

# **CS5228: Knowledge Discovery and Data Mining**

## Lecture 1 — Introduction & Overview

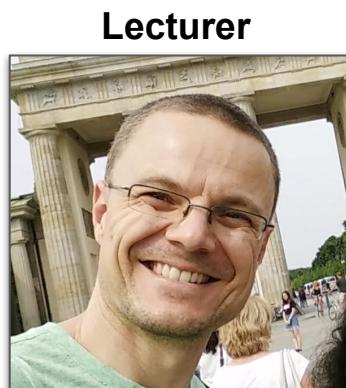
# Outline

- Course Logistics
- Overview
  - What is Knowledge Discovery / Data Mining?
  - Common Data Mining tasks
  - Types of data & data representations
- Data preparation
  - Data quality
  - Exploratory Data Analysis (EDA)
  - Data preprocessing
- Summary

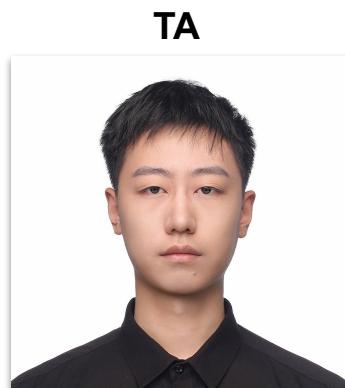
# Course Logistics

- **Lectures & Tutorials**
  - Friday, LT17: 6.30-8.30 pm / 8.30-9.30 pm
  - Physical classes (all recorded)
  - Announcements & materials on Canvas
- **Where to ask questions**
  - Canvas discussion (you are also strongly encouraged to answer questions!)
  - Email to teaching team (for private concerns or sensitive question, e.g., about an assignment)

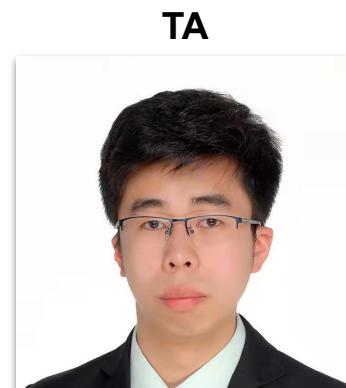
# Teaching Team



Christian von der Weth  
chris@comp.nus.edu.sg



Liu Chenyan  
chenyan@u.nus.edu



Gao Ruize  
e1023891@u.nus.edu



Rajdeep Singh Hundal  
rajdeep@nus.edu.sg



Luo Yang  
yang\_luo@u.nus.edu



Ayush Goyal  
e1124577@u.nus.edu

# Assessments

- **4 assignments (10% each)**
  - Programming tasks + theoretical questions (Python)
  - Discussions allowed, but code and answers must be submitted individually
- **Quiz in the last lecture (10% each)**
  - MCQ/MRQ Canvas quiz
  - Open-book but no Internet (screen recording will be required)
- **Midterm (20%)**
  - MCQ/MRQ + essay questions using Examplify
  - Open-book but blocked Internet
- **Project (30%)**
  - Group project (~4 students per group, more details after enrollment is complete)
  - Kaggle InClass Competition

# Lesson Plan (tentative deadlines; check Canvas!)

Release dates for assignments (Fri);  
submission deadline 2 weeks later (Thu)

Week	Date	Topics	Important Dates
1	Aug 16	Introduction	
2	Aug 23	Clustering I	
3	Aug 30	Clustering II	A1
4	Sep 06	Association Rules	
5	Sep 13	Regression & Classification I	A2
6	Sep 20	Regression & Classification II	
Recess	Sep 27	No class	
7	Oct 04	<b>Midterm Exam</b>	Midterm (Weeks 1-6)
8	Oct 11	Regression & Classification III	A3
9	Oct 18	Recommender Systems	
10	Oct 25	Graph Mining	A4
11	Nov 01	Dimensionality Reduction (recording, WBD)	
12	Nov 08	Data Stream Mining	
13	Nov 15	Review & Outlook + <b>Quiz</b>	Quiz 2 (Weeks 7-12)

# Course Policies

- Zero-Tolerance for Plagiarism
  - Students will be reported to University for disciplinary action for plagiarism/cheating offence
  - Offenders will receive F grade for the module (for any assessment with 10%+ weight!!!)
  - Assignments: discussion allowed but each student must submit their individual solutions
- Resources
  - <https://www.comp.nus.edu.sg/cug/plagiarism/>

# Course Policies

- AI use in class
  - Generally allowed for ideation, brainstorming, self-learning, improve writing
  - Take-home assignments: AI tools permitted but need to be acknowledged
  - Exams (midterm, quiz): AI tools not permitted incl. locally installed tools (e.g., open LLMs)
- Resources
  - <https://libguides.nus.edu.sg/new2nus/acadintegrity>  
(see the "Guidelines on the Use of AI Tools For Academic Work" tab)
  - <https://myportal.nus.edu.sg/studentportal/student-discipline/all/docs/NUS-Plagiarism-Policy.pdf>

# Course Policies

- Right Infringements on NUS Course Materials

*All course participants (including permitted guest students) who have access to the course materials on LumiNUS or any approved platforms by NUS for delivery of NUS modules are not allowed to re-distribute the contents in any forms to third parties without the explicit consent from the module instructors or authorized NUS officials.*

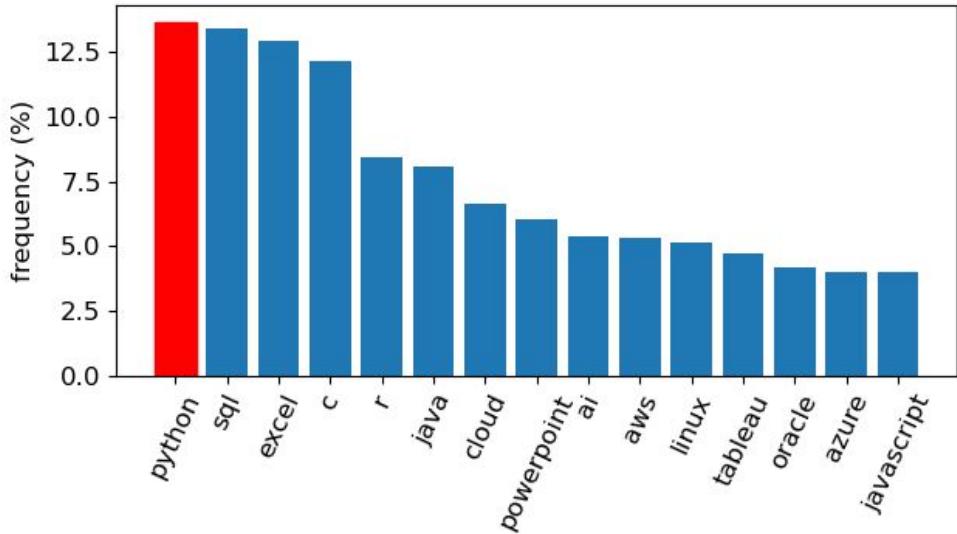
# What You Need

- Programming environment: Python + Jupyter
  - All implementation tasks will be in Python
  - Assignments will include Jupyter notebooks
  - Supplementary [Jupyter notebooks](#) for hands-on practice
- Common packages for data science
  - NumPy
  - pandas
  - NetworkX
  - scikit-learn



# Why Python?

- **Analysis of job descriptions**
  - 15k+ job offers from JobStreet  
(data analyst, data engineer, data scientist)
  - Quick-&-dirty keyword extraction
  - ...but check for yourself! :)



# Learning Outcomes

- Fundamental knowledge about concepts & algorithms in data mining
  - Nature of data: data representations, data and attribute types
  - Common data mining tasks and important algorithms (with their strengths and weaknesses)
  - Problems, risks & ethical issues of "unrestrained" data mining
- Perform data mining tasks for new applications in practice
  - Given a dataset and task, select appropriate techniques to solve the task
  - Justify design and implementation decisions
  - Interpret results and assess limitations

# References

- **Textbooks** (useful but not required)
  - J. Leskovec, A. Rajaraman, J. Ullman: *Mining Massive Datasets*  
(online version available at: <http://www.mmmds.org/>)
  - P. Tan, M. Steinbach, A. Karpatne, V. Kumar: *Introduction to Data Mining*
  - More in Canvas Readings
- ...the Web

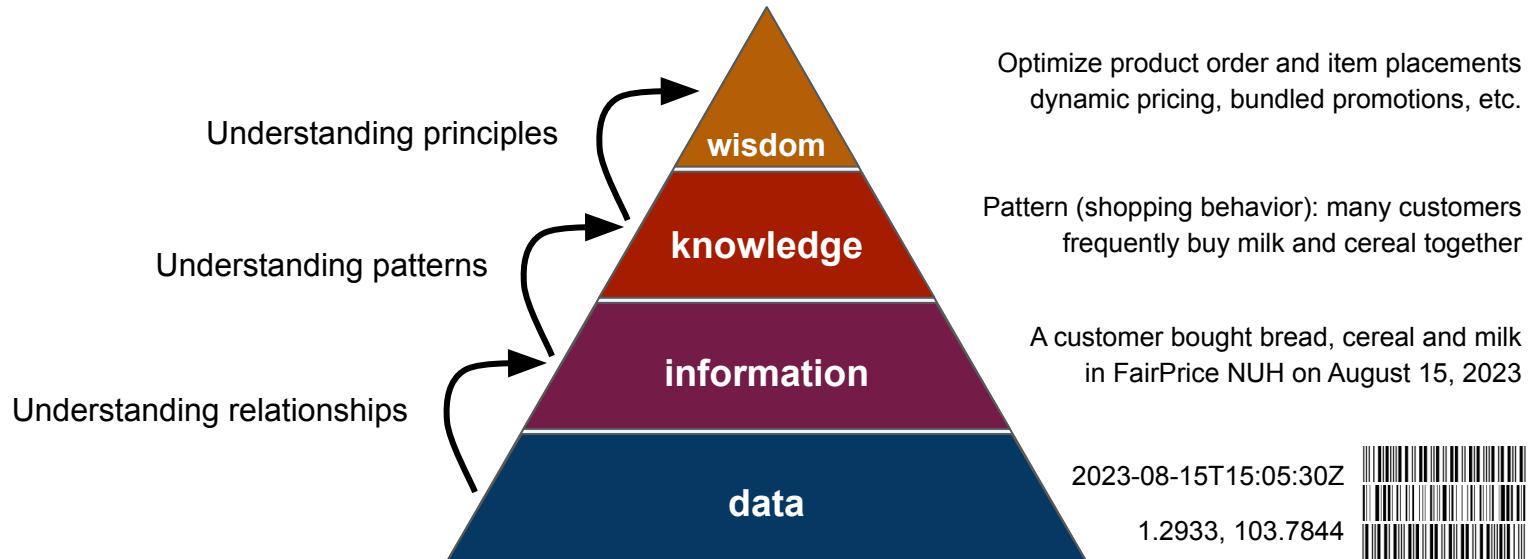
# Outline

- Course Logistics
- **Overview**
  - What is Knowledge Discovery / Data Mining?
  - Common Data Mining tasks
  - Types of data & data representations
- Data preparation
  - Data quality
  - Exploratory Data Analysis (EDA)
  - Data preprocessing
- Summary

# What is Knowledge Discovery & Data Mining

*"The **non-trivial** extraction of **implicit**, previously **unknown**, and potentially **useful** information from data."*

(Frawley, Piatetsky-Shapiro, Matheus; 1991)



# From Data to Knowledge

## Data Selection

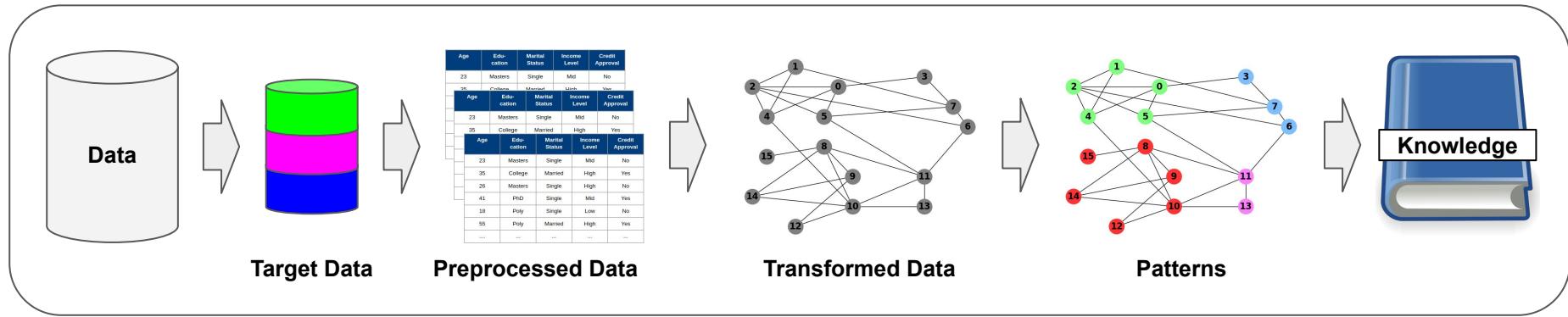
- Identify relevant data to solve a given task

## Data Transformation

- Convert data into suitable representation

## Postprocessing

- Visualization
- Interpretation
- Understanding
- ...



## Data Preprocessing

- Handling missing data
- Duplicate elimination
- Feature selection
- Normalization
- ...

## Data Mining

- Clustering
- Classification
- Regression
- Associations
- Correlations
- ...

# What is NOT Knowledge Discovery & Data Mining?

- Trivial extraction of information/patterns from data
  - Looking up a phone number in phone directory
  - Dividing students based on their degree course
  - Calculating the total sales of a company
- Data analysis not yielding patterns (i.e., new information)
  - Monitoring a patient's heart rate for abnormalities
  - Querying a Web search engine

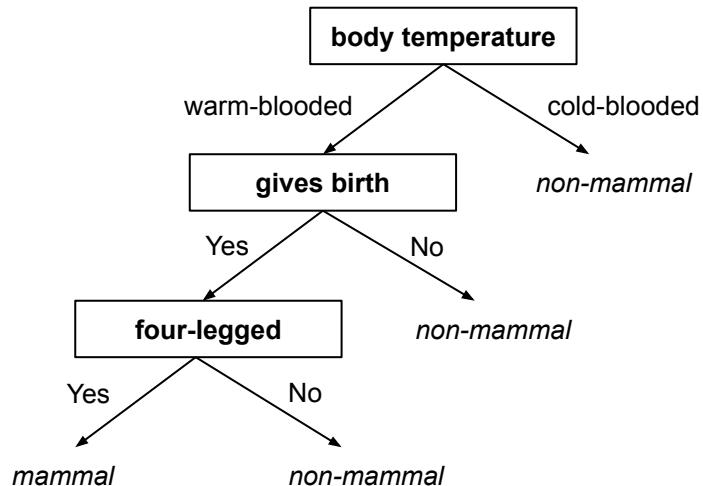
# What Makes a Pattern Useful or Meaningful?

*"If you torture the data long enough, it will confess to anything"*

(Ronald Coase; 1981 — slightly paraphrased)

- Main goal: **Generalizability**
  - Patterns should remain accurate over unseen data
  - Common causes: small and/or biased data

*But what about humans and platypuses, etc.?*

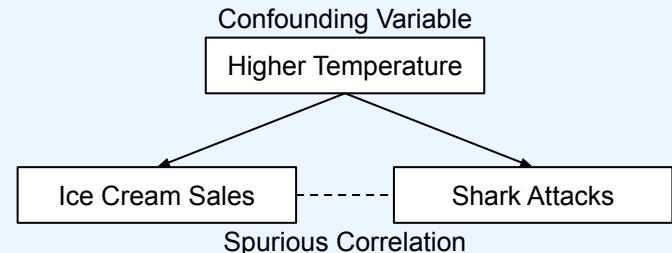


# There is Always Some Pattern in Your Data (even in random data)

- Bizarre and Surprising Insights

- "Female-named hurricanes kill more people than male hurricanes."
- "Users of Chrome and Firefox browsers make better employees."
- "Shark attacks increase when ice cream sales increase"
- "Music taste predicts political affiliation."
- "A job promotion can lead to quitting."
- "Vegetarians miss fewer flights."
- "Smart people like curly fries."
- "Higher status, less polite."

**Important:** Patterns indicate correlations, but correlation does not imply causation!



# Spotting "Shady" Patterns — Reality Check

- What is the (perceived) difference between the 2 statements below?
  - In the context of identifying and/or assessing patterns

*"The higher the sales of ice cream,  
the higher the number of shark attacks."*

vs.

*"The higher the concentration of  
anti-mullerian hormone, the lower the  
concentration of follicle-stimulating hormone."*

# Data Mining Gone Wrong

*"Your scientists were so preoccupied with whether they could,  
they didn't stop to think if they should."*

(Ian Malcolm; Jurassic Park, 1991)

Malaysian Bar troubled over judges using AI for sentencing

Applicant pre-selection by algorithm:  
How to convince an AI of yourself

Algorithms are deciding who gets the  
first vaccines. Should we trust them?

How Target Figured Out A Teen Girl  
Was Pregnant Before Her Father Did

Millions of black people affected by  
racial bias in health-care algorithms

The computers rejecting your job application

# Quick Quiz

What is (arguably) **NOT**  
a "proper" Data Mining task?  
(given a dataset of supermarket transactions)

**A**

Finding the largest sets of products  
most frequently bought together

**B**

Finding groups of similar users  
based on the buying behavior

**C**

Finding all purchases of a bundled  
promotion (i.e., multiple items)

**D**

Finding the products most frequently  
bought on weekends after 6pm

# Outline

- Course Logistics
- Overview
  - What is Knowledge Discovery / Data Mining?
  - **Common Data Mining tasks**
  - Types of data & data representations
- Data preparation
  - Data quality
  - Exploratory Data Analysis (EDA)
  - Data preprocessing
- Summary

# Methods — Association Rules

- Input: transactional data
  - Transaction: data record with set of items
  - Set of items are from a fixed collection
- Pattern: Association rules
  - Rules predicting the occurrence of items based on the occurrence of other items

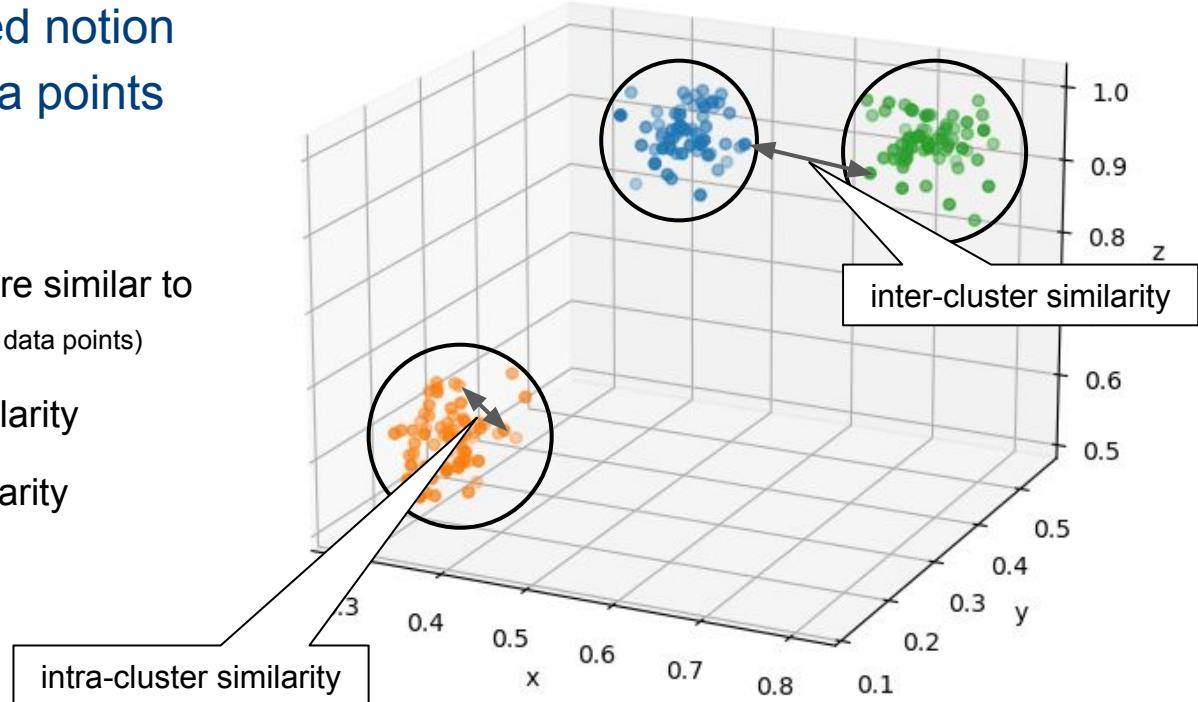
ID	Items
1	covid-19, anosmia, cough, fatigue
2	flu, anosmia, headache
3	covid-19, anosmia, headache, fatigue, fever
4	covid-19, flu, anosmia, fatigue
5	flu, depression, fatigue, fever, headache
...	



{anosmia, fatigue} → {covid-19}

# Methods — Clustering

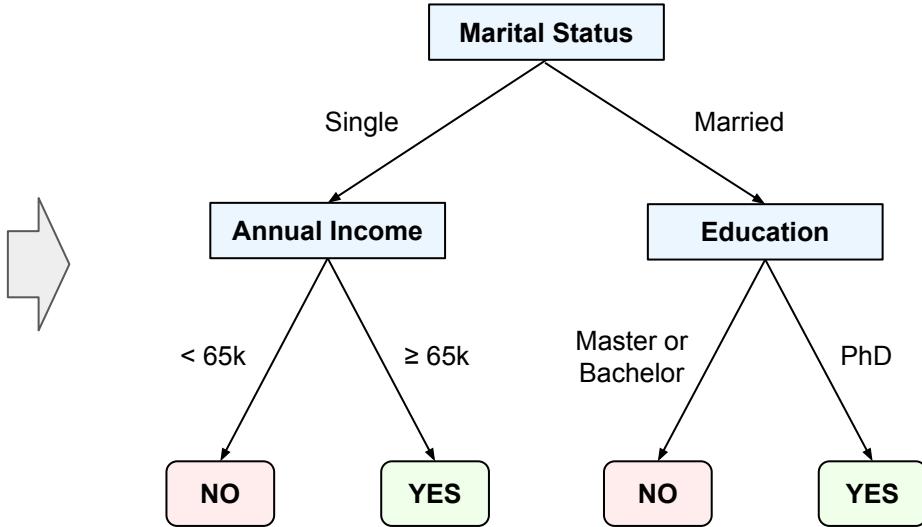
- Input: Data & well-defined notion of similarity between data points
- Pattern: Clusters
  - Groups of data points that are similar to each other (compared to the other data points)
  - Maximize **intra-cluster** similarity
  - Minimize **inter-cluster** similarity



# Methods — Classification

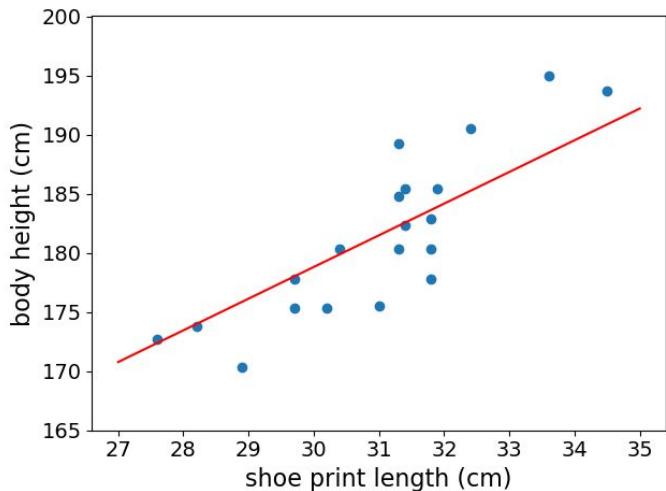
- Input: Dataset with multiple attributes
- Pattern: **Categorical** value of an attribute as function of other attribute values
  - K-Nearest Neighbor, Decision Trees, Linear Classification, etc.

Age	Edu- cation	Marital Status	Annual Income	Credit Default
23	Masters	Single	75k	Yes
35	Bachelor	Married	50k	No
26	Masters	Single	70k	Yes
41	PhD	Single	95k	Yes
18	Bachelor	Single	40k	No
55	Master	Married	85k	No
30	Bachelor	Single	60k	No
35	PhD	Married	60k	Yes
28	PhD	Married	65k	Yes



# Methods — Regression

- Input: Dataset with multiple attributes
- Pattern: **Numerical value of an attribute as function of other attribute values**
  - K-Nearest Neighbor, Regression Trees, Linear Regression, etc.



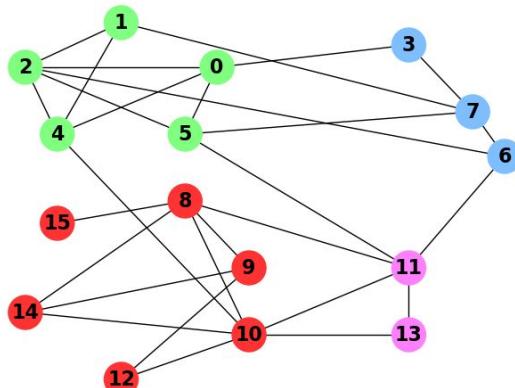
Question: "*What is the expected height of a person that leaves a shoe print of size 32.2cm?*"

Answer: ?

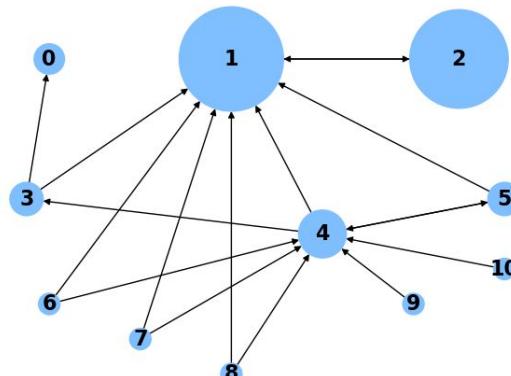
# Methods — Graph Mining

- Input:  $G = (V, E)$ 
  - Set of vertices (or nodes)  $V$  (data points)
  - Set of edges  $E$  (relationship between data points)
- Patterns based on graph structure, e.g.:

Finding communities of nodes



Finding "important" nodes



# Methods — Recommender Systems

- **Input: User-rated items**

(e.g., movies rated by viewers)

- How would Bob rate the movie "Heat"?
- Should "Heat" be recommended to Bob?

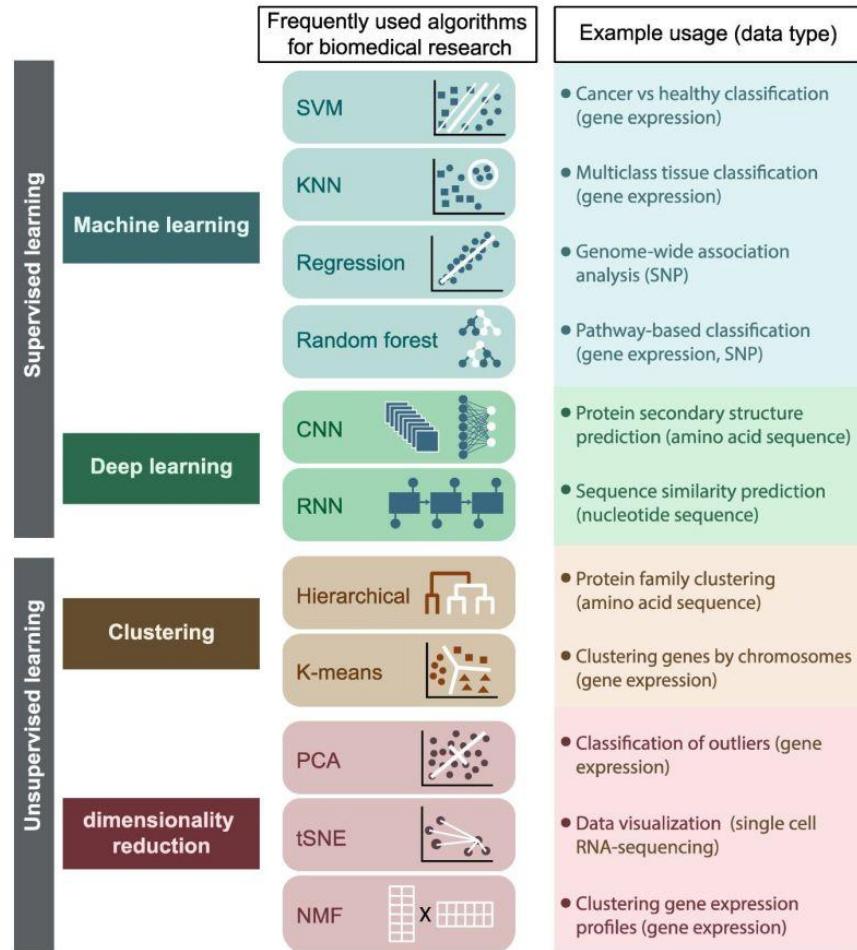
	Clueless	Heat	Jarhead	Big	Rocky
Alice	2	4	5	0	1
Bob	1	???	4	0	2
Claire	1	0	4	3	0
Dave	5	1	2	0	5
Erin	1	5	3	0	3

- **Patterns based on similarities to predict missing values**

- Exploiting features of items
- Exploiting similarities between users or items

# Data Mining in Practice

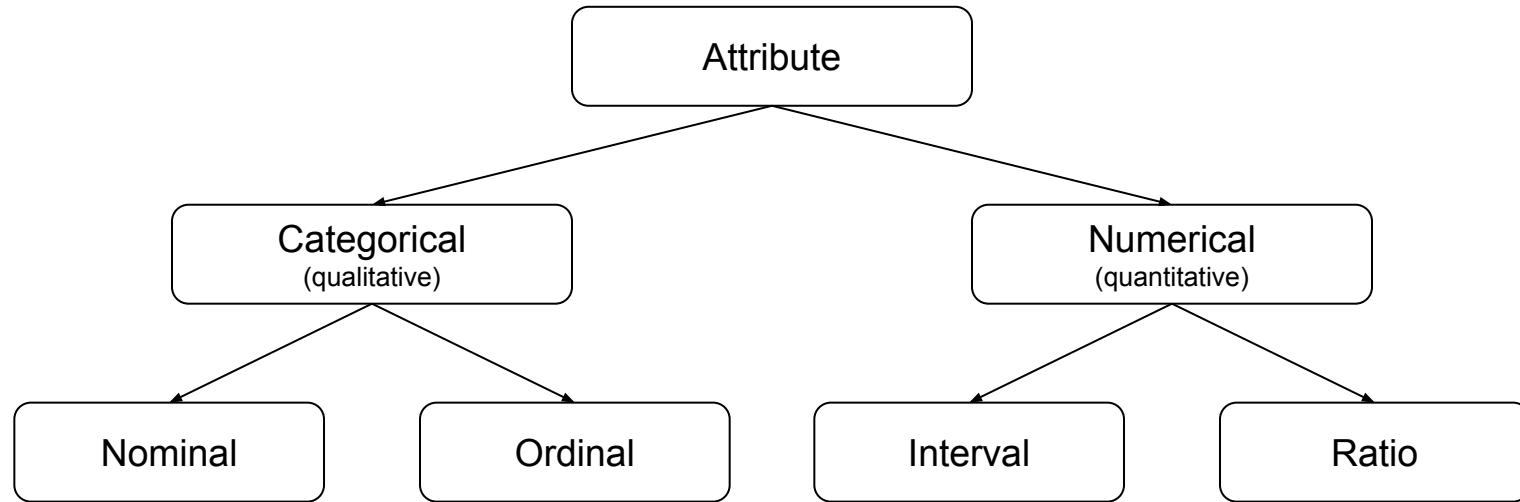
- Example: Biomedical Research
  - Set of important data mining algorithms
  - Relevant for many other fields
  - Many covered here in CS5228  
(main exception: no deep learning)



# Outline

- Course Logistics
- Overview
  - What is Knowledge Discovery / Data Mining?
  - Common Data Mining tasks
  - **Types of data & data representations**
- Data preparation
  - Data quality
  - Exploratory Data Analysis (EDA)
  - Data preprocessing
- Summary

# Types of Attributes



- Values are only labels
- Operations:  
=, ≠
- Examples: sex (m/f), eye color, zip code

- Values are labels with a meaningful order
- Operations:  
=, ≠, <, >
- Examples: street numbers, education level

- Values are measurements with a meaningful distance
- Operations:  
=, ≠, <, >, +, -
- Examples: body temperature in °C, calendar dates

- Values are measurements with a meaningful ratio
- Operations:  
=, ≠, <, >, +, -, \*, /
- Examples: age, weight, income, blood pressure

# Types of Data

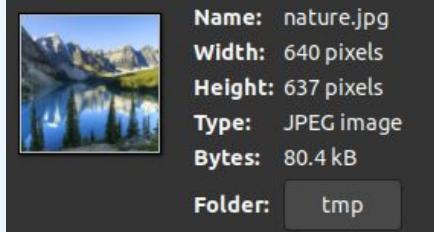
## (Well-)Structured Data

- Highly organized: adheres to predefined data model
- Each object has the same fixed set of attributes
- Easy to search, aggregate, manipulate, analyze data
- Examples: Relational databases, spreadsheets

Age	Education	Marital Status	Income Level	Credit Approval
23	Masters	Single	Mid	No
35	College	Married	High	Yes
26	Masters	Single	High	No
41	PhD	Single	Mid	Yes
18	Poly	Single	Low	No
55	Poly	Married	High	Yes
....	...	...	...	...

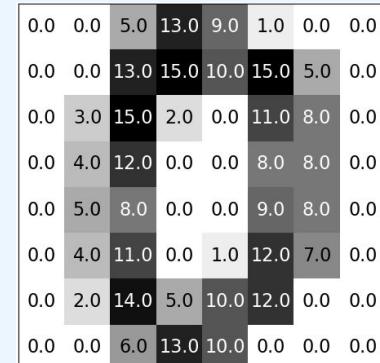
## Semi-Structured Data

- No rigid data model: mix of structured & unstructured data
- Data exchange formats: XML, JSON, CSV
- Tagged unstructured data (e.g., photo + date/time, location, exposure, resolution, flash, etc)



## Unstructured Data

- No fixed data model
- Requires more advanced data analysis techniques
- Examples: images, videos, audio, text, social media



# Types of Data Representations — Record Data

**Data matrix:** collection records; each record consisting of a fixed set of attributes

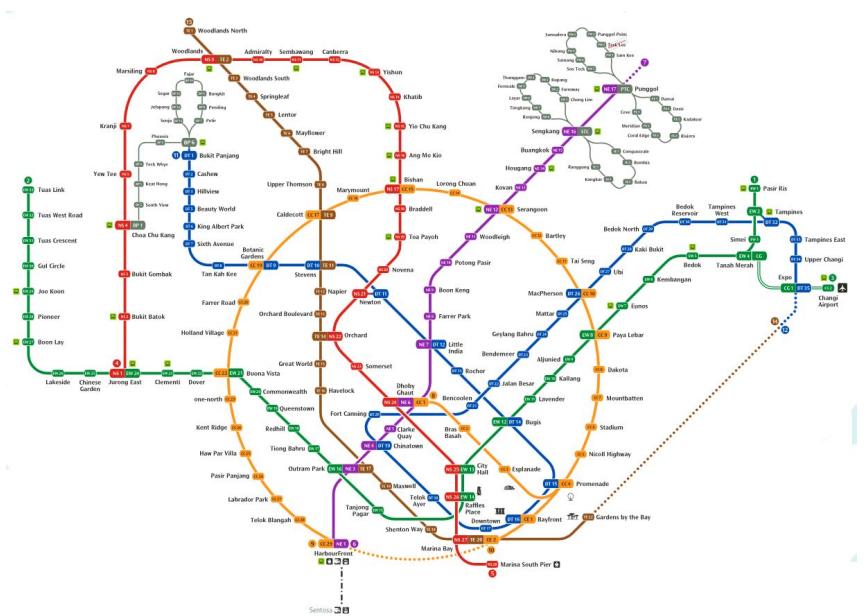
Age	Edu- cation	Marital Status	Annual Income	Credit Approval
23	Masters	Single	75k	Yes
35	Bachelor	Married	50k	No
26	Masters	Single	70k	Yes
41	PhD	Single	95k	Yes
18	Bachelor	Single	40k	No
55	Master	Married	85k	No
30	Bachelor	Single	60k	No
35	PhD	Married	60k	Yes
28	PhD	Married	65k	Yes

**Transaction data:** collection records; each record involves a set of items

ID	Items
1	covid-19, anosmia, cough, fatigue
2	flu, anosmia, headache
3	covid-19, anosmia, headache, fatigue, fever
4	covid-19, flu, anosmia, fatigue
5	flu, depression, fatigue, fever, headache
...	

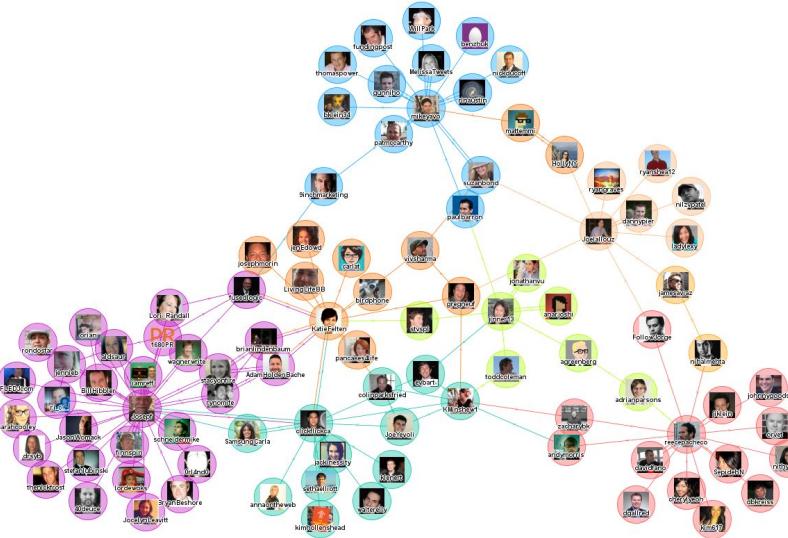
# Types of Data Representations — Graph Data

Example: traffic data



Source: <https://www.lta.gov.sg/>

Example: social network data



Source: <http://touchgraph.com/>

# Types of Data Representations — Ordered Data

Example: stock prices (sequence of data points)



# Quick Quiz

What type of attribute is  
**Annual Income?**

ID	Age	Edu- cation	Marital Status	Annual Income	Credit Approval
101	23	Masters	Single	75k	Yes
102	35	Bachelor	Married	50k	No
103	26	Masters	Single	70k	Yes
104	41	PhD	Single	95k	Yes
105	18	Bachelor	Single	40k	No
...	...	...	...	...	...

A

Nominal

B

Ordinal

C

Interval

D

Ratio

# Quick Quiz

What type of attribute is  
**Education?**

ID	Age	Edu- cation	Marital Status	Annual Income	Credit Approval
101	23	Masters	Single	75k	Yes
102	35	Bachelor	Married	50k	No
103	26	Masters	Single	70k	Yes
104	41	PhD	Single	95k	Yes
105	18	Bachelor	Single	40k	No
...	...	...	...	...	...

A

Nominal

B

Ordinal

C

Interval

D

Ratio

# Quick Quiz

What type of attribute is  
**ID?**

ID	Age	Edu- cation	Marital Status	Annual Income	Credit Approval
101	23	Masters	Single	75k	Yes
102	35	Bachelor	Married	50k	No
103	26	Masters	Single	70k	Yes
104	41	PhD	Single	95k	Yes
105	18	Bachelor	Single	40k	No
...	...	...	...	...	...

**A**

Nominal

**B**

Ordinal

**C**

Interval

**D**

Ratio

# Outline

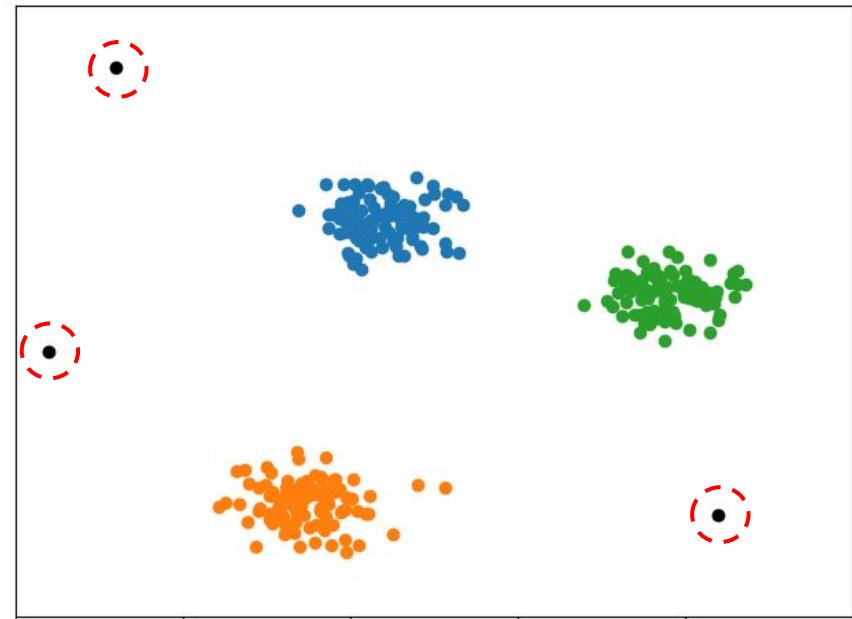
- Course Logistics
- Overview
  - What is Knowledge Discovery / Data Mining?
  - Common Data Mining tasks
  - Types of data & data representations
- Data preparation
  - **Data quality**
  - Exploratory Data Analysis (EDA)
  - Data preprocessing
- Summary

# Data Quality — Noise

- Data = true signal + noise
  - Sensor readings from faulty devices  
(also intrinsic noise or external influences)
  - Errors during data entry  
(by humans or machines)
  - Errors during data transmission
  - Inconsistencies in data formats  
(e.g., iso time vs unix time, DD/MM/YYYY vs. MM/DD/YYYY)
  - Inconsistencies in conventions  
(e.g., meters vs. miles, meters vs. centimeters)

# Data Quality — Outliers

- Outlier: Data point with attribute values considerably different from other points
- Case 1: Outliers are noise
  - Negatively interfere with data analysis
  - (Try to) remove outliers and/or use methods less prone to outliers
- Case 2: Outliers are targets
  - (the goal is to find rare/strange/odd data points)
  - Credit card fraud
  - Intrusion detection



# Data Quality — Missing Values

- Common causes

- Attribute values not collected  
(e.g., broken sensor, person refused to report age)
- Attributes not applicable in all cases  
(e.g., no income information for children)

- Handling missing values

- Remove data points with missing values
- Remove attributes with missing values  
(not all attributes are always equally important)
- (Try to) fill in missing values  
(e.g., average temperature readings of nearby sensors)

Age	Edu- cation	Marital Status	Annual Income	Credit Default
23	Masters	Single	75k	Yes
N/A	Bachelor	Married	N/A	No
26	Masters	Single	70k	Yes
41	PhD	Single	95k	Yes
18	Bachelor	Single	40k	No
55	Master	Married	N/A	No
30	Bachelor	Single	N/A	No
35	PhD	Married	60k	Yes
N/A	PhD	Married	65k	Yes

# Data Quality — Duplicates

- **Duplicates:** Data points referring to the same object/entity

(e.g., two records in a database refer to the same real-world person)

- Exact duplicates: data points have the same attribute values
- Near duplicates: data points (slightly) differ in their attribute values  
(e.g., same person with the same phone number but in different formats)

- **Task: Duplicate Elimination**

- Relatively easy for exact duplicates
- Generally very difficult for near duplicates

**Note:** Duplicates are a major issue when merging data from multiple heterogeneous sources. Due to its complexity, duplicate elimination is beyond the scope of this lecture

# Outline

- Course Logistics
- Overview
  - What is Knowledge Discovery / Data Mining?
  - Common Data Mining tasks
  - Types of data & data representations
- Data preparation
  - Data quality
  - **Exploratory Data Analysis (EDA)**
  - Data preprocessing
- Summary

# Exploratory Data Analysis (EDA)

- EDA — getting to know your data (through basic transformation and visualization)

- Assess data quality
- Basic sanity checks
- Get first insights into data
- Formulate new questions



No formal process with strict rules!

## Running example:

Cardiovascular Disease Dataset  
(modified to make some points)

	<b>id</b>	<b>age</b>	<b>gender</b>	<b>height</b>	<b>weight</b>	<b>ap_hi</b>	<b>ap_lo</b>	<b>cholesterol</b>	<b>gluc</b>	<b>smoke</b>	<b>alco</b>	<b>active</b>	<b>cardio</b>
0	0	18393	2	168	62.0	110	80	1	1	0	0	1	0
1	1	20228	1	156	85.0	140	90	3	1	0	0	1	1
2	2	18857	1	165	64.0	130	70	3	1	0	0	0	1
3	3	17623	2	169	82.0	150	100	1	1	0	0	1	1
4	4	17474	1	156	56.0	100	60	1	1	0	0	0	0

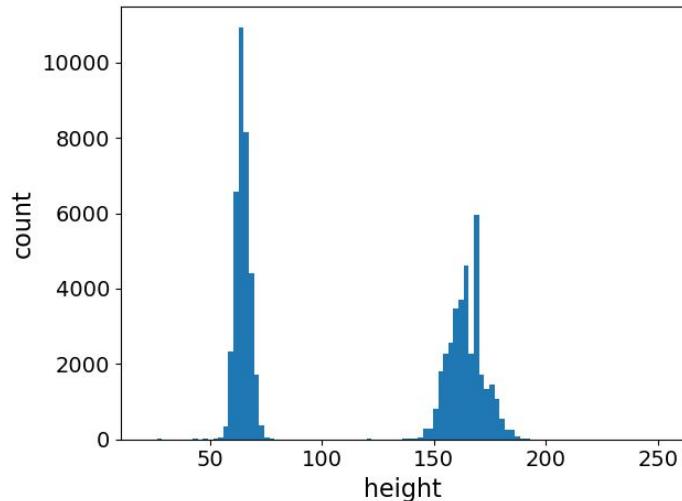
Source: [Cardiovascular Disease dataset](#)

# EDA — Identifying Noise

- Using histograms to inspect distribution of data values

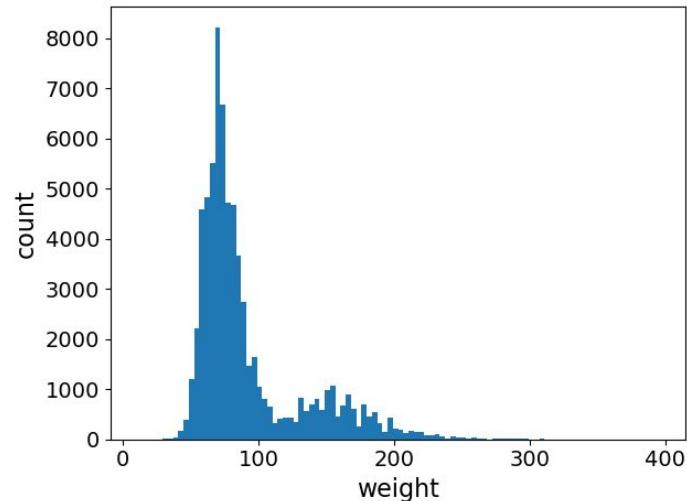
Noise in the height values

- 50% measured in inches
- 50% measured in centimeters



Noise in the weight values

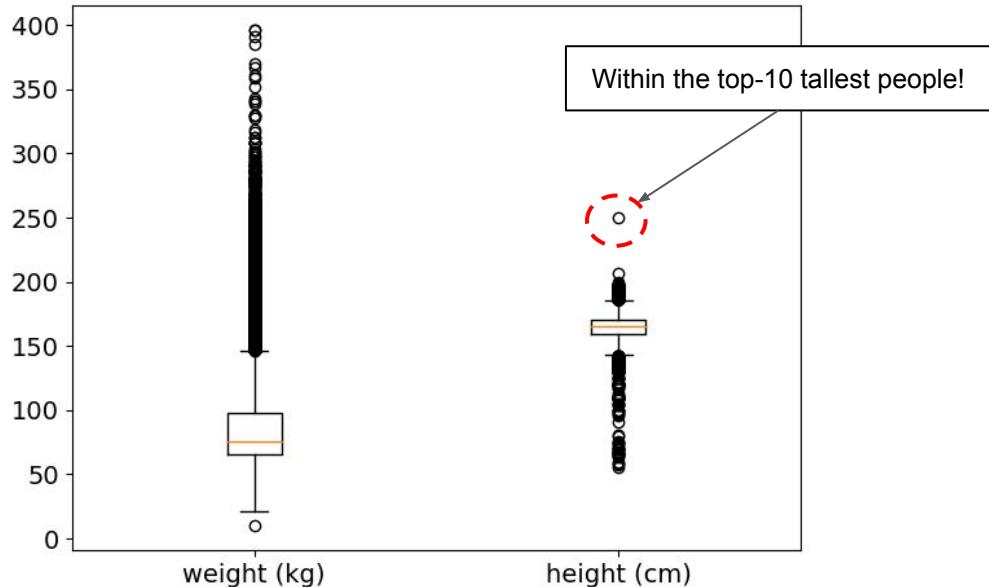
- 80% measured in kilograms
- 20% measured in pounds



# EDA — Identifying Noise / Outliers

- Box plots to inspect distribution of attribute values
  - Make outliers explicit

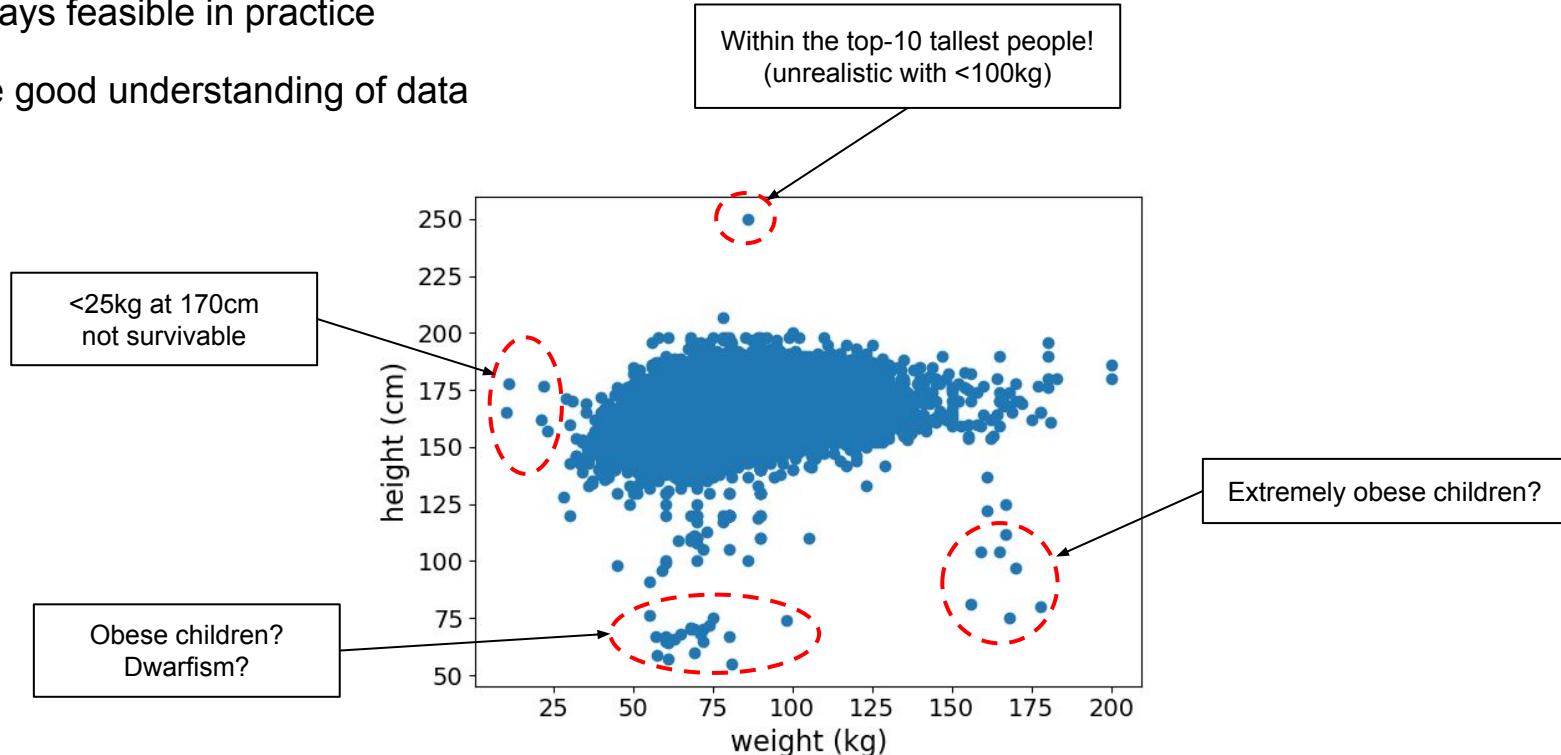
**Note:** Not all outliers are "bad" or considered noise. For example, a CEO's salary is typically much higher than the one of the average employee. Whether it should be removed depends on the goal of the analysis



# EDA — Identifying Noise / Outliers

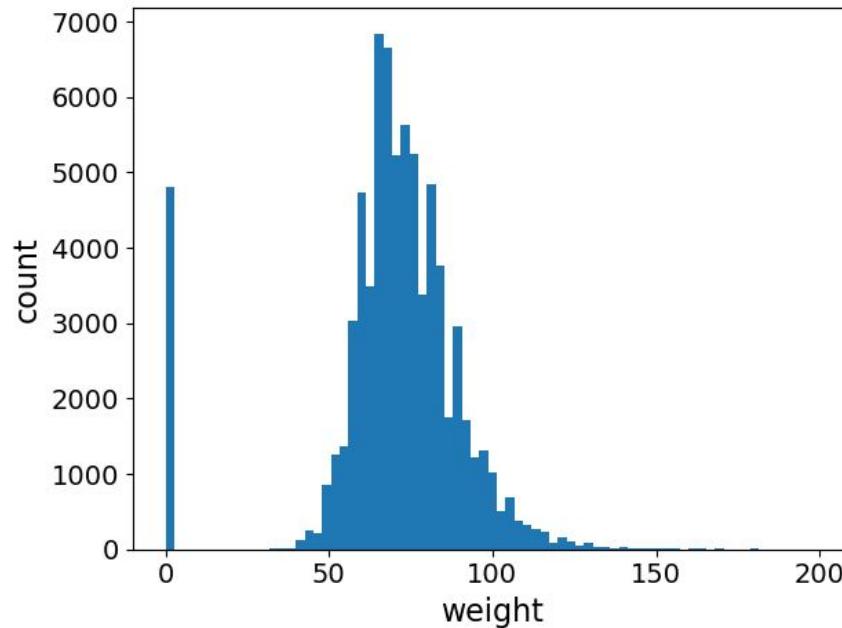
- Using scatter plot to inspect correlations

- Not always feasible in practice
- Require good understanding of data



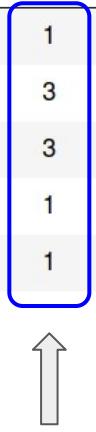
# EDA — Missing Values

- Example: Default value (0) if people did not disclose weight
  - Can already negatively affect simple analysis such as calculating means/averages



# EDA — Attribute Types

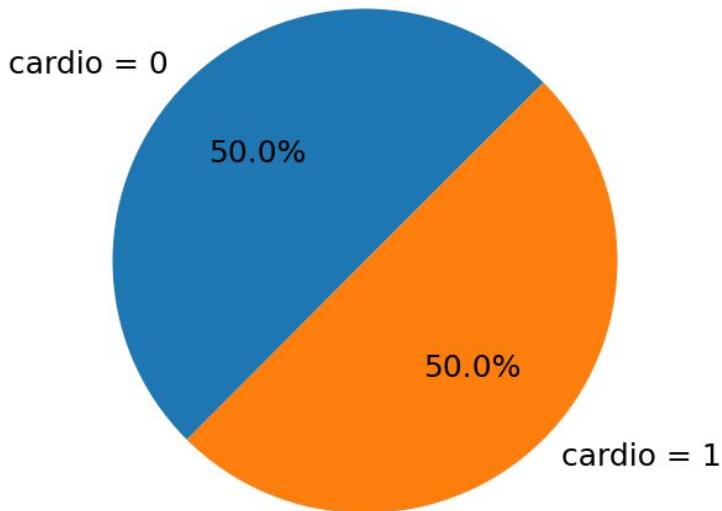
		id	age	gender	height	weight	ap_hi	ap_lo	cholesterol	gluc	smoke	alco	active	cardio
0	0	18393	2	168	62.0	110	80		1	1	0	0	1	0
1	1	20228	1	156	85.0	140	90		3	1	0	0	1	1
2	2	18857	1	165	64.0	130	70		3	1	0	0	0	1
3	3	17623	2	169	82.0	150	100		1	1	0	0	1	1
4	4	17474	1	156	56.0	100	60		1	1	0	0	0	0



- Looks numerical but is categorical (ordinal)  
(1: normal, 2: above normal, 3: well above normal)
- Usually part of the documentation of dataset
- Interpretation requires good understanding of the data  
→ Generally impossible for automated methods

# EDA — Distribution of Class Labels

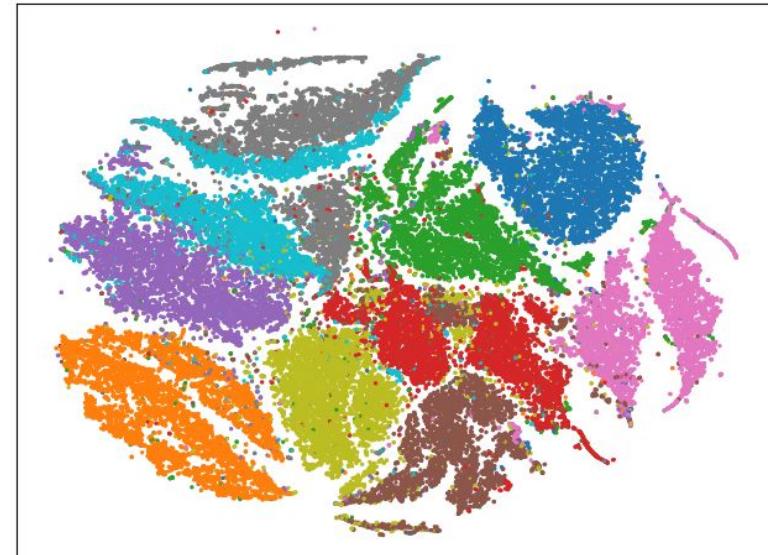
- Classification tasks generally benefit from balanced datasets
  - Balanced = all classes are (almost) equally represented
  - Distribution of classes also affects evaluation of found patterns



# EDA — Visualizing High-Dimensional Data

- Visualization using dimensionality reduction techniques (here: t-SNE)
- MNIST Dataset
  - 60k handwritten digits 0, 1, 2, ..., 9  
(~6k samples for class)
  - 28×28 pixels → 784 features  
(integer grayscale values 0..255)

2 1 0 4 1 4  
9 0 6 9 0 1  
7 3 4 9 6 6



# EDA — Unstructured Data (just some intuitions)

- Plain text
  - Language, (size of) vocabulary
  - Formal vs. informal text (e.g., social media content with slang, emoticons, emojis)
- Images/videos
  - Dimensions and resolutions
  - Color spaces
- Audio
  - Sampling rate and frequency range
  - Types of recording (e.g., voice vs. music)

# Quick Quiz

Which of the statements  
on the right is **True**?

**A**

Outliers are always noise and need  
to be removed before an analysis

**B**

As long as my class labels are  
balanced, I will get good results

**C**

Boxplots are often insufficient to  
identify all outliers in a dataset

**D**

If attribute values show a weird  
distribution, I know something is off

# Outline

- Course Logistics
- Overview
  - What is Knowledge Discovery / Data Mining?
  - Common Data Mining tasks
  - Types of data & data representations
- Data preparation
  - Data quality
  - Exploratory Data Analysis (EDA)
  - **Data preprocessing**
- Summary

# Data Preprocessing

- Main purposes
  - Improve data quality ("*Garbage in, garbage out!*")
  - Generate valid input for data mining algorithms
  - Remove complexity from data to ease analysis
- Core preprocessing task
  - Data cleaning
  - Data reduction
  - Data transformation
  - Data discretization

# Data Cleaning

- Improve data quality

- Remove or fill missing values
- Identify and remove outliers  
(if outliers are not the goal of the analysis)
- Identify and remove/merge duplicates
- Correct errors and inconsistencies  
(e.g., convert inches to centimeters)



Non-trivial tasks and typically  
very application-specific

# Data Reduction

- Reducing the number of data points
  - Sampling — select subset of data points (typically random or stratified sampling)
  - Commonly used for preliminary analysis or when the data size is extremely large
- Reducing the number of attributes
  - Removing irrelevant attributes (e.g., ids or ethically questionable attributes such as religion, sexual orientation, etc.)
  - Dimensionality reduction — mapping the data into a lower-dimensional space (PCA, LDA, t-SNE, etc.)
- Reducing the number of attribute values (form of noise removal)
  - Aggregation or generalization
  - Binning with smoothing

# Reducing the Number of Attribute Values — Examples

## • Aggregation

- Moving up concept hierarchy of numerical attributes (e.g., from days to years)
- Generalization for categorical attributes

	id	age	gender	height
0	0	18393	2	168
1	1	20228	1	156
2	2	18857	1	165
3	3	17623	2	169
4	4	17474	1	156



	id	age	gender	height
0	0	50.0	2	168
1	1	55.0	1	156
2	2	51.0	1	165
3	3	48.0	2	169
4	4	47.0	1	156

## • Binning and smoothing

- Sort by attribute value (e.g., height)
- Split data into bins of equal sizes
- Replace each value with bin mean  
(the means are also rounded in this example)

55 57 59 60 64 65 65 66 67 67 67 68 68 70 70 70 ...

55 57 59 60 64 | 65 65 66 67 67 | 67 68 68 70 70 | 70 ...

59 59 59 59 59 | 66 66 66 66 66 | 69 69 69 69 69 | 72 ...

# Data Transformation

- Some data reduction techniques also transform the data
  - Dimensionality reduction, aggregation/generalization, binning, etc.
- Attribute construction
  - Add or replace attribute inferred from existing attributes
  - Example: weight, volume → density
- Normalization
  - Scaling attribute values to value into a specified range (e.g., [0,1])
  - Standardization: scaling by using mean and standard deviation

# Normalization — Examples

Min-max normalization

$$x_i^{weight} = \frac{x_i^{weight} - \min(x^{weight})}{\max(x^{weight}) - \min(x^{weight})}$$

weight

62.0

85.0

64.0

82.0

56.0

weight

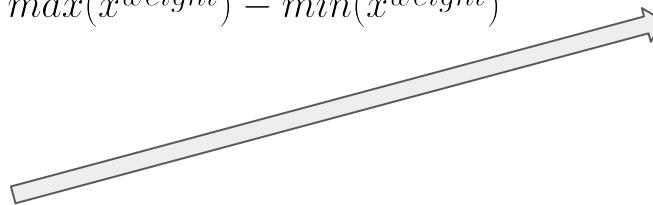
0.273684

0.394737

0.284211

0.378947

0.242105



weight

-0.847867

0.749826

-0.708937

0.541431

-1.264657

$$x_i^{weight} = \frac{x_i^{weight} - \mu^{weight}}{\sigma^{weight}}$$

Standardization  
(z-score normalization)

# Data Discretization

- Converting continuous attributes into ordinal attributes
  - Some algorithms accept only categorical attributes
  - Convert a regression task to a classification task
- Example: Convert weight to a weight category
  - Many existing discretization methods
  - Here: discretization using 3 user-defined bins

	<b>id</b>	<b>age</b>	<b>gender</b>	<b>weight</b>
0	66283	14461	1	86.0
1	4780	14740	1	57.0
2	34457	21090	1	88.0
3	83116	15869	2	60.0
4	70356	20687	1	103.0



	<b>id</b>	<b>age</b>	<b>gender</b>	<b>weight</b>	<b>weight_bin</b>	<b>weight_class</b>
0	66283	14461	1	86.0	(70, 90]	average
1	4780	14740	1	57.0	(0, 70]	light
2	34457	21090	1	88.0	(70, 90]	average
3	83116	15869	2	60.0	(0, 70]	light
4	70356	20687	1	103.0	(90, 150]	heavy

# One-Hot Encoding

- Converting categorical attributes into numerical attributes
  - Converting categorical attributes into a series of binary attributes 0/1
  - Allows the application of any methods for numerical features on categorical attributes
- Example

	<b>id</b>	<b>weight_class</b>
0	66598	light
1	88878	average
2	80363	heavy
3	52546	light
4	17715	average



	<b>id</b>	<b>weight_class_light</b>	<b>weight_class_average</b>	<b>weight_class_heavy</b>
0	66598	1	0	0
1	88878	0	1	0
2	80363	0	0	1
3	52546	1	0	0
4	17715	0	1	0

# Quick Quiz

Which attributes are generally(!)  
not relevant for the analysis and  
**SHOULD be removed?**

ID	Age	Education	Marital Status	Annual Income	Email	Credit Default
101	23	Masters	Single	75k	alice@...	Yes
102	35	Bachelor	Married	50k	bob@...	No
103	26	Masters	Single	70k	claire@...	Yes
104	41	PhD	Single	95k	dave@...	Yes
105	18	Bachelor	Single	40k	erin@...	No
106	24	Masters	Single	65k	fred@...	Yes

A

ID + Email

B

Age + Email

C

ID + Education

D

ID + Marital Status

# Quick Quiz — Side Note



ELSEVIER

Available online at [www.sciencedirect.com](http://www.sciencedirect.com)



ScienceDirect

Journal of Research in Personality 42 (2008) 1116–1122

---

JOURNAL OF  
RESEARCH IN  
PERSONALITY

---

[www.elsevier.com/locate/jrp](http://www.elsevier.com/locate/jrp)

Brief Report

## How extraverted is honey.bunny77@hotmail.de? Inferring personality from e-mail addresses

Mitja D. Back \*, Stefan C. Schmukle, Boris Egloff

*Department of Psychology, University of Leipzig, Seeburgstr. 14-20, 04103 Leipzig, Germany*

Available online 29 February 2008

# Quick Quiz

Which attributes are arguably not relevant or "problematic" and **SHOULD be removed?**

Age	Religion	Education	Has Account	Annual Income	Zodiac Sign	Credit Approval
23	Buddhist	Masters	Yes	75k	Leo	Yes
35	Buddhist	Bachelor	Yes	50k	Gemini	No
26	Muslim	Masters	Yes	70k	Libra	Yes
41	Christian	PhD	Yes	95k	Leo	Yes
18	Buddhist	Bachelor	Yes	40k	Virgo	No
24	Muslim	Masters	Yes	65k	Aries	Yes

A

Religion + Education + Zodiac Sign

B

Religion + Zodiac Sign + Has Account

C

Religion + Zodiac Sign

D

Has Account + Zodiac Sign

# Quick Quiz — Side Note

CNA Insider

## Does a job seeker's horoscope matter? For some companies, the answer is yes

There are companies that turn to unconventional methods like astrology, tarot reading and numerology to help guide hiring decisions. What beliefs are these practices grounded in, and how legitimate are they?



# Outline

- Course Logistics
- Overview
  - What is Knowledge Discovery / Data Mining?
  - Common Data Mining tasks
  - Types of data & data representations
- Data preparation
  - Data quality
  - Exploratory Data Analysis (EDA)
  - Data preprocessing
- Summary

# Summary

- Course Logistics

- Core Concepts

- What is (not) Data Mining?
- Knowledge discovery process
- Overview to common tasks



**Data → Knowledge**

- Data preparation

- Types of data and data quality
- Exploratory data analysis
- Data preprocessing



**Know your data & clean your data!**

# Solutions to Quick Quizzes

- Slide 22: D
- Slide 37: D
- Slide 38: B (A also OK)
- Slide 39: A (in general)
- Slide 55: C
- Slide 65: A
- Slide 67: B (C also OK)

# CS5228: Knowledge Discovery and Data Mining

## Lecture 2 — Clustering I

# Course Logistics — Update

- Project — Team Formation

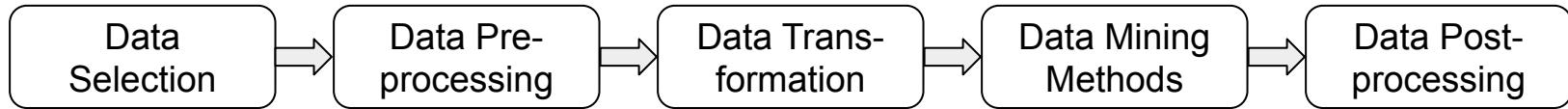
- Default team size: 4 (exception maybe later if needed)
- Final allocation at the teaching team's discretion
- Canvas self-signup group

- Project — Deadline

- Dataset release: Week ~3
- Progress Report: Week ~7
- Final Report: Week 11 (Thu, Oct 30) } Earlier submission are always welcome! :)

# Quick Recap

- Data Mining — from data to knowledge



- Nature of data

- Types of attributes: categorical (nominal / ordinal) vs. numerical (interval / ratio)
- Types of data and data representations  
(e.g., data matrix, transactions, graph, ordered data)
- Data quality

# Quick Recap — Data Preparation

- **Exploratory Data Analysis (EDA)**

- Assess data quality
- Get to know your data

- **Data Preprocessing**

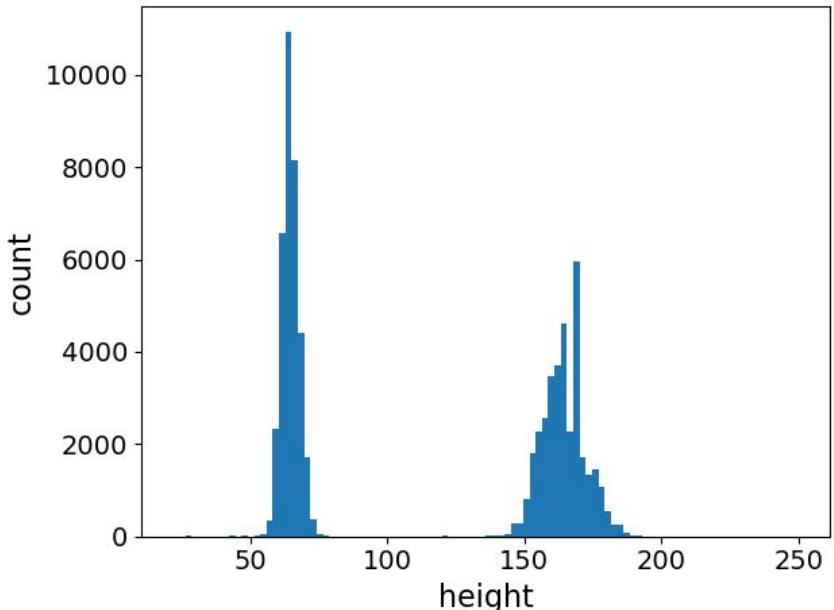
- Improve data quality ("*Garbage in, garbage out!*")
- Generate valid input for data mining algorithms
- Remove complexity from data to ease analysis

Example of noise: missing values

Age	Edu- cation	Marital Status	Annual Income	Credit Default
23	Masters	Single	75k	Yes
N/A	Bachelor	Married	N/A	No
26	Masters	Single	70k	Yes
41	PhD	Single	95k	Yes
18	Bachelor	Single	40k	No
55	Master	Married	N/A	No
30	Bachelor	Single	N/A	No
35	PhD	Married	60k	Yes
N/A	PhD	Married	65k	Yes

# Quick Recap — Clarifications

- Suspicious EDA results  $\Rightarrow$  errors in the data



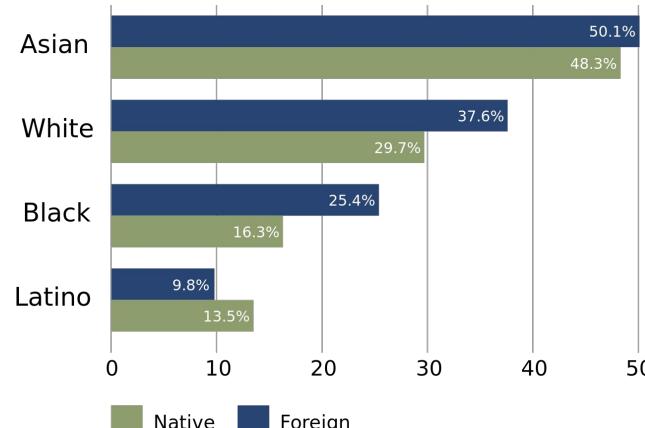
Data may be absolutely correct

Example: infant care dataset (infants + parents)

# Quick Recap — Clarifications

- Removing "questionable" attributes  $\Rightarrow$  better results
  - e.g., removing zodiac sign from credit default prediction might lower accuracy
- Removing "questionable" attributes  $\Rightarrow$  no biases (or perfect privacy)
  - e.g., removing ethnicity not foolproof if it correlates with other attributes

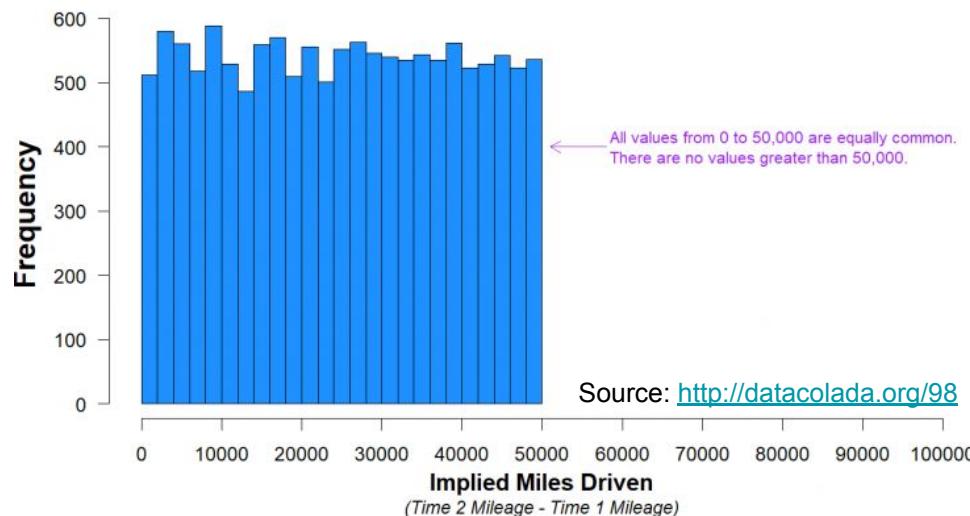
Age	Ethnicity	Education	Annual Income	Credit Approval
23	White	Masters	75k	Yes
35	Buddhist	Bachelor	50k	No
26	Asian	Masters	70k	Yes
41	Asian	PhD	95k	Yes
18	Black	Bachelor	40k	No
...	...	...	...	...



# Quick Recap — EDA: Additional Insights

- Manipulated / fudged data
  - Sometimes data just does not "look right"
  - Example: unexpected/unintuitive data distributions

Figure 1. Histogram of Miles Driven - Car #1 (N=13,488)

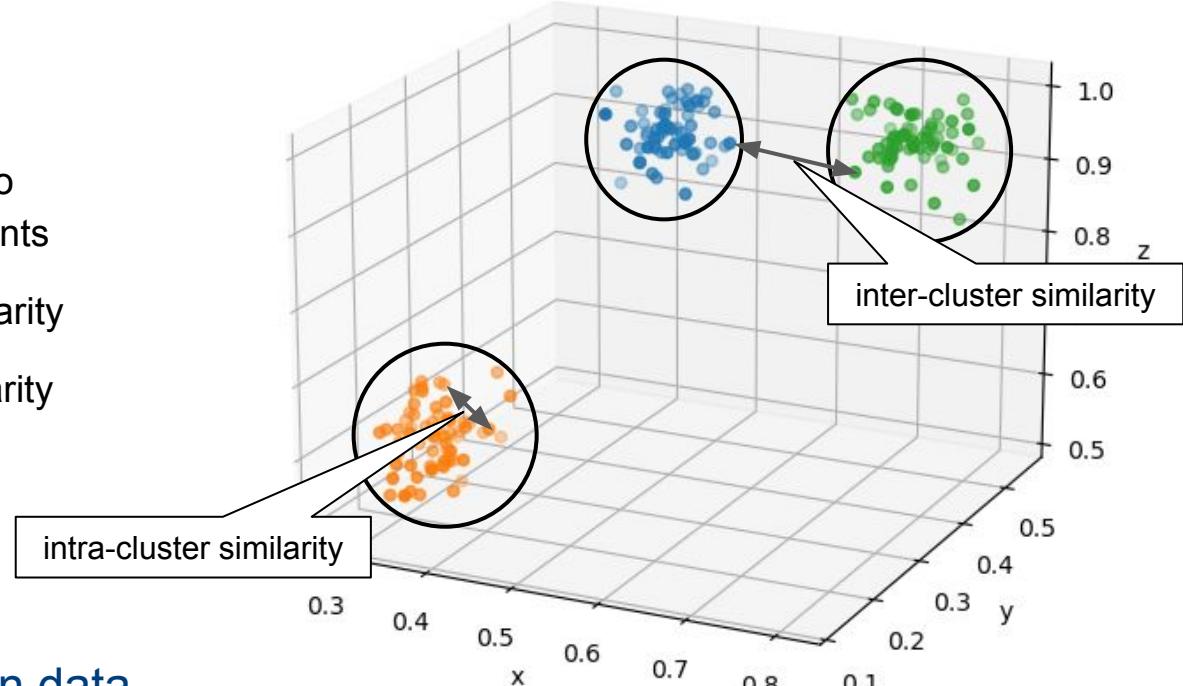


# Outline

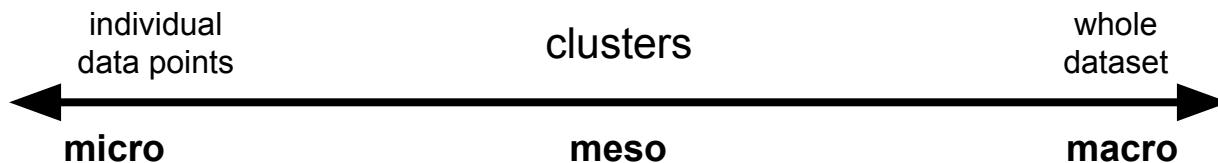
- **Clustering**
  - Overview
  - Applications
  - Concepts
- Clustering algorithms
  - K-Means
  - DBSCAN
  - Hierarchical Clustering
- Cluster Evaluation

# What is Clustering?

- Goal of Clustering
  - Separate **unlabeled** data into groups of **similar** objects/points
  - Maximize **intra-cluster** similarity
  - Minimize **inter-cluster** similarity



- Meso-level perspective on data



# Applications

- Market segmentation
  - Group customers based on behavior and/or preferences
  - Push tailored promotions to all customers in cluster
- Recommender systems
  - Group items (e.g., movies) based on their attributes (e.g., genre, length, budget)
  - Recommend movies from a cluster with movies a user liked
- Web Search Diversification
  - Group Web pages (e.g., news articles) based on content, source (type), etc.
  - Return search results from different clusters to ensure diversity
- ...and many more applications

# Ingredients for Clustering

- Representation of objects, e.g.:
  - (Multidimensional) point coordinates  $x, y$
  - Sets  $A, B$  (e.g., items in a transaction)
  - Vectors  $u, v$  (e.g., TF-IDF)
- Similarity Measure, e.g.:
  - Euclidean Distance
  - Jaccard Similarity
  - Cosine Similarity
- Clustering Algorithm
  - Process that determines if an object belongs to a cluster

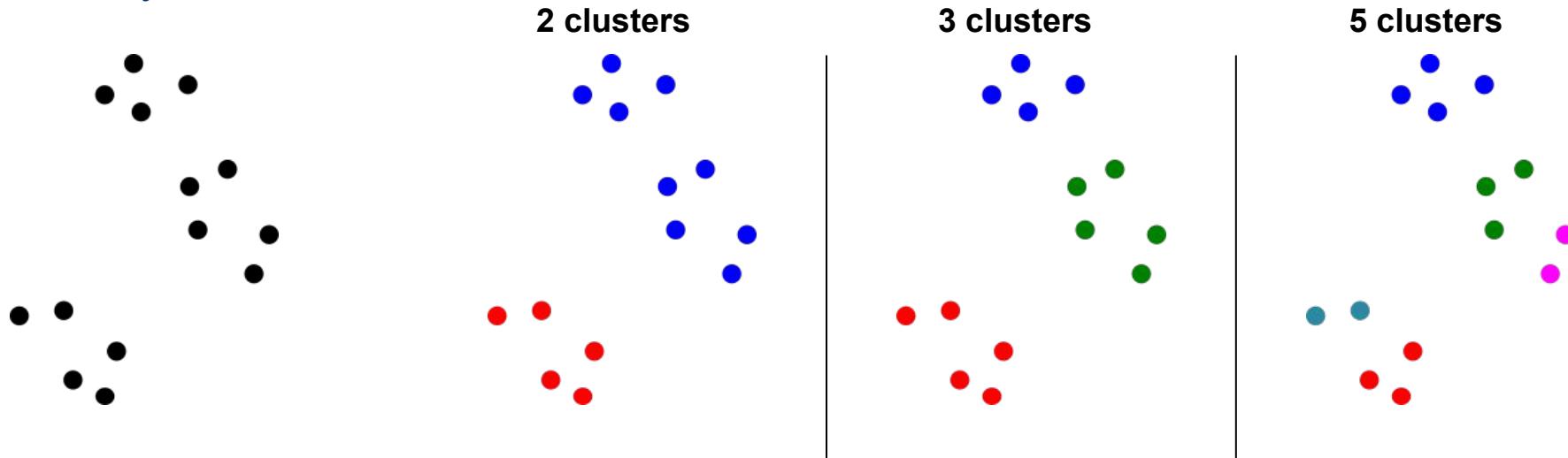
$$dist_{euclidean}(x, y) = \sqrt{\sum_{i=1}^n (x_i - y_i)^2}$$

$$sim_{jaccard}(A, B) = \frac{|A \cap B|}{|A \cup B|}$$

$$sim_{cosine}(u, v) = \frac{u \cdot v}{\|u\| \|v\|}$$

# What makes a clustering "good"?

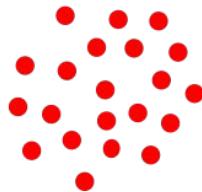
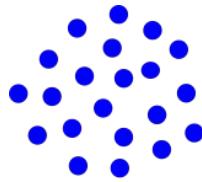
How many clusters?



→ Deciding on a good / meaningful / useful set of clusters not obvious

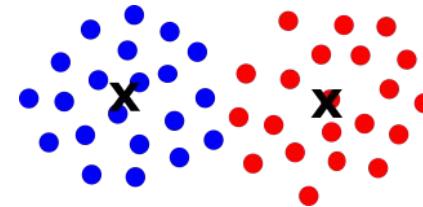
# Types of Clusters

- Well-separated



- Any object in a cluster is closer to every other object in the cluster than to any point outside the cluster

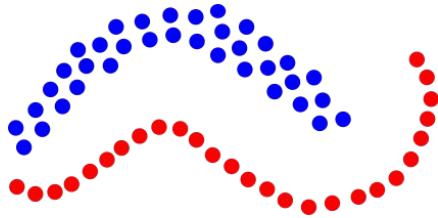
- Center-based



- Any object in a cluster is closer to the "center" of a cluster than to the center of any other cluster
- Example: mean of all data points (in Euclidean space)
- Cluster center commonly called **centroid**

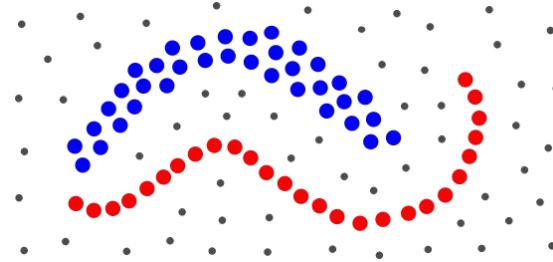
# Types of Clusters

- Contiguity-based



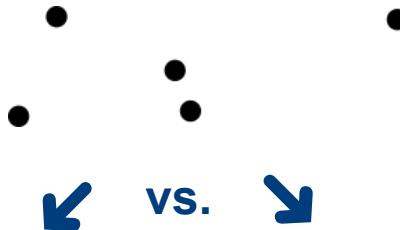
- 2 objects are in a cluster if they are more similar than a specified threshold
- Each object is more similar to some object in that cluster than to any point in a different cluster

- Density-based

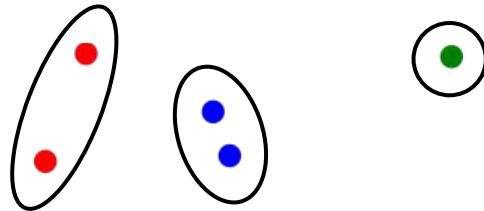


- Cluster = dense( $r$ ) region of objects surrounded by region of low(er) density
- Can typically address noise better than contiguity-based clusters

# Types of Clusterings (i.e., sets of clusters)

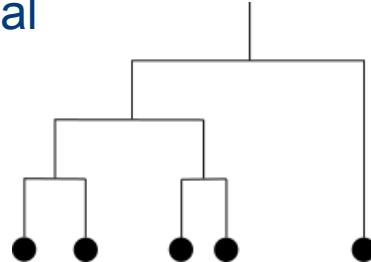


- **Partitional**



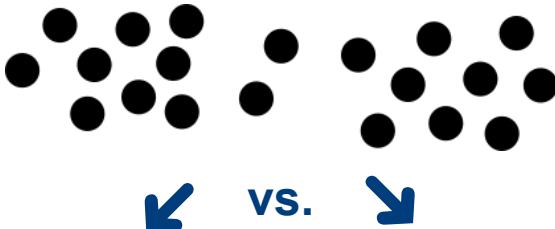
- Division of the set of data objects into non-overlapping subsets (i.e., clusters)
- Each object is in exactly 1 cluster (or in no cluster at all)

- **Hierarchical**

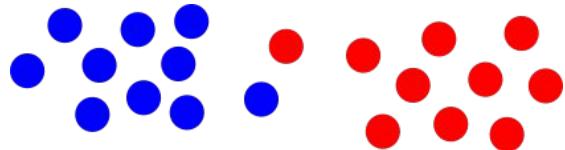


- Clusters can be nested
- A point can belong to different clusters depending on the hierarchy level

# Types of Clusterings (i.e., sets of clusters)

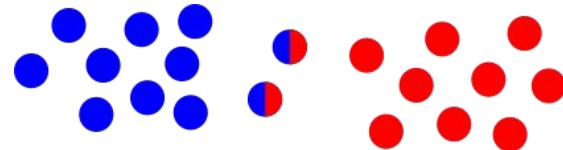


- Exclusive



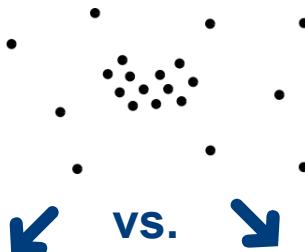
- Each object belongs to 1 cluster

- Non-exclusive / overlapping

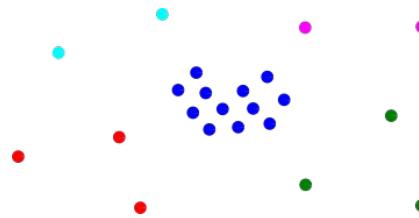


- An object can belong to more than 1 cluster at a time
- Fuzzy clustering: each object belongs to all clusters with a certain probability

# Types of Clusterings (i.e., sets of clusters)

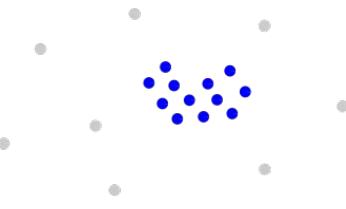


- Complete



- Every object is assigned to (at least) 1 cluster

- Partial



- An object might not belong to a cluster
- Examples: noise, outliers

# Quick Quiz

In what situation can I **NOT** apply clustering on a dataset?

**A**

All attributes of my dataset are nominal attributes

**B**

My dataset is 1-dimensional, i.e., there is only one attribute

**C**

The values of the attributes are not normally distributed

**D**

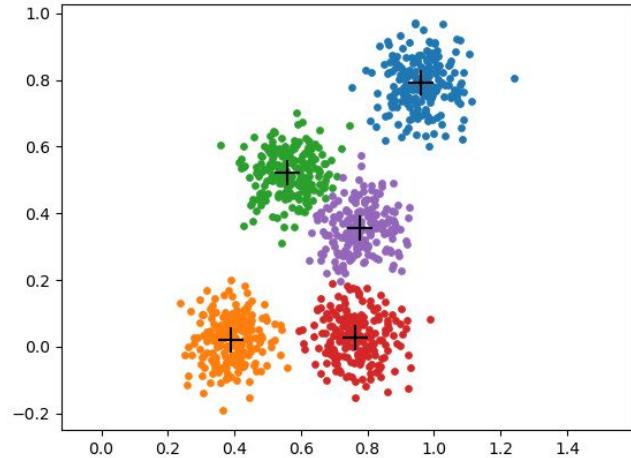
There is no similarity or distance between the data points defined

# Outline

- Clustering
  - Overview
  - Applications
  - Concepts
- Clustering Algorithms
  - K-Means
  - DBSCAN
  - Hierarchical Clustering
- Cluster Evaluation

# K-Means

- Basic characteristics
  - Clusters: centroid-based
  - Clustering: partitional, exclusive, complete
- Inputs (for d-dimensional Euclidean space)
  - $(x_1, x_2, \dots, x_N)$ ,  $x_i \in R^d$
  - Number of clusters K  $\rightarrow c_1, c_2, \dots, c_K$  cluster centers
- Optimization objective
  - Minimize Sum of Squared Error
  - Finding optimal solution is NP-hard  
 $O(N^{Kd+1})$
- Greedy solutions



$$SSE = \sum_{i=1}^K \sum_{x \in C_i} \underbrace{||x - c_i||^2}_{\substack{\text{squared distance between} \\ \text{data point and center}}} \underbrace{C_i}_{\substack{\text{set of points} \\ \text{in i-th cluster}}}$$

# K-Means — How to Define the Centroid of a Cluster?

- Simple case in Euclidean space  $SSE = \sum_{i=1}^K \sum_{x \in C_i} \|x - c_i\|^2$

$$\frac{\delta}{\delta c_k} SSE = \frac{\delta}{\delta c_k} \sum_{i=1}^K \sum_{x \in C_i} (x - c_i)^2$$

$$= \sum_{i=1}^K \sum_{x \in C_i} \frac{\delta}{\delta c_k} (x - c_i)^2$$

$$\Rightarrow \sum_{x \in C_k} 2 \cdot (x - c_k) \stackrel{!}{=} 0$$

# K-Means — How to Define the Centroid of a Cluster?

- Simple case in Euclidean space  $SSE = \sum_{i=1}^K \sum_{x \in C_i} \|x - c_i\|^2$

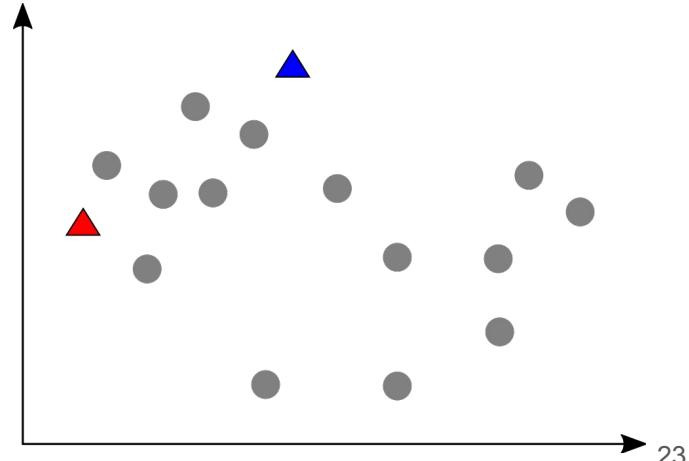
$$\begin{aligned}\frac{\delta}{\delta c_k} SSE &= \frac{\delta}{\delta c_k} \sum_{i=1}^K \sum_{x \in C_i} (x - c_i)^2 \\ &= \sum_{i=1}^K \sum_{x \in C_i} \frac{\delta}{\delta c_k} (x - c_i)^2 \\ \Rightarrow \sum_{x \in C_k} 2 \cdot (x - c_k) \stackrel{!}{=} 0 &\quad \sum_{x \in C_k} 2 \cdot (x - c_k) \stackrel{!}{=} 0 \\ &\Rightarrow \sum_{x \in C_k} x - \sum_{x \in C_k} c_k = 0 \\ &\Rightarrow m_k c_k = \sum_{x \in C_k} x \\ &\Rightarrow c_k = \frac{1}{m_k} \sum_{x \in C_k} x\end{aligned}$$

→ Centroid of cluster = Mean of all points in that cluster

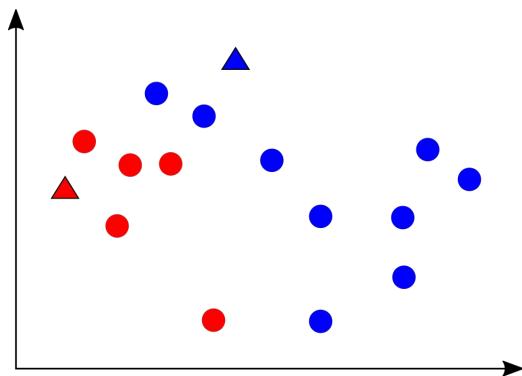
# K-Means — Basic Algorithm (Lloyd's Algorithm)

1. Initialization: Select K points as initial centroids  $c_1, c_2, \dots, c_K$
  2. Repeat
    - 2a) Assignment: assign each point to nearest cluster (i.e., centroid)
    - 2b) Update: move each centroid to the average of its assigned points
- Until no change in assignments

Example: K=2, after initialization

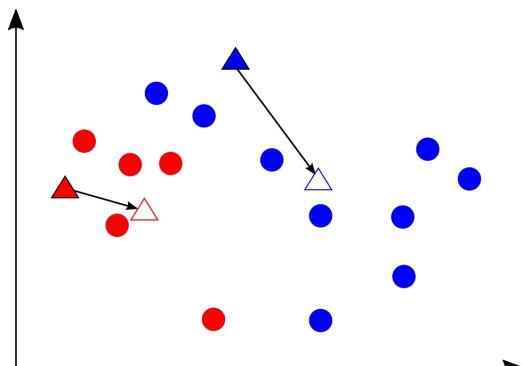


# K-Means — Repeated Steps



- Assignment

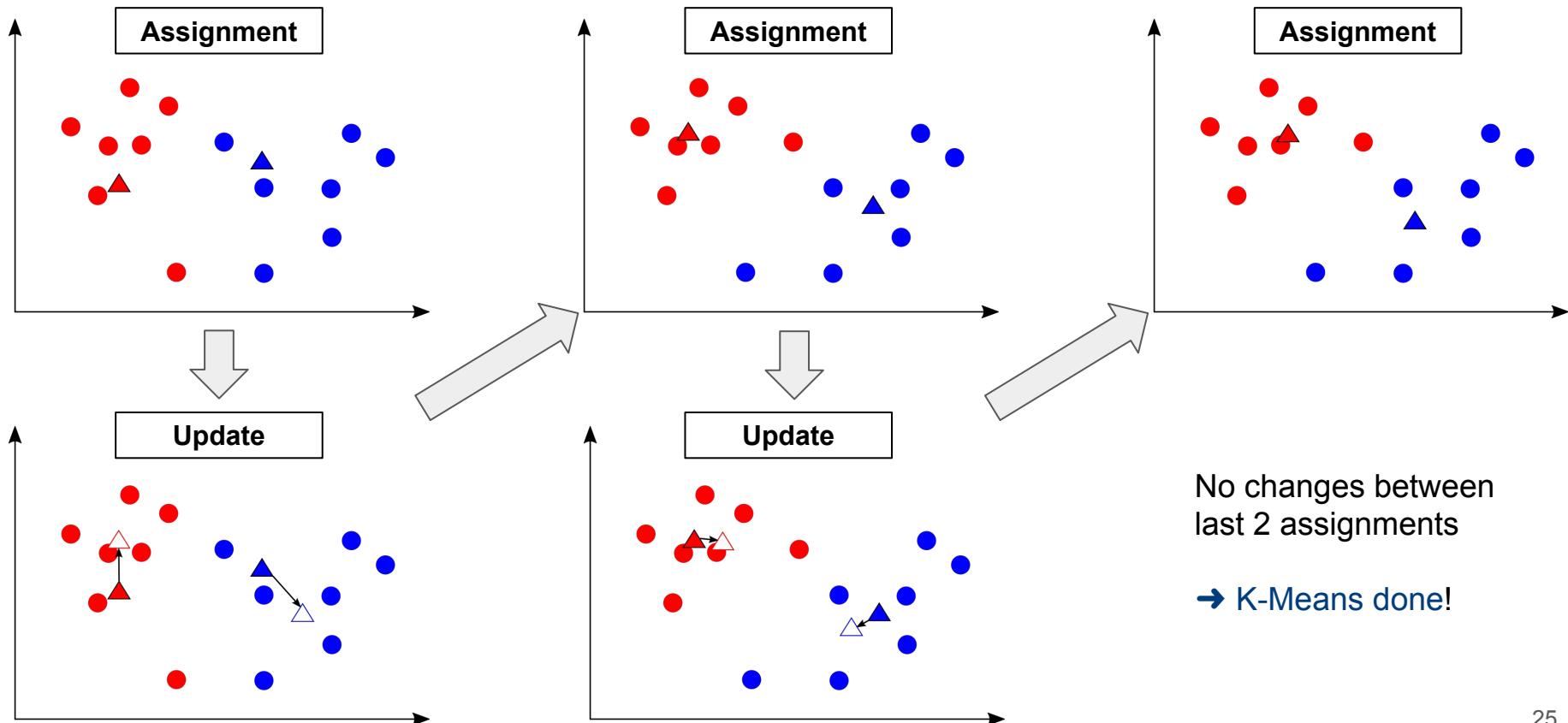
- For each data point  $x$ , find nearest centroid  $c_i$
- Assign  $x$  to cluster  $i$



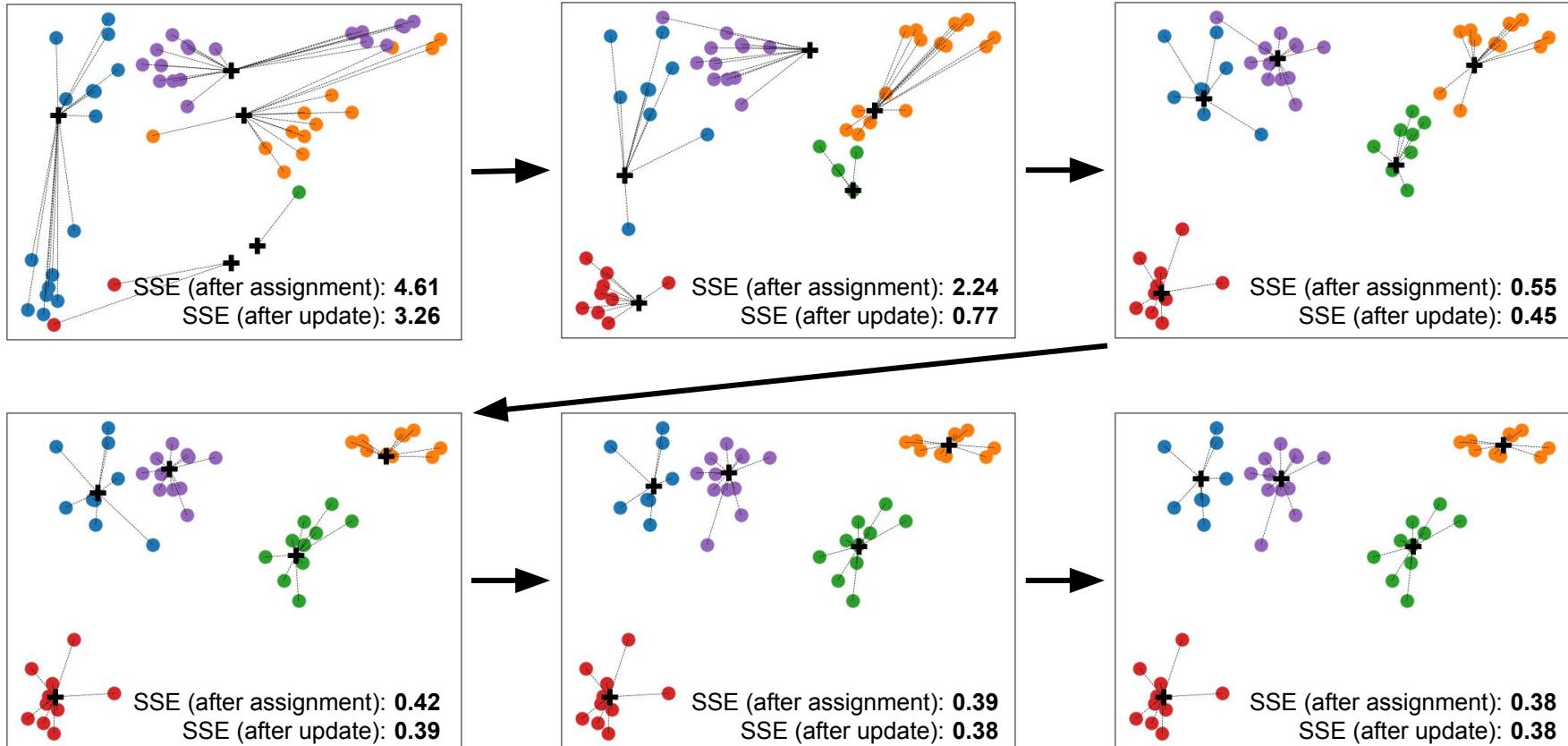
- Update

- Calculate mean of all data points of cluster  $i$
- Set centroid  $c_i$  to mean of cluster  $i$

# K-Means — Iteration until Convergence

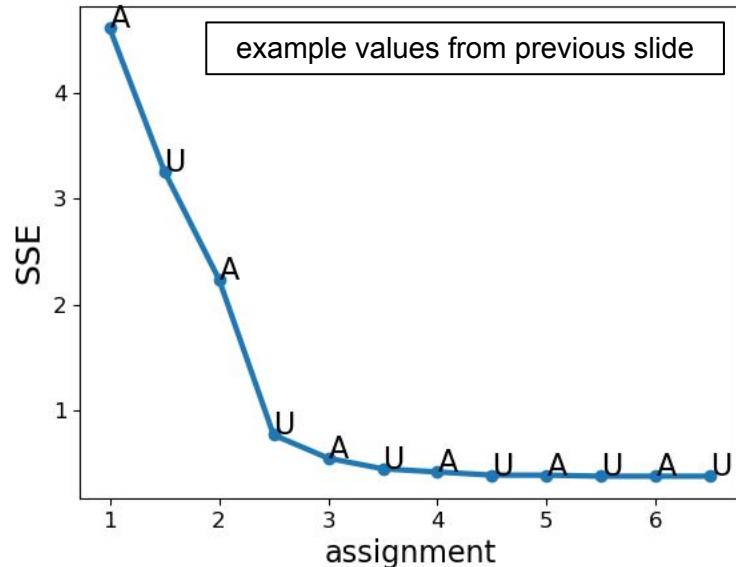


# K-Means — Convergence



# K-Means — Convergence

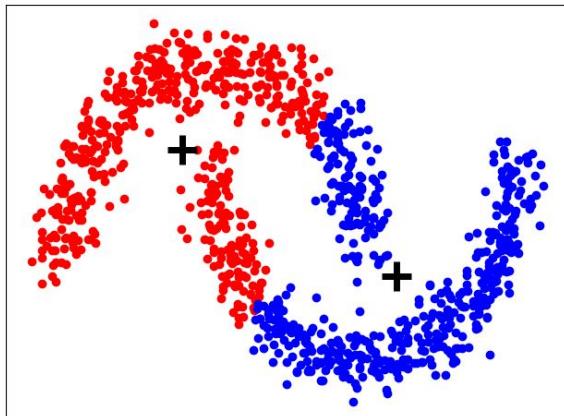
- Good news: K-Means always converges!
  - Both assignment (A) and update (U) reduce SSE (or no changes)
  - Most improvement during the first iterations
- Bad news
  - Lloyd's algorithm returns a **local optimum**, not necessarily a global optimum
  - Important: initialization of centroids  
(discussed in more detail later)



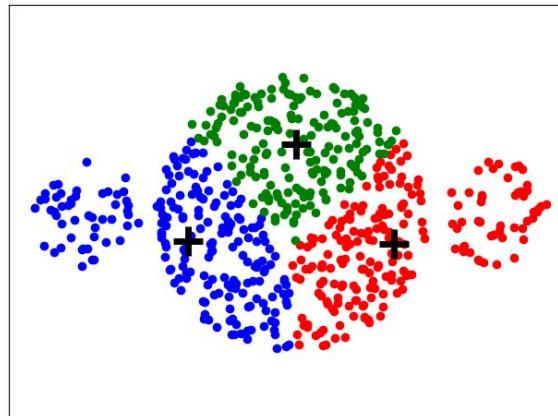
# K-Means — Limitations (Data Distribution)

- K-Means is susceptible to "natural" clusters

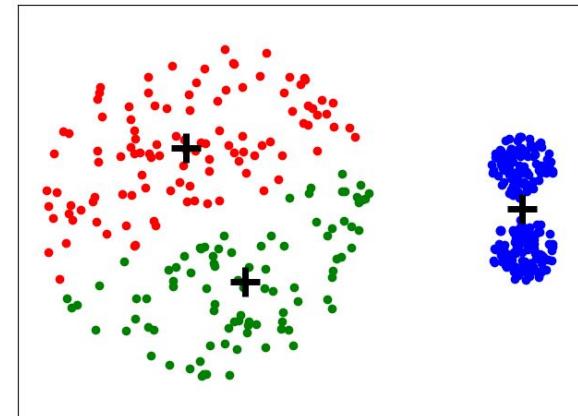
Non-Spherical Clusters



Clusters of different sizes

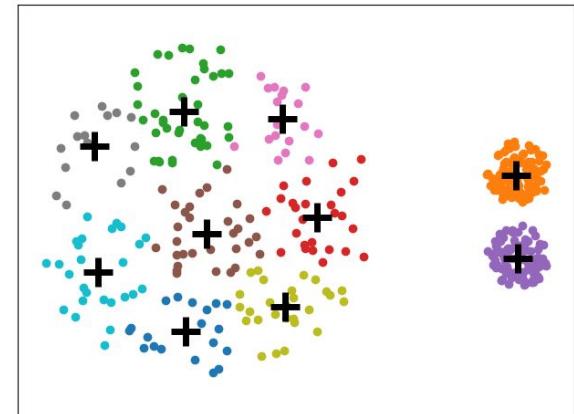
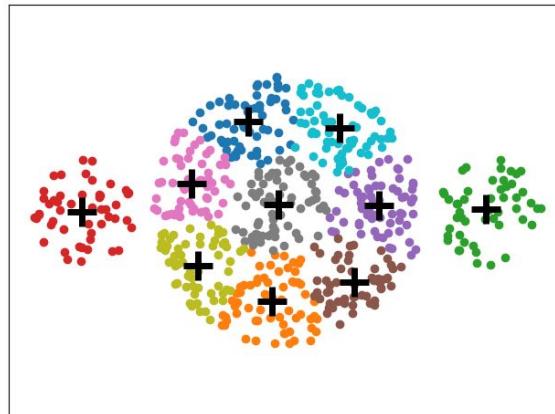
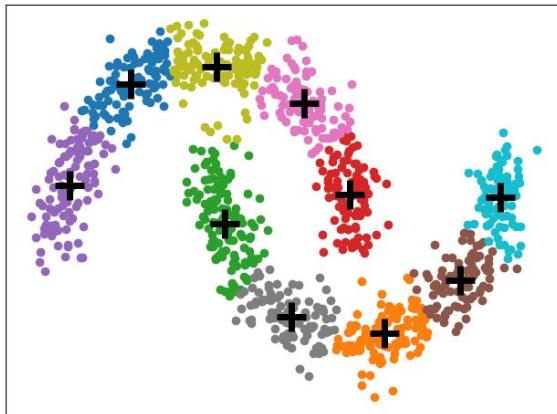


Clusters of different densities



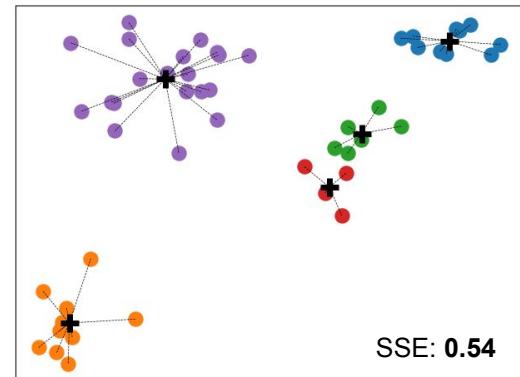
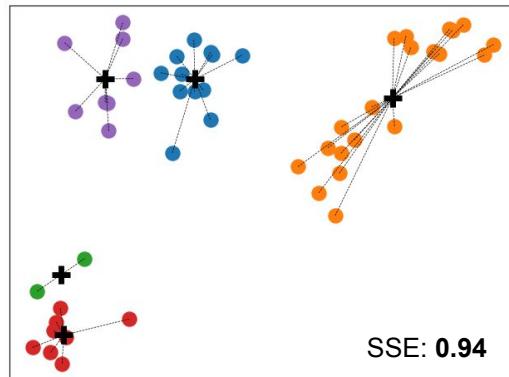
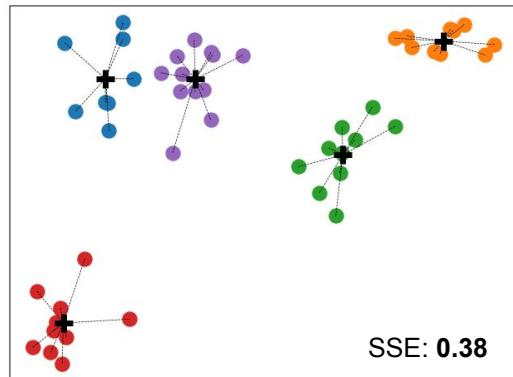
# K-Means — Limitations (Data Distribution)

- Potential workaround: Choose large(r) value for K
  - Intuition: split natural clusters into multiple "well-behaved" (blob-like) subclusters
  - Apply suitable postprocessing steps to merge subclusters



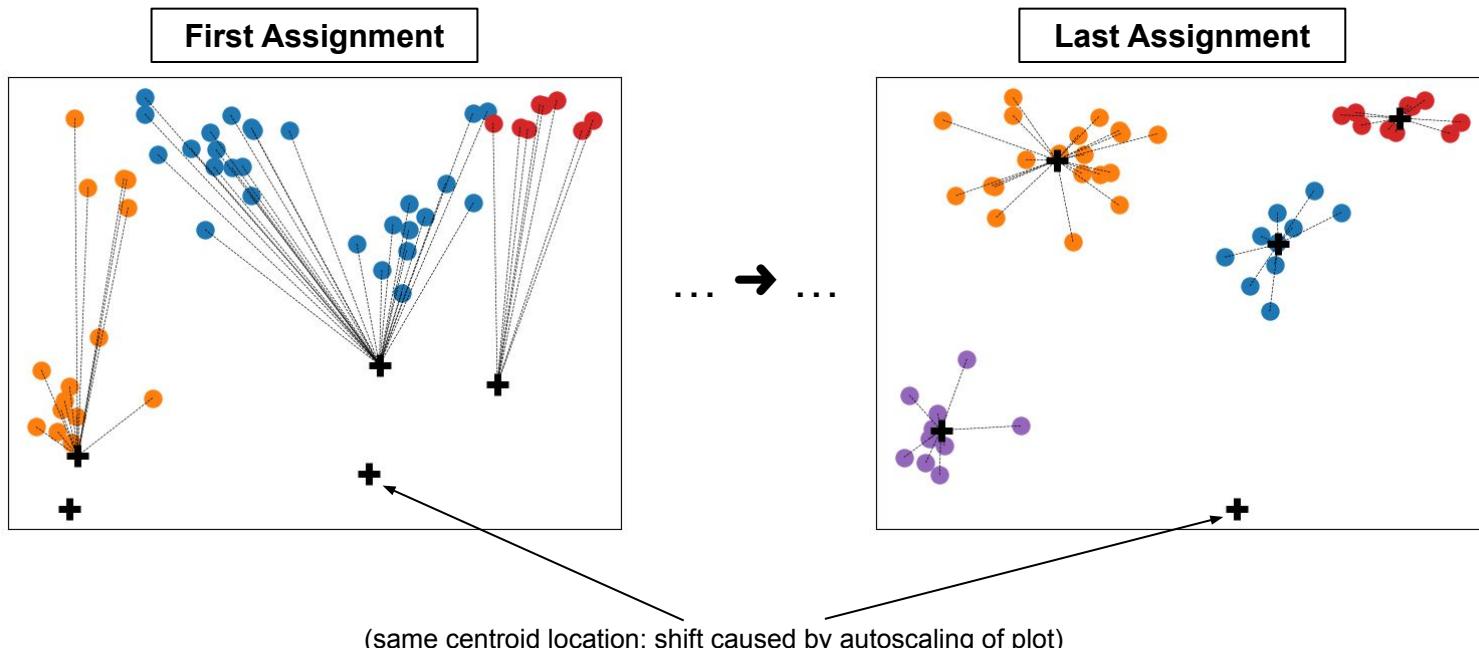
# K-Means — Limitations (Initial Centroids Issue)

- Different initializations of centroids may yield different clusterings
  - Different clusterings typically have different SSEs → global minimum!



# K-Means — Limitations (Initial Centroids Issue)

- Some initialization of centroids can yield empty clusters
  - Occurs when a centroid is "blocked off" data points by other centroids



# Quick Quiz

What is the **maximum number of empty clusters** with K-Means with a really bad initialization of the centroids and  $K \geq 2$ ?

A

0

B

$K - 1$

C

$K$

D

$K + 1$

# K-Means — Handling Empty Clusters

- Artificially fill empty clusters after assignment step (and continue iterations)
  - Replace empty cluster with point that contributes most to SSE
  - Replace empty cluster with a point from the cluster with the highest SSE
- Post processing
  - Split "loose" clusters = clusters with very high SSE
- Modification of Lloyd's algorithm → K-Means variants
  - Typically aim to address the initial centroids issue

# K-Means Variants — K-Means++

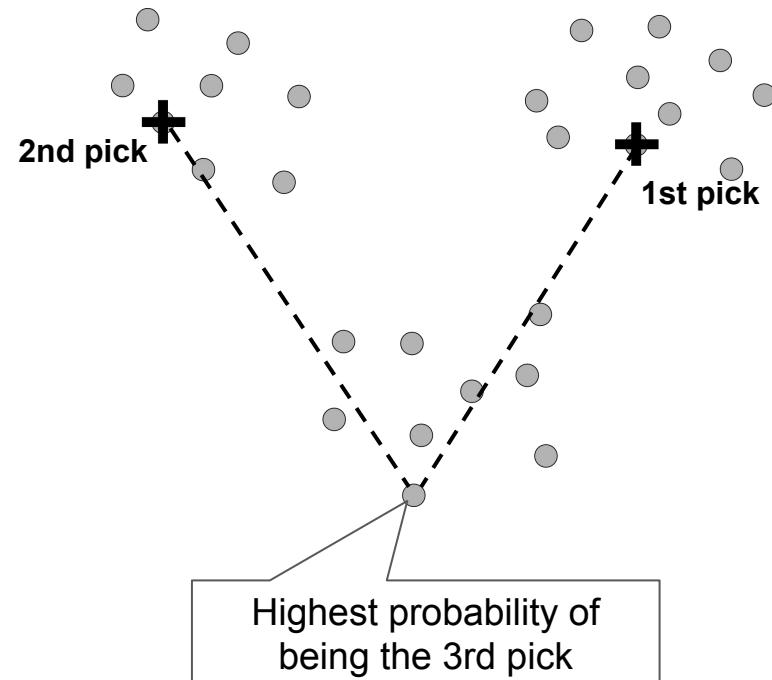
- Only changes initialization of centroids

(assignment/update steps remain the same)

- Goal: spread out centroids
- Better performance in practice
- Theoretical guarantees

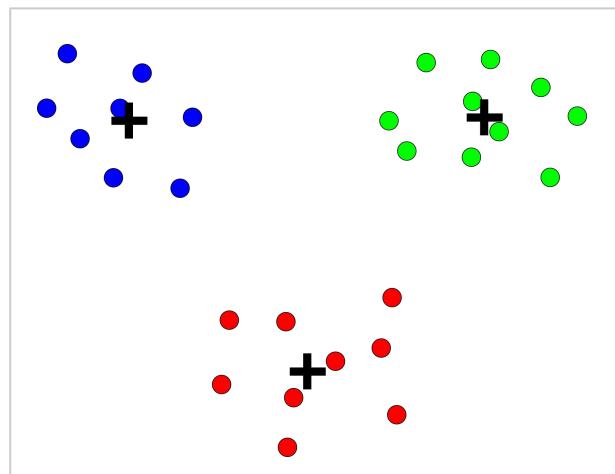
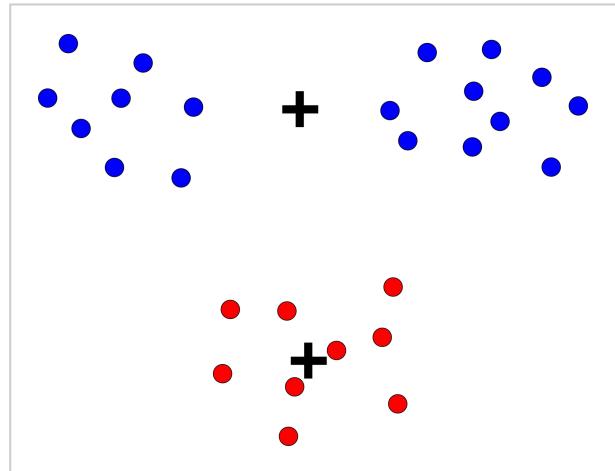
- Initialization process

- Pick random point as first centroid  $c_1$
  - Repeat
    - For each point  $x$ , calculate distance  $d_x$  to nearest existing centroid
    - Pick random point for next centroid with probability proportional to  $d_x^2$
- Until K centroids have been picked



# K-Means Variants — X-Means

- Automatic method to choose K
  - Run K-Means with K=2
  - Iteratively, run K-Means with K=2 over each subcluster
  - Split subcluster only if meaningful w.r.t. a scoring function
- Example scoring functions
  - Bayesian Information criterion (BIC)
  - Akaike information criterion (AIC)
  - Minimum Description Length (MDL)



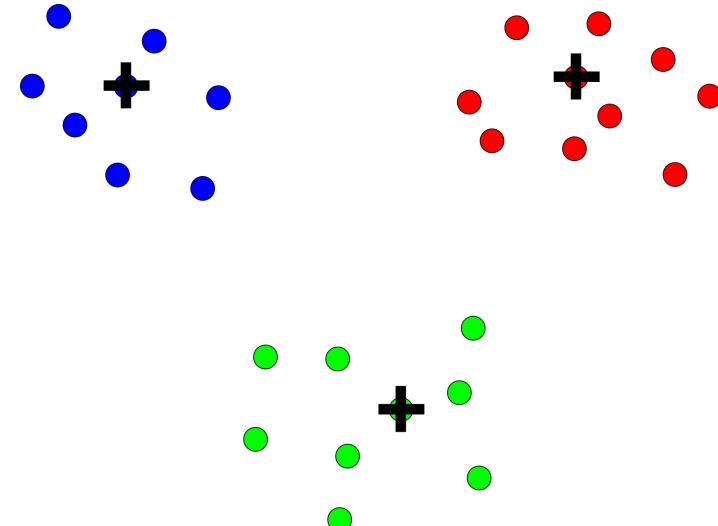
# K-Means Variants — K-Medoids

- **Restriction:** centroids are chosen from the data points

- Does not require the calculation of the averages  
(no average of sets or more complex objects)
- Only notion of distance or similarity still needed  
(e.g., Jaccard similarity for sets or custom metrics)
- More robust to noise and outliers

- **Main issue:** performance

- More expensive Update step
- Swap medoid with each point in cluster and calculate change in cost (e.g., SSE)
- Choose the point as new medoid that minimizes the cost after swapping



# Quick Quiz

Can a theoretically optimal initialization of centroids **guarantee** no empty clusters in any case of running K-Means?

A

No

B

Yes

C

Impossible  
to say

D

Unlikely

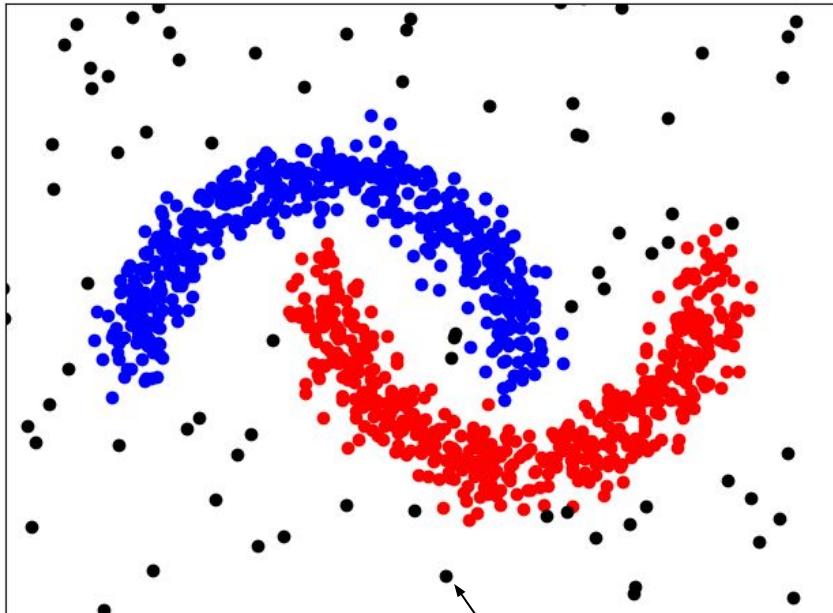
# Outline

- Clustering
  - Overview
  - Applications
  - Concepts
- Clustering Algorithms
  - K-Means
  - **DBSCAN**
  - Hierarchical Clustering
- Cluster Evaluation

# DBSCAN (Density-Based Spatial Clustering of Applications with Noise)

- Basic characteristics
  - Clusters: density-based
  - Clustering: partitional, exclusive, partial
- Inputs (for d-dimensional Euclidean space)
  - $(x_1, x_2, \dots, x_N)$ ,  $x_i \in R^d$
  - $\varepsilon$  — radius defining a points neighborhood
  - $MinPts$  — minimum number of points

$$density = \frac{mass}{volume} = \frac{MinPts}{\varepsilon}$$



noise: not part  
of any cluster

# DBSCAN — Types of Points

- **Core points** ●

- All points with at least  $MinPts$  neighbors with radius  $\mathcal{E}$  (this includes the point itself!)
- Form the **interior** of a cluster

- **Border Points** ○

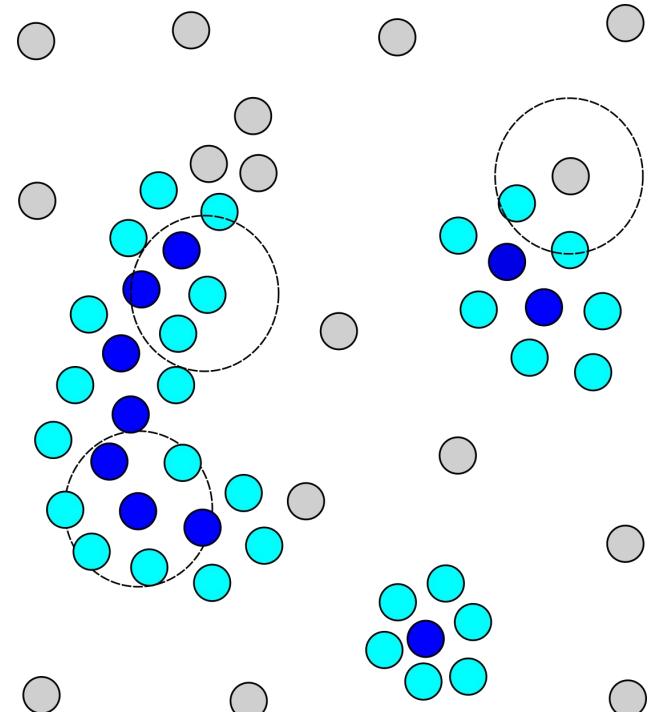
- Non-core points with at least one core point in their neighborhood
- Form the **border** of a cluster

- **Outliers / noise** ○

- All other points
- Default node type

Example:  $\mathcal{E}$  represented by circles

$$MinPts = 5$$



# DBSCAN — Algorithm (2 Iterative Phases)

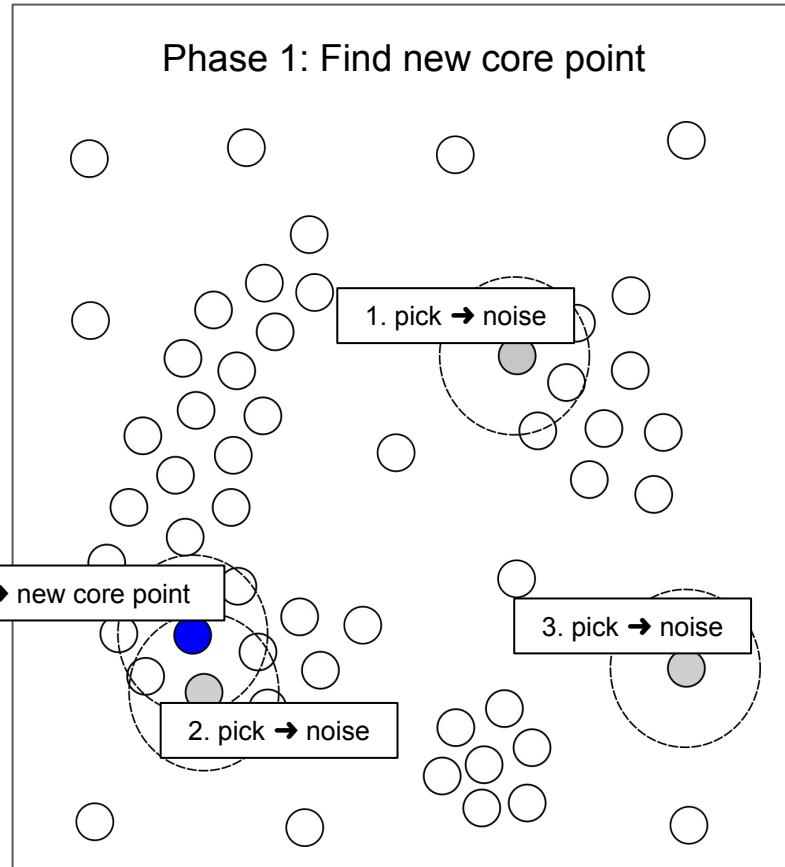
## 1. Find new cluster seed (core point)

Repeat

- 1a) Pick random unexplored point  $x$
- 1b) If  $x$  has less than  $MinPts$  neighbors within  $\varepsilon$ ,  
label  $x$  as noise (might change later)

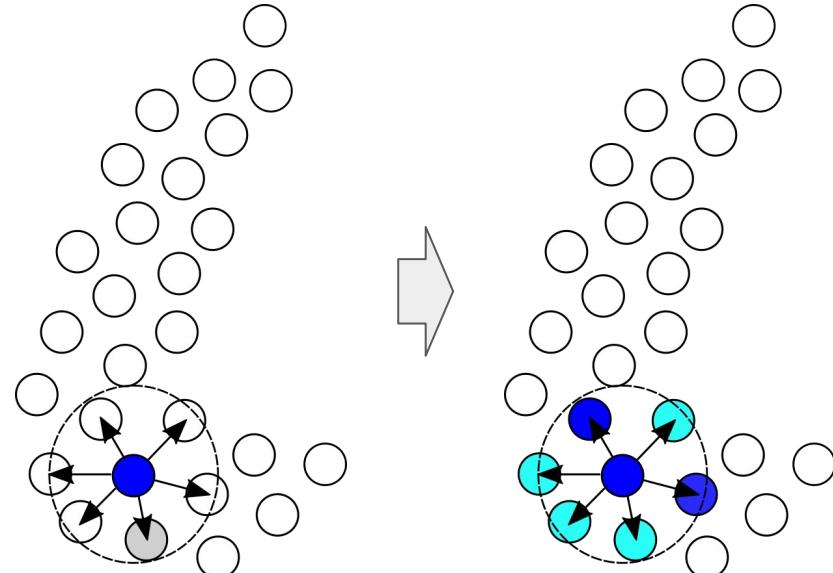
Until  $x$  is a new core point

## 2. Explore new cluster



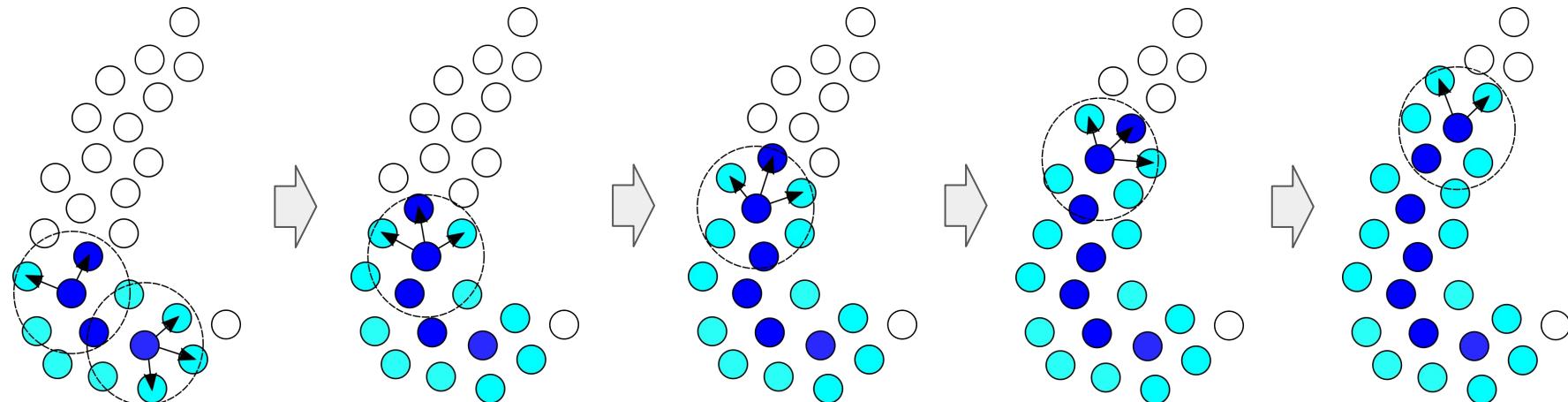
# DBSCAN — Algorithm (Cluster Exploration)

- Explore all neighbors of core point
  - Put all neighbors into the same cluster
  - A neighbor is either a core point or a border point (never noise)
  - New type of neighbor might overwrite previous noise label
- Recursively repeat for each newly found core point



# DBSCAN — Algorithm (Cluster Exploration)

- Example: complete exploration of a cluster
  - Further exploration stops at border points
  - Points beyond border points will become noise or part of a new cluster (after Phase 1)



# DBSCAN — Algorithm (Cluster Exploration)

- **Input:** newly found core  $x_c$  point signifying a new cluster

```
 $S \leftarrow get\_neighbors(x_c, \mathcal{E})$ 
```

WHILE  $|S| \neq 0$

```
 $s \leftarrow S.pop()$ 
```

```
IF  $s.label = "noise"$  THEN  
     $s.label \leftarrow x_c.cluster\_id$ 
```

```
IF  $s.label \neq "unknown"$  THEN  
    CONTINUE
```

```
 $s.label \leftarrow x_c.cluster\_id$ 
```

```
 $neighbors \leftarrow get\_neighbors(s, \mathcal{E})$ 
```

```
IF  $|neighbors| \geq MinPts$  THEN  
     $S \leftarrow S \cup neighbors$ 
```

Initialize  $S$  with the the neighbors of  $x_c$

Pick next point  $s$  from  $S$  until  $S$  is empty

If point  $s$  is (currently) noise,  
add point to current cluster

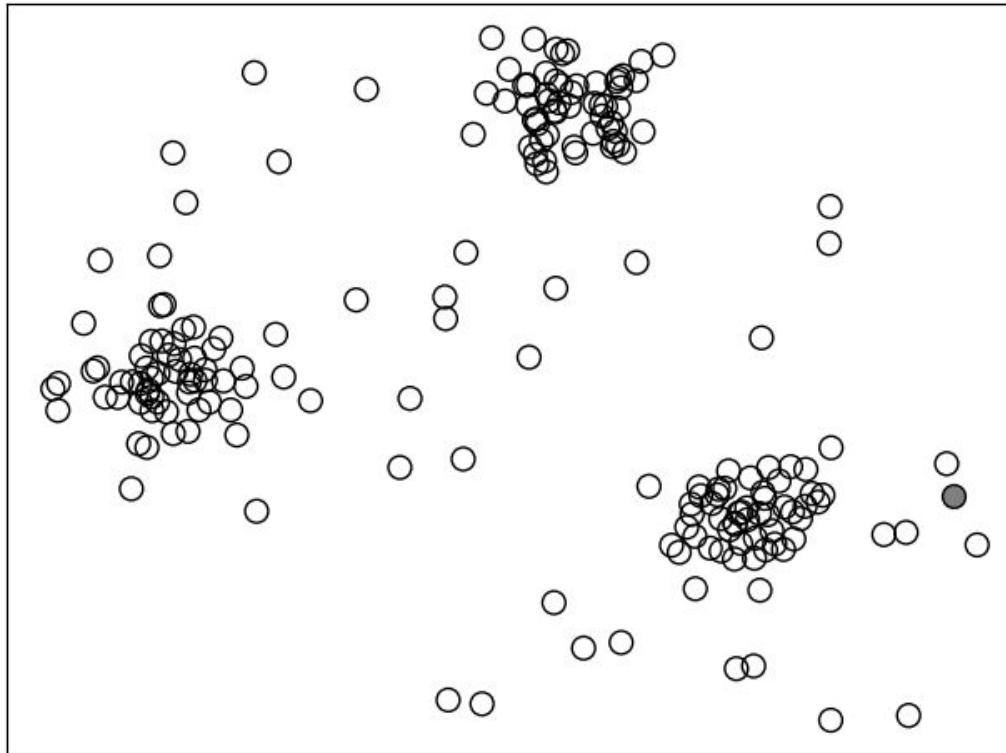
If  $s$  has already been explored,  
continue with next point

Add  $s$  current cluster ( $s$  so far unexplored)

Get all neighbors of  $s$

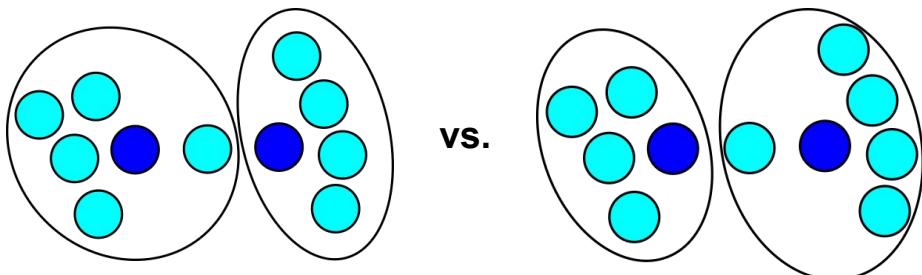
If  $s$  has more than  $MinPts$  neighbors,  $s$  is also a core point  
→ add neighbors to  $S$  (so they will be explored as well)

# DBSCAN — Example with Visualization



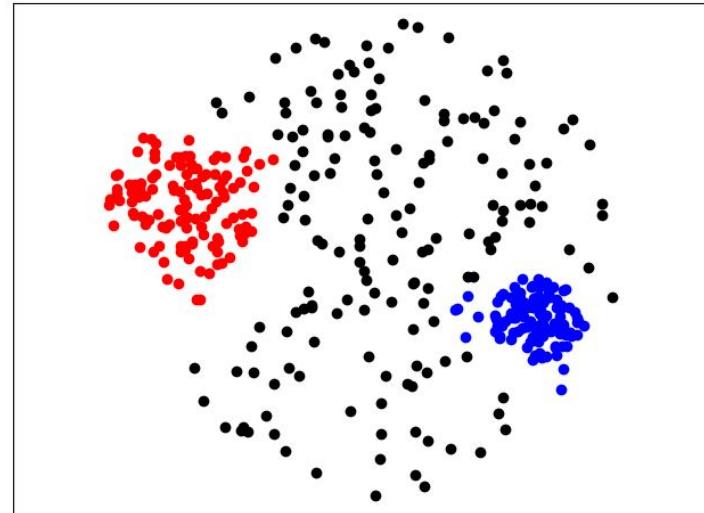
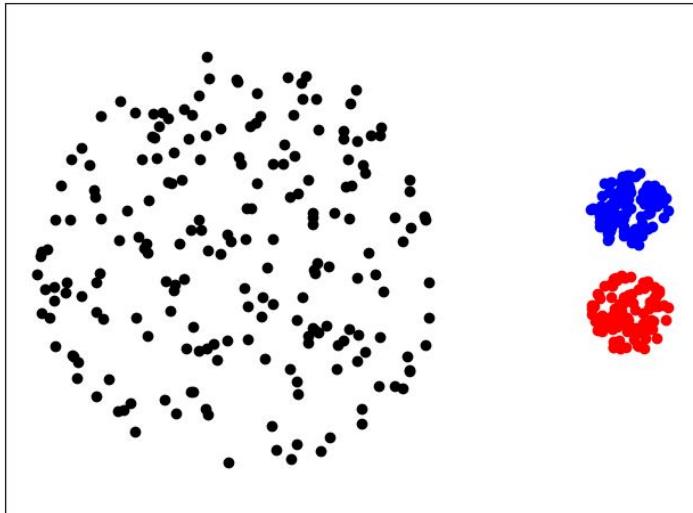
# DBSCAN — Characteristics

- DBSCAN always converges
  - Each data point gets explored (either in Phase 1 or 2)
  - A data point does not change its type (only exception: noise → border)
- DBSCAN is not completely deterministic
  - Phase 1 introduces randomness
  - Border points may be reachable from core points of different clusters
  - Noise and core points deterministic



# DBSCAN — Limitations

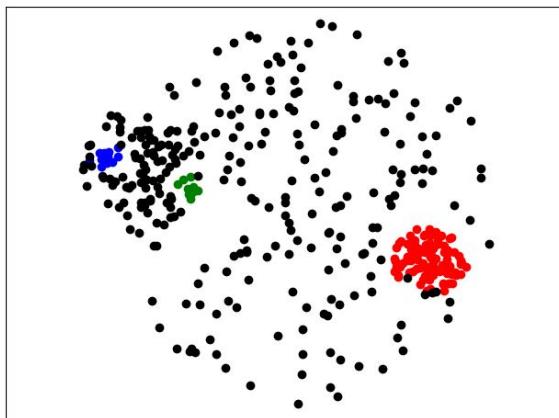
- DBSCAN cannot handle different densities



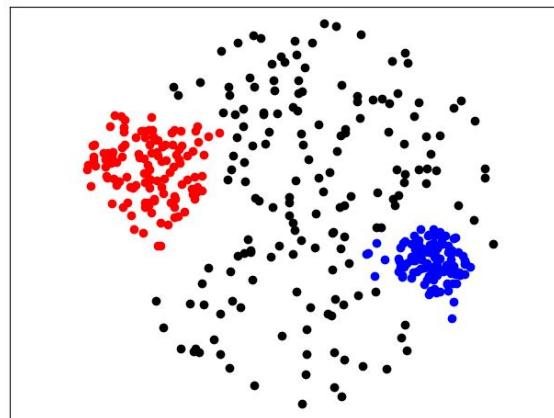
# DBSCAN — Limitations

- DBSCAN is generally very sensitive to parameters
  - Choosing  $\varepsilon$  and  $MinPts$  requires good understanding of data and context

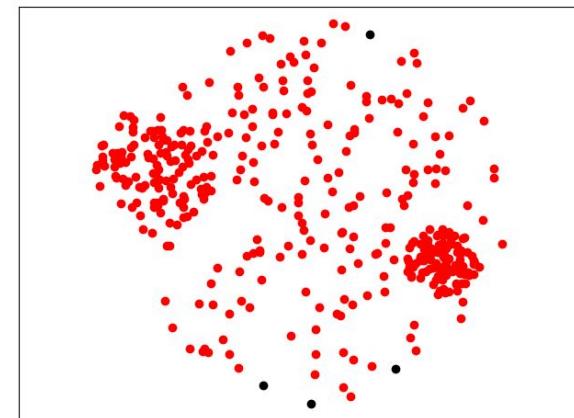
$$\varepsilon = 0.05$$



$$\varepsilon = 0.1$$



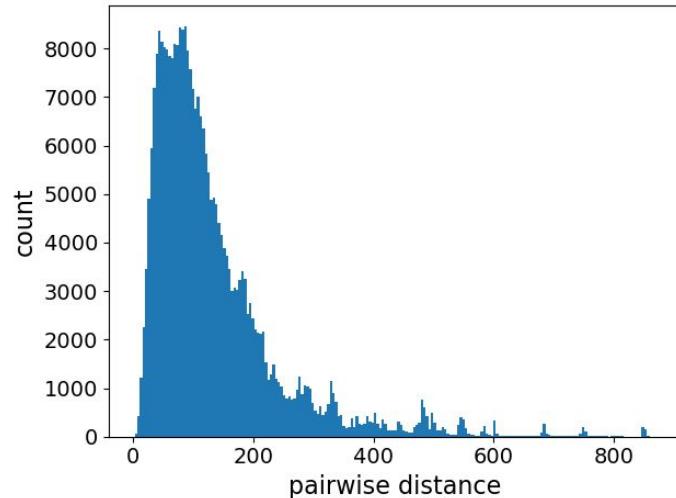
$$\varepsilon = 0.2$$



( $MinPts = 10$  for all three examples)

# DBSCAN — How to Choose Parameter Values?

- Informed by results of EDA, e.g.:



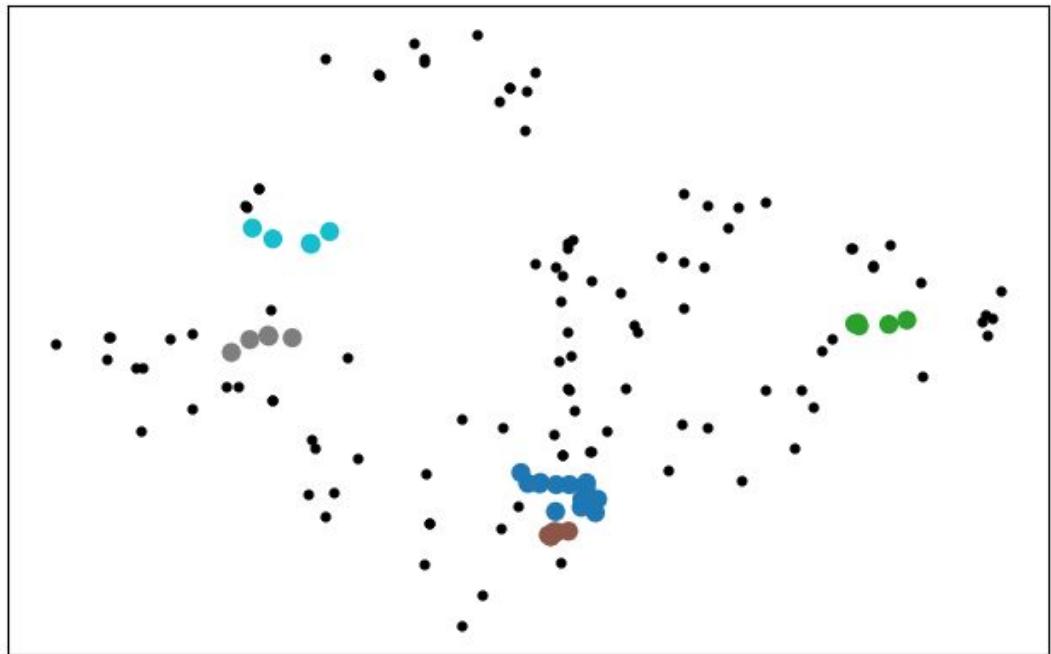
Distribution of all pairwise distances

First insights into suitable values for  $\varepsilon$

- Density of data points has intuitive semantic meaning, e.g.:
  - Geographic distance between bars in a city
  - Task: Find areas (clusters) with more than 10 bars within 500m

# DBSCAN — How to Choose Parameter Values?

- Intuitive interpretations of meaningful parameter values
- Example
  - 141 McDonald's restaurants across Singapore
  - Find areas with more than 5 restaurants within a 500m radius



# Quick Quiz

What is the **smallest cluster size** when using DBSCAN and a given value for  $MinPts$ ?

A

1

B

$MinPts - 1$

C

$MinPts$

D

$MinPts + 1$

# Quick Quiz

Given  $N=1,000$  data points and using DBSCAN with  $MinPts=10$ , what is the **minimum** possible number of clusters?

A

0

B

1

C

MinPts

D

MinPts + 1

# Quick Quiz — Side Note

- Slight inconsistencies across different sources
  - Relevant step in algorithm

$\text{neighbors} \leftarrow \text{get\_neighbors}(s, \mathcal{E})$

Does  $\text{neighbors}$  contain  $s$  itself?

- Original paper: Yes,  $s$  is part of neighborhood

- Smallest cluster size:  $\text{MinPts}$

- Maximum number of clusters:  $\left\lfloor \frac{N}{\text{MinPts}} \right\rfloor$        $N$  = number of data points

# Clustering Algorithms

- K-Means
- DBSCAN
- Hierarchical Clustering (next lecture)

# Outline

- Clustering
  - Overview
  - Applications
  - Concepts
- Clustering Algorithms
  - K-Means
  - DBSCAN
  - Hierarchical Clustering (next lecture)
- Cluster Evaluation

# Summary — Clustering I

- Clustering as fundamental data mining algorithm
  - Cluster provide a "meso-view" on data
  - Required: well-defined notion of similarity between data points
  - No single definition what a good cluster / clustering is
- Wide range of different clustering algorithms
- In this lecture: K-Means & DBSCAN
  - K-Means: split all data points into k clusters based in their **relative similarities**
  - DBSCAN: find clusters based on **absolute similarities** between data points

# Solutions to Quick Quizzes

- Slide 18: D
- Slide 32: B
- Slide 37: A
- Slide 51: C
- Slide 52: A

# **CS5228: Knowledge Discovery and Data Mining**

## Lecture 3 — Clustering II

# Course Logistics — Update

- Assignment 1

- Topics: EDA, Data Preparation & K-Means
- Submission deadline: Thu, Sep 12 (11.59 pm)

- Reminder: Honor Code

- Cheat and plagiarism is serious academic offence
- Do not read, copy, steal, etc. others code
- Lecture slides & videos should easily suffice

- Project

- Team formation about to be finalized
- Kaggle Competition will launch soon.

New submission deadline:  
Thu, Nov 14 (Week 13)

# Quick Recap — Tutorial

city	state	parent
p-b	PA	Bach
p c	OK	S
p-b	FL	ma
p a	CT	
p c	WV	
p b	IN	assoc
p b	CO	
p-b	MP	
p-d	WY	
p-b	MS	

- Alternative encoding of nominal attributes: "*proxy encoding*"

- Replace nominal values with 1 or more numerical values
- Numerical values should reflect underlying assumption of the impact of attribute
- Example: "*What makes 'state' a potentially useful attribute?*"

Interpretation	Encoding through replacement
Average Political leaning	→ Percentage of democrats/republicans
State education budget	→ Dollar-per-student value
School system	→ Rate of homeschooling
Urbanization	→ #universities per capita
...	→ ...

# Quick Recap — Tutorial

city	state	parent
p-b	PA	Bach
p c	OK	S
p-b	FL	ma
p a	CT	
p c	WV	
p b	IN	assoc
p b	CO	
p-b	MP	
p-d	WY	
p-b	MS	

- Important: "careless" encoding may imply questionable interpretation
  - Question: *"What is the interpretation of my encoding, and is it meaningful?"*
  - In practice, often very difficult to answer

Encoding	Interpretation
----------	----------------

- Ordinal values → PA < OK < FL < CT < WV < ...
- Latitude/Longitude → Geographic location of state matters
- #KFC per capita → Proliferation of fast food matters
- ... → ...



It **might** be correct, even if only incidentally!

# Quick Recap — Lecture

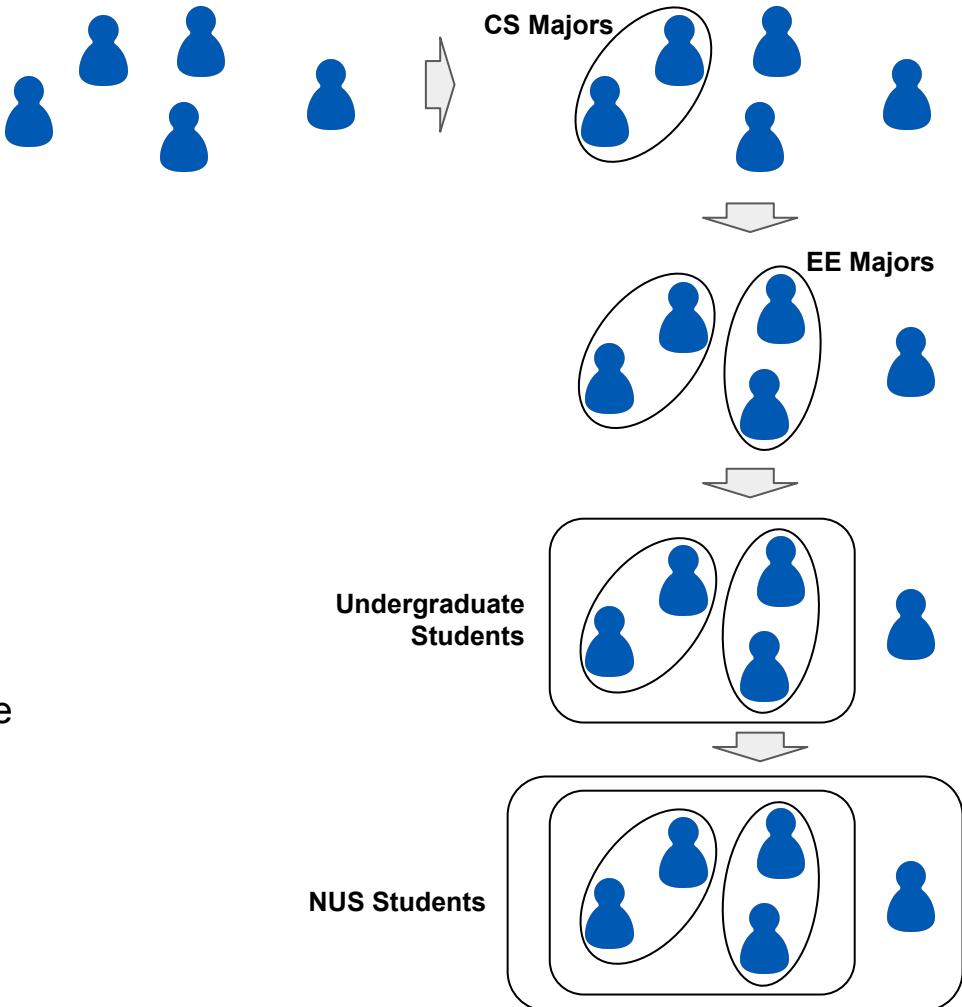
- **Clustering**
  - Grouping data points based on their similarities
  - No single definition for cluster or clustering → different meaningful intuitions
  - General-purpose data mining method ("only" distance/similarity measure required)
- **Algorithms discussed so far:**
  - K-Means (centroid-based, partitional, exclusive, complete)
  - DBSCAN (density-based, partitional, exclusive, partial)

# Outline

- **Clustering**
  - Overview
  - Concepts
  - Applications
- **Clustering algorithms**
  - K-Means
  - DBSCAN
  - Hierarchical Clustering
- Cluster Evaluation

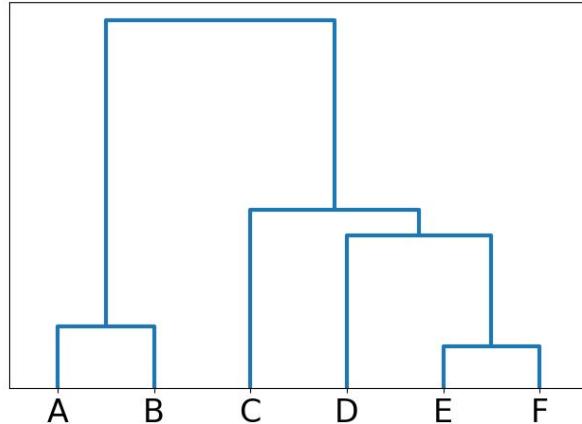
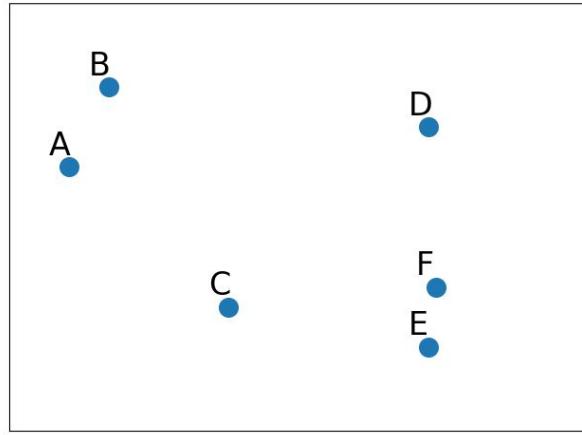
# Hierarchical Clustering

- Basic characteristics
  - Clusters: depends...
  - Clustering: hierarchical (duh!), complete, exclusive (at each level!)
- No parameterization (in principle)
  - In practice, typically number of clusters is specified (similar to K-means)
  - Different choices of measures to calculate distances between clusters



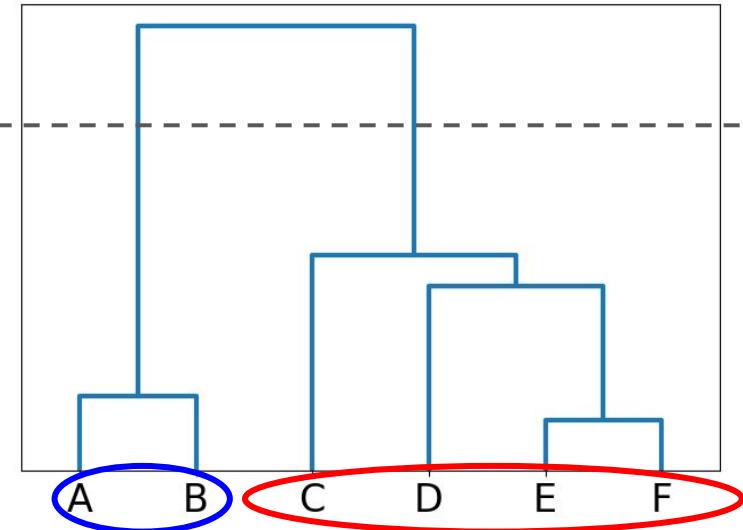
# Dendrograms

- Dendrogram: Visualization of hierarchical relationships
  - Binary tree showing how clusters are hierarchically merged/split
  - Each node is a cluster
  - Each leaf is a singleton cluster
  - Height reflects distance between clusters  
(e.g., large distance between A/B and C/D/E/F clusters)

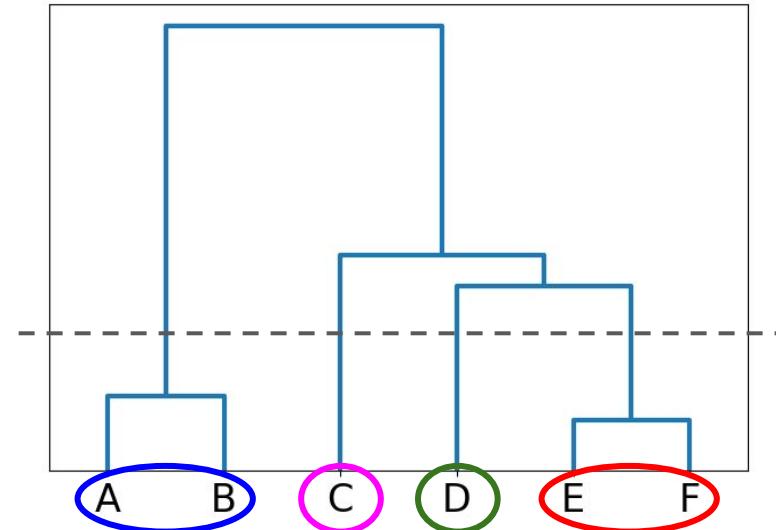


# Hierarchical Clustering — Dendograms

- A clustering can be obtained by cutting a dendrogram at the desired level



2 clusters

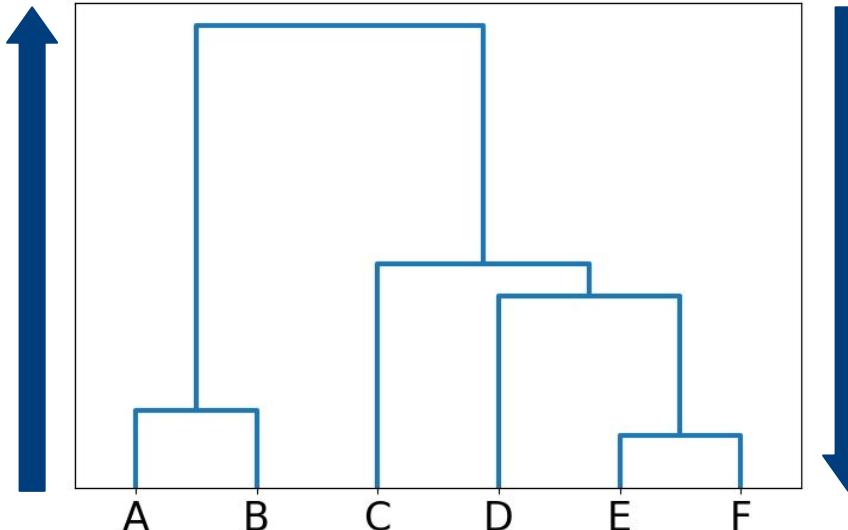


4 clusters

# Hierarchical Clustering — 2 Main Types

## Agglomerative (bottom-up)

- Start with each point being its own cluster
- At each step, merge closest pair of clusters
- Stop when only one cluster is left



**AGNES** (AGglomerative NESting)

## Divisive (top-down)

- Start with one cluster containing all points
- At each step, split a cluster
- Stop when each cluster contains a single point

**DIANA** (DIvise ANAlysis)

# Quick Quiz

What is the **minimum** and **maximum** possible **depth** of a dendrogram for a dataset with  $N$  data points?

(assume O-notation)

A

Min:  $\log_2 N$  Max:  $N$

B

Min:  $\sqrt{N}$  Max:  $N \log_2 N$

C

Min:  $\sqrt{N}$  Max:  $N$

D

Min:  $\log_2 N$  Max:  $N \log_2 N$

# AGNES — Basic Algorithm

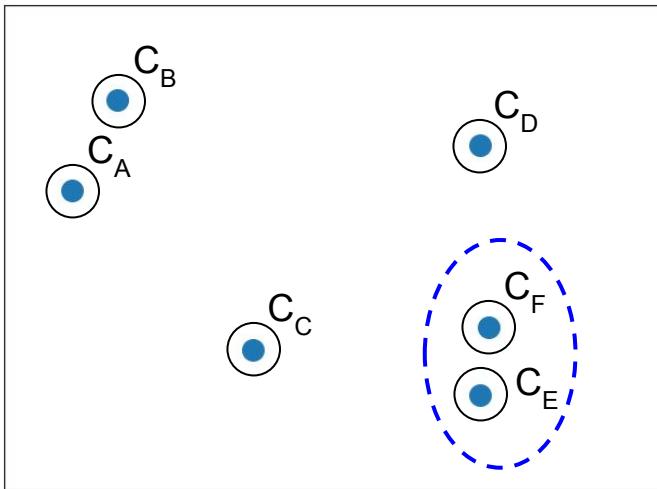
1. Initialization: Each point forms its own cluster
  2. Repeat
    - 2a) **Merge** the two closest clusters into one
- Until** only 1 cluster remains



# AGNES — Implementation

- Implementation using distance matrix

Initial clustering: each cluster, one point



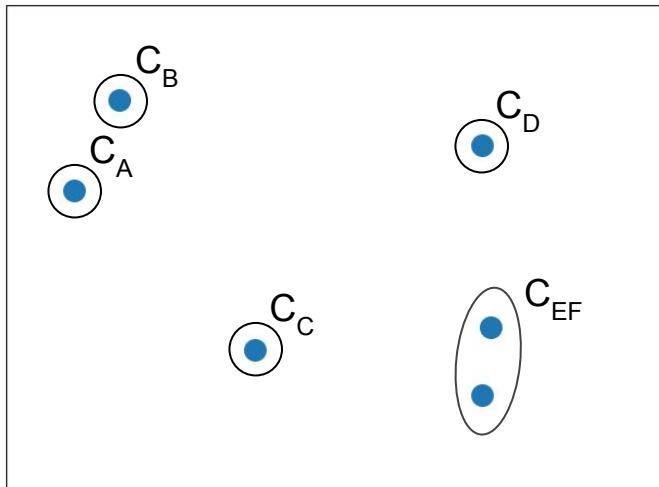
Distance between clusters = distance between points

	C <sub>A</sub>	C <sub>B</sub>	C <sub>C</sub>	C <sub>D</sub>	C <sub>E</sub>	C <sub>F</sub>
C <sub>A</sub>	$\infty$	2.25	5.32	9.06	9.79	9.49
C <sub>B</sub>		$\infty$	6.08	7.85	9.86	9.21
C <sub>C</sub>			$\infty$	6.73	4.81	5.02
C <sub>D</sub>				$\infty$	5.51	4.00
C <sub>E</sub>					$\infty$	<b>1.53</b>
C <sub>F</sub>						$\infty$

# AGNES — Implementation

- What's the distance between clusters? (beyond containing single points)

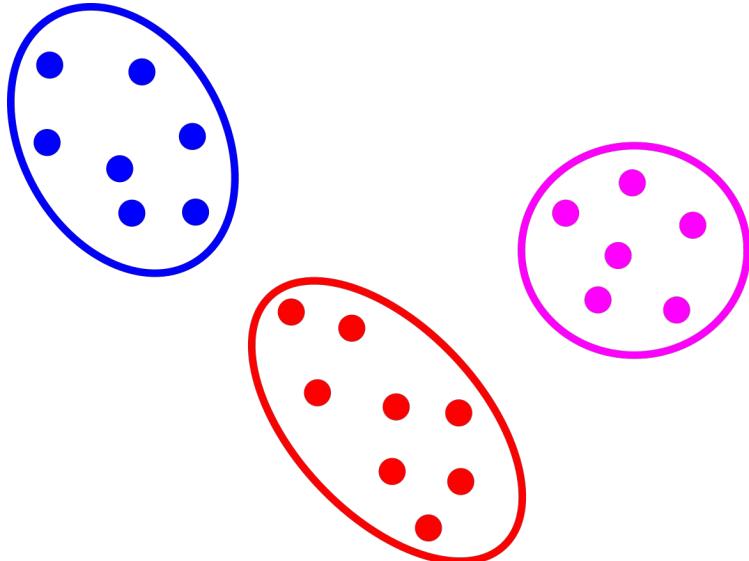
Clustering after merging  $C_E$  and  $C_F$  to  $C_{EF}$



	$C_A$	$C_B$	$C_C$	$C_D$	$C_{EF}$
$C_A$	$\infty$	2.25	5.32	9.06	???
$C_B$		$\infty$	6.08	7.85	???
$C_C$			$\infty$	6.73	???
$C_D$				$\infty$	???
$C_{EF}$					$\infty$

# Quick "Quiz"

Which 2 clusters should  
get merged next?



**A**

Red & Blue

**B**

Red & Magenta

**C**

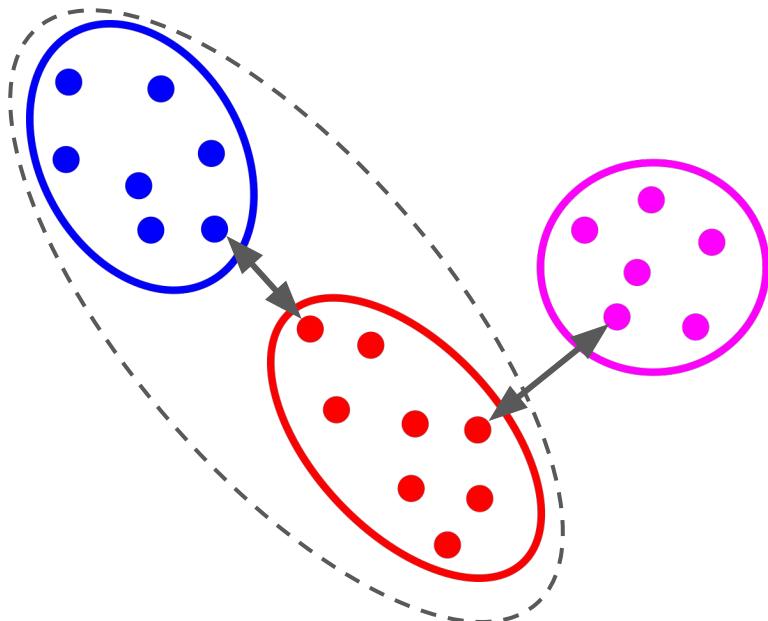
Blue & Magenta

**D**

It's Friday,  
I'm out...

# AGNES — Single Linkage

- Single Linkage Clustering
  - Distance between clusters = **minimum distance** between two points from each cluster



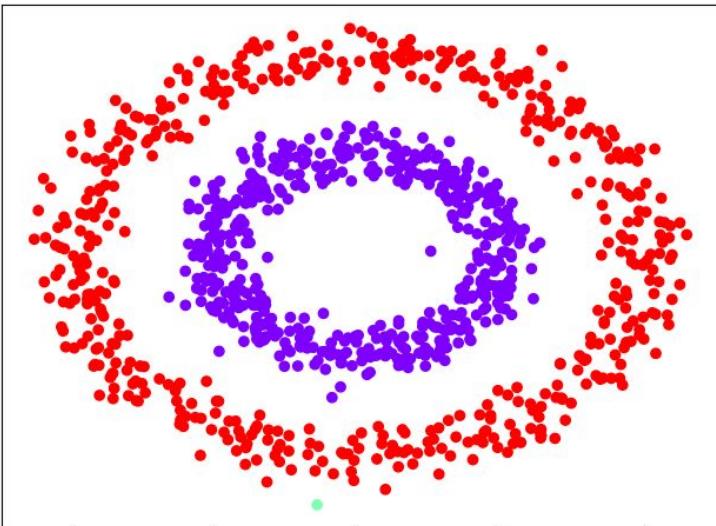
simple pointwise distance

$$d_{single}(C_i, C_j) = \min_{p \in C_i, q \in C_j} d(p, q)$$

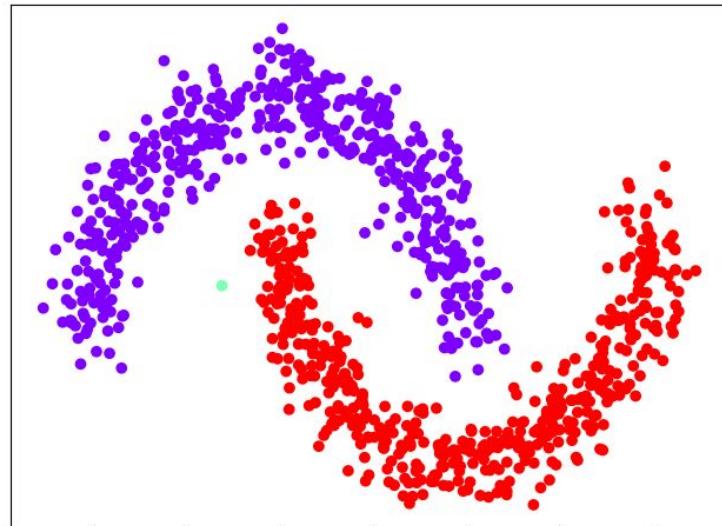
# AGNES — Single Linkage

- Strength: Can handle non-globular shapes

#cluster = 3



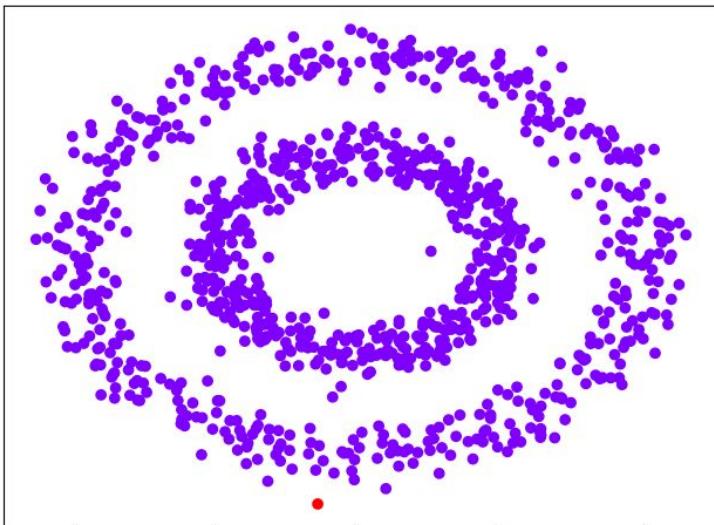
#cluster = 3



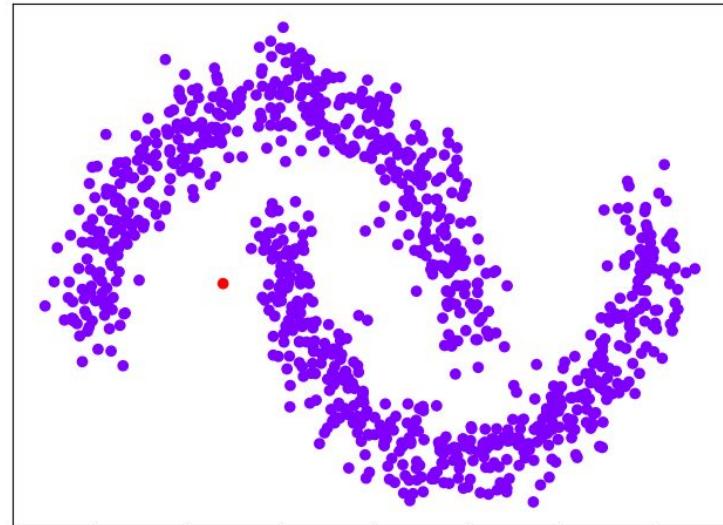
# AGNES — Single Linkage

- Weakness: Very susceptible to noise → "Chaining"
  - A single point may cause two clusters get merged

#cluster = 2

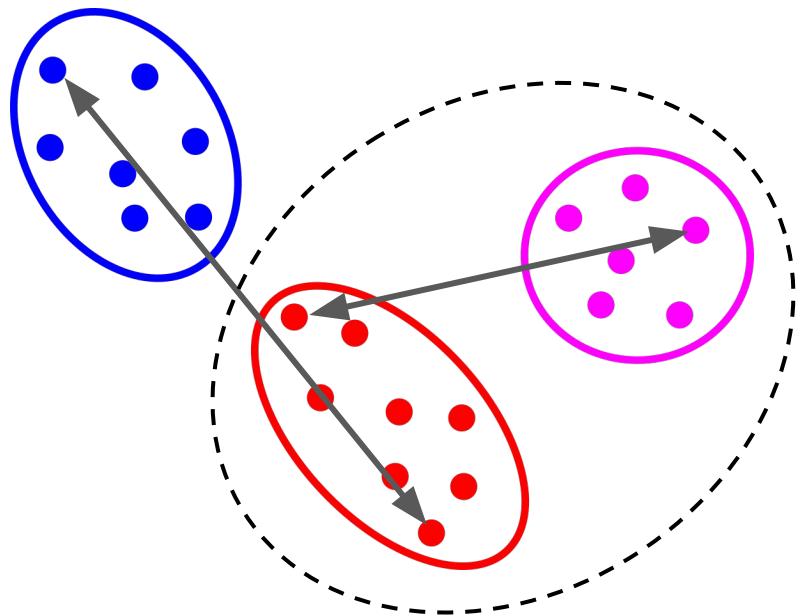


#cluster = 2



# AGNES — Complete Linkage

- Complete Linkage Clustering
  - Distance between clusters = **maximum distance** between two points from each cluster

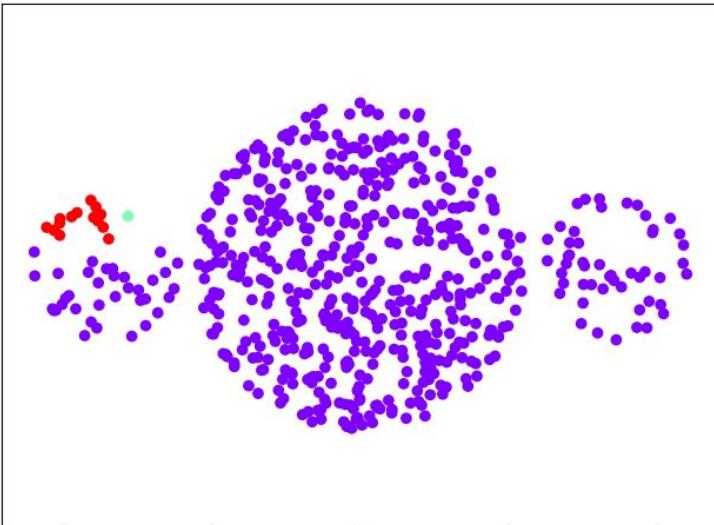


$$d_{complete}(C_i, C_j) = \max_{p \in C_i, q \in C_j} d(p, q)$$

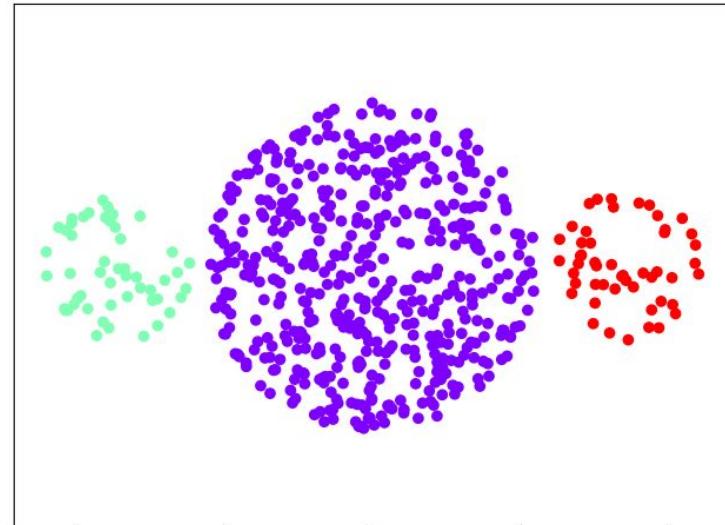
# AGNES — Complete Linkage

- Strength: Less susceptible to noise or outliers

Single Linkage, #cluster = 3



Complete Linkage, #cluster = 3

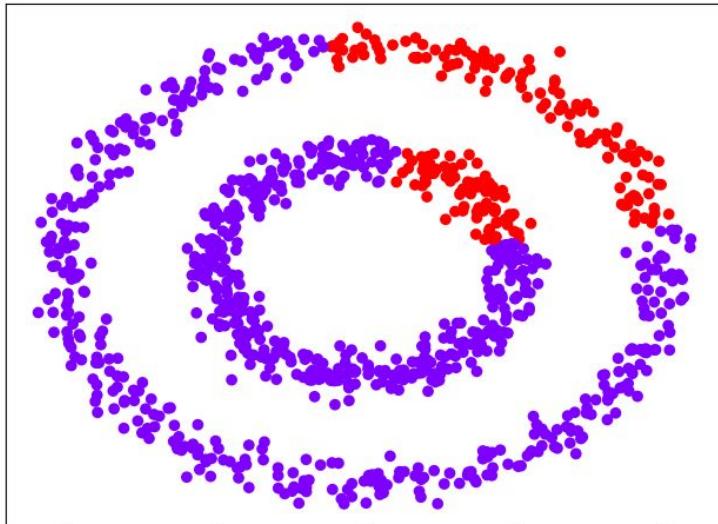


# AGNES — Complete Linkage

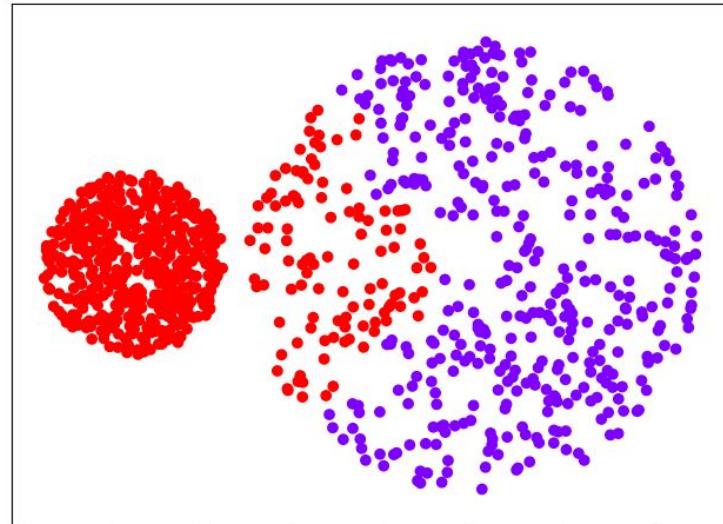
- Weaknesses

- Bias towards globular clusters
- Tends to break large clusters

#cluster = 2

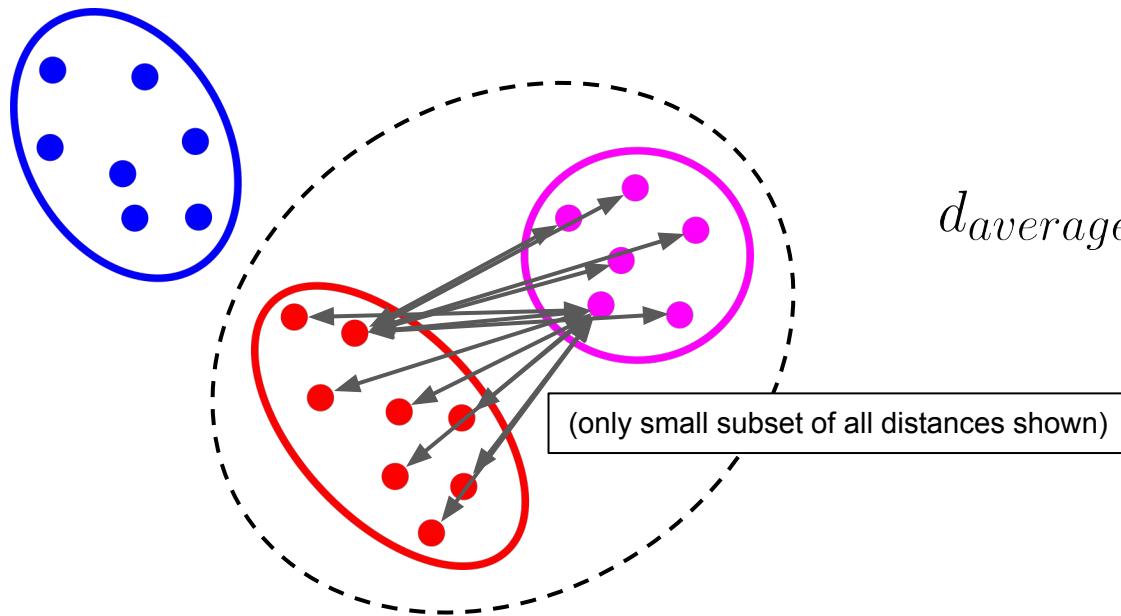


#cluster = 2



# AGNES — Average Linkage

- Complete Linkage Clustering (compromise between single and complete linkage)
  - Distance between clusters = **average distance** between two points from each cluster



$$d_{\text{average}}(C_i, C_j) = \underset{p \in C_i, q \in C_j}{\text{avg}} d(p, q)$$

# AGNES — Linkage Alternatives

- **Centroid linkage**

- Distance between clusters = distance between the centroids of each cluster

$$d_{centroid}(C_i, C_j) = d(\underbrace{m_i, m_j}_{\text{centroid of cluster } i \text{ and } j (\text{m for mean})})$$

- **Ward linkage**

$$d_{Ward}(C_i, C_j) = \overbrace{\sum_{k \in C_i \cup C_j} ||x_k - m_{ij}||^2}^{\text{Variance of } C_{ij}} - \overbrace{\sum_{k \in C_i} ||x_k - m_i||^2}^{\text{Variance of } C_i} - \overbrace{\sum_{k \in C_j} ||x_k - m_j||^2}^{\text{Variance of } C_j}$$

$$= \frac{n_i n_j}{n_i + n_j} ||m_i - m_j||^2$$

$n_i$  = #points in cluster  $C_i$

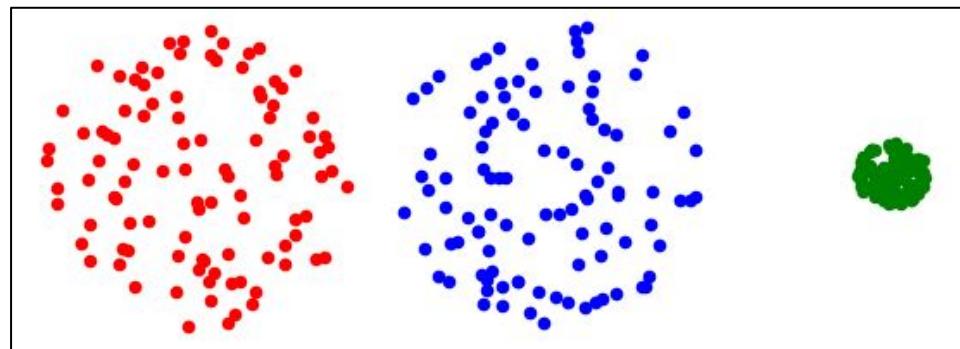
# Ward Linkage — Intuition

$$d_{Ward}(C_i, C_j) = \underbrace{\sum_{k \in C_i \cup C_j} ||x_k - m_{ij}||^2}_{\text{Variance of } C_{ij}} - \underbrace{\sum_{k \in C_i} ||x_k - m_i||^2}_{\text{Variance of } C_i} - \underbrace{\sum_{k \in C_j} ||x_k - m_j||^2}_{\text{Variance of } C_j}$$

- Example for Ward Linkage
  - Each blob: 100 data points

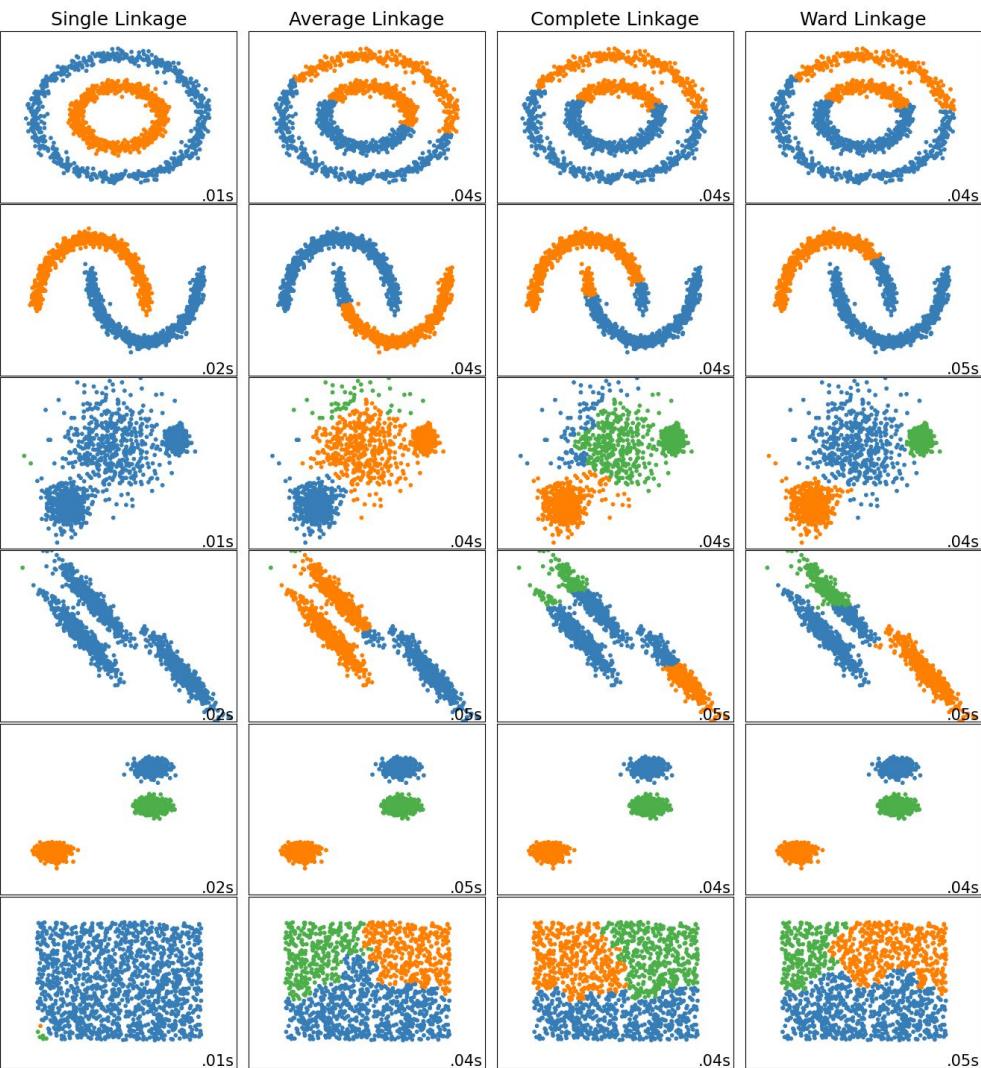
$$d_{Ward}(\text{Red}, \text{Blue}) = 1,635 - 195 - 200 = \mathbf{1,240}$$

$$d_{Ward}(\text{Blue}, \text{Green}) = 1,450 - 200 - 10 = \mathbf{1,240}$$



# AGNES

- Linkage comparison



# Quick Quiz

Which linkage method has intuitively the **highest chance** of returning a clustering where the dendrogram has a **depth of  $N-1$** ?

A

Single

B

Complete

C

Average

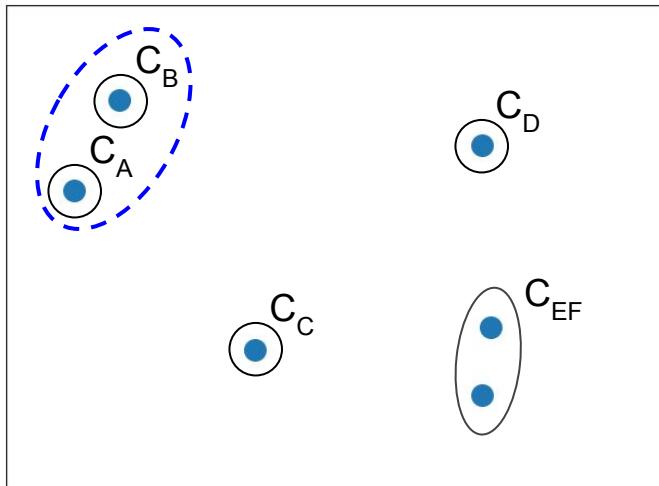
D

No difference

# AGNES — Implementation

- Distance matrix after merging Cluster  $C_E$  and  $C_F$  + Average Linkage

Clustering after merging  $C_E$  and  $C_F$  to  $C_{EF}$

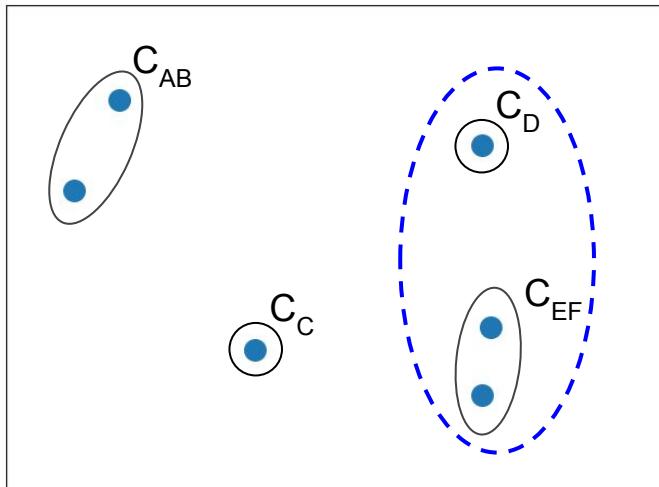


	$C_A$	$C_B$	$C_C$	$C_D$	$C_{EF}$
$C_A$	$\infty$	<b>2.25</b>	5.32	9.06	9.64
$C_B$		$\infty$	6.08	7.85	9.54
$C_C$			$\infty$	6.73	4.92
$C_D$				$\infty$	4.76
$C_{EF}$					$\infty$

# AGNES — Implementation

- Distance matrix after merging Cluster  $C_A$  and  $C_B$  + Average Linkage

Clustering after merging  $C_A$  and  $C_B$  to  $C_{AB}$

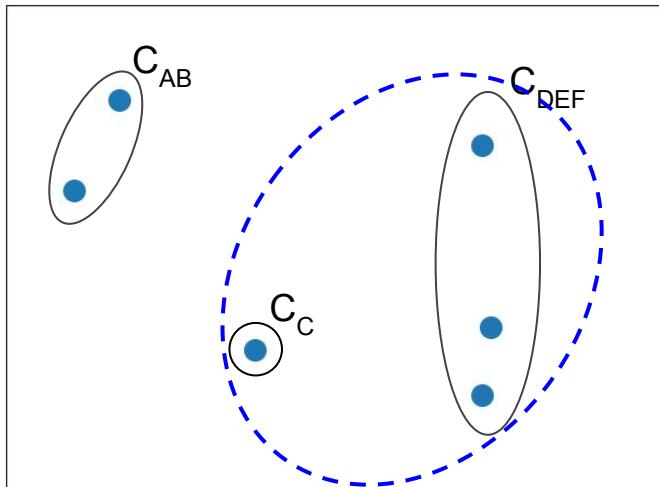


	$C_{AB}$	$C_C$	$C_D$	$C_{EF}$
$C_{AB}$	$\infty$	5.70	8.45	9.59
$C_C$		$\infty$	6.73	4.92
$C_D$			$\infty$	<b>4.76</b>
$C_{EF}$				$\infty$

# AGNES — Implementation

- Distance matrix after merging Cluster  $C_C$  and  $C_{DEF}$  + Average Linkage

Clustering after merging  $C_D$  and  $C_EF$  to  $C_{DEF}$

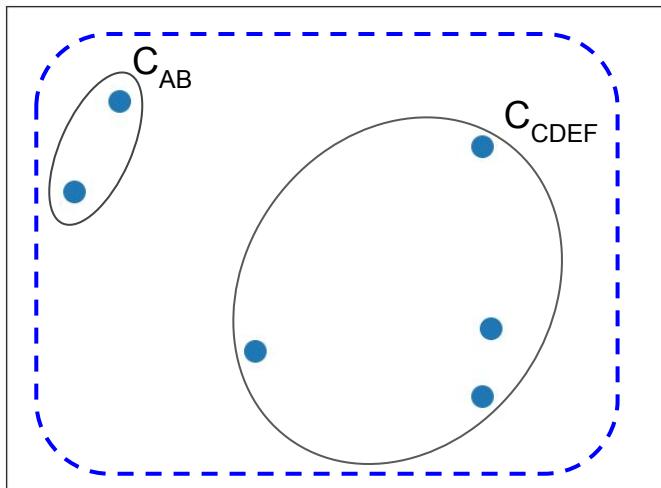


	$C_{AB}$	$C_C$	$C_{DEF}$
$C_{AB}$	$\infty$	5.70	9.21
$C_C$		$\infty$	<b>5.51</b>
$C_{DEF}$			$\infty$

# AGNES — Implementation

- Distance matrix after merging Cluster  $C_{AB}$  and  $C_{CDEF}$  + Average Linkage

Clustering after merging  $C_C$  and  $C_{DEF}$  to  $C_{CDEF}$

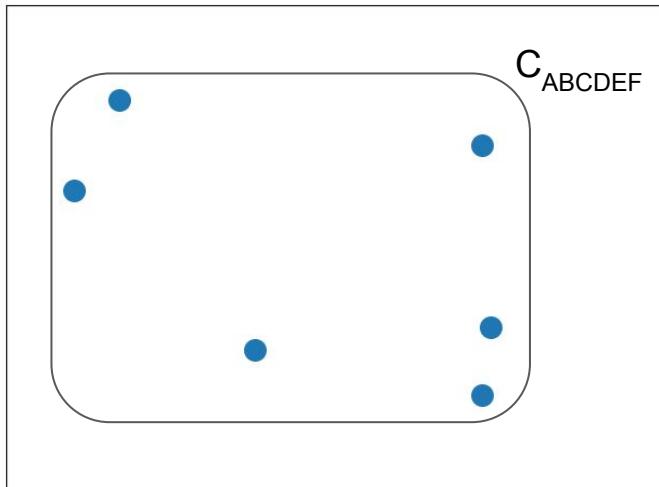


	$C_{AB}$	$C_{CDEF}$
$C_{CDEF}$	$\infty$	<b>8.33</b>

# AGNES — Implementation

- Distance matrix after merging Cluster  $C_{AB}$  and  $C_{CDEF}$  + Average Linkage

Clustering after merging  $C_{AB}$  and  $C_{CDEF}$  to  $C_{ABCDEF}$



	$C_{ABCDEF}$
$C_{ABCDEF}$	$\infty$

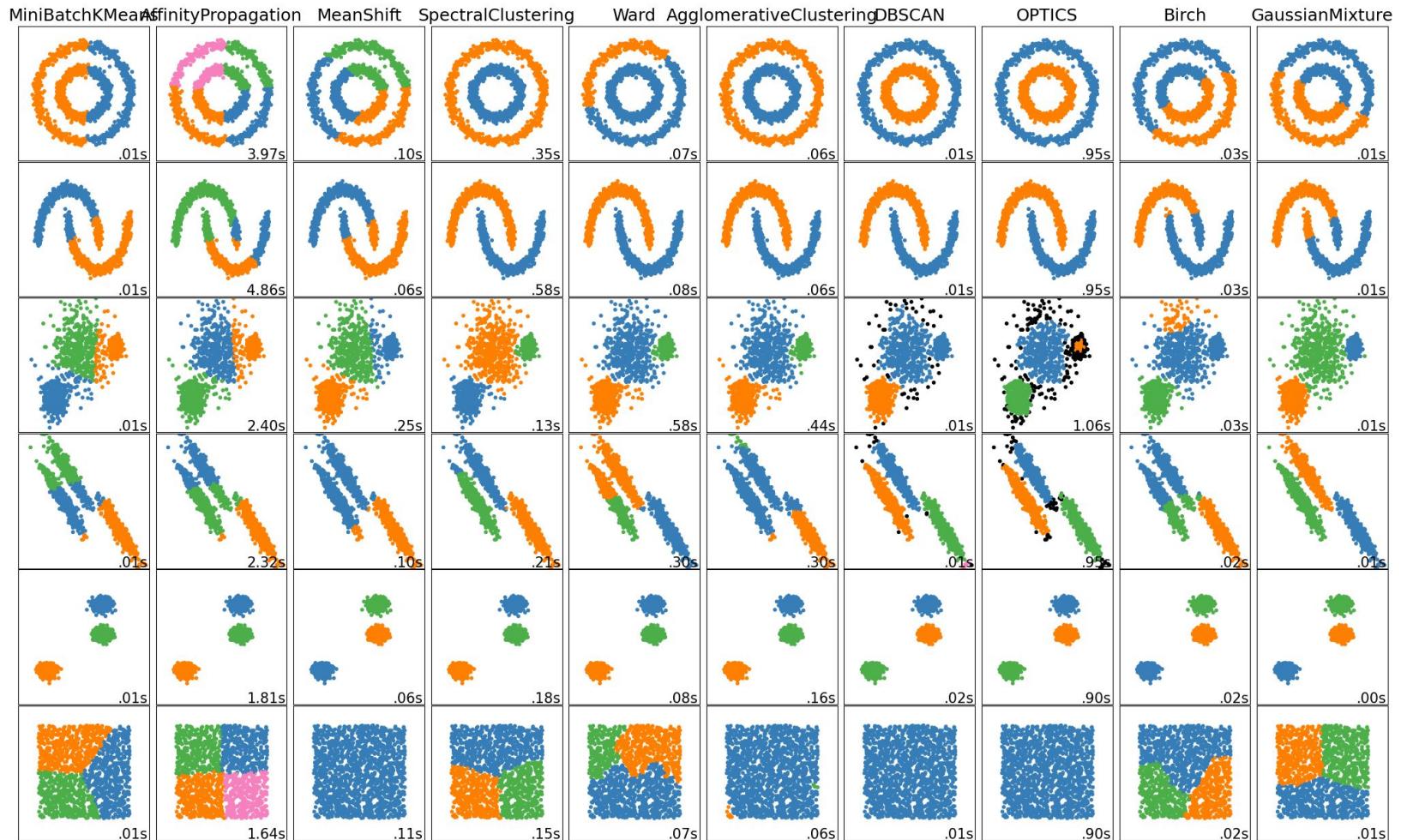
→ Done!

# AGNES — Complexity Analysis

- Space Complexity:  $O(N^2)$ 
  - Storing distance matrix
- Time Complexity
  - Baseline:  $O(N^3)$  — ( $N-1$ ) steps, each step  $O(N^2)$  to scan distance matrix
  - Using more sophisticated data structures, e.g, heap or priority queue:  $O(N^2 \log N)$
  - Special optimization for Single Linkage Clustering:  $O(N^2)$

# DIANA — DIvisive ANAlysis

- Top-Down Hierarchical Clustering
  - Start with all points forming one cluster
  - Recursively split one cluster until all clusters have size 1
- Challenge:  $2^n$  ways to split a cluster with  $n$  points
  - Heuristics needed to restrict search space
  - Generally slower and less common than AGNES
- Cases where DIANA can perform better
  - No complete clustering needed → early stopping
  - Splitting can utilize global knowledge (merging based on local knowledge only)



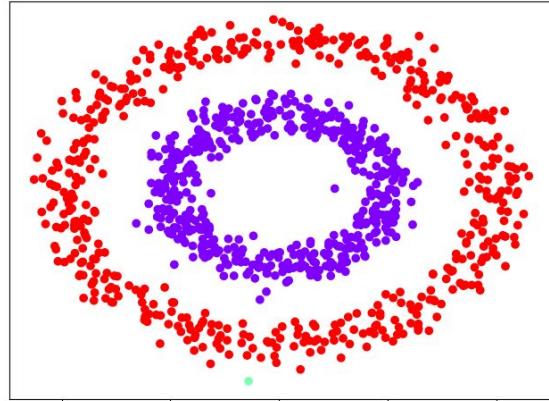
Source: <https://scikit-learn.org/stable/modules/clustering.html>

# Outline

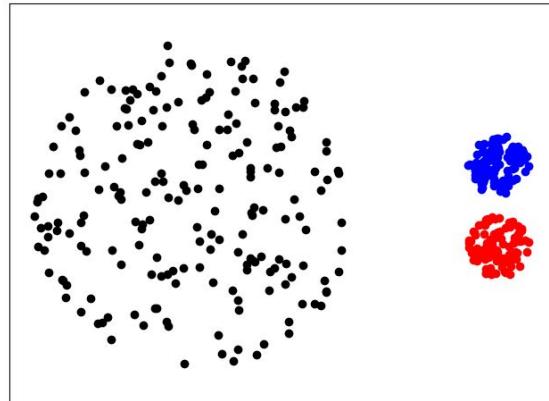
- **Clustering**
  - Overview
  - Concepts
  - Applications
- **Clustering algorithms**
  - K-Means
  - DBSCAN
  - Hierarchical Clustering
- **Cluster Evaluation**

# Cluster Evaluation

- Problem 1: Just eyeballing the clustering is rarely possible
  - High-dimensional data ( $\geq 3$  dimensions) difficult to impossible to visualize
  - Difficult to assess "nature" of clusters a-priori (e.g., variations in shape, size, density, etc)
  - Presence and distribution of noise or outliers



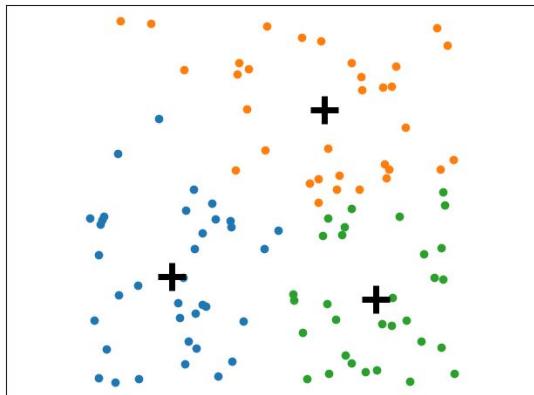
Your data usually does not look like this



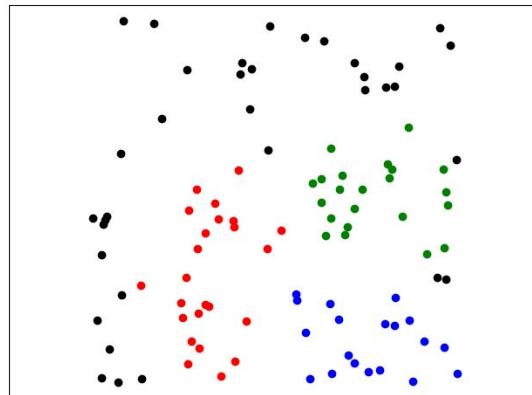
# Cluster Evaluation

- Problem 2: Clustering algorithms will always find some clusters
  - Example: K-Means, DBSCAN and AGNES applied to random data

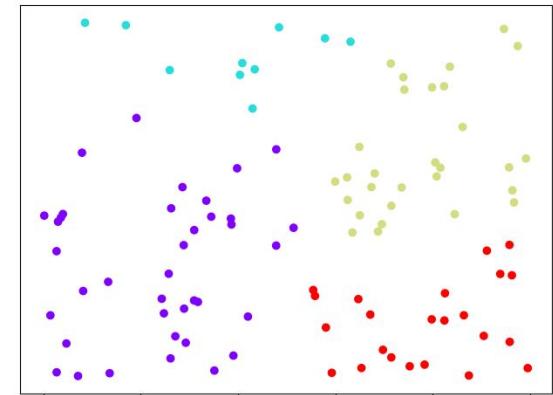
K-Means



DBSCAN



AGNES

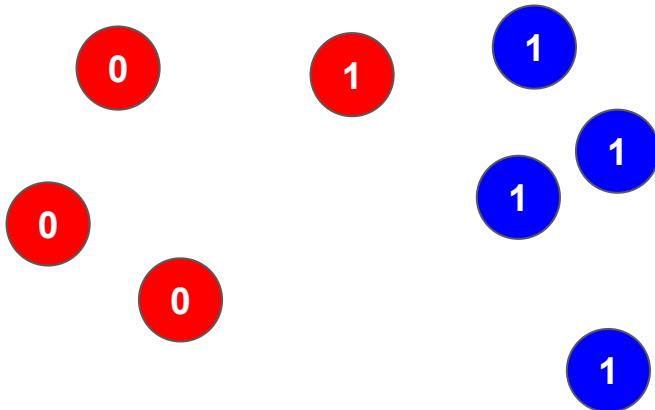


# Cluster Evaluation

- Purpose of cluster evaluation
  - Comparing the results of different clustering algorithms
  - Comparing the results of a clustering algorithm with different parameters
  - Minimizing the effects of noise on the clustering
- Getting a sense of the "goodness" of a clustering
- Two main approaches
  - External quality measures: evaluate a clustering against a ground truth (if available)
  - Internal quality measures: evaluate clustering from the data itself

# Cluster Evaluation — External Quality Measures

- Ground truth: Labeled data
  - Labels indicate that two points "belong together"
  - If cluster reflect this → good clustering



Cluster	Label
Red	0
Red	0
Red	0
Red	1
Blue	1

# External Quality Measures — Cluster Purity

- Cluster purity  $P$

- $N$ : #points,  $C$ : set of cluster,  $L$ : set of labels

$$P = \frac{1}{N} \sum_{c \in C} \overbrace{\max_{l \in L} |c \cap l|}^{\text{\#points with most common label } l \text{ in cluster } c}$$

Purity for example:

$$P = \frac{1}{8}(3 + 4) = 0.875$$

- Limitations

- Purity does not penalize having many cluster  
→  $P=1$  easy to achieve with all cluster containing single point

Cluster	Label
Red	0
Red	0
Red	0
Red	1
Blue	1

# External Quality Measures: Information Retrieval Metrics

- Established metrics from classification tasks

- **TP** — true positives  
same cluster, same label  
(A/B, A/C, B/C, E/F, ..., G/H)
- **TN** — true negatives  
different clusters, different labels  
(A/E, A/F, A/G, A/H, B/E, ..., C/H)
- **FP** — false positives  
same cluster, different labels  
(A/D, B/D, C/D)
- **FN** — false negatives  
different cluster, same label  
(D/E, D/F, D/G, D/H)

For the example:

- **TP = 9**
- **TN = 12**
- **FP = 3**
- **FN = 4**

ID	Custer	Label
A	Red	0
B	Red	0
C	Red	0
D	Red	1
E	Blue	1
F	Blue	1
G	Blue	1
H	Blue	1

# External Quality Measures — Information Retrieval Metrics

- Rand Index RI
  - Reflects accuracy

$$RI = \frac{TP + TN}{TP + FP + FN + TN}$$

$$RI_{example} = 0.75$$

- Precision P, Recall R, F1-Score

$$P = \frac{TP}{TP + FP}$$

$$R = \frac{TP}{TP + FN}$$

$$F1 = \frac{2 \cdot P \cdot R}{P + R}$$

$$P_{example} = 0.75$$

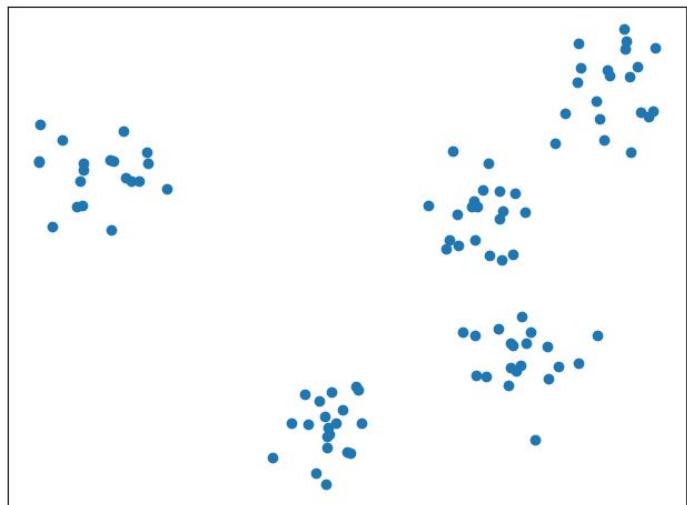
$$R_{example} = 0.69$$

$$F1_{example} = 0.72$$

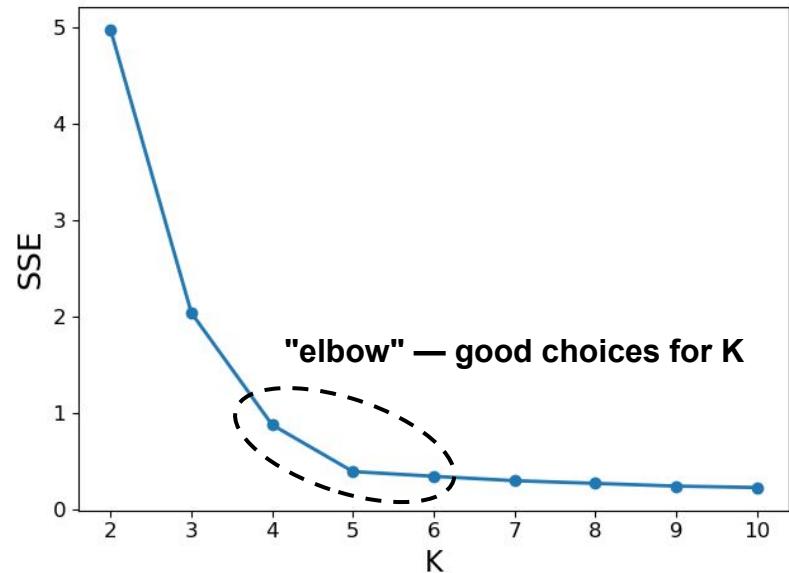
- ...and others using TP, TN, FP, FN

# Internal Quality Measures — SSE

- Use SSE to select number of clusters



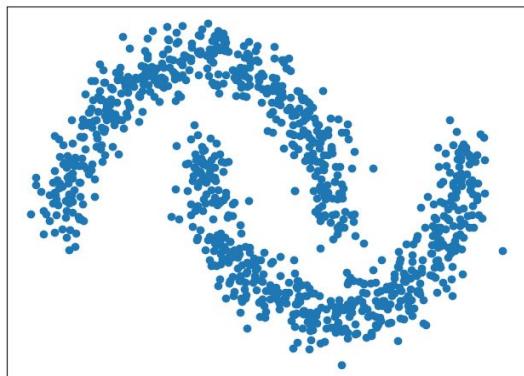
input data



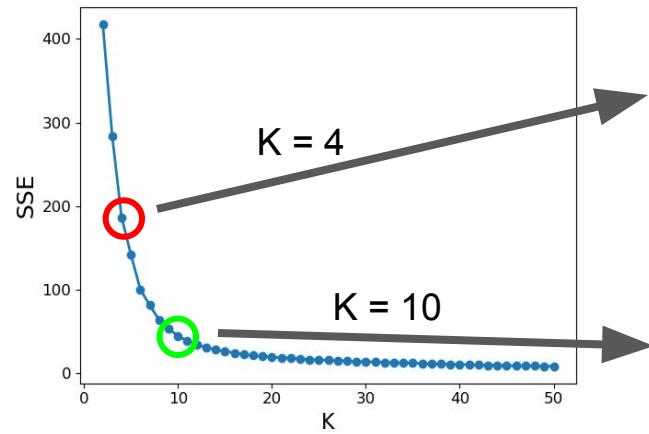
SSE for different K

# Internal Quality Measures — SSE

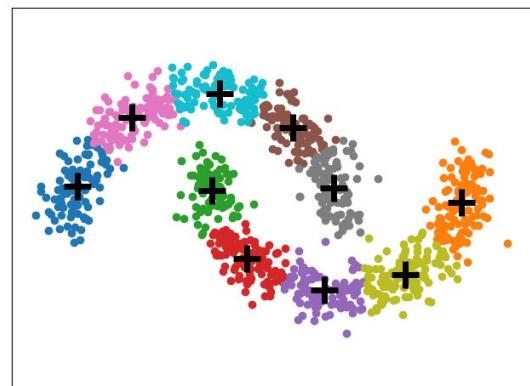
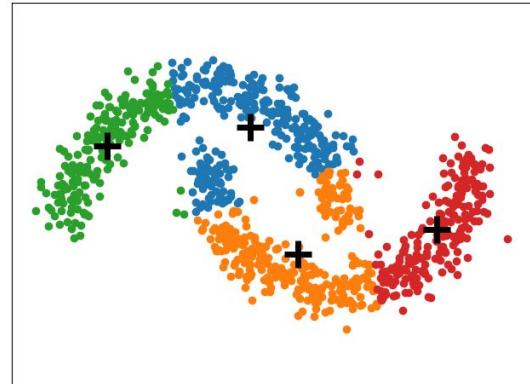
- Also applicable to more complicated data
  - But inherently "favors" globular clusters



input data

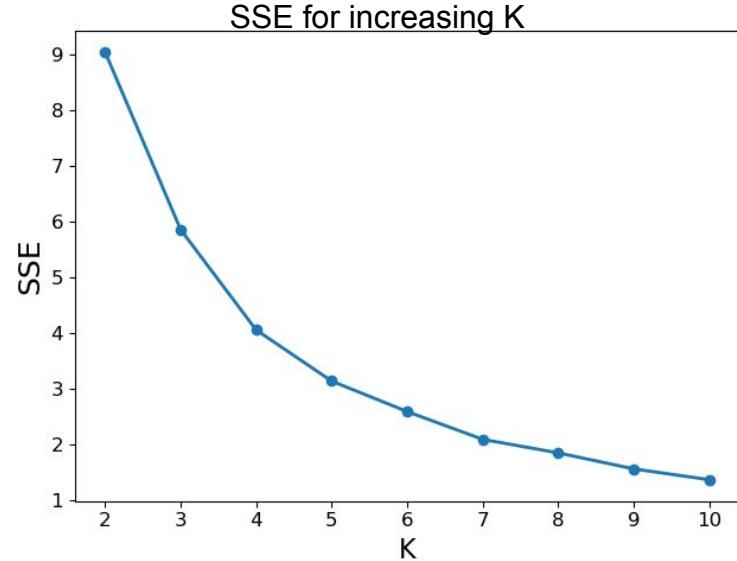
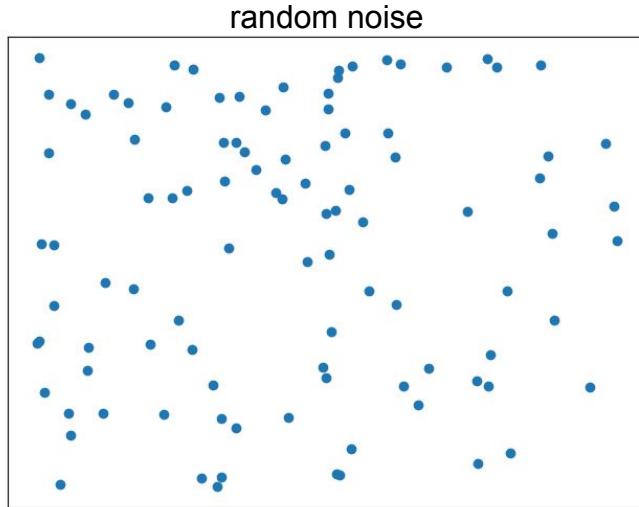


SSE for different K



# Internal Quality Measures — SSE

- Limitation of SSE as quality measure
  - SSE does not penalize large number of clusters
  - SSE decreases for increasing cluster counts
  - Applicable beyond K-Means, but less intuitive interpretation (in case of non-globular clusters)



# Quick Quiz

If  $K \ll N$ , can  $SSE=0$ ?

**Why** or **why not?**

**A**

Yes

**B**

No

# Internal Quality Measures — Silhouette Coefficient

- Intuition: A good clustering has

- High inter-cluster distances
- Low intra-cluster distances

- For each data point  $x$ , define

- **Cohesion**  $a(x)$ : average distance to points in the same cluster
- **Separation**  $b(x)$ : minimum average distance to points in a different cluster
- **Silhouette:**

$$s(x) = \frac{b(x) - a(x)}{\max\{a(x), b(x)\}}, \text{ if } |C_x| > 1$$

$$x \in C_X$$

$$a(x) = \frac{1}{|C_X - 1|} \sum_{p \in C_X, p \neq x} d(x, p)$$

the smaller, the better

$$b(x) = \min_{X \neq K} \frac{1}{|C_K|} \sum_{p \in C_K} d(x, p)$$

the larger, the better

$$s(x) = 0, \text{ if } |C_x| = 1$$

# Internal Quality Measures — Silhouette Coefficient

$$s(x) = \frac{b(x) - a(x)}{\max\{a(x), b(x)\}}$$

- Interpretation

$$-1 \leq s(x) \leq +1$$

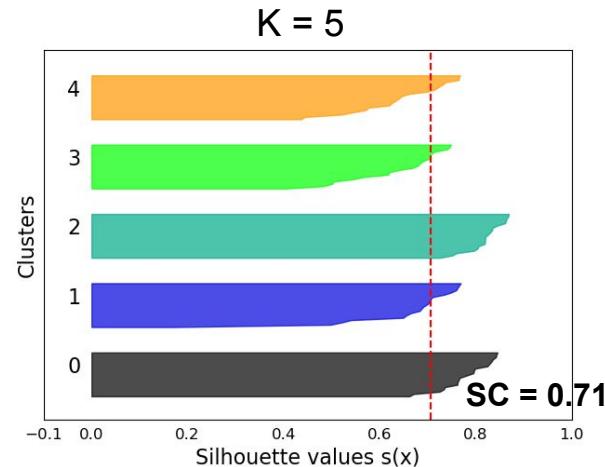
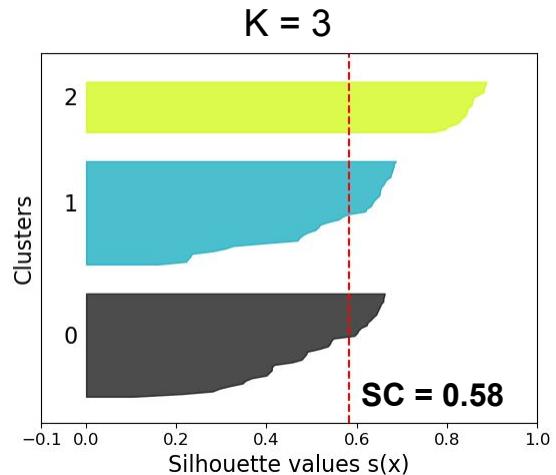
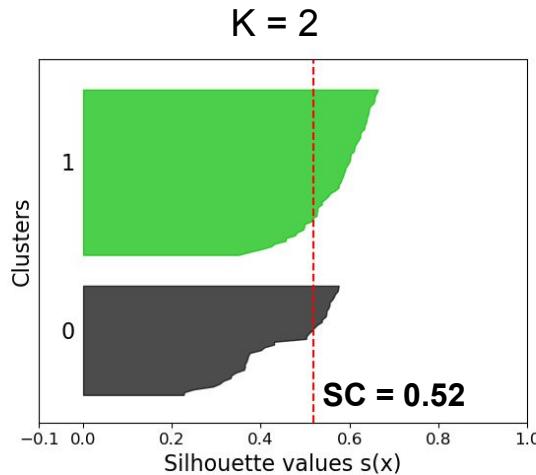
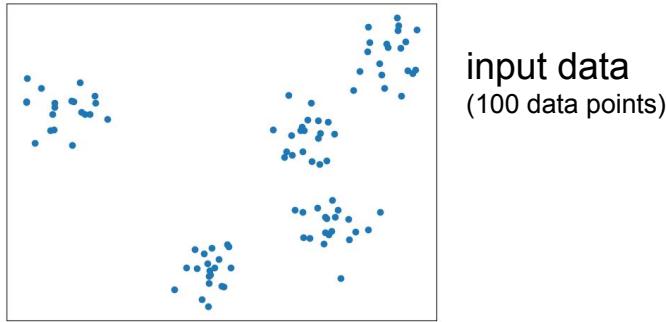
**BAD**                            **GOOD**

- Silhouette Coefficient SC:

$$SC = \frac{1}{N} \sum_{i=1}^N s(x_i)$$

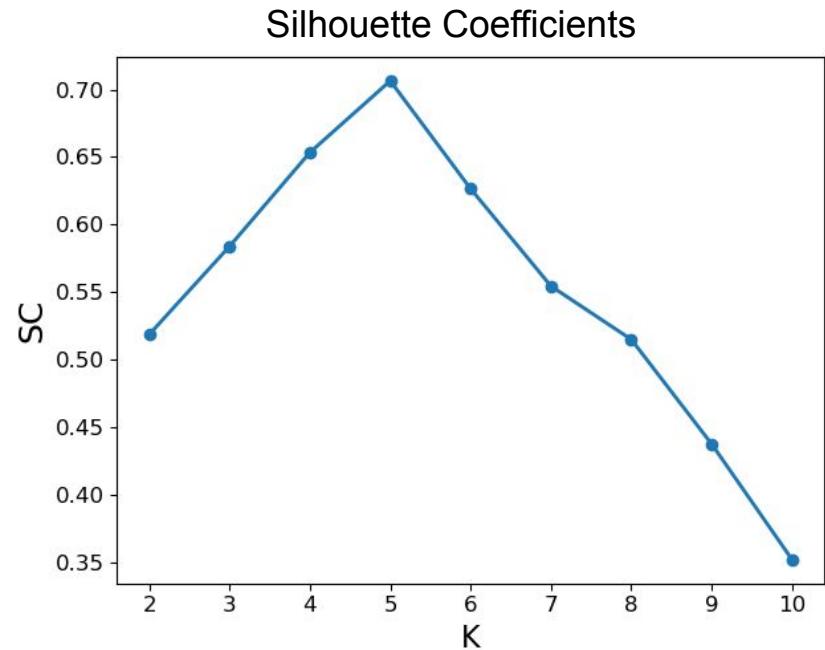
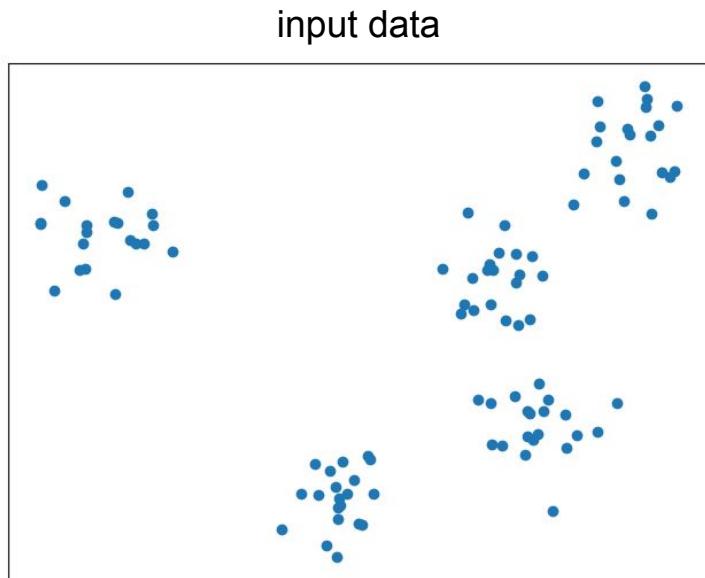
# Internal Quality Measures — Silhouette Coefficient

- Example: K-Means



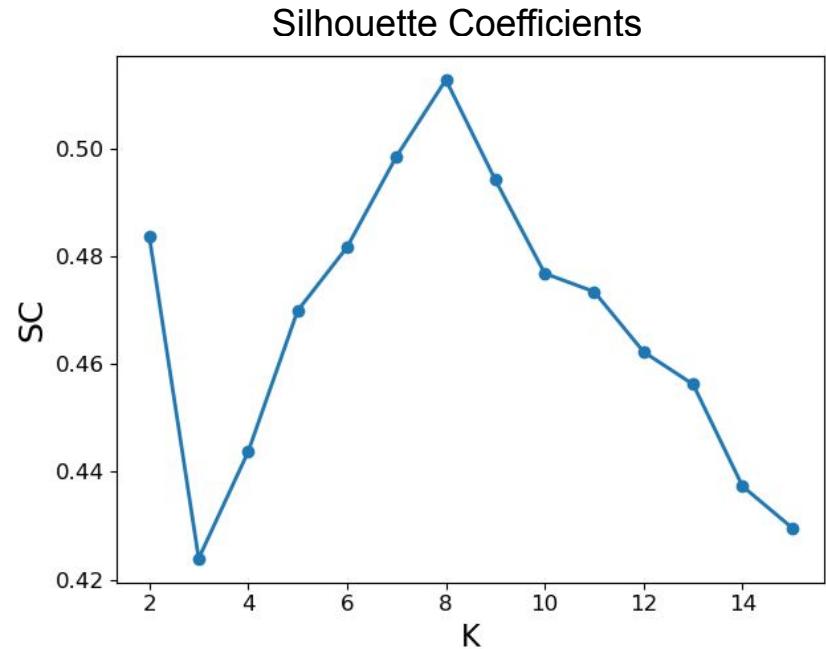
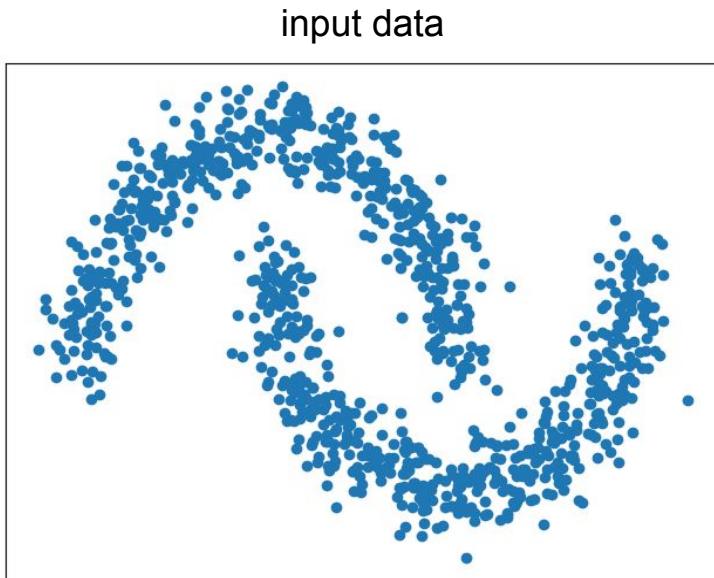
# Internal Quality Measures — Silhouette Coefficient

- Example: K-Means



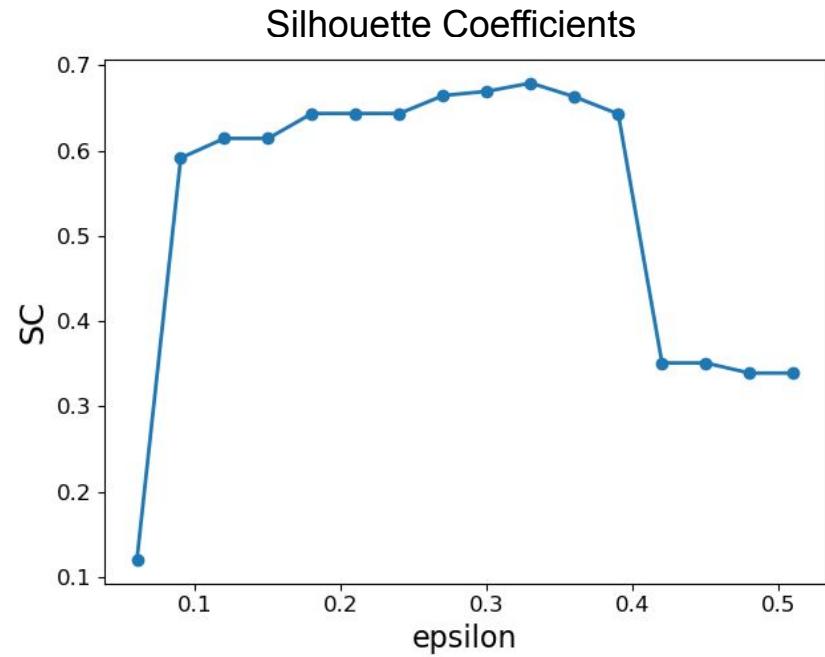
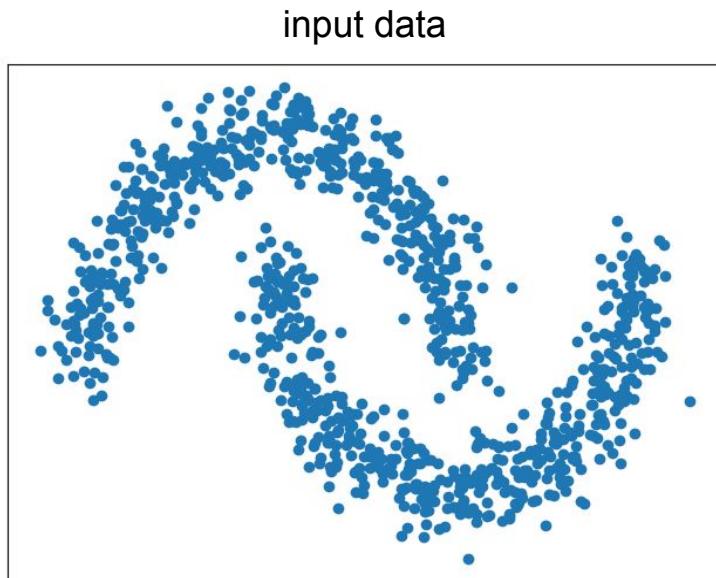
# Internal Quality Measures — Silhouette Coefficient

- Example: K-Means



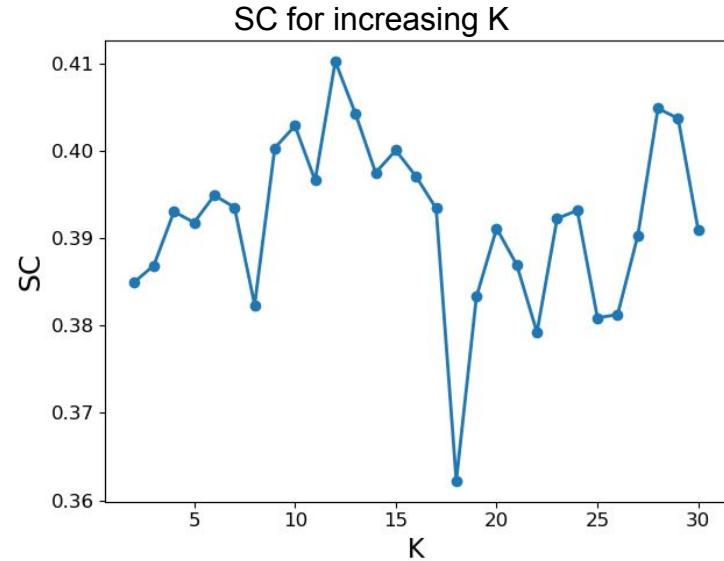
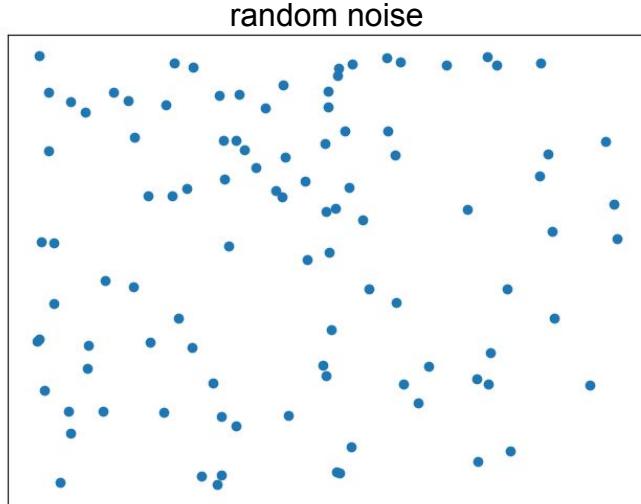
# Internal Quality Measures — Silhouette Coefficient

- Example: DBSCAN



# Internal Quality Measures — Silhouette Coefficient

- SC for random data (K-Means)



**Note:** DBSCAN on random data quickly results in 0 or 1 cluster, for which SC is not defined

# Cluster Evaluation — Comments

- In practice, choice of "best" clustering often more pragmatic:
  - Fixed number of clusters (problematic for DBSCAN)
  - Parameters defined by tasks  
(e.g., "areas with more than 5 McDonalds within a radius of 500m")
  - Maximum, minimum, or average size of clusters
  - Focus in individual clusters instead of whole clustering  
(e.g., biggest/smallest cluster, cluster that contains certain points)
  - Set  $K$  "too high" and merge later if needed
  - ...

# Outline

- Clustering
  - Overview
  - Concepts
  - Applications
- Clustering algorithms
  - K-Means
  - DBSCAN
  - Hierarchical Clustering
- Cluster Evaluation

# Summary — Clustering

- **Clustering:** Finding patterns (here: cluster/groups) in unlabeled data
  - Very important concept in data mining
  - Wide range of clustering algorithms with varying characteristics (pros & cons)  
→ No "one-size-fits-all" algorithm
- **Discussed algorithms:** K-Means, DBSCAN, AGNES
  - Focus on the — arguably intuitive — conceptual inner workings
  - Emphasis on algorithms' strength and weaknesses
  - Many tweaks and optimizations to improve performance
- **Major challenge:** cluster evaluation
  - No fool-proof method to find the best algorithm or parameters (at least for unlabeled data)

# Solutions to Quick Quizzes

- Slide 11: A
- Slide 15: A and/or B
- Slide 26: A
- Slide 46: A (in case of duplicates and  $K < \#\text{unique points}$ )

# **CS5228: Knowledge Discovery and Data Mining**

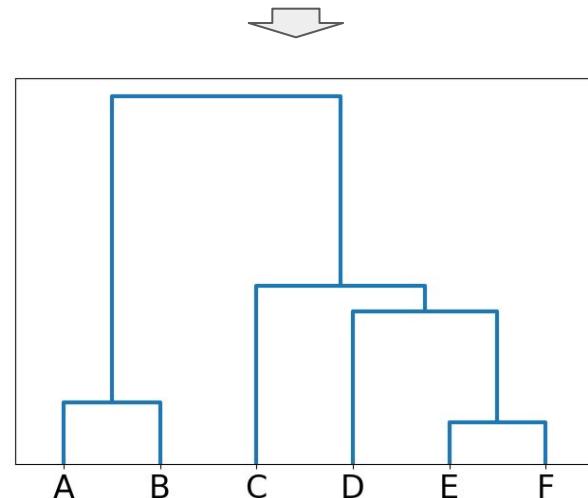
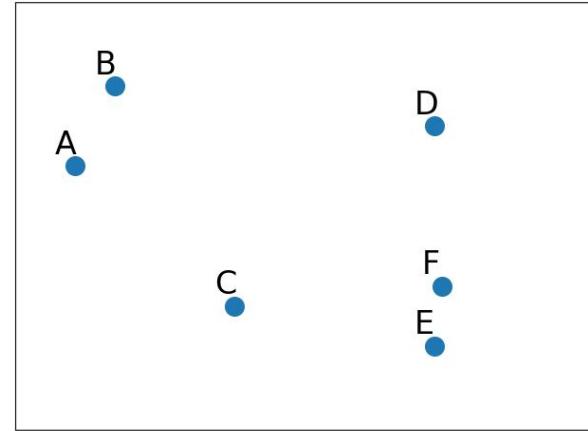
## Lecture 4 — Association Rule Mining

# Course Logistics — Update

- Assignment 1
  - Submission deadline: Thu, Sep 12 (11.59 pm)
  - Honor code: don't cheat, don't copy, don't steal, don't plagiarize, etc.
  - Don't forget to check Discussion and Errata page Canvas
- Project
  - Teams finalized (please get all in contact)
  - Kaggle competition launched

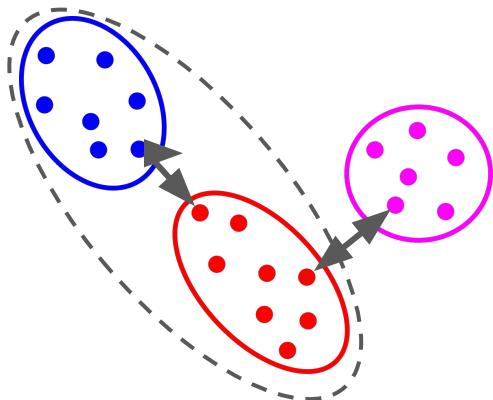
# Recap — Hierarchical Clustering

- AGNES (AGglomerative NESting)
  - Start with  $N$  clusters, one for each data point
  - Iteratively merge nearest clusters into one
  - Stop if all data points are in one cluster
- Core questions: How to calculate distances between clusters?

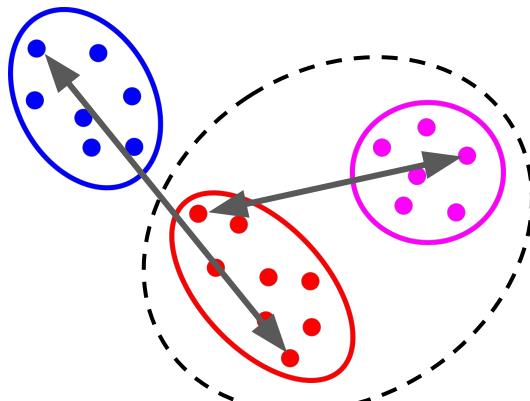


# Recap — Linkage Methods

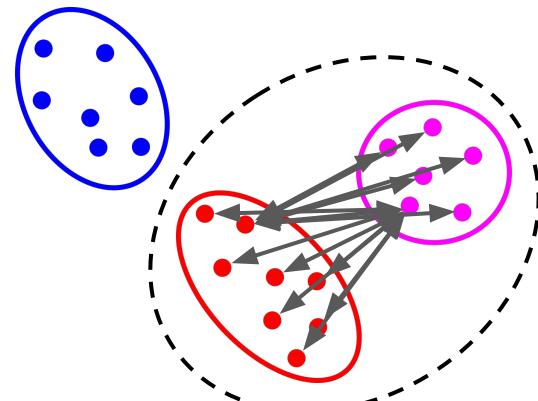
Single Linkage



Complete Linkage



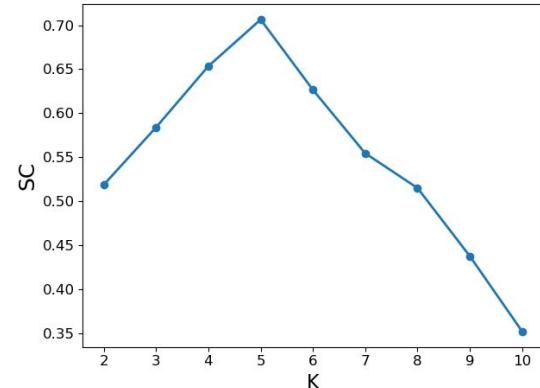
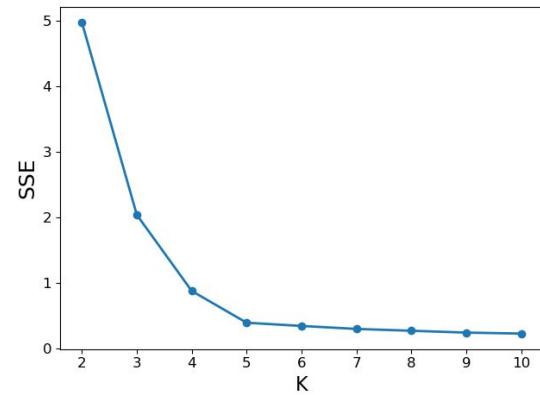
Average Linkage



# Recap — Cluster Evaluation

- If ground truth available: external quality measures
  - Cluster purity
  - TP/TN/FP/FN-based metrics (e.g., Rand index)
- Unlabelled data: internal quality measures
  - Elbow method using SSE
  - Silhouette Coefficient (SC)

} favor blob-like clusters
- Cluster evaluation in practice (unlabeled data)
  - No fool-proof method to find "best" clustering
  - Decision on clustering often rather pragmatic



# Recap — Clustering as a Means to an End

- Clustering as part of EDA
  - SSE plot, SC plot, dendrogram, etc. can provide useful insights into the data
  - Little requirements — "only" similarity/distance between data points needed
  - In the gray area between (simple) EDA and proper data analysis

- Clustering for data preprocessing — example:
    - Cluster persons according to their height into K=10 groups
    - Assign each person new height = centroid of cluster
- 
- form of aggregation or  
binning & smoothing

# Outline

- **Association Rule Mining**
  - Overview
  - Applications
- Definitions
- Algorithms
  - Brute-Force
  - Apriori
- Discussion

# Association Rules — Basic Setup

- Input database:
  - Set of **transactions**
  - Transaction = set of **items**
- Output: **Association Rules**
  - Rules predicting the occurrence of some items based on occurrence of other items

**antecedent → consequent**

$$\{\text{item}_2, \text{item}_3\} \rightarrow \{\text{item}_5\}$$

$$\{\text{item}_1\} \rightarrow \{\text{item}_3\}$$

TID	Items
1	item <sub>1</sub> , item <sub>2</sub> , item <sub>3</sub> , item <sub>4</sub> , item <sub>5</sub>
2	item <sub>2</sub> , item <sub>3</sub> , item <sub>5</sub>
3	item <sub>1</sub> , item <sub>4</sub> , item <sub>5</sub>
4	item <sub>2</sub> , item <sub>3</sub> , item <sub>5</sub> , item <sub>6</sub> , item <sub>7</sub>
5	item <sub>1</sub> , item <sub>3</sub> , item <sub>5</sub> , item <sub>7</sub>
...	

# Applications — Market Basket Analysis

- Understanding customers shopping behavior
  - **Items:** products in supermarket/store
  - **Transaction:** baskets at check-out
- Interesting rules:
  - Customers who buy {a, b} also tend to buy {x, y}
  - Example: {cereal} → {milk}
- Purpose
  - Shelf management / item placement
  - Promotions (product bundles)
  - Recommendations
  - Pricing strategies

TID	Items
1	bread, yogurt
2	bread, milk, cereal, eggs
3	yogurt, milk, cereal, cheese
4	bread, yogurt, milk, cereal
5	bread, yogurt, milk, cheese
...	

# Applications — Medical Data Analysis

- Diagnosis Support Systems

- Items: symptoms, diseases
- Transaction: patient's medical history

ID	Items
1	covid-19, anosmia, cough, fatigue
2	flu, anosmia, headache
3	covid-19, anosmia, headache, fatigue, fever
4	covid-19, flu, anosmia, fatigue
5	flu, depression, fatigue, fever, headache
...	



$\{\text{anosmia, fatigue}\} \rightarrow \{\text{covid-19}\}$

- ADR discovery (adverse drug reaction)

- Items: drugs, reactions/symptoms
- Transaction: patient's medical history

ID	Items
1	$d_1, d_2, d_3, \text{rash, vomit}$
2	$d_1, d_3, \text{headache, nausea, rash,}$
3	$d_2, d_3, \text{nausea, vomit}$
4	$d_1, \text{nausea, rash, vomit}$
5	$d_3, d_4, \text{headache, depression}$
...	



$\{d_1\} \rightarrow \{\text{rash}\}$

# Applications — Census Data Analysis

- Getting insights into a population
  - Items: demographic data
  - Transaction: census record
- Interesting rules:
  - Correlations among groups of people based on shared demographics
  - Example: {uni-grad,  $\geq 30$ }  $\rightarrow$  {high-income}
- Purpose
  - Policy & decision making
  - Resource allocation
  - Urban planning

TID	Items
1	female, $\geq 25$ , uni-grad, hdb, single, high-income
2	male, $\geq 25$ , uni-grad, hdb, single, mid-income
3	male, $\geq 25$ , uni-grad, hdb, condo, high-income
4	male, $\geq 30$ , uni-grad, condo, married, high-income
5	female, $\geq 30$ , uni-grad, condo, married, high-income
...	

# Applications — Behavior Data Analysis

- User preferences & linkings
  - Items: movies, songs, books, etc.
  - Transaction: viewing/listening/reading history
- Interesting rules (movies):
  - Viewer who watched movies {a, b} also watched movies {x, y}
  - Example: {Jaws}→{It}
- Purpose
  - Recommendation systems

TID	Items
1	Jaws, Halloween, Scream, It
2	Alien, Jaws, Scream, It
3	Tenet, Inception, Interstellar
4	Jaws, Halloween, It
5	Alien, Tenet, Jaws, It
...	

# Association Rules — Problem Statement

- Association rules are not "hard" rules
  - e.g.,  $\{\text{cereal}\} \rightarrow \{\text{milk}\}$  does not mean that customers always buy milk when buying cereal
  - each possible combination (e.g.,  $\{\text{yogurt, bread}\} \rightarrow \{\text{milk}\}$ ) is potential association rule
- Given  $d$  unique items  $\rightarrow 3^d - 2^{d+1} + 1$  rules
  - $d = 6 \rightarrow 602$  possible rules!
- Association Rule Mining
  - Finding **interesting/significant** association rules
  - Finding such rules **efficiently**

TID	Items
1	bread, yogurt
2	bread, milk, cereal, eggs
3	yogurt, milk, cereal, cheese
4	bread, yogurt, milk, cereal
5	bread, yogurt, milk, cheese
...	

# Outline

- **Association Rule Mining**
  - Overview
  - Applications
- **Definitions**
- **Algorithms**
  - Brute-Force
  - A-Priori
- **Discussion & Summary**

# Definitions — Itemset, K-itemset

- **Itemset**

- A subset of items

{bread}, {yogurt}, {bread, yogurt}, {milk}, {cereal},  
{eggs}, {bread, milk}, {bread, milk, cereal}, ...

- **K-itemset**

- An itemset containing k items, e.g., k=3:

{bread, milk, cereal}, {bread, yogurt, cheese},  
{yogurt, milk, cereal}, {yogurt, cereal, cheese},  
{milk, cereal, cheese}, {bread, milk, eggs}, ...

TID	Items
1	bread, yogurt
2	bread, milk, cereal, eggs
3	yogurt, milk, cereal, cheese
4	bread, yogurt, milk, cereal
5	bread, yogurt, milk, cheese

# Definitions — Support Count, Support (for itemsets)

- **Support count SC**
  - Number of transactions containing an itemset
  - e.g.,  $SC(\{\text{bread, yogurt, milk}\}) = 2$
- **Support S**
  - Fraction of transactions containing an itemset
  - e.g.,  $S(\{\text{bread, yogurt, milk}\}) = 2/5$

TID	Items
1	bread, yogurt
2	bread, milk, cereal, eggs
3	yogurt, milk, cereal, cheese
4	bread, yogurt, milk, cereal
5	bread, yogurt, milk, cheese

# Definitions — Frequent Itemset

- **Frequent itemset**

- Itemset with a support greater or equal than a minimum threshold  $minsup$
- e.g., all frequent itemsets if

$$minsup = 2/5$$

{yogurt}  
{milk}  
{cheese}  
{cereal}  
{bread}  
{bread, milk}  
{yogurt, milk}  
{bread, cereal}  
{cereal, milk}  
{bread, yogurt}  
{cereal, yogurt}  
{cereal, yogurt, milk}  
{bread, cereal, milk}  
{bread, yogurt, milk}

$$minsup = 3/5$$

{yogurt}  
{milk}  
{cereal}  
{bread}  
{bread, milk}  
{yogurt, milk}  
{cereal, milk}  
{bread, yogurt}

TID	Items
1	bread, yogurt
2	bread, milk, cereal, eggs
3	yogurt, milk, cereal, cheese
4	bread, yogurt, milk, cereal
5	bread, yogurt, milk, cheese

# Definitions — Association Rule

- **Association Rule**

- Implication expression  $X \rightarrow Y$ ,  
where X and Y are itemsets
- e.g.,  $\{yogurt, milk\} \rightarrow \{bread\}$

TID	Items
1	bread, yogurt
2	bread, milk, cereal, eggs
3	yogurt, milk, cereal, cheese
4	bread, yogurt, milk, cereal
5	bread, yogurt, milk, cheese

# Definitions — Support (for association rules)

- **Support of an association rule**

- Fraction of transactions containing all items of an association rule  $X \rightarrow Y$

$$S(X \rightarrow Y) = \frac{SC(X \cup Y)}{N}$$

#transactions

TID	Items
1	bread, yogurt
2	bread, milk, cereal, eggs
3	yogurt, milk, cereal, cheese
4	bread, yogurt, milk, cereal
5	bread, yogurt, milk, cheese
...	

$$S(\{yogurt, milk\} \rightarrow \{bread\}) = \frac{SC(\{yogurt, milk, bread\})}{N} = 2/5$$

$$S(\{yogurt, bread\} \rightarrow \{milk\}) = \frac{SC(\{yogurt, milk, bread\})}{N} = 2/5$$

# Definitions — Confidence

- **Confidence** of an association rule  $X \rightarrow Y$ 
  - Probability of  $Y$  given  $X$

$$C(X \rightarrow Y) = \frac{S(X \rightarrow Y)}{S(X)} = \frac{S(X \cup Y)}{S(X)}$$

TID	Items
1	bread, yogurt
2	bread, milk, cereal, eggs
3	yogurt, milk, cereal, cheese
4	bread, yogurt, milk, cereal
5	bread, yogurt, milk, cheese

$$C(\{yogurt, milk\} \rightarrow \{bread\}) = \frac{S(\{yogurt, milk, bread\})}{S(\{yogurt, milk\})} = 2/3$$

# High Support, High Confidence → Interesting Rules

$X \rightarrow Y$	Low Support	High Support
Low Confidence	<ul style="list-style-type: none"><li>The items in <math>(X \cup Y)</math> do not frequently appear together</li><li>Even if the items in X appear together, they do so often without the items in Y</li></ul>	<ul style="list-style-type: none"><li>The items in <math>(X \cup Y)</math> frequently appear together</li><li>If the items in X appear together, they often do so without the items in Y</li></ul>
High Confidence	<ul style="list-style-type: none"><li>The items in <math>(X \cup Y)</math> do not frequently appear together</li><li>If the items in X appear together, they often do so with the items in Y</li></ul>	<ul style="list-style-type: none"><li>The items in <math>(X \cup Y)</math> frequently appear together</li><li>If the items in X appear together, they do so often with the items in Y</li></ul>

# Quick Quiz

Given an association rule R, **which inequality** regarding the support S(R) and confidence C(R) **holds**?

**A**

$$S(R) > C(R)$$

**B**

$$S(R) \geq C(R)$$

**C**

$$S(R) \leq C(R)$$

**D**

$$S(R) < C(R)$$

# Outline

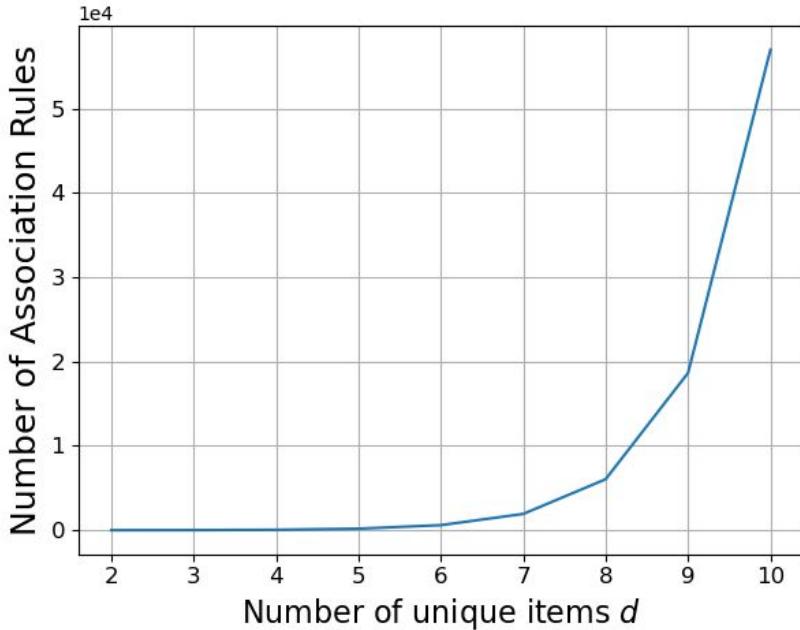
- Association Rule Mining
  - Overview
  - Applications
- Definitions
- Algorithms
  - Brute-Force
  - A-Priori
- Discussion & Summary

# Brute Force Approach — Algorithm

- Given a set of transactions,  
find all association rules  $X \rightarrow Y$  with
  - Support  $S(X \rightarrow Y) \geq \text{minsup}$
  - Confidence  $C(X \rightarrow Y) \geq \text{minconf}$
- Brute force algorithm
  - List all possible association rules  $X \rightarrow Y$
  - Calculate support  $S(X \rightarrow Y)$  and confidence  $C(X \rightarrow Y)$  for each rule
  - Drop rules with  $S(X \rightarrow Y) < \text{minsup}$  and  $C(X \rightarrow Y) < \text{minconf}$

# Brute Force Approach — Computation Complexity

- Given  $d$  unique items  $\rightarrow 3^d - 2^{d+1} + 1 \in O(3^d)$  rules
  - $d = 6 \rightarrow 602$  (theoretically) possible rules!



Average number items carried in a supermarket in 2019

Source: FMI

**28,112**

<https://www.fmi.org/our-research/supermarket-facts>

# Brute Force Approach — Computation Complexity

- Let  $w$  be the maximum number of items in a transaction within the database
  - $N = 5, w = 4 \rightarrow \leq 250$  "available" rules!



The difference between 250 and 602 seems negligible, but this is only because in this toy example,  $d = 6$  and  $w = 4$  are of the same magnitude.

The number 250 also ignores duplicate rules.

$$\rightarrow O(N \cdot (3^w - 2^{w+1} + 1)) \text{ rules}$$

(typically  $w \ll d$ )

TID	Items
1	bread, yogurt
2	bread, milk, cereal, eggs
3	yogurt, milk, cereal, cheese
4	bread, yogurt, milk, cereal
5	bread, yogurt, milk, cheese

$N$



$w$

True number of different rules: 154

# Decoupling Support and Confidence

- Recall  $S(X \rightarrow Y) = \frac{SC(X \cup Y)}{N}$

$$\left. \begin{array}{l} S(\{yogurt, milk\} \rightarrow \{bread\}) \\ S(\{yogurt, bread\} \rightarrow \{milk\}) \\ S(\{milk, bread\} \rightarrow \{yogurt\}) \end{array} \right\} = \frac{SC(\{yogurt, milk, bread\})}{N} = S(\{yogurt, milk, bread\})$$

- **Observation 1**

- A rule  $X \rightarrow Y$  has only sufficient support if  $X \cup Y$  is a frequent itemset
- No need to calculate confidence of rules where  $X \cup Y$  is not a frequent item set

$$S(X \rightarrow Y) \geq \text{minsup} \iff S(X \cup Y) \geq \text{minsup}$$

# Two-Part Algorithm for Mining Association Rules

- Part 1 — Frequent Itemset Generation
  - Generate itemsets with support  $\geq \text{minsup}$
  - "Only"  $2^d - 1$  possible itemsets to check
- Part 2: — Association Rule Generation
  - Generate rules from frequent itemsets
  - Return rules with confidence  $\geq \text{minconf}$

TID	Items
1	bread, yogurt
2	bread, milk, cereal, eggs
3	yogurt, milk, cereal, cheese
4	bread, yogurt, milk, cereal
5	bread, yogurt, milk, cheese



*minsup*

**Frequent itemsets:**

{milk}, {cereal, milk}, {bread, milk}, ...



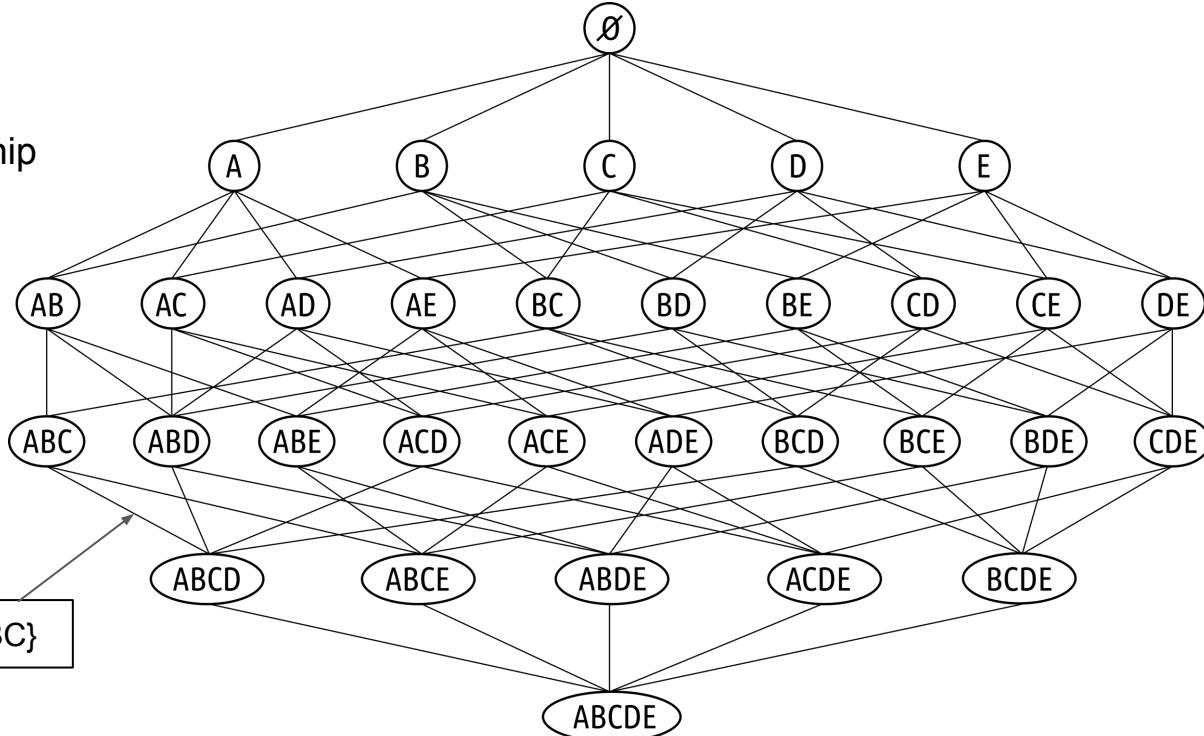
*minconf*

**Association rules:**

{cereal}  $\rightarrow$  {milk}

# Frequent Itemset Generation

- Itemset lattice
  - Node: itemset
  - Edge: containment relationship
- $2^d - 1$  nodes/itemsets
  - $d=5 \rightarrow 31$  itemsets



# Frequent Itemset Generation — Brute Force Algorithm

```
support_counts ← dict({})  
for each transaction  $t$  in database:  
    for  $k$  in  $1..(t.length)$ :  
         $k\_itemsets \leftarrow generate\_itemsets(t, k)$   
        for each itemset in  $k\_itemsets$ :  
            support_counts[itemset] += 1
```

Global counter for all found itemsets

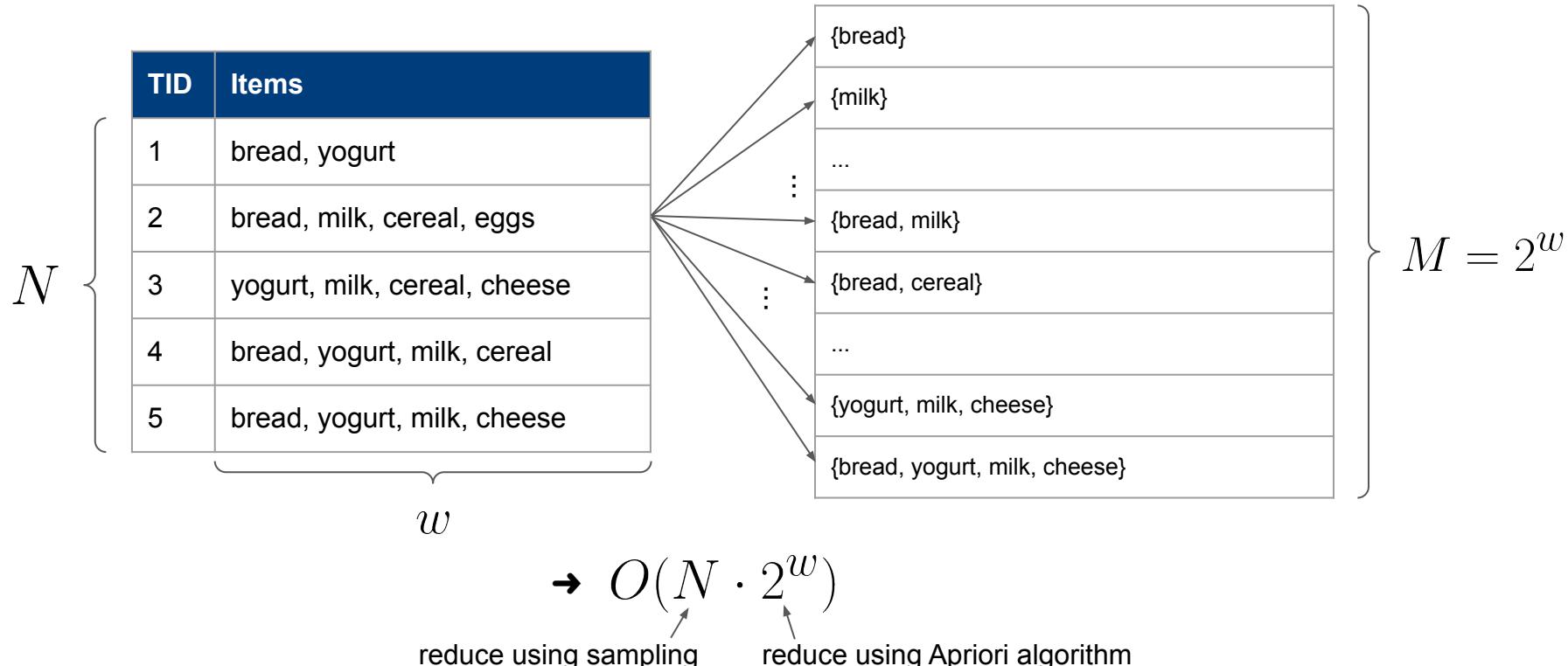
For each transaction, generate  $k$ -itemsets, with  
 $k = 1, 2, 3, \dots$  (up to #items in transaction)

For  $k$ -itemset, increase its global counter by 1

**Question:** Why do we need to count 1-itemsets  
if an association rule requires at least 2 items?

# Frequent Itemset Generation — Brute Force Algorithm

- Complexity Analysis



# Outline

- Association Rule Mining
  - Overview
  - Applications
- Definitions
- Algorithms
  - Brute-Force
  - A-Priori
- Discussion & Summary

# Apriori Principle (Anti-Monotonicity Principle)

TID	Items
1	bread, yogurt
2	bread, milk, cereal, eggs
3	yogurt, milk, cereal, cheese
4	bread, yogurt, milk, cereal
5	bread, yogurt, milk, cheese
...	

TID	Items
1	bread, yogurt
2	bread, milk, cereal, eggs
3	yogurt, milk, cereal, cheese
4	bread, yogurt, milk, cereal
5	bread, yogurt, milk, cheese
...	

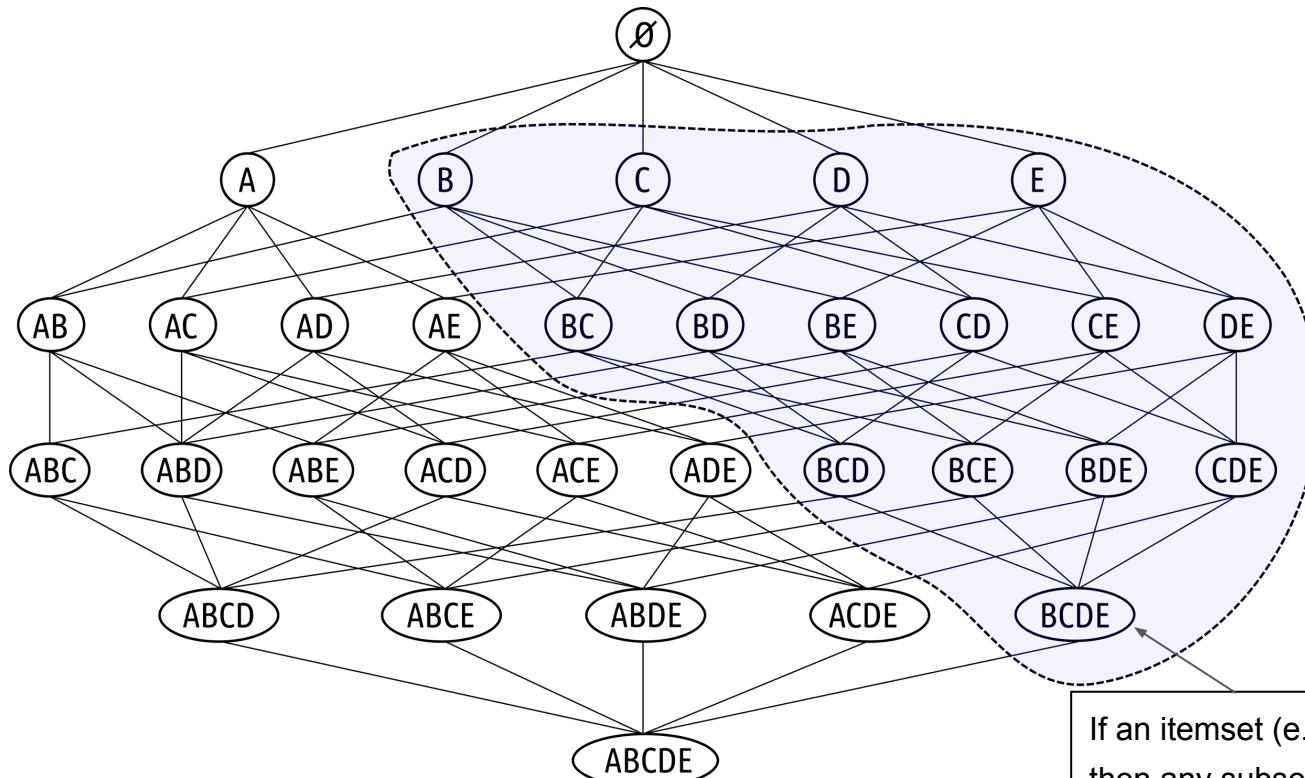
- **Observation 2:** If  $X$  and  $Y$  are itemsets and  $X \subseteq Y$ , then

- $S(X) \geq S(Y)$
- If  $Y$  is frequent, then  $X$  is frequent
- If  $X$  is not frequent, then  $Y$  is not frequent

$$S(\{\text{bread, yogurt}\}) = 3/5$$

$$S(\{\text{bread, yogurt, milk}\}) = 2/5$$

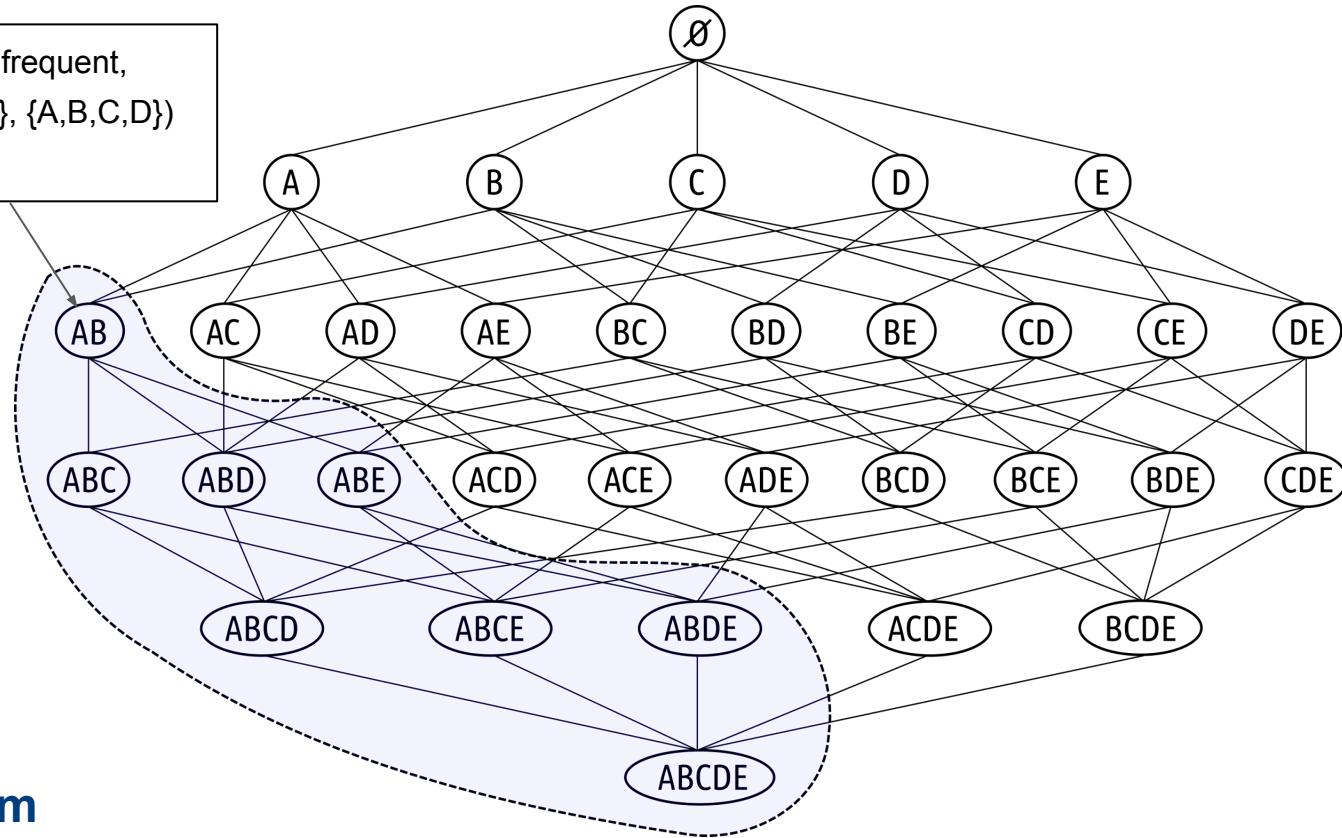
# Apriori Principle (Anti-Monotonicity Principle)



If an itemset (e.g., {B,C,D,E}) is frequent,  
then any subset (e.g., {B,D,E}, {CE}) is also frequent

# Apriori Principle (Anti-Monotonicity Principle)

If an itemset (e.g., {A,B}) is not frequent, then any superset (e.g., {A,B,D}, {A,B,C,D}) is also not frequent



→ Apriori Algorithm

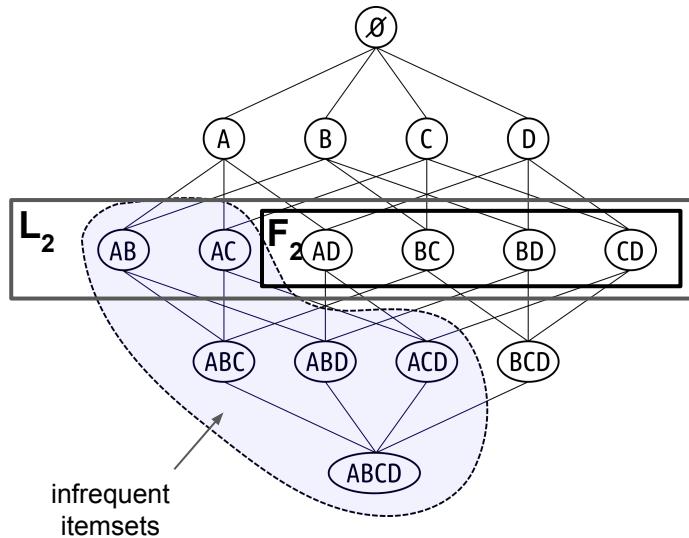
# Apriori Algorithm

- Notations

- $L_k$  — candidate k-itemsets
- $F_k$  — frequent k-itemsets ( $F_k \subseteq L_k$ )

For  $k$  in  $1..w$ :

- **Generate**  $L_k$  from  $F_{k-1}$
- **Prune** k-itemsets from  $L_k$  using  $F_{k-1}$
- **Calculate** SC for remaining  $L_k$  itemsets
- **Filter**  $L_k$  itemsets with insufficient SC  $\rightarrow F_k$
- If  $|F_k| = 0$ , stop



# Quick Quiz

Which of the four steps is generally the **most expensive** one?

For  $k$  in  $1..w$ :

- **Generate**  $L_k$  from  $F_{k-1}$
- **Prune**  $k$ -itemsets from  $L_k$  using  $F_{k-1}$
- **Calculate** SC for remaining  $L_k$  itemsets
- **Filter**  $L_k$  itemsets with insufficient SC →  $F_k$
- If  $|F_k| = 0$ , stop

A

Generate

B

Prune

C

Calculate

D

Filter

# Apriori Algorithm

$\text{minsup} = 0.4 \rightarrow \text{minimum support count: 2}$



Generating

Itemset
{bread}
{cereal}
{cheese}
{eggs}
{milk}
{yogurt}

Calculating

Itemset	SC
{bread}	4
{cereal}	3
{cheese}	2
{eggs}	1
{milk}	4
{yogurt}	4

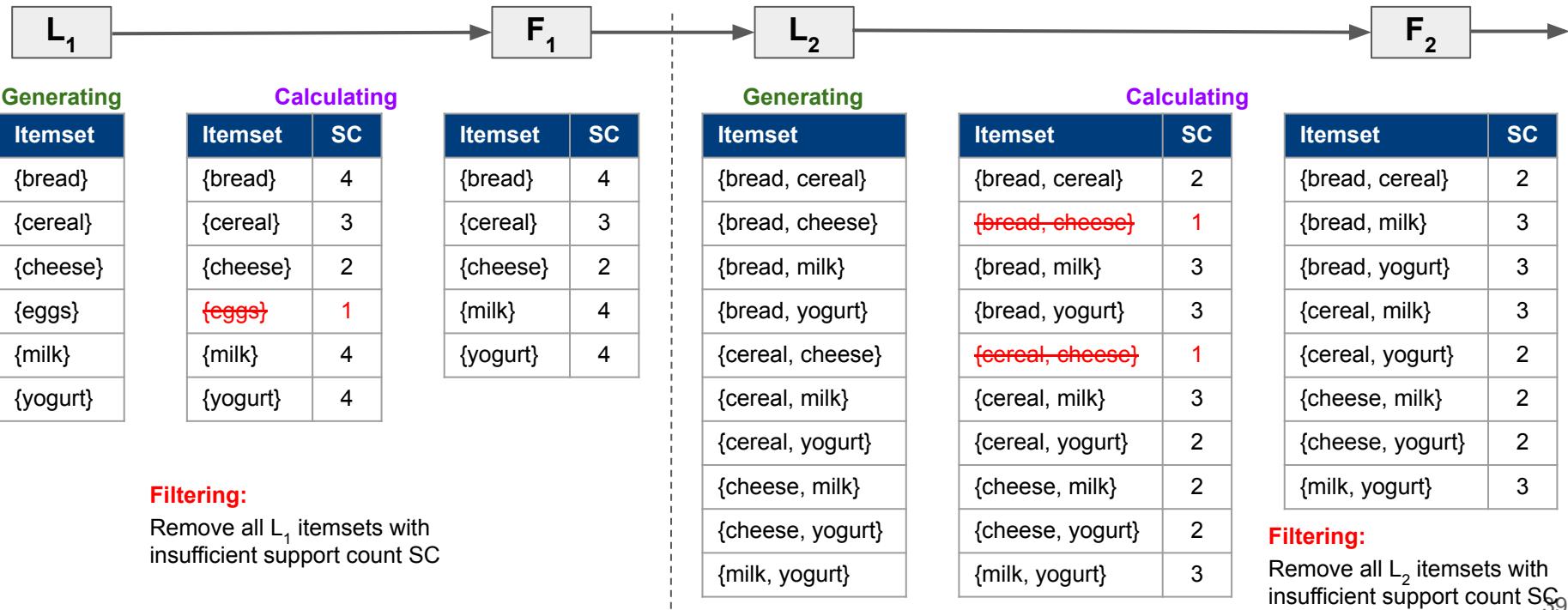
Itemset	SC
{bread}	4
{cereal}	3
{cheese}	2
{eggs}	1
{milk}	4
{yogurt}	4

Filtering:

Remove all  $L_1$  itemsets with insufficient support count SC

# Apriori Algorithm

$\text{minsup} = 0.4 \rightarrow \text{minimum support count: } 2$



# Apriori Algorithm

$\text{minsup} = 0.4 \rightarrow \text{minimum support count: } 2$



k-1 itemsets	
{bread, cheese}	✗
{bread, milk}	✓
{cheese, milk}	✓

Itemset
{bread, cereal, cheese}
{bread, cereal, milk}
{bread, cereal, yogurt}
{bread, cheese, milk}
{bread, cheese, yogurt}
{bread, milk, yogurt}
{cereal, cheese, milk}
{cereal, cheese, yogurt}
{cereal, milk, yogurt}
{cheese, milk yogurt}

## Pruning:

$\{\text{bread, cheese}\} \notin F_2$ ,

$\rightarrow \{\text{bread, cheese, milk}\} \notin F_3$

$\rightarrow \text{SC}(\{\text{bread, cheese, milk}\})$  not needed!

## Generating

Itemset	SC
{bread, cereal, milk}	2
{bread, cereal, yogurt}	1
{bread, milk, yogurt}	2
{cereal, milk, yogurt}	2
{cheese, milk yogurt}	2

## Calculating

Itemset	SC
{bread, cereal, milk}	2
{bread, milk, yogurt}	2
{cereal, milk, yogurt}	2
{cheese, milk yogurt}	2

## Filtering:

Remove all  $L_3$  itemsets with insufficient support count SC

# Apriori Algorithm

minsup = 0.4 → minimum support count: 2



Generating

k-1 itemsets	Itemset	Itemset	sc
{bread, cheese, milk}	{bread, cereal, cheese, milk}		
{bread, cheese, yogurt}	{bread, cereal, milk, yogurt}		
{bread, milk, yogurt}	{bread, cheese, milk, yogurt}		
{cheese, milk, yogurt}	{cereal, cheese, milk, yogurt}		

F4 is empty → done!

## Pruning:

Only {bread, milk, yogurt} is in  $F_3$

→ {bread, cheese, milk, yogurt}  $\notin F_4$

→ SC({bread, cheese, milk, yogurt}) not needed!

# Apriori Algorithm

- Output: All frequent itemsets  $F_i$  with
  - $i \geq 2$  — cannot create rules from a single item
  - $|F_i| > 0$  — set of itemsets is not empty
- Implementation details
  - Generating/Pruning — How to get from  $F_{k-1}$  to  $L_k$ ?
  - Calculating — How to calculate SC for  $L_k$  itemsets efficiently?  
(not covered here as this is done on the implementation level)

Itemset	sc
{bread, cereal}	2
{bread, milk}	3
{bread, yogurt}	3
{cereal, milk}	3
{cereal, yogurt}	2
{cheese, milk}	2
{cheese, yogurt}	2
{milk, yogurt}	3
{bread, cereal, milk}	2
{bread, milk, yogurt}	2
{cereal, milk, yogurt}	2
{cheese, milk yogurt}	2

$F_2$   
 $F_3$

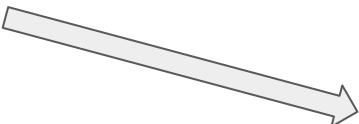
# Generating/Pruning: $F_{k-1} \times F_1$ Method

$F_2$ : frequent 2-itemsets

Itemset
{bread, cereal}
{bread, milk}
{bread, yogurt}
{cereal, milk}
{cereal, yogurt}
{cheese, milk}
{cheese, yogurt}
{milk, yogurt}

## Generating:

Merge frequent  $(k-1)$ -itemsets and frequent 1-itemsets to get all possible  $k$ -itemsets



$L_3$ : 3-itemsets

Itemset
{bread, cereal, cheese}
{bread, cereal, milk}
{bread, cereal, yogurt}
{bread, cheese, milk}
{bread, cheese, yogurt}
{bread, milk, yogurt}
{cereal, cheese, milk}
{cereal, cheese, yogurt}
{cereal, milk, yogurt}
{cheese, yogurt, milk}

## Pruning:

Delete all  $k$ -itemsets with at least one containing  $(k-1)$ -itemset not in  $F_{k-1}$



$L_3$ : 3-itemsets (pruned)

Itemset
{bread, cereal, milk}
{bread, cereal, yogurt}
{bread, milk, yogurt}
{cereal, milk, yogurt}
{cheese, milk, yogurt}

$F_1$ : frequent 1-itemsets

Itemset
{bread}
{cereal}
{cheese}
{milk}
{yogurt}

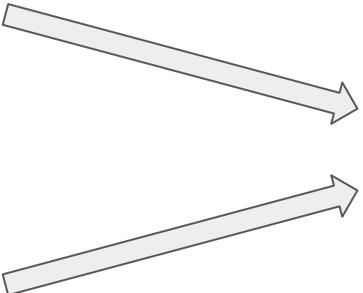
# Generating/Pruning: $F_{k-1} \times F_{k-1}$ Method

$F_2$ : frequent 2-itemsets

Itemset
{bread, cereal}
{bread, milk}
{bread, yogurt}
{cereal, milk}
{cereal, yogurt}
{cheese, milk}
{cheese, yogurt}
{milk, yogurt}

## Generating:

Merge frequent (k-1)-itemsets that overlap in (k-2) items to get all possible k itemsets



$L_3$ : 3-itemsets

Itemset
{bread, cereal, milk}
{bread, cheese, milk}
{bread, cereal, yogurt}
{bread, cheese, yogurt}
{bread, milk, yogurt}
{cereal, cheese, milk}
{cereal, cheese, yogurt}
{cereal, milk, yogurt}
{cheese, milk, yogurt}

## Pruning:

Delete all k-itemsets with at least one containing (k-1)-itemset not in  $F_{k-1}$



$L_3$ : 3-itemsets (pruned)

Itemset
{bread, cereal, milk}
{bread, cereal, yogurt}
{bread, milk, yogurt}
{cereal, milk, yogurt}
{cheese, milk, yogurt}

# Calculating Support Counts

- Calculating SC for each candidate itemset in  $L_k$ 
  - Requires **full scan** of database
  - For **each** transactions T, check for **each** itemset s if  $s \in T$
  - If  $s \in T$ , **update** counter of s

→ This is the step we want to minimize!

$L_3$ : 3-itemsets

Itemset
{bread, cereal, milk}
{bread, cereal, yogurt}
{bread, milk, yogurt}
{cereal, milk, yogurt}
{cheese, milk, yogurt}

Calculating

$L_3$ : 3-itemsets with SC values

Itemset	SC
{bread, cereal, milk}	2
{bread, cereal, yogurt}	1
{bread, milk, yogurt}	2
{cereal, milk, yogurt}	2
{cheese, milk, yogurt}	2

# Two-Part Algorithm for Mining Association Rules

- Part 1 — Frequent Itemset Generation

- General itemsets with support  $\geq \text{minsup}$
- Apriori algorithm



- Part 2: — Association Rule Generation

- Generate rules from frequent itemsets through binary partitioning of itemsets
- Return rules with confidence  $\geq \text{minconf}$

TID	Items
1	bread, yogurt
2	bread, milk, cereal, eggs
3	yogurt, milk, cereal, cheese
4	bread, yogurt, milk, cereal
5	bread, yogurt, milk, cheese



$\text{minsup}$

**Frequent itemsets:**

{milk}, {cereal, milk}, {bread, milk}, ...



$\text{minconf}$

**Association rules:**

$\{\text{cereal}\} \rightarrow \{\text{milk}\}$

# Rule Generation

- For each frequent itemset  $S$ , derive candidate rules  $X \rightarrow Y$ 
  - A rule is a binary split of  $s$ , i.e.,  $Y=S-X$
- For each rule  $X \rightarrow Y$ 
  - Calculate confidence  $C(X \rightarrow Y)$
  - If confidence  $\geq minconf$ , add rule to final result set

}

$2^{|S|-2}$  possible rules for each frequent itemset

}

$$C(X \rightarrow Y) = \frac{SC(X \cup Y)}{SC(X)}$$

Both values have been calculated during Frequent Itemset Generation!

- No need to access database
- Fast

# Apriori Principle (Anti-Monotonicity Principle)

- Given itemset  $S$  and two derived rules  $X_1 \rightarrow Y_1$ ,  $X_2 \rightarrow Y_2$  with  $X_1 \cup Y_1 = X_2 \cup Y_2 = S$

$$C(X_1 \rightarrow Y_1) = \frac{S(X_1 \cup Y_1)}{S(X_1)} \quad C(X_2 \rightarrow Y_2) = \frac{S(X_2 \cup Y_2)}{S(X_2)}$$

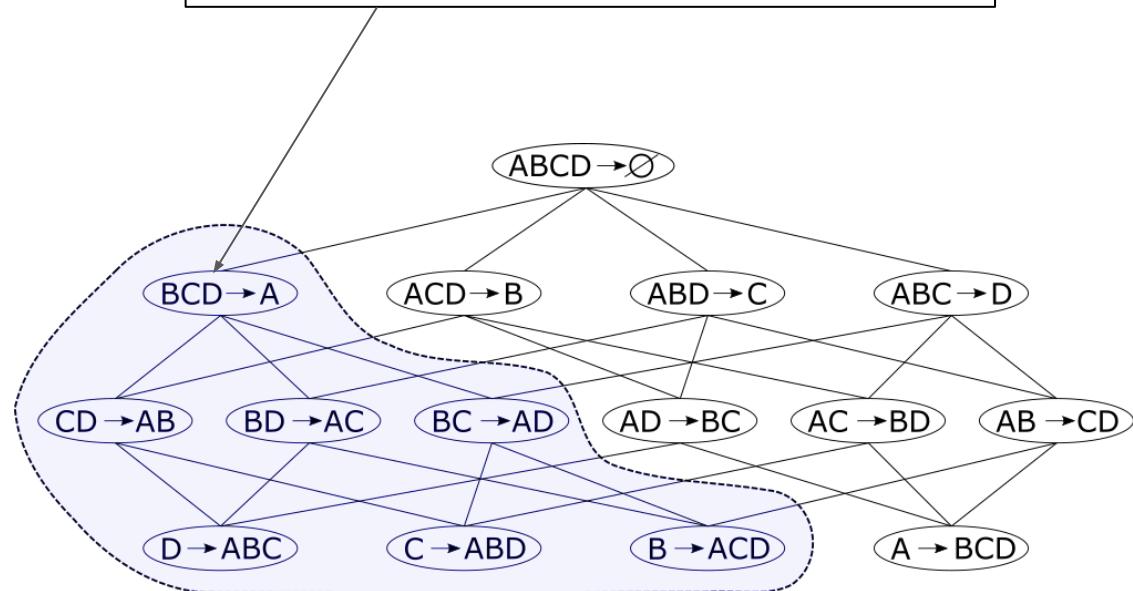
$$\begin{aligned} X_1 \subseteq X_2 \Rightarrow S(X_1) &\geq S(X_2) \\ \Rightarrow C(X_1 \rightarrow Y_1) &\leq C(X_2 \rightarrow Y_2) \end{aligned}$$

- Example: If  $\{A,B,C\} \rightarrow \{D\}$  has low confidence, so have:
  - $\{A\} \rightarrow \{B,C,D\}$ ,  $\{B\} \rightarrow \{A,C,D\}$ ,  $\{C\} \rightarrow \{A,B,D\}$ ,  $\{A,B\} \rightarrow \{C,D\}$ ,  $\{A,C\} \rightarrow \{B,D\}$ ,  $\{B,C\} \rightarrow \{A,D\}$

# Apriori Principle (Anti-Monotonicity Principle)

- Rule lattice
  - Node: association rule
  - Edge: containment relationship w.r.t. antecedent/consequent
- $2^{|S|}-2$  rules
  - $|S|=4 \rightarrow 14$  rules

If rule  $BCD \rightarrow A$  has a insufficient confidence,  
so do all rules containing A in the consequent



# Rule Generation Algorithm (for a single itemset S)

$YS = \{ \{s\} \mid s \in S \}$

**repeat**:

$YS\_valid = \text{evaluate}(S, YS, minconf)$

$YS = \text{generate}(YS\_valid)$

**until**  $|YS| = 0$

$YS = \{ \{A\}, \{B\}, \{C\}, \{D\} \}$

**evaluate**( $S, YS, minconf$ ) :

$YS\_obsolete \leftarrow \{\}$

**for each**  $Y$  **in**  $YS$  :

$X = S - Y$

**if**  $C(X \rightarrow Y) \geq minconf$  :

    output  $(X \rightarrow Y)$  as a valid rule

**else**:

$YS\_obsolete \leftarrow YS\_obsolete \cup Y$

**return**  $YS - YS\_obsolete$

$Y = \{A\}$	$Y = \{B\}$	$Y = \{C\}$	$Y = \{D\}$
$\{BCD\} \rightarrow \{A\}$	$\{ACD\} \rightarrow \{B\}$	$\{ABD\} \rightarrow \{C\}$	$\{ABC\} \rightarrow \{D\}$

$\{ \{B\}, \{C\}, \{D\} \}$

# Rule Generation Algorithm (for a single itemset S)

$YS = \{ \{s\} \mid s \in S \}$

**repeat**:

$YS\_valid = \text{evaluate}(S, YS, minconf)$

$YS = \text{generate}(YS\_valid)$

**until**  $|YS| = 0$

**evaluate**( $S, YS, minconf$ ) :

$YS\_obsolete \leftarrow \{\}$

**for each**  $Y$  **in**  $YS$  :

$X = S - Y$

**if**  $C(X \rightarrow Y) \geq minconf$  :

    output  $(X \rightarrow Y)$  as a valid rule

**else**:

$YS\_obsolete \leftarrow YS\_obsolete \cup Y$

**return**  $YS - YS\_obsolete$

$YS\_valid = \{ \{B\}, \{C\}, \{D\} \}$

$YS = \{ \{B, C\}, \{B, D\}, \{C, D\} \}$

$Y = \{B, C\}$

$\{AD\} \rightarrow \{BC\}$

$Y = \{B, D\}$

$\{AC\} \rightarrow \{BD\}$

$Y = \{C, D\}$

$\{AB\} \rightarrow \{CD\}$

$\{ \{B, C\}, \{B, D\}, \{C, D\} \}$

# Rule Generation Algorithm (for a single itemset S)

$YS = \{ \{s\} \mid s \in S \}$

**repeat**:

$YS\_valid = \text{evaluate}(S, YS, minconf)$

$YS = \text{generate}(YS\_valid)$

**until**  $|YS| = 0$

**evaluate**( $S, YS, minconf$ ) :

$YS\_obsolete \leftarrow \{\}$

**for each**  $Y$  **in**  $YS$  :

$X = S - Y$

**if**  $C(X \rightarrow Y) \geq minconf$  :

    output  $(X \rightarrow Y)$  as a valid rule

**else**:

$YS\_obsolete \leftarrow YS\_obsolete \cup Y$

**return**  $YS - YS\_obsolete$

$YS\_valid = \{ \{B, C\}, \{B, D\}, \{C, D\} \}$

$YS = \{ \{B, C, D\} \}$

$Y = \{B, C, D\}$

$\{A\} \rightarrow \{B, C, D\}$

$\{ \{B, C, D\} \}$

# Rule Generation Algorithm (for a single itemset S)

$YS = \{ \{s\} \mid s \in S \}$

**repeat**:

$YS\_valid = \text{evaluate}(S, YS, minconf)$

$YS = \text{generate}(YS\_valid)$

**until**  $|YS| = 0$

**evaluate**( $S, YS, minconf$ ) :

$YS\_obsolete \leftarrow \{\}$

**for each**  $Y$  **in**  $YS$  :

$X = S - Y$

**if**  $C(X \rightarrow Y) \geq minconf$  :

            output  $(X \rightarrow Y)$  as a valid rule

**else** :

$YS\_obsolete \leftarrow YS\_obsolete \cup Y$

**return**  $YS - YS\_obsolete$

$YS\_valid = \{ \{B, C, D\} \}$

$YS = \{\}$



**Done!**

# Two-Part Algorithm for Mining Association Rules

- Part 1 — Frequent Itemset Generation
  - General itemsets with support  $\geq \text{minsup}$
  - Apriori algorithm



- Part 2: — Association Rule Generation
  - Generate rules from frequent itemsets through binary partitioning of itemsets
  - Return rules with confidence  $\geq \text{minconf}$



TID	Items
1	bread, yogurt
2	bread, milk, cereal, eggs
3	yogurt, milk, cereal, cheese
4	bread, yogurt, milk, cereal
5	bread, yogurt, milk, cheese



**Frequent itemsets:**  
 $\{\text{milk}\}$ ,  $\{\text{cereal, milk}\}$ ,  $\{\text{bread, milk}\}$ , ...



**Association rules:**  
 $\{\text{cereal}\} \rightarrow \{\text{milk}\}$

# Definitions — Lift

- **Lift of an association rule  $X \rightarrow Y$**

- Probability of  $Y$  given  $X$  while controlling for support of  $Y$  (i.e., popularity of  $Y$ )

$$L(X \rightarrow Y) = \frac{S(X \rightarrow Y)}{S(X)S(Y)} = \frac{S(X \cup Y)}{S(X)S(Y)}$$

TID	Items
1	bread, yogurt
2	bread, cereal, milk, eggs
3	yogurt, milk, cereal, cheese
4	bread, cereal, yogurt, milk
5	bread, yogurt, milk, cheese

$$L(\{cereal\} \rightarrow \{bread\}) = \frac{S(\{cereal, bread\})}{S(\{cereal\})S(\{bread\})} = \frac{0.4}{0.6 \cdot 0.8} = 0.833$$

# Lift — Interpretation

$$L(\{cereal\} \rightarrow \{bread\}) = \frac{S(\{cereal, bread\})}{S(\{cereal\})S(\{bread\})} = \frac{0.4}{0.6 \cdot 0.8} = 0.833$$

- Probability of {bread}  
 $S(\{bread\}) = 0.8$
  - Probability of {bread} given {cereal}  
 $C(\{cereal\} \rightarrow \{bread\}) = 0.66$
- Presence of cereal **reduces** probability of bread!  
 $\Rightarrow L(\{cereal\} \rightarrow \{bread\}) \leq 1.0$

- **Usage of lift** (and other metrics for association rules)
  - Further filtering and ranking of association rules
  - Finding "substitution" items

**Note:** Lift is not part of Apriori algorithm since anti-monotonicity principle does not hold here

# Quick "Quiz"

Which association rule would you expect to have the **smallest lift**?

A

$\{cereal\} \rightarrow \{milk\}$

B

$\{coke\} \rightarrow \{pepsi\}$

C

$\{milo\} \rightarrow \{nutella\}$

D

$\{banana\} \rightarrow \{grapes\}$

# Outline

- Association Rule Mining
  - Overview
  - Applications
- Definitions
- Algorithms
  - Brute-Force
  - A-Priori
- Discussion & Summary

# Discussion

- Alternative metric to decide whether a rule is interesting (beyond confidence and lift)
  - Conviction, all-confidence, collective strength, leverage
- Additional useful information to consider, for example:
  - Attributes of items (e.g., quantity and price of products)
  - Sequence of items (e.g., order when products have been added to the cart)
  - Categories of items (e.g., "milk" and "yogurt" are both "dairy" products)
  - User information (e.g., associating multiple transactions to the same user)
- Reminder: Rules indicate correlations / co-occurrences, NOT causality!

# Summary

- **Pattern of interest: Association Rule  $X \rightarrow Y$** 
  - Predicting the occurrence of some items  $Y$  based on occurrence of other items  $X$
  - Applicable to a wider range of task for transactional data
  - Various metrics that define whether a rule is useful (e.g., support, confidence, lift)
- **Practical algorithm to handle complexity**
  - Decoupling calculations of support and confidence
  - Apriori algorithm for Frequent Itemset Generation and Association Rule Generation

# Solutions to Quick Quizzes

- Slide 22: C
- Slide 37: C
- Slide 57: B

# **CS5228: Knowledge Discovery and Data Mining**

## Lecture 5 — Classification & Regression I

# Course Logistics — Update

- Assignment 2
  - Release: Fri, Sep 13
  - Submission deadline: Thu, Oct 03 (11.59 pm)
  - Reminder: Please adhere to naming and uploading guidelines
- Midterm exam preparation
  - Install and check out Examplify (see [student guide](#))
  - Try practice exam (Non-Secure Block Internet)

# Recap — Association Rules

- **Association Rule Mining**
  - Input: database of transactions (i.e., sets of items)
  - Output: rules predicting the occurrence of some items based on occurrence of other items
- **Example: Market Basket Analysis**
  - Item = product in supermarket
  - Transaction = products in basket at check-out
  - Interesting rules = customers who buy X also tend to Y

TID	Items
1	bread, yogurt
2	bread, milk, cereal, eggs
3	yogurt, milk, cereal, cheese
4	bread, yogurt, milk, cereal
5	bread, yogurt, milk, cheese

# Recap — Association Rules

- **Interesting association rules**
  - Different definitions that quantify usefulness of a rule:support, confidence, lift, conviction, leverage, etc.
- **Important observation**
  - Relationship between support and confidence

→ Decoupling Support and Confidence — 2 steps:  
1) Frequent Itemset Generation  
2) Association Rule Generation

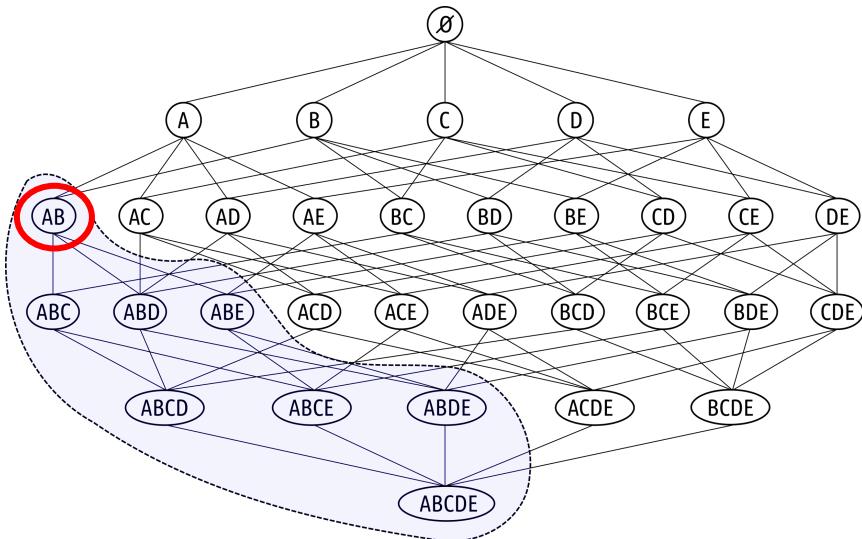
$$C(X \rightarrow Y) = \frac{S(X \rightarrow Y)}{S(X)} = \frac{S(X \cup Y)}{S(X)}$$

- **Anti-monotone** property of support and confidence

→ Apriori Algorithms for Frequent Itemset and Association Rule Generation

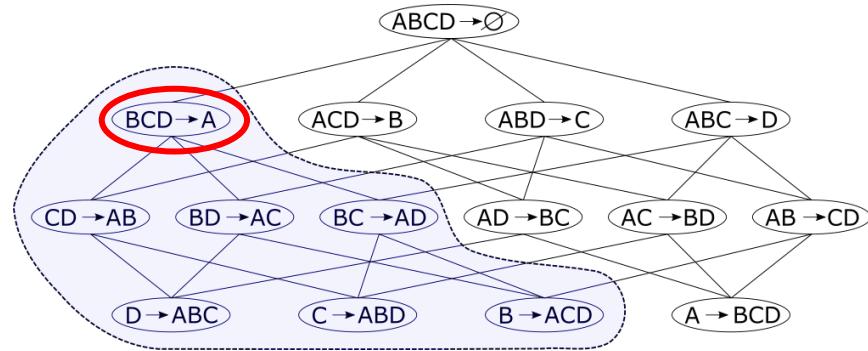
# Recap — Association Rules

Itemset Lattice



If an itemset (e.g., {A,B}) is not frequent, then any superset (e.g., {A,B,D}, {A,B,C,D}) is also not frequent

Rule Lattice



If rule BCD  $\rightarrow$  A has a insufficient confidence, so do all rules containing A in the consequent

# Outline

- **Classification & Regression**
  - Overview & Examples
  - Basic Setup
  - Evaluation
- **Nearest Neighbor Methods**
  - KNN — K-Nearest Neighbor Algorithm
  - Pros, Cons & Caveats

# Classification & Regression: Overview

- **Classification AND Regression**
  - Core tasks of supervised machine learning
  - Training: Find patterns in dataset between the values of **dependent variable(s)** given the values **independent variable(s) / features**
  - Prediction: Use patterns to assign values to dependent variables new/unseen data
- **Classification VS. Regression**
  - Classification: Dependent variable is categorical
  - Regression: Dependent variable is continuous

Independent variables (features)

Age	Edu- cation	Marital Status	Income Level	Credit Approval	Credit Limit
23	Masters	Single	Mid	No	—
35	College	Married	High	Yes	\$S7,000
26	Masters	Single	High	No	—
41	PhD	Single	Mid	Yes	\$S5,000
18	Poly	Single	Low	No	—
55	Poly	Married	High	Yes	\$S10,000
30	College	Single	High	Yes	\$S8,000
35	PhD	Married	High	Yes	\$S10,000
28	Masters	Married	Mid	Yes	\$S5,000
45	Masters	Married	Mid	???	???

Dependent, categorical variable

Dependent, numerical variable

# Application Examples

month	flat_type	block	street_name	floor_area_sqm	flat_model	lease_commence_date	latitude	longitude	subzone	planning_area	resale_price
2013-02	4 room	4B	boon tiona road	91.0	model a	2005	1.286589	103.831950	tiong bahru	bukit merah	663888.0
1997-08	3 room	11	holland drive	65.0	improved	1975	1.309054	103.794063	holland drive	queenstown	228000.0
2007-08	executive	558	jurong west street 42	157.0	maisonette	1985	1.354198	103.717344	hong kah	jurong west	305000.0
1996-02	4 room	354	hougang avenue 7	105.0	new generation	1986	1.372408	103.899433	kangkar	hougang	235000.0
2015-06	3 room	99	old airport road	56.0	standard	1969	1.308590	103.888245	aljunied	geylang	320000.0

- Prediction of flat prices (regression task)
  - Help sellers and buyers to make better decisions
  - Understand the importance of attributes on flat prices

# Application Examples

Has heart disease (1)  
or not (0)

ID	age	sex	chest	rest_bp	chol	rest_ecg	max_hr	oldpeak	slope	num_vessels	thal	class
0	49.2	0	4.00	163.00	181.11	0	148.23	0.94	2	0	3	1
1	53.6	1	1.74	130.23	276.47	2	152.92	0.12	2	0	3	0
2	49.6	1	4.00	147.00	223.30	2	102.35	1.62	2	2	7	1
3	59.0	1	4.00	112.37	187.25	0	158.16	0.00	1	1	7	1
4	51.1	1	1.95	138.03	238.48	0	172.54	1.15	1	1	3	0

- Health analytics: prediction of heart disease
  - Diagnosis and therapy support systems
  - Identify most important risk factors

# Application Examples

- Sentiment Analysis / Opinion Mining (over text)

Regression



94%



90%

TOMATOMETER

AUDIENCE SCORE



What you will be getting when you walk into an inevitably overstuffed movie theater is something singular that reflects our age in a way that none of the MCU films that preceded it have—indeed, very few Hollywood spectacles ever have.

May 1, 2019 | Rating: 3.5/4 | [Full Review...](#)



Oliver Jones

Observer

★ Top Critic

What's missing from "Endgame" is the free play of imagination, the liberation of speculation, the meandering paths and loose ends that start in logic and lead to wonder.

April 28, 2019 | [Full Review...](#)



Richard Brody

New Yorker

★ Top Critic

Endgame consists almost entirely of the downtime scenes that were always secretly everyone's favorite parts of these movies anyway.

April 26, 2019 | [Full Review...](#)



Dana Stevens

Slate

★ Top Critic

Classification



**Fresh**



**Rotten**



**Fresh**



**Rotten**

Tomatometer

Audience

# Application Examples

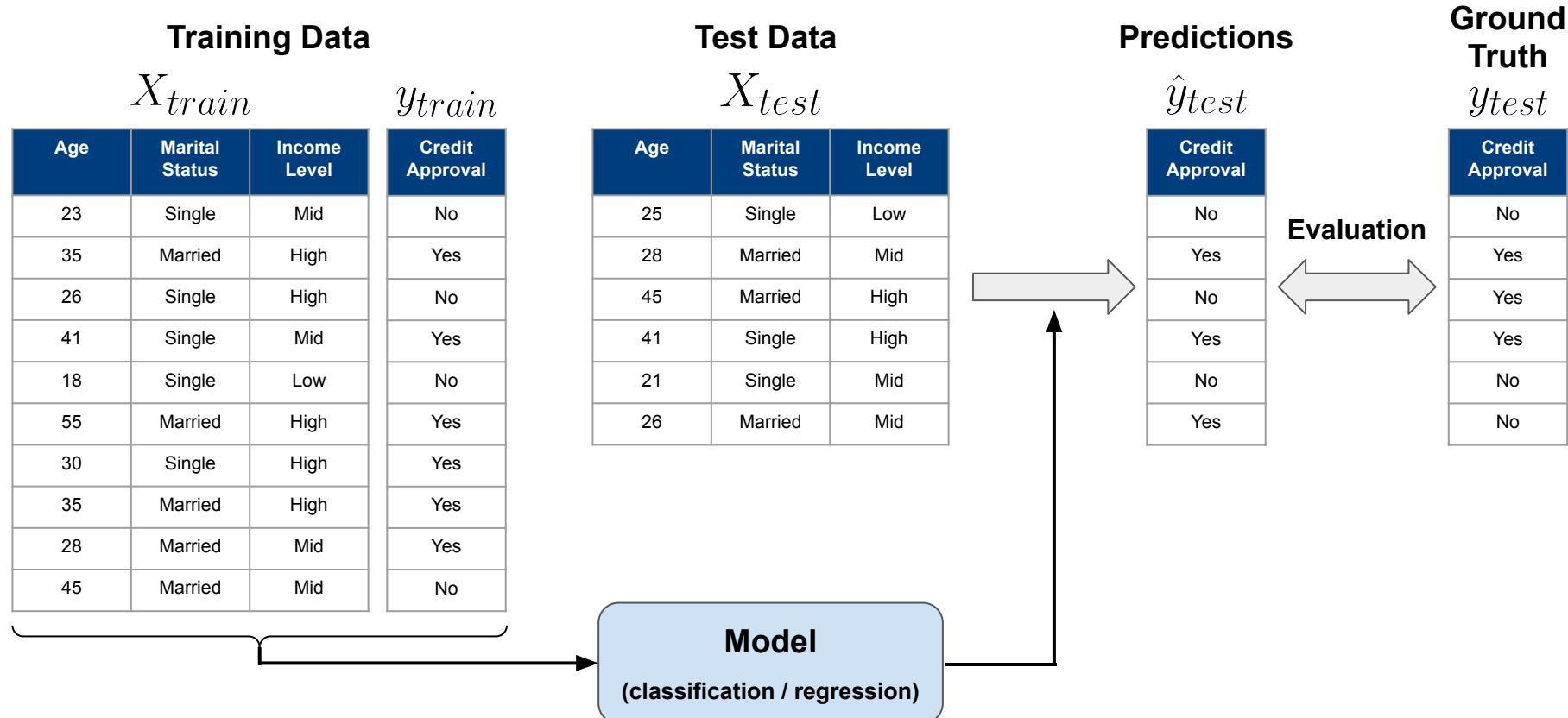
- Self-driving vehicles
  - Input: image & sensor data
- Regression tasks, e.g.:
  - Acceleration
  - Steering angle
  - Event prediction (%)
- Classification tasks, e.g.:
  - Obstacle detection
  - Street sign identification



# Outline

- **Classification & Regression**
  - Overview & Examples
  - **Basic Setup**
  - Evaluation
- **Nearest Neighbor Methods**
  - KNN — K-Nearest Neighbor Algorithm
  - Pros, Cons & Caveats

# Supervised Training/Learning — Basic Setup



# Supervised Training/Learning — Training & Test Data

- Given: Dataset D of pairs  $\{(x, y)\}$ 
  - x — features (independent variables)
  - y — label (dependent variable)
- Split dataset D into:
  - $D_{train}$  — data for training the model
  - $D_{test}$  — data for evaluating the model
  - Important:  $D_{train} \cap D_{test} = \emptyset$   
(the test data is not allowed to be used during training)

Age	Edu- cation	Marital Status	Income Level	Credit Approval
23	Masters	Single	Mid	No
35	College	Married	High	Yes
26	Masters	Single	High	No
41	PhD	Single	Mid	Yes
18	Poly	Single	Low	No
55	Poly	Married	High	Yes
30	College	Single	High	Yes
35	PhD	Married	High	Yes
28	Masters	Married	Mid	Yes
45	Masters	Married	Mid	No

$D_{train}$

$D_{test}$

features      labels

# Supervised Training/Learning

- Model — Parameterized function  $h(x, \Theta) = y$ 
  - Maps input space (features) to the output space (labels)
  - $\Theta$  — trainable/learnable parameters of the model  
(note: not all models are parameterized)
- Model selection — Selection of a "family" of functions  $h(x, \Theta) = y$ 
  - K-Nearest Neighbor, Decision Trees, Linear Models, Neural Networks, ...
- Training / Learning
  - Process of systematically finding the best values for  $\Theta$
  - $\Theta_{best} \Leftrightarrow$  best mapping between features and labels

# Outline

- **Classification & Regression**
  - Overview & Examples
  - Basic Setup
  - **Evaluation**
- **Nearest Neighbor Methods**
  - KNN — K-Nearest Neighbor Algorithm
  - Pros, Cons & Caveats

# Classification & Regression — Evaluation

## Regression

$\hat{y}_{test}$

Credit Limit
\$S9,000
\$S4,000
\$S5,800
\$S10,000
\$S11,000
\$S3,500

$y_{test}$

Credit Limit
\$S10,000
\$S4,000
\$S6,000
\$S12,000
\$S10,000
\$S3,000



Direct comparison of numerical values

Common: Root Mean Squared Error (RMSE):

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

## Classification

$\hat{y}_{test}$

Credit Approval
No
Yes
No
Yes
No
Yes

$y_{test}$

Credit Approval
No
Yes
Yes
Yes
No
No



Question: How to compare classification results?

# Classification: Evaluation

- Base case: Binary classification (2 labels: 0 or 1)
- Example: COVID-19 test

test result	infected
$\hat{y}_{test}$	$y_{test}$
0	0
0	0
0	1
1	1
0	0
1	0



Confusion Matrix

		ground truth label $y$	
		1	0
predicted label $\hat{y}$	1	1	1
	0	1	3

# Classification: Evaluation — Confusion Matrix

		ground truth label $y$	
		1	0
predicted label $\hat{y}$	1	True Positives (TP)	False Positives (FP)
	0	False Negatives (FN)	True Negatives (TN)

- True Positives (TP):** Number of positive classes that have been correctly predicted as positive
- True Negatives (TN):** Number of negative classes that have been correctly predicted as negative
- False Positives (FP):** Number of negative classes that have been incorrectly predicted as positive
- False Negatives (FN):** Number of positive classes that have been incorrectly predicted as negative

# Classification: Evaluation — Popular Metrics

- Accuracy

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN}$$

		$y$	
		1	0
$\hat{y}$	1	TP	FP
	0	FN	TN

- Sensitivity, Specificity

$$Sensitivity = \frac{TP}{TP + FN}$$

		$y$	
		1	0
$\hat{y}$	1	TP	FP
	0	FN	TN

$$Specificity = \frac{TN}{TN + FP}$$

		$y$	
		1	0
$\hat{y}$	1	TP	FP
	0	FN	TN

# Classification: Evaluation — Popular Metrics

- Precision, Recall, F1 Score

Harmonic Mean of  
Precision and Recall

$$Precision = \frac{TP}{TP + FP}$$

		$y$	
		1	0
$\hat{y}$	1	TP	FP
	0	FN	TN

$$Recall = \frac{TP}{TP + FN}$$

		$y$	
		1	0
$\hat{y}$	1	TP	FP
	0	FN	TN

$$F1 = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall}$$

		$y$	
		1	0
$\hat{y}$	1	TP	FP
	0	FN	TN

# Classification: Evaluation — Why so Many Measures?

- Problem: (Highly) imbalanced datasets
- Example use case: COVID-19 test
  - Most people in a population are not infected
  - Assume a test that always(!) returns "negative"

		ground truth label $y$
		1
		0
predicted label $\hat{y}$	1	0
	0	200
		10,000

$$\text{Accuracy} = \frac{0 + 10000}{0 + 0 + 10000 + 200} = 98\%$$

$$\text{Specificity} = \frac{10000}{10000 + 0} = 100\%$$

→ Very high values despite "useless" test

# Classification: Evaluation — Why so Many Measures?

- Observation: FP and FN not always equally problematic

$$Precision = \frac{TP}{TP + FP}$$

$$Recall = \frac{TP}{TP + FN}$$

- Example: Heart disease prediction

- BAD: misclassifying a high-risk person
- OK-ish: misclassifying a healthy person



**Recall > Precision**

- Example: News article classification

(e.g., for search engines such as Google News)

- BAD: showing article of wrong category
- OK: missing a relevant article in result



**Recall < Precision**

# Classification: Evaluation — Numerical Class Scores

- Assumption so far:  
 $\hat{y}$  are **class labels**
  - Output of many models:  
 $\hat{y}$  are **class scores**
    - e.g., probability of a class
- **Thresholding**
  - Use threshold to convert  $\hat{y}$  to a binary variable 0/1

0.45
0.30
0.55
0.25
0.35
0.55
0.30
0.60
0.40
0.45

class scores

$\hat{y}_{0.5}$
$0.45 \rightarrow 0$
$0.30 \rightarrow 0$
$0.55 \rightarrow 1$
$0.25 \rightarrow 0$
$0.35 \rightarrow 0$
$0.55 \rightarrow 1$
$0.30 \rightarrow 0$
$0.60 \rightarrow 1$
$0.40 \rightarrow 0$
$0.45 \rightarrow 0$

threshold=0.5

$\hat{y}_{0.43}$
$0.45 \rightarrow 1$
$0.30 \rightarrow 0$
$0.55 \rightarrow 1$
$0.25 \rightarrow 0$
$0.35 \rightarrow 0$
$0.55 \rightarrow 1$
$0.30 \rightarrow 0$
$0.60 \rightarrow 1$
$0.40 \rightarrow 0$
$0.45 \rightarrow 1$

threshold=0.43

$y$
1
0
0
0
1
1
0
1
0
1
0
1
0
1

ground truth

# Classification: Evaluation — Numerical Class Scores

- Different threshold yield different results

		$y$	
		1	0
$\hat{y}_{0.5}$	1	2	1
	0	3	4
			threshold=0.5

$$F1 = 0.66$$

		$y$	
		1	0
$\hat{y}_{0.43}$	1	4	1
	0	1	4
			threshold=0.43

$$F1 = 0.80$$

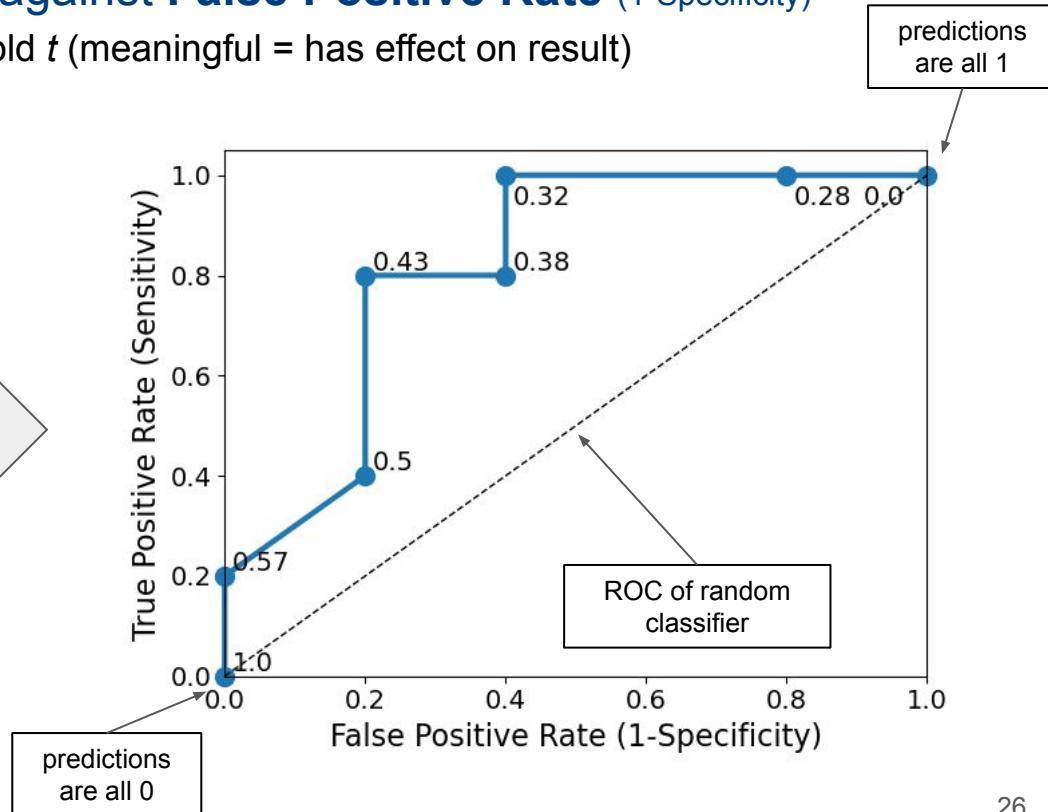
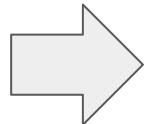
→ Question: What threshold to use? — Answer: Try them all!

# Receiver Operating Characteristic (ROC)

- Plot True Positive Rate (Sensitivity) against False Positive Rate (1-Specificity)
  - Plot values for each meaningful threshold  $t$  (meaningful = has effect on result)

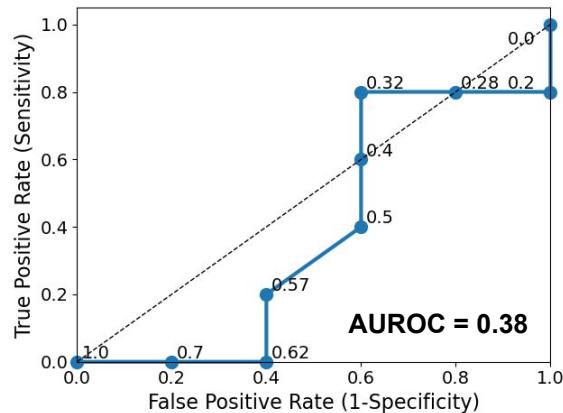
	class scores	ground truth
$t = 0.0 >$	0.25	0
$t = 0.28 >$	0.30	0
$t = 0.32 >$	0.30	0
$t = 0.38 >$	0.35	1
$t = 0.43 >$	0.40	0
$t = 0.45 >$	0.45	1
$t = 0.45 >$	0.45	1
$t = 0.50 >$	0.55	0
$t = 0.55 >$	0.55	1
$t = 0.57 >$	0.60	1
$t = 1.0 >$		

both sorted w.r.t. scores

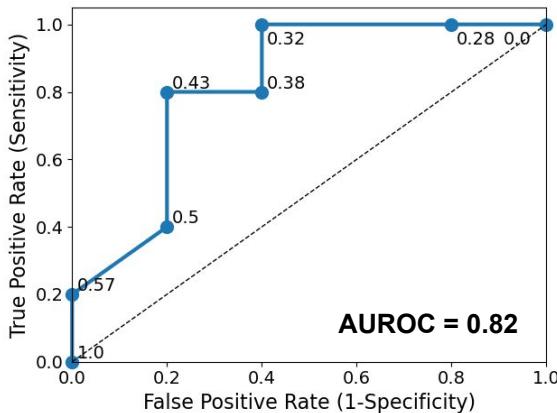


# Receiver Operating Characteristic (ROC) — Comparison

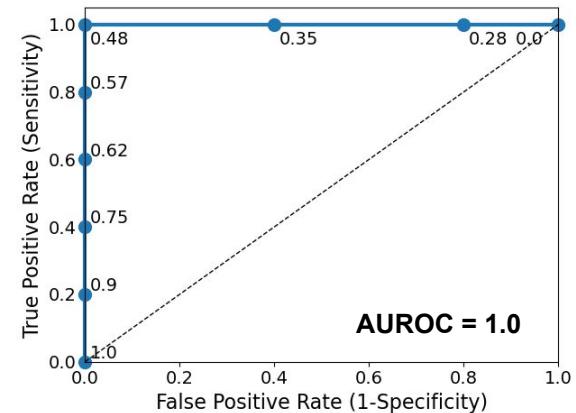
Class scores from 3 different classifiers



Poor classifier



"Decent" classifier

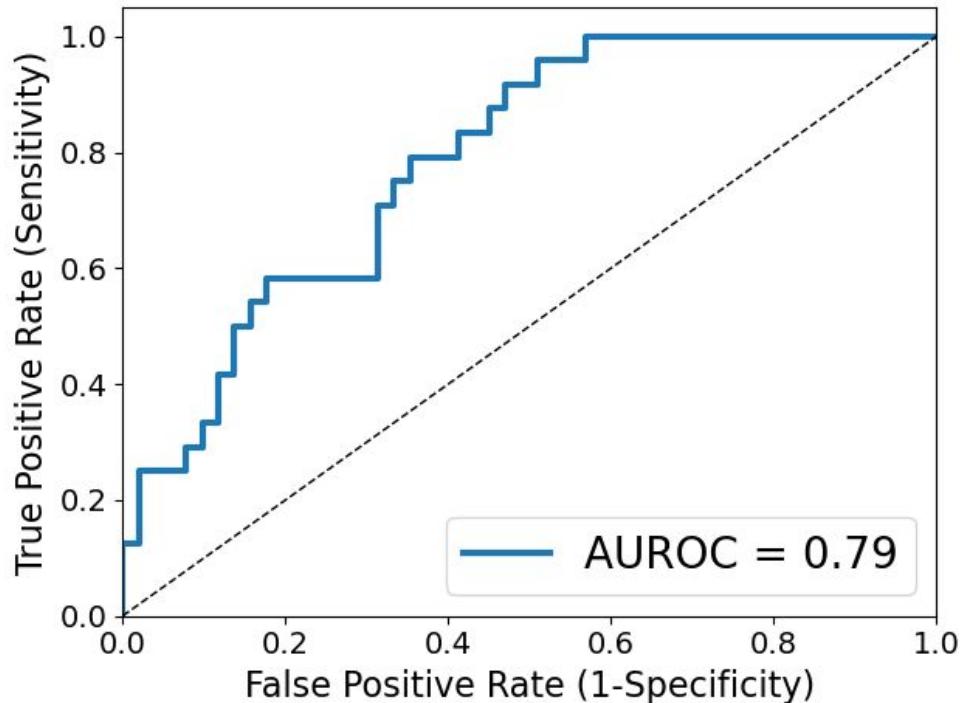


Perfect classifier

- Quantify quality of classifiers using **AUROC** (or AUC)
  - Area Under Receiver Operating Characteristic
  - AUROC of random classifier: 0.5

# ROC / AUROC — Practical Example

- IRIS dataset
  - 3 classes of Iris plants
  - 50 samples per class
  - 4 continuous features (sepal/petal length/width)
- ROC / AUROC
  - More samples, more thresholds
  - Smoother ROC
  - Thresholds typically omitted



# Quick Quiz

In what situation is **accuracy** a good measure?

**A**

The test data contains a sufficient amount of data samples

**B**

There are more than 2 class labels

**C**

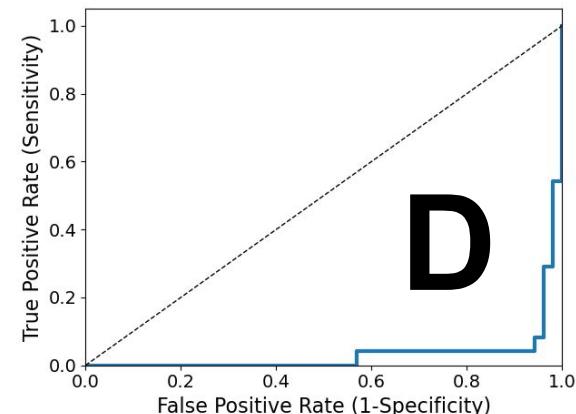
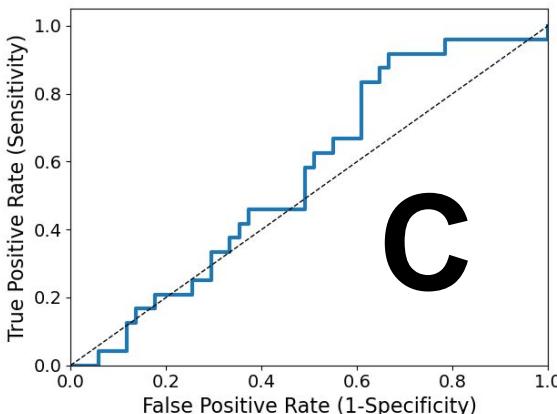
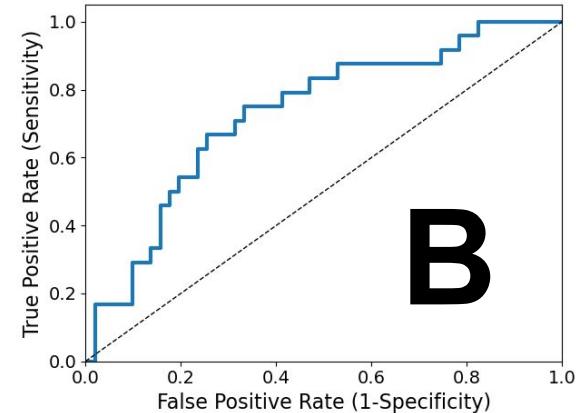
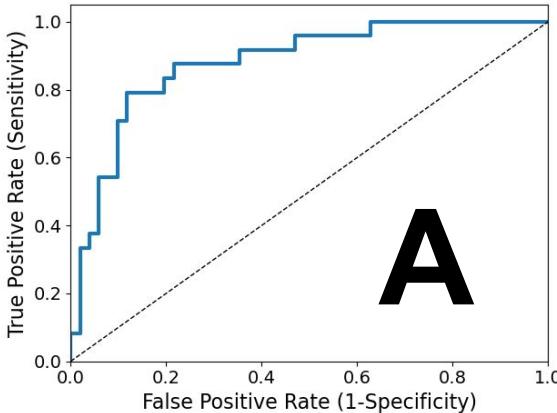
Accuracy is never a good measure

**D**

The dataset is reasonably balanced (similar number of class labels)

# Quick Quiz

For a **binary classification** task, which of the 4 classifiers would you choose?



# Classification: Evaluation — Beyond 2 Classes

- Example: 3 classes, 50 samples

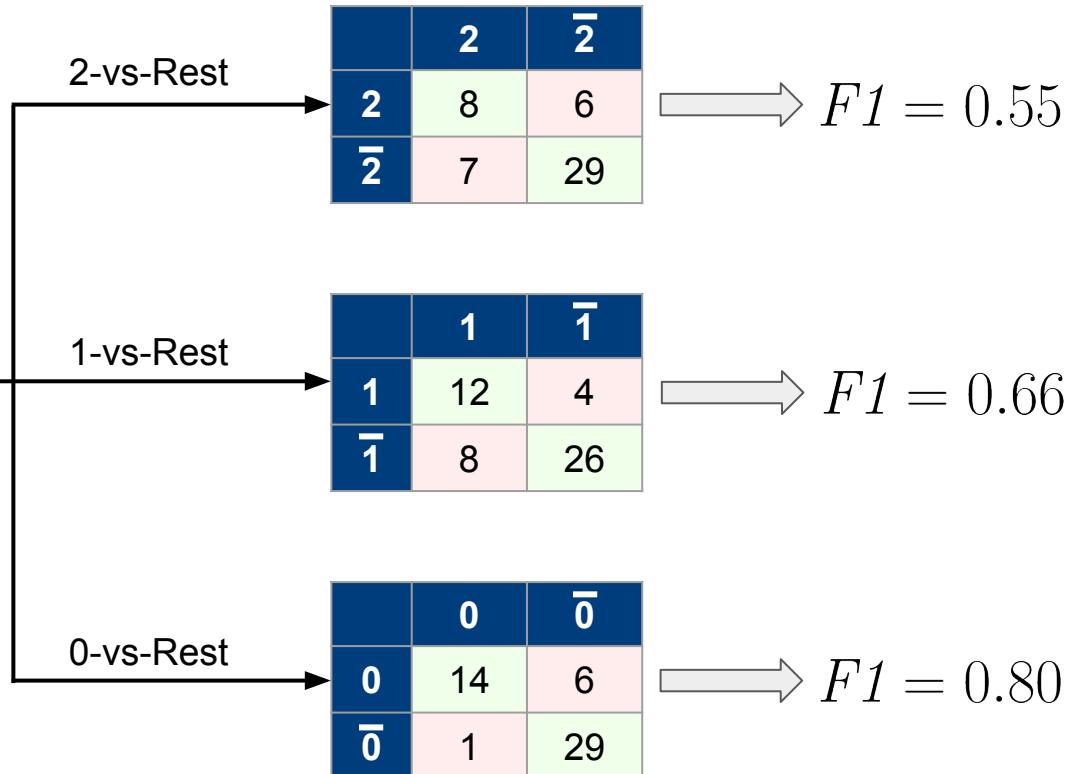
		ground truth label $y$		
		2	1	0
predicted label $\hat{y}$	2	8	6	0
	1	3	12	1
	0	4	2	14

$$\text{Accuracy} = \frac{8 + 12 + 14}{8 + 12 + 14 + 6 + 3 + 1 + 4 + 2} = 0.68$$

# Multiclass Evaluation — One-vs-Rest Confusion Matrices

- Example:

	ground truth label $y$		
	2	1	0
2	8	6	0
1	3	12	1
0	4	2	14



# One-vs-Rest — Micro Averaging

	2	$\bar{2}$
2	8	6
$\bar{2}$	7	29

	1	$\bar{1}$
1	12	4
$\bar{1}$	8	26

	0	$\bar{0}$
0	14	6
$\bar{0}$	1	29

Average over all  
TP, FP, FN, TN

	c	$\bar{c}$
c	11.33	5.33
$\bar{c}$	5.33	28

$$F1 = 0.68$$

# One-vs-Rest — Macro Averaging

	2	$\bar{2}$
2	8	6
$\bar{2}$	7	29

$$\longrightarrow F1 = 0.55$$

	1	$\bar{1}$
1	12	4
$\bar{1}$	8	26

$$\longrightarrow F1 = 0.66$$

	0	$\bar{0}$
0	14	1
$\bar{0}$	6	29

$$\longrightarrow F1 = 0.80$$

Average over  
all metrics



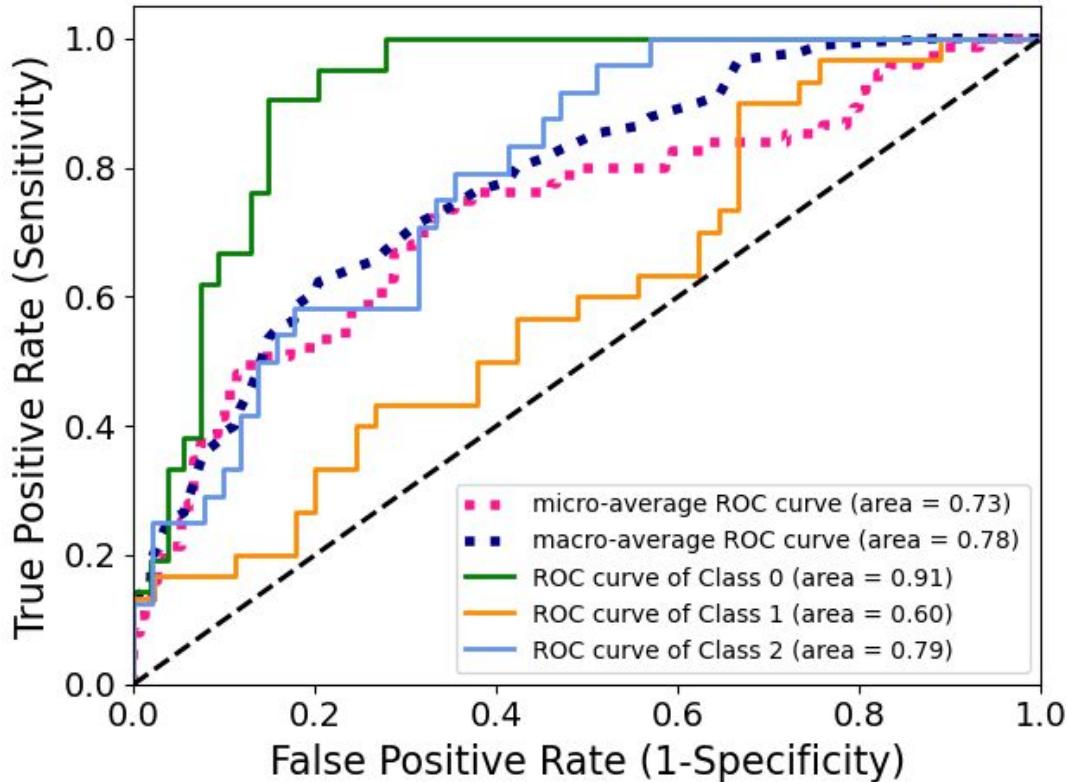
$$F1 = 0.67$$

# One-vs-Rest — Macro vs. Micro Averaging

- Both methods use One-vs-Rest confusion matrices
  - All introduced metrics applicable (incl. ROC/AUROC)
- Micro-averaging
  - Averaging over TP, FP, FN, TN values of all One-vs-Rest confusion matrices
  - Favors bigger classes (since average over counts)
- Macro-averaging
  - Averaging over metrics derived from each One-vs-Rest confusion matrix
  - Treats all class equally (since metrics are normalized)

# ROC / AUROC — Practical Example (Multiclass)

- IRIS dataset
  - 3 classes of Iris plants
  - 50 samples per class
  - 4 continuous features (sepal/petal length/width)



# Quick Quiz

A **2-class** classifier and a **10-class** classifier have a f1-score of 0.6:  
Which classifier does a **better** job?

**A**

The 2-class classifier

**B**

The 10-class classifier

**C**

Both are equally good

**D**

Not comparable

# Outline

- Classification & Regression
  - Overview & Examples
  - Basic Setup
  - Evaluation
- Nearest Neighbor Methods
  - KNN — K-Nearest Neighbor Algorithm
  - Pros, Cons & Caveats

# K-Nearest Neighbor Algorithm (KNN)

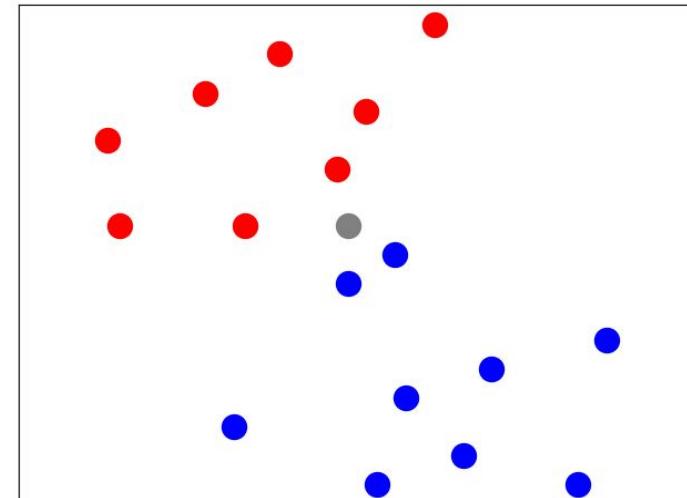
- Intuition behind KNN:

- Label of an unseen data point  $x$  derives from the labels of the  $k$ -nearest neighbors of  $x$
- Similar data points → similar labels

} Required: notion of similarity/distance

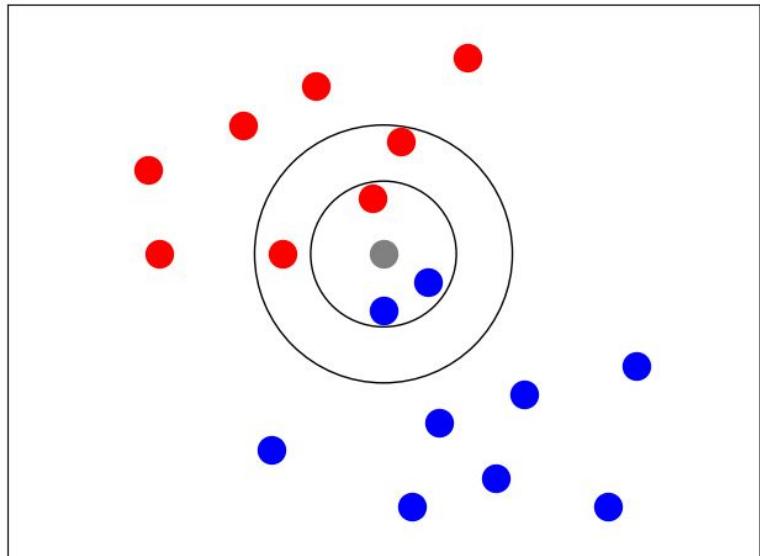
- Example

- What should be the label/color of the unseen (gray) data point?



# KNN for Classification

- "Training"
  - Remember training data
- Prediction for unseen point  $x_i$ 
  - Calculate distances to all training data points  $x_j$ , e.g.:
$$d(p, q) = \sqrt{\sum_i^d (q_i - p_i)^2}$$
Euclidean distance,  $d = \# \text{features}$
  - Get the  $k$ -nearest neighbors
  - Label of  $x_i$  = most frequent label among all  $k$ -nearest neighbors



$k$  is typically odd to minimize the chance of ties

# Quick "Quiz"

Should  $k$  not better be a **prime**  
and not just an odd number?

(e.g., to limit the chance of 3/3/3 ties for  $k=9$ )

**A**

Never set  $k$  to a prime

**B**

No need but also no harm

**C**

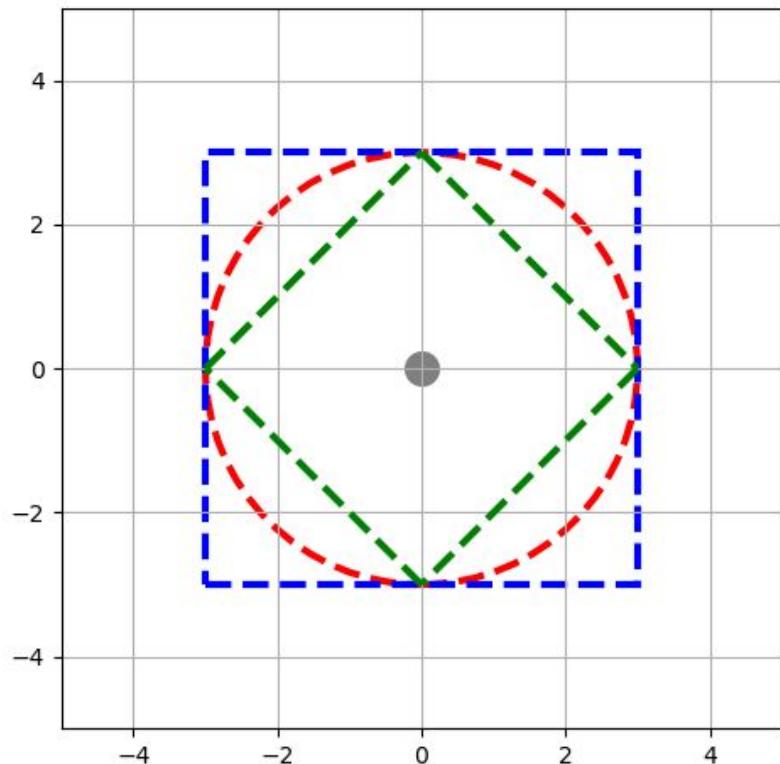
Possible but there are risks

**D**

Setting  $k$  to a prime  
is always preferable

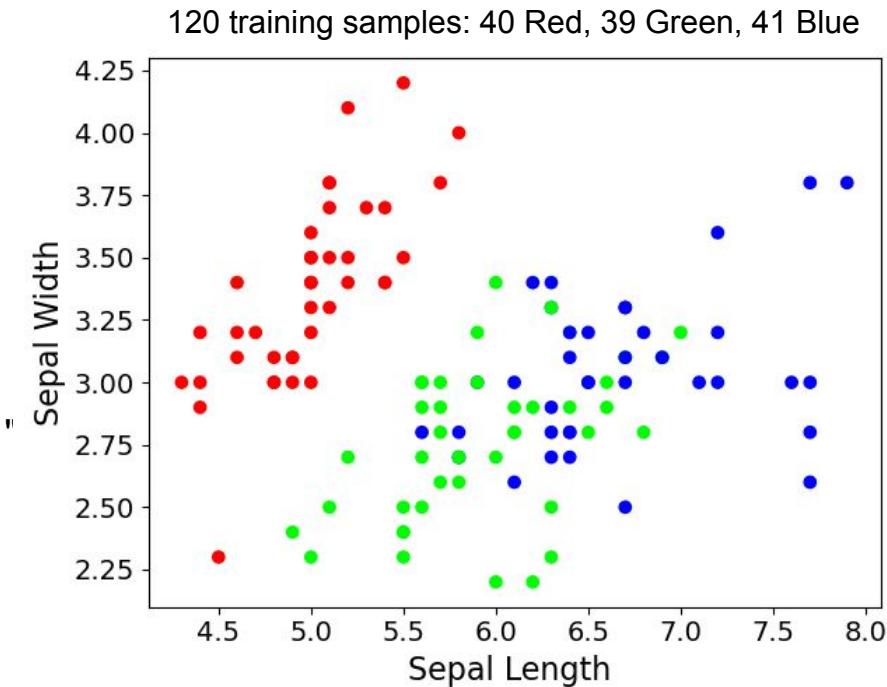
# KNN for Classification — Distance Metrics

- Example for different distance metrics
  - Euclidean distance
  - Manhattan distance
  - Chebyshev distance
- Other metrics, e.g.:
  - Cosine similarity
  - Jaccard similarity
  - ...
  - User-defined metrics



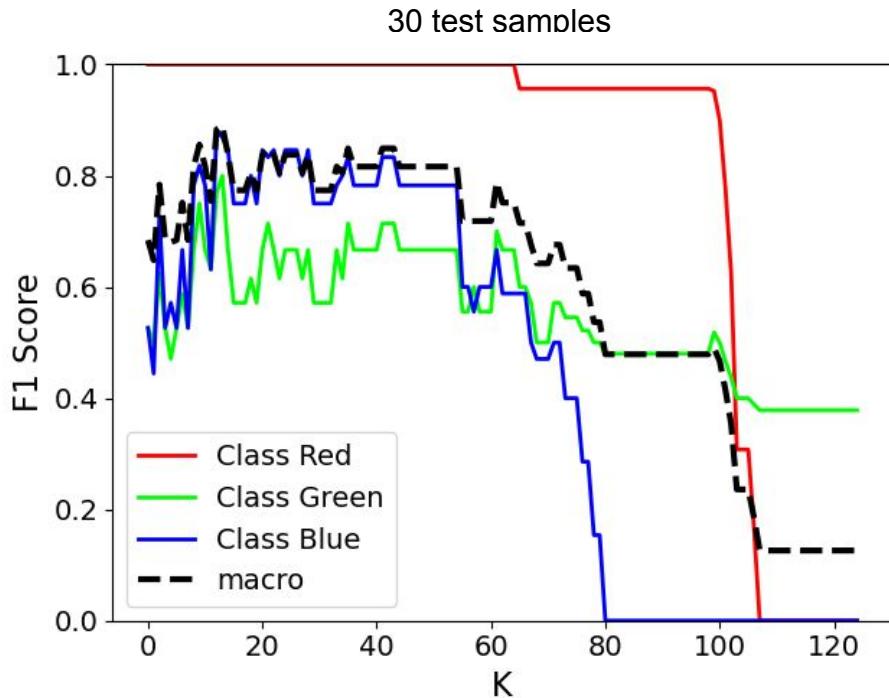
# KNN for Classification — Example

- IRIS dataset
  - 3 classes, 50 samples per class
  - 4 continuous features, but only 2 used: (sepal length & width)
- Basic EDA
  - Class "Red" well separated
  - A lot of overlap between Classes "Green" and "



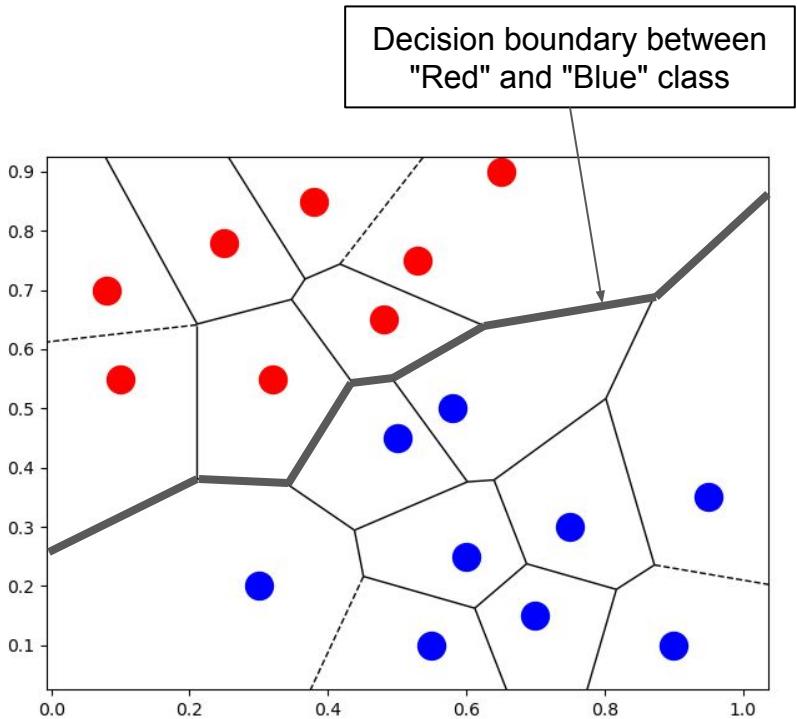
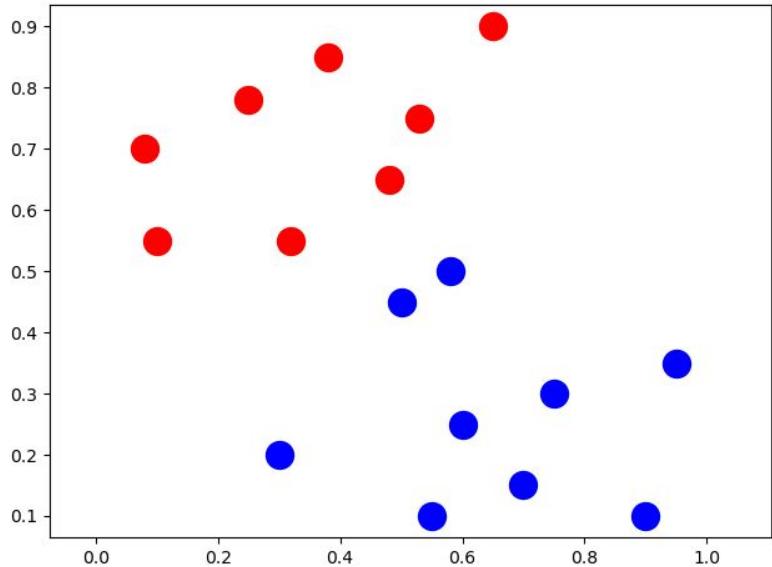
# KNN for Classification — Example

- Common outcomes
  - Different  $k$  yield different results
  - (Very) small  $k$  — results very sensitive to noise and outliers
  - Large  $k$  — insufficient capacity to properly sep
  - Very large  $k$  — most frequent class in training data starts to dominate

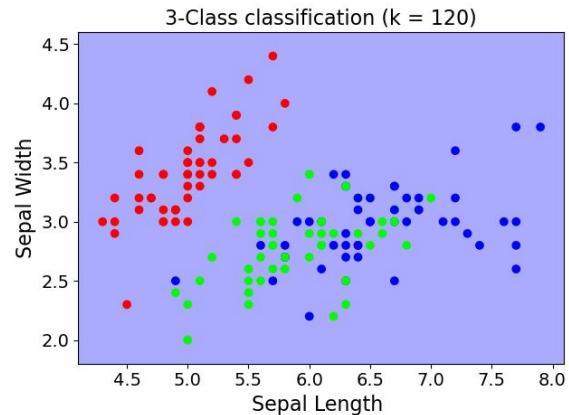
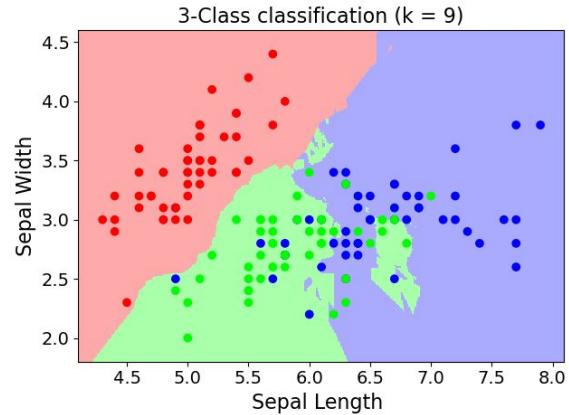
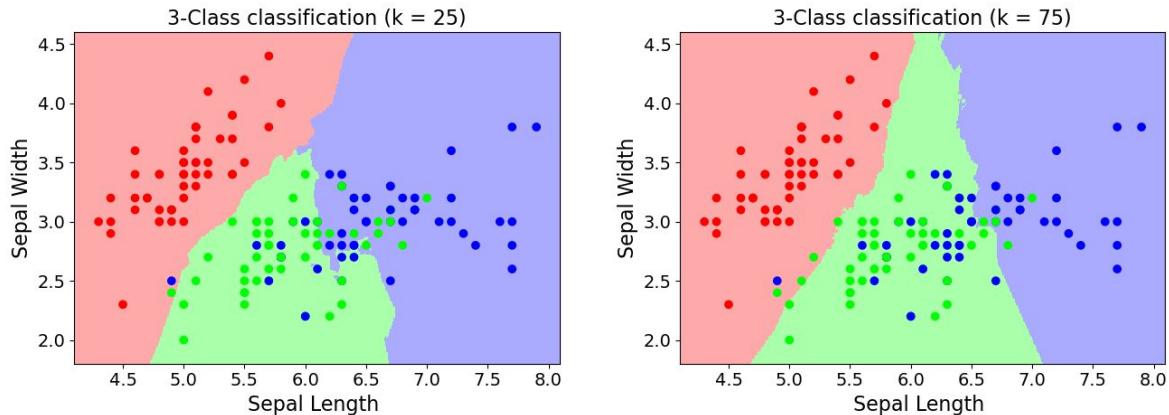
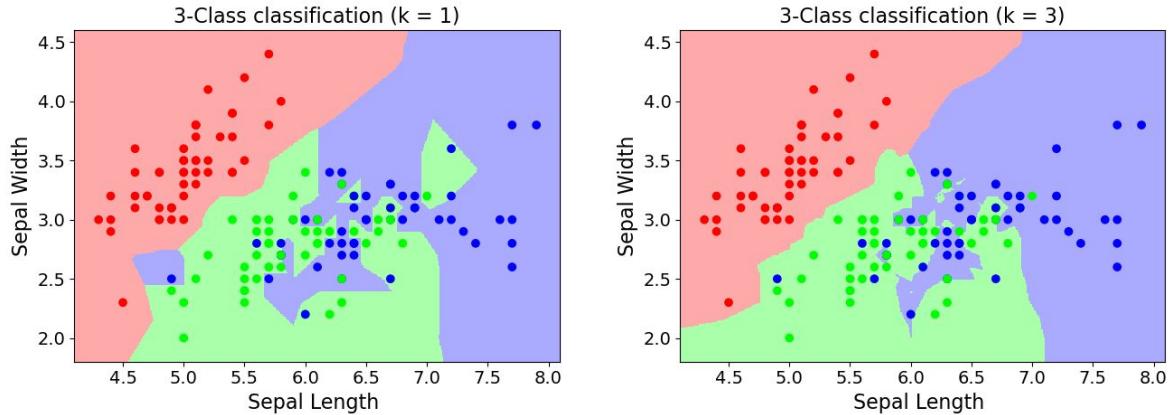
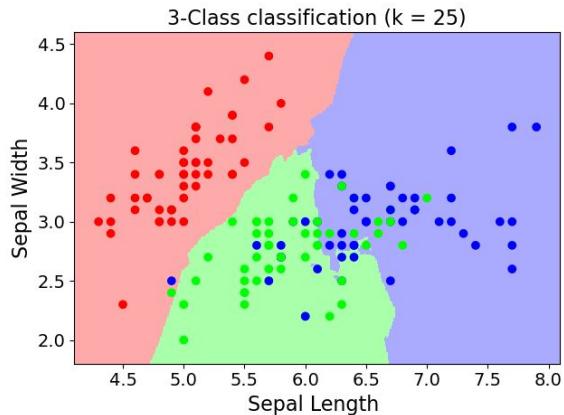
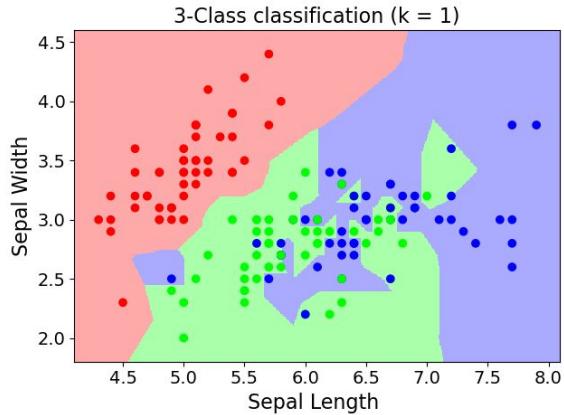


# 1-Nearest Neighbor: Voronoi Tessellation

- Decision boundaries for 1-NN =derived from Voronoi Tessellation
  - Separation of space into cells
  - Cell: set of points nearest to a single data point

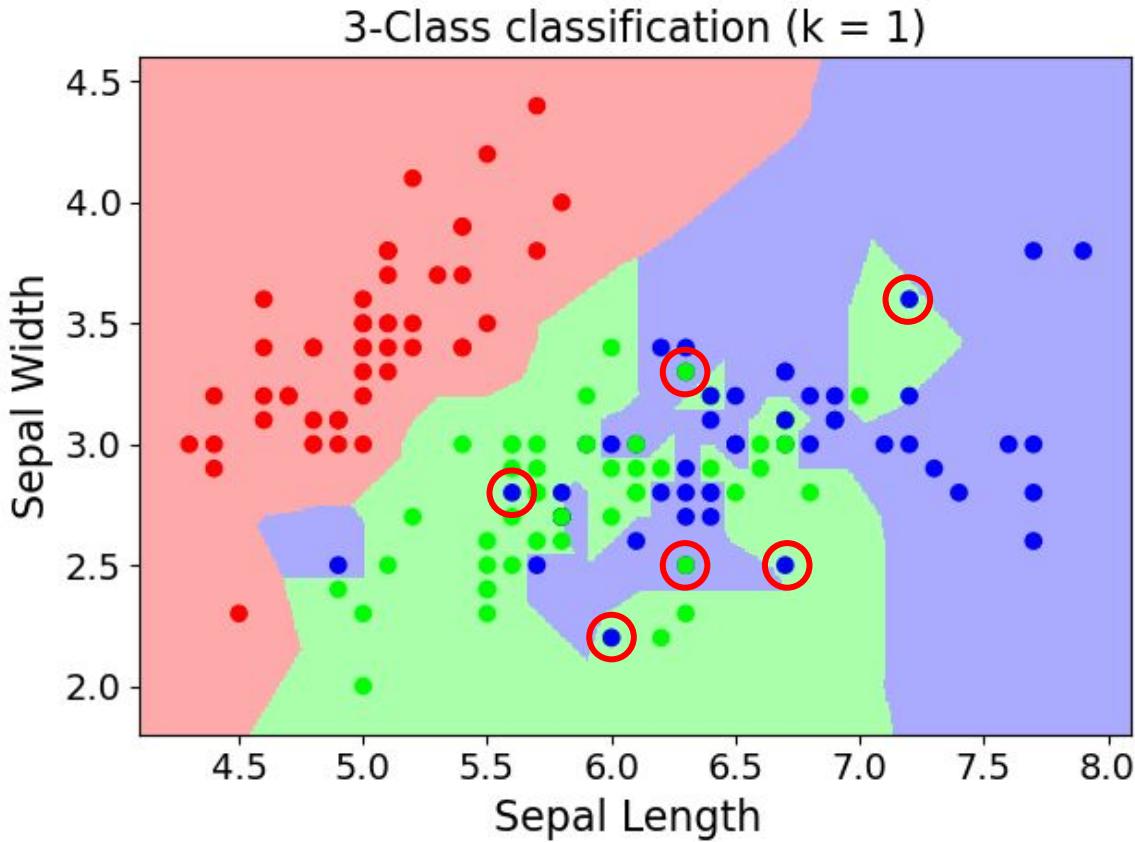


# KNN for Classification — Example



# Quick Quiz

All data points come  
from the **training data**:  
What's going on there?



# KNN for Regression

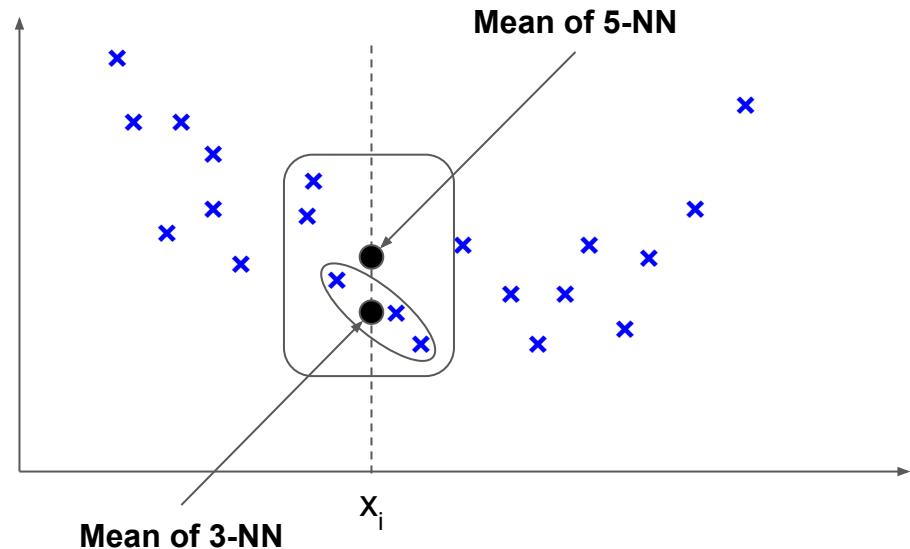
- "Training"
  - Remember training data
- Prediction for unseen point  $x_i$

- Calculate distances to all training data points  $x_j$ , e.g.:

$$d(p, q) = \sqrt{\sum_i^d (q_i - p_i)^2}$$

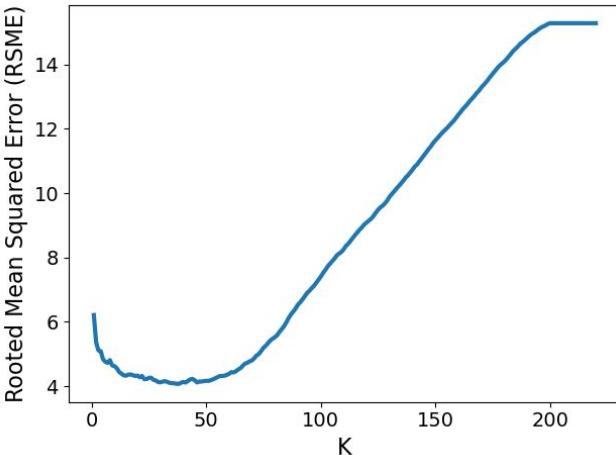
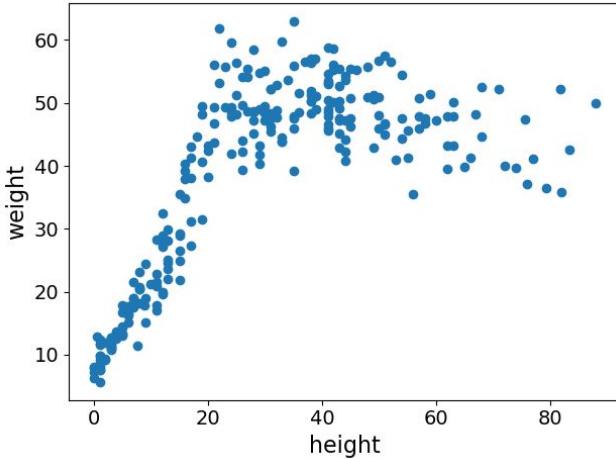
Euclidean distance,  $d = \# \text{features}$

- Get the  $k$ -nearest neighbors
- **Label of  $x_i$  = mean of scores of all  $k$ -nearest neighbors**

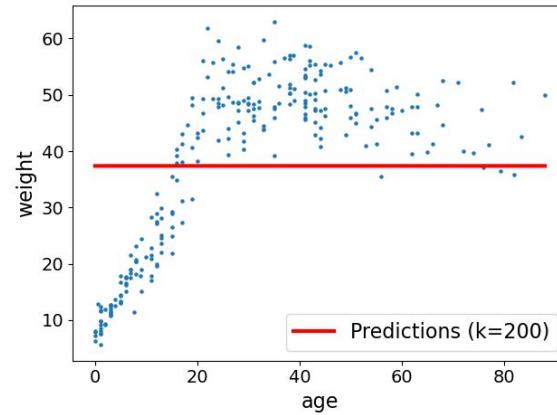
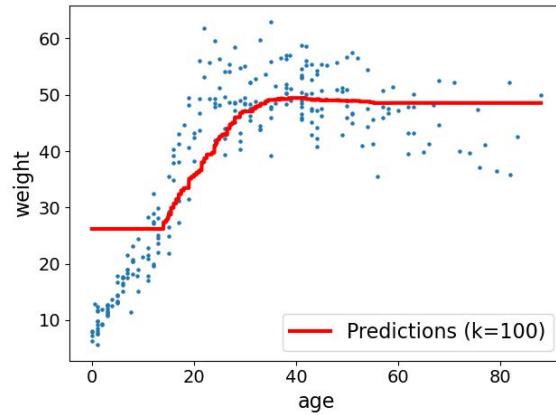
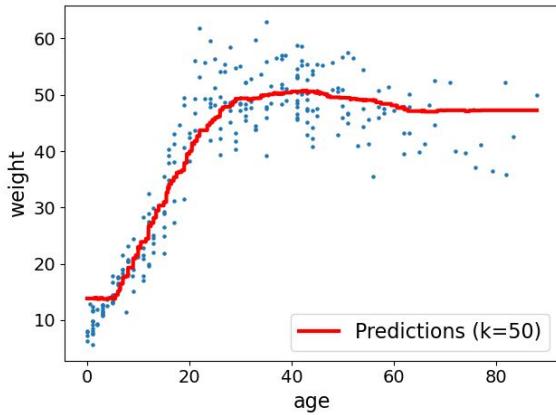
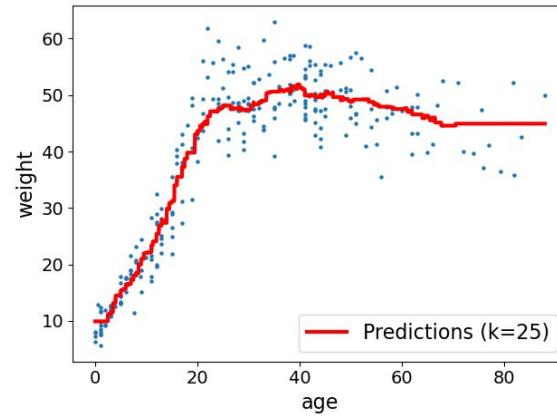
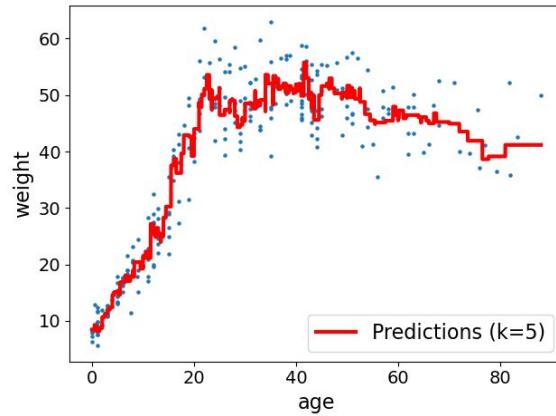
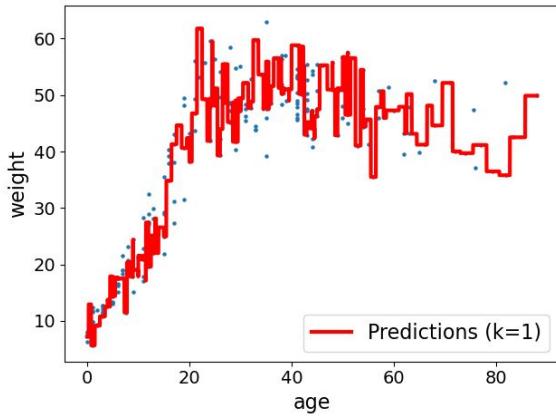


# KNN for Regression — Example

- "age vs. weight" dataset
  - age and weight of 257 males
  - 200 training samples, 57 test samples
- Common outcomes (again)
  - Different  $k$  yield different scores
  - (Very) small  $k$  — predicted scores very sensitive to noise and outliers
  - Large  $k$  — mean over too many neighbors
  - Very large  $k$  — predicted scores converge to the mean over



# KNN for Regression — Example



# Choice of $k$ — Summary

- $k$  too small
    - Predictions sensitive to noise/outliers
    - Very uneven decision boundaries  
(or regression lines)
- }
- Risk of **overfitting**
- 
- $k$  too large
    - Unable to capture local patterns
    - Very smooth decision boundaries  
(or regression lines)
- }
- Risk of **underfitting**

# Outline

- Classification & Regression
  - Overview & Examples
  - Basic Setup
  - Evaluation
- Nearest Neighbor Methods
  - KNN — K-Nearest Neighbor Algorithm
  - Pros, Cons & Caveats

# Pros & Cons...and Caveats

- Pros

- Very simple and intuitive algorithm — but often surprisingly good performance!
- Generic algorithm — applicable as long as distance between points can be "meaningfully" measured
- Can produce arbitrarily shaped decision boundaries
- No training time

- Cons

- Finding neighbors at test time may be slow  
(in practice: data structures and auxiliary algorithms to speed up k-NN search)
- All training data need to be stored

- Caveats

- "...applicable as long as distance between points can be "meaningfully" measured"

# Caveat 1: Feature Values (Range, Magnitude)

- Euclidean distance sensitive to range and magnitude of attribute values

id	weight (kg)	height (cm)	sex
A	80	165	male
B	55	180	female
C	70	180	???

$$d(A, C) = \sqrt{(70 - 80)^2 + (180 - 165)^2} = 18.0$$

$$d(B, C) = \sqrt{(70 - 55)^2 + (180 - 180)^2} = 15.0$$

→ C classified as female

id	weight (kg)	height (m)	sex
A	80	1.65	male
B	55	1.80	female
C	70	1.80	???

$$d(A, C) = \sqrt{(70 - 80)^2 + (1.80 - 1.65)^2} = 10.0$$

$$d(B, C) = \sqrt{(70 - 55)^2 + (1.80 - 1.80)^2} = 15.0$$

→ C classified as male

# Data Normalization

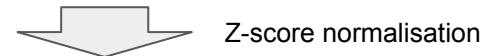
- Z-score normalisation, e.g.:

$$x_i^{\text{weight}} = \frac{x_i^{\text{weight}} - \mu^{\text{weight}}}{\sigma^{\text{weight}}}$$

- Min-max normalization, e.g.:

$$x_i^{\text{weight}} = \frac{x_i^{\text{weight}} - \min(x^{\text{weight}})}{\max(x^{\text{weight}}) - \min(x^{\text{weight}})}$$

id	weight (kg)	height (m)	sex
A	80	1.65	male
B	55	1.80	female
C	70	1.80	???



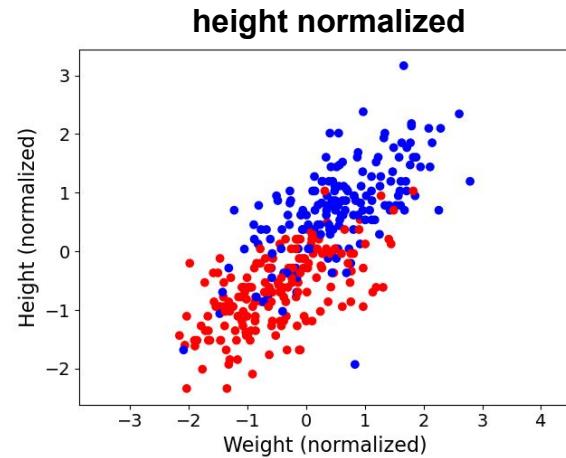
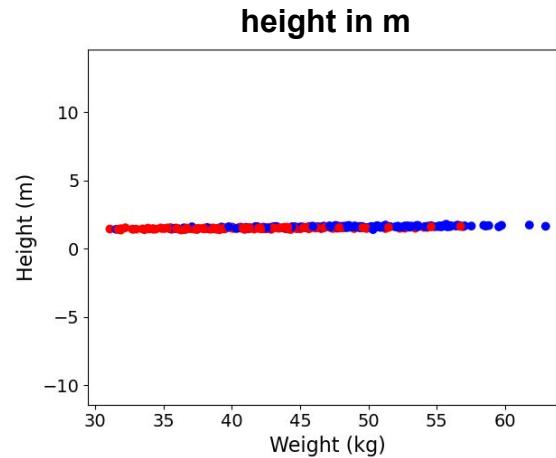
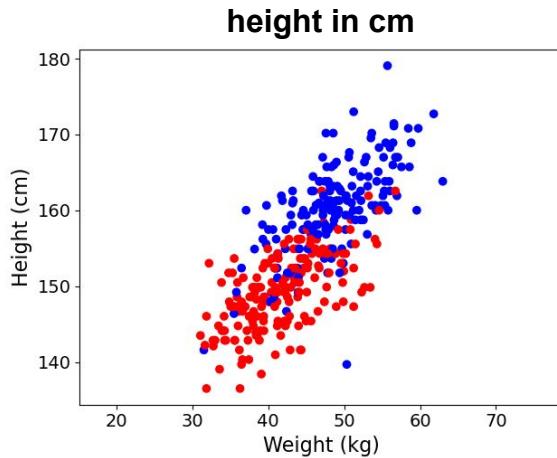
id	weight	height	sex
A	1.14	-1.41	male
B	-1.30	0.71	female
C	0.16	0.71	???

$$d(A, C) = 2.33, \quad d(B, C) = 1.46$$

→ C classified as female

# Data Normalization — Visualized

- (weight, height) data points for 187 females (red) and 165 males (blue)

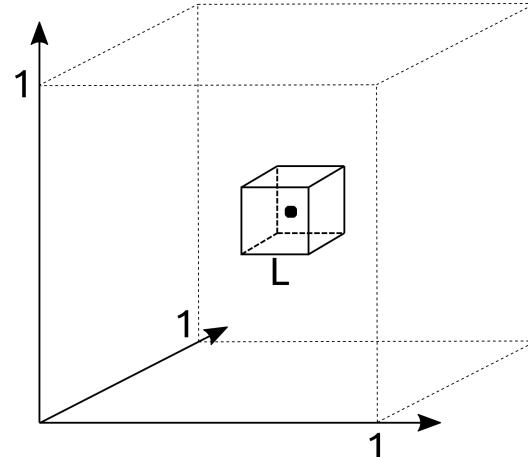


**Note:** Normalization assumes that all features are equally important. This may not always be true!

# Caveat 2: Curse of Dimensionality

- **Effect of high dimensions** (i.e., many features)
  - Data points tend to never be close together
  - Average distance between points converges
- **Intuition**
  - Assume  $N$  data points uniformly distributed within a unit cube with  $d$  dimensions
  - Let  $L$  be the length of the smallest cube containing the  $k$ -NN of a data point

$$L^d \approx \frac{k}{N} \Rightarrow L \approx \left(\frac{k}{N}\right)^{\frac{1}{d}}$$



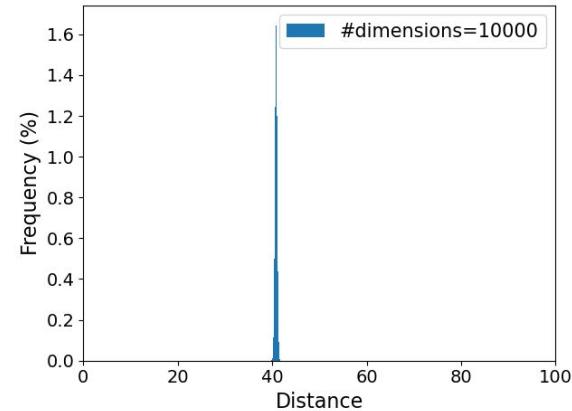
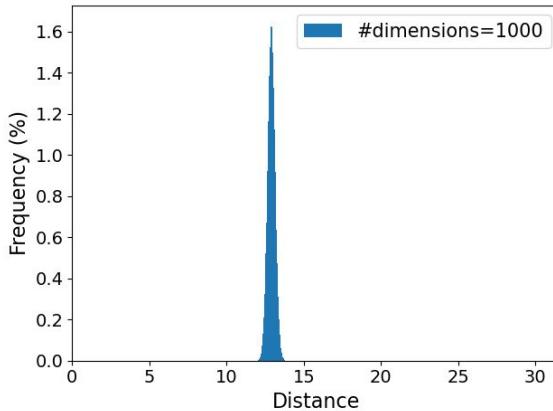
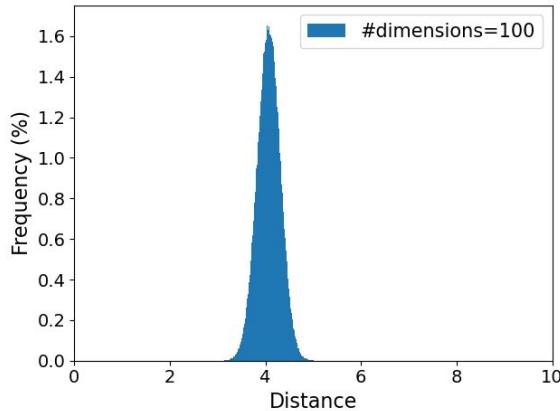
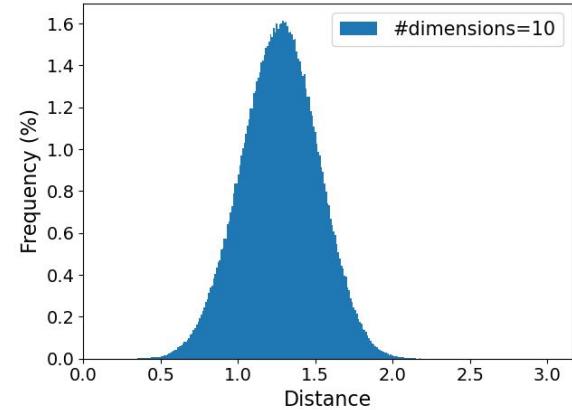
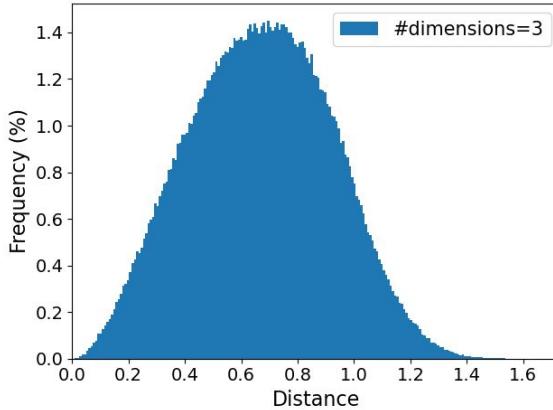
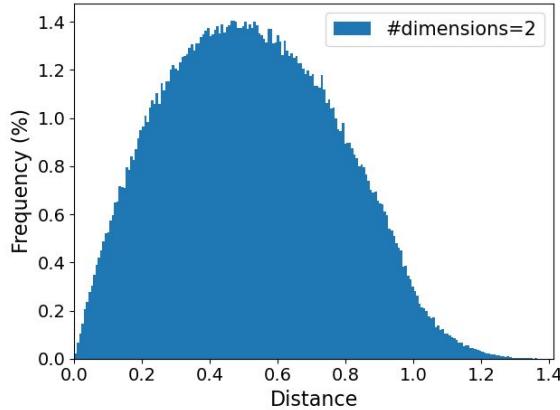
$N=1,000, k=10$

$d$	$L$
2	0.100
3	0.215
10	0.631
100	0.955
1,000	0.995
10,000	0.999

**The cube with the  $k$ -NN is almost the whole unit cube!**

# Curse of Dimensionality

Distribution of pairwise distances between 1,000 random data points and different number of dimensions



# Caveat 3: Non-Numerical Data

- In practice: Features often categorical
  - Ordinal — education level, grades, etc.
  - Nominal — sex, marital status, etc.
- Basic approach: Convert categorical features into numerical features
  - Allows direct use of common distance metrics
  - Does not automatically yield good results

Age	Sex	Edu- cation	Marital Status
23	male	Masters	Single
35	male	College	Married
26	female	Masters	Single
41	female	PhD	Single
18	female	Poly	Single
55	male	Poly	Divorced
30	female	College	Single
35	male	PhD	Married
28	male	Masters	Married
45	female	Masters	Married

# Caveat 3: Non-Numerical Data

- **Binary features**

- Special case of categorical data
- Convert into binary variable 0/1 to indicate absence/presence



Sex
male
male
female
female
female

Sex
0
0
1
1
1

- **Ordinal features**

- Utilize natural order of feature values
- Convert into numerical values while preserving their original order



Education
Masters
College
Masters
PhD
Poly

Education
4
2
4
5
1

# Caveat 3: Non-Numerical Data

- **Nominal features** (with  $n$  different values)
  - **One-hot encoding:** convert nominal feature into  $N$  binary features
  - Convert each new binary feature into binary variable 0/1 to indicate absence/presence
  - Increases dimensionality!  
(particularly if  $N$  is very large)

Marital Status
Single
Married
Single
Single
Single
Divorced



Single	Married	Divorced
1	0	0
0	1	0
1	0	0
1	0	0
1	0	0
0	0	1

# Caveat 3: Non-Numerical Data

- Discussion / Limitations

- One-hot encoding increases dimensionality of data
- Distance between ordinal values often not intuitive

$$5 \text{ (PhD)} - 4 \text{ (Masters)} = 2 \text{ (College)} - \text{Poly (1)}$$

- Typically requires data normalization  
(particularly when combined with numerical features)

Education
Masters
College
Masters
PhD
Poly



Education
4
2
4
5
1

- Alternative approach: Custom distance metric

- Design metric that appropriately quantifies distance between categorical values
- Typically very specific to given dataset and application context.

# Caveat 4: Semantic vs. Low-Level Similarity

- Unstructured data  
(text, image, video, audio)
  - Object is often more than just the sum of its parts
- Example (dog food ad)
  - Pixelwise similarity large; easy to compute
  - Semantic similarity small

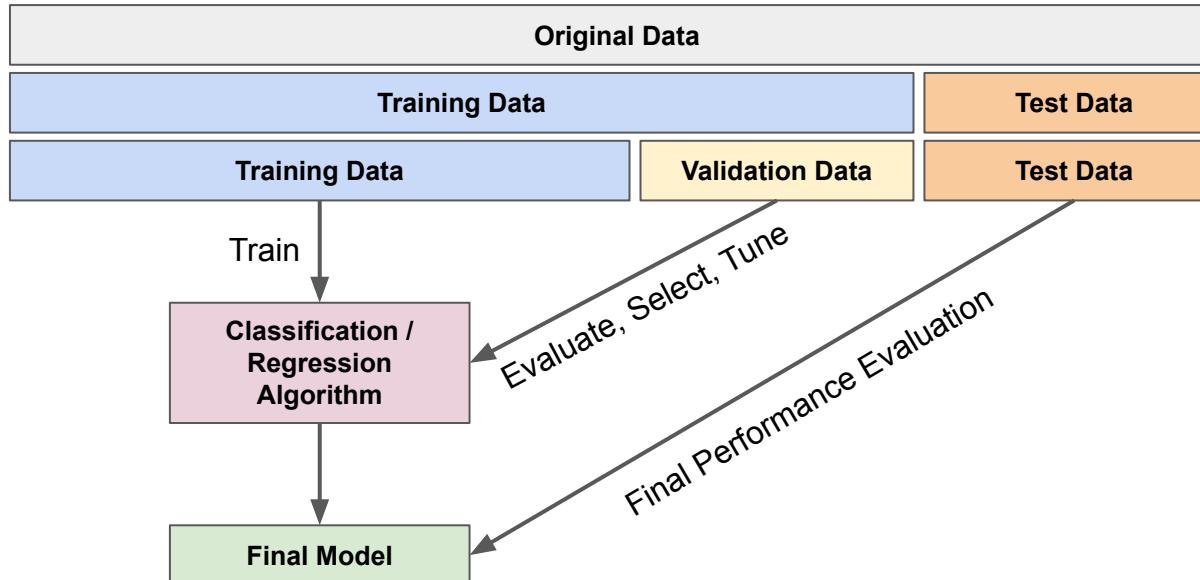


# Supervised Training/Learning — Extended Setup

- Recall basic setup: training data + test data
  - Building a good classifier or regressor
    - Find the best data preprocessing steps
    - Find the best model (model selection)
    - Find the best hyperparameter values
  - **Important:** Not allowed to use test data for this!
    - Test data is supposed to contain truly unseen data
    - Test data no longer unseen when used to optimize/tune model  
(for example, results might be different for a different training-test split)
- 
- The diagram illustrates the iterative nature of building a good classifier or regressor. It shows a list of bullet points. The second bullet point, 'Building a good classifier or regressor', contains three sub-points: 'Find the best data preprocessing steps', 'Find the best model (model selection)', and 'Find the best hyperparameter values'. A large brace is positioned to the right of these three sub-points, grouping them together. An arrow points from the end of the brace to the text 'Iterative evaluation required', indicating that the process of finding the best preprocessing, model, and hyperparameters requires iterative evaluation.
- Iterative evaluation required

# Training & Evaluation Process Using Validation Data

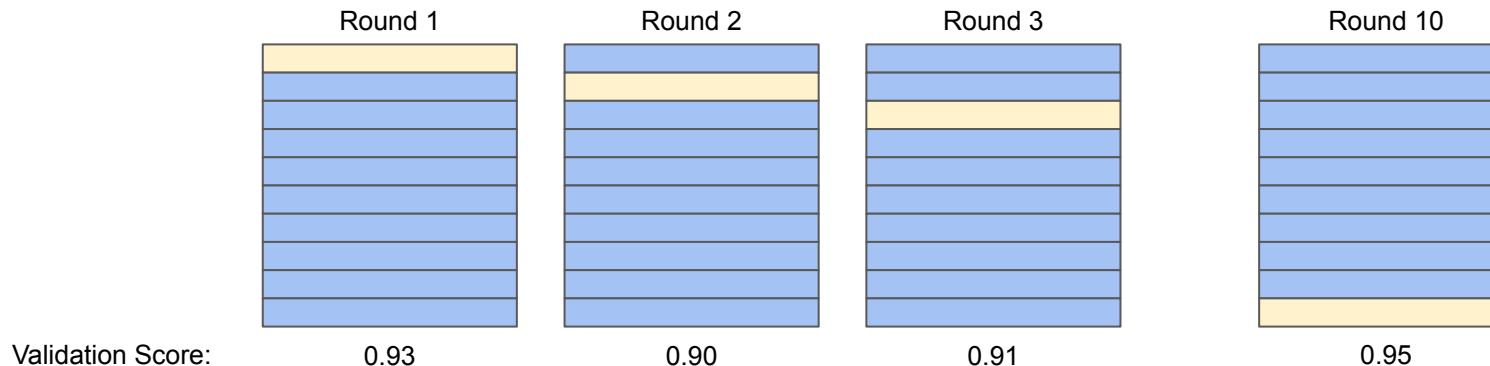
- Common data split to ensure generalizability of results
  - Use test data only at the very end to evaluate performance of final model
  - Use validation data for model selection and hyperparameter tuning



# K-Fold Cross Validation (just one of many validation techniques)

- Core idea of  $k$ -fold cross validation

- Split training data into  $k$  blocks of equal size (e.g.,  $k=10$ )
- Use  $(k-1)$  blocks for training and remaining block for evaluation
- Repeat for  $k$  rounds with different permutations



- Advantages

- Averaged results more reliable
- Variance between results useful indicator

# Information Leakage through Normalization

- Important guidelines
  - Do not normalize before splitting dataset into training and test data!
  - Normalize training and test data but only based on training data!
- Example (pseudo) code using scikit-learn

Why?

```
# Split input data into training and test set
X_train, X_test = split(X, 0.2)

# Fit StandardScaler (i.e., calculate mean and variance)
scaler = preprocessing.StandardScaler().fit(X_train) # CORRECT!
#scaler = preprocessing.StandardScaler().fit(X)        # WRONG!!!

# Fit both training and test data
X_train_transformed = scaler.transform(X_train)
X_test_transformed = scaler.transform(X_test)
```

# Summary

- **Evaluation of classifiers** (straightforward for regressors)
  - Different metric with different interpretations
  - Applicability of metrics depending on data and task
- **K-Nearest Neighbor (KNN) classifier/regressor**
  - Very intuitive supervised learning model
  - Limited applicability for (very) large datasets  
(heavy lifting done during prediction time not the training time)
  - Choice of hyperparameter  $k$  important to address risk of overfitting and underfitting

# Solutions to Quick Quizzes

- Slide 29: D
- Slide 30: Technically D, but A is the more practical result
- Slide 37: B
- Slide 41: B
- Slide 47: Duplicate data points
- Slide 67: Otherwise, risk of data leakage
  - Mean and standard deviation will be affected by test data

# **CS5228: Knowledge Discovery and Data Mining**

## Lecture 6 — Classification & Regression II

# Course Logistics — Update

- Assignment 2
  - Submission deadline: Oct 03, 11.59 pm
  - Don't forget to check the Discussion and Errata page on Canvas
- Midterm
  - Check new Canvas page for midterm exam
  - Report any issues with the practice exam early enough
  - If needed, 2nd practice exam during Recess Week
  - Coming up: survey regarding request for loaner laptop
- Project
  - Submission deadline for progress report: Oct 10, 11.59 pm

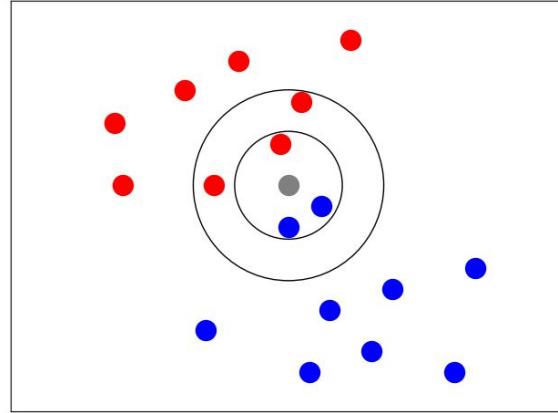
# Quick Recap — Classification & Regression

- Pattern of interest
  - Matching or function between input features and output
  - Goal: use matching to predict outputs for unseen samples
  - Categorical output → classification
  - Numerical output → regression
- Important: Evaluation of predictions
  - Straightforward for regression
  - Series of metrics for classification  
(accuracy, recall, precision, f1, AUC-ROC)

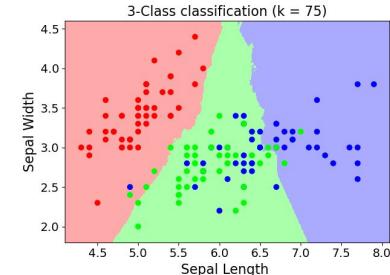
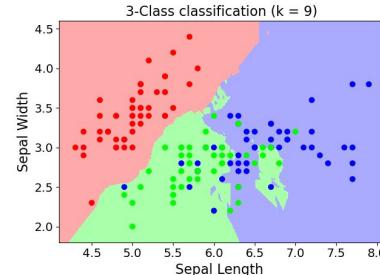
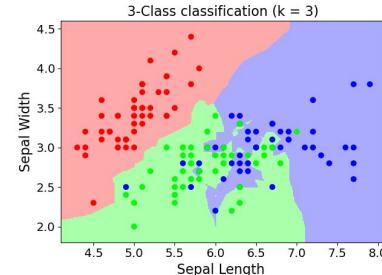
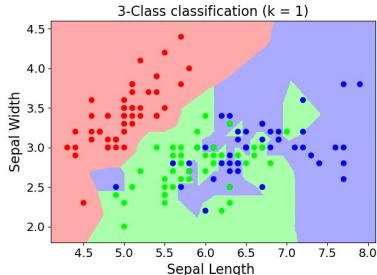
Age	Edu- cation	Marital Status	Income Level	Credit Approval	Credit Limit
23	Masters	Single	Mid	No	\$S5,000
35	College	Married	High	Yes	\$S7,000
26	Masters	Single	High	No	\$S9,000
41	PhD	Single	Mid	Yes	\$S5,000
18	Poly	Single	Low	No	\$S6,000
55	Poly	Married	High	Yes	\$S10,000
30	College	Single	High	Yes	\$S8,000
35	PhD	Married	High	Yes	\$S10,000
28	Masters	Married	Mid	Yes	\$S5,000
45	Masters	Married	Mid	???	???

# Quick Recap — KNN Algorithm

- Intuition behind KNN:
  - Label of an unseen data point  $x$  derives from the labels of the  $k$ -nearest neighbors of  $x$
  - Similar data points → similar labels
  - Caveats due to reliance of similarity metric



- Effects of hyperparameter  $k$ 
  - Tradeoff between (risks of) underfitting and overfitting



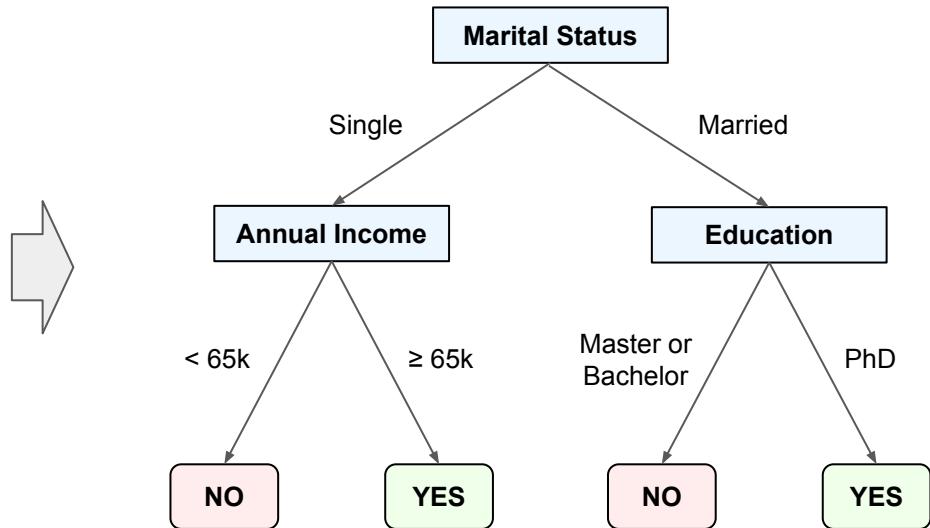
# Outline

- **Decision Trees**
  - Overview
  - Training Decision Trees
  - Overfitting
- **Tree Ensembles**
  - Bagging
  - Random Forest
  - Boosting

# Decision Tree

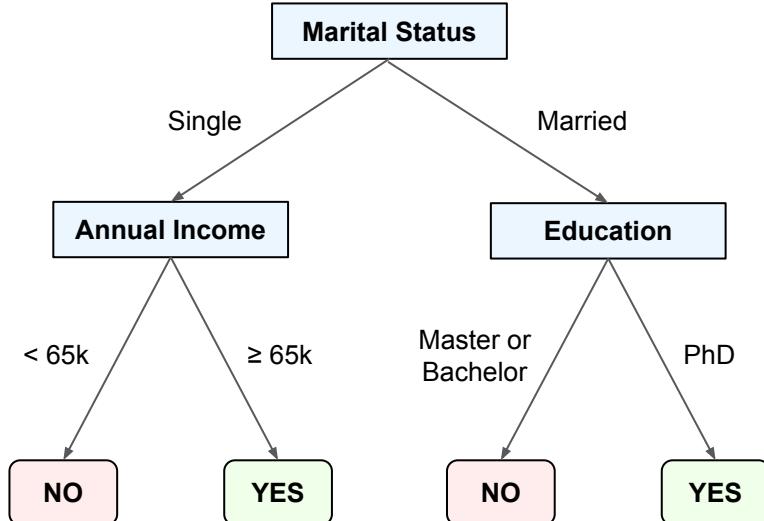
- Example: Decision Tree for classification

Age	Education	Marital Status	Annual Income	Credit Approval
23	Masters	Single	75k	Yes
35	Bachelor	Married	50k	No
26	Masters	Single	70k	Yes
41	PhD	Single	95k	Yes
18	Bachelor	Single	40k	No
55	Masters	Married	85k	No
30	Bachelor	Single	60k	No
35	PhD	Married	60k	Yes
28	PhD	Married	65k	Yes



# Decision Tree

- Decision Tree — idea
  - Represent mapping between features and label/value as flowchart-like structure
- Components (a bit simplified at the moment)
  - (Inner) node — test on a single feature
  - Branch — outcome of a test; corresponds to a feature values or range of values
  - Leaf — label (classification) or real value (regression)



# Decision Tree — Application to Unseen Data

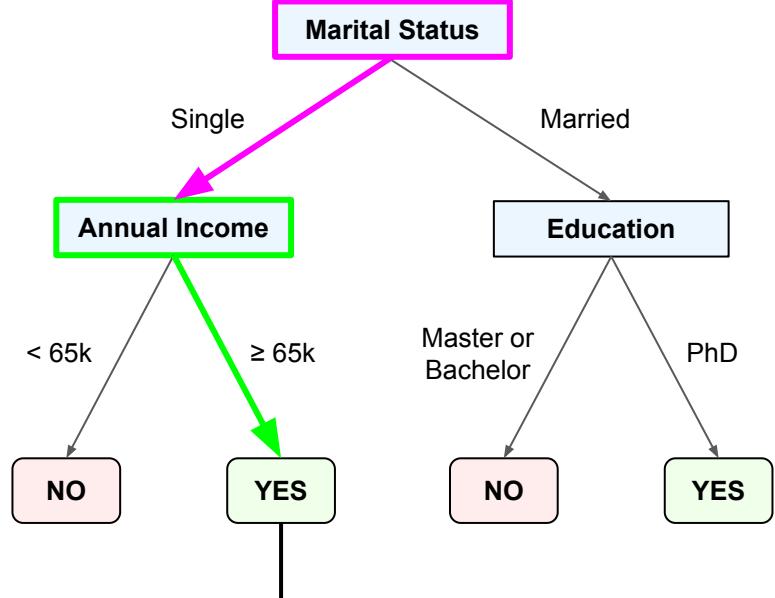
Age	Education	Marital Status	Annual Income	Credit Approval
50	PhD	Single	70k	???



Age	Education	Marital Status	Annual Income	Credit Approval
50	PhD	Single	70k	???



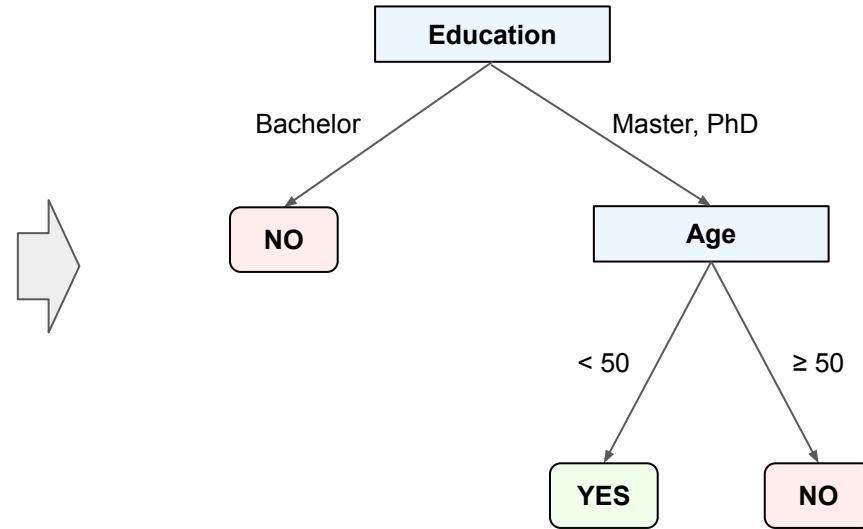
Age	Education	Marital Status	Annual Income	Credit Approval
50	PhD	Single	70k	YES



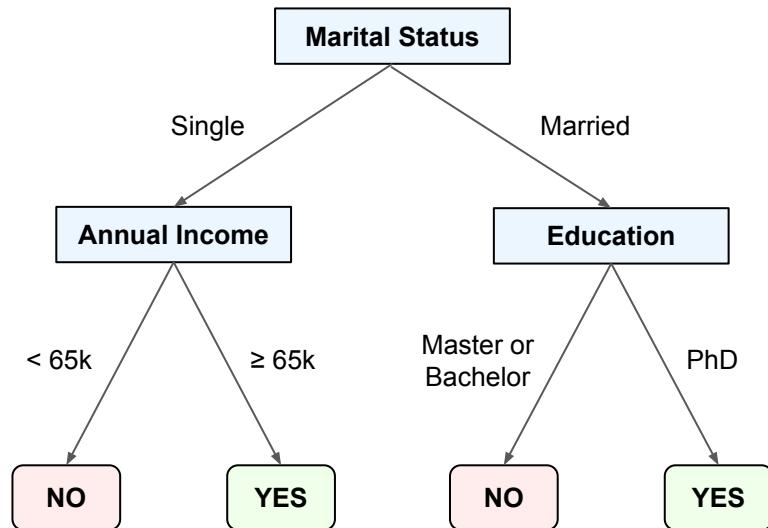
# Decision Tree

- Same dataset, different Decision Tree
  - In general, there are multiple trees that match a dataset

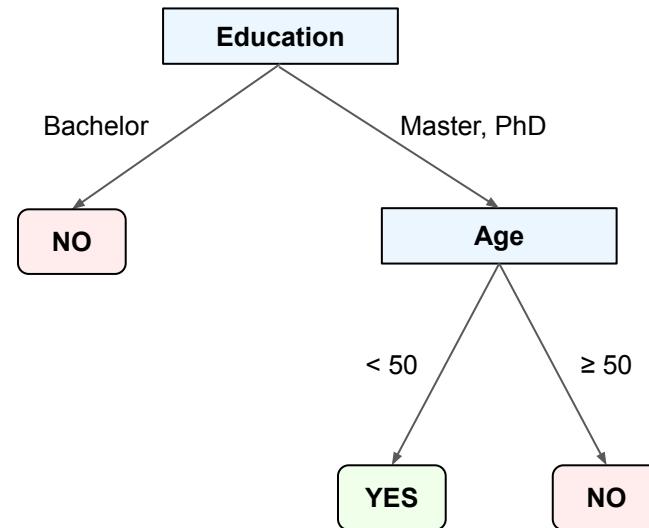
ID	Age	Edu- cation	Marital Status	Annual Income	Credit Approval
1	23	Masters	Single	75k	Yes
2	35	Bachelor	Married	50k	No
3	26	Masters	Single	70k	Yes
4	41	PhD	Single	95k	Yes
5	18	Bachelor	Single	40k	No
6	55	Masters	Married	85k	No
7	30	Bachelor	Single	60k	No
8	35	PhD	Married	60k	Yes
9	28	PhD	Married	65k	Yes



# Which Decision Tree is Better?



Age	Education	Marital Status	Annual Income	Credit Approval
50	PhD	Single	70k	Yes



Age	Education	Marital Status	Annual Income	Credit Approval
50	PhD	Single	70k	No

# Quick Quiz

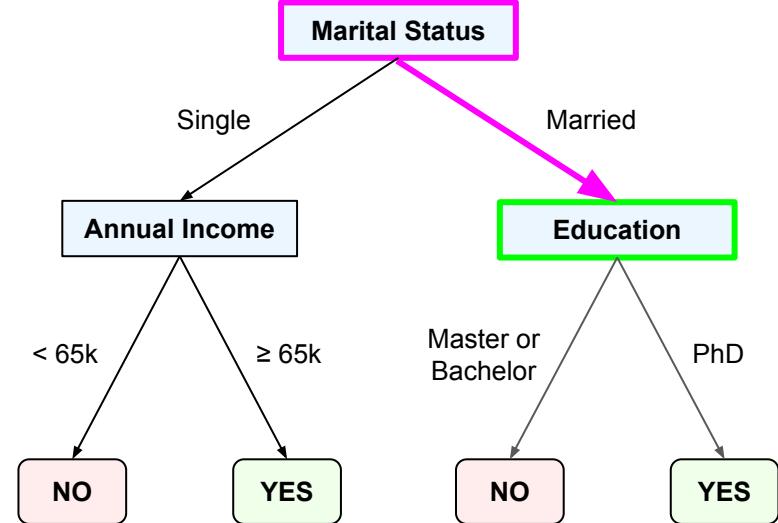
Age	Edu- cation	Marital Status	Annual Income	Credit Approval
50	PhD	Married	70k	???



Age	Edu- cation	Marital Status	Annual Income	Credit Approval
50	Poly	Single	70k	???

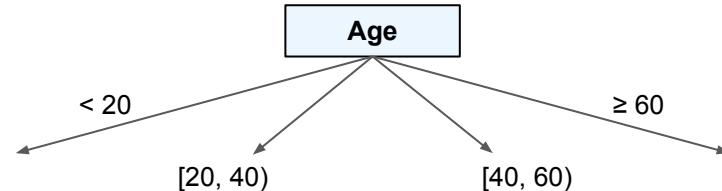
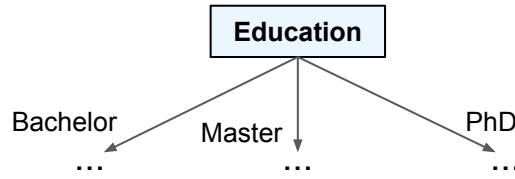


What to do in case  
of **unknown values**  
for a feature?



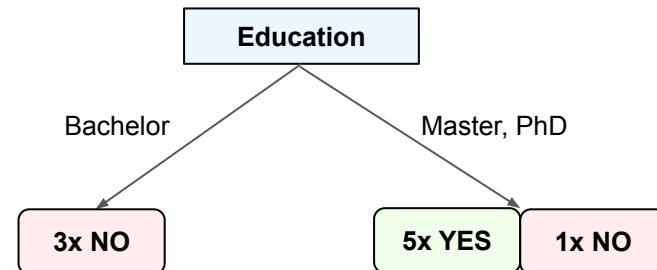
# Decision Tree — Diversity (Sneak Preview)

- Different branching factors



- Different depth

- Leaves can have more than one label or real value
- Required based on dataset or based on choice ( $\rightarrow$  Pruning)
- Final output: majority label (classification) or mean of values (regression)



# Outline

- **Decision Trees**
  - Overview
  - **Training Decision Trees**
  - Overfitting
- **Tree Ensembles**
  - Bagging
  - Random Forest
  - Boosting

# Building a Decision Tree

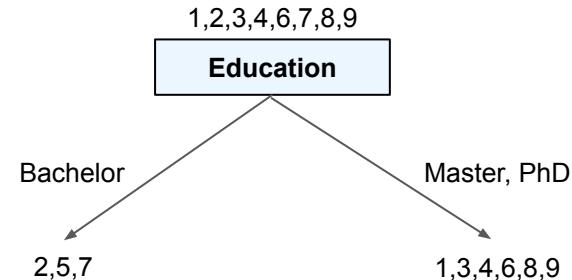
- Notations

- $D_t$  — set of records that reach node  $t$
- $D_0$  — set of all records at root node

- General procedure

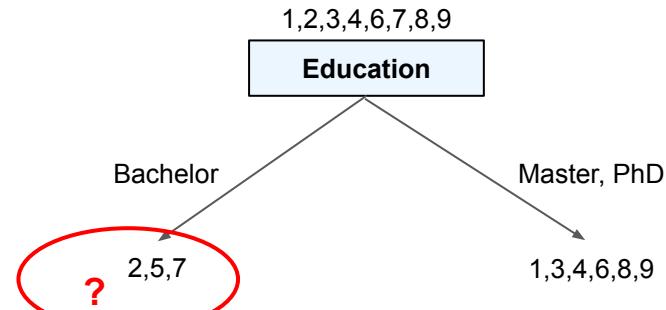
- If  $|D_t| = 1$  or all records in  $D_t$  have the same class or value  $\rightarrow t$  is leaf node
- Otherwise, choose test (feature + conditions) to split  $D_t$  into smaller subsets (i.e., subtrees)
- Recursively apply procedure to each subtree

ID	Age	Education	Marital Status	Annual Income	Credit Approval
1	23	Masters	Single	75k	Yes
2	35	Bachelor	Married	50k	No
3	26	Masters	Single	70k	Yes
4	41	PhD	Single	95k	Yes
5	18	Bachelor	Single	40k	No
6	55	Masters	Married	85k	No
7	30	Bachelor	Single	60k	No
8	35	PhD	Married	60k	Yes
9	28	PhD	Married	65k	Yes



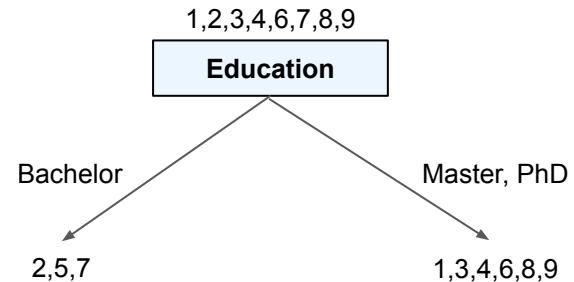
# Building a Decision Tree — Step-by-Step Example

ID	Age	Edu-cation	Marital Status	Annual Income	Credit Approval
1	23	Masters	Single	75k	Yes
2	35	Bachelor	Married	50k	No
3	26	Masters	Single	70k	Yes
4	41	PhD	Single	95k	Yes
5	18	Bachelor	Single	40k	No
6	55	Masters	Married	85k	No
7	30	Bachelor	Single	60k	No
8	35	PhD	Married	60k	Yes
9	28	PhD	Married	65k	Yes



# Building a Decision Tree — Step-by-Step Example

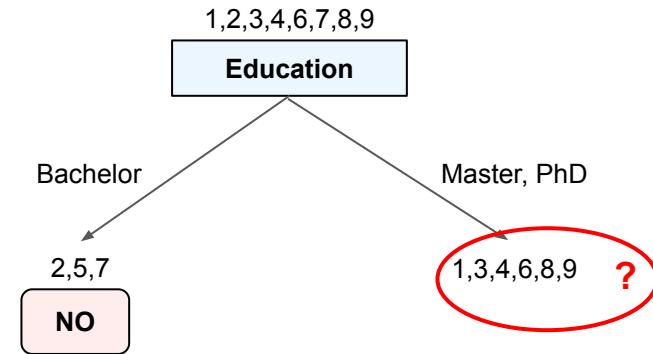
ID	Age	Edu-cation	Marital Status	Annual Income	Credit Approval
2	35	Bachelor	Married	50k	No
5	18	Bachelor	Single	40k	No
7	30	Bachelor	Single	60k	No



All the same labels → leaf node

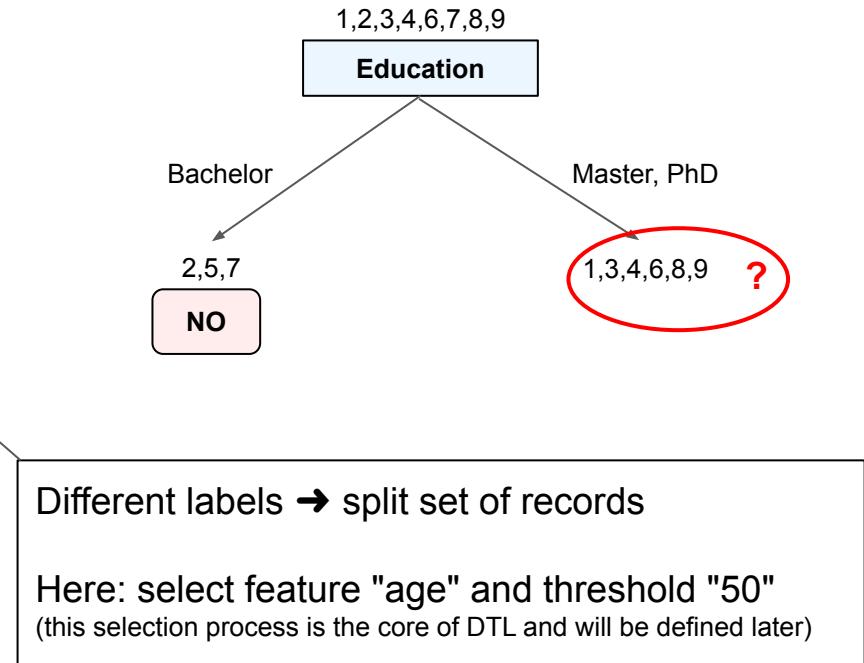
# Building a Decision Tree — Step-by-Step Example

ID	Age	Education	Marital Status	Annual Income	Credit Approval
2	35	Bachelor	Married	50k	No
5	18	Bachelor	Single	40k	No
7	30	Bachelor	Single	60k	No



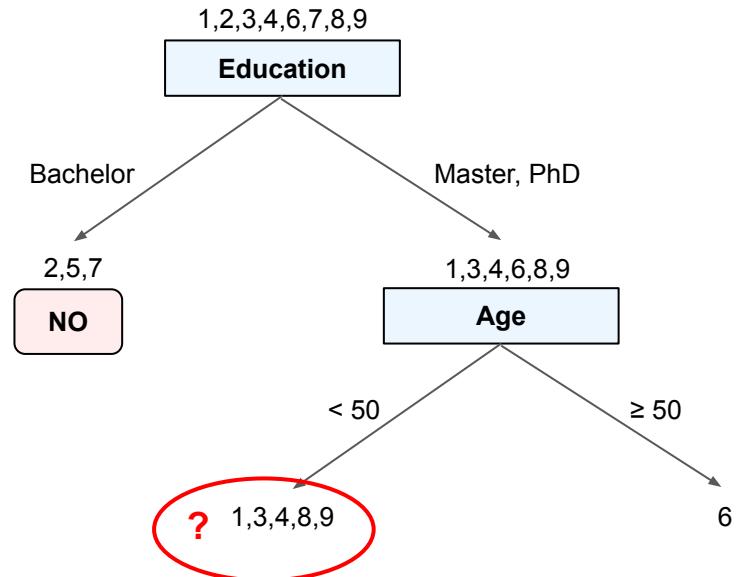
# Building a Decision Tree — Step-by-Step Example

ID	Age	Education	Marital Status	Annual Income	Credit Approval
1	23	Masters	Single	75k	Yes
3	26	Masters	Single	70k	Yes
4	41	PhD	Single	95k	Yes
6	55	Masters	Married	85k	No
8	35	PhD	Married	60k	Yes
9	28	PhD	Married	65k	Yes



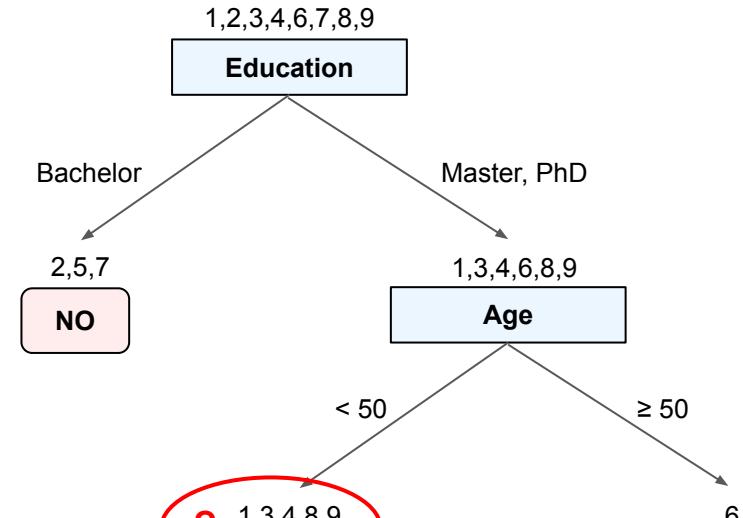
# Building a Decision Tree — Step-by-Step Example

ID	Age	Education	Marital Status	Annual Income	Credit Approval
1	23	Masters	Single	75k	Yes
2	30	Masters	Single	80k	No
3	26	Masters	Single	70k	Yes
4	41	PhD	Single	95k	Yes
5	32	PhD	Single	85k	No
6	55	Masters	Married	85k	No
7	38	PhD	Married	70k	Yes
8	35	PhD	Married	60k	Yes
9	28	PhD	Married	65k	Yes



# Building a Decision Tree — Step-by-Step Example

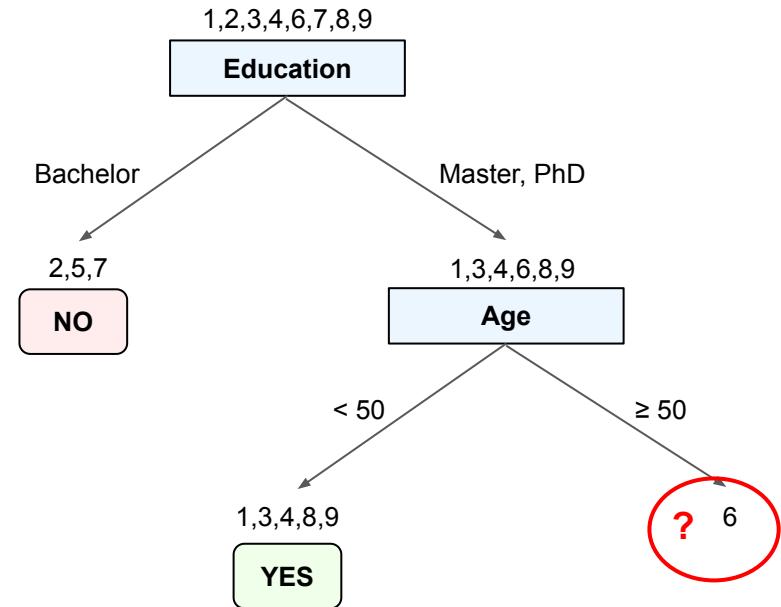
ID	Age	Education	Marital Status	Annual Income	Credit Approval
1	23	Masters	Single	75k	Yes
3	26	Masters	Single	70k	Yes
4	41	PhD	Single	95k	Yes
8	35	PhD	Married	60k	Yes
9	28	PhD	Married	65k	Yes



All the same labels → leaf node

# Building a Decision Tree — Step-by-Step Example

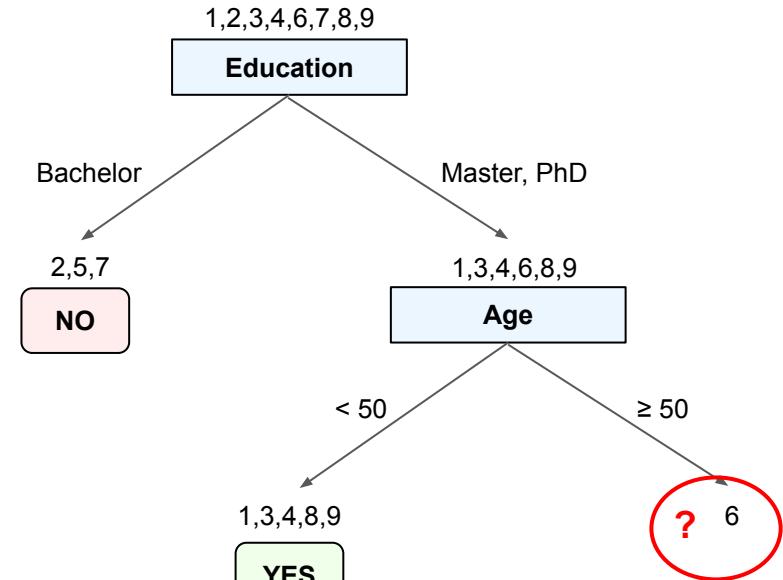
ID	Age	Education	Marital Status	Annual Income	Credit Approval
1	23	Masters	Single	75k	Yes
2	25	Masters	Single	60k	No
3	26	Masters	Single	70k	Yes
4	41	PhD	Single	95k	Yes
5	30	PhD	Married	80k	No
6	35	PhD	Married	60k	Yes
7	28	PhD	Married	65k	Yes
8	32	PhD	Married	70k	Yes
9	28	PhD	Married	65k	Yes



# Building a Decision Tree — Step-by-Step Example

ID	Age	Education	Marital Status	Annual Income	Credit Approval
6	55	Masters	Married	85k	No

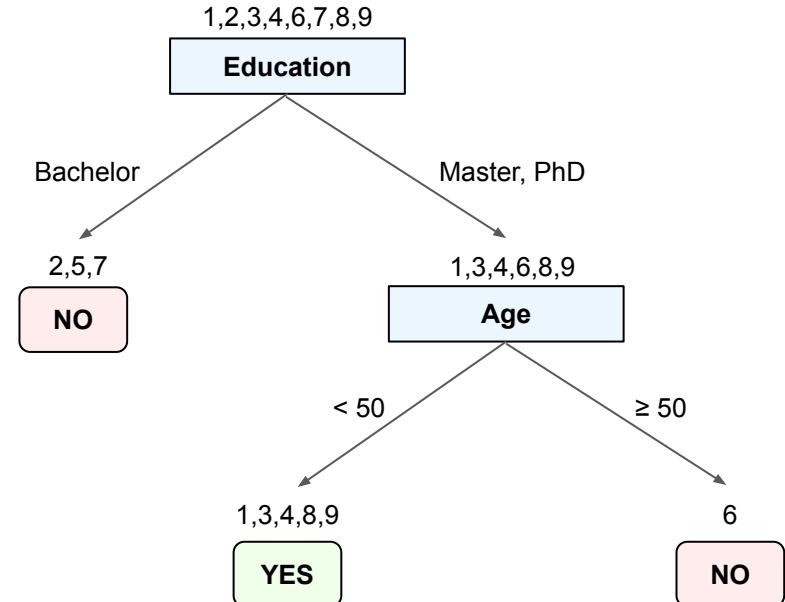
Only one record → leaf node



# Building a Decision Tree — Step-by-Step Example

ID	Age	Education	Marital Status	Annual Income	Credit Approval

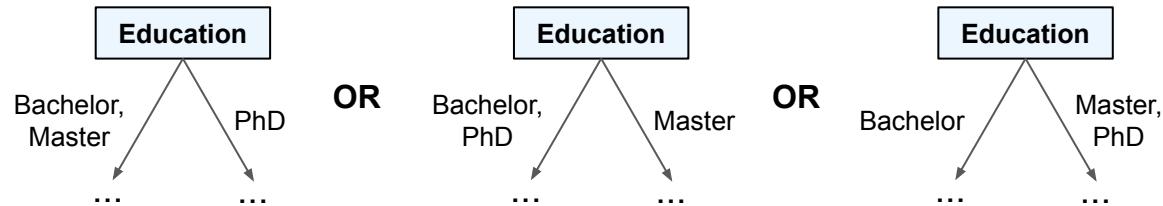
All records covered → Done!



# How to Split? — Nominal Attributes

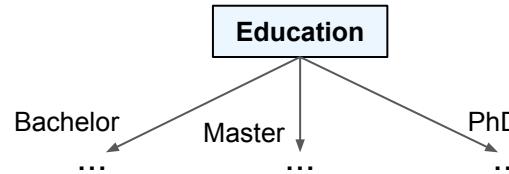
- **Binary split**

- Partition all  $d$  values into two subsets
- $\frac{2^d - 2}{2}$  possible splits



- **Multiway split**

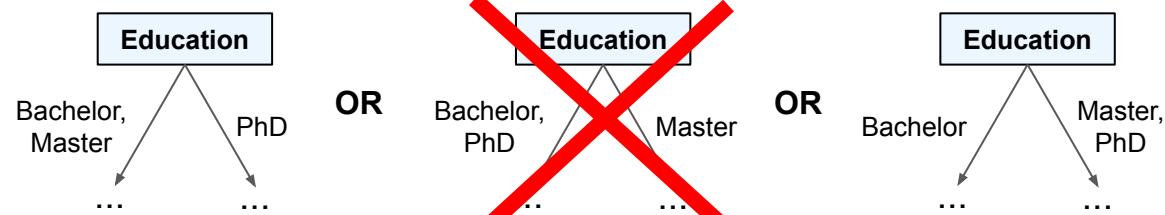
- Each value yields one subtree
- In principle, arbitrary splits into  $2 \leq s \leq d$  subtrees possible, but number of possible splits explodes



# How to Split? — Ordinal Attributes

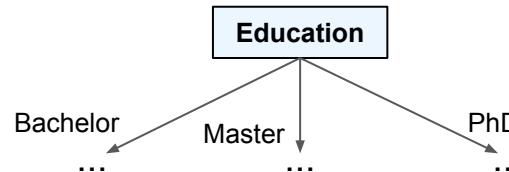
- **Binary split**

- Partition all  $d$  values into two subsets
- Partitions must preserve natural order of values
- $d - 1$  possible splits



- **Multiway split**

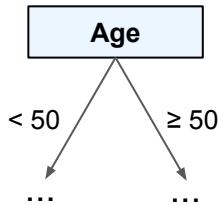
- Each value yields one subtree



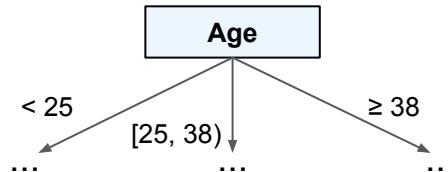
**Note:** Whether "Education" is treated as nominal or ordinal feature is up to interpretation and a design choice of the user

# How to Split? — Numerical Values

- Binary split

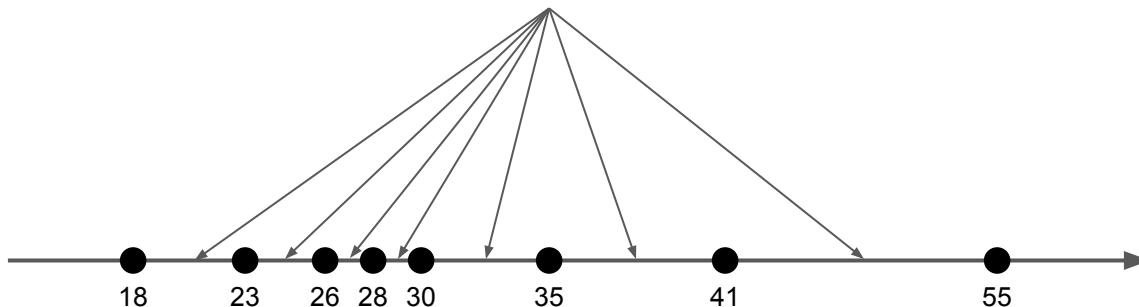


- Multiway split



Age
23
35
26
41
18
55
30
35
28

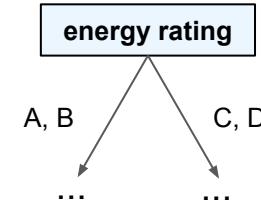
Possible thresholds



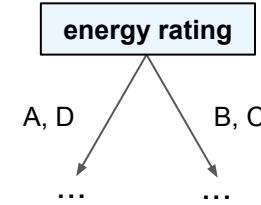
# Quick Quiz

For the **energy rating** of a building,  
what is generally **not** a suitable split?

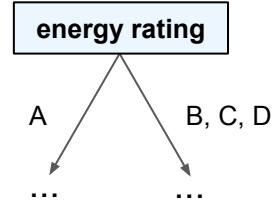
A



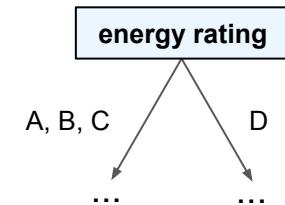
B



C

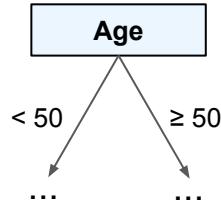


D

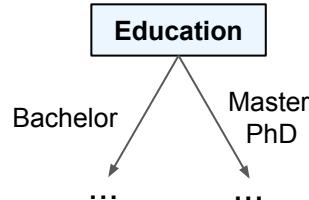


# Finding the Best Splits?

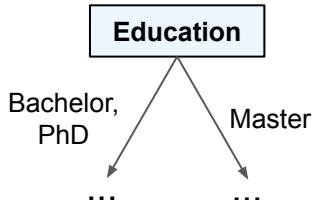
- Which feature to use for splitting the training records



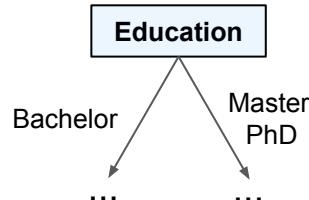
vs.



- How to split the w.r.t. a selected feature?



vs.

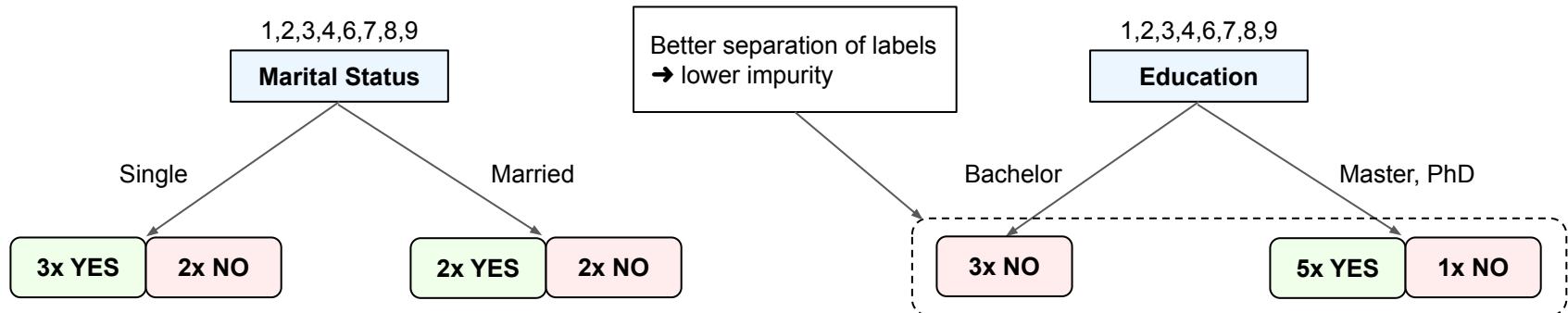


Why choose **Age** over  
**Education**, or vice versa?

What makes **{B,M}/P}** a  
better split than **{B}/{MP}**,  
or vice versa — or any  
other alternative way?

# Finding the Best Splits

- Global optimum
  - Best splits = splits that result in a Decision Tree with the highest accuracy
  - Problem: Finding the optimal tree is NP-complete → not practical for large datasets
- Greedy approach
  - Fast(er) heuristics but no guarantees for optimal results
  - Basic approach: Pick the split that minimizes the **impurity** of subtrees (w.r.t. class labels)



# Finding the Best Splits

- General procedure
  - Calculate impurity  $I(t)$  of node  $t$  before splitting
  - For each possible / considered split, calculate impurity of split  $I_{split}$   
(weighted average of impurities of resulting child nodes)
  - Select split with lowest impurity  $I_{split}$
  - Perform split if  $I_{split} < I(t)$  — (not necessarily always the case)

# Impurity of a Node (Classification)

## Gini Index

$$Gini(t) = 1 - \sum_{c \in C} P(c|t)^2$$

0x YES

6x NO

$$1 - (1.0^2 + 0^2) = 0$$

## Entropy

$$Entropy(t) = - \sum_{c \in C} P(c|t) \log P(c|t)$$

4x YES

2x NO

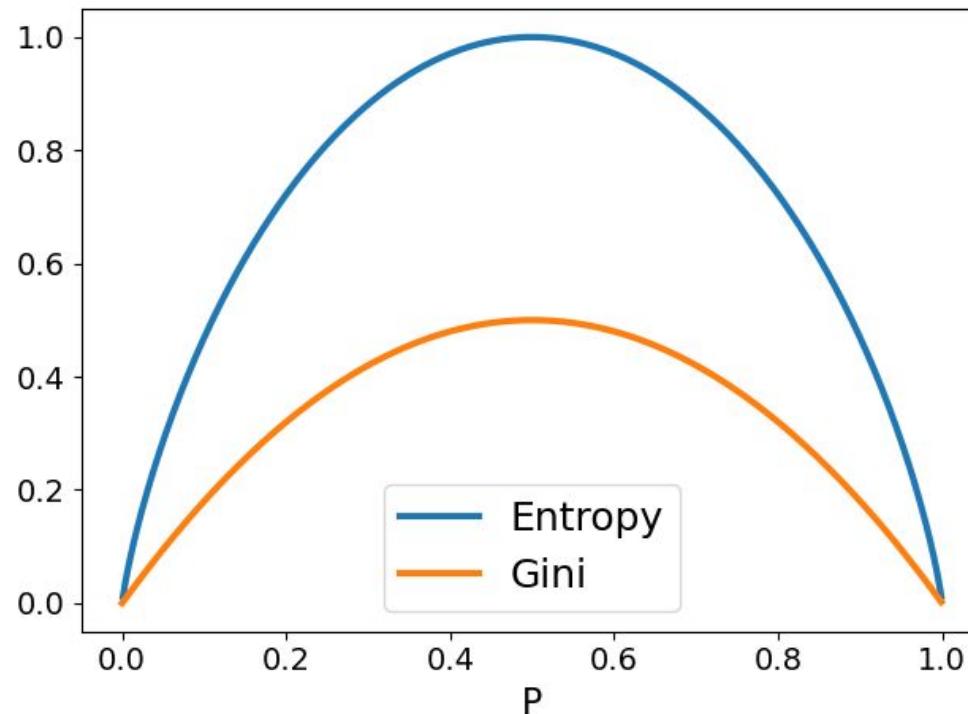
$$1 - ((4/6)^2 + (2/6)^2) = 0.44$$

$$-(4/6 \log_2 4/6 + 2/6 \log_2 2/6) = 0.92$$

$P(c|t)$  = relative frequency of class  $c$  in node  $t$

# Impurity of a Node (Classification)

- Gini Index vs. Entropy for 2-class problem



# Impurity of a Split (Classification)

- Assume node  $t$  is split into  $k$  children

- $n_i$  — number of records at  $i$ -th child
- $n$  — number of records at node  $t$
- $I(i)$  — impurity of node (e.g., Gini, Entropy)

Sum of impurity values of all children, weighted by the number of records at each child.

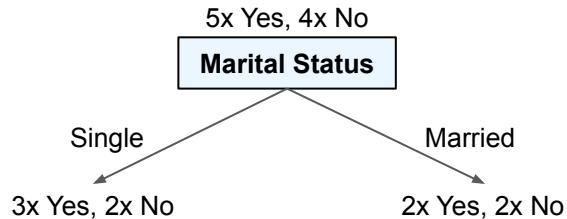
$$I_{split} = \sum_i^k \frac{n_i}{n} I(i)$$

- Information Gain IG

- Difference between  $I(t)$  and  $I_{split}$
- Choose split that minimizes  $I_{split}$  = split that maximizes  $IG$
- Required condition:  $IG > 0$

$$IG = I(t) - I_{split}$$

# Impurity of Split (Classification) — Example

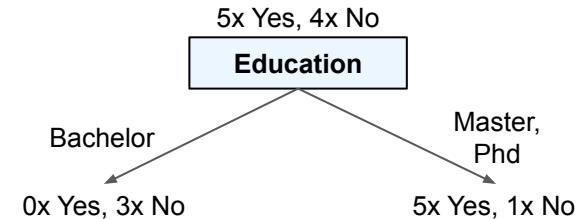


$$Gini(t_{parent}) = 1 - \left( \left(\frac{5}{9}\right)^2 + \left(\frac{4}{9}\right)^2 \right) = 0.49$$

$$Gini(t_{single}) = 1 - \left( \left(\frac{3}{5}\right)^2 + \left(\frac{2}{5}\right)^2 \right) = 0.48$$

$$Gini(t_{married}) = 1 - \left( \left(\frac{2}{4}\right)^2 + \left(\frac{2}{4}\right)^2 \right) = 0.5$$

$$Gini_{split} = \frac{5}{9} \cdot Gini(t_{single}) + \frac{4}{9} \cdot Gini(t_{married}) = 0.49$$



$$Gini(t_{parent}) = 1 - \left( \left(\frac{5}{9}\right)^2 + \left(\frac{4}{9}\right)^2 \right) = 0.49$$

$$Gini(t_B) = 1 - \left( \left(\frac{0}{3}\right)^2 + \left(\frac{3}{3}\right)^2 \right) = 0$$

$$Gini(t_{M/P}) = 1 - \left( \left(\frac{5}{6}\right)^2 + \left(\frac{1}{6}\right)^2 \right) = 0.28$$

$$Gini_{split} = \frac{3}{9} \cdot Gini(t_B) + \frac{6}{9} \cdot Gini(t_{M/P}) = 0.19$$

$$IG = Gini(t_{parent}) - Gini_{split} = 0$$

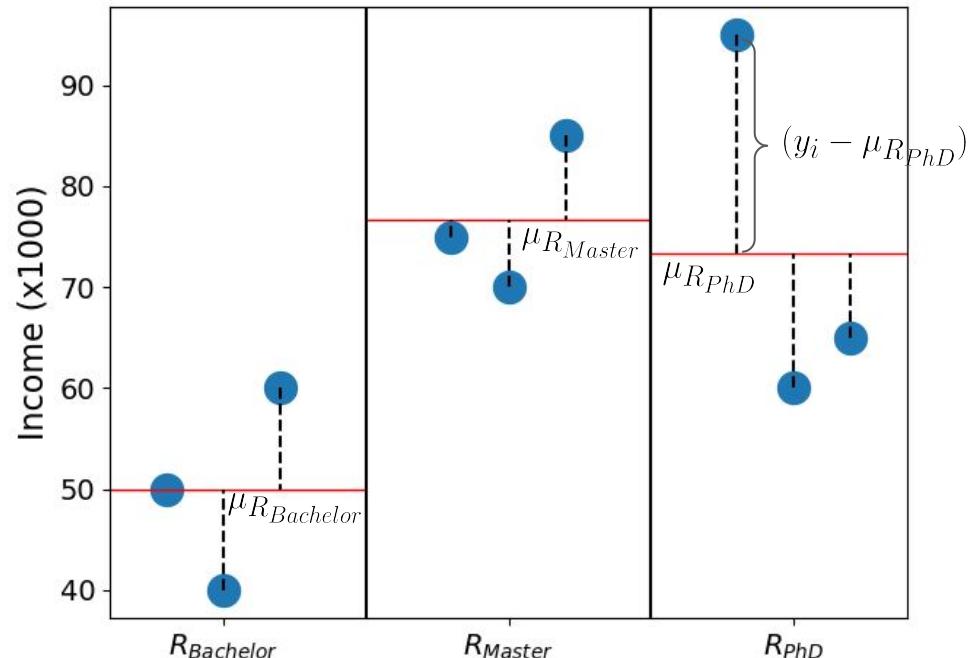
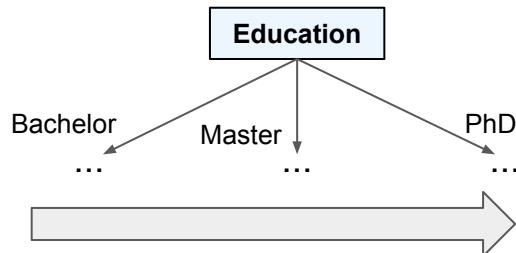
$$IG = Gini(t_{parent}) - Gini_{split} = 0.3$$

# Impurity of a Split (Regression)

## Residual Sum of Squares (RSS)

$$RSS_{split} = \sum_{k=1}^K \sum_{i \in R_k} (y_i - \mu_{R_k})^2$$

Education	Annual Income
Masters	75k
Bachelor	50k
Masters	70k
PhD	95k
Bachelor	40k
Master	85k
Bachelor	60k
PhD	60k
PhD	65k

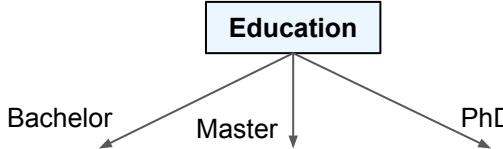


# Impurity of a Split (Regression)

$$RSS_{split} = \sum_{k=1}^K \sum_{i \in R_k} (y_i - \mu_{R_k})^2$$

$$= \sum_{i \in R_{Bachelor}} (y_i - \mu_{R_{Bachelor}})^2 + \sum_{i \in R_{Master}} (y_i - \mu_{R_{Master}})^2 + \sum_{i \in R_{PhD}} (y_i - \mu_{R_{PhD}})^2$$

$$\begin{aligned} &= (50 - 50)^2 + (40 - 50)^2 + (60 - 50)^2 \\ &\quad + (75 - 76.67)^2 + (70 - 76.67)^2 + (85 - 76.67)^2 \\ &\quad + (95 - 73.33)^2 + (60 - 73.33)^2 + (65 - 73.33)^2 \\ &= 1033.34 \end{aligned}$$



$\mu_{R_{Bachelor}} = 50$
$\mu_{R_{Master}} = 76.67$
$\mu_{R_{PhD}} = 73.33$

Edu- cation	Annual Income
Masters	75k
Bachelor	50k
Masters	70k
PhD	95k
Bachelor	40k
Master	85k
Bachelor	60k
PhD	60k
PhD	65k

# Decision Trees — Pros & Cons

- **Pros**

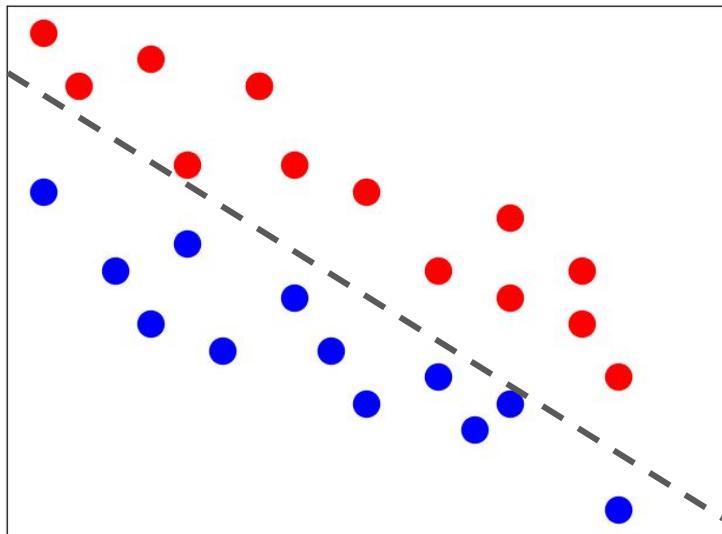
- Inexpensive to train and test
- Easy to interpret (if tree is not too large)
- Can handle categorical and numerical data

- **Cons**

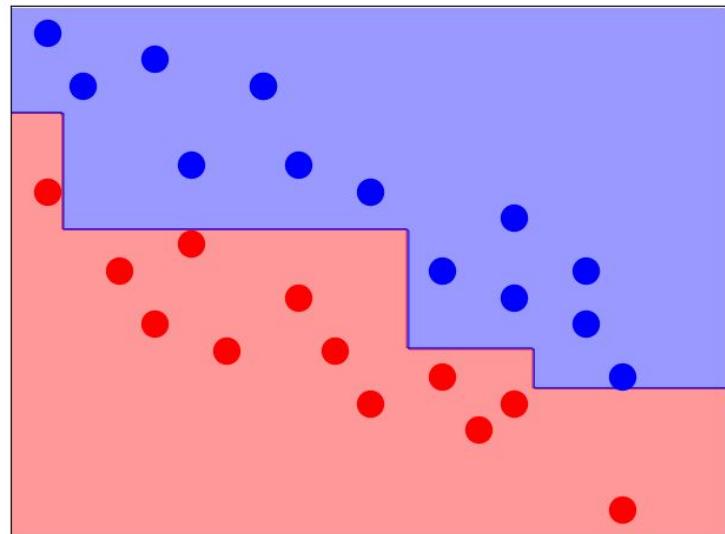
- Sensitive to small changes in the training data  
(Hierarchical structure: errors early on propagate down)
- Greedy approach does not guarantee optimal tree
- Each decision involves only a single feature
- Does not take interactions between features into account

# Decision Trees — Interaction between Features

Optimal decision boundary



Decision Tree boundaries

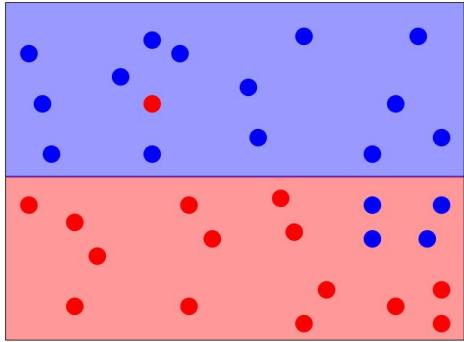


# Outline

- **Decision Trees**
  - Overview
  - Training Decision Trees
  - **Overfitting**
- **Tree Ensembles**
  - Bagging
  - Random Forest
  - Boosting

# Decision Trees — Underfitting & Overfitting

Underfitting



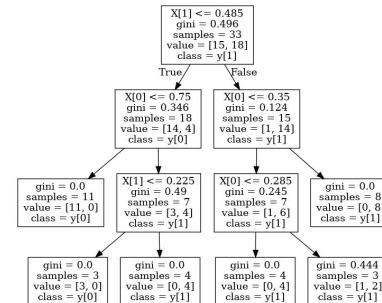
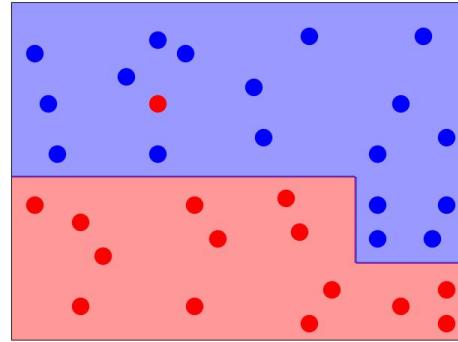
$X[1] \leq 0.485$   
gini = 0.496  
samples = 33  
value = [15, 18]  
class = y[1]

True  
False

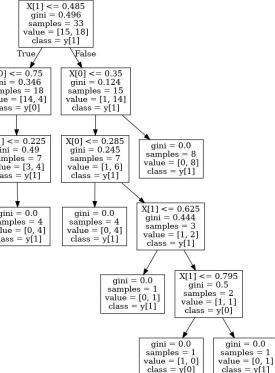
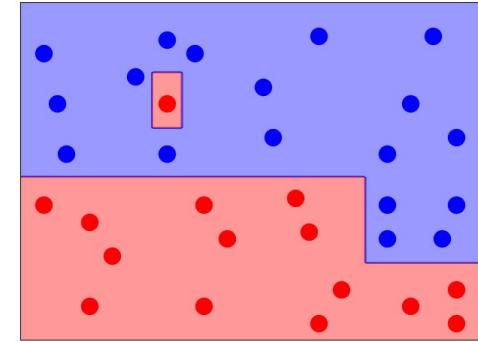
gini = 0.346  
samples = 18  
value = [14, 4]  
class = y[0]

gini = 0.124  
samples = 15  
value = [1, 14]  
class = y[1]

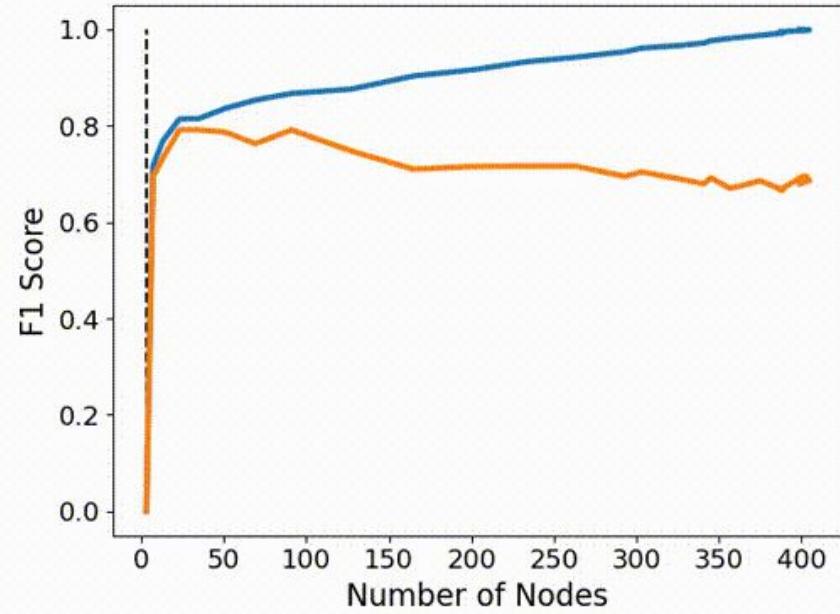
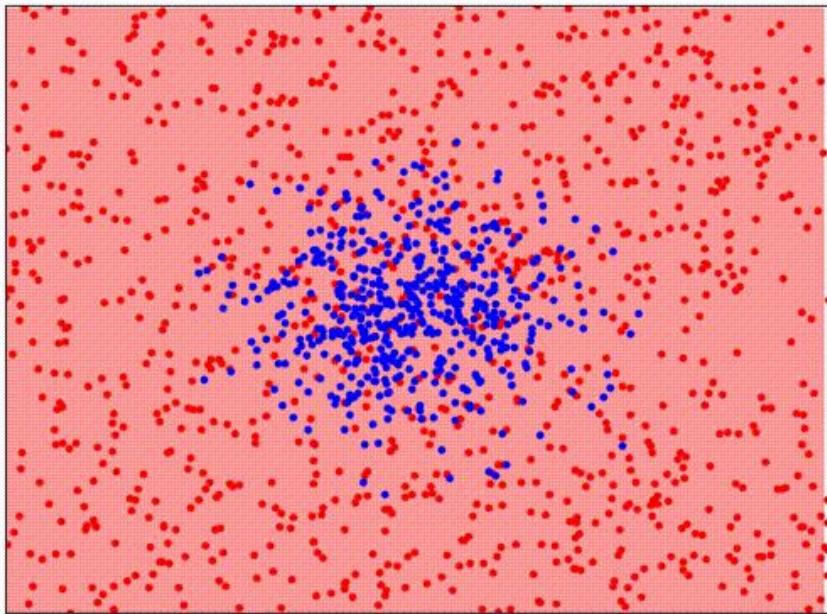
Good fit



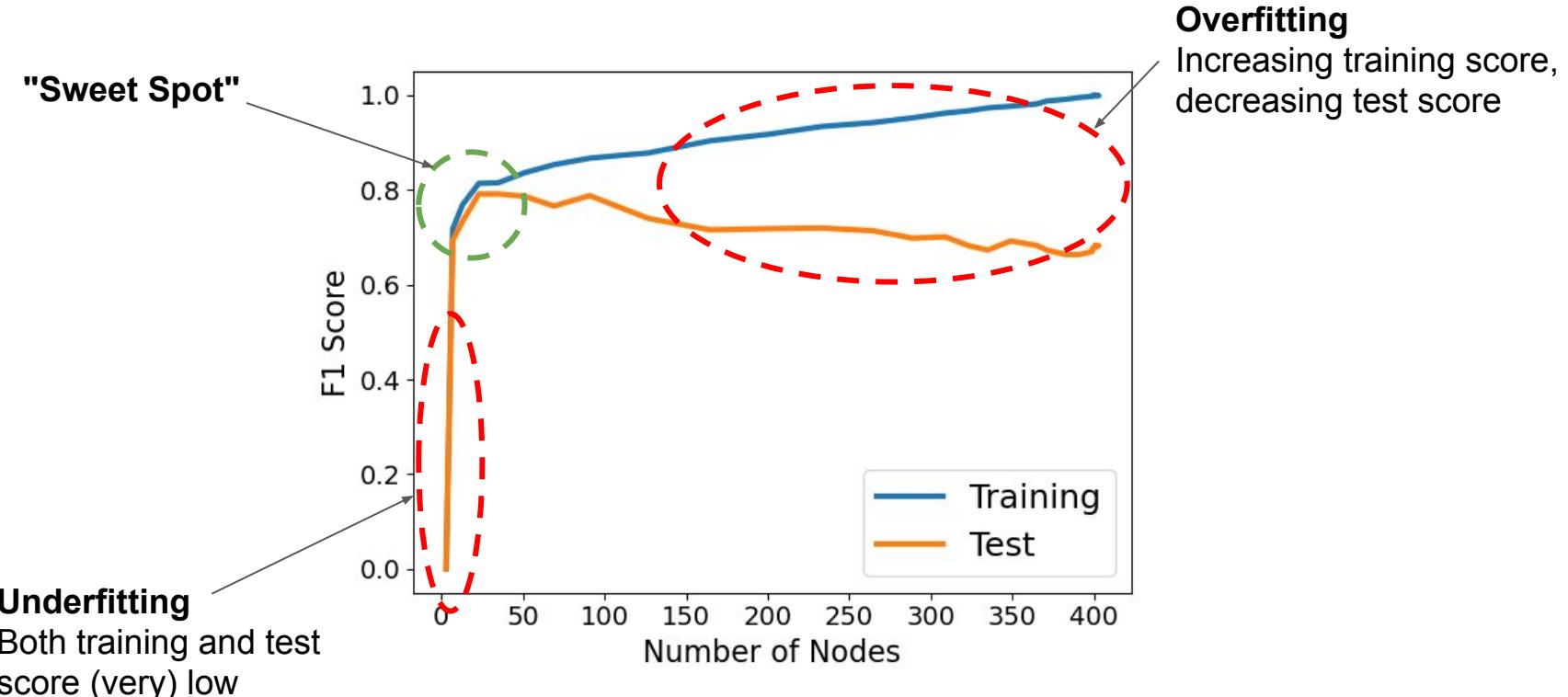
Overfitting



# Decision Trees — Underfitting & Overfitting

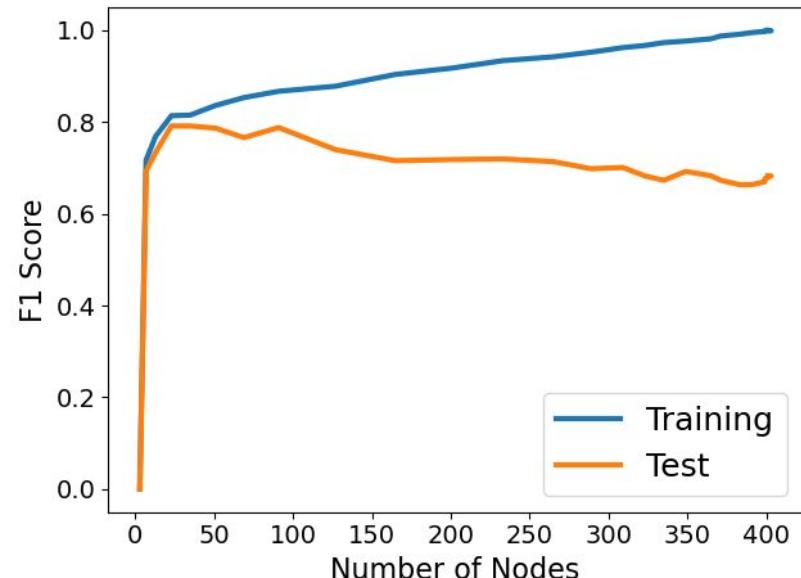


# Decision Trees — Underfitting & Overfitting



# Decision Trees — Overfitting

- Decision Tree algorithm can always split the training data perfectly\*
  - Keep splitting (i.e., increase height of tree) until each leaf contains only one data sample
  - One data sample → 100% pure

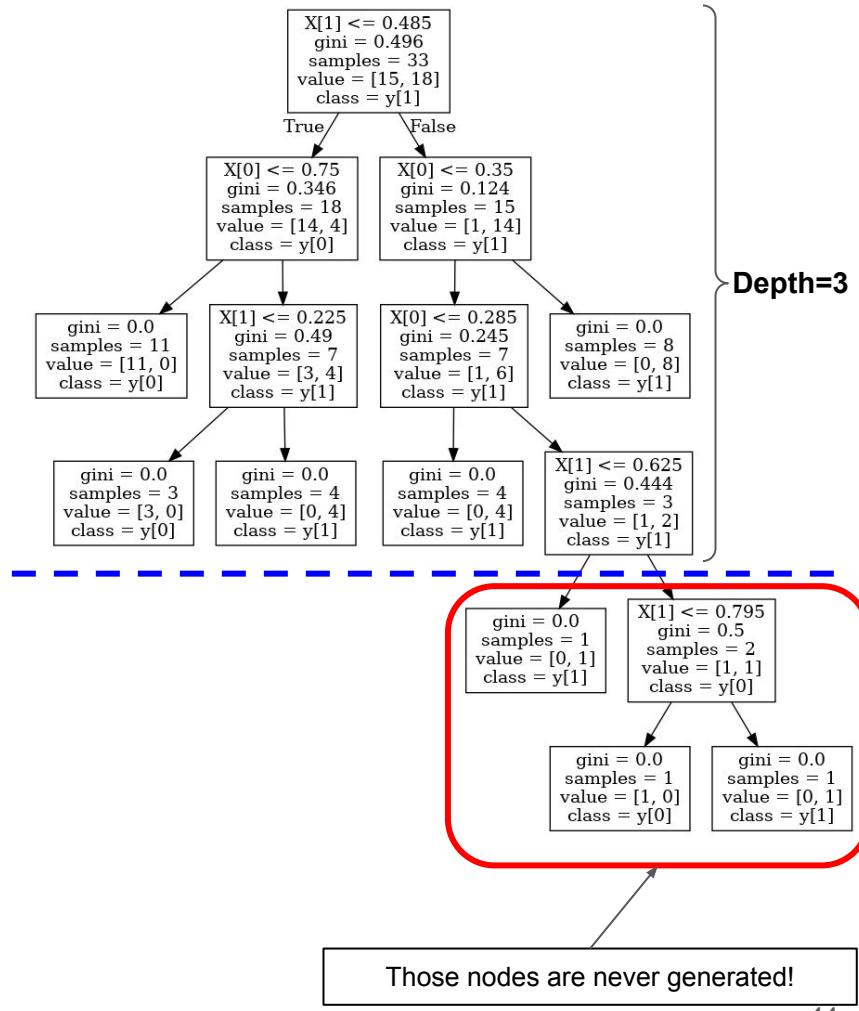
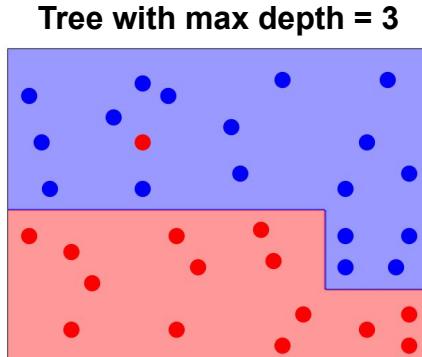
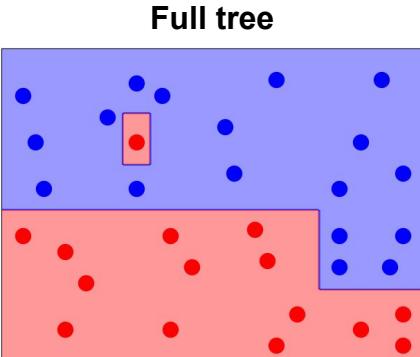


- Solution: Limit size/height of Decision Tree → Pruning
  - Pre-pruning: Stop splitting nodes ahead of time
  - Post-pruning: Build full tree, but then remove leaves/splits if beneficial
  - ... combination of multiple approaches

\*assuming not duplicate data points with different target labels/values

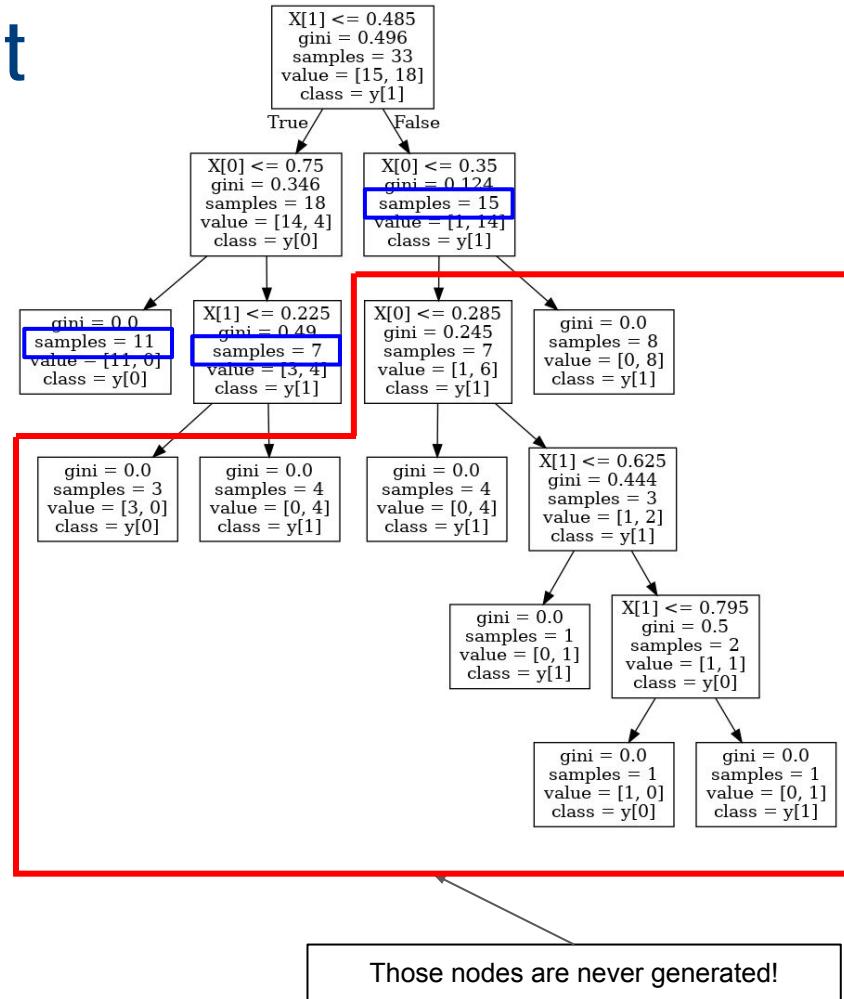
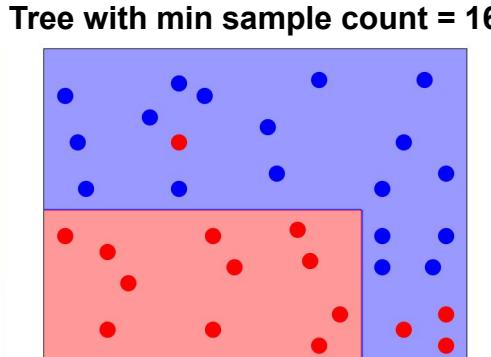
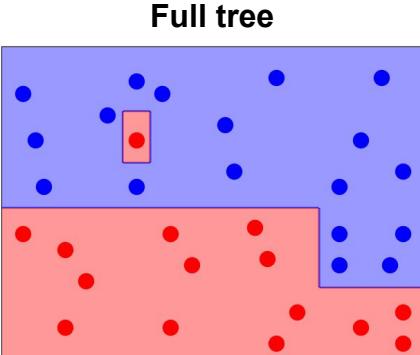
# Pre-Pruning: Maximum Depth

- Define maximum depth/height of tree
  - Stop splitting if maximum depth is reached
- Example: maximum depth = 3

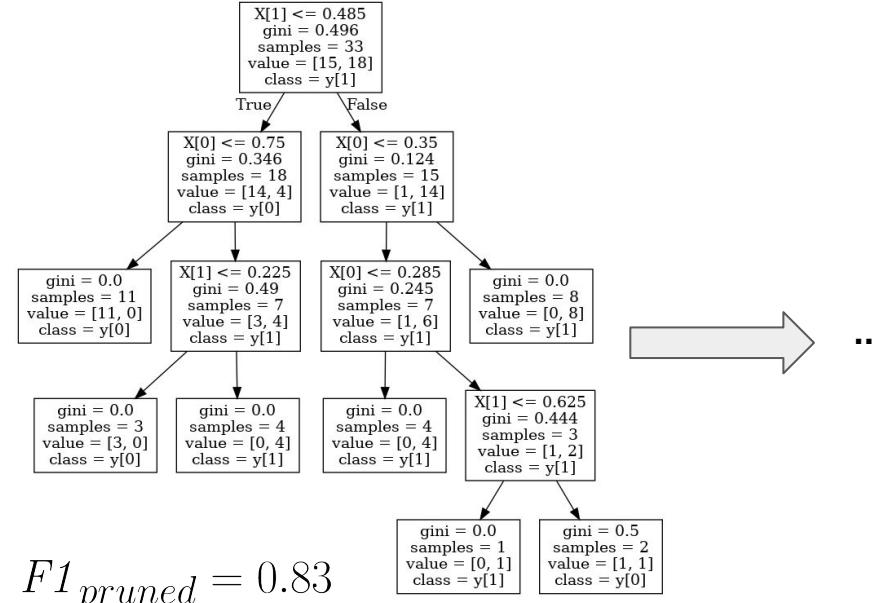
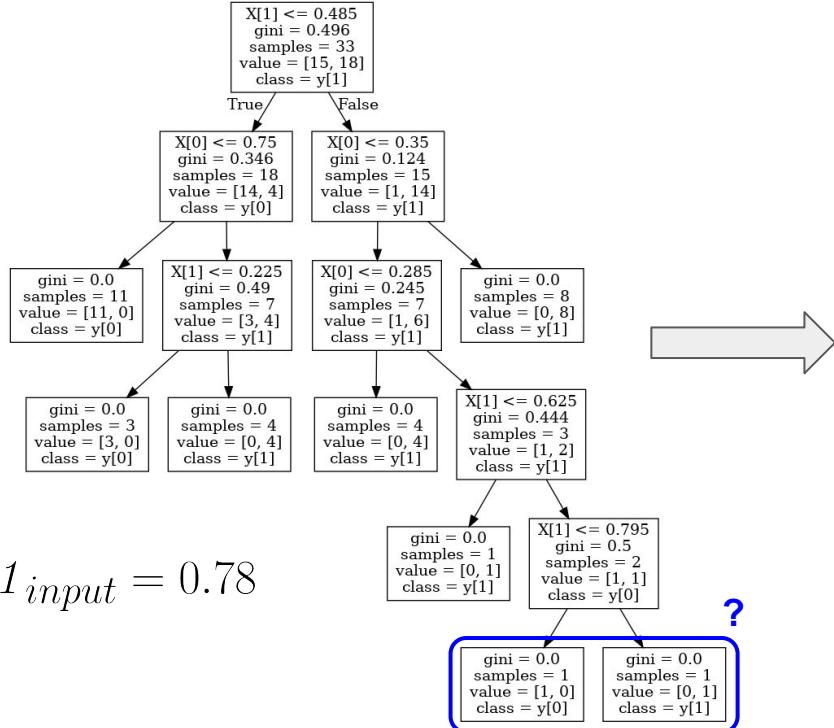


# Pre-Pruning: Minimum Sample Count

- Define minimum number of samples each node must have
  - Stop splitting if node has less than the minimum number of samples
- Example: minimum sample count = 16



# Post-Pruning: Prune Leaves/Splits using Validation



$F1_{pruned} > F1_{input} \rightarrow$  Remove split from Decision Tree (and continue checking next split)

# Quick Quiz

What is the **maximum** possible **depth** of a Decision Tree given a dataset with  $N$  data points?

A

$O(N)$

B

$O(\log N)$

C

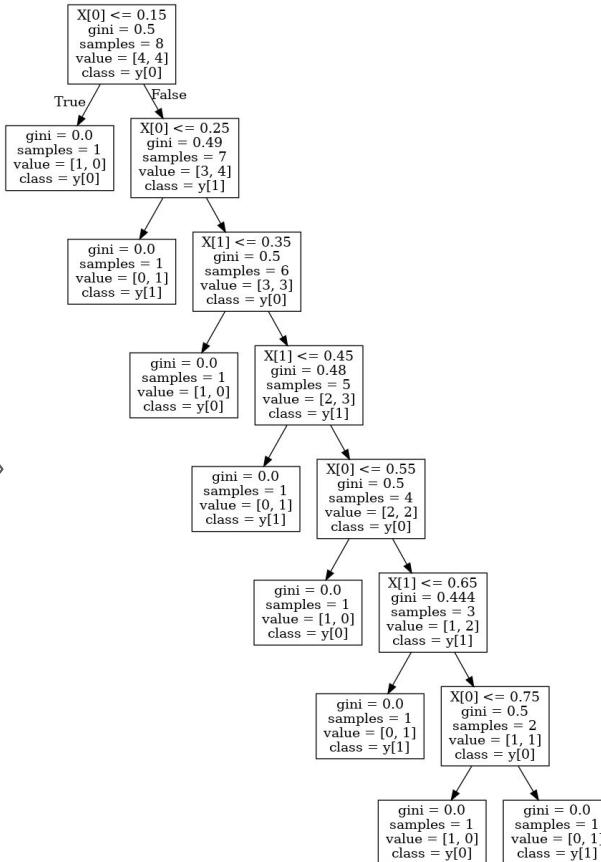
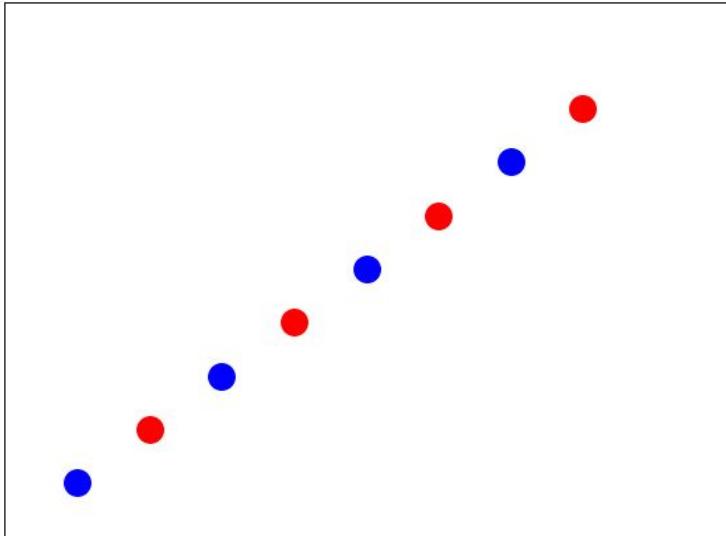
$O(N \log N)$

D

$O(\log \log N)$

# Quick Quiz

$N = 8$



# Outline

- **Decision Trees**
  - Overview
  - Training Decision Trees
  - Overfitting
  
- **Tree Ensembles**
  - Bagging
  - Random Forest
  - Boosting

# Tree Ensembles

- Aim to address limitations of (single) Decision Trees
  - High variance — i.e., sensitivity to small changes in training data
  - Typically not the same accuracy as other approaches

- Countermeasure: Tree Ensembles

Construct many decision trees and combine their predictions

- Bagging
- Random Forests
- Boosting

**Basic trade-off of ensemble methods:**

Higher accuracy, lower variance  
vs.  
Lower interpretability, longer training time

# Outline

- **Decision Trees**
  - Overview
  - Training Decision Trees
  - Overfitting
  
- **Tree Ensembles**
  - Bagging
  - Random Forest
  - Boosting

# Bagging — Bootstrap Aggregation

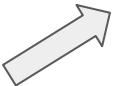
- Bagging — basic idea (not limited to Decision Trees)
  - Train many models (classifiers/regressors) of different training data
  - Combine predictions of each models for final prediction
  - Increases accuracy and lowers variance
- Where to get more training data from? → Bootstrap Sampling
  - Take repeated samples from a single training dataset  $D$
  - Bootstrap sample  $D_i$  sampled from  $D$ , **uniformly** and **with replacement** ( $|D_i| = |D|$ )
  - Train a model over each bootstrap dataset  $D_i$

# Bagging — Bootstrap Aggregation

ID	Age	Education	Credit Approval
1	23	Masters	Yes
2	35	Bachelor	No
3	26	Masters	Yes
4	41	PhD	Yes

Bootstrap Sample 1

ID	Age	Education	Credit Approval
1	23	Masters	Yes
2	35	Bachelor	No
2	35	Bachelor	No
4	41	PhD	Yes



Bootstrap Sample 2

ID	Age	Education	Credit Approval
2	35	Bachelor	No
2	35	Bachelor	No
4	41	PhD	Yes
4	41	PhD	Yes

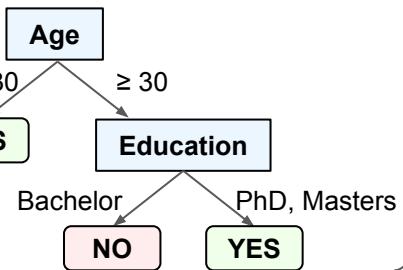
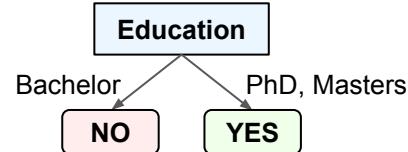
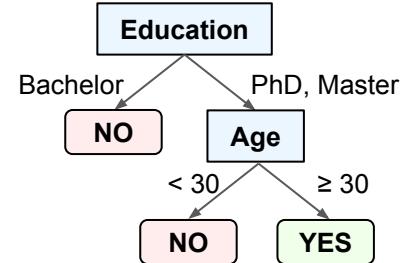


Bootstrap Sample N

ID	Age	Education	Credit Approval
1	23	Masters	Yes
2	35	Bachelor	No
3	26	Masters	Yes
4	41	PhD	Yes

...

ID	Age	Education	Credit Approval
1	23	Masters	Yes
2	35	Bachelor	No
2	35	Bachelor	No
4	41	PhD	Yes



Majority Vote

# Bagging — Bootstrap Aggregation

- Limitations

- Assume original dataset  $D$  has one or more strong predictors — features that yield splits with a (very) high information gain
- Bootstrap samples  $D_i$  are also likely to have those strong predictors

## → Consequences

- Most bagged trees will use strong predictors on top
- Most bagged trees will look very similar
- Predictions of bagged trees will be highly correlated



→ Only limited reduction in variance!

# Outline

- **Decision Trees**
  - Overview
  - Training Decision Trees
  - Overfitting
  
- **Tree Ensembles**
  - Bagging
  - **Random Forest**
  - Boosting

# Random Forests

- Random Forest = bootstrap sampling (bagging) + feature sampling
  - Create bootstrap samples  $D_i$  like for bagging
  - Feature sampling: For each  $D_i$ , consider only a random subset of features of size  $m$

$d$  features

Age	Edu- cation	Marital Status	Annual Income	Credit Approval
23	Masters	Single	75k	Yes
35	Bachelor	Married	50k	No
26	Masters	Single	70k	Yes
41	PhD	Single	95k	Yes
18	Bachelor	Single	40k	No
55	Master	Married	85k	No
30	Bachelor	Single	60k	No
35	PhD	Married	60k	Yes
28	PhD	Married	65k	Yes

bootstrap sampling  
+ feature sampling



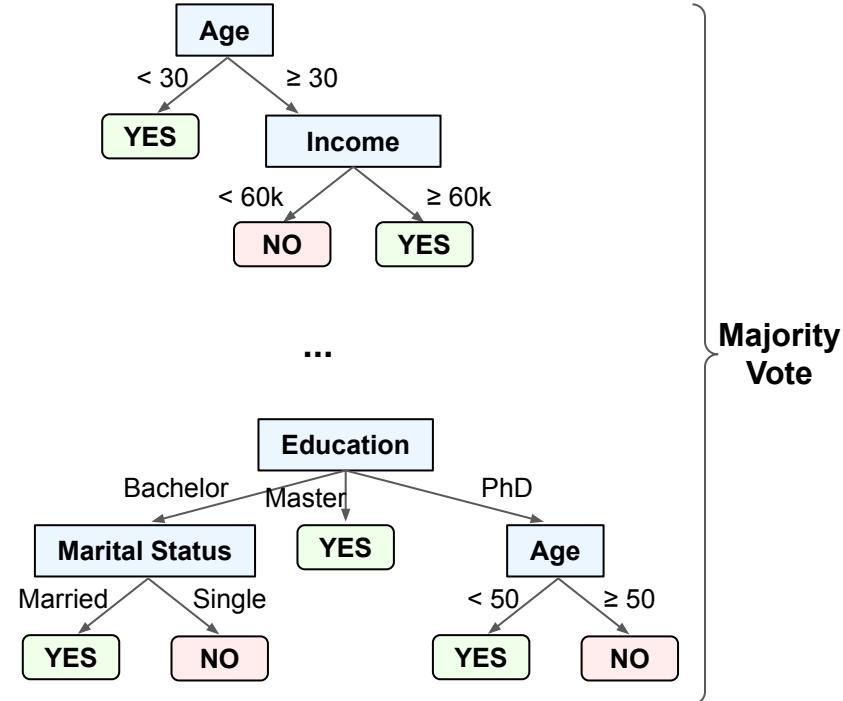
typically  
 $m \approx \sqrt{d}$

$m$  features

Edu- cation	Annual Income	Credit Approval
Masters	75k	Yes
Bachelor	50k	No
Masters	70k	Yes
PhD	95k	Yes
Bachelor	40k	No
Masters	70k	Yes
Masters	75k	Yes
Bachelor	40k	No
Bachelor	40k	No

# Random Forests

- Effects of feature sampling
    - Strong predictors in  $D$  are often absent in  $D_i$
    - Resulting trees often look very different
    - Predictions of trees much less correlated
  - Higher reduction in variance + typically higher accuracy



# Random Forests — Pros & Cons (Compared to Decision Trees)

- Pros

- High accuracy — fairly close to state of the art
- Sampling and training independent across  $D_i \rightarrow$  parallelizable!
- Not much tuning required

- Cons

- Less Interpretable
- Slower training and prediction

# Outline

- **Decision Trees**
  - Overview
  - Training Decision Trees
  - Overfitting
- **Tree Ensembles**
  - Bagging
  - Random Forest
  - Boosting

# Boosting

- Like bagging, boosting combines multiple trees (in general, multiple models)
- So what are the key differences?

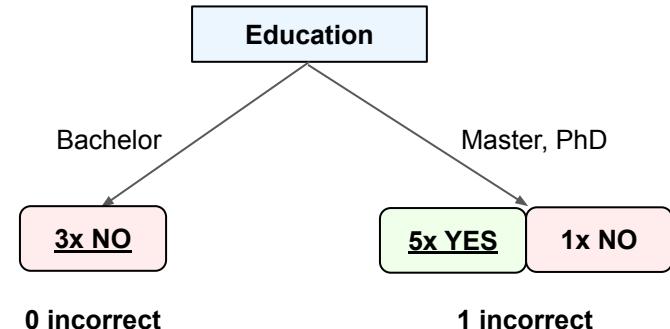
	Bagging	Boosting
Training	Trees are trained independently (and can be done in parallel)	Trees are trained in sequence; (the accuracy of the last tree affects the training of the next tree)
Prediction	All trees have the same amount of say in the final prediction	Trees have different amount of say in the final prediction (depending on their individual accuracy)

# Boosting & Weak Learners

- So far, all models discussed are **Strong Learners**
  - Goal: perform as best as possible on a given classification or regression task

- **Weak Learner**

- Goal: perform (slightly) better than guessing
- Very common weak learner: **Decision Stump**  
(e.g., decision tree of height 1, i.e., only one split)
- Very simple model → very fast training



- **Boosting:** Combine many weak classifiers into a single strong learner
  - Basic idea: subsequent models try to improve the errors of previous models

# AdaBoost — Adaptive Boosting (for Decision Trees)

- AdaBoost
  - Applicable to many classification/regression algorithms to improve performance
  - Very commonly combined with Decision Trees
- Basic training algorithm
  - Train a Weak Learner over  $D_i$  (e.g., Decision Stump)
  - Identify all misclassified samples
  - Calculate error rate of learner to quantify its amount of say
  - Sample  $D_{i+1}$  such that misclassified samples are more likely to be picked than correctly classified samples
  - Repeat...

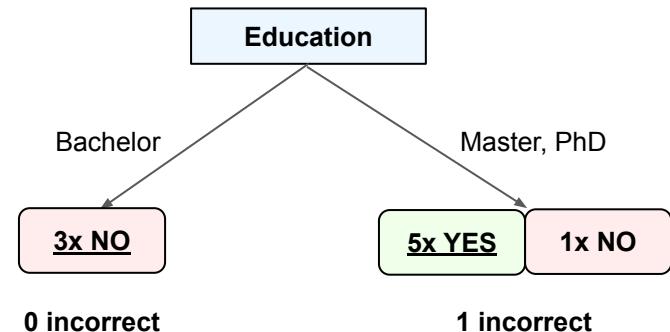
# AdaBoost Training — Step-by-Step Example (in the $m$ -th iteration)

- Step 1:

- Train Decision Stump  $h_m$  over sampled dataset  $D_m$   
(original dataset  $D$  in the beginning)
- Identify all misclassified training samples in  $D$

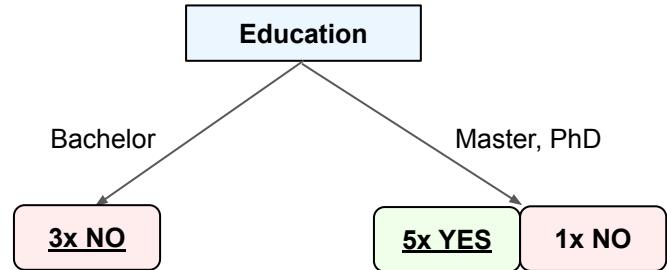
Sample Weight $w$
1/9
1/9
1/9
1/9
1/9
1/9
1/9
1/9
1/9

Age	Education	Marital Status	Annual Income	Credit Approval
23	Masters	Single	75k	Yes
35	Bachelor	Married	50k	No
26	Masters	Single	70k	Yes
41	PhD	Single	95k	Yes
18	Bachelor	Single	40k	No
55	Master	Married	85k	No
30	Bachelor	Single	60k	No
35	PhD	Married	60k	Yes
28	PhD	Married	65k	Yes



**Initial step:** assign each data sample in  $D$  with a weight  $w_i=1/|D|$

# AdaBoost Training — Step-by-Step Example (in the $m$ -th iteration)



- **Step 2**

2a) Calculate total error  $\epsilon_m$

$$\epsilon_m = \sum_i^N w_i \cdot \underbrace{\delta(h_m(x_i) \neq y_i)}_{\begin{array}{l} 0, \text{ if } x_i \text{ is correctly classified} \\ 1, \text{ if } x_i \text{ is misclassified} \end{array}}$$

$$\epsilon_m = 1/9$$

2b) Calculate "amount of say"  $\alpha_m$  of  $h_m$

$$\alpha_m = \frac{1}{2} \ln \frac{1 - \epsilon_m}{\epsilon_m}$$

$$\alpha_m = 1.04$$

# AdaBoost Training — Step-by-Step Example (in the $m$ -th iteration)

- Step 3

- 3a) Update sample weights

$$w_i = w_i \cdot \begin{cases} e^{\alpha_m}, & \text{if } x_i \text{ was misclassified} \\ e^{-\alpha_m}, & \text{if } x_i \text{ was correctly classified} \end{cases}$$

- 3b) Normalize sample weights

$$w_i = \frac{w_i}{\sum_i^N w_i}$$

Sum up to 1

Age	Education	Marital Status	Annual Income	Credit Approval
23	Masters	Single	75k	Yes
35	Bachelor	Married	50k	No
26	Masters	Single	70k	Yes
41	PhD	Single	95k	Yes
18	Bachelor	Single	40k	No
55	Master	Married	85k	No
30	Bachelor	Single	60k	No
35	PhD	Married	60k	Yes
28	PhD	Married	65k	Yes



Sample Weight w	3a)	3b)	Sample Weight w
1/9	0.04	0.0635	0.0635
1/9	0.04	0.0635	0.0635
1/9	0.04	0.0635	0.0635
1/9	0.04	0.0635	0.0635
1/9	0.04	0.0635	0.0635
1/9	0.04	0.0635	0.0635
1/9	<b>0.31</b>	<b>0.492</b>	0.492
1/9	0.04	0.0635	0.0635
1/9	0.04	0.0635	0.0635
1/9	0.04	0.0635	0.0635



# AdaBoost Training — Step-by-Step Example (in the $m$ -th iteration)

- Step 4

4a) Generate new  $D_i$  based on sample weights

(misclassified samples are much more likely to be picked)

4b) With new  $D_i$ , go to Step 1 and continue

Sample Weight w	Age	Edu- cation	Marital Status	Annual Income	Credit Approval
0.0635	23	Masters	Single	75k	Yes
0.0635	35	Bachelor	Married	50k	No
0.0635	26	Masters	Single	70k	Yes
0.0635	41	PhD	Single	95k	Yes
0.0635	18	Bachelor	Single	40k	No
0.492	55	Master	Married	85k	No
0.0635	30	Bachelor	Single	60k	No
0.0635	35	PhD	Married	60k	Yes
0.0635	28	PhD	Married	65k	Yes



New input for Step 1

Age	Edu- cation	Marital Status	Annual Income	Credit Approval
23	Masters	Single	75k	Yes
55	Master	Married	85k	No
26	Masters	Single	70k	Yes
41	PhD	Single	95k	Yes
55	Master	Married	85k	No
55	Master	Married	85k	No
26	Masters	Single	70k	Yes
35	PhD	Married	60k	Yes
55	Master	Married	85k	No

# AdaBoost Training — Basic Algorithm

**Initialization:** Dataset  $D$ ,  $|D|=N$ , with initial sample weights  $w_i = \frac{1}{N}$

**for**  $m = 1$  to  $M$  **do:**

    Generate  $D_m$  by sampling from  $D$  w.r.t. sampling weights  $w$

    Train Decision Stump  $h_m$  over  $D_m$

    Apply  $h_m$  to all samples in  $D$  and identify misclassified samples

    Calculate total error

$$\epsilon_m = \sum_i^N w_i \cdot \delta(h_m(x_i) \neq y_i)$$

    Calculate amount of say

$$\alpha_m = \frac{1}{2} \ln \frac{1 - \epsilon_m}{\epsilon_m}$$

    Update sample weights

$$w_i = w_i \cdot \begin{cases} e^{\alpha_m}, & \text{if } x_i \text{ was misclassified} \\ e^{-\alpha_m}, & \text{if } x_i \text{ was correctly classified} \end{cases} \quad \& \quad w_i = \frac{w_i}{\sum_i^N w_i}$$

**end for**

# AdaBoost Prediction

- Assume 8 boosted Decision Stumps  $h_1, \dots, h_8$ 
  - Each tree has an "amount of say"  $\alpha_m$
  - Let  $h_1, h_3, h_8$  say "Yes"; all other trees say "No"

$$\alpha_m = \frac{1}{2} \ln \frac{1 - \epsilon_m}{\epsilon_m}$$

$h_1$	$\alpha_1 = 0.34$
$h_3$	$\alpha_3 = 1.20$
$h_8$	$\alpha_8 = 0.97$
$h_2$	$\alpha_2 = 0.14$
$h_4$	$\alpha_4 = 0.58$
$h_5$	$\alpha_5 = 0.09$
$h_6$	$\alpha_6 = 0.62$
$h_7$	$\alpha_7 = 0.45$

}

$$0.34 + 1.20 + 0.97 = 2.51$$



Final prediction: "Yes"

$$0.14 + 0.58 + 0.09 + 0.62 + 0.45 = 1.88$$

# Gradient Boosted Trees

- Gradient Boosting
  - Mainly applied to regression algorithms to improve performance
  - Very commonly combined with Decision Trees (for regression)
- Basic training algorithm
  - Start with a initial prediction (e.g., mean over all values)
  - Calculate residuals = error between true value and current prediction
  - Train Decision Stump to predict residuals
  - Update predictions based on predicted residuals
  - Repeat...

# GB Training — Step-by-Step Example (in the $m$ -th iteration)

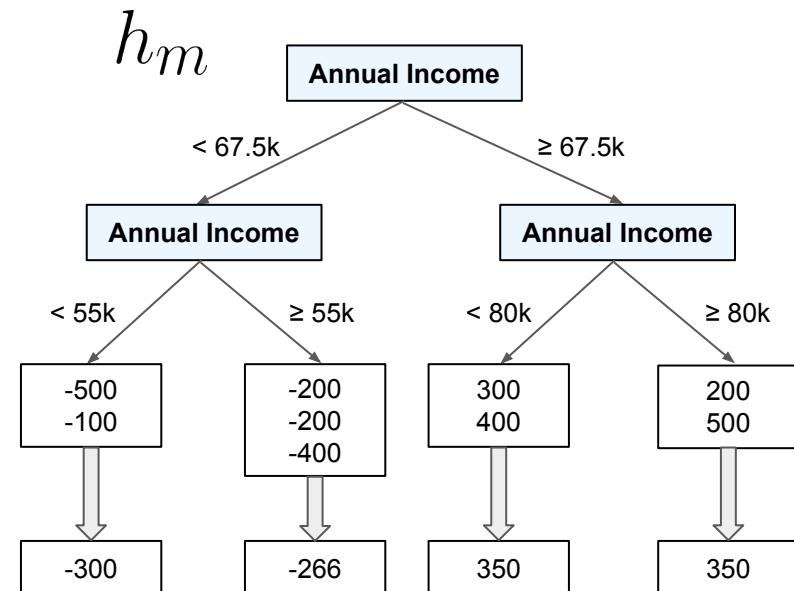
- Step 1:

1a) Calculate residuals  $r_{i,m} = y_i - f_{m-1}(x_i)$

1b) Fit Decision Stump  $h_m$  to residuals  $r_{i,m}$

Age	Education	Marital Status	Annual Income	Credit Limit y	$f_{m-1}(x)$	$r_m(x)$
23	Master	Single	75k	1,400	1,000	400
35	Bachelor	Married	50k	900	1,000	-100
26	Master	Single	70k	1,300	1,000	300
41	PhD	Single	95k	1,500	1,000	500
18	Bachelor	Single	40k	500	1,000	-500
55	Master	Married	85k	1,200	1,000	200
30	Bachelor	Single	60k	800	1,000	-200
35	PhD	Married	60k	800	1,000	-200
28	PhD	Married	65k	600	1,000	-400

Assume  $m = 1 \quad f_0(x_i) = 1000$



# GB Training — Step-by-Step Example (in the $m$ -th iteration)

- Step 2:

- a) Calculate predicted residuals  $h_m(x_i)$  for all training samples
- b) Calculate new predictions  $f_m(x_i) = f_{m-1} + \eta \cdot h_m(x_i)$  (here:  $\eta = 0.1$ )
- c) Set  $m = m+1$ , go to Step 1

Age	Edu- cation	Marital Status	Annual Income	Credit Limit $y$	$f_{m-1}(x)$	$r_m(x)$	$h_m(x)$	$f_m(x)$
23	Master	Single	75k	1,400	1,000	400	350	1,035
35	Bachelor	Married	50k	900	1,000	-100	-300	970
26	Master	Single	70k	1,300	1,000	300	350	1,035
41	PhD	Single	95k	1,500	1,000	500	350	1,035
18	Bachelor	Single	40k	500	1,000	-500	-300	970
55	Master	Married	85k	1,200	1,000	200	350	1,035
30	Bachelor	Single	60k	800	1,000	-200	-266	973
35	PhD	Married	60k	800	1,000	-200	-266	973
28	PhD	Married	65k	600	1,000	-400	-266	973

**Note:** long-term trend

- The residuals  $r_m$  go towards 0
- The predicted values  $f_m$  are closer to the true values  $y$

# GB Training — Step-by-Step Example (in the $m$ -th iteration)

- Output for after Step 1 & 2 for  $m+1$

Includes building new Decision Stump  $h_{m+1}$

**Step 1**      **Step 2**

The table below shows the data for 10 individuals across various features and the resulting values for each step.

Age	Education	Marital Status	Annual Income	Credit Limit y	$f_{m-1}(x)$	$r_m(x)$	$h_m(x)$	$f_m(x)$	$r_{m+1}(x)$	$h_{m+1}(x)$	$f_{m+1}(x)$
23	Master	Single	75k	1,400	1,000	400	350	1,035	365	315	1,067
35	Bachelor	Married	50k	900	1,000	-100	-300	970	-70	-270	943
26	Master	Single	70k	1,300	1,000	300	350	1,035	265	315	1,067
41	PhD	Single	95k	1,500	1,000	500	350	1,035	465	315	1,067
18	Bachelor	Single	40k	500	1,000	-500	-300	970	-470	-270	943
55	Master	Married	85k	1,200	1,000	200	350	1,035	165	315	1,067
30	Bachelor	Single	60k	800	1,000	-200	-266	973	-173	-240	949
35	PhD	Married	60k	800	1,000	-200	-266	973	-173	-240	949
28	PhD	Married	65k	600	1,000	-400	-266	973	-373	-240	949

# Gradient Boosting Training — Basic Algorithm

**Initialization:** Dataset  $D$ ,  $f_0(x_i) = \text{mean}(y)$ ,  $\eta = 0.1$

**for**  $m = 1$  to  $M$  **do:**

    Calculate residuals  $r_{i,m} = y_i - f_{m-1}(x_i)$

    Train Decision Stump  $h_m$  over  $D$  with  $r_{i,m}$  as targets

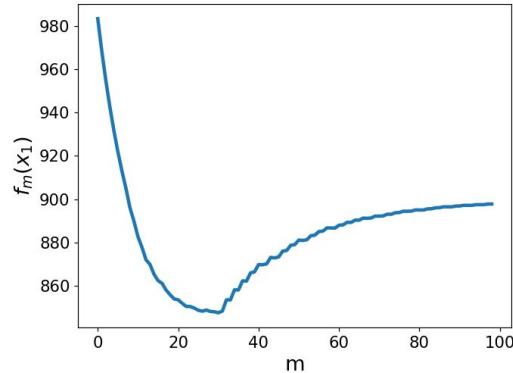
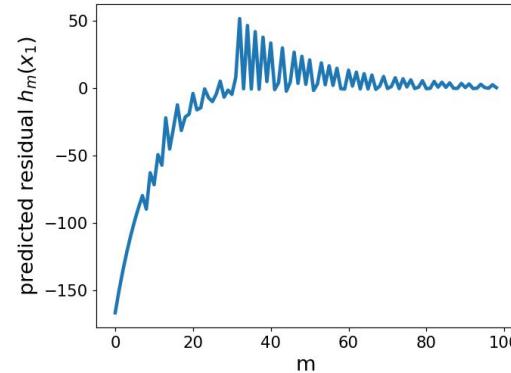
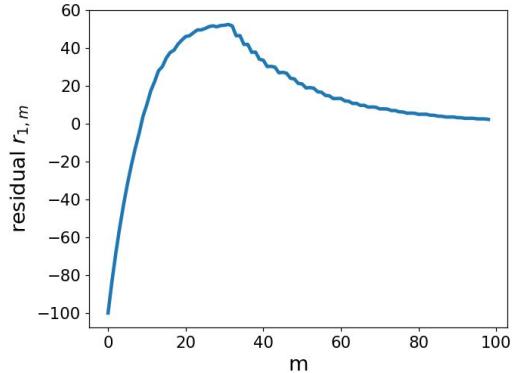
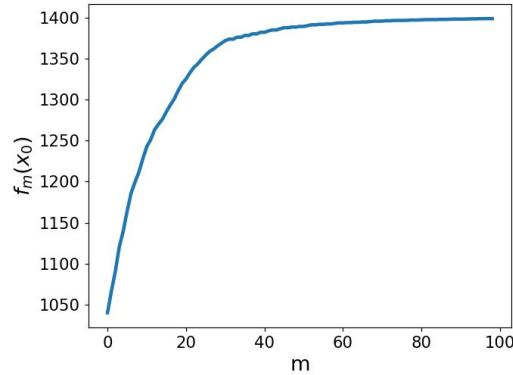
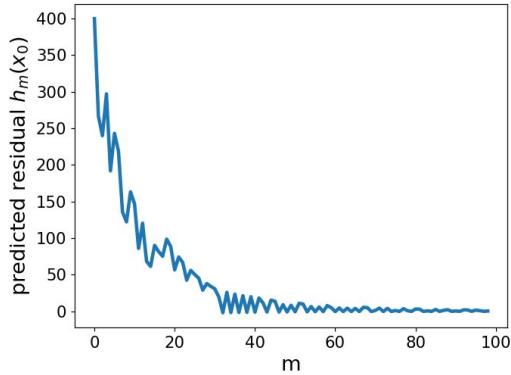
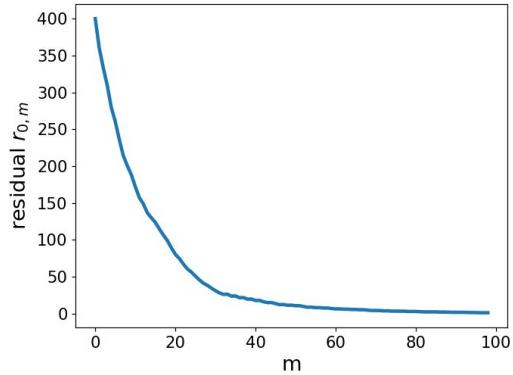
    Predicted residuals  $h_m(x_i)$  for all training samples

    Calculate new predictions  $f_m(x_i) = f_{m-1} + \eta \cdot h_m(x_i)$

**end for**

**Output:**  $M$  Decision Stumps  $h_1, h_2, \dots, h_M$

# Gradient Boosting Training — Convergence for $x_0$ and $x_1$

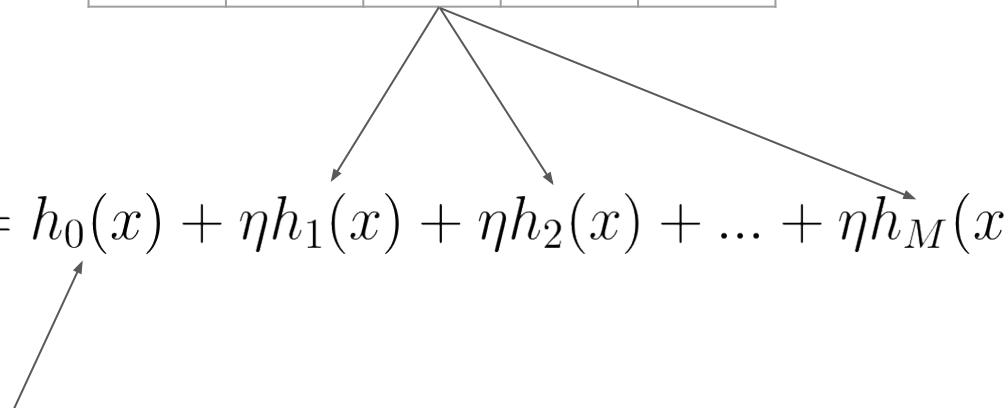


# Gradient Boosting Prediction

Age	Edu- cation	Marital Status	Annual Income	Credit Limit
50	PhD	Single	70k	???

$$h(x) = h_0(x) + \eta h_1(x) + \eta h_2(x) + \dots + \eta h_M(x)$$

Initial prediction  $f_0(x)$   
(e.g.,  $f_0(x) = 1000$ )



# Boosting Methods — Pros & Cons (Compared to Decision Trees)

- Pros
  - High accuracy — often state of the art
- Cons
  - Less Interpretable (arguably even less compared to Random Forests)
  - Slower training and prediction → sequential processing → not parallelizable

# Summary

- **Decision Trees**
  - Intuitive model for classification and regression → interpretable!
  - Can handle categorical and numerical data (although tricky in practice)
  - Typically good but not great results
- **Tree Ensembles**
  - Aim to address limitations of single decision trees (particularly high variance)
  - Ensembles of independent models: Bagging, Random Forests
  - Ensembles of dependent models: AdaBoost, Gradient Boosted Trees
  - State of the art in many application contexts

# Solutions to Quick Quizzes

- Slide 11: Throw error (safe default), use majority label of subtree
- Slide 27: B ("natural" ranking of energy labels is broken)
- Slide 47: A

# **CS5228: Knowledge Discovery and Data Mining**

## Lecture 7 — Classification & Regression III

# Course Logistics — Update

- Assignment 3
  - Available on Canvas (since Oct 13)
  - Submission deadline: Thu Oct 24, 11.59 pm
- Project
  - Progress reports completed + feedback provided to almost most team  
(there will be an announcement with a short summary)
  - 1st TEAMMATES session live (deadline: Oct 17, 11.59 pm)

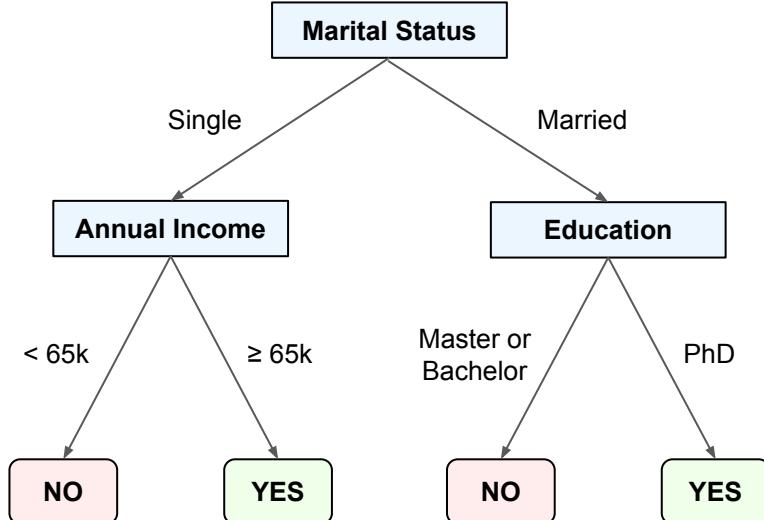
# Quick Recap — Decision Trees

- **Decision Trees**

- Flowchart-like structure mapping input features to output labels or values
- Applicable to classification & regression tasks
- Support for categorical & numerical features
- Typically quite interpretable

- **Building Decision Trees**

- Greedy algorithm iteratively finding the best splits
- Best split = split that minimizes impurity

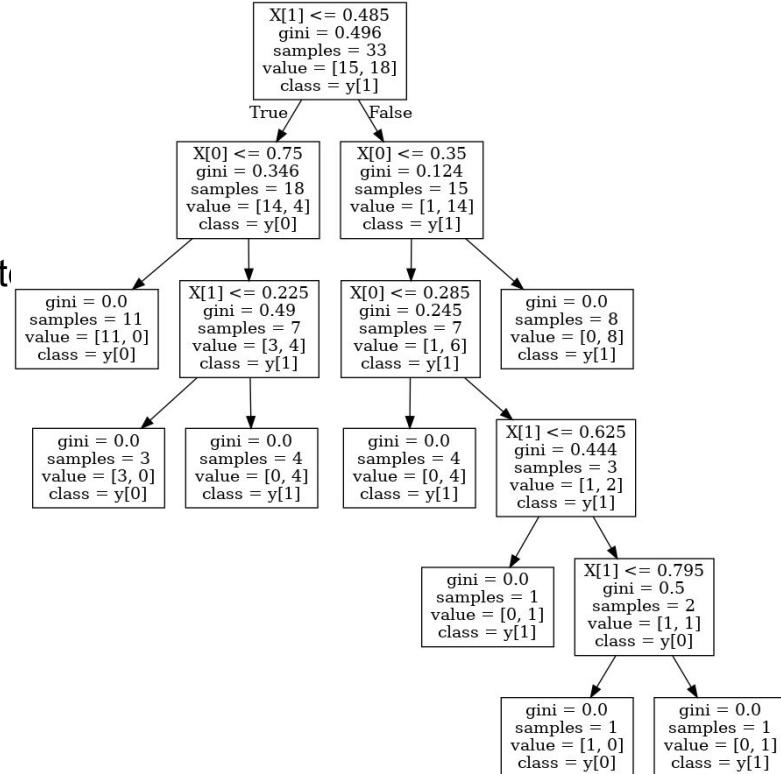


# Quick Recap — Decision Trees

- Challenges

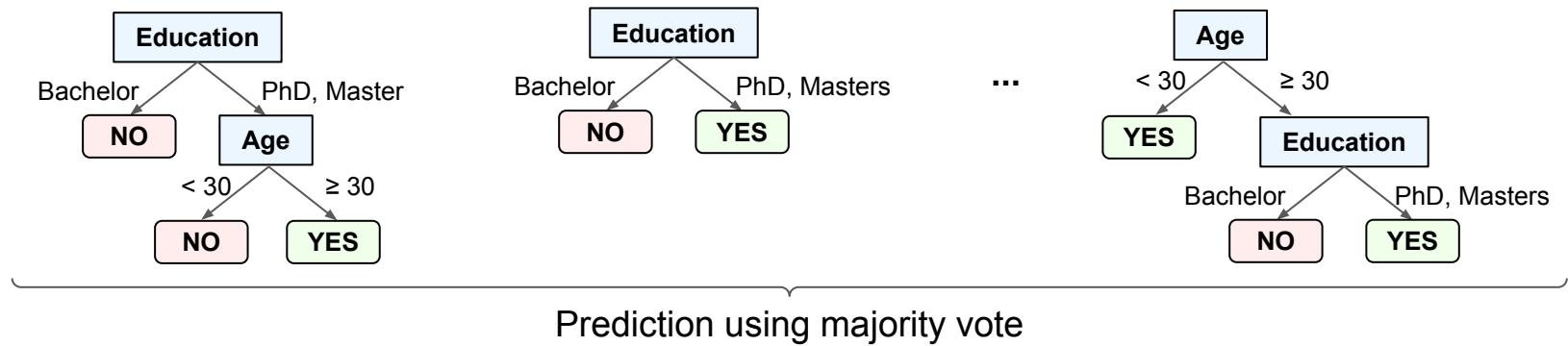
- Sensitive to small changes in training data  
→ High variance
- High chance of overfitting in case of maximum/completely pure tree  
→ Pruning of tree to avoid overfitting
- In practice, Decision Trees do not show state-of-the-art performance

→ Tree Ensemble methods



# Quick Recap — Tree Ensembles

- **Tree ensembles:** Construct many decision trees and combine their predictions
  - Pros: Higher accuracy, lower variance
  - Cons: Lower interpretability, longer training time
- **Ensembles of independent models:**
  - Bagging — Train decision trees over re-sampled training data (bootstrap sampling)
  - Random Forests — bootstrap sampling + feature sampling



# Quick Recap — Tree Ensembles

- Ensembles of dependent models (boosting methods)
  - Sequential training of decision trees
  - Next tree tries to improve errors of previous trees
  - Trees have different amount of say in predictions
- Two introduced boosting methods:
  - AdaBoost — resample training data to favour previously misclassified samples
  - Gradient Boosted Trees — start with initial prediction, improve based on predicted errors

# Outline

- **Linear Models**
  - Basic setup
- **Linear Regression**
  - Problem formulation
  - Normal Equation (analytical solution)
  - Gradient Descent (iterative optimization)
  - Polynomial Linear Regression (overfitting, regularization)
  - Interpretation of Coefficients
- **Logistic Regression**
  - Problem formulation
  - Gradient Descent

# Linear Models

- Basic setup

- Dataset of  $n$  samples  $\{(x_i, y_i)\}_{i=1}^n$
- Input data with  $d$  features  $x_i = (x_{i1}, x_{i2}, \dots, x_{id})$

- Assumption

- There exists linear relationship between  $x_i$  and dependent variable  $y_i$

$$\hat{y}_i = h_\theta(x_i) = f(\theta_0 x_{i0} + \theta_1 x_{i1} + \theta_2 x_{i2} + \dots + \theta_d x_{id})$$

Predicted value which  
is hopefully close to  $y_i$

$$\theta = \{\theta_0, \theta_1, \theta_2, \dots, \theta_d\}, \theta_i \in \mathbb{R}$$

Age	Edu- cation	Marital Status	Income Level	Credit Approval	Credit Limit
23	Masters	Single	Mid	No	\$S5,000
35	College	Married	High	Yes	\$S7,000
26	Masters	Single	High	No	\$S9,000
...	...	...	...	...	...

# Linear Models

- Vector representation

- Introduce constant feature  $x_{i0}$

$$h_\theta(x_i) = f(\underbrace{\theta_0 x_{i0} + \theta_1 x_{i1} + \theta_2 x_{i2} + \dots + \theta_d x_{id}}_{= 1})$$

- Represent  $x_i$  with new constant feature

$$x_i = (1, x_{i1}, x_{i2}, \dots, x_{id})$$

- Rewrite linear relationship using vectors representing  $x_i$  and  $\theta$

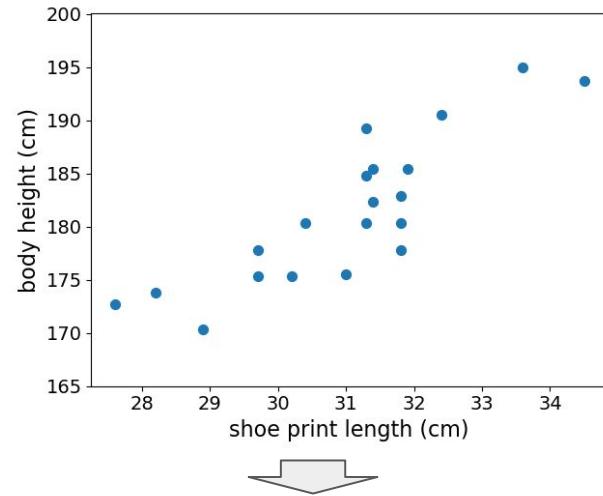
$$h_\theta(x_i) = f(\theta^T x_i)$$

# Outline

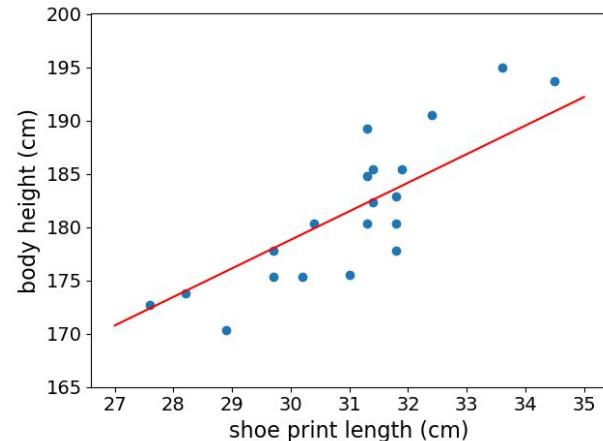
- **Linear Models**
  - Basic setup
- **Linear Regression**
  - **Problem formulation**
  - Normal Equation (analytical solution)
  - Gradient Descent (iterative optimization)
  - Polynomial Linear Regression (overfitting, regularization)
  - Interpretation of Coefficients
- **Logistic Regression**
  - Problem formulation
  - Gradient Descent

# Linear Regression — Simple Example

- Crime scene investigation (CSI)
  - Found a shoe print of size 32.2cm
  - What is the estimated height of the suspect?



- Approach: Linear Regression
  - Collect a dataset of (size, height)-pairs
  - Quantify linear relationship between shoe print size and body height  $\rightarrow h_{\theta}(size) = \theta_0 + \theta_1 size$
  - Predict suspect's height via  $\hat{y} = h_{\theta}(32.2)$



# Linear Regression

- Regression → Real-valued predictions

- Function  $f$  is the identity function  $f(x) = x$

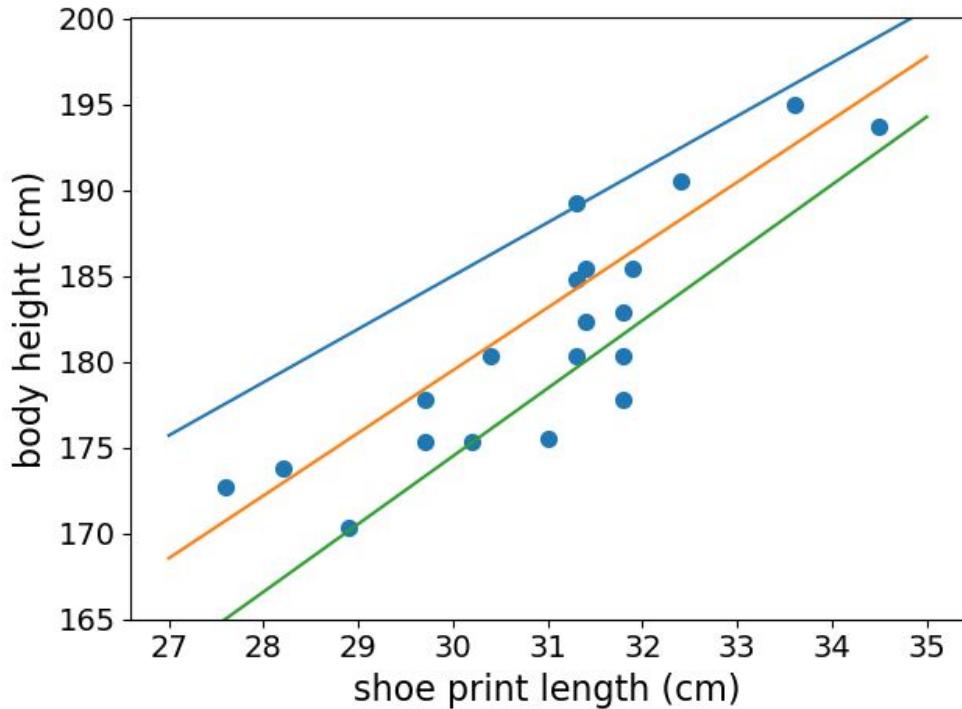
$$\hat{y}_i = h_{\theta}(x_i) = f(\theta^T x_i) = \theta^T x_i$$

$$\hat{y}_i = \theta^T x_i$$

$$\hat{y} = X\theta$$

$$\hat{y} = \begin{bmatrix} \hat{y}_1 \\ \hat{y}_2 \\ \vdots \\ \hat{y}_n \end{bmatrix} \quad X = \begin{bmatrix} 1 & x_{11} & x_{12} & \dots & x_{1d} \\ 1 & x_{21} & x_{22} & \dots & x_{2d} \\ 1 & \vdots & \vdots & \ddots & \vdots \\ 1 & x_{n1} & x_{n2} & \dots & x_{nd} \end{bmatrix} \quad \theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \vdots \\ \theta_d \end{bmatrix}$$

# Quick Quiz



Which is the best line?

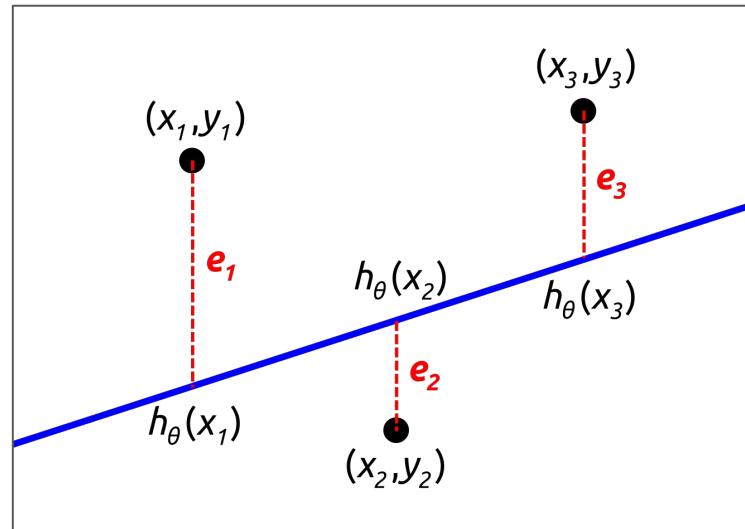
Why?

How to find it?

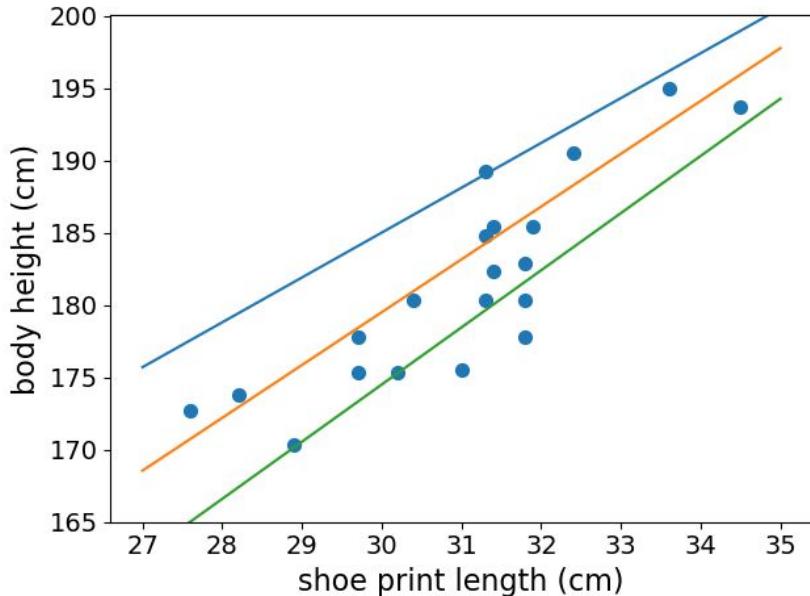
# Linear Regression — Loss Function

- **Loss function** (also: cost function, error function)
  - Quantifies how good or bad a given set of values for  $\theta$  is?
  - Measures the difference between predictions  $\hat{y}$  and true values  $y$
- Loss function for Linear Regression:  
**Mean Squared Error (MSE)**

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



# Losses for CSI Example



$$h_{blue} = 92 + 3.10 \cdot x$$
$$h_{orange} = 69 + 3.61 \cdot x$$
$$h_{green} = 56 + 3.95 \cdot x$$
$$h_{random} = 100 - 5.00 \cdot x$$

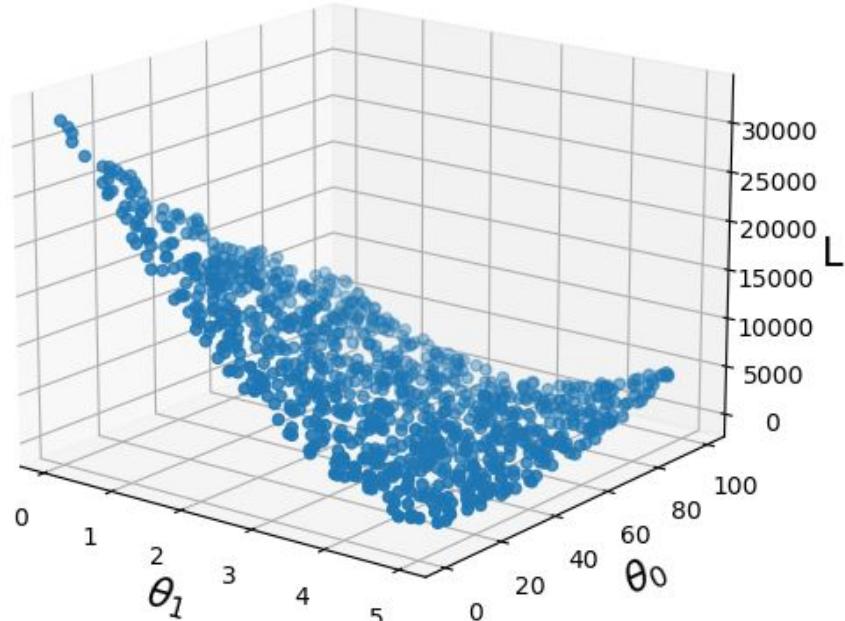
$$L_{blue} = 57.47$$
$$L_{orange} = 12.12$$
$$L_{green} = 20.83$$
$$L_{random} = 56, 129.23$$

→ How to find the best values for  $\theta$ ?

# Method 1: Random Search (the "stupid" way)

- Repeat "enough" times
  - Select random values for  $\theta = \{\theta_0, \theta_1, \dots, \theta_d\}$
  - Calculate loss L for current  $\theta$
- Return  $\theta$  with smallest loss
- Limitation:
  - Not practical beyond toy examples
- Don't do that! :)

Plot of 1,000 losses



# Outline

- **Linear Models**
  - Basic setup
- **Linear Regression**
  - Problem formulation
  - **Normal Equation** (analytical solution)
  - Gradient Descent (iterative optimization)
  - Polynomial Linear Regression (overfitting, regularization)
  - Interpretation of Coefficients
- **Logistic Regression**
  - Problem formulation
  - Gradient Descent

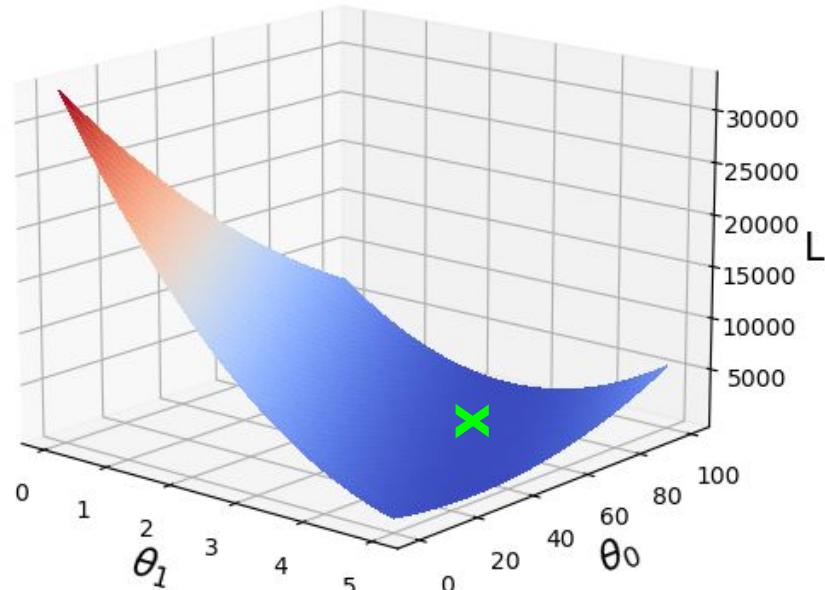
# Method 2: Find Minimum of $L$ Analytically (the proper way)

- Minimum of loss function  $L \rightarrow$  Calculus to the rescue!

- Partial derivatives w.r.t. to all  $\theta_i$  are 0

$$\frac{\partial L}{\partial \theta_0} = 0, \frac{\partial L}{\partial \theta_1} = 0, \dots, \frac{\partial L}{\partial \theta_d} = 0$$

- $d+1$  equations with  $d+1$  unknowns
- (no need to check if minimum or maximum)



# Method 2: Find Minimum of $L$ Analytically (the proper way)

- Rewrite loss function  $L$ 
  - Vector representation mathematically more convenient to handle
  - Avoids dealing with  $d$  equations

$$\begin{aligned} L &= \frac{1}{n} \sum_{i=1}^n (\hat{y}_i - y_i)^2 \\ &= \frac{1}{n} \sum_{i=1}^n (\theta^T x_i - y_i)^2 \\ &= \frac{1}{n} \|X\theta - y\|^2 \end{aligned}$$

- Derive  $L$  w.r.t. to  $\theta$ 
  - Using chain rule
- Set  $\partial L / \partial \theta$  to 0

$$\frac{\partial L}{\partial \theta} = \frac{2}{n} X^T (X\theta - y)$$

$$\frac{2}{n} X^T (X\theta - y) \stackrel{!}{=} \overrightarrow{0}$$

# Linear Regression — Normal Equation

- Solve for  $\theta$

$$\frac{2}{n} X^T (X\theta - y) = \vec{0}$$

$$X^T X\theta = X^T y$$

$$(X^T X)^{-1} X^T X\theta = (X^T X)^{-1} X^T y$$

$$\theta = (X^T X)^{-1} X^T y$$

$$\theta = X^\dagger y, \text{ with } X^\dagger = \underbrace{(X^T X)^{-1} X^T}_{\text{"pseudo inverse" of } X}$$

# Pseudo Inverse $X^\dagger$

$$X^\dagger = (X^T X)^{-1} X^T$$

$$\begin{matrix} X \\ X^T \end{matrix} \begin{matrix} X \\ (d+1) \times n \end{matrix} \left[ \begin{matrix} X \\ n \times (d+1) \end{matrix} \right] = \left[ \begin{matrix} X^T X \\ (d+1) \times (d+1) \end{matrix} \right] \longrightarrow \underbrace{\left[ \begin{matrix} (X^T X)^{-1} \\ (d+1) \times (d+1) \end{matrix} \right]^{-1}}_{(d+1) \times n} \left[ \begin{matrix} X^T \\ (d+1) \times n \end{matrix} \right]$$

## • Performance analysis

- Most expensive operation: calculating the inverse of  $(X^T X)^{-1}$
- Calculation of inverse depends on number of features  $d$ , not on number of data samples  $n$
- Complexity of calculating inverse of a  $d \times d$  matrix:  $O(d^3)$

# Quick Quiz

What condition is **not required** for a matrix A to be **invertible**?

**A**

The determinant of A is non-zero

**B**

A is a square matrix

**C**

A has full rank

**D**

All diagonal values are non-zero

# Quick Quiz

When will  $X^T X$  **not** be invertible?

A

There are more features  $d$  than data samples  $n$

B

The data is not normalized

C

In practice,  $X^T X$  will always be invertible

D

$X$  has no determinant

# Outline

- **Linear Models**
  - Basic setup
- **Linear Regression**
  - Problem formulation
  - Normal Equation (analytical solution)
  - **Gradient Descent** (iterative optimization)
  - Polynomial Linear Regression (overfitting, regularization)
  - Interpretation of Coefficients
- **Logistic Regression**
  - Problem formulation
  - Gradient Descent

# Method 2: Find Minimum of $L$ Analytically (the proper way)

- **Algorithm**

- Construct matrix  $X$  and vector  $y$  from data
- Calculate pseudo inverse  $X^\dagger = (X^T X)^{-1} X^T$
- Return  $\theta = X^\dagger y$

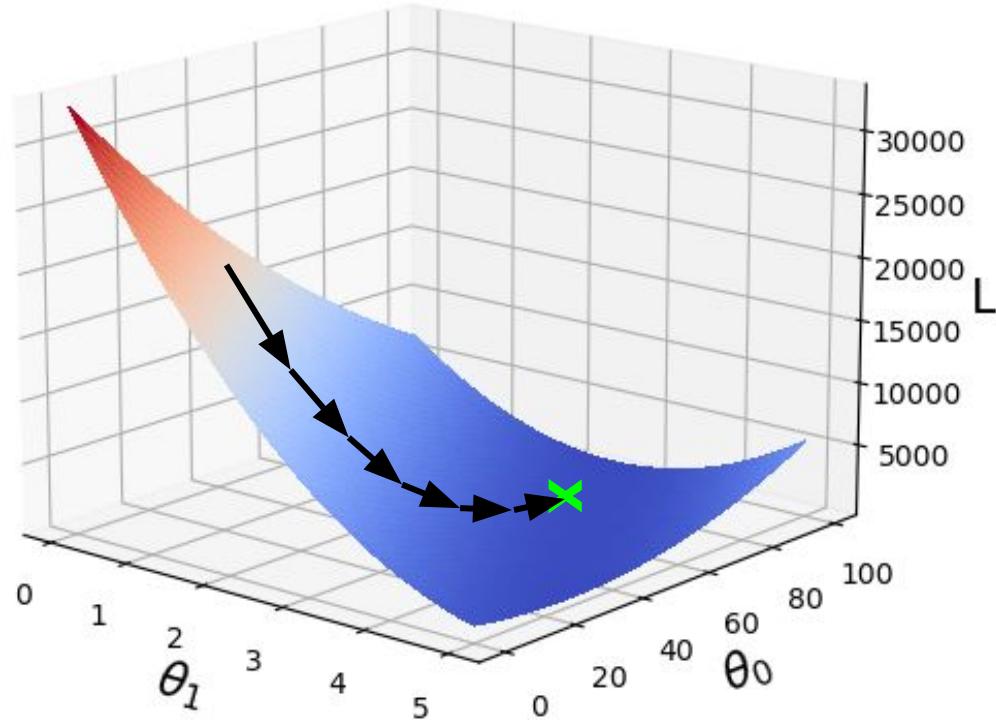
- For the CSI example:

$$\theta = \begin{bmatrix} 20.0 & 620.2 \\ 620.0 & 19284.9 \end{bmatrix}^{-1} X^T y = \begin{bmatrix} 18.4 & -0.6 \\ -0.6 & 0.02 \end{bmatrix} X^T y = \begin{bmatrix} 69.5 \\ 3.6 \end{bmatrix}$$

$$h_\theta(\text{size}) = 69.5 + 3.6\text{size} \quad h_\theta(32.2) = 185.4$$

# Method 3: Gradient Descent

- Core idea
  - Start with a random setting of  $\theta$
  - Adjust  $\theta$  iteratively to minimize  $L$



# Gradient — Quick Refresher

- Gradient

- Vector of partial derivatives of a multivariable function (e.g.,  $\theta_0, \theta_1, \dots, \theta_d$ )
- Partial derivative: slope w.r.t. to a single variable given a current set of values for all  $\theta_0, \theta_1, \dots, \theta_d$
- Points in the direction of the steepest ascent

$$\nabla_{\theta} L = \frac{\partial L}{\partial \theta} = \begin{bmatrix} \frac{\partial L}{\partial \theta_0} \\ \frac{\partial L}{\partial \theta_1} \\ \frac{\partial L}{\partial \theta_2} \\ \vdots \\ \frac{\partial L}{\partial \theta_d} \end{bmatrix}$$

# Gradient for CSI Example

Best value:  $\theta_0 = 69.5, \theta_1 = 3.6$

- Assume  $\theta_0 = 60$  and  $\theta_1 = 4$

$$\nabla_{\theta} L = \frac{2}{n} X^T (X\theta - y) = \frac{2}{n} X^T (X \cdot \begin{bmatrix} 60 \\ 4 \end{bmatrix} - y) = \begin{bmatrix} 5.2 \\ 163.9 \end{bmatrix}$$

- Interpretation of  $\nabla_{\theta} L = \begin{bmatrix} 5.2 \\ 163.9 \end{bmatrix}$ 
  - Both values positive: a small increase of  $\theta_0$  or  $\theta_1$  will increase the loss
  - A small change in  $\theta_1$  affects the loss more than the same change in  $\theta_0$
  - Absolute values of gradient not a direct indicator of how to update  $\theta$

# Gradient Descent Algorithm

- Important concept: learning rate
  - Scaling factor for gradient (typical range: 0.01 - 0.0001)

**Input** : data  $(X, y)$ , loss function  $L$ , learning rate  $\eta$

**Initialization** : Set  $\theta$  to random values

**while true :**

    Calculate gradient  $\nabla_{\theta} L$

$\theta \leftarrow \theta - (\eta \cdot \nabla_{\theta} L)$

In practice: stop loop  
when  $\theta$  converges

# Gradient Descent for CSI Example

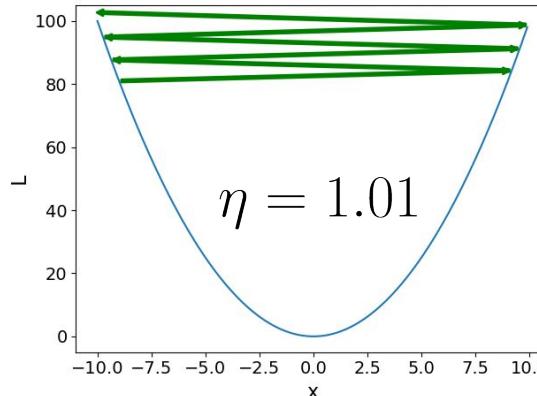
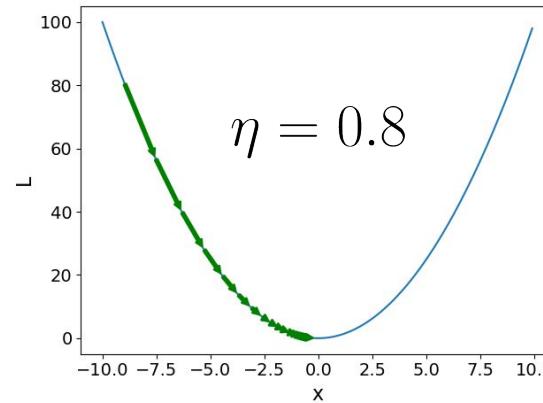
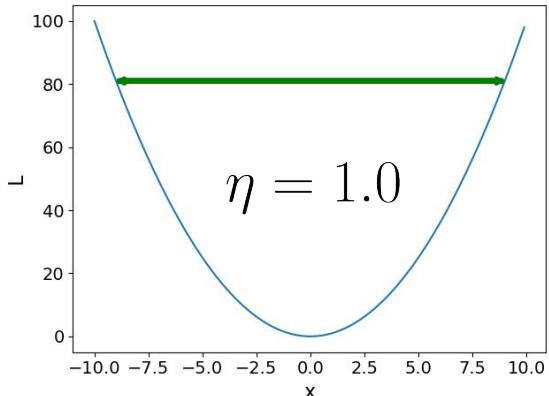
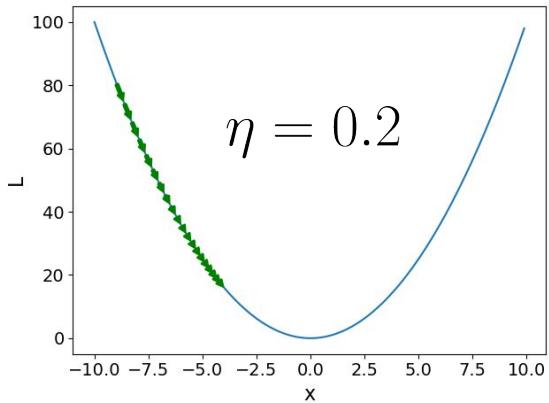
- Input
  - $\eta = 0.0001$
- Initialization
  - $\theta_0 = 100$
  - $\theta_1 = 50$

```
[001] theta0=99.70619, theta1=40.86448, loss=2163825.04200
[002] theta0=99.46909, theta1=33.49256, loss=1409025.31406
[003] theta0=99.27776, theta1=27.54378, loss=917521.53239
[004] theta0=99.12336, theta1=22.74340, loss=597468.46605
[005] theta0=98.99877, theta1=18.86972, loss=389059.15347
[006] theta0=98.89823, theta1=15.74385, loss=253349.02868
[007] theta0=98.81709, theta1=13.22143, loss=164978.51518
[008] theta0=98.75161, theta1=11.18595, loss=107434.18922
[009] theta0=98.69877, theta1=9.54342, loss=69962.98624
[010] theta0=98.65613, theta1=8.21798, loss=45562.82115
[011] theta0=98.62172, theta1=7.14841, loss=29674.13840
[012] theta0=98.59395, theta1=6.28532, loss=19327.88711
[013] theta0=98.57153, theta1=5.58885, loss=12590.70710
[014] theta0=98.55344, theta1=5.02683, loss=8203.65008
[015] theta0=98.53884, theta1=4.57330, loss=5346.92525
[016] theta0=98.52706, theta1=4.20733, loss=3486.70856
[017] theta0=98.51754, theta1=3.91201, loss=2275.38917
[018] theta0=98.50986, theta1=3.67370, loss=1486.61298
[019] theta0=98.50366, theta1=3.48140, loss=972.98470
[020] theta0=98.49866, theta1=3.32622, loss=638.52480
...
[098] theta0=98.47656, theta1=2.67760, loss=14.17658
[099] theta0=98.47655, theta1=2.67760, loss=14.17658
[100] theta0=98.47653, theta1=2.67760, loss=14.17658
```

**Quick Quiz:** Why not just increase the learning rate to speed things up?

# Effects of Learning Rate for

$$L = x^2, \frac{\partial L}{\partial x} = 2x, \text{ 20 steps}$$

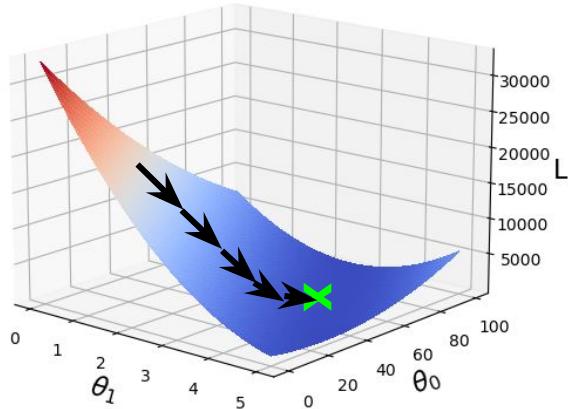


# Gradient Descent — Variations

- (Basic) Gradient Descent
  - Calculate gradient und update  $\theta$  for whole dataset
- Stochastic Gradient Descent (SGD)
  - Calculate gradient und update  $\theta$  for each data sample
- Mini-batch Gradient Descent
  - Calculate gradient und update  $\theta$  for batches of sample
  - e.g., batch = 64 data samples
  - In practice often referred to as SGD

# Gradient Descent — Variations

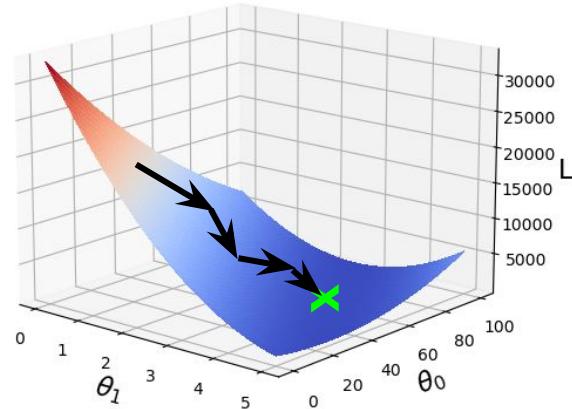
Gradient Descent



Gradient averaged over all data items

- Smooth descent
- Small(er) gradients
- Small(er) update steps

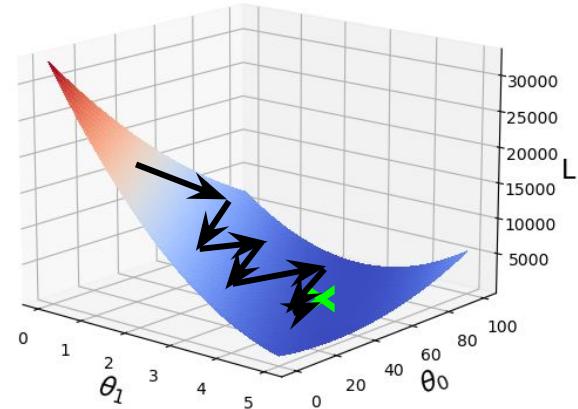
Mini-Batch Gradient Descent



Gradient averaged over some data items

- Well, "somewhere in-between" :)

Stochastic Gradient Descent



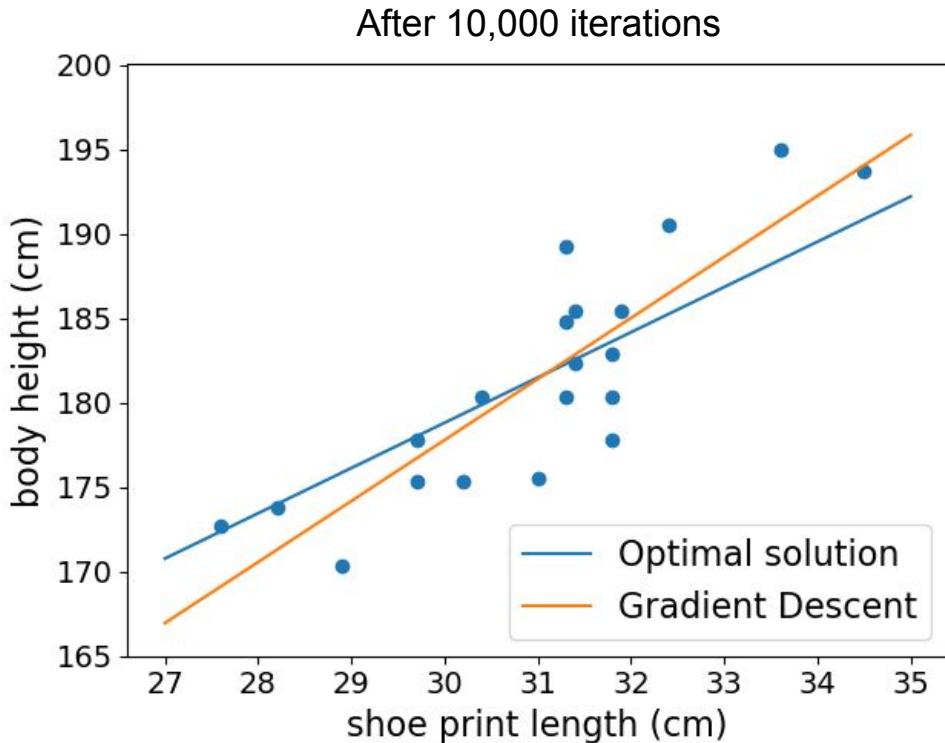
Gradient for each data item considered

- Choppy descent
- Large(r) gradients
- Large(r) steps

# Normal Equation vs. Gradient Descent

- Gradient Descent
  - Works well even if  $d$  is large
  - Works even if  $X^T X$  is non-invertible
  - Iterative process; may not find optimal solution in practice
  - Learning rate a critical hyperparameter
  
- Normal Equation
  - Finds optimal solutions
  - Non-iterative; no need of learning rate
  - Calculation of  $(X^T X)^{-1}$  in  $O(d^3)$

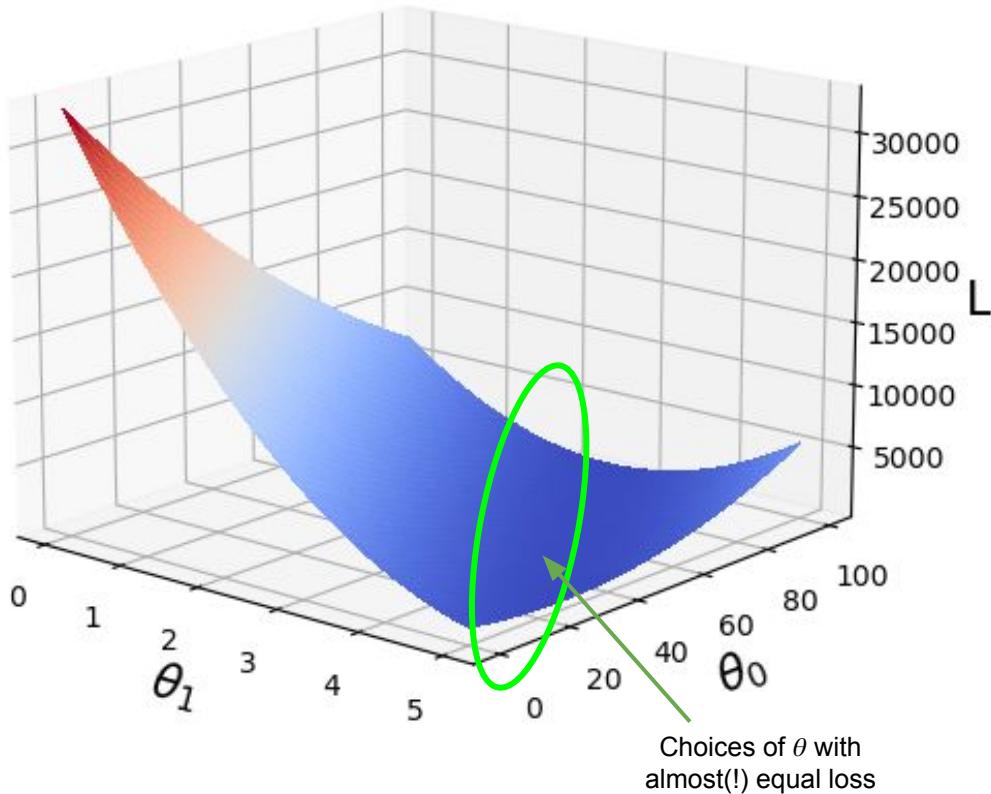
# Quick Quiz



Gradient Descent not reaching optimal solution after 10k iterations.

Why?

# Quick Quiz



Region of "near-plateau":

- Gradient  $\nabla_{\theta} L$  very small
- Step  $\eta \nabla_{\theta} L$  extremely small
- Very slow convergence

# Outline

- **Linear Models**
  - Basic setup
- **Linear Regression**
  - Problem formulation
  - Normal Equation (analytical solution)
  - Gradient Descent (iterative optimization)
  - **Polynomial Linear Regression** (overfitting, regularization)
  - Interpretation of Coefficients
- **Logistic Regression**
  - Problem formulation
  - Gradient Descent

# Polynomial Linear Regression

- Linear Regression  $\not\rightarrow$  line / plane / hyperplane
- Polynomial Linear Regression
  - Allows to capture nonlinear relationships between  $X$  and  $y$
  - Polynomial regression model for 1 input feature

Still linear in  $\theta$ !

$$\hat{y}_i = \theta_0 1 + \theta_1 x_i + \theta_2 x_i^2 + \dots + \theta_p x_i^p$$

- Matrix representation (again, 1 input feature!)

$$X^{(1)} = \begin{bmatrix} 1 & x_1 \\ 1 & x_2 \\ \vdots & \vdots \\ 1 & x_n \end{bmatrix} \quad X^{(2)} = \begin{bmatrix} 1 & x_1 & x_1^2 \\ 1 & x_2 & x_2^2 \\ \vdots & \vdots & \vdots \\ 1 & x_n & x_n^2 \end{bmatrix} \quad X^{(3)} = \begin{bmatrix} 1 & x_1 & x_1^2 & x_1^3 \\ 1 & x_2 & x_2^2 & x_2^3 \\ \vdots & \vdots & \vdots & \vdots \\ 1 & x_n & x_n^2 & x_n^3 \end{bmatrix} \quad X^{(p)} = \begin{bmatrix} 1 & x_1 & x_1^2 & x_1^3 & \dots & x_1^p \\ 1 & x_2 & x_2^2 & x_2^3 & \dots & x_2^p \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_n & x_n^2 & x_n^3 & \dots & x_n^p \end{bmatrix}$$

# Polynomial Linear Regression

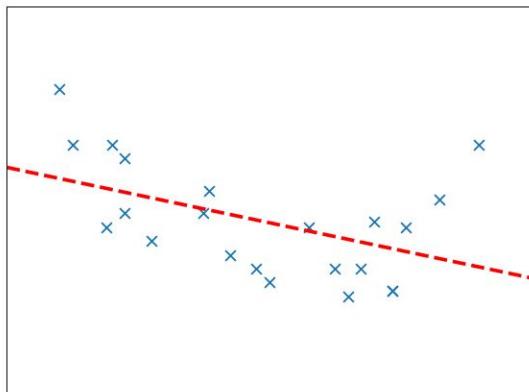
- Example: 1 input feature, 3 data samples

$$X^{(1)} = \begin{bmatrix} 1 & 2 \\ 1 & 3 \\ 1 & 1 \end{bmatrix} \quad X^{(2)} = \begin{bmatrix} 1 & 2 & 4 \\ 1 & 3 & 9 \\ 1 & 1 & 1 \end{bmatrix} \quad X^{(3)} = \begin{bmatrix} 1 & 2 & 4 & 8 \\ 1 & 3 & 9 & 27 \\ 1 & 1 & 1 & 1 \end{bmatrix}$$

- Polynomial terms "look the same" as additional features
- Finding best  $\theta$  using the Normal Equation or Gradient Descent

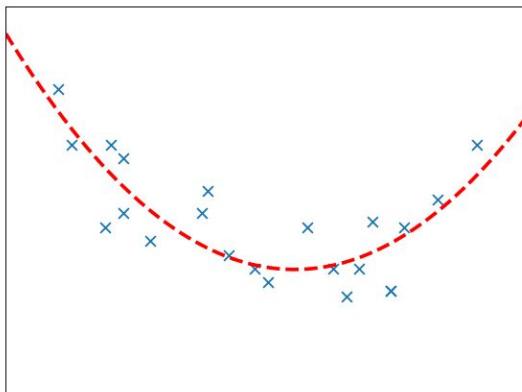
# Polynomial Linear Regression — Example

$p = 1$



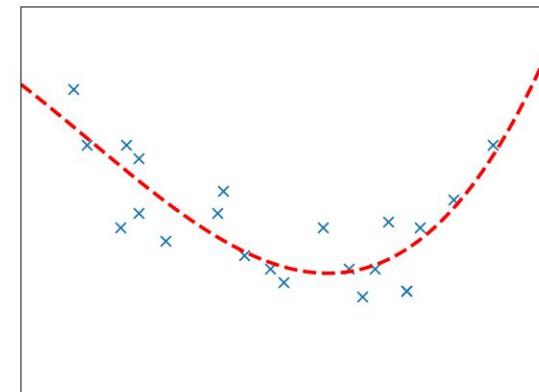
Underfitting

$p = 2$



Good fit

$p = 3$



Overfitting?

# Polynomial Linear Regression — Overfitting

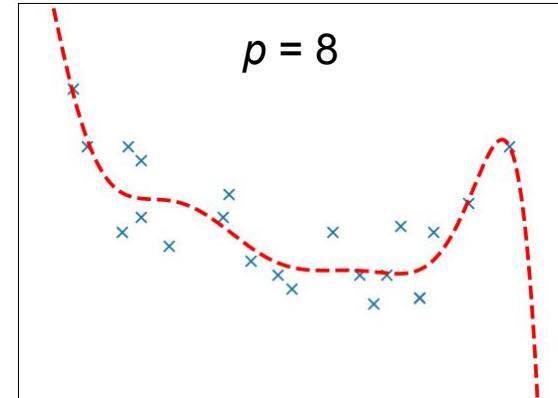
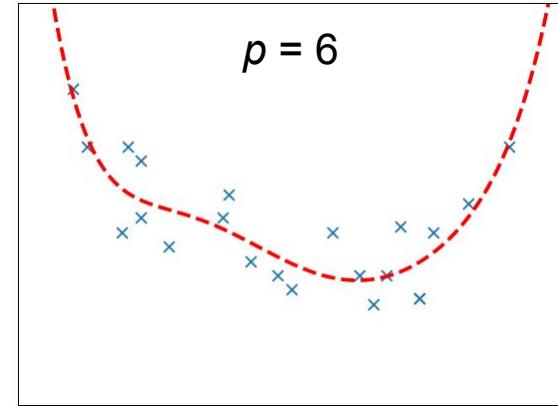
- Increasing degree of polynomial  $p$ 
  - More capacity to capture nonlinear relationships
  - Much higher sensitivity to noise and outliers
- Countermeasure: **Regularization**
  - Extend loss function to "punish" large values of  $\theta$

$$L = \frac{1}{n} \|X\theta - y\|^2 + \lambda \frac{1}{n} \|\theta\|_2^2$$

Regularization parameter

$$\|\theta\|_2^2 = \sum_{i=1}^d \theta_i^2$$

Note: excludes  $\theta_0$ !



# Polynomial Linear Regression — Minimizing Loss $L$

- Normal Equation

$$\theta = (X^T X + \lambda \begin{bmatrix} 0 & & & \\ & 1 & & \\ & & \ddots & \\ & & & 1 \end{bmatrix})^{-1} X^T y$$

- Gradient Descent

$$\nabla_{\theta} L = \frac{2}{n} X^T (X\theta - y) + \lambda \frac{2}{n} \theta$$

# Polynomial Linear Regression — More than 1 Feature

- Polynomial of degree  $p=2$  and two input features ( $d=2$ )

$$\hat{y}_i = \theta_0 + \theta_1 x_{i1} + \theta_2 x_{i2} + \theta_3 x_{i1}^2 + \theta_4 x_{i2}^2 + \underbrace{\theta_5 x_{i1} x_{i2}}_{\text{interaction terms (cross terms)}}$$

- Polynomial of degree  $p=3$  and two input features ( $d=2$ )

$$\begin{aligned}\hat{y}_i = & \theta_0 + \theta_1 x_{i1} + \theta_2 x_{i2} + \theta_3 x_{i1}^2 + \theta_4 x_{i2}^2 + \theta_5 x_{i1}^3 + \theta_6 x_{i2}^3 + \\ & \theta_7 x_{i2}^2 x_{i1} + \theta_8 x_{i2} x_{i1}^2 + \theta_9 x_{i2} x_{i1}\end{aligned}$$

- Polynomial of degree  $p=2$  and three input features ( $d=3$ )

$$\begin{aligned}\hat{y}_i = & \theta_0 + \theta_1 x_{i1} + \theta_2 x_{i2} + \theta_3 x_{i3} + \theta_4 x_{i1}^2 + \theta_5 x_{i2}^2 + \theta_6 x_{i3}^2 \\ & \theta_7 x_{i2} x_{i1} + \theta_8 x_{i3} x_{i1} + \theta_9 x_{i3} x_{i2}\end{aligned}$$

# Polynomial Linear Regression — More than 1 Feature

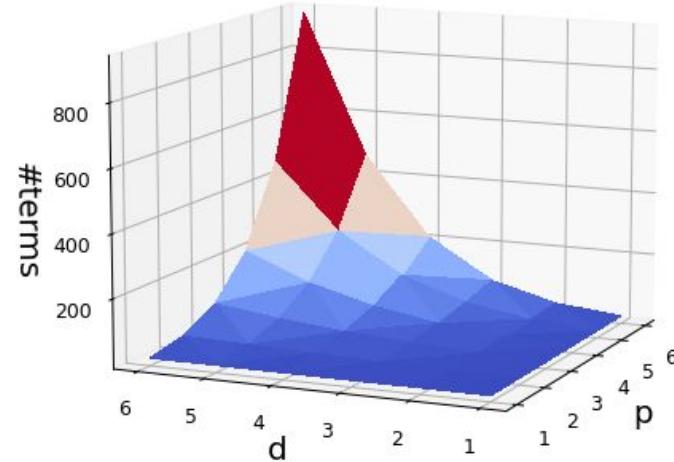
- Number of terms in multivariate polynomial given given  $p, d$

$$\theta_i, 0 \leq i \leq M, \text{ with } M = \binom{p+d}{p}$$

$$\binom{n}{k} = \frac{n!}{k!(n-k)!}$$

- Practical considerations

- Limited to small number of features and small polynomial degrees
- In principle, terms can be dropped (e.g., all interaction terms)



} risk of overfitting + low interpretability

} justification typically not obvious

# Outline

- **Linear Models**
  - Basic setup
- **Linear Regression**
  - Problem formulation
  - Normal Equation (analytical solution)
  - Gradient Descent (iterative optimization)
  - Polynomial Linear Regression (overfitting, regularization)
  - **Interpretation of Coefficients**
- **Logistic Regression**
  - Problem formulation
  - Gradient Descent

# Linear Regression — Interpretation of $\theta_i$

- Example: prediction of house prices

$$\left. \begin{array}{l} \text{price} = \theta_0 + \theta_1(\#\text{rooms}) + \theta_2(\text{area}) + \theta_3(\text{floor}) + \dots \\ \text{price}' = \theta_0 + \theta_1(\#\text{rooms} + 1) + \theta_2(\text{area}) + \theta_3(\text{floor}) + \dots \end{array} \right\} \Delta(\text{price}) = \text{price}' - \text{price} = \theta_1$$

- Interpretation
  - Change of value of feature  $i$  by 1 unit → change of output value by  $\theta_i$
  - Assumption: all other features values remain the same

→ What about normalizing the data?

# Data Normalization — Yes or No? (standardization, min-max scaling)

- Data normalization does not affect model performance  
(assuming basic Linear Regression without regularization)
- In favor of "No"
  - Preserves unit of feature  $i \rightarrow$  direct interpretation of  $\theta_i$
  - Better for comparing  $\theta_i$  for the same features across different datasets
- In favour of "Yes"
  - When using regularization
  - When using Polynomial Linear Regression
  - Better for comparing  $\theta_i$  within a model (e.g.,  $\theta_i > \theta_j \rightarrow$  feature  $i$  more important than feature  $j$ )

# Quick Quiz

Which of the statements regarding Linear Regression is **True**?

**A**

It's impossible to overfit given a dataset with only 1 feature

**B**

Scaling the data will change the coefficients

**C**

Gradient Descent can get stuck in local minimum

**D**

Regularization can improve the training loss/error

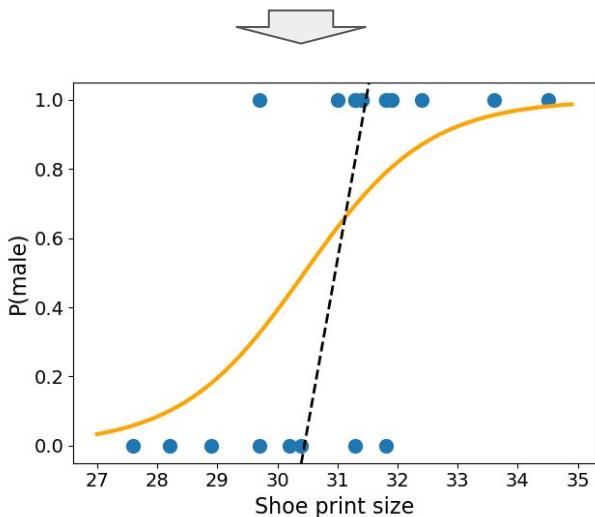
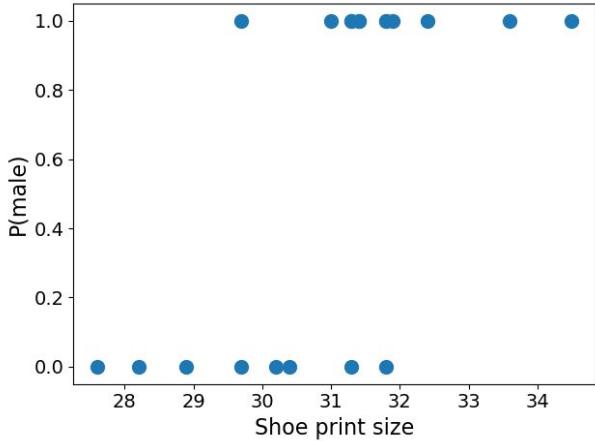
# Overview

- **Linear Models**
  - Basic setup
- **Linear Regression**
  - Problem formulation
  - Normal Equation (analytical solution)
  - Gradient Descent (iterative optimization)
  - Polynomial Linear Regression (overfitting, regularization)
  - Interpretation of Coefficients
- **Logistic Regression**
  - Problem formulation
  - Gradient Descent

# Logistic Regression — Simple Example

- Crime scene investigation (CSI)
  - Found a shoe print of size 32.2cm
  - Is the suspect a male or not?
- Approach: Logistic Regression for binary classification  $y_i \in \{0, 1\}$ 
  - Collect a dataset of (size, sex)-pairs
  - Train linear classifier such that

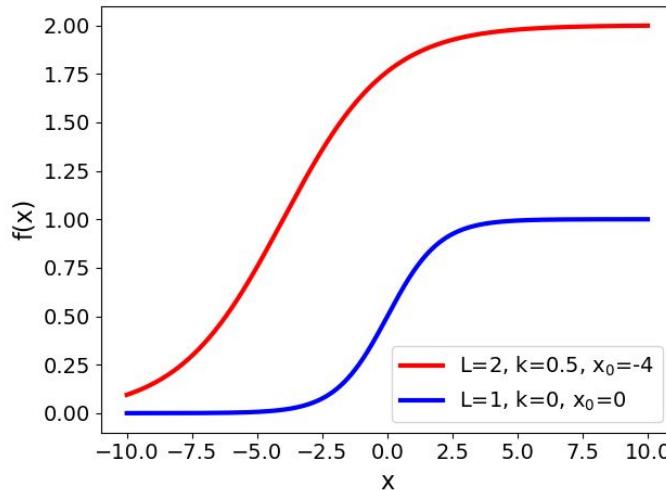
$$0 \leq h_{\theta}(x_i) \leq 1$$



# Logistic Regression

- Logistic Regression → Real-valued predictions interpreted as probability
  - Function  $f$  is the standard **Logistic Function** (Sigmoid function)

$$f(x) = \frac{L}{1 + e^{-k(x-x_0)}} \xrightarrow{L=1, k=1, x_0=0} f(x) = \frac{1}{1 + e^{-x}}$$



# Logistic Regression — Probabilistic Interpretation

- $\hat{y}$  interpreted as a probability

$$\hat{y} = h_{\theta}(x) = f(\theta^T x) = \frac{1}{1 + e^{-\theta^T x}} \quad \text{with } \hat{y} \in [0, 1]$$

→  $\hat{y} = h_{\theta}(x)$  is the estimated probability that  $y_i = 1$  (male) given  $x$  and  $\theta$

$$\hat{y} = P(y = 1|x, \theta)$$

→ Given only discrete 2 outcomes:  $P(y = 1|x, \theta) + P(y = 0|x, \theta) = 1$

$$\hat{y} = 1 - P(y = 0|x, \theta)$$

# Logistic Regression — Probabilistic Interpretation

$$\hat{y} = P(y = 1|x, \theta) = 1 - P(y = 0|x, \theta)$$

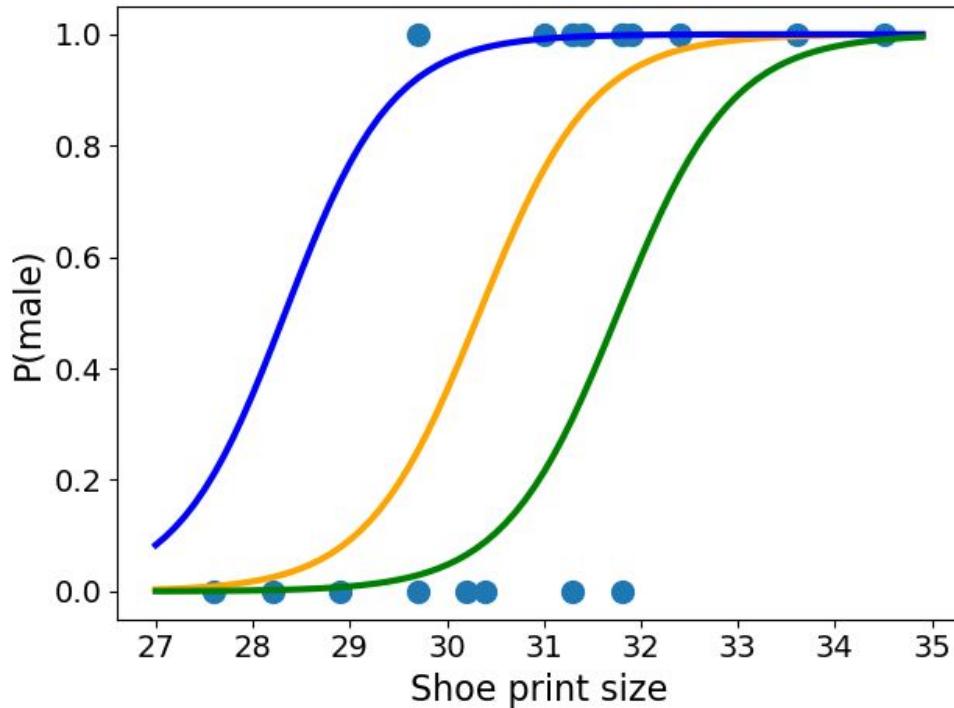
→  $P(y|x)$  is a Bernoulli distribution

$$P(y|x) = \begin{cases} \hat{y} & , y = 1 \\ 1 - \hat{y} & , y = 0 \end{cases}$$

$$P(y|x) = \hat{y}^y (1 - \hat{y})^{1-y}$$

$$\hat{y} = \frac{1}{1 + e^{-\theta^T x}}$$

# Logistic Regression



Which  $h_{\theta}(x)$  is the best?

How to find it  $\theta$ ?

# Logistic Regression — Loss Function

$$\hat{y} = \frac{1}{1 + e^{-\theta^T x}}$$

- Goal: Maximize probability of true  $y$  label given training sample  $x$

- Find  $\theta$  that **maximizes**

$$P(y|x) = \hat{y}^y (1 - \hat{y})^{1-y}$$

$$\begin{aligned}\log P(y|x) &= \log [\hat{y}^y (1 - \hat{y})^{1-y}] \\ &= y \log \hat{y} + (1 - y) \log (1 - \hat{y})\end{aligned}$$

- Find  $\theta$  that **minimizes**

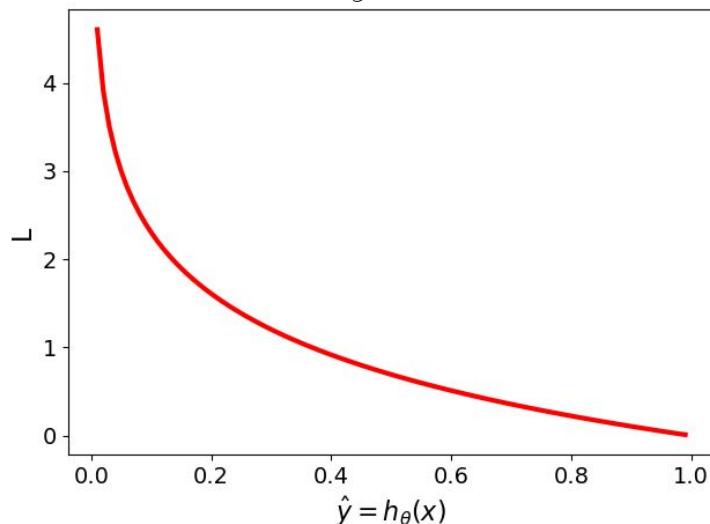
$$L = -P(y|x) = -[y \log \hat{y} + (1 - y) \log (1 - \hat{y})]$$

Cross-Entropy Loss

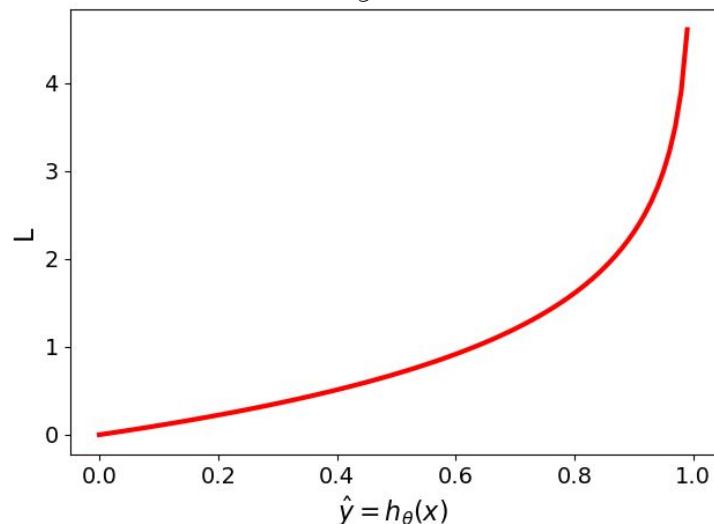
# Cross-Entropy Loss — Visualization

$$L = -[y \log \hat{y} + (1 - y) \log (1 - \hat{y})]$$

if  $y = 1$



if  $y = 0$



# Quick Quiz

What happens if  
Logistic Regression gets a  
training sample **correct**?

A

No loss will be calculated

B

The loss will be 0

C

The loss will be small

D

The loss will be large

# Overview

- **Linear Models**
  - Basic setup
- **Linear Regression**
  - Problem formulation
  - Normal Equation (analytical solution)
  - Gradient Descent (iterative optimization)
  - Polynomial Linear Regression (overfitting, regularization)
  - Interpretation of Coefficients
- **Logistic Regression**
  - Problem formulation
  - Gradient Descent

# Logistic Regression — Loss Function

$$\hat{y} = \frac{1}{1 + e^{-\theta^T x}}$$

- Loss for all training samples

$$\begin{aligned} L &= -\frac{1}{n} \sum_{i=1}^n [y_i \log \hat{y}_i + (1 - y_i) \log (1 - \hat{y}_i)] \\ &= -\frac{1}{n} \sum_{i=1}^n [y_i \log h_\theta(x_i) + (1 - y_i) \log (1 - h_\theta(x_i))] \\ &= -\frac{1}{n} \sum_{i=1}^n \left[ y_i \log \frac{1}{1 + e^{\theta^T x_i}} + (1 - y_i) \log \left(1 - \frac{1}{1 + e^{\theta^T x_i}}\right) \right] \end{aligned}$$

- Required for minimizing the loss:  $\nabla_\theta L = \frac{\partial L}{\partial \theta} = ???$

# Logistic Regression — Loss Function

- After lots of tedious math...

$$\frac{\partial L}{\partial \theta_j} = \frac{1}{n} \sum_{i=1}^n [h_\theta(x_i) - y_i] x_{ij}$$

$$\nabla_\theta L = \frac{1}{n} X^T (h_\theta(X) - y)$$

- Problem:  $\frac{1}{n} X^T (h_\theta(X) - y) \stackrel{!}{=} 0$  has no closed-form solution for  $\theta$

→ Gradient Descent!

# Logistic Regression in Practice (CSI Example)

Vanilla implementation

```
Start training for 1,000,000 iterations...
Loss: 0.6931471805599453      0.0%
Loss: 0.765069661957756      10.0%
Loss: 0.6093119537276577     20.0%
Loss: 0.5066673325457598     30.0%
Loss: 0.4551164039897514     40.0%
Loss: 0.4280635319707213     50.0%
Loss: 0.4154389042684111     60.0%
Loss: 0.4152887492109594     70.0%
Loss: 0.4151849953246913     80.0%
Loss: 0.41511286000667447    90.0%
loss: 0.41506245481850296   100.0%
Training finished in 0:00:09.617970
```

`sklearn.linear_model.LogisticRegression`  
(with default L-BFGS-B solver)

```
Start training...
Training finished in 0:00:00.000035 (#iterations: 21)
```

$$\theta = \begin{bmatrix} \theta_0 \\ \theta_1 \end{bmatrix} = \begin{bmatrix} -43.45 \\ 1.42 \end{bmatrix}$$

$$\theta = \begin{bmatrix} \theta_0 \\ \theta_1 \end{bmatrix} = \begin{bmatrix} -44.91 \\ 1.47 \end{bmatrix}$$

# Logistic Regression in Practice (CSI Example)

Vanilla implementation

```
Start training for 78,000 iterations...
Loss: 0.6931471805599453      0.0%
Loss: 0.6565544350291496      10.0%
Loss: 0.6475478825116517      20.0%
Loss: 0.6389819932273285      30.0%
Loss: 0.6308342158505089      40.0%
Loss: 0.6230827400881452      50.0%
Loss: 0.6157065622831859      60.0%
Loss: 0.6086855316895313      70.0%
Loss: 0.6020003798552925      80.0%
Loss: 0.5956327355164103      90.0%
loss: 0.5895658866765062     100.0%
Training finished in 0:00:00.784118
```

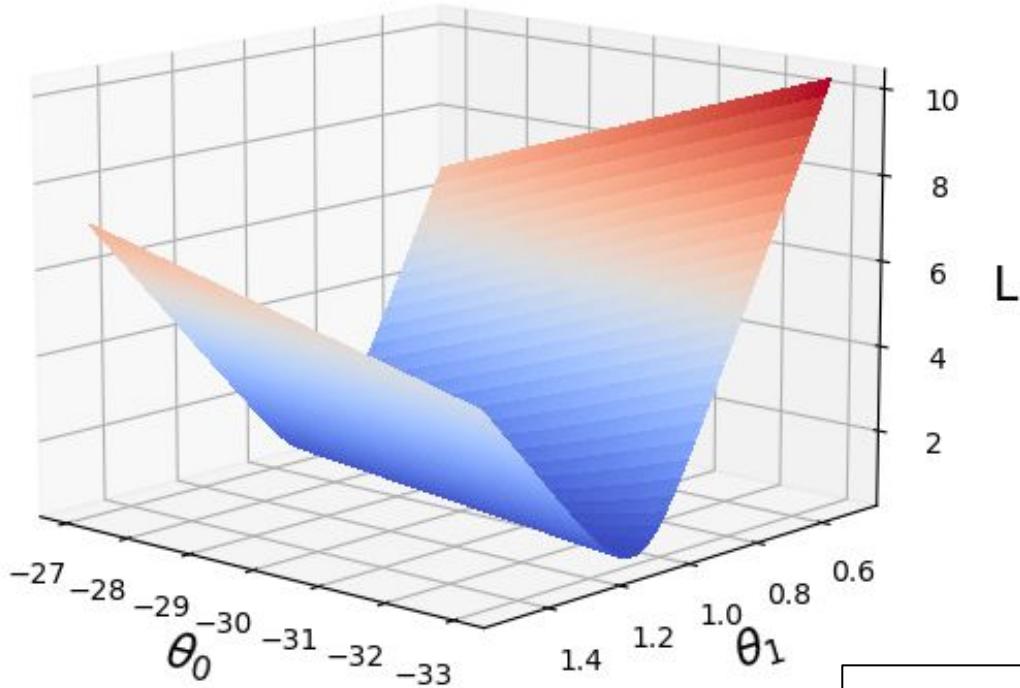
sklearn.linear\_model.LogisticRegression  
(with SAG solver)

```
Start training...
Training finished in 0:00:00.007022 (#iterations: 5820)
```

$$\theta = \begin{bmatrix} \theta_0 \\ \theta_1 \end{bmatrix} = \begin{bmatrix} -5.44 \\ 0.19 \end{bmatrix}$$

$$\theta = \begin{bmatrix} \theta_0 \\ \theta_1 \end{bmatrix} = \begin{bmatrix} -5.45 \\ 0.19 \end{bmatrix}$$

# Logistic Regression in Practice (CSI Example)



Region of "near-plateau":

- Gradient  $\nabla_{\theta} L$  very small
- Step  $\eta \nabla_{\theta} L$  extremely small
- Very slow convergence

**Note:** The Cross-Entropy loss of Logistic Regression is convex  
→ There always exists exactly one minimum (global minimum)!

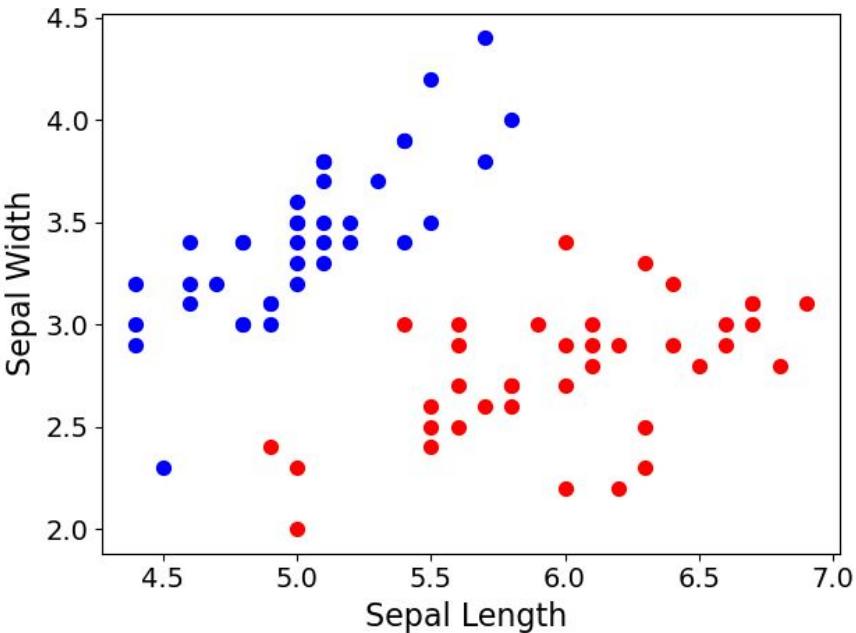
# Logistic Regression in Practice

- Logistic Regression with vanilla Gradient Descent (also Linear Regression)
  - Works in principle — the math is sound!
  - Often poor performance compared to more sophisticated implementations
- Many techniques to boost performance
  - Smart(er) initialization of  $\theta$
  - Adaptive learning rates
  - Extensions to Gradient Descent
  - Regularization
  - ...and others

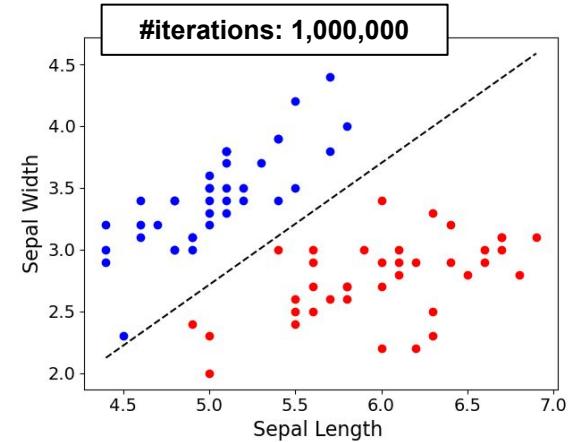
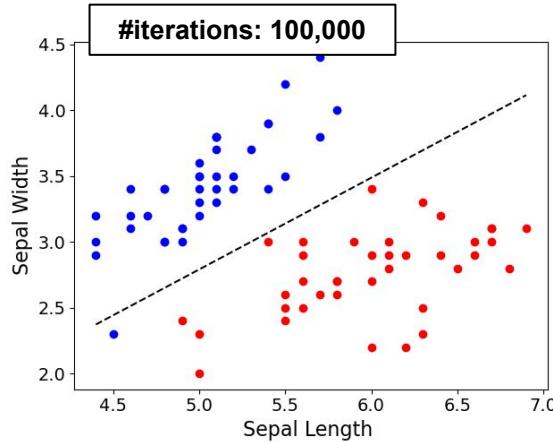
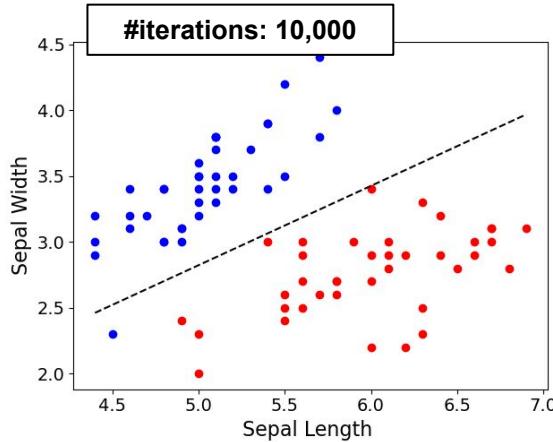
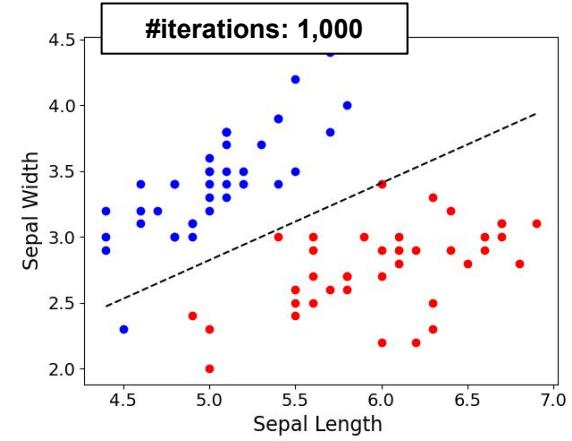
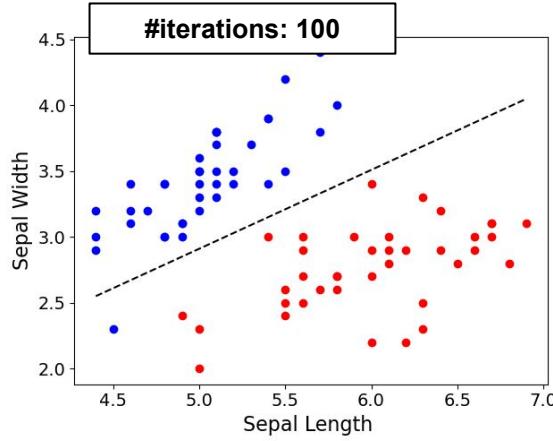
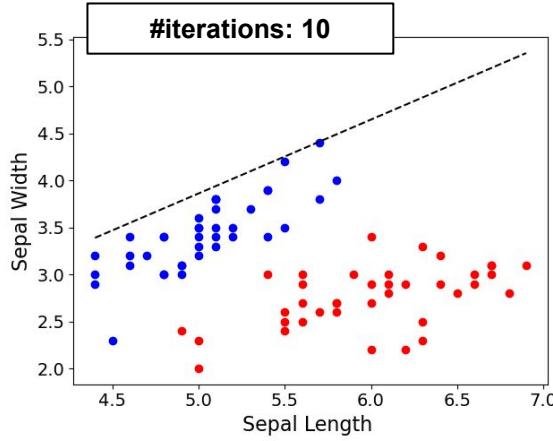
# Logistic Regression — 2D Example

- IRIS dataset

- 3 classes of Iris plants  
(only 2 considered)
- 50 samples per class
- 4 continuous features  
(only sepal length/width considered)



# Logistic Regression — 2D Example (Vanilla Gradient Descent)



# Polynomial Logistic Regression

- Analogous to Polynomial Linear Regression
  - Allows to capture nonlinear relationships between  $X$  and  $y$
  - Polynomial Logistic Regression model for 1 input feature

$$\hat{y}_i = \frac{1}{1 + e^{-\theta^T x_i}}, \quad \text{with } \theta^T x_i = \theta_0 + \theta_1 x_i + \theta_2 x_i^2 + \dots + \theta_p x_i^p$$

- Identical practical considerations

- Solve using Gradient Descent as usual  
(optionally with regularization to avoid overfitting)
- Limited to small number of features  
and small polynomial degrees

$$\nabla_{\theta} L = \frac{1}{n} X^T (h_{\theta}(X) - y) + \lambda \frac{2}{n} \theta$$

# Overview

- **Linear Models**
  - Basic setup
- **Linear Regression**
  - Problem formulation
  - Normal Equation (analytical solution)
  - Gradient Descent (iterative optimization)
  - Polynomial Linear Regression (overfitting, regularization)
  - Interpretation of Coefficients
- **Logistic Regression**
  - Problem formulation
  - Gradient Descent

# Summary

- **Linear Models**
  - Assume linear relationship between input features and output  
(i.e., output = sum of weighted feature values)
  - However, regression line / decision boundary not always a line/plane/hyperplane  
(data transformation to add polynomial terms based on input features)
  - Generally good interpretability
- **Linear Regression & Logistic Regression**
  - Very fundamental and popular regression and classification methods
  - Gradient Descent to solve both tasks + Normal Equation to solve Linear Regression
  - Effects of data normalization mainly(!) on explainability / interpretability of results
  - Straightforward extension of Logistic Regression beyond 2 classes (not covered here)

# Solutions to Quick Quizzes

- Slide 13: Yellow (smallest average error)
- Slide 22: D
- Slide 23: A
- Slide 35: "near plateau" → very small gradients and updates
- Slide 48: B
- Slide 57: C

# CS5228: Knowledge Discovery and Data Mining

## Lecture 8 — Recommender Systems

# Course Logistics

- **Deadline Reminders**
  - Submission of A3: Thu Oct 24, 11.59 pm – answer please in English only :)
  - Submission of project report: Nov 14, 11.59 pm
  - Extended TEAMMATES deadline: Sat, Oct 19, 11.59pm
- **Marks upload**
  - A2 + midterm result soon ready

# Quick Recap — Linear Models

- Basic Assumption

- Linear relationship between  $x_i$  and dependent variable  $y_i$

$$\hat{y}_i = h_{\theta}(x_i) = f(\underbrace{\theta_0 x_{i0} + \theta_1 x_{i1} + \theta_2 x_{i2} + \dots + \theta_d x_{id}}_{= 1})$$

Predicted value which is hopefully close to  $y_i$

1, 2, ...,  $d$  input features

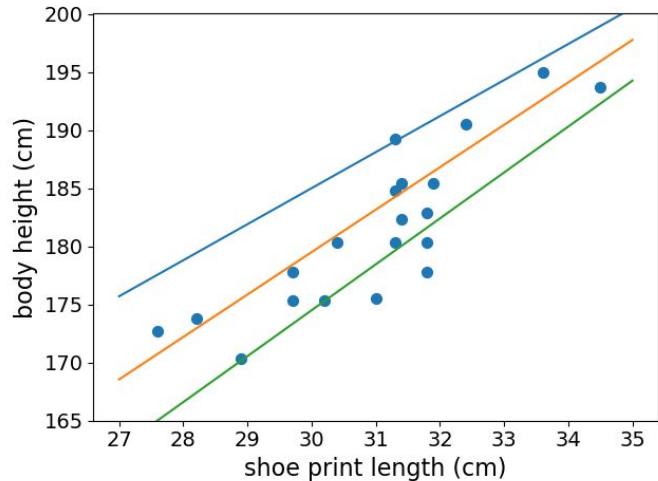
The diagram illustrates the linear model equation  $\hat{y}_i = h_{\theta}(x_i) = f(\theta_0 x_{i0} + \theta_1 x_{i1} + \theta_2 x_{i2} + \dots + \theta_d x_{id})$ . A box labeled "Predicted value which is hopefully close to  $y_i$ " has an upward-pointing arrow from it to the equation. Another box labeled "1, 2, ...,  $d$  input features" has three arrows pointing to the first term ( $\theta_0 x_{i0}$ ), the last term ( $\theta_d x_{id}$ ), and the constant term (= 1).

- Learned parameters of model:  $\theta = \{\theta_0, \theta_1, \theta_2, \dots, \theta_d\}$ ,  $\theta_i \in \mathbb{R}$

# Quick Recap — Linear Regression

- Find  $\theta$  that minimizes MSE loss  $L$

$$\begin{aligned} L &= \frac{1}{n} \sum_{i=1}^n (\hat{y}_i - y_i)^2 \\ &= \frac{1}{n} \|X\theta - y\|^2 \end{aligned}$$



- Solve using Normal Equation

$$\frac{\partial L}{\partial \theta} = \frac{2}{n} X^T (X\theta - y) \rightarrow \frac{2}{n} X^T (X\theta - y) \stackrel{!}{=} \vec{0} \rightarrow \theta = (X^T X)^{-1} X^T y$$

- Solve using Gradient Descent

$$\nabla_{\theta} L = \frac{2}{n} X^T (X\theta - y) \rightarrow \text{repeat: } \theta \leftarrow \theta - (\eta \cdot \nabla_{\theta} L)$$

# Quick Recap — Logistic Regression

- Regression model for classification
  - Interpret  $\hat{y}$  as probability that  $x$  belongs to Class 1

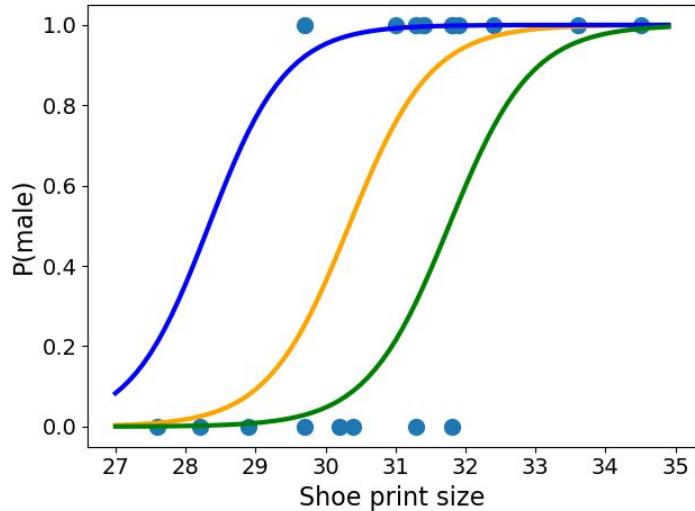
$$\hat{y} = h_{\theta}(x) = f(\theta^T x) = \frac{1}{1 + e^{-\theta^T x}}$$

- Minimize **Cross-Entropy Loss L**

$$L = -[y \log \hat{y} + (1 - y) \log (1 - \hat{y})]$$

- Solve using **Gradient Descent**

$$\nabla_{\theta} L = \frac{1}{n} X^T (h_{\theta}(X) - y) \quad \rightarrow \quad \text{repeat: } \theta \leftarrow \theta - (\eta \cdot \nabla_{\theta} L)$$



# Quick Recap — Linear Models

- **Polynomial Linear/Logistic Regression**

- Data transformation to include polynomial terms of features
- No change of algorithms needed

$$X^{(1)} = \begin{bmatrix} 1 & x_1 \\ 1 & x_2 \\ \vdots & \vdots \\ 1 & x_n \end{bmatrix} \quad X^{(2)} = \begin{bmatrix} 1 & x_1 & x_1^2 \\ 1 & x_2 & x_2^2 \\ \vdots & \vdots & \vdots \\ 1 & x_n & x_n^2 \end{bmatrix} \quad X^{(3)} = \begin{bmatrix} 1 & x_1 & x_1^2 & x_1^3 \\ 1 & x_2 & x_2^2 & x_2^3 \\ \vdots & \vdots & \vdots & \vdots \\ 1 & x_n & x_n^2 & x_n^3 \end{bmatrix}$$

- **Regularization to avoid overfitting**

- Extend loss function to "punish" large values of  $\theta$
- Only minor changes to Normal Equation and calculation of Gradient

$$L = \frac{1}{n} \|X\theta - y\|^2 + \lambda \frac{1}{n} \|\theta\|_2^2$$

# Outline

- **Overview**
  - Motivation
  - Naive / alternative approaches
- **Content-based recommendation systems**
  - Pairwise item-item similarity
  - User-item similarity
- **Collaborative filtering (CF)**
  - Memory-based CF
  - Model-based CF

# Recommender Systems — Motivation

- In the online world: information and item overload
  - Too many items: products, songs, movies, news articles, restaurants, etc.
  - More choices require better filters → recommendation engines

User perspective	Provider perspective
<ul style="list-style-type: none"><li>• Identify relevant items</li><li>• Minimize effort (to find relevant items)</li><li>• Maximize satisfaction</li><li>• Optimize spending of money and attention</li></ul>	<ul style="list-style-type: none"><li>• Maximize sales / transactions</li><li>• Maximize user engagement (e.g., to maximize ad revenue)</li><li>• Gain competitive advantage</li></ul>

# Editorial Recommendations

- **Recommendations by "experts"**
  - Expert = person with expertise about item(s)  
(e.g., movie or restaurant critic, staff writers, journalists)
  - Objective, elaborate, trustworthy, credible  
(at least in an ideal world...)
  - Writing editorial recommendations is generally paid work

A Foodie's Guide to the Best Burgers in Singapore

**The best burgers in Singapore**

15 Best Burgers In Singapore So Good, You Won't Stop At Just Bun

Burgers, burgers, burgers: The best to sink your teeth into whether dine-in or delivered

Best Burger Joints You Must Try in Singapore

**Snackdown review: The best burgers in Singapore**

**The best restaurants with burger delivery options in Singapore**

# Peer Recommendations

- **Recommendations by normal users**

- Online word-of-mouth recommendations  
(however, users are typically strangers)
- Common feature on shopping/booking sites
- Typically subjective, short, biased
- Many reviews per item create average view  
but represents again information overload

● ● ● ○○ Reviewed 11 November 2016 □ via mobile

## One of the best burgers in town

Food n drink above average. Service has much room for improvement. Ordered the beef burger which is huge n delicious. However the server failed to check for the preference on the done-ness of the patty.

### VALUE FOR MONEY

Created on 08/22/2020



“ basic bike, not that suitable for actual MTB but can't beat the value for easy bike riding ”



by Anand ✓ Verified Purchase

28 May 2020

The product looks good and very difficult to tilt vertically otherwise it's a worth the money. Due to Circuit Breaker, the shipping took a Long time.

### 5/5 Excellent

#### Verified traveller

Travelled with family, Travelled with group

25 Oct 2019

☺ Liked: Cleanliness, staff & service, amenities, property conditions & facilities

It was too crowded and busy hotel. Otherwise everything was good

### ★★★★★ A dream come true.

Reviewed in the United States on August 1, 2019

#### Verified Purchase

A lot of us thought we would never see all these characters in one movie, but these guys did it. And they gave us some of the best movies we know now. What a spectacular journey it was, starting with Iron Man and now here. Very well done everyone who was apart of it, and thank you.

# Manual Recommendations — Pros & Cons

- Pros
  - Semantically rich (ratings, plain text, images, videos, etc.)
  - Explainability / Interpretability
- Cons
  - Manual effort — What is the incentive for writing a review?
  - Lack of personalization

# Recommendation Fraud

Online reviews 'used as blackmail'

'Why I write fake online reviews'

Spotify tests sponsorship of full-screen album recommendations

A new study analyses the murky world of fake Amazon reviews

How merchants use Facebook to flood Amazon with fake reviews

Army of fake reviewers being built to dupe buyers, drive online sales

Influencer Marketing Fraud: The Shady Side of Social Media

Can We Trust Social Media Influencers?

## Buy Lazada Review - Votes - Sells

### How to Get Paid to Write Reviews

Amazon's Fake Review Problem Is Getting Worse

Threatening a business with a bad review is ugly bullying

Amazon is filled with fake reviews and it's getting harder to spot them

# Quick Quiz

How many Amazon reviews for electronic products are "**fake**"?

(according to a study from 2018)

**A**

~10%

**B**

~30%

**C**

~60%

**D**

~90%

# Outline

- **Overview**
  - Motivation
  - **Naive / alternative approaches**
- **Content-based recommendation systems**
  - Pairwise item-item similarity
  - User-item similarity
- **Collaborative filtering (CF)**
  - Memory-based CF
  - Model-based CF

# Recommendations Simple Aggregations

- Rank items based on aggregated scores

**Rotten Tomatoes Top-100 Movies**

**BEST OF RT**

Movies with 40 or more critic reviews vie for their place in history at Rotten Tomatoes. Eligible movies are ranked based on their Adjusted Scores.

Rank	Rating	Title	No. of Reviews
1.	96%	Black Panther (2018)	516
2.	94%	Avengers: Endgame (2019)	531
3.	93%	Us (2019)	536
4.	97%	Toy Story 4 (2019)	445
5.	99%	Lady Bird (2017)	394
6.	100%	Citizen Kane (1941)	94
7.	97%	Mission: Impossible - Fallout (2018)	430
8.	98%	The Wizard of Oz (1939)	120
9.	96%	The Irishman (2019)	441
10.	96%	BlacKkKlansman (2018)	438

**IMDB Top-250 Movies**

Top Rated Movies  
Top 250 as rated by IMDb Users

SHARE

Rank & Title	IMDb Rating	Your Rating	Add
1. The Shawshank Redemption (1994)	★ 9.2	☆	+
2. The Godfather (1972)	★ 9.1	☆	+
3. The Godfather: Part II (1974)	★ 9.0	☆	+
4. The Dark Knight (2008)	★ 9.0	☆	+
5. 12 Angry Men (1957)	★ 8.9	☆	+
6. Schindler's List (1993)	★ 8.9	☆	+
7. The Lord of the Rings: The Return of the King (2003)	★ 8.9	☆	+
8. Pulp Fiction (1994)	★ 8.8	☆	+
9. The Good, the Bad and the Ugly (1966)	★ 8.8	☆	+
10. The Lord of the Rings: The Fellowship of the Ring (2001)	★ 8.8	☆	+

# Simple Aggregations — Pros & Cons

- Pros

- Relatively easy to compute (typically weighted aggregated based on different factors)
- Typically good/safe recommendations (particularly for new/unknown users)

- Cons

- Requires sufficient number of ratings per items
- High risk of popularity bias; lack of diversity  
("rich get richer" effects, "few get richer" effects)
- Lack of personalization

# Personalized Recommendations

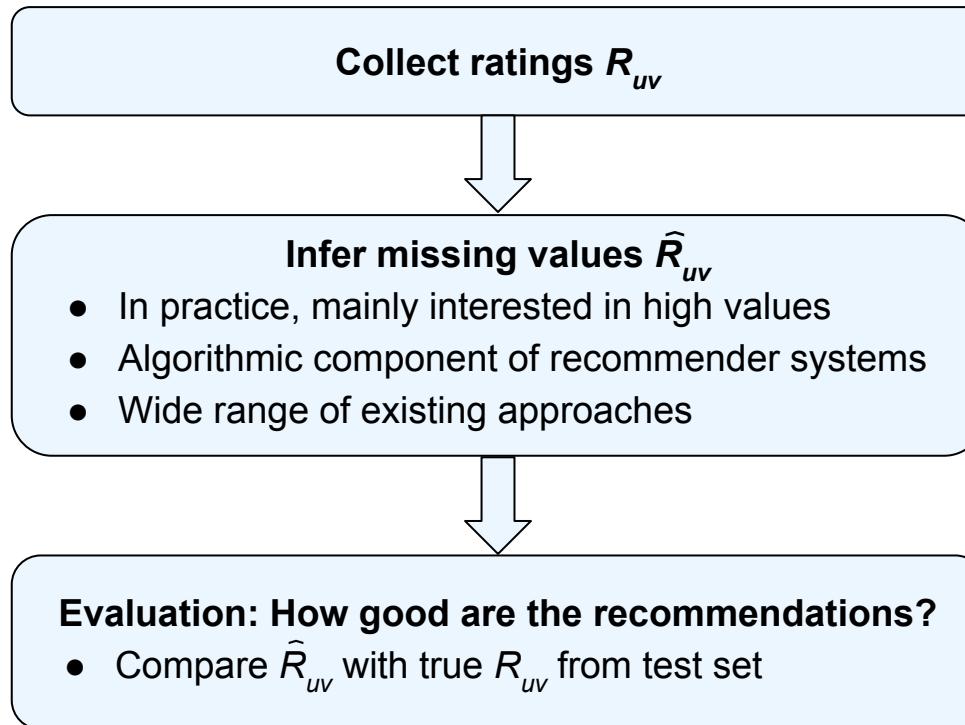
- Users have different preferences that define the relevance of items
  - Preferences = interests, tastes, likings, needs, wants, desires, etc.
  - Relevant items = items that match users' preferences best

- Basic setup

- Set of users  $U = \{u_1, u_2, \dots, u_n\}$
- Set of items  $V = \{v_1, v_2, \dots, v_m\}$
- Rating matrix  $R$  with  
 $|U|$  rows and  $|V|$  columns
- Matrix element  $R_{uv}$ :  $u$ 's rating of  $v$   
(e.g., 1-5 stars, binary 0/1)

	movies						
users		2	5		1		
		4		5	5		
		4	2				
	2		3		2		3
					4		
	5		5			2	
						4	3
	3			1			
			2		2		
		5		1			4

# Personalized Recommendations — Core Tasks



# Collecting Ratings

- **Explicit**
  - Ask/invite/encourage users to rate items
  - Pay users to rate items (e.g., crowdsourcing)

- **Implicit — derive ratings from users' behavior, e.g.:**

- Product bought
- Video watched
- Article read
- Link clicked
- ...



→ High ratings

(But how to get low ratings?)

**Key challenge: Rating matrix R  
is in practice very sparse!**

# Evaluation

- Split R into training and test set
- Performance metrics
  - Root Mean Squared Error (for numerical ratings)

$$\sqrt{\frac{1}{|S|} \sum_{(u,v) \in S} (\hat{R}_{uv} - R_{uv})^2}$$

Set of  $(u, v)$  pairs in test set

- Precision, Recall, F1 score, etc.  
(TP, TN, FP, FN for binary ratings or binary recommendation after converting numerical ratings)
- Precision@k, Recall@k  
(precision and recall w.r.t. to the top-k highest predicted ratings)
- Compare rankings induced by  $\hat{R}_{uv}$  and  $R_{uv}$  with  $(u, v) \in S$   
(also consider the order of the top-k highest ratings)

	2	5		1		
		4		5	5	
		4	2			
2		3			2	3
				4		
5		5				2
				4	3	
3				1		
		2		2		
		5		1		4

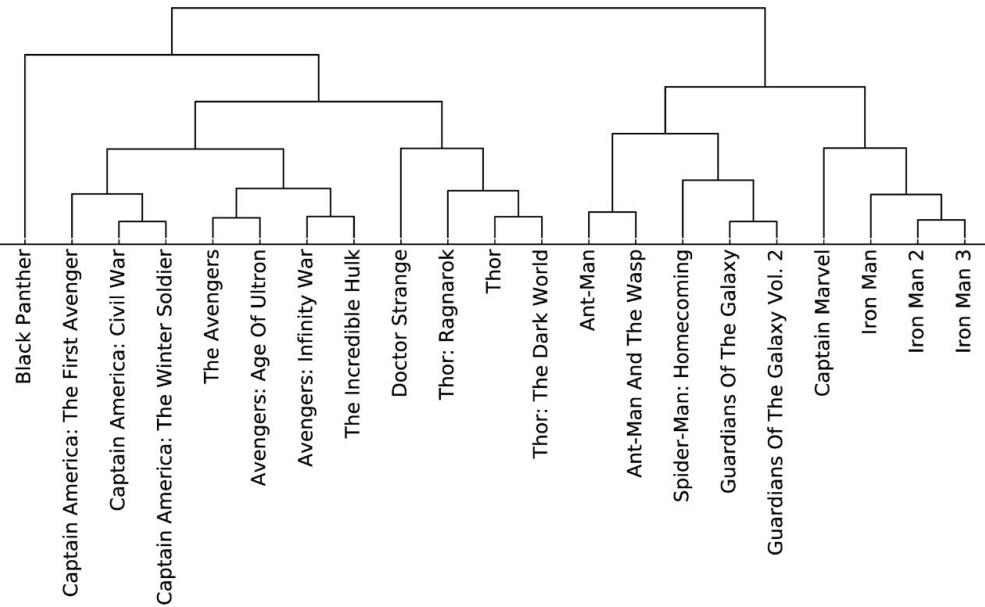
# Recommendations Using Association Rules

- User preferences & likings
  - Items: movies, songs, books, etc.
  - Transaction: viewing/listening/reading history
- Interesting rules (movies):
  - Viewer who watched movies {a, b} also watched movies {x, y}
  - Example: {Jaws} → {It}
- Limitations
  - Basic AR algorithm ignores ratings
  - Popularity bias: user with very unique tastes likely to get subpar recommendations

TID	Items
1	Jaws, Halloween, Scream, It
2	Alien, Jaws, Scream, It
3	Tenet, Inception, Interstellar
4	Jaws, Halloween, It
5	Alien, Tenet, Jaws, It
...	

# Recommendations Using Clustering

Example: Hierarchical clustering of MCU movies



## ● Approach

- Cluster movies based on "useful" features (genre, director, writer, length, ...)
- Recommend movies from clusters with movies a user has rated highly

## ● Limitations

- Find good feature in practice very difficult (we come back to that)
- Unsystematic: no well-defined process to pick recommendations

# Recommendations Using Regression (or Classification)

- Example approach: Linear Regression

- Independent variable: movie features
- Dependent variable: user rating

→ Build a linear regression model for each users

- Limitations

- Requires good features for each item
- Cold-start problem: requires a lot of user ratings to build a good model

Rated movies of an individual user

original	powerful	absorbing	comical	romantic	...	Rating
0.20	0.95	0.80	0.00	0.10	...	4.5
0.80	0.25	0.50	0.50	0.75	...	4.0
0.05	0.2	0.20	0.95	0.90	...	2.0
0.60	0.20	0.80	0.00	0.85	..	3.5
0.90	0.95	0.90	0.10	0.40	..	5.0
0.45	0.50	0.20	0.60	0.30	..	2.0
0.10	0.40	0.55	0.90	0.30	..	2.5
0.75	0.50	0.50	0.40	0.40	..	3.5
0.80	0.80	0.85	0.10	0.10	...	4.5
...	...	...	...	...	...	...

# Outline

- Overview
  - Motivation
  - Naive / alternative approaches
- Content-based recommendation systems
  - Pairwise item-item similarity
  - User-item similarity
- Collaborative filtering (CF)
  - Memory-based CF
  - Model-based CF

# Content-Based Recommender System

- Intuition
  - Recommend item  $v$  to user  $u$  that are similar to  $v$  and  $u$  has rated highly
  - Examples: movies of the same genre, songs from the same artist, articles about the same topic, products with similar features, etc.
- Basic requirement: **item profiles** = feature vector for each item, e.g.:
  - Movie: genre, director, writer, cast, length, year, ...
  - Product: type, brand, price, weight, color, ...
  - Article: set of (important) words / tf-idf vector / ...

# Running Example

- MovieLens dataset
  - Items: Movies of different genres
  - Features: 20 genres (incl. "uncategorized")
  - Ratings: 1-5 stars (incl. half stars)
- Numbers for "Small" dataset
  - 610 users, 9,742 movies
  - ~100k ratings → sparsity: ~1.7%

	comedy	action	romance	drama	fantasy	thriller	...
<i>Clueless</i>	1	0	1	0	0	0	...
<i>Heat</i>	0	1	0	0	0	1	...
<i>Bad Boys</i>	1	1	0	1	0	1	...
<i>Leon</i>	0	1	0	1	0	1	...
<i>Alice</i>	1	0	1	1	1	0	...
<i>Jarhead</i>	0	1	0	1	0	0	...
<i>Rocky</i>	0	0	0	1	0	0	...
<i>Big</i>	1	0	1	1	1	0	...
<i>Krull</i>	0	1	0	0	1	0	...
...	...	...	...	...	...	...	...

	$v_1$	$v_2$	$v_3$	$v_4$	$v_5$	$v_6$	$v_7$	$v_8$	...
$u_1$	3.0					4.5			...
$u_2$	3.5	5.0						2.0	...
$u_3$				3.0		5.0			...
$u_4$		4.5	2.0				2.0		...
$u_5$	3.5		2.5						...
$u_6$					3.0			3.0	...
$u_7$					3.5	3.0			...
$u_8$			2.0				3.0	3.0	...
$u_9$	5.0				4.5				...
...	...	...	...	...	...	...	...	...	...

Source: [MovieLens Dataset](#)

# Simple Approach — Pairwise Item Similarity

- **Pairwise item similarity  $sim(x,y)$**

- $x, y$  — feature vectors of movies
- Common metric: cosine similarity

$$sim(x, y) = \cos(\theta) = \frac{x \cdot y}{\|x\| \|y\|}$$

- **Limitation: Requires reference item, e.g.:**

- Movie(s) the user was most recently watching
- Movie(s) the user has rated the highest
- Movie(s) the user is currently browsing

$sim(Heat, Heat) = 1.0$

$sim(Heat, Clueless) = 0.0$

$sim(Heat, Bad Boys) = 0.77$

$sim(Heat, Jarhead) = 0.33$

$sim(Heat, Alice) = 0.0$

Similar titles you might also like [What is this?](#)

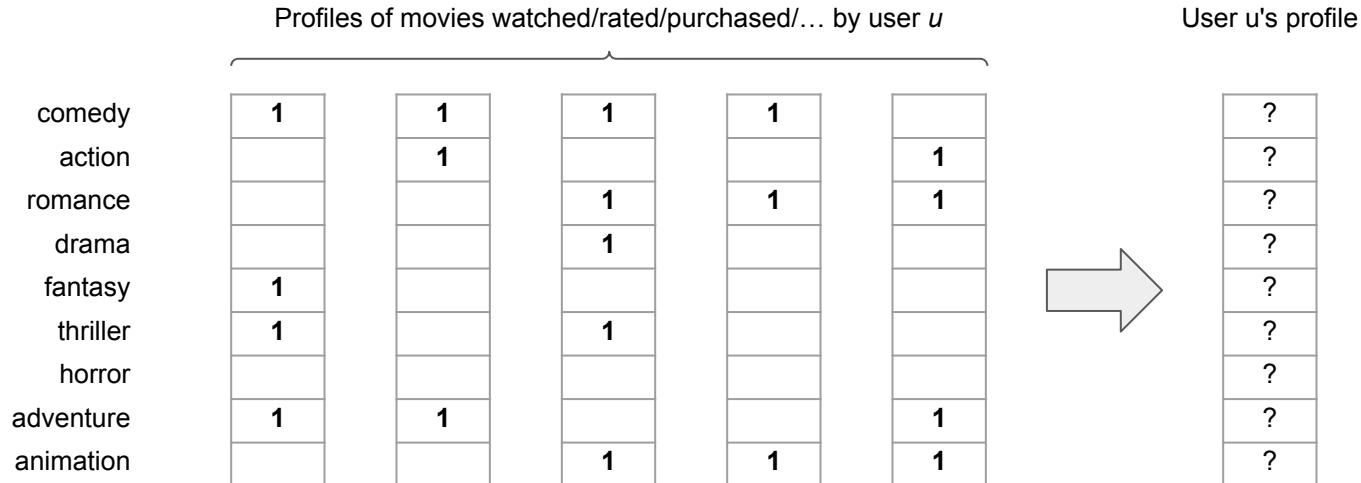


# Outline

- Overview
  - Motivation
  - Naive / alternative approaches
- Content-based recommendation systems
  - Pairwise item-item similarity
  - User-item similarity
- Collaborative filtering (CF)
  - Memory-based CF
  - Model-based CF

# User-Item Similarities

- Needed: **user profiles** = feature vector for each user
  - Requirement: same shape as item profiles to calculate similarities
  - Approach: user profile = "some aggregate" over item profiles rated by the user



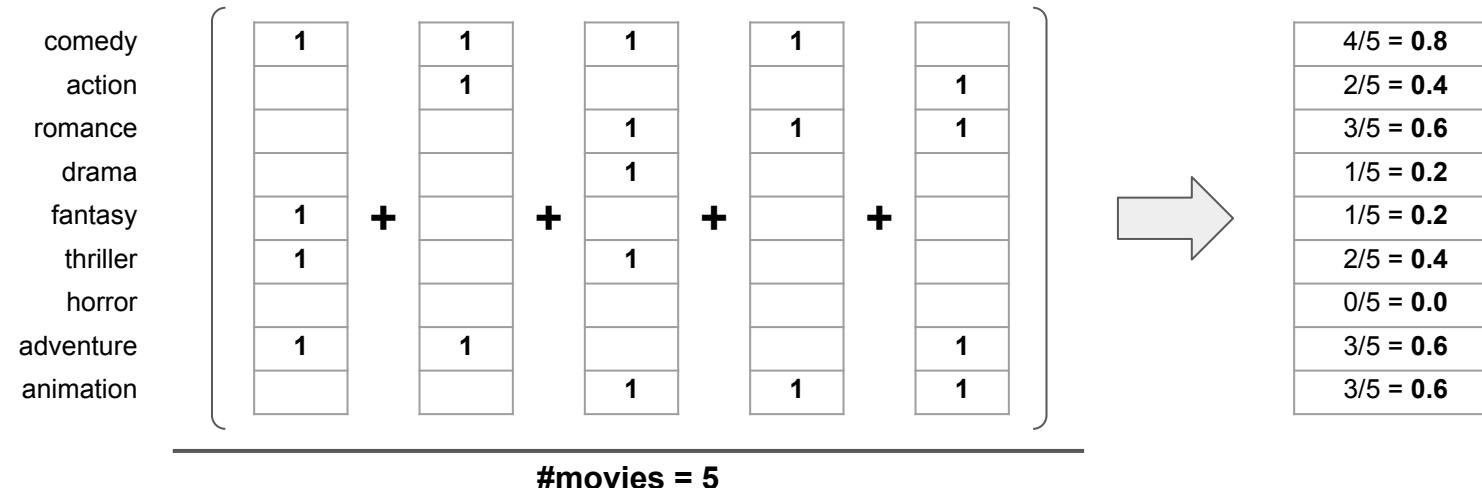
# User-Item Similarities — Binary Utility Matrix

- $R_{uv} \in \{0, 1\}$  — for example,  $R_{uv} = 1$  if

- User u bought movie v
- User u watched movie v

} Implicit rating that  $u$  liked  $v$  (no explicit ratings available here; but also no implicit dislikes!)

- Simple Average



# User-Item Similarities — Real-Valued Utility Matrix

- $R_{uv} \in \mathbb{R}$ — for example,  $R_{uv} \in \{1.0, 1.5, 2.0, 2.5, \dots, 5.0\}$  star rating
  - Explicit rating of user  $u$  for movie  $v$
  - Important: semantic interpretation — ratings express both likes and dislikes (despite all ratings positive)
  - Use rating as weights for features for a weighted aggregation

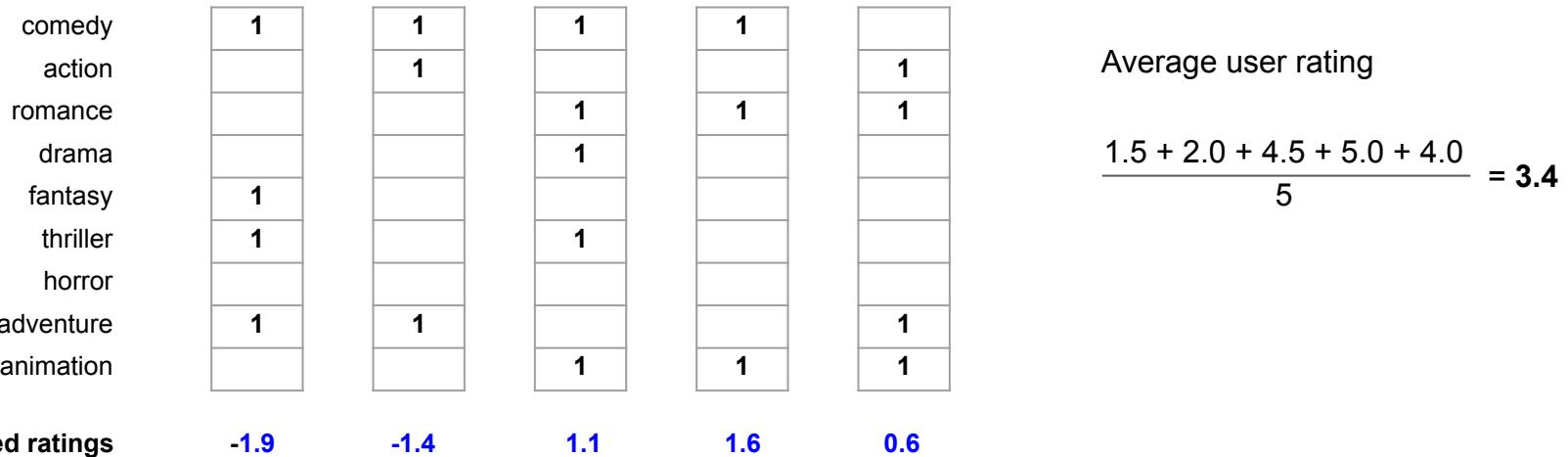
comedy	1	1	1	1	
action		1			1
romance			1		1
drama			1		
fantasy	1				
thriller	1				
horror			1		
adventure	1	1		1	
animation			1		1
<b>u's ratings</b>	<b>1.5</b>	<b>2.0</b>	<b>4.5</b>	<b>5.0</b>	<b>4.0</b>

## Intuition

- The user likes romantic and animated movies
- The user dislikes fantasy and adventure movies

# User-Item Similarities — Real-Valued Utility Matrix

- Step 1: Normalize ratings
  - Subtract average user rating from each movie rating
  - Converts ratings into positive (liked) and negative (disliked) scale
  - Distinguishes "generous" users (mostly rate highly and a 3.0 is a low rating) from more "grumpy" users (mostly rate low and a 3.0 is a high rating)



# User-Item Similarities — Real-Valued Utility Matrix

- Step 2: Calculate weighted features for user profile
    - The weights are the normalized weights

The diagram shows the calculation of weighted average ratings for User u's profile. It starts with a list of movies and their normalized ratings:

	...	...	...	...	...
adventure	1	1	1	1	1
animation					
<b>u's normalized ratings</b>	<b>-1.9</b>	<b>-1.4</b>	<b>1.1</b>	<b>1.6</b>	<b>0.6</b>

Below these, two weighted average calculations are shown:

$$w_{adventure} = \frac{1 \cdot (-1.9) + 1 \cdot (-1.4) + 1 \cdot 0.6}{3} = -0.9$$

$$w_{animation} = \frac{1 \cdot 1.1 + 1 \cdot 1.6 + 1 \cdot 0.6}{3} = 1.1$$

To the right, a large arrow points from the user's ratings to a vertical column labeled "User u's profile" containing the following values:

-0.15	comedy
-0.40	action
1.10	romance
1.10	drama
-1.90	fantasy
-0.40	thriller
0.00	horror
-0.90	adventure
1.10	animation

# Quick Quiz

Given the example calculation above what should we **ensure** to get good profiles?

$$w_{adventure} = \frac{1 \cdot (-1.9) + 1 \cdot (-1.4) + 1 \cdot 0.6}{3} = -0.9$$

**A**

A user has given mostly high ratings to different movies

**B**

Enough movies of the same genre have been rated

**C**

A user has rated enough movies of the same genre

**D**

A user's ratings for the same genre need to be diverse

# User-Item Similarities

- Pairwise item similarity  $sim(u, v)$

- $u$  — user profile;  $v$  — item profile
- Suitable metric: cosine similarity (note that user and item profiles can have different magnitudes)

→ Recommend items  $v_i$  to user  $u$  with max. similarities  $sim(u, v_i)$

- Practical considerations

- Top  $k$  most similar items always the same → add some randomization for diversity  
(the set of top  $k$  most similar items might only change over time if the user rates more items)
- Top  $k$  most similar items might include items the user has already rated → remove those items  
(in practice, recommending known items not uncommon — e.g., YouTube recommendations)
- More sophisticated ways to aggregate item profiles to user profiles conceivable  
(for example: ignore underrepresented features, e.g., if a user rated only one comedy movie)

# Content-Based Recommender System — Pros & Cons

- **Pros**

- Recommendations for user  $u$  do not depend on other users  
(this also allows for good recommendations for users with very unique tastes)
- Recommendations can also include new or unpopular items (i.e., with no or very few ratings)
- Good explainability (features that had most effect on the high similarity)

- **Cons**

- Cold-start problem: How to build a profile for new users?  
(naive approach: recommend generally popular items to new users)
- Finding good features (and values!) for items a non-trivial task  
(Question: Are genres a good feature set to represent movies?)
- Overspecialization: By default, no recommendations outside a user's profile  
(in practice: add some randomization into the recommendation process)

# Outline

- **Overview**
  - Motivation
  - Naive / alternative approaches
- **Content-based recommendation systems**
  - Pairwise item-item similarity
  - User-item similarity
- **Collaborative filtering (CF)**
  - **Memory-based CF**
  - Model-based CF

# Collaborative Filtering

- Idea: Utilize the opinions of others
  - Recommend items that other user with similar tastes/preferences/needs have liked
  - Does not require item or user-specific features
- Two perspectives
  - User-based — two users are similar if they rated the same items similarly
  - Item-based — two items are similar if they are equally rated by users

# User-Based CF — Calculating Similarities

- Example: movie ratings

- How much might Bob like the movie "Heat"?

	Clueless	Heat	Jarhead	Big	Rocky
Alice	2	4	5	0	1
Bob	1	???	4	0	2
Claire	1	0	4	3	0
Dave	5	1	2	0	5
Erin	1	5	3	0	3

- Intuitions given the dataset

- Alice and Bob have similar tastes, so Bob might rate "*Heat*" similar to Alice
- Claire and Bob have similar tastes, but Claire has not rated "*Heat*"
- Dave and Bob have very different tastes, so Dave's opinion about "*Heat*" shouldn't matter  
(if anything, it should be an indicator that Bob will like "*Heat*"; usually not relevant in practice)

→ How can we capture and quantify these intuitions?

# User-Based CF — Calculating Similarities

- Represent all users by their rating vectors  $v$

- Rows of rating matrix

$$r_A = (2, 4, 5, 0, 1)^T$$

$$r_B = (1, 0, 4, 0, 2)^T$$

$$r_C = (1, 0, 4, 3, 0)^T$$

$$r_D = (5, 1, 2, 0, 5)^T$$

$$r_E = (1, 5, 3, 0, 3)^T$$

Using cosine similarity 

$$\text{sim}(r_A, r_B) = 0.77$$

$$\text{sim}(r_D, r_B) = 0.67$$

Too similar to capture our intuition

	Clueless	Heat	Jarhead	Big	Rocky
Alice	2	4	5	0	1
Bob	1	???	4	0	2
Claire	1	0	4	3	0
Dave	5	1	2	0	5
Erin	1	5	3	0	3

## Problem

- Missing values (0) are treated as negative
  - All ratings are positive values
- No explicit notion of dissimilarity  
(only less or more similar)

# User-Based CF — Calculating Similarities

- Idea: Normalize rating vectors
  - Mean-centering — subtract row mean from each rating vector
  - Missing values (0) now represent the average rating
  - Bad ratings (i.e., below average) now represented by negative values

	Clueless	Heat	Jarhead	Big	Rocky
Alice	2-3 = <b>-1</b>	4-3 = <b>1</b>	5-3 = <b>2</b>	<b>0</b>	1-3 = <b>-2</b>
Bob	1-2.33 = <b>-1.33</b>	<b>???</b>	4-2.33 = <b>1.67</b>	<b>0</b>	2-2.33 = <b>-0.33</b>
Claire	1-2.67 = <b>-1.67</b>	<b>0</b>	4-2.67 = <b>1.33</b>	3-2.67 = <b>0.33</b>	<b>0</b>
Dave	5-3.25 = <b>1.75</b>	1-3.25 = <b>-2.25</b>	2-3.25 = <b>-1.25</b>	<b>0</b>	5-3.25 = <b>1.75</b>
Erin	1-3 = <b>-2</b>	5-3 = <b>2</b>	3-3 = <b>0</b>	<b>0</b>	3-3 = <b>0</b>


$$\text{sim}(r_A, r_B) = 0.78$$
$$\text{sim}(r_D, r_B) = -0.65$$

Cosine similarity between mean-centered vectors → Pearson Correlation Coefficient

# User-Based CF — Predicting Ratings

- $\hat{R}_{uv}$  = weighted average of ratings from similar users
  - N — set of  $k$  users most similar to  $u$  who have already rated item  $v$

$$\hat{R}_{uv} = \frac{\sum_{w \in N} sim(u, w) \cdot R_{wv}}{\sum_{w \in N} sim(u, w)}$$

- For the example

- With  $k = 2$

$$\hat{R}_{Bob,Heat} = \frac{0.78 \cdot 4 + 0.44 \cdot 5}{0.78 + 0.44} = 4.3$$

Alice                            Erin

	Clueless	Heat	Jarhead	Big	Rocky
Alice	2	4	5	0	1
Bob	1	???	4	0	2
Claire	1	0	4	3	0
Dave	5	1	2	0	5
Erin	1	5	3	0	3

# Item-Based CF

- Analog to user-based approach
  - For an item  $v$ , find the most similar items  
(2 items are similar, if their ratings across all users are similar)
  - $\hat{R}_{uv} = \text{weighted average of ratings of similar items}$
  - $M$  — set of  $k$  items most similar to  $v$  who have already been rated by  $u$

$$\hat{R}_{uv} = \frac{\sum_{i \in M} sim(i, v) \cdot R_{ui}}{\sum_{i \in M} sim(i, v)}$$

**Note:** Recall that in content-based recommender systems, measuring the similarity between items relied on item profiles / feature vectors. In case of item-item CF, the rating vector of an item represents its profile.

# Item-Based CF — Example

Calculate mean-centered rating vectors  
(here: columns of rating matrix)

$$v_C = (0, -1, -1, 3, -1)^T$$

$$v_H = (0.67, 0, 0, -2.33, 1.67)^T$$

$$v_J = (1.4, 0.4, 0.4, -1.6, -0.6)^T$$

$$v_B = (0, 0, 0, 0, 0)^T$$

$$v_R = (-1.75, -0.75, 0, 2.25, 0.25)^T$$

	Clueless	Heat	Jarhead	Big	Rocky
Alice	2	4	5	0	1
Bob	1	???	4	0	2
Claire	1	0	4	3	0
Dave	5	1	2	0	5
Erin	1	5	3	0	3

Calculate distances between "Heat" and all other movies Bob has already rated

$$\text{sim}(v_H, v_C) = -0.85$$

$$\text{sim}(v_H, v_J) = 0.55$$

$$\text{sim}(v_H, v_R) = -0.69$$

Even for k=2, there is only 1 movie with a positive Pearson correlation coefficient

$$\widehat{R}_{uv} = \frac{\overbrace{0.55 \cdot 4}^{\text{Jarhead}}}{0.55} = 4$$

# Collaborative Filtering — User-Based vs. Item-Based

- In theory: user-based and item-based are dual approaches
  - In practice: item-based typically outperforms user-based
    - Items are "simpler" than users
    - Items can be more easily described
    - User can have very varied tastes
- 
- Item-item similarity typically more meaningful

# Outline

- **Overview**
  - Motivation
  - Naive / alternative approaches
- **Content-based recommendation systems**
  - Pairwise item-item similarity
  - User-item similarity
- **Collaborative filtering (CF)**
  - Memory-based CF
  - Model-based CF

# Model-Based Collaborative Filtering

- Latent factor models

- Latent representation:  $k$ -dimensional vector for each user  $u$  and item  $v$
- Learn latent representations from the data  
(in contrast to content-based systems where feature vectors are constructed)
- Estimate unknown ratings  $\hat{R}_{uv} = w_u^T h_v$

- Approach: Matrix Factorization

- Put all user vectors into a matrix  $W$
- Put all item vector into a matrix  $H$

→ Find  $W, H$  such that  $R = WH$

or at least approximately

# Matrix Factorization — Basic Setup

- Given: ratings matrix  $R$

- $m$  — number of users  $|W|$
- $n$  — number of items  $|H|$

$$\left[ \begin{array}{c} R \\ m \times n \end{array} \right] = \left[ \begin{array}{c} W \\ m \times k \end{array} \right] \times \left[ \begin{array}{c} H \\ k \times n \end{array} \right]$$

- Hyperparameter  $k$

- Size of latent representations

# Finding Matrices W, H

- Minimize loss function

$$L = \sum_{R_{uv} > 0} e_{uv} = \sum_{R_{uv} > 0} (R_{uv} - \hat{R}_{uv})^2 = \sum_{R_{uv} > 0} (R_{uv} - w_u^T h_v)^2$$

→ with regularization:

$$L = \sum_{R_{uv}} (R_{uv} - w_u^T h_v)^2 + \lambda(\|w_u\|^2 + \|h_v\|^2)$$

- Using Gradient Descent

Calculate gradients

$$\frac{\partial e_{uv}}{\partial w_u} = -2(R_{uv} - w_u^T h_v)h_v + 2\lambda w_u$$

$$\frac{\partial e_{uv}}{\partial h_v} = -2(R_{uv} - w_u^T h_v)w_u + 2\lambda h_v$$



Update rules

$$w_u \leftarrow w_u - \eta \frac{\partial e_{uv}}{\partial w_u}$$

$$h_v \leftarrow h_v - \eta \frac{\partial e_{uv}}{\partial h_v}$$

# Finding Matrices $W, H$ — Algorithm

**Input** : rating matrix  $R^{(m \times n)}$ , latent vector size  $k$ , #iterations  $T$

**Initialization** :  $W^{(m \times k)}$ ,  $H^{(k \times n)}$  with values 0..1

**for** 1 **to**  $T$

**for all**  $u, v$  **with**  $R_{uv} > 0$

$$w_u \leftarrow w_u + \eta [2(R_{uv} - w_u^T h_v)h_v - 2\lambda w_u]$$

$$h_v \leftarrow h_v + \eta [2(R_{uv} - w_u^T h_v)w_u - 2\lambda h_h]$$

**return**  $W, H$

# Finding Matrices W, H — Example

	v <sub>1</sub>	v <sub>2</sub>	v <sub>3</sub>	v <sub>4</sub>	v <sub>5</sub>	v <sub>6</sub>	v <sub>7</sub>
u <sub>1</sub>	4	0	0	5	1	0	0
u <sub>2</sub>	5	5	4	0	0	0	0
u <sub>3</sub>	0	0	0	2	4	5	0
u <sub>4</sub>	0	3	0	0	0	0	3

Calculate W, H



k = 100

$\lambda = 0.1$

#iterations = 10k

W · H

	v <sub>1</sub>	v <sub>2</sub>	v <sub>3</sub>	v <sub>4</sub>	v <sub>5</sub>	v <sub>6</sub>	v <sub>7</sub>
u <sub>1</sub>	3.9	3.5	3.4	4.8	1	2.4	3.1
u <sub>2</sub>	4.9	4.8	4	3.7	3.2	5.3	4.4
u <sub>3</sub>	3.4	3.3	3.6	2	3.8	4.9	3.9
u <sub>4</sub>	3.1	2.9	3.2	2.5	2.3	3.8	3

- Effects of regularization

- Increase  $\lambda$ : worse fit of known ratings, "smoother" values for all ratings
- Decrease  $\lambda$ : better fit of known ratings, more "extreme" values of unknown ratings

# Collaborative Filtering — Pros & Cons

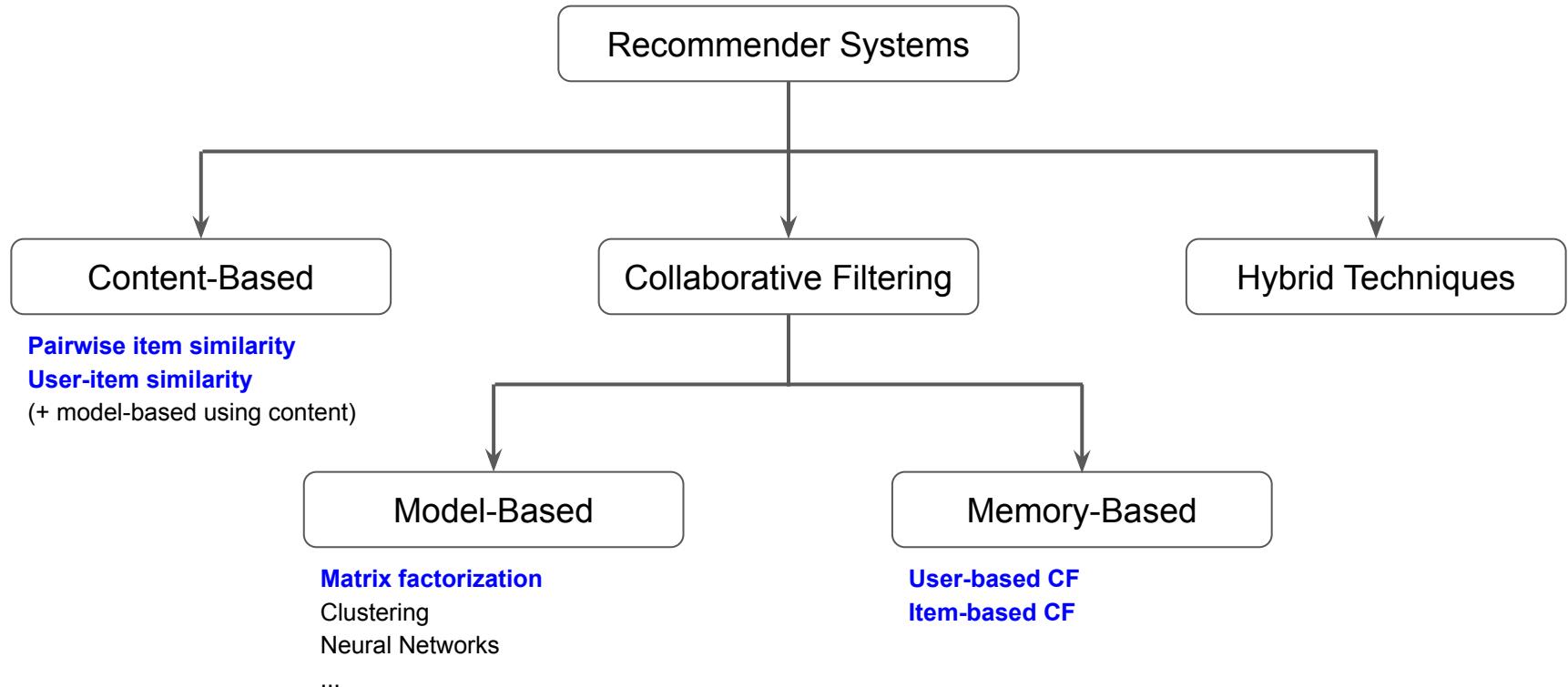
- Pros

- No need to find and create good features  
(such as genres for movies)
- Intuitive approach

- Cons

- Similarity calculations rely on sufficient number of ratings
- Cold-start problem in case of new users or items
- Popularity bias: user with very unique tastes likely to get subpar recommendations  
(because even the  $k$  most similar users will not be truly very similar)
- Naive implementation very expensive: Finding  $k$  most similar users/items  $\in O(|R|)$   
(optimization techniques needed: e.g. clustering of users/items to limit search space)

# Recommender Systems — Summary



# Quick Quiz

What does **not** affect the recommendations made by Collaborative Filtering?

**A**

A new movie is added

**B**

A new user is added

**C**

A user rates a movie

**D**

A movie's details get updated

# Quick Quiz

What does generally affect the recommendations made by Collaborative Filtering the **least**?

**A**

An old user has rated an old movie

**B**

An old user has rated a new movie

**C**

A new user has rated an old movie

**D**

A new user has rated a new movie

# The Dangers of (Over-)Personalization

- 2 infamous side-effects

(particularly when recommending news or social media posts)

- Filter bubbles
- Echo chambers

- Core problems

- No incentive for service providers to ensure (sufficient) diversity
- Users do not know what content is shown and why (or why not!)

**Why is TikTok creating filter bubbles based on your race?**

**How social media filter bubbles and algorithms influence the election**

**When Algorithms Decide Whose Voices Will Be Heard**

*Social Media Giants Support Racial Justice. Their Products Undermine It.*

**Facebook reportedly ignored its own research showing algorithms divided users**

# Outline

- Overview
  - Motivation
  - Naive / alternative approaches
- Content-based recommendation systems
  - Pairwise item-item similarity
  - User-item similarity
- Collaborative filtering (CF)
  - Memory-based CF
  - Model-based CF

# Summary

- **Recommender systems**
  - More specifically: personalized recommender systems
  - Integral component of many online platforms
  - User: find relevant items + providers: present relevant items
  - BUT: risk of over-personalized recommendations
- **Implementing recommender systems**
  - Wide range of data mining techniques applicable
  - No "one-size-fits-all" solutions
  - In practice, hybrid approaches most successful

# Solutions to Quick Quizzes

- Slide 13: C
- Slide 34: C
- Slide 54: D
- Slide 55: A

# **CS5228: Knowledge Discovery and Data Mining**

## Lecture 9 — Graph Mining

# Course Logistics

- **Deadline Reminders**
  - Submission of A4: Nov 14, 11.59 pm
  - Submission of project report: Nov 14, 11.59 pm
- **Project**
  - Check your TEAMMATES result

# Quick Recap

- **Recommender Systems**

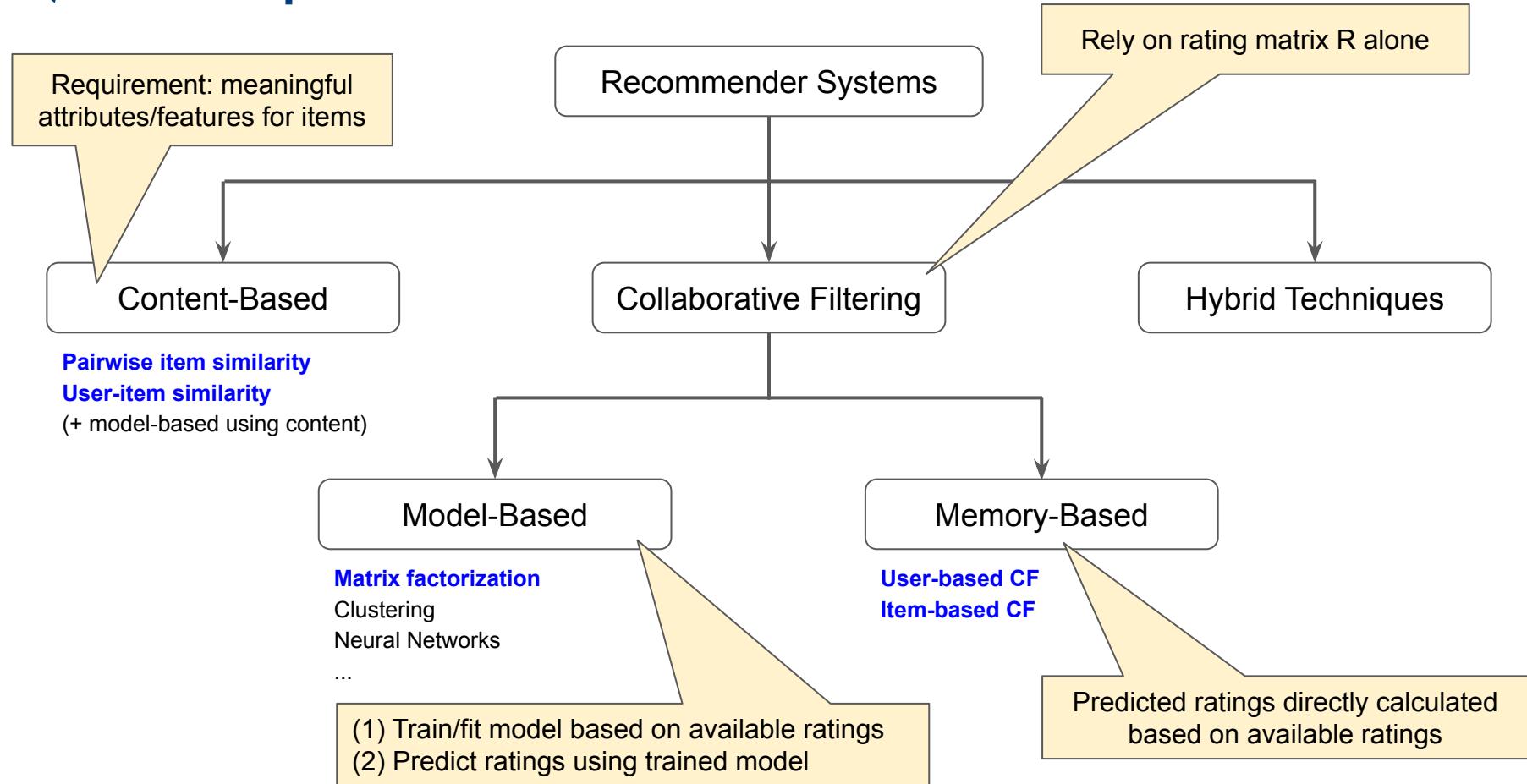
- Problem: information overload (too many items: products, songs, movies, news articles, restaurants, etc.)
- Goal: identify relevant items matching a user's preferences

- **Basic setup**

- Set of users  $U$ , set of items  $V$
- Rating matrix  $R$  with  $|U|$  rows and  $|V|$  columns
- Matrix element  $R_{uv}$ :  $u$ 's rating of  $v$  (e.g., 1-5 stars, binary 0/1)
- If available: information (features) about each item

	items						
users		2	5		1		
			4		5	5	
			4	2			
	2		3		2		3
					4		
	5		5			2	
						4	3
	3				1		
			2		2		
			5		1		4

# Quick Recap



# Outline

- **Graph Mining**
  - Application Examples
  - Basic definitions
- **Community Detection**
  - Basic definition & goals
  - Overview to different algorithms
- **Centrality**
  - Basic definition & goals
  - Overview to different algorithms
- **Summary**

# Graphs are Everywhere

## Transportation Networks

**Nodes:** MRT stations

**Edges:** Direct train connections between stations

### Applications:

- Find shortest travels
- Find busy stations
- Plan bus routes



Source: <https://www.lta.gov.sg/>

# Graphs are Everywhere

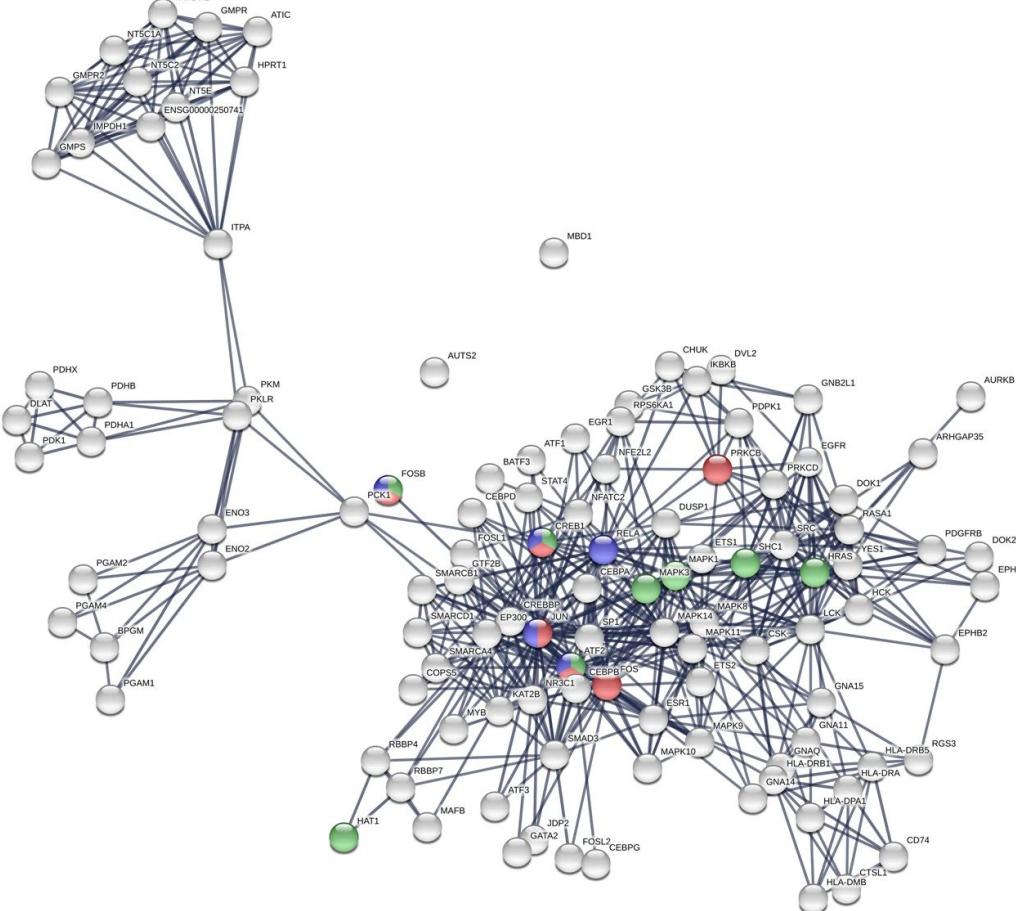
# Protein-Protein Interactions

## Nodes: Proteins

**Edges:** Physical interactions between two proteins

## **Applications:**

- Understanding of diseases  
(disease prognosis, disease susceptibility)
  - Drug discovery



# Graphs are Everywhere

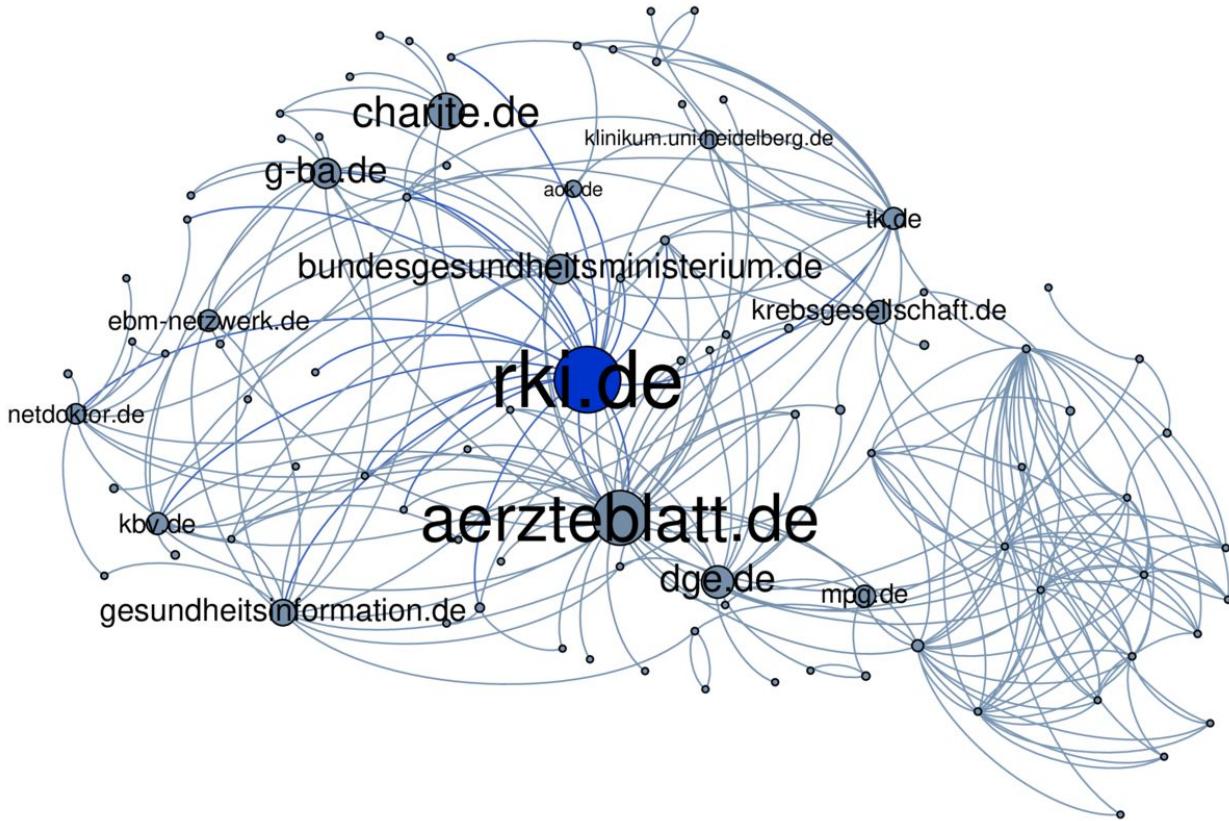
## Web Graph

**Nodes:** Web pages

**Edges:** Hyperlinks

### Applications:

- Identify authoritative sites
- Effective Web search



# Graphs are Everywhere

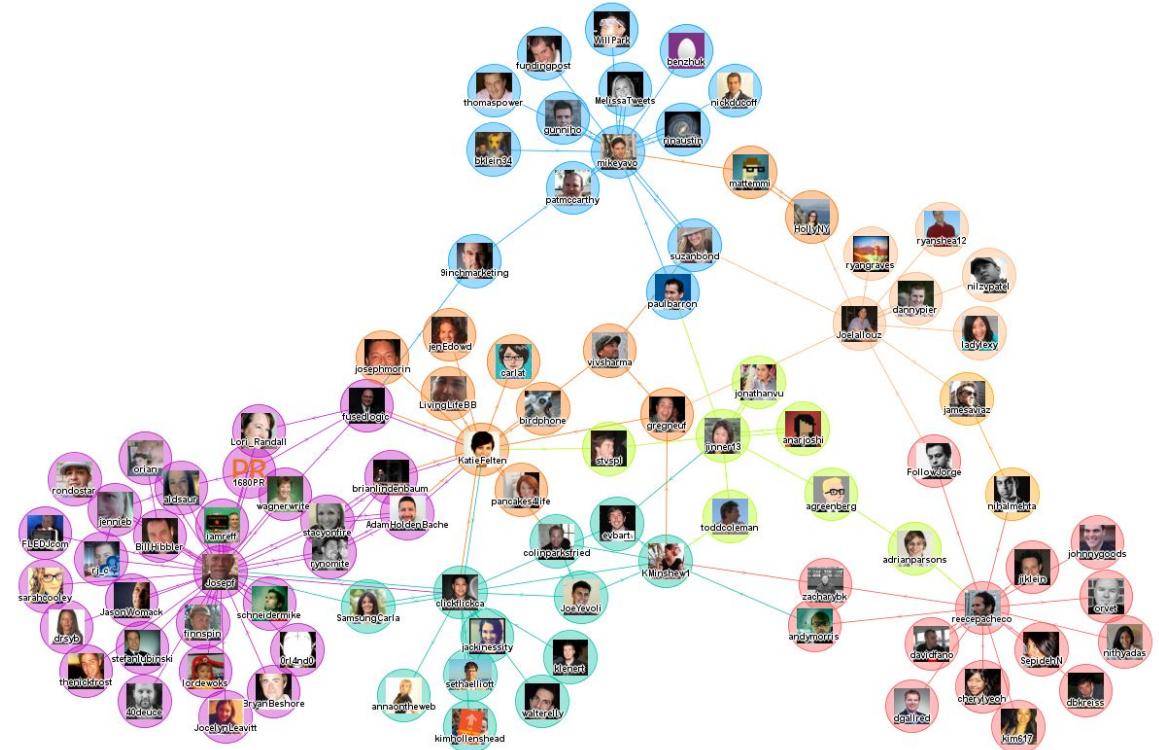
## Social Networks

**Nodes:** People, users

**Edges:** friendship relationships

### Applications:

- Identify communities
- Find influential people
- Friendship recommendations
- Effective Information diffusion  
(e.g., advertising, political campaigns)



# Outline

- **Graph Mining**
  - Application Examples
  - **Basic definitions**
- **Community Detection**
  - Basic definition & goals
  - Overview to different algorithms
- **Centrality**
  - Basic definition & goals
  - Overview to different algorithms
- **Summary**

# Graphs — Mathematical Definition

- Graph: Formalism for representing **relationships** between **items**

- A graph is a tuple  $G = (V, E)$

- Set of vertices (or nodes)  $V$

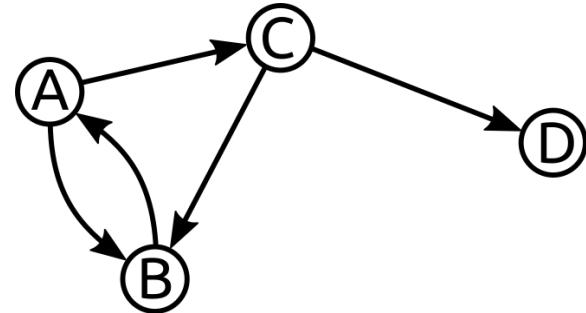
$$V = \{v_1, v_2, \dots, v_n\}$$

- Set of edges  $E$

$$E = \{e_1, e_2, \dots, e_m\}$$

where an edge is a pair of vertices

$$e_i = (v_j, v_k)$$



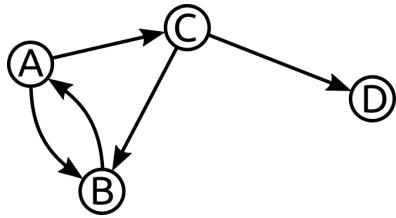
$$V = \{A, B, C, D\}$$

$$E = \{(A, B), (A, C), (C, D), (B, A), (C, B)\}$$

# Types of Graphs

	undirected	directed
unweighted	<p>friendships on Facebook directly connected MRT stations</p>	<p>hyperlinks between web pages followers on Twitter</p>
weighted	<p>co-authorship with number of papers transport network with travel times</p>	<p>#retweets between Twitter users borrower network with money owed</p>

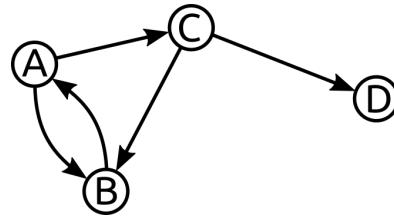
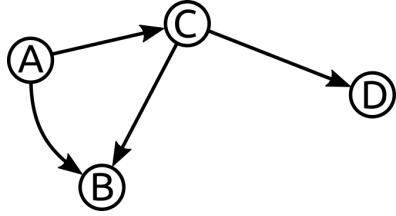
# Types of Graphs



Cyclic Graph

vs.

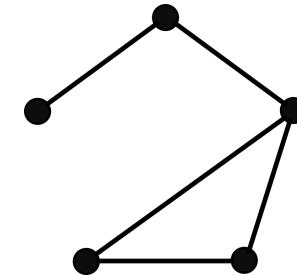
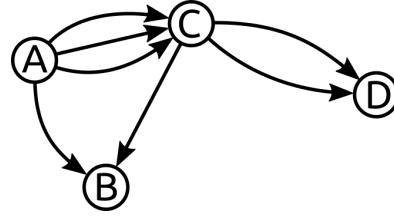
Acyclic Graph



(Simple) Graph

vs.

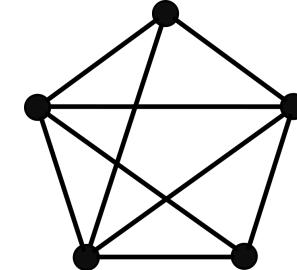
Multigraph



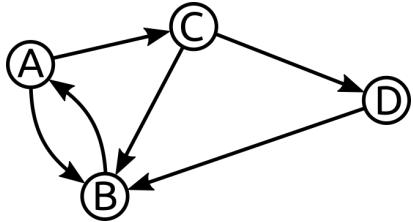
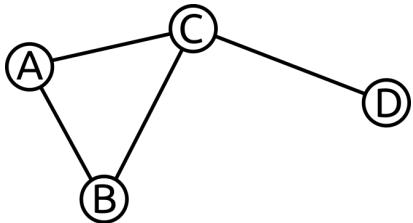
Sparse Graph

vs.

Dense Graph

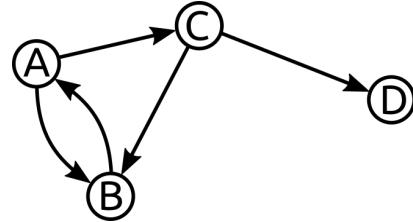


# Types of Graphs



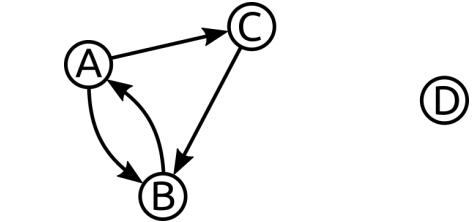
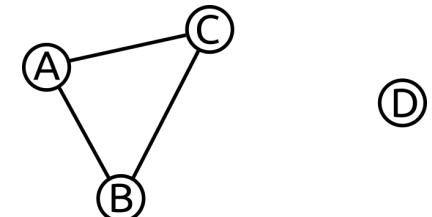
## (Strongly) Connected Graph

There exists a path from each node to every other node



## Weakly Connected Graph

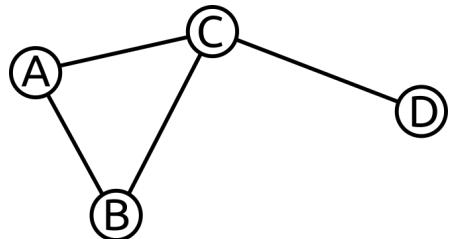
A directed graph where the underlying undirected graph is (strongly) connected



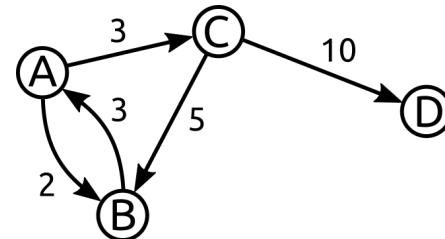
## Disconnected Graph

A graph that is not connected

# Representing Graphs — Adjacency Matrix



$$A = \begin{bmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$



$$A = \begin{bmatrix} 0 & 2 & 3 & 0 \\ 3 & 0 & 0 & 0 \\ 0 & 5 & 0 & 10 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

# Quick Quiz

When representing the **MRT** network as a graph  $G$  which **type of graph** will  $G$  definitely be?

A

Connected

B

Undirected

C

Weighted

D

Unweighted

# Quick Quiz

Which type of graph is **not suitable** to represent the SBS bus network? Why?

A

Sparse

B

Undirected

C

Weighted

D

Unweighted

# Overview

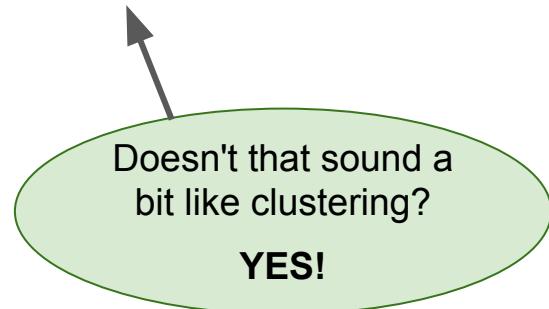
- **Graph Mining**
  - Application Examples
  - Basic definitions
- **Community Detection**
  - Basic definition & goals
  - Overview to different algorithms
- **Centrality**
  - Basic definition & goals
  - Overview to different algorithms
- **Summary**

# Community Detection

- No formal definition of a community
  - From "Networks: An Introduction" (Mark Newman):

*"Loosely stated, [community detection] is the problem of finding the natural divisions of a network into groups of vertices such that there are many edges within groups and few edges between groups. What exactly we mean by "many" or "few," however, is debatable, [...]"*

- Wide range of applications
  - Identifying groups in social networks
  - Recommendation systems
  - Market segmentation
  - Outlier/anomaly detection



# Community Detection — Modularity

- Modularity  $Q \in [-1/2, 1]$  of an undirected graph  $G$  with adjacency matrix  $A$ 
  - Measures the relative density of edges inside communities with respect to edges outside communities
  - Optimizing modularity is NP-hard → Practical algorithms based on heuristics

$$Q = \frac{1}{2m} \sum_{vw} \left[ A[v, w] - \frac{k_v k_w}{2m} \right] \underbrace{\delta(c_v, c_w)}_{\delta(c_v, c_w) = \begin{cases} 1 & , c_v = c_w \\ 0 & , \text{otherwise} \end{cases}}$$

Diagram illustrating the components of the modularity formula:

- $A[v, w]$  — weight of edge between nodes  $v$  and  $w$
- $k_i$  — sum of the weights of edges attached to node  $i$
- $c_i$  — community of a node  $i$
- $m$  — sum of weights of all edges

# Community Detection — Louvain Algorithm

- Method for the optimization of modularity

**Initialization:** Each node is a community

**Repeat** until no further change

## Phase 1: Modularity Optimization

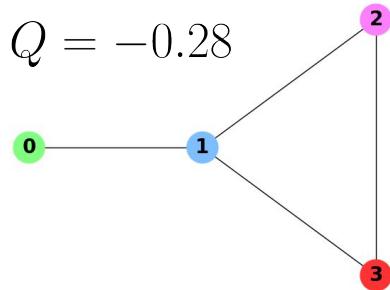
- For each node  $v$ , check if moving it to an adjacent community improves modularity
- Move  $v$  to community that maximizes modularity

## Phase 2: Graph Aggregation

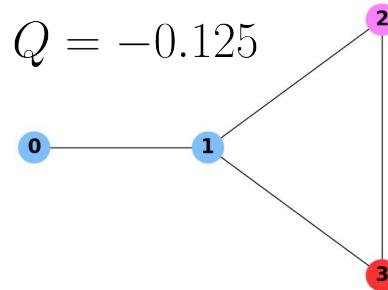
- Represent each community as a new node
- Update weights between new nodes

# Louvain Algorithm — Modularity Optimization

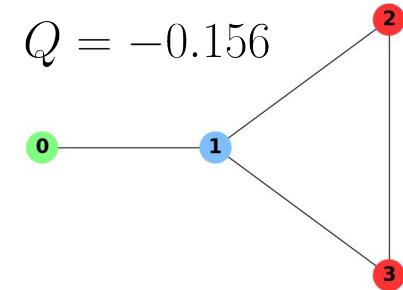
Input: Each node a community



Moving community 0 to community 1

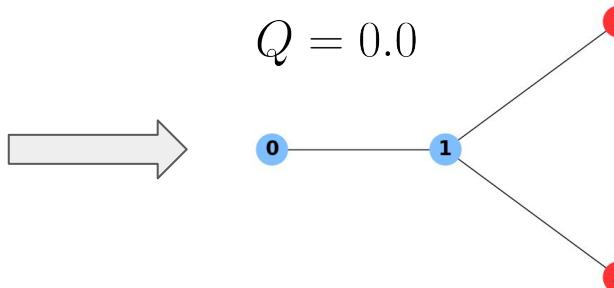


Moving community 2 to community 3



→ Move community 0 to community 1 as it maximizes modularity

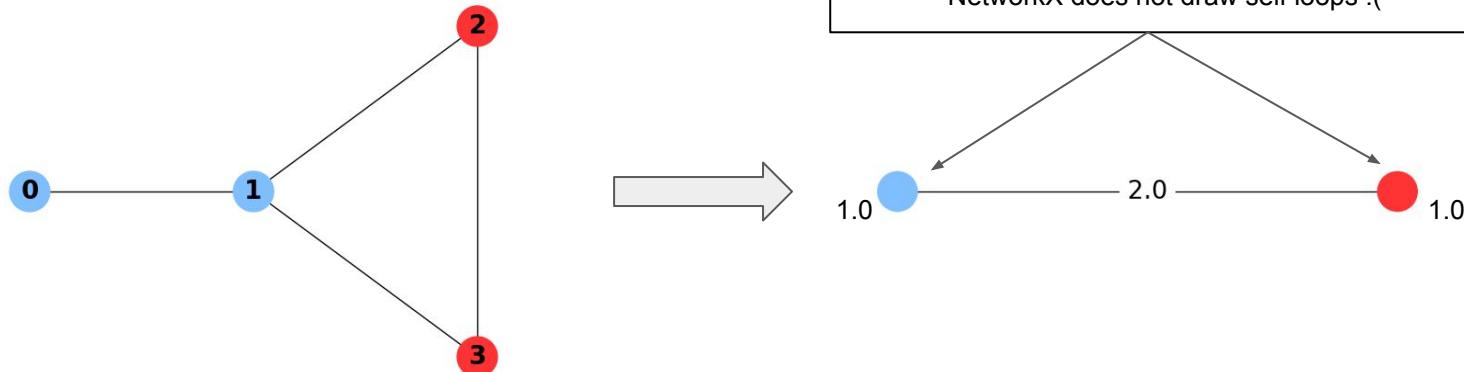
Output after  
all iterations



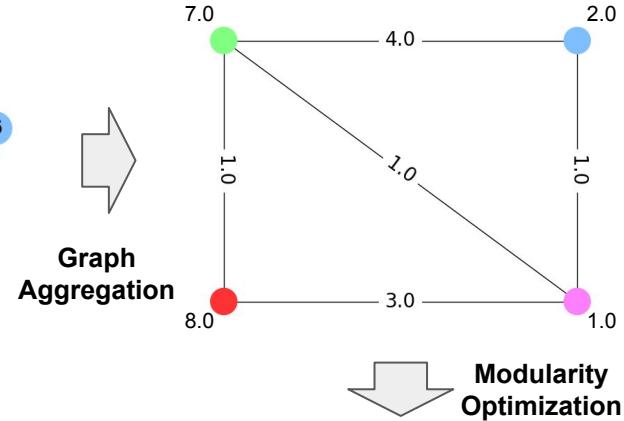
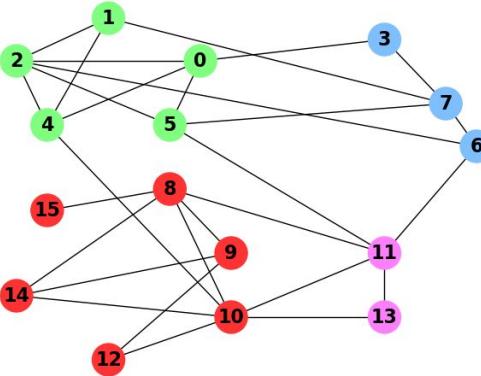
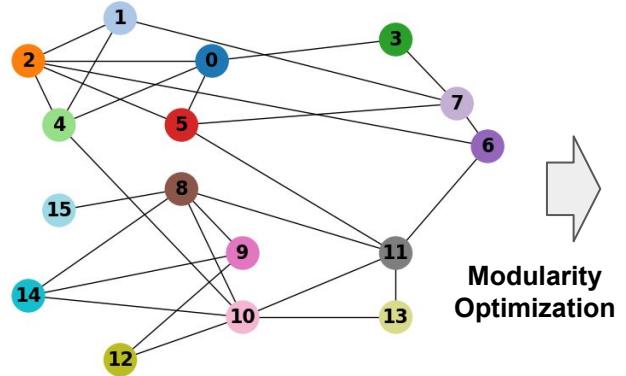
Note: Moving a node may not be permanent but can change again in later iteration before convergence.

# Louvain Algorithm — Graph Aggregation

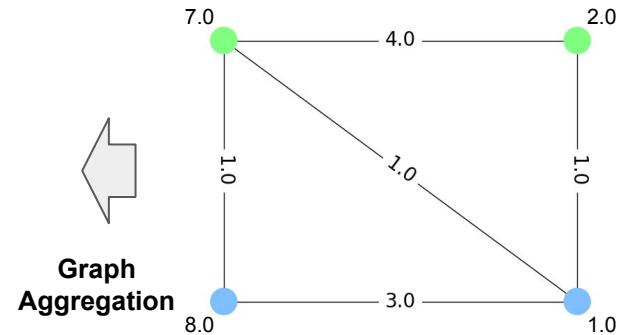
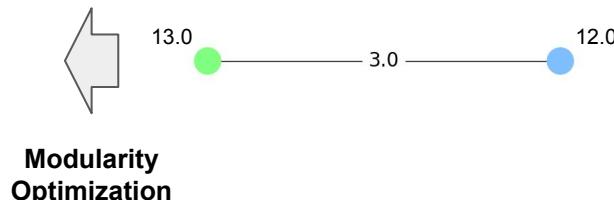
- 2 main steps
  - Merge all nodes of a community into a new single node
  - Aggregate edge weights accordingly



# Louvain Algorithm — Full Example



No change in  
communities  
→ Done!



# Louvain Algorithm — Remarks

- **Heuristic**
  - Optimizes modularity locally on all nodes
  - No guarantees for optimal modularity globally
    - (in practice often superior to other methods)
- **Performance optimization**
  - Phase 1 (Modularity Optimization) requires to calculate change in modularity  $\Delta Q$ 
    - (difference between modularity of G before and after moving communities)
  - Calculating  $\Delta Q$  can be done based on local changes in community assignments
    - (does not require the recalculate the modularity G after each change)

Full details in paper: [Fast Unfolding of Communities in Large Networks](#) (Blondel et al., 2008)

# Community Detection — Girvan-Newman Algorithm

- **Divisive hierarchical approach**
  - Start with whole graph representing a community
  - Iteratively remove edges until community is split into 2 sub-communities  
(continue splitting sub-communities recursively if needed)
- Criteria for removing edges: **Edge Betweenness Centrality**
  - Sum of fraction of all-pairs shortest paths that pass through an edge e

$$c_B(e) = \sum_{v,w \in V} \frac{\sigma(v, w|e)}{\sigma(v, w)}$$

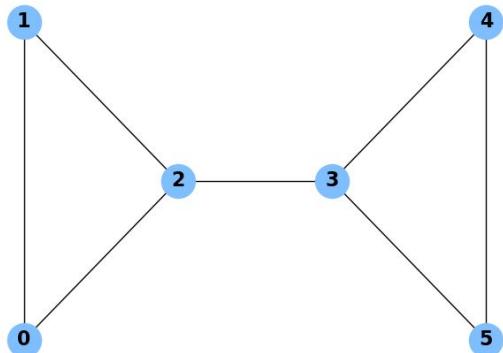
number of shortest paths from v to w going through edge e

number of shortest paths from v to w

Full details in paper: [On Variants of Shortest-Path Betweenness Centrality and their Generic Computation](#) (Brandes et al., 2008)

# Girvan-Newman Algorithm — Edge Betweenness Centrality

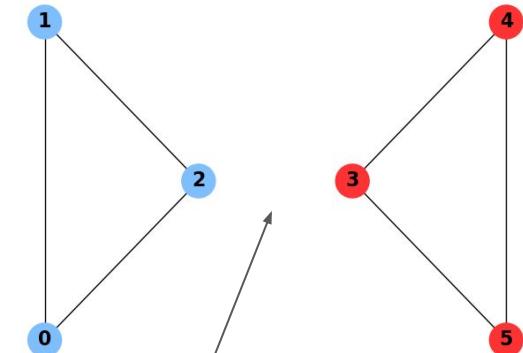
Input community



Betweenness Centrality  
for all edges

$e$	$c_B(e)$
(0, 1)	0.067
(0, 2)	0.267
(1, 2)	0.267
<b>(2, 3)</b>	<b>0.600</b>
(3, 4)	0.267
(3, 5)	0.267
(4, 5)	0.067

Output communities



Remove edge  $e$  with max.  $c_B(e)$

Continue recursively until community is split  
(not needed in this simple example)

# Girvan-Newman Algorithm

- Algorithm splits a graph  $G(V, E)$  into 2 disconnected components

**Repeat**

Calculate  $c_B(e)$  for all  $e \in E$

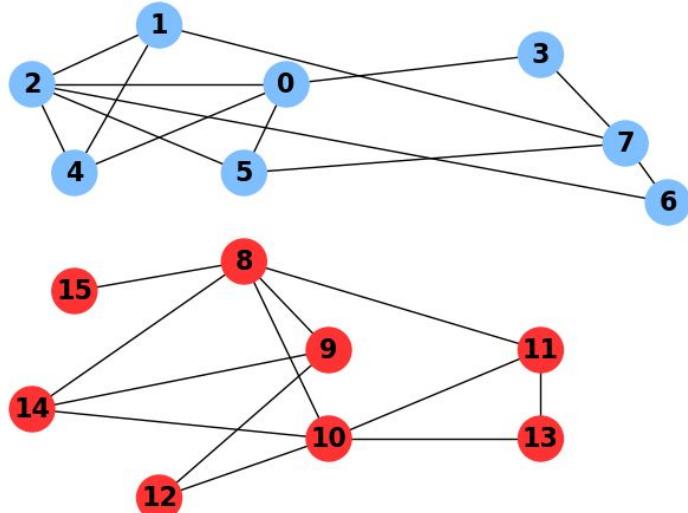
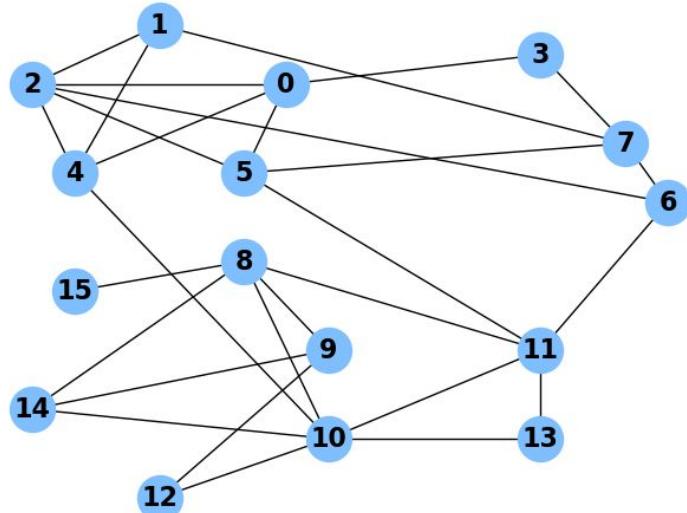
Remove edge from with max.  $c_B(e)$

**Until**  $G$  is split into 2 components

- Recursive step

- Apply algorithm to each new component
- Stops if a component contains only single node  
(or early stop based on user specifications)

# Girvan-Newman Algorithm — Full Example



# Girvan-Newman Algorithm — Remarks

- Complexity Analysis
  - Core concept of algorithm: Edge Betweenness Centrality
  - Requires to solve the **All-Pairs Shortest Path (APSP)** problem
  - Various algorithms and complexities depending on type of graph  
(directed vs. undirected, cyclic vs. acyclic, with or without negative weights, etc.)

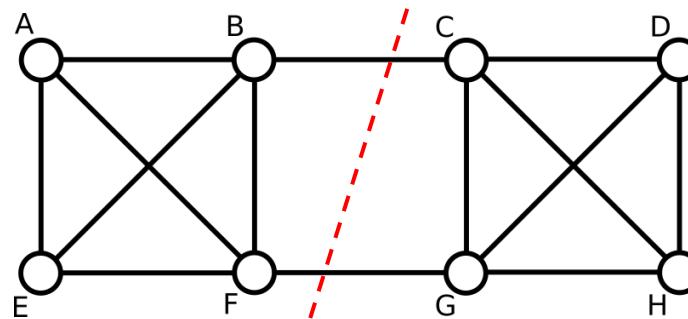
Time Complexity
$n^3$
$n^3(\log n)/\log n^{1/3}$
$n^3(\log n/\log n)^{1/2}$
$n^3/(\log n)^{1/2}$
$n^3(\log n/\log n)^{5/7}$
$n^3 \log n/\log n$
$n^3(\log n)^{1/2}/\log n$
$n^3/\log n$
$n^3(\log n/\log n)^{5/4}$
$n^3(\log n)^3/(\log n)^2$
$n^3(\log n)/(\log n)^2$

Table taken from: [A Survey of Shortest-Path Algorithms](#)  
(Madkour et al., 2017) —  $n = |V|$ , number of nodes

# Karger's Algorithm for Min-Cut

- **Min-Cut Problem**

- Given a graph  $G$ , cut  $G$  into 2 components such that the number of edges between both components is minimal



$$\rightarrow |\text{Min-Cut}| = 2$$

- Fundamental problem in graph theory → many existing algorithms  
(with varying focus and support for different graph types — e.g., directed vs. undirected)

# Karger's Algorithm for Min-Cut

- Karger's algorithm
  - Randomized method to find Min-Cut
  - Applicable to undirected graphs with positive weights  
(this includes unweighted graphs where the weight can be considered 1)

**While**  $|V| > 2$

Randomly pick a remaining edge  $e = (v, u)$

Merge/contract  $v$  and  $u$  into a new node

- Update edges to neighbors of  $v$  and  $u$
- Remove self-loops

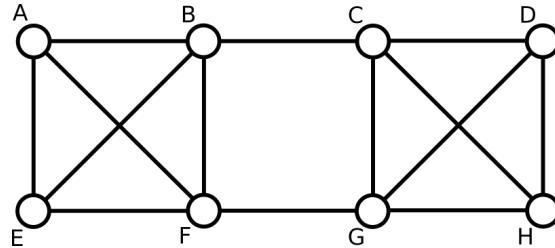
**Return** edges between the final 2 nodes as Min-Cut

**Intuition:** Edges that are in the Min-Cut have a lower probability to get picked!

Basic runtime:  $O(|V|^2)$

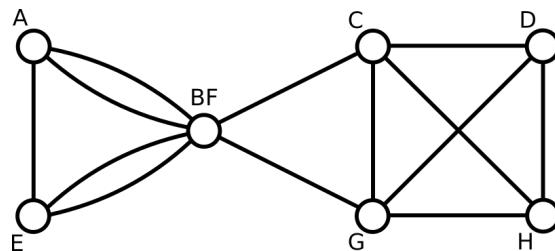
(but further optimizations exists)

# Karger's Algorithm for Min-Cut — Example



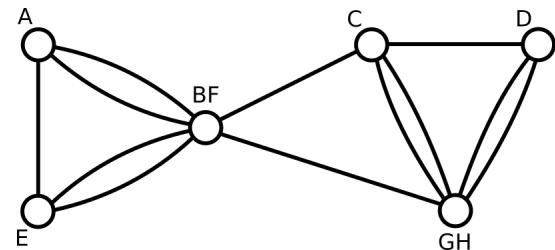
Number of edges: 14

Pick  $e = (B, F)$  with  $P(e) = 1/14$



Number of edges: 13

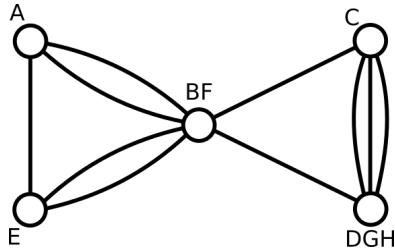
Pick  $e = (G, H)$  with  $P(e) = 1/13$



Number of edges: 12

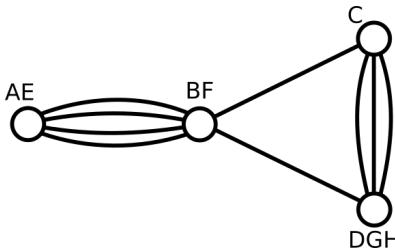
Pick  $e = (D, GH)$  with  $P(e) = 1/6$

# Karger's Algorithm for Min-Cut — Example



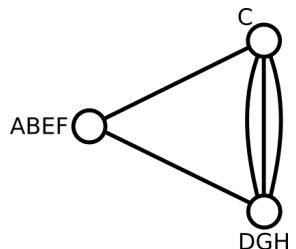
Number of edges: 10

Pick  $e = (A, E)$  with  $P(e) = 1/10$



Number of edges: 9

Pick  $e = (AE, BF)$  with  $P(e) = 4/9$



Number of edges: 5

Pick  $e = (C, DGH)$  with  $P(e) = 3/5$



2 node left → Done, with  $|\text{Min-Cut}| = 2$

# Karger's Algorithm for Min-Cut — Analysis

- What is the probability that the algorithm finds the "correct" Min-Cut?
- For an undirected graph  $G = (V, E)$ , with  $n = |V|$  and  $m = |E|$

The average degree of a node is  $\frac{1}{n} \sum_{v \in V} \text{degree}(v) = \frac{2m}{n}$

The size of the Min-Cut is limited to  $|Min-Cut| \leq \frac{2m}{n}$  since  $\forall v \in V : |Min-Cut| \leq \text{degree}(v)$

→  $P(\text{randomly selected edge is in Min-Cut}) \leq \frac{\frac{2m}{n}}{m} = \frac{2}{n}$

# Karger's Algorithm for Min-Cut — Analysis

- Let  $P(\text{success}) = P(\text{final cut is Min-Cut})$

$$P(\text{success}) = P(\text{1st selected edge not in Min-Cut}) \times \\ P(\text{2nd selected edge not in Min-Cut}) \times \dots$$

$$\begin{aligned} P(\text{success}) &\geq \left(1 - \frac{2}{n}\right) \left(1 - \frac{2}{n-1}\right) \left(1 - \frac{2}{n-2}\right) \dots \left(1 - \frac{2}{4}\right) \left(1 - \frac{2}{3}\right) \\ &= \frac{n-2}{n} \cdot \frac{n-3}{n-1} \cdot \frac{n-4}{n-3} \cdot \dots \cdot \frac{2}{4} \cdot \frac{1}{3} \\ &= \frac{2}{n(n-1)} = \binom{n}{2}^{-1} \end{aligned}$$

That's a rather low probability :(  
→ Run algorithm multiple times!  
But how often?

# Karger's Algorithm for Min-Cut — Analysis

- If the algorithm is run  $k$  times and take the smallest cut found
  - What is the probability  $P(\text{failure})$  that it is not a Min-Cut?

$$P(\text{failure}) = \left[1 - \binom{n}{2}^{-1}\right]^k$$

$$\text{with } k = \binom{n}{2} \ln n \quad \rightarrow \quad P(\text{failure}) \leq \left(\frac{1}{e}\right)^{\ln n} = \frac{1}{n}$$

**Note:** For any  $x \geq 1$ :

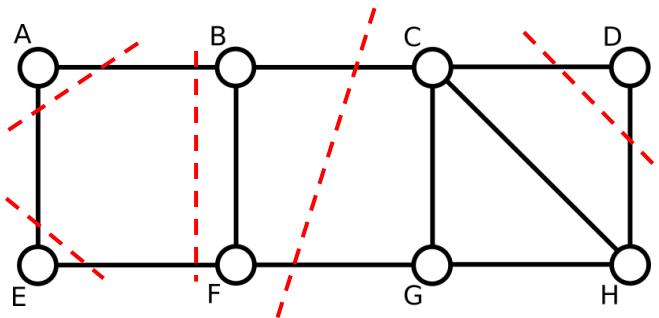
$$\frac{1}{4} \leq \left(1 - \frac{1}{x}\right)^{cx} \leq \left(\frac{1}{e}\right)^c$$

With  $k \in O(n^2 \log n)$  → Total runtime of Karger's Algorithm is in  $O(n^4 \log n)$

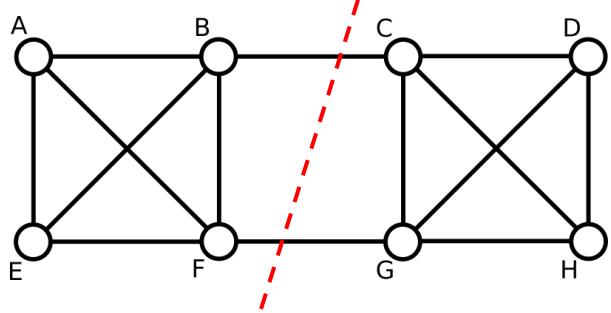
(with a Min-Cut with a high probability)

# Karger's Algorithm for Min-Cut — Remarks

- In general, a graph has multiple possible Min-Cuts



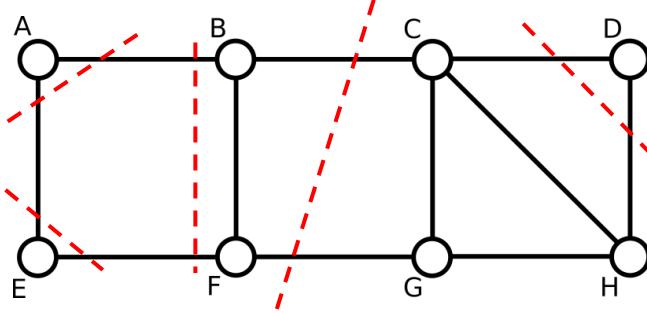
- Choice of Min-Cut often application-specific, e.g.,
  - Favor Min-Cuts where the 2 components are of similar size (e.g., similar number of nodes)
  - Ignore Min-Cuts where the size of a component is below a threshold



```

|Min-Cut| = 5 -- Components: A,B ### C,D,E,F,G,H
|Min-Cut| = 4 -- Components: A,C,D,E,F,G,H ### B
|Min-Cut| = 3 -- Components: A,B,C,D,E,F,G ### H
|Min-Cut| = 3 -- Components: A,B,C,D,E,F,G ### H
|Min-Cut| = 5 -- Components: A,B,C,D,E,F ### G,H
|Min-Cut| = 2 -- Components: A,B,E,F ### C,D,G,H
|Min-Cut| = 2 -- Components: A,B,E,F ### C,D,G,H
|Min-Cut| = 3 -- Components: A,B,C,D,E,F,G ### H
|Min-Cut| = 4 -- Components: A,B,D,E,F,G,H ### C
|Min-Cut| = 5 -- Components: A,F ### B,C,D,E,G,H
|Min-Cut| = 4 -- Components: A,B,C,E,F,G ### D,H
|Min-Cut| = 5 -- Components: A,B,C,D,E,F ### G,H
|Min-Cut| = 4 -- Components: A,B,C,E,F ### D,G,H
|Min-Cut| = 3 -- Components: A,B,C,E,F,G,H ### D
|Min-Cut| = 3 -- Components: A,B,C,D,E,F,G ### H
|Min-Cut| = 4 -- Components: A,B,C,E,F,G ### D,H
|Min-Cut| = 4 -- Components: A,B,C,D,E,G,H ### F
|Min-Cut| = 2 -- Components: A,B,E,F ### C,D,G,H
|Min-Cut| = 2 -- Components: A,B,E,F ### C,D,G,H
|Min-Cut| = 2 -- Components: A,B,E,F ### C,D,G,H

```



```

|Min-Cut| = 2 -- Components: A,B,C,E,F,G,H ### D
|Min-Cut| = 2 -- Components: A,B,C,E,F,G,H ### D
|Min-Cut| = 2 -- Components: A,B,E,F ### C,D,G,H
|Min-Cut| = 2 -- Components: A,B,C,E,F,G,H ### D
|Min-Cut| = 2 -- Components: A,E ### B,C,D,E,F,G,H
|Min-Cut| = 3 -- Components: A,B,C,D,E,F,G ### H
|Min-Cut| = 2 -- Components: A,E ### B,C,D,E,F,G,H
|Min-Cut| = 3 -- Components: A,B,C,D,E,F,H ### G
|Min-Cut| = 3 -- Components: A,B,C,E,F,G ### D,H
|Min-Cut| = 2 -- Components: A,B,C,D,E,F,H ### G
|Min-Cut| = 3 -- Components: A,B,C,D,E,F,H ### G
|Min-Cut| = 3 -- Components: A,B,C,D,E,F,G ### H
|Min-Cut| = 2 -- Components: A,E ### B,C,D,E,F,G,H
|Min-Cut| = 2 -- Components: A,B,C,E,F,G,H ### D
|Min-Cut| = 3 -- Components: A,B,C,D,E,F,H ### G
|Min-Cut| = 3 -- Components: A,B,C,D,E,F,G ### H
|Min-Cut| = 2 -- Components: A,E ### B,C,D,E,F,G,H
|Min-Cut| = 2 -- Components: A,B,C,E,F,G,H ### D
|Min-Cut| = 2 -- Components: A,B,E,F ### C,D,G,H
|Min-Cut| = 2 -- Components: A,B,C,D,F,G,H ### E
|Min-Cut| = 2 -- Components: A,B,E,F ### C,D,G,H
|Min-Cut| = 2 -- Components: A,B,C,E,F,G,H ### D
|Min-Cut| = 2 -- Components: A,B,E,F ### C,D,G,H
|Min-Cut| = 2 -- Components: A,B,E,F ### C,D,G,H

```

# Overview

- **Graph Mining**
  - Application Examples
  - Basic definitions
- **Community Detection**
  - Basic definition & goals
  - Overview to different algorithms
- **Centrality**
  - Basic definition & goals
  - Overview to different algorithms
- **Summary**

# Centrality

- **Centrality — Centrality measures**

- Quantify the **importance** of a node given its topological position in a graph  
(centrality is commonly assigned to nodes but can be extended to edges, cf. Edge Betweenness Centrality)
- Different measures favor different "flavours" of importance

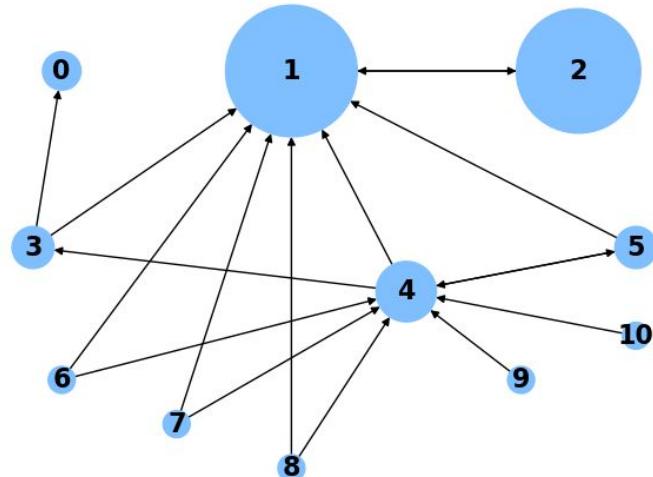
## → What makes a node important?

(or more important compared to other nodes)

- **Wide range of applications**

- Identify influential social network users
- Identify superspreaders of diseases
- Identify key infrastructure nodes in infrastructure networks

Example: PageRank of 10 web pages



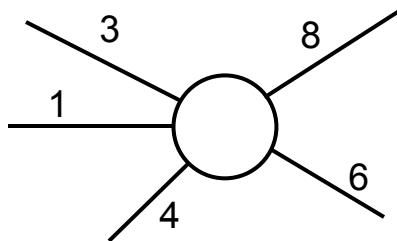
# Degree Centrality

- Centrality of a node only dependent on direct neighborhood edges

Undirected graph

Sum of weights of connected edges

$$c_d(v_i) = \sum_{v_j \in V} A[i, j]$$

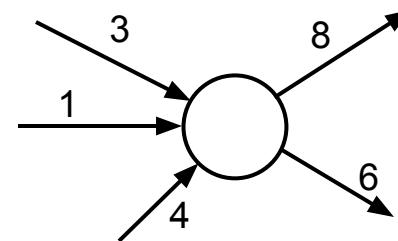


$$c_d(v) = 22$$

Directed graph

Sum of weights of incoming edges

$$c_{d\_in}(v_i) = \sum_{v_j \in V} A[j, i]$$



$$c_{d\_in}(v) = 8$$

Sum of weights of outgoing edges

$$c_{d\_out}(v_i) = \sum_{v_j \in V} A[i, j]$$

$$c_{d\_out}(v) = 14$$

Note: For unweighted graphs,  $A[i,j] = 1$  for existing edges → sum of weights = edge count

# Degree Centrality — Pros & Cons

- **Pros**

- Simple measure → easy and fast to calculate
- For many applications "good enough"

- **Cons**

- Local measure — does not take any extended topological information of a node into account
- Treats all connected edges of a node equally (i.e., neighboring source or target node does not matter)
- Depending on application, easy to manipulate

## Examples of manipulation attacks

- Create fake pages with links to a target page to bump up its search result rank
- Create fake followers on social media to increase reputation and attract sponsors
- Create fake feedback to create on e-commerce sites to increase reputation

# Eigenvector Centrality

- Idea: Centrality of a node depends on the centrality of its neighbors
  - Basic definition applicable to undirected graph
  - Recursive definition — How to calculate it? (well, it's in the name)

$$c_{ev}(v_i) = \frac{1}{\lambda} \sum_{v_j \in V} [A[i, j] \cdot c_{ev}(v_j)]$$

$\lambda$  — some normalization constant

In matrix form:  $\lambda c_{ev} = A c_{ev}$

$c_{ev}$  — vector of centrality degrees of all nodes

Common Eigenvector equation:

$$A\vec{x} = \lambda\vec{x}$$

All  $\vec{x}$  solving this equation are the eigenvectors of  $A$  and  $\lambda$  are their corresponding eigenvalues:

$$(A - \lambda I)\vec{x} = 0$$

$c_{ev}$  is the largest eigenvector of adjacency matrix  $A$ !

# Eigenvector Centrality — Power Iteration / Power Method

- Power Iteration — numerical method to find largest eigenvector of a matrix
  - Solving eigenvector equation analytically not tractable for large matrices

**Input:** matrix  $M$ , error threshold  $\varepsilon$ , #max. iterations  $T$

**Initialization:**  $t = 0$ ,  $x_0 = [1/|V|, 1/|V|, \dots, 1/|V|]$

**Repeat**

$$t = t+1$$

$$x_t = Ax_{t-1}$$

$$x_t = x_t / \|x_t\| \quad \# \text{ Normalize vector}$$

$$\delta = \|x_t - x_{t-1}\| \quad \# \text{ Calculate difference}$$

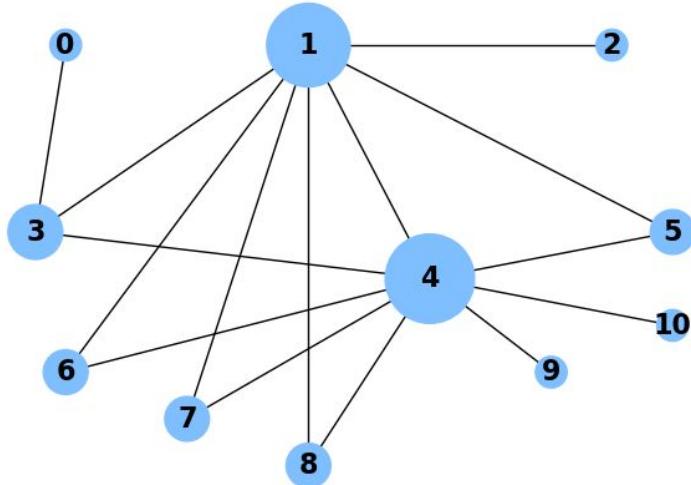
**Until**  $\delta < \varepsilon$  **or**  $t > T$

**Return**  $x_t$

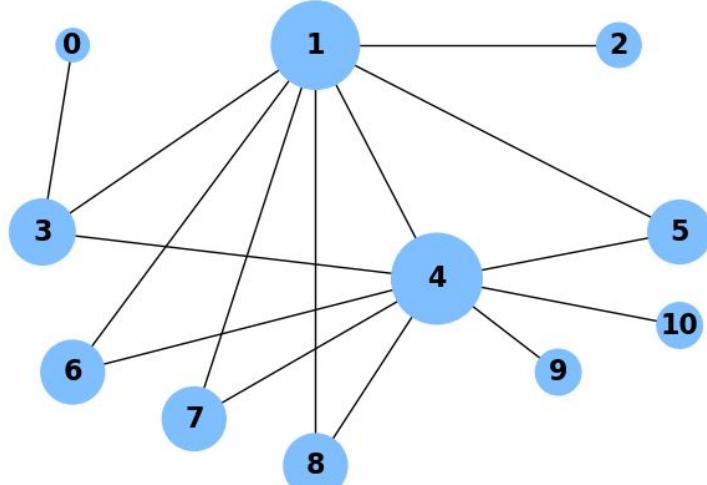
# Degree vs. Eigenvector Centrality

(size of nodes reflect centrality scores)

Degree



Eigenvector



- Low-degree nodes benefit from connections to high-degree nodes

# PageRank

- Goal: Find important pages in the web graph
  - The set of vertices  $V$  is set of all (indexed) web pages
  - An edge  $e_{ij} \in E$  indicates that there is a hyperlink for page  $i$  to page  $j$

- PageRank — a page  $v_j$  has a high PageRank if

- Many other pages link to  $v_j$
- Those pages linking to  $v_j$  have
  - (a) a high PageRank themselves
  - (b) not many other outgoing links



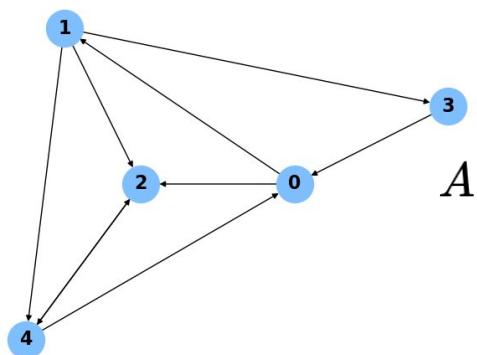
$$PR(v_j) = \sum_{i \rightarrow j} \frac{1}{\text{outdegree}(i)} PR(v_i)$$



Matrix notation for all vertices

$$PR(v) = M \cdot PR(v)$$

# PageRank — Transition Matrix M



Adjacency matrix A

$$A = \begin{bmatrix} 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 \end{bmatrix}$$

- Transpose A
- Normalize columns  
(column entries sum up to 1)

$$\xrightarrow{\hspace{1cm}} M =$$

Transition matrix M

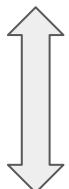
$$M = \begin{bmatrix} 0 & 0 & 0 & 1 & \frac{1}{2} \\ \frac{1}{2} & 0 & 0 & 0 & 0 \\ \frac{1}{2} & \frac{1}{3} & 0 & 0 & \frac{1}{2} \\ 0 & \frac{1}{3} & 0 & 0 & 0 \\ 0 & \frac{1}{3} & 1 & 0 & 0 \end{bmatrix}$$

**Note:** M is a so-called column-stochastic matrix.

# Computing PageRank Scores

- PageRank is an Eigenvector problem
  - Solvable with Power Iteration Method

$$\lambda PR(v) = M \cdot PR(v)$$



Since  $M$  is column-stochastic,  
the largest eigenvalue  $\lambda = 1$ .

$$PR(v) = M \cdot PR(v)$$

Common Eigenvector equation:

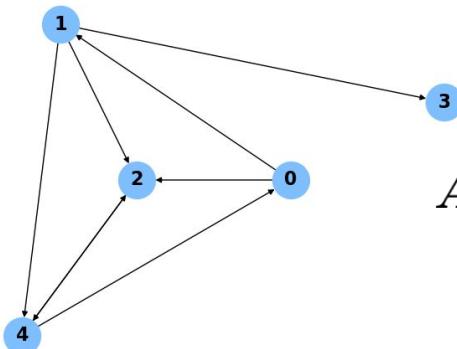
$$A\vec{x} = \lambda\vec{x}$$

All  $\vec{x}$  solving this equation are the eigenvectors  
of  $A$  and  $\lambda$  are their corresponding eigenvalues:

$$\vec{x} \quad \lambda \quad (A - \lambda I) \vec{x} = 0$$

# Computing PageRank Scores — Challenge

- Requirement: Graph must be strongly connected
  - Each node can be reached from any other node
  - Requirement does not hold for the web graph → dangling nodes
  - Computing PageRank scores using Power Iteration Method not working



Adjacency matrix A

$$A = \begin{bmatrix} 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 \end{bmatrix}$$

- Transpose A
- Normalize columns  
(column entries sum up to 1?)



Transition matrix M

$$M = \begin{bmatrix} 0 & 0 & 0 & 0 & \frac{1}{2} \\ \frac{1}{2} & 0 & 0 & 0 & 0 \\ \frac{1}{2} & \frac{1}{3} & 0 & 0 & \frac{1}{2} \\ 0 & \frac{1}{3} & 0 & 0 & 0 \\ 0 & \frac{1}{3} & 1 & 0 & 0 \end{bmatrix}$$

# Computing PageRank Scores — Handling Dangling Nodes

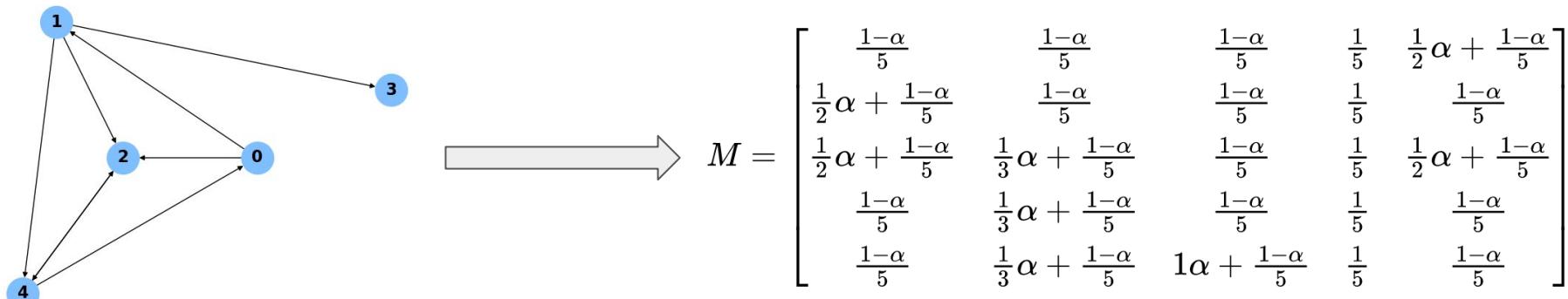
- PageRank implements **Random Surfer** model
    - $\alpha$  probability of following a link on a page to the next
    - $(1 - \alpha)$  probability of jumping to a random page (i.e., not following a link)
- Any page can be reached, whether linked or not!

$$PR(v) = M \cdot PR(v) \iff PR(v) = \alpha \cdot M \cdot PR(v) + (1 - \alpha) \cdot E$$

$$\text{with } E = \begin{bmatrix} 1/|V| \\ 1/|V| \\ \vdots \\ 1/|V| \end{bmatrix}$$

# Random Surfer Model — Effects on Transition Matrix

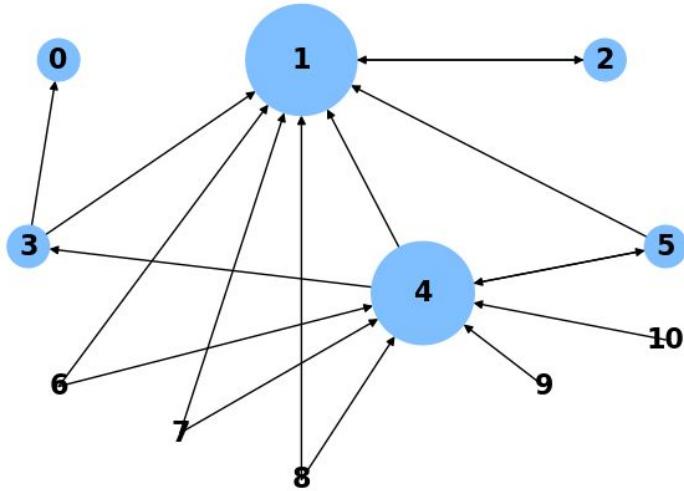
- Random Surfer model introduces "virtual edges"
  - Each node can be reached from any other → virtual links → fully connected graph
  - PageRank scores can be calculated using Power Iteration Method without problems



**\*Note:** Matrix  $M$  is not materialized, but this is the matrix that reflects the PageRank formula!

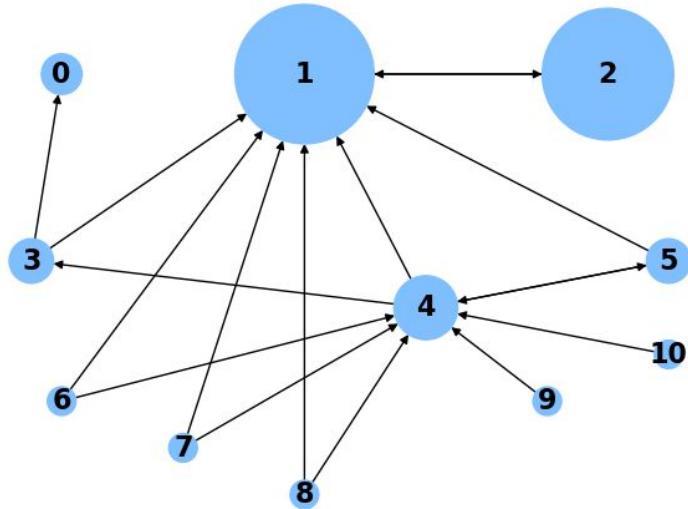
# InDegree vs. PageRank (size of nodes reflect centrality scores)

InDegree



- Nodes 1 and 4 very similar centrality scores
- Node 2 with only a low centrality score
- Scores of 0 for nodes with no incoming edges

PageRank



- Clear difference between Nodes 1 and 4
- Node 2 with a high centrality score since linked to from Node 1 with a very high score
- Non-zero scores for nodes with no incoming edges

# Quick Quiz

Given a Graph with 10 nodes and a setting of  $\alpha=0.85$ , what is the **lowest possible** PageRank score?

$$c_{pr}(v_i) = \alpha M c_{pr}(v_i) + (1 - \alpha)E$$

A

0.0

B

0.015

C

0.085

D

0.15

# Eigenvector-Based Measures — Remarks

- Measuring centrality by solving an Eigenvector problem
  - Recursive definition of centrality very intuitive
  - Many other similar measures (e.g., HITS, SALSA, Katz)
  - Many application-specific extensions to basic measures  
(e.g., personalization of PageRank where random jumps are no longer uniform)
  - More complex calculation than for local measures but calculation of largest Eigenvectors very scalable through parallelization

# Closeness Centrality

- Intuition: A node  $t$  is central if the distance to all other nodes is small
  - Small distance to node  $t$  = short paths from all other nodes to  $t$
  - For directed paths, the closeness of node  $t$  can differ greatly when considering incoming or outgoing edges for calculating distances
  - Basic definition applicable to unweighted graph  
(generalized definitions for weighted graphs have been proposed)

$$c_{cl}(v) = \frac{N - 1}{\sum_{w \in V} d(w, v)}$$

$N$  — number of nodes from which  $v$  is reachable

$d(w, v)$  — length of shortest path from  $w$  to  $v$

**Note:** For directed graphs, this definition calculates closeness using the nodes' incoming edges — more common case.  
To consider outgoing edges,  $d(w, v)$  becomes  $d(v, w)$ , and  $N$  becomes the number of nodes reachable from  $v$ .

# Betweenness Centrality

- Intuition: A node  $t$  is central if many shortest paths between all other nodes pass through node  $t$ 
  - Removing such nodes would cause the most "disruption" in a graph
  - Directly applicable to directed/undirected and weighted/unweighted graphs  
(since the notion of shortest path is well-defined for all graph types)

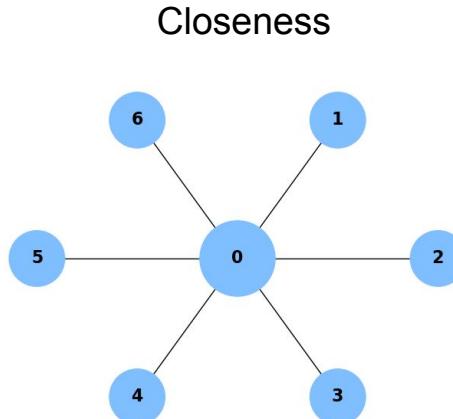
$$c_b(v) = \sum_{s,t \in V; s \neq v \neq t} \frac{\sigma_{st}(v)}{\sigma_{st}}$$

number of shortest paths from  $s$  to  $t$  passing through node  $v$

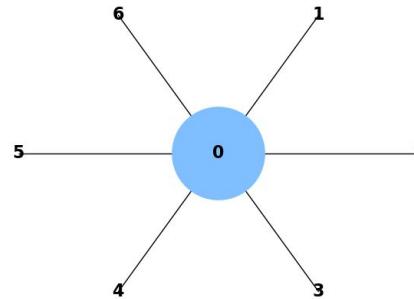
total number of shortest paths from  $s$  to  $t$

# Closeness vs. Betweenness (size of nodes reflect centrality scores)

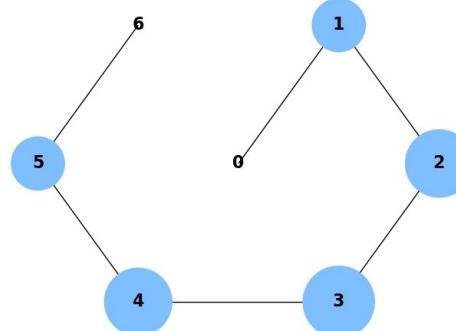
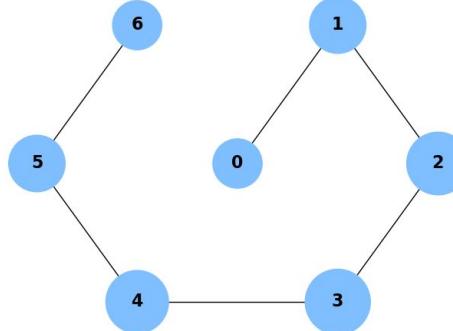
Star network graph



Betweenness

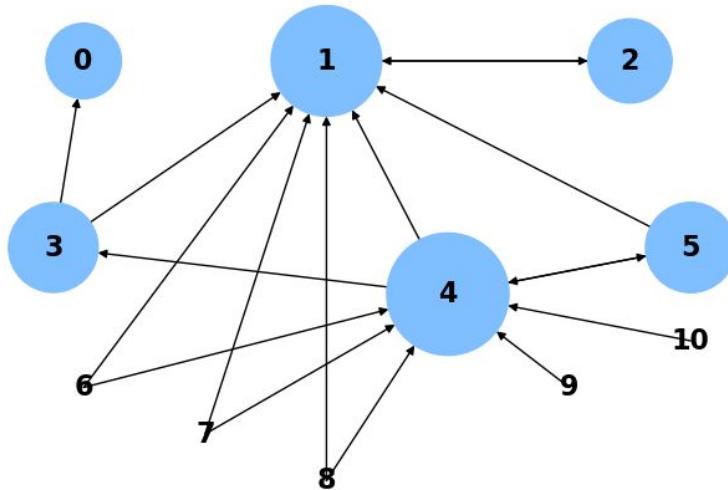


Sequence graph

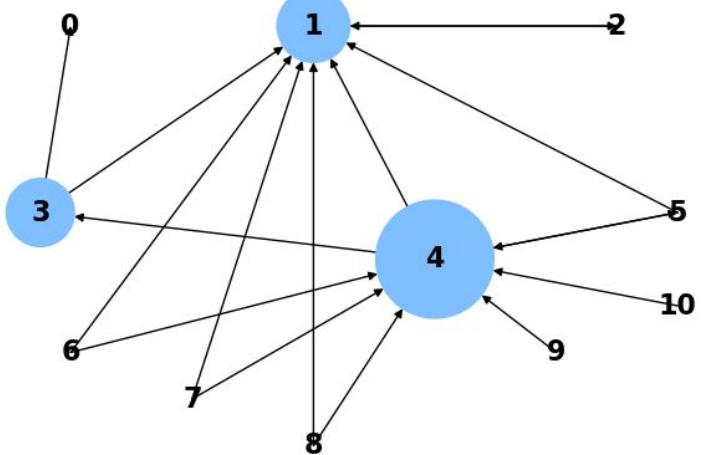


# Closeness vs. Betweenness (size of nodes reflect centrality scores)

Closeness



Betweenness



# Closeness & Betweenness — Remarks

- Distance-based measures

- Both measures rely on the notion of shortest paths between nodes
- Requires to solve the **All-Pairs Shortest Path (APSP)** problem
- Various algorithms and complexities depending on type of graph  
(directed vs. undirected, cyclic vs. acyclic, with or without negative weights, etc.)

Time Complexity
$n^3$
$n^3(\log n)/\log n^{1/3}$
$n^3(\log n/\log n)^{1/2}$
$n^3/(\log n)^{1/2}$
$n^3(\log n/\log n)^{5/7}$
$n^3 \log n/\log n$
$n^3(\log n)^{1/2}/\log n$
$n^3/\log n$
$n^3(\log n/\log n)^{5/4}$
$n^3(\log n)^3/(\log n)^2$
$n^3(\log n)/(\log n)^2$

Table taken from: [A Survey of Shortest-Path Algorithms](#)  
(Madkour et al., 2017) —  $n = |V|$ , number of nodes

# Centrality — Discussion

- Centrality = importance of nodes on a graph
  - Important concept of graph mining with many applications
  - Wide range of proposed measures that differ in their definition of a node's importance
  - Not all measures are applicable (or suitable) to all types of graphs
- Overview to a selected set of popular measures
  - Local measures — Degree, InDegree, OutDegree
  - Eigenvector-based measures — Eigenvector Centrality, PageRank
  - Distance-based measures — Closeness, Betweenness

# Quick Quiz

Given an **undirected** and **connected** graph G, which centrality measure can yield **scores of 0**?

A

Eigenvector

B

Closeness

C

Degree

D

Betweenness

# Quick Quiz

Which MRT station has the highest  
**Closeness centrality** score?

(graph does not include LRT stations)

A

Dhoby Ghaut

B

Buona Vista

C

City Hall

D

Stevens

# Quick Quiz

Which MRT station has the highest  
**Betweenness centrality** score?

(graph does not include LRT stations)

A

Bugis

B

Botanic Gardens

C

Clementi

D

Outram Park

# Solutions to Quick Quizzes

- Slide 16: A
- Slide 17: B
- Slide 54: B
- Slide 62: D
- Slide 63: A
- Slide 64: B

# **CS5228: Knowledge Discovery and Data Mining**

## Lecture 7 — Dimensionality Reduction

# Outline

- **Dimensionality Reduction**
  - Motivation
  - Naive approaches
- Dimensionality Reduction Techniques
  - PCA — Principal Component Analysis
  - LDA — Linear Discriminant Analysis
  - t-SNE — t-distributed Stochastic Neighbor Embedding

# Dimensionality Reduction — Motivation

- High number of dimensions = high number of data features
  - $m$  ("mass") — number of data points
  - $V$  ("volume") — data space described by dimensions

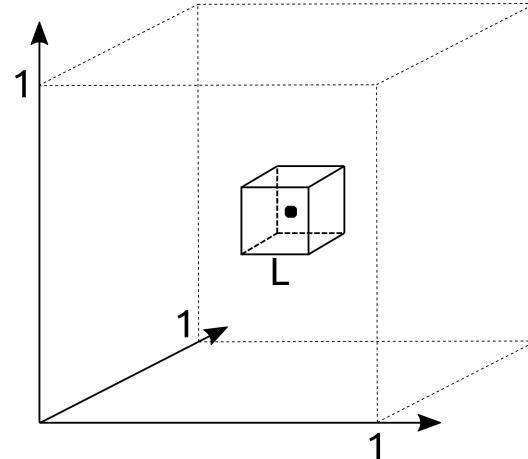
$$\text{density } \rho = \frac{m}{V} \qquad \rightarrow \text{High-dimensional data quickly becomes very sparse!}$$

- Effects of many features and data sparsity
  - Higher computational cost
  - Problematic for any method that requires statistical significance → high risk of overfitting ("good" data points and noise/outliers look more and more indistinguishable)
  - Obscures similarities between data points  
(points similar in low dimensions but not in high dimensions)

# Curse of Dimensionality

- **Effect of high dimensions** (i.e., many features)
  - Data points tend to never be close together
  - Average distance between points converges
- **Intuition**
  - Assume  $N$  data points uniformly distributed within a unit cube with  $d$  dimensions
  - Let  $L$  be the length of the smallest cube containing the  $k$ -NN of a data point

$$L^d \approx \frac{k}{N} \Rightarrow L \approx \left(\frac{k}{N}\right)^{\frac{1}{d}}$$



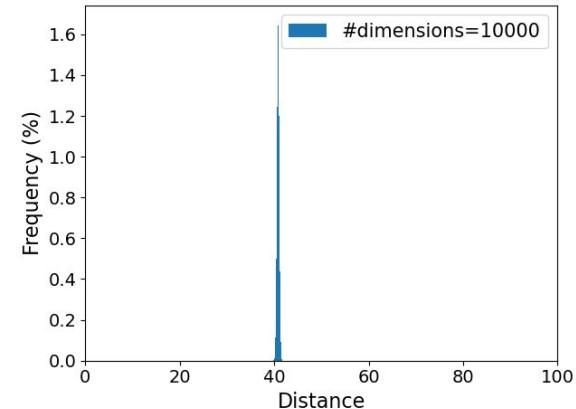
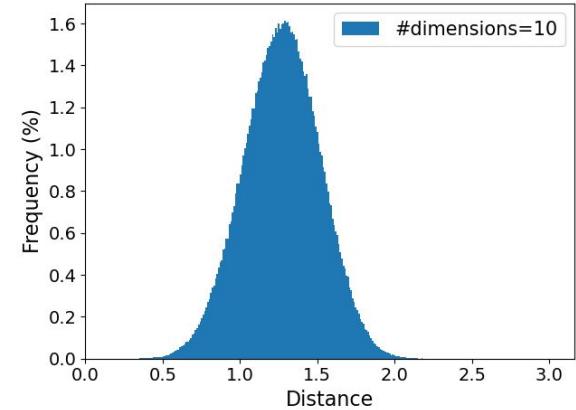
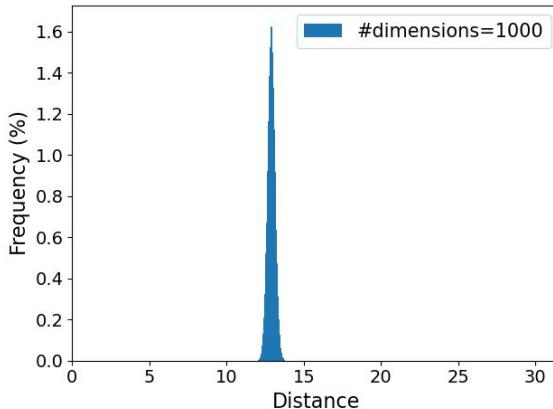
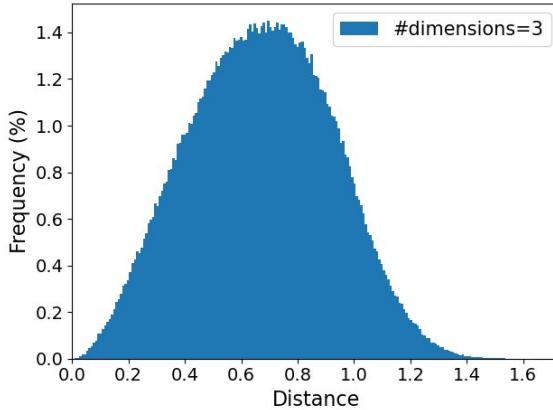
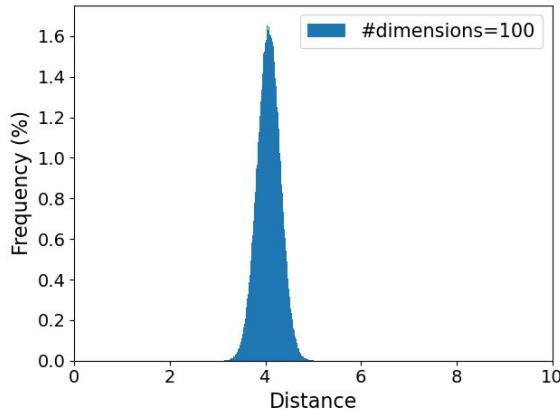
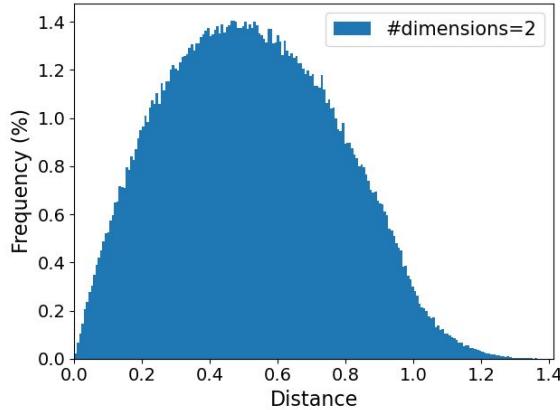
N=1,000, k=10

$d$	$L$
2	0.100
3	0.215
10	0.631
100	0.955
1,000	0.995
10,000	0.999

The cube with the  
k-NN is almost the  
whole unit cube!

# Curse of Dimensionality

Distribution of pairwise distances between 1,000 random data points and different number of dimensions



# Dimensionality Reduction — Feature Selection

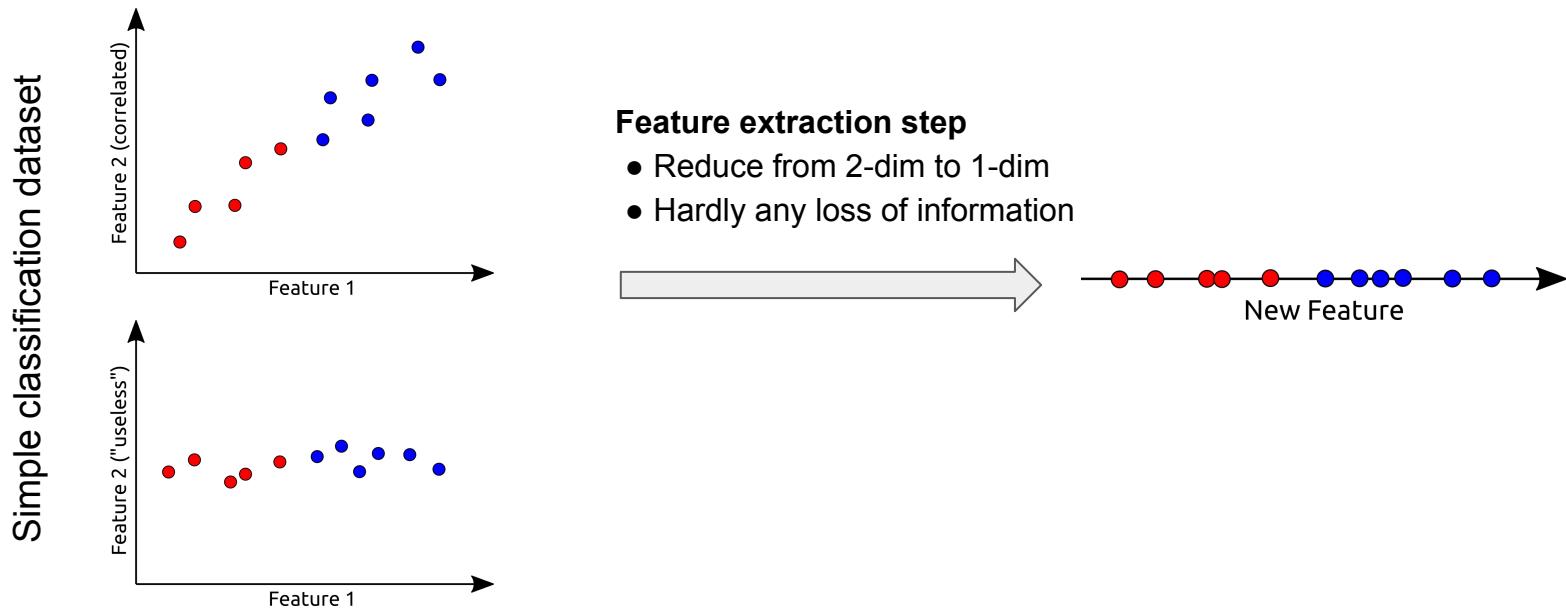
- Feature Selection — Removal of features before analysis  
(alternatively: keep only a certain subset of features)
- Common feature selection
  - Remove "unimportant" features based on expert knowledge  
(e.g., birth date of a person is unlikely to affect his/her spending behavior)
  - Remove features that might introduce ethical biases  
(e.g., sexual orientation and ethnicity for credit card approval)
  - Remove features with very low variance → not useful
  - Remove features that are strongly correlated with other features → not needed  
(basic method: calculate pairwise correlations and remove one of the features in case of high correlation)
  - Remove features with low ranks w.r.t. to their power to discriminate data points  
(basic approach of Decision Trees where feature near the root node yield purer subtrees)

# Feature Selection — Pros & Cons

- Pros
  - Relatively straightforward to implement
  - Does not change features themselves → effects on analysis results are preserved
- Cons
  - Selecting important features based on expert knowledge often not trivial/obvious
  - Finding meaningful thresholds — e.g., min. variance or correlation — not obvious

# Dimensionality Reduction — Feature Extraction

- Feature Extraction — basic idea
  - Generate new features as form of summary of original features
  - Feature extraction algorithms utilize discriminatory power and correlations among features



# Outline

- Dimensionality Reduction
  - Motivation
  - Naive approaches
- Dimensionality Reduction Techniques
  - PCA — Principal Component Analysis
  - LDA — Linear Discriminant Analysis
  - t-SNE — t-distributed Stochastic Neighbor Embedding

# Principal Component Analysis (PCA)

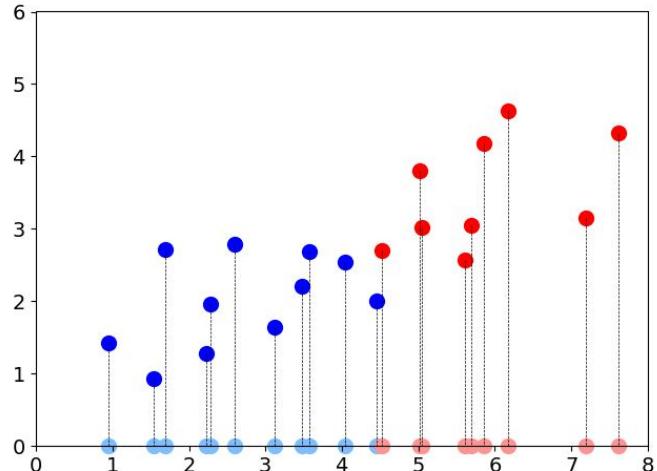
- PCA — Dimensionality reduction through linear transformation
  - New output features are a linear combination of original input features
  - Transforms data to a new coordinate systems
  - Unsupervised approach (independent from any kind of class labels)
- Basic setup
  - Dataset  $X$  ( $n$  samples,  $d$  features)
  - Find matrix  $W$  to transform  $X$  into a  $p$ -dimensional dataset ( $p < d$ )

$$\begin{bmatrix} X \\ n \times d \end{bmatrix} \begin{bmatrix} W \\ d \times p \end{bmatrix} = \begin{bmatrix} P \\ n \times p \end{bmatrix}$$

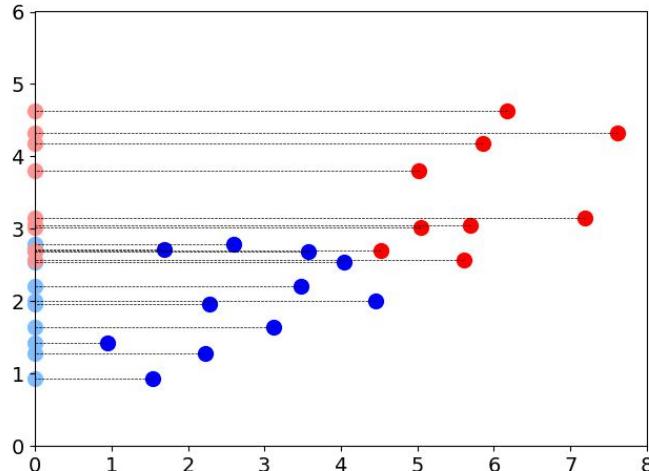
→ What makes a good transformation matrix  $W$  and how to find it?

# PCA — Intuition Using Naive Transformations

Mapping of data to x-axis:  $W = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$



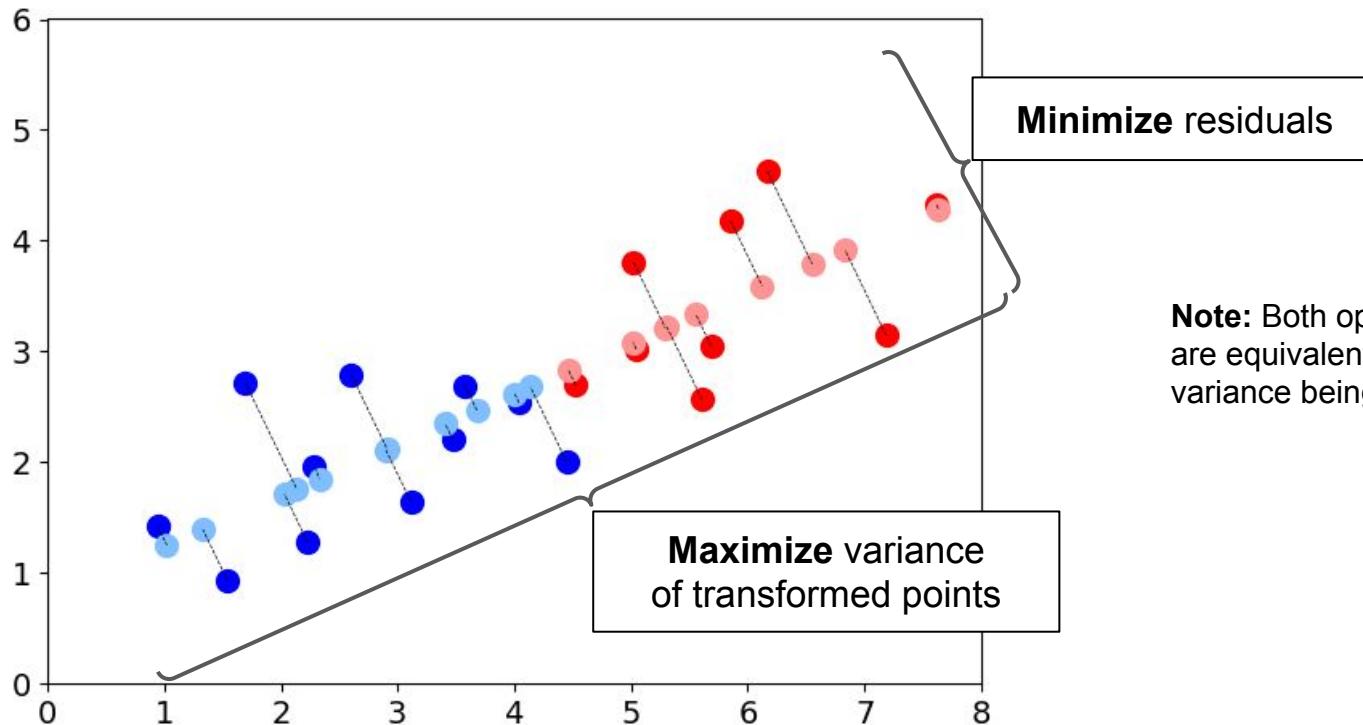
Mapping of data to x-axis:  $W = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$



Which transformation is the better one? Why?

Is there an even better transformation?

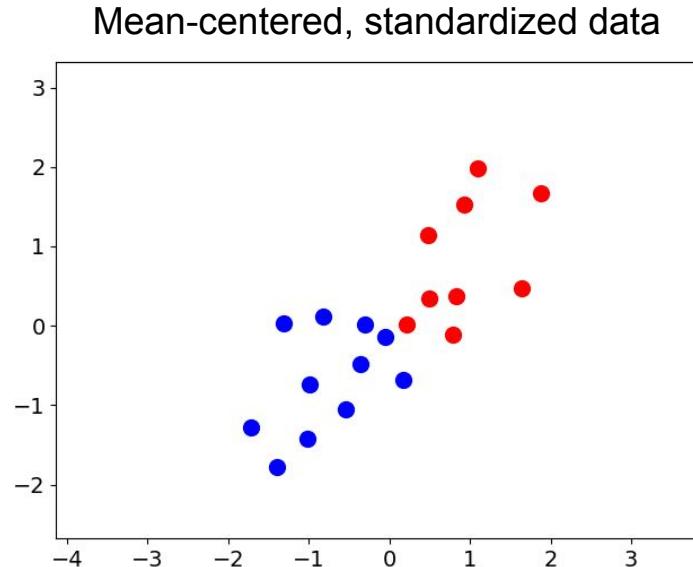
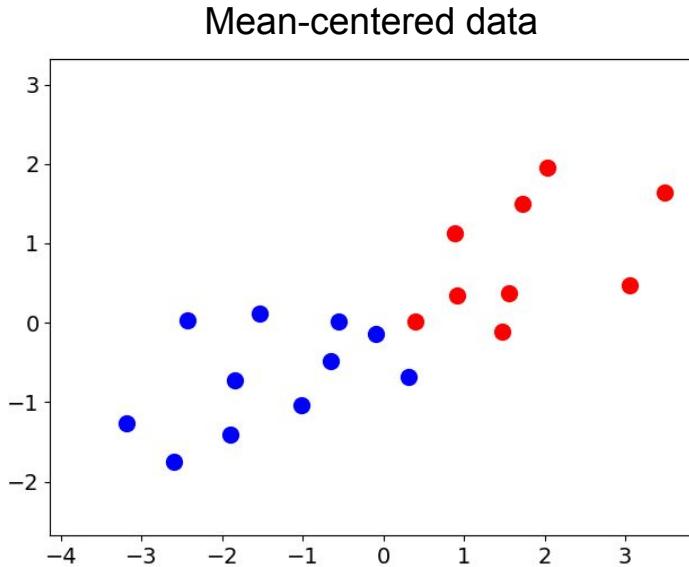
# PCA — Equivalent Objectives for Finding Transformation



**Note:** Both optimization objectives are equivalent, with maximizing the variance being easier to handle

# PCA — Data Normalization

- Data normalization steps
  - Mean-centering — does not affect results but makes the math much easier
  - Standardizing (divide by standard deviation) — optional; will affect the results



# PCA — Finding the 1st Principal Component

- 1st Principal Component of data  $X$

- Unit vector  $w_1$  that maximizes the variance of transformed data

$$w_1 = \underset{\|w\|=1}{\operatorname{argmax}} \underbrace{\frac{1}{n} \sum_i (p_i - 0)^2}_{\text{Variance of transformed data}}$$

$p_i$  — data point  $x_i$  after transformation

0 because of mean-centered data

# PCA — Finding the 1st Principal Component

- 1st Principal Component of data  $X$

- Unit vector  $w_1$  that maximizes the variance of transformed data

$$w_1 = \underset{\|w\|=1}{\operatorname{argmax}} \underbrace{\frac{1}{n} \sum_i (p_i - 0)^2}_{\text{Variance of transformed data}}$$

$p_i$  — data point  $x_i$  after transformation

0 because of mean-centered data

$$\begin{aligned} w_1 &= \underset{\|w\|=1}{\operatorname{argmax}} \frac{1}{n} \sum_i (x_i \cdot w)^2 \\ &= \underset{\|w\|=1}{\operatorname{argmax}} \frac{1}{n} \|Xw\|^2 \\ &= \underset{\|w\|=1}{\operatorname{argmax}} \frac{1}{n} w^T X^T X w \\ &= \underset{\|w\|=1}{\operatorname{argmax}} w^T \frac{X^T X}{n} w \\ &= \underset{\|w\|=1}{\operatorname{argmax}} w^T C_X w \end{aligned}$$

→  $C_X$  is the covariance matrix of  $X$

**Note:** Sometimes  $C_X = X^T X$  (instead of  $C_X = X^T X/n$ ). In this case  $C_X$  is the unnormalized covariance matrix (scatter matrix). This only affects the magnitude of the eigenvalues but the eigenvectors for  $W$ .

# PCA — Finding the 1st Principal Component

$$w_1 = \underset{\|w\|=1}{\operatorname{argmax}} w^T C_X w$$

$$= \underset{\|w\|=1}{\operatorname{argmax}} \frac{w^T C_X w}{w^T w}$$

Note that  $w^T w = \|w\| = 1$

*Rayleigh Quotient*

→  $w_1$  is the largest eigenvector of the covariance matrix  $C_X$

# PCA — Finding the k-th Principal Component

- Subtract (k-1) principal components from  $X$

$$X_k = X - \sum_{s=1}^{k-1} X w_s w_s^T$$

- k-th Principal Component of data  $X_k$

- Unit vector  $w_k$  that maximizes the variance of transformed data — after transforming  $X_k$

$$w_k = \underset{\|w\|=1}{\operatorname{argmax}} w^T \frac{X_k^T X_k}{n} w = \underset{\|w\|=1}{\operatorname{argmax}} w^T C_x^{(k)} w = \underset{\|w\|=1}{\operatorname{argmax}} \frac{w^T C_x^{(k)} w}{w^T w}$$

→  $w_k$  is the largest eigenvector of the covariance matrix  $C_x^{(k)}$

# PCA — The Math

$$\frac{dJ(w)}{dw} = \frac{\frac{d(w^T C w)}{dw} \cdot w^T w - w^T C w \cdot \frac{d(w^T w)}{dw}}{(w^T w)^2}$$

$$= \frac{w^T (C + C^T)}{w^T w} - \frac{w^T C w \cdot 2w}{(w^T w)^2}$$

$$= \frac{2Cw}{w^T w} - \frac{w^T C w \cdot 2w}{(w^T w)^2}$$

$$= \frac{2}{w^T w} \left( Cw - \frac{w^T C w}{w^T w} w \right)$$

$$J(w) = \frac{w^T C w}{w^T w}$$

Quotient Rule

$$\frac{x^T A x}{dx} = x^T (A + A^T)$$

$C$  is symmetric  $\rightarrow C + C^T = 2C$

scalar value!

$$\frac{dJ(w)}{dw} = \frac{2}{w^T w} (Cw - J(w)w) \stackrel{!}{=} 0 \quad \Leftrightarrow \quad Cw = \overbrace{J(w)w}^{}$$

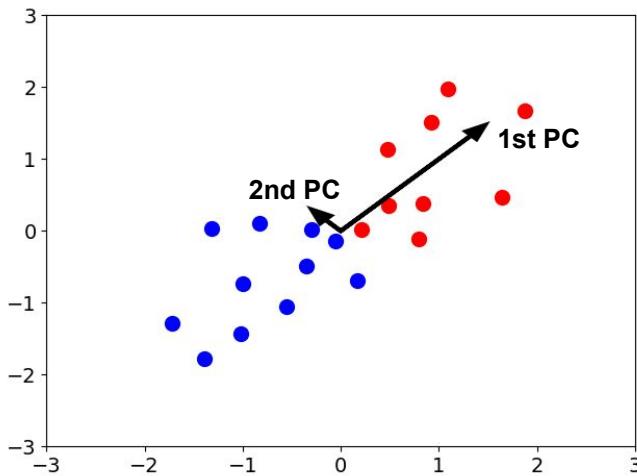
$\rightarrow$  Eigenvalue problem!

# PCA — Getting all Principal Components

- Mathematical convenience

- Largest eigenvector of  $C_X^{(k)}$  = k-largest eigenvector of  $C_X$

→ Principal Components of  $X$  = eigenvectors of covariance matrix  $C_X$



## Interpretation

- 1st PC points into the direction of maximum variance
- 2nd PC points into the direction of maximum variance after 1st PC removed from dataset  $X$
- ...

# PCA in Python Code (using numpy library)

```
1 def normalize(X):
2     X -= np.mean(X, 0) # Mean-center the data
3     X /= np.std(X, 0) # Standardize data (optional; in line with sklearn.decomposition.PCA)
4     return X
5
6
7 def covariance(X):
8     return np.dot(X.T, X) / X.shape[0] # Assumes mean-centered data matrix X
9
10
11 def pca(X, num_pc=None):
12     # Prepare data
13     X = normalize(X)
14     C = covariance(X)
15     # Calculate all eigenvectors and eigenvalues
16     eigenval, eigenvec = np.linalg.eigh(C)
17     # find the the num_pc largest eigenvalues
18     top_idx = np.argsort(eigenval)[::-1][:num_pc]
19     # Create transformation matrix W using eigenvectors with the largest eigenvalues
20     W = eigenvec[:, top_idx]
21     # Return the transformed data
22     return np.dot(X, W)
```

# PCA — Transforming Original Dataset X

- $C_x$  is a  $(d \times d)$  matrix  $\rightarrow d$  eigenvectors and eigenvalues
  - How to choose  $1 \leq p \leq d$  to get transformation matrix  $W$  of shape  $(d \times p)$ ?

- **Explained Variance Ratio**

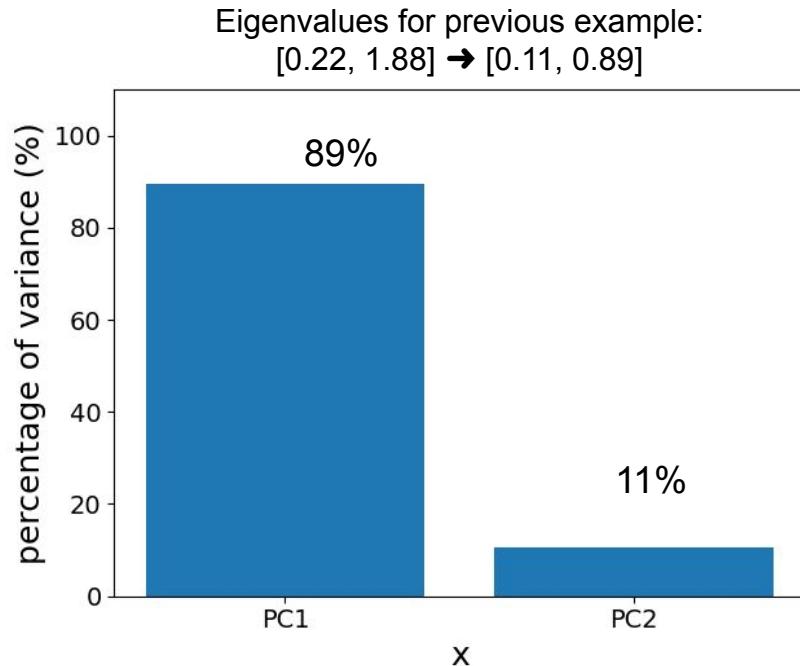
- Percentage of variance that is attributed by each principal component
  - normalized eigenvalues
- Choose  $p$  such that  $p$  largest PCs explain a minimum amount of variance

$$W = \begin{bmatrix} | \\ w_1 \\ | \end{bmatrix}$$

Using only PC1

$$W = \begin{bmatrix} | & | \\ w_1 & w_2 \\ | & | \end{bmatrix}$$

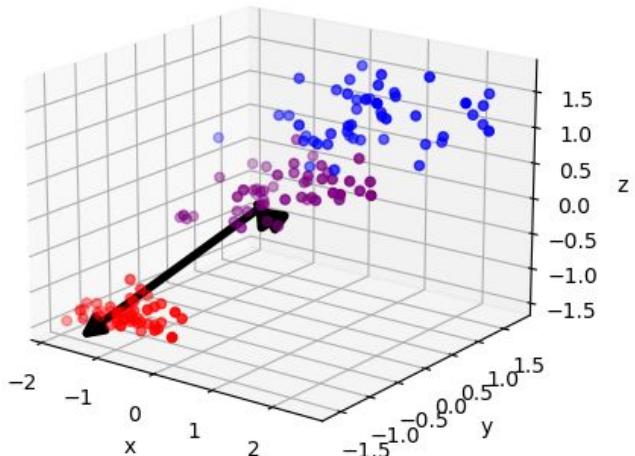
Using PC1 and PC2



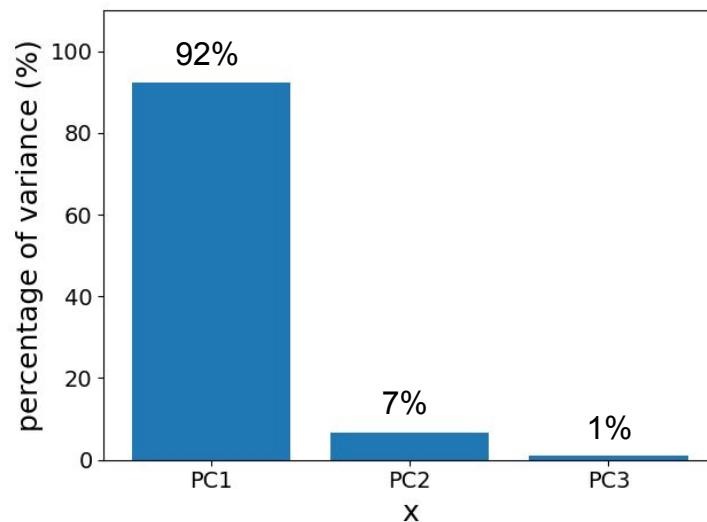
# PCA — Full Example (IRIS dataset)

- IRIS dataset
  - Only 3 (out of 4) features considered — only to allow for easy visualization

Dataset with the 3 principal components



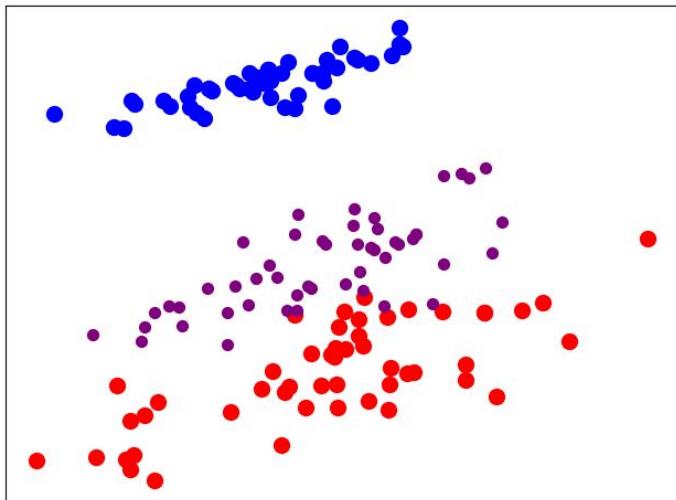
Explained variance of the 3 PCs



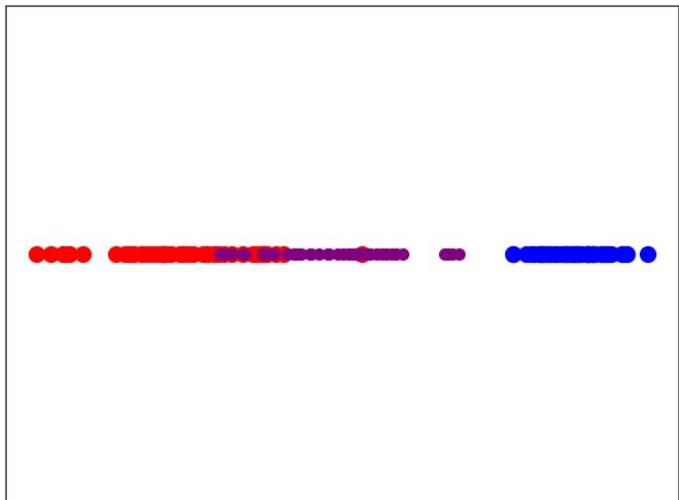
# PCA — Full Example (IRIS dataset)

- Transformation of  $X$  using principal components

Using PC1 and PC2  
(99% of variance explained)



Using only PC1  
(92% of variance explained)



# PCA — Pros & Cons

- **Pros**

- Intuitive — exploit knowledge about correlated and low-variance features
- Can significantly reduce the amount of data
- Improves performance of algorithms and reduces risk of overfitting
- Visualization of high-dimensional data (even just when applied during EDA)

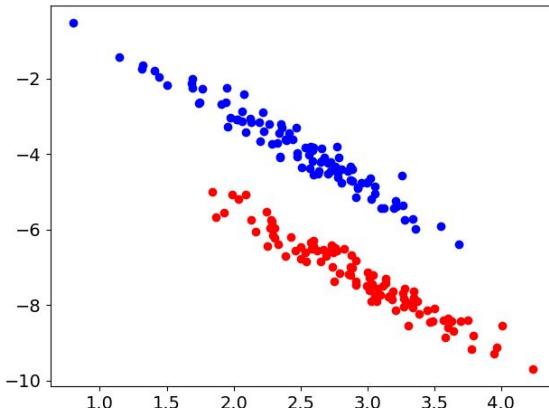
- **Cons**

- Most fundamentally: loss of information
- Assumes linear correlations among features
- Assumes that large variance equals high importance (does not always have to be the case)
- Does not take class labels into account (in case of classification tasks)

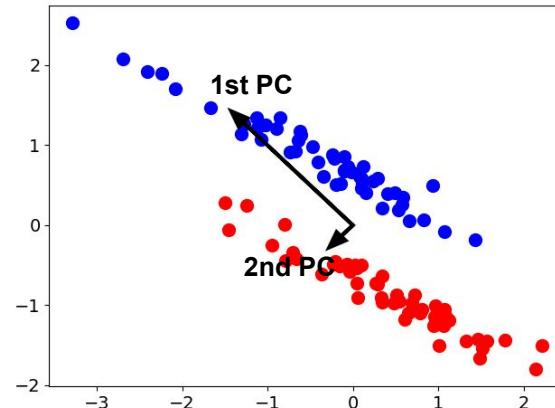
# PCA — Limitation for Classification Datasets

- PCA applied to labelled datasets for classification (pathological example)
  - PCA maximizes w.r.t. the variance of the whole dataset
  - PCA ignores any information from the class labels

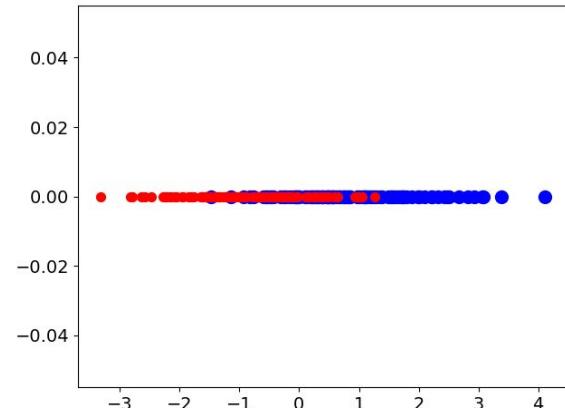
2-class dataset



principal components



transformation using 1st PC



# Outline

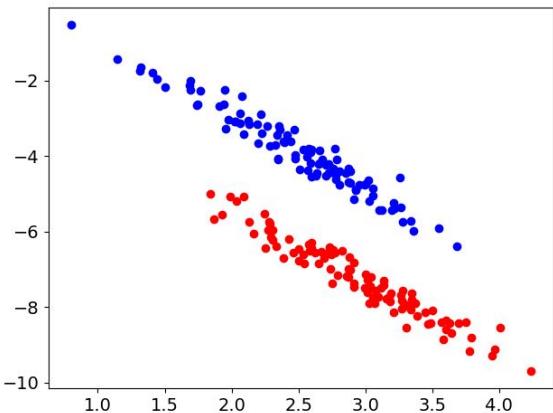
- Dimensionality Reduction
  - Motivation
  - Naive approaches
- Dimensionality Reduction Techniques
  - PCA — Principal Component Analysis
  - LDA — Linear Discriminant Analysis
  - t-SNE — t-distributed Stochastic Neighbor Embedding

# Linear Discriminant Analysis (LDA)

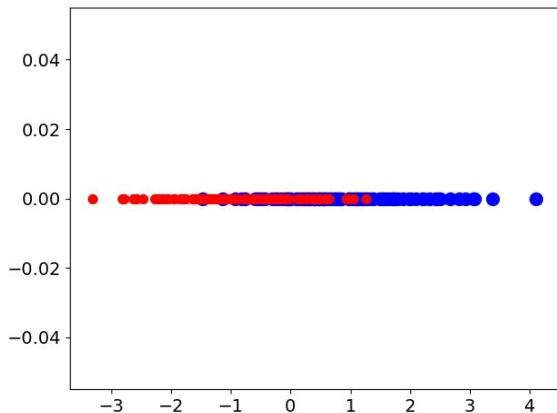
- Similarity to PCA
  - Linear transformation technique
  - Output: Matrix  $W$  to transform dataset  $X$  into a lower-dimensional space
- Main difference: 2 optimization objectives
  - Minimize variance of transformed points within each class  
(recall that PCA maximizes the variance across the whole dataset)
  - Maximize separation between classes

# LDA vs. PCA — Example

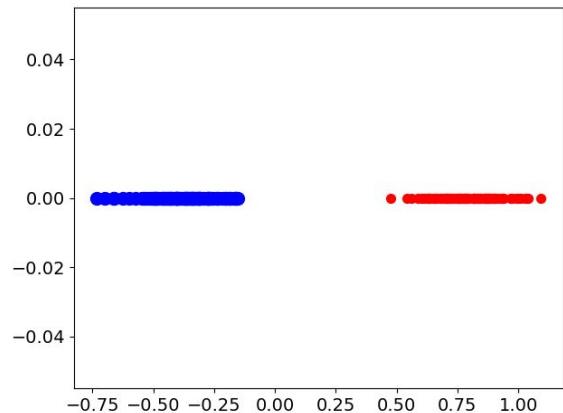
2-class dataset



PCA



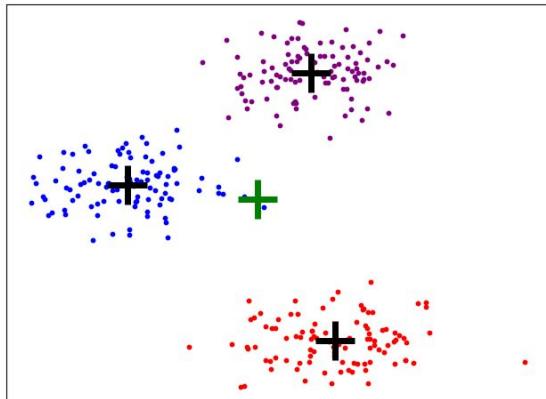
LDA



# LDA Concepts — Between-Class Variance

- Variance of distances between class means and (overall) mean

- $C$  — number of classes
- $\mu^{(i)}$  — class mean vector  
(mean of data points of class  $i$ )
- $\mu$  — (overall) mean vector  
(mean of all data points)



$$\begin{aligned} [m^{(i)} - m]^2 &= [w^T \mu^{(i)} - w^T \mu]^2 \\ &\quad \text{projected mean vectors} \\ &= w^T \underbrace{[\mu^{(i)} - \mu] [\mu^{(i)} - \mu]^T}_{\text{scatter of class means}} w \\ &= w^T S_B^{(i)} w \\ &\quad \downarrow \text{over all classes} \\ &= w^T S_B w, \text{ with } S_B = \sum_{i=1}^C n^{(i)} S_B^{(i)} \end{aligned}$$

# LDA ( $S_B$ ) in Python Code (using numpy library)

```
1 def calc_S_b(X, y):
2     C = np.unique(y) # C = set of class labels
3     d = X.shape[1]   # d = number of features
4
5     mu = np.mean(X, axis=0) # Calculate overall mean
6
7     S_b = np.zeros((d, d)) # Initialize S_b matrix
8
9     for c in C:
10         # n_i number of samples labeled c
11         n_i = X[y==c,:].shape[0]
12         # mu_i mean of samples labeled c
13         mu_i = np.mean(X[y==c], axis=0)
14         # Calculate difference vector
15         # (= mean-centering class means)
16         diff = (mu_i - mu).reshape(-1, 1)
17         # Add scatter of difference vector
18         S_b += n_i * np.dot(diff, diff.T)
19
20     return S_b
21
```

$$= w^T \underbrace{[\mu^{(i)} - \mu] [\mu^{(i)} - \mu]^T}_\text{scatter of class means} w$$

$$= w^T S_B^{(i)} w$$

 over all classes

$$= w^T S_B w, \text{ with } S_B = \sum_{i=1}^C n^{(i)} S_B^{(i)}$$

# LDA Concepts — Within-Class Variance

- Variance of data points of the same class

$$\begin{aligned} \sum_{j=1}^{n^{(i)}} \left[ p_j^{(i)} - m^{(i)} \right]^2 &= \sum_{j=1}^{n^{(i)}} \left[ w^T x_j^{(i)} - w^T \mu^{(i)} \right]^2 \\ &= \sum_{j=1}^{n^{(i)}} w^T \underbrace{\left[ x_j^{(i)} - \mu^{(i)} \right] \left[ x_j^{(i)} - \mu^{(i)} \right]^T}_\text{scatter of data points of class } i w \\ &= w^T S_W^{(i)} w \\ &\quad \downarrow \text{over all classes} \\ &= w^T S_W w, \text{ with } S_W = \sum_{i=1}^C S_W^{(i)} \end{aligned}$$

# LDA ( $S_W$ ) in Python Code (using numpy library)

```
1 def calc_S_w(X, y):
2     C = np.unique(y)    # C = set of class labels
3     d = X.shape[1]      # d = number of features
4
5     S_w = np.zeros((d, d))  # Initialize S_w matrix
6
7     for c in C:
8         # Get all samples labeled c
9         X_c = X[y==c]
10        # Mean-center the data
11        X_c -= np.mean(X_c, 0)
12        # Add scatter matrix of X_c
13        S_w += np.dot(X_c.T, X_c)
14
15    return S_w
```

$$= \sum_{j=1}^{n(i)} w^T [x_j^{(i)} - \mu^{(i)}] [x_j^{(i)} - \mu^{(i)}]^T w$$

$$= w^T S_W^{(i)} w$$

↓ over all classes

$$= w^T S_W w, \text{ with } S_W = \sum_{i=1}^C S_W^{(i)}$$

# LDA — Optimization Objective

Maximize:  $J(w) = \frac{w^T S_B w}{w^T S_W w}$

scatter of projected class means  
scatter of projected data points (per class)

*Generalized Rayleigh Quotient*

→ Generalized eigenvalue problem:  $S_W^{-1} S_B w = J(w)w$

scalar value!

Optimal projection vectors =

eigenvectors of largest the eigenvalues of matrix  $S_W^{-1} S_B$

# LDA — The Math

$$J(w) = \frac{w^T A w}{w^T B w}$$

$$\frac{dJ(w)}{dw} = \frac{\frac{d(w^T A w)}{dw} \cdot w^T B w - w^T A w \cdot \frac{d(w^T B w)}{dw}}{(w^T B w)^2}$$

Quotient Rule

$$= \frac{w^T (A + A^T)(w^T B w) - w^T (B + B^T)(w^T A w)}{(w^T B w)^2}$$

$$\frac{x^T A x}{dx} = x^T (A + A^T)$$

$$= \frac{2Aw(w^T B w) - 2Bw(w^T A w)}{(w^T B w)^2}$$

$X$  is symmetric  $\rightarrow X + X^T = 2X$

$$\frac{dJ(w)}{dw} = \frac{2}{w^T B w} (Aw - J(w)Bw) \stackrel{!}{=} 0 \quad \Leftrightarrow \quad \begin{aligned} Aw &= \overbrace{J(w)Bw}^{\text{scalar value!}} \\ B^{-1}Aw &= J(w)w \end{aligned}$$

→ Generalized Eigenvalue problem!

# Note on the Number of Eigenvectors

- Definition of  $S_B$  includes two constraints

- $S_B$  is the sum of  $C$  matrices of rank 1 or less

- The mean  $\mu$  is constraint by  $\mu = \frac{1}{C} \sum_{i=1}^C \mu_i$

→  $S_B$  has rank of  $(C-1)$  or less

→ only  $(C-1)$  eigenvectors are non-zero! (the respective eigenvalues are 0)

**Note:** In practice, these remaining  $d-c+1$  eigenvalues are only very close to zero due to floating pointing imprecisions

# LDA — Algorithm

1. Calculate mean vectors  $\mu$  and  $\mu^{(i)}$  for all C classes
2. Calculate scatter matrices  $S_w$  and  $S_B$
3. Calculate eigenvectors and eigenvalues of  $S_w^{-1}S_B$
4. Select  $p$  eigenvectors  $w_p$  with the largest eigenvectors

$$\mathbf{W} = \begin{bmatrix} | & | & \dots & | \\ w_1 & w_2 & \dots & w_p \\ | & | & \dots & | \end{bmatrix} \quad \text{with } p \leq C-1$$

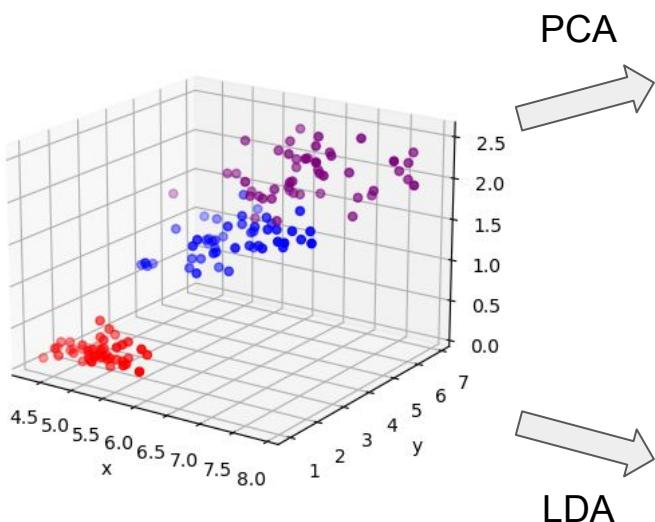
5. Project dataset X into new space via  $XW$

# LDA in Python Code (using numpy library)

$$S_W^{-1} S_B$$

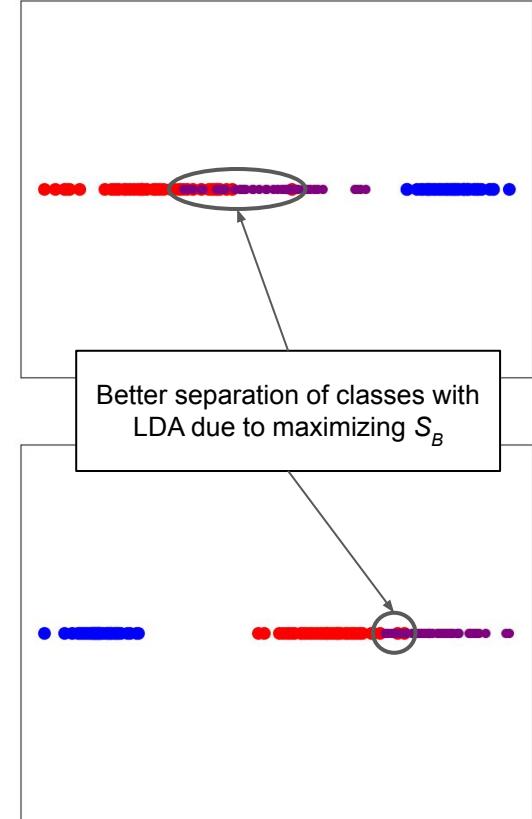
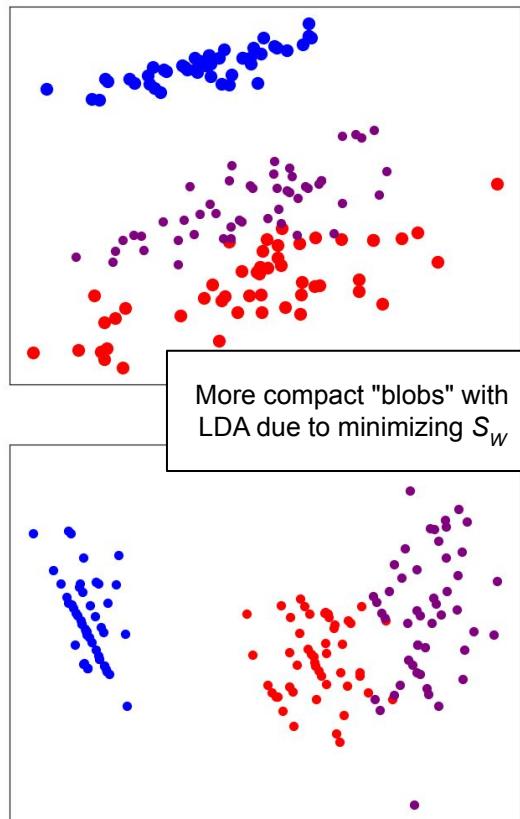
```
1 def lda(X, y, num_pc=None):
2     # Calculate scatter matrices
3     S_w = calc_S_w(X, y)
4     S_b = calc_S_b(X, y)
5     # Calculate all eigenvectors and eigenvalues
6     eigenval, eigenvec = np.linalg.eig(np.linalg.inv(S_w).dot(S_b))
7     # find the the num_pc largest eigenvalues
8     top_idx = np.argsort(eigenval)[::-1][:num_pc]
9     # Create transformation matrix W using eigenvectors with the largest eigenvalues
10    W = eigenvec[:, top_idx]
11    # Return the transformed data
12    return np.dot(X, W)
```

# LDA — Full Example (IRIS dataset)



PCA

LDA



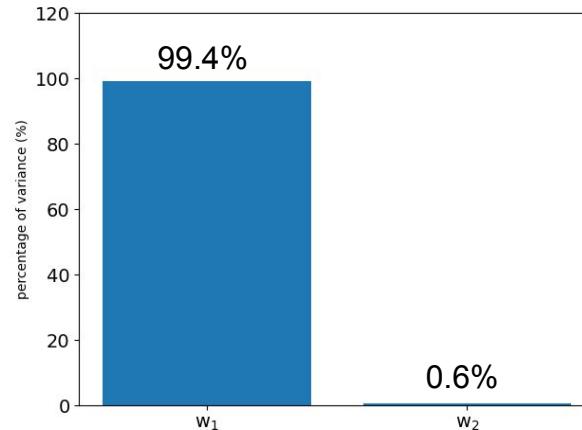
# LDA — Full Example (IRIS dataset)

- Resulting  $d$  eigenvalues (sorted)

1.	26.9
2.	0.17
3.	$2.03e^{-15}$

$\left. \right\} C-1 = 3-1 = 2$  non-zero eigenvalues  
 $\left. \right\} d-C+1 = 3-3+1 = 1$  zero eigenvalues

- Choosing  $p \leq C-1$  using Explained Variance Ratio



→  $p=1$  a good choice

# LDA — Pros & Cons

- Pros

- Intuitive extension to PCA yielding similar benefits  
(reduction in amount of data, reduced risk of overfitting, visualization, etc.)
- Consideration of class labels (generally more suitable than PCA for labelled dataset)

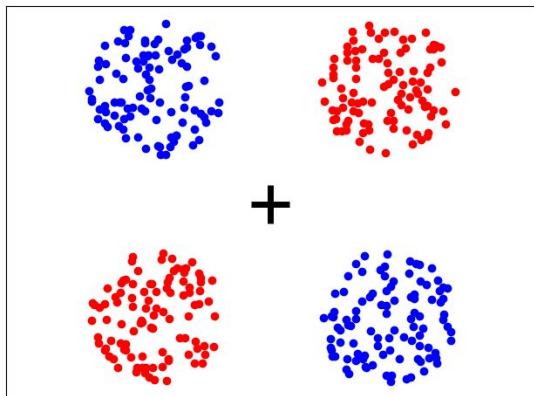
- Cons

- Similar cons as PCA (loss of information, assumes linear correlations, etc.)
- Assumes unimodal Gaussian distributions
- Assumes that means are the most discriminant features

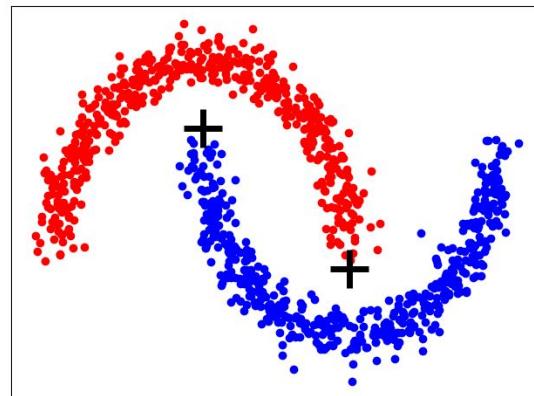
# LDA — Problematic Cases

- Data distributions that are (significantly) non-unimodal Gaussian

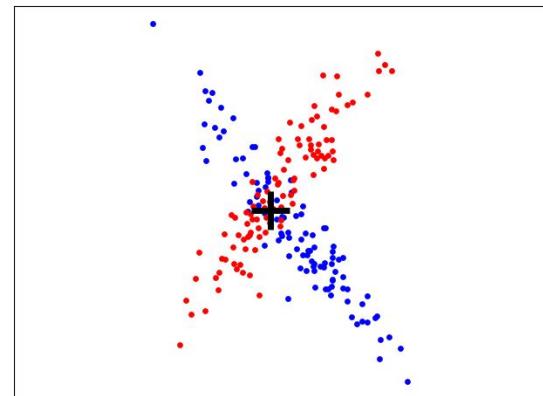
bimodal with  $\mu = \mu_1 = \mu_2$



non-Gaussian



unimodal but  $\mu = \mu_1 = \mu_2$



# Outline

- Dimensionality Reduction
  - Motivation
  - Naive approaches
- Dimensionality Reduction Techniques
  - PCA — Principal Component Analysis
  - LDA — Linear Discriminant Analysis
  - t-SNE — t-distributed Stochastic Neighbor Embedding

# t-Distributed Stochastic Neighbor Embedding (t-SNE)

- t-SNE — Non-linear dimensionality reduction technique
  - Unsupervised approach (independent from any kind of class labels)
  - Iterative algorithm:
    - 1) Start with random lower-dimensional representation  $Y$
    - 2) Change  $Y$  until a loss function converges to a minimum
- Intuition behind t-SNE
  - Convert Euclidean distances in  $X$  and  $Y$  to conditional probabilities  
(e.g., if data points  $x_i$  and  $x_j$  are close  $\rightarrow$  conditional probability  $p_{ij}$  should be high)
  - Iteratively change  $Y$  such that both probability distributions become more similar
- Optimization objective: Points that are close in  $X$  are also close in  $Y$

# t-SNE — Convert Euclidean Distances to Cond. Probs

- Mathematical formulation
  - For data points in  $X$

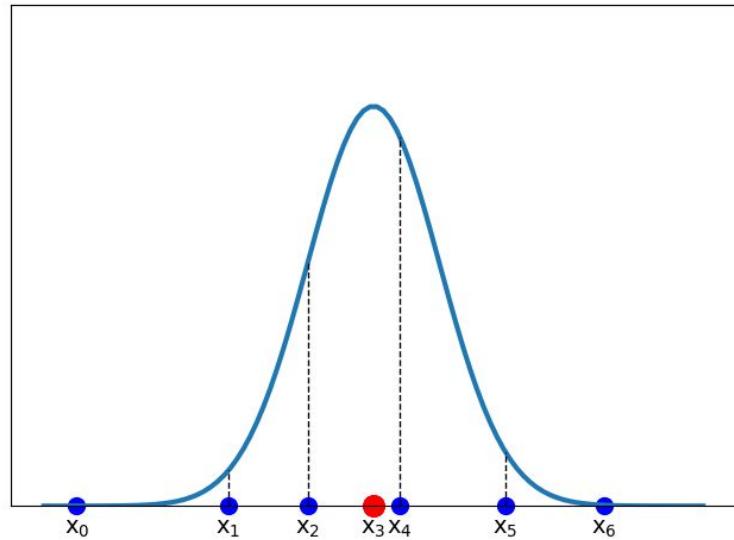
$$p_{j|i} = \frac{\exp(-\|x_i - x_j\|^2 / 2\sigma_i^2)}{\sum_{k \neq i} \exp(-\|x_i - x_k\|^2 / 2\sigma_i^2)}$$

- Visual explanation (for  $i = 3$ )
  - Assume  $d$ -dim Gaussian centered at  $x_3$
  - Calculate  $p_{j|3}$  proportional to Gaussian (proportional to heights of dashed lines from each point  $x_j$ )

$p_{4 3}$	$p_{2 3}$	$p_{5 2}$	$p_{1 3}$	$p_{6 3}$	$p_{0 3}$
0.84	0.16	~0.0	~0.0	~0.0	~0.0

## Interpretation

$p_{j|i}$  is the probability that  $x_i$  would pick  $x_j$  as neighbor



# t-SNE

- Calculate joint probabilities

$$p_{ij} = \frac{p_{j|i} + p_{i|j}}{2n}$$

Joint probabilities  $P = \{p_{ij}\}$  for previous example

$$\begin{bmatrix} 0 & 0.071 & 0 & 0 & 0 & 0 & 0 \\ 0.071 & 0 & 0.09 & 0 & 0 & 0 & 0 \\ 0 & 0.09 & 0 & 0.057 & 0.09 & 0 & 0 \\ 0 & 0 & 0.057 & 0 & 0.129 & 0.001 & 0 \\ 0 & 0 & 0.009 & 0.129 & 0 & 0.025 & 0 \\ 0 & 0 & 0 & 0.001 & 0.025 & 0 & 0.117 \\ 0 & 0 & 0 & 0 & 0 & 0.117 & 0 \end{bmatrix}$$

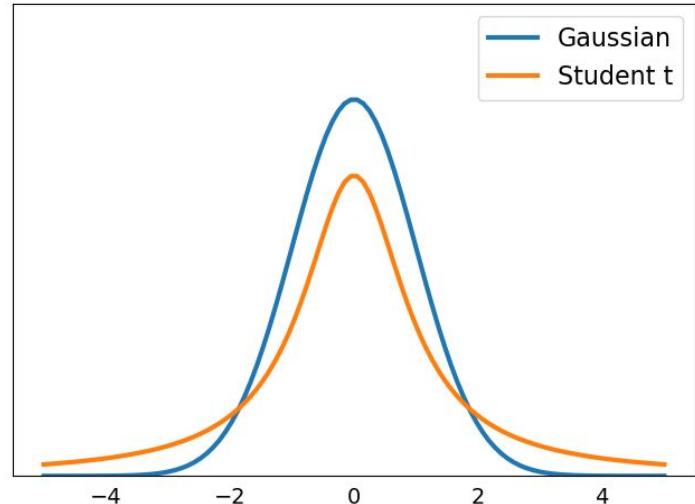
Assume  $Q = \{q_{ij}\}$  being the joint probabilities of data points  $y_i$  in  $Y$

→ How to modify all  $y_i$  such that  $P \approx Q$ ?

# t-SNE

- t-SNE uses a Student t distribution
  - Heavier tails yield better results in practice
  - 1 degree of freedom

$$q_{ij} = \frac{(1 + \|y_i - y_j\|^2)^{-1}}{\sum_{k \neq l} (1 + \|y_i - y_j\|^2)^{-1}}$$



- Loss function: Kullback-Leibler divergence between P and Q

$$L = KL(P||Q) = \sum_i \sum_j p_{ij} \log \frac{p_{ij}}{q_{ij}}$$

**Note:** The KL divergence is a measure of how one probability distribution is different from another probability distribution

# t-SNE — Minimizing Loss

- Minimize  $L$  using Gradient Descent

$$L = KL(P||Q) = \sum_i \sum_j p_{ij} \log \frac{p_{ij}}{q_{ij}}$$

$$\frac{\partial L}{\partial y_i} = \dots = 4 \sum_j (p_{ij} - q_{ij})(y_i - y_j)(1 + \|y_i - y_j\|^2)^{-1}$$

# t-SNE — Basic Algorithm

**Input** : Dataset  $X$ , number of iterations  $T$ , learning rate  $\eta$

**Initialization** : Sample  $Y^{(0)} = \{y_1, y_2, \dots, y_n\}$  from  $\mathcal{N}(0, 10^{-4}\mathbf{I})$

Calculate  $p_{j|i}$  and  $p_{ij}$  for all  $(x_i, x_j)$ -pairs

**for**  $t = 1$  **to**  $T$  **do**

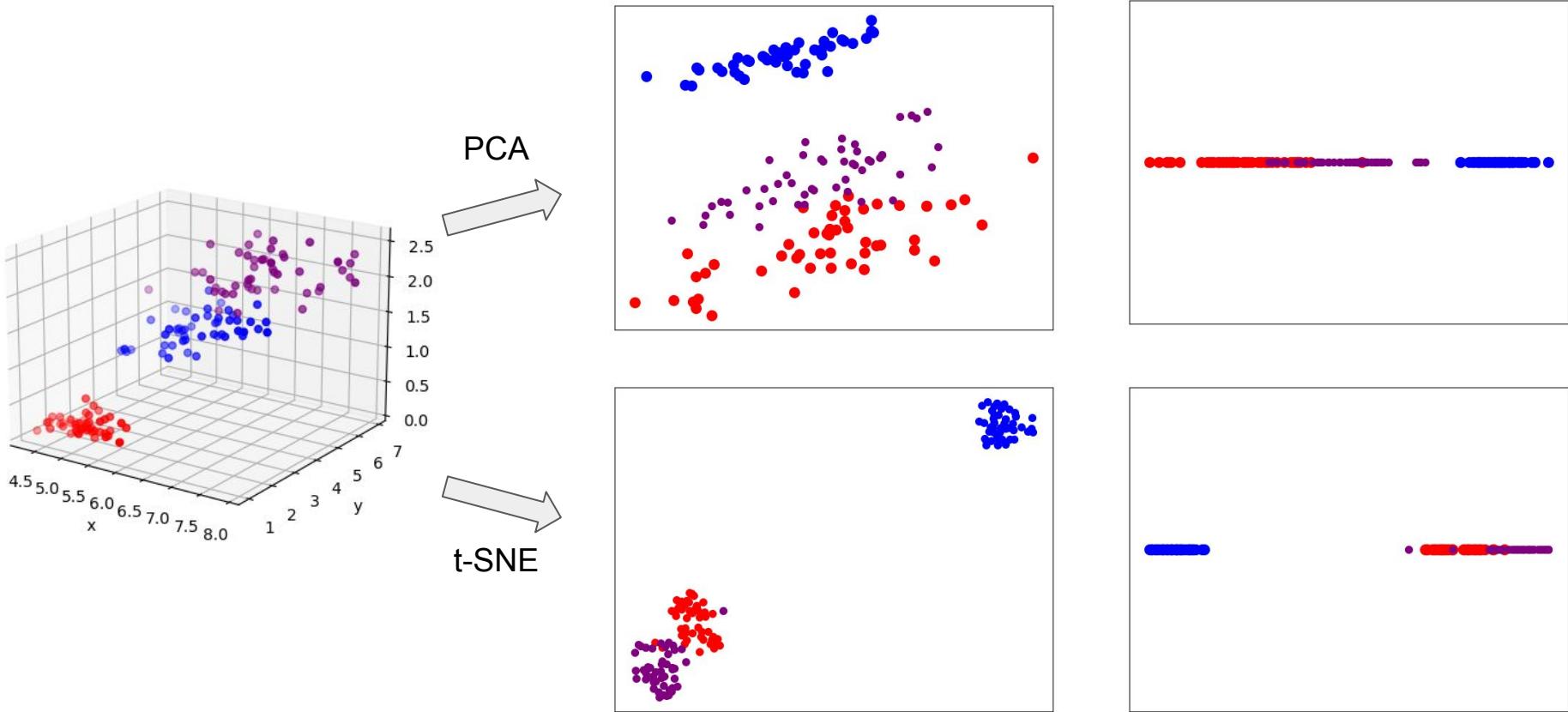
    Calculate  $q_{ij}$  for all  $(y_i, y_j)$ -pairs

    Calculate gradients  $\frac{\partial L}{\partial y_i}$

    Update  $y_i^t \leftarrow y_i^{t-1} - \eta \frac{\partial L}{\partial y_i}$

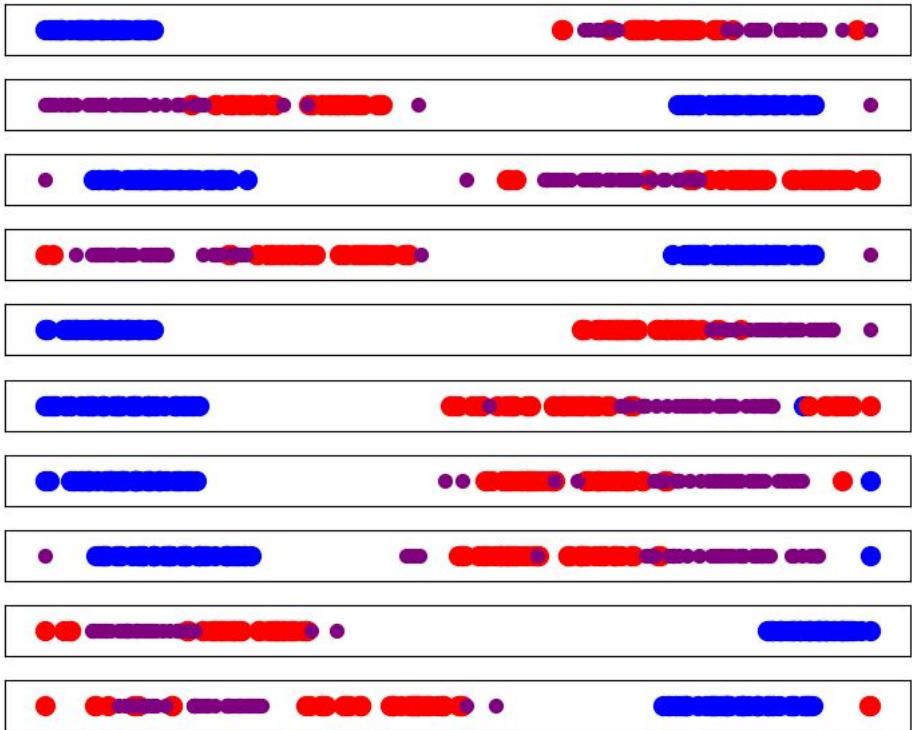
**return**  $Y^{(T)}$

# t-SNE — Full Example (IRIS dataset)



# t-SNE — Non-Determinism

- t-SNE is non-deterministic
  - $Y^{(0)}$  is randomly sampled
  - Different runs will generally yield different projections
  - In practice, perform multiple runs to get an understanding of the data

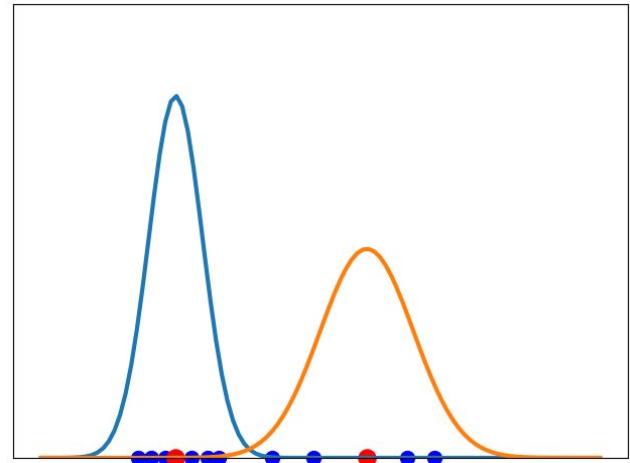


# t-SNE — Calculating $p_{j|i}$ (implementation detail)

- How to pick the values for  $\sigma_i$ ?

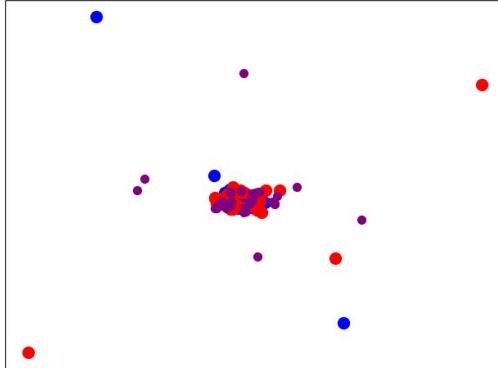
$$p_{j|i} = \frac{\exp(-\|x_i - x_j\|^2 / 2\sigma_i^2)}{\sum_{k \neq i} \exp(-\|x_i - x_k\|^2 / 2\sigma_i^2)}$$

- Intuition: Set  $\sigma_i$  based on density around  $x_i$ 
  - High density  $\rightarrow$  smaller  $\sigma_i$  / Low density  $\rightarrow$  larger  $\sigma_i$
  - Controls how many  $x_j$  with effective  $p_{j|i}$
- Calculate best based  $\sigma_i$  on hyperparameter **perplexity**
  - Larger perplexity: more neighbors have effective  $p_{j|i}$
  - Common perplexity values in practice: 5..50

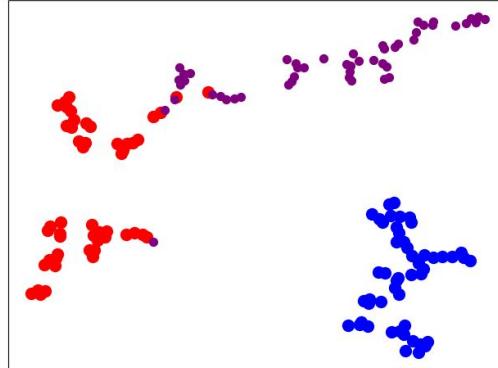


# t-SNE — Effects of Perplexity Parameter

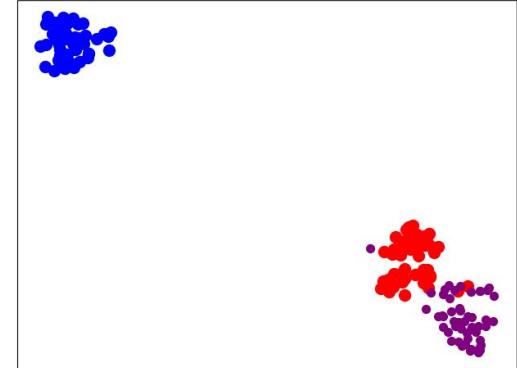
perplexity = 1



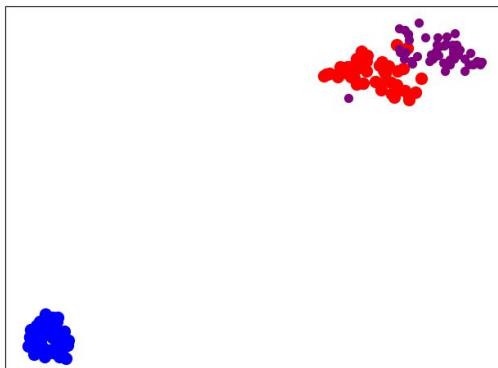
perplexity = 5



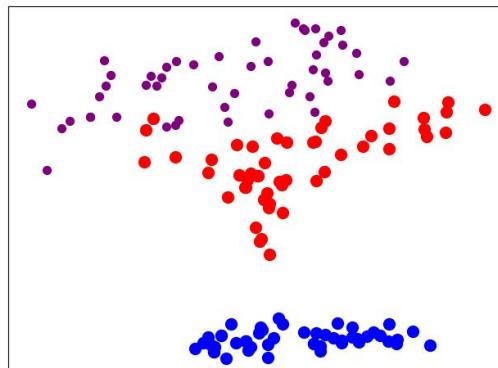
perplexity = 25



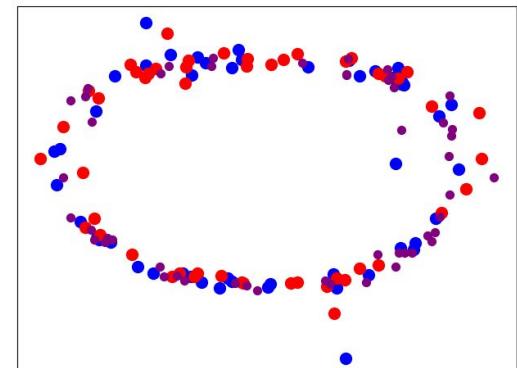
perplexity = 50



perplexity = 100



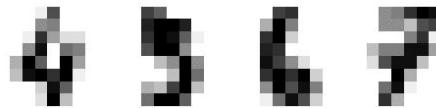
perplexity = 200



# t-SNE — Pros & Cons

- Pros
  - Can handle non-linear data
  - Works very well for data visualization
- Cons
  - Computational very expensive on very high-dimensional data (compared to, e.g., PCA)
  - Non-deterministic behavior; might need multiple runs
  - Several hyperparameters affecting the output (perplexity, learning rate, number of iterations, initialization)

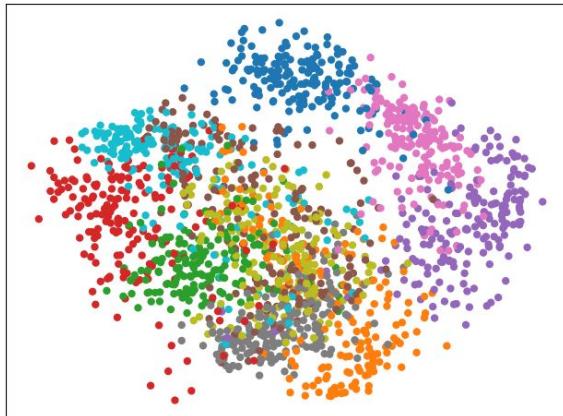
# Example — Digits Dataset



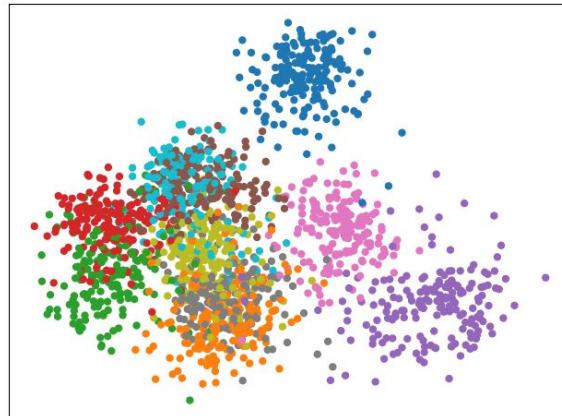
- **Digits Dataset**

- 1,797 handwritten digits 0, 1, 2, ..., 9  
(~180 samples for class)
- 8x8 pixels → 64 features  
(integer grayscale value 0..16)

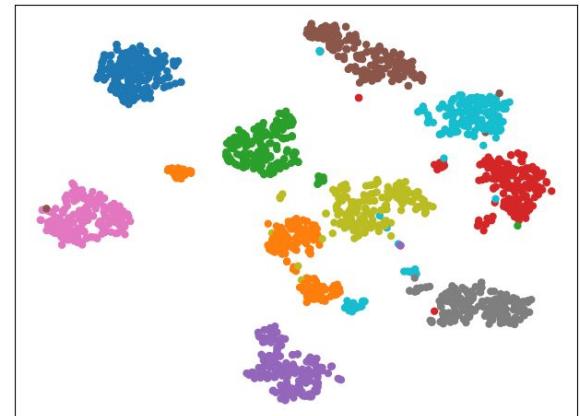
PCA



LDA



t-SNE



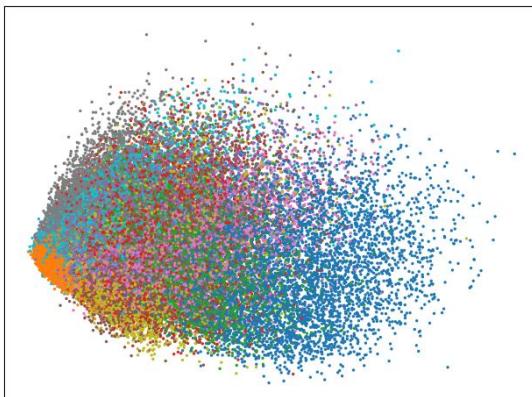
# Example — MNIST

- Digits Dataset

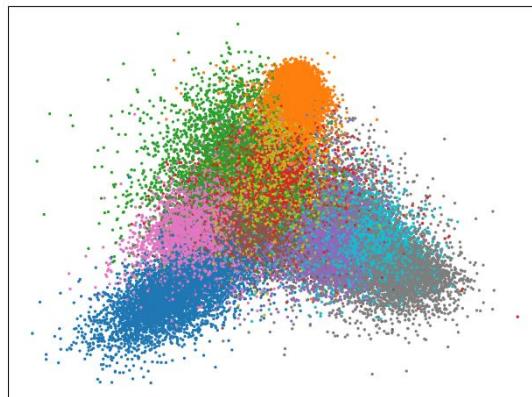
- 60k handwritten digits 0, 1, 2, ..., 9  
(~6k samples for class)
- 28×28 pixels → 784 features  
(integer grayscale values 0..255)

2 1 0 4 1 4  
9 0 6 9 0 1  
7 3 4 9 6 6

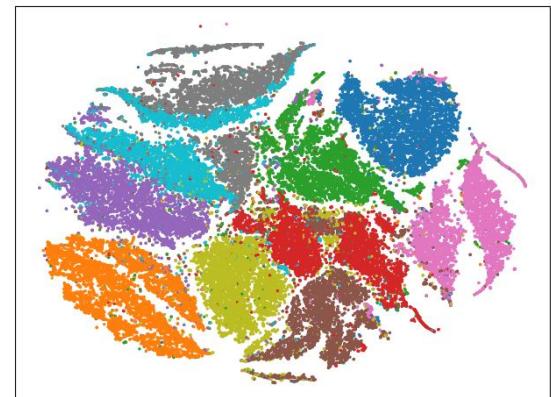
PCA (4.9 sec)



LDA (5.1 sec)

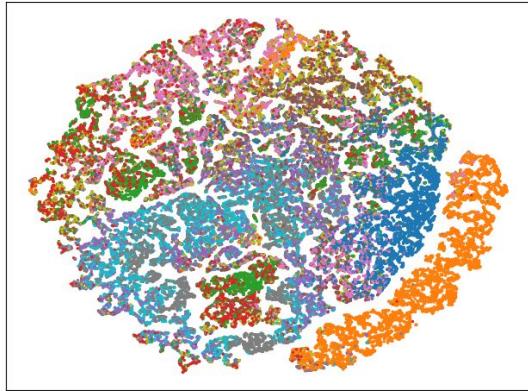


t-SNE (~58 min!)

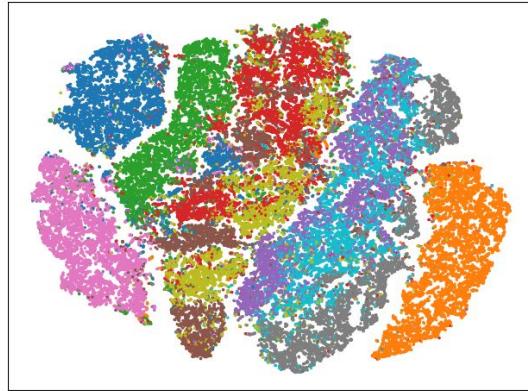


# PCA + t-SNE

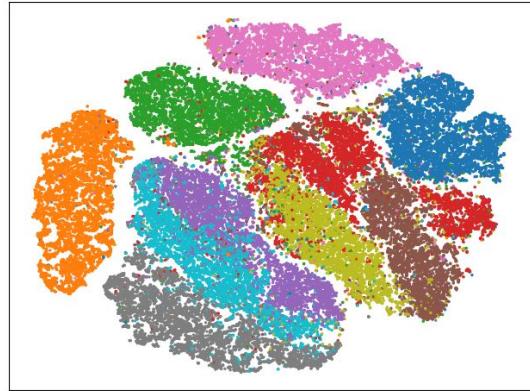
PCA/3 + t-SNE (2:04 min)



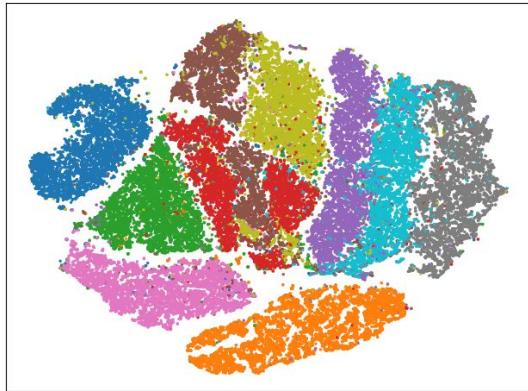
PCA/7 + t-SNE (2:30 min)



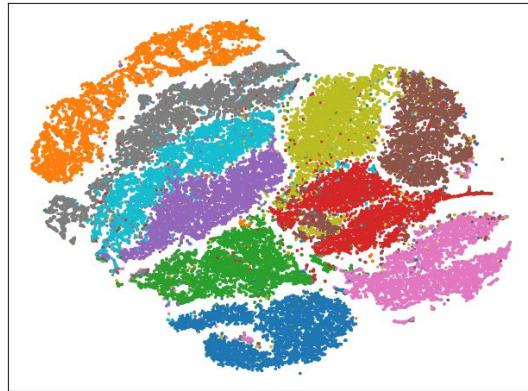
PCA/14 + t-SNE (2:56 min)



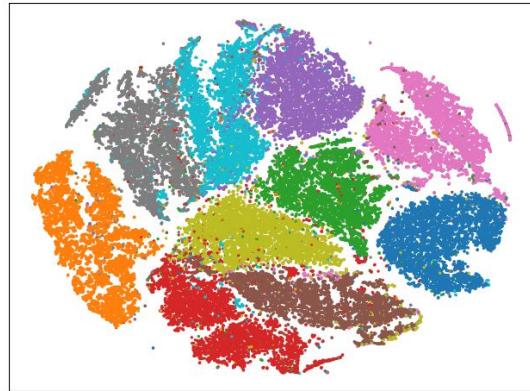
PCA/28 + t-SNE (4:31 min)



PCA/56+ t-SNE (8:16 min)



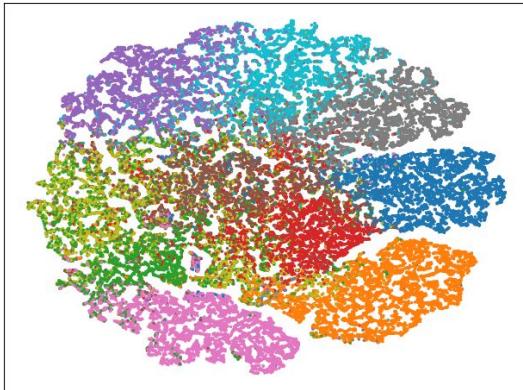
PCA/128 + t-SNE (14:00 min)



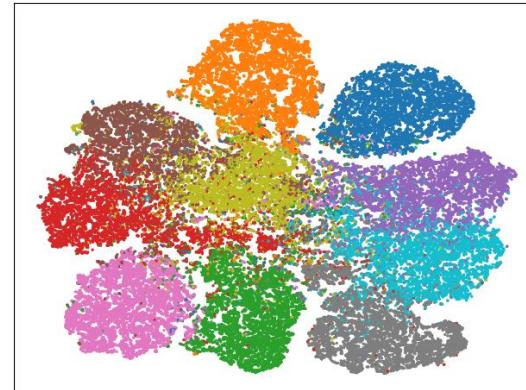
# LDA + t-SNE

- Recall restriction of LDA (compared to PCA)
  - Only up to ( $C-1$ ) non-zero eigenvalues
  - Range of dimensionality reduction by:  $1 \leq p \leq 9$  for MNIST

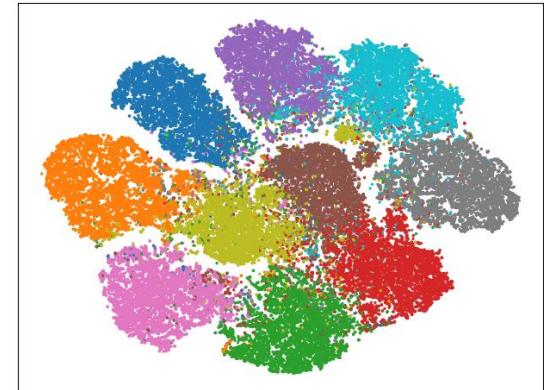
**LDA/3 + t-SNE** (2:30 min)



**LDA/6+ t-SNE** (2:41 min)



**LDA/9 + t-SNE** (2:55 min)

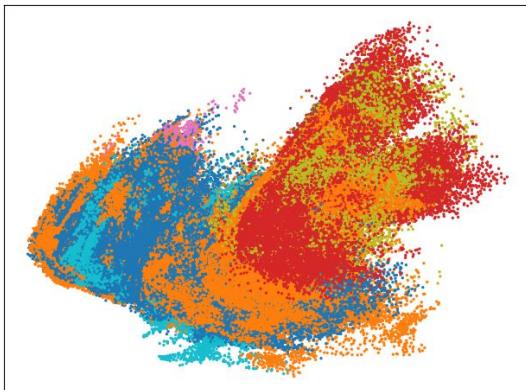


# Example — Forest Cover Type

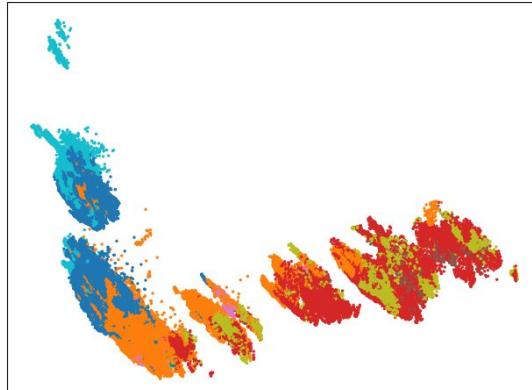
- Forest cover type classification dataset

- 581k samples of 30×30m cells with 1 of 7 forest types  
(Spruce/Fir, Lodgepole Pine, Ponderosa Pine, Cottonwood/Willow, Aspen, Douglas-fir, Krummholz)
- 54 features (elevation, aspect, slope, etc.)

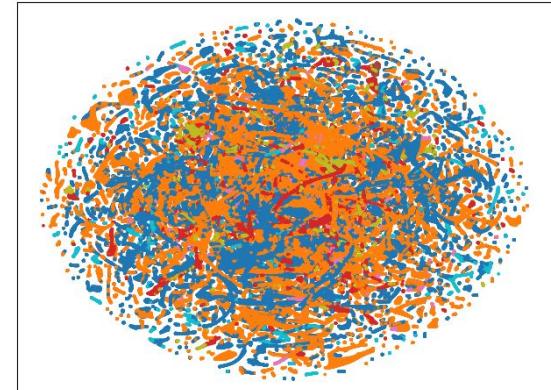
PCA (1.2sec)



LDA (3.2sec)

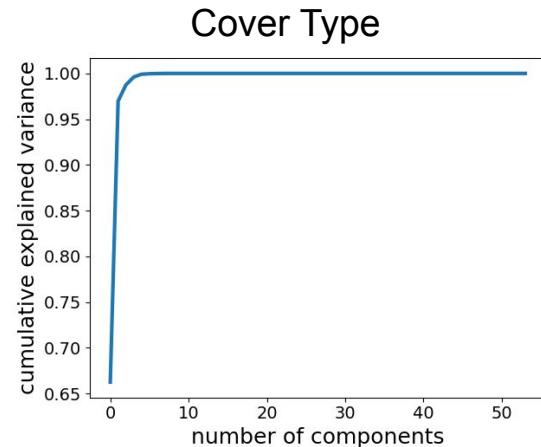
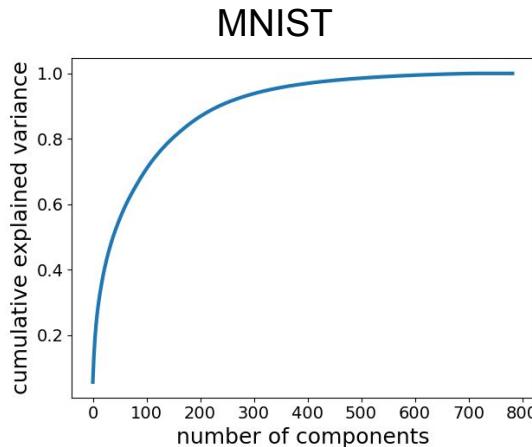
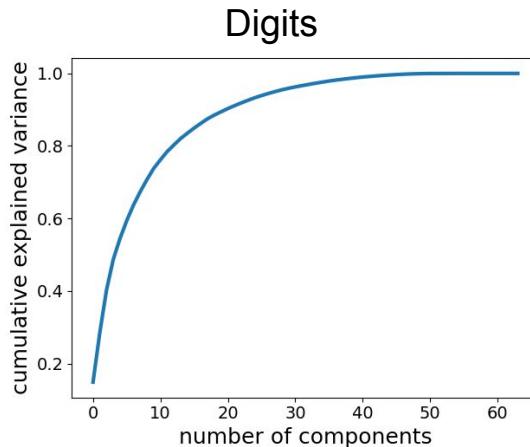


t-SNE (~30min)



# Example — Explained Variance (PCA)

- Cumulative explained variance
  - Data distribution more homogeneous for Digits and MNIST
  - Cover type dataset with many features have only 0 values (low/no variance)



# Summary

- Problem: high-dimensional dataset (i.e., large number of features)
  - Higher risk of overfitting
  - Higher computational costs
- Two basic types of countermeasures
  - Feature selection — "manually" remove subset of features
  - Feature extraction — create new features based on original features
- Dimensionality reduction techniques for feature extraction
  - Linear techniques: PCA (unsupervised) and LDA (supervised)
  - Probabilistic techniques: t-SNE

# **CS5228: Knowledge Discovery and Data Mining**

## Lecture 11 — Data Stream Mining

# Course Logistics

- Reminder for submission deadlines
  - A4: Nov 14, 11.59 pm
  - Project Report: Nov 4, 11.59 pm
- Project submission
  - Submission = project report (PDF, max 8 pages) + source code
  - Only 1 submission per team needed
  - Submission files should include team name
- Last Lecture Quiz
  - Friday, Nov 15, ~19:15 (30 min)
  - MCQs/MRQs, Lectures 7-11

# Quick Recap — Graph Mining

- **Community Detection**

- Identification of "interesting" subgraphs  
(≈ nodes in subgraph more tightly compared to other nodes)
- Similar to the task of clustering  
(clustering algorithms can be adopted to find communities)

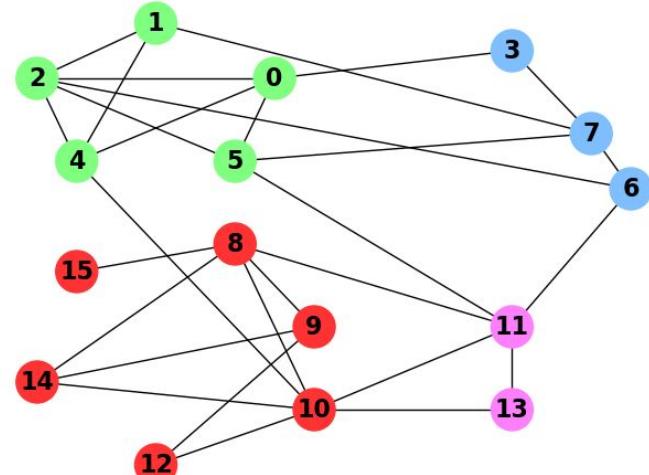
- No single definition of "community"

→ Many algorithms for community detection

- Similarity between nodes (e.g., AGNES)
- Density-based (modularity + Louvain algorithm)
- Split-based (Edge Betweenness, Min-Cut)

Girvan Newman algorithm

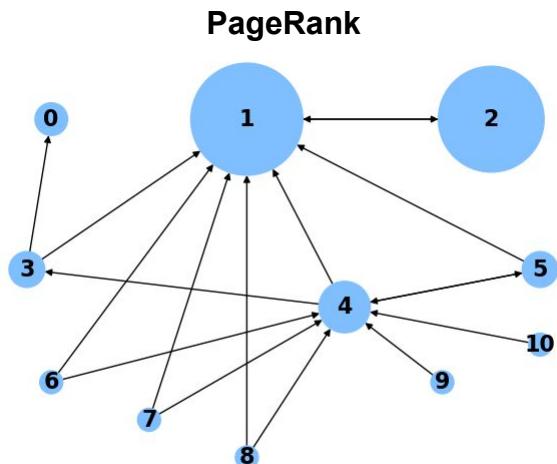
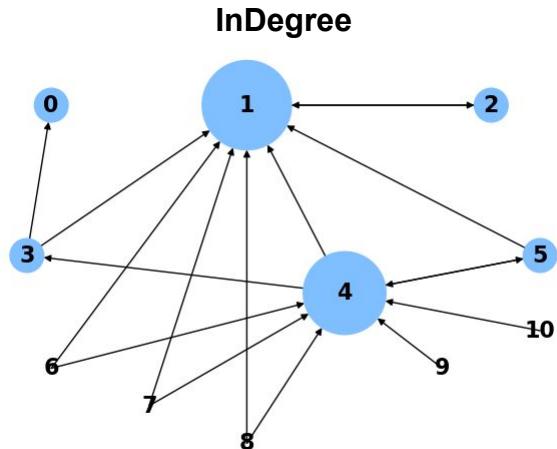
Karger's algorithm



Shared focus: **connectedness**

# Quick Recap — Graph Mining

- **Centrality = importance of a node**
  - Based on a node's topological position in a graph
  - Different centrality measures focusing on different topological features
  - Not all measures applicable to all types of graphs
- **Popular centrality measures covered**
  - Local measures (Degree, InDegree, OutDegree)
  - Eigenvector-based measures (Eigenvector Centrality, PageRank)
  - Distance-based or path-based measures (Closeness, Betweenness)



# Outline

- **Motivation**
  - Basic setup
  - Example Applications
- **Core Techniques**
  - Sampling
  - Filtering
  - Counting (distinct items)
- **Summary**

# Data Stream Mining

- Data Mining so far
    - Access to complete dataset (at the same time)
    - Virtually unlimited storage and computing resources  
(also: runtime of algorithms typically not that important, compared to the results)
    - Support of arbitrary complex patterns
  - Now: data items arrive one-by-one  
in real time...like a stream
    - All data never fully available
    - Often very high arrival speeds
    - Often limited amount of resources
    - Often time-critical decisions  
(common: execution in main memory only)
- 
- Focus on simple patterns (but not too simple)

# Quick Quiz

Given is a stream of temperature sensor values in °C.

What is the only **non-trivial** pattern to monitor?

A

temp > 100 °C

B

Average of all temperature values

C

Median of all temperature values

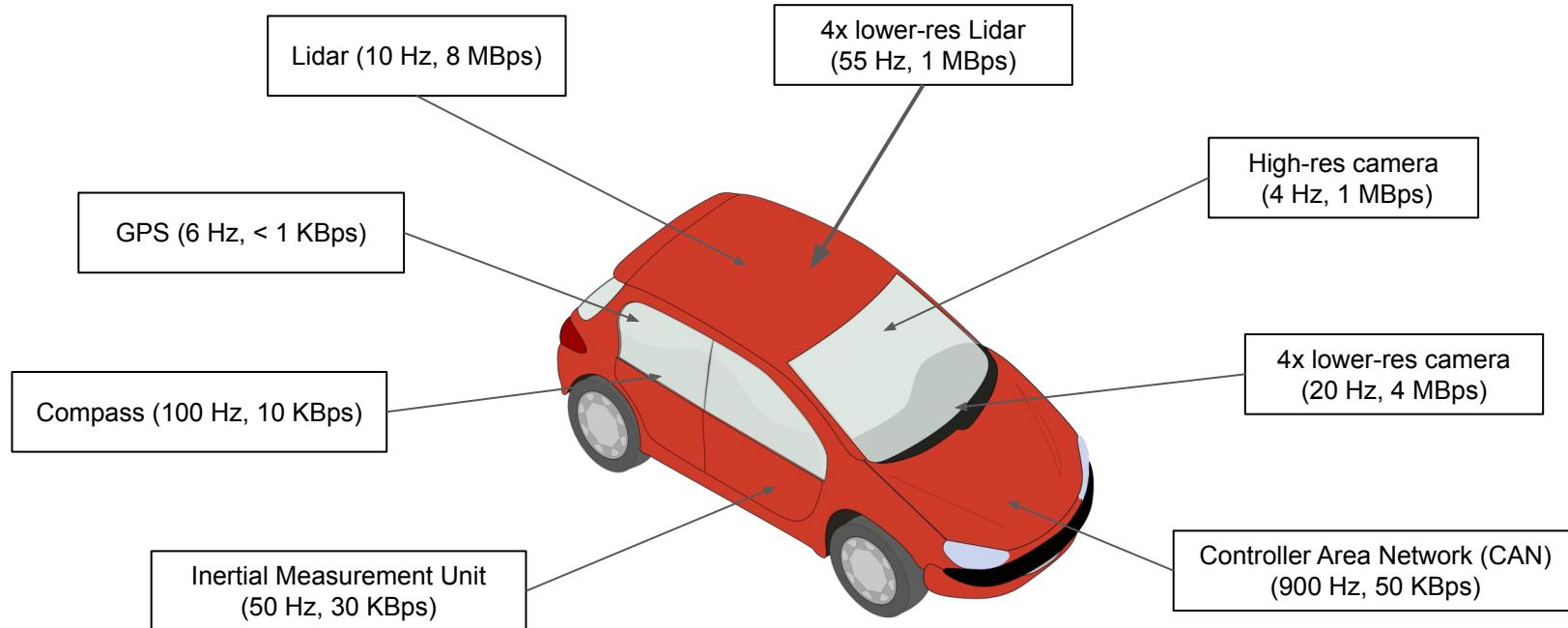
D

Maximum range of all temperature values

# (Self-Driving) Cars

- Sensor data generated during a 40-minutes trip: 30 GB (13 MBps)

Source: [Exploring big volume sensor data with Vroom](#) (Moll et al., 2017)



# Smart Cities

- Example: Lamppost-as-a-Platform (LaaP)

Source: <https://www.developer.tech.gov.sg/technologies/sensor-platforms-and-internet-of-things/lamppost-as-a-platform>



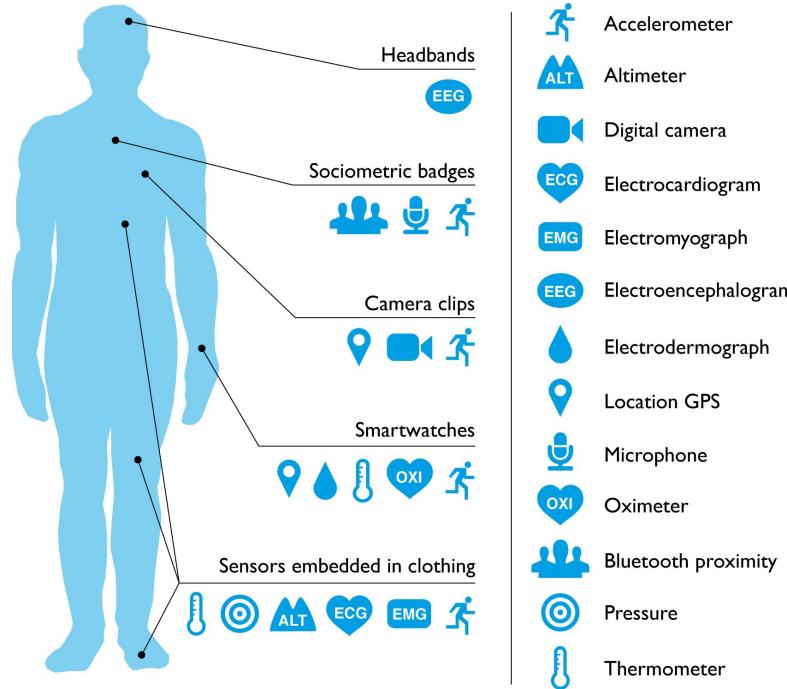
- Camera
- Temperature
- Humidity
- Gas (e.g., carbon monoxide)
- Air quality
- Rain

# Health Monitoring

- Consumer Health Wearables

Source: <https://journals.plos.org/plosmedicine/article?id=10.1371/journal.pmed.1001953>

- Smartphones
- Smartwatches
- Fitness bands
- Body cameras



# Social Media

- Example: Daily data volume on Facebook

Source: <https://blog.wishpond.com/post/115675435109/40-up-to-date-facebook-facts-and-stats>

- 4.5 billion Likes
- 17 billion location-tagged posts
- 350 million photos uploaded
- 4.75 billion items shared
- 10 billion messages sent



# Quick Sidenotes

- Covered techniques and algorithms not specific to streams
  - Applicable to many use cases involving large volumes of data
- Not of interest here: sliding window approaches
  - Keep the  $k$  most recent data items
  - Ignore all older items (delete from window)

} → window = complete dataset (snapshot)
- Commonly used throughout the lecture: hash functions
  - We will assume "well-behaved" hash functions  
(same input → same output, minimize duplication, avalanche effect)
  - No deeper discussion what this means here

# Outline

- Motivation
  - Basic setup
  - Example Applications
- Core Techniques
  - Sampling
  - Filtering
  - Counting (distinct items)
- Summary

# Sampling

- Sampling — basic definition

- Process of selecting members of a population of interest (e.g., HDB residents)
- Consideration of whole population typically impractical (e.g., too costly to survey all HDB residents)
- Goal: statistical analysis of population (e.g., average happiness, most common complaints)

- Sampling data streams

- Population = stream of incoming data items
- Time and/or resource constraint generally make it impossible to consider all data items
- Relevant patterns based on statistical analysis



→ Core challenge: How to get a representative sample?

# Problems with Naive Approach

- Toy example setup
  - Stream of search queries — items are tuples (user, query, time)
  - Goal: Fraction of queries issued more than once (here: twice) in the last 24h (by the average user)
  - Restriction only 10% of all tuples should be stored
- Naive approach: Store latest tuple with a probability of 1/10
  - Let  $s$  = number of time a user issued a query **once**
  - Let  $d$  = number of time a user issued a query **twice**

**Correct estimate:**  
(assuming all tuples)

$$\frac{d}{s + d}$$

**Estimate based on  
naive sample:**

$$\frac{d}{10s + 19d}$$

# Problems with Naive Approach — Explanation

- The issue: of  $d$  queries issued twice

- $d/100$  will be both in the sample

$$\left(\frac{1}{10}\right) \left(\frac{1}{10}\right) \cdot d = \frac{1}{100}d$$

- $18d/100$  will be only once in the sample

$$2 \cdot \left(\frac{1}{10}\right) \left(\frac{9}{10}\right) \cdot d = \frac{18}{100}d$$

- Fractions of queries issued twice in the sample:

$$\frac{\frac{1}{100}d}{\frac{1}{100}d + \frac{18}{100}d + \frac{1}{10}s} = \frac{d}{10s + 19d}$$

# Sample with a Given Probability

- General Approach
  - Given: items/tuples with  $n$  components, e.g., (user, query, time)
  - Select subset of components as key on which the selection of the sample is based  
(for toy example: key = "user" → consider only 10% of users instead of arbitrary tuples)
  - Choice of key depends on goal of analysis
- Question: How to obtain a sample consisting of any fraction  $a/b$  of keys?  
(in other words: How to decide whether to keep or discard a new tuple?)
- Common implementation via hash function
  - Define hash function  $h$  that maps  $h(key)$  in to  $b$  buckets  $1..b$
  - For each new tuple, if  $h(key) \leq a$  (with  $a \leq b$ ), add tuples to sample

# Sample with a Given Size Limit — Reservoir Sampling

- Goal: Maintain a uniform random sample of fixed size  $B$ 
  - Uniform random = each item has the same probability to be sampled
  - Allows to approximate basic statistics such as mean, variance, median, etc.
- Basic algorithm
  - Input stream of items  $\{a_1, a_2, \dots\}$
  - Maximum reservoir size  $B$
  - It can be shown that each item in the reservoir was sampled with the same probability  $B/t$  (at any time  $t$ )

- 1) Add  $a_t$  with  $t \leq B$  to reservoir
- 2) When receiving  $a_t$  with  $t > B$ :  
with probability  $B/t$ , replace random item in reservoir with new item  $a_t$

# Reservoir Sampling — Example

- Initialization:  $t = 0, B = 3$

Stream:



$t = 1$	<b>A</b>
$t = 2$	<b>B</b>
$t = 3$	<b>C</b>
$t = 4$	<b>A</b>
$t = 5$	<b>C</b>
$t = 6$	<b>B</b>
$t = 7$	<b>B</b>
$t = 8$	<b>A</b>
$t = 9$	<b>C</b>

Reservoir:


- 1) Add  $a_t$  with  $t \leq B$  to reservoir
  - 2) When receiving  $a_t$  with  $t > B$ :  
with probability  $B/t$ , replace random item in reservoir with new item  $a_t$

# Reservoir Sampling — Example

- $t = 3$

Stream:

$t = 1$	A
$t = 2$	B
$t = 3$	C
$t = 4$	A
$t = 5$	C
$t = 6$	B
$t = 7$	B
$t = 8$	A
$t = 9$	C



Reservoir:

A
B
C

Just fill up the reservoir...

- 1) Add  $a_t$  with  $t \leq B$  to reservoir
- 2) When receiving  $a_t$  with  $t > B$ :  
with probability  $B/t$ , replace random item in reservoir with new item  $a_t$

# Reservoir Sampling — Example

- $t = 4$

Stream:

$t = 1$	A
$t = 2$	B
$t = 3$	C
 $t = 4$	A
$t = 5$	C
$t = 6$	B
$t = 7$	B
$t = 8$	A
$t = 9$	C

Reservoir:

A
B
<del>C</del> A

- 1) Add  $a_t$  with  $t \leq B$  to reservoir
- 2) When receiving  $a_t$  with  $t > B$ :  
with probability  $B/t$ , replace random item in reservoir with new item  $a_t$

Probability  $B/t = 3/4$

→ Randomized decision: add  $a_4 = A$  to reservoir

→ Replace random element — here  $B_3 = C$  with A

# Reservoir Sampling — Example

- $t = 5$

Stream:

$t = 1$	A
$t = 2$	B
$t = 3$	C
$t = 4$	A
 $t = 5$	C
$t = 6$	B
$t = 7$	B
$t = 8$	A
$t = 9$	C

Reservoir:

A
<del>B C</del>
A

- 1) Add  $a_t$  with  $t \leq B$  to reservoir
- 2) When receiving  $a_t$  with  $t > B$ :  
with probability  $B/t$ , replace random item in reservoir with new item  $a_t$

Probability  $B/t = 3/5$

→ Randomized decision: add  $a_5 = C$  to reservoir

→ Replace random element — here  $B_2 = B$  with C

# Reservoir Sampling — Example

- $t = 6$

Stream:

$t = 1$	<b>A</b>
$t = 2$	<b>B</b>
$t = 3$	<b>C</b>
$t = 4$	<b>A</b>
$t = 5$	<b>C</b>
 $t = 6$	<b>B</b>
$t = 7$	<b>B</b>
$t = 8$	<b>A</b>
$t = 9$	<b>C</b>

Reservoir:

<b>A</b>
<b>C</b>
<b>A</b>

- 1) Add  $a_t$  with  $t \leq B$  to reservoir
- 2) When receiving  $a_t$  with  $t > B$ :  
with probability  $B/t$ , replace random item in reservoir with new item  $a_t$

Probability  $B/t = 3/6$

→ Randomized decision: discard  $a_6 = B$

# Reservoir Sampling — Example

- $t = 7$

Stream:

$t = 1$	A
$t = 2$	B
$t = 3$	C
$t = 4$	A
$t = 5$	C
$t = 6$	B
$t = 7$	B
$t = 8$	A
$t = 9$	C



Reservoir:

A
C
<del>A</del> B

- 1) Add  $a_t$  with  $t \leq B$  to reservoir
- 2) When receiving  $a_t$  with  $t > B$ :  
with probability  $B/t$ , replace random item in reservoir with new item  $a_t$

Probability  $B/t = 3/7$

→ Randomized decision: add  $a_7 = B$  to reservoir

→ Replace random element — here  $B_3 = A$  with B

# Proof Sketch — All Elements in $B$ sampled with $B/t$

- Obvious case:  $i = t$ 
  - $a_i$  was inserted into  $B$  with probability  $B/t$  (direct result from algorithm)
- Otherwise:  $i < t$ 
  - Observation: in step  $i$ , an item gets replaced with probability  $B/i * 1/B = 1/i$
  - Probability of an item in reservoir

$$B/i \cdot \left(1 - \frac{1}{i+1}\right) \cdot \left(1 - \frac{1}{i+2}\right) \cdot \dots \cdot \left(1 - \frac{1}{t}\right) = B/t$$

The diagram illustrates the derivation of the formula for the probability of an item being in the reservoir. It consists of four rectangular boxes at the bottom, each containing a probability statement, with arrows pointing upwards to the corresponding term in the product formula above.

- The first box contains "Probability that  $a_i$  was inserted in  $B$ ". An arrow points from this box to the first term  $B/i$ .
- The second box contains "Probability that  $a_i$  was NOT replaced in  $B$  at step  $(i+1)$ ". An arrow points from this box to the second term  $\left(1 - \frac{1}{i+1}\right)$ .
- The third box contains "Probability that  $a_i$  was NOT replaced in  $B$  at step  $(i+2)$ ". An arrow points from this box to the third term  $\left(1 - \frac{1}{i+2}\right)$ .
- The fourth box contains "Probability that  $a_i$  was NOT replaced in  $B$  at step  $t$ ". An arrow points from this box to the final term  $\left(1 - \frac{1}{t}\right)$ .

# Outline

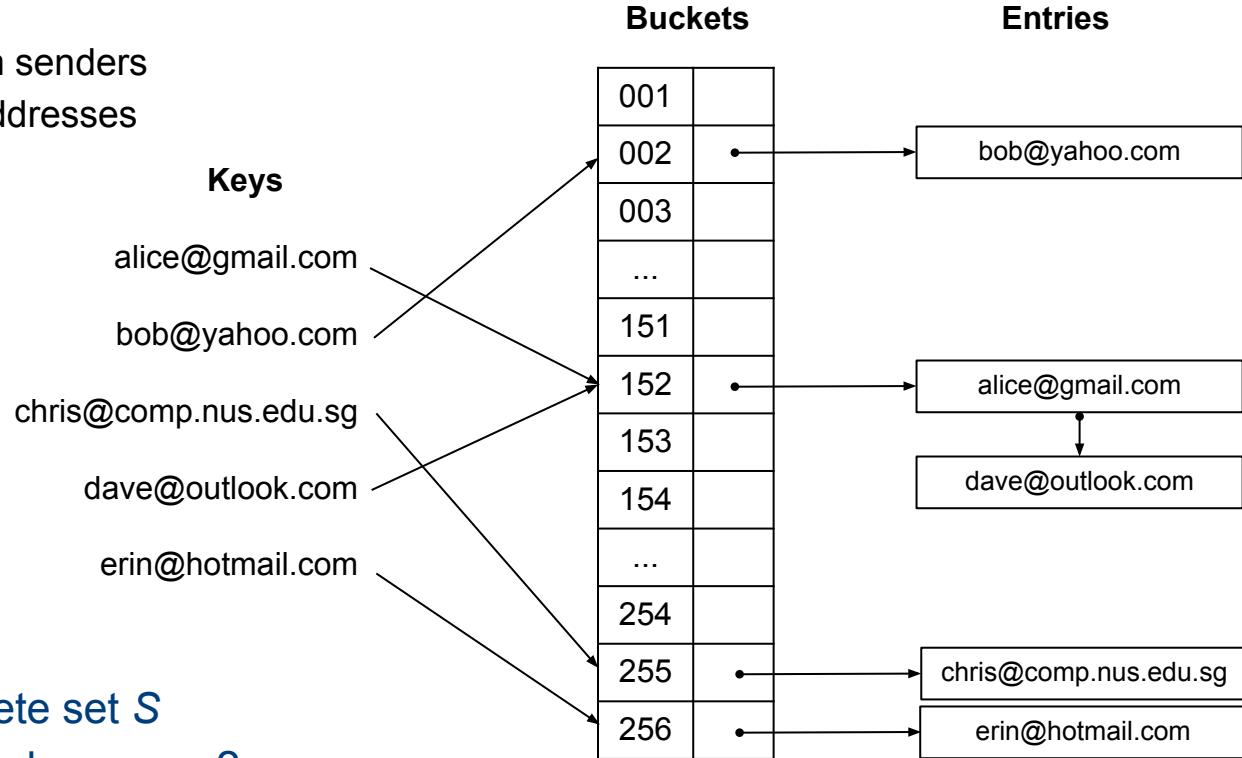
- Motivation
  - Basic setup
  - Example Applications
- Core Techniques
  - Sampling
  - Filtering
  - Counting (distinct items)
- Summary

# Filtering Data Streams

- Goal: Only accept items that meet certain criterion
- Simple: criterion is a property of item that can be (easily) calculated, e.g.:
  - Search queries with more than 2 keywords
  - Sensor values with valid status code
- Challenge: criterion involves lookup for membership in a (very large) set  $S$ , e.g.:
  - Emails with sender addresses that are whitelisted (spam filter)
  - Page visits to a predefined set of websites
  - Tweets from a selected group of users

# Basic Solution — Create Hash Table for Set S

- Example: spam filter
  - Only accept emails from senders with whitelisted email addresses



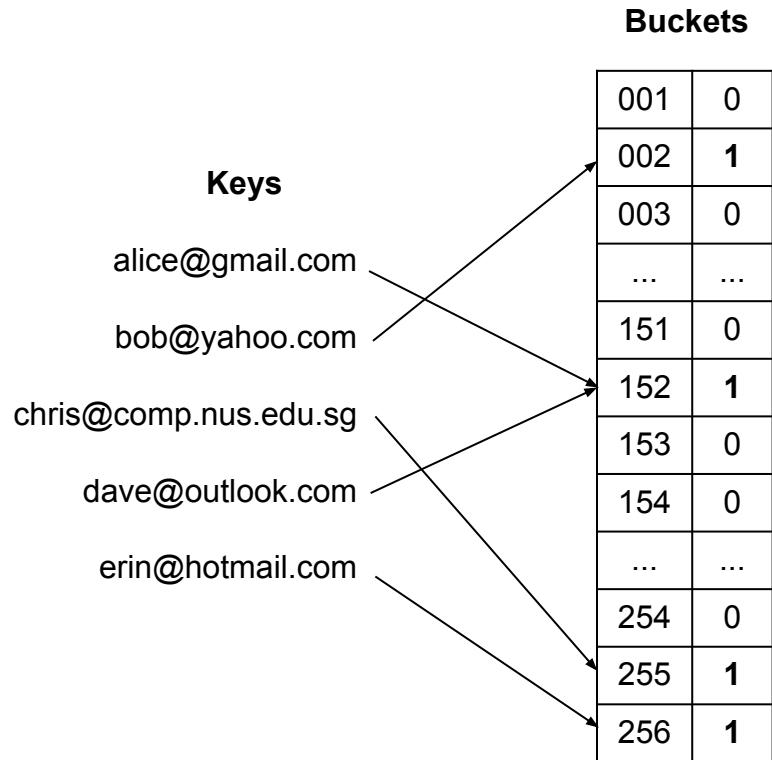
- Requires to store complete set S
- What if there is not enough memory?

# Hashing without Storing $S$

- Create lookup table
    - Create bit array  $B$  with  $1..n$  bits and  $B[i] = 0$
    - Choose hash function  $h(key) \in [1, n]$
    - For each key  $s \in S$  set  $B[h(s)] = 1$
  - Filter step for new data item with key  $k$ 
    - Accept if  $B[h(k)] = 1$ , discard otherwise
- Problem: False Positives
- $a \in S, b \notin S$  and  $h(a) = h(b)$



How bad is it?



# Quick Quiz

Can **false negatives** occur?

**A**

No

**B**

Yes, but the probability is negligible

**C**

Yes, but they do not matter for  
the application context.

**D**

Yes, and that's why  
this is a bad approach.

# False Positive Rate — Analysis

Probability of  $B[i] = 1$  after inserting **one key** from  $S$ :

$$\frac{1}{|B|}$$

Probability of  $B[i] = 0$  after inserting **one key** from  $S$ :

$$1 - \frac{1}{|B|}$$

Probability of  $B[i] = 0$  after inserting **all keys** from  $S$ :

$$\left(1 - \frac{1}{|B|}\right)^{|S|} = \left(1 - \frac{1}{|B|}\right)^{|B|\frac{|S|}{|B|}} \approx e^{-\frac{|S|}{|B|}}$$

Probability of  $B[i] = 1$  after inserting **all keys** from  $S$ :

$$1 - e^{-\frac{|S|}{|B|}}$$



Probability of  $B[h(s)] = 1$  with key  $s \notin S$ :

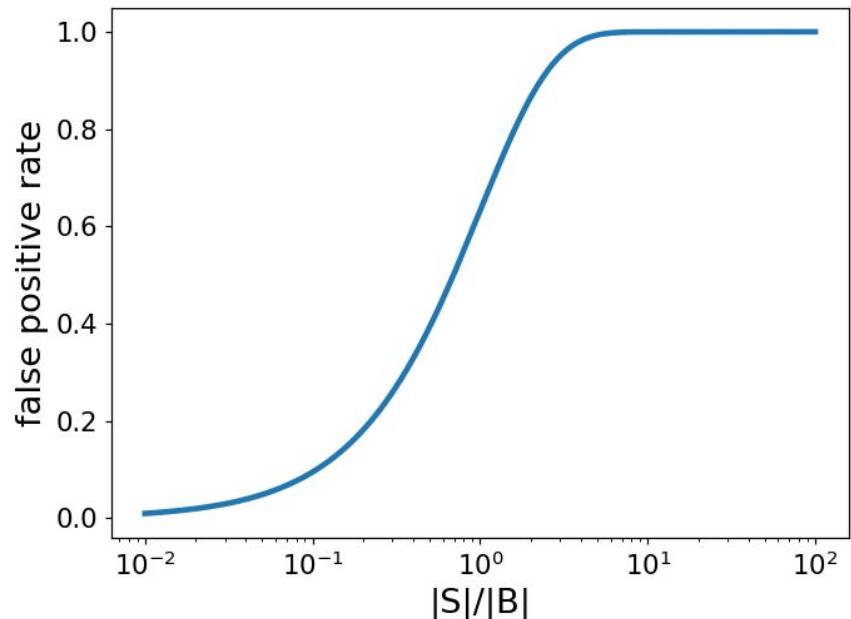
(probability of a false positive)

$$1 - e^{-\frac{|S|}{|B|}}$$

$$e^p = \left[ \lim_{n \rightarrow \infty} \left(1 + \frac{1}{n}\right)^n \right]^p$$

# False Positive Rate — Visualization

- Effect of ratio  $|S|/|B|$  on false positive rate
  - Example calculation
    - $|S| = 10^9$  whitelisted email addresses
    - $|B| = 8 \cdot 10^9$  (1 GB of main memory)
- $|S|/|B| = 1/8$
- False positive rate:  $1 - e^{-1/8} = 11.75\%$



**How to reduce false positive rate?**  
(without increasing the size of bit array B)

# Bloom Filters

- Bloom Filters — basic idea

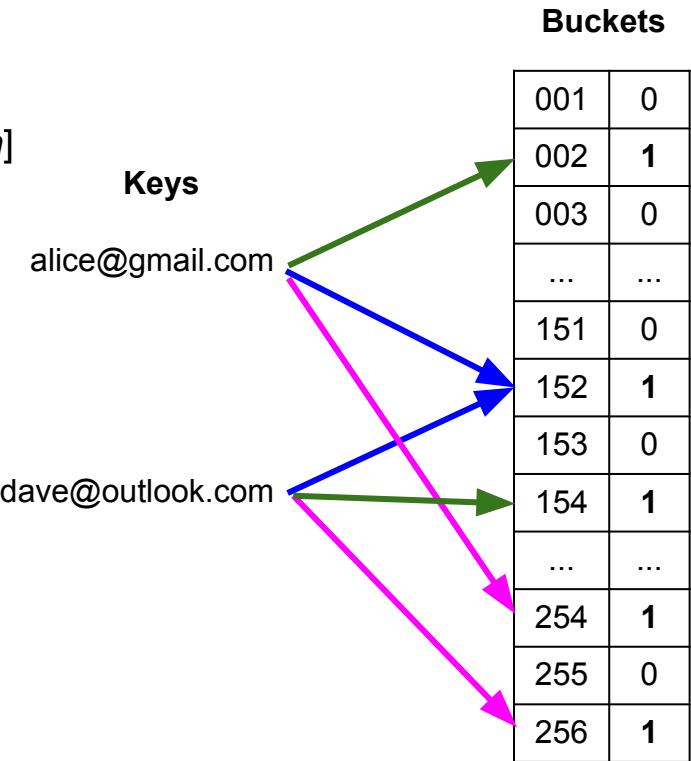
- Create bit array  $B$  with  $1..n$  bits and  $B[i] = 0$
- Choose  $m$  independent hash functions  $h_i(key) \in [1, n]$
- For each key  $s \in S$  set  $B[h_i(s)] = 1$ , with  $1 \leq i \leq m$

- Example

- 3 hash functions:  $h_1$ ,  $h_2$ ,  $h_3$

- Filter step for new data item with key  $k$

- Accept if  $B[h_i(k)] = 1$  for all  $1 \leq i \leq m$ ,  
discard otherwise



# False Positive Rate — Analysis

Probability of  $B[h(s)] = 1$  with key  $s \notin S$ :  
(probability of a false positive)

$$1 - e^{-\frac{|S|}{|B|}}$$



from 1 to  $m$  hash functions

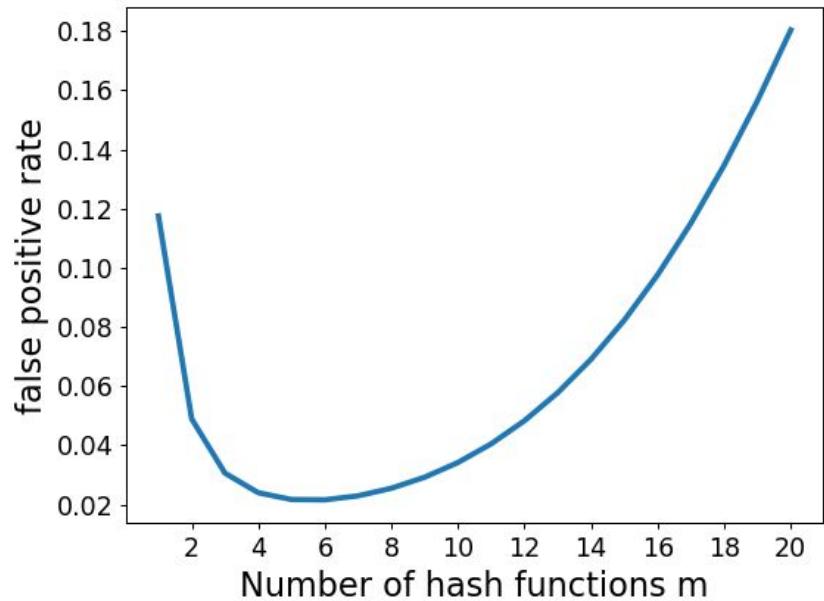
Probability of  $B[h_i(s)] = 1$  ( $1 \leq i \leq m$ ) with key  $s \notin S$ :  
(probability of a false positive)

$$\left(1 - e^{-m \frac{|S|}{|B|}}\right)^m$$

# False Positive Rate — Visualization

- Effect of number of hash functions  $m$  on false positive rate
  - Here,  $|S|/|B| = 1/8$  (see previous example)
- Global optimum — intuition: large  $m$ 
  - More hash results need all be 1
  - But: also more 1s in bit array
- Optimal value for  $m$

$$m_{best} = \frac{|B|}{|S|} \ln 2$$



$$m_{best} = \frac{8}{1} \ln 2 = 5.5 \approx 6$$

→ False positive rate: **2.2%**

# Bloom Filter — Discussion

- **Memory and space consumption**
  - Time complexity:  $O(m)$
  - Space complexity:  $O(|B|)$
- **Limitation: no support of removing keys from  $S$** 
  - E.g.: not able to remove a whitelisted email address
  - Only workaround: rebuild lookup table (bit array) from scratch

Why?

# Extension — Counting Bloom Filters

- Counting Bloom Filters

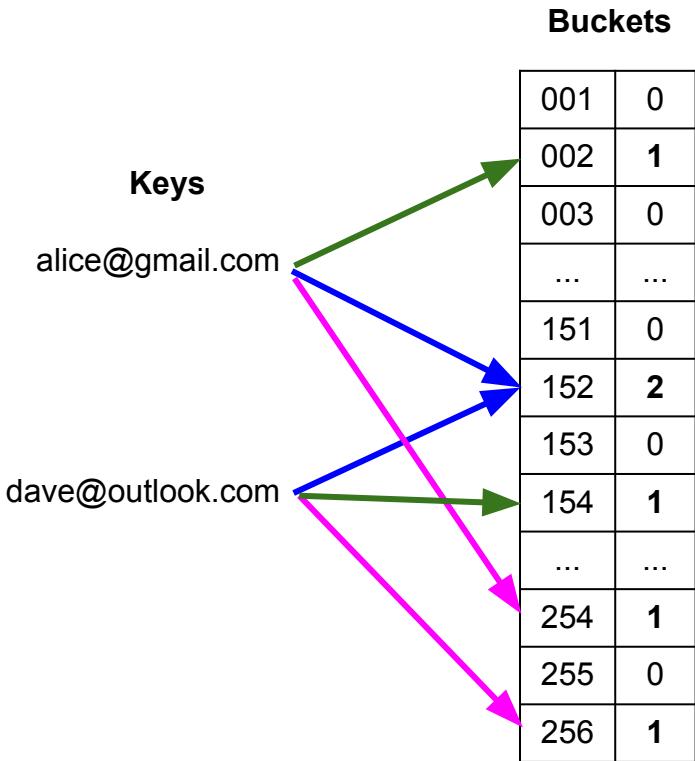
- Replace bits for counters (typically 3-4 bits per counter)
- Increased size of lookup table (3-4 times)

- Operations

- Insert  $s$ :  $B[h_i(s)] \leftarrow 1$ , with  $1 \leq i \leq m$
- Delete  $s$ :  $B[h_i(s)] \leftarrow 0$ , with  $1 \leq i \leq m$
- Lookup  $k$ : Accept if  $B[h_i(k)] > 0$  for all  $1 \leq i \leq m$

- False positive rate

- Same as normal Bloom Filter:  $\left(1 - e^{-m\frac{|S|}{B}}\right)^m$



# Outline

- Motivation
  - Basic setup
  - Example Applications
- Core Techniques
  - Sampling
  - Filtering
  - **Counting** (distinct items)
- Summary

# Counting Unique/Distinct Elements

- Applications

- Number of unique Facebook visitors (identified by user id)
- Number of unique website visitors (identified by IP address)
- Number of unique words in a (large) text document

- Straightforward solution

- Maintain set  $S$  of items seen so far
- Number distinct elements  $\rightarrow |S|$

→ What if set  $S$  can grow very large?

```
1 S = set()
2
3 with open('data/ip-only-nasa-access.log') as file:
4     for line in file:
5         ip = line.strip() # Get IP address
6         S.add(ip)          # Add IP address to set
7
8 print('|S| = {}'.format(len(S)))
|S| = 7637
```

# Flajolet-Martin Algorithm

- Approximation approach
  - Estimate distinct count in an unbiased way
  - Accept errors but minimize their probability
- Basic algorithm

- (1) Choose hash function that maps each of the  $n$  elements to at least  $\log_2 n$  bits
- (2) For each element  $s$  in the stream
  - (a) Calculate hash  $h(s)$  → bit string of length  $\log_2 n$
  - (b) Let  $r(s) =$  number of trailing 0s in  $h(s)$
  - (c) Keep track of largest  $r(s)$  →  $R$
- (3) Return estimate for distinct count as  $2^R$

} Example:  $2^{32}$  IPv4 addresses  
→ at least 32 bits

# Flajolet-Martin Algorithm — Example

a: IPv4 address	h(a)
13.66.139.0	01010100110100100110111000001001
157.48.153.185	1100101001001001001001011011111 <u>00</u>
157.48.153.185	1100101001001001001001011011111 <u>00</u>
216.244.66.230	0001110001010101100111010001001 <u>0</u>
54.36.148.92	010101010001010110010111000011 <u>00</u>
92.101.35.224	10100011100001100000000100000001
73.166.162.225	00100100100111100100101101011 <u>000</u>
73.166.162.225	00100100100111100100101101011 <u>000</u>
54.36.148.108	00001100010100011000000001010001
54.36.148.1	011001011000101000101100100001 <u>00</u>
162.158.203.24	10111000010010010011100010111 <u>0</u>
157.48.153.185	1100101001001001001011011011 <u>00</u>
157.48.153.185	1100101001001001001011011011 <u>00</u>
...	...

R = 3 (largest number of trailing 0s so far)

→ Estimate for distinct count:  $2^3 = 8$

# Quick "Quiz"

Someone is telling you that s/he flipped a fair coin **3 times** and got **3 Heads** after ***k* tries**?

Which **number of tries** is the most believable?

**A**

$k = 1$

**B**

$k = 10$

**C**

$k = 100$

**D**

$k = 1,000$

# Flajolet-Martin Algorithm — Intuition & Proof Sketch

- Basic intuition

- More distinct elements → more different hash values → "unusual" hash values more likely
- "Unusual" hash value = hash value with rare bit pattern (e.g., number of trailing 0s)

Probability that  $h(a)$  ends in **at least**  $k$  trailing 0s

$$\underbrace{\frac{1}{2} \cdot \frac{1}{2} \cdot \dots \cdot \frac{1}{2}}_{k \text{ factors}} = \frac{1}{2^k} = s^{-k}$$

Probability that  $h(a)$  ends in **less than**  $k$  trailing 0s

$$1 - \frac{1}{2^k}$$

Given  $m$  distinct elements, probability that  
**all**  $h(a)$  end in **less than**  $k$  trailing 0s

$$\left(1 - \frac{1}{2^k}\right)^m$$

Given  $m$  distinct elements, probability that  $R \geq k$   
(i.e., at least one of the  $m$  elements has  $h(a)$  with at least  $k$  trailing 0s)

$$1 - \left(1 - \frac{1}{2^k}\right)^m$$

# Flajolet-Martin Algorithm — Proof Sketch

Given  $m$  distinct elements, probability that  $\mathbf{R} \geq \mathbf{k}$   
(i.e., at least one of the  $m$  elements has  $h(a)$  with at least  $k$  trailing 0s)

$$e^p = \left[ \lim_{n \rightarrow \infty} \left( 1 + \frac{1}{n} \right)^n \right]^p$$

$$1 - \left( 1 - \frac{1}{2^k} \right)^m = 1 - \left( 1 - \frac{1}{2^k} \right)^{\frac{2^k m}{2^k}} \approx 1 - e^{-\frac{m}{2^k}}$$

**Case 1:**  $2^k \ll m \rightarrow 1 - e^{-\frac{m}{2^k}} \approx 1 - 0 = 1$

**Case 2:**  $2^k \gg m \rightarrow 1 - e^{-\frac{m}{2^k}} \approx 1 - \underbrace{\left( 1 - \frac{m}{2^k} \right)}_{\text{First 2 terms of the Taylor expansion of } e^x} \approx \frac{m}{2^k} \approx 0$

$$e^x = 1 + \frac{x}{1!}, \text{ if } x \ll 1$$

First 2 terms of the  
Taylor expansion of  $e^x$

$$e^x = 1 + \frac{x}{1!} + \frac{x^2}{2!} + \frac{x^3}{3!} + \dots$$

# Flajolet-Martin Algorithm — Proof Sketch

- Interpretation

**Case 1:**  $2^k \ll m \rightarrow P(R \geq k) \approx 1$

The probability to get an  $h(a)$  with  
**enough trailing 0s** is rather high

**Case 2:**  $2^k \gg m \rightarrow P(R \geq k) \approx 0$

The probability to get an  $h(a)$  with  
**too many trailing 0s** is rather low

}  $\rightarrow R$  is typically in the right ballpark

# Flajolet-Martin Algorithm — Problems & Extensions

- Obvious problem with basic Flajolet-Martin algorithm — given  $2^R$ 
  - An estimate is always a power of 2
  - If  $R$  is off by just 1, estimates doubles or halves
- Practical solution: use multiple hash functions  $h_i(a)$  — e.g.:
  - $p \cdot q$  hash functions  $\rightarrow p \cdot q R$  values  $\rightarrow p \cdot q$  estimates for the distinct counts
  - Put all estimates into  $p$  groups, each of size  $q$
  - Calculate median of each group  $\rightarrow p$  medians
  - Calculate the mean over all  $p$  medians

# Outline

- Motivation
  - Basic setup
  - Example Applications
- Core Techniques
  - Sampling
  - Filtering
  - Counting (distinct items)
- Summary

# Summary

- Data streams — main challenges
  - Large data volumes + high arrival speeds
  - Limited resources + real-time requirements



→ patterns = statistical analysis

- Common tasks on streams

(or very large datasets in general)

- Sampling
- Filtering
- Counting  
(distinct elements)



→ trade-off: speed / resource-efficiency vs. accuracy / errors

# Solutions to Quick Quizzes

- Slide 7: C
- Slide 30: A
- Slide 42: B