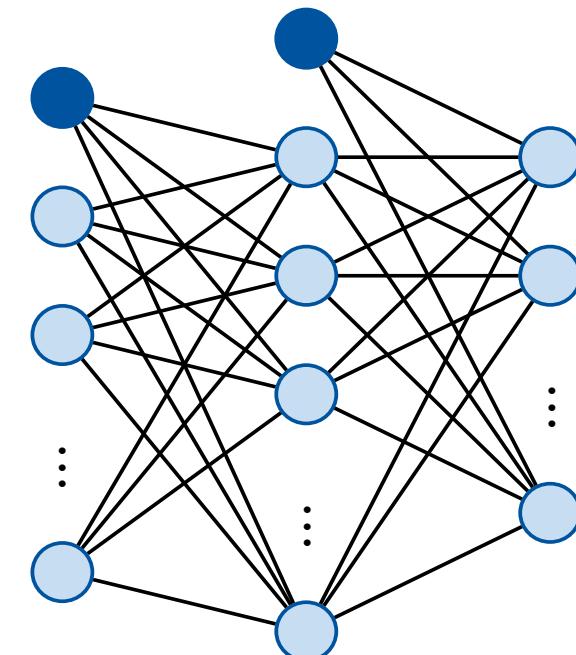
$$\frac{\partial z}{\partial x} = \frac{\partial z}{\partial y_1} \frac{\partial y_1}{\partial x} + \frac{\partial z}{\partial y_2} \frac{\partial y_2}{\partial x} + \dots = \sum_{i=1}^k \frac{\partial z}{\partial y_i} \frac{\partial y_i}{\partial x}$$

- With increasing depth, there will be exponentially many paths!



Approach 2: Numerical Differentiation

- Given the current state $\mathbf{w}^{(\tau)}$, we can evaluate $E(\mathbf{w}^{(\tau)})$.
- Idea: Make small changes to $\mathbf{w}^{(\tau)}$ and accept those that improve $E(\mathbf{w}^{(\tau)})$.
- Need several forward passes for each weight – over the whole dataset.
- This is horribly inefficient!

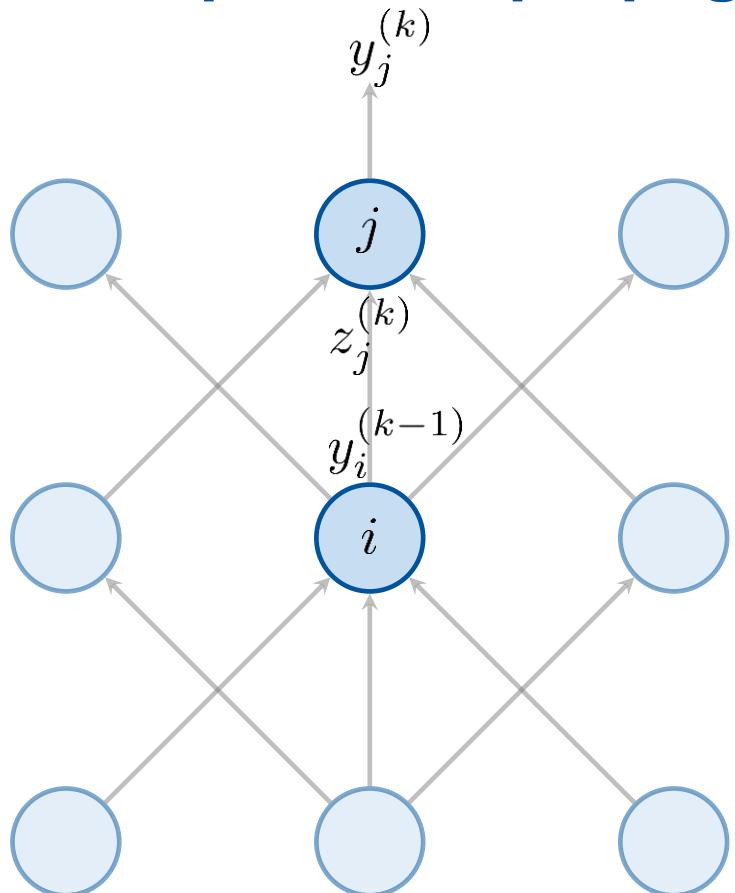


Approach 3: Incremental Analytical Differentiation

- Idea: compute the gradients layer by layer.
- Each layer below builds upon the results of the layer above.
- The gradient is propagated backwards through the layers.
- This is the [backpropagation](#) algorithm.

$$\begin{array}{c} \frac{\partial E(\mathbf{w})}{\partial y_i} \\ \downarrow \\ \frac{\partial E(\mathbf{w})}{\partial z_i} \\ \downarrow \\ \frac{\partial E(\mathbf{w})}{\partial x_i} \end{array} \quad \begin{array}{l} \searrow \\ \frac{\partial E(\mathbf{w})}{\partial w_{ij}^{(2)}} \end{array} \quad \begin{array}{l} \searrow \\ \frac{\partial E(\mathbf{w})}{\partial w_{ij}^{(1)}} \end{array}$$

Example: Backpropagation for MLPs



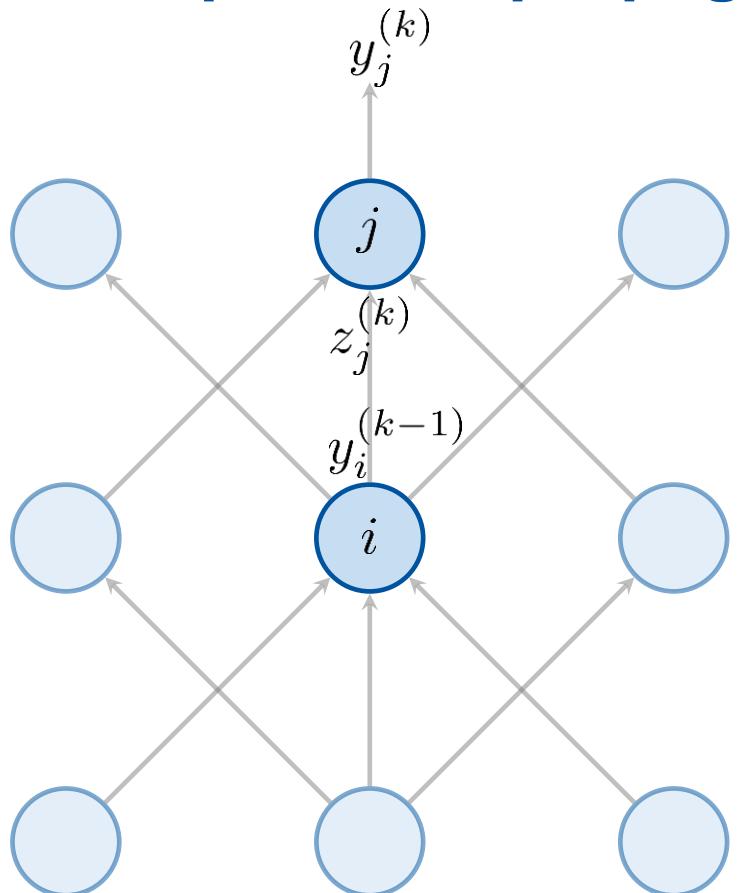
Input of layer k :
$$z_j^{(k)} = \sum_i w_{ji}^{(k-1)} y_i^{(k-1)}$$

Output of layer k :
$$y_j^{(k)} = g\left(z_j^{(k)}\right)$$

$$\frac{\partial E}{\partial z_j^{(k)}} = \frac{\partial y_j^{(k)}}{\partial z_j^{(k)}} \frac{\partial E}{\partial y_j^{(k)}} = \frac{\partial g\left(z_j^{(k)}\right)}{\partial z_j^{(k)}} \frac{\partial E}{\partial y_j^{(k)}}$$

$$\frac{\partial E}{\partial y_i^{(k-1)}} = \sum_j \frac{\partial z_j^{(k)}}{\partial y_i^{(k-1)}} \frac{\partial E}{\partial z_j^{(k)}}$$

Example: Backpropagation for MLPs



Input of layer k :
$$z_j^{(k)} = \sum_i w_{ji}^{(k-1)} y_i^{(k-1)}$$

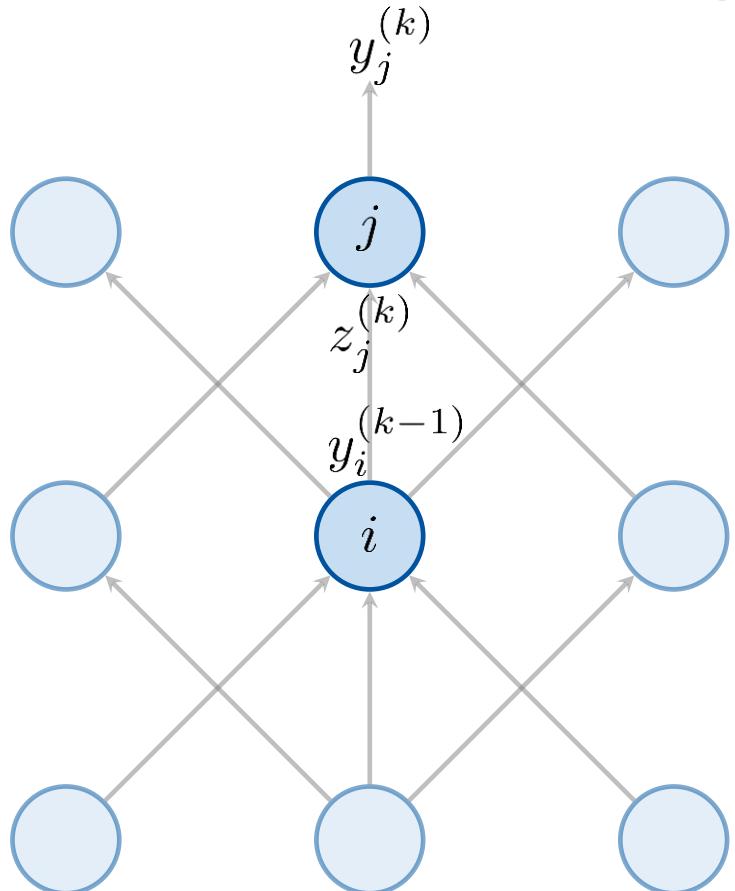
Output of layer k :
$$y_j^{(k)} = g\left(z_j^{(k)}\right)$$

$$\frac{\partial E}{\partial z_j^{(k)}} = \frac{\partial y_j^{(k)}}{\partial z_j^{(k)}} \frac{\partial E}{\partial y_j^{(k)}} = \frac{\partial g\left(z_j^{(k)}\right)}{\partial z_j^{(k)}} \frac{\partial E}{\partial y_j^{(k)}}$$

$$\frac{\partial E}{\partial y_i^{(k-1)}} = \sum_j \frac{\partial z_j^{(k)}}{\partial y_i^{(k-1)}} \frac{\partial E}{\partial z_j^{(k)}} = \sum_j w_{ji}^{(k-1)} \frac{\partial E}{\partial z_j^{(k)}}$$

$$\frac{\partial z_j^{(k)}}{\partial y_i^{(k-1)}} = w_{ji}^{(k-1)}$$

Example: Backpropagation for MLPs



Input of layer k :
$$z_j^{(k)} = \sum_i w_{ji}^{(k-1)} y_i^{(k-1)}$$

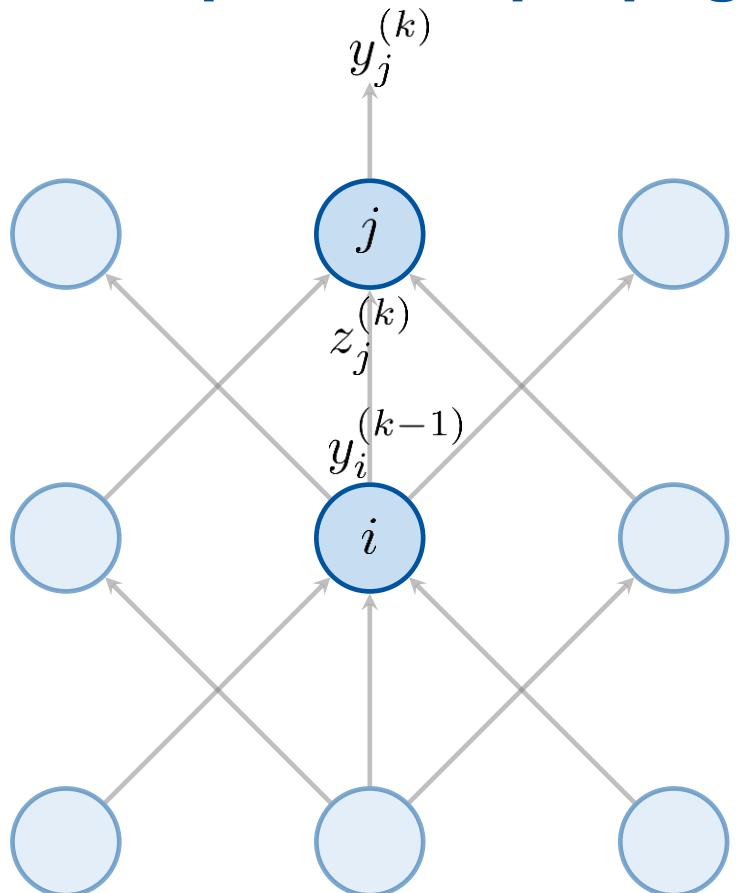
Output of layer k :
$$y_j^{(k)} = g\left(z_j^{(k)}\right)$$

$$\frac{\partial E}{\partial z_j^{(k)}} = \frac{\partial y_j^{(k)}}{\partial z_j^{(k)}} \frac{\partial E}{\partial y_j^{(k)}} = \frac{\partial g\left(z_j^{(k)}\right)}{\partial z_j^{(k)}} \frac{\partial E}{\partial y_j^{(k)}}$$

$$\frac{\partial E}{\partial y_i^{(k-1)}} = \sum_j \frac{\partial z_j^{(k)}}{\partial y_i^{(k-1)}} \frac{\partial E}{\partial z_j^{(k)}} = \sum_j w_{ji}^{(k-1)} \frac{\partial E}{\partial z_j^{(k)}}$$

$$\frac{\partial E}{\partial w_{ji}^{(k-1)}} = \frac{\partial z_j^{(k)}}{\partial w_{ji}^{(k-1)}} \frac{\partial E}{\partial z_j^{(k)}}$$

Example: Backpropagation for MLPs



Input of layer k :
$$z_j^{(k)} = \sum_i w_{ji}^{(k-1)} y_i^{(k-1)}$$

Output of layer k :
$$y_j^{(k)} = g\left(z_j^{(k)}\right)$$

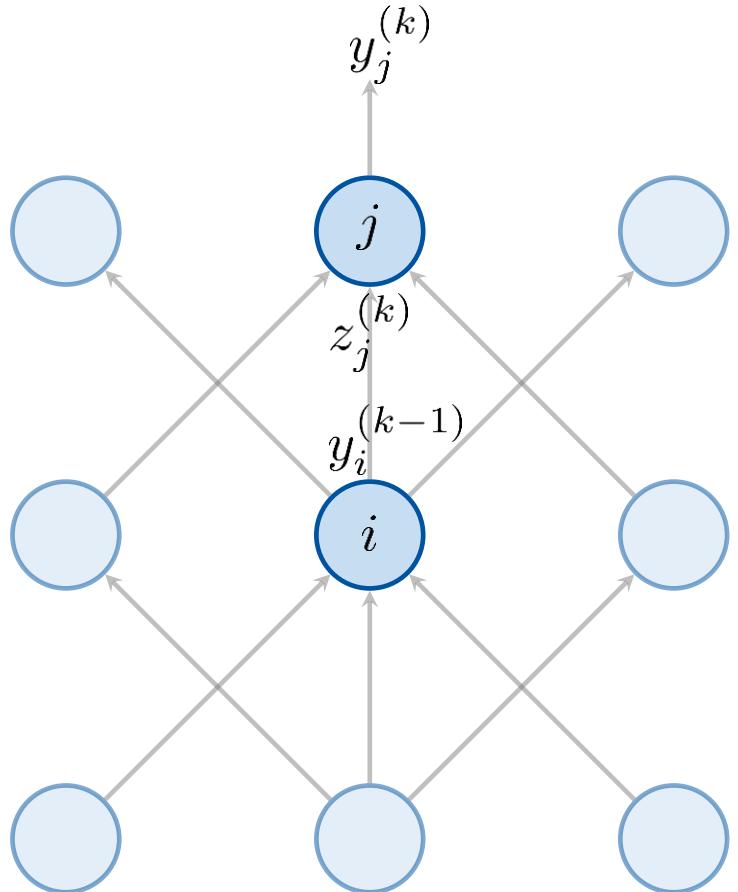
$$\frac{\partial E}{\partial z_j^{(k)}} = \frac{\partial y_j^{(k)}}{\partial z_j^{(k)}} \frac{\partial E}{\partial y_j^{(k)}} = \frac{\partial g\left(z_j^{(k)}\right)}{\partial z_j^{(k)}} \frac{\partial E}{\partial y_j^{(k)}}$$

$$\frac{\partial E}{\partial y_i^{(k-1)}} = \sum_j \frac{\partial z_j^{(k)}}{\partial y_i^{(k-1)}} \frac{\partial E}{\partial z_j^{(k)}} = \sum_j w_{ji}^{(k-1)} \frac{\partial E}{\partial z_j^{(k)}}$$

$$\frac{\partial E}{\partial w_{ji}^{(k-1)}} = \frac{\partial z_j^{(k)}}{\partial w_{ji}^{(k-1)}} \frac{\partial E}{\partial z_j^{(k)}} = y_i^{(k-1)} \frac{\partial E}{\partial z_j^{(k)}}$$

$$\frac{\partial z_j^{(k)}}{\partial w_{ji}^{(k-1)}} = y_i^{(k-1)}$$

Example: Backpropagation for MLPs



Input of layer k :
$$z_j^{(k)} = \sum_i w_{ji}^{(k-1)} y_i^{(k-1)}$$

Output of layer k :
$$y_j^{(k)} = g\left(z_j^{(k)}\right)$$

$$\frac{\partial E}{\partial z_j^{(k)}} = \frac{\partial y_j^{(k)}}{\partial z_j^{(k)}} \frac{\partial E}{\partial y_j^{(k)}} = \frac{\partial g\left(z_j^{(k)}\right)}{\partial z_j^{(k)}} \frac{\partial E}{\partial y_j^{(k)}}$$

$$\frac{\partial E}{\partial y_i^{(k-1)}} = \sum_j \frac{\partial z_j^{(k)}}{\partial y_i^{(k-1)}} \frac{\partial E}{\partial z_j^{(k)}} = \sum_j w_{ji}^{(k-1)} \frac{\partial E}{\partial z_j^{(k)}}$$

$$\frac{\partial E}{\partial w_{ji}^{(k-1)}} = \frac{\partial z_j^{(k)}}{\partial w_{ji}^{(k-1)}} \frac{\partial E}{\partial z_j^{(k)}} = y_i^{(k-1)} \frac{\partial E}{\partial z_j^{(k)}}$$

Discussion Backpropagation

Advantages

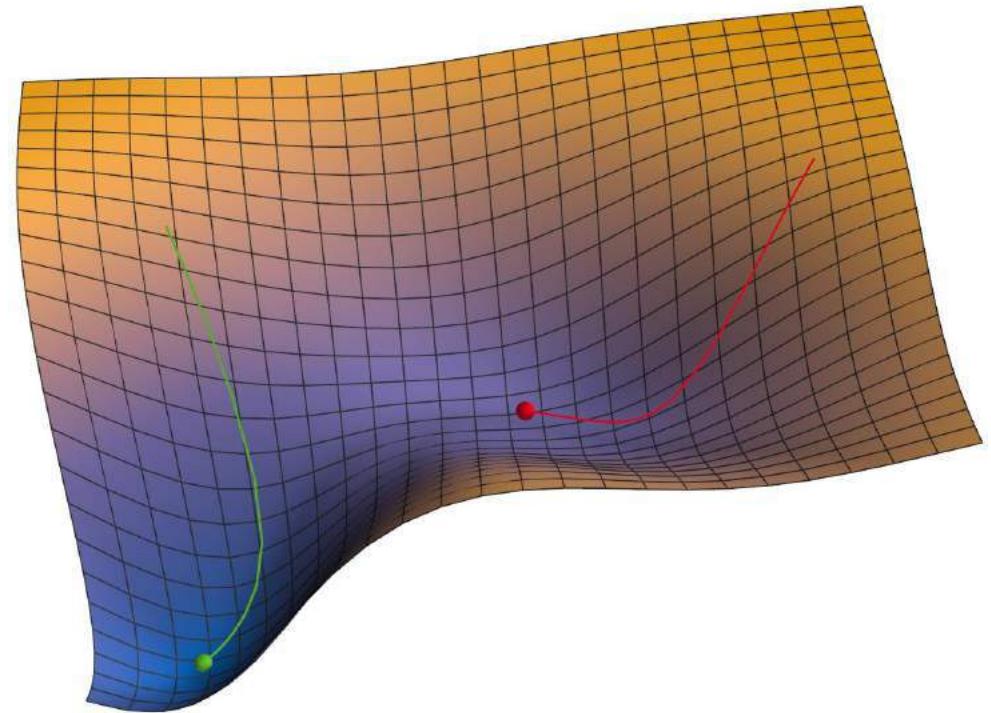
- Very general algorithm, widely used.
- Efficiently computes all gradients in the network using dynamic programming.
- The same concept can be applied to any differentiable function.
 - This makes it possible to define other types of layers.

Limitations

- Efficient evaluation of backpropagation requires storing all unit activations from forward pass.
 - The amount of memory necessary for this imposes a practical limit on the size of the network.
- Successful learning relies on the gradients to be propagated to the early network layers.
 - Numerical challenges may arise here.

Neural Network Basics

1. Perceptrons
2. Multi-Layer Perceptrons
3. Loss Functions
4. Backpropagation
5. **Stochastic Gradient Descent**

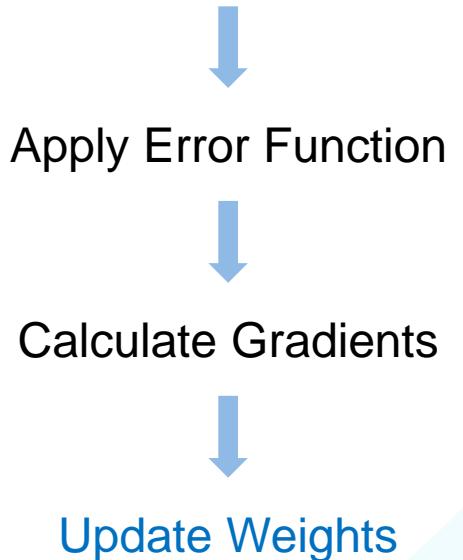
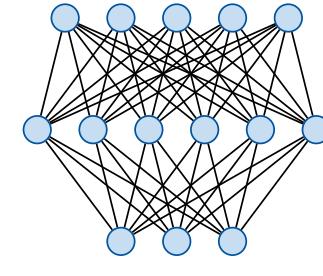


Stochastic Gradient Descent

- Now that we have the gradients, we need to update the weights.
- We already know the basic equation for this
 - (1st-order) Gradient Descent

$$w_{kj}^{(\tau+1)} = w_{kj}^{(\tau)} - \eta \frac{\partial E(\mathbf{w})}{\partial w_{kj}} \Big|_{\mathbf{w}^{(\tau)}}$$

- Remaining Questions:
 - On what data do we want to apply this?
 - How should we choose the learning rate η ?



Stochastic vs. Batch Learning

- **Batch Learning**
 - Process the full dataset in one batch.
 - Compute the gradient based on all training examples.
- **Stochastic Learning**
 - Choose a single example from the training set.
 - Compute the gradient only based on this example.
 - This estimate will generally be noisy, which has some advantages.

$$w_{kj}^{(\tau+1)} = w_{kj}^{(\tau)} - \eta \frac{\partial E(\mathbf{w})}{\partial w_{kj}} \Big|_{\mathbf{w}^{(\tau)}}$$

$$E(\mathbf{w}) = \sum_{n=1}^N E_n(\mathbf{w})$$

$$w_{kj}^{(\tau+1)} = w_{kj}^{(\tau)} - \eta \frac{\partial E_n(\mathbf{w})}{\partial w_{kj}} \Big|_{\mathbf{w}^{(\tau)}}$$

Batch Learning

- Conditions of convergence are well understood.
- Many acceleration techniques only work in batch learning.
- Theoretical analysis of the weight dynamics and convergence rates are simpler.

Stochastic Learning

- Usually much faster than batch learning.
- Often results in better solutions.
- Can be used for tracking changes when the target distribution shifts.

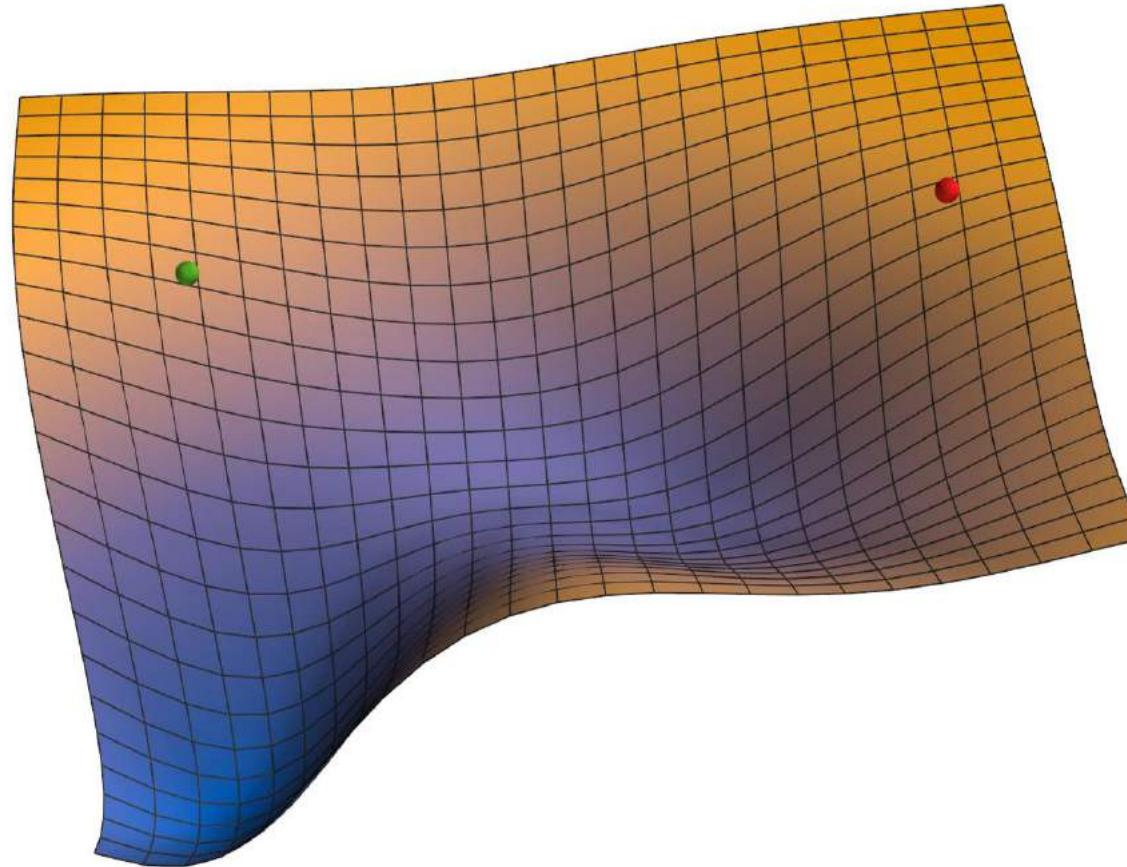
Middle ground: Minibatches

Minibatches

- Idea
 - Process only a small batch of training examples together. $\mathcal{B} = \{(\mathbf{x}_1, t_1), \dots, (\mathbf{x}_B, t_B)\} \subset \mathcal{D}$
 - Start with a small batch size & increase it as training proceeds.
- Advantages
 - Gradients will be more stable than for stochastic gradient descent, but still faster to compute than with batch learning.
 - Take advantage of redundancies in the training set.
 - Matrix operations are more efficient than vector operations.
- Caveat
 - Need to normalize error function by the minibatch size to use the same learning rate between minibatches

$$E(\mathbf{w}) = \frac{1}{N} \sum_{n=1}^N L(t_n, y(\mathbf{x}_n; \mathbf{w})) + \frac{\lambda}{N} \Omega(\mathbf{w})$$

Example



Choosing the Right Learning Rate

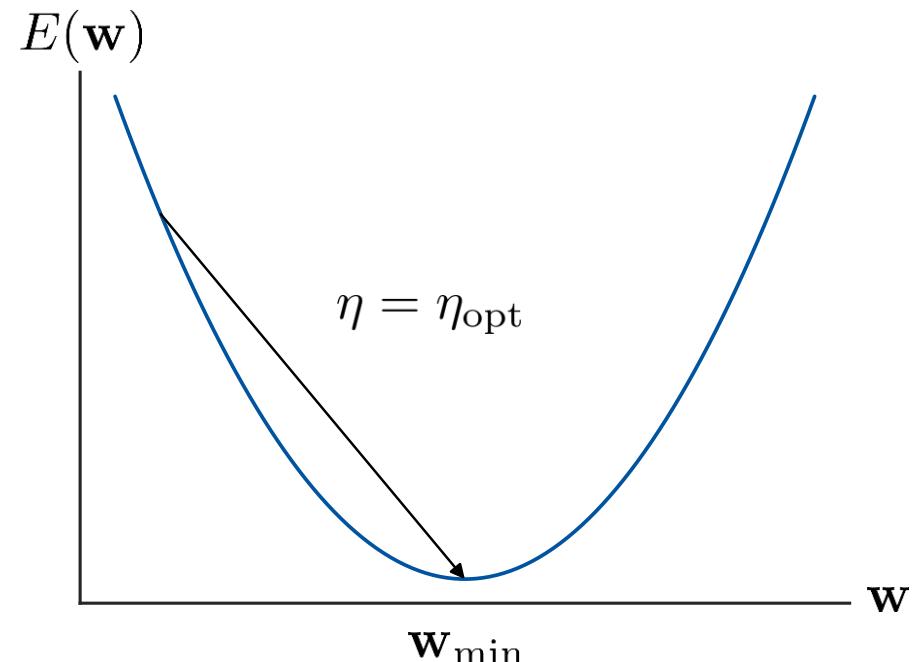
- Consider a simple 1D example:

$$w^{(\tau+1)} = w^{(\tau)} - \eta \frac{\partial E(w)}{\partial w}$$

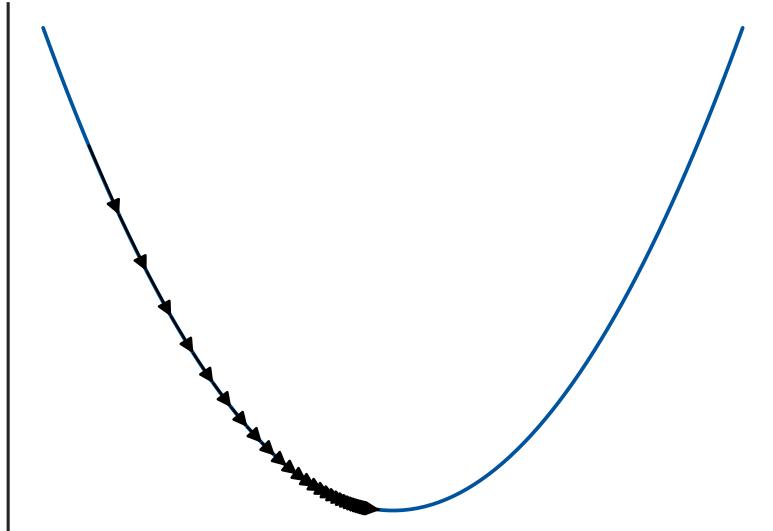
- What is the optimal learning rate η_{opt} ?
- If E is quadratic, the optimal learning rate is given by the inverse of the Hessian:

$$\eta_{\text{opt}} = \left(\frac{\partial^2 E(w^{(\tau)})}{\partial w^2} \right)^{-1}$$

- For neural networks, the Hessian is usually infeasible to compute.

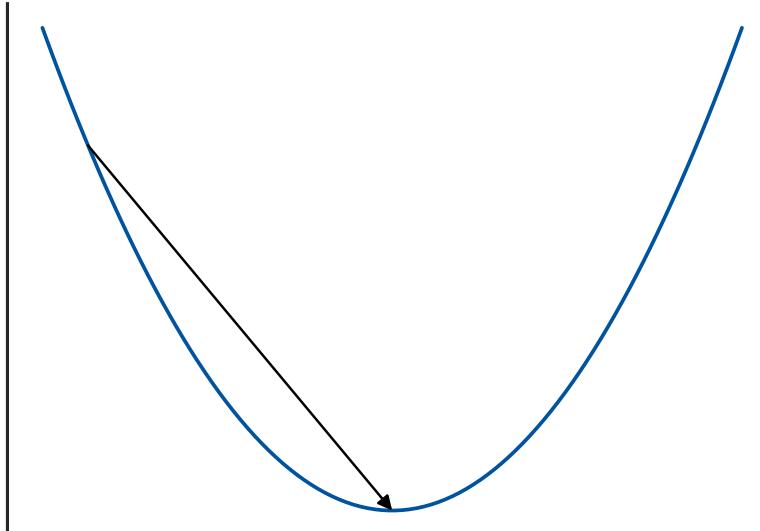


η too small



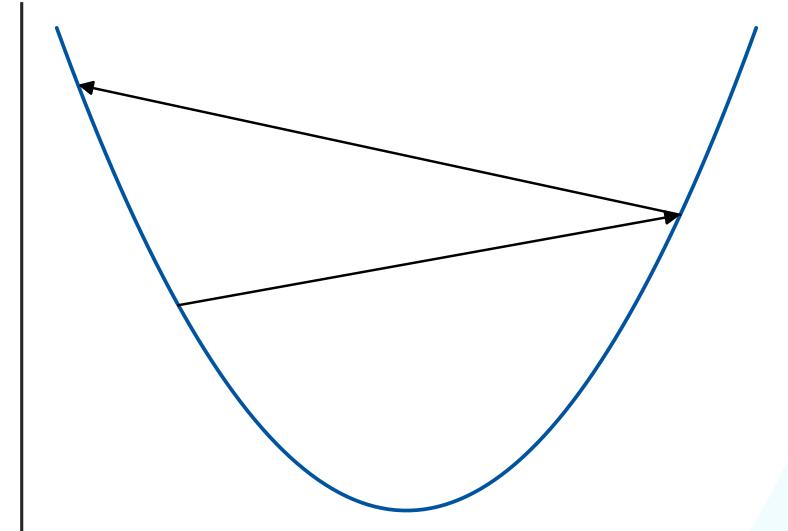
Convergence is slow

η_{opt}



Converges ideally in
a single step

η too large



Might not converge

Discussion: Stochastic Gradient Descent

Advantages

- Very simple, but still quite robust method.
- Minibatches offer a compromise between stability and faster computation.
- Stochasticity in minibatches is often beneficial for learning

Limitations

- Finding a good setting for the learning rate is very important for fast convergence.
 - Choosing the right learning rate is a challenge and requires experience.
 - A different learning rate may be optimal for different parts of the network
- Following the direction of steepest descent is not always the fastest way to the optimum
 - E.g., in highly correlated data

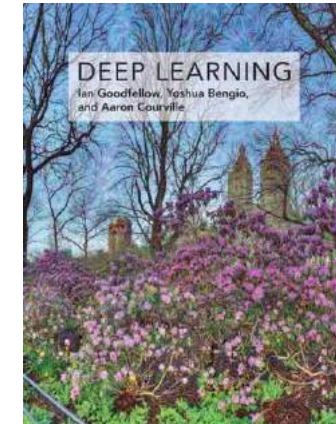
$$\frac{\partial E(\mathbf{w})}{\partial \mathbf{w}}$$


References and Further Reading

- More information about [Neural Networks and Deep Learning](#) is available in the following book.

I. Goodfellow, Y. Bengio, A. Courville
Deep Learning
MIT Press, 2016

<https://www.deeplearningbook.org/>



Elements of Machine Learning & Data Science

Process Mining

Lecture 20

Prof. Wil van der Aalst

Marco Pegoraro, M.Sc.

Tsunghao Huang, M.Sc.

Nina Graves, M.Sc.

Part I: Introduction to Process Mining

Event data, process models, software, applications

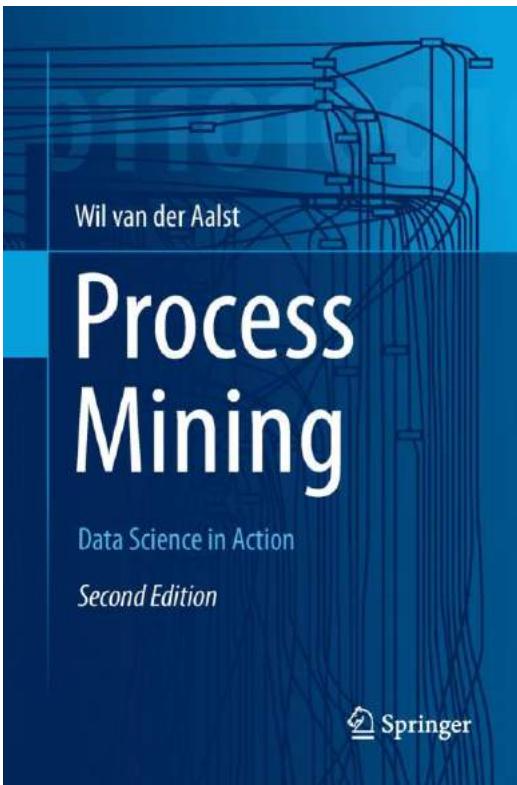
Part II: Unsupervised Process Mining

Process discovery (including Inductive Mining)

Part III: Supervised Process Mining

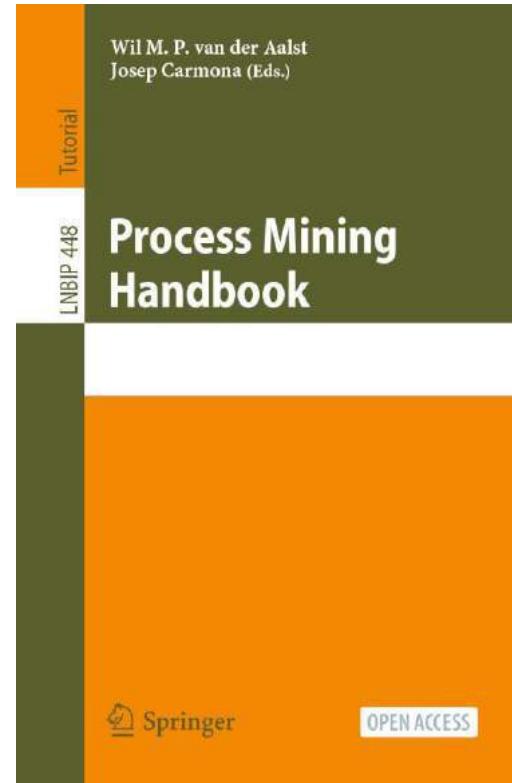
Conformance checking and link to ML (including token-based replay)

Sources for this lecture



W. van der Aalst.
Process Mining: Data Science in Action
2016, Springer
<https://link.springer.com/book/10.1007/978-3-662-49851-4>

Open Access Online!



W. van der Aalst, Josep Carmona
Process Mining Handbook
2022, Springer
<https://link.springer.com/book/10.1007/978-3-031-08848-3>

Part I: Introduction to Process Mining

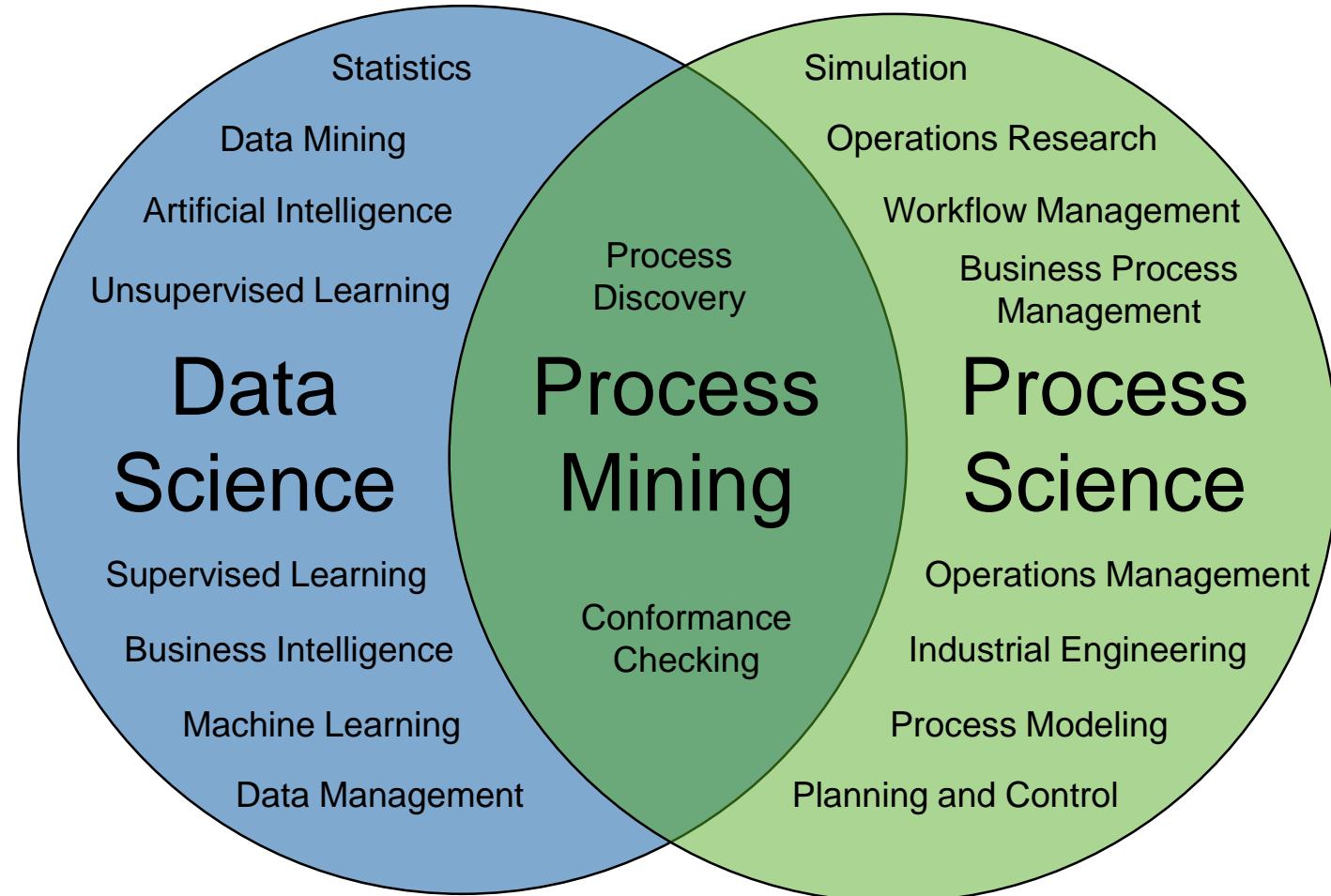
Introduction to Process Mining

- 1. Process Mining and Event Data**
2. Process Models
3. Software Tools
4. Applications

Link Between Data and Process Science

Traditionally:

- not process-centric
- focus on specific tasks or decisions



Traditionally:

- not data-driven
- focus on modeling (languages) and automation

Event Data in Process Mining

ID	Activity	Time
11152	Register Order	15.12.22 12:25
11152	Send Invoice	15.12.22 12:45
11152	Pay	15.12.22 13:01
11153	Register Order	15.12.22 13:05
11153	Send Invoice	15.12.22 13:08
11152	Confirm Payment	15.12.22 13:11
11153	Pay	16.12.22 15:03
11152	Make Delivery	17.12.22 8:10

Normal Event Log

We considered timestamped data before. For example:

- **Time series:** numerical features with equidistant timestamps (determined by the sampling rate).
- **Sequence mining:** a very specific setting where we focus on sequences of itemsets.

Event data:

- The occurrence of an event has a meaning, i.e., timestamps are not equidistant.
- Event refers to (at least) a case identifier, activity name, and timestamp.
- Very general!

Event Data in Process Mining

ID	Activity	Time
11152	Register Order	15.12.22 12:25
11152	Send Invoice	15.12.22 12:45
11152	Pay	15.12.22 13:01
11153	Register Order	15.12.22 13:05
11153	Send Invoice	15.12.22 13:08
11152	Confirm Payment	15.12.22 13:11
11153	Pay	16.12.22 15:03
11152	Make Delivery	17.12.22 8:10

Normal Event Log

Each row refers to an event and per event, there are three **mandatory** attributes:

- Case identifier
- Activity name
- Timestamp

But there can be any number of **additional** attributes, such as:

- Costs
- Duration
- Location
- Resource
- Etc.

Event Data in Process Mining

ID	Activity	Time
11152	Register Order	15.12.22 12:25
11152	Send Invoice	15.12.22 12:45
11152	Pay	15.12.22 13:01
11153	Register Order	15.12.22 13:05
11153	Send Invoice	15.12.22 13:08
11152	Confirm Payment	15.12.22 13:11
11153	Pay	16.12.22 15:03
11152	Make Delivery	17.12.22 8:10

Normal Event Log

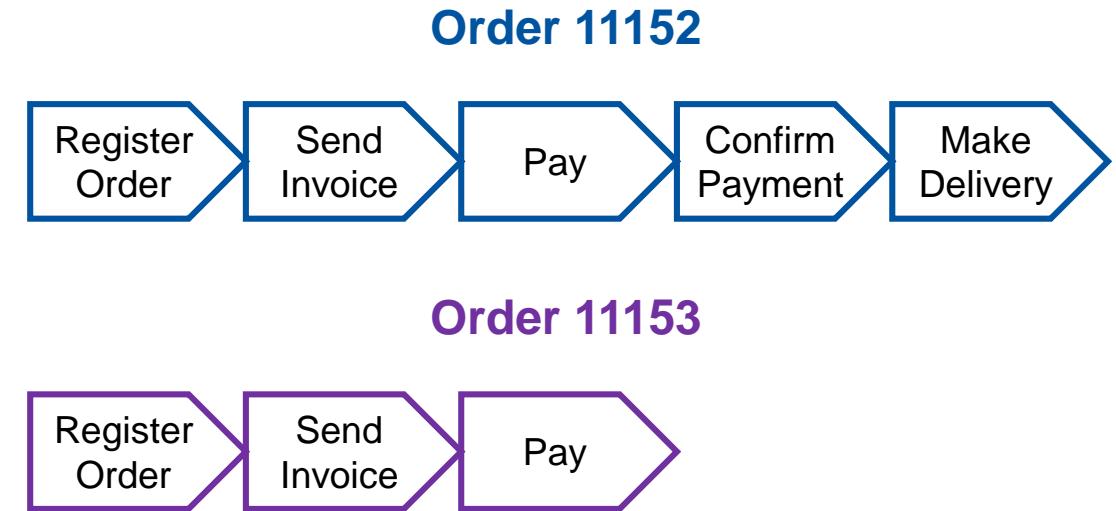


Simplified Event Log

Event Data in Process Mining

ID	Activity	Time
11152	Register Order	15.12.22 12:25
11152	Send Invoice	15.12.22 12:45
11152	Pay	15.12.22 13:01
11153	Register Order	15.12.22 13:05
11153	Send Invoice	15.12.22 13:08
11152	Confirm Payment	15.12.22 13:11
11153	Pay	16.12.22 15:03
11152	Make Delivery	17.12.22 8:10

Normal Event Log

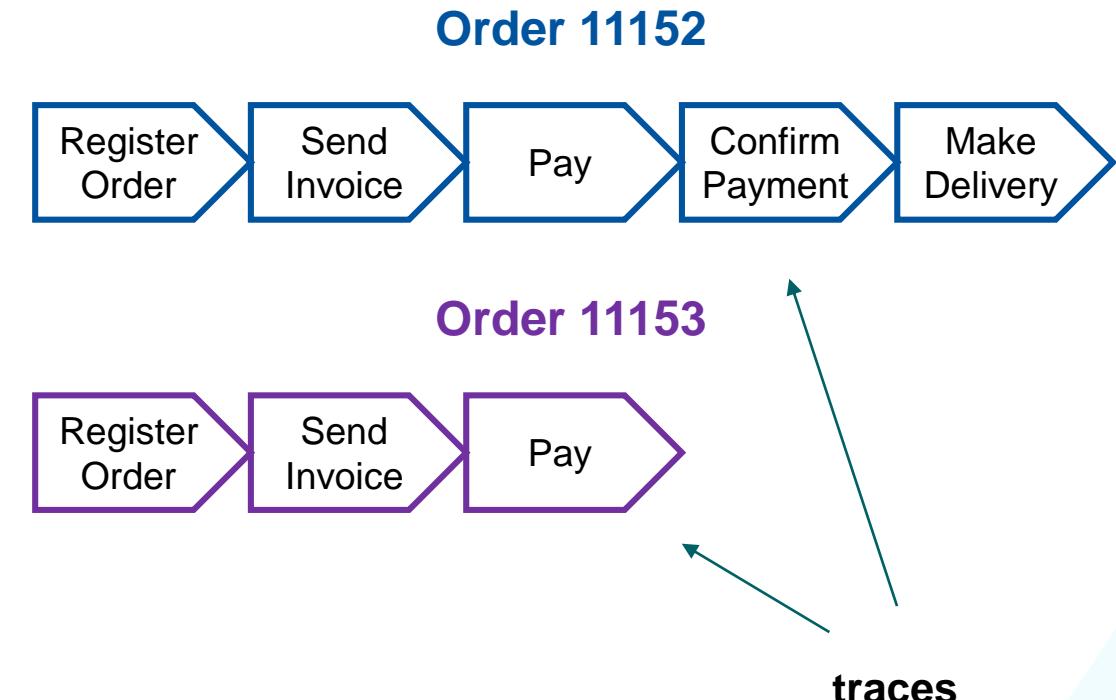


Simplified Event Log

Event Data in Process Mining

ID	Activity	Time
11152	Register Order	15.12.22 12:25
11152	Send Invoice	15.12.22 12:45
11152	Pay	15.12.22 13:01
11153	Register Order	15.12.22 13:05
11153	Send Invoice	15.12.22 13:08
11152	Confirm Payment	15.12.22 13:11
11153	Pay	16.12.22 15:03
11152	Make Delivery	17.12.22 8:10

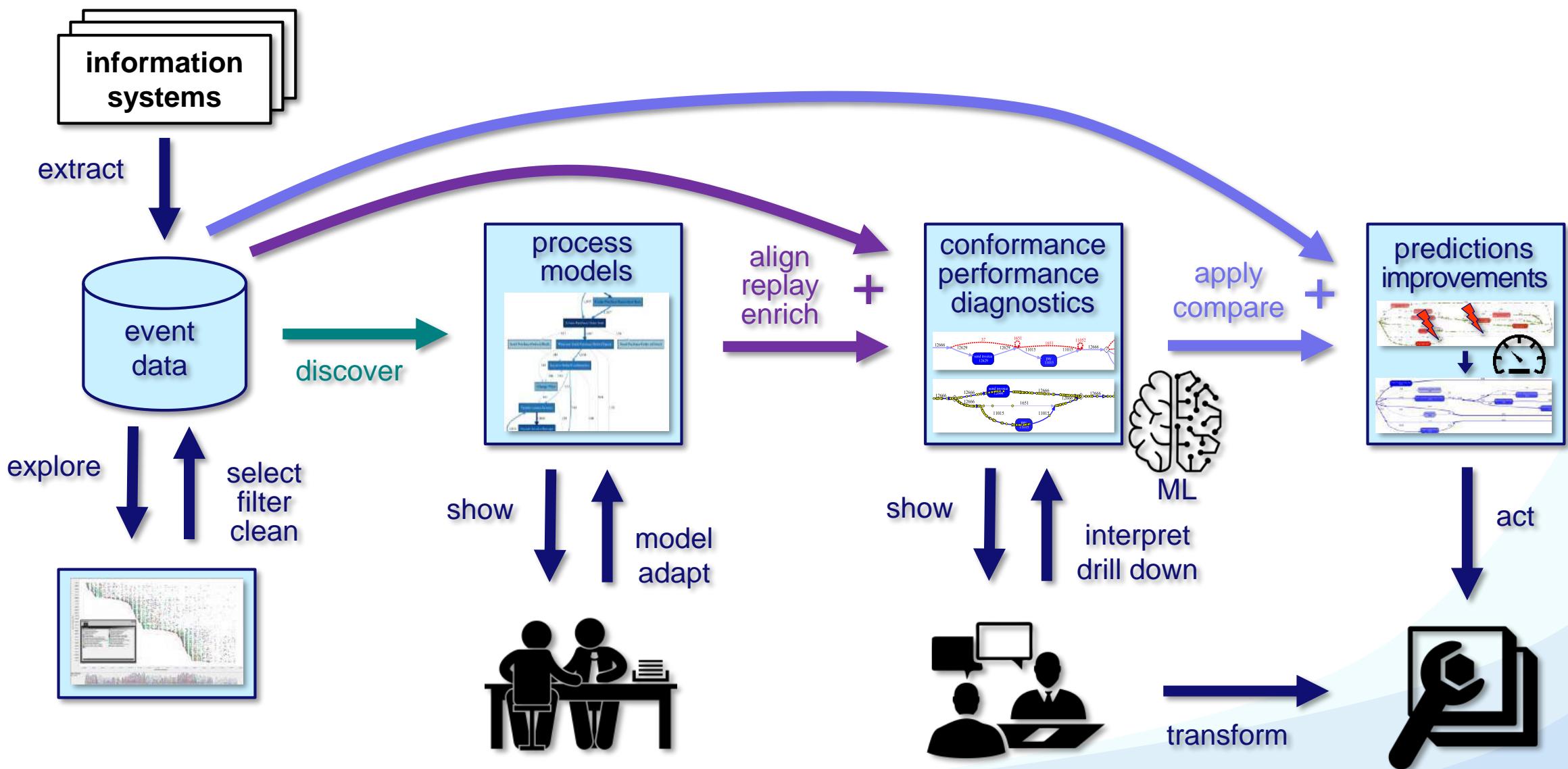
Normal Event Log



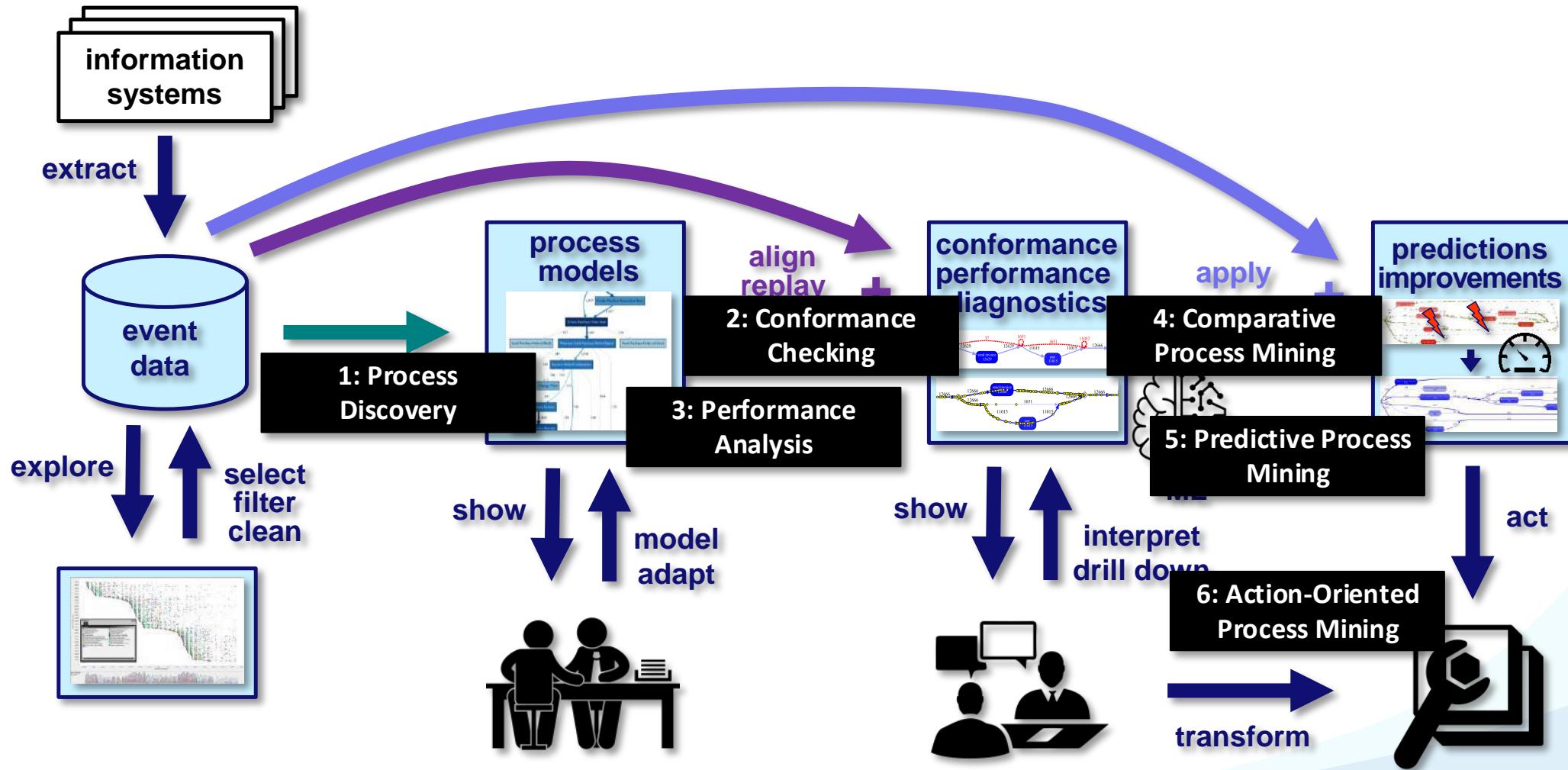
Simplified Event Log

Simplified event log is a multiset of traces

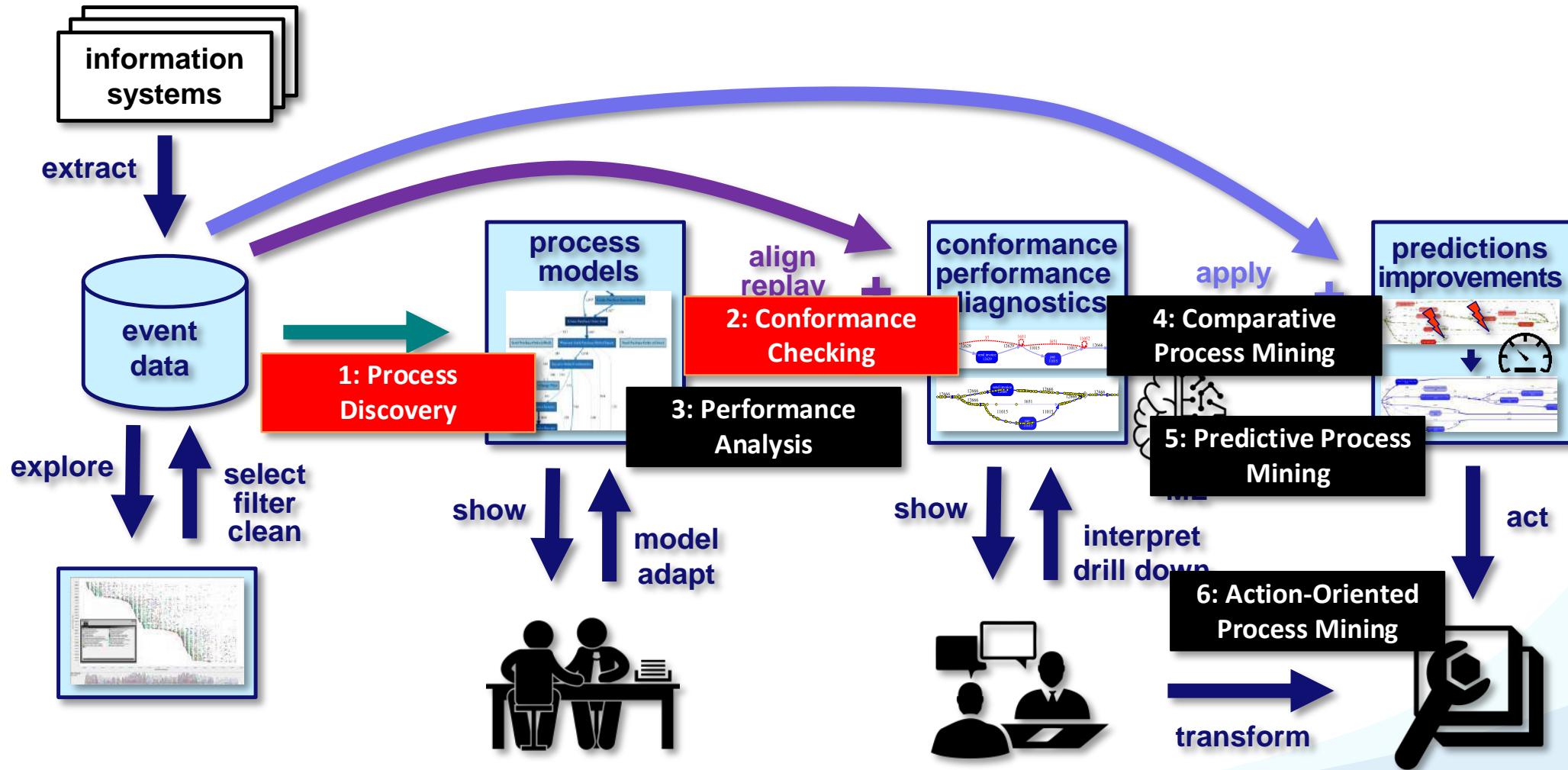
Process Mining Overview



Six Tasks in Process Mining



Six Tasks in Process Mining



Introduction to Process Mining

1. Process Mining and Event Data
2. **Process Models**
3. Software Tools
4. Applications

Four Common Types of Process Models

The same process can be visualized in many ways:

- DFGs (Directly-Follows Graphs)

Supported by all process mining tools (simple, but no concurrency)

- Petri nets

The oldest model for concurrent processes and the de facto standard in process mining research

- Process trees

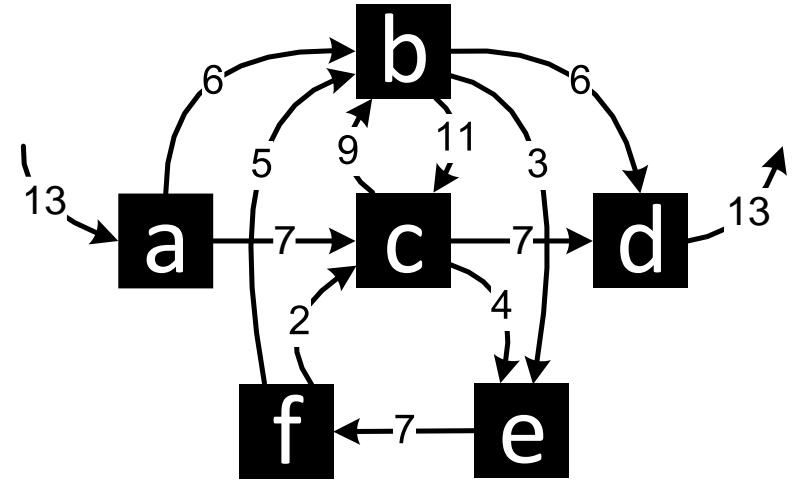
Frequently used in process mining because it is block structured and sound by construction

- BPMN (Business Process Model and Notation)

The industry standard (we use a small subset) related to UML Activity Diagrams (not explained in detail)

Directly-Follows Graphs

- Simplest notation
- Marks all edges between activities that occurred
- Helps to get first impression about the data
- Used by process discovery algorithms (e.g., Inductive Miner)



Directly-Follows Graphs – Example

Assume we have a simplified log:

$$L = [\langle a, b, d, e, h \rangle, \langle a, d, c, e, g \rangle, \langle a, c, d, e, f, b, d, e, g \rangle, \langle a, d, b, e, h \rangle, \langle a, c, d, e, f, d, c, e, f, c, d, e, h \rangle, \langle a, c, d, e, g \rangle]$$

a = register request

b = examine thoroughly

c = examine casually

d = check ticket

e = decide

f = reinitiate request

g = pay compensation

h = reject request

Directly-Follows Graphs – Example

Assume we have a simplified log:

$$L = [\langle a, b, d, e, h \rangle, \langle a, d, c, e, g \rangle, \langle a, c, d, e, f, b, d, e, g \rangle, \langle a, d, b, e, h \rangle, \langle a, c, d, e, f, d, c, e, f, c, d, e, h \rangle, \langle a, c, d, e, g \rangle]$$

$a =$ register request

$b =$ examine thoroughly

$c =$ examine casually

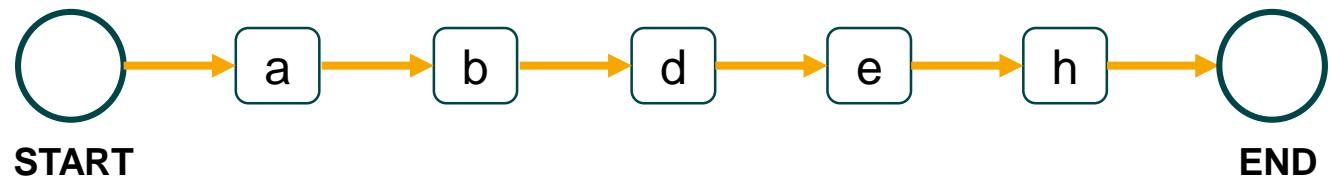
$d =$ check ticket

$e =$ decide

$f =$ reinitiate request

$g =$ pay compensation

$h =$ reject request



Directly-Follows Graphs – Example

Assume we have a simplified log:

$$L = [\langle a, b, d, e, h \rangle, \langle a, d, c, e, g \rangle, \langle a, c, d, e, f, b, d, e, g \rangle, \langle a, d, b, e, h \rangle, \langle a, c, d, e, f, d, c, e, f, c, d, e, h \rangle, \langle a, c, d, e, g \rangle]$$

a = register request

b = examine thoroughly

c = examine casually

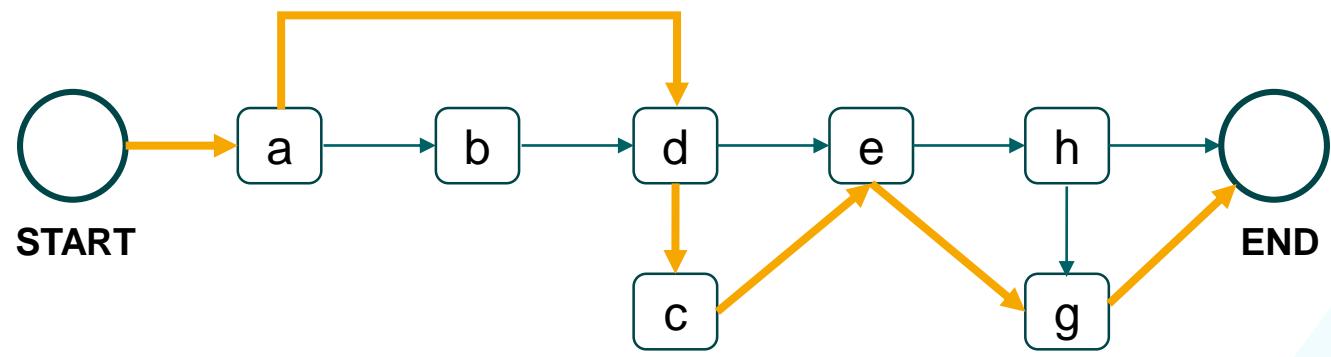
d = check ticket

e = decide

f = reinitiate request

g = pay compensation

h = reject request



Directly-Follows Graphs – Example

Assume we have a simplified log:

$$L = [\langle a, b, d, e, h \rangle, \langle a, d, c, e, g \rangle, \langle a, c, d, e, f, b, d, e, g \rangle, \langle a, d, b, e, h \rangle, \langle a, c, d, e, f, d, c, e, f, c, d, e, h \rangle, \langle a, c, d, e, g \rangle]$$

a = register request

b = examine thoroughly

c = examine casually

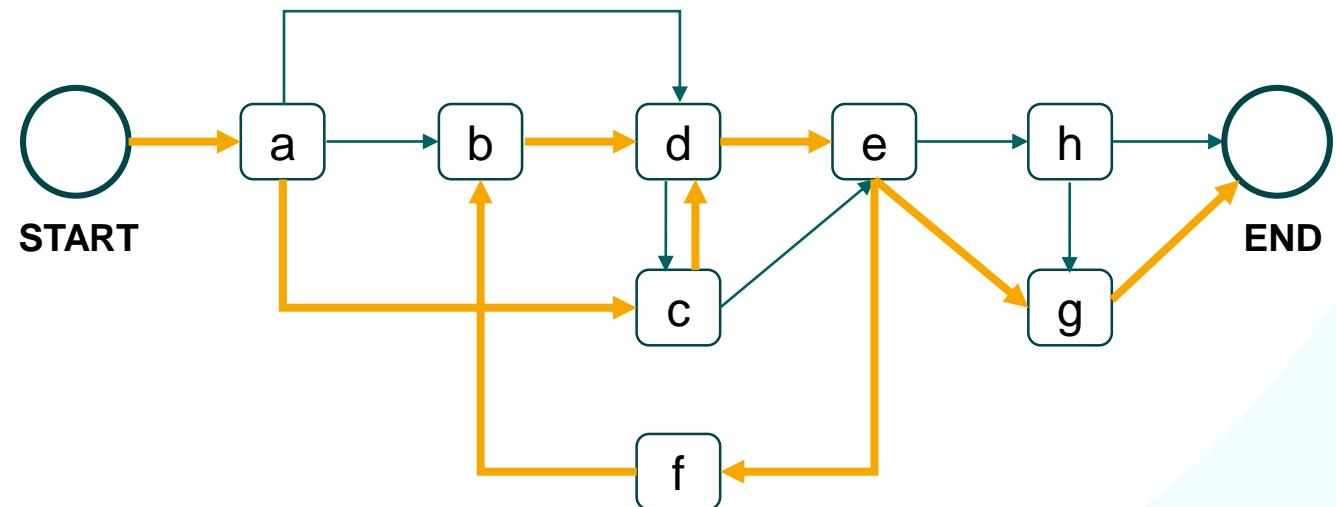
d = check ticket

e = decide

f = reinitiate request

g = pay compensation

h = reject request



Directly-Follows Graphs – Example

Assume we have a simplified log:

$$L = [\langle a, b, d, e, h \rangle, \langle a, d, c, e, g \rangle, \langle a, c, d, e, f, b, d, e, g \rangle, \langle a, d, b, e, h \rangle, \langle a, c, d, e, f, d, c, e, f, c, d, e, h \rangle, \langle a, c, d, e, g \rangle]$$

a = register request

b = examine thoroughly

c = examine casually

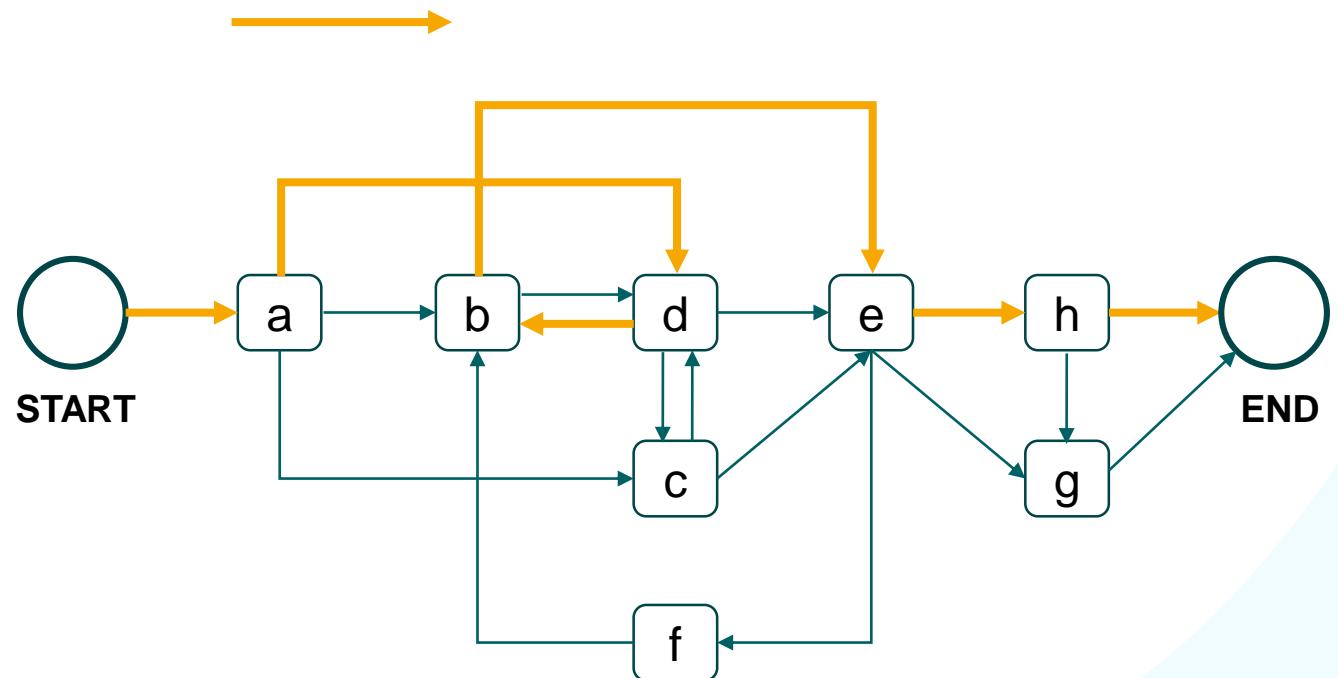
d = check ticket

e = decide

f = reinitiate request

g = pay compensation

h = reject request



Directly-Follows Graphs – Example

Assume we have a simplified log:

$$L = [\langle a, b, d, e, h \rangle, \langle a, d, c, e, g \rangle, \langle a, c, d, e, f, b, d, e, g \rangle, \langle a, d, b, e, h \rangle, \langle a, c, d, e, f, d, c, e, f, c, d, e, h \rangle, \langle a, c, d, e, g \rangle]$$

a = register request

b = examine thoroughly

c = examine casually

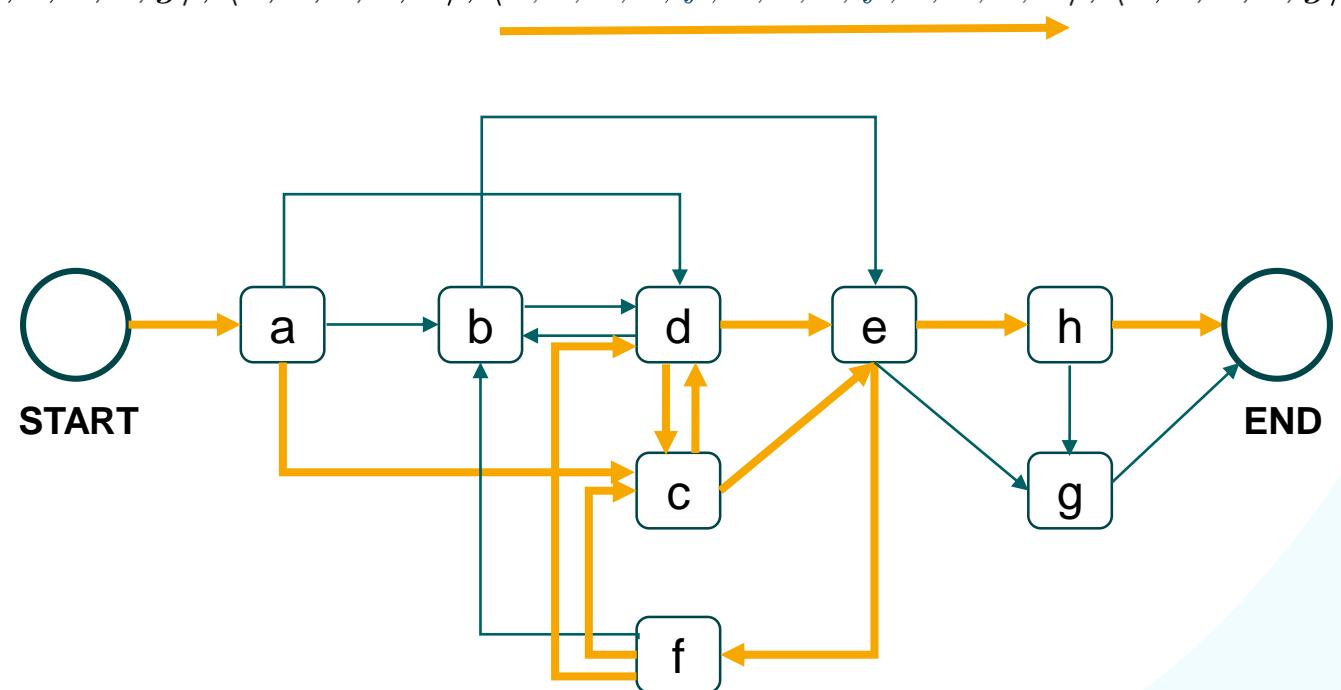
d = check ticket

e = decide

f = reinitiate request

g = pay compensation

h = reject request



Directly-Follows Graphs – Example

Assume we have a simplified log:

$$L = [\langle a, b, d, e, h \rangle, \langle a, d, c, e, g \rangle, \langle a, c, d, e, f, b, d, e, g \rangle, \langle a, d, b, e, h \rangle, \langle a, c, d, e, f, d, c, e, f, c, d, e, h \rangle, \langle a, c, d, e, g \rangle]$$



a = register request

b = examine thoroughly

c = examine casually

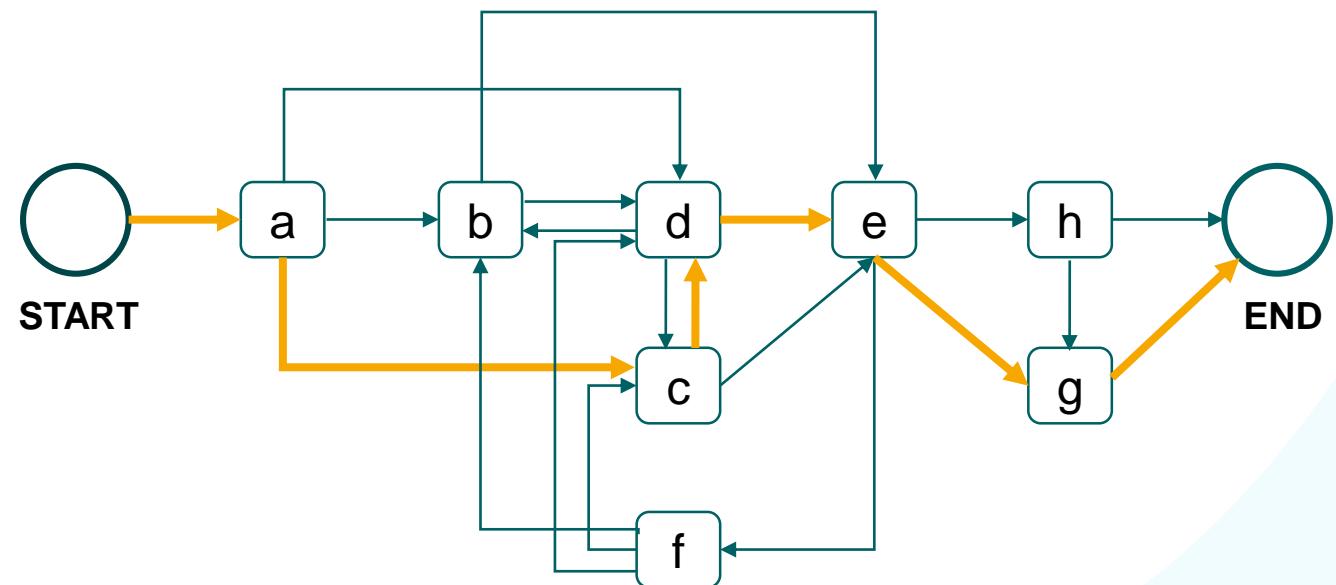
d = check ticket

e = decide

f = reinitiate request

g = pay compensation

h = reject request



Directly-Follows Graphs – Example

Assume we have a simplified log:

$$L = [\langle a, b, d, e, h \rangle, \langle a, d, c, e, g \rangle, \langle a, c, d, e, f, b, d, e, g \rangle, \langle a, d, b, e, h \rangle, \langle a, c, d, e, f, d, c, e, f, c, d, e, h \rangle, \langle a, c, d, e, g \rangle]$$

a = register request

b = examine thoroughly

c = examine casually

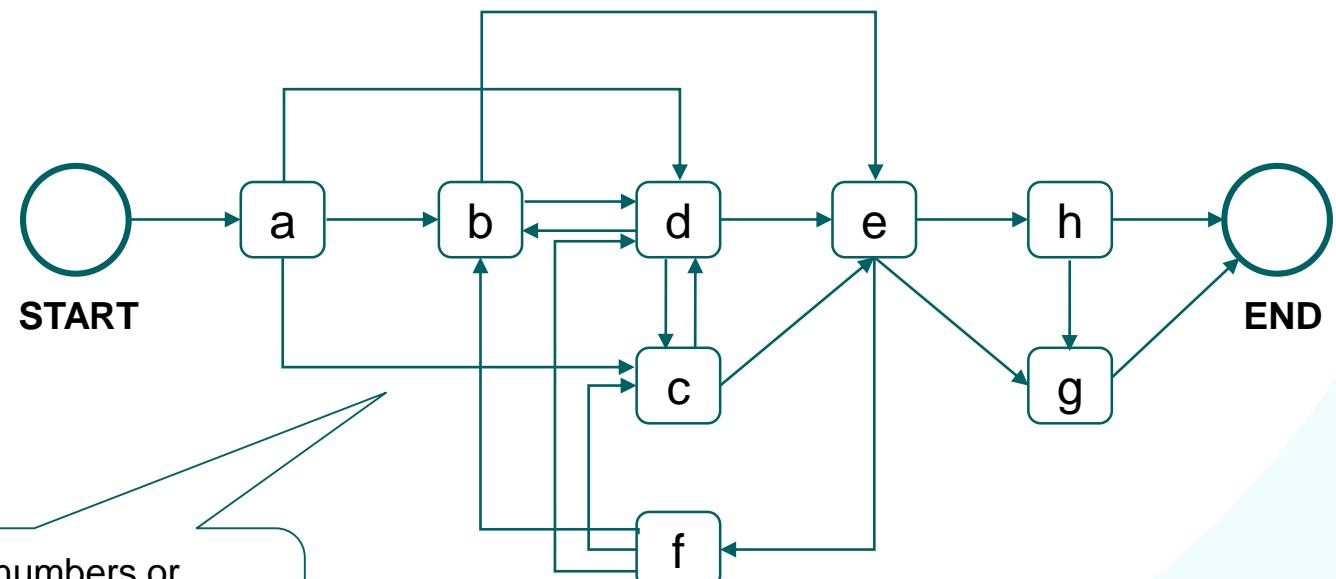
d = check ticket

e = decide

f = reinitiate request

g = pay compensation

h = reject request



Often we also use numbers or width of edges to show the frequency of particular edges

Directly-Follows Graphs – Example

Assume we have a simplified log:

$$L = [\langle a, b, d, e, h \rangle, \langle a, d, c, e, g \rangle, \langle a, \underline{c}, d, e, f, b, d, e, g \rangle, \langle a, d, b, e, h \rangle, \langle \underline{a}, \underline{c}, d, e, f, d, c, e, f, c, d, e, h \rangle, \langle a, \underline{c}, d, e, g \rangle]$$

a = register request

b = examine thoroughly

c = examine casually

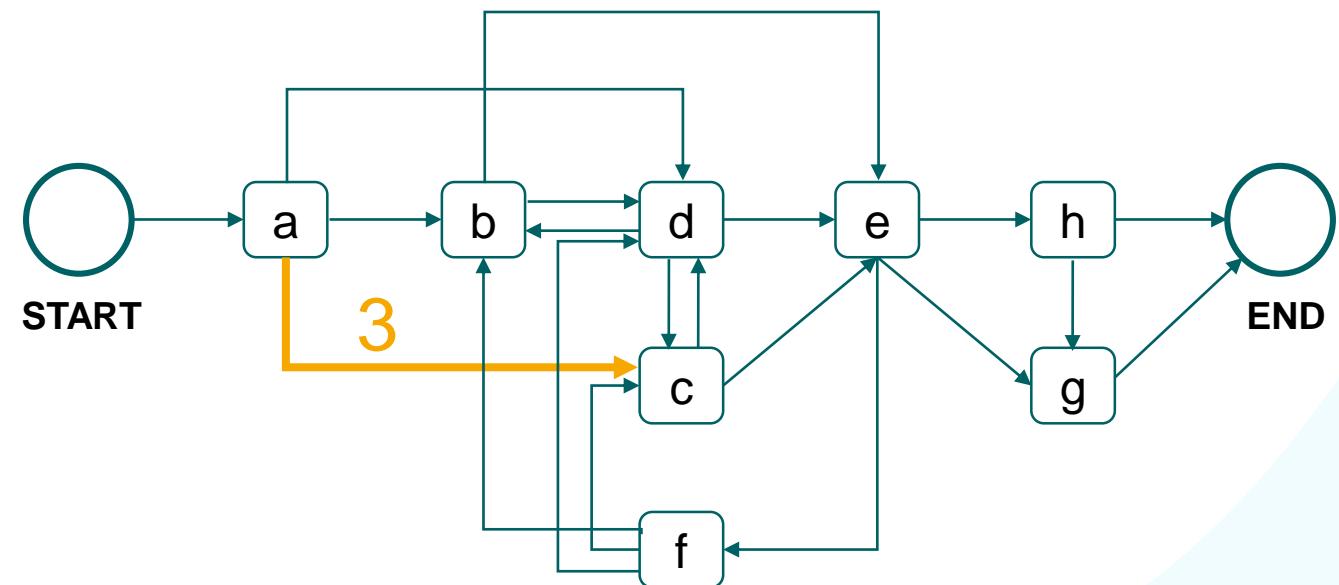
d = check ticket

e = decide

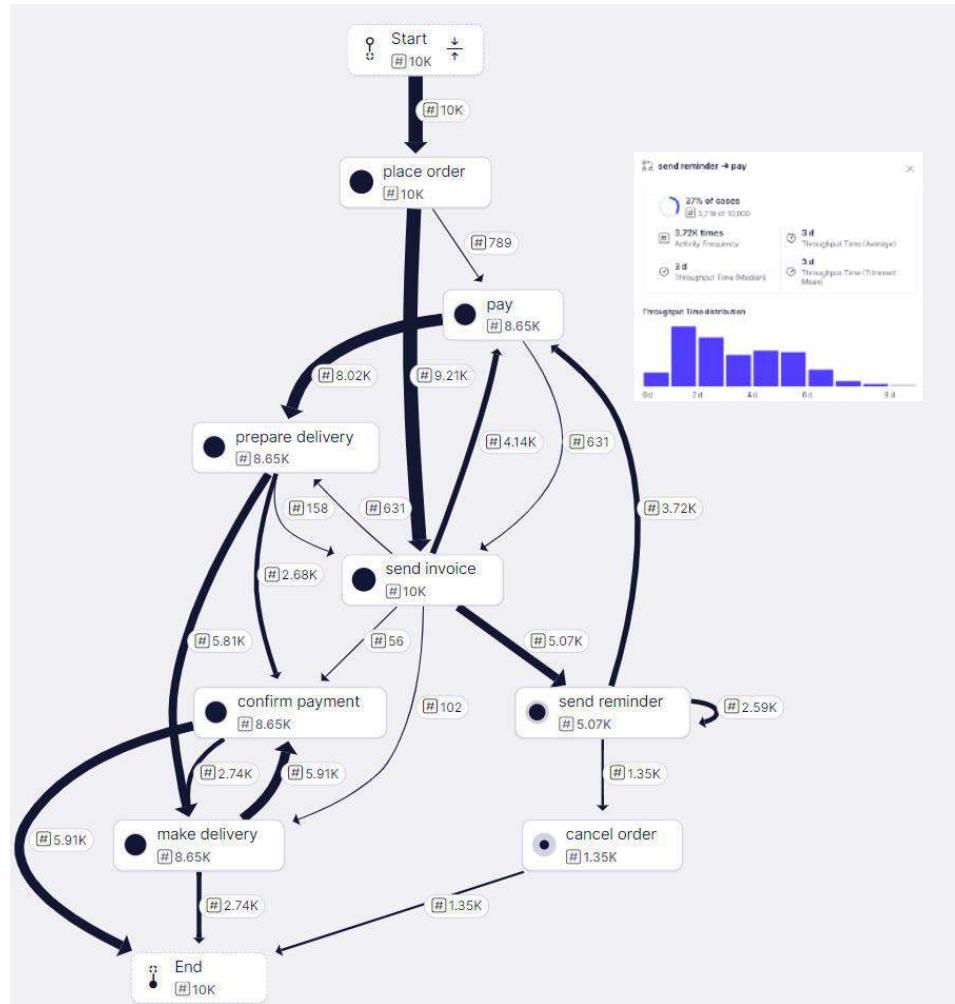
f = reinitiate request

g = pay compensation

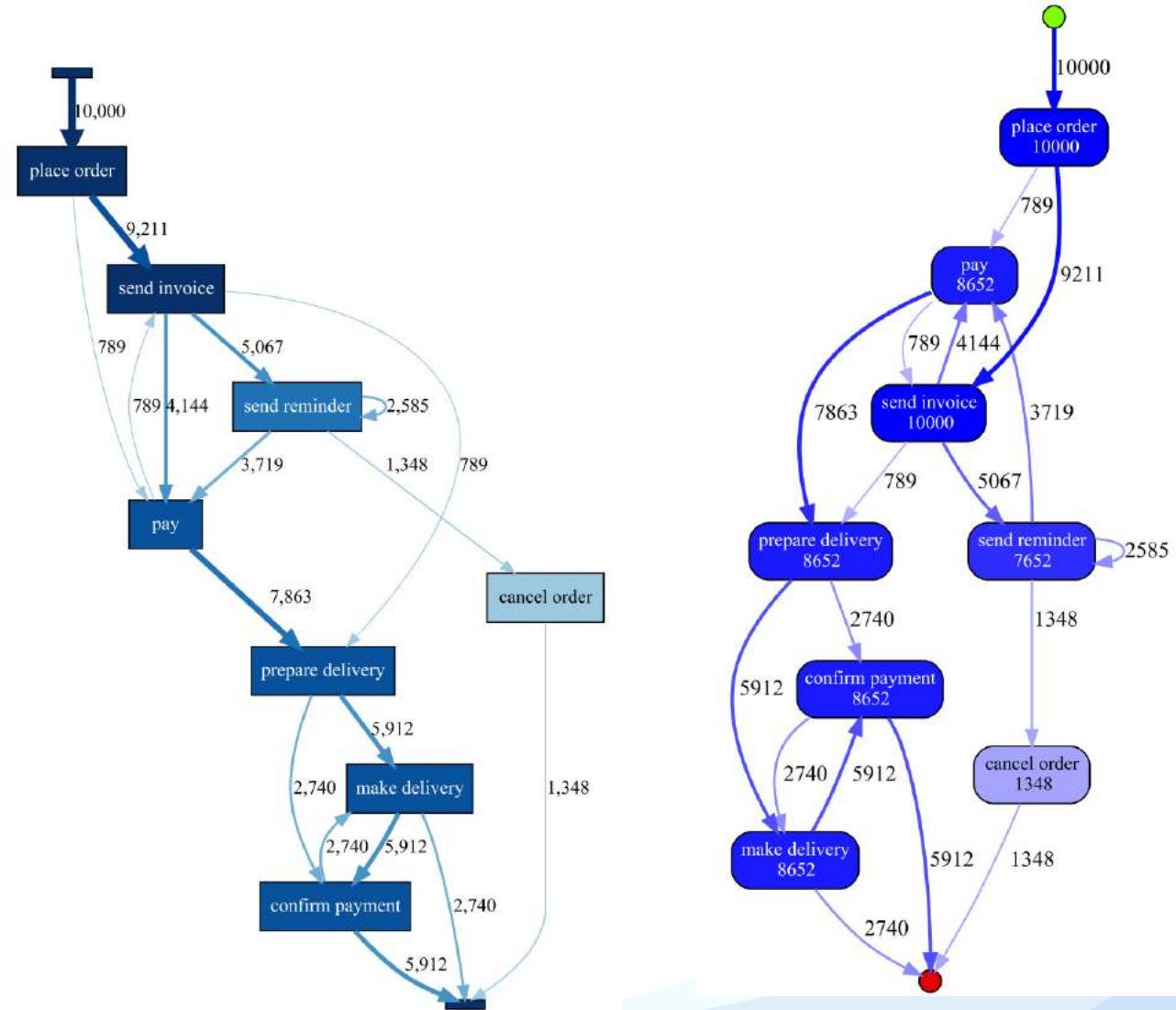
h = reject request



Directly-Follows Graphs – Example Visualization in Tools



DFG in Celonis

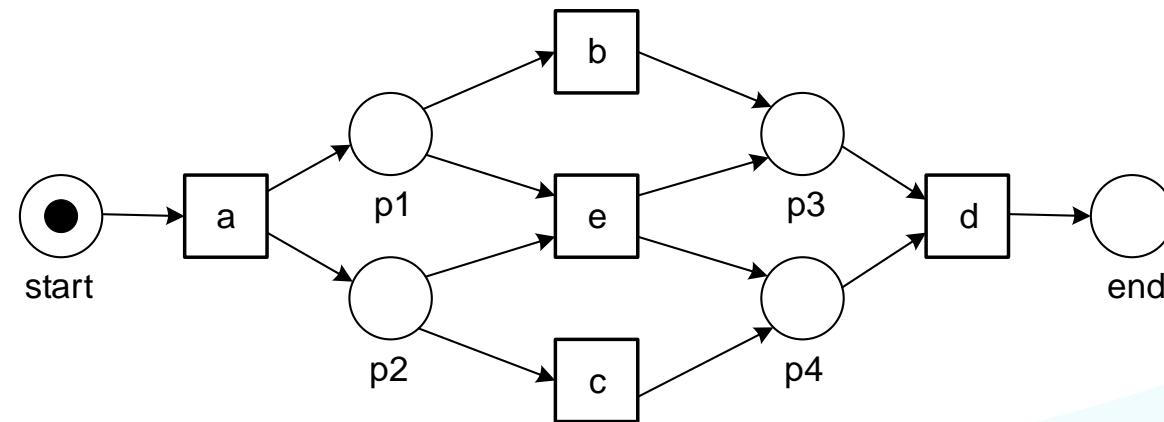


DFGs in ProM

Petri Nets

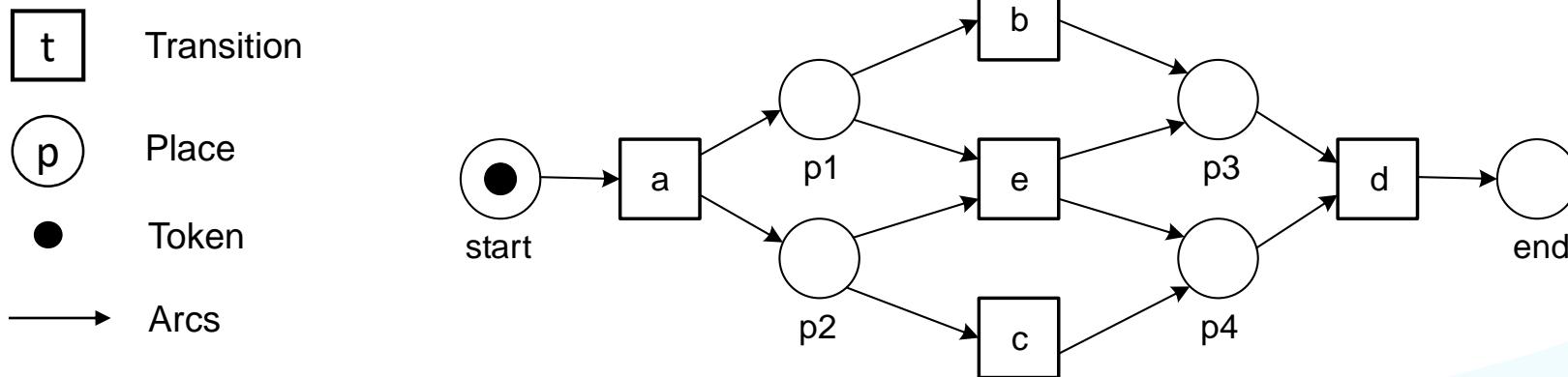
- Wide variety of application domains
- Oldest and most investigated process modeling language
- Petri net is a bipartite graph consisting of **places** and **transitions**

 Transition
 Place
 Token
→ Arcs

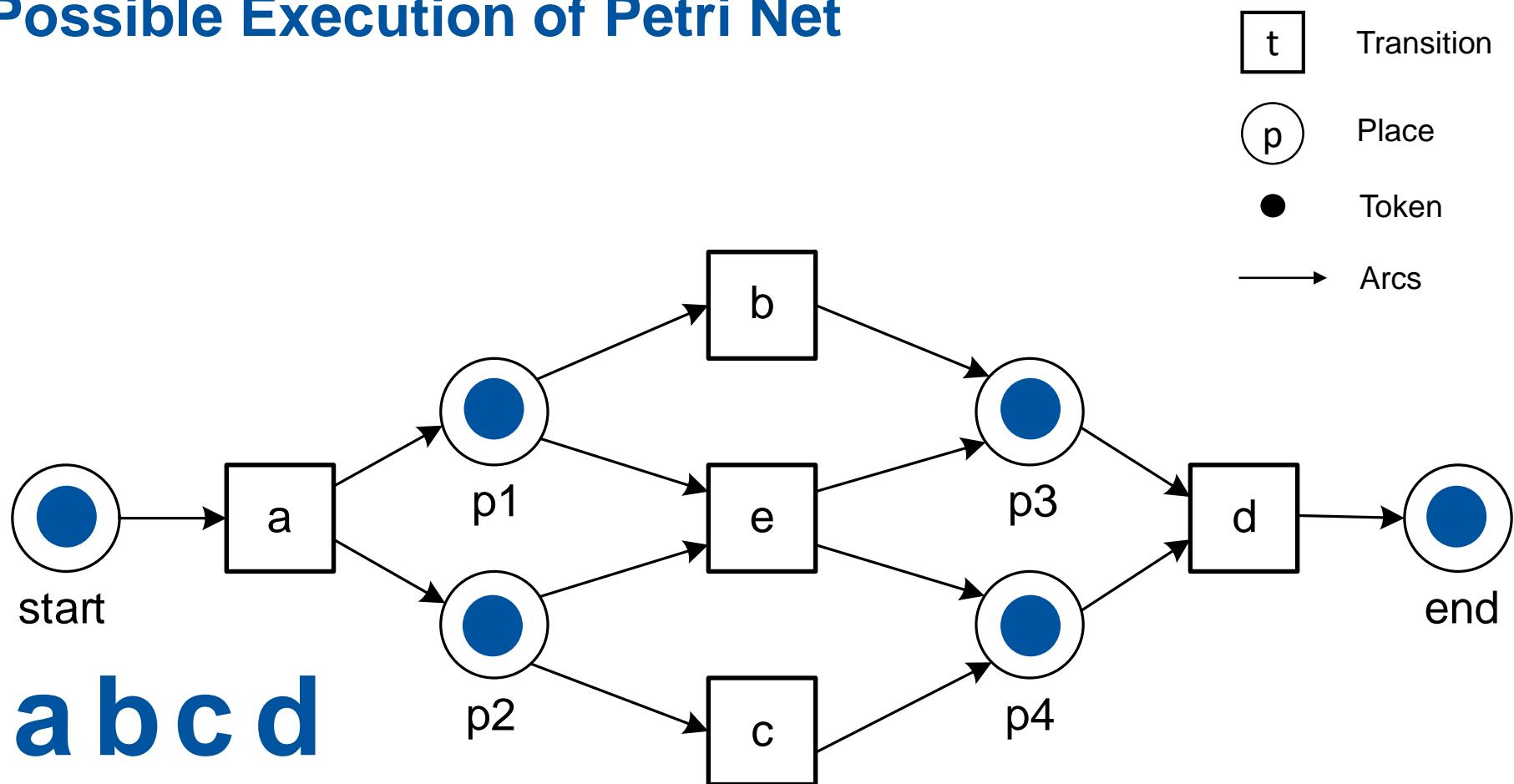


Petri Nets

- **Transitions** can **fire** if all input places have tokens (when firing they produce tokens in **all output places** and consume tokens from **all input places**)
- In process mining we use mainly **accepting Petri nets**
(accepting Petri nets have a defined START and END state)



One Possible Execution of Petri Net

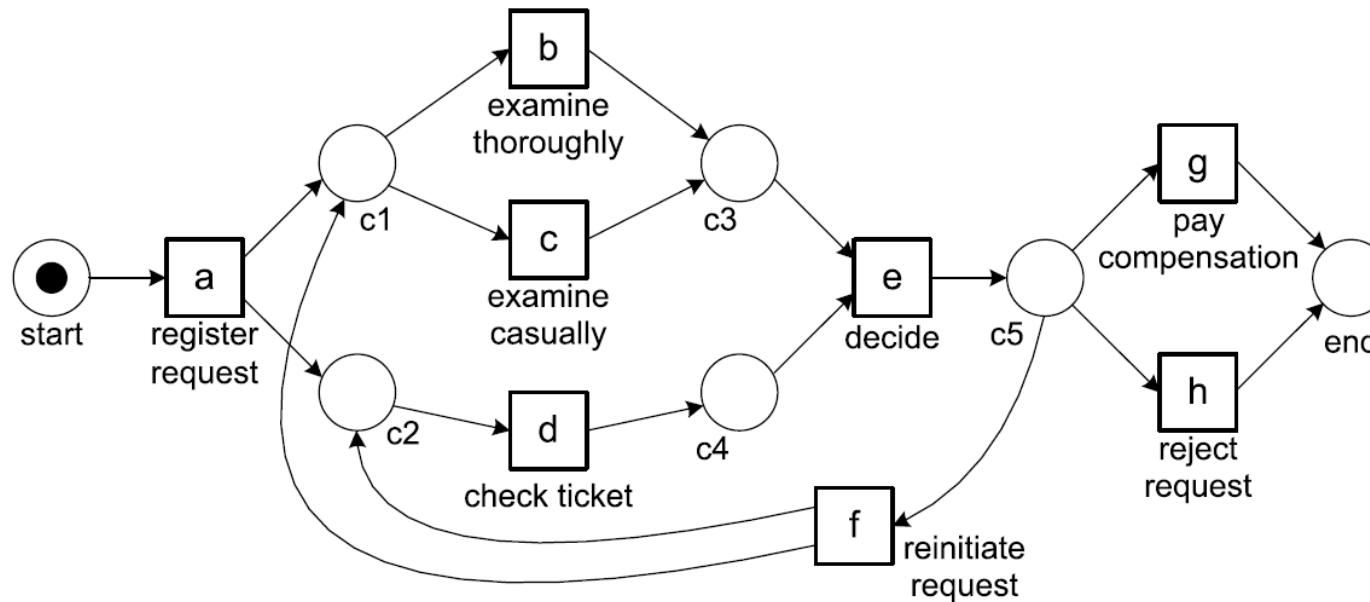


Initial marking [start], final marking [end]

Petri Nets – Example

The same log previously used to build a DFG can be used to discover the following Petri net:

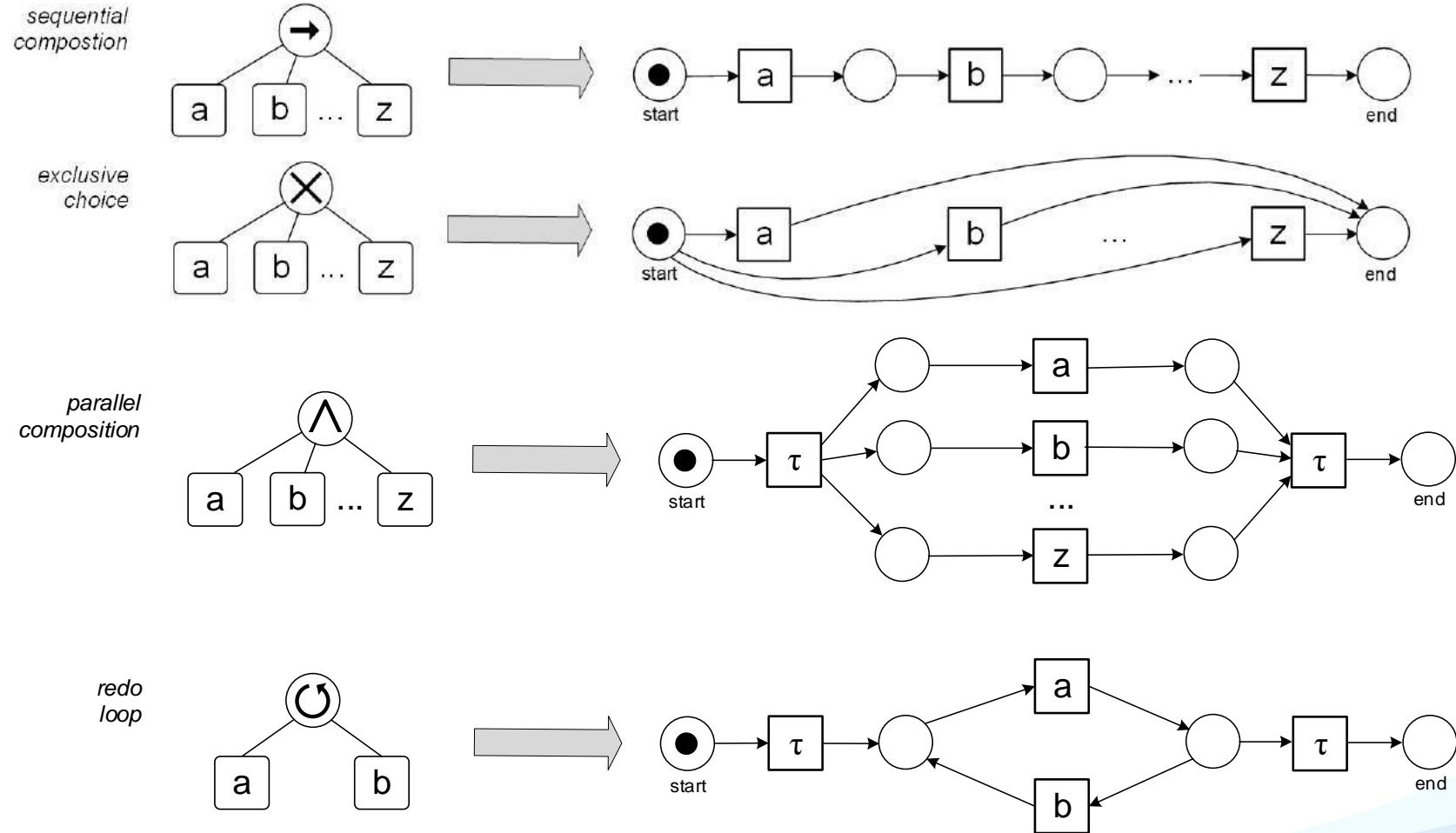
$$L = [\langle a, b, d, e, h \rangle, \langle a, d, c, e, g \rangle, \langle a, c, d, e, f, b, d, e, g \rangle, \langle a, d, b, e, h \rangle, \langle a, c, d, e, f, d, c, e, f, c, d, e, h \rangle, \langle a, c, d, e, g \rangle]$$



Process Tree – Four Types of Operators

- Often used in process mining because it guarantees favourable properties simplifying many applications
- Hierarchically structures the process into behavioural blocks (represented as a tree)
- The behaviour of a block is defined by operators

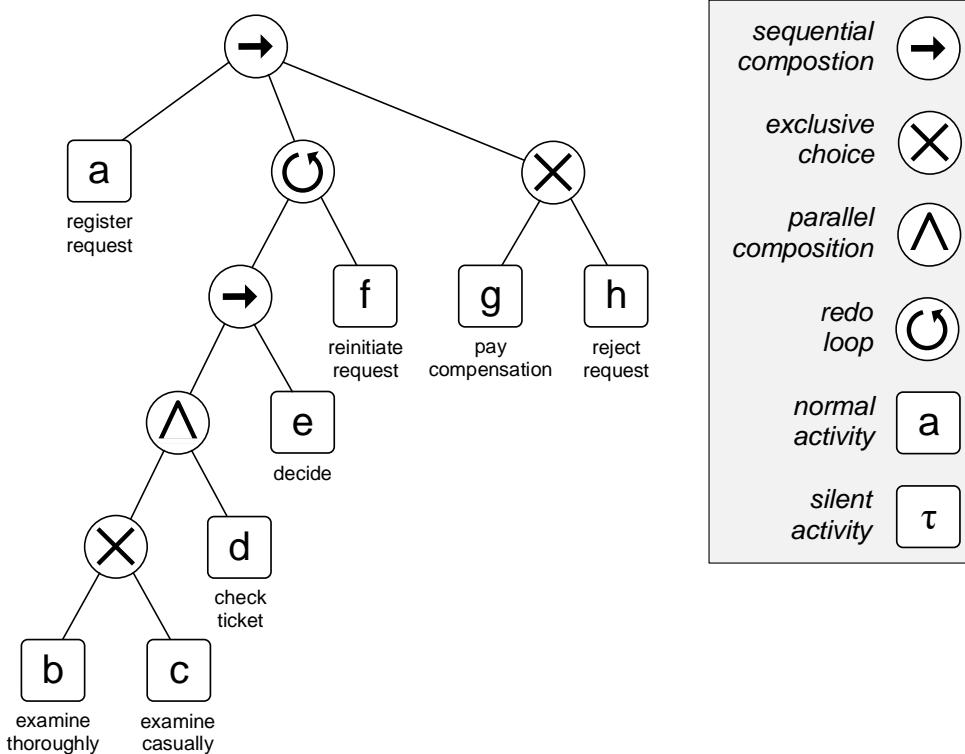
Process Tree – Four Types of Operators



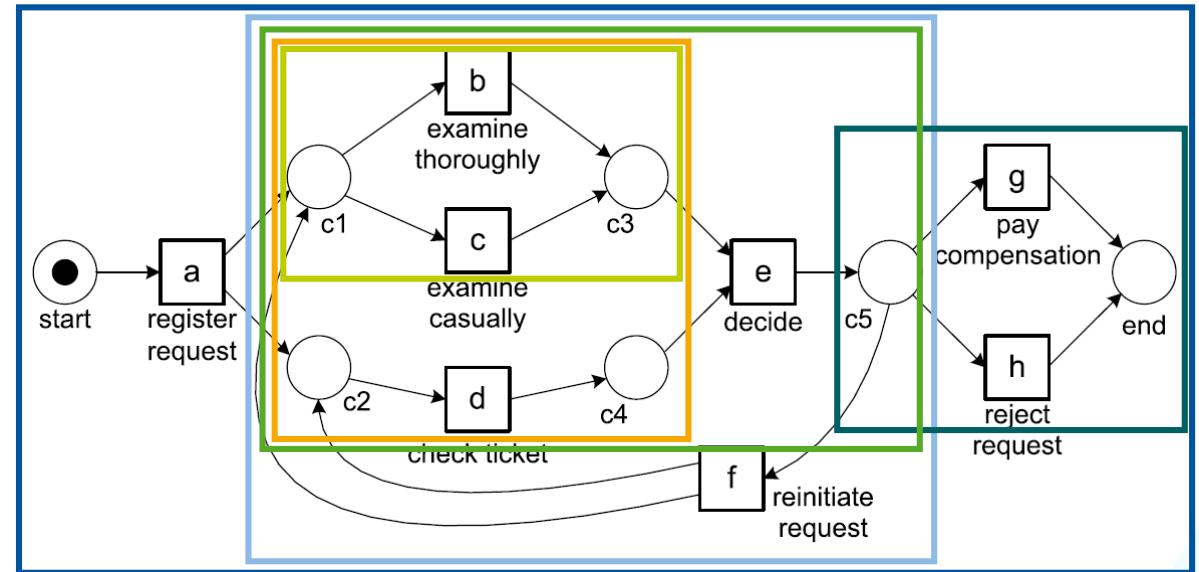
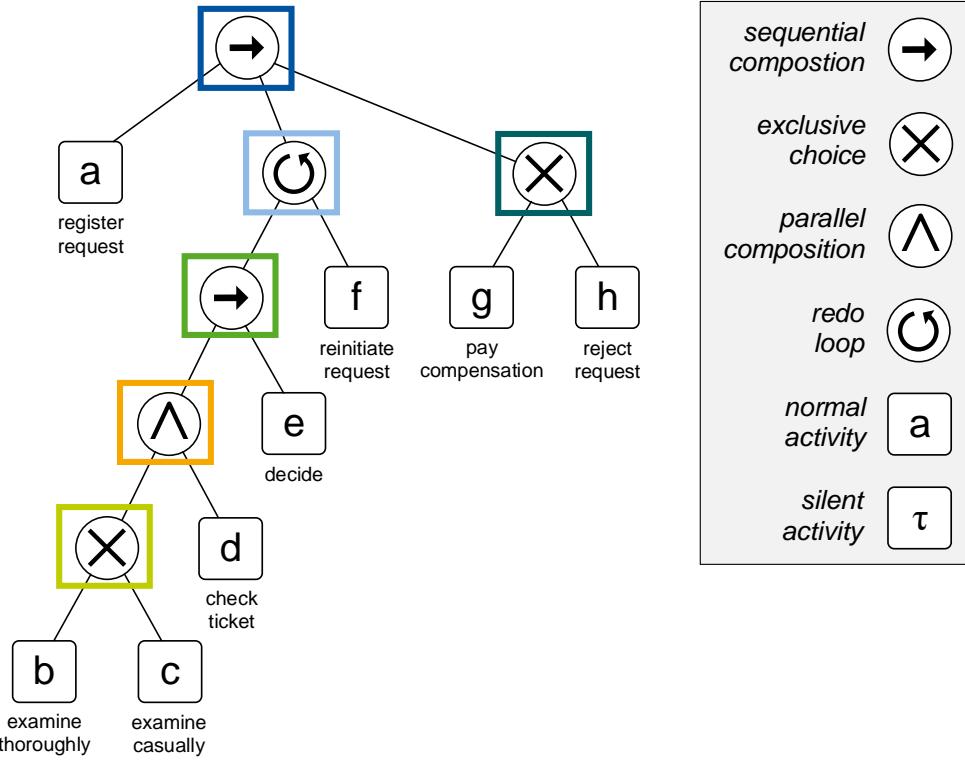
Process Tree – Example

The same log previously used to build a DFG can be used to discover the following process tree:

$$L = [\langle a, b, d, e, h \rangle, \langle a, d, c, e, g \rangle, \langle a, c, d, e, f, b, d, e, g \rangle, \langle a, d, b, e, h \rangle, \langle a, c, d, e, f, d, c, e, f, c, d, e, h \rangle, \langle a, c, d, e, g \rangle]$$



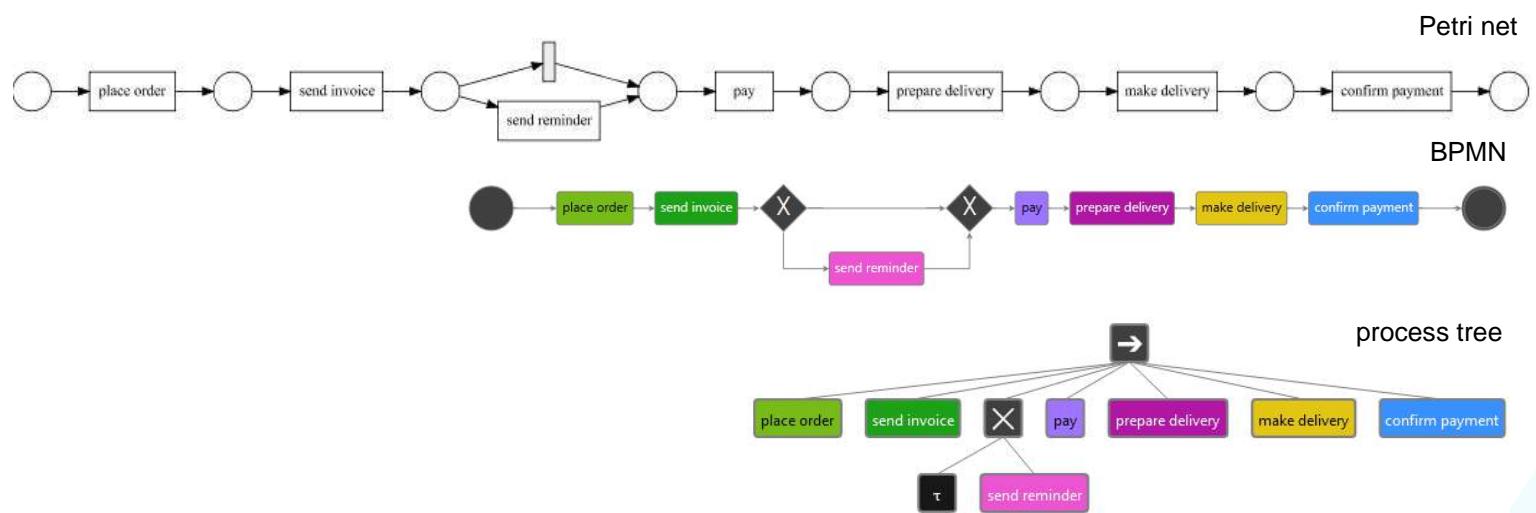
Process Tree Semantics



Process Models

The same process can be visualized in many ways:

- DFGs
- Petri Nets
- Process Trees
- BPMN models

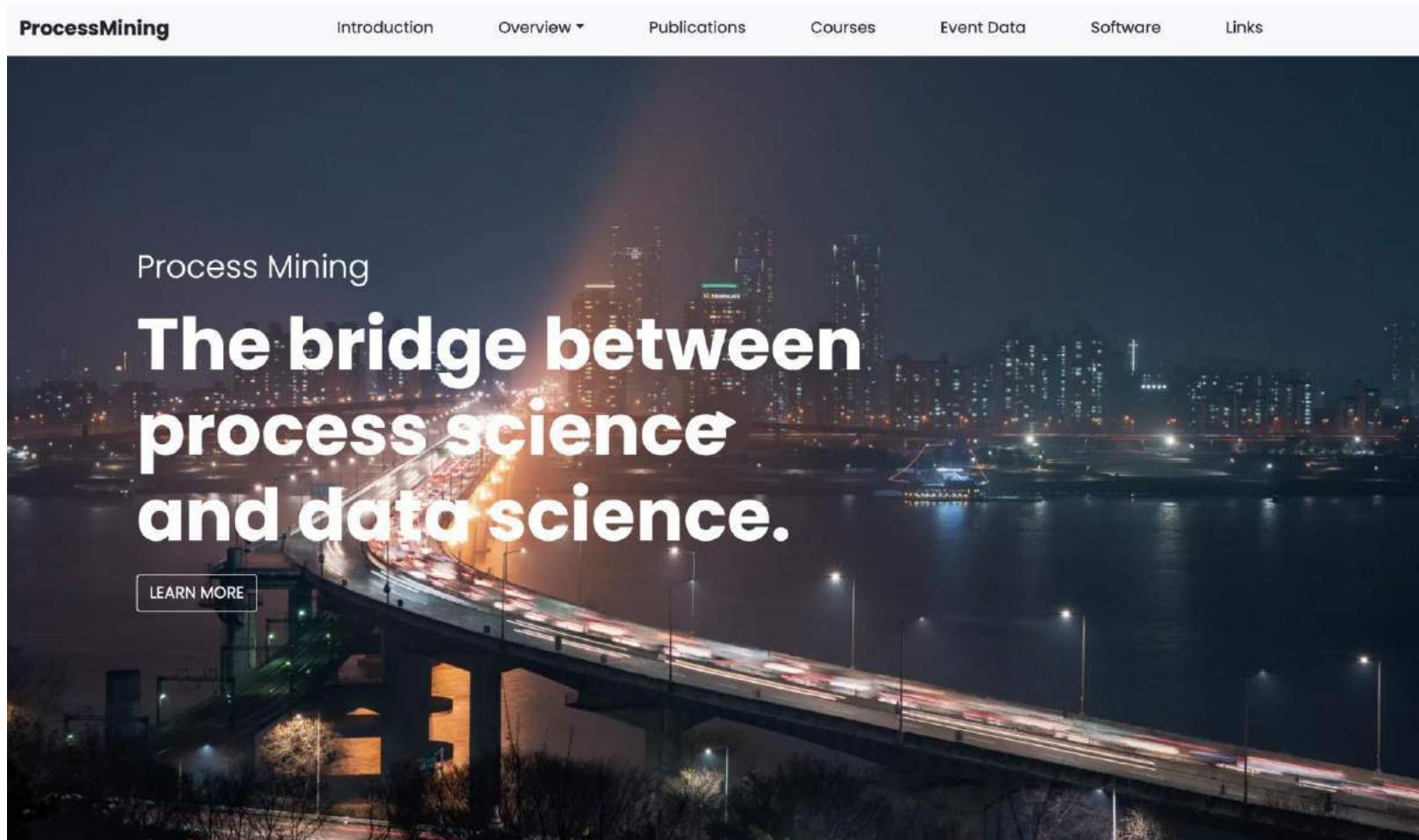


The conversion of process mining results into desired notations is relatively easy.

Introduction to Process Mining

1. Process Mining and Event Data
2. Process Models
- 3. Software Tools**
4. Applications

Over 40 commercial process mining tools (see processmining.org)



For learning resources and more information about tools check out processmining.org

Process Mining Demo

event_log-12666-orders.xlsx - Excel

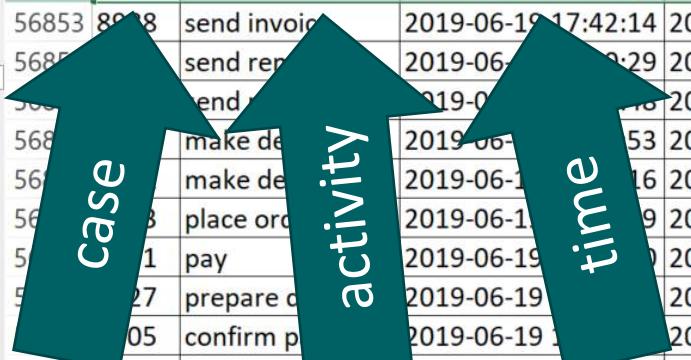
Aalst, W.M.P. van der

FILE HOME INSERT PAGE LAYOUT FORMULAS DATA REVIEW VIEW ACROBAT

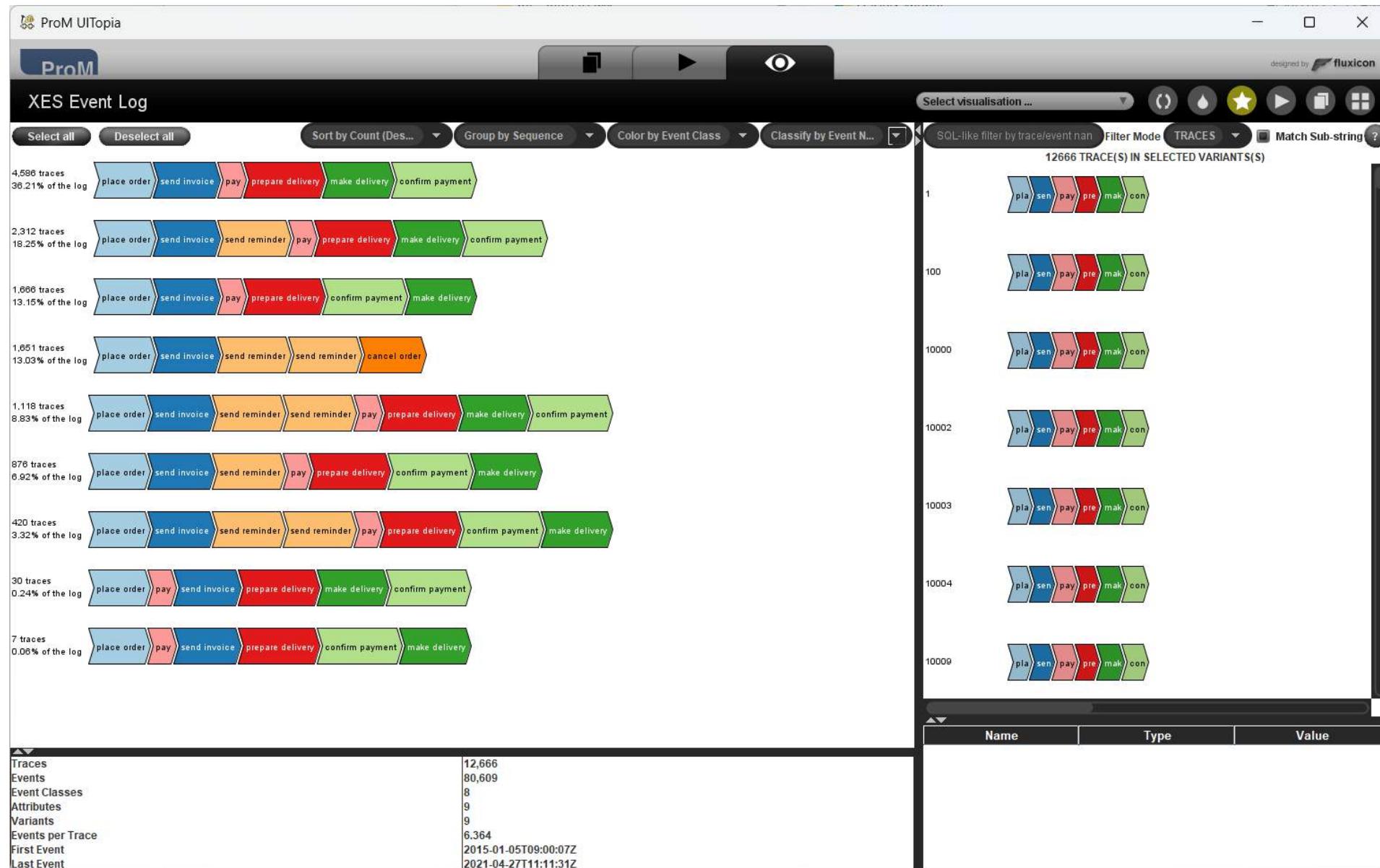
A56852 X ✓ fx 9012

	A	B	C	D	E	F	G	H	I
1	case	activity	start time	end time	resource	product	prod-price	quantity	address
56849	8993	send invoice	2019-06-19 17:02:14	2019-06-19 17:07:13	Jack	APPLE iPhone 6 16 GB	639.0	5	NL-7948DN-12a
56850	8996	send invoice	2019-06-19 17:04:52	2019-06-19 17:08:50	Emily	APPLE iPhone 5s 16 GB	449.0	4	NL-9491BG-41
56851	8918	prepare delivery	2019-06-19 17:19:01	2019-06-19 17:22:58	Aiden	APPLE iPhone 6 16 GB	639.0	3	NL-7826GD-9
56852	9012	place order	2019-06-19 17:27:31	2019-06-19 17:33:46	Sophia	MOTOROLA Moto G	199.0	2	NL-7828AM-11a
56853	8918	send invoice	2019-06-19 17:42:14	2019-06-19 17:47:22	Lily	SAMSUNG Core Prime G361	135.0	2	NL-7907EJ-42
56854	8927	send reminder	2019-06-19 17:48:29	2019-06-19 18:21:58	Luke	SAMSUNG Galaxy S4	329.0	1	NL-7822AW-5
56855	8918	send invoice	2019-06-19 18:18:18	2019-06-19 18:21:11	Luke	APPLE iPhone 6 16 GB	639.0	5	NL-9521KJ-34
56856	8927	make delivery	2019-06-19 18:53	2019-06-19 18:25:46	Avery	SAMSUNG Galaxy S4	329.0	2	NL-7948BX-10
56857	8927	make delivery	2019-06-19 18:56	2019-06-19 18:30:34	Abigail	SAMSUNG Galaxy S4	329.0	6	NL-9468HG-14
56858	8913	place order	2019-06-19 19:19	2019-06-19 19:17:16	Emma	MOTOROLA Moto G	199.0	2	NL-7822AW-5
56859	8911	pay	2019-06-19 19:20	2019-06-19 19:22:48	Emily	APPLE iPhone 6 16 GB	639.0	-5	
56860	8927	prepare order	2019-06-19 19:20	2019-06-19 22:21:48	Lucas	APPLE iPhone 6 16 GB	639.0	36	
56861	8905	confirm payment	2019-06-19 19:20	2019-06-19 20:05:02	Lily	SAMSUNG Galaxy S4	329.0	2	
56862	9014	place order	2019-06-19 22:02:32	2019-06-19 22:08:02	Aiden	SAMSUNG Galaxy S4	329.0	25	
56863	8922	send reminder	2019-06-19 22:18:26	2019-06-19 22:35:06	Luke	SAMSUNG Galaxy S4	329.0	4	
56864	8927	confirm payment	2019-06-19 22:21:12	2019-06-19 22:30:05	Lily	APPLE iPhone 6 16 GB	639.0	2	NL-7931TV-36
56865	9015	place order	2019-06-20 07:16:24	2019-06-20 07:22:23	Emma	APPLE iPhone 6s Plus 64 GB	969.0	7	NL-7944BB-6
56866	8903	cancel order	2019-06-20 08:59:43	2019-06-20 09:07:33	Lily	SAMSUNG Galaxy S4	329.0	1	NL-7942GT-2
56867	9003	send invoice	2019-06-20 09:11:11	2019-06-20 09:19:46	Jack	SAMSUNG Galaxy S4	329.0	1	NL-7948DN-12a
56868	8836	make delivery	2019-06-20 09:36:17	2019-06-20 10:59:53	Ella	APPLE iPhone 6s Plus 64 GB	969.0	4	NL-7833HT-15
56869	8950	send reminder	2019-06-20 09:36:54	2019-06-20 09:59:18	Abigail	SAMSUNG Galaxy J5	219.99	4	NL-7887AC-13
56870	8938	pay	2019-06-20 09:57:31	2019-06-20 10:04:09	Lily	SAMSUNG Galaxy S4	329.0	3	NL-7826GD-9
56871	9016	place order	2019-06-20 10:00:10	2019-06-20 10:04:01	Aiden	SAMSUNG Galaxy S4	329.0	4	NL-7918AE-48b

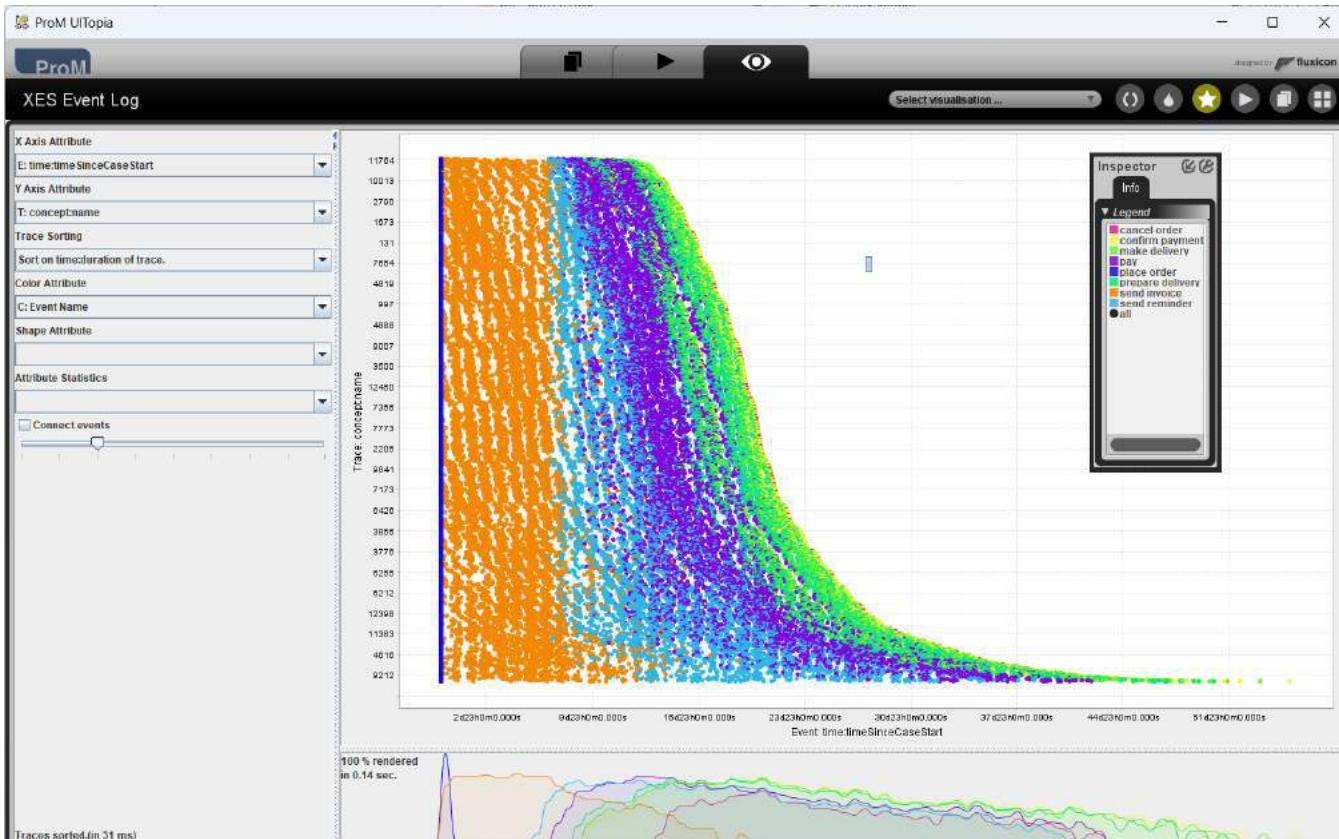
80,609 events
12,666 cases (= orders)
8 unique activities



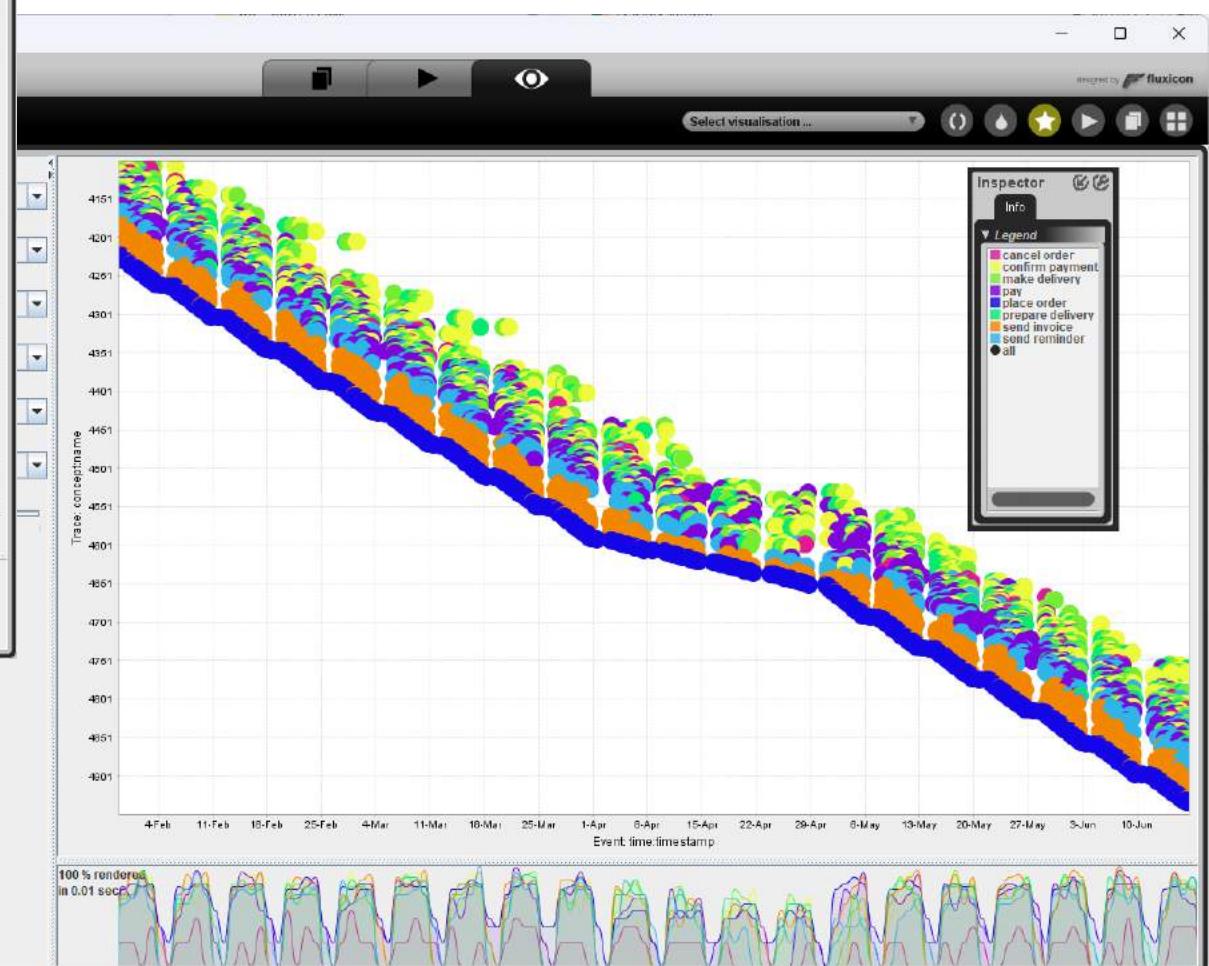
Process Mining Demo – ProM



Process Mining Demo – ProM



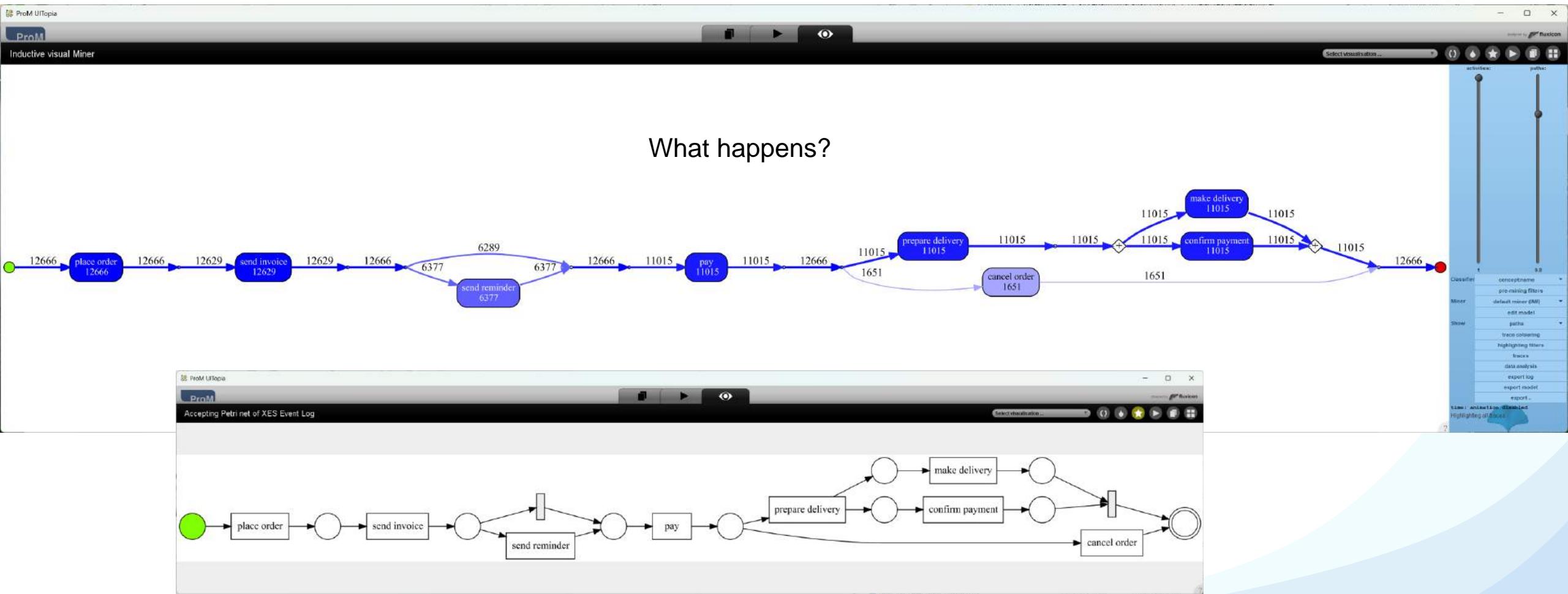
80,609 dots, one for each event



Process Mining Demo – ProM

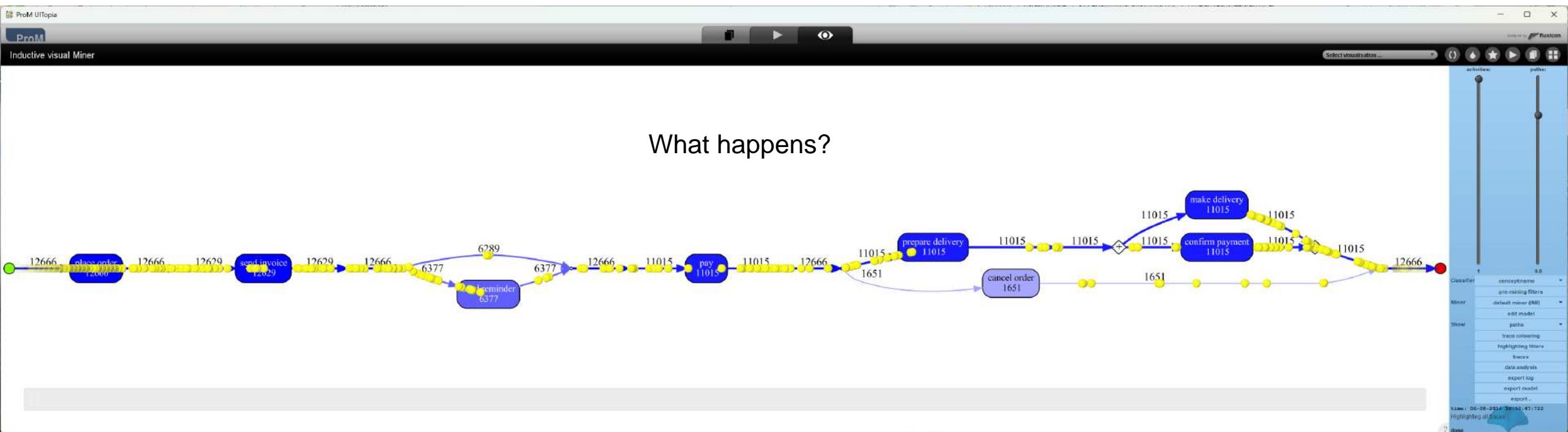
Process Model Discovered Using the Inductive Miner

What happens?



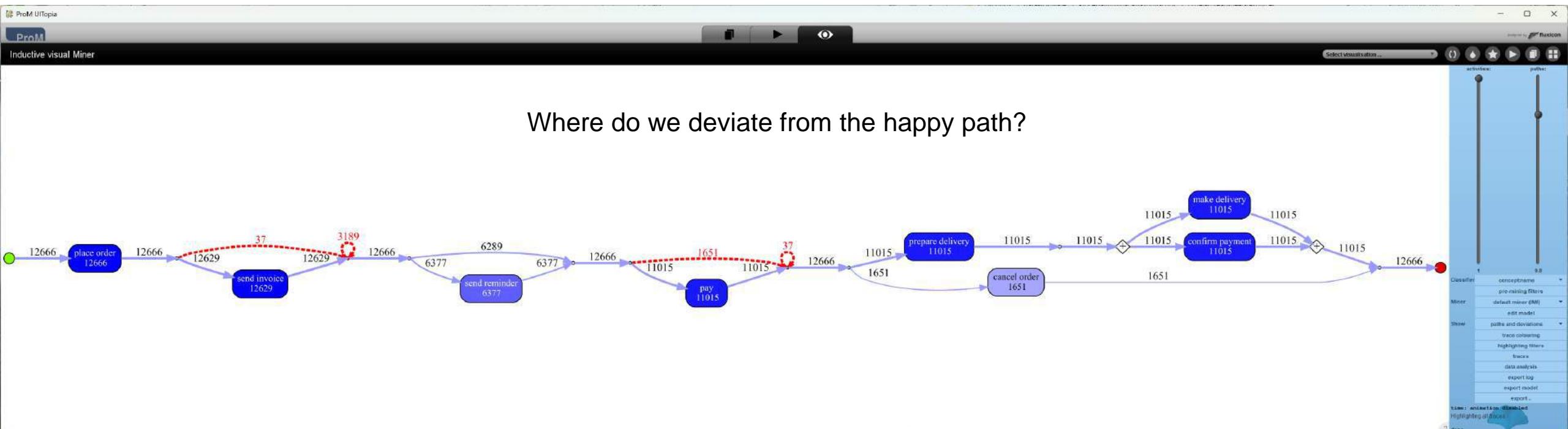
Process Mining Demo – ProM

Process Model Discovered Using the Inductive Miner



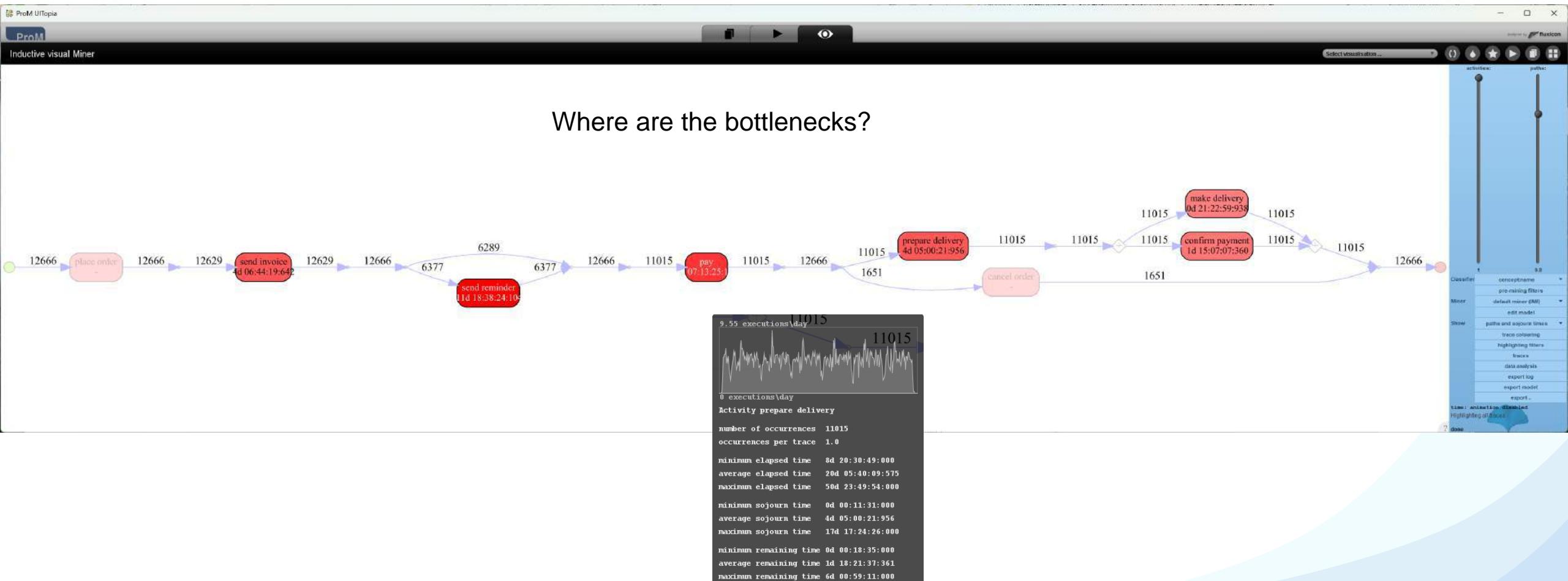
Process Mining Demo – ProM

Using Conformance Checking to See Process Deviations



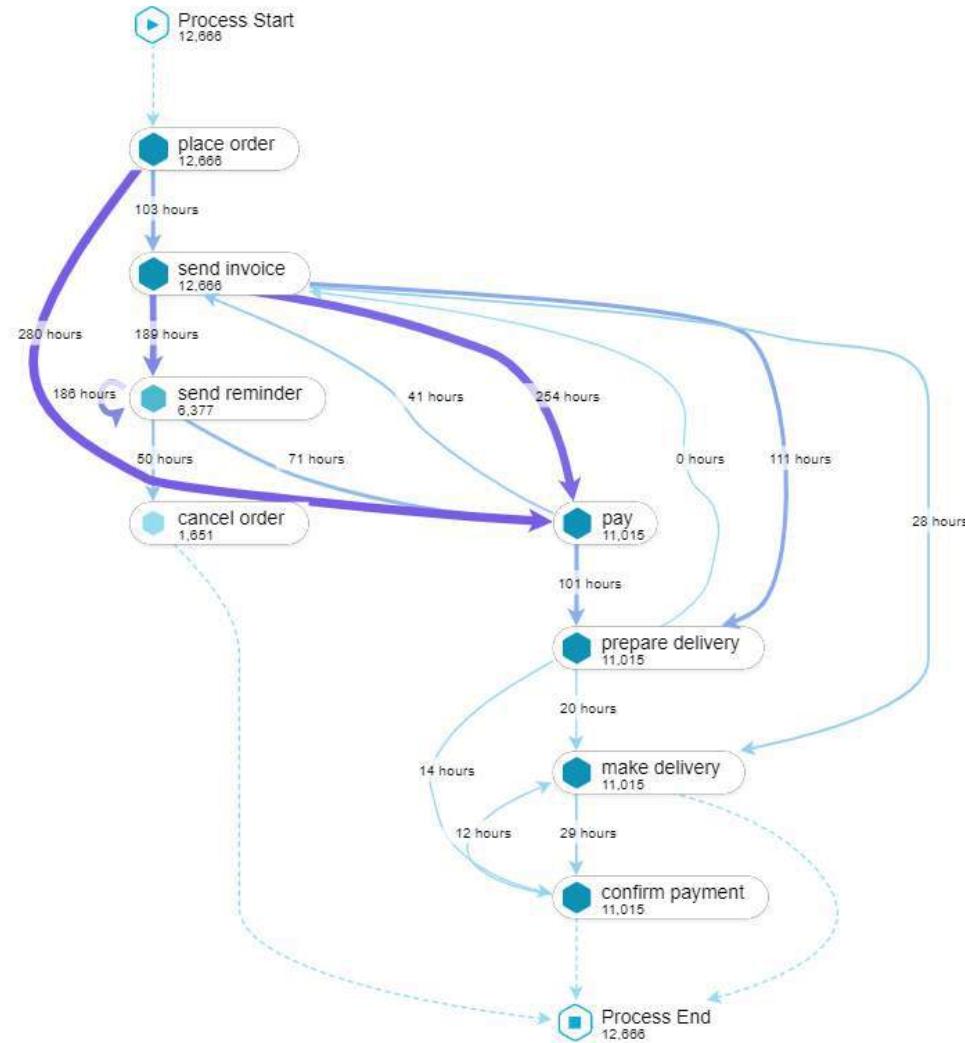
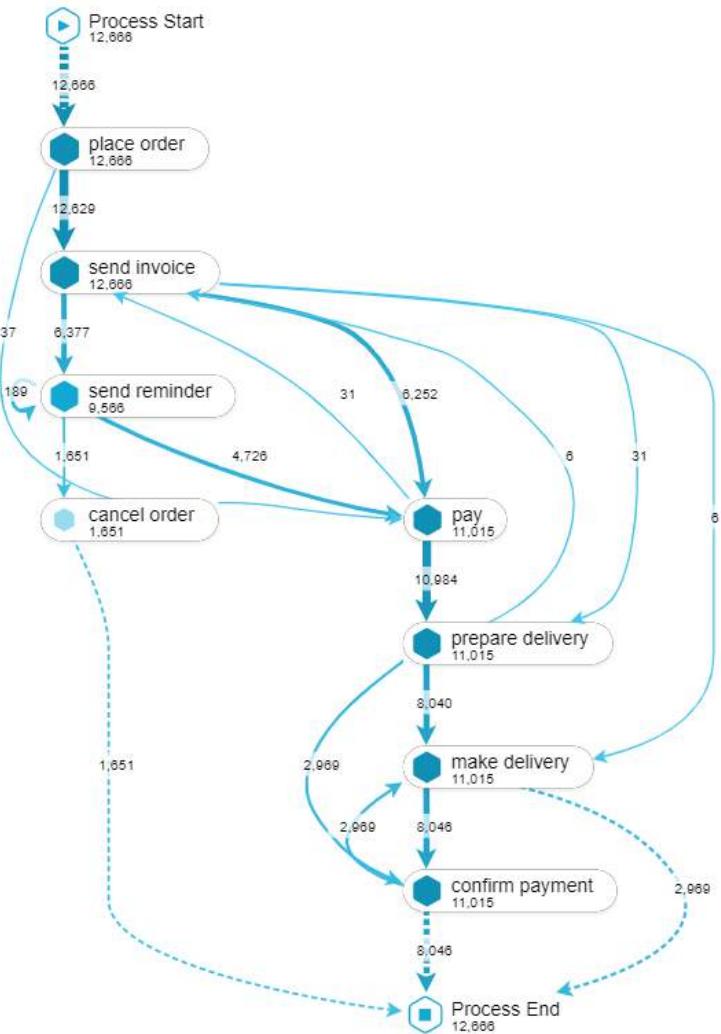
Process Mining Demo – ProM

Bottleneck Analysis – Enriching the Model with Performance Information



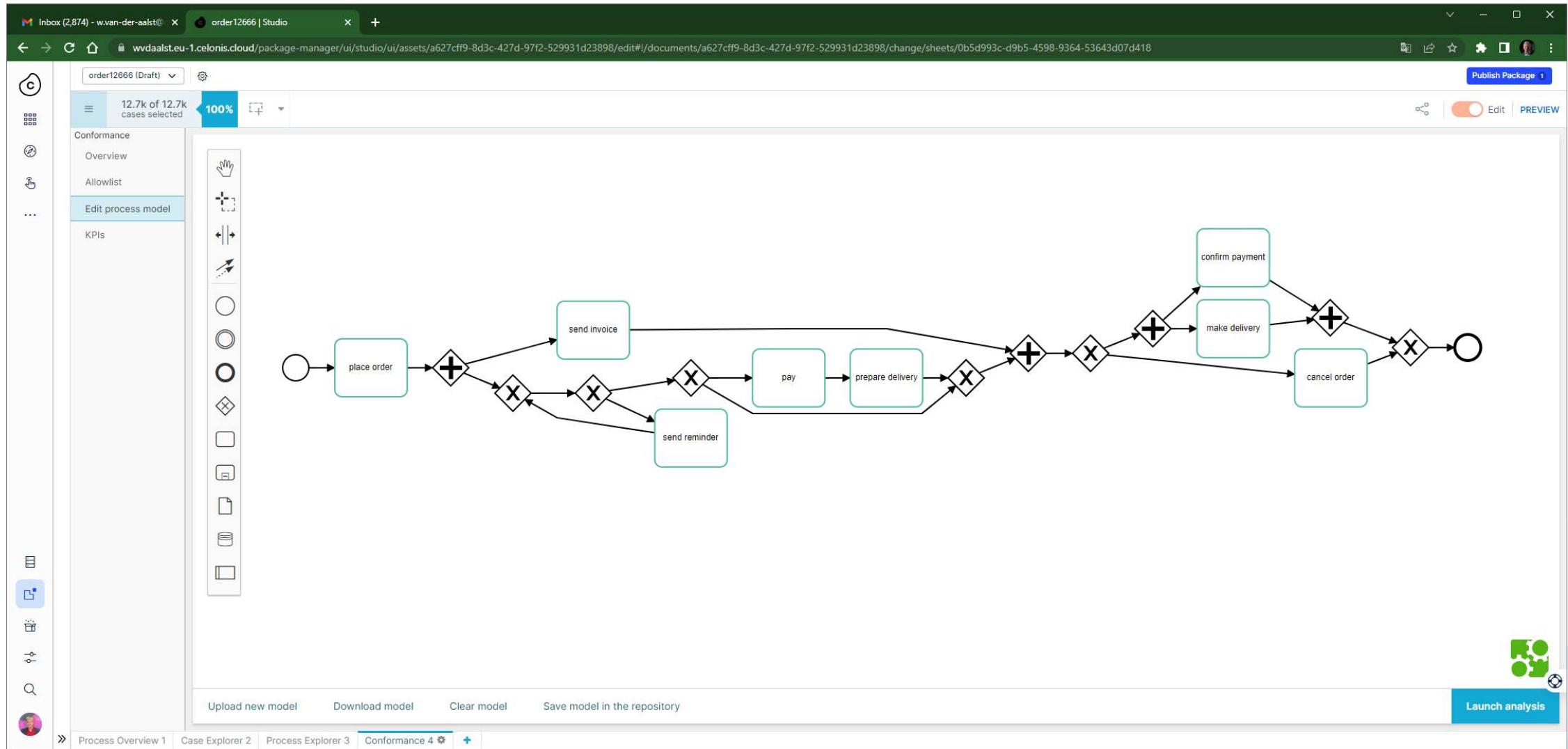
Process Mining Demo – Celonis

Frequencies and Times (Using the same event data)



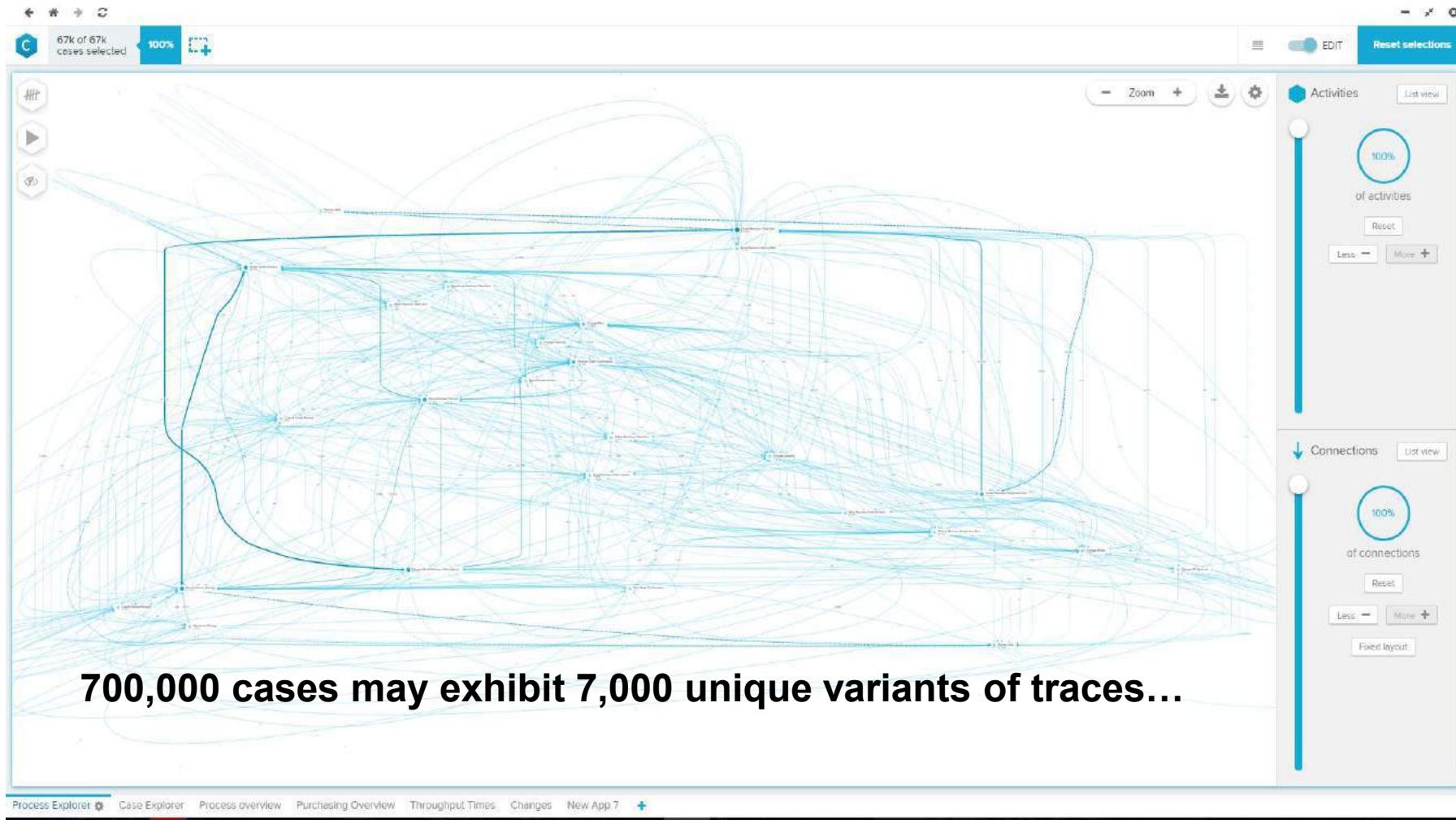
Process Mining Demo – Celonis

Process Model Discovered Using the Inductive Mining Algorithm



Reality Is Not So Simple

Real Processes May Look Like This



Examples of Tools

- **ProM** is the most complete open-source process tool that served as an example for all later tools
 - Download from <https://promtools.org/>
- **Celonis** is the leading commercial tool (there are 40+ other commercial tools)
 - Get via <https://signup.celonis.com/>
 - Free course: <https://www.celonis.com/wils-process-mining-class/>
- In this course, we will mostly use **PM4Py**
 - Python-based process mining library
 - Easy to combine with other data science techniques
 - Collaborative effort PADS@RWTH and Fraunhofer FIT

PM4PY

Fraunhofer
FIT

PADS
Chair of Process and Data Science

RWTH AACHEN
UNIVERSITY

Introduction to Process Mining

1. Process Mining and Event Data
2. Process Models
3. Software Tools
- 4. Applications**

Process Mining is Used in All Domains!

- finance and insurance (Rabobank, Wells Fargo, ADAC, APG, Suncorp, VTB, etc.)
- logistics and transport (Uber, Deutsche Bahn, Lufthansa, Airbus, Vanderlande, etc.)
- production (ABB, Siemens, BMW, Fiat, Bosch, AkzoNobel, Bayer, Neste, etc.)
- healthcare, biomedicine, and pharmacy (Uniklinik RWTH Aachen, Charite University Hospital, GE Healthcare, Philips, Medtronic, Pfizer, Bayer, AstraZeneca, etc.)
- telecom (Deutsche Telekom, Vodafone, A1 Telekom Austria, Telekom Italia, etc.)
- food and retail (Edeka, MediaMarkt, Globus, Zalando, AB InBev, etc.)
- energy (Uniper, Chevron, Shell, BP, E.ON, etc.)
- IT services (Dell, Xerox, IBM, Nokia, ServiceNow, etc.)
- consultancy (Deloitte, Ernst & Young, KPMG, PwC, etc.)

Process Mining Example – Airports



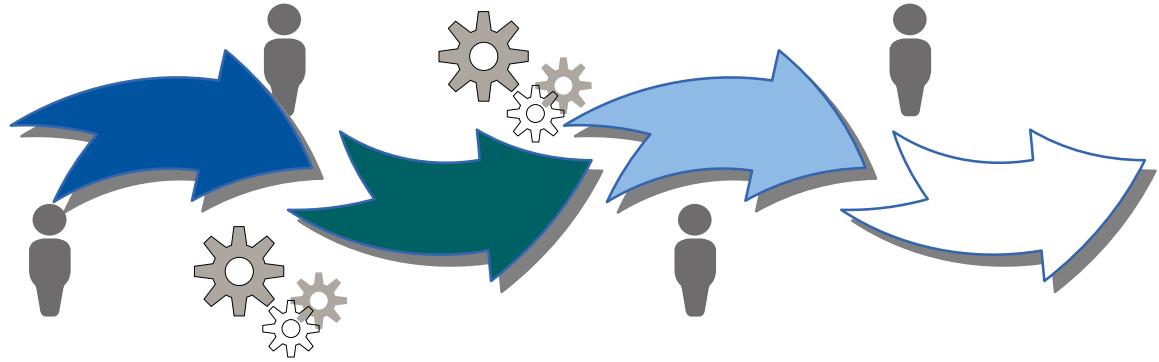
- Why do bags miss a plane?
- Why do I need to wait so long for my bags?
- When and why does the system break down?
- Am I using the available capacity properly?

Part II: Unsupervised Process Mining

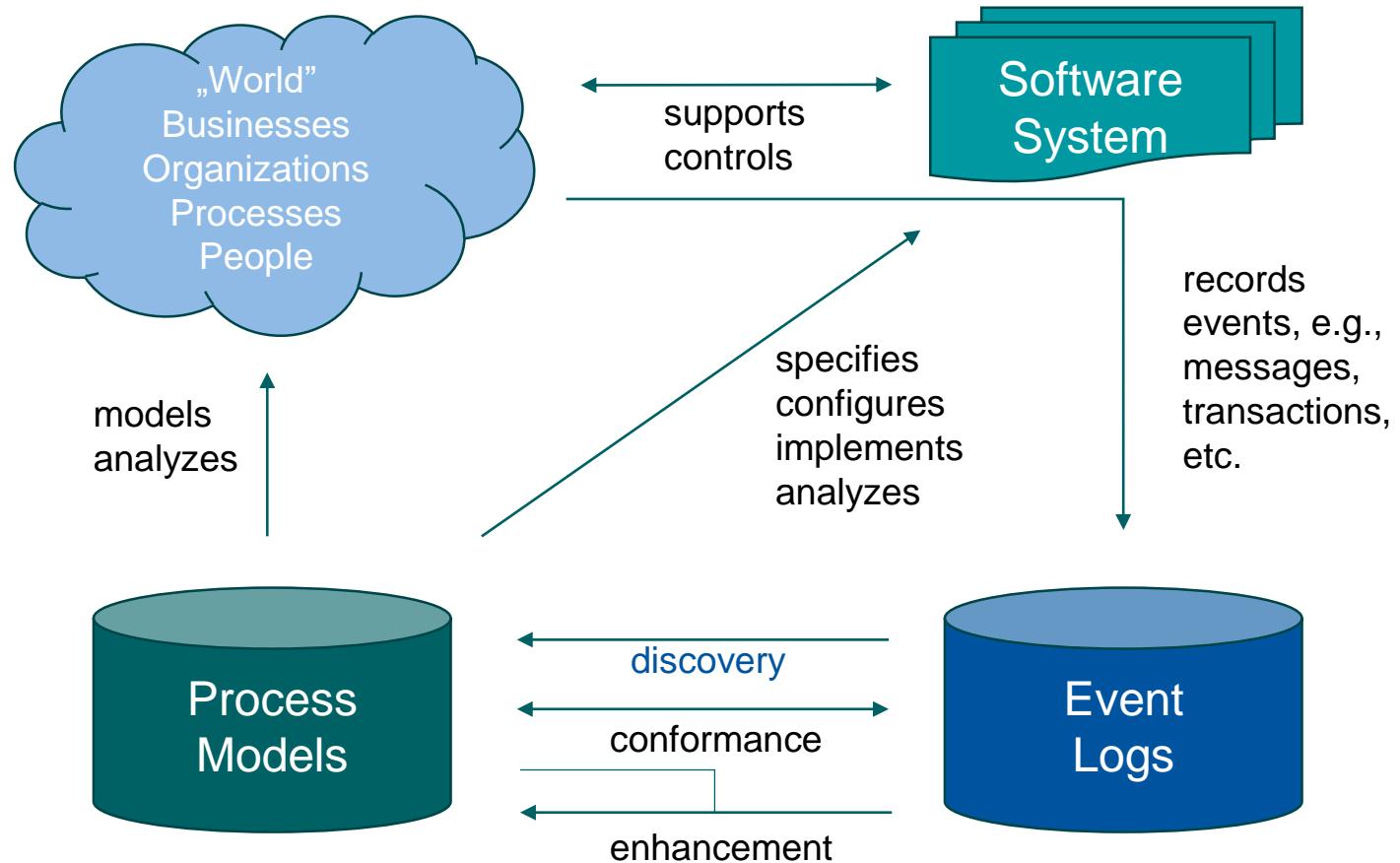
Process Discovery

Unsupervised Process Mining

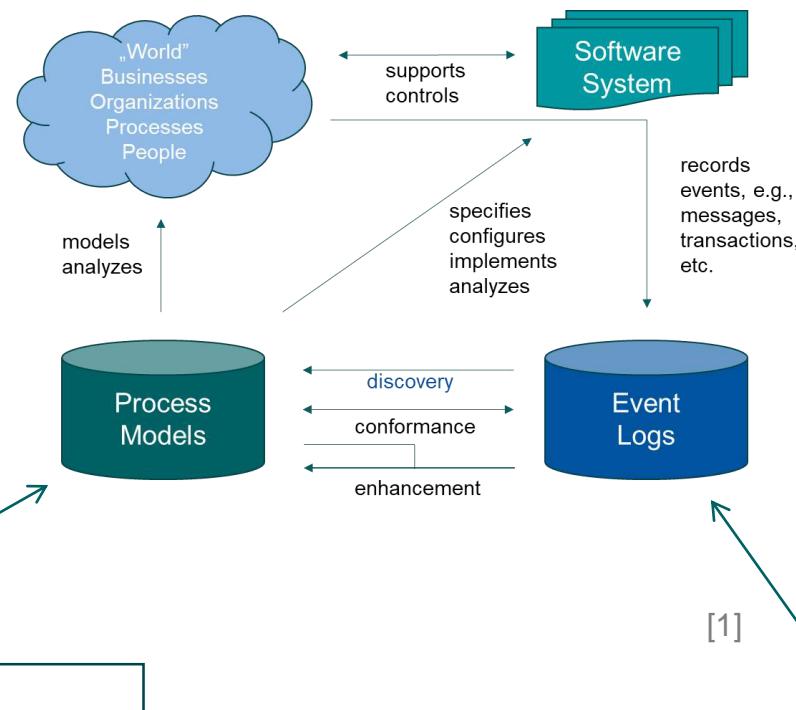
1. **Process Discovery**
2. Bottom-Up Discovery (very brief)
3. Top-Down Discovery (IM)



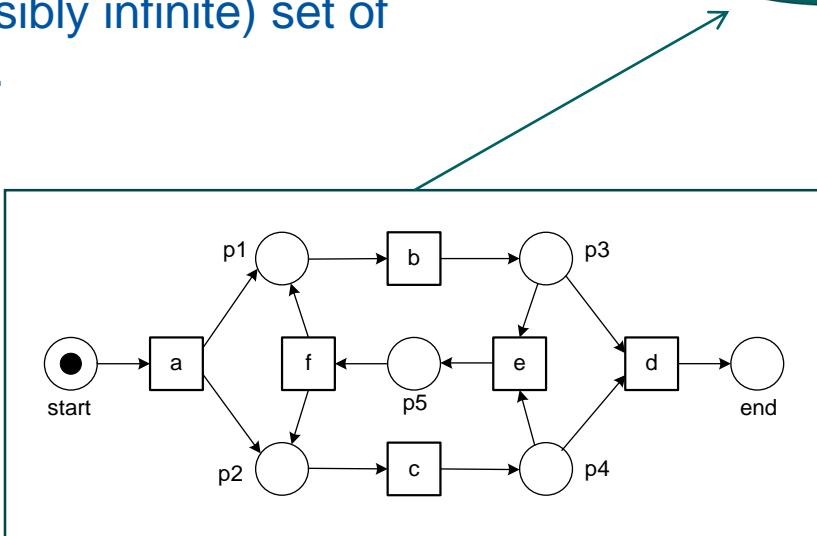
Positioning Process Discovery



Let's Consider the Simplest Setting Possible



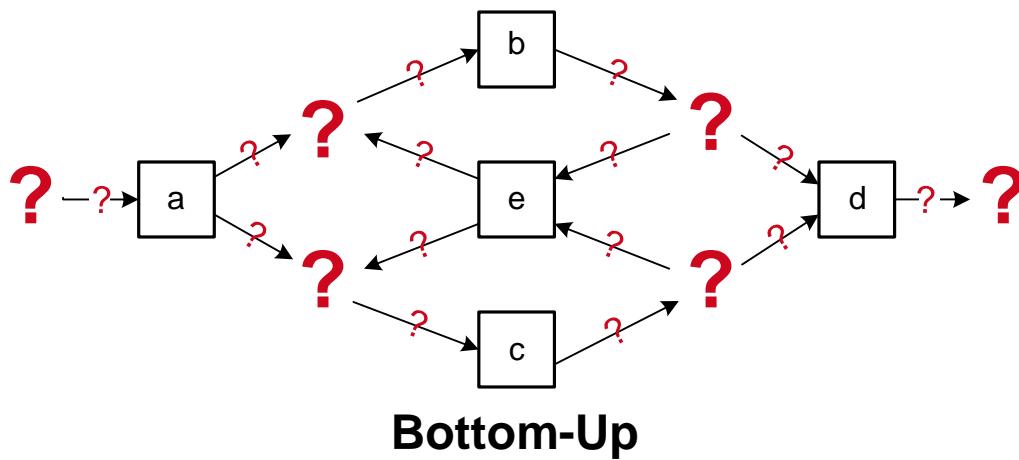
A process model describes a (possibly infinite) set of traces.



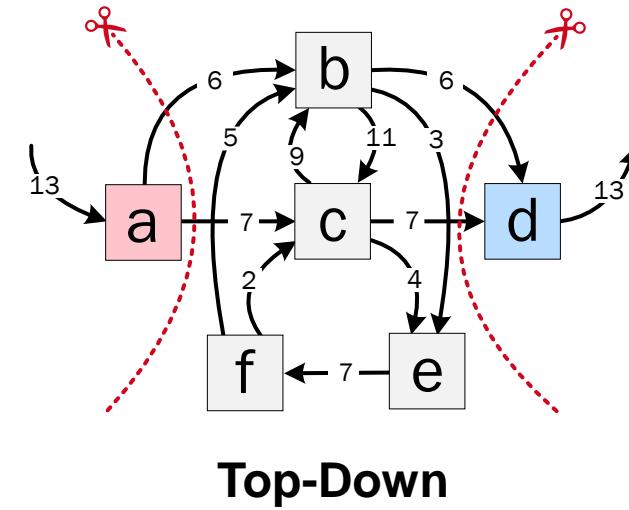
A simplified event log is just a multiset of traces, and each trace is a sequence of activities.

$$\text{Log} = [\langle a, b, c, d \rangle^3, \langle a, c, b, d \rangle^4, \langle a, b, c, e, f, b, c, d \rangle^2, \langle a, b, c, e, f, c, b, d \rangle, \langle a, c, b, e, f, b, c, d \rangle^2, \langle a, c, b, e, f, b, c, e, f, c, b, d \rangle]$$

Process Discovery Approaches



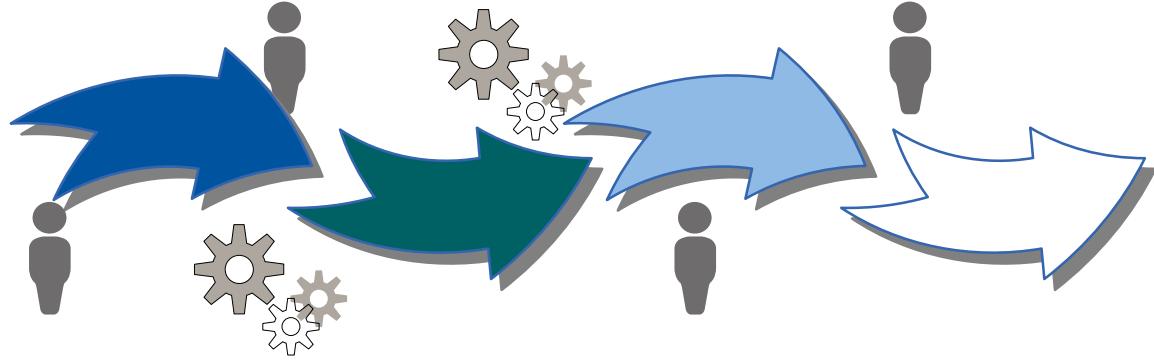
Bottom-Up



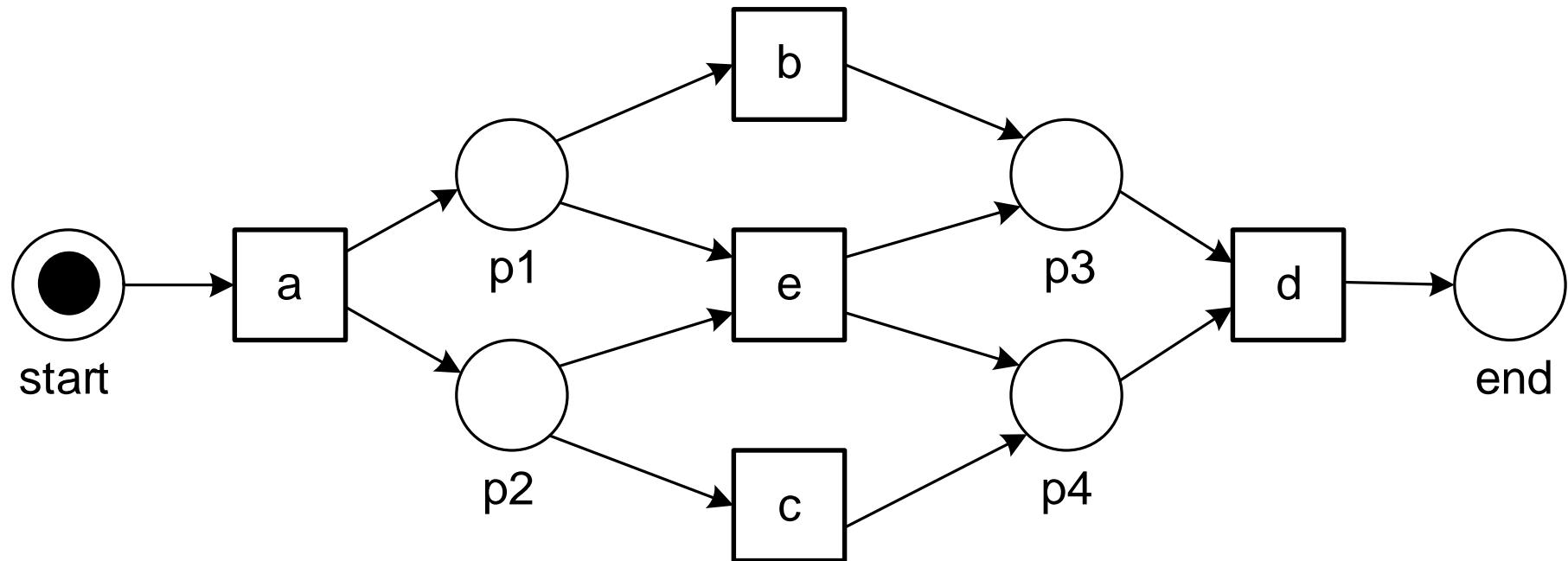
Top-Down

Unsupervised Process Mining

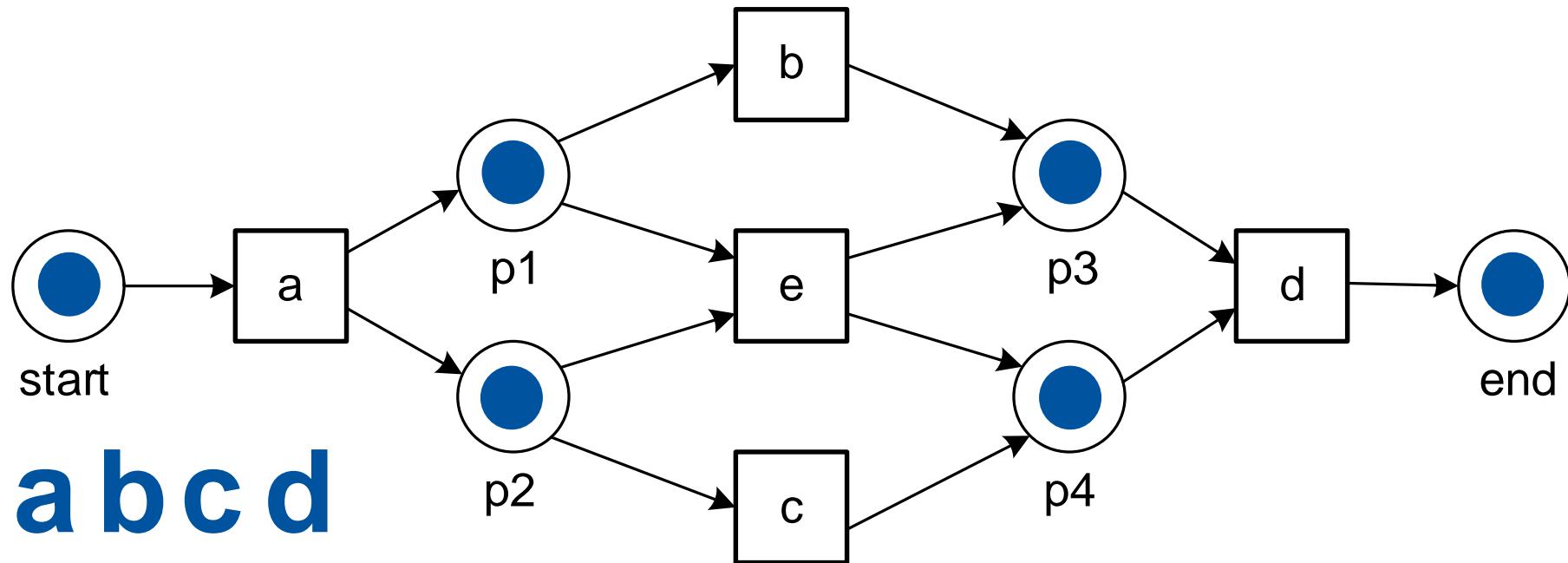
1. Process Discovery
2. **Bottom-Up Discovery**
3. Top-Down Discovery



Explaining Bottom-Up Approach Using Accepting Petri Nets

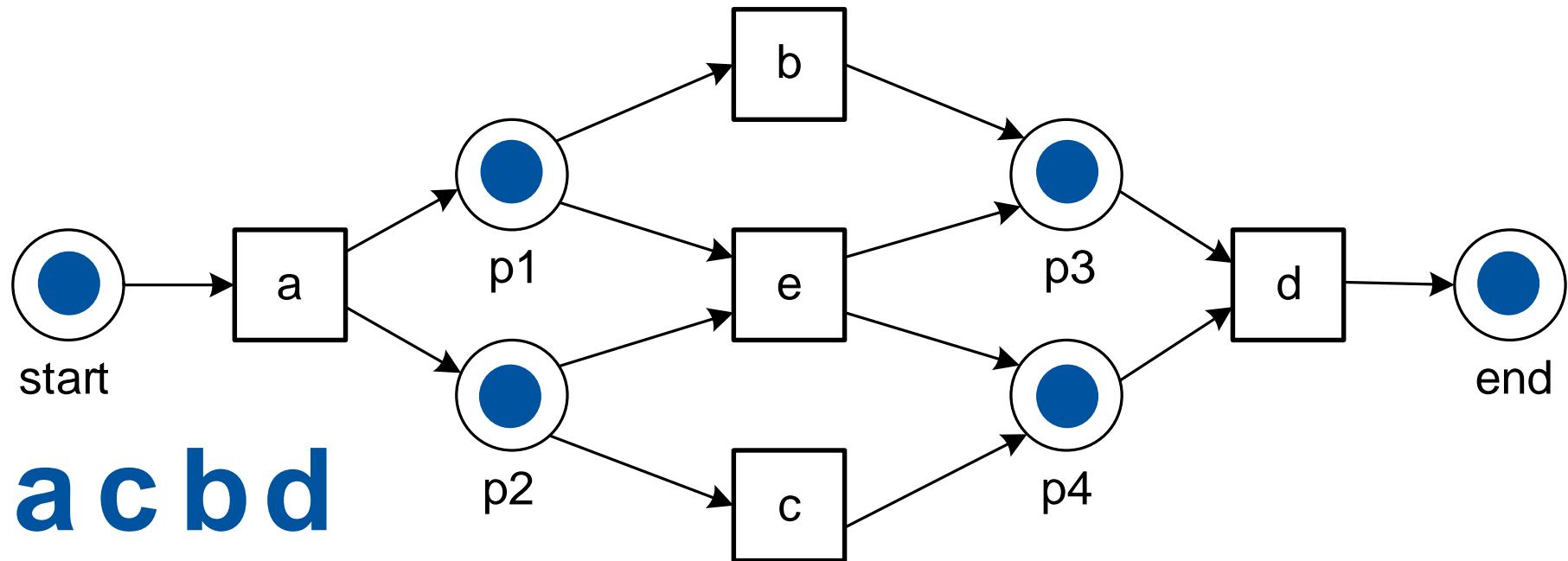


Example Trace (1/3)



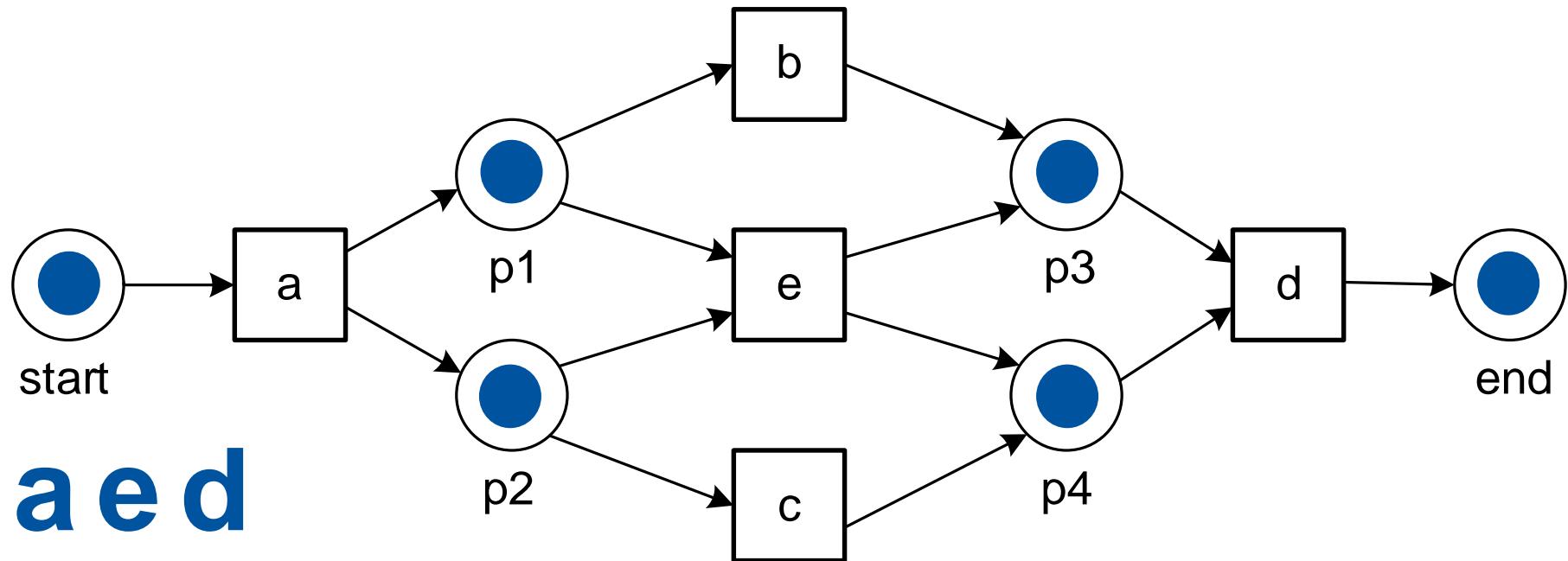
Initial marking [start], final marking [end]

Example Trace (2/3)



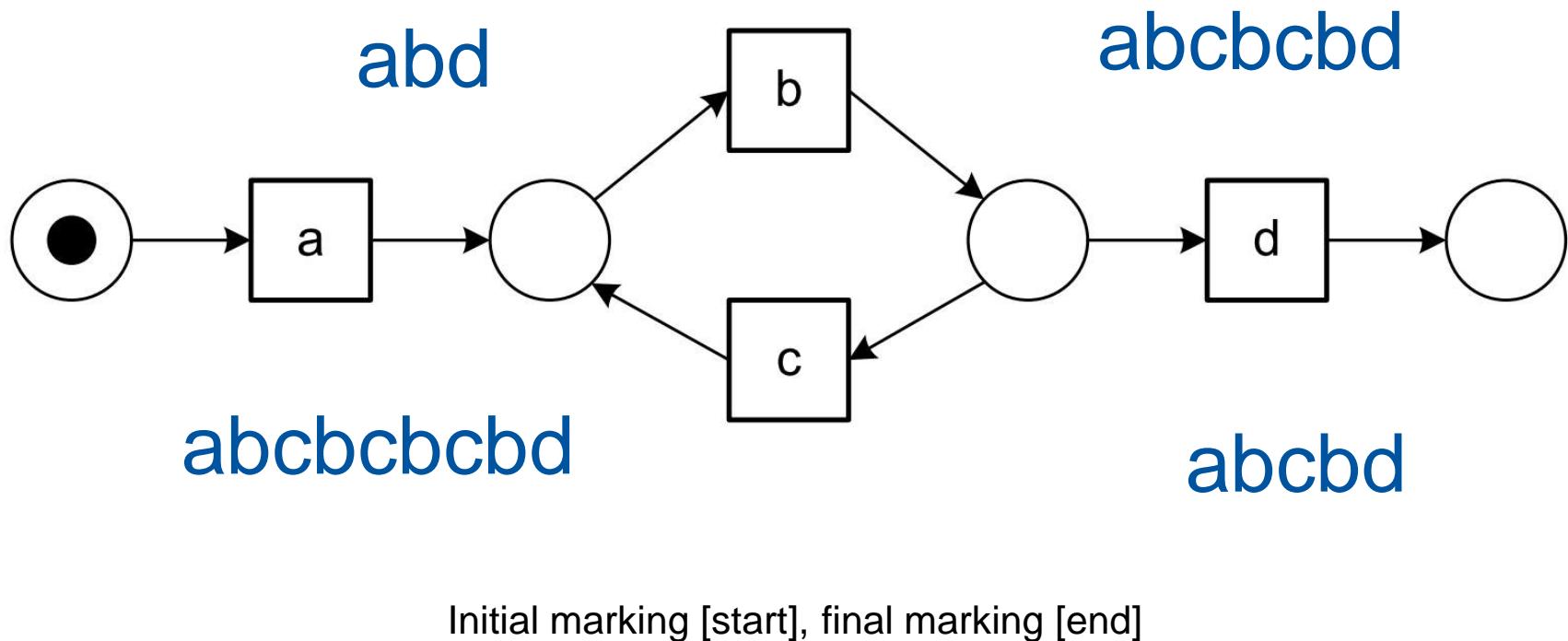
Initial marking [start], final marking [end]

Example Trace (3/3)



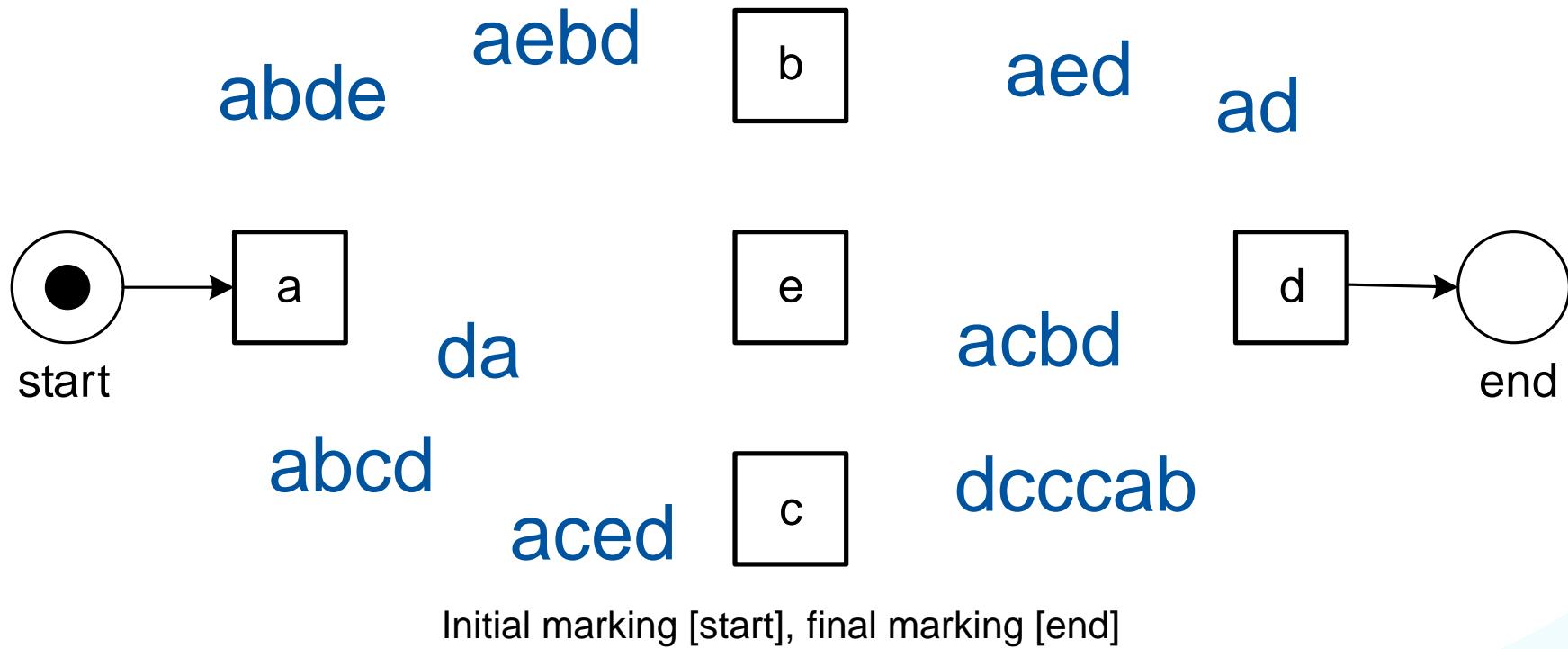
Another Example – Loop

Infinitely many possible traces



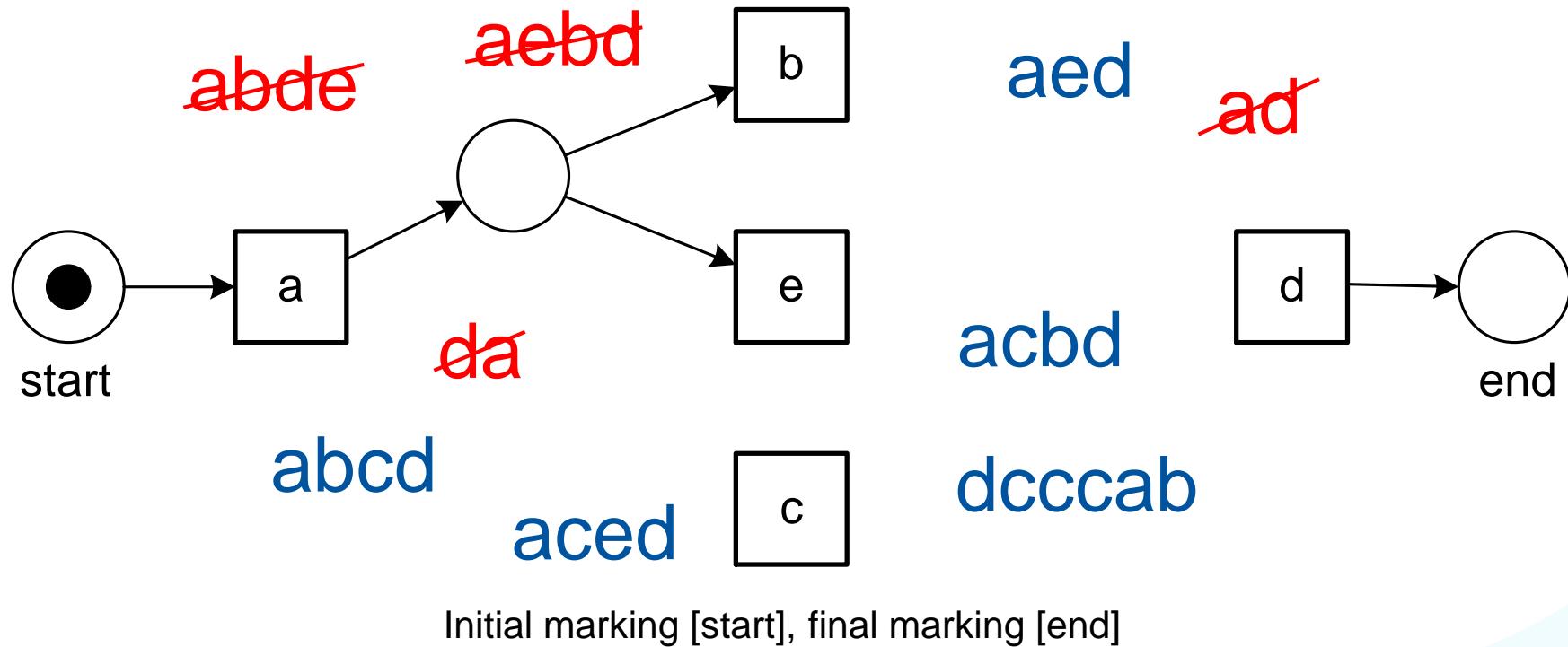
Places are Constraints

- Places cannot have ‘negative tokens’
- Must have the correct number of tokens in the end (indicated by final marking)

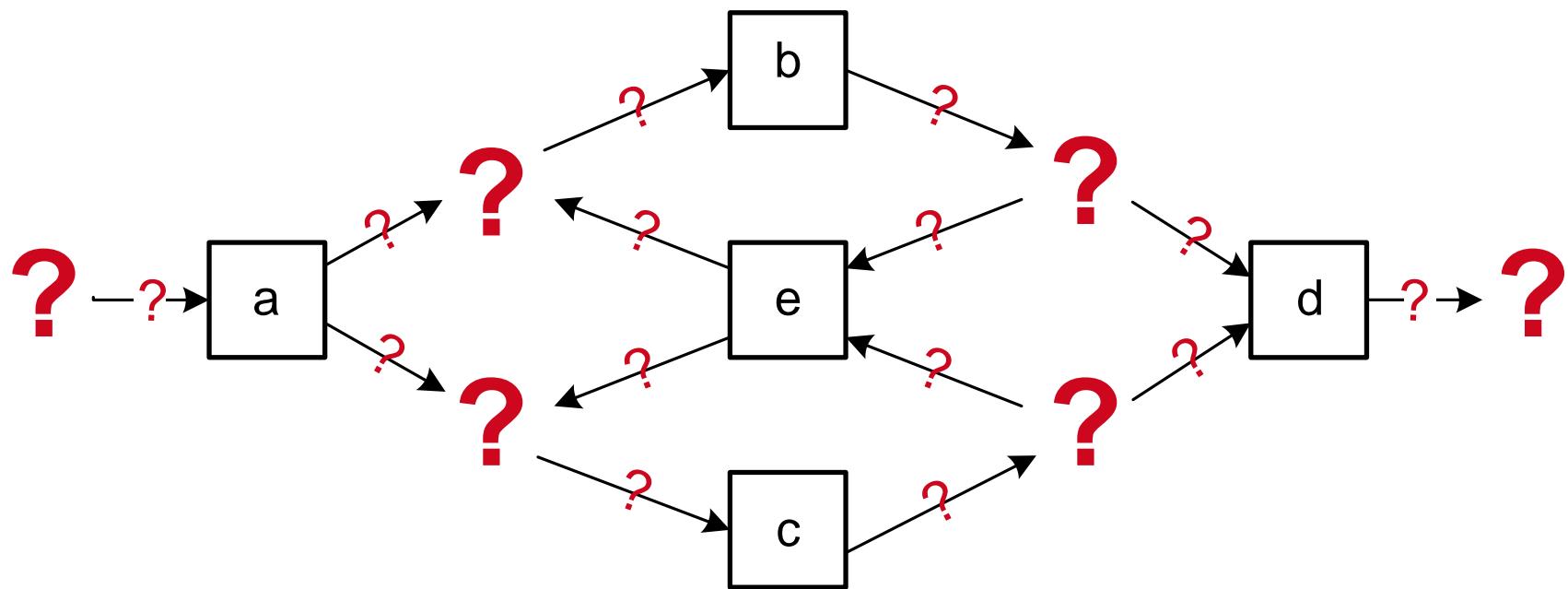


Places are Constraints

- Places cannot have ‘negative tokens’
- Must have the correct number of tokens in the end (indicated by final marking)



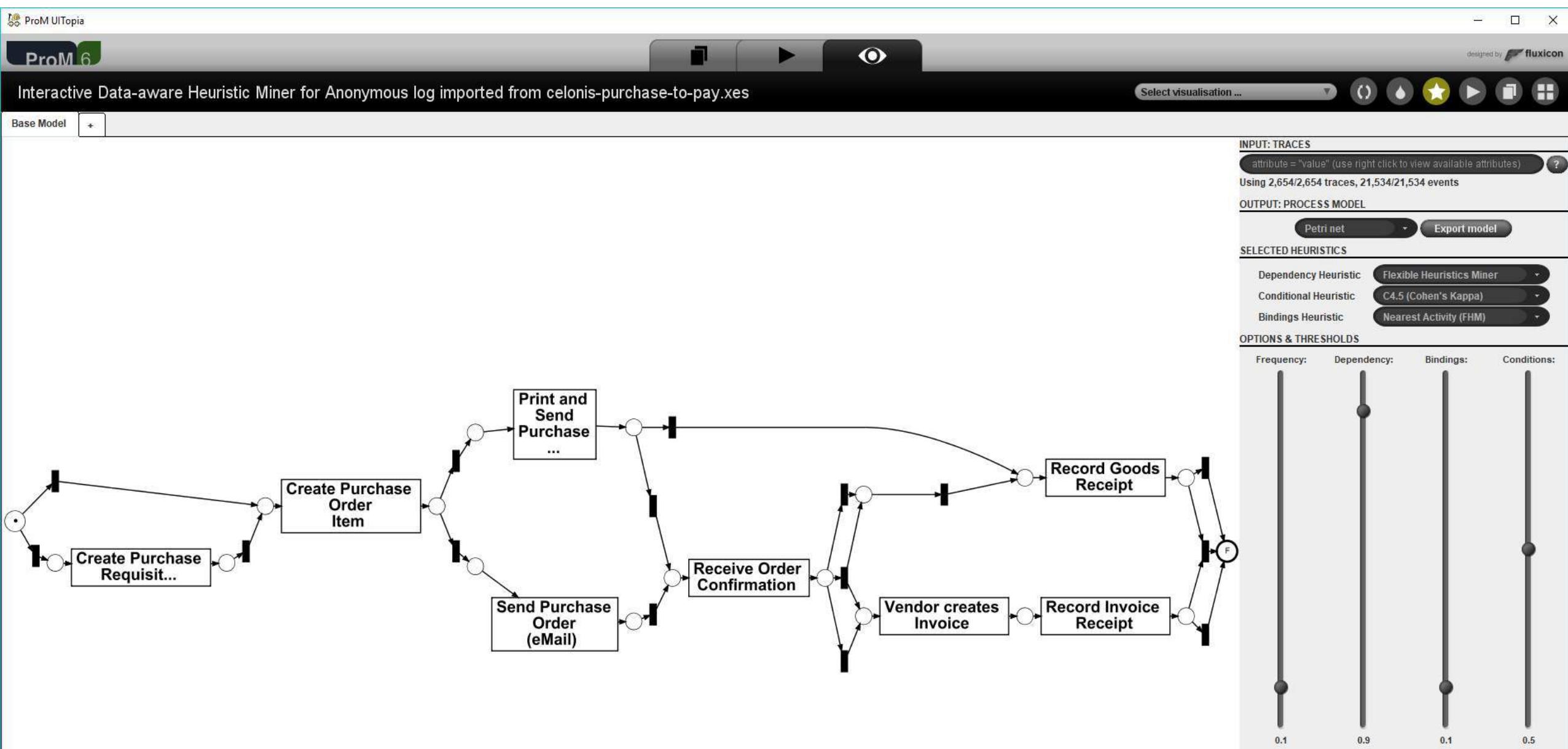
Process Discovery – Finding Places



Many Approaches Possible

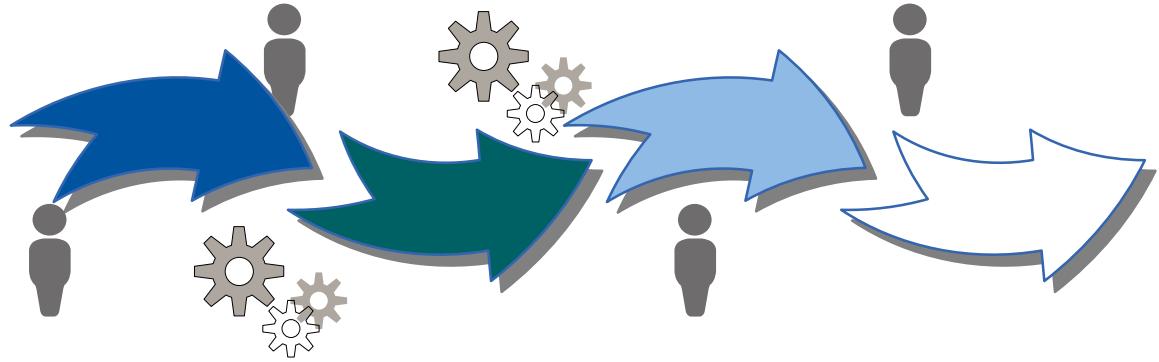
- Heuristics that provide only guarantees for limited classes of models (e.g., Alpha algorithm and heuristic miner)
- Approaches that formally guarantee perfect replayability of the event log (e.g., state-based regions)
- Genetic and other evolutionary approaches (very flexible)
- Optimization-based approaches that turn discovery into an optimization problem (e.g., ILP miner)
- Brute-force approaches that exploit monotonicity properties (apriori-style algorithms)

Example – Heuristic Miner Applied to SAP Data



Unsupervised Process Mining

1. Process Discovery
2. Bottom-Up Discovery
3. **Top-Down Discovery**



Example Top-Down Algorithm Approach: Inductive Mining

- Based on work done by Sander Leemans, Dirk Fahland, and Wil van der Aalst
- Family of approaches with different guarantees and scalability characteristics
(all can ensure replayability of the whole event log)

$$L_3 = [\langle a, b, c, d, e, f, b, d, c, e, g \rangle, \langle a, b, d, c, e, g \rangle^2 \\ \langle a, b, c, d, e, f, b, c, d, e, f, b, d, c, e, g \rangle]$$

$$L_4 = [\langle a, c, d \rangle^{45}, \langle b, c, d \rangle^{42}, \langle a, c, e \rangle^{38}, \langle b, c, e \rangle^{22}]$$

$$L_5 = [\langle a, b, e, f \rangle^2, \langle a, b, e, c, d, b, f \rangle^3, \langle a, b, c, e, d, b, f \rangle^2, \langle a, b, c, d, e, b, f \rangle^4, \\ \langle a, e, b, c, d, b, f \rangle^3]$$

$$L_6 = [\langle a, c, e, g \rangle^2, \langle a, e, c, g \rangle^3, \langle b, d, f, g \rangle^2, \langle b, f, d, g \rangle^4]$$

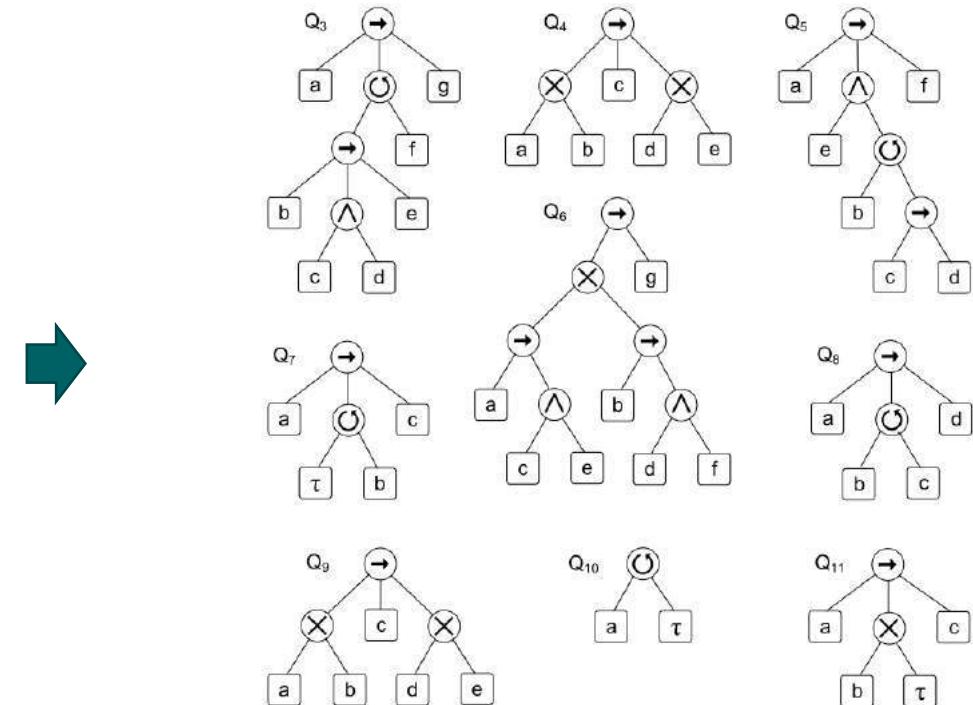
$$L_7 = [\langle a, c \rangle^2, \langle a, b, c \rangle^3, \langle a, b, b, c \rangle^2, \langle a, b, b, b, b, c \rangle]$$

$$L_8 = [\langle a, b, d \rangle^3, \langle a, b, c, b, d \rangle^2, \langle a, b, c, b, c, b, d \rangle]$$

$$L_9 = [\langle a, c, d \rangle^{45}, \langle b, c, e \rangle^{42}]$$

$$L_{10} = [\langle a, a \rangle^{55}]$$

$$L_{11} = [\langle a, b, c \rangle^{20}, \langle a, c \rangle^{30}]$$



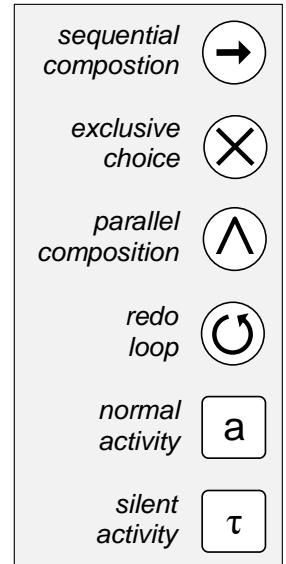
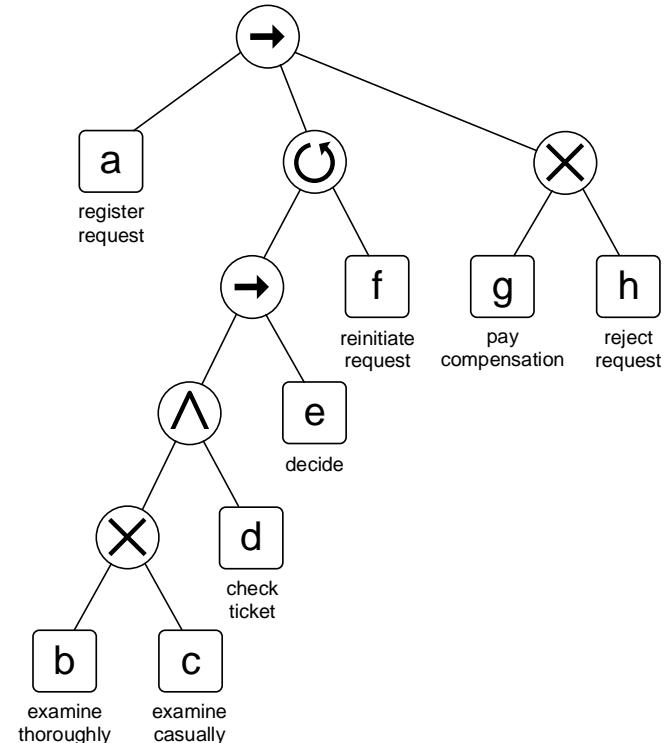
Inductive Mining

$$L = [\langle a, b, d, e, h \rangle^3, \\ \langle a, d, c, e, g \rangle^4, \\ \langle a, c, d, e, f, b, d, e, g \rangle^2, \\ \langle a, d, b, e, h \rangle^2, \\ \langle a, c, d, e, f, d, c, e, f, c, d, e, h \rangle^2, \\ \langle a, c, d, e, g \rangle]$$

Input – simplified event log

Inductive Mining

$L = [\langle a, b, d, e, h \rangle^3,$
 $\langle a, d, c, e, g \rangle^4,$
 $\langle a, c, d, e, f, b, d, e, g \rangle^2,$
 $\langle a, d, b, e, h \rangle^2,$
 $\langle a, c, d, e, f, d, c, e, f, c, d, e, h \rangle^2,$
 $\langle a, c, d, e, g \rangle]$



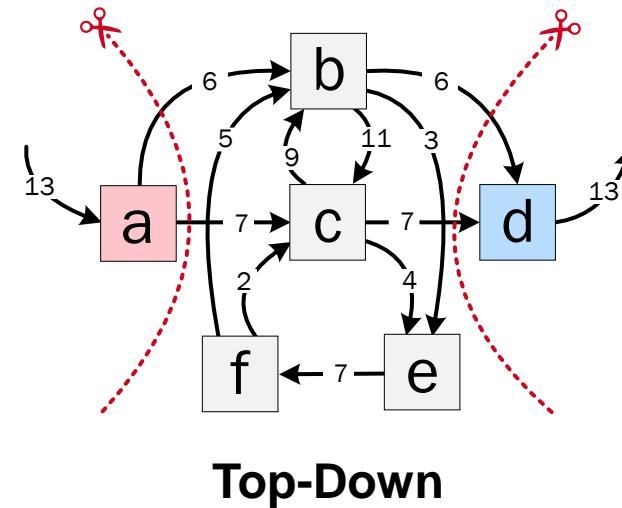
Input – simplified event log

Output – process tree

Inductive Mining in Steps

Apply **recursively** (split into multiple sublogs):

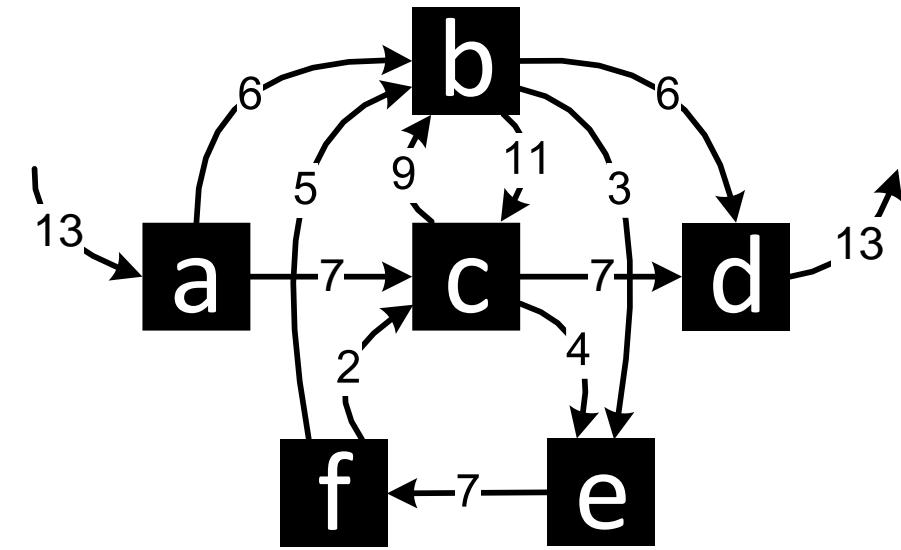
1. Create DFG based on the event log
2. Find a **cut** in the DFG
3. Partition event log based on chosen cut
4. Handle **base cases**
5. Recurse on non-base cases



Applying Inductive Mining Recursively

Step 1 – Create DFG Based On the Event Log

3x	a	b	c	d								
4x	a	c	b	d								
2x	a	b	c	e	f	b	c	d				
2x	a	c	b	e	f	b	c	d				
1x	a	b	c	e	f	c	b	d				
1x	a	c	b	e	f	b	c	e	f	c	b	d

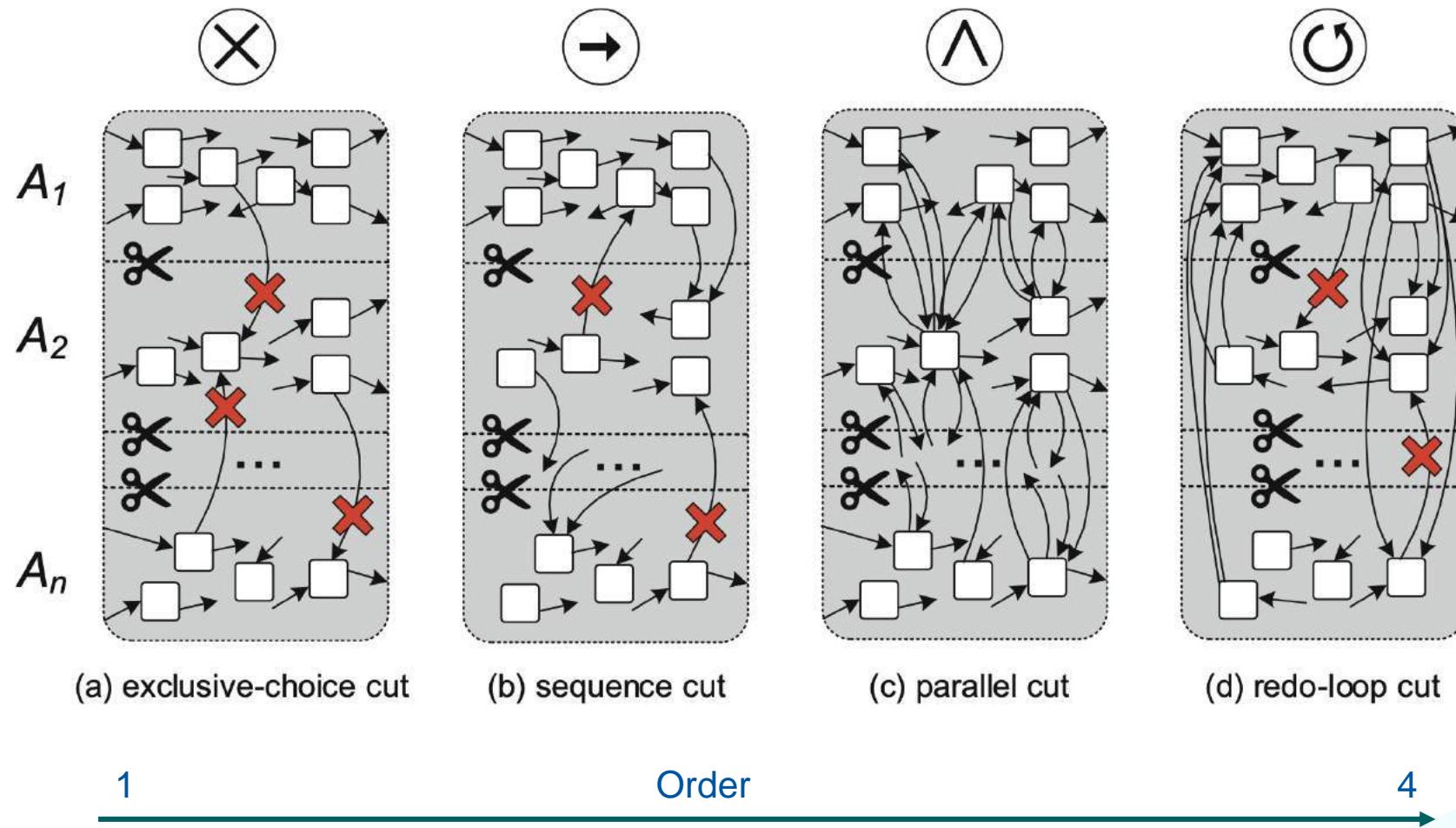


[3]

Input – simplified event log

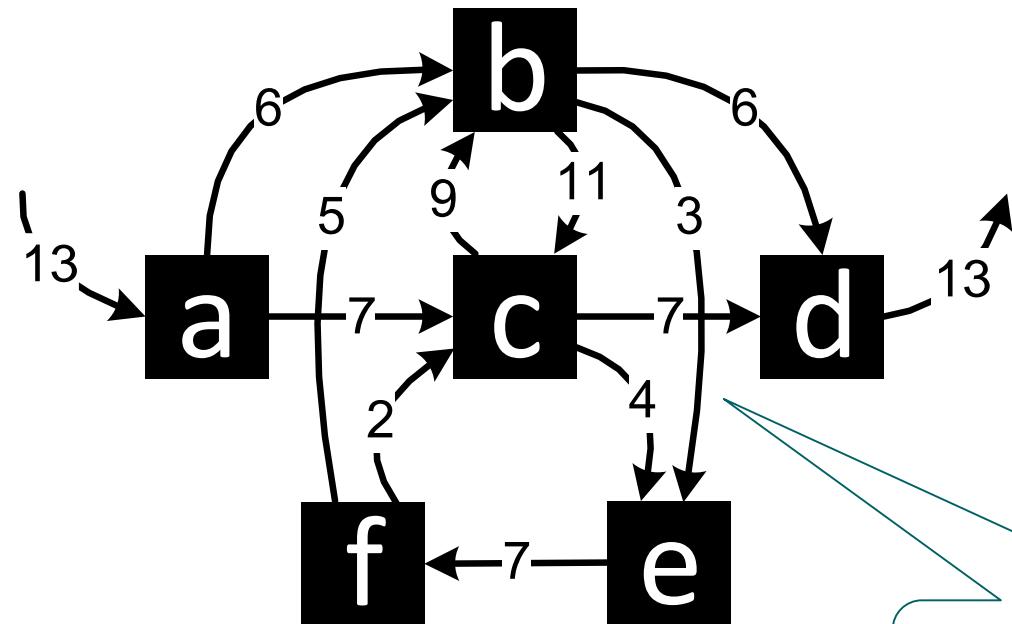
Directly-follows graph

Inductive Mining – Possible Cuts



Applying Inductive Mining Recursively

Step 2 – Choose Cut

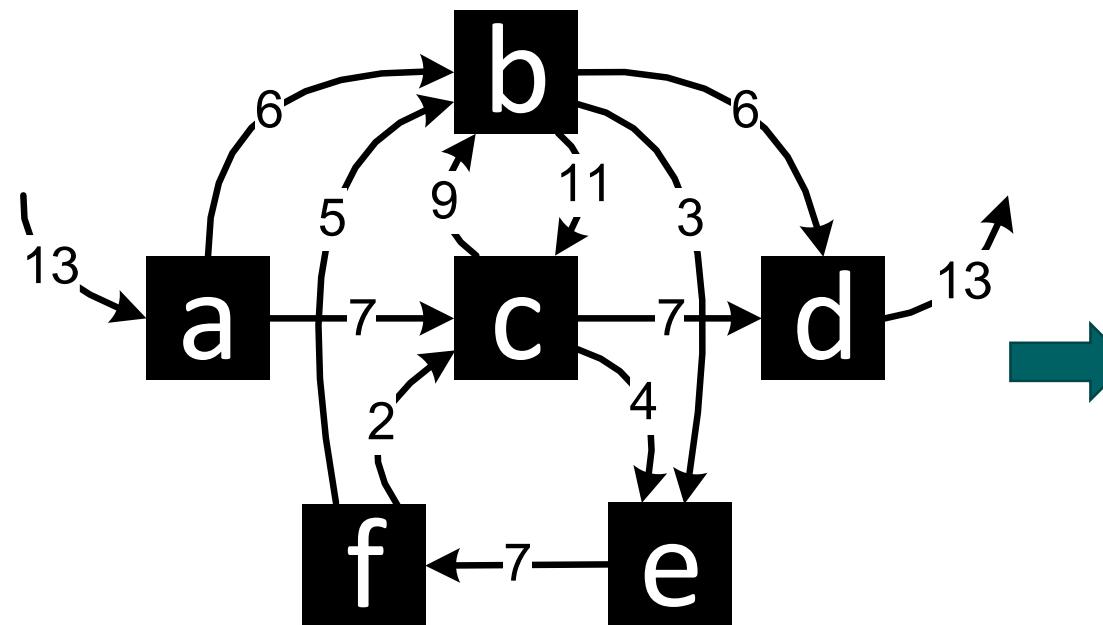


Directly-follows graph

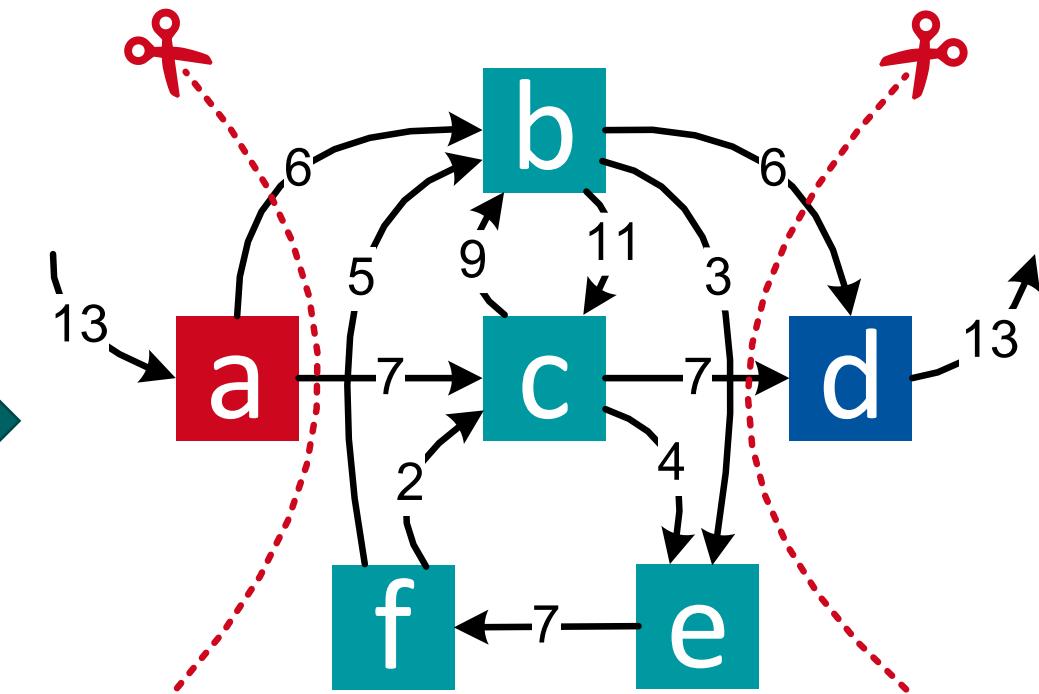
Exclusive choice cut not possible,
but we can apply [sequence cut!](#)

Applying Inductive Mining Recursively

Step 2 – Choose Cut



Directly-follows graph

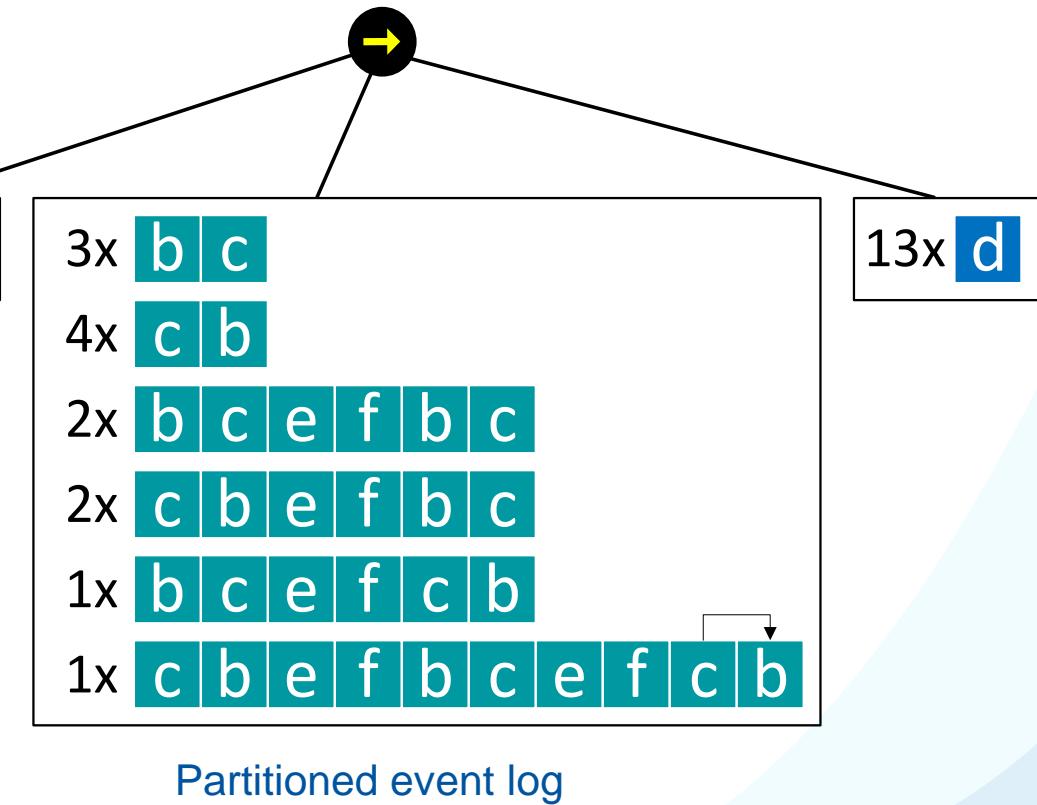
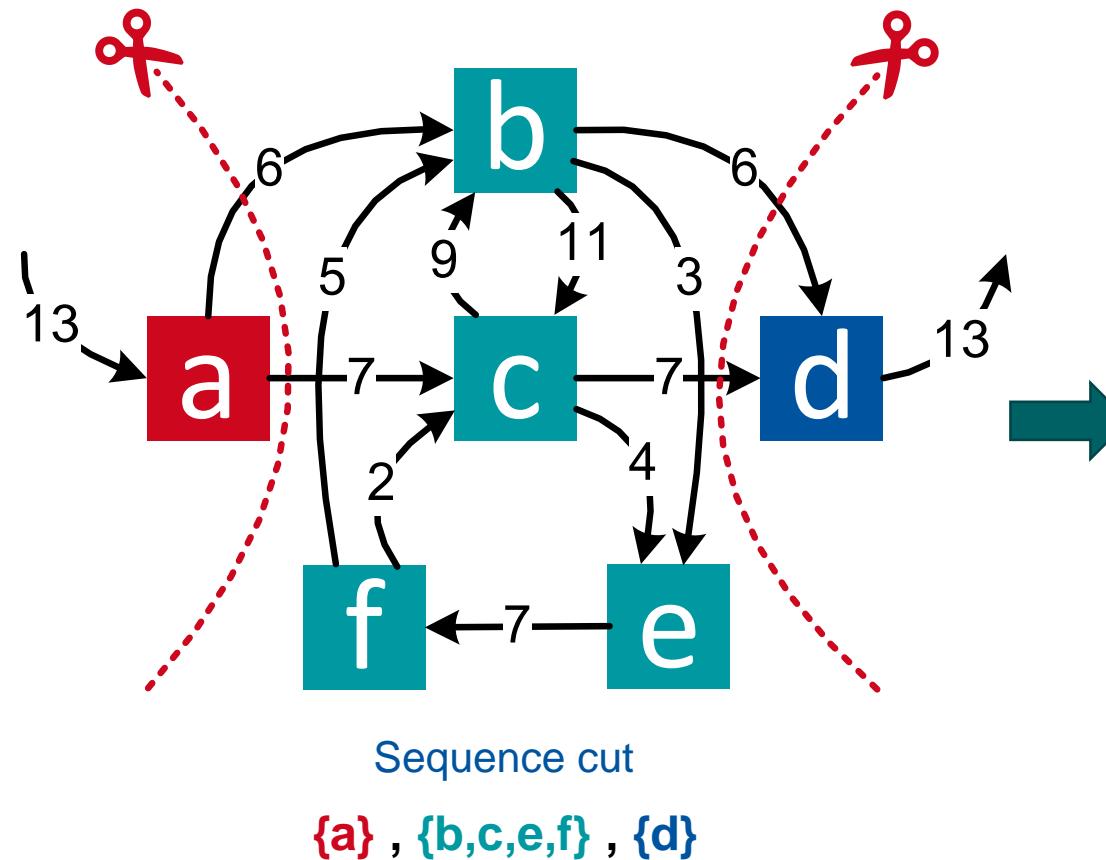


Sequence cut

$\{a\}$, $\{b,c,e,f\}$, $\{d\}$

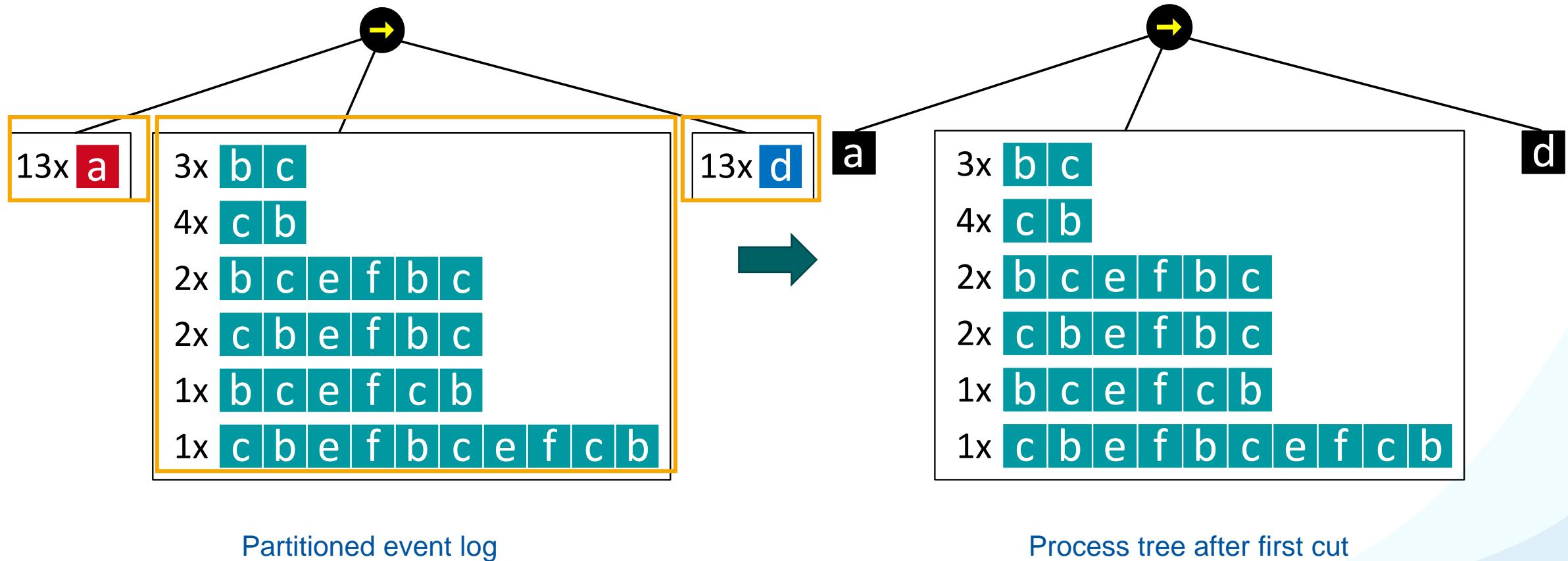
Applying Inductive Mining Recursively

Step 3 – Partition Event Log Based on Chosen Cut



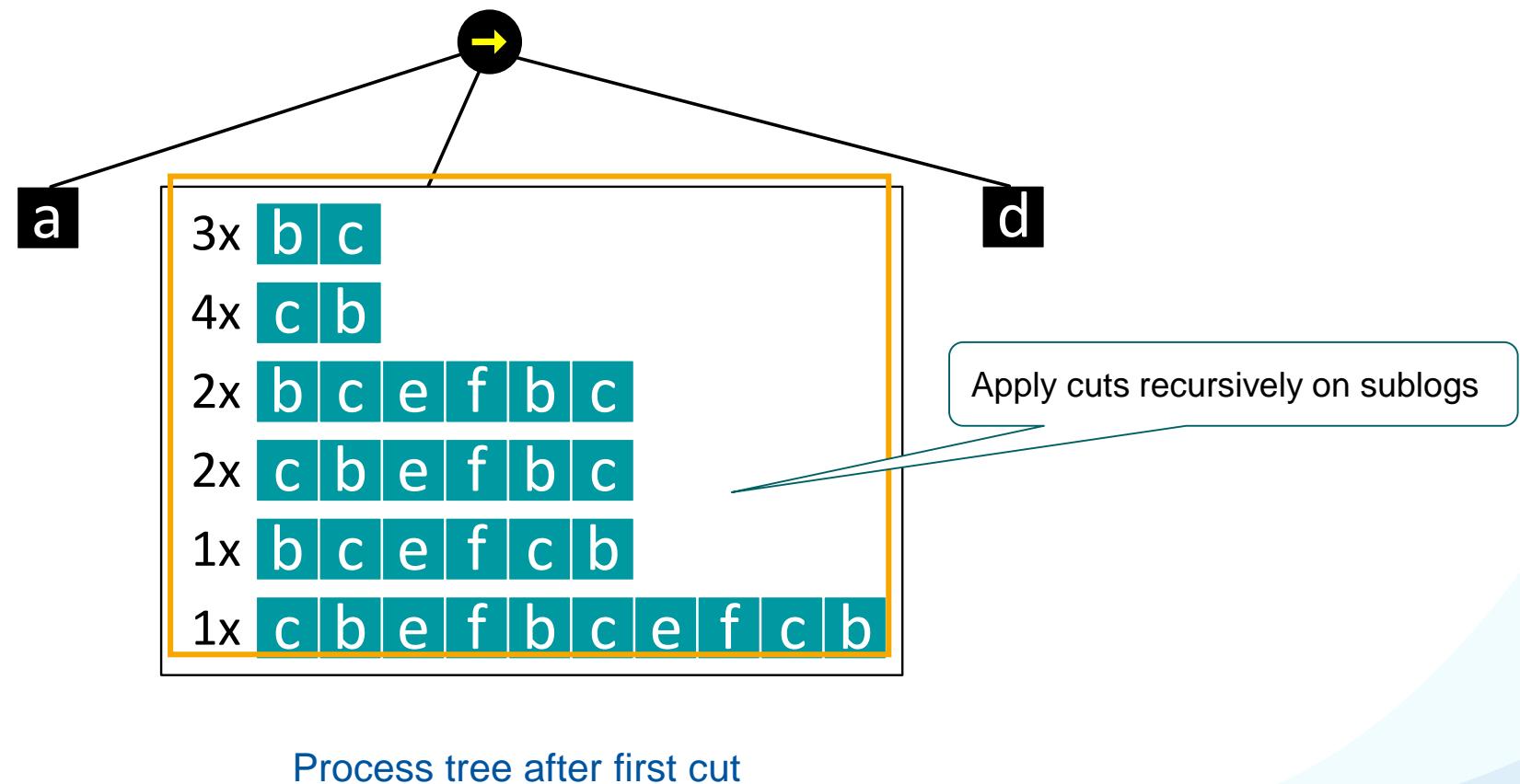
Applying Inductive Mining Recursively

Step 4 – Handle Base Cases



Applying Inductive Mining Recursively

Step 5 – Recurse on Non-Base Cases



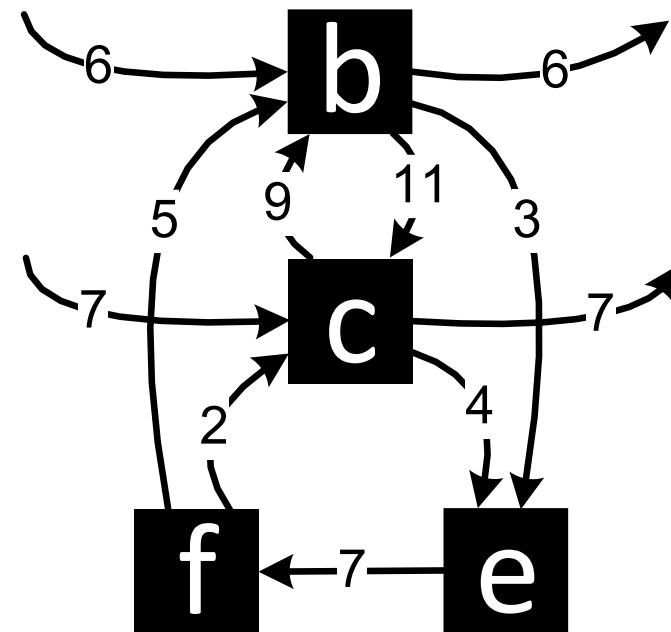
Applying Inductive Mining Recursively

Step 1 – Create DFG Based On the Event Log

3x	b	c								
4x	c	b								
2x	b	c	e	f	b	c				
2x	c	b	e	f	b	c				
1x	b	c	e	f	c	b				
1x	c	b	e	f	b	c	e	f	c	b



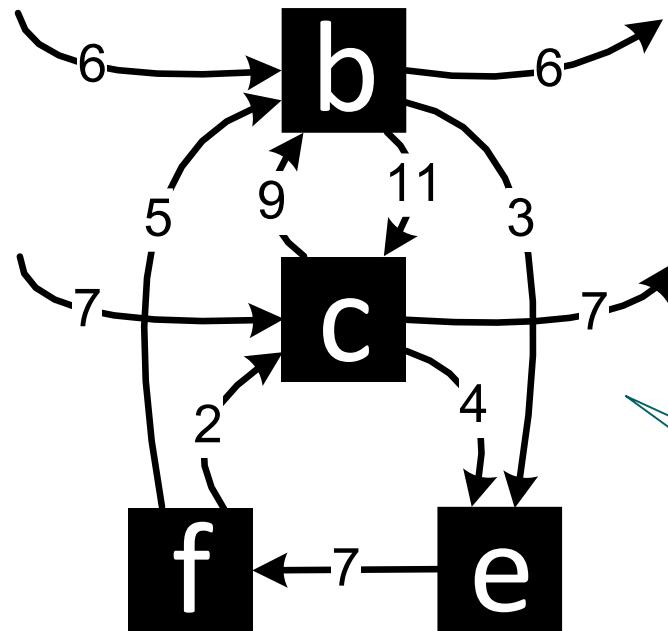
Input – simplified event log



Directly-follows graph

Applying Inductive Mining Recursively

Step 2 – Choose Cut

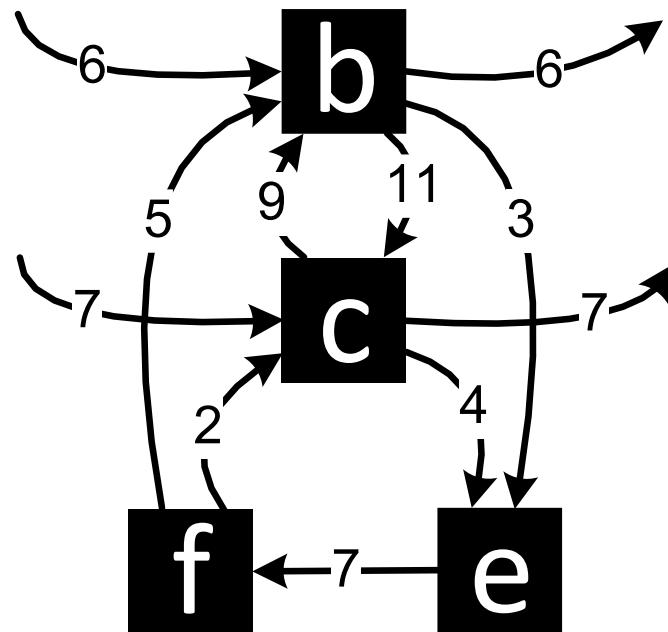


Directly-follows graph

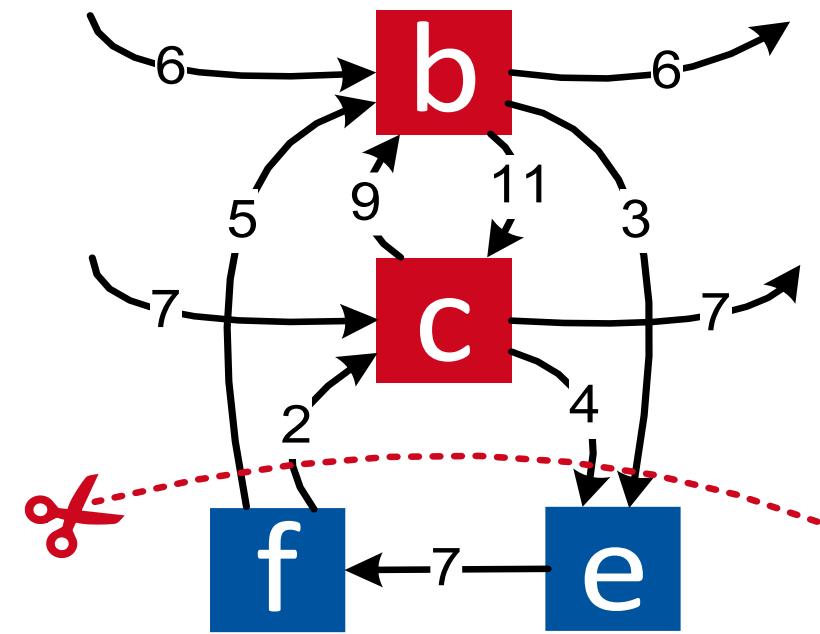
Exclusive choice, sequence cut and parallel cuts not possible, but we can apply [loop cut!](#)

Applying Inductive Mining Recursively

Step 2 – Choose Cut



Directly-follows graph

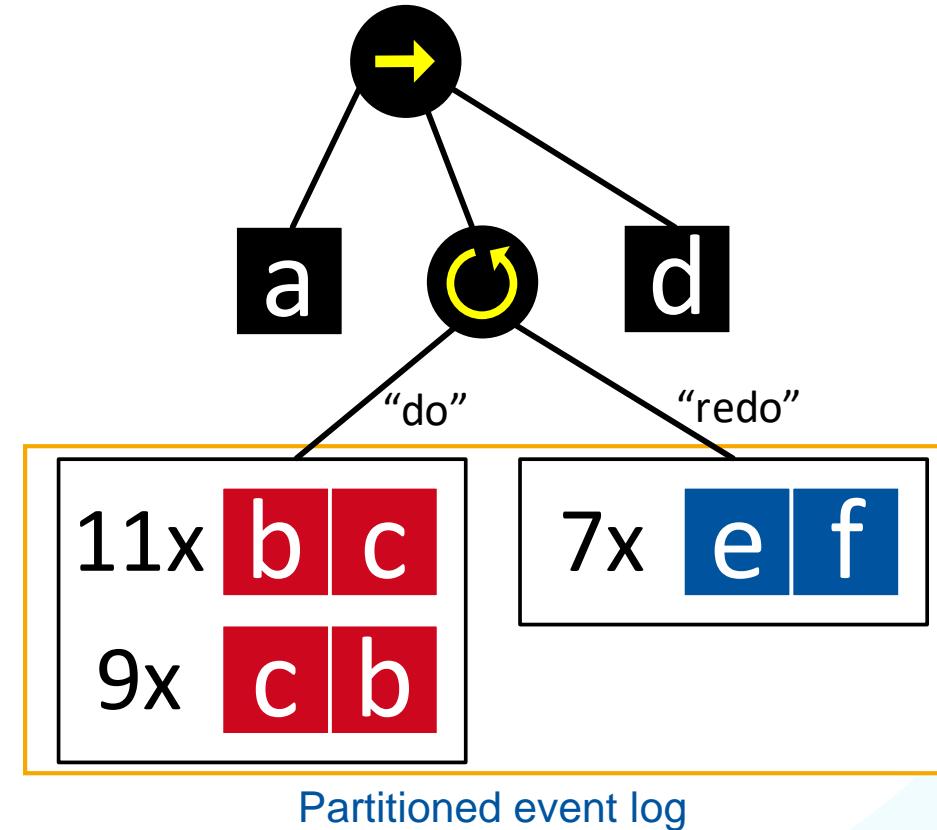
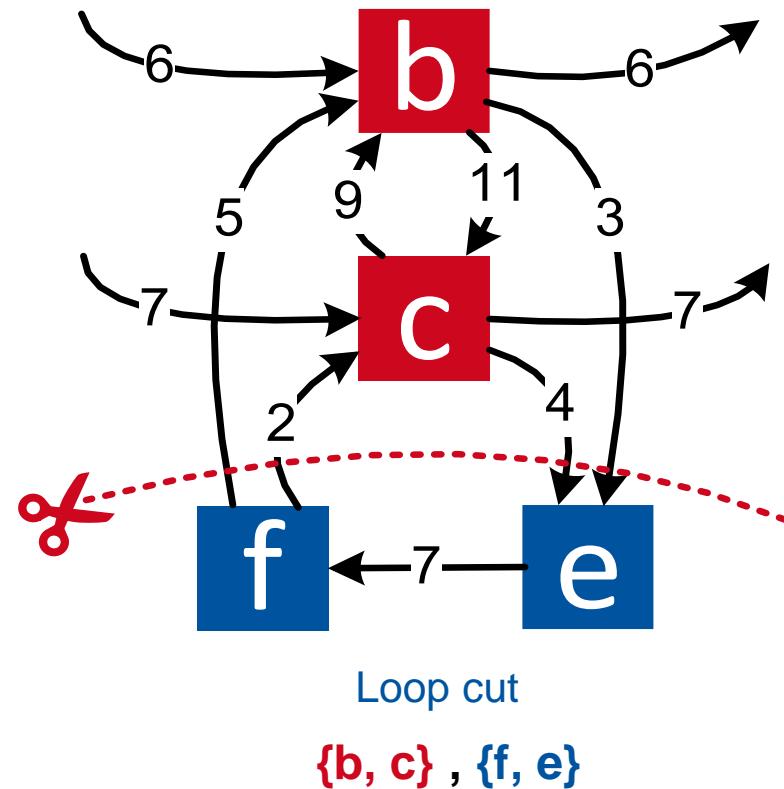


Loop cut

$\{b, c\}$, $\{f, e\}$

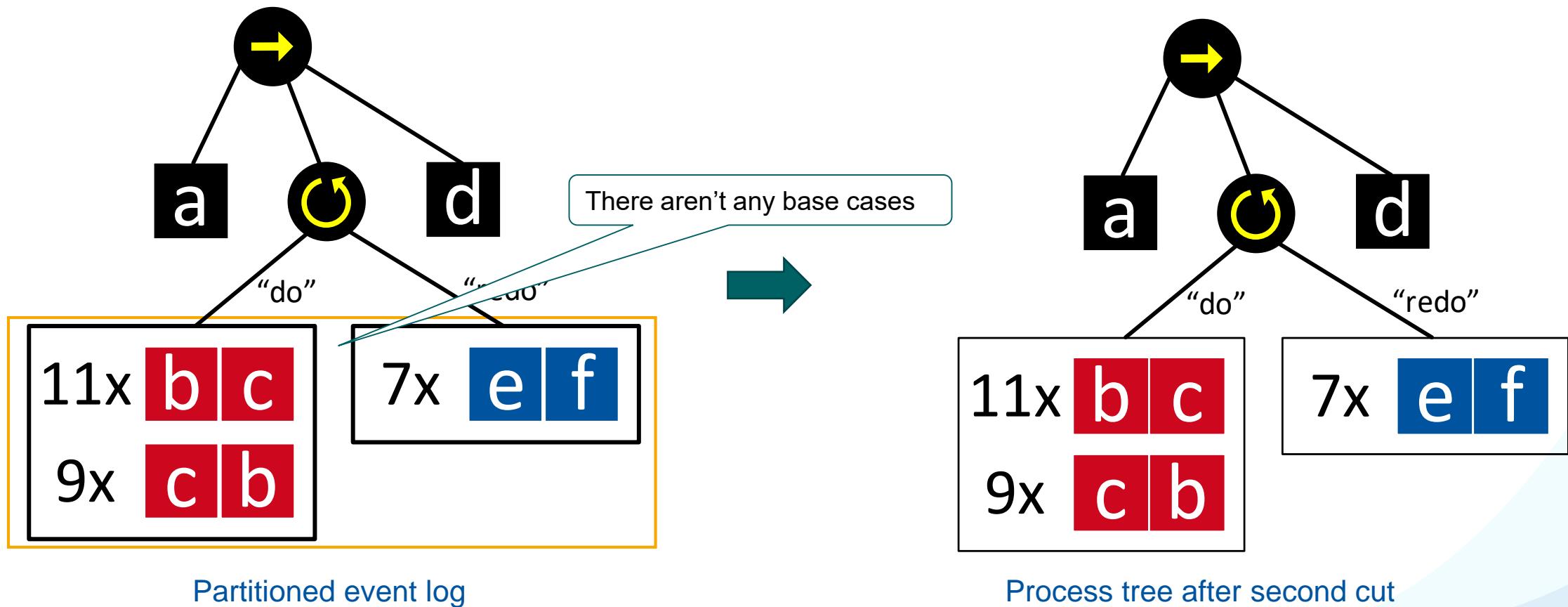
Applying Inductive Mining Recursively

Step 3 – Partition Event Log Based on Chosen Cut



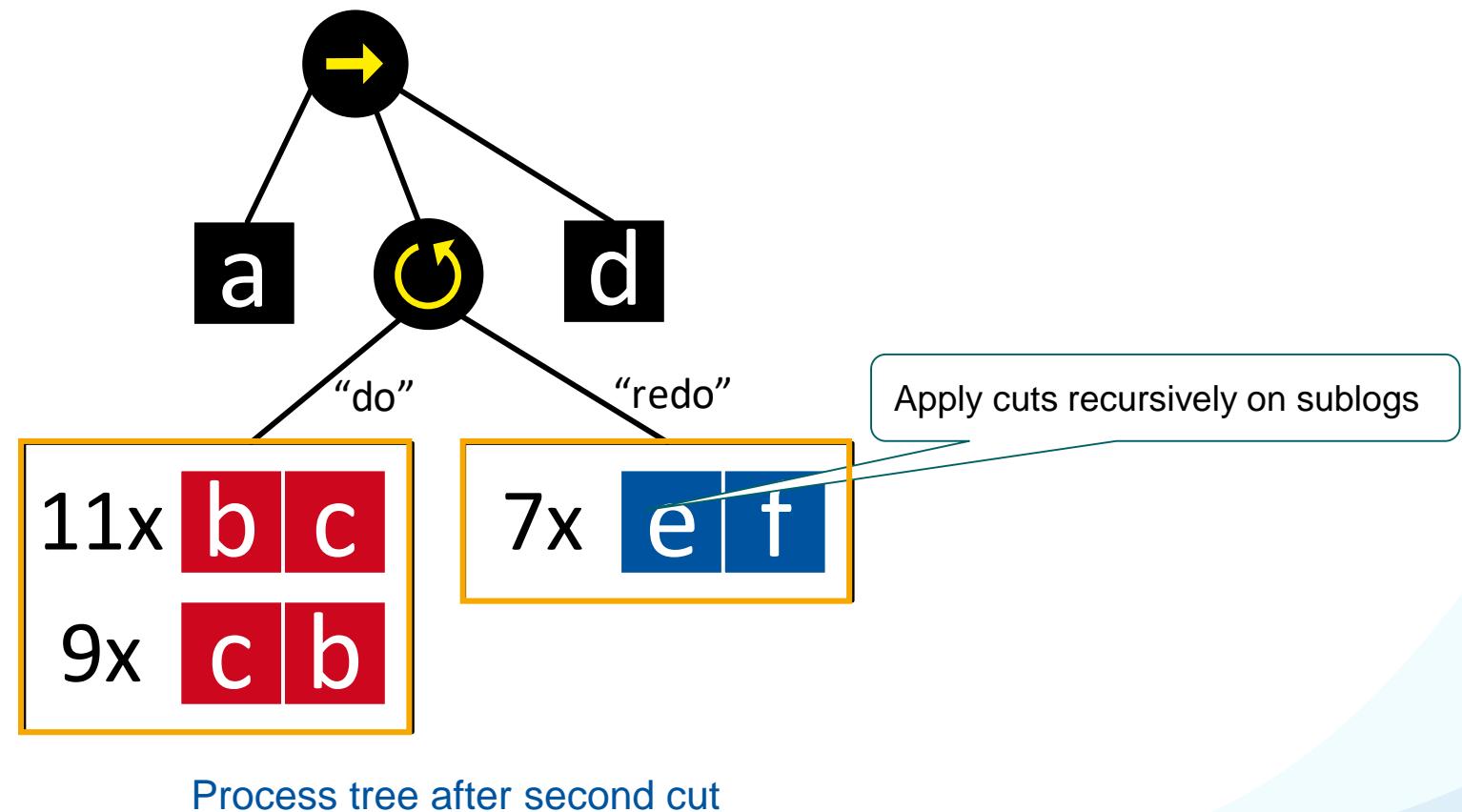
Applying Inductive Mining Recursively

Step 4 – Handle Base Cases



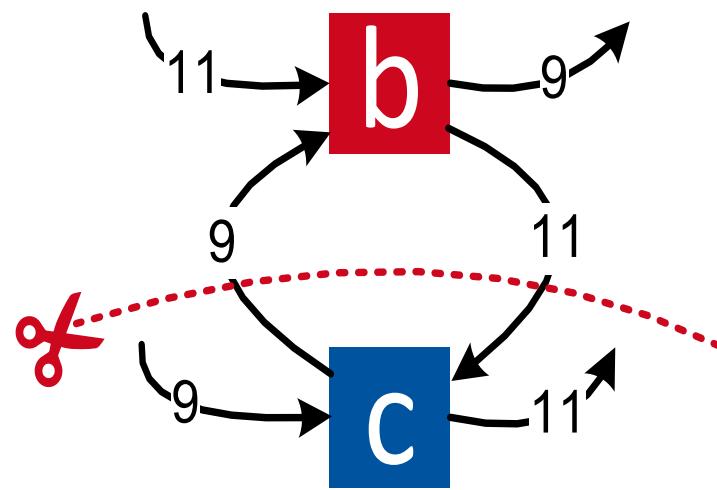
Applying Inductive Mining Recursively

Step 5 – Recurse on Non-Base Cases

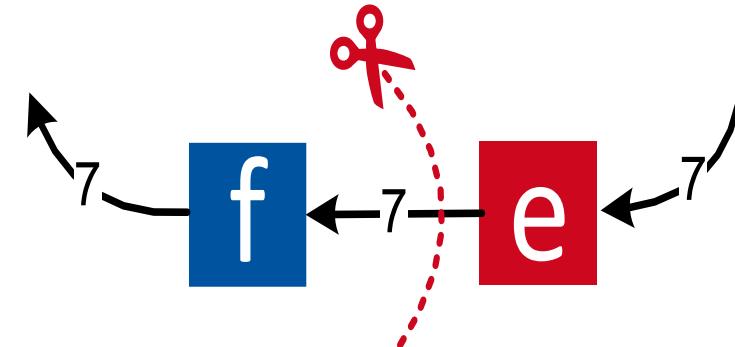


Applying Inductive Mining Recursively

Repeat All These Steps on the Sublogs



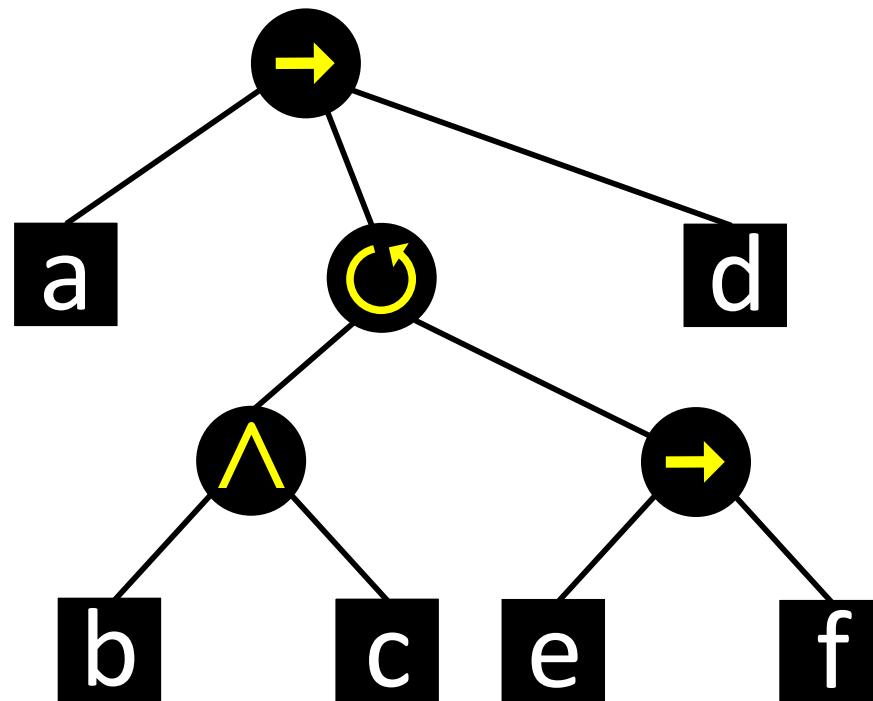
Parallel cut



Sequence cut

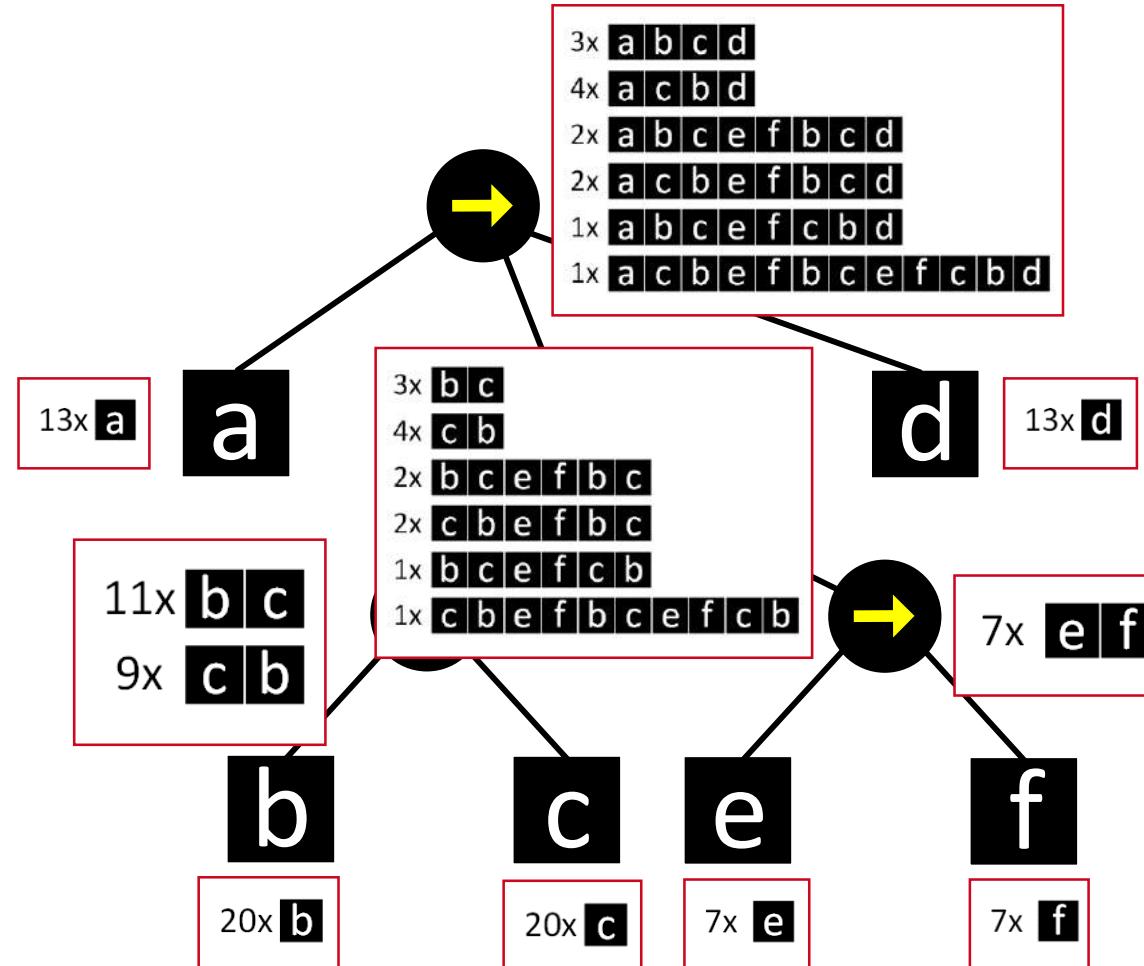
Applying Inductive Mining Recursively

Final Process Tree

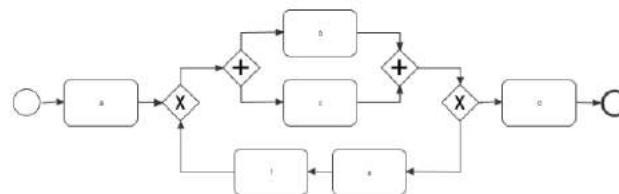
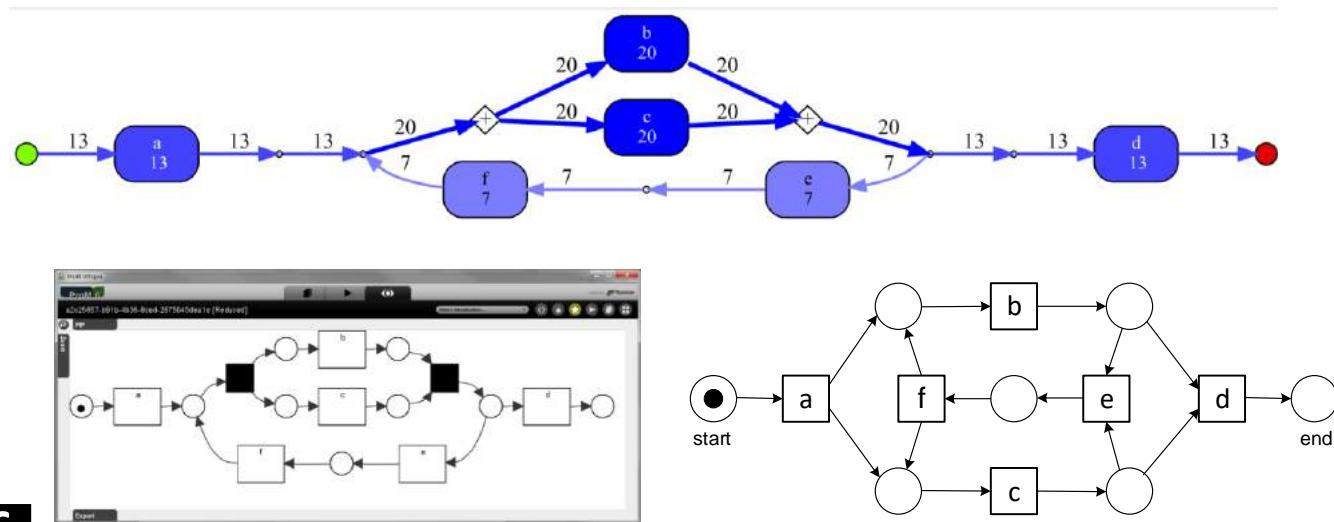
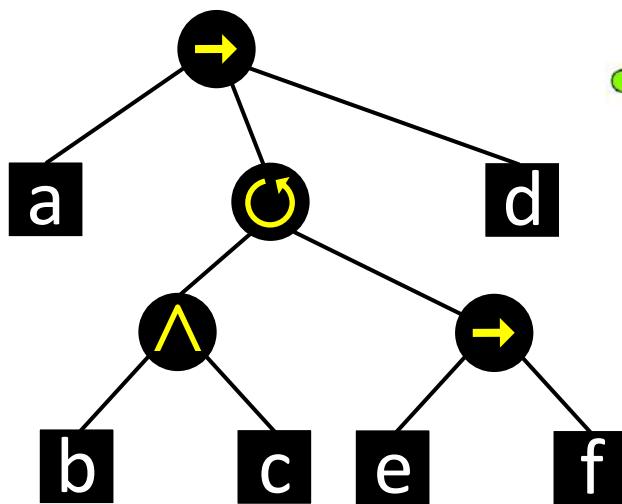


Applying Inductive Mining Recursively

Top-Down Process



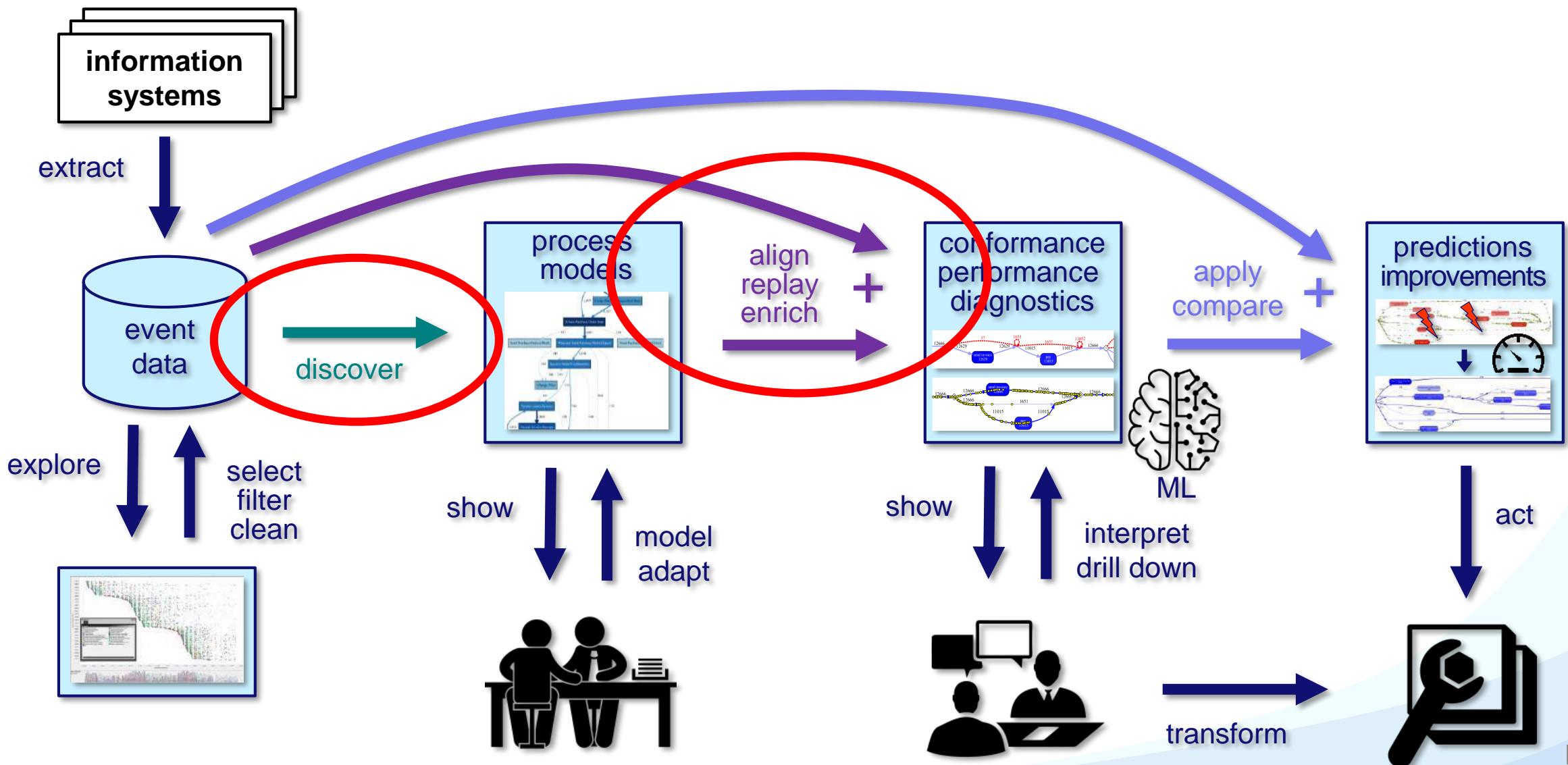
Alternative Notations



Inductive Mining Properties

- Basic algorithm formally **guarantees** that the original event log can be fully replayed
- Models satisfy formal properties that greatly facilitate further analysis (**soundness**)
- If the event log was generated from a basic process tree (no duplication of labels), then an **equivalent model** will be found
- Extensions exist to deal with **infrequent** behavior and **incomplete** event logs
- **Highly scalable** – dealing with billions of events, millions of cases, and thousands of unique activities
- Allows for distribution, streaming, etc.

From Process Discovery to Conformance Checking

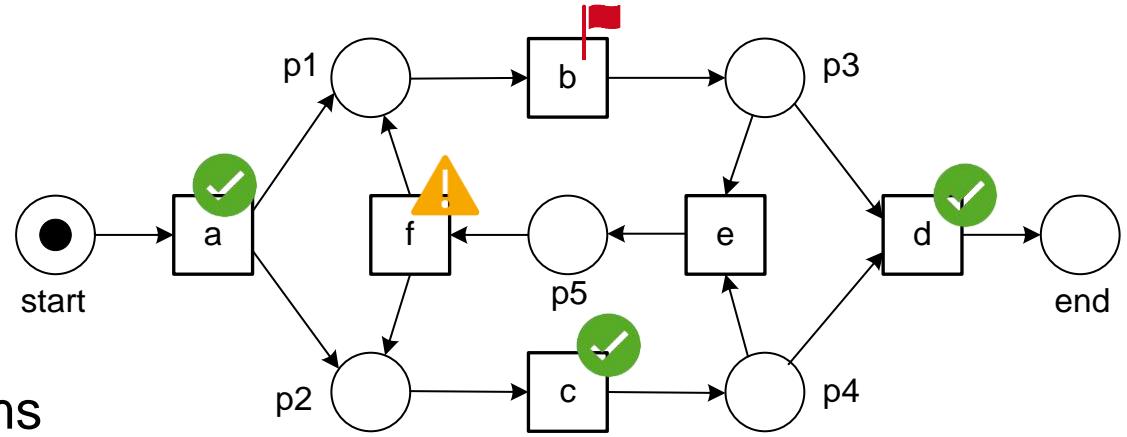


Part III: Supervised Process Mining

Conformance Checking and the Connection to “Mainstream ML”

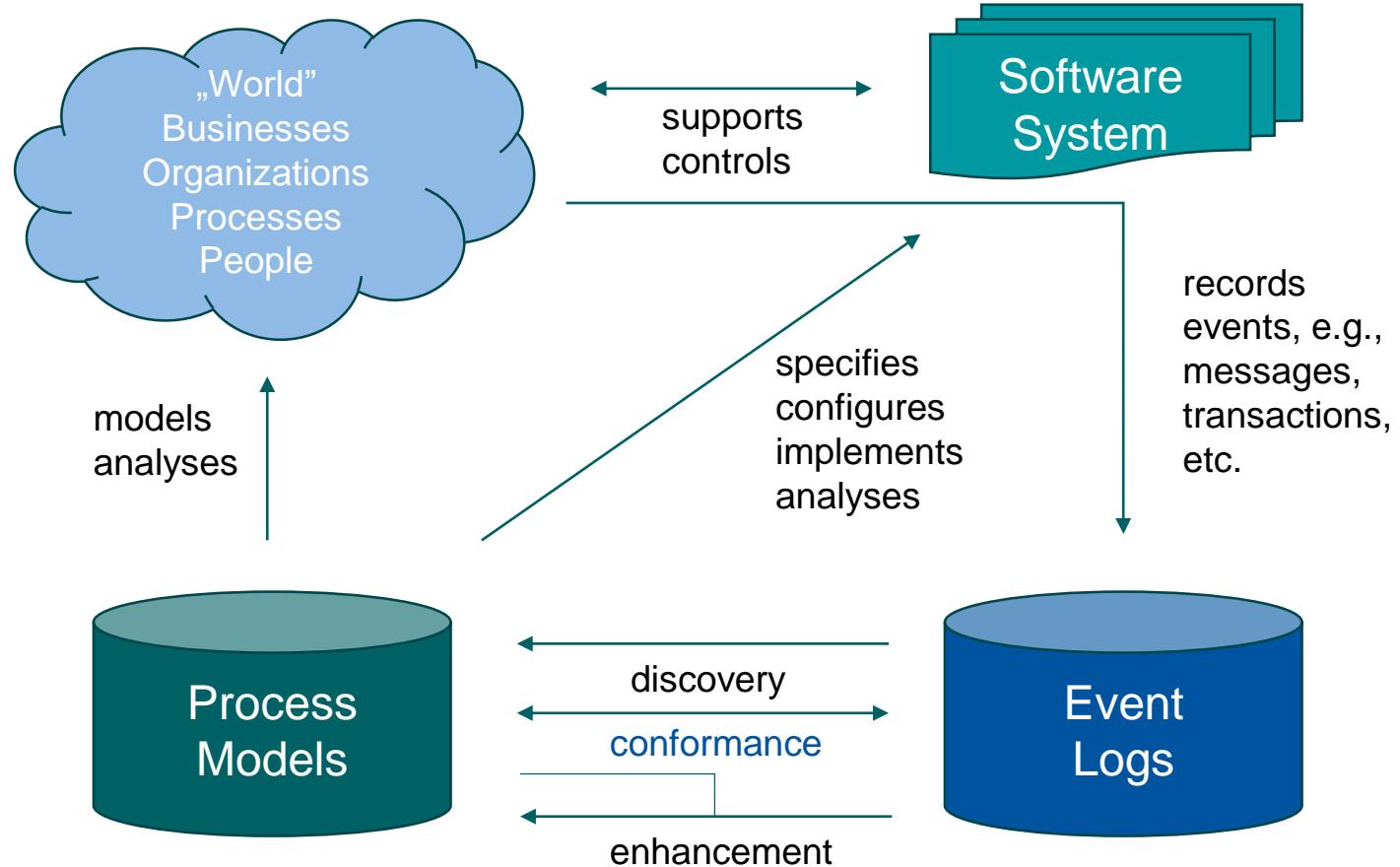
Supervised Process Mining

1. **Token-Based Replay**
2. Token-Based Replay Examples
3. Fitness at the Log Level
4. Generating Supervised Learning Problems



Conformance checking and the link to other data science techniques (e.g., machine learning).

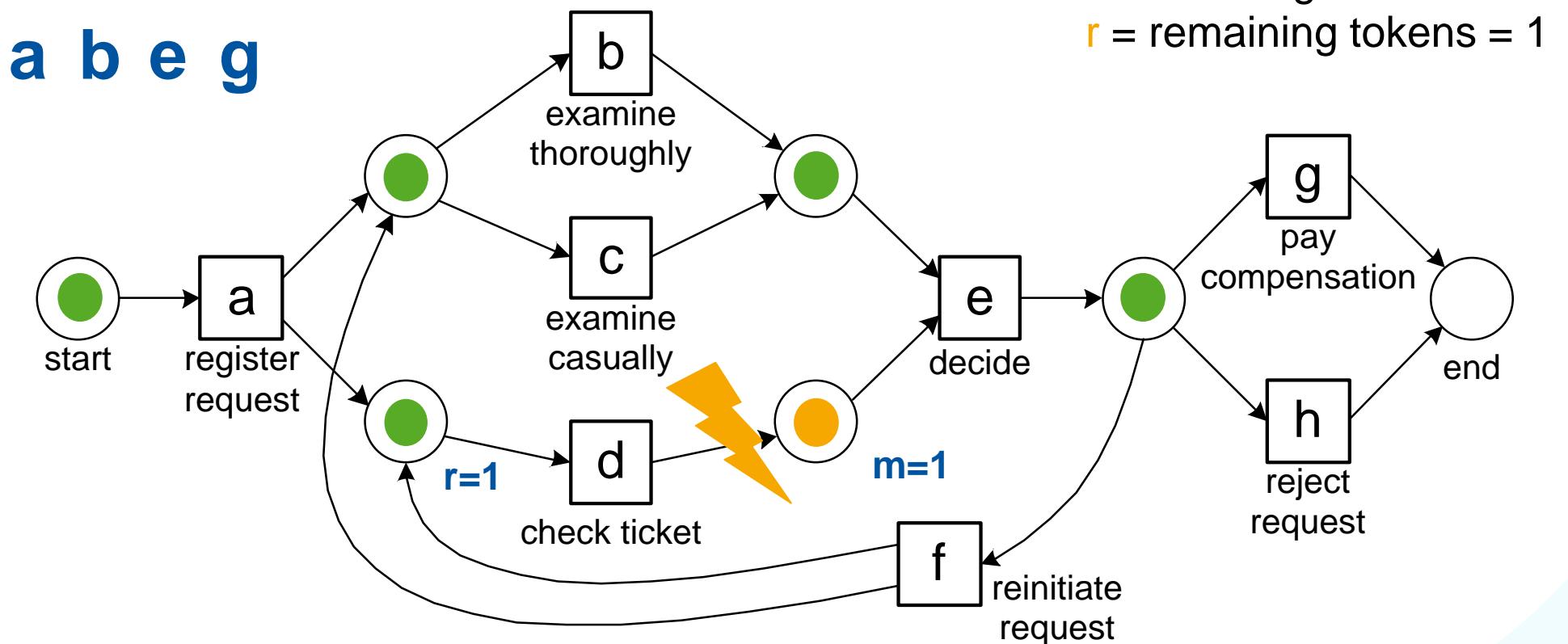
Positioning Conformance Checking



[1]

There are several conformance-checking techniques,
here we focus on “token-based replay”.

Counting Tokens While Replaying



Fitness at the Trace Level

$$\text{fitness}(\sigma, N) = \frac{1}{2} \left(1 - \frac{m}{c}\right) + \frac{1}{2} \left(1 - \frac{r}{p}\right)$$

Fitness at the Trace Level

$$\text{fitness}(\sigma, N) = \frac{1}{2} \left(1 - \frac{1}{6}\right) + \frac{1}{2} \left(1 - \frac{1}{6}\right) = \frac{5}{6} \approx 0.83$$

p = produced tokens = 6

c = consumed tokens = 6

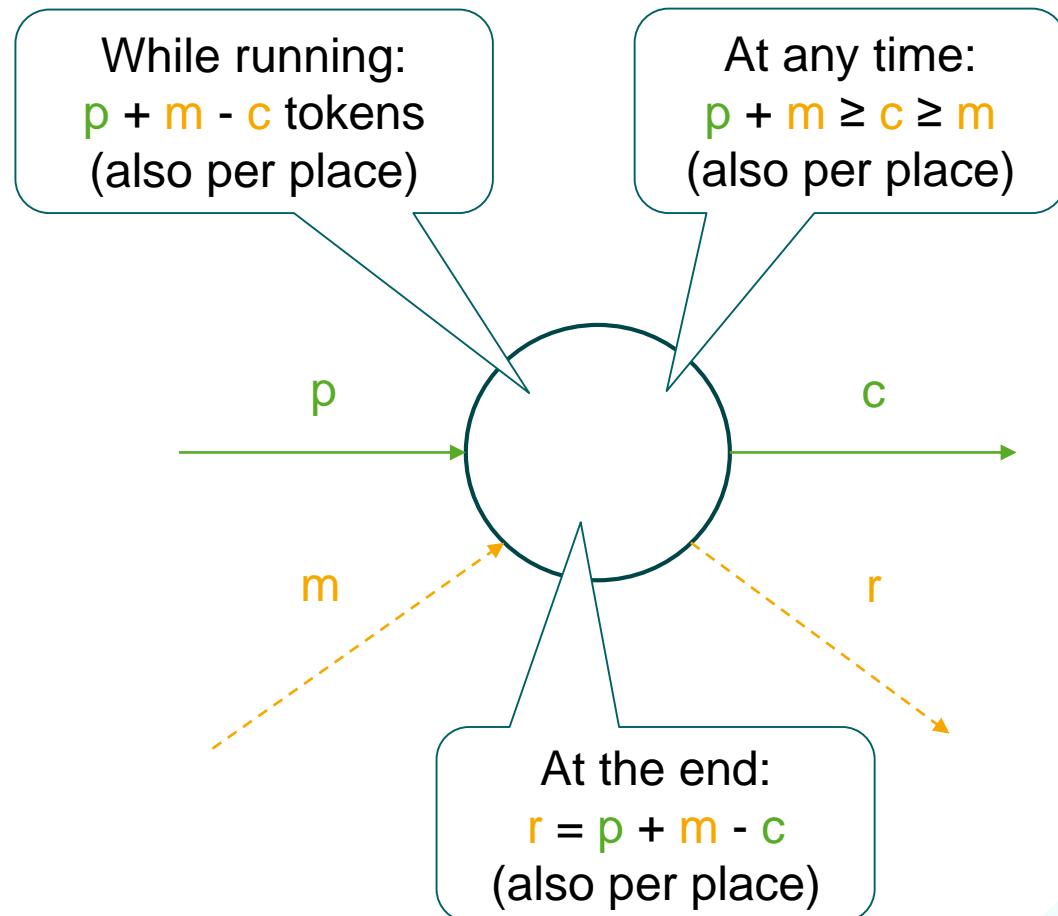
m = missing tokens = 1

r = remaining tokens = 1

Token-Based Replay Approach

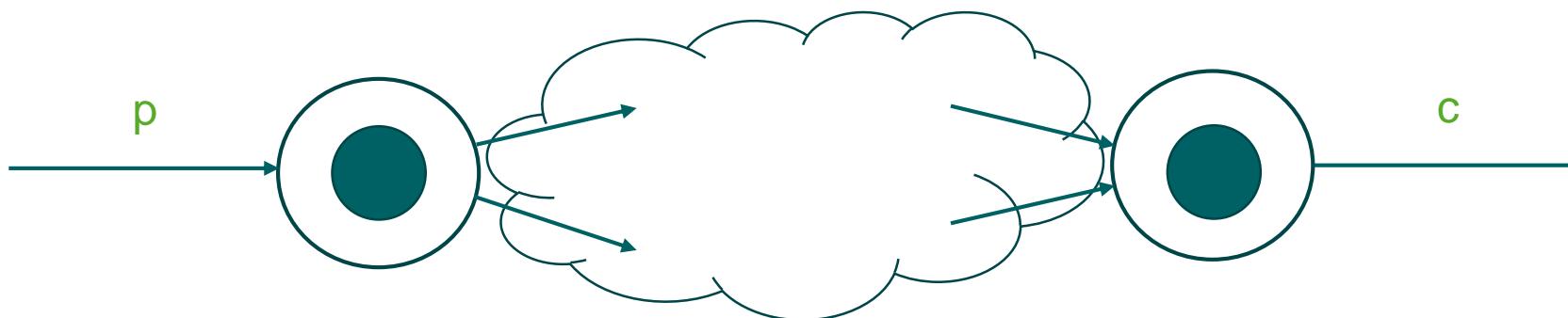
Use four counters:

- p = produced tokens
- c = consumed tokens
- m = missing tokens
(consumed while not there)
- r = remaining tokens
(produced but not consumed)



Token-Based Replay Approach

- Initially, a token is **produced** for the start place – **increment p**
- Finally, a token is **consumed** from the end place (also if it is missing) – **increment c**
(possibly also **m**)

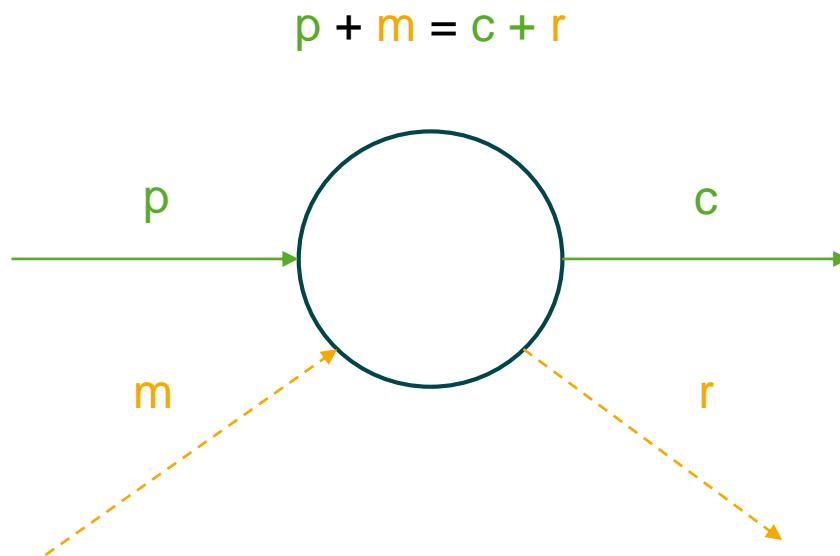


Diagnostics

Dimensions

- Per place or sum for all places in the model
- Per trace or sum for all traces in the event log

Four possible combinations



Four counters:

- p = produced tokens
- c = consumed tokens
- m = missing tokens
- r = remaining tokens

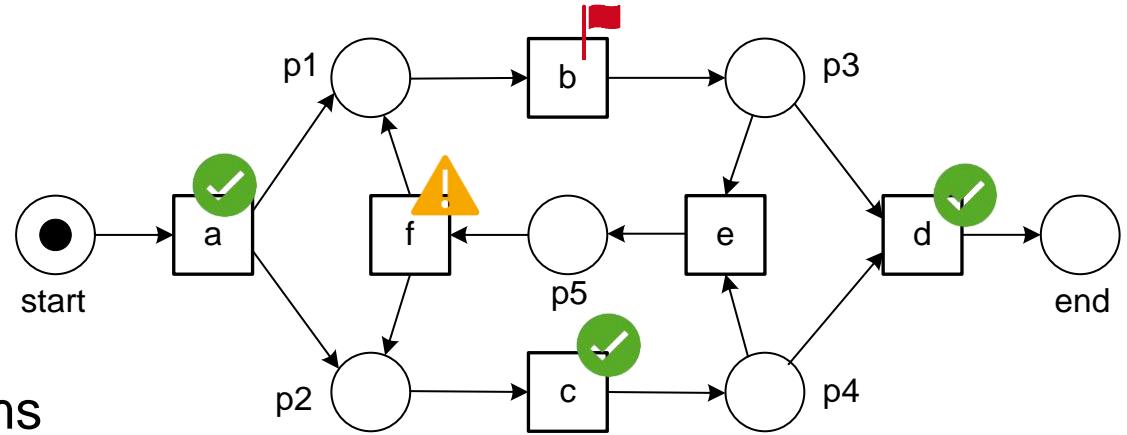
Summary Token-Based Replay Approach

- Pick a trace and initialize the process model by producing a token for the start place
- Fire the transition that corresponds to the next activity in the trace and update the counters for produced and consumed tokens. If not possible, add the required missing tokens first and update the missing tokens counter
- Repeat the above step until the end of the trace is reached
- Consume a token from the end place. If not possible, add the required missing token first and update the missing tokens counter
- Update the remaining tokens counter for each remaining token
- Compute:

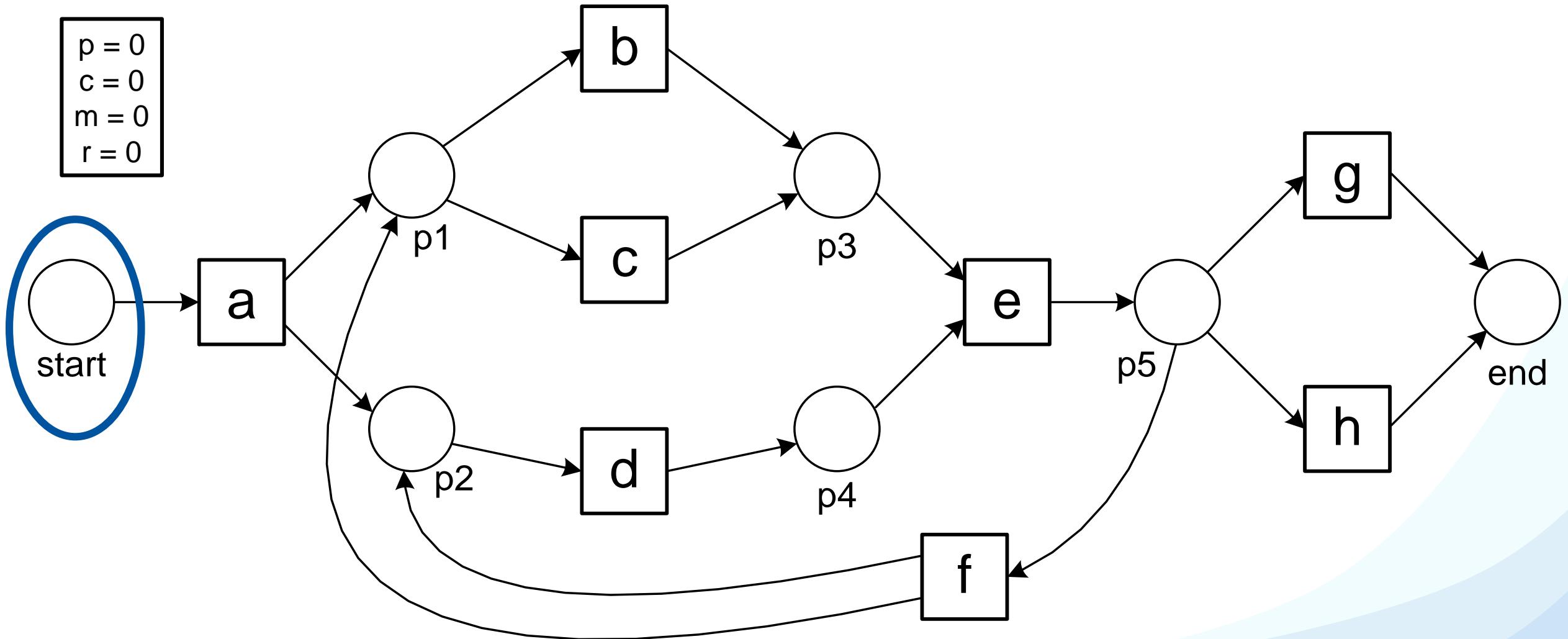
$$\text{fitness}(\sigma, N) = \frac{1}{2} \left(1 - \frac{m}{c} \right) + \frac{1}{2} \left(1 - \frac{r}{p} \right)$$

Supervised Process Mining

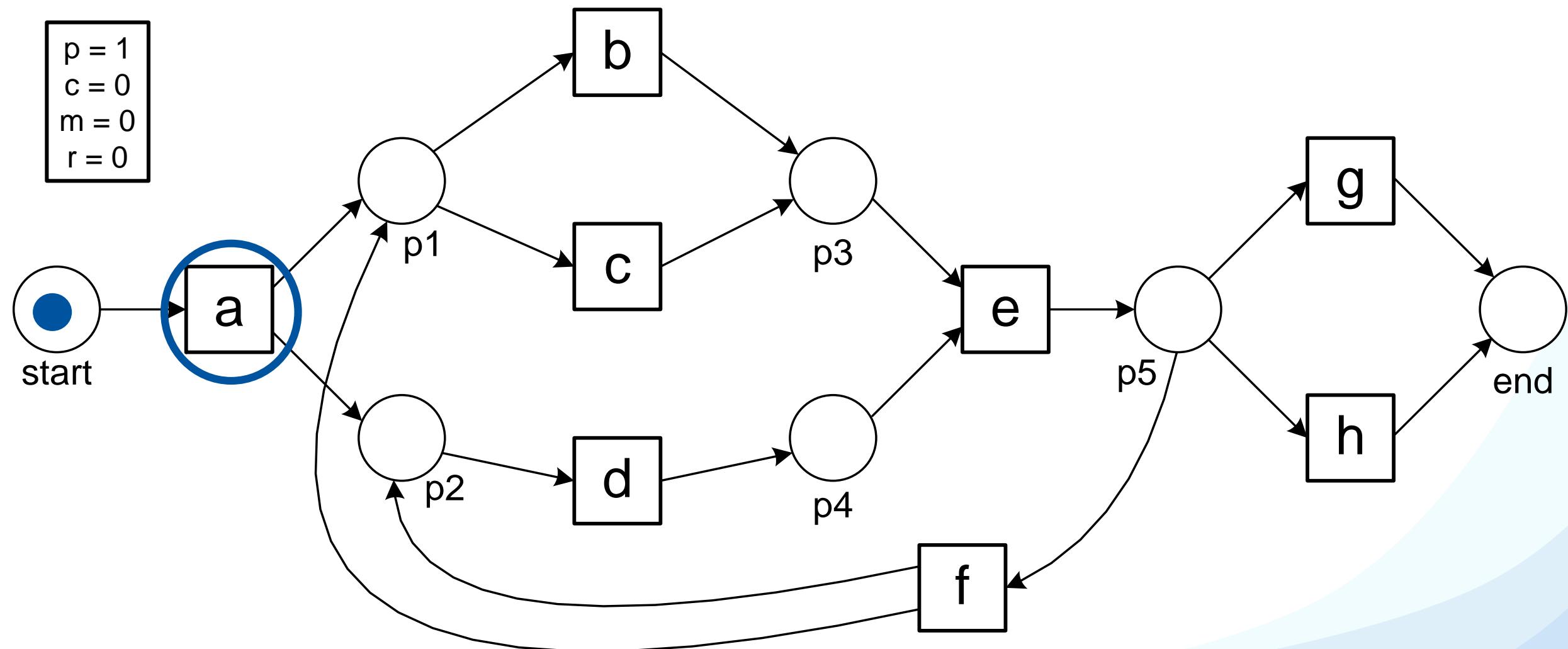
1. Token-Based Replay
2. **Token-Based Replay Examples**
3. Fitness at the Log Level
4. Generating Supervised Learning Problems



Replaying $\sigma_1 = \langle a, c, d, e, h \rangle$

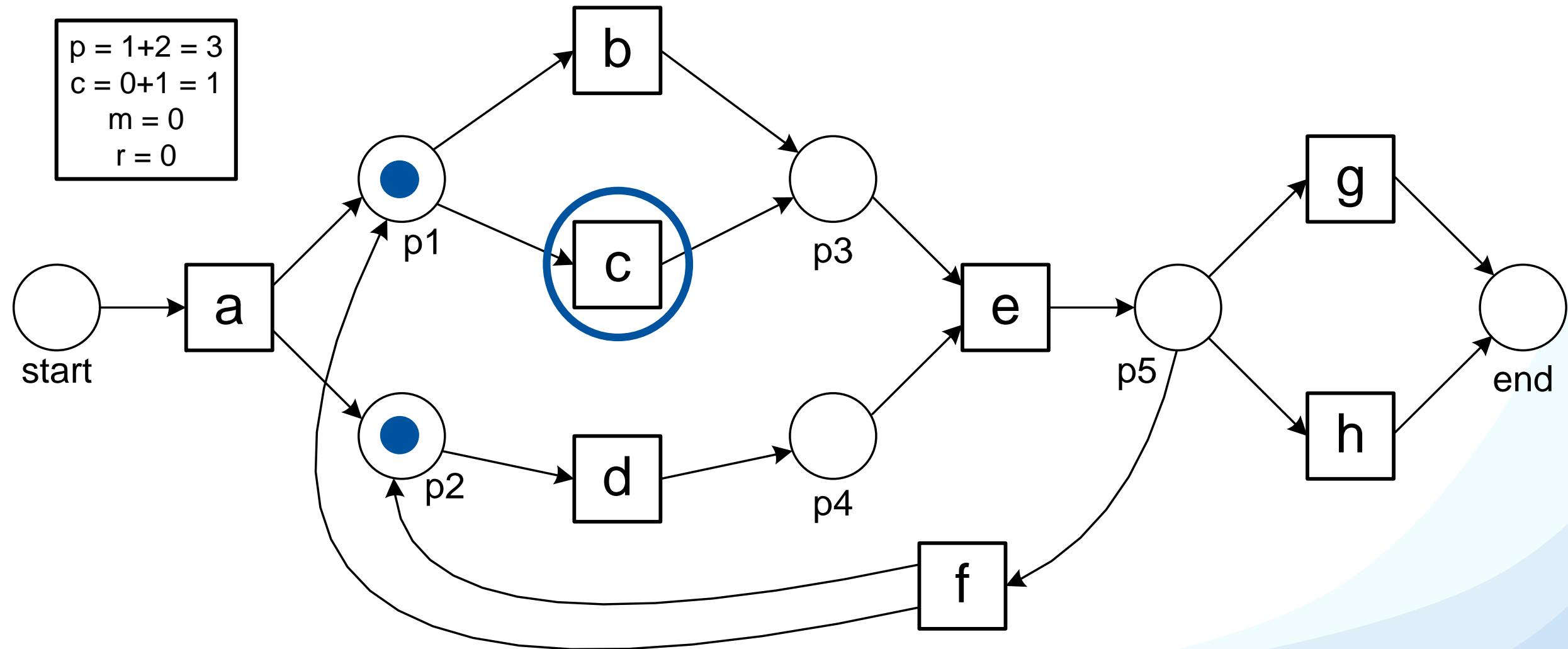


Replaying $\sigma_1 = \langle a, c, d, e, h \rangle$

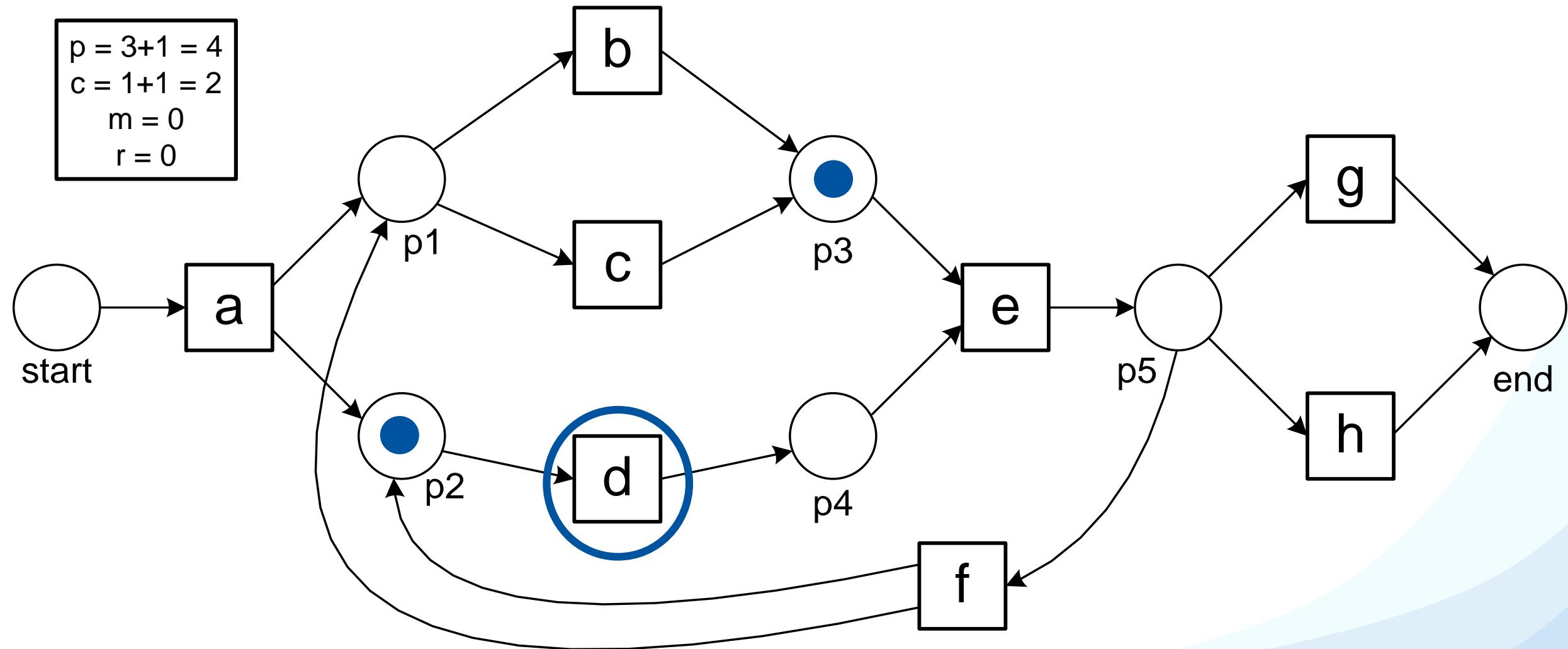


Replaying $\sigma_1 = \langle a, c, d, e, h \rangle$

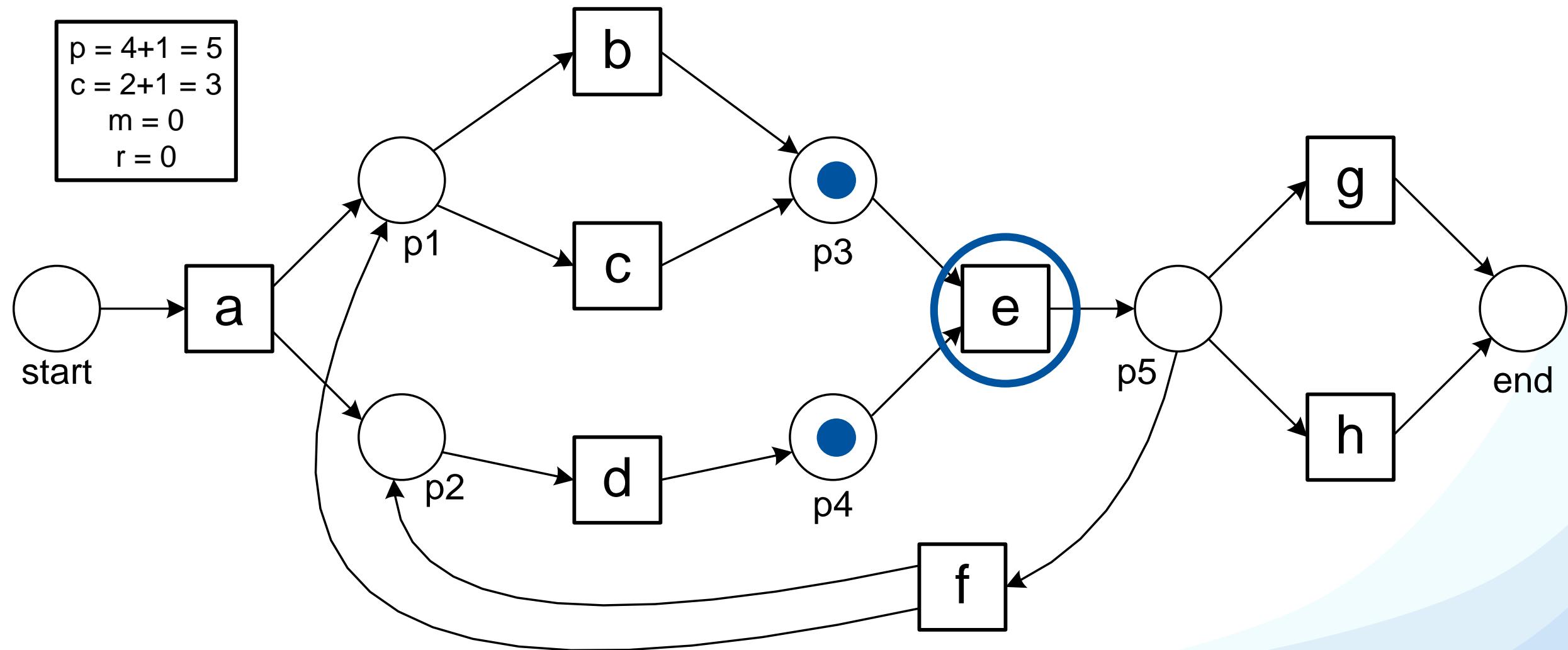
$$\begin{aligned} p &= 1+2 = 3 \\ c &= 0+1 = 1 \\ m &= 0 \\ r &= 0 \end{aligned}$$



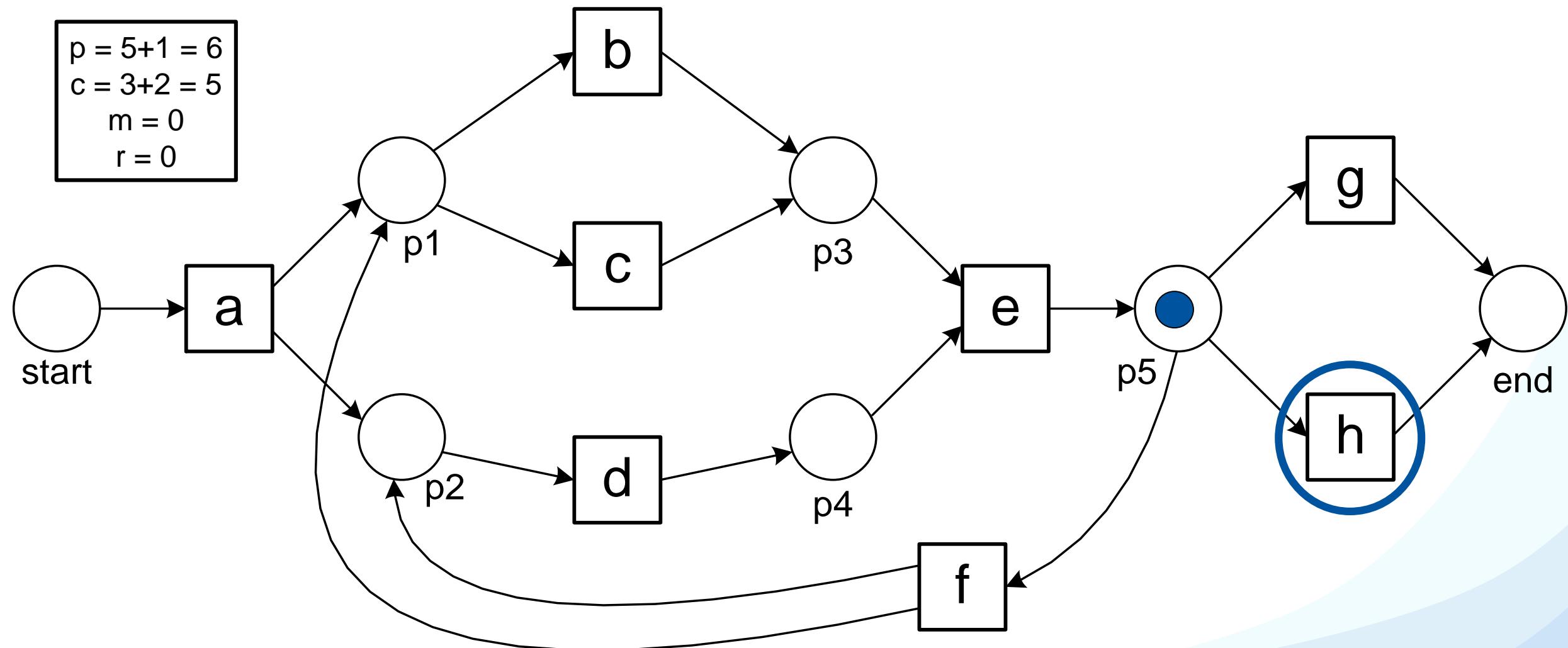
Replaying $\sigma_1 = \langle a, c, \textcolor{blue}{d}, e, h \rangle$



Replaying $\sigma_1 = \langle a, c, d, e, h \rangle$

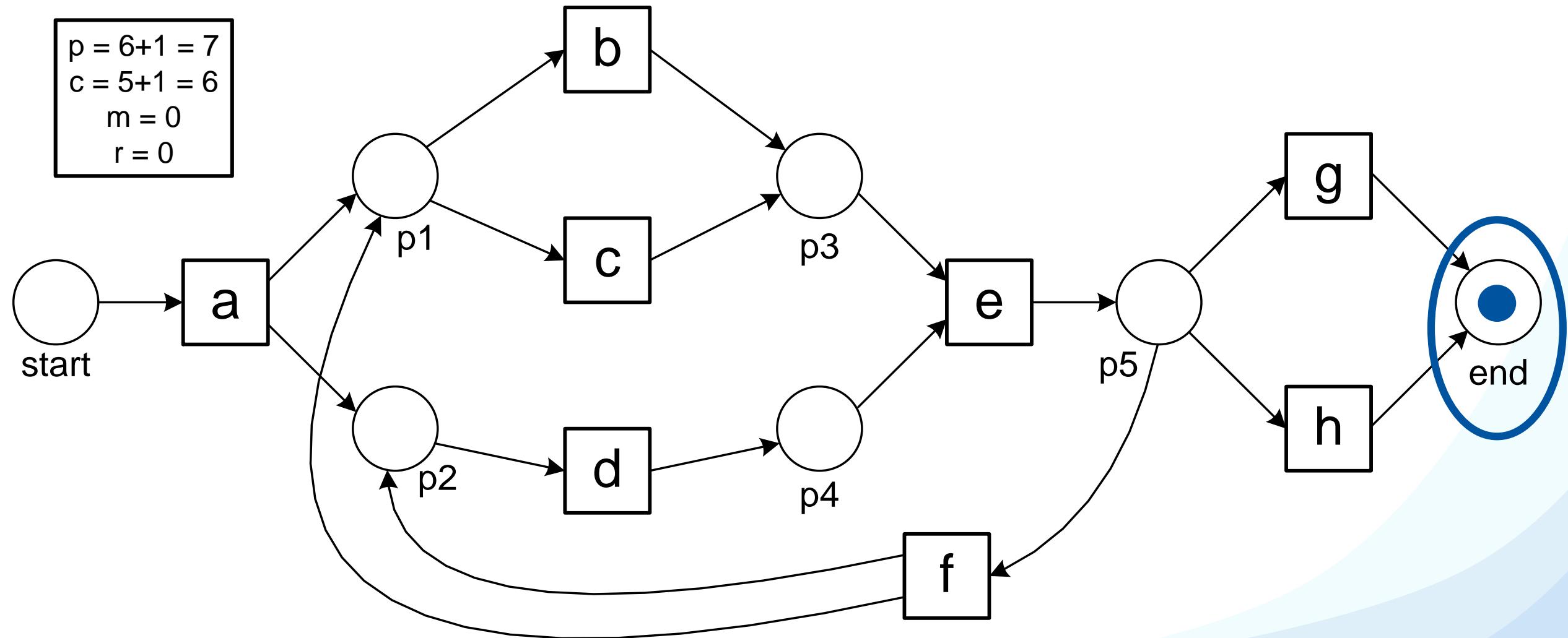


Replaying $\sigma_1 = \langle a, c, d, e, h \rangle$

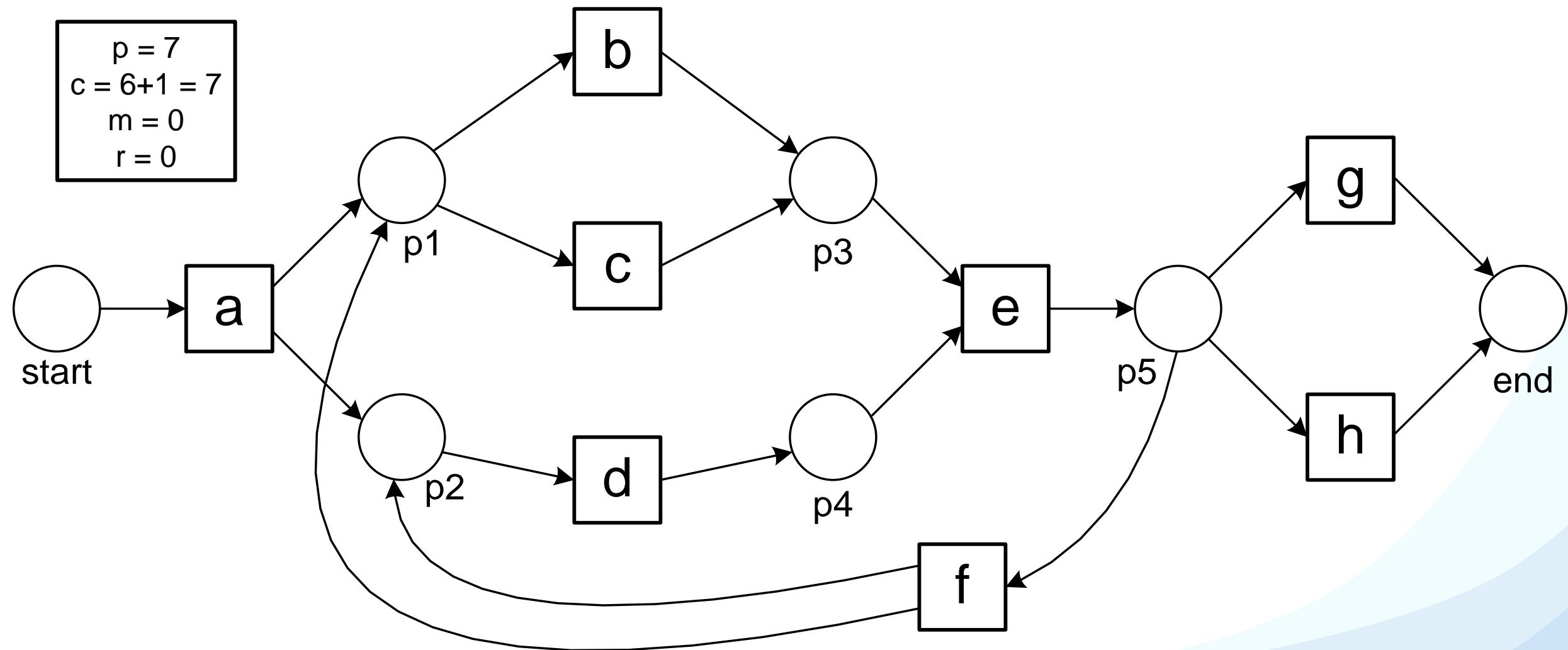


Replaying $\sigma_1 = \langle a, c, d, e, h \rangle$

$p = 6+1 = 7$
 $c = 5+1 = 6$
 $m = 0$
 $r = 0$



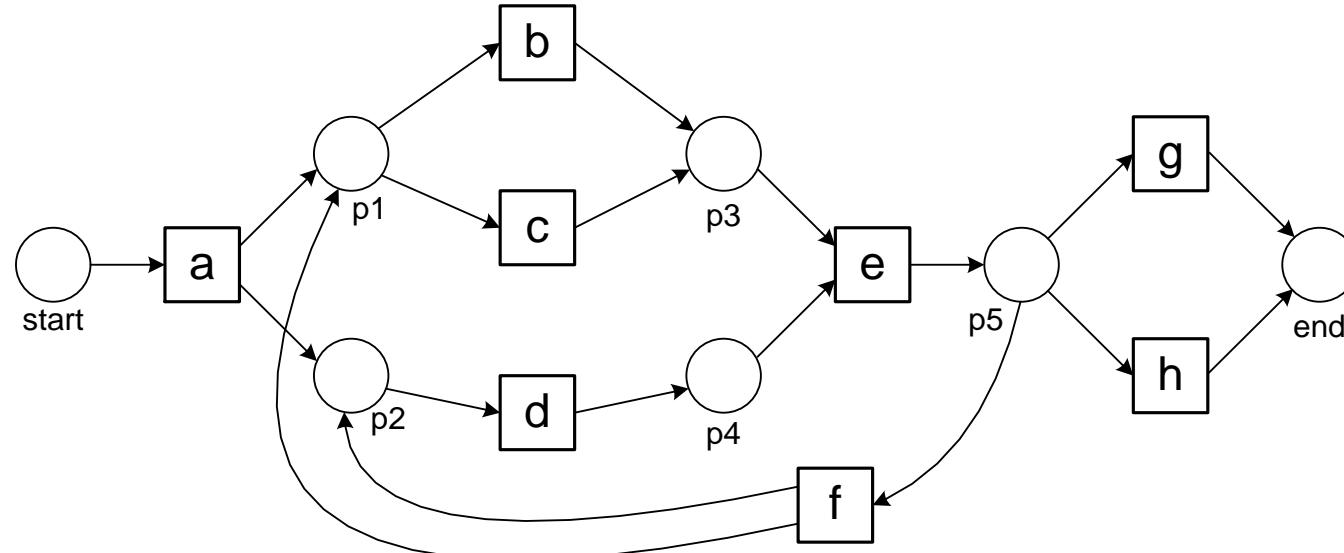
Replaying $\sigma_1 = \langle a, c, d, e, h \rangle$



Fitness at the Trace Level

$p = 7$
$c = 7$
$m = 0$
$r = 0$

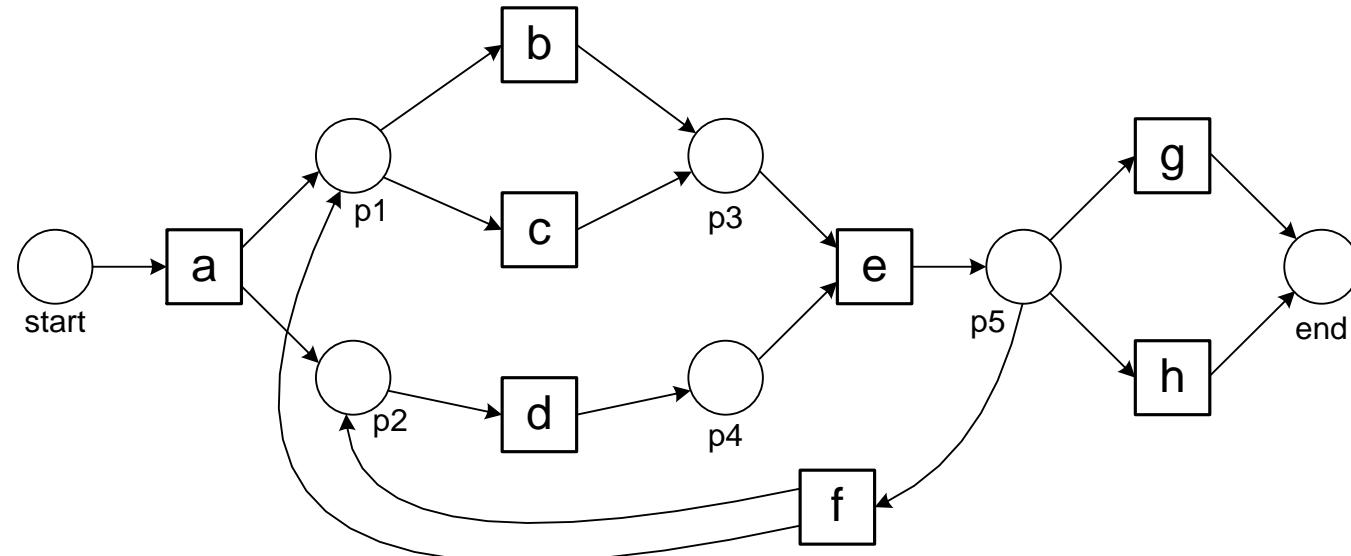
$$\sigma_1 = \langle a, c, d, e, h \rangle$$



$$\text{fitness}(\sigma_1, N) = \frac{1}{2} \left(1 - \frac{m}{c}\right) + \frac{1}{2} \left(1 - \frac{r}{p}\right)$$

Fitness at the Trace Level

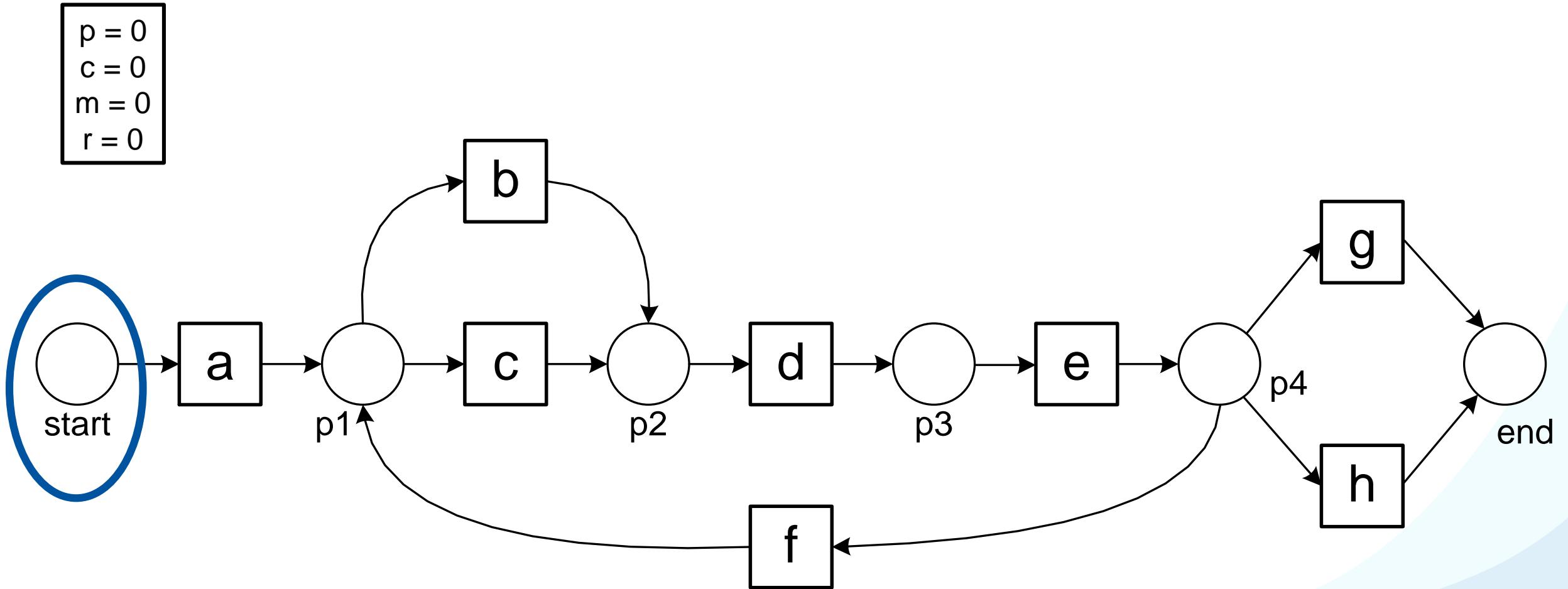
$$\begin{array}{l} p = 7 \\ c = 7 \\ m = 0 \\ r = 0 \end{array}$$



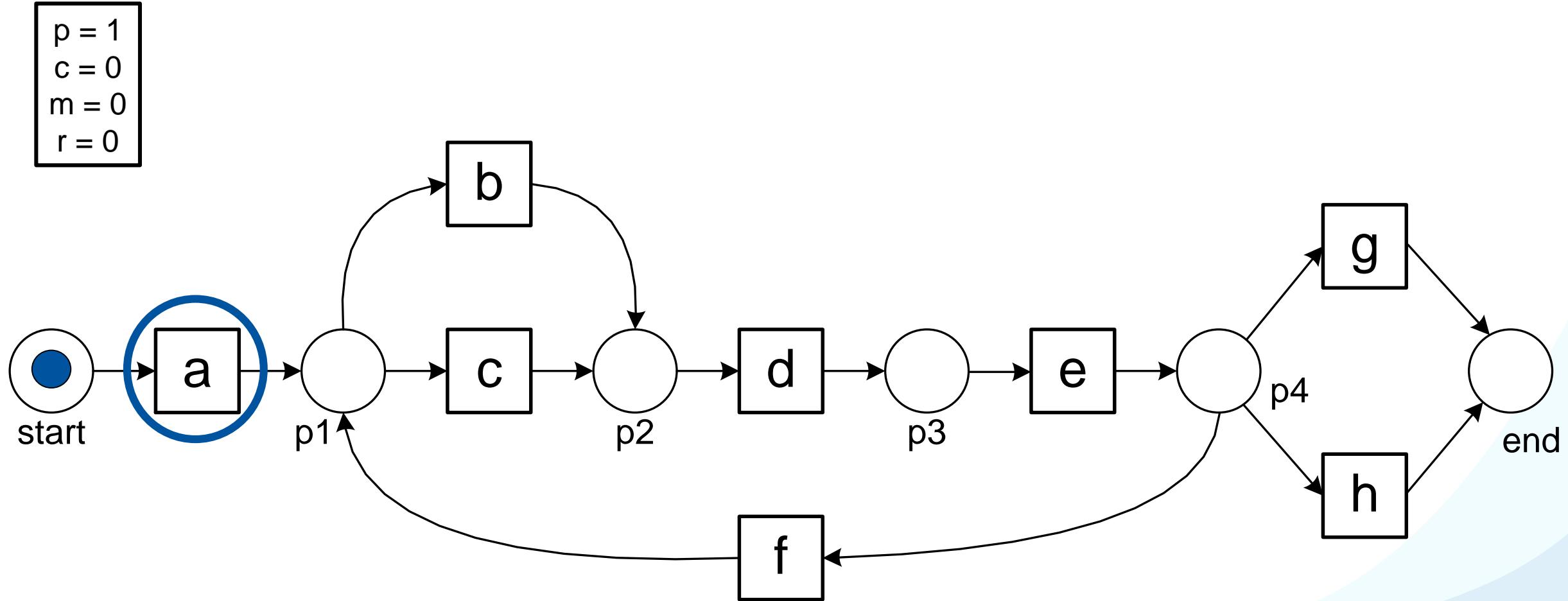
$$\sigma_1 = \langle a, c, d, e, h \rangle$$

$$\text{fitness}(\sigma_1, N) = \frac{1}{2} \left(1 - \frac{0}{7}\right) + \frac{1}{2} \left(1 - \frac{0}{7}\right) = 1$$

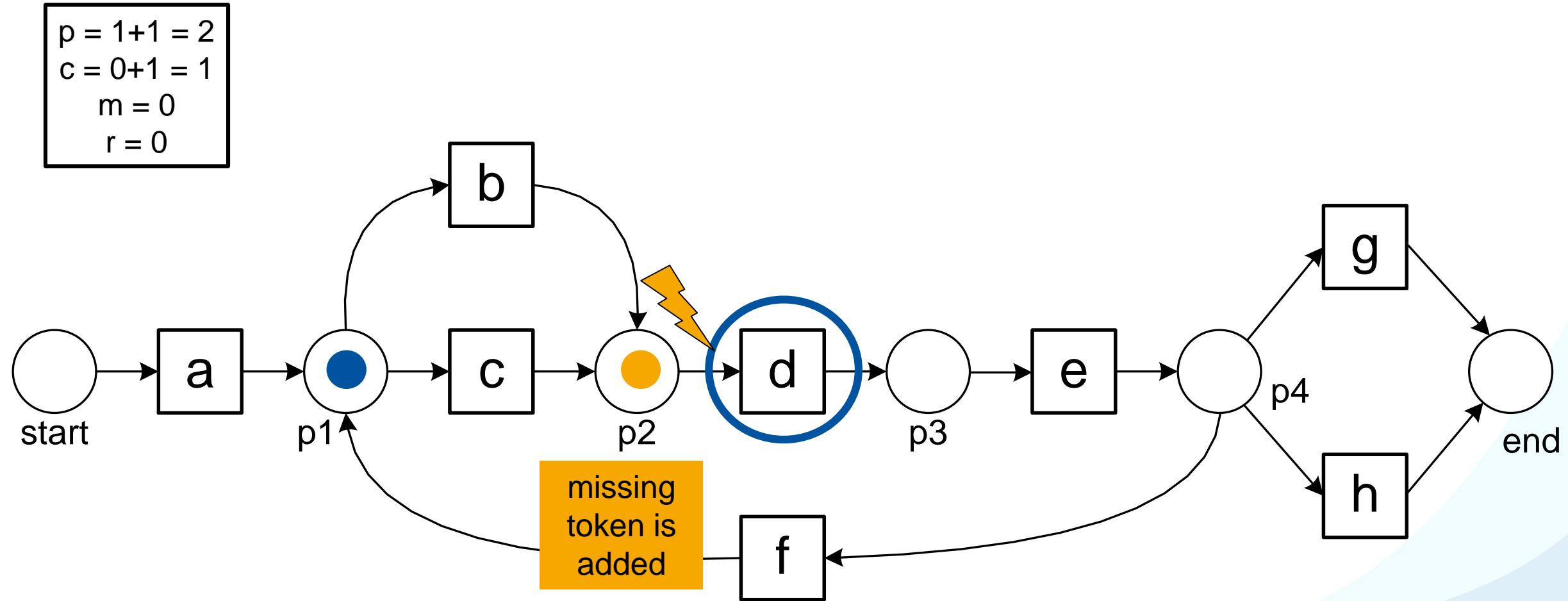
Replaying $\sigma_2 = \langle a, d, c, e, h \rangle$



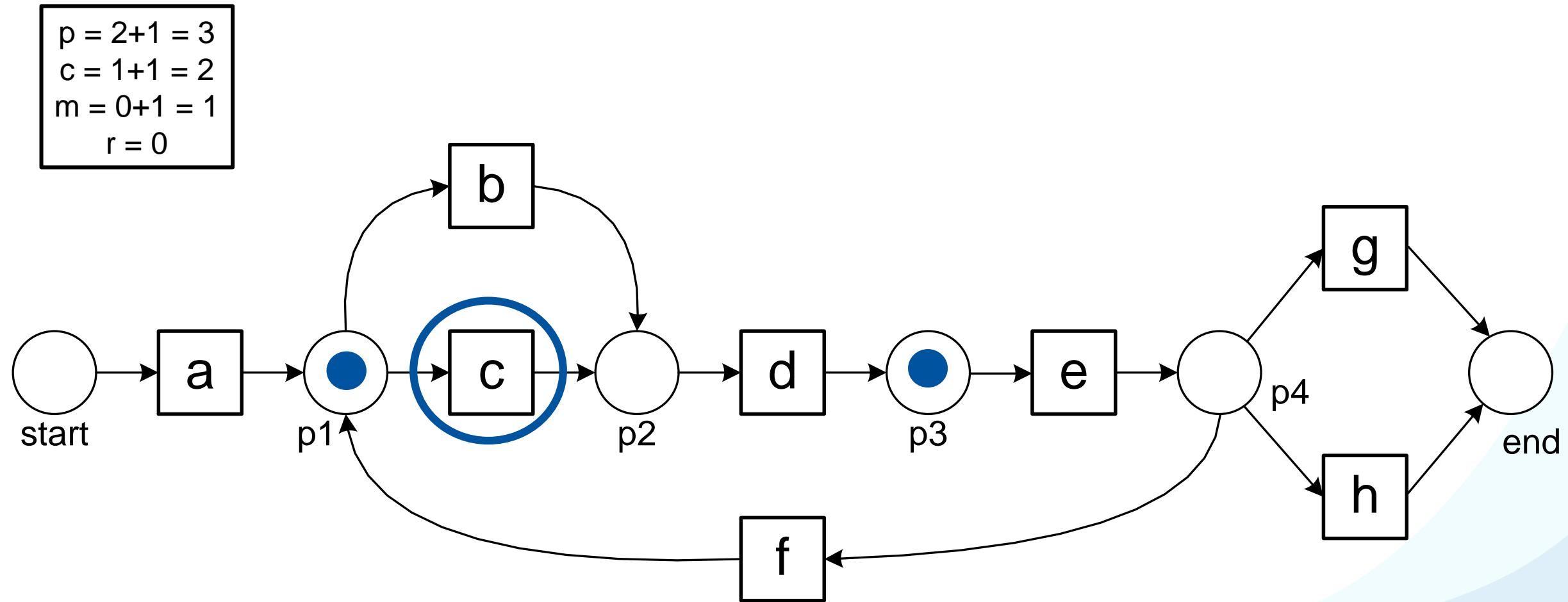
Replaying $\sigma_2 = \langle a, d, c, e, h \rangle$



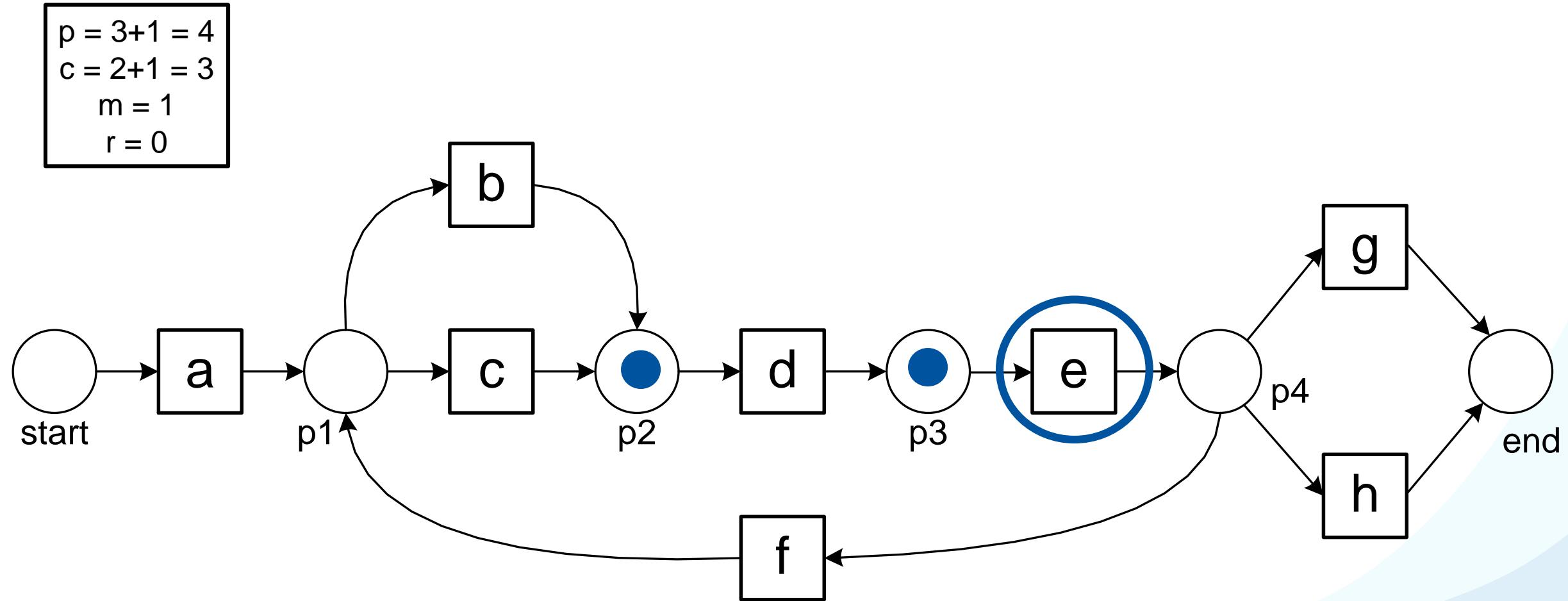
Replaying $\sigma_2 = \langle a, d, c, e, h \rangle$



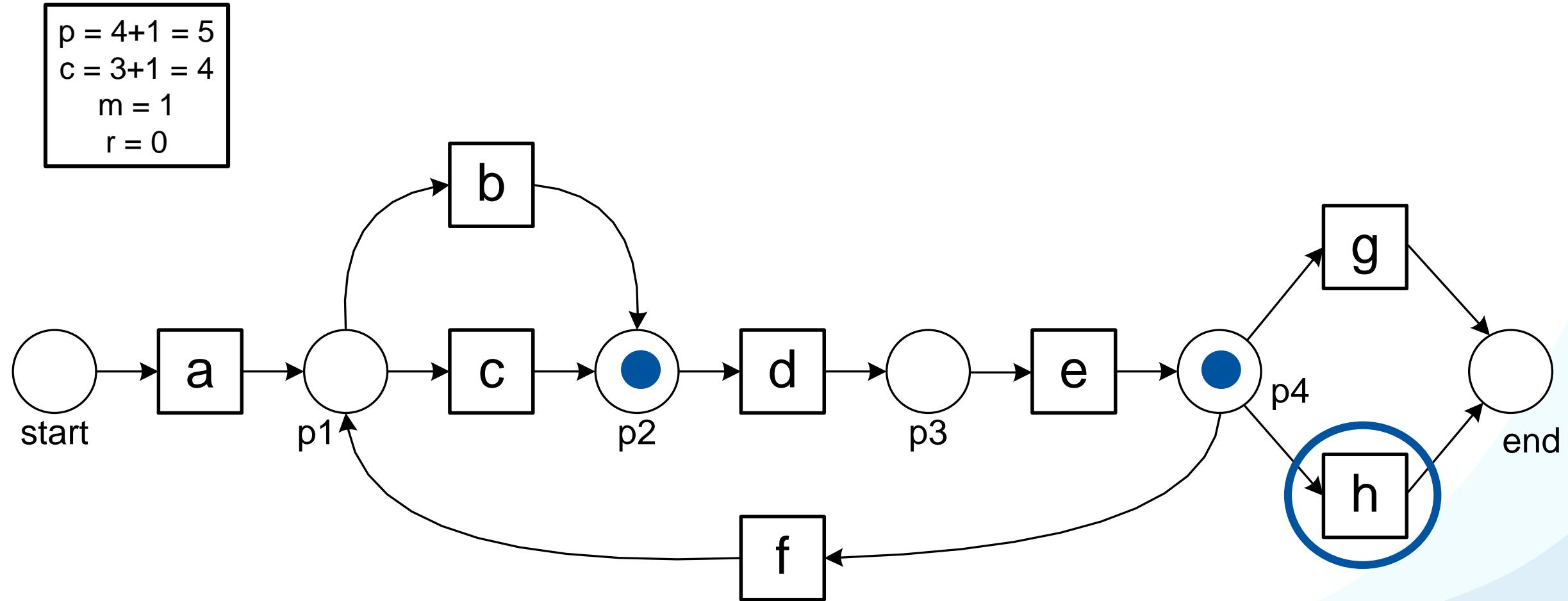
Replaying $\sigma_2 = \langle a, d, \textcolor{blue}{c}, e, h \rangle$



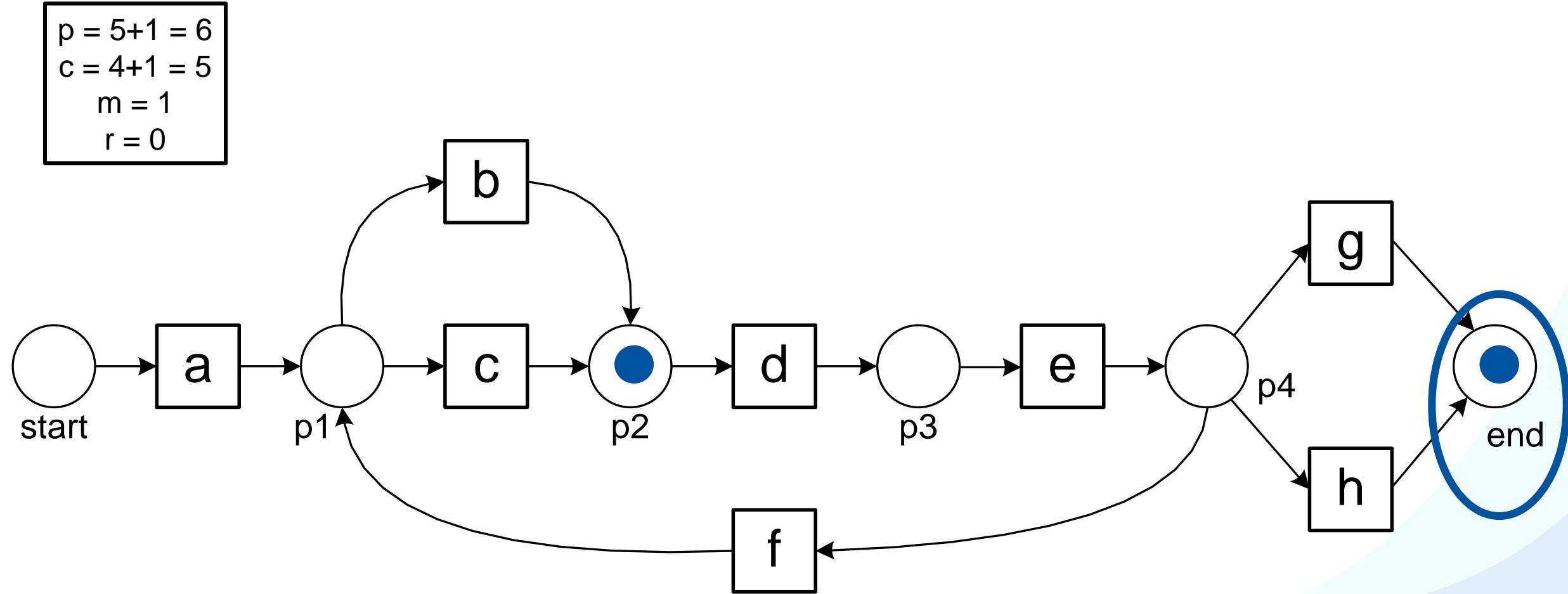
Replaying $\sigma_2 = \langle a, d, c, e, h \rangle$



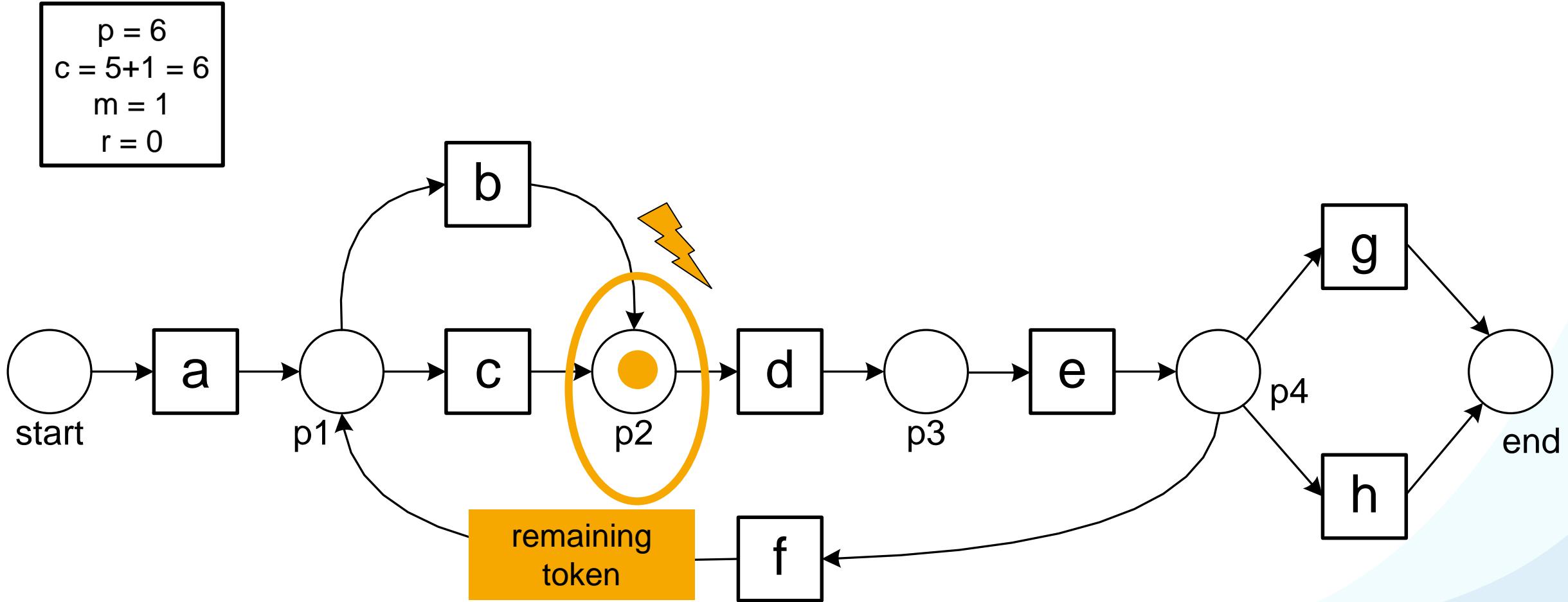
Replaying $\sigma_2 = \langle a, d, c, e, h \rangle$



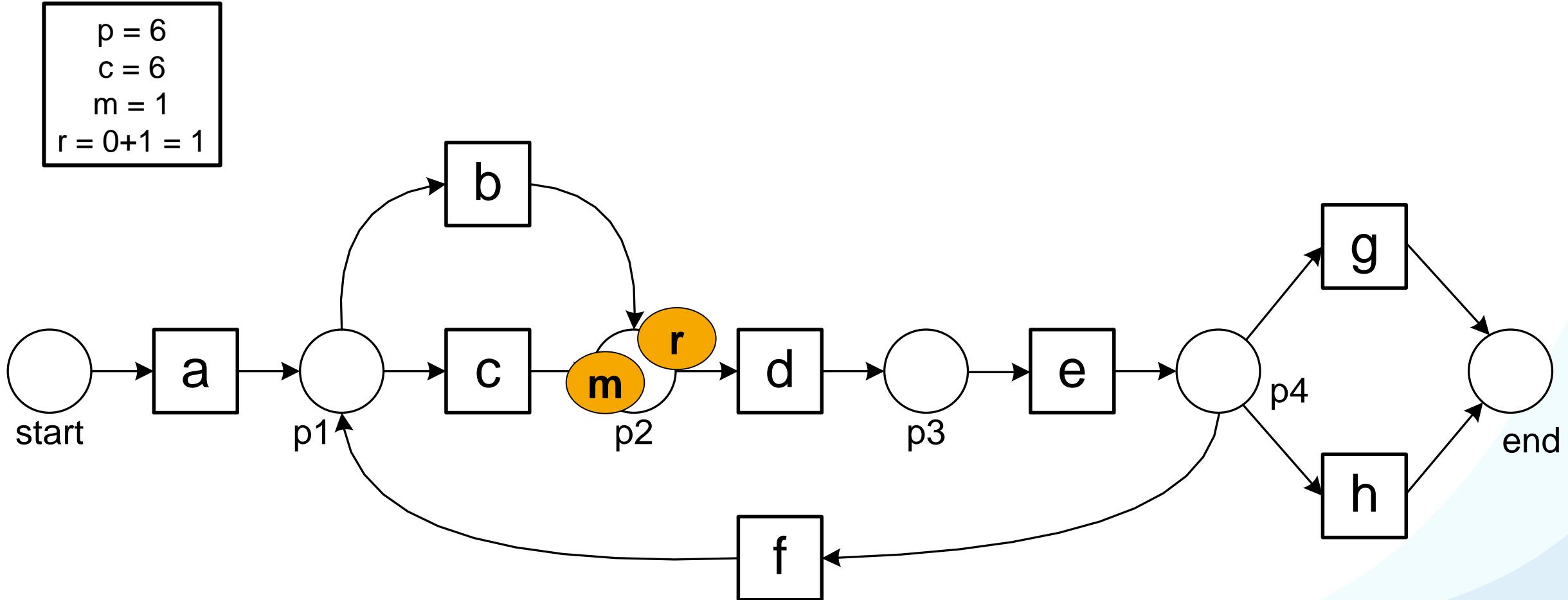
Replaying $\sigma_2 = \langle a, d, c, e, h \rangle$



Replaying $\sigma_2 = \langle a, d, c, e, h \rangle$

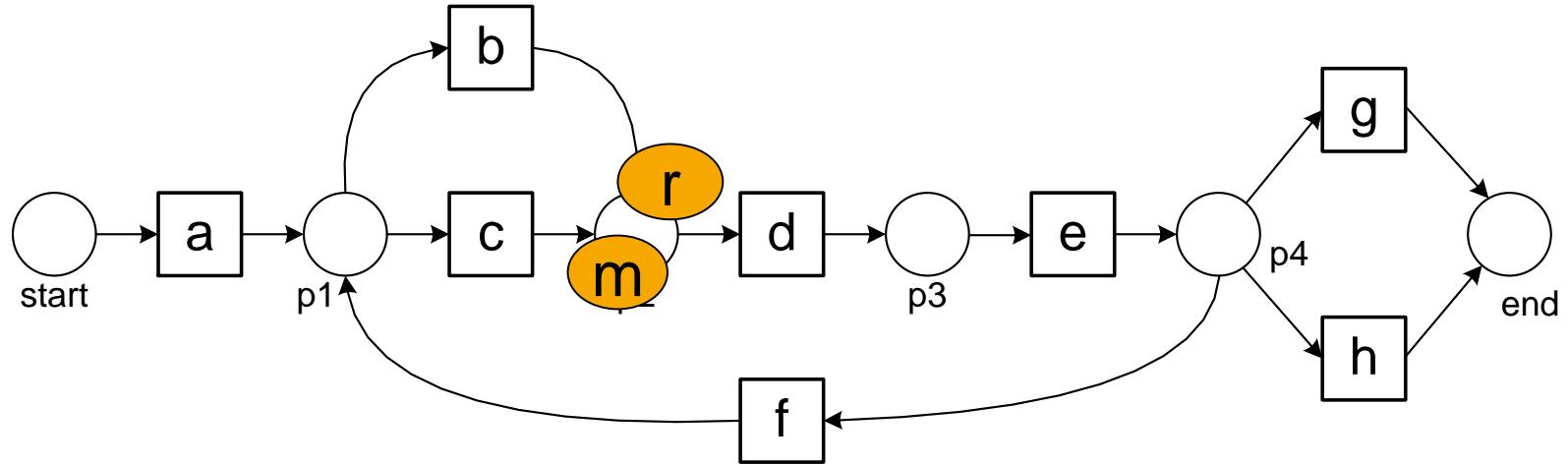


Replaying $\sigma_2 = \langle a, d, c, e, h \rangle$



Fitness at the Trace Level

$p = 6$
$c = 6$
$m = 1$
$r = 1$

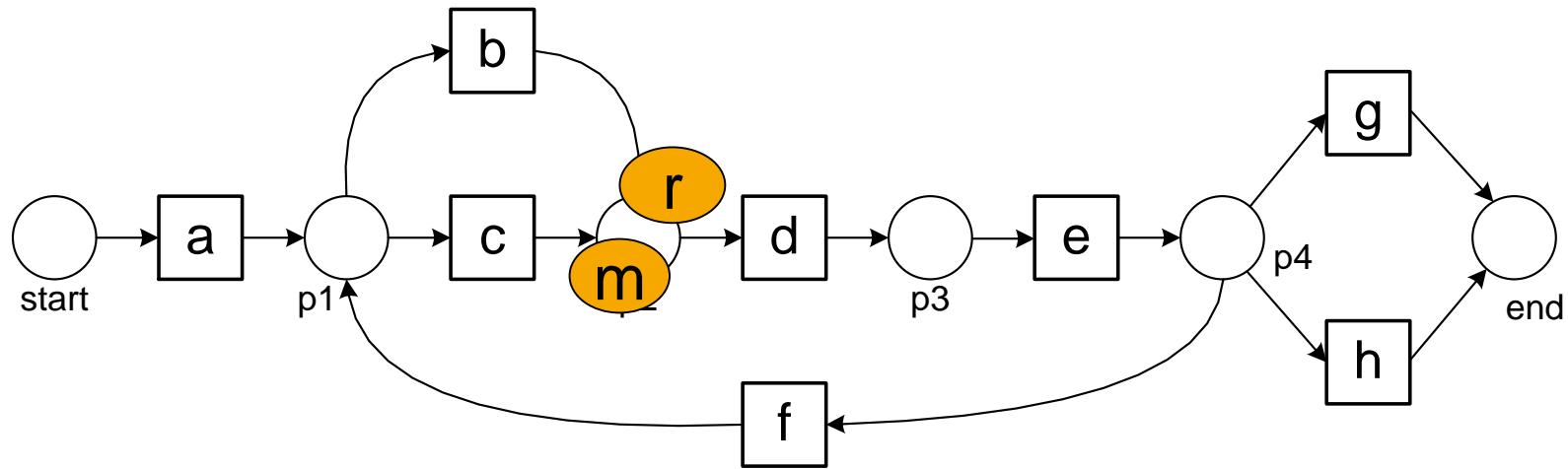


$$\sigma_2 = \langle a, d, c, e, h \rangle$$

$$\text{fitness}(\sigma_2, N) = \frac{1}{2} \left(1 - \frac{m}{c}\right) + \frac{1}{2} \left(1 - \frac{r}{p}\right)$$

Fitness at the Trace Level

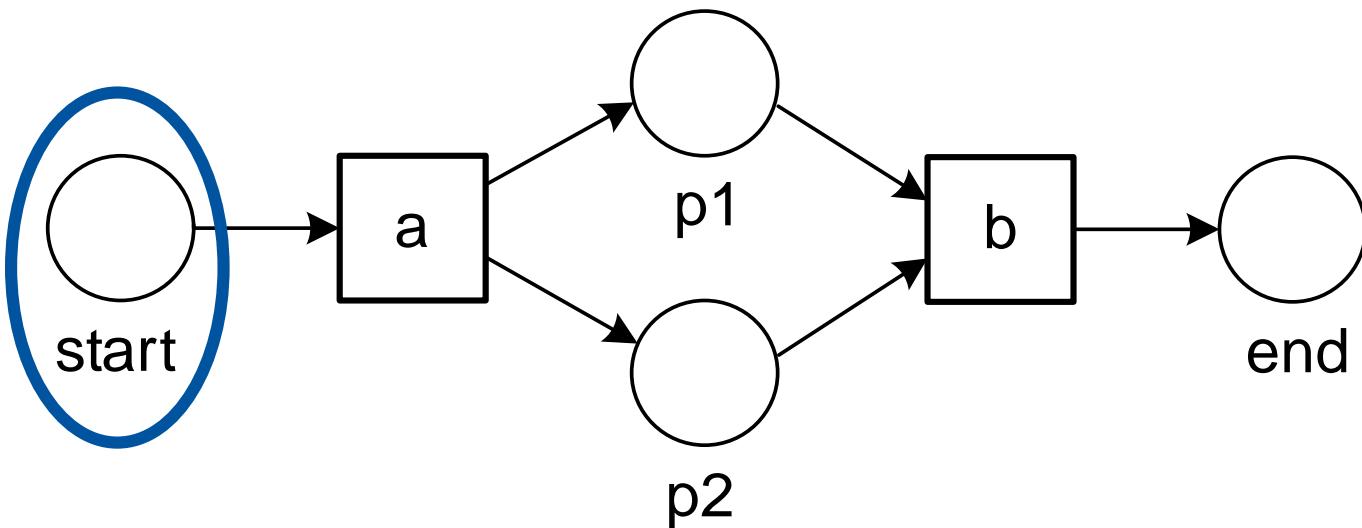
$$\begin{array}{l} p = 6 \\ c = 6 \\ m = 1 \\ r = 1 \end{array}$$



$$\sigma_2 = \langle a, d, c, e, h \rangle$$

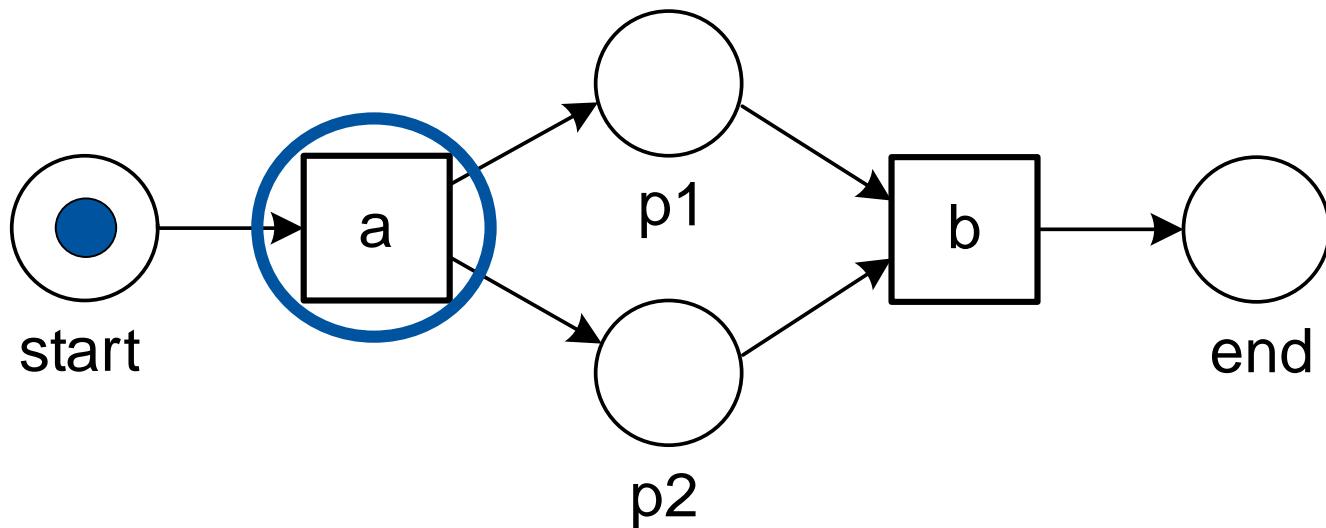
$$\text{fitness}(\sigma_2, N) = \frac{1}{2} \left(1 - \frac{1}{6}\right) + \frac{1}{2} \left(1 - \frac{1}{6}\right) = \frac{5}{6} \approx 0.83$$

Replaying $\sigma_3 = \langle a \rangle$



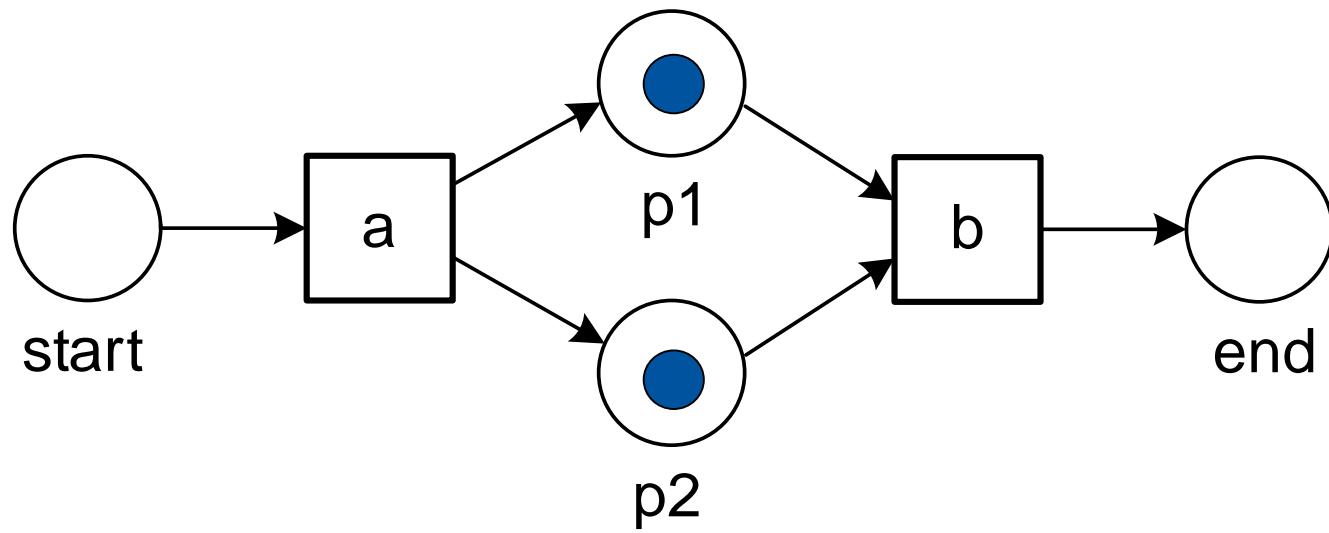
$p = 0$
$c = 0$
$m = 0$
$r = 0$

Replaying $\sigma_3 = \langle a \rangle$



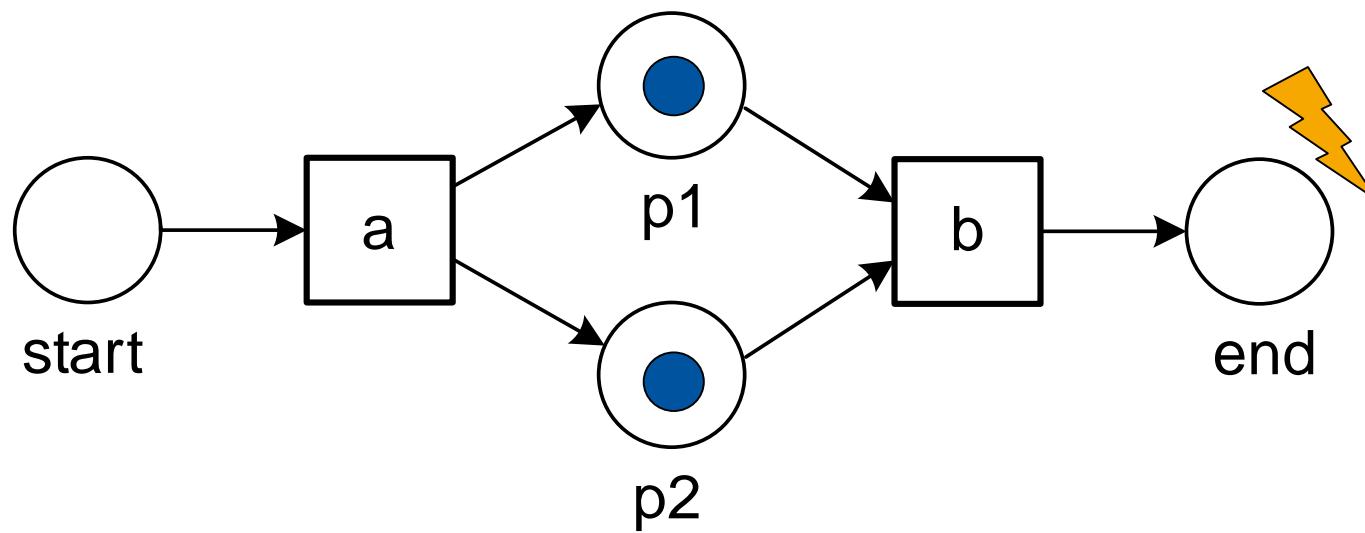
$$\begin{aligned} p &= 0 + 1 = 1 \\ c &= 0 \\ m &= 0 \\ r &= 0 \end{aligned}$$

Replaying $\sigma_3 = \langle a \rangle$



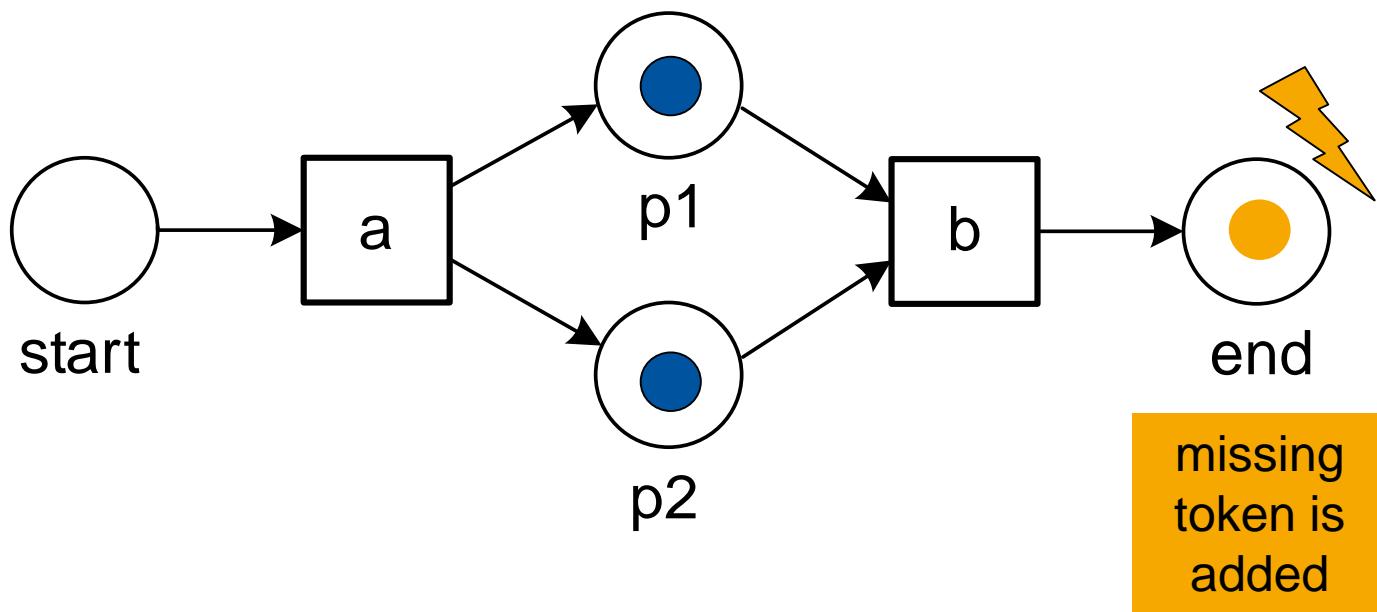
$$\begin{aligned} p &= 1 + 2 = 3 \\ c &= 0 + 1 = 1 \\ m &= 0 \\ r &= 0 \end{aligned}$$

Replaying $\sigma_3 = \langle a \rangle$



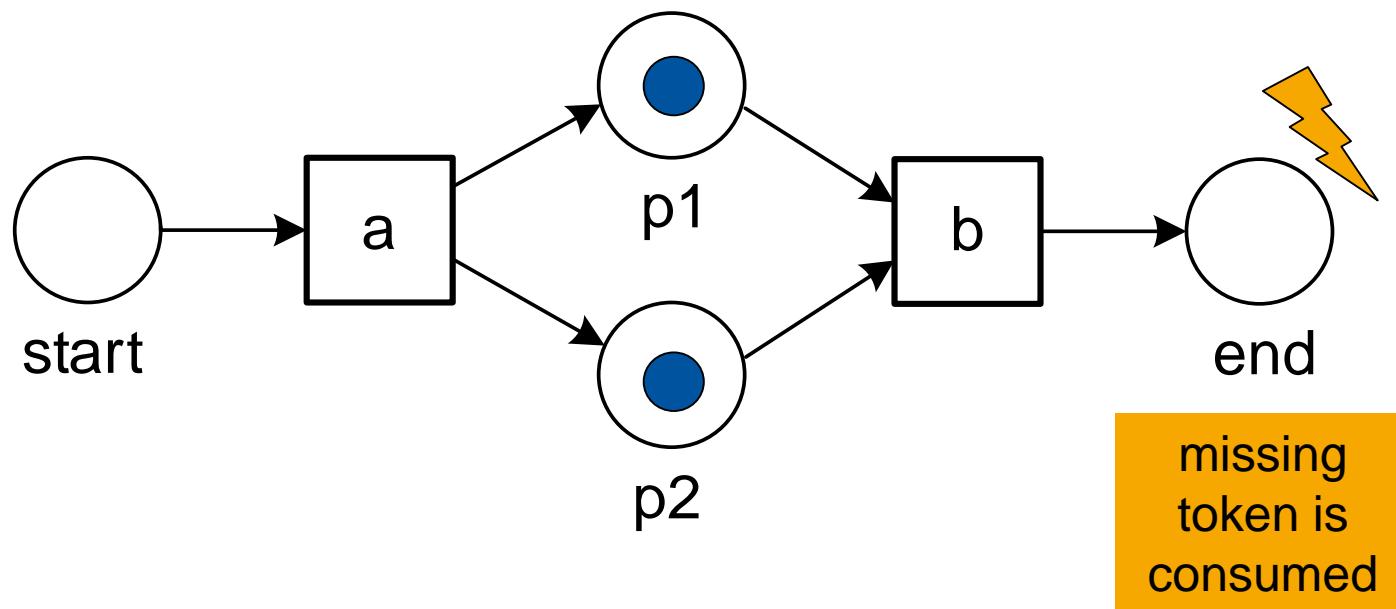
$p = 3$
$c = 1$
$m = 0$
$r = 0$

Replaying $\sigma_3 = \langle a \rangle$



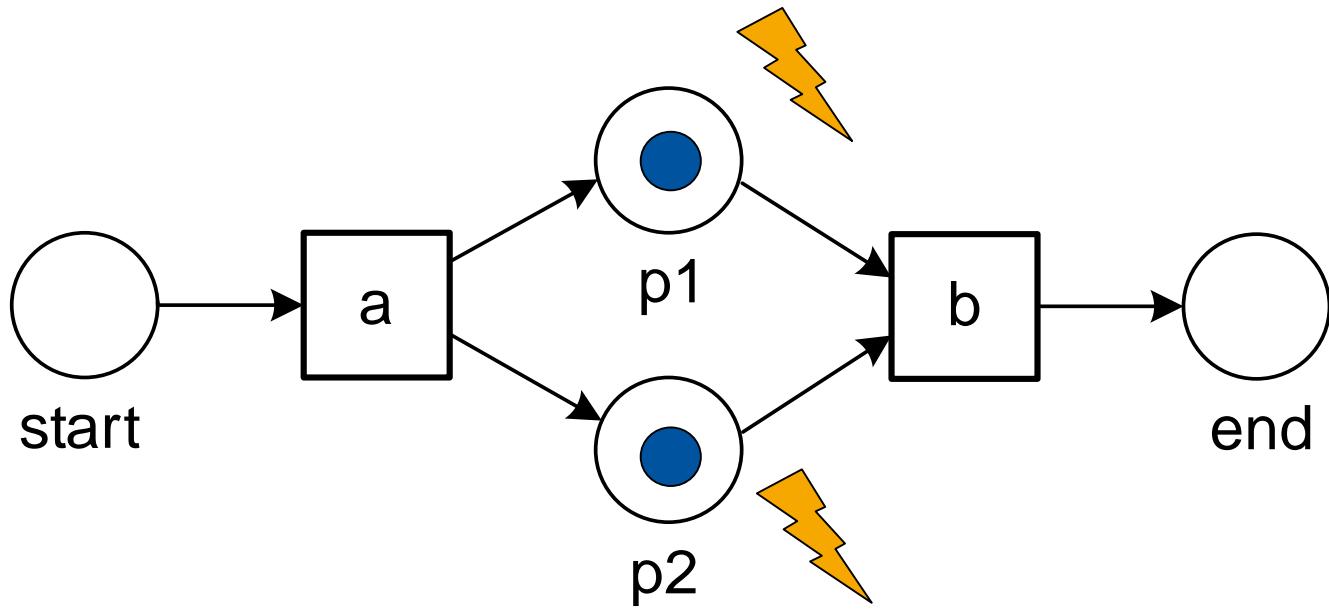
$p = 3$
 $c = 1$
 $m = 0 + 1 = 1$
 $r = 0$

Replaying $\sigma_3 = \langle a \rangle$



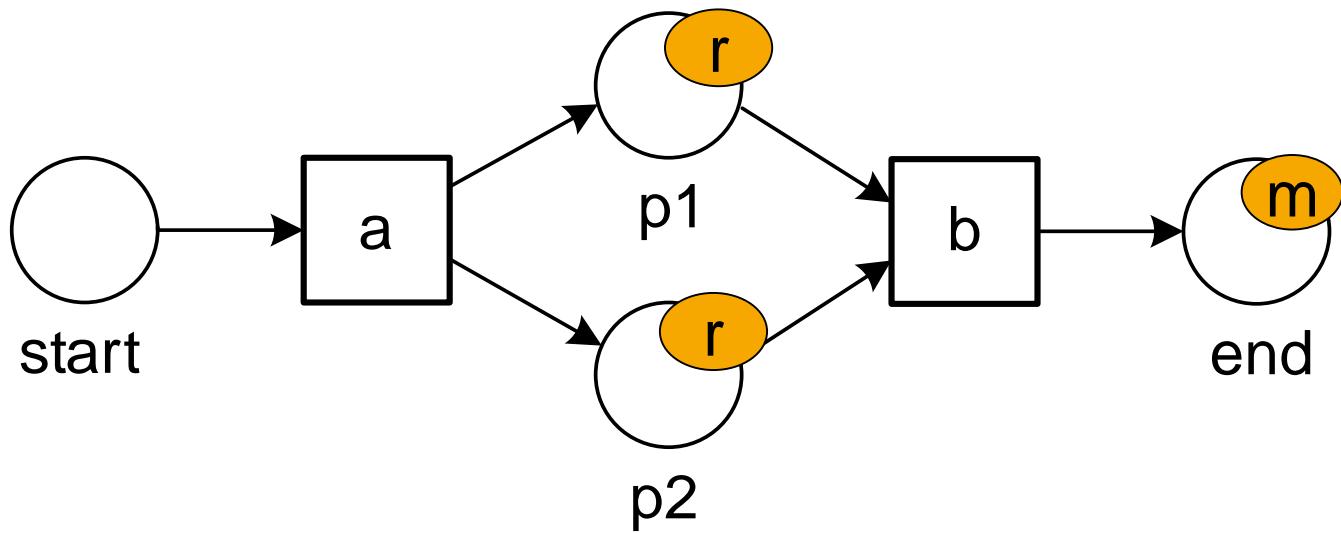
$p = 3$
$c = 1 + 1 = 2$
$m = 1$
$r = 0$

Replaying $\sigma_3 = \langle a \rangle$



$p = 3$
$c = 2$
$m = 1$
$r = 0 + 2 = 2$

Fitness at the Trace Level

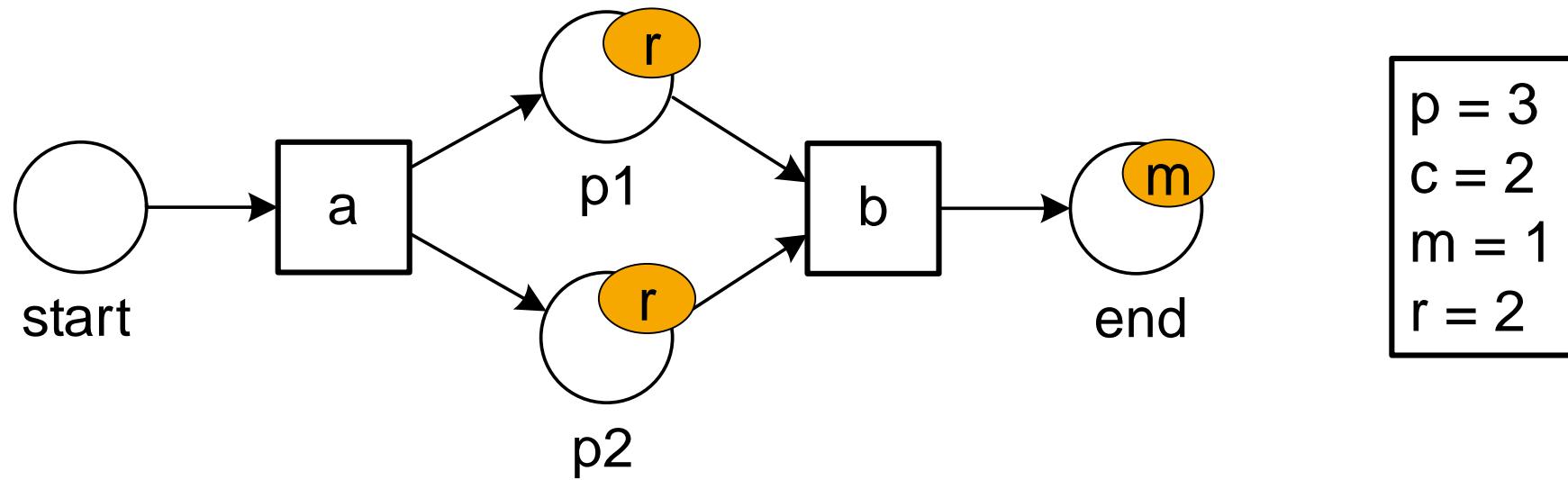


$p = 3$
$c = 2$
$m = 1$
$r = 2$

$$\sigma_3 = \langle a \rangle$$

$$\text{fitness}(\sigma_3, N) = \frac{1}{2} \left(1 - \frac{m}{c} \right) + \frac{1}{2} \left(1 - \frac{r}{p} \right)$$

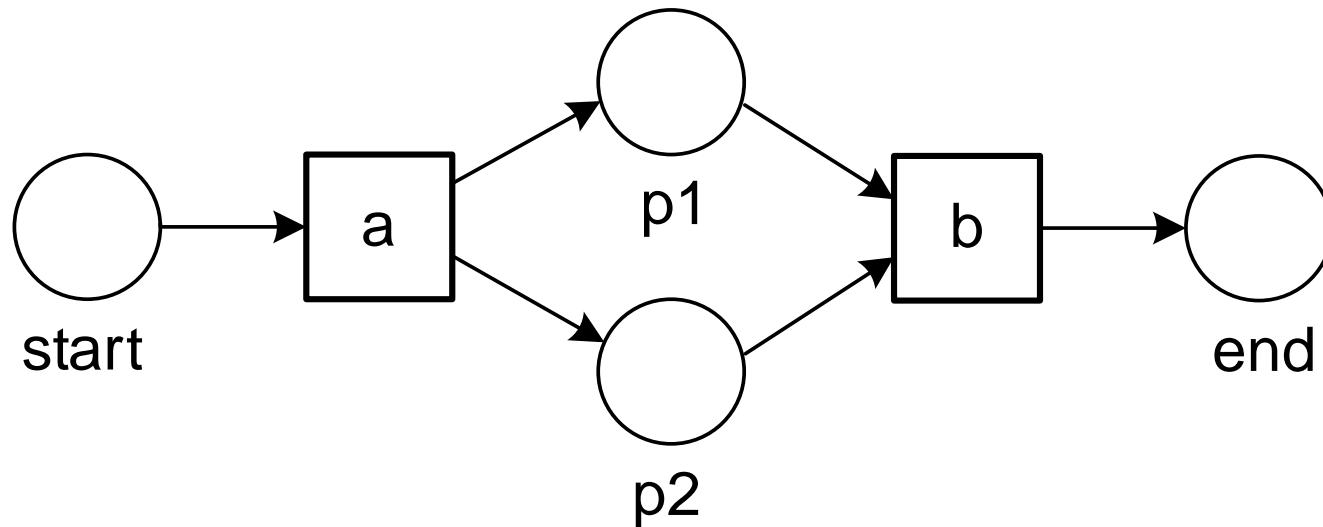
Fitness at the Trace Level



$$\sigma_3 = \langle a \rangle$$

$$\text{fitness}(\sigma_3, N) = \frac{1}{2} \left(1 - \frac{1}{2} \right) + \frac{1}{2} \left(1 - \frac{2}{3} \right) = \frac{5}{12} \approx 0.42$$

What is the Worst Case Scenario?

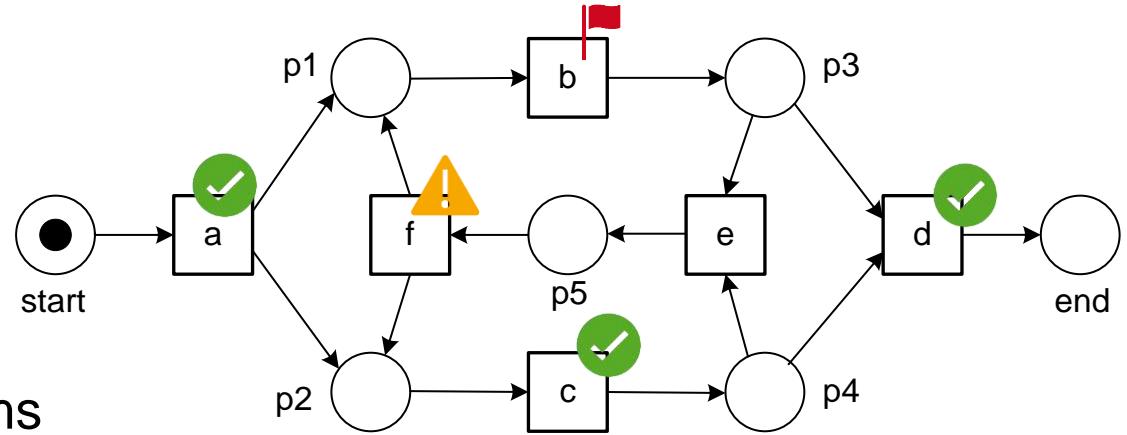


$$\text{fitness}(\sigma_{bad}, N) = \frac{1}{2} \left(1 - \frac{m}{c}\right) + \frac{1}{2} \left(1 - \frac{r}{p}\right) = 0$$

$$\begin{cases} p = r \\ c = m \end{cases}$$

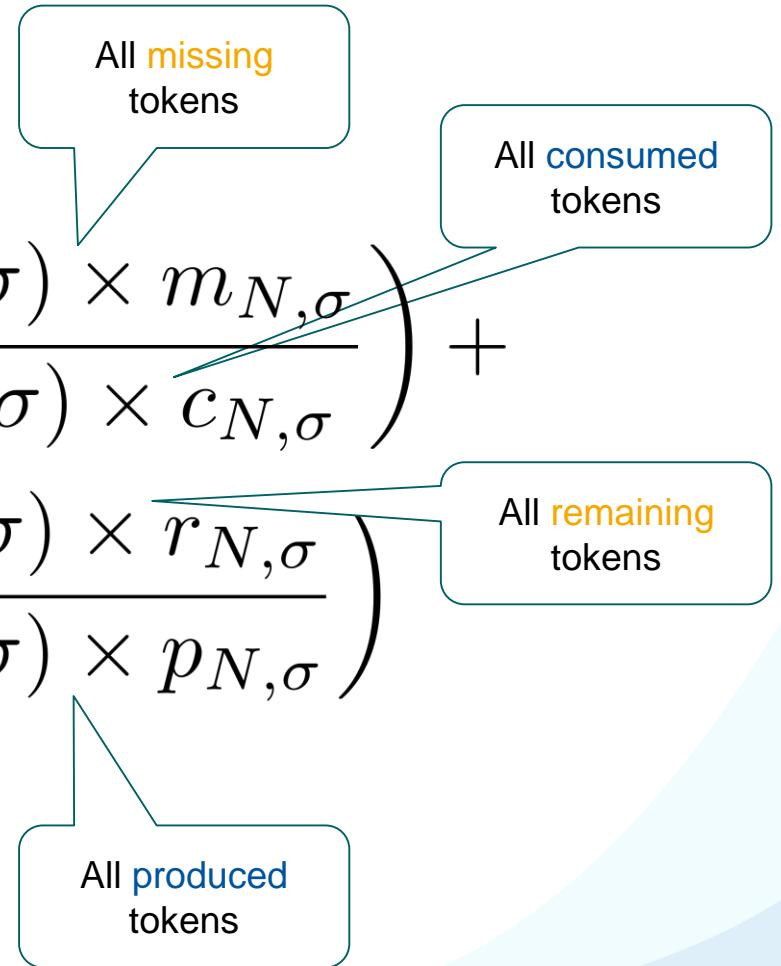
Supervised Process Mining

1. Token-Based Replay
2. Token-Based Replay Examples
3. **Fitness at the Log Level**
4. Generating Supervised Learning Problems



Fitness at the Log Level

$$\text{fitness}(L, N) = \frac{1}{2} \left(1 - \frac{\sum_{\sigma \in L} L(\sigma) \times m_{N,\sigma}}{\sum_{\sigma \in L} L(\sigma) \times c_{N,\sigma}} \right) + \frac{1}{2} \left(1 - \frac{\sum_{\sigma \in L} L(\sigma) \times r_{N,\sigma}}{\sum_{\sigma \in L} L(\sigma) \times p_{N,\sigma}} \right)$$

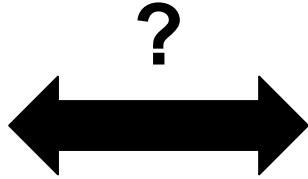


Less scary than it looks:

The **sums** of p , c , m , and r over
all traces in the **entire event log** ...

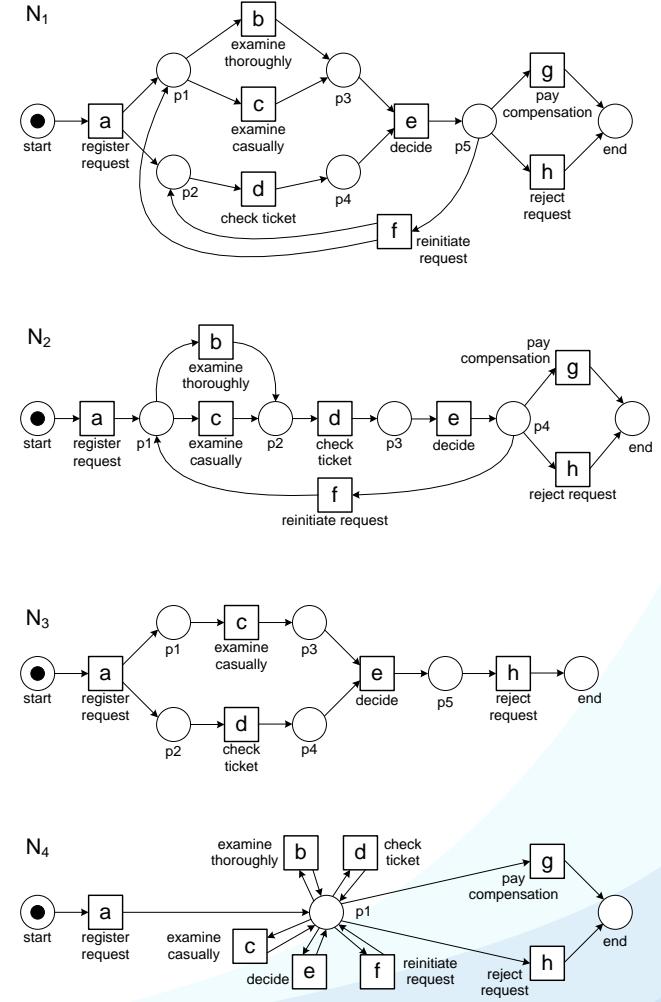
Computing Fitness

#	trace
455	acdeh
191	abdeg
177	adceh
144	abdeh
111	acdeg
82	adceg
56	adbeh
47	acdefdfbeh
38	adbeg
33	acdefbdbeh
14	acdefbddeg
11	acdefdbeg
9	adcefcdbeh
8	adcefdbeh
5	adcefbdeg
3	acdefbdefdbeg
2	adcefdbeg
2	adcefbdefbdeg
1	adcefdbefbdeg
1	adbefbdefdbeg
1	adcefdbefcdefdbeg
1391	



$$\text{fitness } (L, N) = \frac{1}{2} \left(1 - \frac{\sum_{\sigma \in L} L(\sigma) \times m_{N,\sigma}}{\sum_{\sigma \in L} L(\sigma) \times c_{N,\sigma}} \right) + \frac{1}{2} \left(1 - \frac{\sum_{\sigma \in L} L(\sigma) \times r_{N,\sigma}}{\sum_{\sigma \in L} L(\sigma) \times p_{N,\sigma}} \right)$$

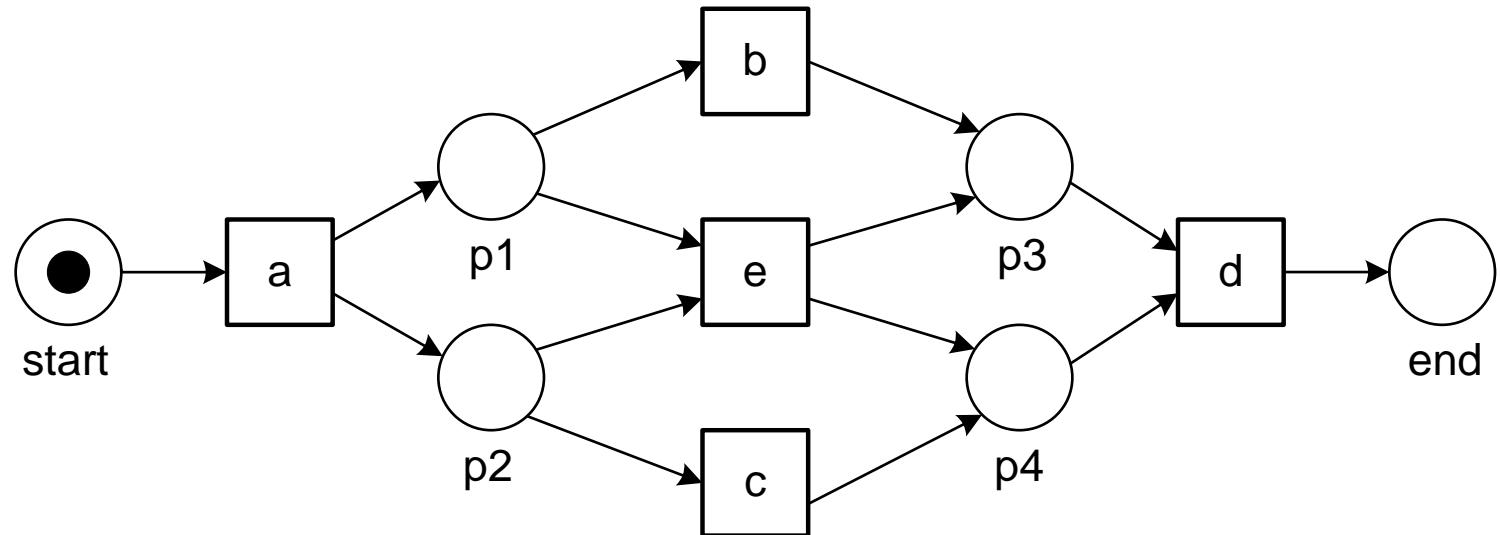
fitness $(L_{\text{full}}, N_1) = 1$
 fitness $(L_{\text{full}}, N_2) = 0.9504$
 fitness $(L_{\text{full}}, N_3) = 0.8797$
 fitness $(L_{\text{full}}, N_4) = 1$



Computing Fitness – Example

Trace	Frequency
abcd	10
acbd	10
aed	10
abd	2
acd	1
ad	1
abbd	1

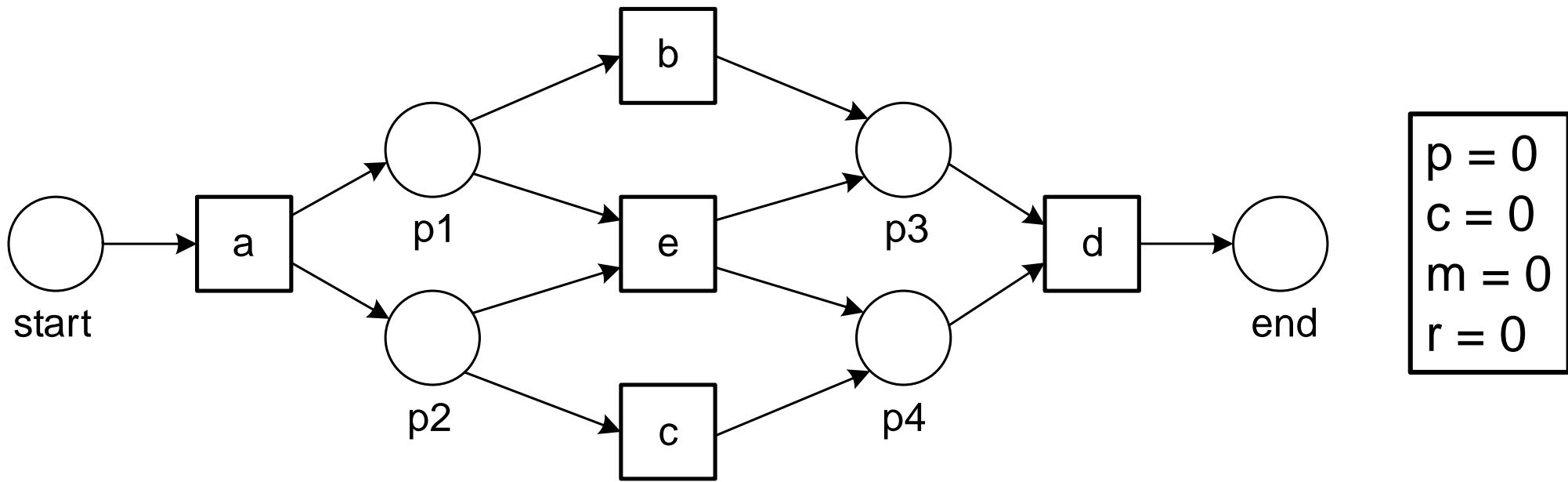
multiset of traces in tabular format



- Consider the event log containing 35 cases
- What is the **fitness** of this process model?

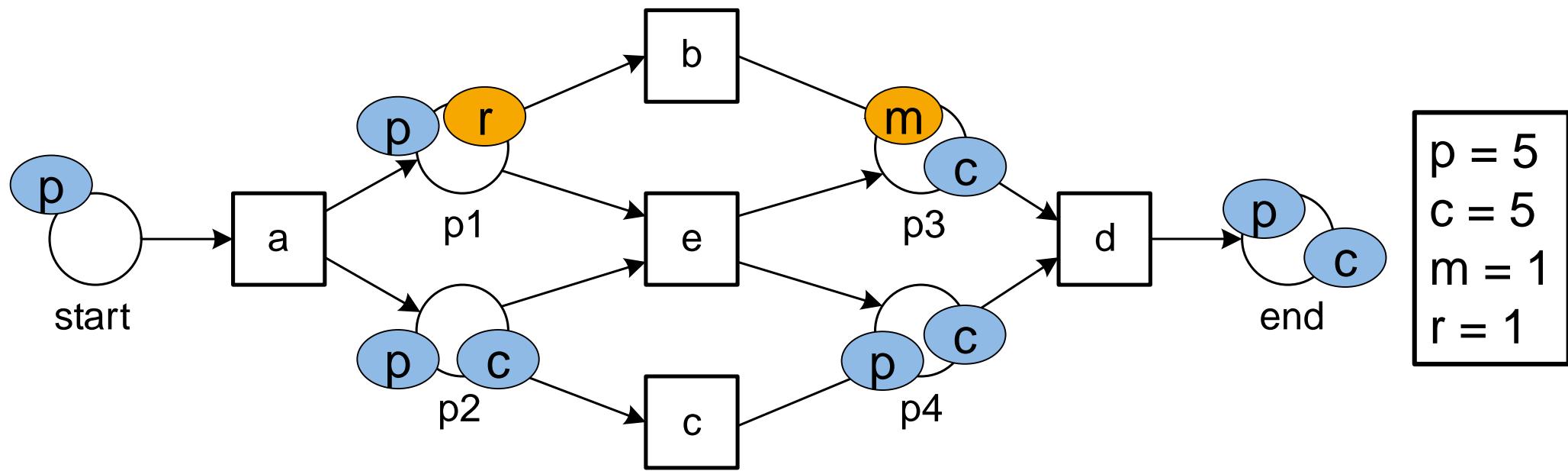
Computing Fitness – Example

Consider Trace **acd** ($\sigma = \langle a, c, d \rangle$)



Computing Fitness – Example

Consider Trace **acd** ($\sigma = \langle a, c, d \rangle$)

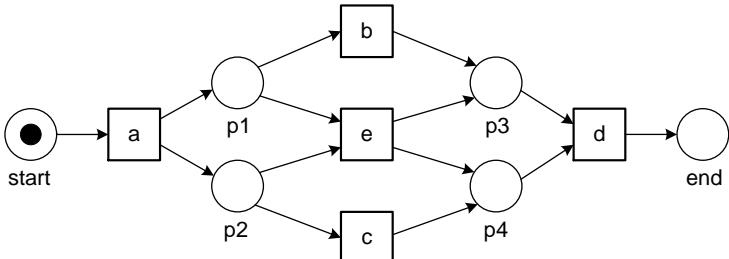


Computing Fitness – Example

Trace	Frequency	Produced (p)	Remaining (r)	Consumed (c)	Missing (m)	Produced (all)	Remaining (all)	Consumed (all)	Missing (all)
abcd	10	6	0	6	0	60	0	60	0
acbd	10	6	0	6	0	60	0	60	0
aed	10	6	0	6	0	60	0	60	0
abd	2	5	1	5	1	10	2	10	2
acd	1	5	1	5	1	5	1	5	1
ad	1	4	2	4	2	4	2	4	2
abbd	1	6	2	6	2	6	2	6	2

acd:

$$\begin{aligned}
 p &= 5 \\
 c &= 5 \\
 m &= 1 \\
 r &= 1
 \end{aligned}$$

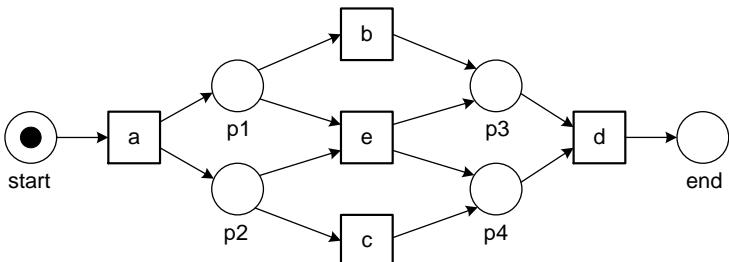


Computing Fitness – Example

Trace	Frequency	Produced (p)	Remaining (r)	Consumed (c)	Missing (m)	Produced (all)	Remaining (all)	Consumed (all)	Missing (all)
abcd	10	6	0	6	0	60	0	60	0
acbd	10	6	0	6	0	60	0	60	0
aed	10	6	0	6	0	60	0	60	0
abd	2	5	1	5	1	10	2	10	2
acd	1	5	1	5	1	5	1	5	1
ad	1	4	2	4	2	4	2	4	2
abbd	1	6	2	6	2	6	2	6	2

acd:

$$\begin{aligned}
 p &= 5 \\
 c &= 5 \\
 m &= 1 \\
 r &= 1
 \end{aligned}$$



Computing Fitness – Example

Trace	Frequency	Produced (p)	Remaining (r)	Consumed (c)	Missing (m)	Produced (all)	Remaining (all)	Consumed (all)	Missing (all)
abcd	10	6	0	6	0	60	0	60	0
acbd	10	6	0	6	0	60	0	60	0
aed	10	6	0	6	0	60	0	60	0
abd	2	5	1	5	1	10	2	10	2
acd	1	5	1	5	1	5	1	5	1
ad	1	4	2	4	2	4	2	4	2
abbd	1	6	2	6	2	6	2	6	2
Sum						205	7	205	7

$$\text{fitness}(L, N) = \frac{1}{2} \left(1 - \frac{\sum_{\sigma \in L} L(\sigma) \times m_{N,\sigma}}{\sum_{\sigma \in L} L(\sigma) \times c_{N,\sigma}} \right) + \frac{1}{2} \left(1 - \frac{\sum_{\sigma \in L} L(\sigma) \times r_{N,\sigma}}{\sum_{\sigma \in L} L(\sigma) \times p_{N,\sigma}} \right)$$

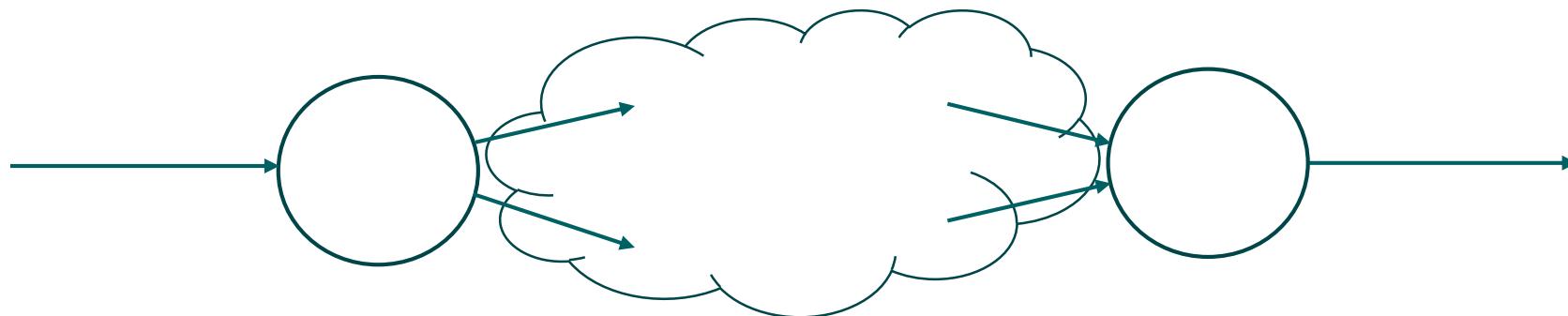
Computing Fitness – Example

Trace	Frequency	Produced (p)	Remaining (r)	Consumed (c)	Missing (m)	Produced (all)	Remaining (all)	Consumed (all)	Missing (all)
abcd	10	6	0	6	0	60	0	60	0
acbd	10	6	0	6	0	60	0	60	0
aed	10	6	0	6	0	60	0	60	0
abd	2	5	1	5	1	10	2	10	2
acd	1	5	1	5	1	5	1	5	1
ad	1	4	2	4	2	4	2	4	2
abbd	1	6	2	6	2	6	2	6	2
Sum						205	7	205	7

$$\begin{aligned}
 \text{fitness}(L, N) &= \frac{1}{2} \left(1 - \frac{\sum_{\sigma \in L} L(\sigma) \times m_{N,\sigma}}{\sum_{\sigma \in L} L(\sigma) \times c_{N,\sigma}} \right) + \frac{1}{2} \left(1 - \frac{\sum_{\sigma \in L} L(\sigma) \times r_{N,\sigma}}{\sum_{\sigma \in L} L(\sigma) \times p_{N,\sigma}} \right) \\
 &= \frac{1}{2} \left(1 - \frac{7}{205} \right) + \frac{1}{2} \left(1 - \frac{7}{205} \right) \approx 0.966
 \end{aligned}$$

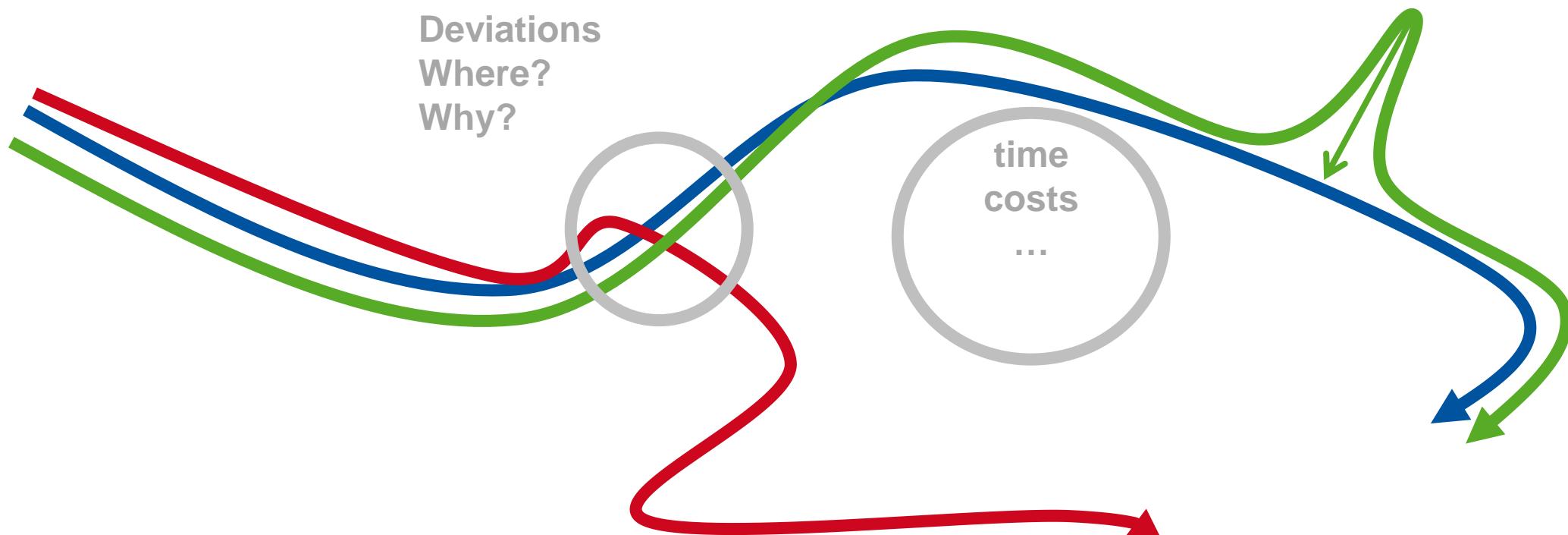
Limitations of Token-Based Approach

- Basic replay approach assumes **visible & uniquely labeled** transitions.
- Most implementations (ProM, PM4Py, Celonis, etc.) use **heuristics** to deal with silent transitions and multiple transitions having the same label
- Conformance values are sometimes **too optimistic**
- Local decision-making may lead to misleading results



Alignments

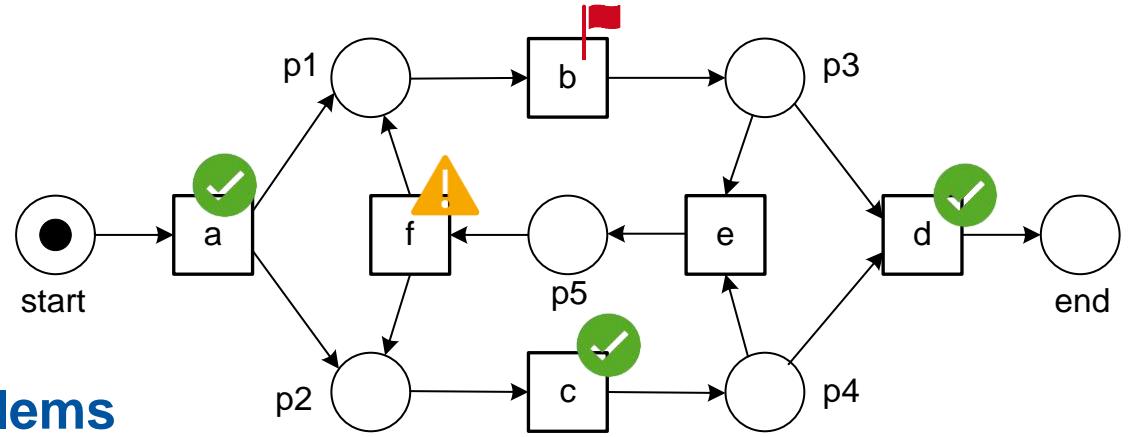
A Better, but More Expensive Way to Check Compliance



- Find the “closest path” in the model
- Outside of the scope of this course

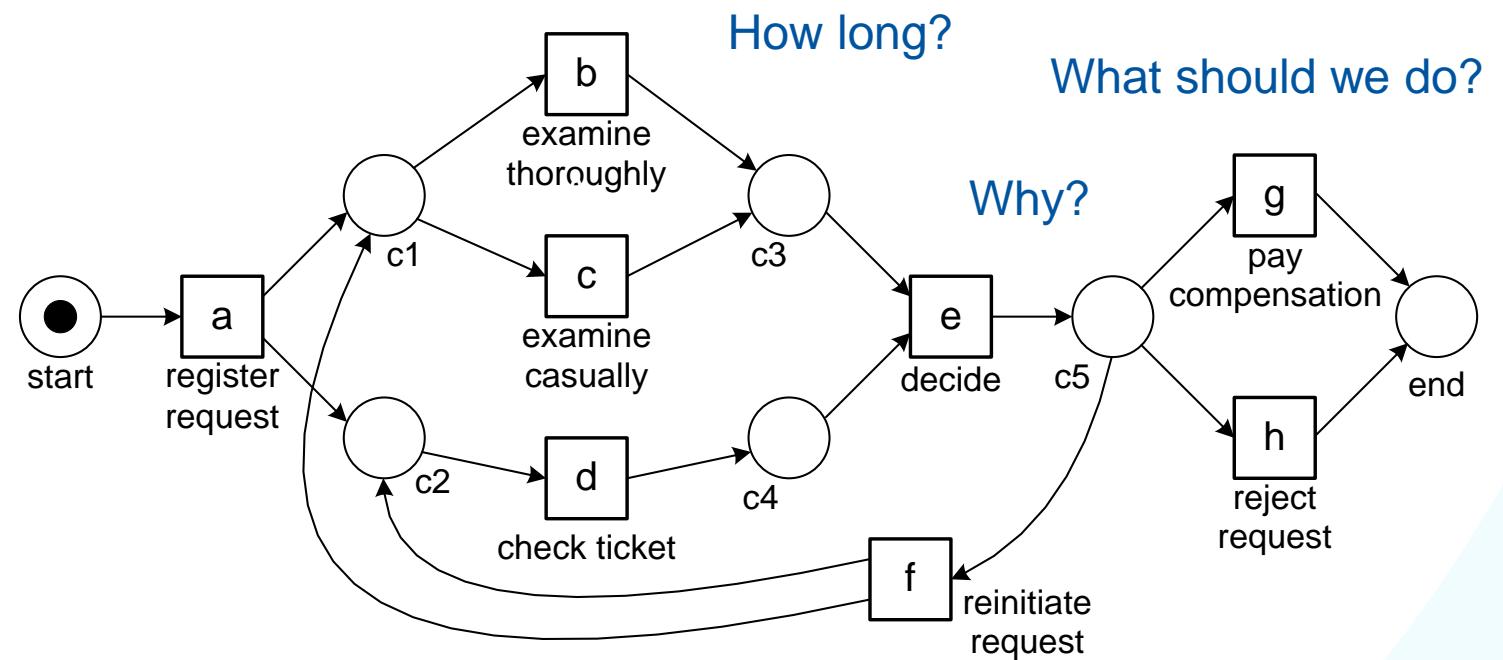
Supervised Process Mining

1. Token-Based Replay
2. Token-Based Replay Examples
3. Fitness at the Log Level
4. **Generating Supervised Learning Problems**

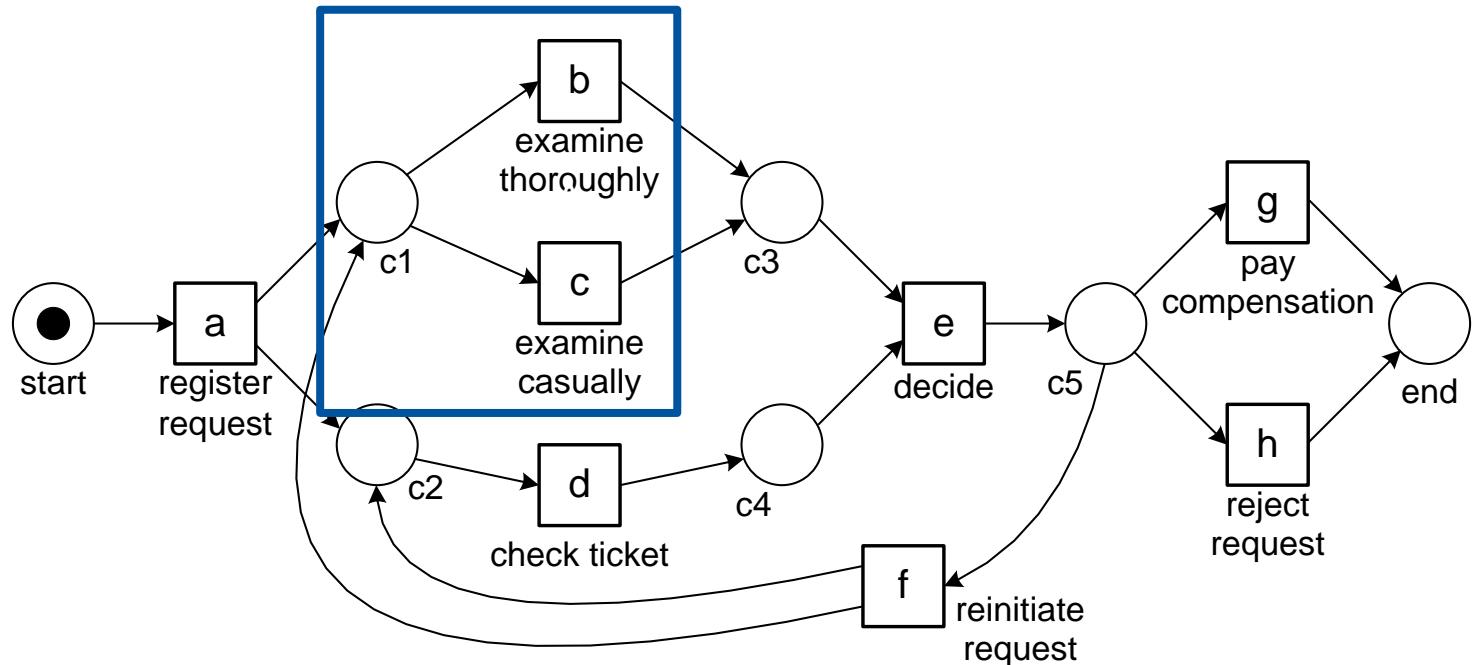


Connection to Machine Learning

- Machine learning and many other data science techniques are **not process-centric**
- Consider an information system with thousands of tables. **How to get started?**
- Process mining can **generate** valuable **machine-learning** problems



Decision Mining

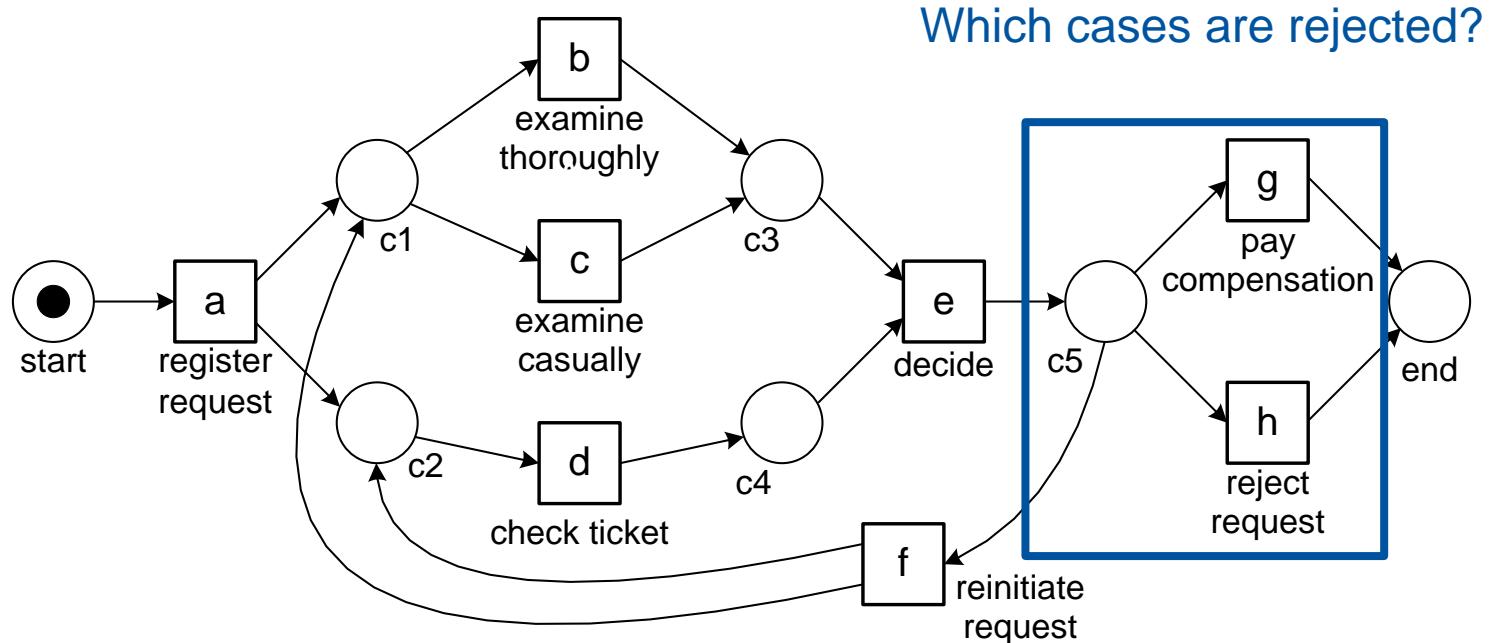


For example:

- Cases handled by John
- Cases handled in January
- Cases that were submitted late
- Cases of new customers
- ...

Which cases require a thorough examination?

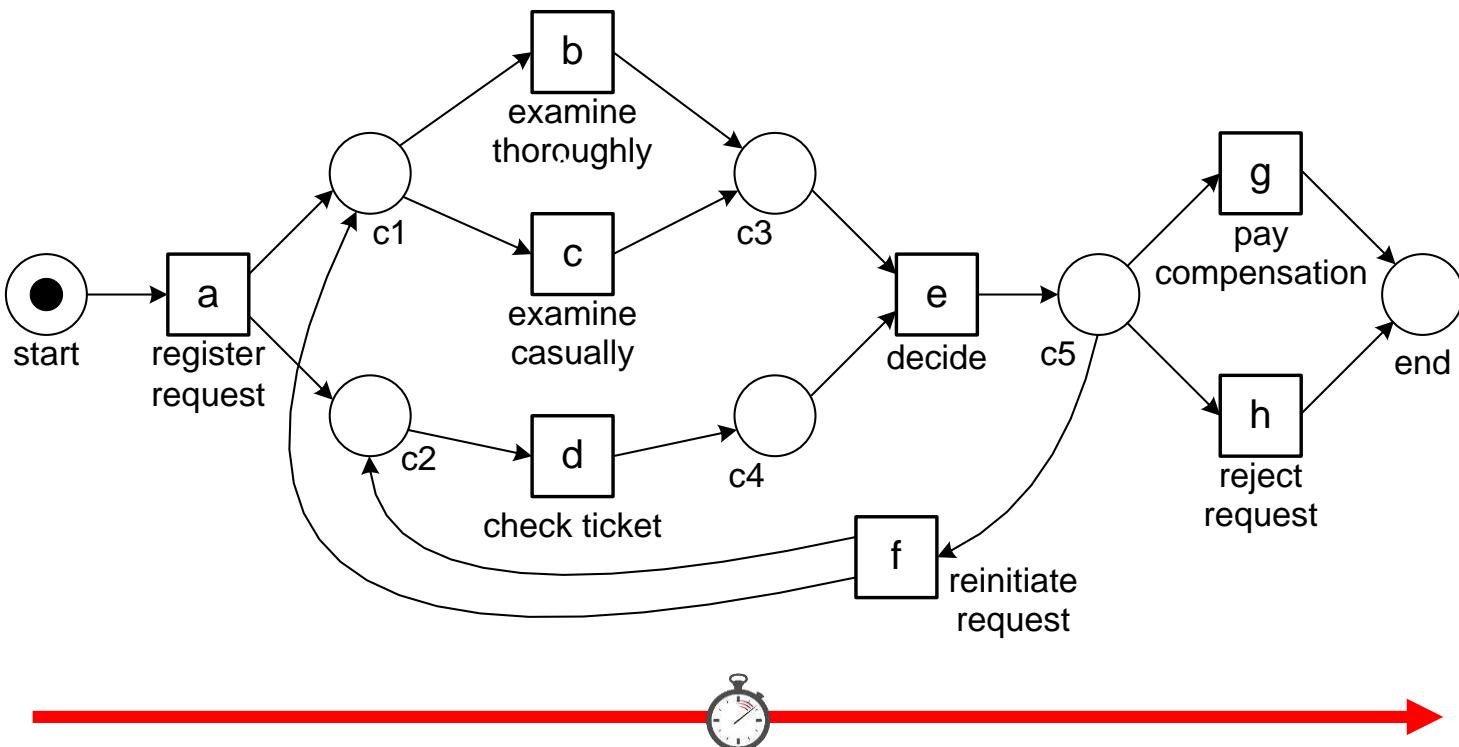
Decision Mining



For example:

- Cases above €500
- Cases that required multiple checks
- Cases that got delayed
- ...

Performance Mining



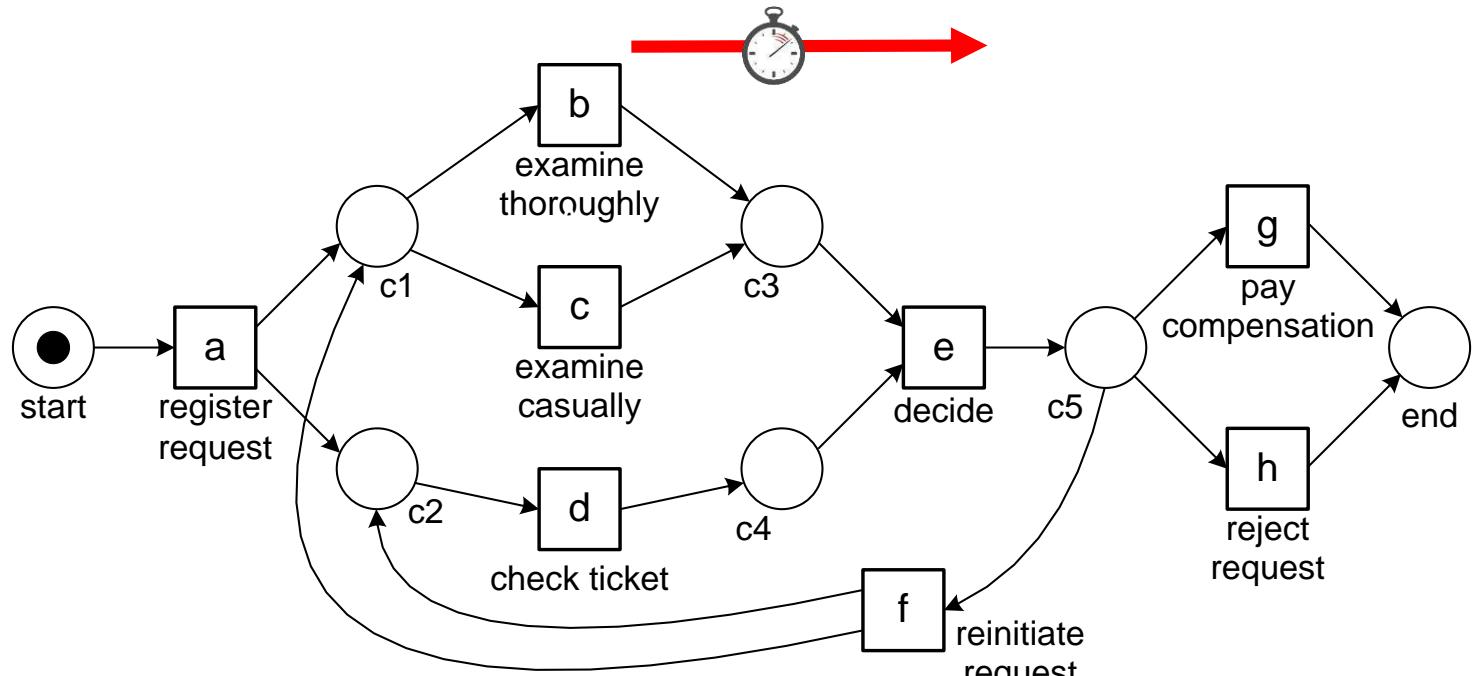
For example:

- Cases handled by Mary
- Cases that required multiple checks
- ...



Which cases took more than two months?

Performance Mining



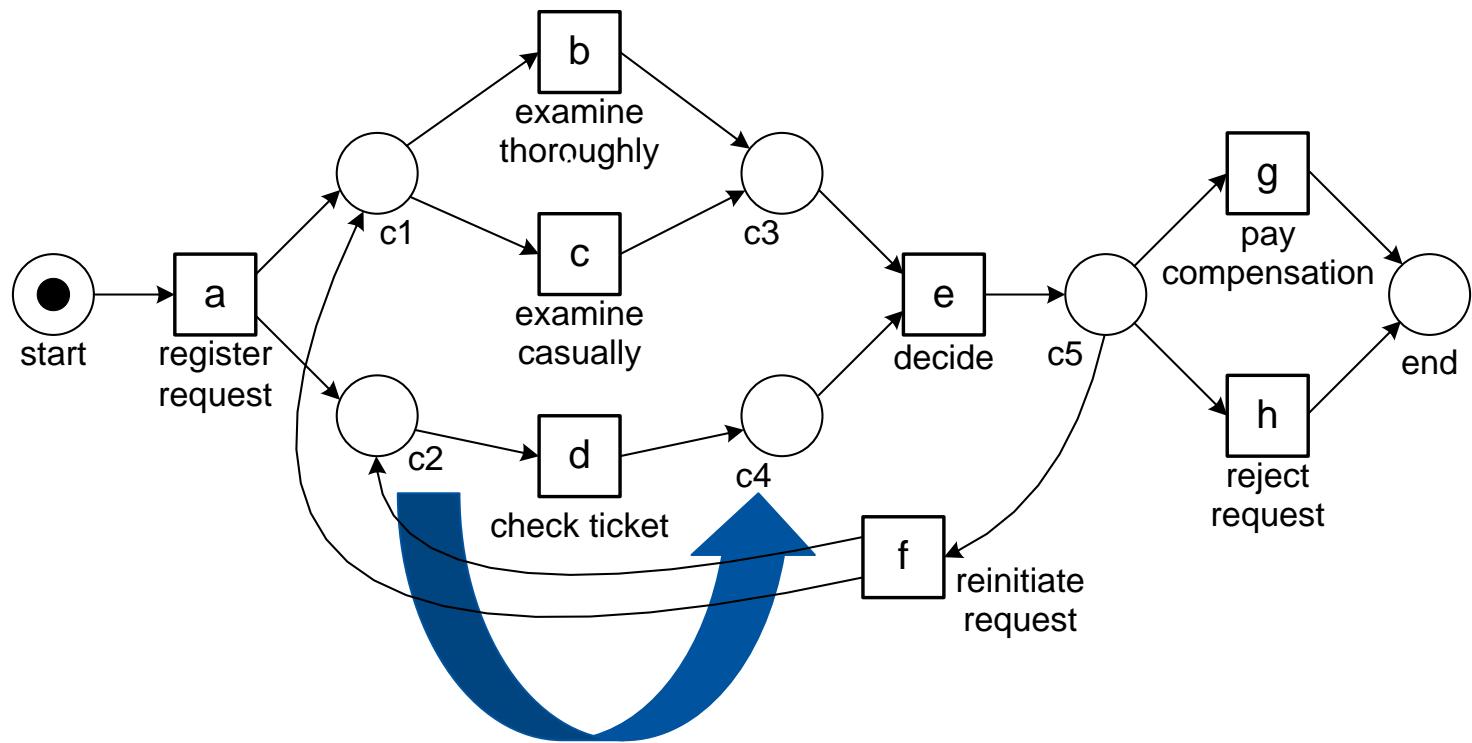
For example:

- A lack of resources due to illness
- An unusual percentage or rework
- ...



What caused the delays in decision making in May?

Deviation Mining

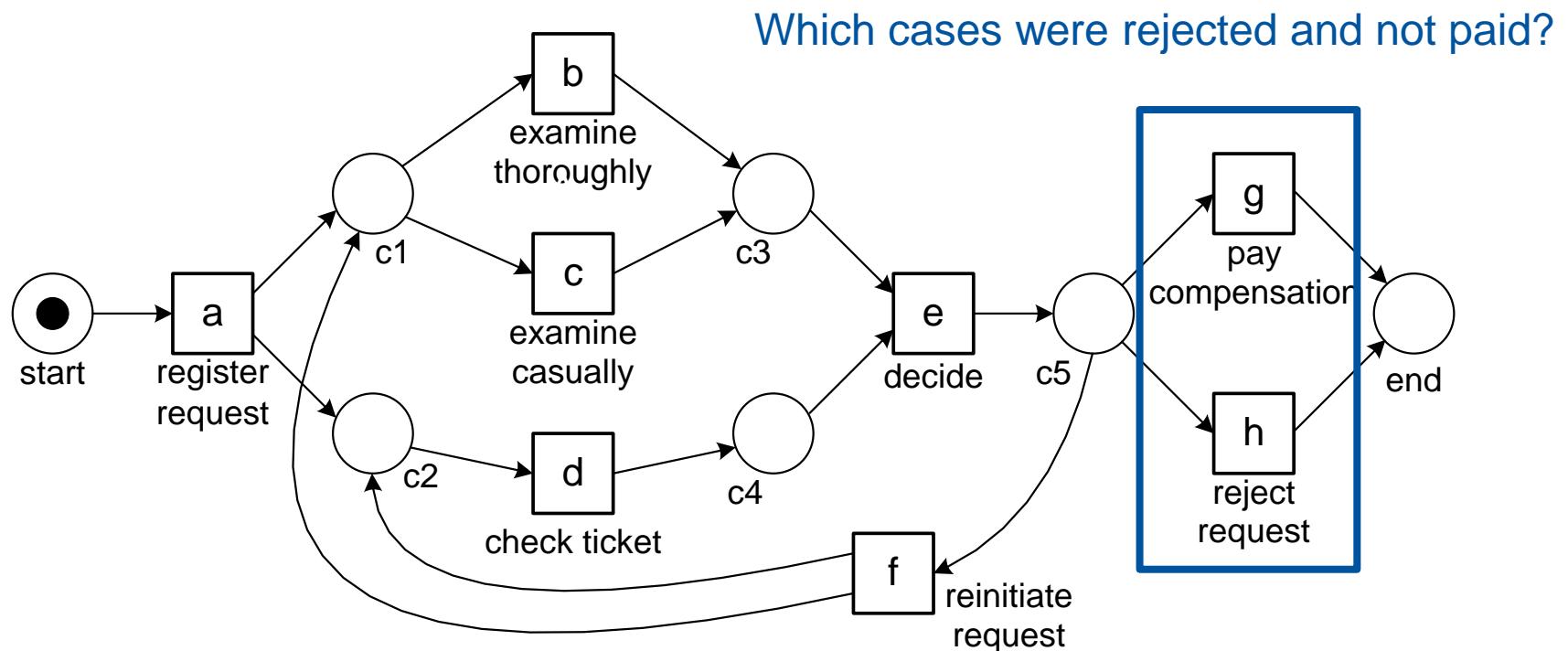


For which cases was the ticket not checked?

For example:

- Cases handled by Mary
- Cases initiated by the downtown office
- ...

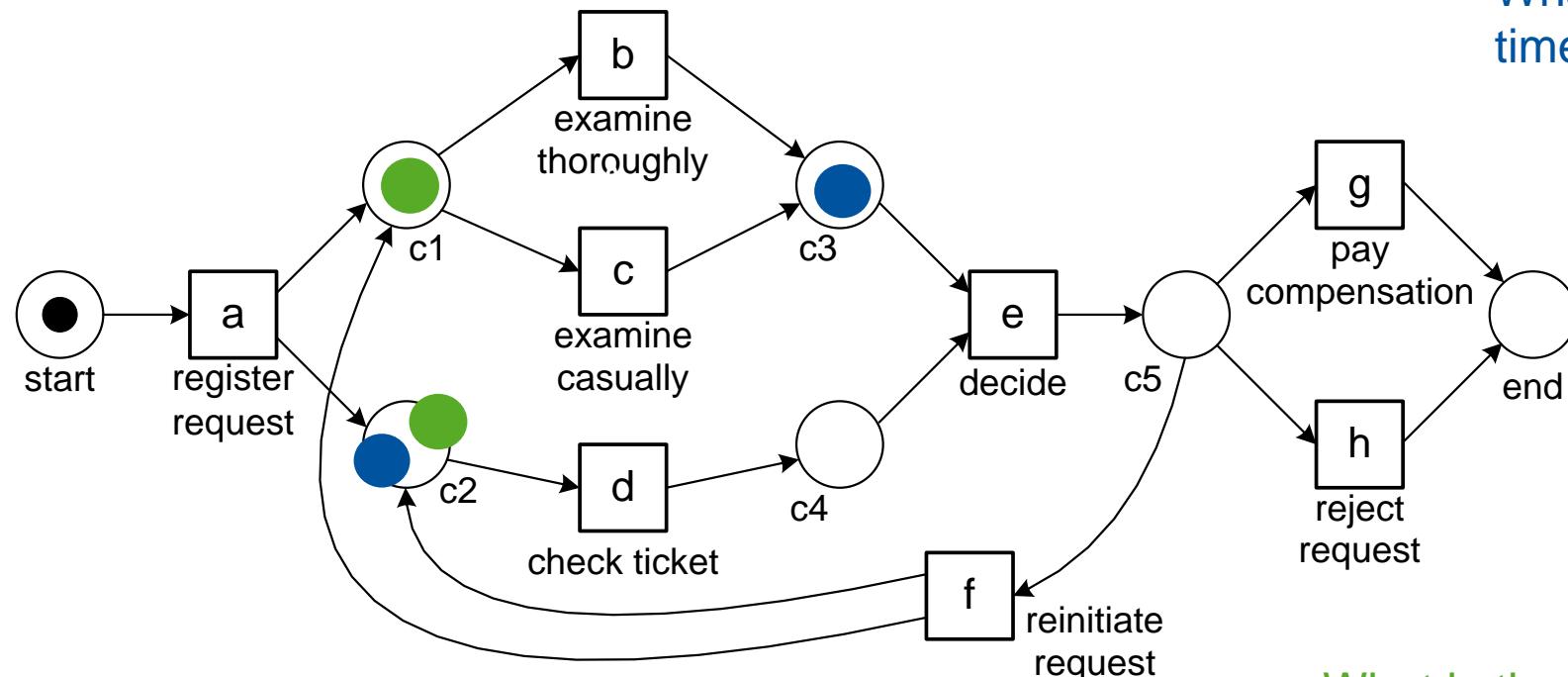
Deviation Mining



For example:

- Cases handled by Pete
- Cases that arrived in June
- ...

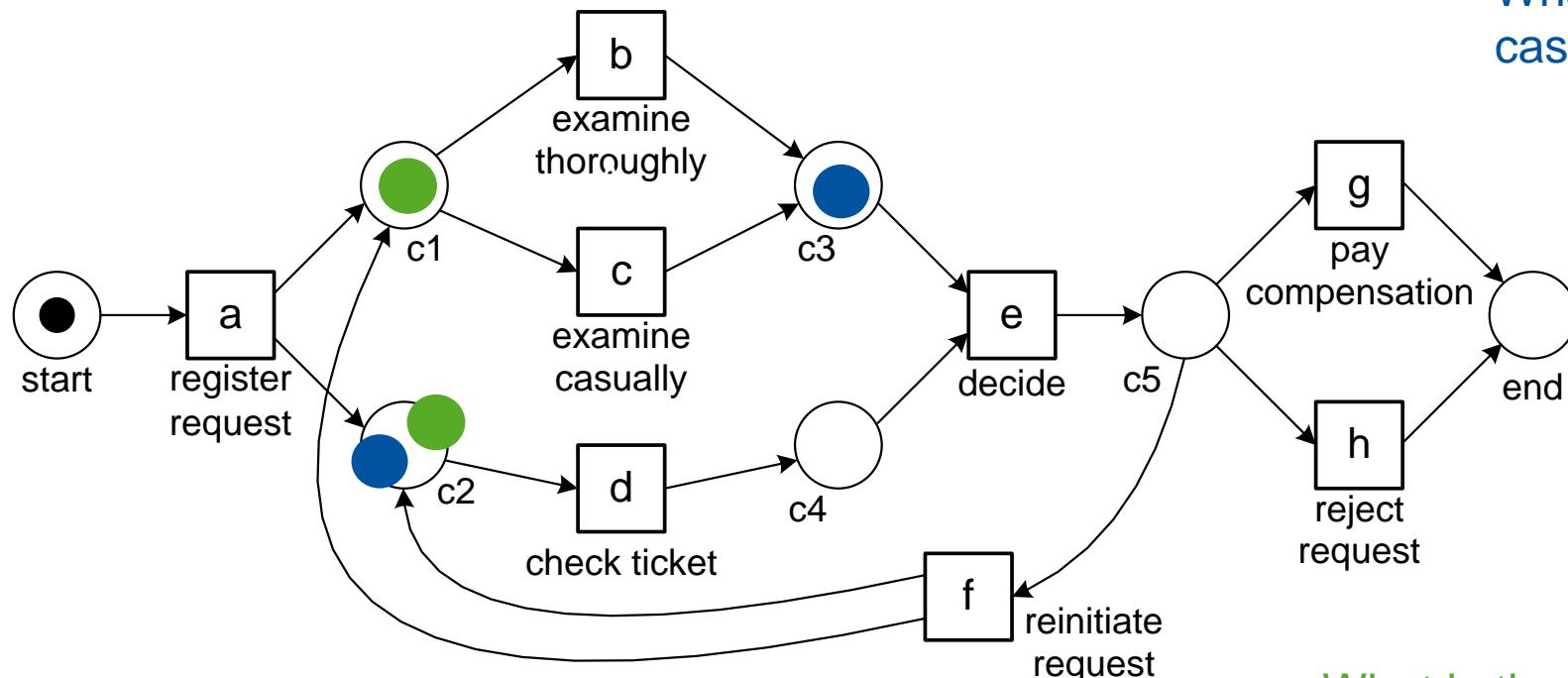
Operational Support Using Process Mining



What is the expected remaining flow time of the blue case?

What is the expected remaining flow time of the green case?

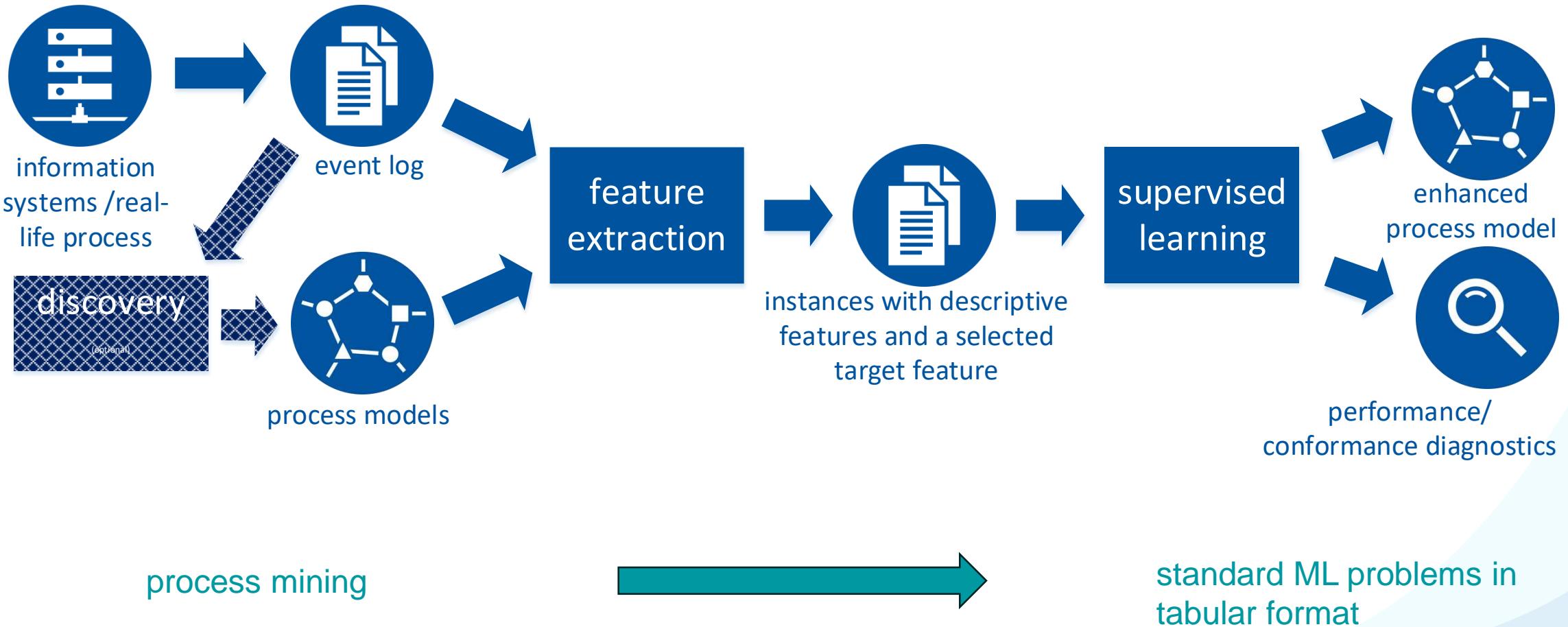
Operational Support Using Process Mining



What is the probability that the blue case will be rejected?

What is the probability that the green case will need two decisions?

General Pattern



Process Mining

- Event data are omnipresent (just like text data or image data).
- A very interdisciplinary field! Connections with traditional **data science**, **process management**, **simulation**, **machine learning**, ...
- We focused on two tasks:
 - **Process Discovery**: obtain a process model from historic **event data**
 - **Conformance Checking**: obtain a measure of **deviation** between **expected** and **actual** behavior
- A relatively young field of research: Many foundational questions are still open, but already widely adopted in industry.

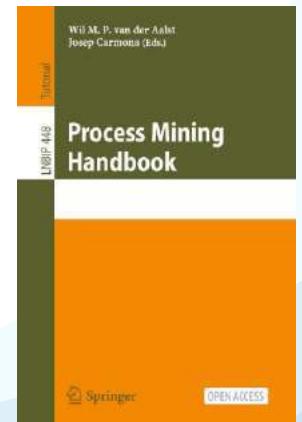
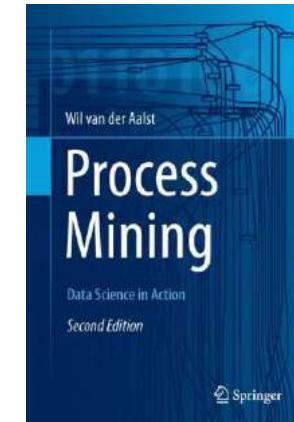
Learn More About Process Mining?

- Consider taking **Business Process Intelligence (BPI)** in the next semester.
- Many opportunities to go deeper, e.g., Advanced Process Mining (APM), seminars, etc.
- Also, we created several **online courses** on Coursera and edX.
- Visit <https://www.pads.rwth-aachen.de/> for thesis projects, etc.



A screenshot of the PADS - Teaching website. The header features the PADS logo and the text "PADS - Teaching". Below the header, there is a navigation menu with links to "ACADEMICS", "RESEARCH", and "THE CHAIR". The main content area shows a photograph of students in a lecture hall. On the left side, there is a sidebar with various links such as "Glossary", "Data Examples", "Signups", "Labs", "Thesis Projects", "Cooperative Themes", "Days Of Oral Defense", "Partnerships", "New Income Learning Course by RWTH and Celonis", "Cooperative Data Science in Action", and "Registration Academic Center of Excellence".

A screenshot of the edX course page for "A Hands-on Introduction to Process Mining". The course is led by Prof. Dr. Wil van der Aalst, the godfather of Process Mining. The page includes a brief description of the course, starting date (08.11.2023), and a QR code for registration. There is also a portrait of Prof. Dr. Wil van der Aalst.



Elements of Machine Learning & Data Science

Text Mining

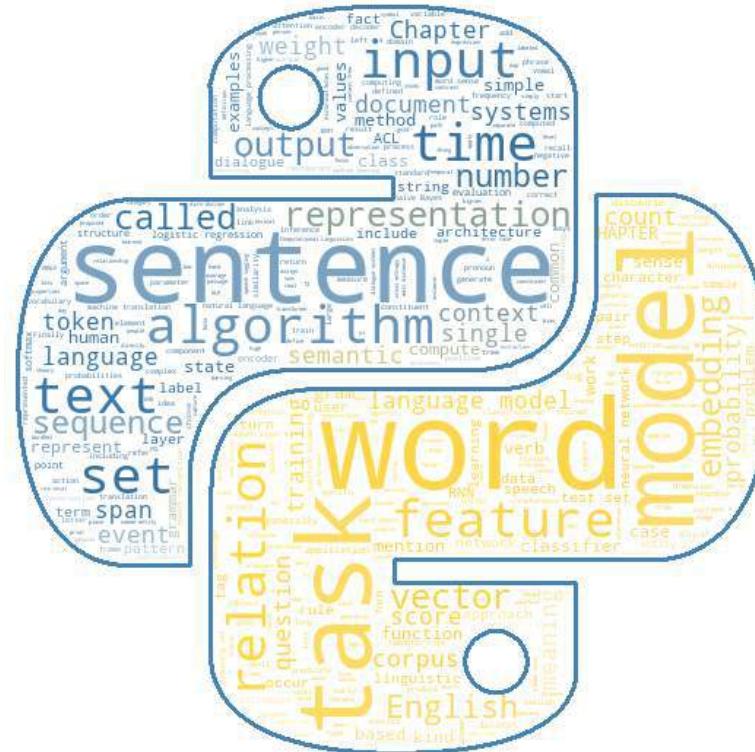
Lecture 21

Prof. Wil van der Aalst

Marco Pegoraro, M.Sc.
Christian Rennert, M.Sc.

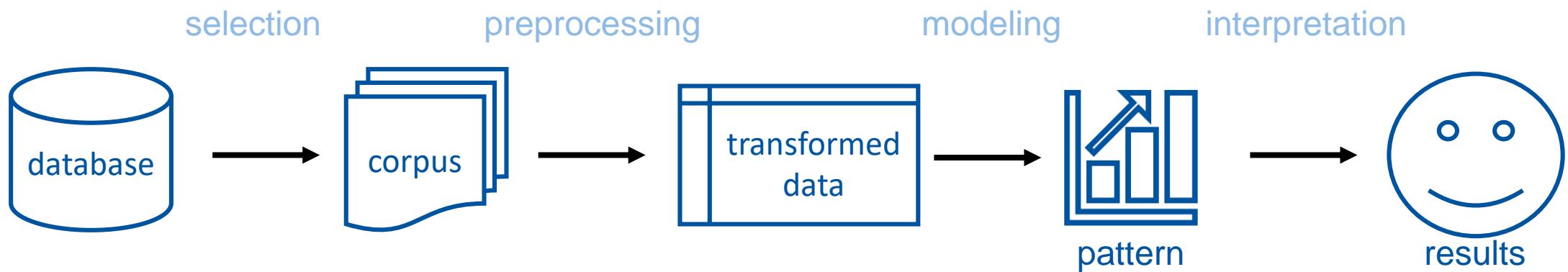
Text Mining

1. Introduction to Text Mining
 2. Text Preprocessing
 3. Modeling
 4. Enriching Text: Linked Data
 5. N-gram Model
 6. Learning a Representation
 7. Word2vec

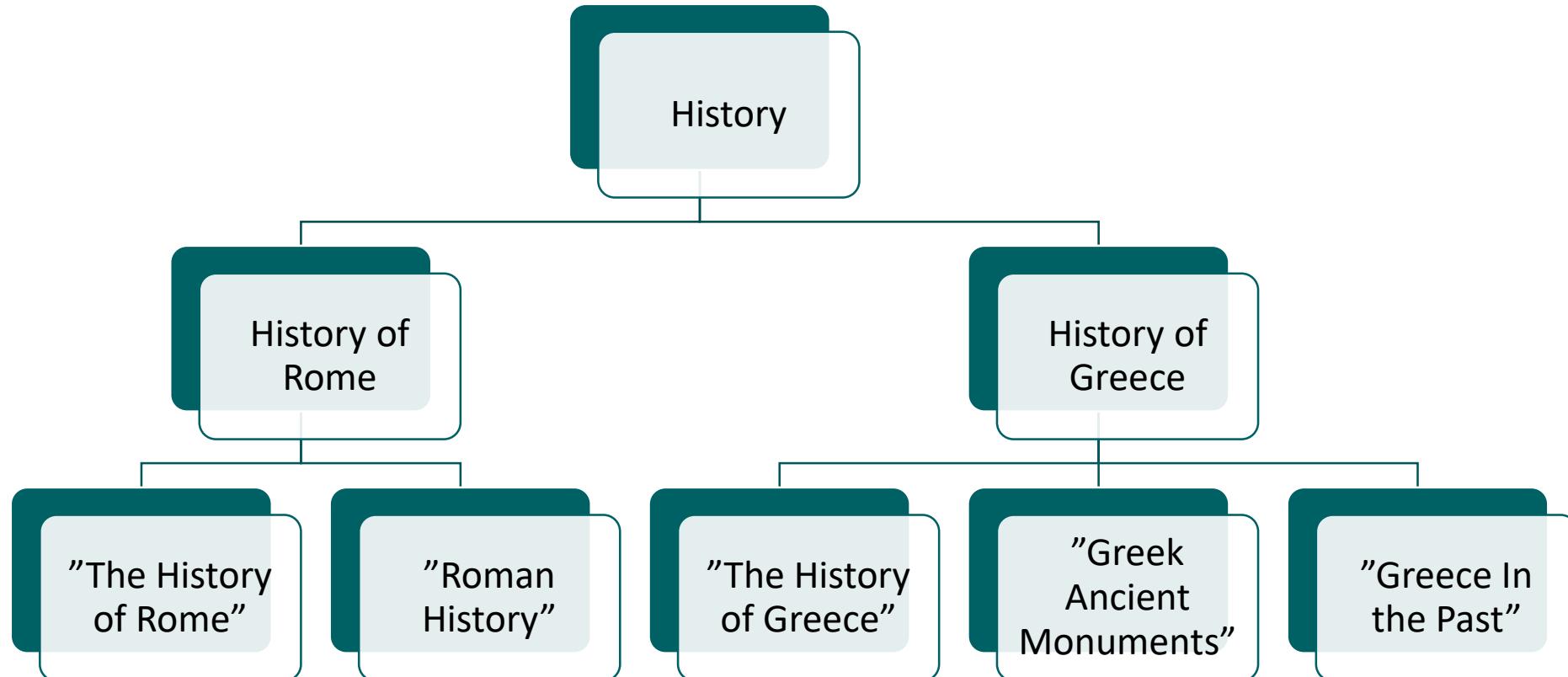


Text Mining Pipeline

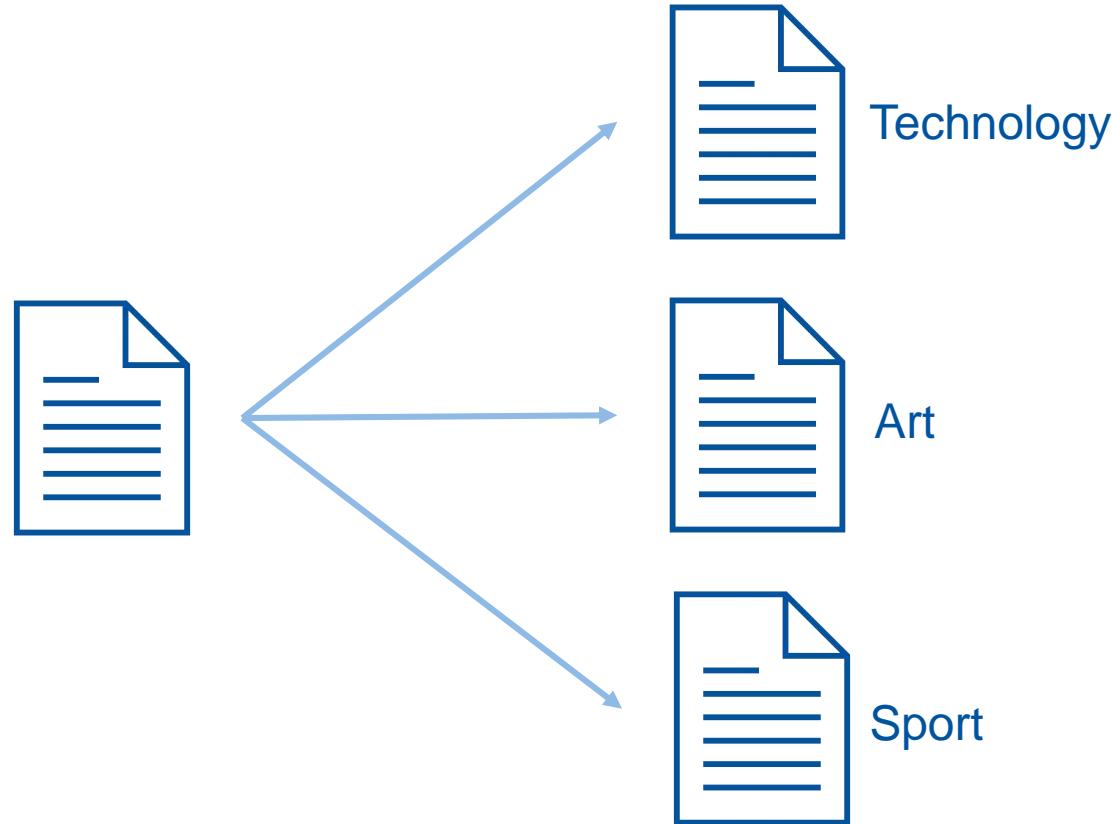
Text mining is the extraction of (structured) knowledge from (unstructured) text



Text Mining Applications – Document Clustering



Text Mining Applications – Document Classification



Text Mining Applications – Named Entity Recognition

Named Entity Recognition (NER)

- recognizes named entities in a text
- classifies them according to their contextual information

"I was at this beach", said **Jenny**. She had been traveling through **Florida** before. "I'm sure that **Max** would love to visit **Miami** as well"

person

place

Text Mining Applications – Sentiment Analysis

Sentiment Analysis:

Identifies positive/negative attitudes through recognition of emotions, opinions or moods

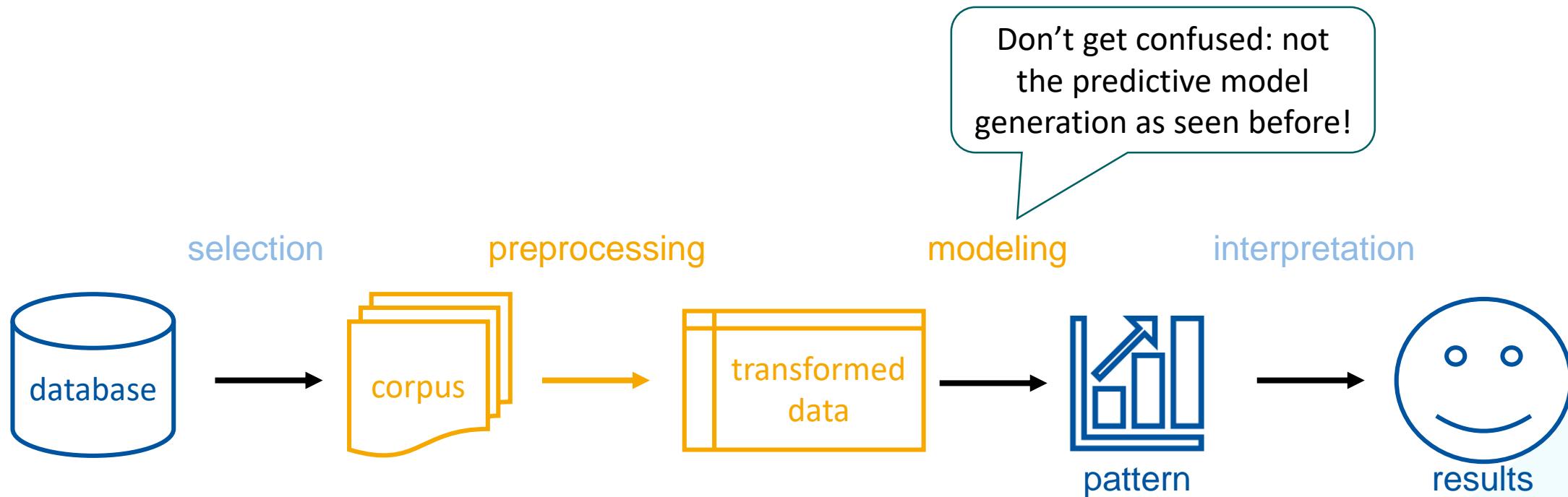
Trust	"Amazing news for Bitcoin!"
Fear	"Suspicious assault in Berlin"
Surprise	"What happened with the President of the US?"

Text Mining Applications – Other Examples

- Part-Of-Speech (POS) tagging – label words with their corresponding part of a speech (noun, verb, adjective, etc.)
- Coreference resolution – identification of words and expressions that refer to the same entity in a text (Max – he, Jenny – her, etc.)
- Keyword extraction – automatic identification of significant terms in a text
- Machine translation – automatic translation from one language to another

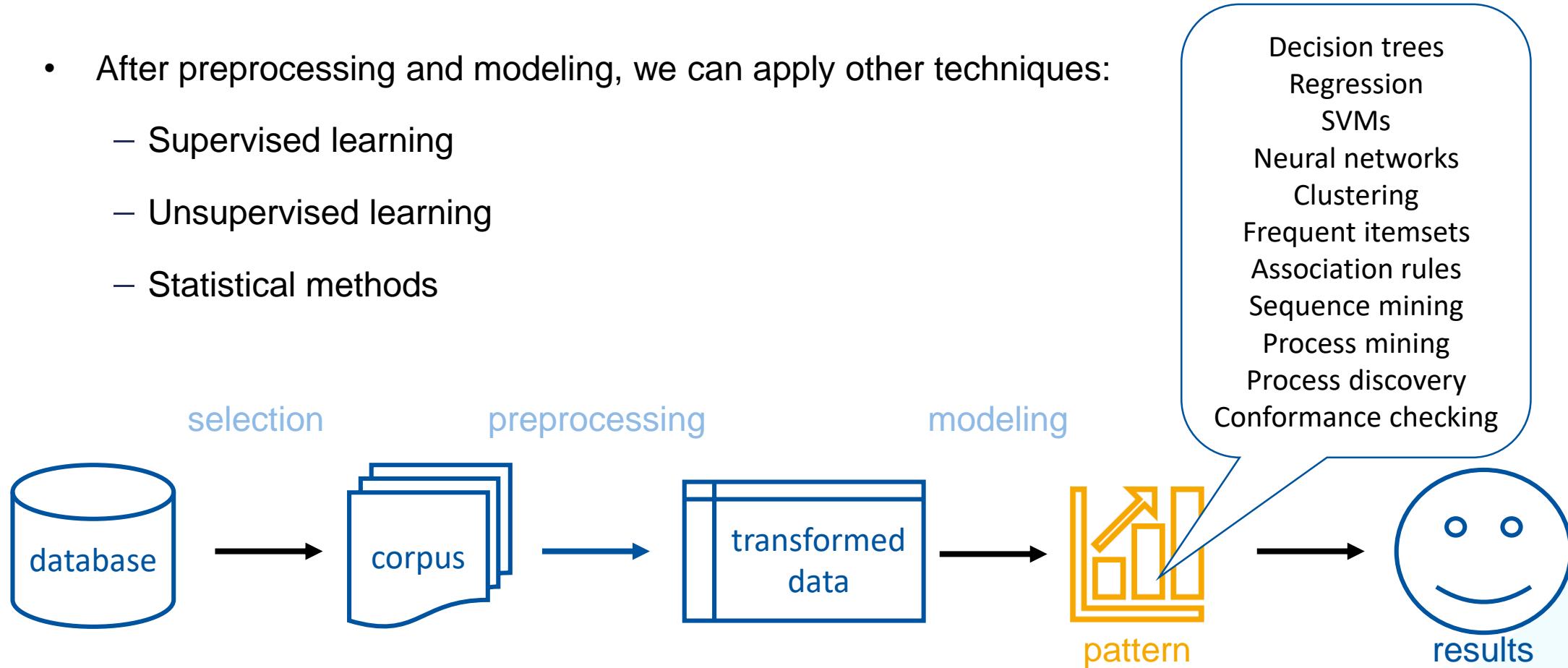
Important steps

It's crucial to preprocess and model the text properly



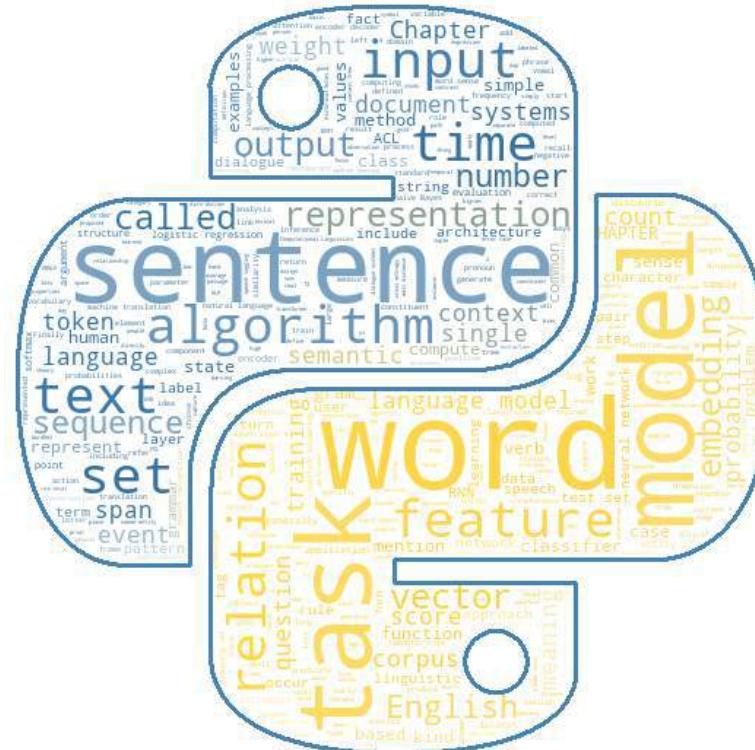
Important steps

- After preprocessing and modeling, we can apply other techniques:
 - Supervised learning
 - Unsupervised learning
 - Statistical methods



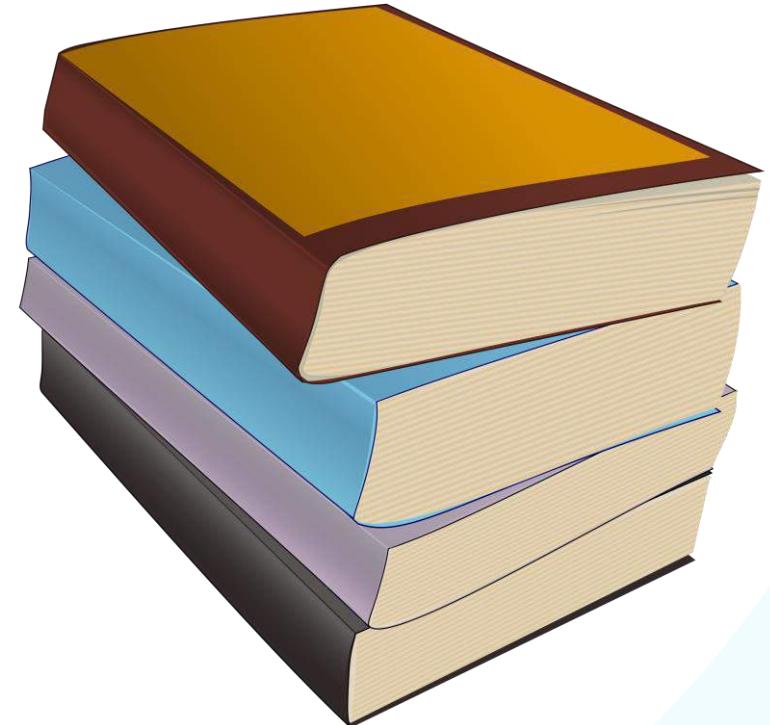
Text Mining

1. Introduction to Text Mining
 2. **Text Preprocessing**
 3. Modeling
 4. Enriching Text: Linked Data
 5. N-gram Model
 6. Learning a Representation
 7. Word2vec



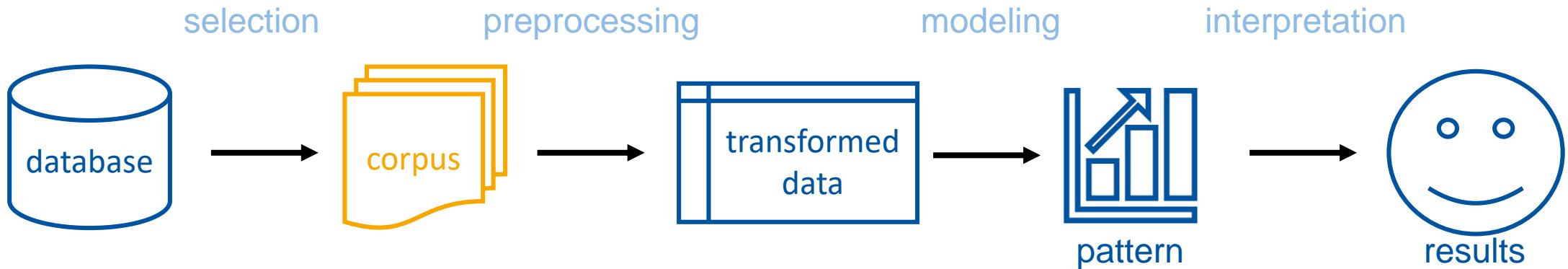
Structuring Text

- Challenge: from unstructured data (text) to structured data (ideally numbers)
- Texts are usually unstructured:
 - Do we consider sentences or words?
 - Lengths of single units of information vary.
 - What are text features and instances?



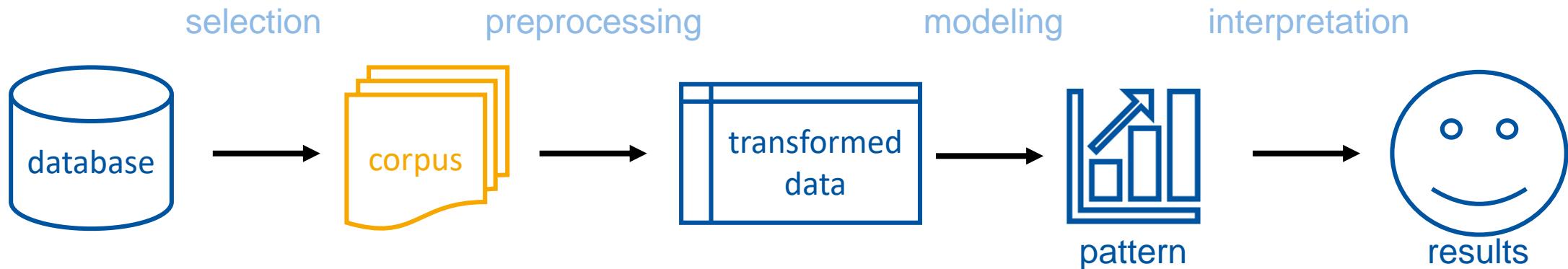
Corpus

Text mining pipeline – Step 1: Extract a [corpus](#)



Corpus

Text mining pipeline – Step 1: Extract a **corpus**



- A **corpus** is a collection of pieces of a text
- **Pieces** can be words, sentences, paragraphs, tweets, posts, ...
- Pieces in the corpus are often called **documents** (regardless of their nature and size)

Annotated Corpus

- Annotated corpus: units (or fractions of units) of a text are annotated with additional information in order to work as a training set for a specific application
- Corpora are usually annotated by hand

Annotated Corpus

for domain-specific (medical) named entity recognition

age

gender

medical history

medical history

"Patient is a 55-year-old male with a history of hypertension and type 2 diabetes mellitus. On physical

anatomical location

examination, there was mild tenderness over the right upper quadrant and laboratory testing revealed

laboratory findings

elevated liver enzymes. Based on these findings, the patient was diagnosed with non-alcoholic fatty

diagnosis

liver disease and started on a treatment plan consisting of lifestyle modifications and medication."

Preprocessing

Text mining pipeline – Step 2: the corpus goes through the preprocessing



Preprocessing

Text mining pipeline – Step 2: the corpus goes through the preprocessing



Presented Techniques:

- Tokenization
- Stop word removal
- Token normalization: **stemming** or **lemmatization**

Tokenization

- Splitting the text into smaller units: **tokens**
- Usually tokens are words (word tokenization)
- Could also be characters, ideograms, phonemes, syllables, sentences, phrases, clauses, ...

Word Tokenization

Easy! Just split after spaces, right?

“He’s been talking to Bill de Blasio, the 109th New York City mayor.”

‘He’s’, ‘been’, ‘talking’ ‘to’ ‘Bill’ ‘de’ ‘Blasio,’ ‘the’, ‘109th’, ‘New’, ‘York’, ‘City’, ‘mayor.’

Word Tokenization

Not always that easy...

“He’s been talking to Bill de Blasio, the 109th New York City mayor.”

This one token should actually be 2



Tokenization Is Not Trivial

- Other languages differ from English and can have 40- or even 60-letter words that are a combination of a few single words
- Example: German
Is this one token?



Tokenization Is Not Trivial

- Other languages differ from English and can have 40- or even 60-letter words that are a combination of a few single words
- There exists specific software to tokenize
- Usually tokenization has to be specifically designed ad-hoc for a certain task
- Tokenization is application-dependent



Stop Word Removal

- Removing words that are not informative
- A stop list commonly contains:
 - "to be" and "to have" verbs: is, are, am, was, have, has, ...
 - articles: the, a, an, ...
 - auxiliary verbs: will, should, would, shall, must, ...
 - prepositions: in, to, from, of, by, on, ...
 - interrogative words: who, what, which, where, when, how, why, ...
- Stop lists are language-dependent

Stop Word Removal

Why do we remove stop words?

What happens if we compare the following two sentences by counting the number of identical words?

“The cats, which were seven, started to climb the tree.”

“The Saxons, which were outnumbered, started to prepare the siege.”

Stop Word Removal

Why do we remove stop words?

6/10 words are the same! → These sentences should be very similar.

“The cats, which were seven, started to climb the tree.”

“The Saxons, which were outnumbered, started to prepare the siege.”

Stop Word Removal

Why do we remove stop words?

However, 5 out of 6 common words are common stop words...

“The cats, which were seven, started to climb the tree.”

“The Saxons, which were outnumbered, started to prepare the siege.”

Stop Word Removal

Why do we remove stop words?

- After removing stop words, sentences are not so similar anymore
- We went from 60% overlap to 10% overlap

“~~The~~ cats, ~~which were~~ seven, started ~~to~~ climb ~~the~~ tree.”

“~~The~~ Saxons, ~~which were~~ outnumbered, started ~~to~~ prepare ~~the~~ siege.”

Stop Word Removal

- Be careful about what you remove!
- Assuming our previous stop list:

The Who's seventh studio album is titled "The Who by Numbers"



seventh studio album titled Numbers

Stop Word Removal

- Designing a good stop list is **not** a trivial task
- Deciding which words should be stop words depends on the **context/goal**
- Stop lists can be **domain-dependent**
 - Example: in a healthcare domain, "patient" and "hospital" could be stop words

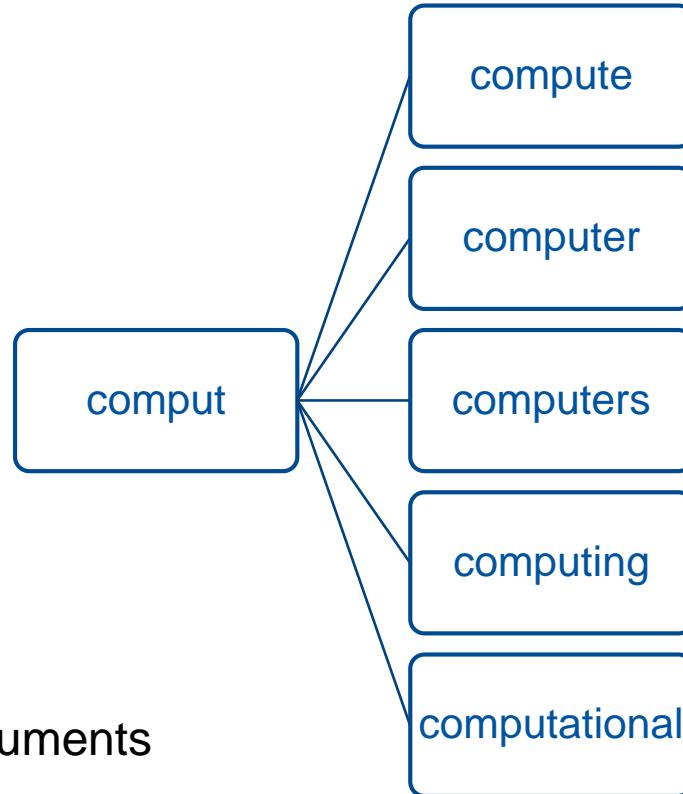


Token Normalization

- **Goal:** Transforming tokens to make them comparable
- **Main forms:**
 - Stemming
 - Lemmatization
- Other forms:
 - Case-folding – converting everything into lowercase
 - Alternative spelling (color/colour)
 - Transliterations
 - ä → ae, ö → oe

Stemming

- Reducing words to their word stem (base or root form)
- Works *most* of the time
- A stem does not need to be a word
- Usually a word stem carries the meaning
- Stemming algorithms can be **aggressive** or **conservative**
 - **Aggressive stemmers** focus on similarities between documents
 - **Conservative stemmers** carefully consider differences



Stemming - Example

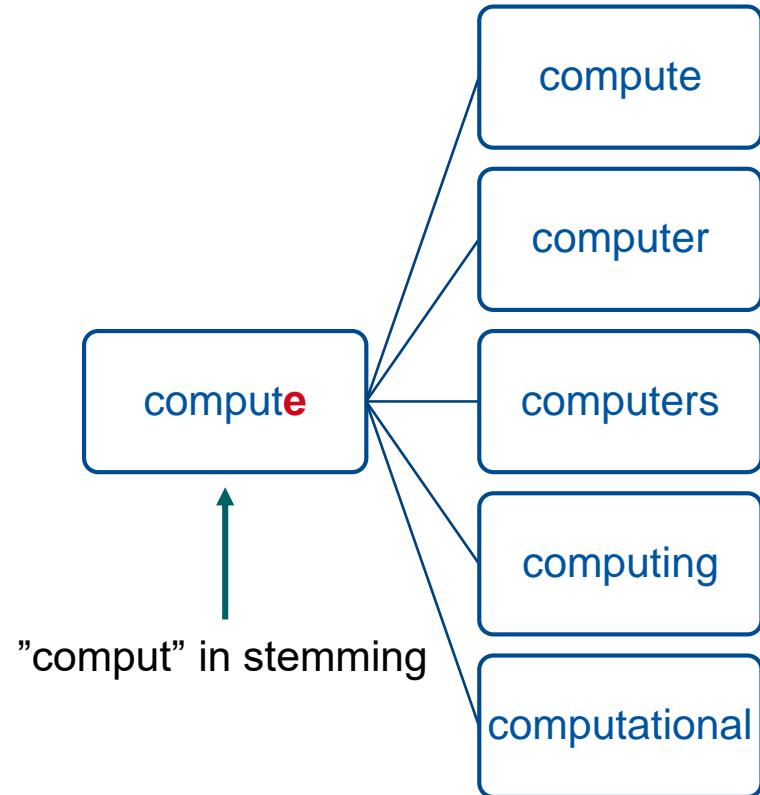
" As I walked through the dark forest, the rustling of leaves under my feet echoed through the silent night. I could feel the chill of the autumn air seeping through my jacket, and the moonlight shone down in a pale glow. Suddenly, a twig snapped, and I froze. Was someone or something watching me? I held my breath, waiting for a sign, but all I heard was the sound of my own heartbeat, thumping in my ears. Slowly, I continued on, my nerves on edge, unsure of what lays ahead."



as i walk through the dark forest the rustl of leav under my feet echo through the silent night i could feel the chill of the autumn air seep through my jacket and the moonlight shone down in a pale glow sudden a twig snap and i froze was someon or someth watch me i held my breath wait for a sign but all i heard was the sound of my own heartbeat thump in my ear slowli i continu on my nerv on edg unsur of what lay ahead

Lemmatization

- Lemmatization applies vocabulary and morphological analysis
- Goal:
 - To remove inflectional endings only
 - To return the base or dictionary form of a word
 - This form is known as the lemma
- More sophisticated than stemming
- Often rule-based (every language has exceptions)



Lemmatization - Example

" As I walked through the dark forest, the rustling of leaves under my feet echoed through the silent night. I could feel the chill of the autumn air seeping through my jacket, and the moonlight shone down in a pale glow. Suddenly, a twig snapped, and I froze. Was someone or something watching me? I held my breath, waiting for a sign, but all I heard was the sound of my own heartbeat, thumping in my ears. Slowly, I continued on, my nerves on edge, unsure of what lay ahead."



as I walk through the dark forest the rustle of leaf under my foot echo through the silent night I could feel the chill of the autumn air seep through my jacket and the moonlight shine down in a pale glow suddenly a twig snap and I freeze be someone or something watch me I hold my breath wait for a sign but all I hear be the sound of my own heartbeat thump in my ear slowly I continue on my nerve on edge unsure of what lay ahead

Stemming vs Lemmatization

as i walk through the dark forest the **rustl** of leav
under my **feet** echo through the silent night i
could feel the chill of the autumn air seep through
my jacket and the moonlight **shone** down in a
pale glow **sudden** a twig snap and i **froze** **was**
someon or **someth** watch me i **held** my breath
wait for a sign but all i heard **was** the sound of my
own heartbeat thump in my ear **slowli** i **continu** on
my **nerv** on **edg unsur** of what lay ahead

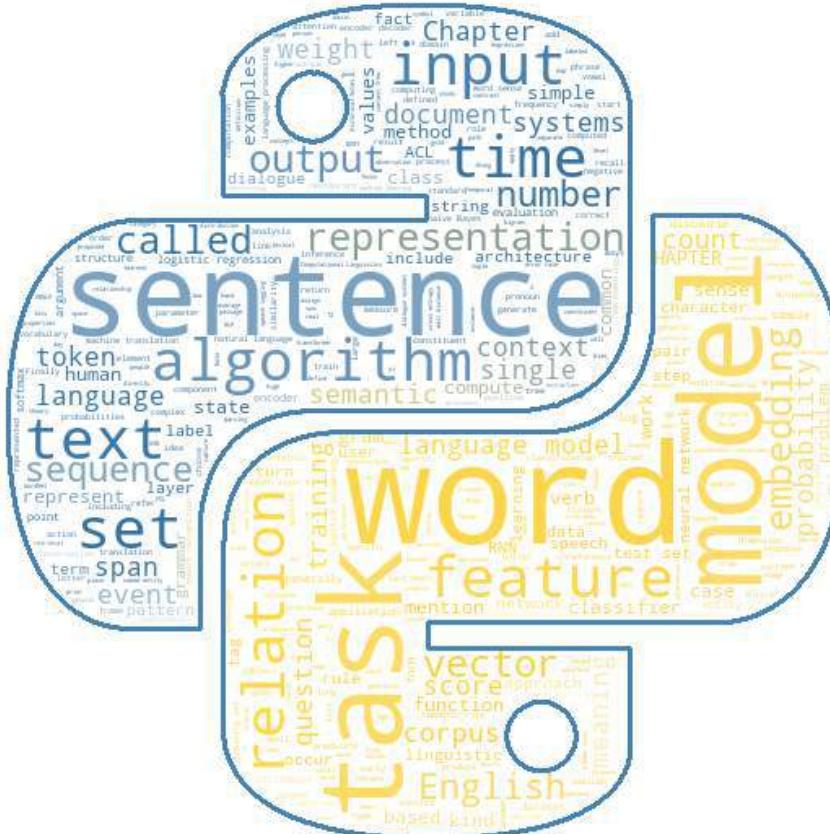
as I walk through the dark forest the **rustle** of **leaf**
under my **foot** echo through the silent night I
could feel the chill of the autumn air seep through
my jacket and the moonlight **shine** down in a pale
glow **suddenly** a twig snap and I **freeze** **be**
someone or **something** watch me I **hold** my
breath wait for a sign but all I hear **be** the sound
of my own heartbeat thump in my ear **slowly** I
continue on my **nerve** on **edge** **unsure** of what lay
ahead

Other Challenges

- Prepositional attachment (e.g., what does a prepositional phrase “with” refer to)
“eating spaghetti **with** chopsticks” vs “eating spaghetti **with** meatballs”
- Anaphora resolution (an expression whose interpretation depends on another expression)
“He convinced his roommate to buy a TV for **himself**”
- Other hidden information
“He **quit smoking**” → he used to smoke
- Homonyms (same word can have different meanings, be a noun or verb, etc.)
sign, firm, tie, watch, tear, bark...

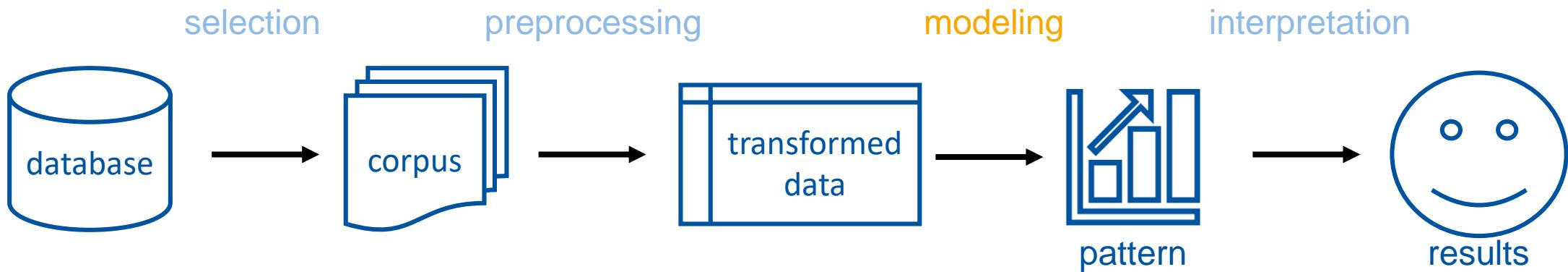
Text Mining

1. Introduction to Text Mining
 2. Text Preprocessing
 3. **Modeling**
 4. Enriching Text: Linked Data
 5. N-gram Model
 6. Learning a Representation
 7. Word2vec



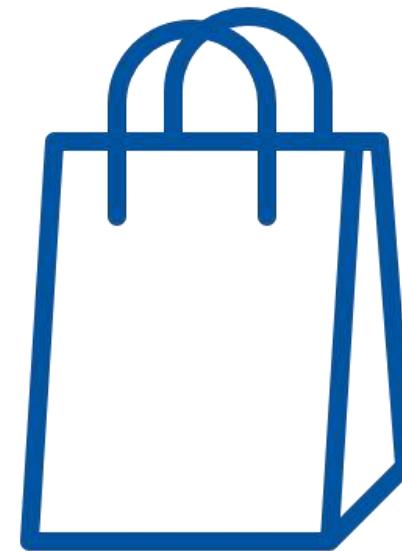
Modeling

After preprocessing the data, we can focus on building a usable **model** (i.e., representation)



Bag of Words Model

- Bag of Words (BoW) is the simplest model
- It represents each document d in the corpus as a bag of words (a multisets of words)



Bag of Words Model – Examples

“Process discovery and conformance checking are part of process mining.”

(apply stemming and
stop word removal)

[‘process’², ‘discover’¹, ‘conform’¹, ‘check’¹, ‘part’¹, ‘min’¹]

Bag of Words Model – Examples

“Process discovery and conformance checking are part of process mining.”

[‘process’², ‘discover’¹, ‘conform’¹, ‘check’¹, ‘part’¹, ‘min’¹]

”I love to eat pizza. Pizza is my favorite food. When I want to eat, I always choose pizza.”

[‘pizza’³, ‘eat’², ‘love’¹, ...]



Bag of Words Model

- A naïve model - multiset representation **loses the order** of the items:

“James loves watching movies, Kate hates it.”

“Kate loves watching movies, James hates it.”

[“James”, ‘love’, ‘watch’,
‘movie’, ‘Kate’, ‘hate’]

“Chickens hatch from eggs.”

“Eggs hatch from chickens.”

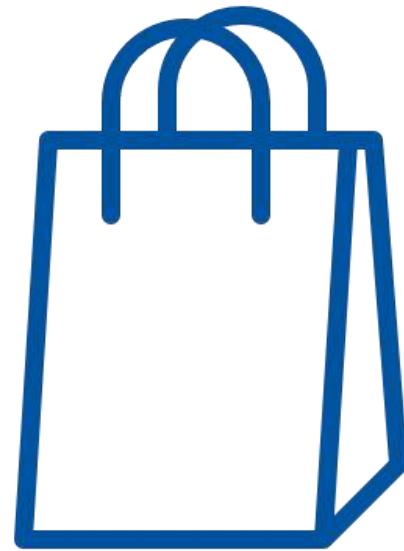
[“chicken”, ‘hatch’, ‘egg’]

→ BoW can be **unsuitable** for applications that strongly depend on **word order**

Bag of Words Model

Why is it still useful?

- Easy to implement
- Works well enough in many applications
- Was successfully used in the past
(e.g., for spam detection, information retrieval)
- Today, it's often used as an intermediate step ([feature extraction](#)) in combination with more advanced techniques



Document-Term Matrix

Tabular form for text data representation – Bag of Words in matrix form

Document	love	eat	pizza	favorite	food
Doc1	1	2	2	1	1
Doc2	0	1	2	0	0
Doc3	1	1	1	0	1

Frequency of
'love' in Doc3



Term Frequency (TF)

- A document-term matrix is a [mathematical matrix](#) that contains frequencies of the terms (words) found in documents
- These frequencies are called [term frequencies \(tf\)](#)

$$tf(w, d) = \text{number of occurrences of word } w \text{ in a document } d$$

Term Frequency (TF) - Example

$tf(w, d)$ = number of occurrences of word w in a document d

d = "I love to eat pizza. Pizza is my favorite food. When I want to eat, I always choose pizza. Pizza toppings can vary, but my go-to toppings are pepperoni and mushrooms. I could eat pizza every day and I would never get tired of it."



$$tf(\text{pizza}, d) = 5$$

$$tf(\text{I}, d) = 5$$

$$tf(\text{eat}, d) = 3$$

$$tf(\text{to}, d) = 2$$

$$tf(\text{toppings}, d) = 2$$

$$tf(\text{my}, d) = 2$$

$$tf(\text{and}, d) = 2$$

$$tf(\text{mushrooms}, d) = 1$$

...

Term Frequency (TF) - Example

$tf(w, d)$ = number of occurrences of word w in a document d

d = "I love to eat pizza. Pizza is my favorite food. When I want to eat, I always choose pizza. Pizza toppings can vary, but my go-to toppings are pepperoni and mushrooms. I could eat pizza every day and I would never get tired of it."



What is the topic of the document?

$$tf(\text{pizza}, d) = 5$$

$$tf(\text{I}, d) = 5$$

$$tf(\text{eat}, d) = 3$$

$$tf(\text{to}, d) = 2$$

$$tf(\text{toppings}, d) = 2$$

$$tf(\text{my}, d) = 2$$

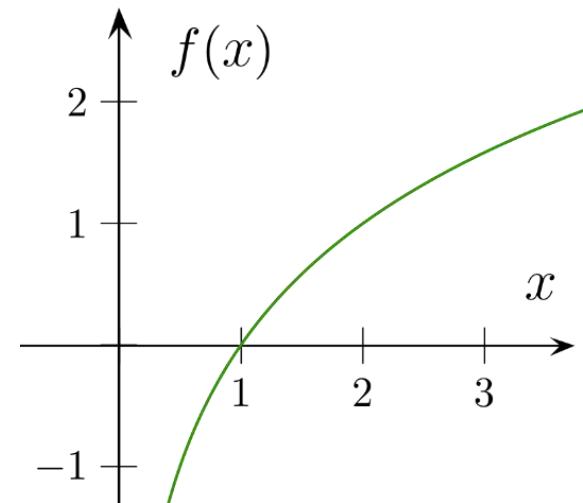
$$tf(\text{and}, d) = 2$$

$$tf(\text{mushrooms}, d) = 1$$

...

Inverse Document Frequency (IDF)

- IDF reflects how "special" the word is in the corpus c in terms of its frequency
- Intuition: The more unlikely the word is, the higher the value



$$idf(w, c) = \log_2\left(\frac{|c|}{\text{number of documents in } c \text{ that contain } w \text{ at least once}}\right)$$

Inverse Document Frequency (IDF) - Example

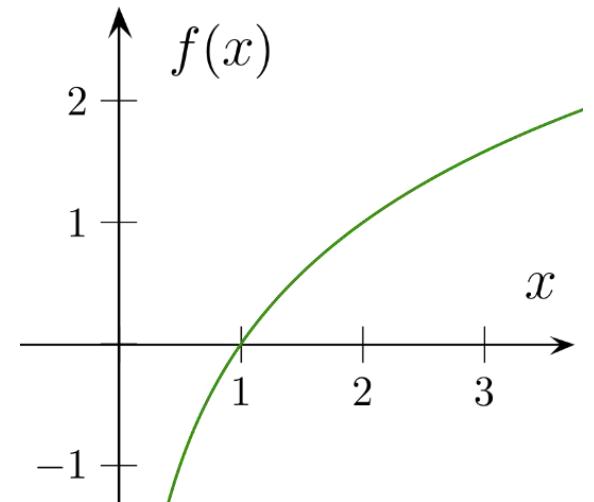
$$idf(w, c) = \log_2\left(\frac{|c|}{\text{number of documents in } c \text{ that contain } w \text{ at least once}}\right)$$

Consider an imaginary corpus of 100 recipes from a cookbook c :

$$idf(\text{water}, c) = \log_2\left(\frac{100}{92}\right) = 0.12$$

$$idf(\text{flour}, c) = \log_2\left(\frac{100}{60}\right) = 0.74$$

$$idf(\text{strawberries}, c) = \log_2\left(\frac{100}{15}\right) = 2.74$$



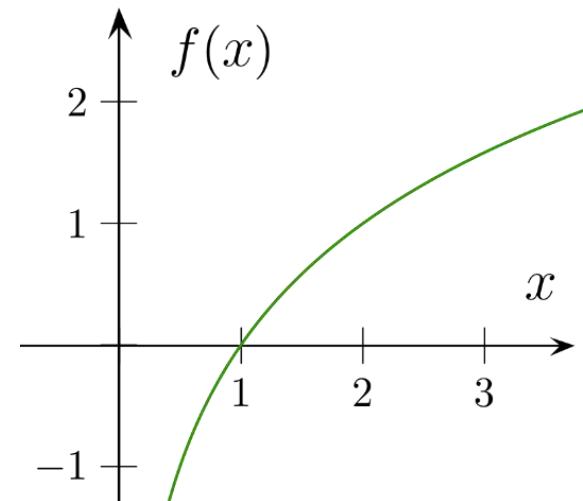
Inverse Document Frequency (IDF)

- Words with **low idf score** appear in many documents
- These words are **not useful** for distinguishing between the documents

$$idf(\text{water}, c) = \log_2\left(\frac{100}{92}\right) = 0.12$$

$$idf(\text{flour}, c) = \log_2\left(\frac{100}{60}\right) = 0.74$$

$$idf(\text{strawberries}, c) = \log_2\left(\frac{100}{15}\right) = 2.74$$



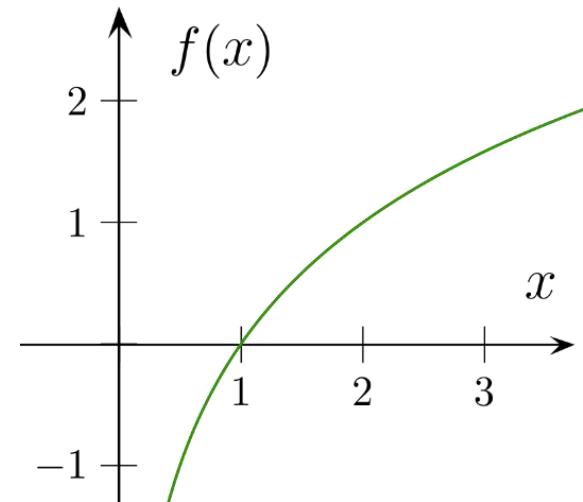
Inverse Document Frequency (IDF)

- Words with **low idf score** appear in many documents
- These words are **not useful** for distinguishing between the documents
- Low-scored words can be **stop word candidates** (add to a **stop list**)

$$idf(\text{water}, c) = \log_2\left(\frac{100}{92}\right) = 0.12$$

$$idf(\text{flour}, c) = \log_2\left(\frac{100}{60}\right) = 0.74$$

$$idf(\text{strawberries}, c) = \log_2\left(\frac{100}{15}\right) = 2.74$$



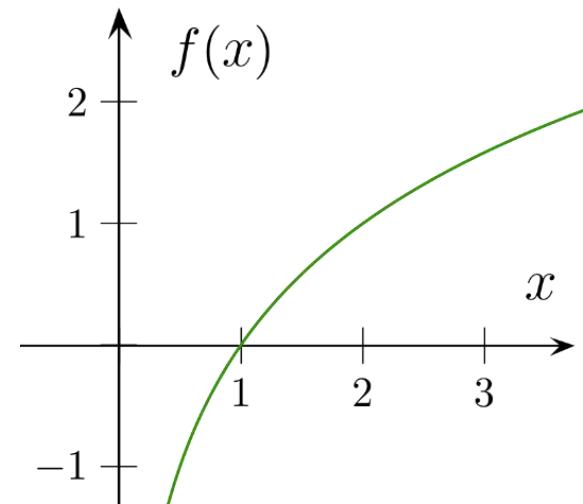
TF-IDF Scoring

- Combination of the **tf** and **idf** scoring
 - tf: **strength of the association** between a word and a document
 - idf: the **relevance** of a word in a whole corpus (how “special” it is)

$$tfidf(w, d, c) = tf(w, d) \cdot idf(w, c)$$

- Essential in **information retrieval**

$idf(w, c) < 1$: w occurs in more than half of the documents
 $idf(w, c) > 1$: w occurs in less than half of the documents



TF-IDF Scoring - Example

Four documents d_1, d_2, d_3 and d_4 in corpus c

$d_1 = \text{'Cats are the only pet of the felines family, while dogs are canids.'}$

stem: **feline**

$d_2 = \text{'Cats are the third-most popular pet in the US.'}$

$d_3 = \text{'Dogs have been selected for millennia as pet animals.'}$

$d_4 = \text{'Normally, dogs are not aggressive towards other dogs outside their territory.'}$

$$tf(\text{feline}, d_1) = 1$$

$$tf(\text{feline}, d_2) = 0$$

$$tf(\text{feline}, d_3) = 0$$

$$tf(\text{feline}, d_4) = 0$$



$$idf(\text{feline}, c) = \log_2(4/1) = 2$$

$$tfidf(\text{feline}, d_1, c) = 1 \cdot 2 = 2$$

$$tfidf(\text{feline}, d_2, c) = 0 \cdot 2 = 0$$

$$tfidf(\text{feline}, d_3, c) = 0 \cdot 2 = 0$$

$$tfidf(\text{feline}, d_4, c) = 0 \cdot 2 = 0$$

TF-IDF Scoring - Example

Four documents d_1, d_2, d_3 , and d_4 in corpus c

stem: cat

d_1 = ‘Cats are the only pet of the felines family, while dogs are canids.’

d_2 = ‘Cats are the third-most popular pet in the US.’

d_3 = ‘Dogs have been selected for millennia as pet animals.’

d_4 = ‘Normally, dogs are not aggressive towards other dogs outside their territory.’

$$tf(cat, d_1) = 1$$

$$tf(cat, d_2) = 1$$

$$tf(cat, d_3) = 0$$

$$tf(cat, d_4) = 0$$

$$idf(cat, c) = \log_2(4/2) = 1$$

$$tfidf(cat, d_1, c) = 1 \cdot 1 = 1$$

$$tfidf(cat, d_2, c) = 1 \cdot 1 = 1$$

$$tfidf(cat, d_3, c) = 0 \cdot 1 = 0$$

$$tfidf(cat, d_4, c) = 0 \cdot 1 = 0$$



TF-IDF Scoring - Example

Four documents d_1, d_2, d_3 and d_4 in corpus c

stem: dog

d_1 = ‘Cats are the only pet of the felines family, while dogs are canids.’

d_2 = ‘Cats are the third-most popular pet in the US.’

d_3 = ‘Dogs have been selected for millennia as pet animals.’

d_4 = ‘Normally, dogs are not aggressive towards other dogs outside their territory.’

$$tf(\text{dog}, d_1) = 1$$

$$tf(\text{dog}, d_2) = 0$$

$$tf(\text{dog}, d_3) = 1$$

$$tf(\text{dog}, d_4) = 2$$

$$idf(\text{dog}, c) = \log_2(4/3) = 0.42$$

$$tfidf(\text{dog}, d_1, c) = 1 \cdot 0.42 = 0.42$$

$$tfidf(\text{dog}, d_2, c) = 0 \cdot 0.42 = 0$$

$$tfidf(\text{dog}, d_3, c) = 1 \cdot 0.42 = 0.42$$

$$tfidf(\text{dog}, d_4, c) = 2 \cdot 0.42 = 0.84$$



TF-IDF Scoring

- Many querying systems rely on TF-IDF scoring (or a variation)
- Simple algorithm:
 - Input: a query (a set of words) and a corpus c
 - For each document d in the corpus compute
$$score(\text{query}, d, c) = \sum_{w \in \text{query}} tfidf(w, d, c)$$
 - Rank documents by their scores
 - Return first x documents

$$tf(w, d) \cdot idf(w, c)$$

Document-Term Matrix with TF-IDF

- The document-term matrix can also be built with **TF-IDF** scores

each column is
a word/term

	word1	word2	word3	word4	...
doc1					
doc2					
...					

each row is
a document

each cell
contains the
TF-IDF score

- Tabular data with instances (documents) and features (words) – other features can be added
- The matrix allows to **apply a wide range of data science techniques**

Document Classification

Document	love	eat	pizza	favorite	food	Article class
Doc1	1	2	2	1	1	recipe
Doc2	0	1	2	0	0	restaurant menu
Doc3	1	1	1	0	1	cookbook

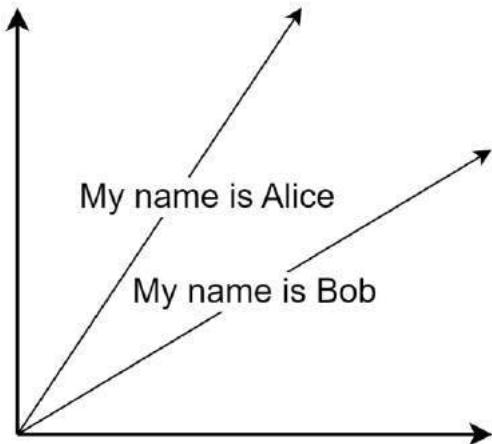
Every document is **represented** by a vector of a constant length
(term frequencies or TF-IDF scores)

Target label for
classification (e.g.,
to train a neural
network)

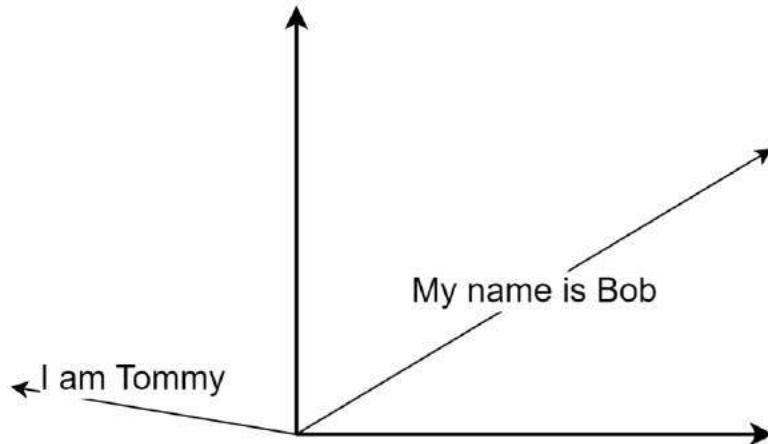
Document Clustering

- Having a fixed length vector, we need a **distance/similarity measure** to perform clustering
- Recall the Clustering lecture— we can use **cosine similarity**

$$\text{sim}(\mathbf{x}_i, \mathbf{x}_j) = \frac{\mathbf{x}_i \cdot \mathbf{x}_j}{\|\mathbf{x}_i\| \|\mathbf{x}_j\|} = \cos(\text{angle})$$



Similar vectors
Positive, approaches 1.0



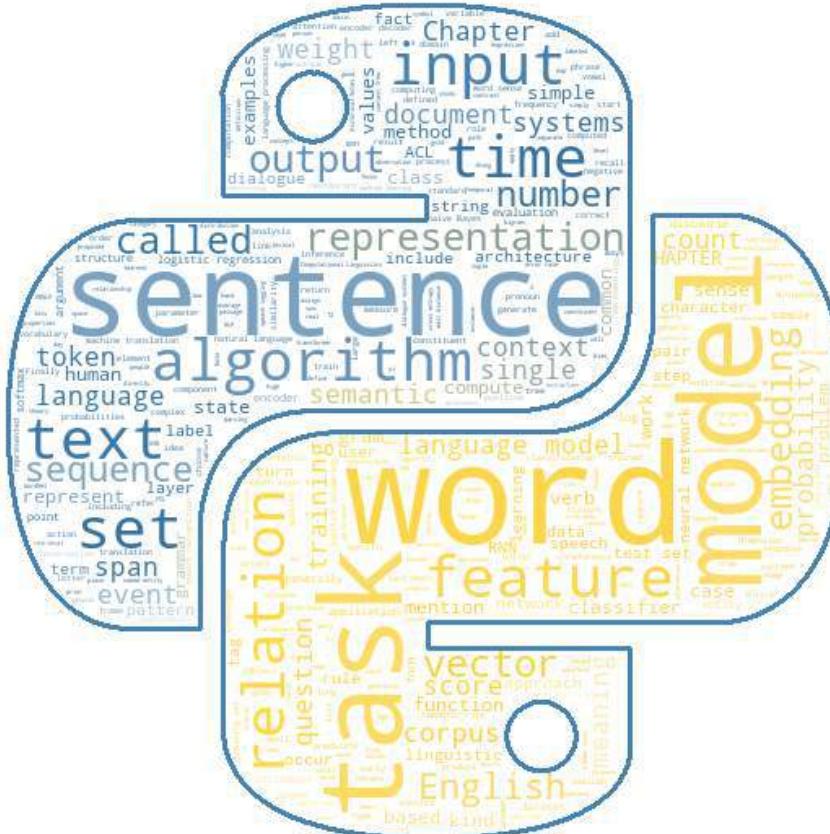
Different vectors
Negative, approaches -1.0

Document Clustering

- Cosine similarity is well-suited to compute for sparse vectors or vectors with different lengths (but in principle other metrics are possible)
- With a distance metric for text data, we can perform clustering (e.g., K-means, K-medoids, DBSCAN, etc.)

Text Mining

1. Introduction to Text Mining
 2. Text Preprocessing
 3. Modeling
 4. **Enriching Text: Linked Data**
 5. N-gram Model
 6. Learning a Representation
 7. Word2vec



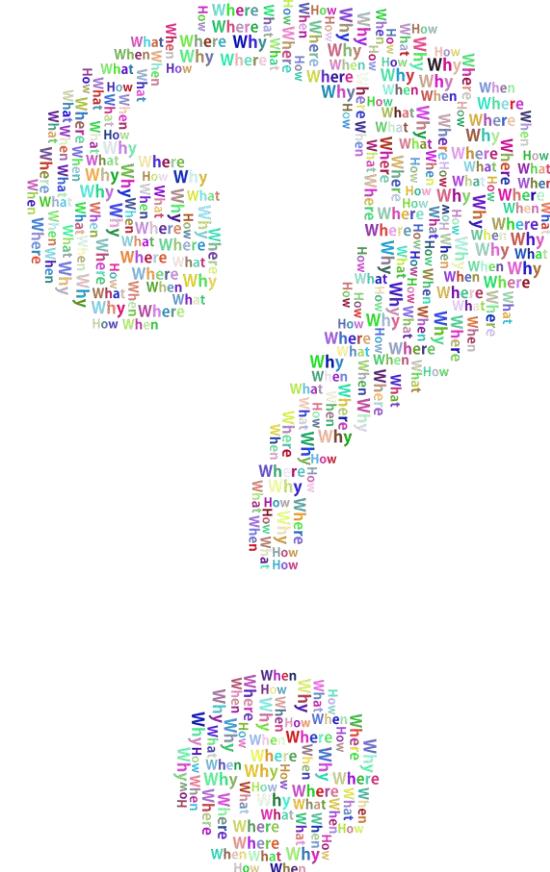
Information Structure

- Representing text using a matrix makes it "processable", but connections between concepts and their meaning are missing

A dense, colorful cloud of words representing semantic connections between the question words 'How', 'What', and 'Where'. The words are repeated in various colors (red, green, blue, yellow, purple) and orientations (rotated, mirrored, etc.) to show their interconnectedness.

Information Structure

- Representing text using a matrix makes it "processable", but **connections between concepts and their meaning** are missing
- Solution: **databases** that are able to store connections between terms
- Querying these databases allows to navigate words on the basis of **lexical relationships**

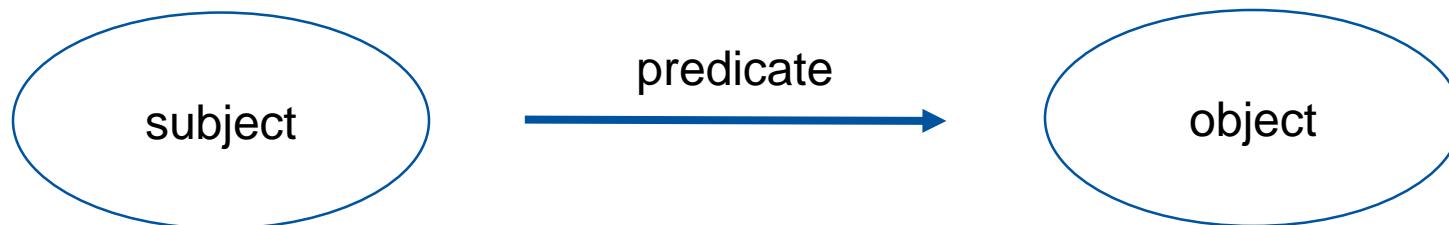


WordNet

- A [lexical network](#) for English
- 155.000+ words
- Organized in graphs and divided in [synsets](#) (sets of [synonyms](#), i.e., semantically equivalent)
- [Relationship information](#):
 - antonyms (cold vs warm),
 - hyponyms (daisy and rose are hyponyms of a flower, i.e., “is a” relation),
 - meronyms (wheel and engine are meronyms of a car, i.e., “part of” relation),
 - ...

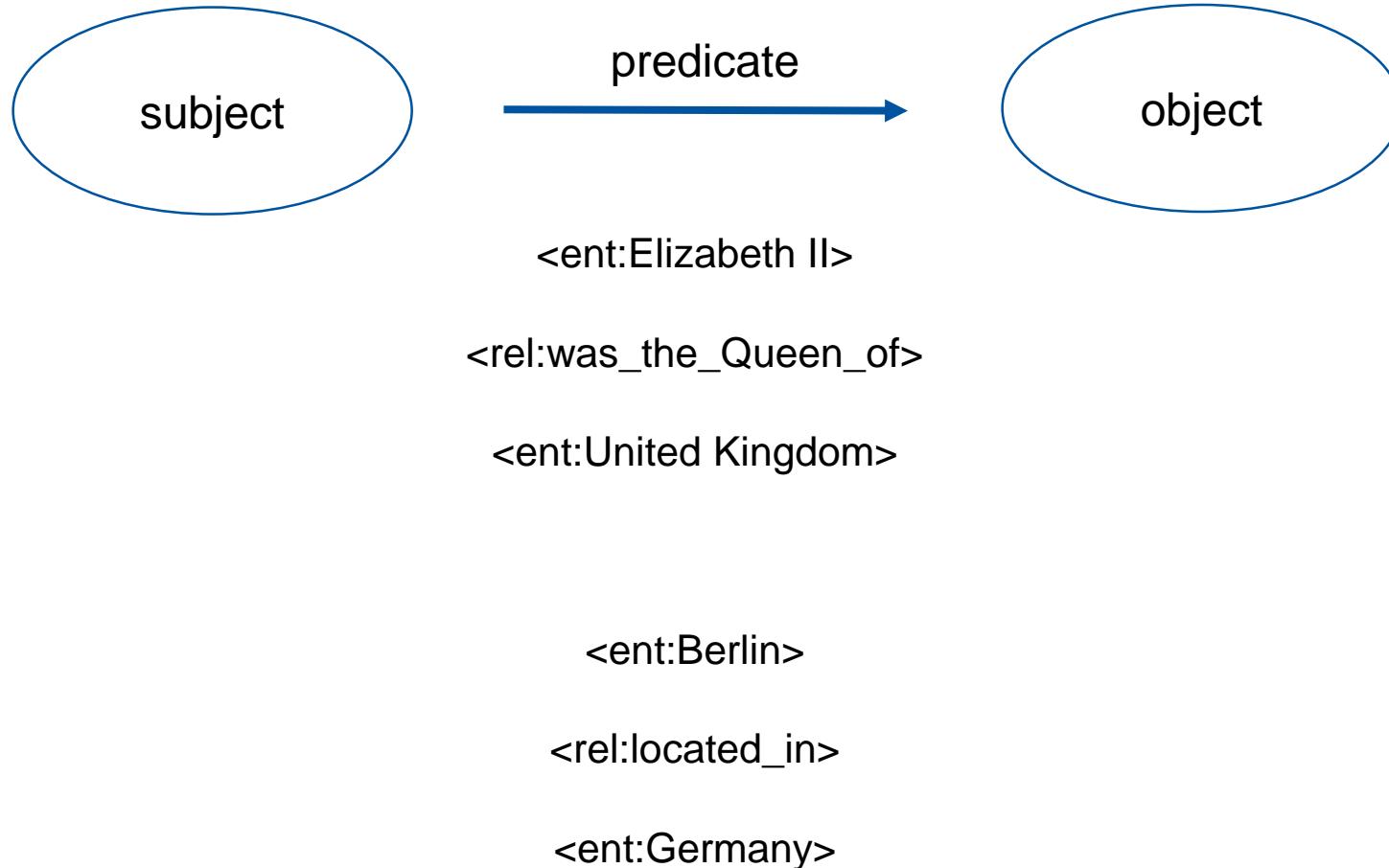
Resource Description Framework (RDF)

- One step further: a general data model
- Describes **relationships** between **things** – connections between concepts can carry **any meaning**
- Based on triples - statements of the following form:



- Looks simple, but with a database large enough, we can answer very complex queries

Resource Description Framework (RDF) - Example



Uniform Resource Identifier (URI)

- Unique and unambiguous identifiers for entities
- Easy to define for limited domains
 - Student ID is a URI for students
 - ISBN is a URI for books
- If we want to identify all entities, it becomes more difficult

DBpedia

Wikipedia for machines

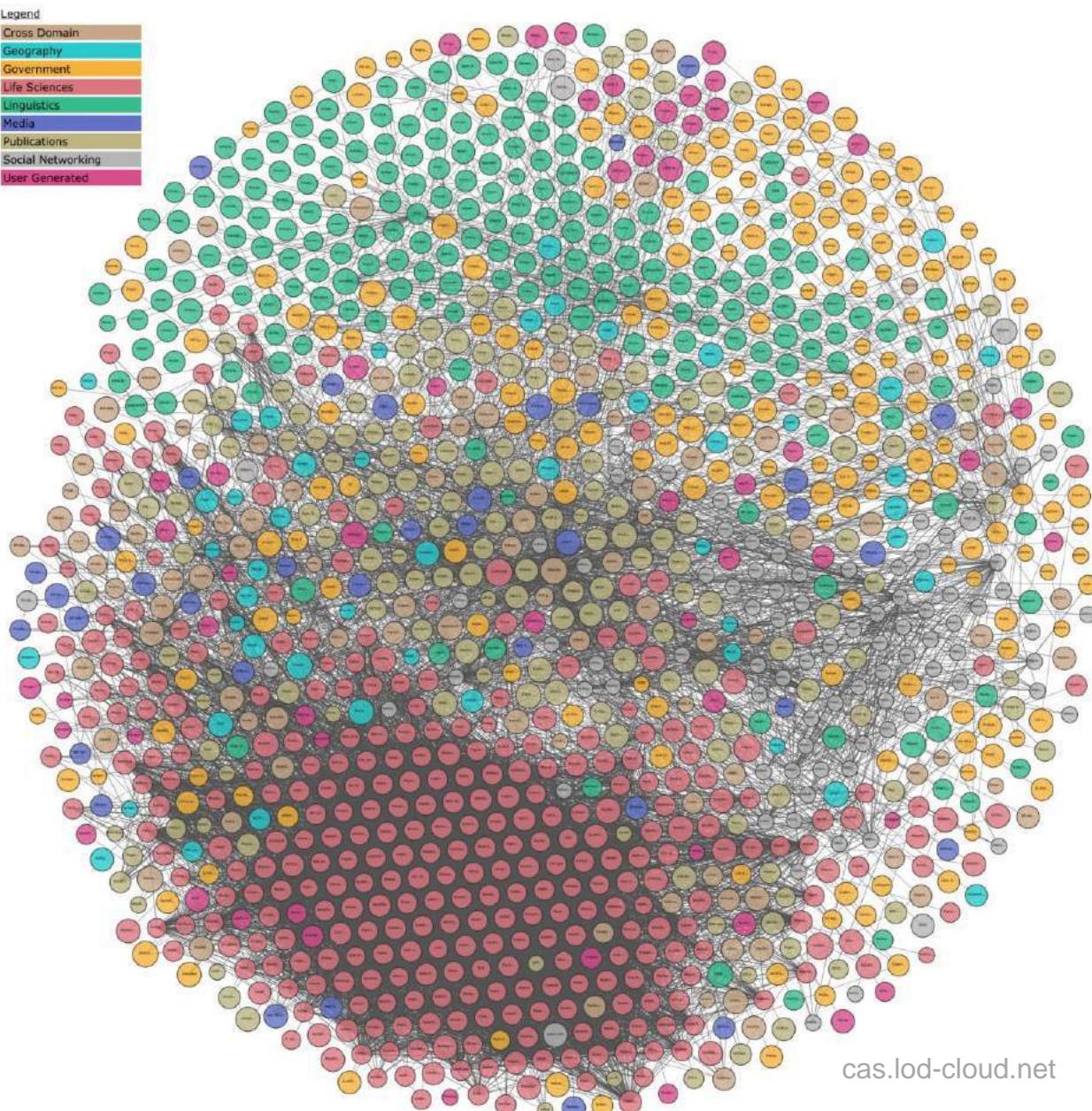
Information extracted from Wikipedia, but organized in RDF triples

Property	Value
dbo:abstract	<ul style="list-style-type: none"> Pizza (Italian: ['pittsa], Neapolitan: ['pittsa]) is a dish of Italian origin consisting of a usually round, flat base of leavened wheat-based dough topped with tomatoes, cheese, and often various other ingredients (such as various types of sausage, anchovies, mushrooms, onions, olives, vegetables, meat, ham, etc.), which is then baked at a high temperature, traditionally in a wood-fired oven. A small pizza is sometimes called a pizzetta. A person who makes pizza is known as a pizzaiolo. In Italy, pizza served in a restaurant is presented unsliced, and is eaten with the use of a knife and fork. In casual settings, however, it is cut into wedges to be eaten while held in the hand. The term pizza was first recorded in the 10th century in a Latin manuscript from the Southern Italian town of Gaeta in Lazio, on the border with Campania. Modern pizza was invented in Naples, and the dish and its variants have since become popular in many countries. It has become one of the most popular foods in the world and a common fast food item in Europe, North America and Australasia; available at pizzerias (restaurants specializing in pizza), restaurants offering Mediterranean cuisine, via pizza delivery, and as street food. Various food companies sell ready-baked pizzas, which may be frozen, in grocery stores, to be reheated in a home oven. In 2017, the world pizza market was US\$128 billion, and in the US it was \$44 billion spread over 76,000 pizzerias. Overall, 13% of the U.S. population aged 2 years and over consumed pizza on any given day. The Associazione Verace Pizza Napoletana (lit. True Neapolitan Pizza Association) is a non-profit organization founded in 1984 with headquarters in Naples that aims to promote traditional Neapolitan pizza. In 2009, upon Italy's request, Neapolitan pizza was registered with the European Union as a Traditional Speciality Guaranteed dish, and in 2017 the art of its making was included on UNESCO's list of intangible cultural heritage. Raffaele Esposito is often considered to be the father of modern pizza. <p>(en)</p>
dbo:hasVariant	<ul style="list-style-type: none"> dbr:Calzone dbr:Stromboli_(food) dbr:Panzerotti
dbo:ingredient	<ul style="list-style-type: none"> dbr:Cheese dbr:Tomato_sauce dbr:Dough
dbo:ingredientName	Dough, sauce (usually tomato sauce), cheese
dbo:region	dbr:Campania
dbo:servingTemperature	Hot or warm



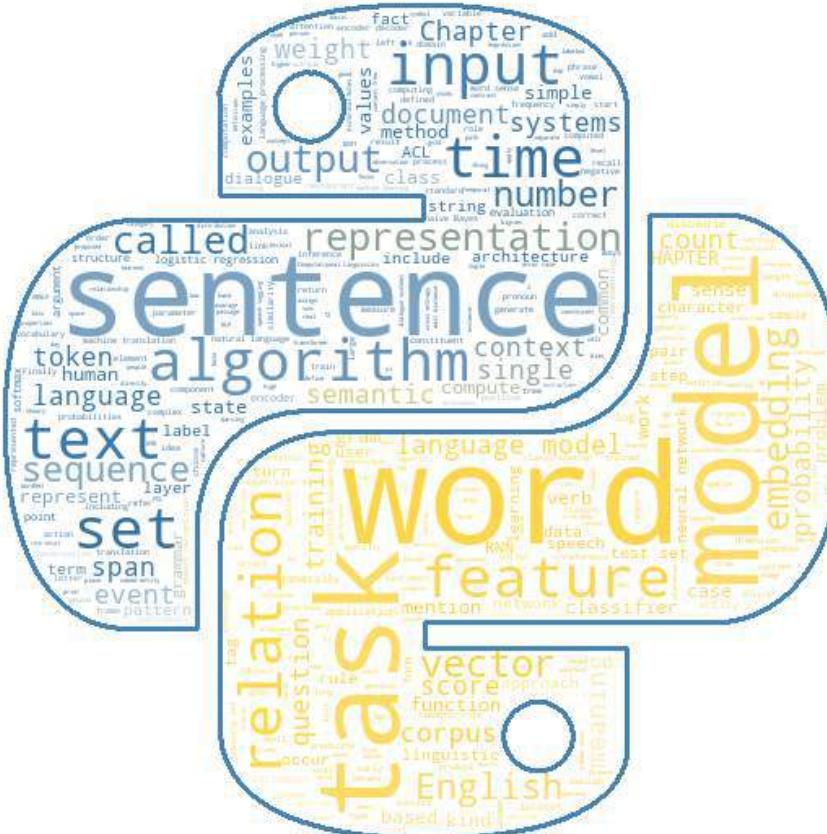
Linked Open Data

- A number of open source databases are interconnected and form the [Linked Open Data](#)
 - a large database of statements
- Publicly available and reusable
- Combined dimensions – tens of billions of RDF triples!
- Example: <http://cas.lod-cloud.net/>
- Word in text can be related to entities in such a database



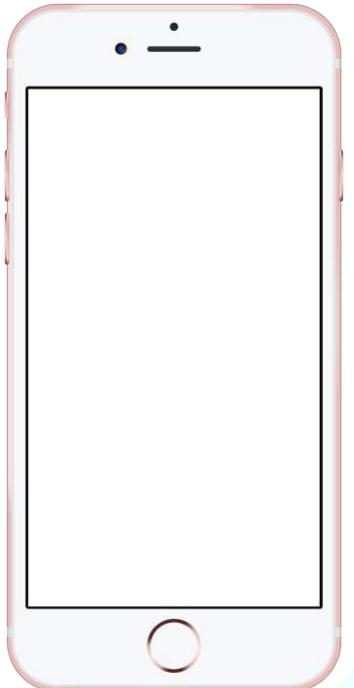
Text Mining

1. Introduction to Text Mining
 2. Text Preprocessing
 3. Modeling
 4. Enriching Text: Linked Data
 5. **N-gram Model**
 6. Learning a Representation
 7. Word2vec



Completion Prediction

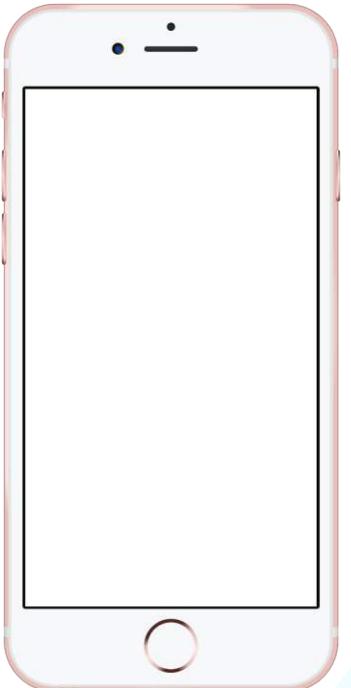
- Completion prediction is another text mining application
- Given a sequence of words, predict the next word
- Examples:
 - Apple gets the most of its revenue from selling cell _____
 - Your code crashed, it has a _____



Completion Prediction

- Completion prediction is another text mining application
- Given a sequence of words, predict the next word
- Examples:
 - Apple gets the most of its revenue from selling cell _____
 - Your code crashed, it has a _____

phones



Completion Prediction

- Completion prediction is another text mining application
- Given a sequence of words, predict the next word
- Examples:

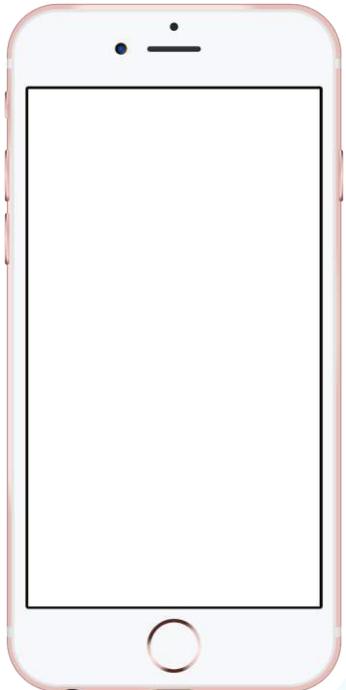
– Apple gets the most of its revenue from selling cell _____ phones

– Your code crashed, it has a _____ bug



Completion Prediction

- Completion prediction is another text mining application
- Given a sequence of words, predict the next word
- Examples:
 - Apple gets the most of its revenue from selling cell _____ phones
 - Your code crashed, it has a _____ bug
- The BoW model is not useful in this case
- We need a model that is capable to retain information about the word order



N-gram Model

- N-gram models use **sequences of consecutive tokens**, instead of individual tokens
- The **N** in N-gram indicates the length of a sequence

N-gram Model (N=1)

“Apples|are|good|for|you.”

Unigram model:

[‘apples’, ‘are’, ‘good’, ‘for’, ‘you’]

The unigram model is identical to BoW!

N-gram Model (N=2)

“Apples are good for you.”

Unigram model:

[‘apples’, ‘are’, ‘good’, ‘for’, ‘you’]

Bigram model:

[('apples', 'are'), ('are', 'good'), ('good', 'for'), ('for', 'you')]

N-gram Model (N=3)

“Apples are good for you.”

Unigram model:

['apples', 'are', 'good', 'for', 'you']

Bigram model:

[('apples', 'are'), ('are', 'good'), ('good', 'for'), ('for', 'you')]

Trigram model:

[('apples', 'are', 'good'), ('are', 'good', 'for'), ('good', 'for', 'you')]

Preprocessing

- Preprocessing steps **are context and application dependent**
- For example, stop word removal is useful for the **document classification**, however, it should not be utilized for **completion prediction**
- Examples in this video **do not** use any preprocessing steps
(no stop word removal, stemming or lemmatization)

Motivation

- We use N-grams is to estimate the probability of a word occurrence given its prior context
- An N-gram uses N-1 tokens for the context
 - **Unigram:** $P(\text{phone})$  Probability that a random word in the corpus is “phone”
 - **Bigram:** $P(\text{phone} \mid \text{cell})$
 - **Trigram:** $P(\text{phone} \mid \text{your cell})$

Motivation

- We use N-grams is to estimate the probability of a word occurrence given its prior context
- An N-gram uses N-1 tokens for the context
 - **Unigram:**
 $P(\text{phone})$ Probability that a random word in the corpus is “phone” given that the previous is “cell”
 - **Bigram:**
 $P(\text{phone} \mid \text{cell})$
 - **Trigram:**
 $P(\text{phone} \mid \text{your cell})$

Motivation

- We use N-grams is to estimate the probability of a word occurrence given its prior context
- An N-gram uses N-1 tokens for the context
 - **Unigram:**
 $P(\text{phone})$
 - **Bigram:**
 $P(\text{phone} \mid \text{cell})$ Probability that a random word in the corpus is “phone” given that the two previous words are “your” and “cell”
 - **Trigram:**
 $P(\text{phone} \mid \text{your cell})$

Example

If you wait too long for the perfect moment, the **perfect moment** will pass you by.
2-gram

If you wait too long for the perfect moment, **the perfect moment** will pass you by.
3-gram

If you wait too long for the perfect **moment**, **the perfect moment** will pass you by.
4-gram

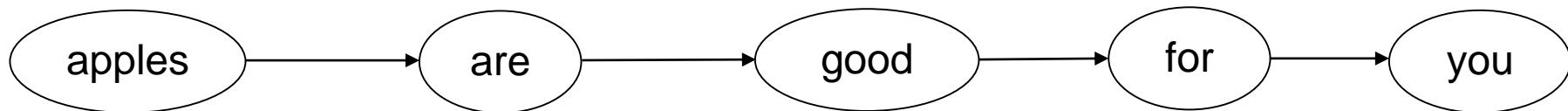
If you wait too long for the **perfect moment**, **the perfect moment** will pass you by.
5-gram

If you wait too long for the **perfect moment**, **the perfect moment** will pass you by.
6-gram



N-gram Model

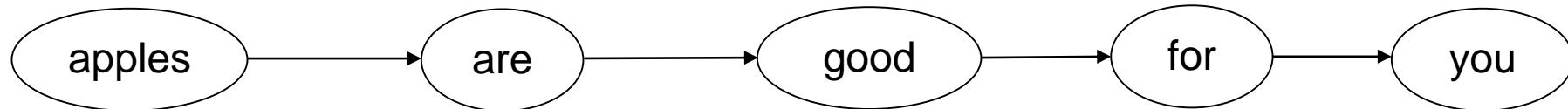
- Our goal is to quantify the **strength** of the relationship represented by the arrows
- Use N-grams to learn a function that quantifies the probability of a word “w” given its prior context



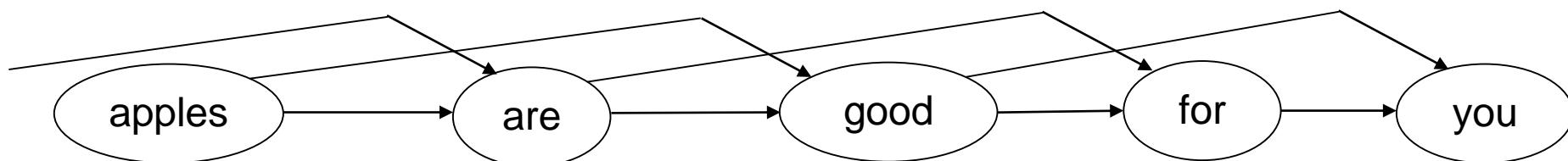
Bigram: arrows from w_1 to w_2 imply that “ w_2 depends on w_1 ”.

N-gram Model

- Our goal is to quantify the **strength** of the relationship represented by the arrows
- Use N-grams to learn a function that quantifies the probability of a word “w” given its prior context



Bigram: arrows from w_1 to w_2 imply that “ w_2 depends on w_1 ”.



Trigram: “ w_k depends on w_{k-1} and w_{k-2} ”

N-gram Model – Computing Probabilities

Word sequences $w_1^n = w_1 \dots w_n$

Chain rule: $P(w_1^n) = P(w_1)P(w_2 | w_1)P(w_3 | w_1^2) \dots P(w_n | w_1^{n-1}) = \prod_{k=1}^n P(w_k | w_1^{k-1})$

$$w_1^n = w_1 \dots w_n$$

$$w_1^{k-1} = w_1 \dots w_{k-1}$$

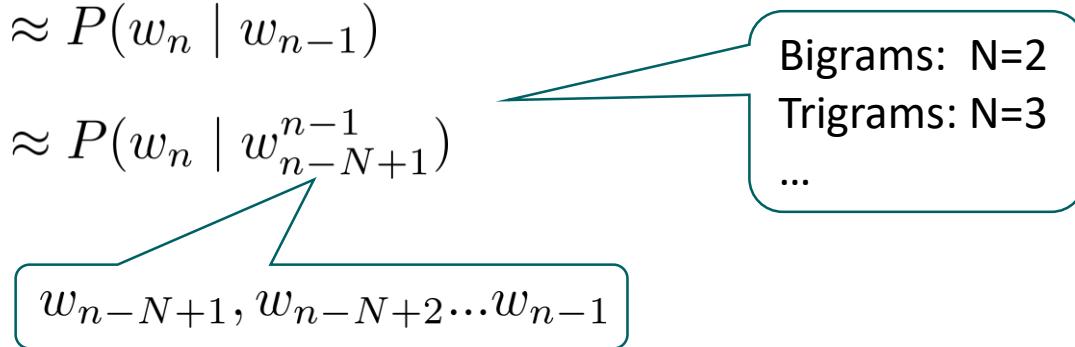
Example ($n = 5$):

$$P(I, \text{ like, pizza, with, mushrooms}) = P(I) \cdot P(\text{like} | I) \cdot P(\text{pizza} | I, \text{ like}) \cdot P(\text{with} | I, \text{ like, pizza}) \cdot P(\text{mushrooms} | I, \text{ like, pizza, with})$$

1 2 3 4 5
 1 2 1 3 2 4 3 5 4

N-gram Model – Markov Assumption

- The Markov assumption: probability of a certain word depends only on a limited context
- Assumption on bigrams: $P(w_n \mid w_1^{n-1}) \approx P(w_n \mid w_{n-1})$
- Assumption on N-grams: $P(w_n \mid w_1^{n-1}) \approx P(w_n \mid w_{n-N+1}^{n-1})$
- Formalization of ‘the last N-1 words matter’
- Note that we need to handle the special case $n - N + 1 < 1$ by using default values



N-gram Model – Computing Probabilities

- Based on the **Markov assumption** and the **chain rule**:

→ Chain rule: $P(w_1^n) = P(w_1)P(w_2 \mid w_1)P(w_3 \mid w_1^2)\dots P(w_n \mid w_1^{n-1}) = \prod_{k=1}^n P(w_k \mid w_1^{k-1})$

Example (n=5)

$$\begin{array}{ccccc} 1 & 2 & 3 & 4 & 5 \\ P(I, \text{ like, pizza, with, mushrooms}) = & & & & \\ P(I) \cdot P(\text{like} \mid I) \cdot P(\text{pizza} \mid I, \text{ like}) \cdot P(\text{with} \mid I, \text{ like, pizza}) \cdot P(\text{mushrooms} \mid I, \text{ like, pizza, with}) \\ 1 & 2 & 1 & 3 & 2 & 4 & 3 & 5 & 4 \end{array}$$

N-gram Model – Computing Probabilities

- Based on the **Markov assumption** and the **chain rule**:
- Bigram approximation:** $P(w_1^n) \approx \prod_{k=1}^n P(w_k \mid w_{k-1})$

The last word

$$\rightarrow \text{Chain rule: } P(w_1^n) = P(w_1)P(w_2 \mid w_1)P(w_3 \mid w_1^2)\dots P(w_n \mid w_1^{n-1}) = \prod_{k=1}^n P(w_k \mid w_1^{k-1})$$

Example (n=5)

$$P(I, \text{ like, pizza, with, mushrooms}) = P(I) \cdot P(\text{like} \mid I) \cdot P(\text{pizza} \mid \cancel{I}, \text{like}) \cdot P(\text{with} \mid \cancel{I}, \cancel{\text{like}}, \text{pizza}) \cdot P(\text{mushrooms} \mid \cancel{I}, \cancel{\text{like}}, \cancel{\text{pizza}}, \text{with})$$

1 2 3 4 5

Special case
(missing context)

N-gram Model – Computing Probabilities

- Based on the **Markov assumption** and the **chain rule**:
 - **Trigram approximation:** $P(w_1^n) \approx \prod_{k=1}^n P(w_k \mid w_{k-2}^{k-1})$
- Chain rule: $P(w_1^n) = P(w_1)P(w_2 \mid w_1)P(w_3 \mid w_1^2)\dots P(w_n \mid w_1^{n-1}) = \prod_{k=1}^n P(w_k \mid w_1^{k-1})$

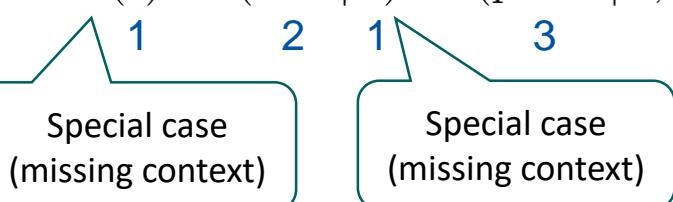
The last 2 words

Example (n=5)

$$P(I, \text{ like, pizza, with, mushrooms}) =$$

$$P(I) \cdot P(\text{like} \mid I) \cdot P(\text{pizza} \mid I, \text{ like}) \cdot P(\text{with} \mid \cancel{I}, \text{ like, pizza}) \cdot P(\text{mushrooms} \mid \cancel{I}, \cancel{\text{like}}, \cancel{\text{pizza}}, \text{ with})$$

1 2 3 4 5



N-gram Model – Computing Probabilities

- Based on the **Markov assumption** and the **chain rule**:

- **N-gram approximation:** $P(w_1^n) \approx \prod_{k=1}^n P(w_k | w_{k-N+1}^{k-1})$

The last N-1 words

$$\rightarrow \text{Chain rule: } P(w_1^n) = P(w_1)P(w_2 | w_1)P(w_3 | w_1^2) \dots P(w_n | w_1^{n-1}) = \prod_{k=1}^n P(w_k | w_1^{k-1})$$

Maximum Likelihood Estimation in N-gram Models

- How to estimate these probabilities?
→ Maximum Likelihood Estimation method (MLE)
- Count function: $C(w_1^n) \rightarrow$ the number of occurrences of w_1^n in the corpus
- Bigram MLE estimation: $P(w_n \mid w_{n-1}) = \frac{C(w_{n-1}w_n)}{C(w_{n-1})}$

the most likely w_n given
the last word

Maximum Likelihood Estimation in N-gram Models

- How to estimate these probabilities?
→ Maximum Likelihood Estimation method (MLE)
- Count function: $C(w_1^n) \rightarrow$ the number of occurrences of w_1^n in the corpus

- Bigram MLE estimation: $P(w_n \mid w_{n-1}) = \frac{C(w_{n-1}w_n)}{C(w_{n-1})}$

the most likely w_n given
the last word

- N-gram MLE estimation: $P(w_n \mid w_{n-N+1}^{n-1}) = \frac{C(w_{n-N+1}^{n-1}w_n)}{C(w_{n-N+1}^{n-1})}$

the most likely w_n given
the last N-1 words

Example

Bigrams (2-grams)

Corpus of 3 sentences:

<s> we want to eat <e>

<s> we have pizza <e>

<s> today we have pizza for dinner <e>

Bigram MLE estimation:

$$P(w_n \mid w_{n-1}) = \frac{C(w_{n-1}w_n)}{C(w_{n-1})}$$

→ Ensure context for N-grams: add special ‘words’ on sentence beginning and end

Example

Bigrams (2-grams)

Corpus of 3 sentences:

`<s> we want to eat <e>`

`<s> we have pizza <e>`

`<s> today we have pizza for dinner <e>`

Bigram MLE estimation:

$$P(w_n \mid w_{n-1}) = \frac{C(w_{n-1}w_n)}{C(w_{n-1})}$$

$$P(\text{we} \mid \text{<s>}) = \frac{2}{3}$$

$$P(\text{<e>} \mid \text{pizza}) = \frac{1}{2}$$

$$P(\text{today} \mid \text{<s>}) = \frac{1}{3}$$

$$P(\text{pizza} \mid \text{have}) = \frac{2}{2}$$

$$P(\text{have} \mid \text{we}) = \frac{2}{3}$$

$$P(\text{want} \mid \text{we}) = \frac{1}{3}$$

Example

Bigrams (2-grams)

Corpus of 3 sentences:

<s> we want to eat <e>

<s> we have pizza <e>

<s> today we have pizza for dinner <e>

Bigram MLE estimation:

$$P(w_n | w_{n-1}) = \frac{C(w_{n-1}w_n)}{C(w_{n-1})}$$

$$P(\text{we} | \text{<s>}) = \frac{2}{3}$$

$$P(\text{<e>} | \text{pizza}) = \frac{1}{2}$$

$$P(\text{today} | \text{<s>}) = \frac{1}{3}$$

$$P(\text{pizza} | \text{have}) = \frac{2}{2}$$

$$P(\text{have} | \text{we}) = \frac{2}{3}$$

$$P(\text{want} | \text{we}) = \frac{1}{3}$$

Example

Bigrams (2-grams)

Corpus of 3 sentences:

`<s> we` want to eat `<e>`

`<s> we` have pizza `<e>`

`<s> today` we have pizza for dinner `<e>`

Bigram MLE estimation:

$$P(w_n \mid w_{n-1}) = \frac{C(w_{n-1}w_n)}{C(w_{n-1})}$$

$$P(\text{we} \mid \text{<s>}) = \frac{2}{3}$$

$$P(\text{<e>} \mid \text{pizza}) = \frac{1}{2}$$

$$P(\text{today} \mid \text{<s>}) = \frac{1}{3}$$

$$P(\text{pizza} \mid \text{have}) = \frac{2}{2}$$

$$P(\text{have} \mid \text{we}) = \frac{2}{3}$$

$$P(\text{want} \mid \text{we}) = \frac{1}{3}$$

Example

Bigrams (2-grams)

Corpus of 3 sentences:

<s> we want to eat <e>

<s> we have pizza <e>

<s> today we have pizza for dinner <e>

Bigram MLE estimation:

$$P(w_n \mid w_{n-1}) = \frac{C(w_{n-1}w_n)}{C(w_{n-1})}$$

$$P(\text{we} \mid \text{<s>}) = \frac{2}{3}$$

$$P(\text{<e>} \mid \text{pizza}) = \frac{1}{2}$$

$$P(\text{today} \mid \text{<s>}) = \frac{1}{3}$$

$$P(\text{pizza} \mid \text{have}) = \frac{2}{2}$$

$$P(\text{have} \mid \text{we}) = \frac{2}{3}$$

$$P(\text{want} \mid \text{we}) = \frac{1}{3}$$

Example 2

Estimation for a whole sentence

- Use the [Markov assumption](#) and the [chain rule](#)
- Multiply the bigrams (N-gram) probabilities

$$P(<\text{s}>, \text{I}, \text{ like}, \text{ pizza}, \text{ with}, \text{ mushrooms}, <\text{e}>) \approx$$

$$P(\text{I} | <\text{s}>) \cdot P(\text{like} | \text{I}) \cdot P(\text{pizza} | \text{like}) \cdot P(\text{with} | \text{pizza}) \cdot P(\text{mushrooms} | \text{with}) \cdot P(<\text{e}> | \text{mushrooms})$$

N-grams: Effectiveness

Sentences generated by sampling from the distributions of different N-gram models trained on Shakespeare's sonnets:

- 1 gram** “of hour loved worship sweet metre moving fore rank and the for of fair better a art careful graciously with“
- 2 gram** “thou wilt prove me thus by day by but day by their physicians know not to the bath for blunting the“
- 3 gram** “methinks no face so gracious is as mine importune thee root pity in the face sweet love remembered such wealth brings that“
- 4 gram** “yet this shall I neer know but live in doubt till my bad angel fire my good one out“

Limitation: Sparseness

- Sparseness
- Assume we use sentences based on a vocabulary of 30 000 words.
 - Bigrams: $30000^2 = 900000000$
 - Trigrams: $30000^3 = 27000000000000$
 - 4-grams: $30000^4 = 810000000000000000$
- In a vector model based on N-grams, this is the number of dimensions
- The dataset would contain mostly zeroes
→ this method is non-applicable for classification problems (as-is)

Limitation: Sparseness

- Imagine we split into **test** and **training data**
 - Train an N-gram model on the training corpus
 - Test it on sequences of length N-1 in the test corpus
- Likely, there are sequences in the test corpus that are **unseen** in the training data
 - The model assigns probability 0
 - The model “as is” is **overfitting** the training data

→ Solution: **Smoothing**

Solution: Smoothing

- Chip-off some probability from likely sequences
- Distribute it to unseen sequences (\rightarrow small but non-zero probability)
- Example: Laplace Smoothing

$$P(w_n \mid w_{n-1}) = \frac{C(w_{n-1} w_n)}{C(w_{n-1})} \longrightarrow P_{Laplace}(w_n \mid w_{n-1}) = \frac{C(w_{n-1} w_n) + 1}{C(w_{n-1}) + V}$$

Ensure non-zero probability

Normalize using size of the vocabulary, to make the sum of all probabilities 1

N-grams on Characters

- The basic version of the N-gram model is often applied to **characters** instead of words
- The vocabulary is just the alphabet (much smaller than all possible words)

4-grams

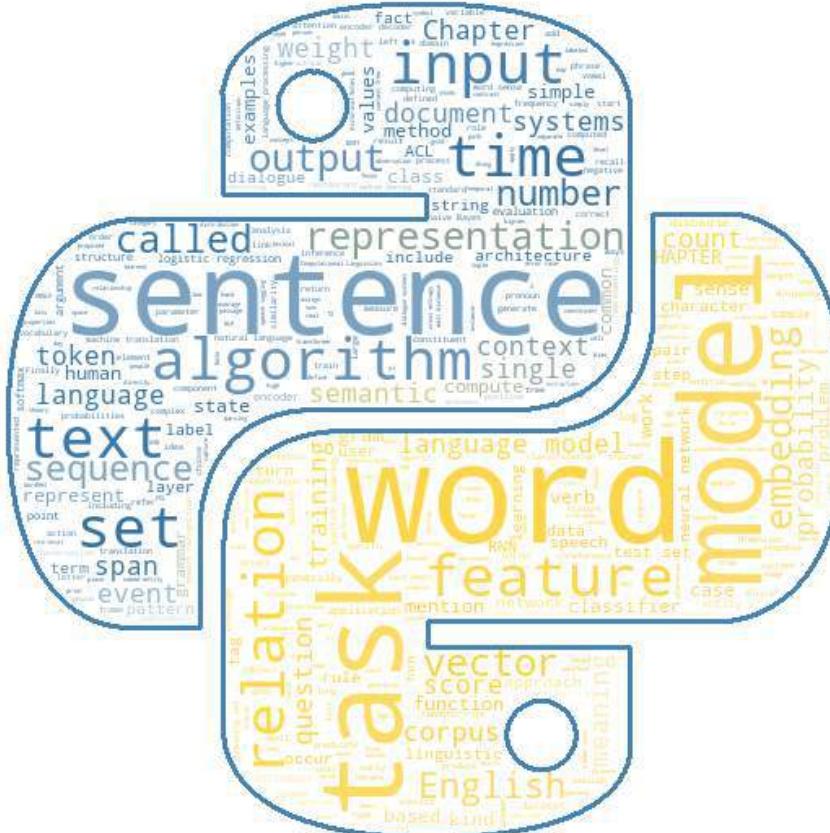
Once_upon_a_time: Once
Once_upon_a_time: nce_
Once_upon_a_time: ce_u
Once_upon_a_time: e_up

Unknowns

- What if a test corpus contains words that **do not occur** in the training corpus?
- These are called **Out of Vocabulary** words (OOV) or **unknowns**
→ the model cannot predict them
- We can model such unknown words using a specific pseudo-word: **<UNK>**

Text Mining

1. Introduction to Text Mining
2. Text Preprocessing
3. Modeling
4. Enriching Text: Linked Data
5. N-gram Model
6. Learning a Representation
7. Word2vec



Sparseness

- The representations of words are based on the length of the dictionary (BoW, N-grams)
→ tend to be very long
- This is because words behave like the categorical data
→ needs one-hot encoding, one feature per word

Sparseness

- Categorical data needs one-hot representation
→ A vector of length equal to the number of possible values

Rome = [1, 0, 0, 0, ..., 0]

Paris = [0, 1, 0, 0, ..., 0]

Italy = [0, 0, 1, 0, ..., 0]

France = [0, 0, 0, 0, ..., 1]

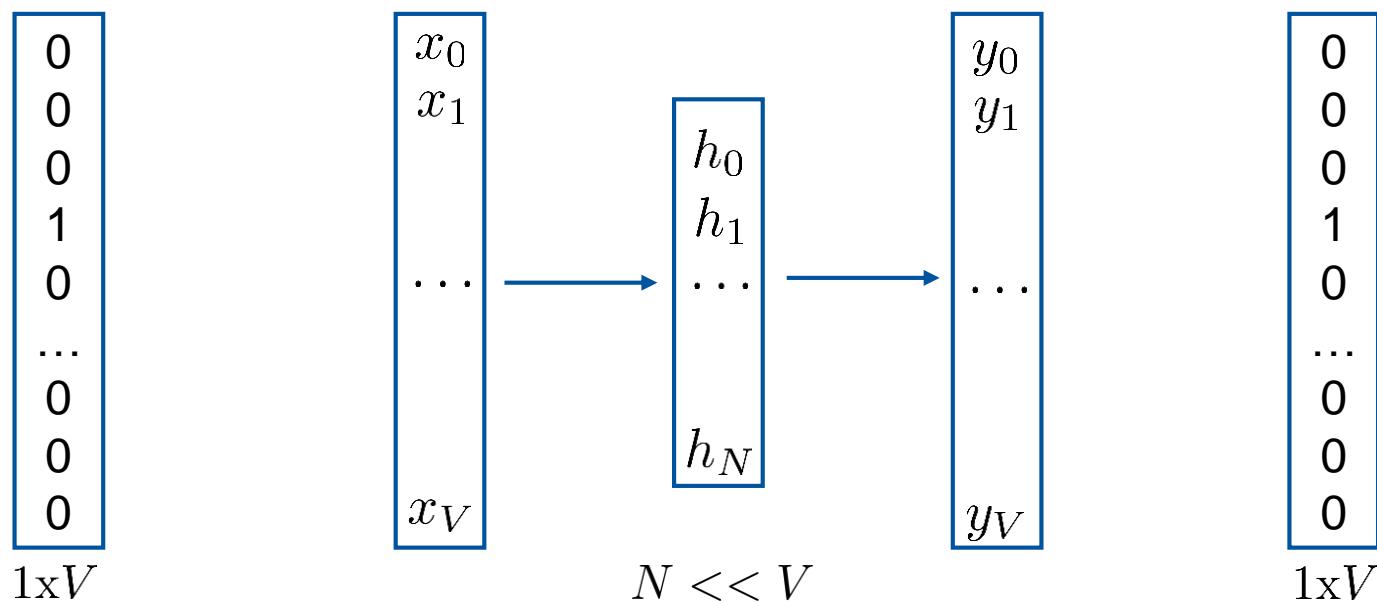
- Text mining: the number of possible values is the **size of the vocabulary**

Learning a Representation

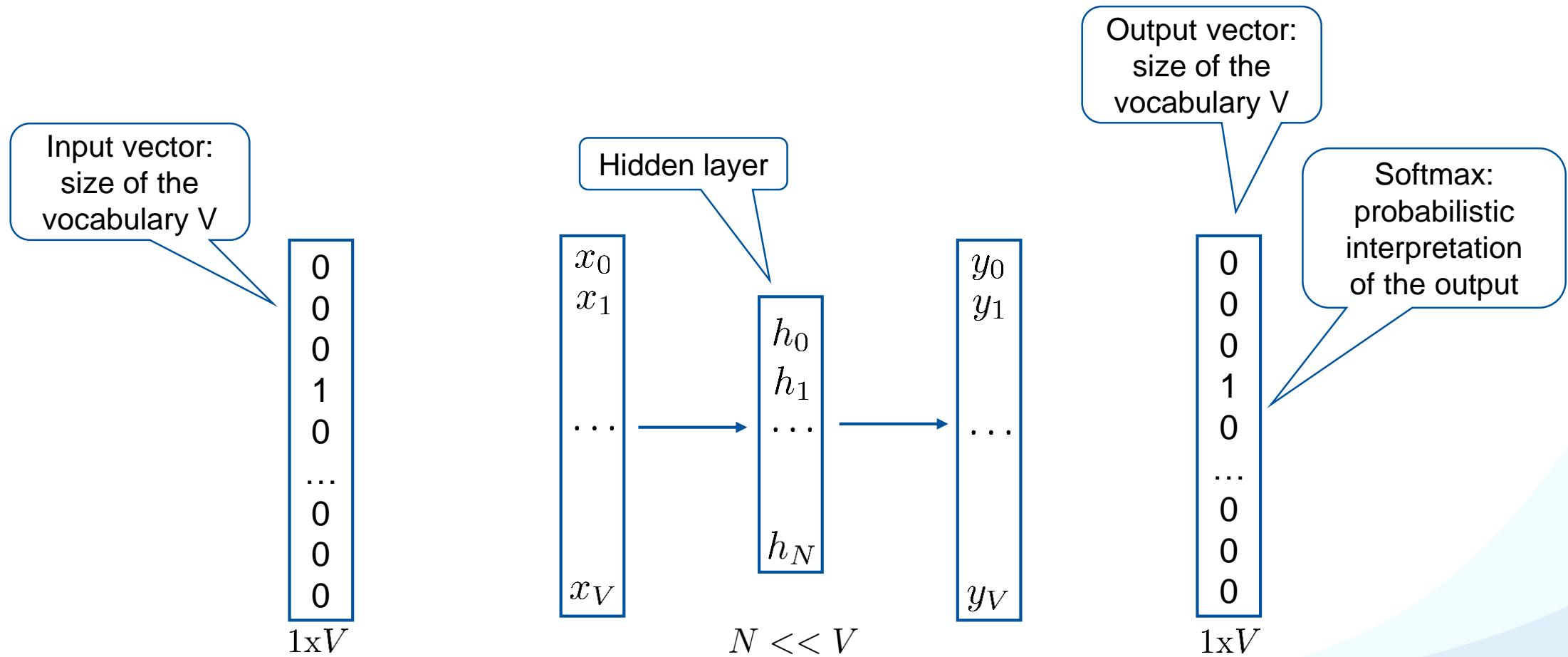
- Solution: use machine learning to automatically identify a smaller data representation
- Autoencoders use neural networks

Autoencoders

- Consider a neural network with a specific structure:
 - Input and output layers of dimension V
 - Hidden layer of dimension N with $N \ll V$



Autoencoders



Autoencoders

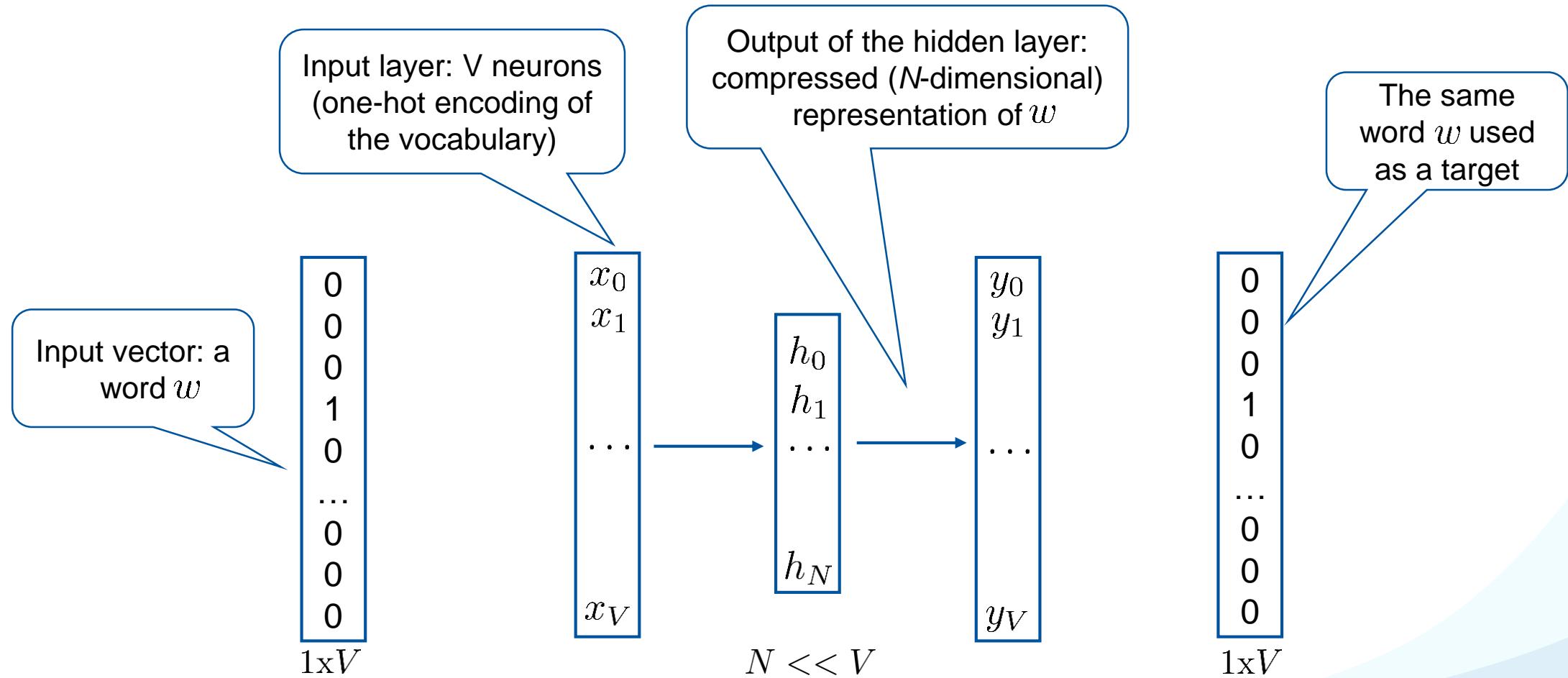
Softmax: transform the output of the NN into interpretable probabilities



Autoencoders - Training

- Train network with the same data in the input and output
 - Convert the corpora into one-hot encoded vectors
 - Feed these one by one into the neural network
 - Perform backpropagation comparing the output with the input
- We obtain a network that outputs (almost) the same one hot encoded vector given as input

Autoencoders

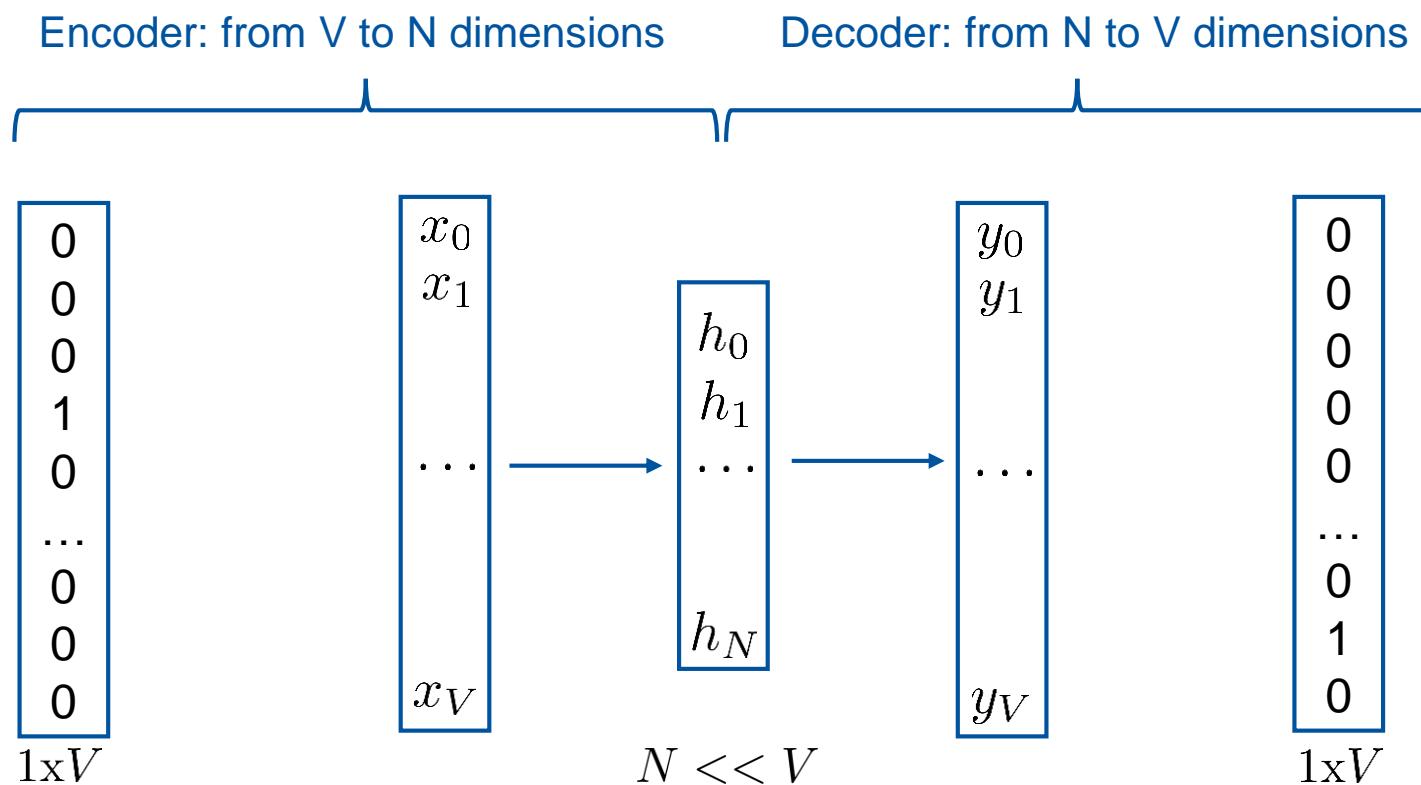


Autoencoders

- The output of the **hidden layer** gives us a **compressed representation** of the input word
- The compression ratio is V/N
- These neural networks are called **autoencoders**
→ one way to automatically learn a representation of data

Autoencoders

It is possible to split an autoencoder in the **encoding** and **decoding** part after the training

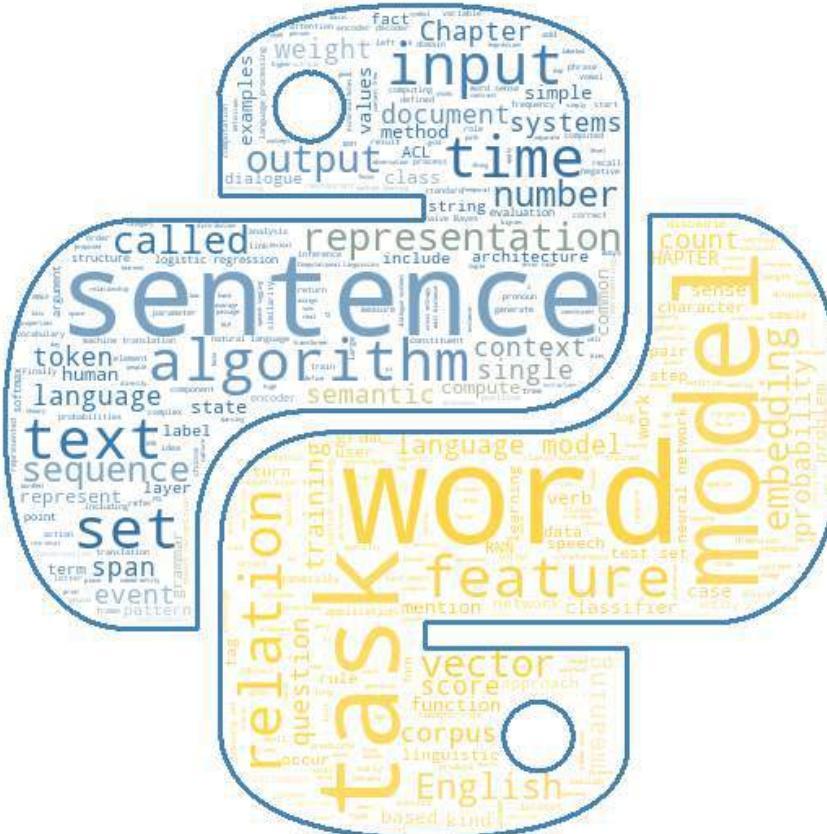


Learning a Representation

- Autoencoders are one way to automatically learn a representation of data
- Learning a smaller data representation is often called [embedding](#)
- When applied to a text, it's referred to as [word embeddings](#)

Text Mining

1. Introduction to Text Mining
 2. Text Preprocessing
 3. Modeling
 4. Enriching Text: Linked Data
 5. N-gram Model
 6. Learning a Representation
 7. **Word2vec**



Word Embeddings

- Compression is very useful, but [autoencoders](#) do not incorporate context
- N-grams allow to consider the [word order](#) and the [context](#)
- Can we obtain a word [embedding](#) that contains order and context?
- → use an N-grams generalization: [skip-grams](#)

Skip-grams

K-skip N-grams

- A skip-gram is an N-gram that allows to skip words
- Skip-grams constructed for a certain skip distance K allow a total of K or less skips for N-grams
- Example:
 - A 3-skip-gram includes: 3 skips, 2 skips, 1 skip or no skip (in total)
 - A 3-skip-trigram (x,y,z) covers “xyz”, “x_yz”, “xy_z”, “x_y_z”, “x__yz”, “xy__z”, “x__y_z”, “x_y__z”, “x___yz”, “xy____z” in a document

Skip-grams - Example

"Hi Jen did you eat the cake?"

1-skip trigrams: ['Hi Jen did', 'Hi Jen you', 'Jen did you', 'Jen did eat', 'Hi did you', 'did you eat', 'did you the', 'you eat cake', ...]

2-skip trigrams: ['Hi Jen eat', 'did you cake', 'Jen you the', ...]

Recall: 2-skip-trigrams also include all 1-skip-trigrams.

Skip-grams - Example

- Skip-grams are can associate a more **general** notion of a context compared to N-grams
- The surrounding context can be partially skipped:
this implies less **overfitting** when used in a learning phase

Word2vec

- Word2vec: one of the most popular techniques to learn word embeddings using shallow neural networks
- Idea: an extension of the autoencoder method
 - Word2vec learns a compressed representation of a word NOT from itself, but from its context
- Step 1: build a training set extracting skip-grams from a text
 - For each word, consider the context in a sliding window around it
 - Create tuples with the word and every skip-gram in the window that does not contain it

Word2vec – Training Set

Example with 2-skip trigrams and a window size of 2 on each side

The little goose swims on a large pond.

→ (the, [<s>, <s>, little]), (the, [<s>, <s>, goose]),
(the, [<s>, little, goose]),

The little goose swims on a large pond.

→ (little, [the, goose, swims]), (little, [<s>, the, goose]),
(little, [<s>, the, swims]), (little, [<s>, goose, swims])

The little goose swims on a large pond.

→ (goose, [the, little, swims]), (goose, [the, little, on]),
(goose, [the, swims, on]), (goose, [little, swims, on])

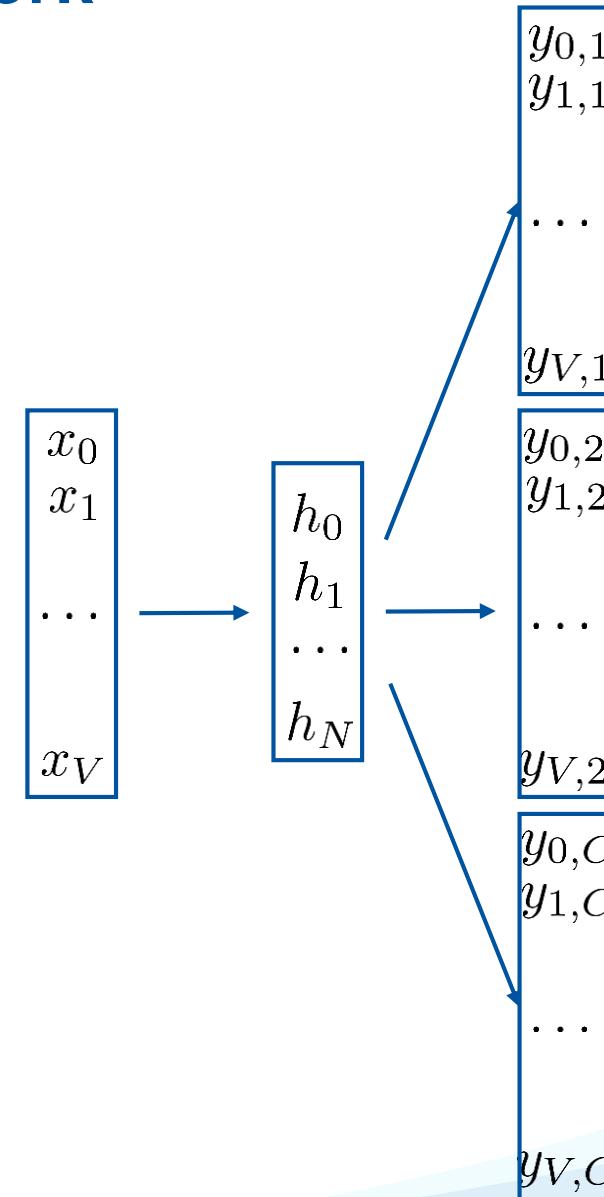
The little goose swims on a large pond.

→ (swims, [little, goose, on]), (swims, [little, goose, a]),
(swims, [little, on, a]), (swims, [goose, on, a])

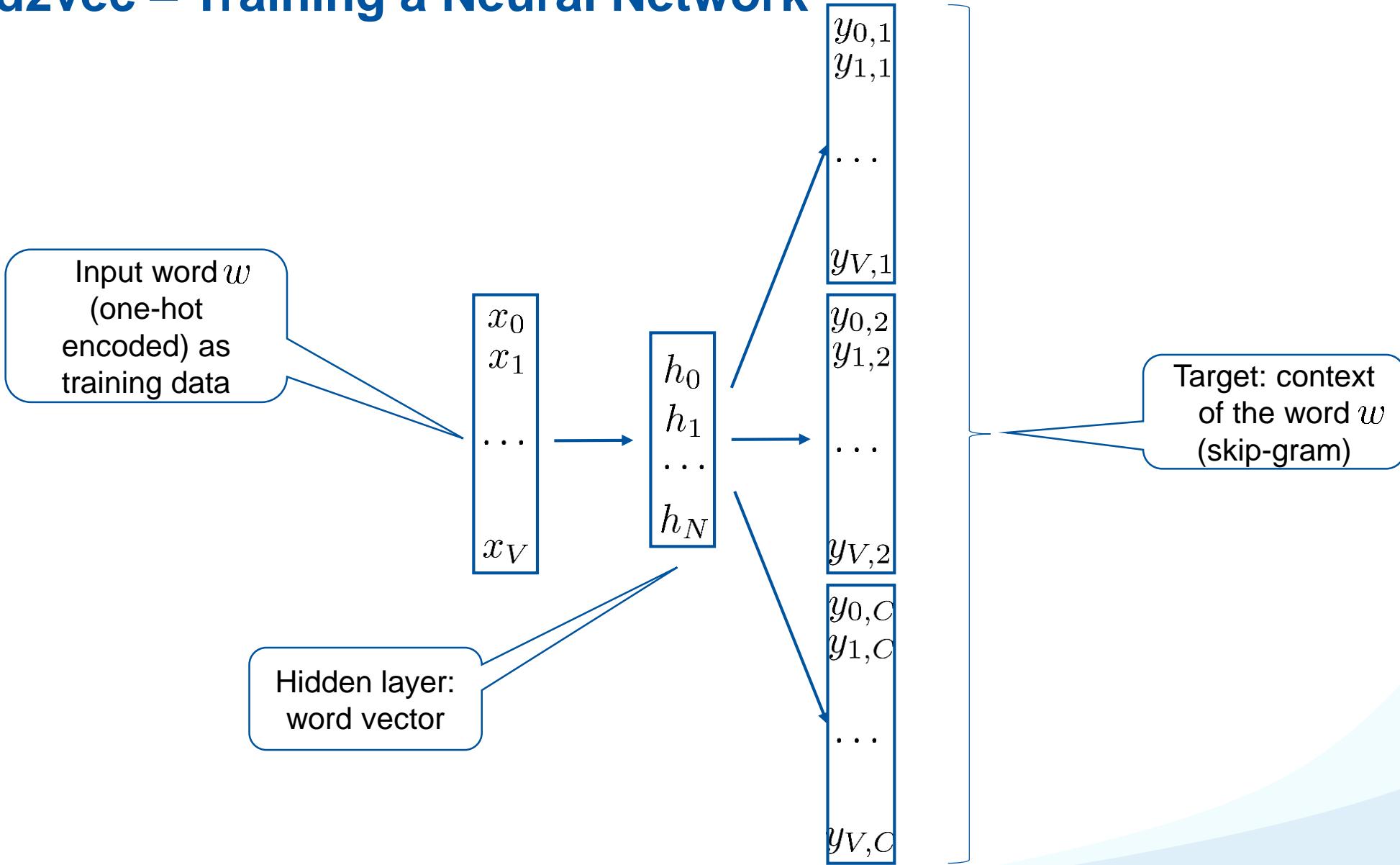
The context has fixed length representation and its order matters!

Word2vec – Training a Neural Network

- Step 2: train a neural network with a word as input and its context as output (using the tuples we built in Step 1)
- This is what the network looks like for our example training set (using skip-trigrams)



Word2vec – Training a Neural Network



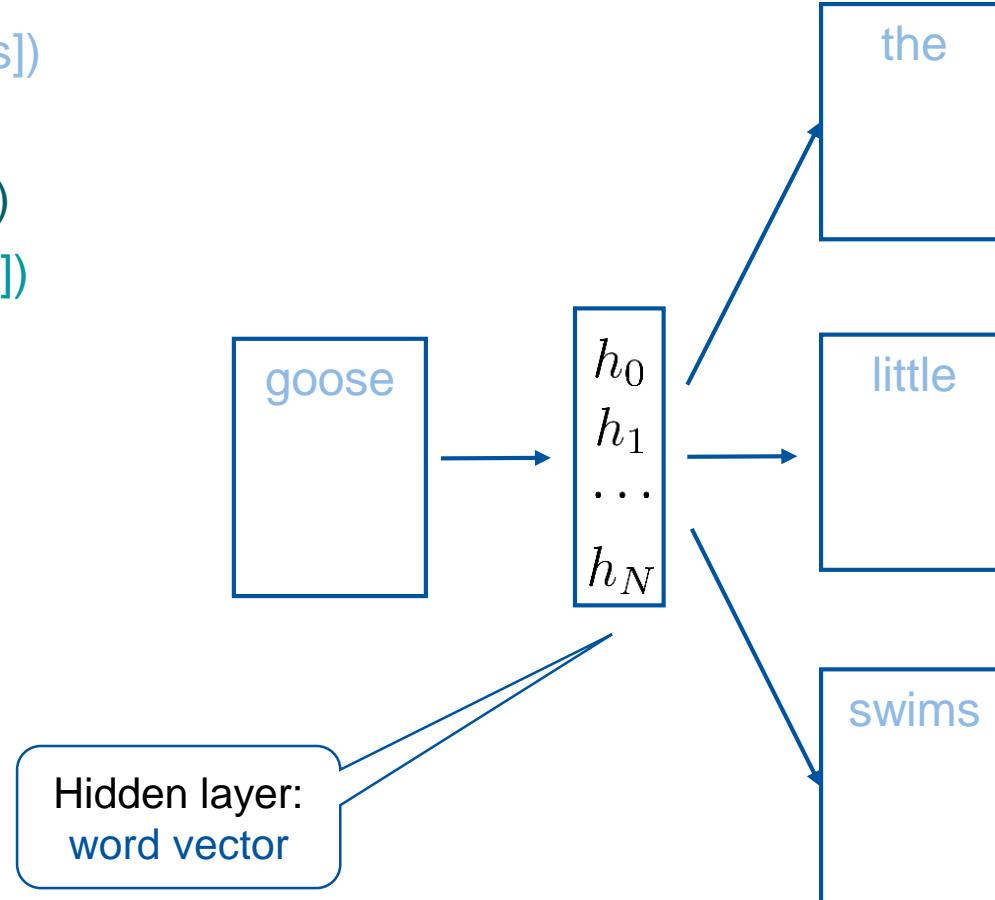
Word2vec – Training Example

(goose, [the, little, swims])

(goose, [the, little, on])

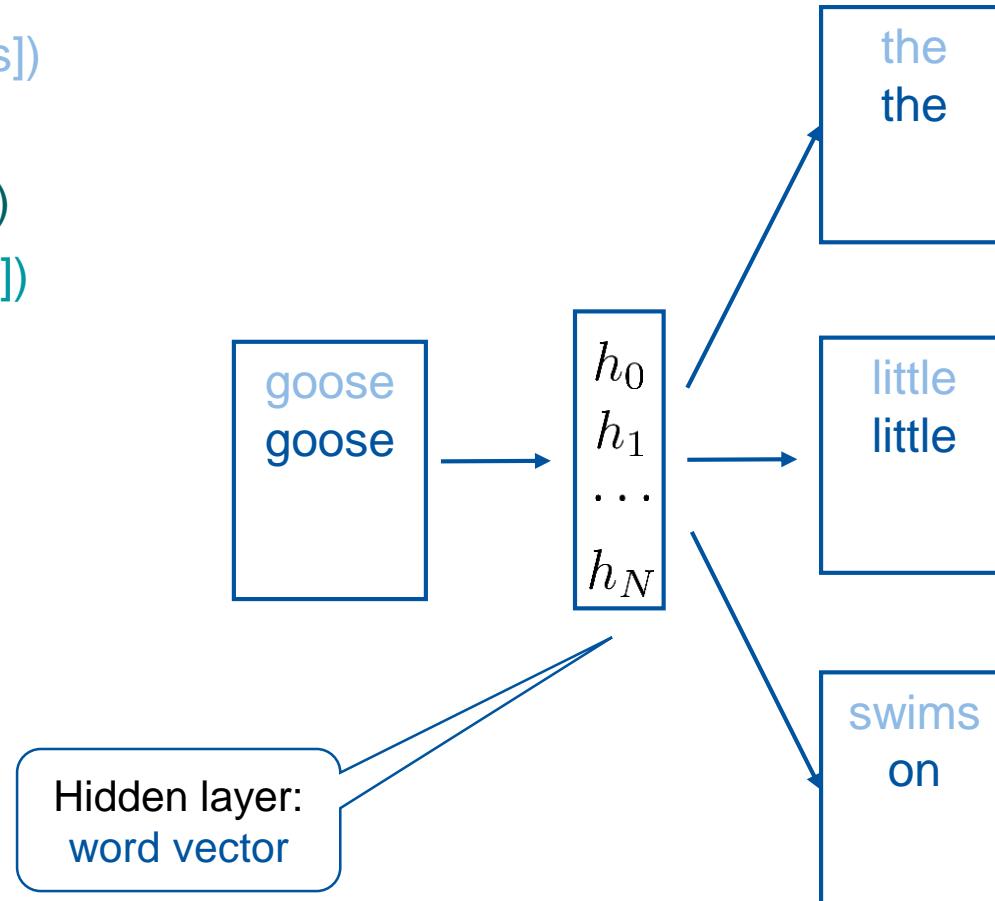
(goose, [the, swims, on])

(goose, [little, swims, on])



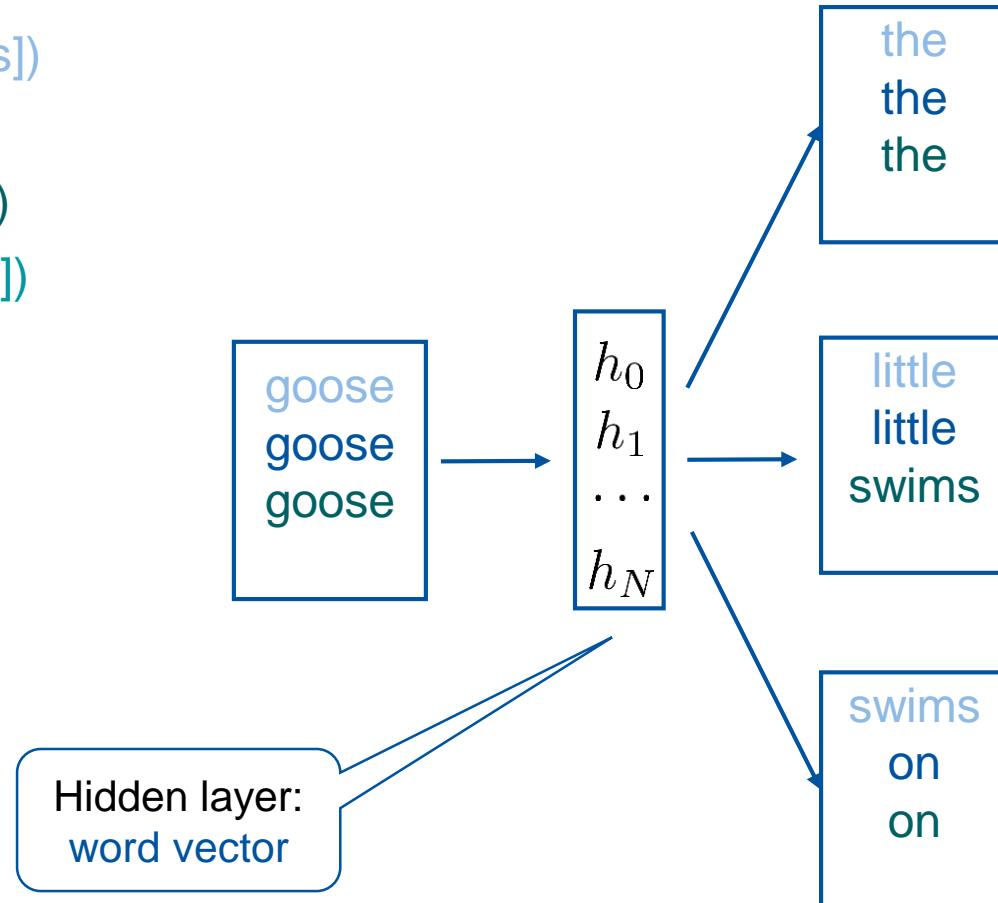
Word2vec – Training Example

(goose, [the, little, swims])
(goose, [the, little, on])
(goose, [the, swims, on])
(goose, [little, swims, on])



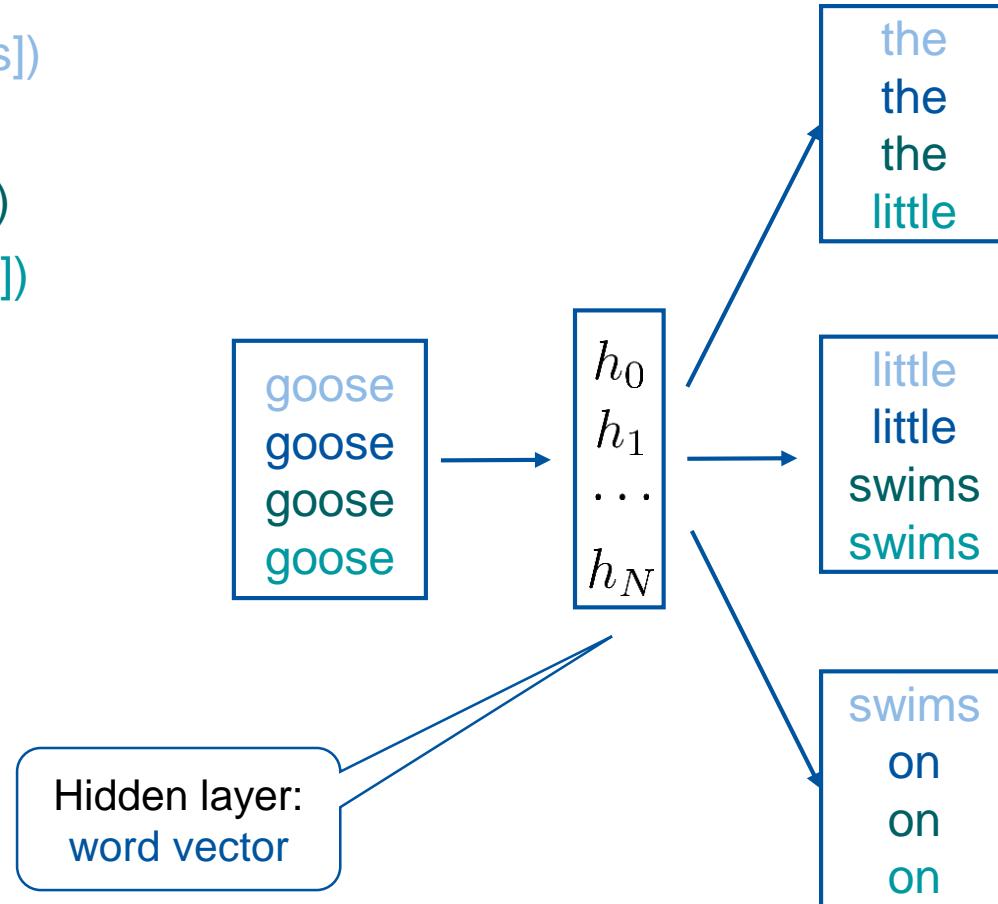
Word2vec – Training Example

(goose, [the, little, swims])
(goose, [the, little, on])
(goose, [the, swims, on])
(goose, [little, swims, on])



Word2vec – Training Example

(goose, [the, little, swims])
(goose, [the, little, on])
(goose, [the, swims, on])
(goose, [little, swims, on])



Word2vec – Training Example

(goose, [the, little, swims])

(goose, [the, little, on])

(goose, [the, swims, on])

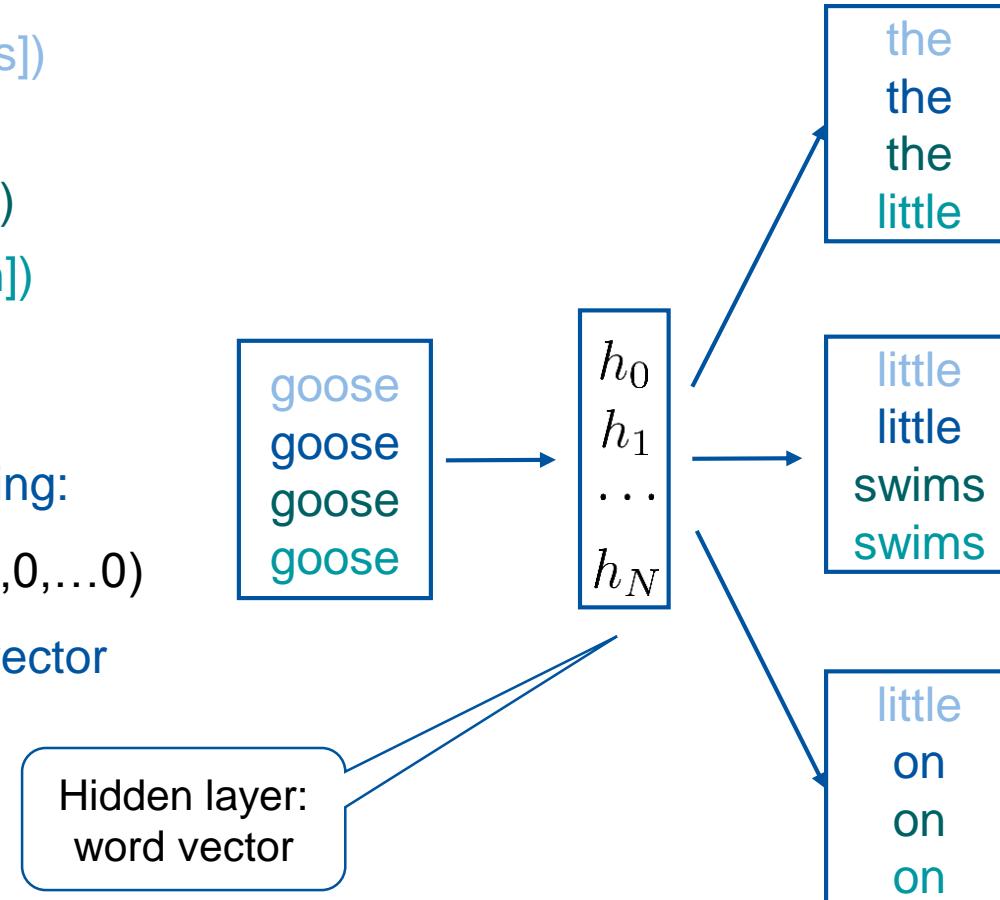
(goose, [little, swims, on])

Hidden Layer after training:

brown = (0,0,0,0,0,0,0,1,0,...0)

is mapped to the word vector

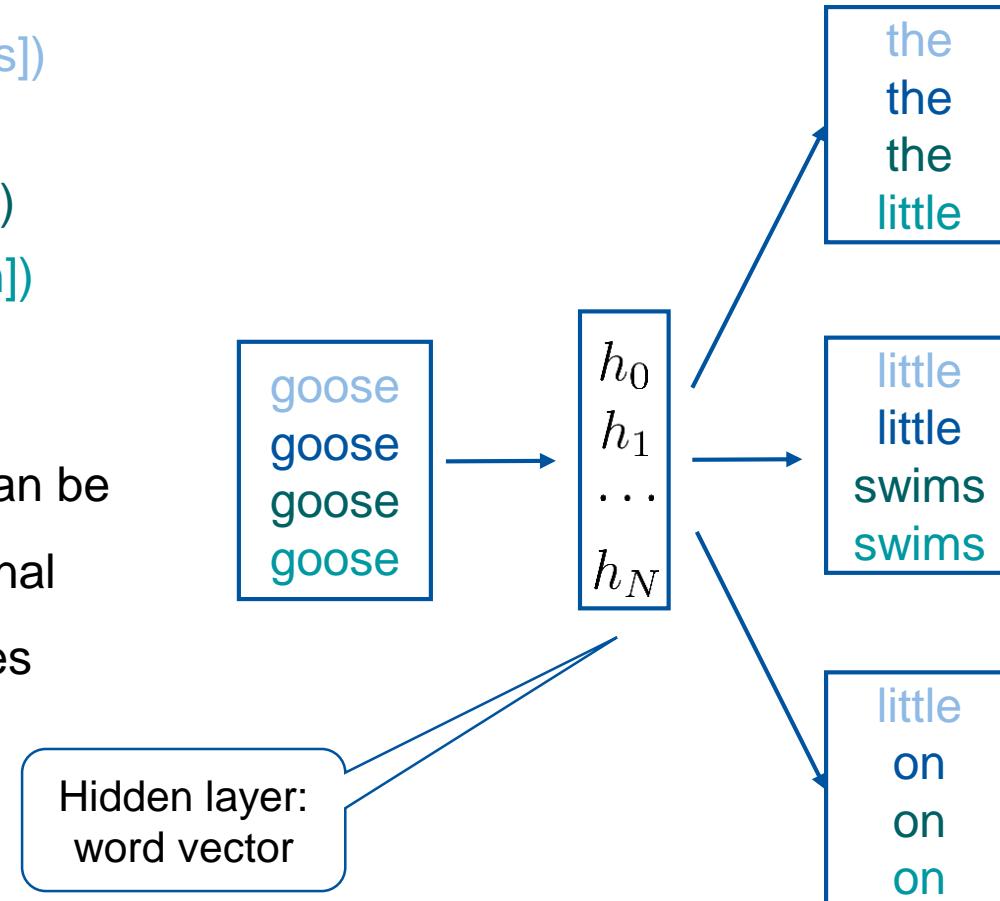
(0.23, 0.74, 0.10)



Word2vec – Training Example

(goose, [the, little, swims])
(goose, [the, little, on])
(goose, [the, swims, on])
(goose, [little, swims, on])

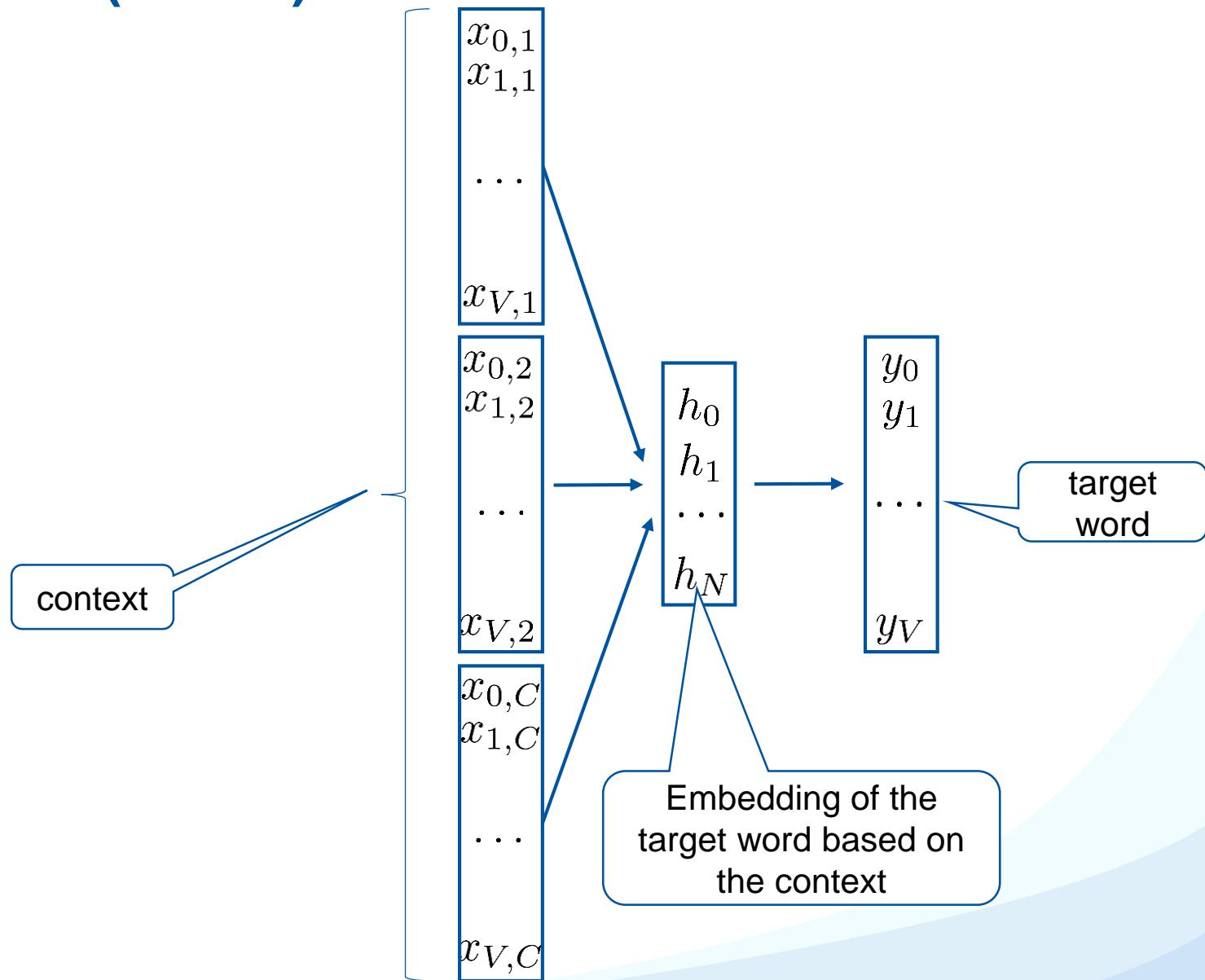
Resulting word vector can be used as lower dimensional input for other techniques



Continuous Bag of Words (CBoW)

Another Variant

The input is the context and the output is the word



Word2vec – Advantages

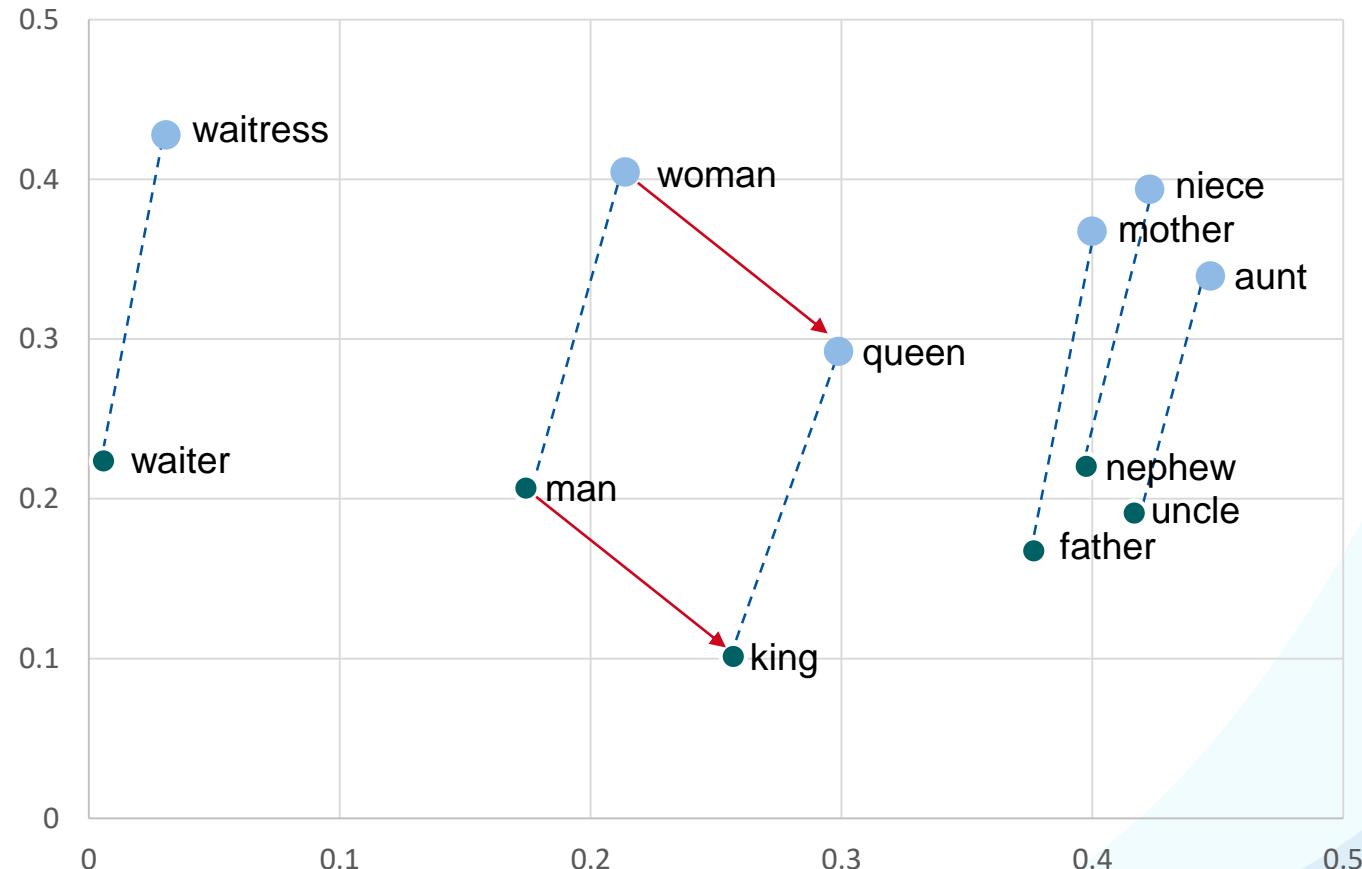
- Word2vec captures the meaning of a word using **its context**
- Returns a powerful and meaningful word representation
 - Vectors for similar words are close to each other
 - It's possible to **add and remove context with vector operations**

$$\text{king} - \text{man} + \text{woman} = \text{queen}$$

$$\text{Father} - \text{man} + \text{woman} = \text{mother}$$

$$\text{Waitress} - \text{woman} + \text{man} = \text{waiter}$$

...



Doc2vec

- Same principles as Word2vec
- Uses **one-hot encoded documents**
- **Training set** of tuples: (one-hot encoded document, skip-gram from that document)

Doc2vec

Example with **2-skip trigrams** (but any feature is possible):

0001 'Cats are the only pet of the felines family, while dogs are canids.'

0010 'Cats are the third-most popular pet in the US.'

0100 'Dogs have been selected for millennia as pet animals.'

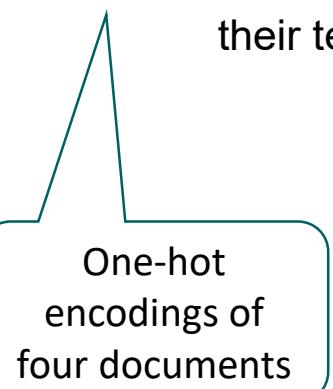
1000 'Normally, dogs are not aggressive towards other dogs outside their territory.'

(0001, [<s>, cats, are]), (0001, [cats, are, the]),
(0001, [cats, are, only]), ...

(0010, [<s>, cats, are]), (0010, [cats, are, the]),
(0010, [cats, are, third]), ...

(0100, [<s>, dogs, have]), (0100, [dogs, have, been]),
(0100, [dogs, have, selected]), ...

(1000, [<s>, normally, dogs]),
(1000, [normally, dogs, are]),
(1000, [normally, dogs, not]), ...



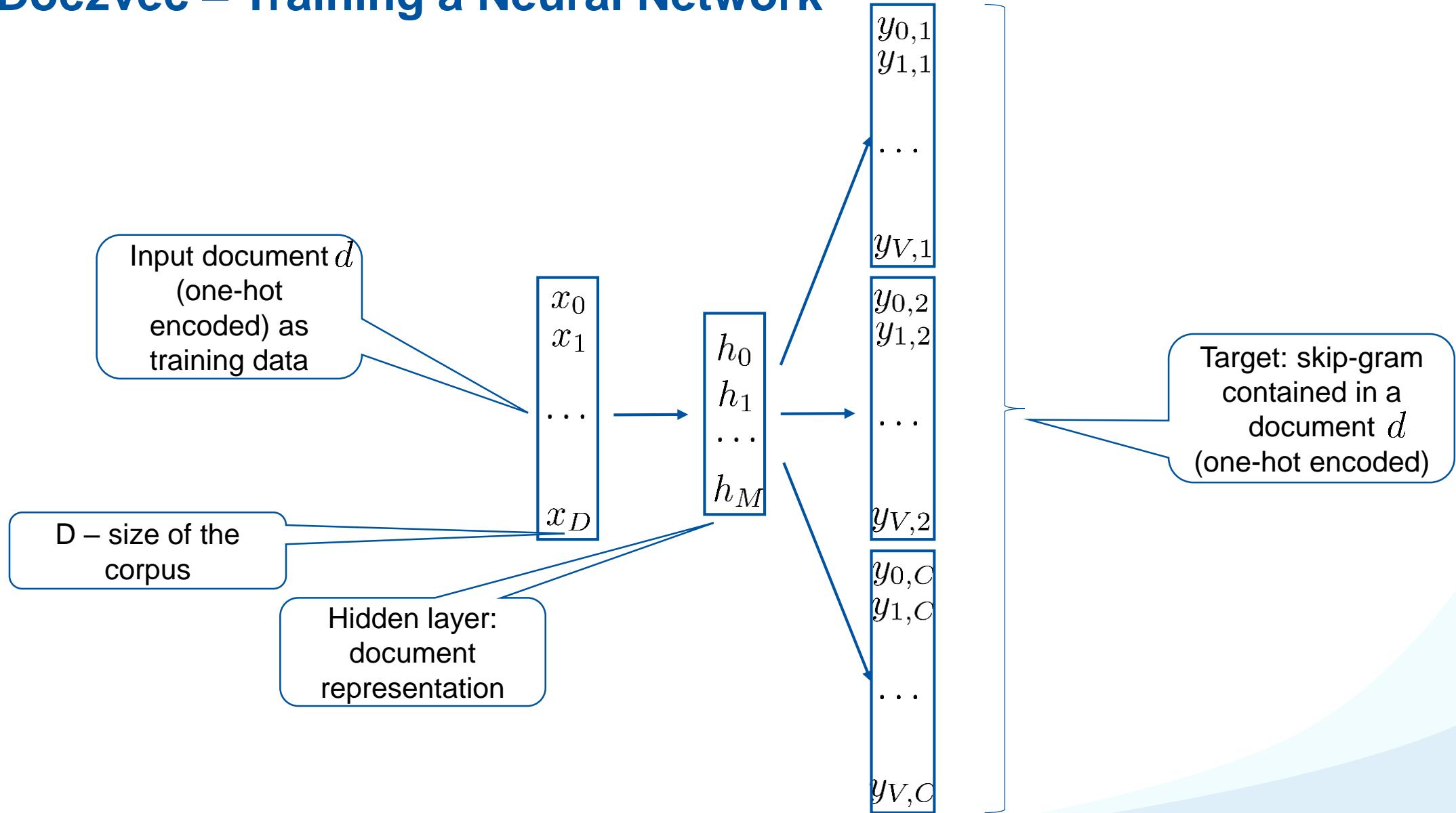
Doc2vec – Training a Neural Network

Learn a representation for a document by training a neural network

- Same architecture as before
- the input is the one-hot encoding of the document
- the output is a skip-gram of the document

→ Hidden layer gives a compact representation of the document

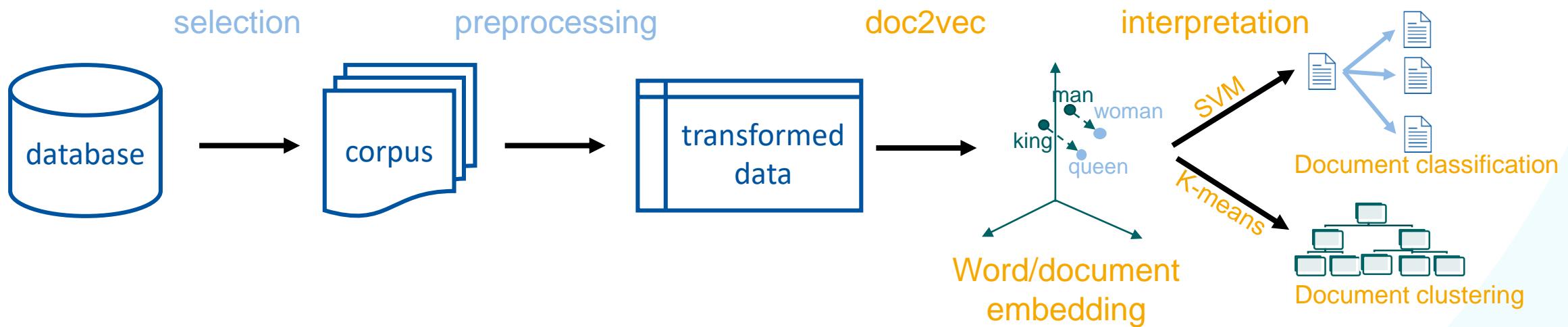
Doc2vec – Training a Neural Network



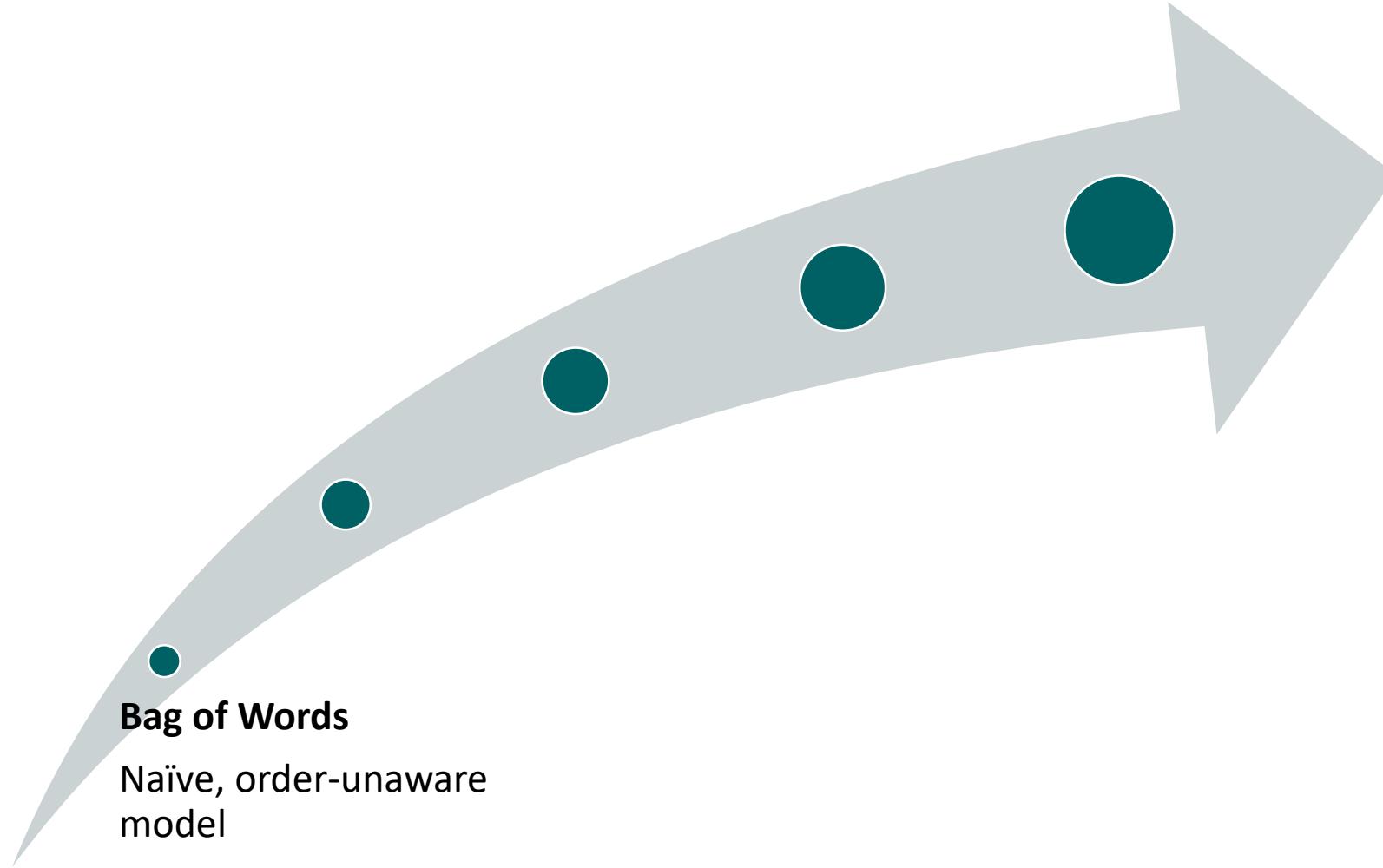
Doc2vec

- Doc2vec: find a **low-dimensional, expressive, powerful** and **context-aware** vector representation for documents in the corpus
- Considers the problems we have seen previously:
 - Keeps track of the **order** of words
 - Avoid the problem of **high sparseness** of textual data
 - Gives a **fixed length representation** of the documents
 - suitable as input for other data science applications (e.g. clustering)
- Many other variations possible...

Text Mining Pipeline With Doc2vec



Outlook



Outlook

N-Grams

Able to account for
order and context



Bag of Words

Naïve, order-unaware
model



Outlook

N-Grams

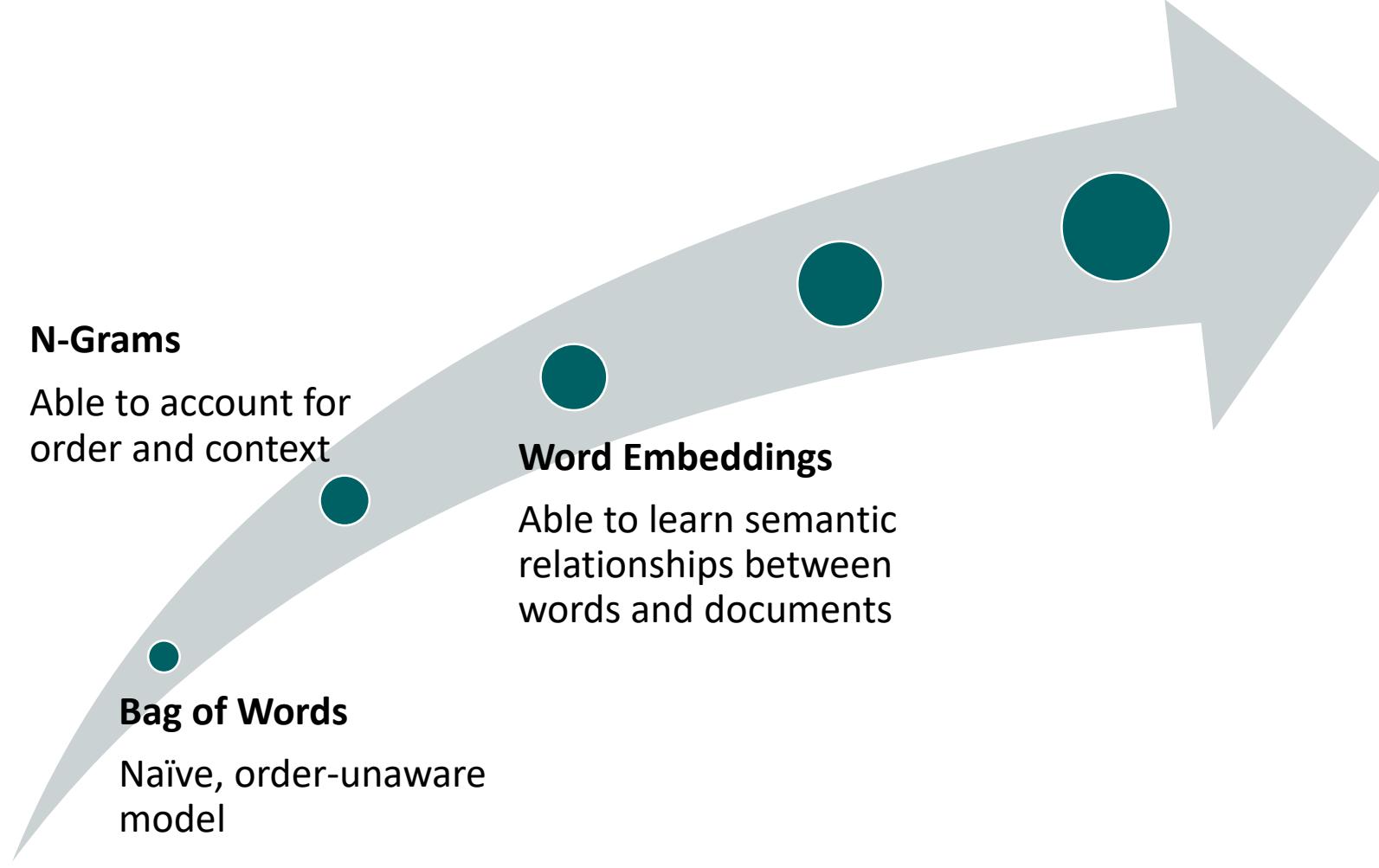
Able to account for
order and context

Word Embeddings

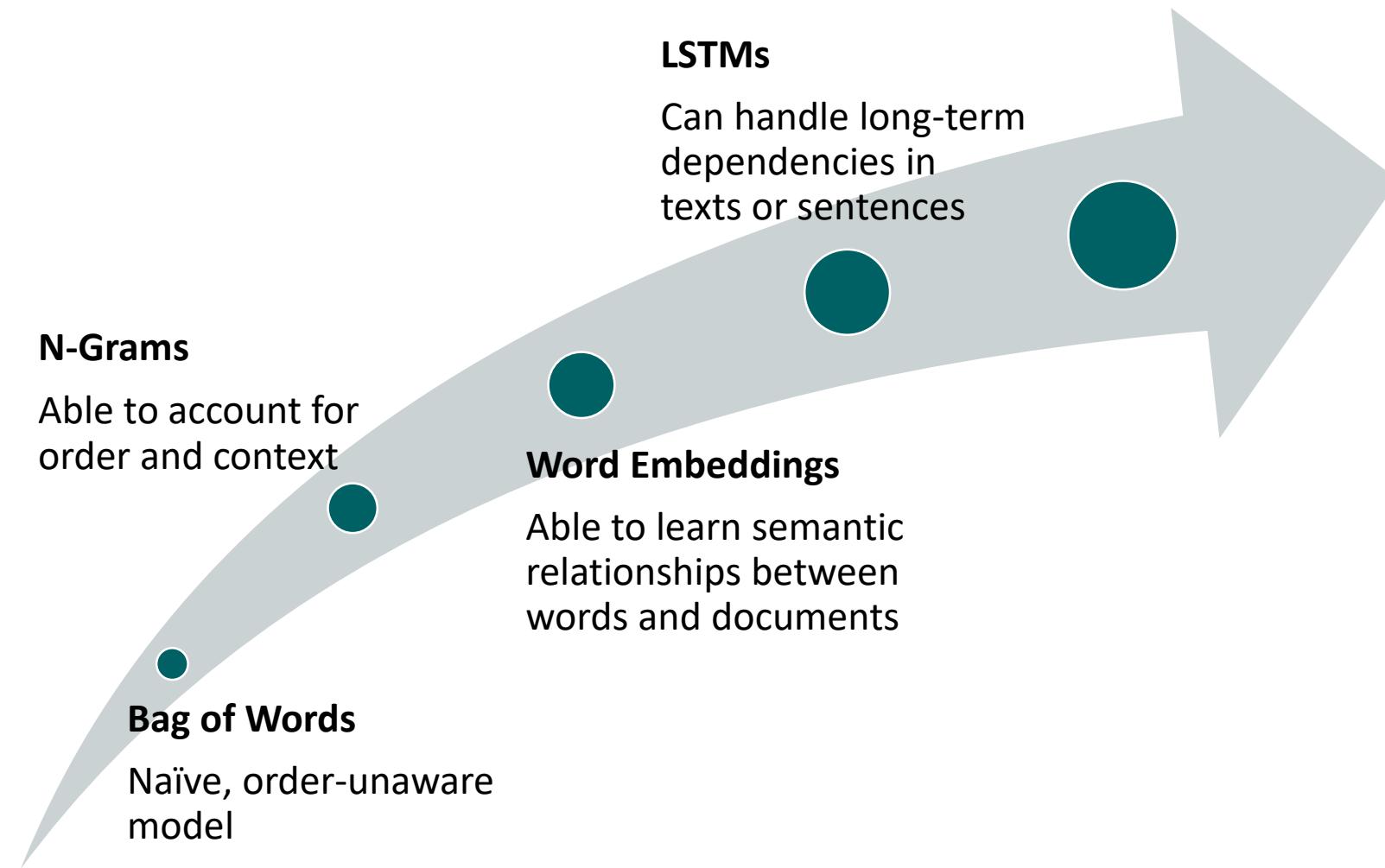
Able to learn semantic
relationships between
words and documents

Bag of Words

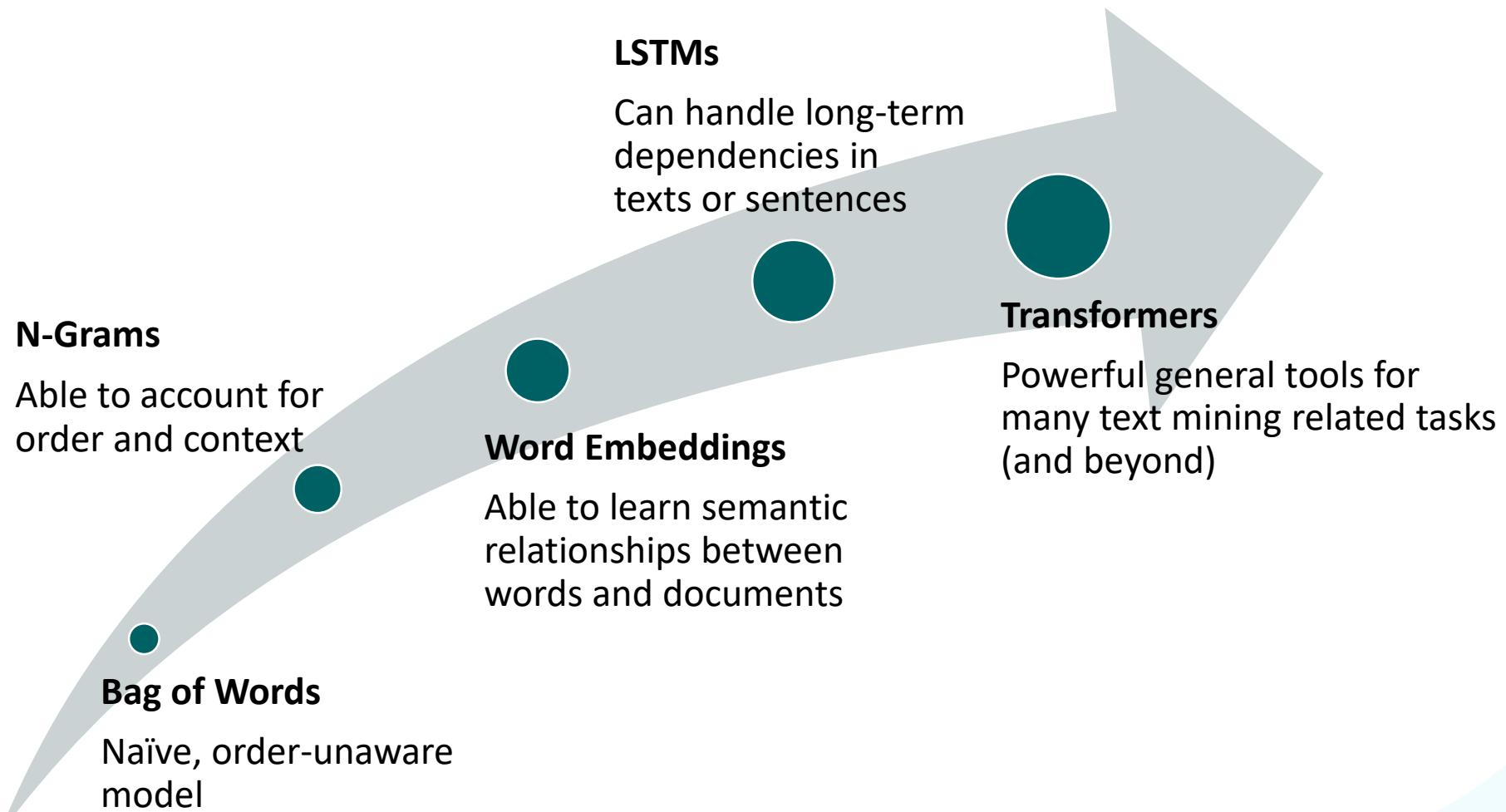
Naïve, order-unaware
model



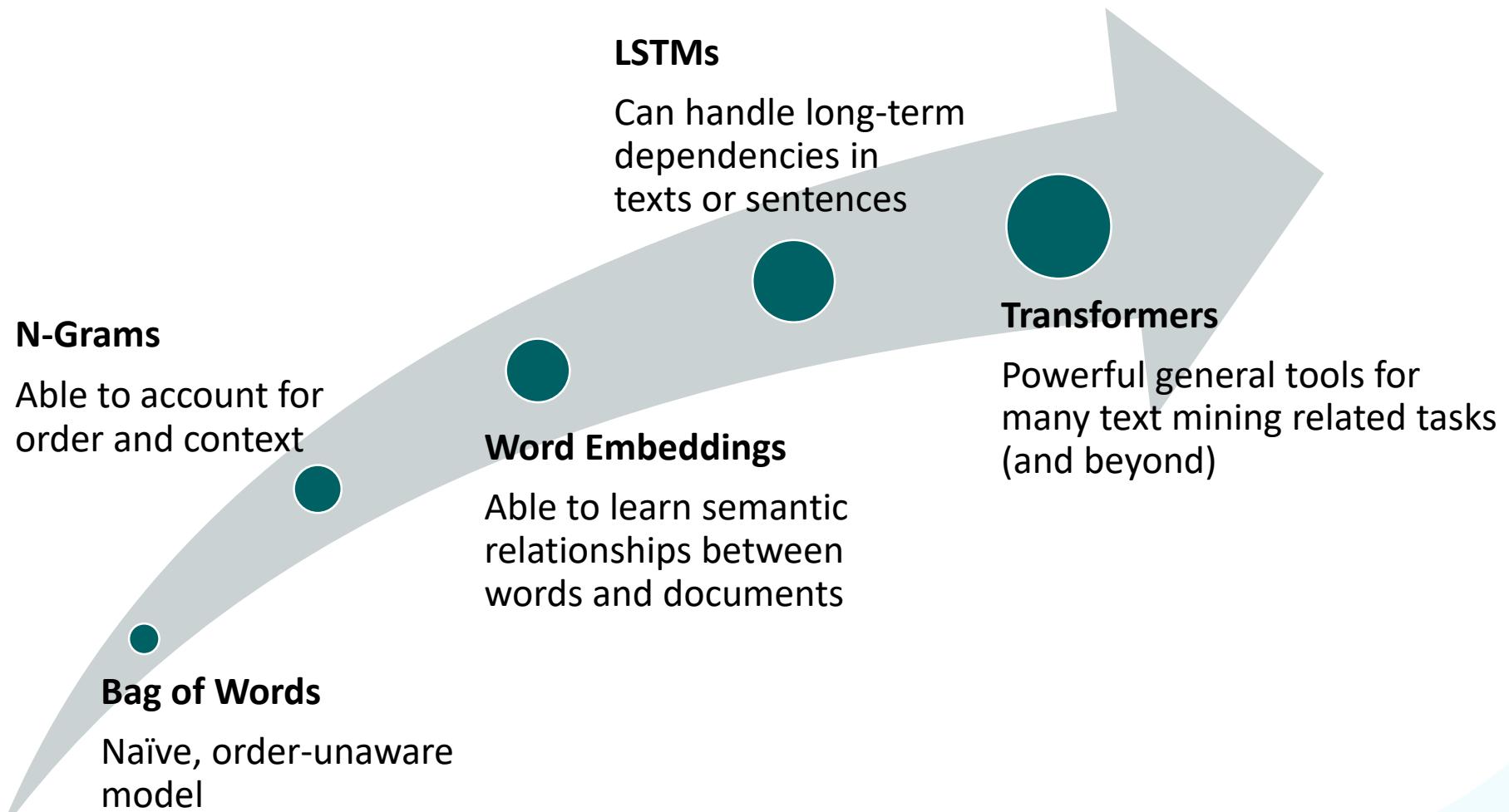
Outlook



Outlook



Outlook



Next up: **Responsible Data Science**

Elements of Machine Learning & Data Science

Responsible Data Science

Lecture 22

Prof. Wil van der Aalst

Marco Pegoraro, M.Sc.

Nina Graves, M.Sc.

Part I: Introduction to RDS

Fairness, accuracy, confidentiality, transparency

Part II: Confidentiality

Risks, encryption, anonymization, quasi-identifiers, K-Anonymity, L-Diversity, and T-Closeness

Part III: Fairness

Fairness measures, itemsets/association rules revisited, effect (rule/outcome), making decision trees fair

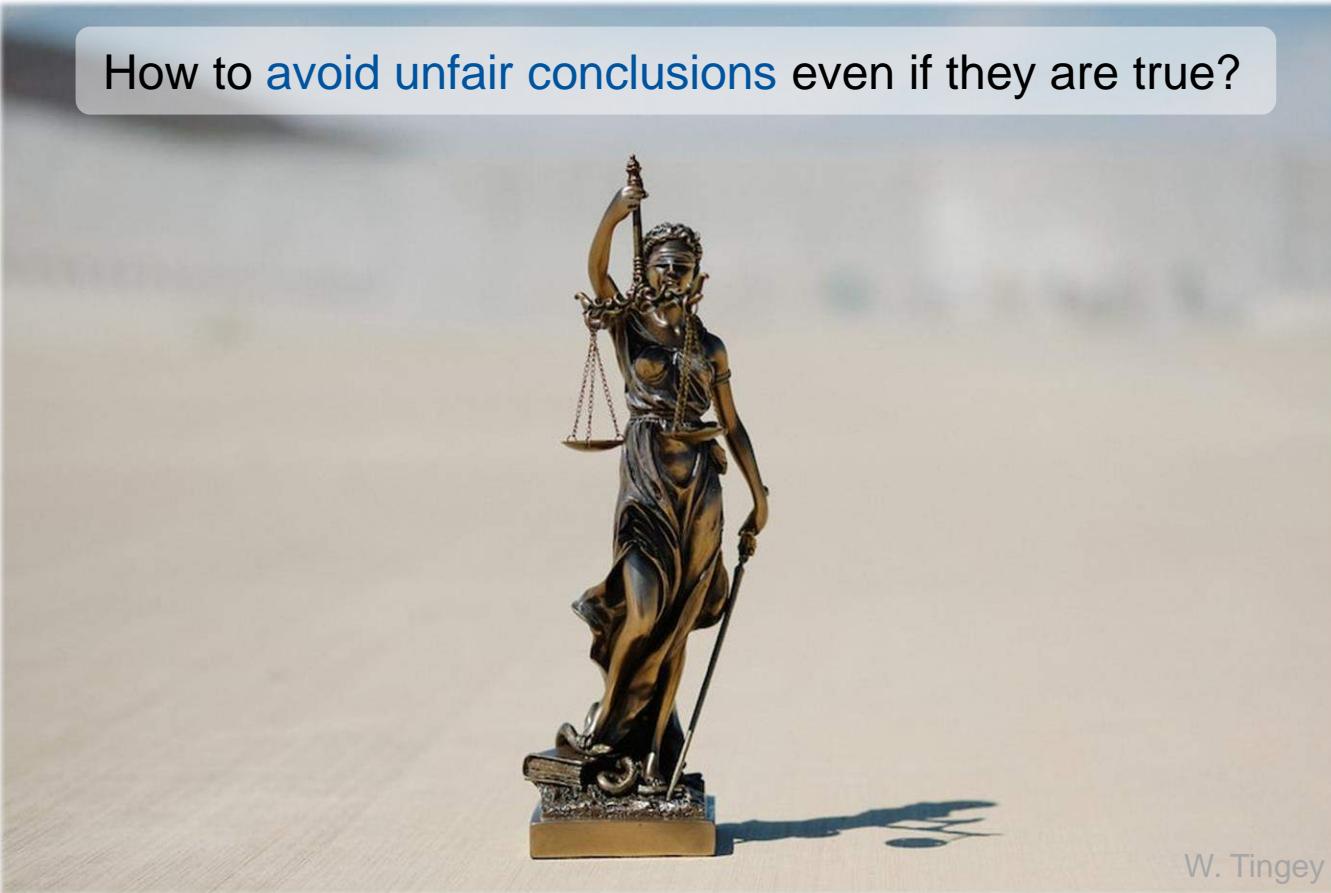
Part I: Introduction to Responsible Data Science

Responsible Data Science

- Fairness
- Accuracy
- Confidentiality
- Transparency



Fairness – Data Science Without Prejudice



Fairness – Data Science Without Prejudice



Banking



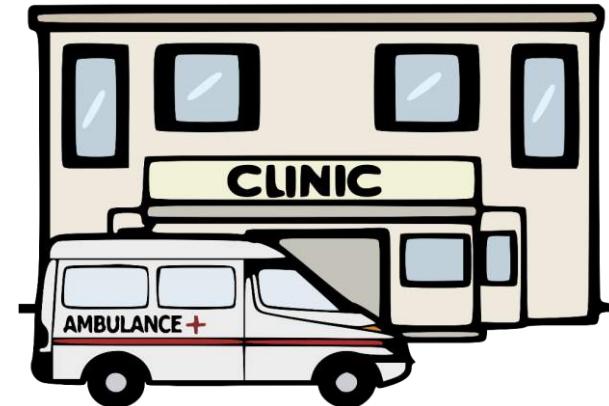
Insurance



Hiring



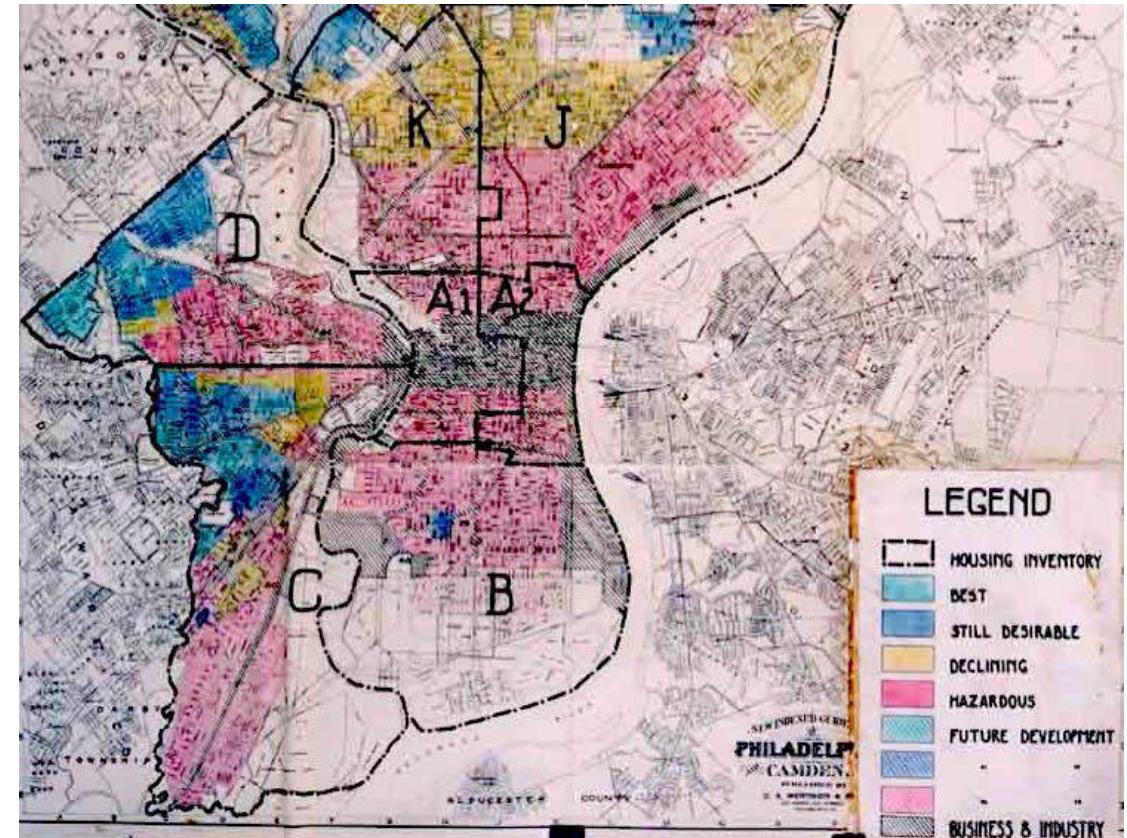
Admission



Health

Not New – Redlining

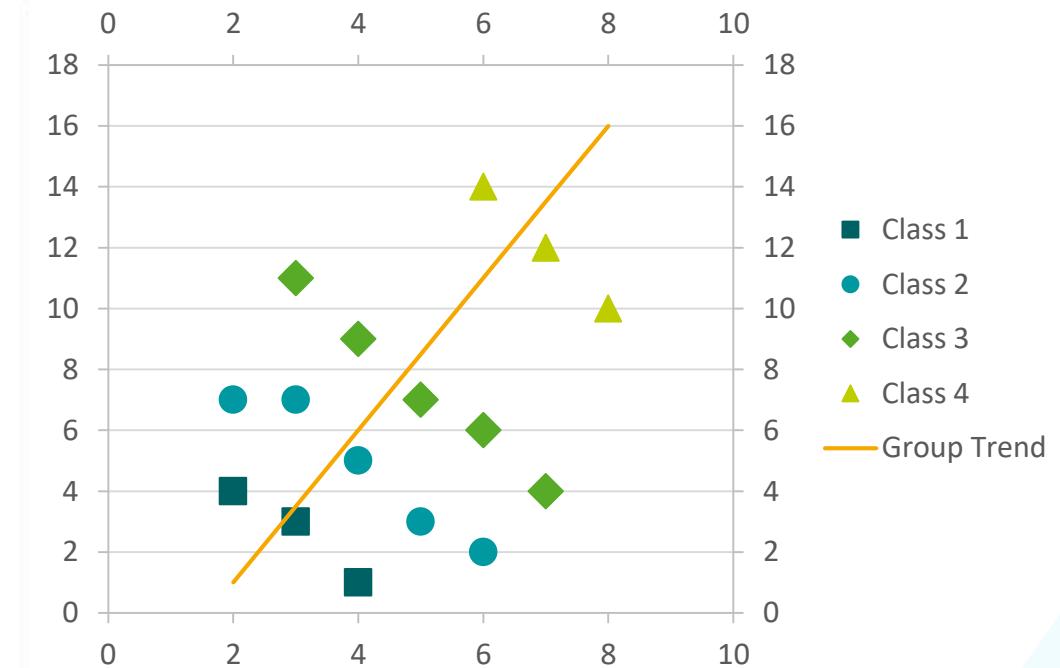
- Redlining, a discriminatory practice of denying affordable services based on geographical locations
- You can remove race, gender, age, etc., but if this correlates with your zip code...



1936 security map of Philadelphia showing redlining of lower income neighborhoods (estimated risk of mortgage loans)

Remember – Simpson's Paradox

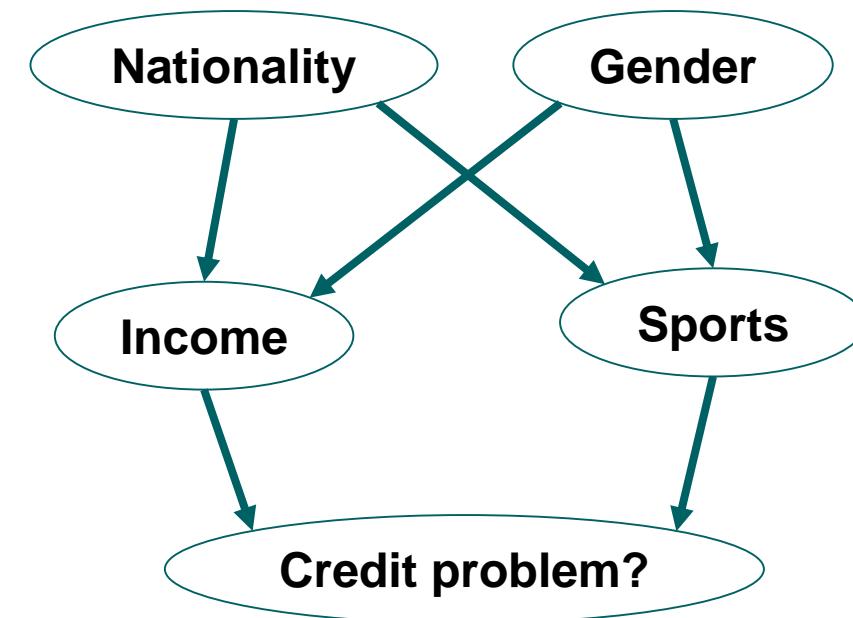
A trend appears in several different groups of data but disappears or reverses when these groups are combined.



Fairness – How to Avoid Unfair Conclusions?

Even if they are true...

- Removing sensitive features often does not work!
- Enforcing fairness may lead to **less accurate predictions**
- Not so easy to define fairness
 - Percentage of patients dying – academic vs regional hospitals, experienced vs inexperienced doctors, etc.
 - Waiting times for a resource – busy vs idle resources, part-time vs full-time, etc.



Fairness – How to Avoid Unfair Conclusions?

Even if they are true...

- There may be intentional or unintentional discrimination when making data-driven decisions
- Training data may be biased (wrong or outdated) or the sample may not be representative (never enough evidence for some groups)
- Even when the data are correct, optimizing for a particular target feature may lead to discrimination (accuracy for old cases does not imply fairness for new cases !)

Accuracy – Data Science Without Guesswork

How to answer questions with a [guaranteed level of accuracy](#)?



See “Computer says no” episode of Little Britain from 2004.
Contrastingly, “ML never says no”.

Accuracy

- ML algorithms **always** return a result, but:
 - The instance may be **close to a decision boundary**
 - There may be **too little training data**
- When testing many null hypotheses, just **by chance**, one will be rejected. **Carlo Emilio Bonferroni** already indicated that one needs to correct for this in 1936.



Generated using DALL-E 3

Accuracy – The Curse of Dimensionality

Find the terrorists

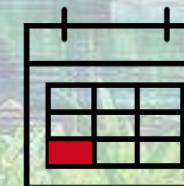
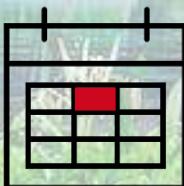


Assumptions:

- 18 million people in NL
- 1800 hotels
- 100 guests per hotel per night
- Hence, on average a person visits a hotel every 100 days

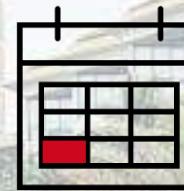
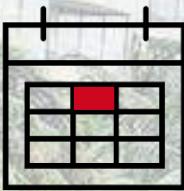
Suspicious event - two persons stay in the same hotel on two different dates

How many suspicious events in a 1000-day period (i.e., less than 3 years)?



Accuracy – Curse of Dimensionality

Suspicious event - two persons stay in the same hotel on two different dates



- The probability that two persons p_1 and p_2 visit a hotel on a given day (d): $\frac{1}{100} \times \frac{1}{100} = 10^{-4}$
- The probability that p_1 and p_2 visit the same hotel on the day (d): $10^{-4} \times \frac{1}{1800} = 5.55 * 10^{-8}$
- The probability that p_1 and p_2 visit the same hotel on two different dates: $(5.55 \times 10^{-8})^2$
-

Probability is 0.0000000000003086!

Accuracy – Curse of Dimensionality

- The probability that two persons p_1 and p_2 visit a hotel on a given day (d): $\frac{1}{100} \times \frac{1}{100} = 10^{-4}$
- The probability that p_1 and p_2 visit the same hotel on the day: $10^{-4} \times \frac{1}{1800} = 5.55 * 10^{-8}$
- The probability that p_1 and p_2 visit the same hotel on two different dates d_1 and d_2 : $(5.55 \times 10^{-8})^2 = 0.00000000000003086$

But how many suspicious events in a 1000-day period?

- Number of candidate events $(\{d_1, d_2\}, \{p_1, p_2\})$: $\binom{1000}{2} \times \binom{18 \times 10^6}{2} = 8.09 \times 10^{19}$
- Hence, the expected number of suspicious events is equal to $(5.55 \times 10^{-8})^2 \times 8.09 \times 10^{19}$
-

These are 249.750 events!

Curse of Dimensionality

- When looking for **patterns** in data (e.g., correlations) and the number of **possible patterns** is as large as the number of data points you have, then, by chance, some of these patterns **will be found!**
- In statistics, the **Bonferroni correction** is a method to counteract the “multiple comparisons problem”.
- Consider **statistical hypothesis testing**, which is based on rejecting the **null hypothesis** if the **likelihood of the observed data under the null hypothesis** is low (e.g., p-value is below 0.05).
- If many hypotheses are tested, then the probability that the null hypothesis is rejected ($p<0.05$) increases.
- In other words, **when the number of potential patterns is large compared to the number of instances, then you will find these patterns in the training data.**
- Related to **overfitting**: If the number of weights in the neural network is larger than the number of instances, then one will perfectly fit any training data. However, this does not mean anything.

Confidentiality – Data Science That Ensures Confidentiality

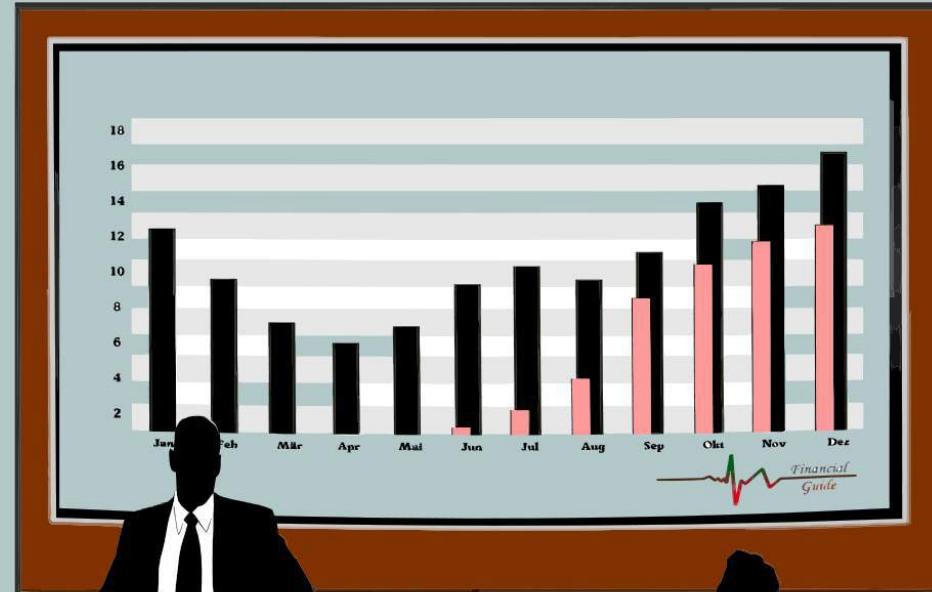


How to answer questions without revealing secrets?

If You are Not Paying, You Are the Product!

Social media

Search engines



General Data Protection Regulation (GDPR)

- The General Data Protection Regulation (GDPR) applies to member states of the European Union
- It came into effect on 25 May 2018
- Companies that are found guilty of misusing data can be fined up to €20 million or 4% of the company's annual turnover
- GDPR states that controllers must make sure it's the case that personal data is processed lawfully, transparently, and for a specific purpose
- This implies that people must understand why their data is being processed and how it is being processed

General Data Protection Regulation (GDPR)

- **Communication** – explain why user should leave personal information
- **Consent** – get clear consent to the processing of personal data
- **Data Transfer Outside the EU** – only when adequate level of protection is guaranteed
- **Sensitive Data** – ensure specific safety for sensitive data like race
- **Access** – users should have access to their information
- **Profiling** – individuals have the right to appeal against the decisions when it is based on automated processing
- **Erase Data** – users can request to delete data
- **Warnings** – companies have to notify authorities about data breaches
- **Marketing** – people should be able to give up direct marketing that uses their data



Another AI Regulation: Artificial Intelligence Act (AI Act)

(not yet enforced, more related to fairness)



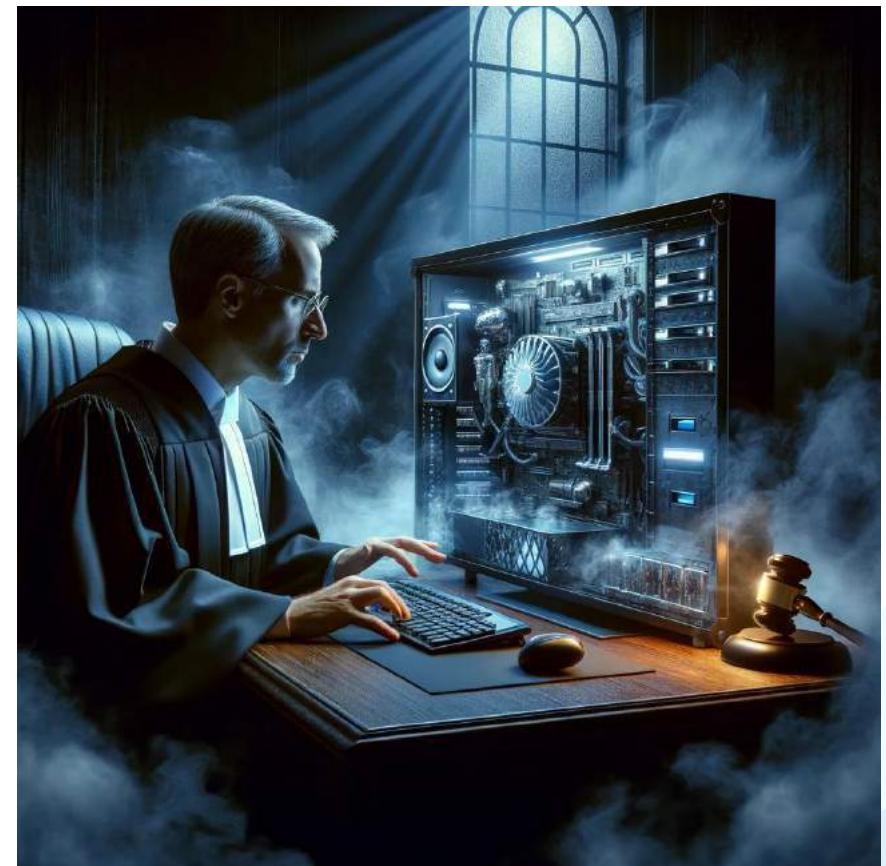
- It aims to **classify** and **regulate** artificial intelligence applications based on their **risk of causing harm**.
- The classification includes four categories of risk ("unacceptable", "high", "limited" and "minimal") plus one additional category for general-purpose AI (e.g., foundation models like GPT).

unacceptable	prohibited	Social scoring, biometric identification and categorization of people, real-time and remote biometric identification systems, such as facial recognition, etc.
high	conformity assessment	Employment, HRM, education, critical infrastructure, law enforcement, etc.
limited	transparency obligation	Chatbots, deepfakes, etc.
minimal	no restrictions	Spam filters, video games and inventory-management systems, etc.

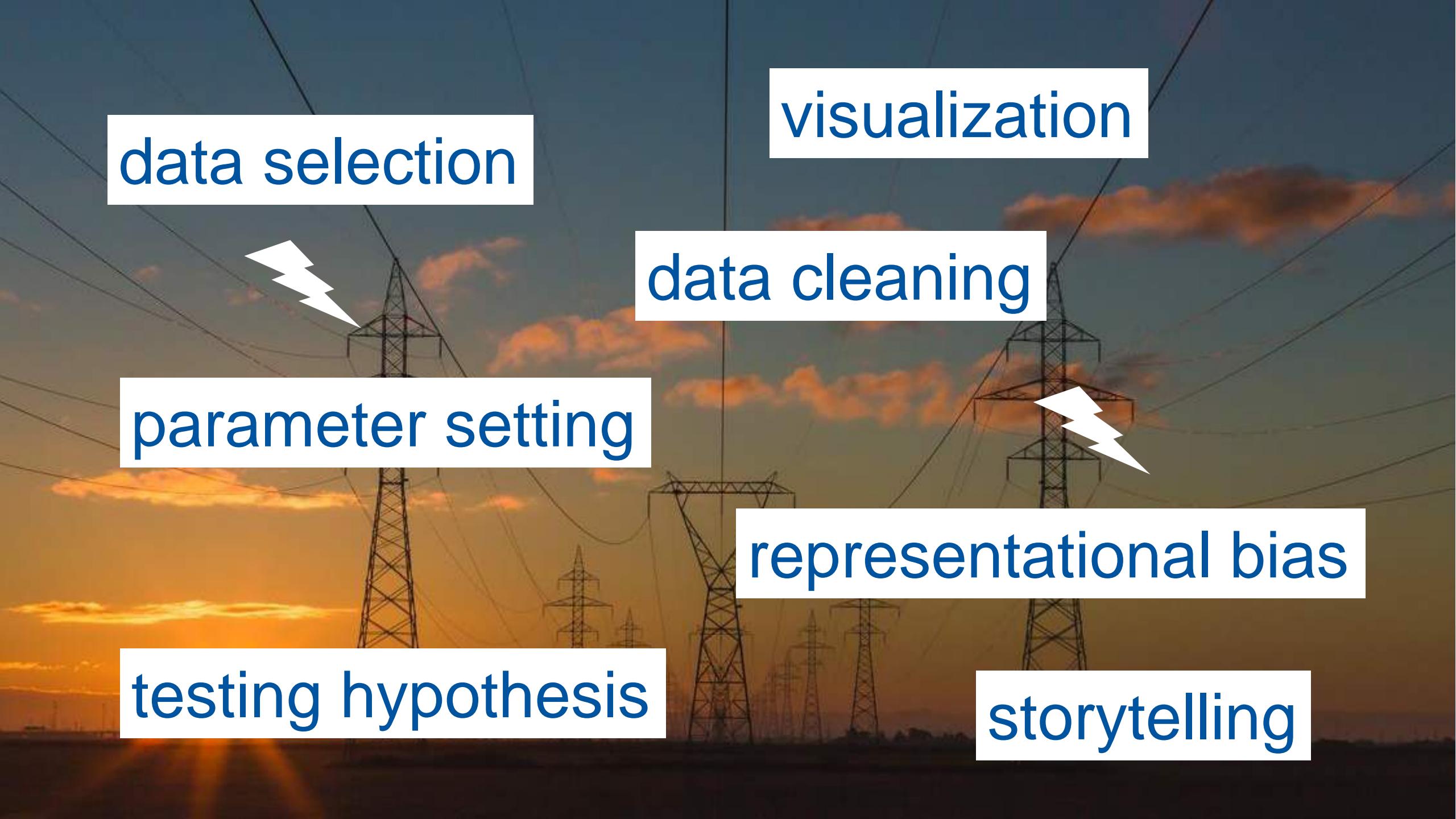
Transparency – Data Science That Provides Transparency

How to clarify answers such that they become indisputable?

- How was the prediction / decision made?
- What happened in the data pipeline?
- Do people understand the result?
- Can the result be explained?



Generated using DALL-E 3

A photograph of a power transmission system at sunset. Multiple tall, lattice-style electricity pylons are silhouetted against a sky filled with orange and yellow clouds. Power lines fan out from the towers. The foreground is dark and out of focus.

data selection

visualization

A white lightning bolt icon is positioned above the text "parameter setting".

parameter setting

A white lightning bolt icon is positioned above the text "data cleaning".

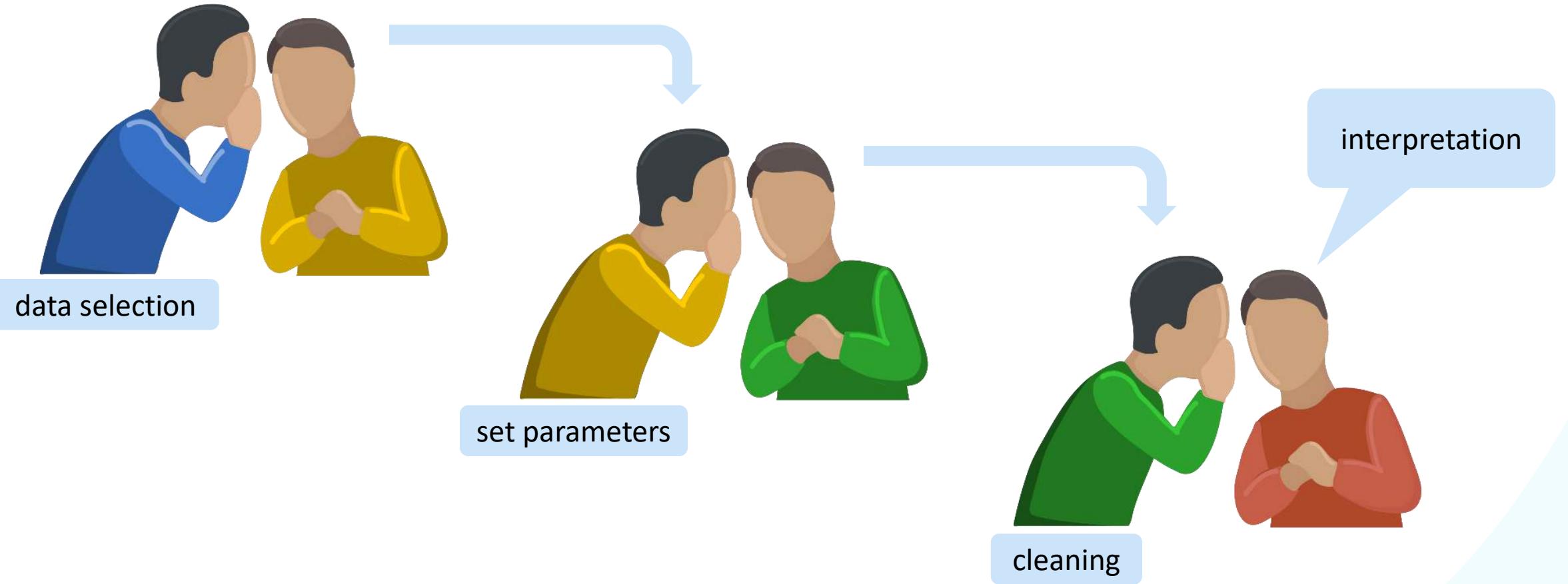
data cleaning

A white lightning bolt icon is positioned above the text "representational bias".

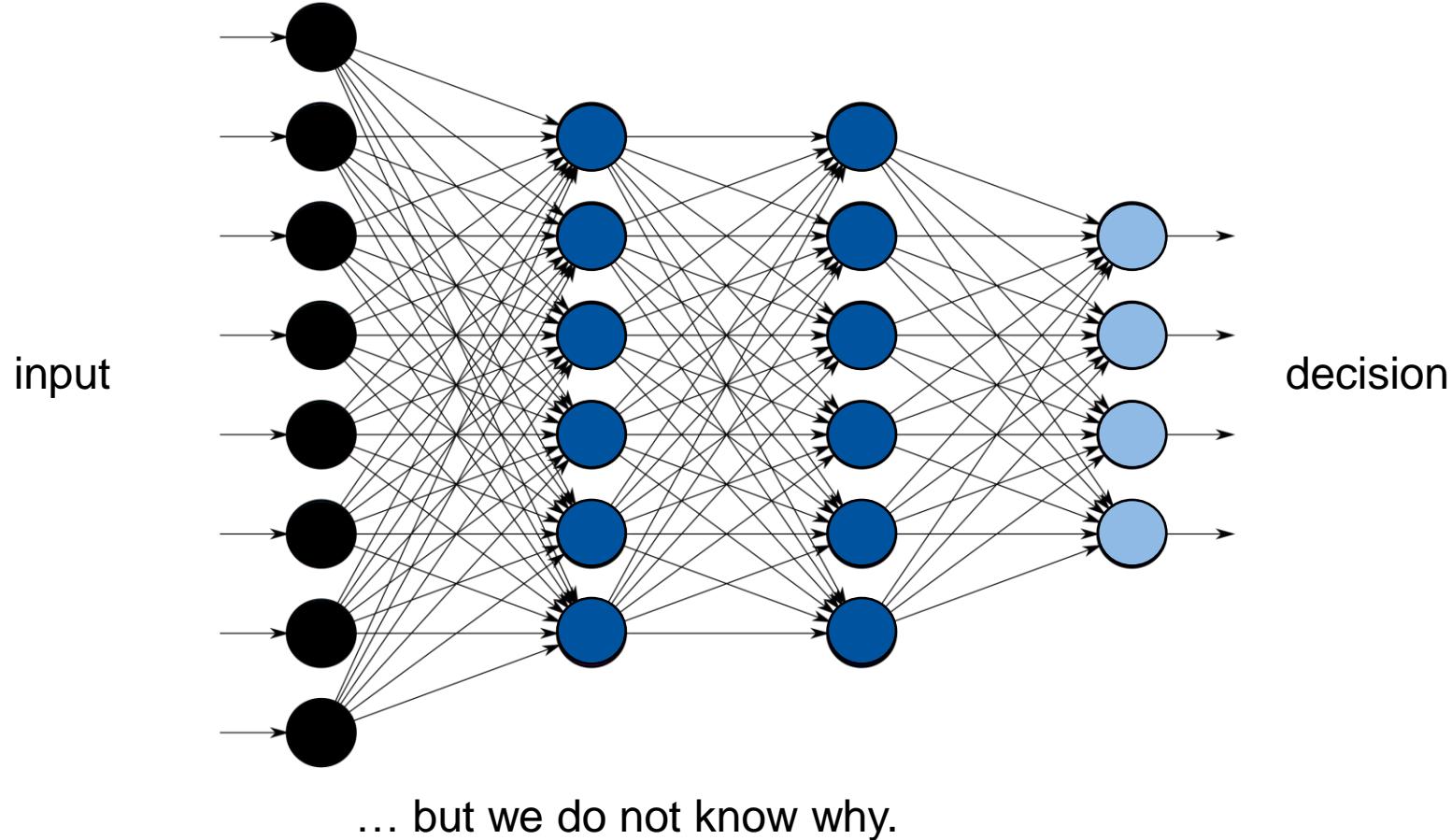
representational bias

testing hypothesis

storytelling

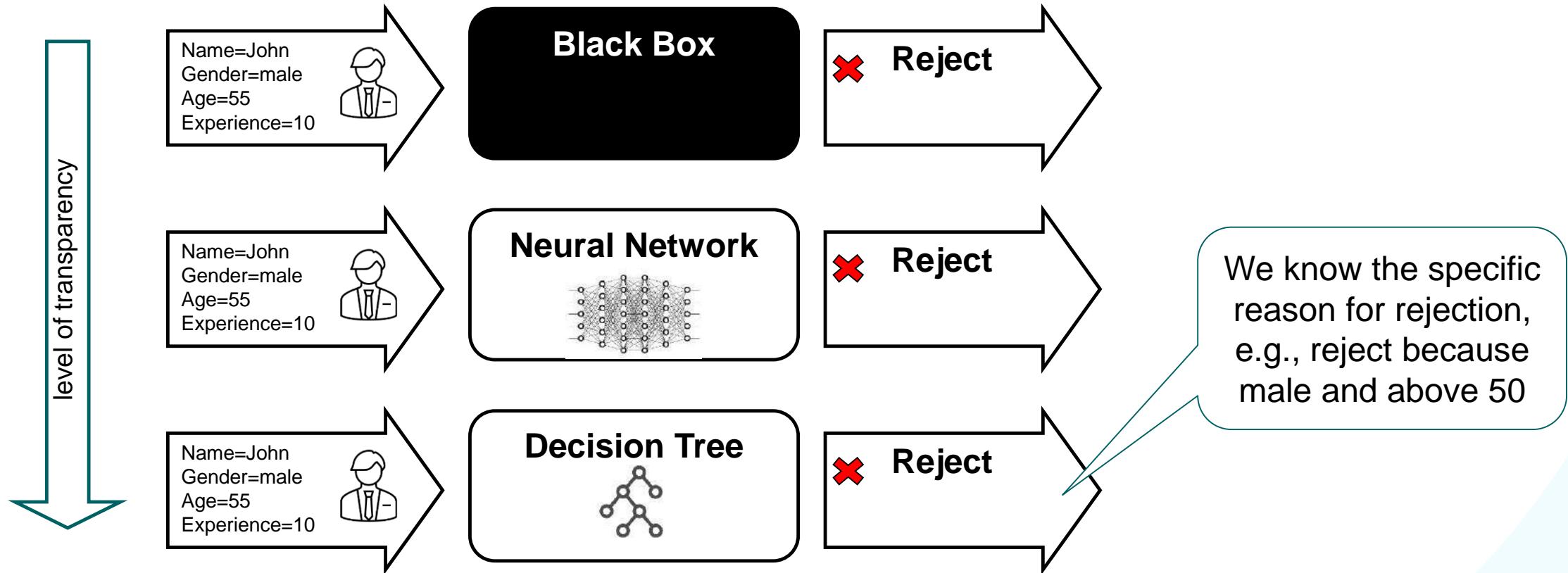


You Are Guilty, Not Selected, Not Treated, ...



Generated using DALL-E 3

Transparency



Summary

- **Fairness**
 - Ensure data-driven decision-making does not amplify existing biases or discrimination.
- **Accuracy**
 - Aim for data-driven models that make correct predictions.
- **Confidentiality**
 - Protect personal and private information throughout the data lifecycle.
- **Transparency**
 - Provide clear and understandable explanations

Part II: Confidentiality

Responsible Data Science (Confidentiality)

1. Confidentiality Risks
2. Using Encryption to Ensure Confidentiality
3. Anonymization Operations
4. K-Anonymity
5. L-Diversity and T-Closeness



4 Types of Features

Name	Age	Gender	ZIP-code	Job	Disease
Smith	27	Male	47577	Engineer	Hepatitis
Johnson	32	Male	47602	Dancer	Hepatitis
Williams	19	Female	47578	Writer	Hepatitis
Brown	55	Male	47905	Engineer	HIV
Jones	31	Male	47609	Dancer	HIV
Garcia	38	Female	47606	Lawyer	HIV
Davis	23	Female	47505	Lawyer	Heart
Martinez	47	Female	47973	Writer	Heart
Taylor	60	Female	47907	Engineer	Heart
Anderson	29	Male	47505	Dancer	Heart

4 Types of Features – Explicit Identifier

An **explicit identifier** is a set of features containing information that explicitly identifies the instance owner

Name	Age	Gender	ZIP-code	Job	Disease
Smith	27	Male	47677	Engineer	Hepatitis
Johnson	42	Male	47502	Dancer	Hepatitis
Williams	19	Female	47678	Writer	Hepatitis
Brown	55	Male	47905	Engineer	HIV
Jones	31	Male	47909	Dancer	HIV
Garcia	38	Female	47906	Lawyer	HIV
Davis	23	Female	47605	Lawyer	Heart
Martinez	47	Female	47673	Writer	Heart
Taylor	60	Female	47507	Engineer	Heart
Anderson	29	Male	47505	Dancer	Heart

4 Types of Features – Quasi-Identifiers

A **quasi-identifier** is a set of features containing information that potentially identifies the instance owner

Name	Age	Gender	ZIP-code	Job	Disease
Smith	27	Male	47677	Engineer	Hepatitis
Johnson	42	Male	47502	Dancer	Hepatitis
Williams	19	Female	47678	Writer	Hepatitis
Brown	55	Male	47905	Engineer	HIV
Jones	31	Male	47909	Dancer	HIV
Garcia	38	Female	47906	Lawyer	HIV
Davis	23	Female	47605	Lawyer	Heart
Martinez	47	Female	47673	Writer	Heart
Taylor	60	Female	47507	Engineer	Heart
Anderson	29	Male	47505	Dancer	Heart

4 Types of Features – Sensitive Features

Sensitive features are sensitive person-specific information about the instance owner

Name	Age	Gender	ZIP-code	Job	Disease
Smith	27	Male	47677	Engineer	Hepatitis
Johnson	42	Male	47502	Dancer	Hepatitis
Williams	19	Female	47678	Writer	Hepatitis
Brown	55	Male	47905	Engineer	HIV
Jones	31	Male	47909	Dancer	HIV
Garcia	38	Female	47906	Lawyer	HIV
Davis	23	Female	47605	Lawyer	Heart
Martinez	47	Female	47673	Writer	Heart
Taylor	60	Female	47507	Engineer	Heart
Anderson	29	Male	47505	Dancer	Heart

4 Types of Features

**explicit
identifier**

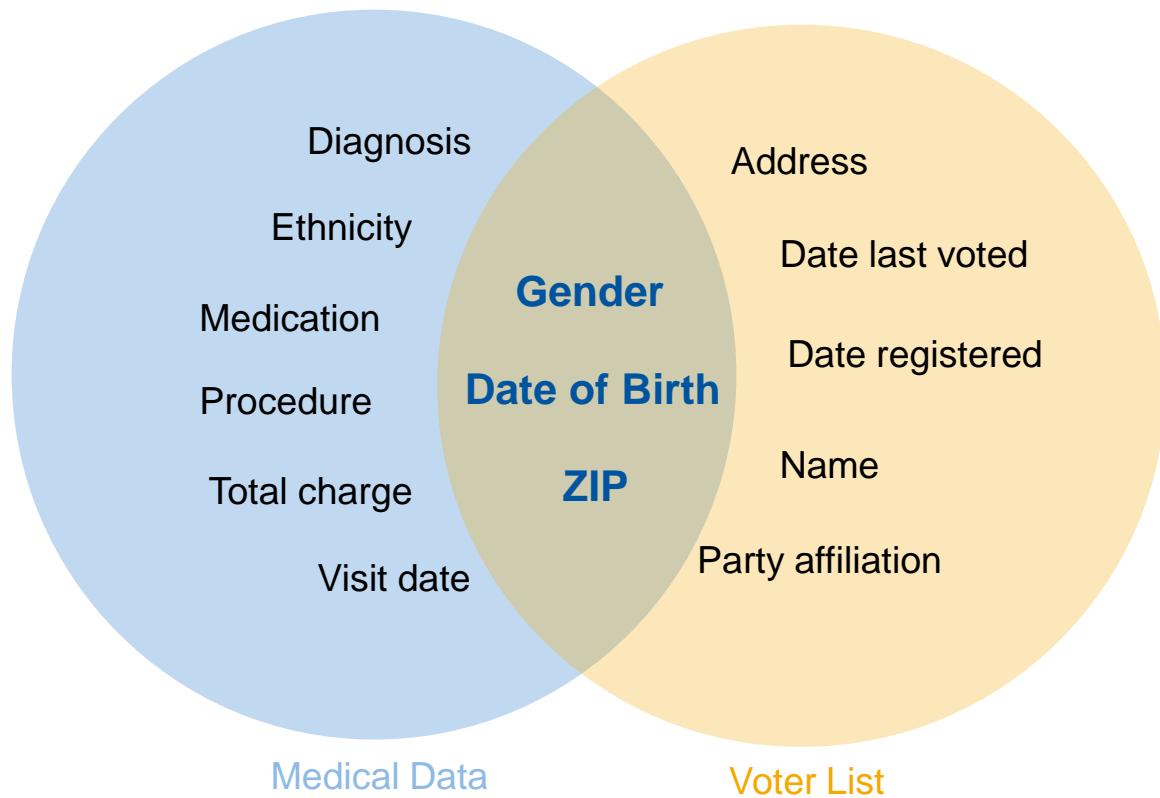
quasi-identifiers

**sensitive
feature**

Name	Age	Gender	ZIP-code	Job	Disease
Smith	27	Male	47577	Engineer	Hepatitis
Johnson	32	Male	47602	Dancer	Hepatitis
Williams	19	Female	47578	Writer	Hepatitis
Brown	55	Male	47905	Engineer	HIV
Jones	31	Male	47609	Dancer	HIV
Garcia	38	Female	47606	Lawyer	HIV
Davis	23	Female	47505	Lawyer	Heart
Martinez	47	Female	47973	Writer	Heart
Taylor	60	Female	47907	Engineer	Heart
Anderson	29	Male	47505	Dancer	Heart

other features (left out)

Problem – Example



87% of the U.S. population had reported characteristics that made them unique based on only such quasi-identifiers

Risks Related to Confidentiality and Privacy

Name	Age	Gender	ZIP-code	Job	Disease
[REDACTED]	27	Male	47577	Engineer	Hepatitis
	32	Male	47602	Dancer	Hepatitis
	19	Female	47578	Writer	Hepatitis
	55	Male	47905	Engineer	HIV
	31	Male	47609	Dancer	HIV
	38	Female	47606	Lawyer	HIV
	23	Female	47505	Lawyer	Heart
	47	Female	47973	Writer	Heart
	60	Female	47907	Engineer	Heart
	29	Male	47505	Dancer	Heart



I know my 38 year old female employee is in the data set, what disease does she have?

I know my friend is in the data set, and she started in May 2023, what is her salary?

Responsible Data Science (Confidentiality)

1. Confidentiality Risks
2. **Using Encryption to Ensure Confidentiality**
3. Anonymization Operations
4. K-Anonymity
5. L-Diversity and T-Closeness



Cryptosystem

- **Cryptosystem:** Can be used to ensure **confidentiality** when **storing** or **exchanging** sensitive data
- There is a **wide variety** of cryptosystems:
 - *Symmetric* cryptosystem
 - *Asymmetric* cryptosystem
 - *Deterministic* cryptosystem
 - *Probabilistic* cryptosystem
 - *Homomorphic* cryptosystem
 - Etc

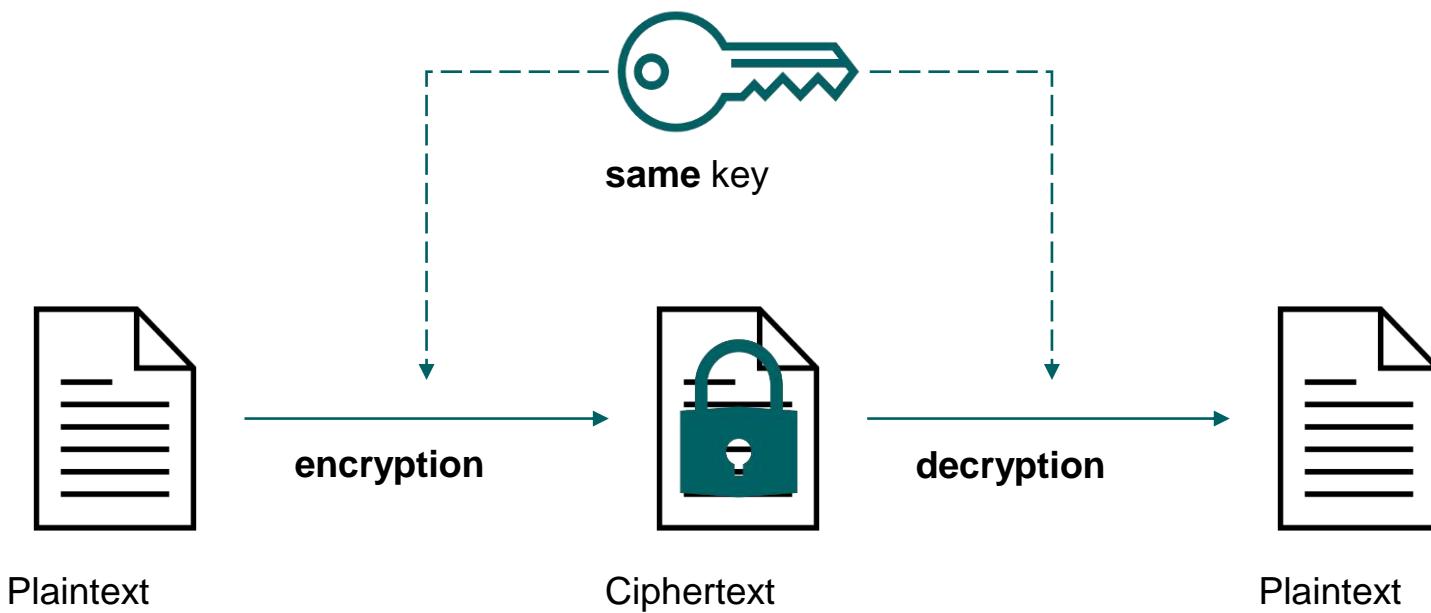
Name	Age	Gender	ZIP-code	Job	Disease
Smith	27	Male	47577	Engineer	Hepatitis
Johnson	32	Male	47602	Dancer	Hepatitis
Williams	19	Female	47578	Writer	Hepatitis
Brown	55	Male	47905	Engineer	HIV
Jones	31	Male	47609	Dancer	HIV
Garcia	38	Female	47606	Lawyer	HIV
Davis	23	Female	47505	Lawyer	Heart
Martinez	47	Female	47973	Writer	Heart
Taylor	60	Female	47907	Engineer	Heart
Anderson	29	Male	47505	Dancer	Heart



0110100101110011001000000110000100100000011001100111010101101100
 011011000010000001110000011100100110111101100110010101110011
 01110011011011101110010001000000110000101110100010000001010010
 01010111010100010010000010000001011000010110001011000110110100
 0011001010110110001000000101010101101100110100101110110110010
 1011100100111001101101001011101000111100100101100001000000110110
 00110010101100001011001000110100101101110 01100111 00100000 ...

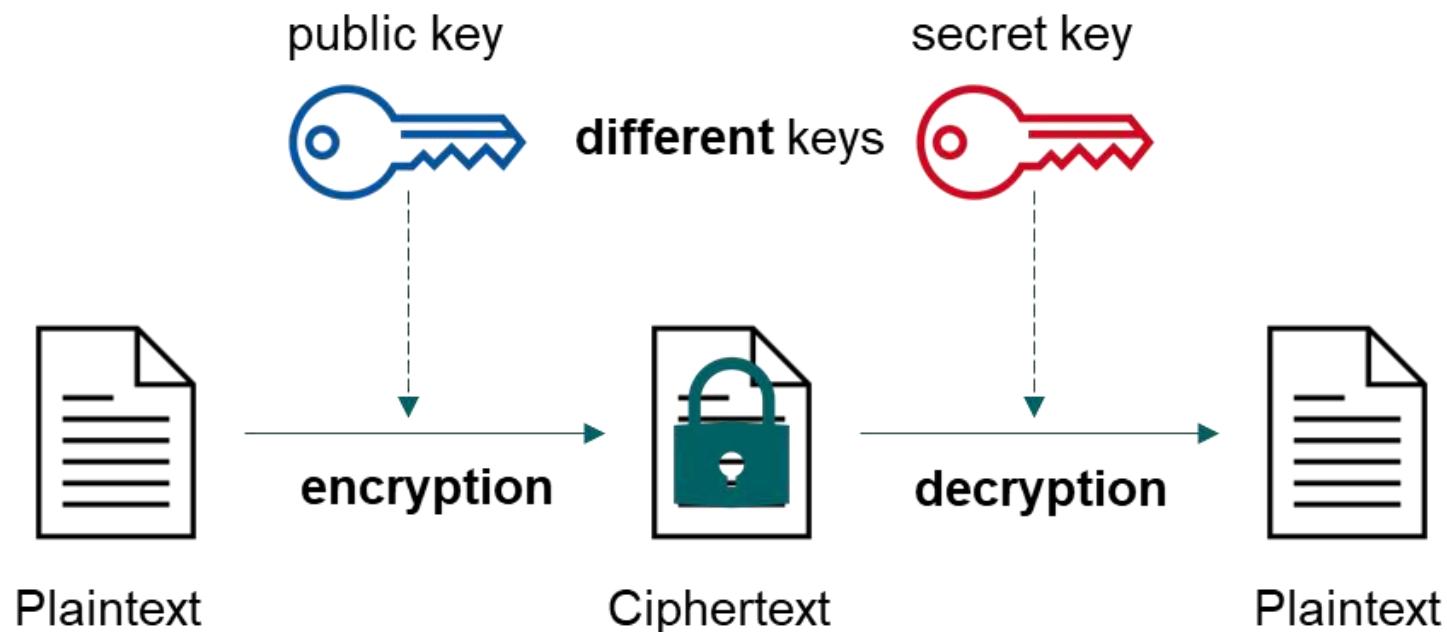
Symmetric Cryptosystem

e.g., AES



Asymmetric Cryptosystem

e.g., RSA

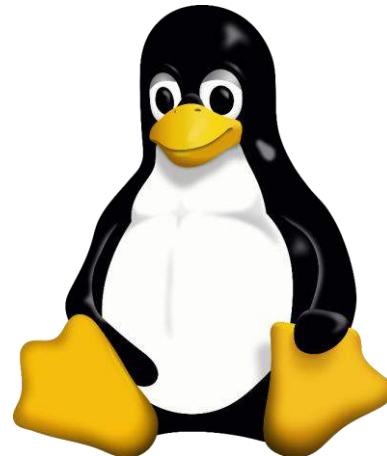


Deterministic Cryptosystem

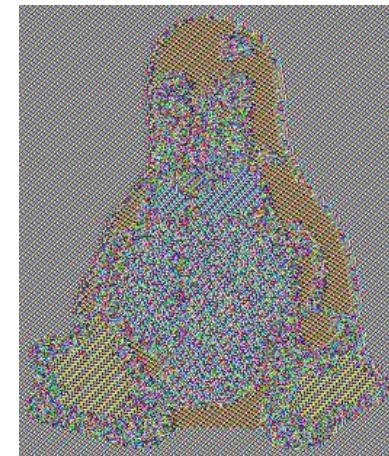
e.g., AES-ECB

A **deterministic** cryptosystem always produces the **same ciphertext** for a given plaintext and key (even over separate executions of the encryption algorithm)

- If we know that certain patterns (e.g., words, phrases, etc.) happen often, we can recognize them due to repeating patterns.
- Most common letters, e.g., “e” (13%) and “a” (8%), N-grams, or words (“the”, “of”, “and”, “to”, etc.)



Original image

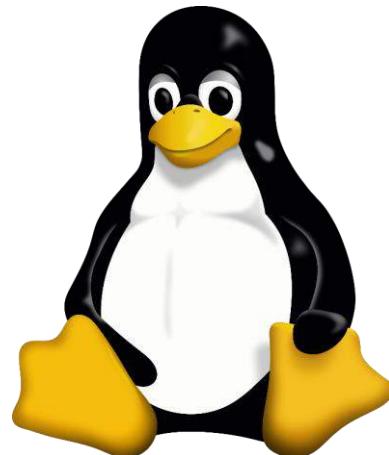


AES-ECB

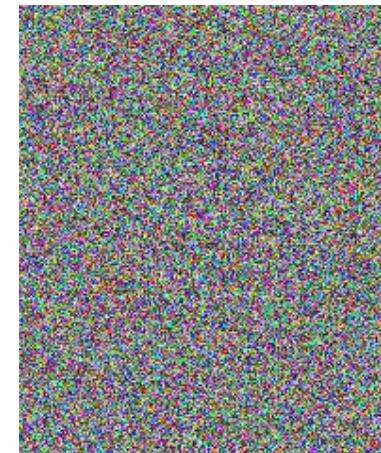
Probabilistic Cryptosystem

e.g., AES-CTR

A **probabilistic** cryptosystem as opposed to deterministic cryptosystem uses **randomness** in an encryption algorithm: when **encrypting the same plaintext** several times it produces **different ciphertexts**.



Original image

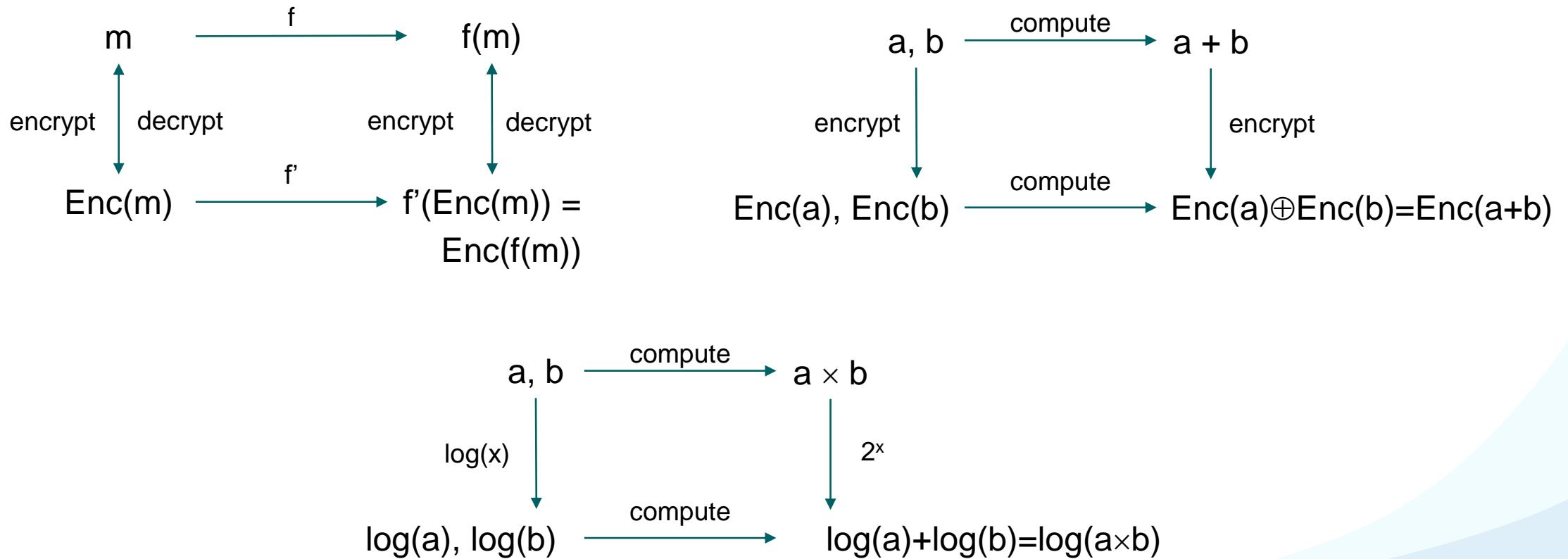


AES-CTR

Homomorphic Cryptosystem

e.g., Pallier

Homomorphic encryption allows computation on ciphertexts without decryption



Cryptosystem

- **Cryptosystem:** Protects sensitive data from unauthorized access when stored or transmitted
- Types of cryptosystems:
 - *Symmetric*
 - Keys are shared between parties
 - *Asymmetric*
 - Public and private keys
 - *Deterministic*
 - Always produces the same ciphertext
 - *Probabilistic*
 - Produces different ciphertexts (randomness)
 - *Homomorphic*
 - Computes on ciphertexts without decryption
 - ...



Responsible Data Science (Confidentiality)

1. Confidentiality Risks
2. Using Encryption to Ensure Confidentiality
3. **Anonymization Operations**
4. K-Anonymity
5. L-Diversity and T-Closeness



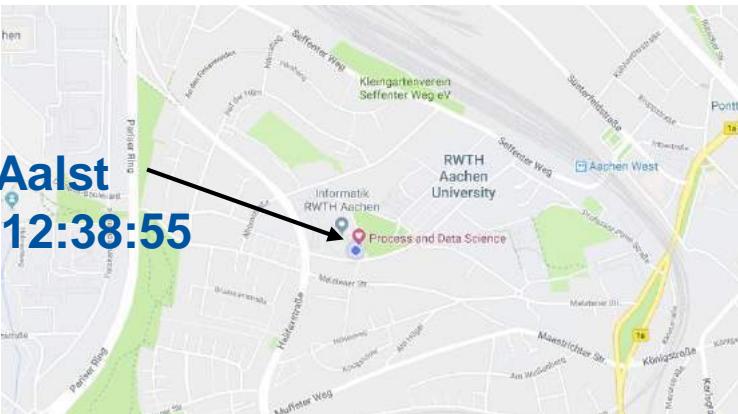
Anonymization Operations

- Provide privacy requirements:
Modify the data by applying a sequence of anonymization operations
- Anonymization operations:
 - Generalization
 - Suppression
 - Data swapping
 - Adding noise
 - Anatomization

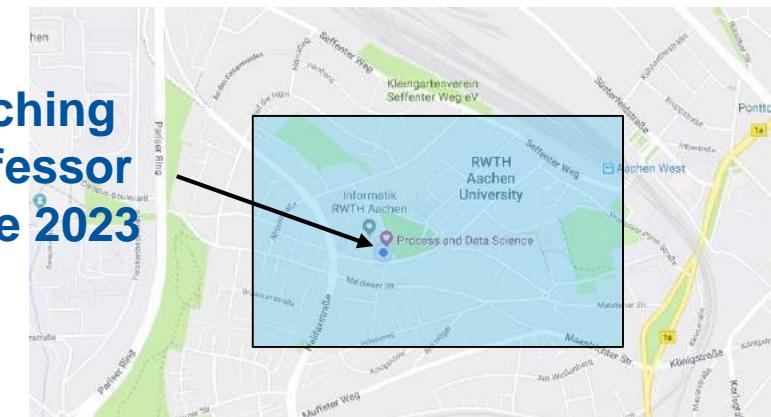
Generalization

- Reduce the granularity of representation to increase the privacy
- Can cause some loss of effectiveness of data management or mining algorithms

Lecture
Wil van der Aalst
23-6-2023T012:38:55

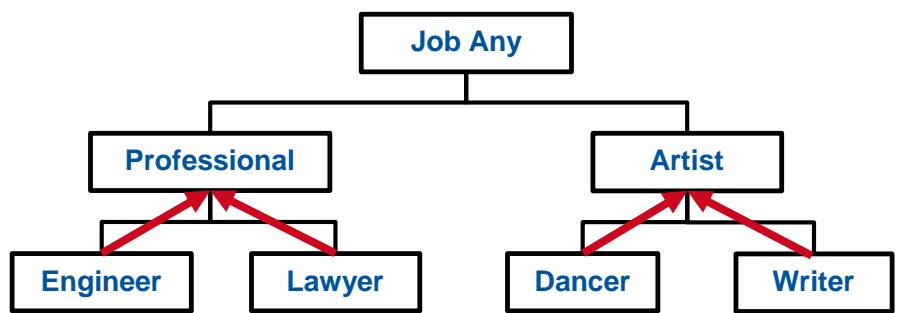


Teaching
Professor
June 2023



Generalization

Example: values of the job feature are generalized to a higher level of abstraction



A curved arrow points from the original 'Job' column to the generalized 'Job' column, indicating the mapping of specific job titles to broader categories.

Name	Gender	Job	Name	Gender	Job
Smith	Male	Engineer	Smith	Male	Professional
Johnson	Male	Dancer	Johnson	Male	Artist
Williams	Female	Writer	Williams	Female	Artist
Brown	Male	Engineer	Brown	Male	Professional
Jones	Male	Dancer	Jones	Male	Artist
Garcia	Female	Lawyer	Garcia	Female	Professional
Davis	Female	Lawyer	Davis	Female	Professional
Martinez	Female	Writer	Martinez	Female	Artist
Taylor	Female	Engineer	Taylor	Female	Professional
Anderson	Male	Dancer	Anderson	Male	Artist

Suppression

Replace some [values with a placeholder value](#), indicating that the replaced values are not disclosed

Name	Gender	Job	Disease
Smith	Male	Engineer	Hepatitis
Johnson	Male	Dancer	Hepatitis
Williams	Female	Writer	Hepatitis
Brown	Male	Engineer	HIV
Jones	Male	Dancer	HIV
Garcia	Female	Lawyer	HIV
Davis	Female	Lawyer	Heart
Martinez	Female	Writer	Heart
Taylor	Female	Engineer	Heart
Anderson	Male	Dancer	Heart



Name	Gender	Job	Disease
*	*	*	*
Johnson	Male	Dancer	Hepatitis
*	*	*	*
Brown	Male	Engineer	HIV
Jones	Male	Dancer	HIV
Garcia	Female	Lawyer	HIV
Davis	Female	Lawyer	Heart
Martinez	Female	Writer	Heart
Taylor	Female	Engineer	Heart
Anderson	Male	Dancer	Heart

Record suppression – refers to suppressing an entire instance (i.e., row)

Suppression

Replace some **values with a placeholder value**, indicating that the replaced values are not disclosed

Name	Gender	Job	Disease
Smith	Male	Engineer	Hepatitis
Johnson	Male	Dancer	Hepatitis
Williams	Female	Writer	Hepatitis
Brown	Male	Engineer	HIV
Jones	Male	Dancer	HIV
Garcia	Female	Lawyer	HIV
Davis	Female	Lawyer	Heart
Martinez	Female	Writer	Heart
Taylor	Female	Engineer	Heart
Anderson	Male	Dancer	Heart



Name	Gender	Job	Disease
Smith	*	Engineer	Hepatitis
Johnson	*	Dancer	Hepatitis
Williams	*	Writer	Hepatitis
Brown	*	Engineer	HIV
Jones	*	Dancer	HIV
Garcia	*	Lawyer	HIV
Davis	*	Lawyer	Heart
Martinez	*	Writer	Heart
Taylor	*	Engineer	Heart
Anderson	*	Dancer	Heart

Column suppression – refers to suppressing a feature (i.e., column)

Many more possibilities!

Data Swapping

Anonymize the data by **exchanging values** of **sensitive features** among individuals

Name	Gender	Job	Disease
Smith	Male	Engineer	Hepatitis
Johnson	Male	Dancer	Hepatitis
Williams	Female	Writer	Hepatitis
Brown	Male	Engineer	HIV
Jones	Male	Dancer	HIV
Garcia	Female	Lawyer	HIV
Davis	Female	Lawyer	Heart
Martinez	Female	Writer	Heart
Taylor	Female	Engineer	Heart
Anderson	Male	Dancer	Heart

Name	Gender	Job	Disease
Smith	Male	Engineer	Hepatitis
Johnson	Male	Dancer	Heart
Williams	Female	Writer	Heart
Brown	Male	Engineer	HIV
Jones	Male	Dancer	HIV
Garcia	Female	Lawyer	HIV
Davis	Female	Lawyer	Heart
Martinez	Female	Writer	Hepatitis
Taylor	Female	Engineer	Hepatitis
Anderson	Male	Dancer	Heart

Frequencies remain unchanged

Additive Noise

Replace the original sensitive value s with $s+r$ where r is a random variable drawn from some distribution

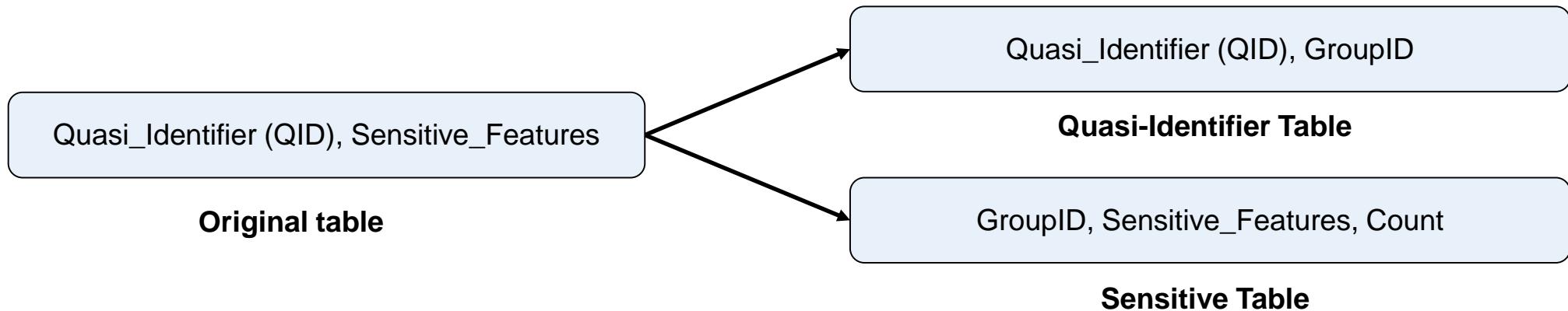
Name	Gender	Job	Salary
Smith	Male	Engineer	10000
Johnson	Male	Dancer	5000
Williams	Female	Writer	9500
Brown	Male	Engineer	12000
Jones	Male	Dancer	9000
Garcia	Female	Lawyer	12500
Davis	Female	Lawyer	6800
Martinez	Female	Writer	11000
Taylor	Female	Engineer	8000
Anderson	Male	Dancer	8500

$$\text{Salary} = \text{Salary} + \text{Gaussian}(10000, 4000)$$

Name	Gender	Job	Salary
Smith	Male	Engineer	22112
Johnson	Male	Dancer	10177
Williams	Female	Writer	13708
Brown	Male	Engineer	23365
Jones	Male	Dancer	13519
Garcia	Female	Lawyer	23954
Davis	Female	Lawyer	13451
Martinez	Female	Writer	24410
Taylor	Female	Engineer	14898
Anderson	Male	Dancer	21154

Anatomization: Decouple instead of delete, change, or swap

- Does not modify the quasi-identifier or the sensitive feature
- Instead, de-associates the relationship between the two



Anatomization – Example

Gender	Job	Disease
Male	Engineer	Hepatitis
Male	Dancer	Hepatitis
Female	Writer	Hepatitis
Male	Engineer	HIV
Male	Dancer	HIV
Female	Lawyer	HIV
Female	Lawyer	Heart
Female	Writer	Heart
Female	Engineer	Heart
Male	Dancer	Heart

Original table

Apply generalization to decrease the number of equivalence classes

generalization

Gender	Job	Disease
Male	Professional	Hepatitis
Male	Artist	Hepatitis
Female	Artist	Hepatitis
Male	Professional	HIV
Male	Artist	HIV
Female	Professional	HIV
Female	Professional	Heart
Female	Artist	Heart
Female	Professional	Heart
Male	Artist	Heart

Generalized data

Anatomization – Example

Gender	Job	Disease
Male	Professional	Hepatitis
Male	Artist	Hepatitis
Female	Artist	Hepatitis
Male	Professional	HIV
Male	Artist	HIV
Female	Professional	HIV
Female	Professional	Heart
Female	Artist	Heart
Female	Professional	Heart
Male	Artist	Heart

Generalized data

Apply suppression to decrease the number of equivalence classes even further

suppression



Gender	Job	Disease
*	Professional	Hepatitis
*	Artist	Hepatitis
*	Artist	Hepatitis
*	Professional	HIV
*	Artist	HIV
*	Professional	HIV
*	Professional	Heart
*	Artist	Heart
*	Professional	Heart
*	Artist	Heart

Generalized suppressed data

These suppression/generalization steps may be driven by k-anonymity or l-diversity (see later)

Anatomization – Example

Gender	Job	Disease
*	Professional	Hepatitis
*	Artist	Hepatitis
*	Artist	Hepatitis
*	Professional	HIV
*	Artist	HIV
*	Professional	HIV
*	Professional	Heart
*	Artist	Heart
*	Professional	Heart
*	Artist	Heart

anatomization

Group 1 are all the professionals Only one professional with hepatitis

Gender	Job	GroupID
*	Professional	1
*	Artist	2
*	Artist	2
*	Professional	1
*	Artist	2
*	Professional	1
*	Professional	1
*	Artist	2
*	Professional	1
*	Artist	2

GroupID	Disease	Count
1	Hepatitis	1
1	HIV	2
1	Heart	2
2	Hepatitis	2
2	HIV	1
2	Heart	2

Sensitive table

Quasi identifier table

Anatomization – Example

Gender	Job	GroupID
*	Professional	1
*	Artist	2
*	Artist	2
*	Professional	1
*	Artist	2
*	Professional	1
*	Professional	1
*	Artist	2
*	Professional	1
*	Artist	2

reinsert the
original quasi
identifiers

Output

Gender	Job	GroupID
Male	Engineer	1
Male	Dancer	2
Female	Writer	2
Male	Engineer	1
Male	Dancer	2
Female	Lawyer	1
Female	Lawyer	1
Female	Writer	2
Female	Engineer	1
Male	Dancer	2

Quasi identifier table

GroupID	Disease	Count
1	Hepatitis	1
1	HIV	2
1	Heart	2
2	Hepatitis	2
2	HIV	1
2	Heart	2

Sensitive table

Now only group-
level probabilities

Anonymization Operations

Anonymization operations:

- Generalization: Replacing specific values with a more general category.
- Suppression: Replacing some values with a placeholder.
- Data swapping: Exchanging data points or attributes between different entities.
- Adding noise: Injecting random noise.
- Anatomization: Decoupling data by grouping.



Responsible Data Science (Confidentiality)

1. Confidentiality Risks
2. Using Encryption to Ensure Confidentiality
3. Anonymization Operations
4. **K-Anonymity**
5. L-Diversity and T-Closeness



Equivalence Class

- **Equivalence class** of an anonymized data table:
a set of instances with the same values for the **quasi-identifiers**
- k-anonymity requires that each equivalence class contains **at least k** instances

Gender	Job	Disease
Male	Professional	Hepatitis
Male	Artist	Hepatitis
Female	Artist	Hepatitis
Male	Professional	HIV
Male	Artist	HIV
Female	Professional	HIV
Female	Professional	Heart
Female	Artist	Heart
Female	Professional	Heart
Male	Artist	Heart

1-Anonymity

Each instance is a separate equivalence class

quasi-identifiers			sensitive feature
Name	Age	ZIP-code	Disease
*	27	47577	Hepatitis
*	32	47602	Hepatitis
*	19	47578	Hepatitis
*	55	47905	HIV
*	31	47609	HIV
*	38	47606	HIV
*	23	47505	Heart
*	47	47973	Heart
*	60	47907	Heart
*	29	47505	Heart

1-Anonymity to 3-Anonymity

1-anonymity

quasi-identifiers			sensitive feature
Name	Age	ZIP-code	Disease
*	27	47577	Hepatitis
*	32	47602	Hepatitis
*	19	47578	Hepatitis
*	55	47905	HIV
*	31	47609	HIV
*	38	47606	HIV
*	23	47505	Heart
*	47	47973	Heart
*	60	47907	Heart
*	29	47505	Heart

sort by Age

quasi-identifiers			sensitive feature
Name	Age	ZIP-code	Disease
*	19	47578	Hepatitis
*	23	47505	Heart
*	27	47577	Hepatitis
*	29	47505	Heart
*	31	47609	HIV
*	32	47602	Hepatitis
*	38	47606	HIV
*	47	47973	Heart
*	55	47905	HIV
*	60	47907	Heart

1-Anonymity to 3-Anonymity

sorted by Age

quasi-identifiers			sensitive feature
Name	Age	ZIP-code	Disease
*	19	47578	Hepatitis
*	23	47505	Heart
*	27	47577	Hepatitis
*	29	47505	Heart
*	31	47609	HIV
*	32	47602	Hepatitis
*	38	47606	HIV
*	47	47973	Heart
*	55	47905	HIV
*	60	47907	Heart



generalization

quasi-identifiers			sensitive feature
Name	Age	ZIP-code	Disease
*	<30	475**	Hepatitis
*	<30	475**	Heart
*	<30	475**	Hepatitis
*	<30	475**	Heart
*	3*	476**	HIV
*	3*	476**	Hepatitis
*	3*	476**	HIV
*	≥40	479**	Heart
*	≥40	479**	HIV
*	≥40	479**	Heart

1-Anonymity vs 3-Anonymity

1-anonymity

quasi-identifiers			sensitive feature
Name	Age	ZIP-code	Disease
*	27	47577	Hepatitis
*	32	47602	Hepatitis
*	19	47578	Hepatitis
*	55	47905	HIV
*	31	47609	HIV
*	38	47606	HIV
*	23	47505	Heart
*	47	47973	Heart
*	60	47907	Heart
*	29	47505	Heart

3-anonymity (**each** equivalence class has **at least 3 instances**)

quasi-identifiers			sensitive feature
Name	Age	ZIP-code	Disease
*	<30	475**	Hepatitis
*	<30	475**	Heart
*	<30	475**	Hepatitis
*	<30	475**	Heart
*	3*	476**	HIV
*	3*	476**	Hepatitis
*	3*	476**	HIV
*	≥40	479**	Heart
*	≥40	479**	HIV
*	≥40	479**	Heart

The Three Equivalence Classes

quasi-identifiers			sensitive feature	
Name	Age	ZIP-code	Disease	
*	<30	475**	Hepatitis	
*	<30	475**	Heart	Equivalence Class 1
*	<30	475**	Hepatitis	
*	<30	475**	Heart	
*	3*	476**	HIV	
*	3*	476**	Hepatitis	Equivalence Class 2
*	3*	476**	HIV	
*	≥40	479**	Heart	
*	≥40	479**	HIV	Equivalence Class 3
*	≥40	479**	Heart	

Confidence of Identifying Instances

Consider

- a table that satisfies k-anonymity for some value k
 - an adversarial who knows the quasi-identifier values of one individual
- The adversarial cannot identify the instance corresponding to that individual with confidence greater than $1/k$

Example: confidence in guessing in the given example is not greater than $1/3$

quasi-identifiers			sensitive feature
Name	Age	ZIP-code	Disease
*	<30	475**	Hepatitis
*	<30	475**	Heart
*	<30	475**	Hepatitis
*	<30	475**	Heart
*	3*	476**	HIV
*	3*	476**	Hepatitis
*	3*	476**	HIV
*	≥40	479**	Heart
*	≥40	479**	HIV
*	≥40	479**	Heart

Problems

- K-anonymity protects against identity disclosure
- is insufficient to prevent feature disclosure
- K-anonymity focuses on **quasi-identifiers (QID)** such that each QID tuple occurs in at least k instances
 → **Sensitive features are not considered!**

quasi-identifiers			sensitive feature
Name	Age	ZIP-code	Disease
*	<30	475**	Hepatitis
*	<30	475**	Heart
*	<30	475**	Hepatitis
*	<30	475**	Heart
*	3*	476**	HIV
*	3*	476**	Hepatitis
*	3*	476**	HIV
*	≥40	479**	Heart
*	≥40	479**	HIV
*	≥40	479**	Heart

Possible Attacks

quasi-identifiers			sensitive feature
Name	Age	ZIP-code	Disease
*	<30	475**	Hepatitis
*	<30	475**	Hepatitis
*	<30	475**	Hepatitis
*	<30	475**	Hepatitis
*	3*	476**	HIV
*	3*	476**	Hepatitis
*	3*	476**	HIV
*	≥40	479**	Heart
*	≥40	479**	HIV
*	≥40	479**	Heart

Homogeneity attack:

The sensitive feature value for all the instances of this equivalence class is the same

Background knowledge attack:

- When knowing Mr. Brown's age and zip code, one can conclude that he corresponds to an instance in this equivalence class.
- Also knowing that he has a very low risk for heart disease, one can conclude that he likely has HIV.

Responsible Data Science (Confidentiality)

1. Confidentiality Risks
2. Using Encryption to Ensure Confidentiality
3. Anonymization Operations
4. K-Anonymity
5. **L-Diversity and T-Closeness**



L-Diversity

- Addresses the issue of homogeneity attacks and background knowledge attacks, e.g., all instances in an equivalence class have the same sensitive feature
- An equivalence class has ℓ -diversity, if there are **at least ℓ “well-represented” values** for the sensitive feature
- A table has ℓ -diversity if **every equivalence class** in the table has ℓ -diversity
- Different interpretations of the term “well-represented”:
 - **Distinct ℓ -diversity**
 - **Entropy ℓ -diversity**

Distinct L-Diversity

- The simplest understanding of “well represented”: Ensure **there are at least l distinct values** for the sensitive feature **in each equivalence class**
- Distinct l-diversity does not prevent a background knowledge attack

quasi-identifiers			sensitive feature
Name	Age	ZIP-code	Disease
*	<30	475**	Hepatitis
*	<30	475**	Heart
*	<30	475**	Hepatitis
*	<30	475**	Heart
*	3*	476**	HIV
*	3*	476**	Hepatitis
*	3*	476**	HIV
*	≥40	479**	Heart
*	≥40	479**	HIV
*	≥40	479**	Heart

Distinct 2-diverse table

Problem Distinct L-Diversity

quasi-identifiers			sensitive feature
Name	Age	ZIP-code	Disease
*	<30	475**	Hepatitis
*	<30	475**	Hepatitis
...
*	<30	475**	Hepatitis
*	<30	475**	Heart
*	3*	476**	Hepatitis
*	3*	476**	HIV
*	3*	476**	HIV
...
*	3*	476**	HIV
*	≥40	479**	Heart
*	≥40	479**	HIV
*	≥40	479**	Heart

99 times Hepatitis

100 instances and having two distinct values but 99% Hepatitis

99 times HIV

100 instances and having two distinct values but 99% HIV

Entropy L-Diversity

- The entropy of an equivalence class E is defined as:

$$H(E) = - \sum_{s \in S} P(E, s) \cdot \log_2(P(E, s))$$



- S : the domain of the sensitive feature
 - $P(E, s)$: the fraction of instances in E that have the sensitive value s
 - Example: $H(E) = -\left(\frac{7}{14} \cdot \log_2\left(\frac{7}{14}\right) + \frac{3}{14} \cdot \log_2\left(\frac{3}{14}\right) + \frac{4}{14} \cdot \log_2\left(\frac{4}{14}\right)\right) = 1.49261$
- A data table has **entropy l-diversity** if for every equivalence class E:

$$H(E) \geq -\log_2\left(\frac{1}{l}\right) = \log_2(l)$$
 - This corresponds to a higher entropy than l equally distributed sensitive values
 - Higher entropy is good because it is harder to guess the actual value!**

Entropy L-Diversity – Example

quasi-identifiers			sensitive feature
Name	Age	ZIP-code	Disease
*	<30	475**	Hepatitis
*	<30	475**	Heart
*	<30	475**	Hepatitis
*	<30	475**	Heart
*	3*	476**	HIV
*	3*	476**	Hepatitis
*	3*	476**	HIV
*	≥40	479**	Heart
*	≥40	479**	HIV
*	≥40	479**	Heart

What is the maximal value of l for entropy l -diversity?

$$H(E) \geq \log_2(l)$$

$$\log_2(l) = 0.92$$

$$l = 2^{0.92} = 1.89$$

Only 1-diversity!

Entropy L-Diversity – Example

quasi-identifiers			sensitive feature
Name	Age	ZIP-code	Disease
*	<30	475**	Hepatitis
*	<30	475**	Hepatitis
...
*	<30	475**	Hepatitis
*	<30	475**	Heart
*	3*	476**	Hepatitis
*	3*	476**	HIV
*	3*	476**	HIV
...
*	3*	476**	HIV
*	≥40	479**	Heart
*	≥40	479**	HIV
*	≥40	479**	Heart

99 times

$$H(E_1) = -\left(\frac{99}{100} \cdot \log_2\left(\frac{99}{100}\right) + \frac{1}{100} \cdot \log_2\left(\frac{1}{100}\right)\right) = 0.02432$$

99 times

$$H(E_2) = -\left(\frac{1}{100} \cdot \log_2\left(\frac{1}{100}\right) + \frac{99}{100} \cdot \log_2\left(\frac{99}{100}\right)\right) = 0.02432$$

$$H(E_i) \geq \log_2(l)$$

$$\log_2(l) = 0.02432$$

$$l = 2^{0.02432} = 1.017$$

(Close to 1, so one can guess
the value with high confidence)

Entropy L-Diversity

- A table has **entropy l-diversity** if for every equivalence class E:
$$H(E) \geq \log_2(l)$$
- Example:
If $H(E) = 2.9$, then entropy 7-diversity holds, but entropy 8-diversity doesn't hold
- To have entropy l-diversity for each equivalence class, the entropy of the entire table has to be at least $\log_2(l)$
- Can be too restrictive: the entropy of the entire table may be low if a few values are very common

$$\log_2(7) = 2.807 \quad \log_2(8) = 3$$

T-Closeness

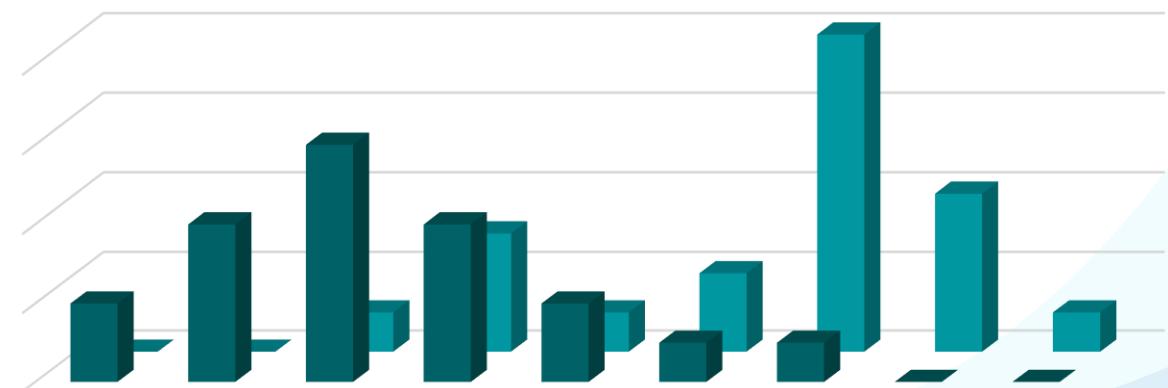
- An equivalence class has t-closeness if the **distance** between the distribution of a sensitive feature **in this class** and the distribution of it **in the whole table** is no more than a **threshold t**

$$\text{Distance}(\text{DE}, \text{DT}) \leq t$$

distribution of the sensitive
feature in the equivalence class

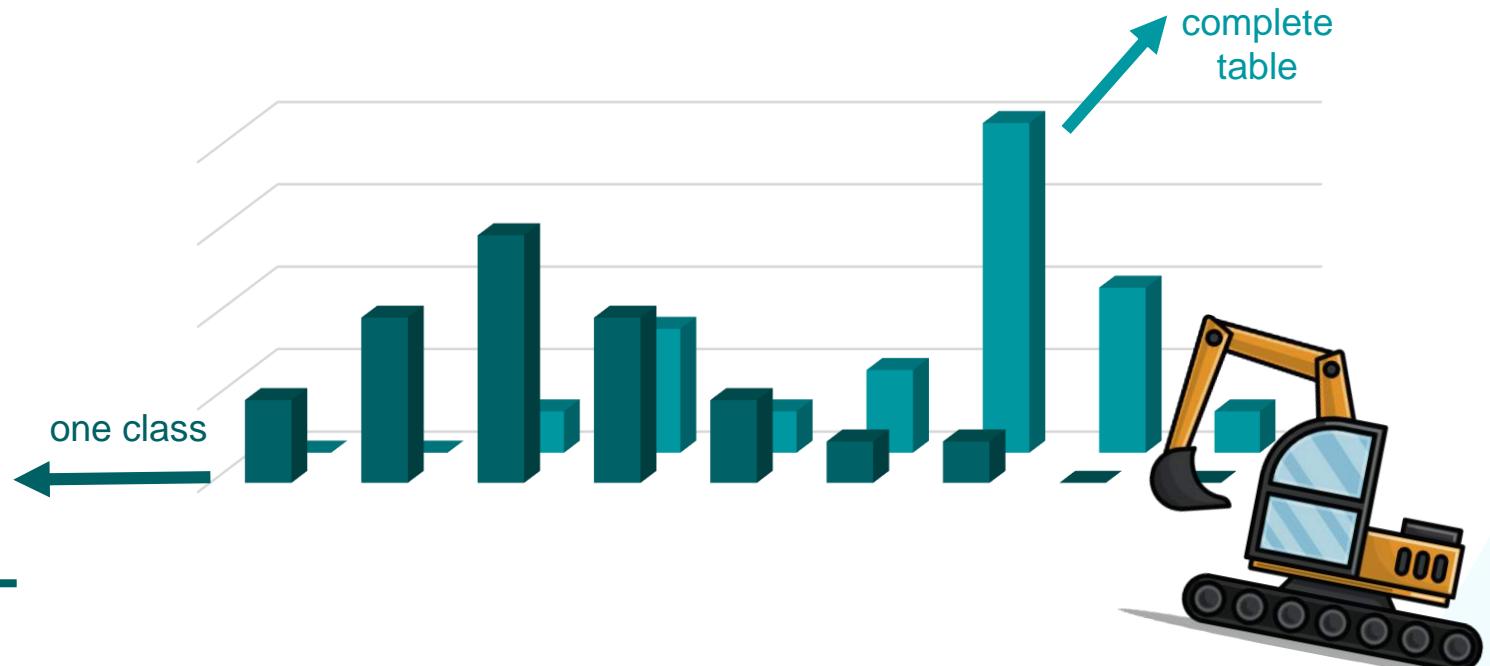
distribution of the sensitive
feature in the whole table

- A table has t-closeness if **all** equivalence classes have t-closeness
- Distance measure should reflect the semantic distance among values
→ **Earth Mover's Distance**



T-Closeness

quasi-identifiers			sensitive feature
Name	Age	ZIP-code	Disease
*	<30	475**	Hepatitis
*	<30	475**	Heart
*	<30	475**	Hepatitis
*	<30	475**	Heart
*	3*	476**	HIV
*	3*	476**	Hepatitis
*	3*	476**	HIV
*	≥40	479**	Heart
*	≥40	479**	HIV
*	≥40	479**	Heart



Confidentiality

- **Important!** If not addressed properly, people will resist data science applications
- Even after removing explicit identifiers, there may be (un)intentional information sharing:
 - Through quasi-identifiers, it may be possible to uniquely identify instances
 - Sensitive features in an equivalence class may not be diverse enough

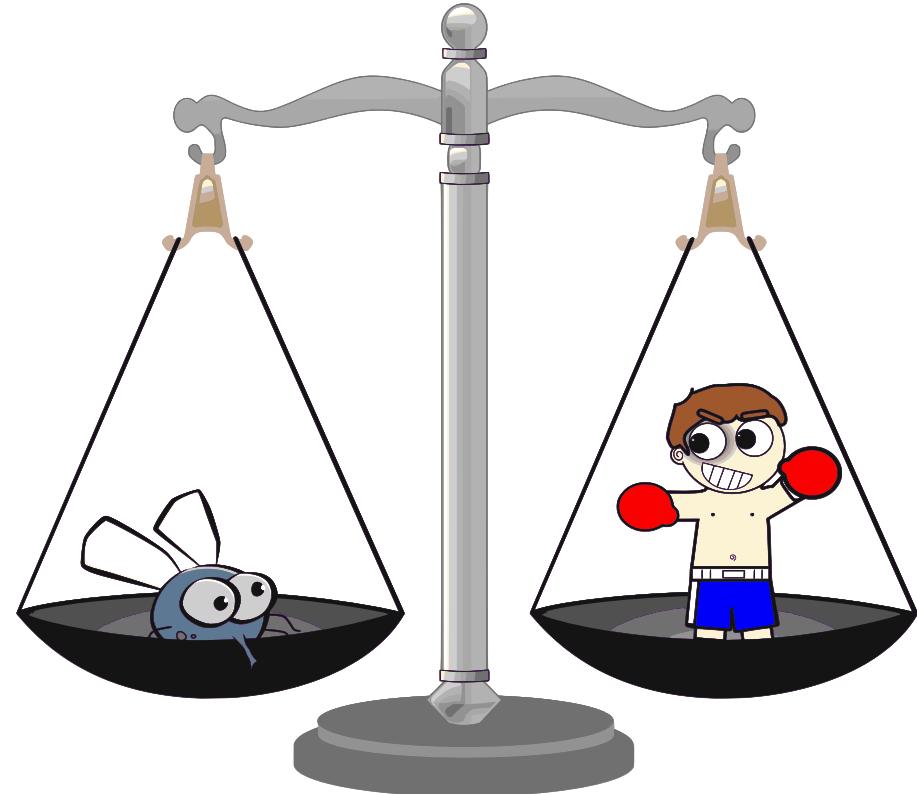
Confidentiality

- **Important!** If not addressed properly, people will resist data science applications
- Can be tackled through [encryption](#) and [anonymization](#)
- We can [measure confidentiality](#), examples:
 - [K-Anonymity](#) focusing on instances having the same quasi-identifiers
 - [L-Diversity](#) and [T-Closeness](#) focusing on the sensitive feature
- Tradeoffs between [utility](#) and [confidentiality](#)

Part III: Fairness

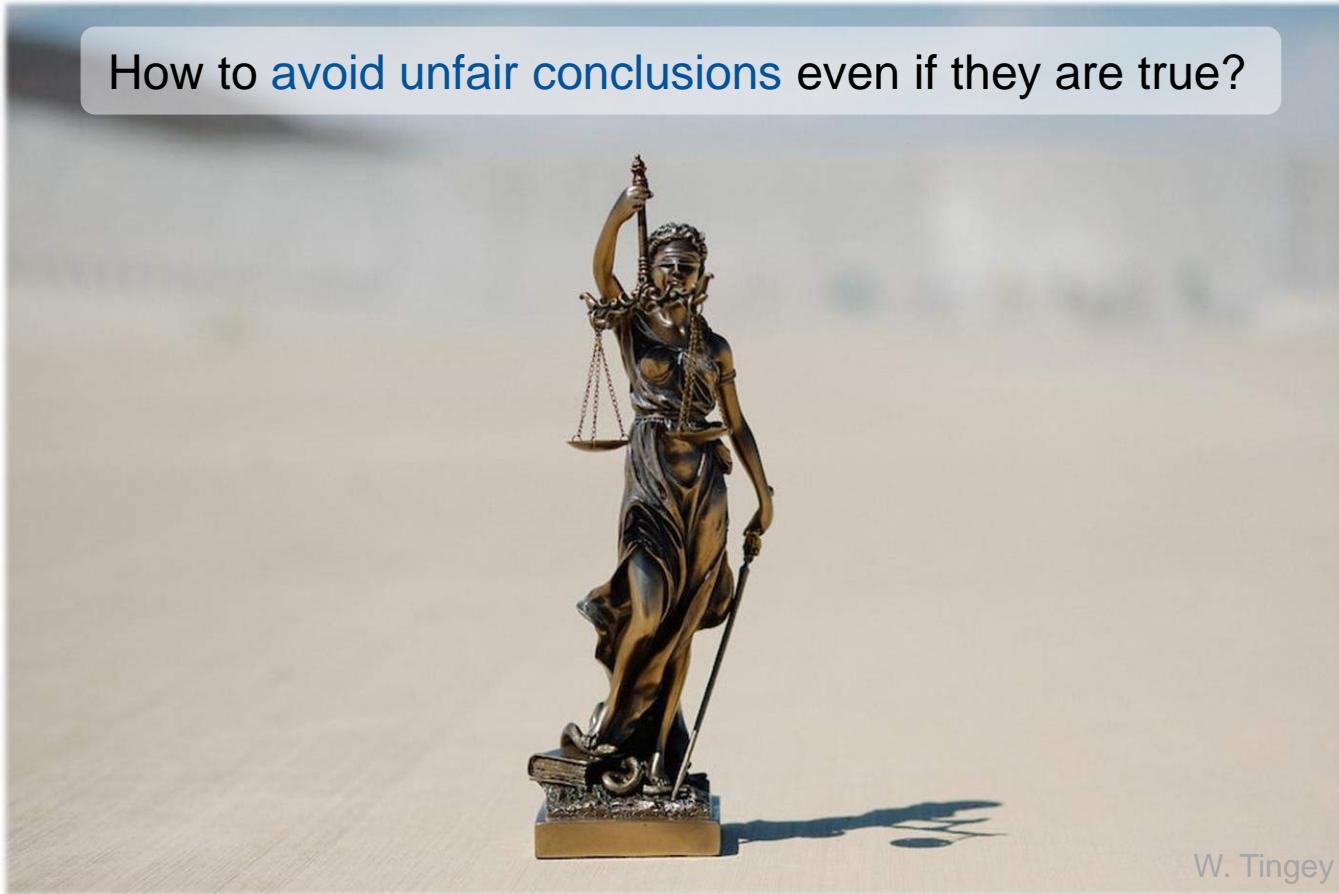
Responsible Data Science (Fairness)

1. Motivation
2. Preliminaries
3. Fairness Measures
4. Fair Decision Trees



Fairness – Data Science Without Prejudice

How to **avoid unfair conclusions** even if they are true?



It Is Hard To Define Fairness

Correct Does Not Imply Fair



Is a company that optimizes its profits by excluding minority neighborhoods from its services fair?

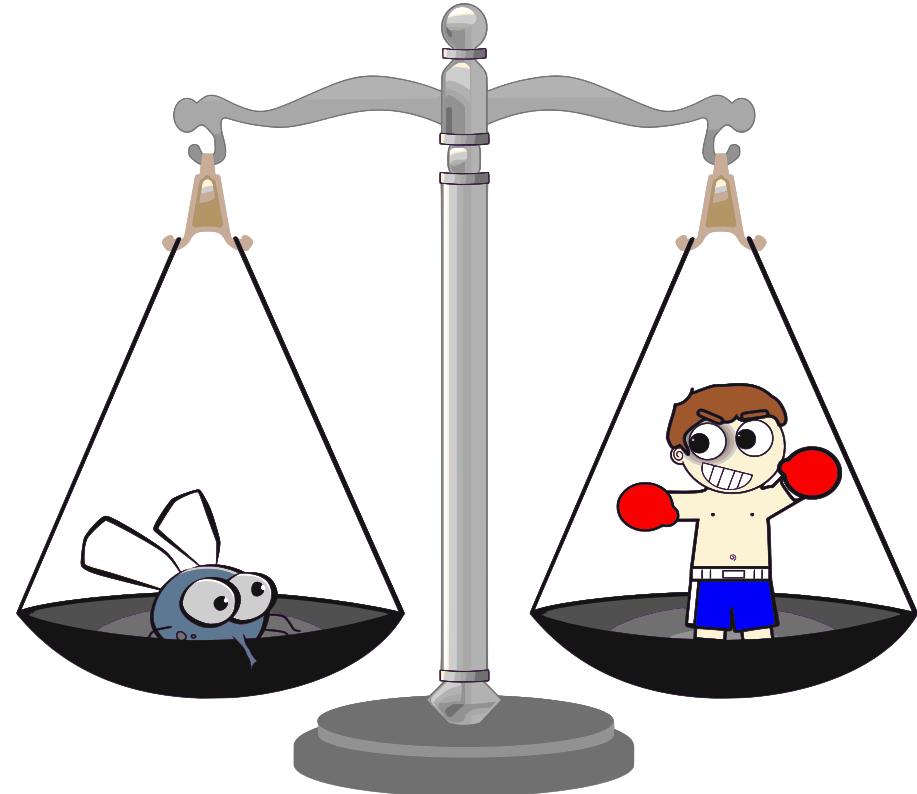
Sometimes we do not want the model with the highest accuracy.

There is often a tradeoff between (1) maximizing the accuracy of a prediction based on training data and (2) fairness incorporating contextual factors.

How to avoid self-fulfilling prophecies?

Responsible Data Science (Fairness)

1. Motivation
2. **Preliminaries**
3. Fairness Measures
4. Fair Decision Trees



Normal Itemsets

Bread	Butter	Chips	Beer
✓	✓		
		✓	✓
✓	✓	✓	✓
		✓	✓
...



Note: here we ignore quantities

{Bread, Butter}
 {Chips, Beer}
 {Bread, Butter, Chips, Beer}
 {Chips, Beer}
 ...

- Any dataset having instances and features can be converted into a multiset of transactions $\mathcal{X} \in \mathbb{M}(\mathbb{P}(\mathcal{I}))$
- Support of an itemset is defined as

$$\text{support}(\mathcal{A}) = \frac{|[\mathcal{T} \in \mathcal{X} | \mathcal{A} \subseteq \mathcal{T}]|}{|\mathcal{X}|}$$

$$\text{support}(\{\text{Bread}\}) = \frac{2}{4}$$

$$\text{support}(\{\text{Chips}, \text{Beer}\}) = \frac{3}{4}$$

$$\text{support}(\{\text{Bread}, \text{Beer}\}) = \frac{1}{4}$$

Itemsets Encoding Quantities

Bread	Butter	Chips	Beer
5	2	0	0
0	0	1	2
2	1	2	1
0	0	2	2
...



{Bread=5, Butter=2, Chips=0, Beer=0}
 {Bread=0, Butter=0, Chips=1, Beer=2}
 {Bread=2, Butter=1, Chips=2, Beer=1}
 {Bread=0, Butter=0, Chips=2, Beer=2}

...

$$\text{support}(\mathcal{A}) = \frac{|[\mathcal{T} \in \mathcal{X} | \mathcal{A} \subseteq \mathcal{T}]|}{|\mathcal{X}|}$$

$$\text{support}(\{\text{Bread} = 5\}) = \frac{1}{4}$$

$$\text{support}(\{\text{Bread} = 0\}) = \frac{2}{4}$$

$$\text{support}(\{\text{Bread} = 2, \text{Beer} = 1\}) = \frac{1}{4}$$

Itemsets Encoding Any Value

Age	City	Income	Gender
34	Bonn	2400	Male
45	Köln	1200	Male
39	Aachen	4200	Female
41	Bonn	2500	Female
...



{Age=34, City=Bonn, Income=2400, Gender=Male}
 {Age=45, City=Köln, Income=1200, Gender=Male}
 {Age=39, City=Aachen, Income=4200, Gender=Female}
 {Age=41, City=Bonn, Income=2500, Gender=Female}

...

$$\text{support}(\mathcal{A}) = \frac{|[\mathcal{T} \in \mathcal{X} | \mathcal{A} \subseteq \mathcal{T}]|}{|\mathcal{X}|}$$

$$\text{support}(\{\text{City} = \text{Bonn}\}) = \frac{2}{4}$$

$$\text{support}(\{\text{Age} = 34, \text{Income} = 0\}) = 0$$

$$\text{support}(\{\text{Age} = 34, \text{Gender} = \text{Male}\}) = \frac{1}{4}$$

Itemsets Encoding Ranges of Values

Age	City	Income	Gender
34	Bonn	2400	Male
45	Köln	1200	Male
39	Aachen	4200	Female
41	Bonn	2500	Female
...



$\{\text{Age} < 40, \text{City} = \text{Bonn}, \text{Income} < 3000, \text{Gender} = \text{Male}\}$
 $\{\text{Age} \geq 40, \text{City} = \text{Köln}, \text{Income} < 3000, \text{Gender} = \text{Male}\}$
 $\{\text{Age} < 40, \text{City} = \text{Aachen}, \text{Income} \geq 3000, \text{Gender} = \text{Female}\}$
 $\{\text{Age} \geq 40, \text{City} = \text{Bonn}, \text{Income} < 3000, \text{Gender} = \text{Female}\}$
 ...

$$\text{support}(\mathcal{A}) = \frac{|[\mathcal{T} \in \mathcal{X} | \mathcal{A} \subseteq \mathcal{T}]|}{|\mathcal{X}|}$$

$$\text{support}(\{\text{Age} < 40\}) = \frac{2}{4}$$

$$\text{support}(\{\text{Age} \geq 40, \text{Income} < 3000\}) = \frac{2}{4}$$

$$\text{support}(\{\text{Age} < 40, \text{Gender} = \text{Male}\}) = \frac{1}{4}$$

Itemsets Encoding Real Values

Age	City	Income (pred)	Income (real)	Gender
34	Bonn	2400	1667	Male
45	Köln	1200	1456	Male
39	Aachen	4200	3987	Female
41	Bonn	2500	2420	Female
...



{Age=34, City=Bonn, Income=1667, Gender=Male}
 {Age=45, City=Köln, Income=1456, Gender=Male}
 {Age=39, City=Aachen, Income=3987, Gender=Female}
 {Age=41, City=Bonn, Income=2420, Gender=Female}

...

- The itemsets can be partly based on predicted values
- Here, the **real values** are used for the income

By using real data values,
 we can check for
data bias

Itemsets Encoding Predicted Values

Age	City	Income (pred)	Income (real)	Gender
34	Bonn	2400	1667	Male
45	Köln	1200	1456	Male
39	Aachen	4200	3987	Female
41	Bonn	2500	2420	Female
...



{Age=34, City=Bonn, Income=2400, Gender=Male}
{Age=45, City=Köln, Income=1200, Gender=Male}
{Age=39, City=Aachen, Income=4200, Gender=Female}
{Age=41, City=Bonn, Income=2500, Gender=Female}

...

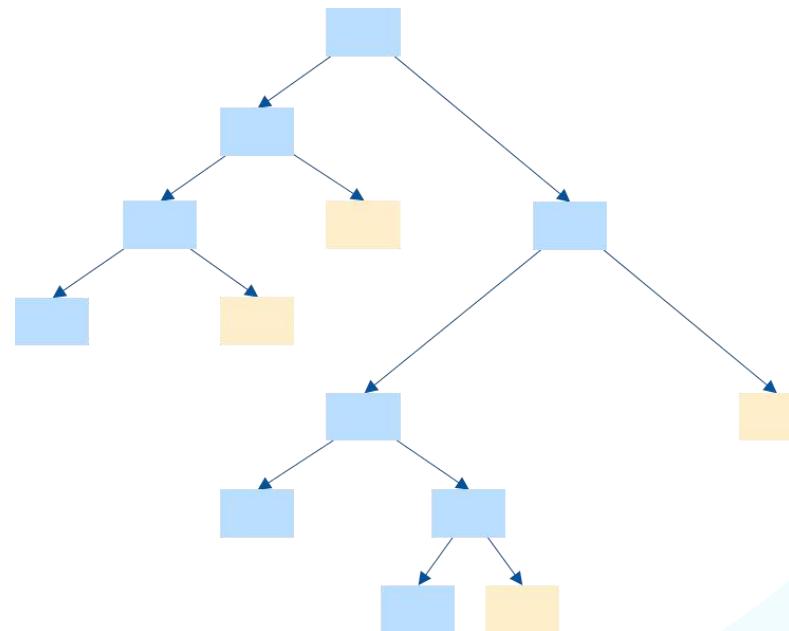
- The itemsets can be partly based on predicted values
- Here, the predicted values are used for the income

By using predicted values, we can check for **algorithmic (model) bias**

Itemsets Encoding Predicted Values

Age	City	Income (pred)	Income (real)	Gender
34	Bonn	2400	1667	Male
45	Köln	1200	1456	Male
39	Aachen	4200	3987	Female
41	Bonn	2500	2420	Female
...

Unfairness can be in the data and/or in the model. We cannot change the past, but we can change the model!



Association Rules

Bread	Butter	Chips	Beer
✓	✓		
		✓	✓
✓	✓	✓	✓
		✓	✓
...



{Bread, Butter}
 {Chips, Beer}
 {Bread, Butter, Chips, Beer}
 {Chips, Beer}

...

- Dataset $\mathcal{X} \in \mathbb{M}(\mathbb{P}(\mathcal{I}))$
- Association rules are of the form $\mathcal{A} \Rightarrow \mathcal{B}$ with $\mathcal{A} \subseteq \mathcal{I}, \mathcal{B} \subseteq \mathcal{I}$ and $\mathcal{A} \cap \mathcal{B} = \emptyset$
- For any rule, we can calculate its **confidence** as follows:

$$\text{conf}(\mathcal{A} \Rightarrow \mathcal{B}) = \frac{\text{support}(\mathcal{A} \cup \mathcal{B})}{\text{support}(\mathcal{A})}$$

$$\begin{aligned}
 \text{conf}(\{Bread\} \Rightarrow \{Butter\}) &= 1 \\
 \text{conf}(\{Chips\} \Rightarrow \{Beer\}) &= 1 \\
 \text{conf}(\{Chips, Beer\} \Rightarrow \{Bread\}) &= \frac{1}{3} \\
 \text{conf}(\{Bread\} \Rightarrow \{Chips, Beer\}) &= \frac{1}{2}
 \end{aligned}$$

Association Rules Using Feature Values

Age	City	Income	Gender
34	Bonn	2400	Male
45	Köln	1200	Male
39	Aachen	4200	Female
41	Bonn	2500	Female
...



{Age=34, City=Bonn, Income=2400, Gender=Male}
 {Age=45, City=Köln, Income=1200, Gender=Male}
 {Age=39, City=Aachen, Income=4200, Gender=Female}
 {Age=41, City=Bonn, Income=2500, Gender=Female}

...

$$\text{conf}(\mathcal{A} \Rightarrow \mathcal{B}) = \frac{\text{support}(\mathcal{A} \cup \mathcal{B})}{\text{support}(\mathcal{A})}$$

$$\begin{aligned}\text{conf}(\{Age = 34\} \Rightarrow \{City = Bonn\}) &= 1 \\ \text{conf}(\{City = Bonn\} \Rightarrow \{Age = 34\}) &= \frac{1}{2}\end{aligned}$$

Association Rules Using Value Ranges

Age	City	Income	Gender
34	Bonn	2400	Male
45	Köln	1200	Male
39	Aachen	4200	Female
41	Bonn	2500	Female
...



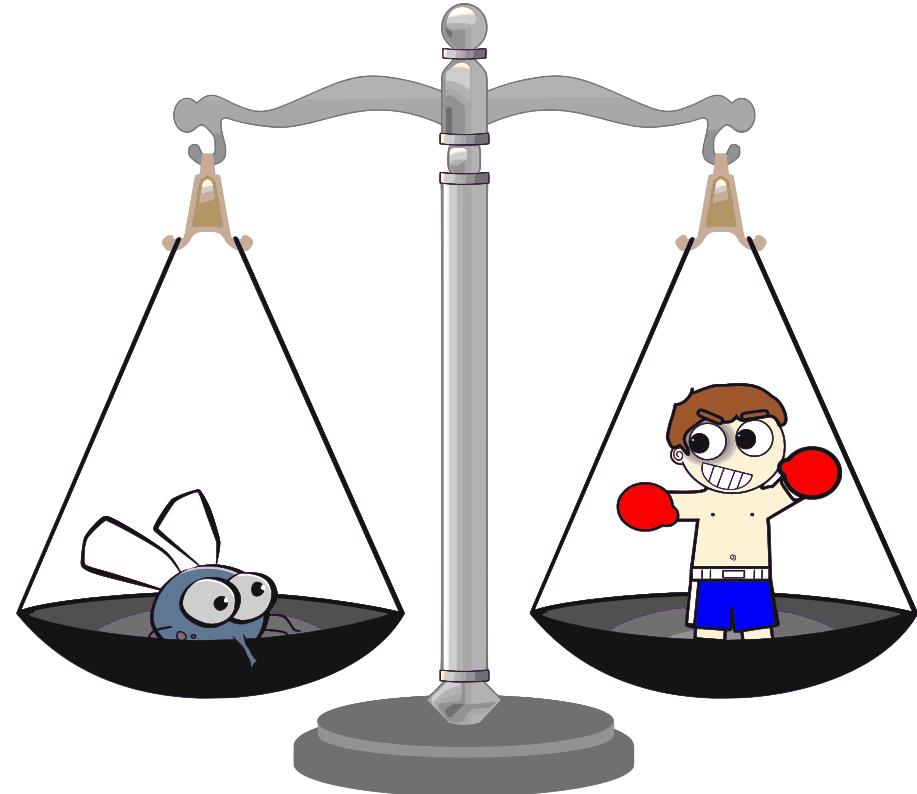
$\{\text{Age} < 40, \text{City} = \text{Bonn}, \text{Income} < 3000, \text{Gender} = \text{Male}\}$
 $\{\text{Age} \geq 40, \text{City} = \text{Köln}, \text{Income} < 3000, \text{Gender} = \text{Male}\}$
 $\{\text{Age} < 40, \text{City} = \text{Aachen}, \text{Income} \geq 3000, \text{Gender} = \text{Female}\}$
 $\{\text{Age} \geq 40, \text{City} = \text{Bonn}, \text{Income} < 3000, \text{Gender} = \text{Female}\}$
 ...

$$\text{conf}(\mathcal{A} \Rightarrow \mathcal{B}) = \frac{\text{support}(\mathcal{A} \cup \mathcal{B})}{\text{support}(\mathcal{A})}$$

$$\begin{aligned} \text{conf}(\{\text{Age} < 40\} \Rightarrow \{\text{Income} < 3000\}) &= \frac{1}{2} \\ \text{conf}(\{\text{Age} \geq 40\} \Rightarrow \{\text{Income} < 3000\}) &= \frac{2}{4} \\ \text{conf}(\{\text{City} = \text{Bonn}\} \Rightarrow \{\text{Income} < 3000\}) &= 1 \end{aligned}$$

Responsible Data Science (Fairness)

1. Motivation
2. Preliminaries
- 3. Fairness Measures**
4. Fair Decision Trees



Effect: Quantifying The Influence of a Potentially Discriminating Itemset

- A dataset with instances and features can be converted into a multiset of transactions $\mathcal{X} \in \mathbb{M}(\mathbb{P}(\mathcal{I}))$ (see previous video)
- A **potentially discriminating itemset** $\mathcal{D} \subseteq \mathcal{I}$ is an itemset that we do **not** want to have an effect on our results (rules, predictions, outcomes, etc.).
- The **effect** of a potentially discriminating itemset \mathcal{D} on an **association rule** $\mathcal{A} \Rightarrow \mathcal{B}$ is defined as
$$\text{effect}_{\mathcal{D}}(\mathcal{A} \Rightarrow \mathcal{B}) = \frac{\text{conf}(\mathcal{A} \cup \mathcal{D} \Rightarrow \mathcal{B})}{\text{conf}(\mathcal{A} \Rightarrow \mathcal{B})}$$
- If adding the potentially discriminating itemset has **no effect** on the confidence of the rule, then
$$\text{effect}_{\mathcal{D}}(\mathcal{A} \Rightarrow \mathcal{B}) \approx 1$$

Effect: Interpretation In The Context Of An Association Rule

- The **effect** of a potentially discriminating itemset \mathcal{D} on an association rule $\mathcal{A} \Rightarrow \mathcal{B}$ in the context of some data set $\mathcal{X} \in \mathbb{M}(\mathbb{P}(\mathcal{I}))$ is defined as

$$\text{effect}_{\mathcal{D}}(\mathcal{A} \Rightarrow \mathcal{B}) = \frac{\text{conf}(\mathcal{A} \cup \mathcal{D} \Rightarrow \mathcal{B})}{\text{conf}(\mathcal{A} \Rightarrow \mathcal{B})}$$

- If adding the potentially discriminating itemset has **a positive effect** on the confidence of the rule, then

$$\text{effect}_{\mathcal{D}}(\mathcal{A} \Rightarrow \mathcal{B}) > 1$$

- If adding the potentially discriminating itemset, has **a negative effect** on the confidence of the rule, then

$$\text{effect}_{\mathcal{D}}(\mathcal{A} \Rightarrow \mathcal{B}) < 1$$

Effect – Example

Age	City	Income	Gender	Health Status
34	Bonn	2400	Male	Good
45	Köln	4800	Male	Medium
39	Aachen	4200	Female	Medium
41	Bonn	2400	Female	Bad
25	Aachen	1000	Male	Good
55	Bonn	5000	Female	Good
34	Aachen	2200	Female	Good
22	Köln	1500	Male	Bad
29	Bonn	2300	Male	Medium
44	Aachen	2600	Male	Medium
...

$$\text{effect}_{\mathcal{D}}(\mathcal{A} \Rightarrow \mathcal{B}) = \frac{\text{conf}(\mathcal{A} \cup \mathcal{D} \Rightarrow \mathcal{B})}{\text{conf}(\mathcal{A} \Rightarrow \mathcal{B})}$$

$$\begin{aligned}\mathcal{A} &= \{City = Bonn\} \\ \mathcal{B} &= \{Health Status = Good\} \\ \mathcal{D} &= \{Income < 2500\}\end{aligned}$$

$$\text{conf}(\mathcal{A} \cup \mathcal{D} \Rightarrow \mathcal{B}) = \frac{\frac{1}{10}}{\frac{3}{10}} = \frac{1}{3}$$

$$\text{conf}(\mathcal{A} \Rightarrow \mathcal{B}) = \frac{\frac{2}{10}}{\frac{4}{10}} = \frac{1}{2}$$

$$\text{effect}_{\mathcal{D}}(\mathcal{A} \Rightarrow \mathcal{B}) = \frac{\frac{1}{3}}{\frac{1}{2}} = \frac{2}{3} < 1$$

Adding the potentially discriminating itemset has a negative effect on the confidence of the rule

Effect – Example

Age	City	Income	Gender	Health Status
34	Bonn	2400	Male	Good
45	Köln	4800	Male	Medium
39	Aachen	4200	Female	Medium
41	Bonn	2400	Female	Bad
25	Aachen	1000	Male	Good
55	Bonn	5000	Female	Good
34	Aachen	2200	Female	Good
22	Köln	1500	Male	Bad
29	Bonn	2300	Male	Medium
44	Aachen	2600	Male	Medium
...

$$\text{effect}_{\mathcal{D}}(\mathcal{A} \Rightarrow \mathcal{B}) = \frac{\text{conf}(\mathcal{A} \cup \mathcal{D} \Rightarrow \mathcal{B})}{\text{conf}(\mathcal{A} \Rightarrow \mathcal{B})}$$

$$\mathcal{A} = \{City = Aachen\}$$

$$\mathcal{B} = \{Health Status = Good\}$$

$$\mathcal{D} = \{Income < 2500\}$$

$$\text{conf}(\mathcal{A} \cup \mathcal{D} \Rightarrow \mathcal{B}) = \frac{\frac{2}{10}}{\frac{2}{10}} = 1$$

$$\text{conf}(\mathcal{A} \Rightarrow \mathcal{B}) = \frac{\frac{2}{4}}{\frac{10}{10}} = \frac{1}{2}$$

$$\text{effect}_{\mathcal{D}}(\mathcal{A} \Rightarrow \mathcal{B}) = \frac{1}{\frac{1}{2}} = 2 > 1$$

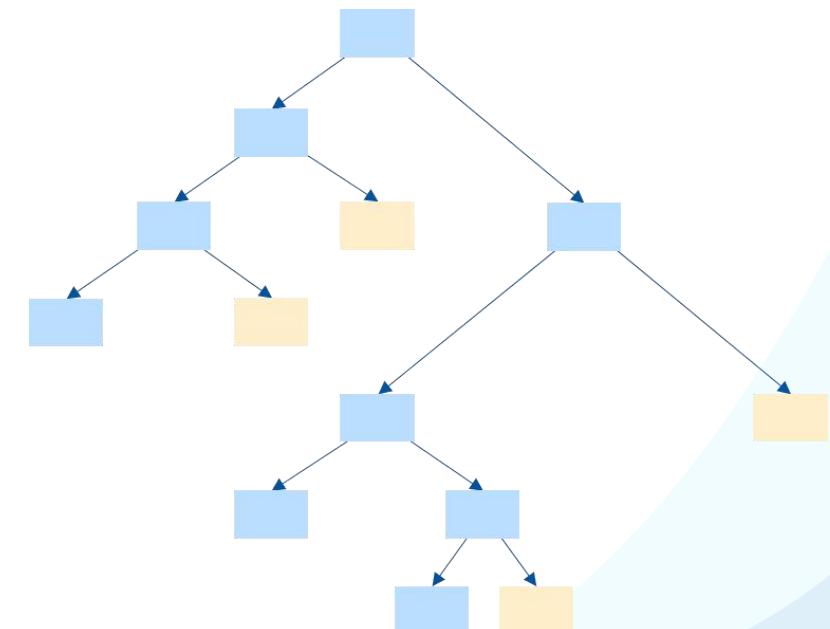
Adding the potentially discriminating itemset has a positive effect on the confidence of the rule

Effect – Outcome (e.g., from a decision tree)

- Assume a data set $\mathcal{X} \in \mathbb{M}(\mathbb{P}(\mathcal{I}))$, but now we consider the effect of potentially discriminating itemset $\mathcal{D} \subseteq \mathcal{I}$ on some outcome $\mathcal{B} \subseteq \mathcal{I}$ (e.g., decision to hire someone)
- The effect of a potentially discriminating itemset \mathcal{D} on some outcome \mathcal{B} is defined as:

$$\text{effect}_{\mathcal{D}}(\mathcal{B}) = \frac{\text{support}(\mathcal{B} \cup \mathcal{D})}{\text{support}(\mathcal{B}) \cdot \text{support}(\mathcal{D})}$$

- If there is **no effect**, we expect to see
 $\text{effect}_{\mathcal{D}}(\mathcal{B}) \approx 1$
- If there is **a positive effect** on the likelihood of outcome \mathcal{B} , then
 $\text{effect}_{\mathcal{D}}(\mathcal{B}) > 1$
- If there is **a negative effect** on the likelihood of outcome \mathcal{B} , then
 $\text{effect}_{\mathcal{D}}(\mathcal{B}) < 1$



Recall, we can change the predicted value by tweaking our model!

Effect – Outcome (Example)

Age	City	Income	Gender	Health Status
34	Bonn	2400	Male	Good
45	Köln	4800	Male	Medium
39	Aachen	4200	Female	Medium
41	Bonn	2400	Female	Bad
25	Aachen	1000	Male	Good
55	Bonn	5000	Female	Good
34	Aachen	2200	Female	Good
22	Köln	1500	Male	Bad
29	Bonn	2300	Male	Medium
44	Aachen	2600	Male	Medium
...

$$\text{effect}_{\mathcal{D}}(\mathcal{B}) = \frac{\text{support}(\mathcal{B} \cup \mathcal{D})}{\text{support}(\mathcal{B}) \cdot \text{support}(\mathcal{D})}$$

$$\mathcal{B} = \{\text{Health Status} = \text{Good}\}$$

$$\mathcal{D} = \{\text{Income} > 4000\}$$

$$\text{support}(\mathcal{B} \cup \mathcal{D}) = \frac{1}{10}$$

$$\text{support}(\mathcal{B}) \cdot \text{support}(\mathcal{D}) = \frac{4}{10} \cdot \frac{3}{10} = \frac{12}{100}$$

$$\text{effect}_{\mathcal{D}}(\mathcal{B}) = \frac{\frac{1}{10}}{\frac{12}{100}} = \frac{5}{6} < 1$$

There is a **slightly negative effect** on the likelihood of the outcome (the value is close to one meaning that there is nearly no effect)

Effect – Outcome (Example)

Age	City	Income	Gender	Health Status
34	Bonn	2400	Male	Good
45	Köln	4800	Male	Medium
39	Aachen	4200	Female	Medium
41	Bonn	2400	Female	Bad
25	Aachen	1000	Male	Good
55	Bonn	5000	Female	Good
34	Aachen	2200	Female	Good
22	Köln	1500	Male	Bad
29	Bonn	2300	Male	Medium
44	Aachen	2600	Male	Medium
...

$$\text{effect}_{\mathcal{D}}(\mathcal{B}) = \frac{\text{support}(\mathcal{B} \cup \mathcal{D})}{\text{support}(\mathcal{B}) \cdot \text{support}(\mathcal{D})}$$

$$\mathcal{B} = \{\text{Health Status} = \text{Good}\}$$

$$\mathcal{D} = \{\text{Age} < 35\}$$

$$\text{support}(\mathcal{B} \cup \mathcal{D}) = \frac{3}{10}$$

$$\text{support}(\mathcal{B}) \cdot \text{support}(\mathcal{D}) = \frac{4}{10} \cdot \frac{5}{10} = \frac{20}{100}$$

$$\text{effect}_{\mathcal{D}}(\mathcal{B}) = \frac{\frac{3}{10}}{\frac{20}{100}} = \frac{3}{2} > 1$$

There is a positive effect on the likelihood of the outcome

Discrimination – Association Rules

- Assume a data set $\mathcal{X} \in \mathbb{M}(\mathbb{P}(\mathcal{I}))$
- The level of discrimination given a potentially discriminating itemset $\mathcal{D} \subseteq \mathcal{I}$ on some association rule $\mathcal{A} \Rightarrow \mathcal{B}$ is defined as

$$\text{disc}_{\mathcal{D}}(\mathcal{A} \Rightarrow \mathcal{B}) = |\text{conf}(\mathcal{A} \cup \mathcal{D} \Rightarrow \mathcal{B}) - \text{conf}(\mathcal{A} \Rightarrow \mathcal{B})|$$

- This yields a value between 0 and 1 where:
 - 0 – no discrimination
 - 1 – maximal discrimination

Discrimination – Outcome

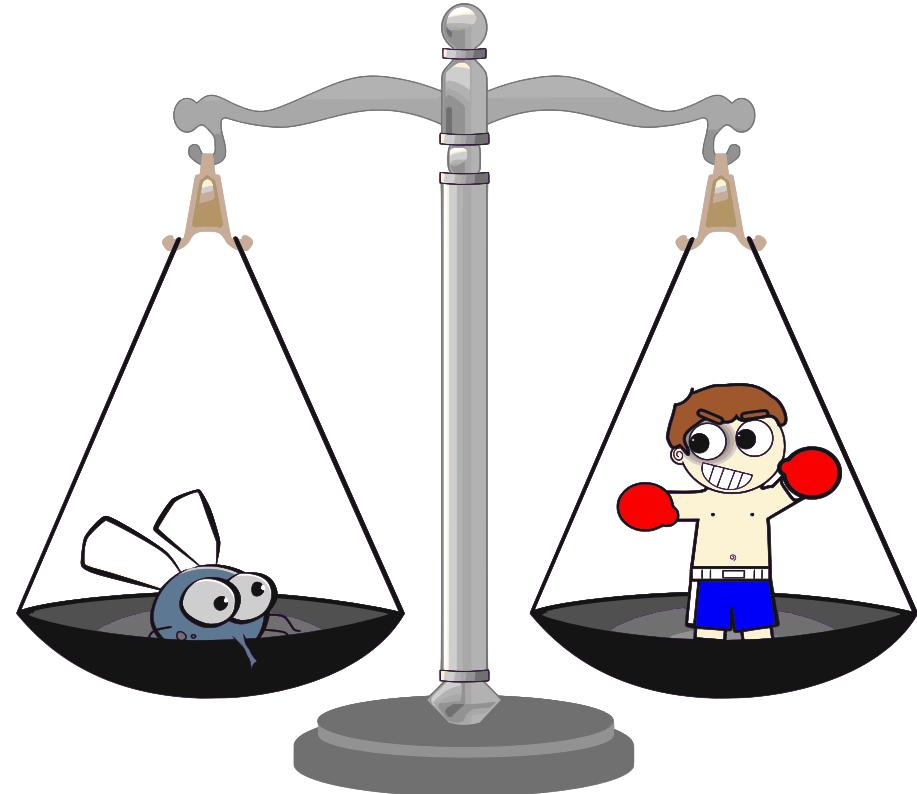
- Assume a data set $\mathcal{X} \in \mathbb{M}(\mathbb{P}(\mathcal{I}))$
- The level of discrimination given a potentially discriminating itemset $\mathcal{D} \subseteq \mathcal{I}$ on some outcome $\mathcal{B} \subseteq \mathcal{I}$ is defined as

$$\text{disc}_{\mathcal{D}}(\mathcal{B}) = |\text{support}(\mathcal{B} \cup \mathcal{D}) - \text{support}(\mathcal{B}) \cdot \text{support}(\mathcal{D})|$$

- This yields a value between 0 and 1 where:
 - 0 – no discrimination
 - 1 – maximal discrimination

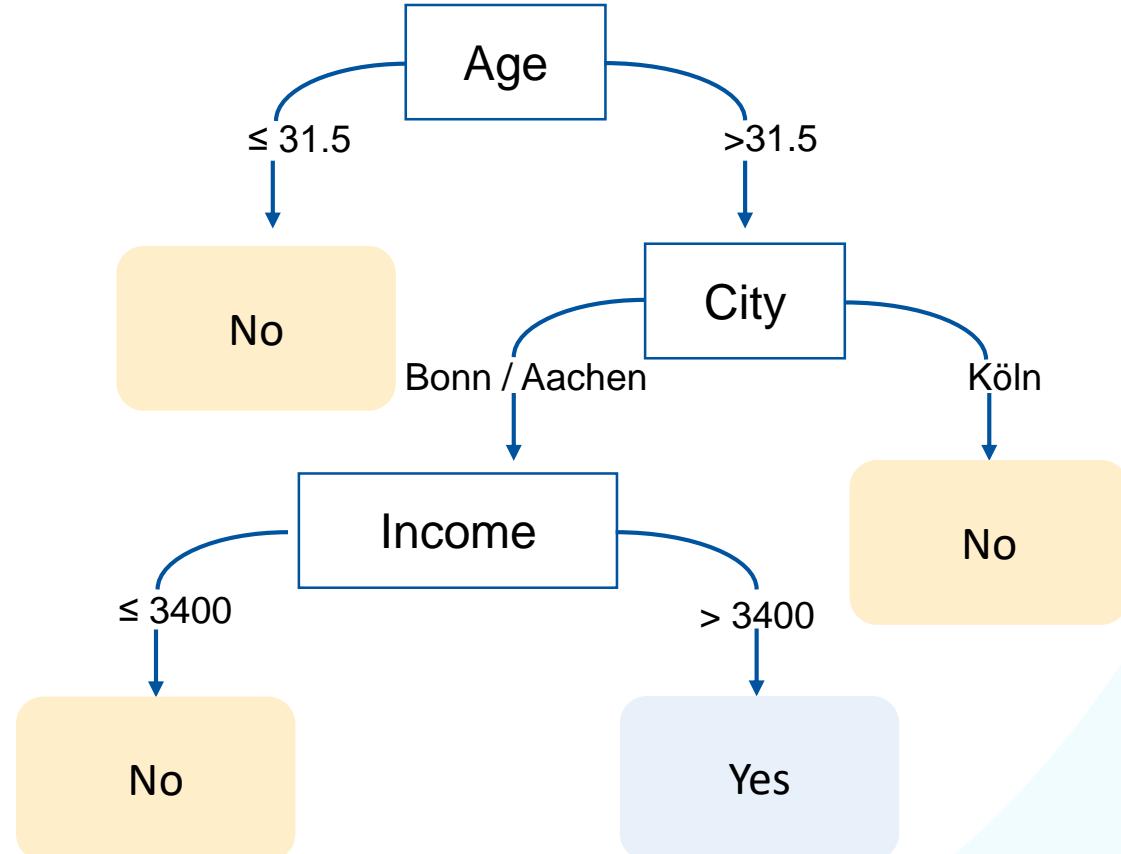
Responsible Data Science (Fairness)

1. Motivation
2. Preliminaries
3. Fairness Measures
4. **Fair Decision Trees**



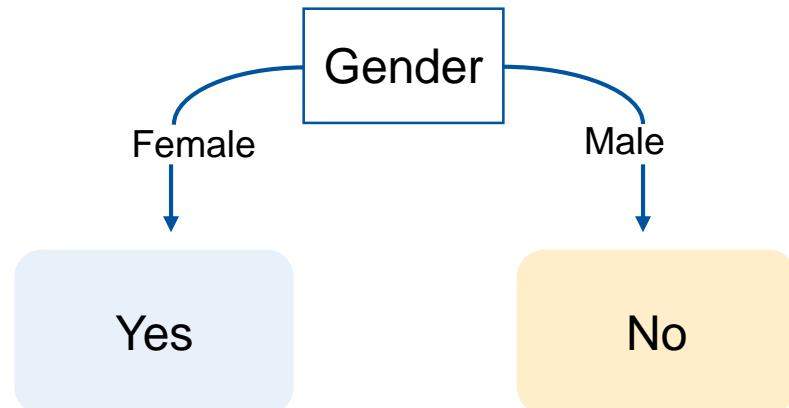
Biased Historic Data May Affect Decisions

Age	City	Income	Flat Ownership
34	Bonn	2400	No
45	Köln	4800	No
39	Aachen	4200	Yes
41	Bonn	2400	Yes
25	Aachen	1000	No
55	Bonn	5000	Yes
34	Aachen	3500	Yes
22	Köln	1500	No
29	Bonn	2300	No
44	Aachen	2600	No
...



Biased Historic Data May Affect Decisions

Age	City	Income	Gender	Flat Ownership
34	Bonn	2400	Male	No
45	Köln	4800	Male	No
39	Aachen	4200	Female	Yes
41	Bonn	2400	Female	Yes
25	Aachen	1000	Male	No
55	Bonn	5000	Female	Yes
34	Aachen	3500	Female	Yes
22	Köln	1500	Male	No
29	Bonn	2300	Male	No
44	Aachen	2600	Male	No
...	



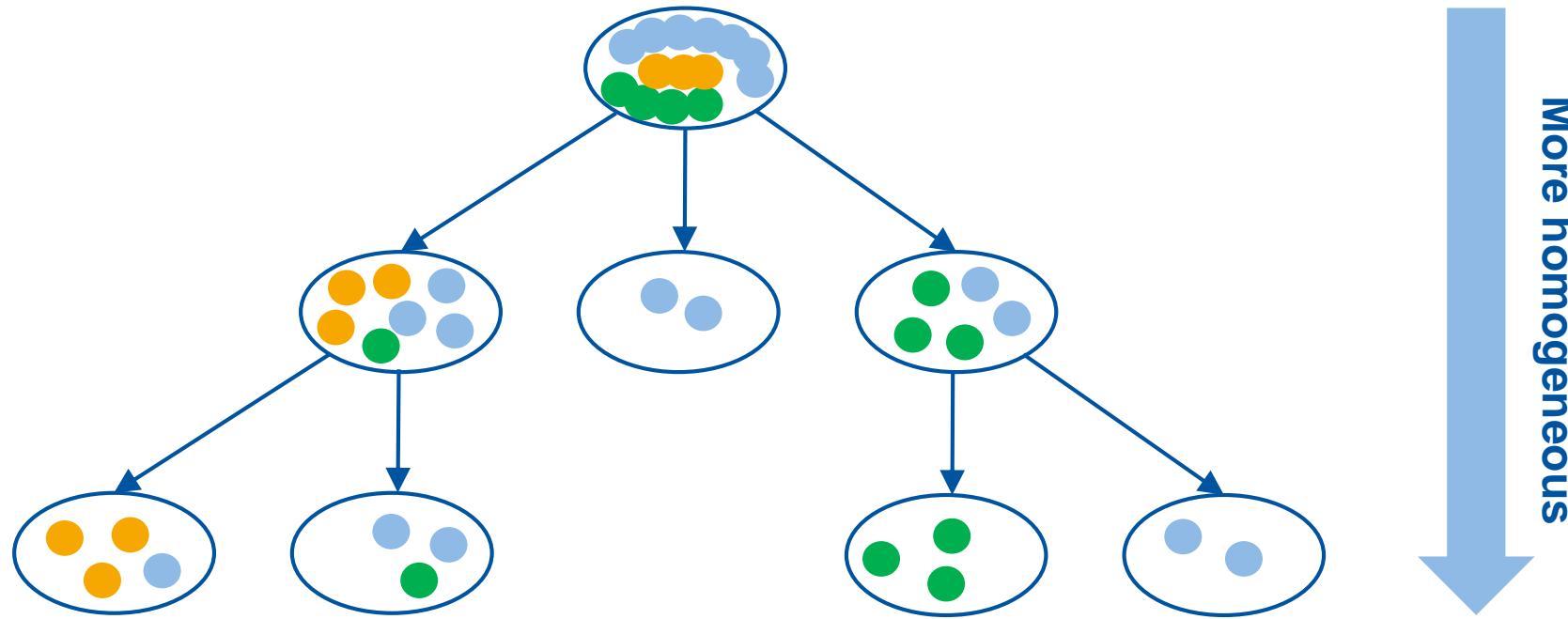
Attribute we do not want to have impact
(due to biased data or fairness reasons)

Making Decision Trees Fair – Three Approaches

- **Pre-processing**
 - Removing discriminatory features from data
 - Removing or duplicating instances
 - Problem: indirect discrimination
- **In-processing**
 - Considering a dependency on discriminatory features and the accuracy of the split while making a decision tree
- **Post-processing**
 - Relabeling leaves in a way that discrimination is lowered
 - Problem: loss in the accuracy

Always a trade-off between fairness and accuracy!

Traditional Decision Tree Learning Using Information Gain



Check slides on
decision trees
for details.

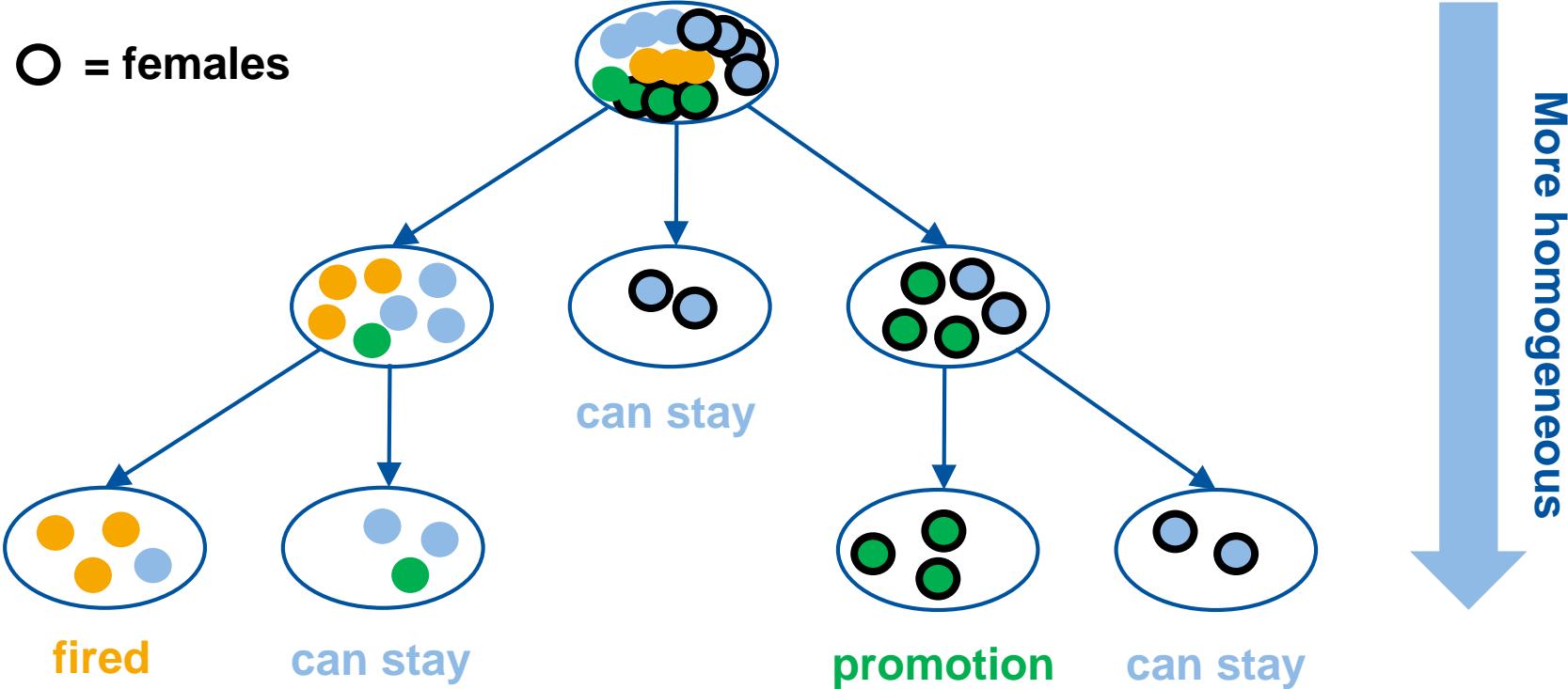
Information gain = improvement in knowledge

$$IG(d) = H(t) - H_W^d(t)$$

$$H(t) = - \sum_{k=1}^K (P(t = k) \cdot \log_s(P(t = k)))$$

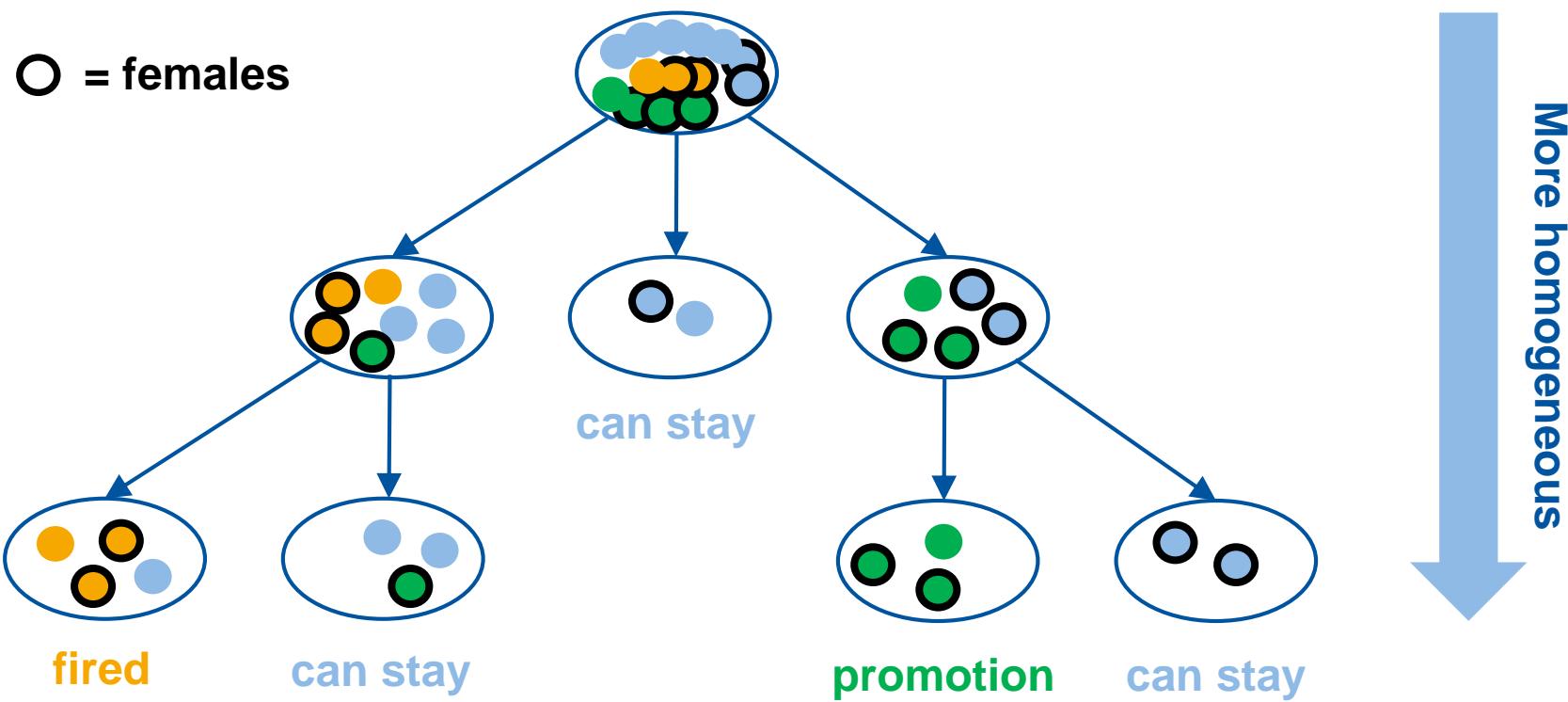
$$H_W(t) = \sum_{node \in nodes(d)} \left(\frac{|node|}{N} \cdot H_{node}(t) \right)$$

What If



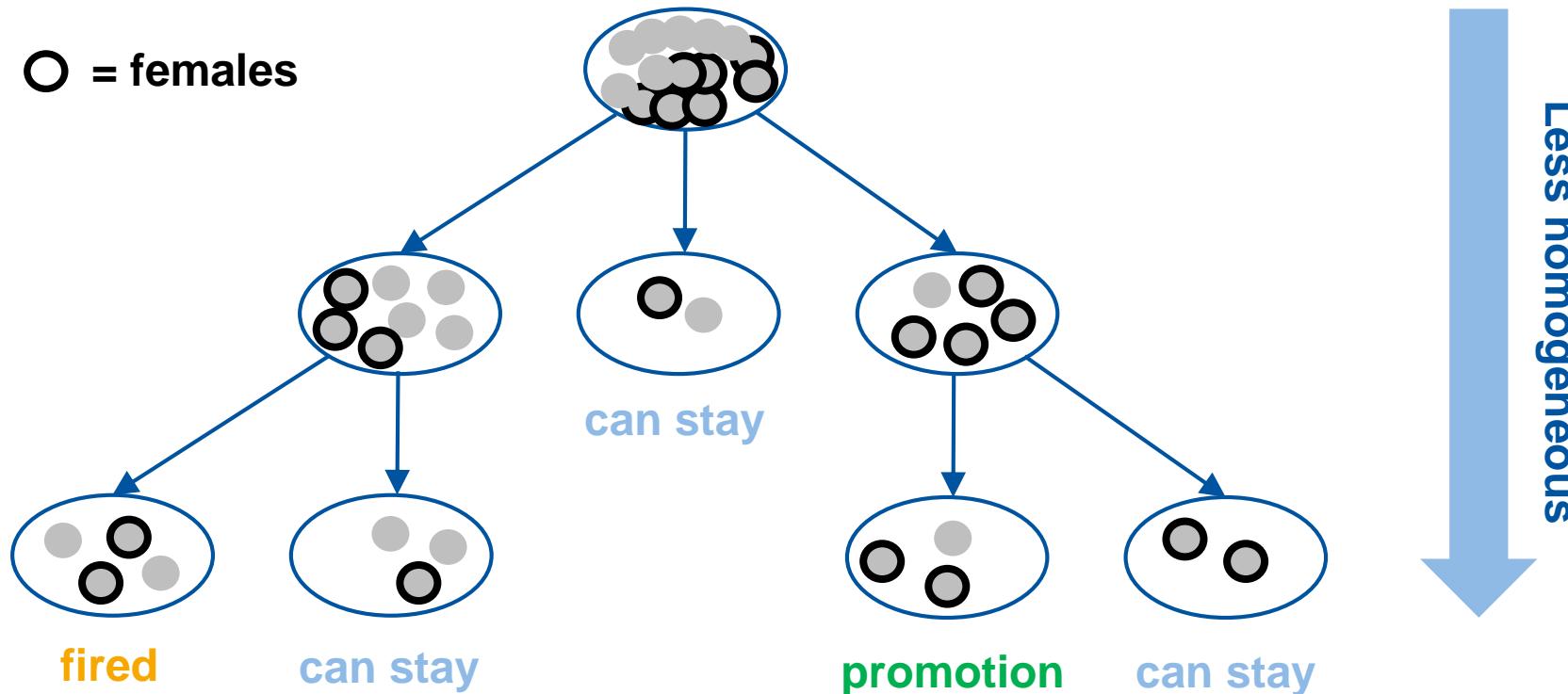
Unfair?

What If



Fair?

Solution (In-Processing) – Information Gain in Sensitivity



$$IGS(d) = H(b) - H_W^d(b)$$

(lower is better)

$$H(b) = - \sum_{k=1}^K (P(b = k) \cdot \log_s(P(b = k)))$$

$$H_W(b) = \sum_{node \in nodes(d)} \left(\frac{|node|}{N} \cdot H_{node}(b) \right)$$

Solution (In-Processing) – Two Forces When Splitting

IGC = classical information gain

$$IGC(d) = H(t) - H_W^d(t)$$

maximize

IGS = gain in sensitivity



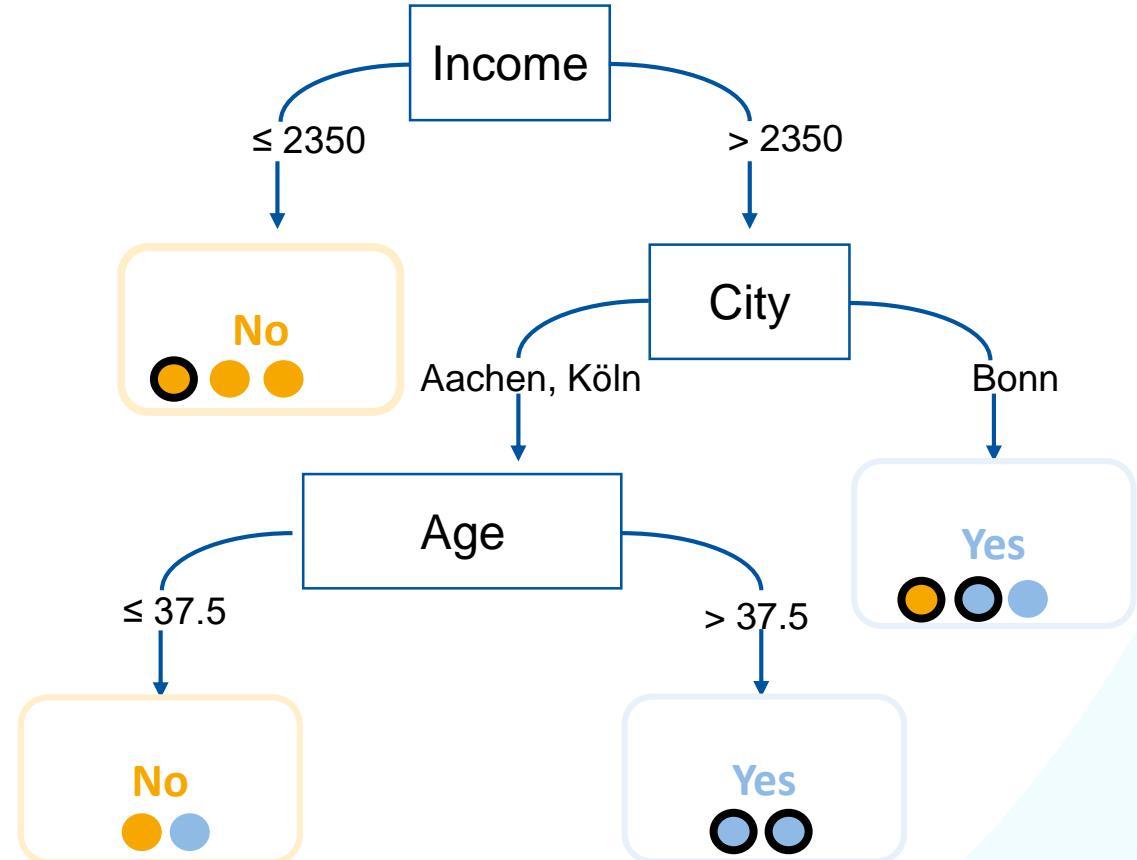
$$IGS(d) = H(b) - H_W^d(b)$$

minimize

Combine both!

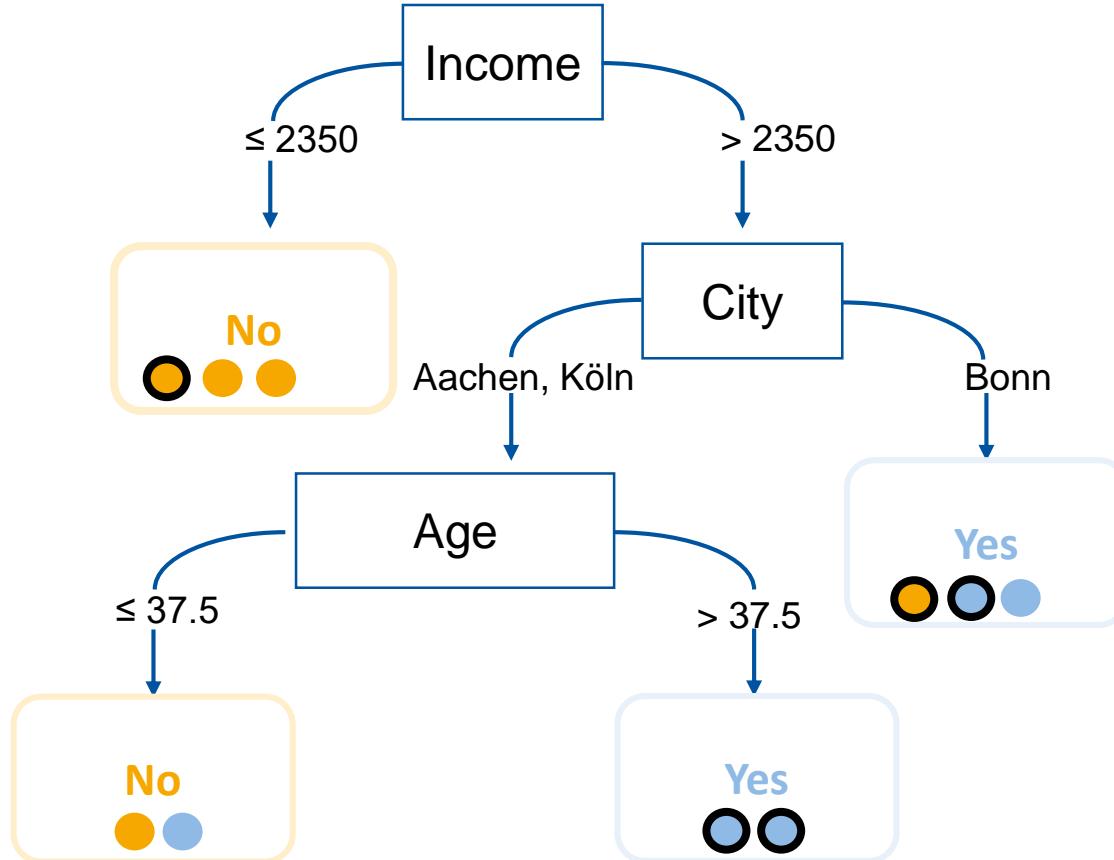
Is This Decision Tree Fair?

Age	City	Income	Gender (disc feature)	Flat Ownership (target)
34	Bonn	2400	Male	Yes
36	Bonn	4800	Female	Yes
39	Aachen	4200	Female	Yes
41	Köln	2400	Female	Yes
25	Aachen	2600	Male	Yes
55	Bonn	5000	Female	No
34	Aachen	3500	Male	No
22	Köln	1500	Male	No
29	Bonn	2300	Male	No
39	Aachen	2200	Female	No
...



$$\text{Accuracy} = \frac{8}{10} = 80\%$$

Solution (Post-Processing) – Measuring Discrimination



Compute discrimination (outcome)

$$\text{disc}_{\mathcal{D}}(\mathcal{B}) = |\text{support}(\mathcal{B} \cup \mathcal{D}) - \text{support}(\mathcal{B}) \cdot \text{support}(\mathcal{D})|$$

\mathcal{B} : outcome

(target itemset, e.g., Flat Ownership = Yes)

\mathcal{D} : potentially discriminating itemset

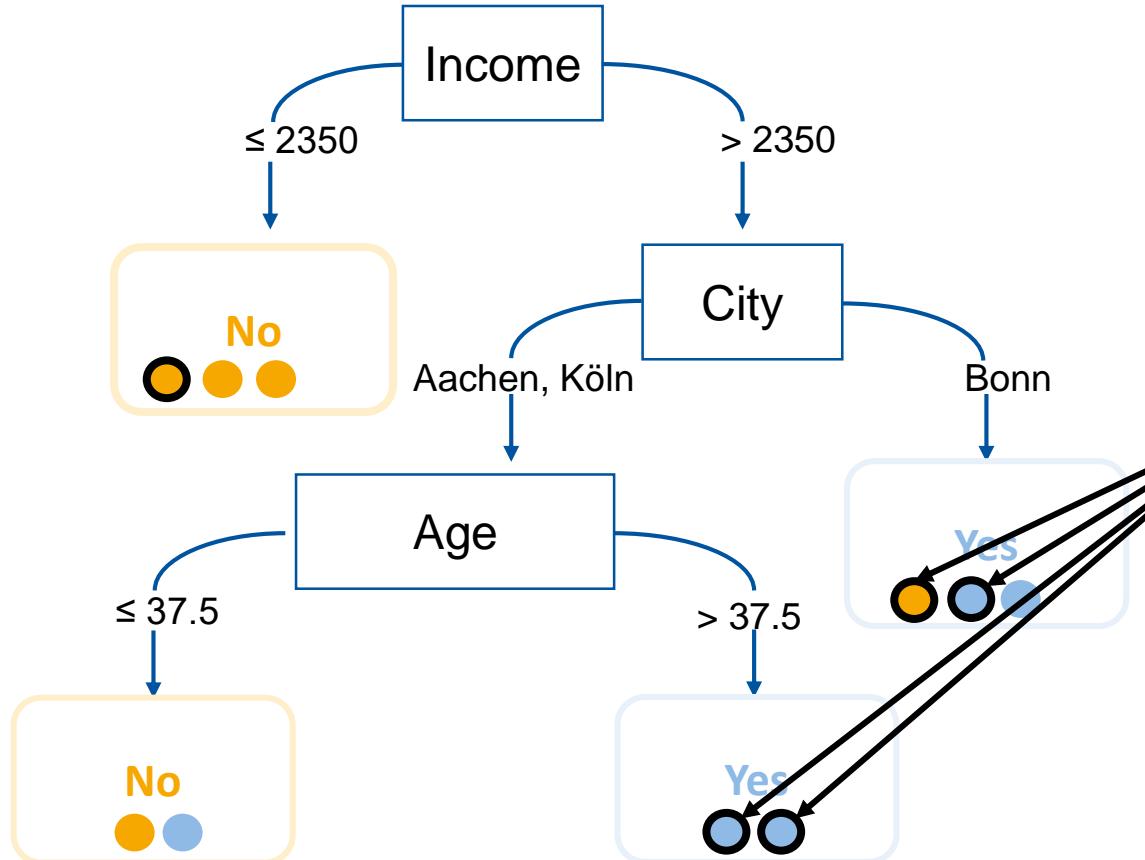
(e.g., Gender = Female)

→ A discrimination close to 0 means no discrimination

→ A discrimination close to 1 means maximal discrimination

$$\text{Accuracy} = \frac{8}{10} = 80\%$$

Solution (Post-Processing) – Measuring Discrimination



Compute discrimination (outcome)

\mathcal{B} : Flat Ownership = Yes

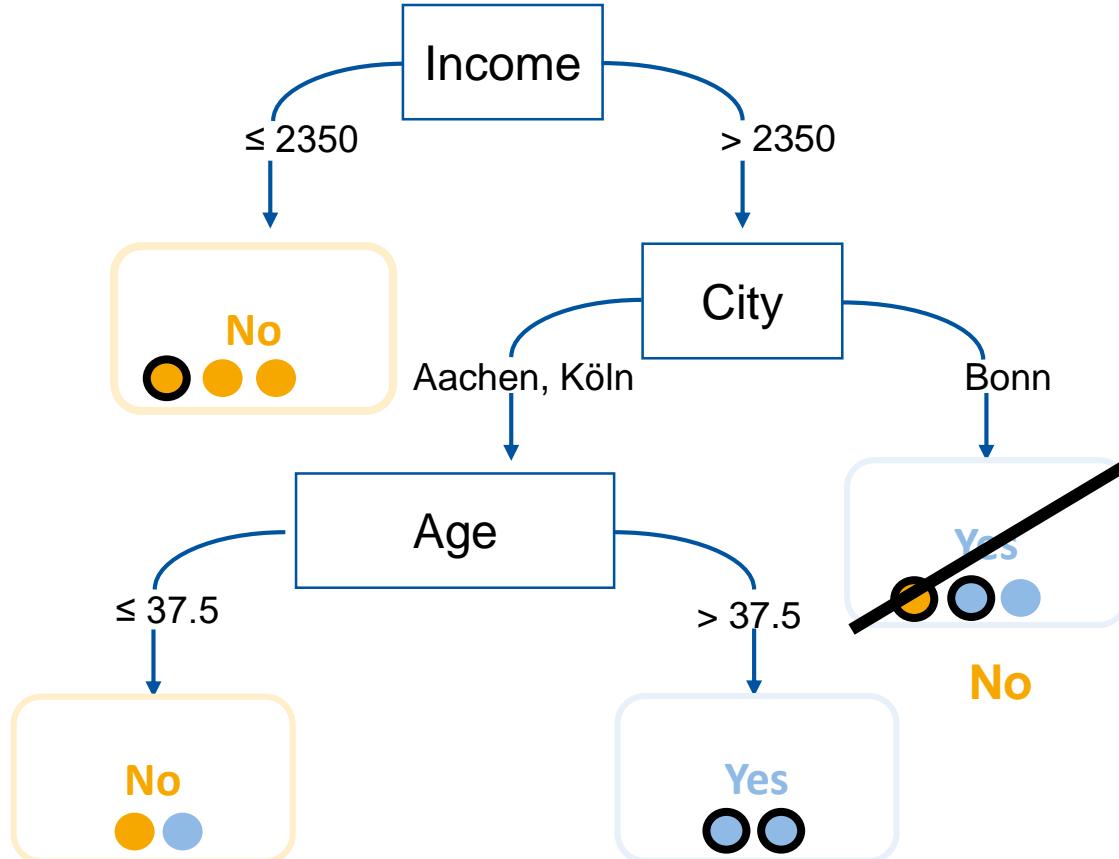
\mathcal{D} : Gender = Female

$$\text{disc}_{\mathcal{D}}(\mathcal{B}) = |\text{support}(\mathcal{B} \cup \mathcal{D}) - \text{support}(\mathcal{B}) \cdot \text{support}(\mathcal{D})|$$

$$= \left| \frac{4}{10} - \frac{5}{10} \cdot \frac{5}{10} \right| = 0.15$$

$$\text{Accuracy} = \frac{8}{10} = 80\%$$

Solution (Post-Processing) – Relabeling Leaves



Compute discrimination (outcome)

$$\begin{aligned} \text{disc}_{\mathcal{D}}(\mathcal{B}) &= |\text{support}(\mathcal{B} \cup \mathcal{D}) - \text{support}(\mathcal{B}) \cdot \text{support}(\mathcal{D})| \\ &= \left| \frac{2}{10} - \frac{2}{10} \cdot \frac{5}{10} \right| = 0.1 \end{aligned}$$

$$\text{Accuracy} = \frac{7}{10} = 70\%$$

- Reduced discrimination
- Reduced accuracy

Conclusion: Responsible Data Science

- Four **key concerns**: Fairness, Accuracy, Confidentiality, and Transparency (FACT, not FAIR).
- **Confidentiality**: Encryption, anonymization, K-Anonymity, L-Diversity, and T-Closeness
- Measuring **fairness** (effect) and making models fair.



Generated using DALL-E 3

With great power comes great responsibility!



Generated using DALL-E 3



Learn more? Visit: www.vdaalst.com & www.pads.rwth-aachen.de

Preparation for class on Friday, 19 January 2024 (mandatory):

Read the following research paper:

Oded Maron and Andrew Moore: Hoeffding Races: Accelerating Model Selection Search for Classification and Function Approximation. Advances in Neural Information Processing Systems 6 (NIPS 1993): 59-66, 1993.
(The paper is available online at <https://proceedings.neurips.cc/>.)

Focus on the following questions (which will be further explored in TPS exercises in class):

- (1) What is the fundamental problem when using cross-validation
(or performance on a validation set) to select between different ML models?
- (2) What is the key idea behind Hoeffding races and how does it address the problem identified in (1)?
- (3) What is the role of the parameters Δ and δ , respectively?

Bring your answers to these questions (which can be in the form of bullet points) to class;
they will be the basis for TSP exercises).

NB: Full understanding of the proof in Section 3 is desirable but not essential.



Elements of Machine Learning & Data Science

Winter semester 2023/24

Automated Machine Learning (1)

Prof. Holger Hoos

Key questions:

- How good is an ML model?
- How good could an ML model be?



Key questions:

- How good is an ML model?
 - Is it “fit for use” (i.e., good enough for deployment)?
 - What are its strengths and weaknesses?
 - Might anything have gone wrong during training?

Key questions:

- How good is an ML model?
 - How do we assess whether it is “fit for use”
(i.e., good enough for deployment)?
 - How do we assess its strengths and weaknesses?
 - How do we detect if anything has gone wrong during training?

Key questions:

- How good could an ML model be?
 - Are we using the best possible ML method / model?
 - Have we configured and trained it in the best possible way?
 - Can we further improve performance?

Key questions:

- How good could an ML model be?
 - How can we ensure we are using a good ML method / model?
 - How can we configure and train it for optimised performance?
 - How can we further improve performance?

High-level learning goals:

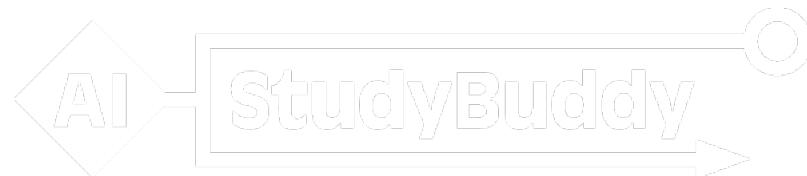
Be able to ...

- answer these key questions in a technical manner;
- recognise weaknesses in the empirical performance of ML models using standard tools and methods;
- explain these analysis tools and methods at a technical level;
- use standard tools and methods for selecting models and optimising their hyperparameters;
- explain these AutoML tools and methods at a technical level.

TPS Exercise

You want to solve a supervised classification problem.

Question:



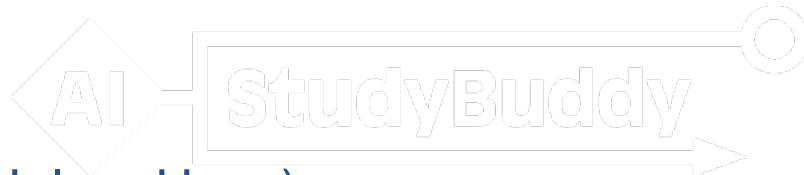
How to make sure you to construct the best possible classifier,
i.e., which methods and techniques should be applied?

Automated Machine Learning

Key idea:

Automate the decisions that have to be made when constructing effective machine learning models / pipelines

Methods/approaches:

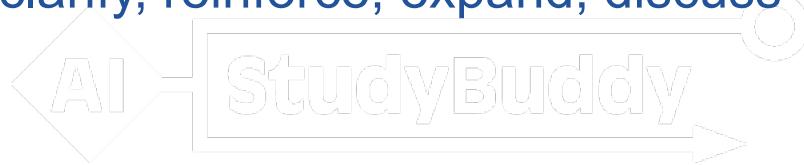


- model selection -> today (Holger Hoos)
- hyper-parameter optimisation (HPO) -> next Tuesday (Anja Jancovic)
- neural architecture search (NAS) -> next Friday (Marie Anastacio)
- combined algorithm selection and hyperparameter optimization (CASH), automating data science -> Tuesday, 30 January (Holger Hoos)

All four classroom sessions: Inverted classroom approach

Key idea:

- You learn relevant concepts and approaches via assigned pre-class homework (reading, answering questions, ...)
- We use classroom time to clarify, reinforce, expand, discuss



Advantages:

- You have more control over your individual learning process (but also more responsibility)
- We use classroom time more effectively
- Classes should be less boring and more fun

Note: For this to work, it is essential that you do homework, participate and take notes.

Preparation for today:

Read the following research paper:

Oded Maron and Andrew Moore: Hoeffding Races: Accelerating Model Selection Search for Classification and Function Approximation. Advances in Neural Information Processing Systems 6 (NIPS 1993): 59-66, 1993.
(The paper is available online at <https://proceedings.neurips.cc/>.)

Focus on the following questions (which will be further explored in TPS exercises in class):

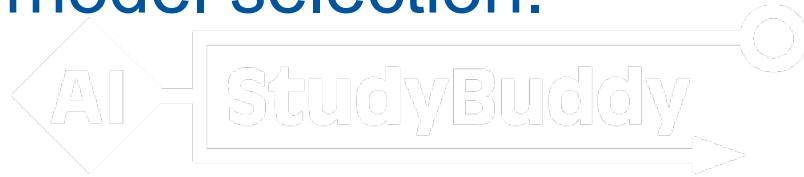
- 
- (1) What is the fundamental problem when using cross-validation
(or performance on a validation set) to select between different ML models?
 - (2) What is the key idea behind Hoeffding Races and how does it address the problem identified in (1)?
 - (3) What is the role of the parameters Δ and δ , respectively?

Bring your answers to these questions (which can be in the form of bullet points) to class;
they will be the basis for TSP exercises).

NB: Full understanding of the proof in Section 3 is desirable but not essential.

TPS Exercise (T part = done as preparation for class)

You have read the research paper by Maron & Moore about Hoeffding Races for model selection.

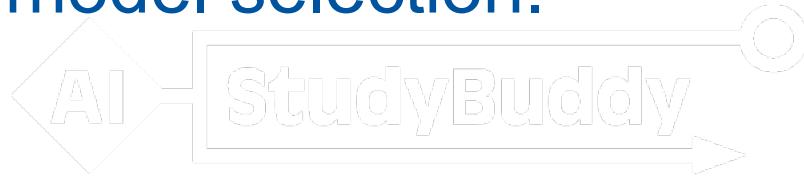


Question:

What is the fundamental problem when using cross-validation (or performance on a validation set) to select between different ML models?

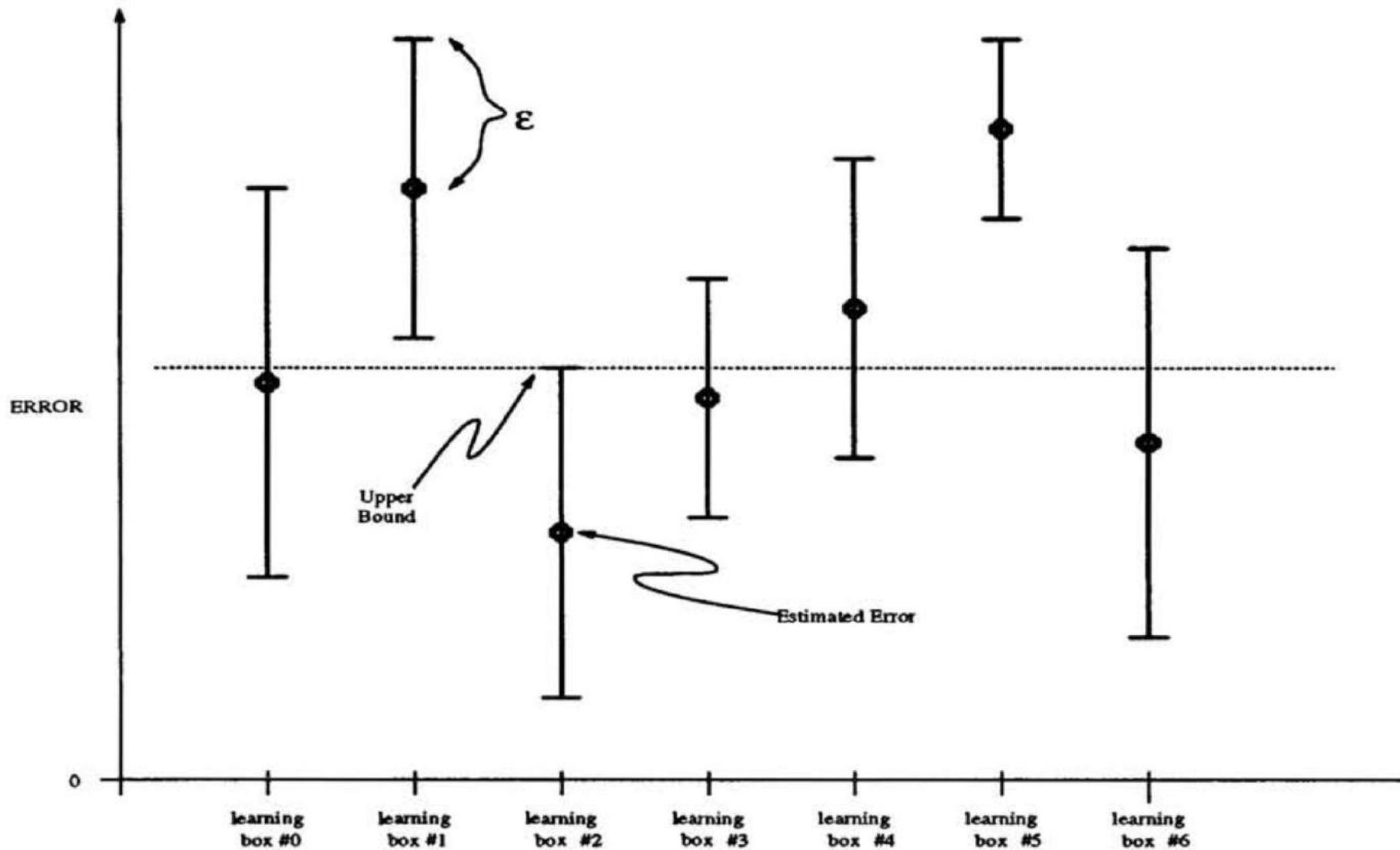
TPS Exercise (T part = done as preparation for class)

You have read the research paper by Maron & Moore about Hoeffding Races for model selection.

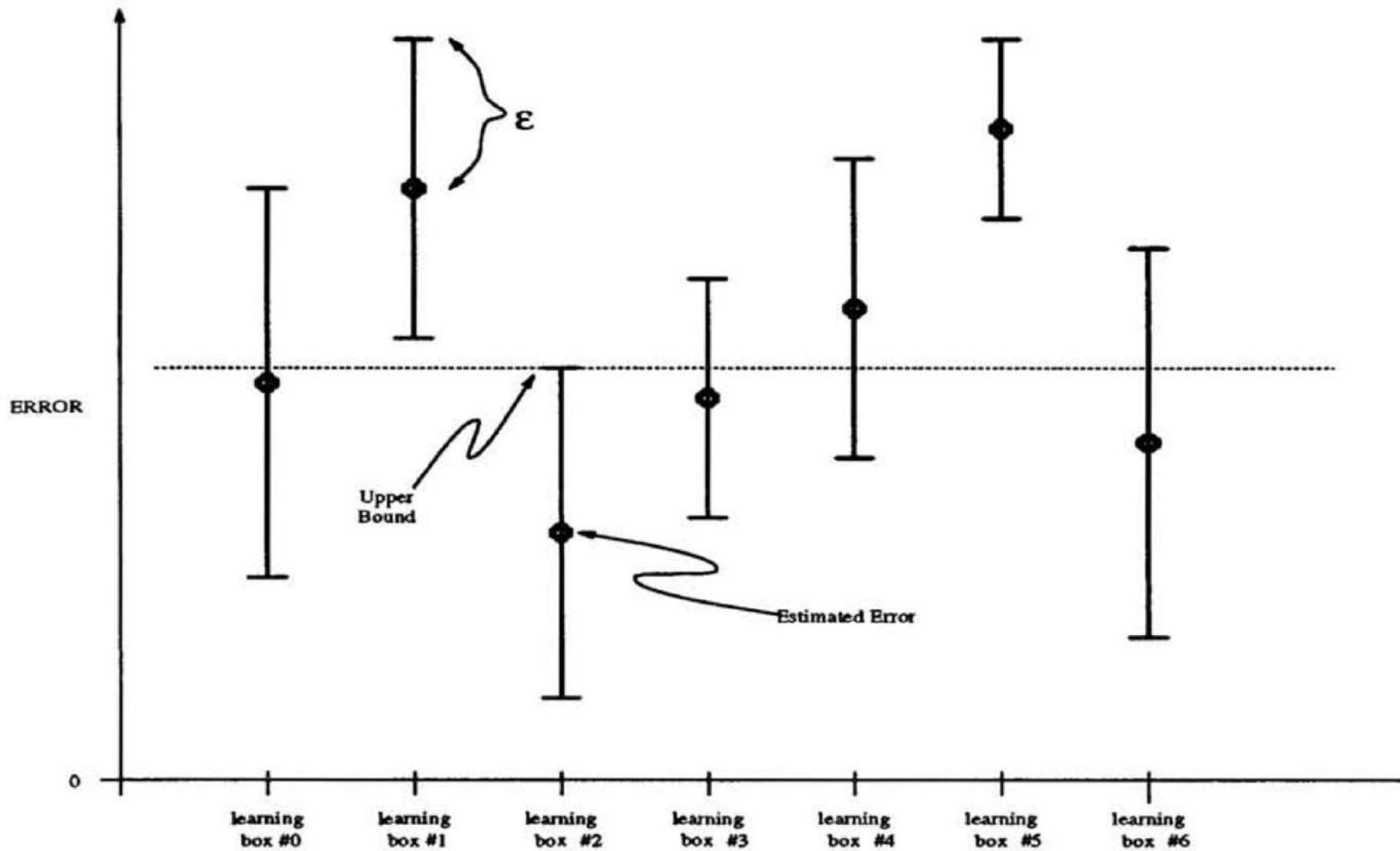


Question:

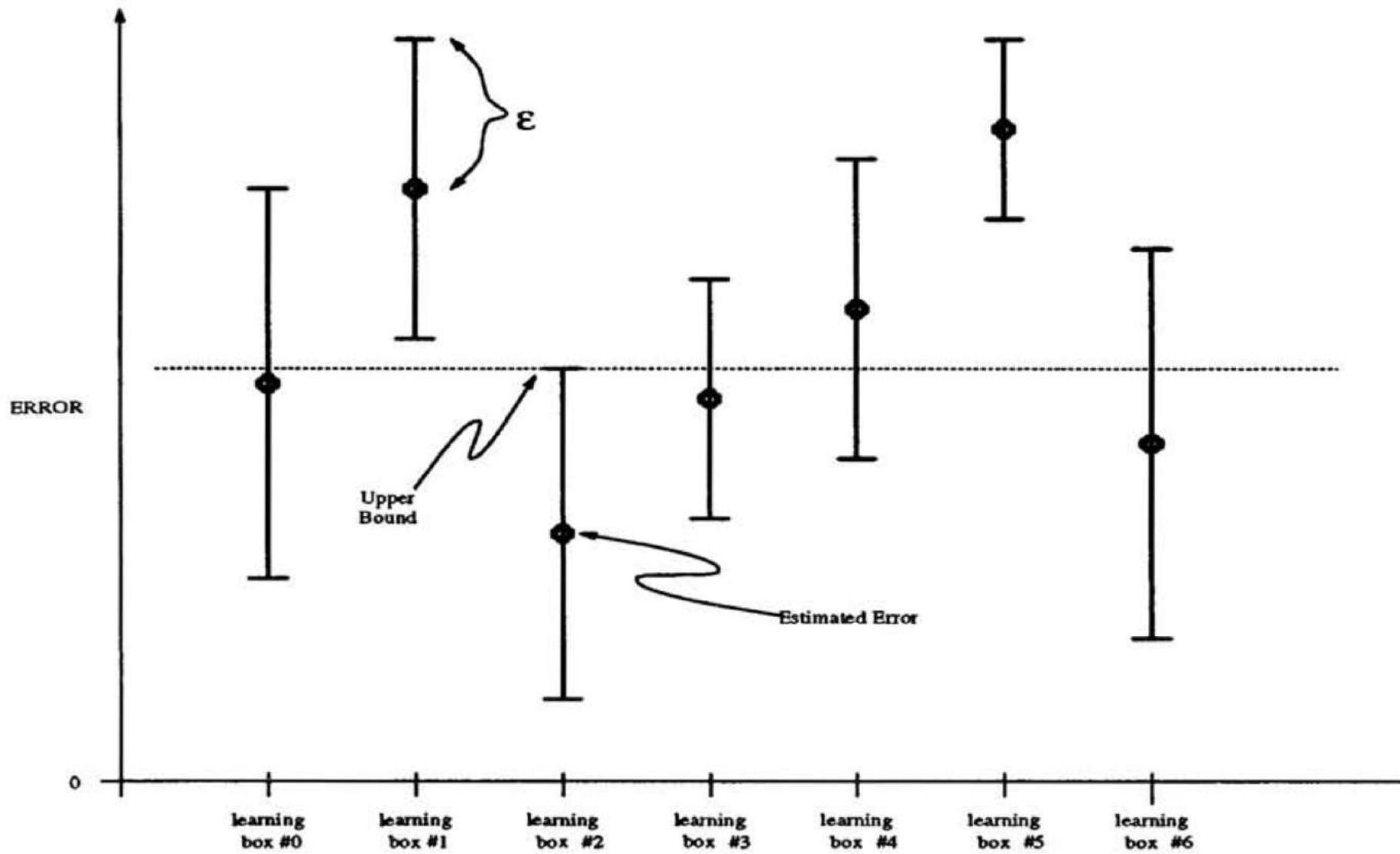
What is the key idea behind Hoeffding Races?



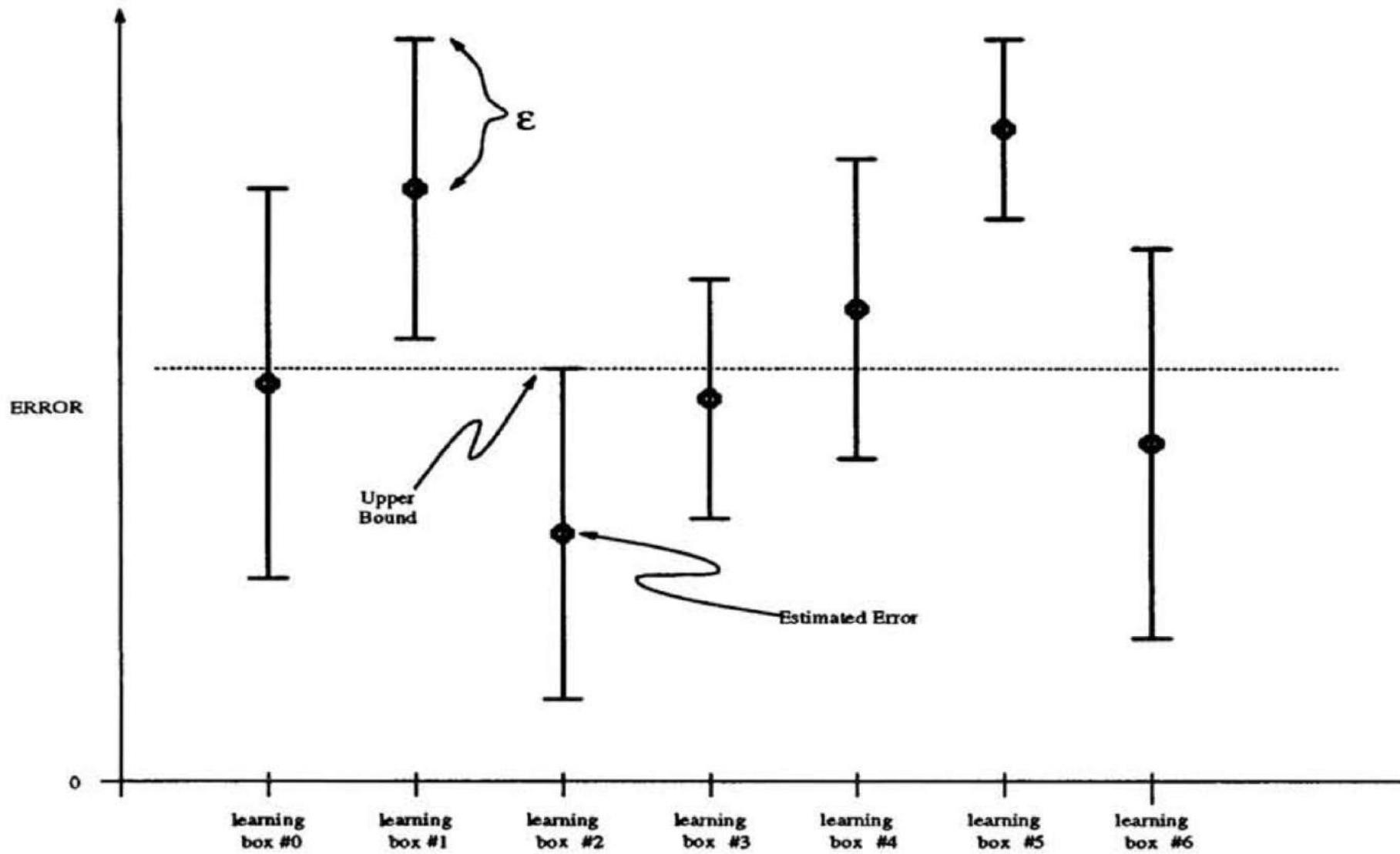
Which models (“learning boxes”) can we safely eliminate?



Which models (“learning boxes”) can we safely eliminate? #1, #5



What happens as more & more inputs are being tested?



What happens as more & more inputs are being tested? Error bars shrink

At each point in the algorithm, we randomly select a point from the test set. We compute the error at that point for all learning boxes, and update each learning box's estimate of its own total error rate. In addition, we use Hoeffding's bound to calculate how close the current estimate is to the true error for each learning box. We then eliminate those learning boxes whose best possible error (their lower bound) is still greater than the worst error of the best learning box (its upper bound); see Figure 2. The intervals get smaller as more points are tested, thereby "racing" the good learning boxes, and eliminating the bad ones.



Why is it important to select (uniformly) at random?

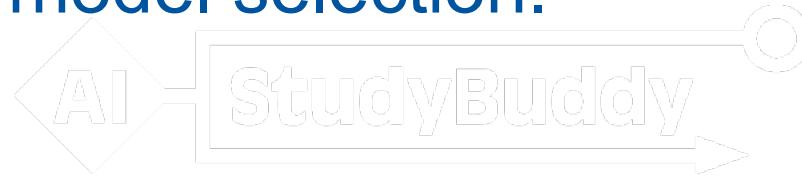
We repeat the algorithm until we are left with just one learning box, or until we run out of points. The algorithm can also be stopped once ϵ has reached a certain threshhold. The algorithm returns a set of learning boxes whose error rates are insignificantly (to within ϵ) different after N test points.



What should we do if we run out of points with more than one model ("learning box") left?

TPS Exercise (T part = done as preparation for class)

You have read the research paper by Maron & Moore about Hoeffding Races for model selection.



Question:

What is the role of the parameters Δ and δ , respectively?

δ : upper bound on $P(|\text{true error} - \text{estimated error}| > \varepsilon)$
for *one* algorithm, during *one stage (iteration)* of the race

i.e., lower $\delta \rightarrow$ lower probability that models get eliminated incorrectly

Δ : overall probability that best model gets eliminated incorrectly

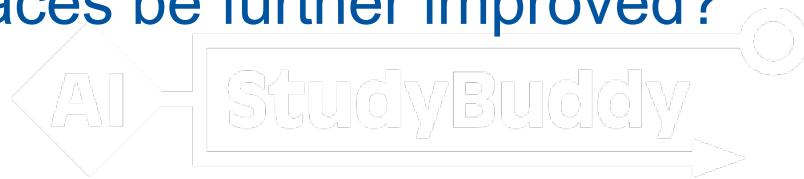
$$\Delta = \delta * \#\text{models} * |\text{test set}|$$

Note: Choose Δ , derive δ from this, then ε (used within HR)

TPS Exercise

Question:

How could Hoeffding Races be further improved?



TPS Exercise

Question:

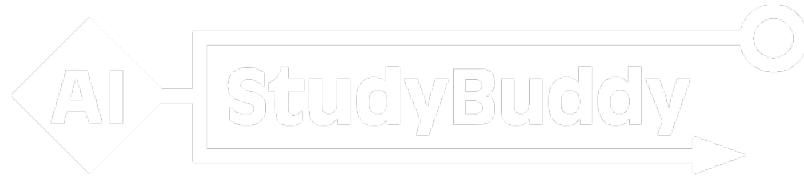
How could Hoeffding Races be further improved?



- Better training data -> better models to start with
- Predict model performance to pre-select models
- Use hyper-parameter optimisation to improve models

Key concepts covered today:

- AutoML
- model selection
- brute-force search
- Hoeffding Races



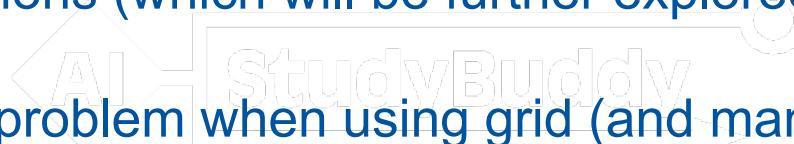
Preparation for class on Tuesday, 23 January 2024 (mandatory):

Read the following research paper:

James Bergstra and Yoshua Bengio: Random Search for Hyper-Parameter Optimization. Journal of Machine Learning Research (JAIR): 281-305, 2012.

(The paper is available online at <https://www.jmlr.org/papers/volume13/bergstra12a/bergstra12a.pdf>)

Focus on the following questions (which will be further explored in TPS exercises in class):

- 
- (1) What is the fundamental problem when using grid (and manual) search for hyper-parameter optimization? Explain what a response surface is in simple terms.
 - (2) What is low effective dimensionality and how does it allow for success of random search to address the problem identified in (1)?
 - (3) What is the key strength of the sequential combination of manual and grid search?

Bring your answers to these questions (which can be in the form of bullet points) to class; they will be the basis for TPS exercises).

NB: Full understanding of the theoretical sections (e.g., 2.1) is desirable but not essential.

Elements of Machine Learning & Data Science

Winter semester 2023/24

Automated Machine Learning (2)

Anja Jankovic & Holger Hoos

Key questions:

- ...

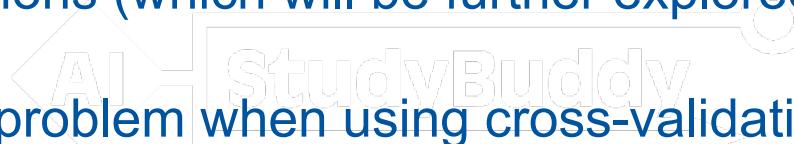


Preparation for today:

Read the following research paper:

Oded Maron and Andrew Moore: Hoeffding Races: Accelerating Model Selection Search for Classification and Function Approximation. Advances in Neural Information Processing Systems 6 (NIPS 1993): 59-66, 1993.
(The paper is available online at <https://proceedings.neurips.cc/>)

Focus on the following questions (which will be further explored in TPS exercises in class):

- 
- (1) What is the fundamental problem when using cross-validation
(or performance on a validation set) to select between different ML models?
 - (2) What is the key idea behind Hoeffding races and how does it address the problem identified in (1)?
 - (3) What is the role of the parameters Δ and δ , respectively?

Bring your answers to these questions (which can be in the form of bullet points) to class;
they will be the basis for TSP exercises).

NB: Full understanding of the proof in Section 3 is desirable but not essential.

TPS Exercise (T part = done as homework)

Question:

**How to assess predictive models for multi-class classification?
(> 2 target classes, e.g., on time, mildly delayed, severely delayed)**

TPS Exercise

You have used supervised ML to train a predictive model for a binary classification problem. The model gives you a numerical **prediction score between 0 and 1.**



Question:

How to assess the quality of the model?

Key concepts covered today:

...

