

COMP4131: Data Modelling and Analysis

Lecture 1: Introduction to Data Modelling and Analysis

Kian Ming Lim

University of Nottingham Ningbo China

kian-ming.lim@nottingham.edu.cn

February 15, 2025

Outline

- ① Teaching Team
- ② Overview of the Module
- ③ Communication, Attendance, Academic Integrity
- ④ An Insight into Data Modelling and Analysis
- ⑤ Data Modelling and Analysis Pipeline

Teaching Team

Introduction to the Teaching Team - Lecturers



Assoc. Prof. Dr. Kian Ming Lim
PMB-424

Kian-Ming.Lim@nottingham.edu.cn

Office Hours: Monday, 09:00 - 10:00, 14:00 - 15:00

Research Profile: [Link](#)



Asst. Prof. Dr. Daokun Zhang
IAMET-229

Daokun.Zhang@nottingham.edu.cn

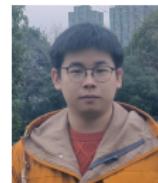
Office Hours: Monday, 09:00 - 11:00

Research Profile: [Link](#)

Introduction to the Teaching Team - Tutors



PhD Student Yue Yang
scxxy2@nottingham.edu.cn



PhD Student Qinglin Mao
scxqm1@nottingham.edu.cn

Overview of the Module

Overview of the Module

- To introduce the principles, techniques, and applications of data analysis and modelling.
- To enable students to recognize the most widely-used data analysis and modelling techniques and determine the appropriate technique for specific applications.
- To enable students to understand and apply computer-based data analysis and modelling techniques in practice.

Learning Outcomes

- **Knowledge and Understanding:**
 1. Understanding the capabilities, strengths and limitations of data analysis and modelling methods
 2. An appreciation of different data analysis and modelling techniques
- **Intellectual Skills:**
 1. The ability to understand complex ideas and relate them to specific situations
- **Professional Skills:**
 1. The ability to implement selected data analysis and modelling methods for real world applications
 2. The ability to evaluate data analysis and modelling techniques and select those appropriate to a given task
- **Transferable Skills:**
 1. The ability to address real problems and assess the value of their proposed solutions
 2. The ability to retrieve and analyse information from a variety of sources and produce detailed written reports on the result

Learning Activities

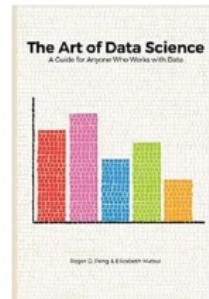
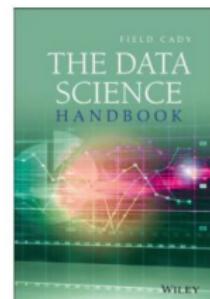
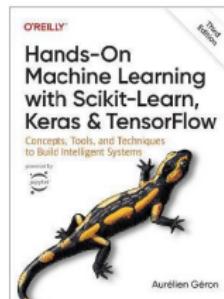
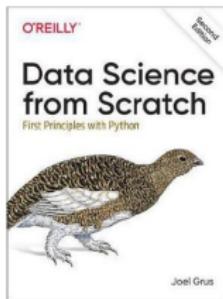
- 2 hours Lecture and 2 hours Computing / 10 weeks
 - Lecture: Monday 12.00pm - 2.00pm (DB C06)
 - Computing: Monday 4.00pm - 6.00pm (IAMET 406)
- Module managed through Moodle (for all resources)

Assessment

- Coursework 1 (25%)
 - Lab submission
 - You will need to complete and submit the lab. Each lab submission 2.5%.
- Coursework 2 (75%)
 - Data analysis study (code and 4000 words)
 - You will need to select a data set and use all the data modelling and analysis steps you have learnt to analyse and wrangle the data set, and create and compare models.
 - You will write your work up as an academic paper - comparing and analysing your results at every stage of the data analysis and modelling pathway.

Module Resources

- Textbook:



- Reading list: <https://rl.talis.com/3/notts/lists/F44CC47F-9CE2-1B7B-8FD5-903D6A49B5B2.html>
- Programming language: Python
- Software: Jupyter Notebook (Anaconda)

Expected Workload

Activity	Hrs by Week	Hours
Lecture – deliver key material	2×10	20
Computing – putting theory into practice	2×10	20
Self-study – revise lecture material	2×10	20
Lab submission (25%) – complete the lab exercises		40
Coursework (75%) – data modelling and analysis to solve a real-world problem, and writing an academic paper.		100
Total (20 credits)		200

Schedule for the Module

Week	Lecture	Lab
1	Introduction to Data Modelling and Analysis	Jupyter Notebooks + Basic Data Manipulation
2	Data Wrangling and Pre-processing	Data Wrangling and Pre-processing
3	Data Visualisation	Visualisation
4	Analysis and Modelling	Exploratory Data Analysis
5	Unsupervised Learning	Unsupervised Machine Learning
6	Linear Regression and Logistic Regression	Linear Regression and Logistic Regression
7	Gaussian Process Regression and Classification	Gaussian Process Regression and Classification
8	Naïve Bayes and K Nearest Neighbors	Naïve Bayes and K Nearest Neighbors
9	Decision Tree and Random Forest	Decision Tree and Random Forest
10	Support Vector Machines and Kernel Methods	Support Vector Machines and Kernel Methods

- **Core Knowledge:** DMA is key for understanding data. You'll learn to represent real-world data for computer processing.
- **Computational Thinking:** It improves your computational thinking. You'll learn to break down complex data problems. When pre-processing data for machine learning, you'll clean, normalize, and extract features. These skills are vital for creating intelligent algorithms.
- **Interdisciplinary Applications:** DMA has strong interdisciplinary connections. In today's digital age, data is everywhere, and the knowledge from this module can be applied across different fields.
- **Research Skills:** If you're interested in research, this module gives you the tools. You'll work with modern data analysis tools and techniques, which are essential for research in big data, AI, and data-driven optimization.

Data Modelling and Analysis and your Career

- Data modelling and analysis are fundamental skills for any data-oriented professional.
- The ability to perform data modelling and analysis is highly sought after in the industry. It is a key component of many data-related job roles, such as data analyst, data scientist, data engineer, business intelligence analyst, and many more.

Data Science vs Machine Learning

- Coursera
- <https://www.coursera.org/articles/data-science-vs-machine-learning>

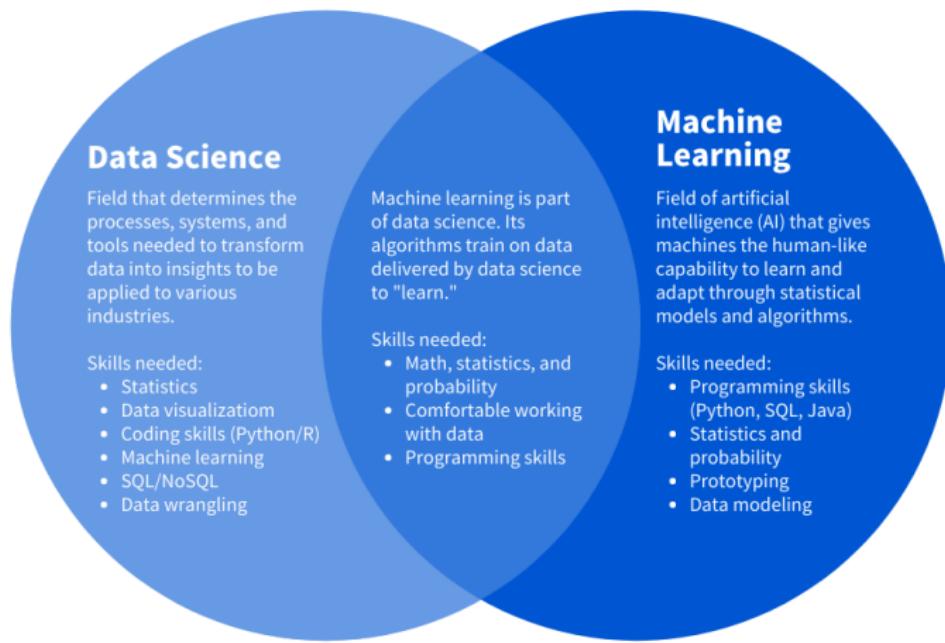


Image Source: Data science vs. machine learning: What's the difference?



Communication, Attendance, Academic Integrity

Module Communication and Q&A

- All module communication will be performed via the “Announcement” forum on Moodle.
 - Please check the forum regularly for important updates.
- If you have a question about the module, please post it on the “Q&A” forum on Moodle.
 - We will respond to your question as soon as possible.
 - If you have a question about the module, it is likely that other students have the same question. Therefore, please post your question on the forum rather than emailing the teaching team directly.
- If you have a question about your personal circumstances, please email the teaching team directly.

Attendance

- Attendance is compulsory for all lectures and labs.
- Attendance monitoring is performed by the University - the teaching team does not mark attendance, nor have the ability to change your attendance record.
- If you are unable to attend, you must obtain an authorised absence via the University's "Extenuating Circumstances" procedure.
- Please attend the lab session on your timetable.
 - Lab groups are organized by the University timetabling team. The teaching team cannot change your assigned group.

Academic Integrity and Practical Advice

- You are expected to complete all module work independently.
- Please be familiar with the University's Academic Misconduct policy.
<https://www.nottingham.edu.cn/en/academic-services/academic-misconduct/academic-misconduct.aspx>
- We do check every submission for plagiarism. Every year, students are caught plagiarising and are penalised accordingly. You do not want to be one of these students.
- Our practical advice are:
 - If you are unsure about what constitutes plagiarism, please ask the teaching team.
 - Do not copy code from the internet without referencing it.
 - Do not share your code with other students.
 - Do not share your code on public repositories (e.g. GitHub).
 - Be cautious of your dorm-mates and friends asking for your code.

An Insight into Data Modelling and Analysis

How Much Data Is Collected Every Minute of the Day?

Some stats as of late 2024:

- 5.52 billion people – approximately 67.5% of the global population – are online
- Total amount of data created, captured, copied, and consumed globally is expected to reach 149 zettabytes by the end of 2024, with projections surpassing 394 zettabytes by 2028.



[https://www.domo.com/learn/infographic/data-never-sleeps-12](https://www.domo.com/learn/infoographic/data-never-sleeps-12)

DMA as a Quest for Efficient Planning

Sumerian Cuneiform Tablet Account of workers.

Some of the earliest records of data collection and analysis:

- Sumerian Cuneiform Clay Tablets Recording Labour Workforce Data 4000 BC
- To plan food requirements for each member of the population
- Record beer given to workers as part of their daily rations

Sources: 1 2 3



Sumerian Cuneiform Tablet Record of Beer.

DMA as a Quest for Survival: Data Visualisation

- Florence Nightingale the Statistician who knew how to communicate data
- Source: [here](#)

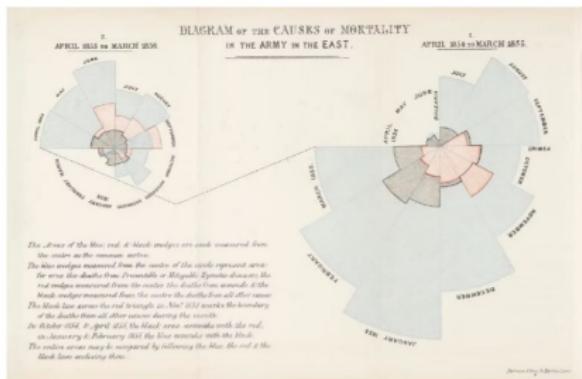


Image Source: Diagram originally from Florence Nightingale's book, Notes on matters affecting the health, efficiency, and hospital administration of the British Army.

<https://wellcomecollection.org/works/jxwtskzc/items>

Video: Florence Nightingale - Joy of Stats

Interactive Visualisation: The Joy of Stats

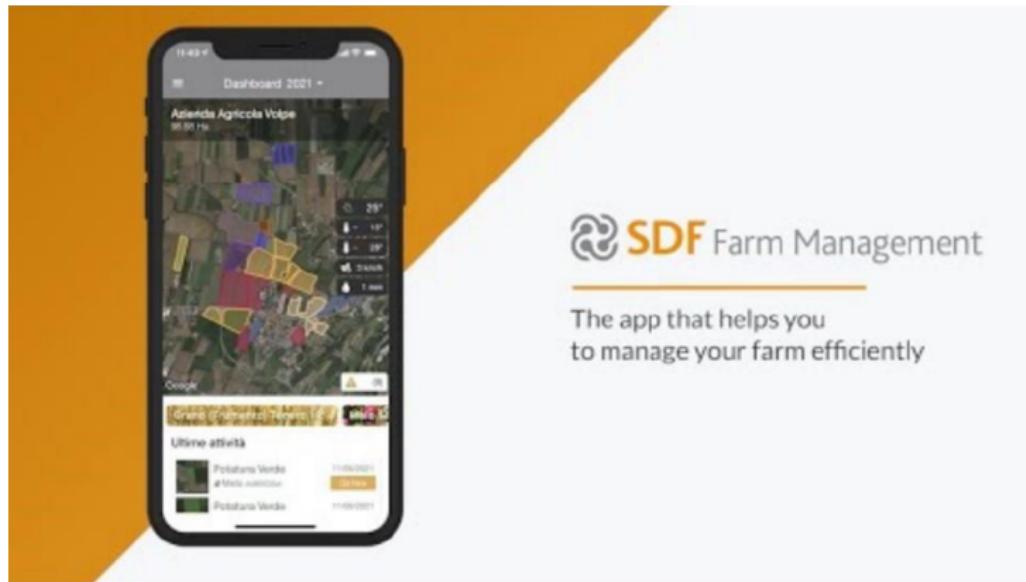
- Hans Rosling
- Gapminder Tools: Interactive Bubbles Chart



Video Source: Hans Rosling: The Joy of Stats ([YouTube Link](#))

Ubiquitous Computing – Everywhere and Anywhere Data

- Sensemaking environment of connected devices, interactive visualizations, and multiple temporal and contextually varying data sources and dependencies.
- Tools for augmenting and transforming our cognitive activities



Video Source: SDF Farm Management ([YouTube Link](#))

ACTIVITY . THINK-PAIR-SHARE

- Pair up with the person next to you and discuss the challenges and difficulties that will be faced, as shown in the video on the previous slide.
- Share your answers with us!



The app that helps you
to manage your farm efficiently

SDF Farm Management

Ubiquitous Computing – Everywhere and Anywhere Data

- Is the farmer better informed to carry out their role than your doctor?
- What are the ethical issues of transferring this model to people?
- The Reality: link



Video Source: <https://www.youtube.com/watch?v=aThi1z0lL4>



Data Modelling and Analysis Pipeline

Data Modelling and Analysis Pipeline

- The building block of Data Science

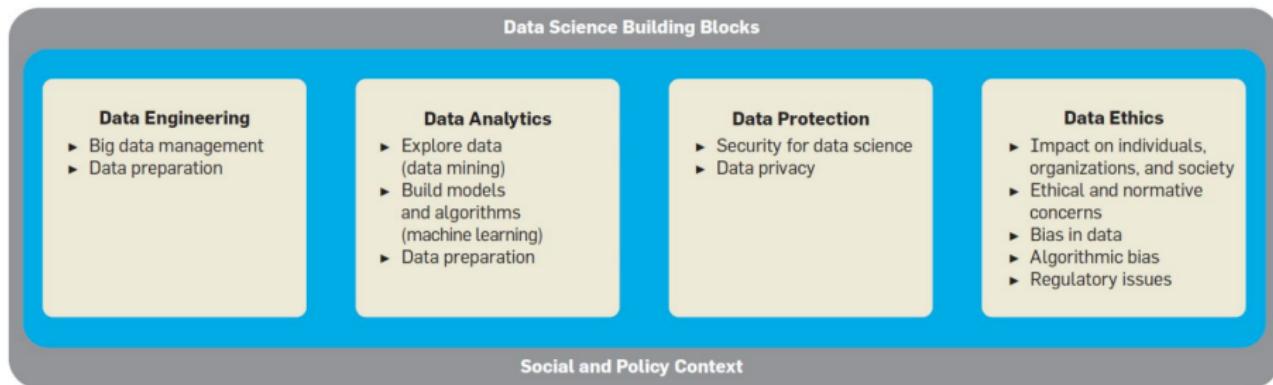
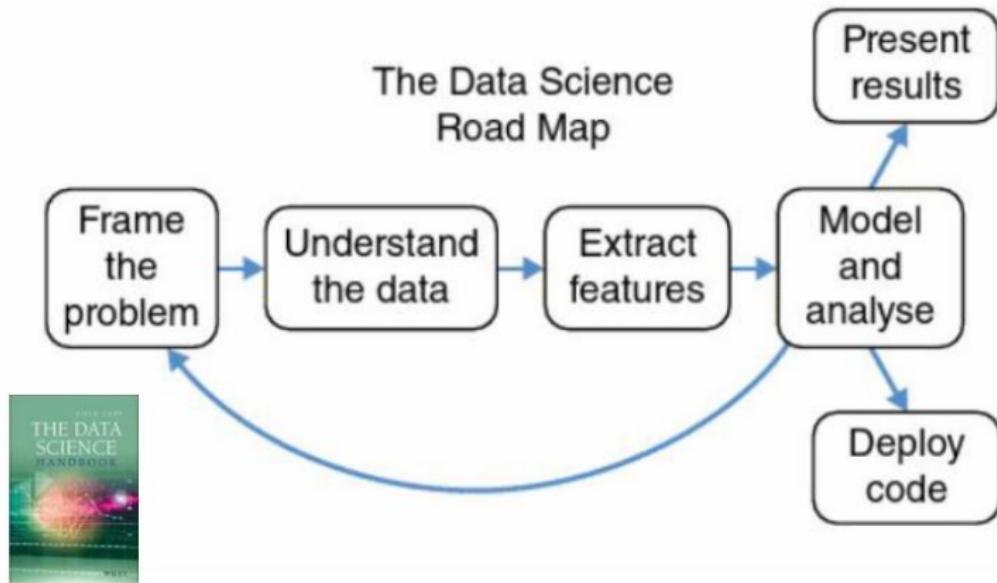


Image Source: Data Science by Ozsu

Data Modelling and Analysis Pipeline

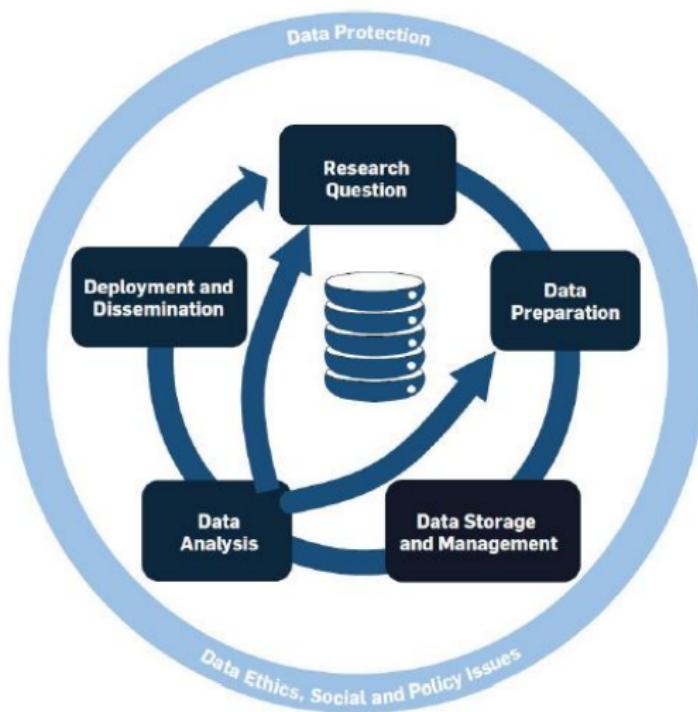
- Understand how to apply the Data Science investigation cycle



From: The Data Science Handbook, Field Cady

Data Modelling and Analysis Pipeline

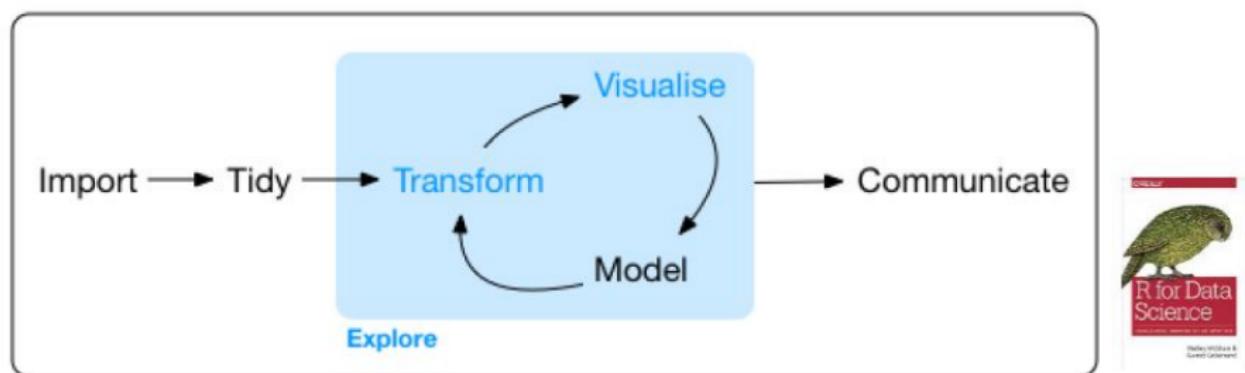
- Understand how to apply the Data Science investigation cycle



From: Data Science – A Systematic Treatment, Ozsu, Comms ACM 2023

Data Modelling and Analysis Pipeline

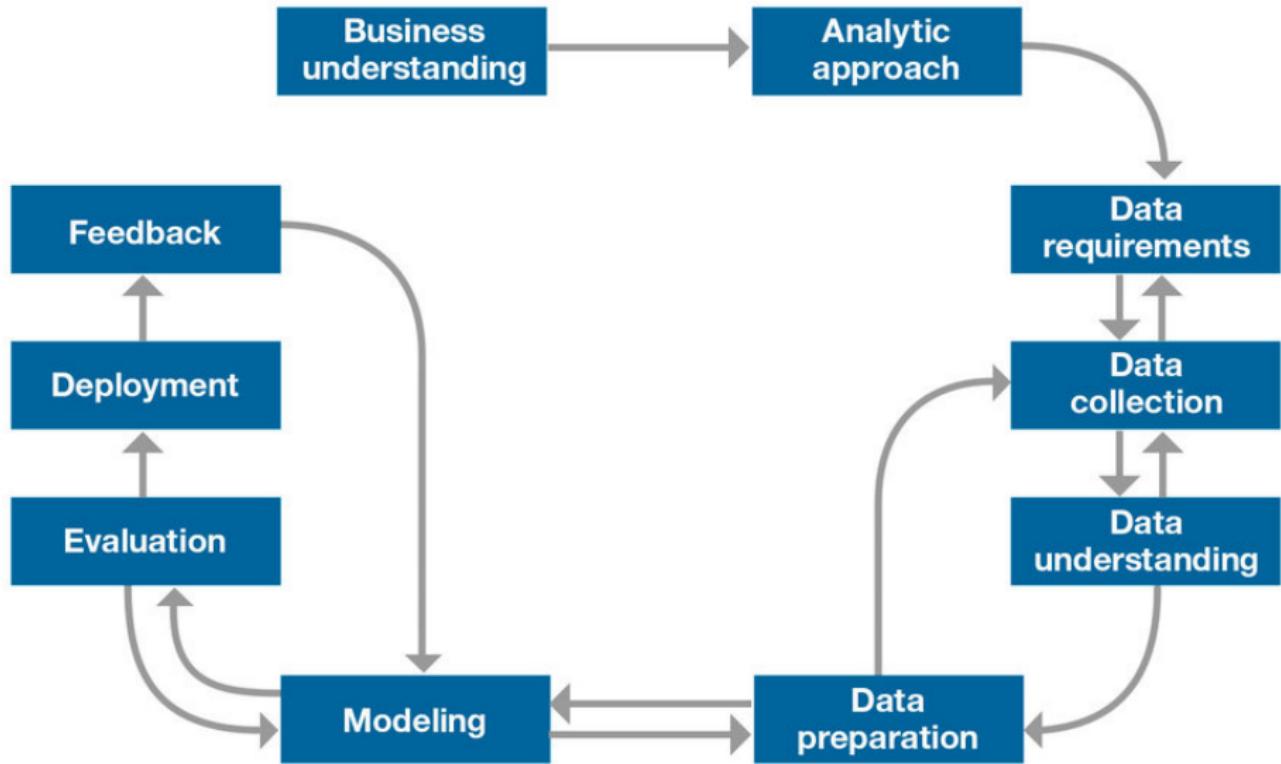
- Understand how to apply the Data Science investigation cycle



From: R for Data Science, Wickham and Grolemund

Data Modelling and Analysis Pipeline

- IBM



Data Modelling and Analysis Pipeline

- Google Cloud
- <https://cloud.google.com/blog/topics/developers-practitioners/intro-data-science-google-cloud>

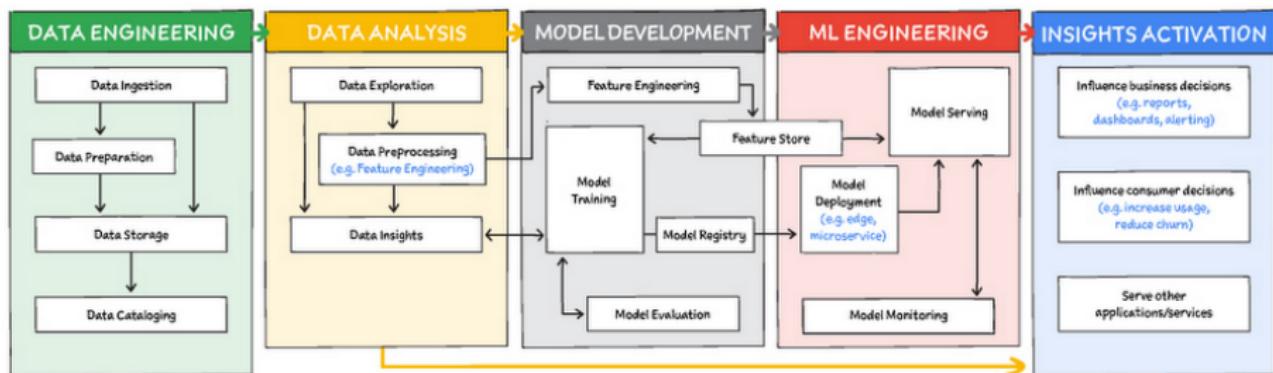


Image Source: Intro to Data Science on Google Cloud

Data Engineering

Objective: Prepare and organize raw data for downstream analysis and model development.

1. Data Ingestion:

- Collect data from various sources (databases, APIs, streaming data, etc.).
- Formats: CSV, JSON, databases, etc.

2. Data Preparation:

- Cleaning, deduplication, and integration of datasets.
- Handling missing values and ensuring data consistency.

3. Data Storage:

- Centralized storage using data lakes, warehouses, or cloud storage.
- Examples: AWS S3, Google BigQuery.

4. Data Cataloging:

- Metadata management for better discoverability.
- Tools: Data catalog services to enable seamless data discovery.

Data engineering ensures data quality and accessibility to support reliable analysis and modeling.

Data Analysis

Objective: Extract insights and prepare data for modeling.

1. Data Exploration:

- Perform exploratory data analysis (EDA) using visualizations and summary statistics.
- Tools: Python (pandas, matplotlib), SQL, Tableau, etc.

2. Data Preprocessing (Feature Engineering):

- Transform raw data into meaningful features for modeling.
- Techniques: normalization, encoding categorical variables, creating derived features.

3. Data Insights:

- Generate actionable insights to understand trends and patterns.
- Example: Identifying correlation, seasonality, or data anomalies.

Data analysis bridges the gap between raw data and useful insights, setting the stage for successful model development.

Model Development

Objective: Train, evaluate, and optimize machine learning models.

1. Feature Engineering:

- Improve model performance by selecting, transforming, and creating features.

2. Model Training:

- Train machine learning models using algorithms such as linear regression, decision trees, neural networks, etc.
- Tools: Scikit-learn, TensorFlow, PyTorch.

3. Model Evaluation:

- Assess the model's performance using metrics (e.g., accuracy, precision, recall, F1-score).
- Perform cross-validation to validate the model's robustness.

4. Model Registry:

- Version control and management of models to ensure traceability.

Model development transforms features into actionable predictive capabilities through model training and evaluation.



Objective: Deploy and monitor machine learning models for real-world applications.

1. Model Deployment:

- Deploy models as APIs, microservices, or edge devices.
- Deployment types: Batch, real-time, and on-device.

2. Model Serving:

- Serve predictions to applications or dashboards.
- Infrastructure tools: Kubernetes, Docker, AWS Lambda.

3. Model Monitoring:

- Track model performance in production (accuracy drift, latency).
- Monitor data for changes that can affect model performance.

ML engineering ensures that models are effectively integrated into business workflows and remain accurate over time.

Insights Activation

Objective: Leverage model outputs and data insights to support decision-making.

1. Influence Business Decisions:

- Use insights to generate reports, dashboards, and alerts.
- Example: Dashboards showing real-time KPIs for decision-making.

2. Influence Consumer Decisions:

- Personalize recommendations, offers, and experiences.
- Example: Reducing customer churn or increasing user engagement.

3. Serve Other Applications/Services:

- Integrate insights with external systems (e.g., customer service platforms, fraud detection).

Insights activation drives business value by turning model outputs into meaningful actions that improve processes and user outcomes.

COMP4131: Data Modelling and Analysis

Lecture 2: Data Wrangling and Pre-processing

Kian Ming Lim

University of Nottingham Ningbo China

kian-ming.lim@nottingham.edu.cn

February 20, 2025

Outline

- ① Introduction to Data Wrangling and Pre-processing
- ② The Data
- ③ Data Cleaning
- ④ Data Transformation
- ⑤ Data Integration and Reduction
- ⑥ The Complete ML Workflow

Introduction to Data Wrangling and Pre-processing

Data Wrangling and Pre-processing

Data wrangling and data pre-processing are interconnected processes in the data preparation pipeline that involve cleaning, transforming, and organizing raw data into a structured and optimized format suitable for analysis, modeling, or decision-making.

Data Wrangling and Pre-processing

- Data wrangling is the process of cleaning, transforming, and organizing raw data into a usable format.
- It involves handling inconsistencies, missing values, and errors, as well as integrating data from multiple sources to prepare it for analysis or further processing.



Workflow of the Data Wrangling Process

Data Wrangling and Pre-processing

- Data pre-processing is the process of preparing data for analysis or machine learning after it has been wrangled.
- It ensures the data is optimized for specific analytical or modeling purposes via techniques such as:
 - Scaling / normalization
 - Encoding categorical variables
 - Handling outliers
 - Splitting data into training and testing sets

Why is Data Wrangling and Pre-processing Important?

Real-World Data Challenges:

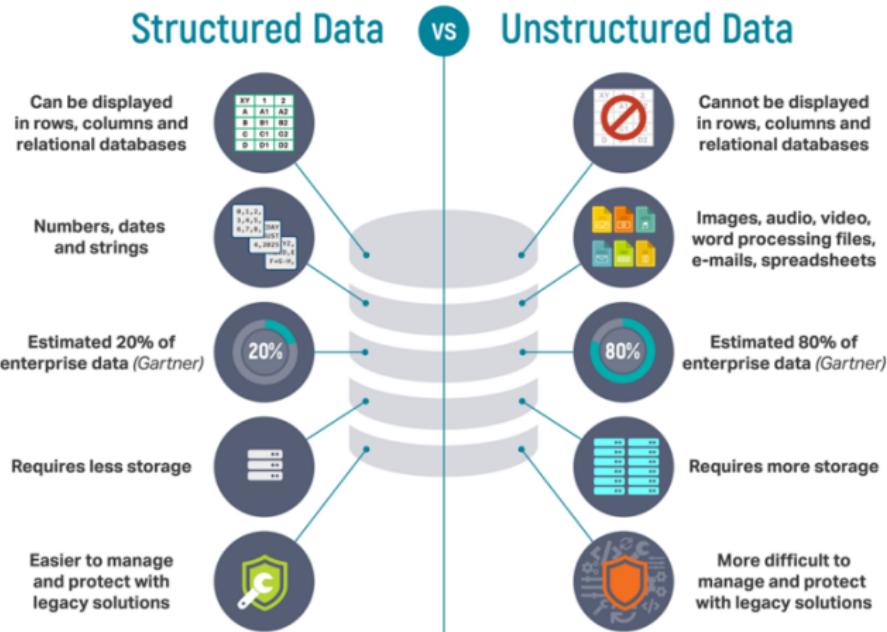
- **Missing values:** Incomplete data that can lead to biased results.
- **Duplicates:** Repeated records that inflate or skew analysis.
- **Inconsistent formats:** Non-standardized data, such as mixed date formats or varying units.
- **Outliers:** Extreme values that can distort statistical calculations.

Impact on Analysis:

- Poor-quality data leads to **inaccurate insights** and **unreliable models**.
- **Garbage in, garbage out (GIGO):** Results are only as good as the data used.

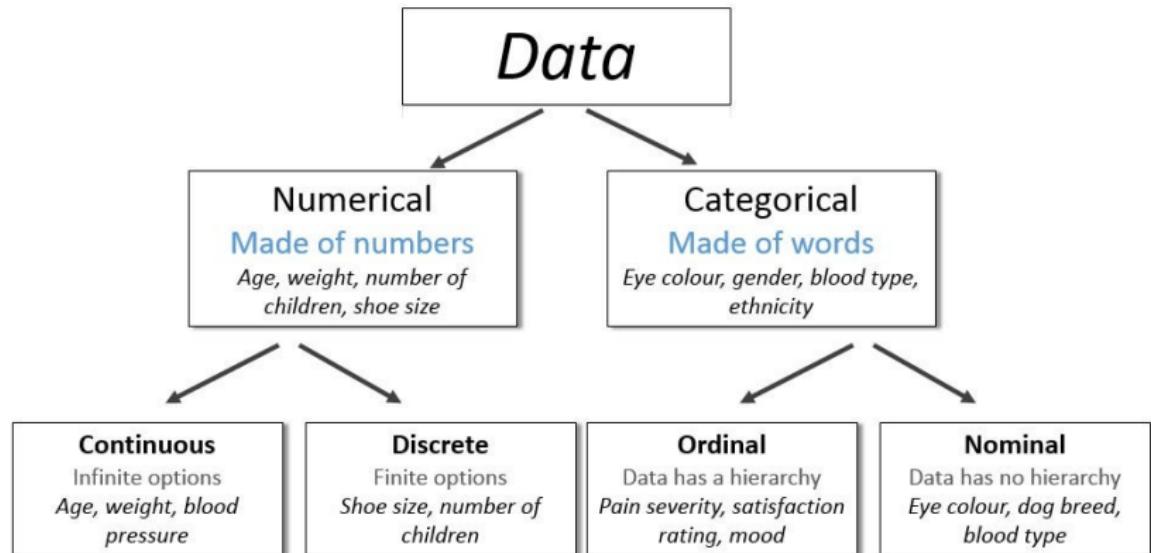
The Data

Types of Data



Source: Lawtomated, Structured vs. Unstructured Data: What are they and why care?

Types of Data



Source: Medium: Data Science Basics.

Common Notations

**Examples
(Samples,
Instances,
Observations)**

Feature (Attribute)		Label (Class, Output, Target)	
housingMedianAge (feature)	totalRooms (feature)	totalBedrooms (feature)	medianHouseValue (label)
15	5612	1283	66900
19	7650	1901	80100
17	720	174	85700
14	1501	337	73400
20	1454	326	65500

Feature and Label

Features:

- A feature is an input variable—the x variable in machine learning.
- A simple machine learning project might use a single feature, while a more sophisticated machine learning project could use millions of features, specified as:

$$x_1, x_2, \dots, x_N$$

Labels:

- A label is the thing we're predicting—the y variable in machine learning.

Examples

Examples:

- An **example** is a particular instance of data, \mathbf{x} . (We put \mathbf{x} in boldface to indicate that it is a vector.)
- Examples can be categorized into:
 - **Labeled examples** - Includes both feature(s) and the label. That is:

Labeled examples: $\{\text{features, label}\} : (\mathbf{x}, y)$

- **Unlabeled examples** - Contains features but not the label. That is:

Unlabeled examples: $\{\text{features}\} : (\mathbf{x})$

Data Cleaning

Handling Missing Values

What are Missing Values?

- Missing data can occur due to errors in data collection, storage, or processing.

Impact:

- Missing data can lead to bias or incorrect analysis.

	A	B	C	D
0	1.0	2.0	3.0	4.0
1	5.0	6.0	NaN	8.0
2	10.0	11.0	12.0	NaN

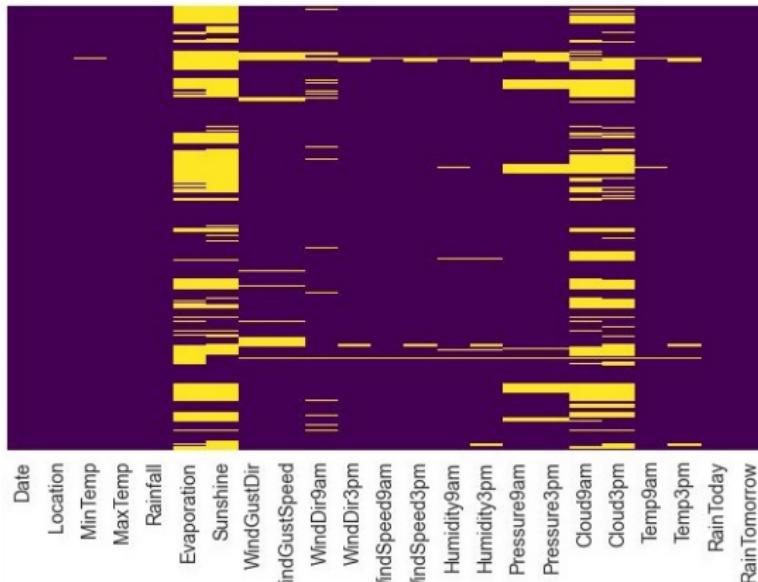
Identifying Missing Data

Methods to Identify Missing Data:

- Visual inspection (e.g., heatmaps).
- Programmatic methods: `isnull()` or `isna()` in Pandas.

Example:

- A dataset with missing values in columns.



Handling Missing Values

Common Techniques:

- Remove missing values (rows/columns).
- Impute missing values (mean, median, or mode).
- Forward/backward fill: use previous or next values to fill missing data.

The diagram illustrates two ways to handle missing data in a DataFrame. On the left, a DataFrame has three rows (0, 1, 2) and four columns (A, B, C, D). Row 1 contains a missing value (NaN) in column C. An arrow labeled "Remove rows" points from the original DataFrame to a smaller version on the right, which only contains rows 0 and 2. Another arrow labeled "Remove cols" points from the original DataFrame to a smaller version on the right, which only contains columns A and B for all three rows.

	A	B	C	D
0	1.0	2.0	3.0	4.0
1	5.0	6.0	NaN	8.0
2	10.0	11.0	12.0	NaN

	A	B
0	1.0	2.0
1	5.0	6.0
2	10.0	11.0

Handling Missing Values

Mean Imputation:

$$x_{\text{new}} = \frac{\sum x_i}{n}$$

	A	B	C	D		A	B	C	D	
0	1.0	2.0	3.0	4.0		0	1.0	2.0	3.0	4.0
1	5.0	6.0	NaN	8.0	Impulation	1	5.0	6.0	7.5	8.0
2	10.0	11.0	12.0	NaN		2	10.0	11.0	12.0	6.0

Outlier Detection and Treatment

What are Outliers?

- A data point that significantly deviates from other observations.

Impact:

- Outliers can distort statistical analysis and model performance.

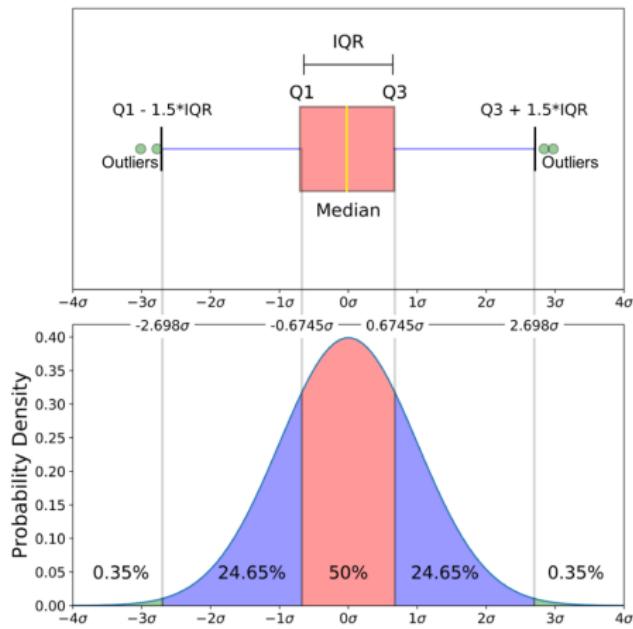
Methods to Identify Outliers:

- Z-score.
- IQR (Interquartile Range).

Methods to Identify Outliers

Z-score:

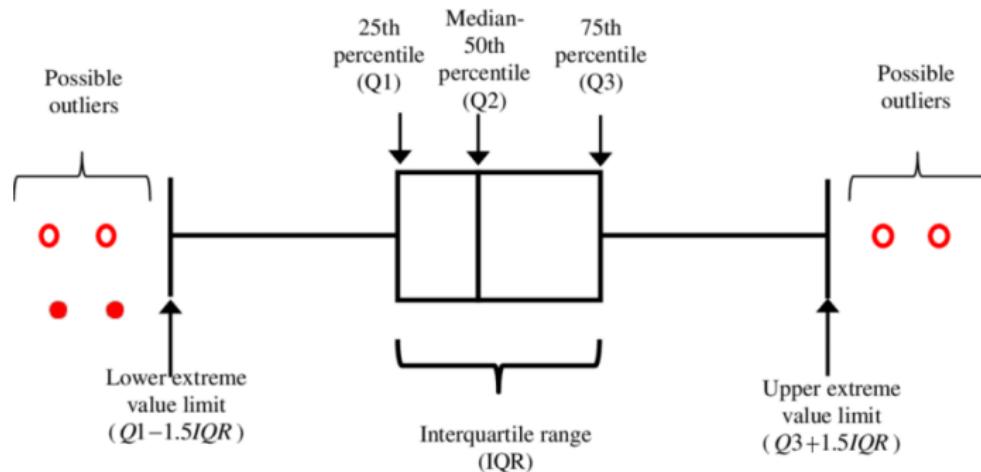
- Measures how far a data point is from the mean in terms of standard deviations.
- Formula: $Z = \frac{(x-\mu)}{\sigma}$



Methods to Identify Outliers

Interquartile Range (IQR):

- Identifies outliers as data points outside $1.5 \times \text{IQR}$.
- Formula: $\text{IQR} = Q3 - Q1$



Handling Outliers

Remove:

- Delete outlier rows.

Cap:

- Replace outliers with a threshold value (e.g., upper/lower bounds).

Transform:

- Apply log or square root transformations to reduce the impact of outliers.

Dealing with Duplicates

What are Duplicates?

- Duplicates can occur due to data entry errors or merging datasets.

Impact:

- Duplicates can lead to overcounting and biased results.

Methods:

- Identify and remove duplicates.

Data Transformation

Introduction to Data Transformation

What is Data Transformation?

- The process of converting data into a suitable format for analysis or modeling.

Why is it Important?

- Improves the performance of the models.
- Ensures data is on a consistent scale.
- Handles categorical and continuous data appropriately.

Key Techniques:

- Feature scaling.
- Encoding categorical data.
- Data binning.
- Feature engineering.
- Handling imbalanced data.

Feature Scaling

What is Feature Scaling?

- Rescaling features to a specific range (e.g., 0 to 1) or standardizing them to have a mean of 0 and a standard deviation of 1.

Why is it Important?

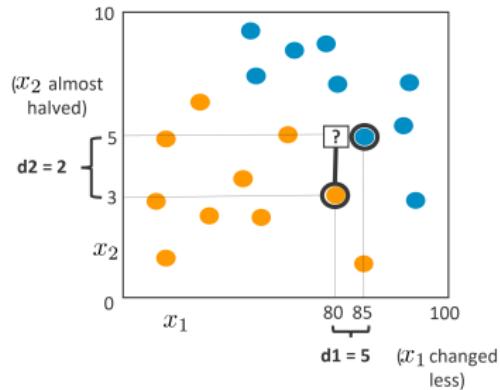
- Ensures all features contribute equally to the model.
- Prevents features with larger scales from dominating.

Common Methods:

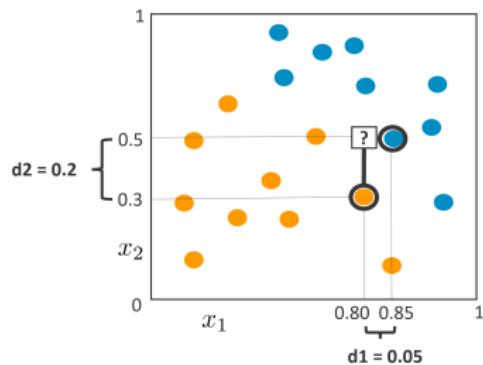
- Normalization (Min-Max scaling).
- Standardization (Z-score scaling).

Feature Scaling

Unscaled features: closer to



Scaled features: closer to



Source: AWS Machine Learning University (MLU)

Normalization vs. Standardization

Normalization (Min-Max Scaling):

- Rescales data to a range of [0, 1].
- Formula: $X_{\text{norm}} = \frac{X - X_{\min}}{X_{\max} - X_{\min}}$

Standardization (Z-score Scaling):

- Rescales data to have a mean of 0 and a standard deviation of 1.
- Formula: $X_{\text{std}} = \frac{X - \mu}{\sigma}$

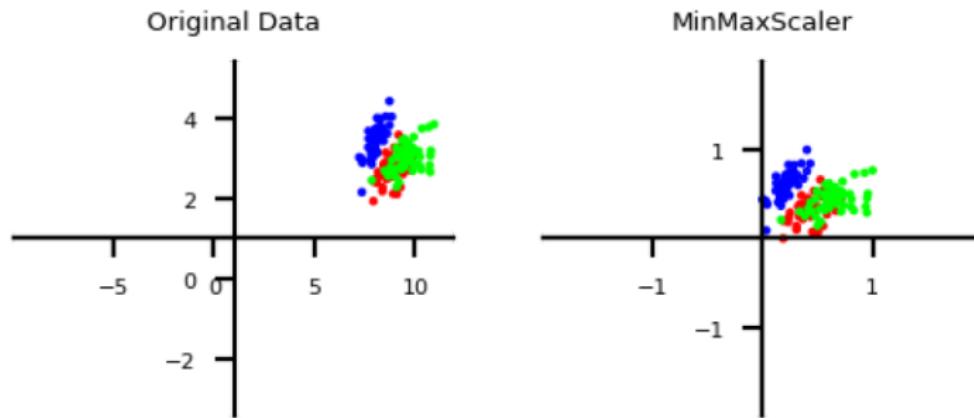
When to Use:

- Normalization: When data distribution is unknown or not Gaussian.
- Standardization: When data follows a Gaussian distribution.

Normalization

Normalization (Min-Max Scaling):

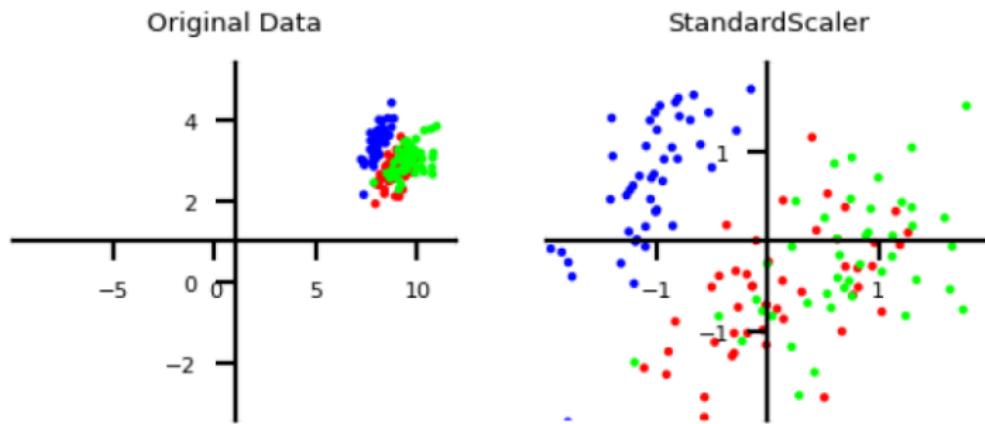
- Scales all features between a given min and max value (e.g. 0 and 1).
- Makes sense if min/max values have meaning in your data.
- Sensitive to outliers.



Standardization

Standardization (Z-score Scaling):

- Generally most useful, assumes data is more or less normally distributed.
- Per feature, subtract the mean value μ , scale by standard deviation σ



Encoding Categorical Data

What is Categorical Data?

- Data that represents categories (e.g., gender, color, country).

Why Encode Categorical Data?

- Most machine learning algorithms require numerical input.

Common Encoding Techniques:

- One-hot encoding.
- Label encoding.
- Binary encoding.

One-hot Encoding, Label Encoding, Binary Encoding

One-hot Encoding:

- Converts each category into a binary vector.
- Example: Red = [1, 0, 0], Green = [0, 1, 0], Blue = [0, 0, 1].

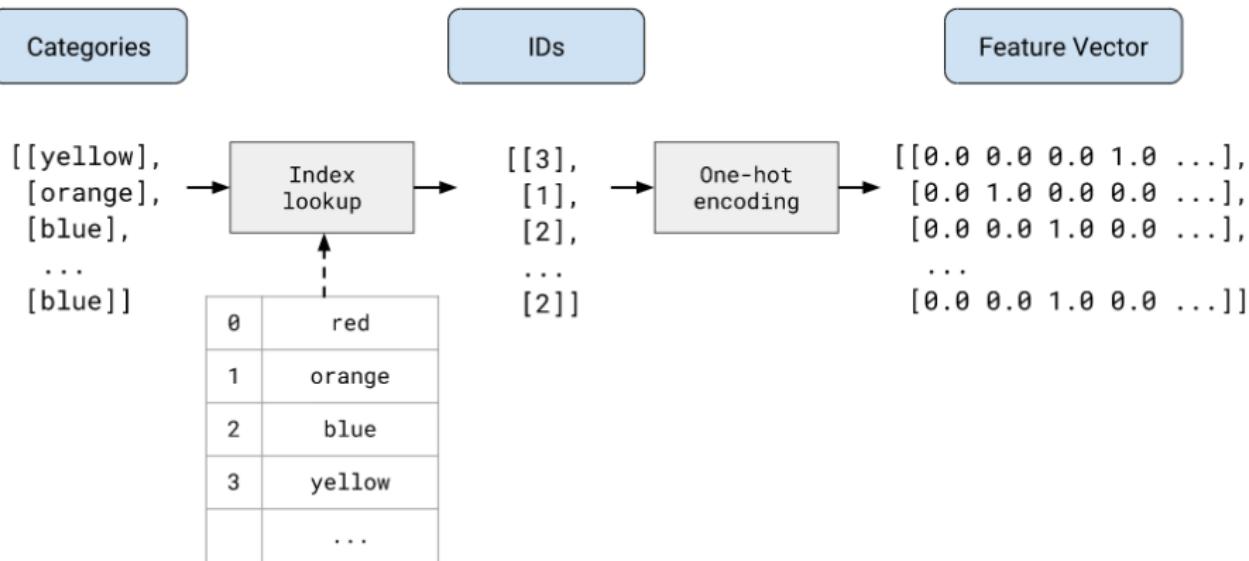
Label Encoding:

- Assigns a unique integer to each category.
- Example: Red = 0, Green = 1, Blue = 2.

Binary Encoding:

- Combines label encoding with binary representation.
- Example: Red = 00, Green = 01, Blue = 10.

One-hot Encoding



Data Binning

What is Data Binning?

- Converting continuous data into discrete categories (bins).

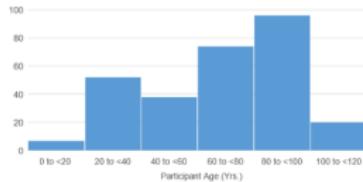
Why Use Data Binning?

- Simplifies complex data.
- Handles outliers.
- Improves model performance for certain algorithms.

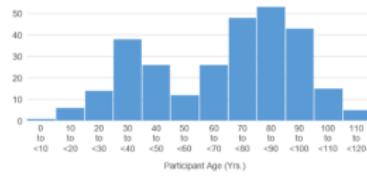
Example (Age):

- 0-18 = "Child",
- 19-35 = "Young Adult",
- 36-60 = "Adult",
- 60+ = "Senior".

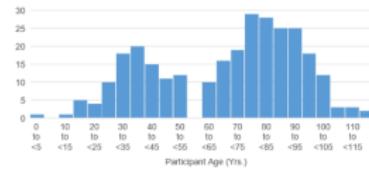
Data Binning



6 bins



12 bins



24 bins

Feature Engineering

What is Feature Engineering?

- The process of creating new features or modifying existing ones to improve model performance.

Techniques:

- Adding new features (e.g., calculating ratios or differences).
- Modifying features (e.g., log transformation).
- Dropping irrelevant or redundant features.

Example:

- Creating a "BMI" feature from height and weight.

Handling Imbalanced Data

What is Imbalanced Data?

- When one class significantly outnumbers the other(s) in a classification problem.

Why is it a Problem?

- Models may become biased toward the majority class.

Techniques to Handle Imbalanced Data:

- Resampling: Oversampling the minority class or undersampling the majority class.
- Synthetic Data Generation: Using techniques like SMOTE, GAN.
- Algorithmic Approaches: Using class-weighted models.

ACTIVITY . THINK-PAIR-SHARE

- Pair up with the person next to you and identify what are the issues in the following heart disease classification data set. What are the solutions?

PatientID	Age	Sex	ChestPainType	RestingBP	Cholesterol	FastingBS	RestingECG	MaxHR	ExerciseAngina	Oldpeak	ST_Slope	HeartDisease
1	52	M	TA	118	186	0	LVH	190	N	0	Flat	0
2	58	M	ASY	136	203	1	Normal	123	Y	1.2	Flat	1
3	44	M	ASY	110	197	0	LVH	177	N	0	Up	1
4	43	M	TA	120	291	0	ST	155	N	0	Flat	1
5	55	F	ATA	132	342	0	Normal	166	N	1.2	Up	0
6	66	M	ASY	112	212	0	LVH	132	Y	0.1	Up	1
7	55	M	NAP			0	Normal	155	N	1.5	Flat	1
8	53	M	ASY	123	282	0	Normal	95	Y	2	Flat	1
9	42	M	ASY	136	315	0	Normal	125	Y	1.8	Flat	1
10	43	M	ASY	110	211	0	Normal	161	N	0	Up	0
11	59	M	ASY	125		1	Normal	119	Y	0.9	Up	1
12	46	M	ASY	140	311	0	Normal	120	Y	1.8	Flat	1
13	42	M	ATA	120	198	0	Normal	155	N	0	Up	0
14	39	M	ASY	110	273	0	Normal	132	N	0	Up	0
15	50	F	ASY	110	254	0	LVH	159	N	0	Up	0
16	60	M	ASY	142	216	0	Normal	110	Y	2.5	Flat	1
17	56	M	ASY	115		1	ST	82	N	-1	Up	1
18	44	F	NAP	118	242	0	Normal	149	N	0.3	Flat	0
19	60	M	ASY	136	195	0	Normal	126	N	0.3	Up	0
20	32	M	ATA	125	254	0	Normal	155	N	0	Up	0
21	58	M	ATA	125	220	0	Normal	144	N	0.4	Flat	0
22	37	F	NAP	120	215	0	Normal	170	N	0	Up	0
23	57	M	ASY	140		1	Normal	100	Y	0	Flat	1
24	69	M	ASY	145	289	1	ST	110	Y	1.8	Flat	1
25	53	F	NAP	128	216	0	LVH	115	N	0	Up	0
26	44	M	ATA	120	184	0	Normal	142	N	1	Flat	0
27	34	M	ATA	98	220	0	Normal	150	N	0	Up	0
28	77	M	ASY	125	304	0	LVH	162	Y	0	Up	1
29	74	F	ATA	120	269	0	LVH	121	Y	0.2	Up	0
30	47	M	NAP	108	243	0	Normal	152	N	0	Up	1
31	63	M	ASY	96	305	0	ST	121	Y	1	Up	1
32	56	M	NAP	155		0	ST	99	N	0	Flat	1
33	32	M	TA	95		1	Normal	127	N	0.7	Up	1
34	65	F	ASY	150	225	0	LVH	114	N	1	Flat	1
35	65	M	ASY	144	312	0	LVH	113	Y	1.7	Flat	1

Data Integration and Reduction

Data Integration and Reduction

What is Data Integration?

- The process of combining data from different sources into a unified dataset.
- Ensures consistency and usability for analysis.

What is Data Reduction?

- The process of reducing the size or complexity of the dataset.
- Aims to retain important information while improving efficiency.

Key Techniques:

- Combining datasets (merging, concatenation, joining).
- Feature selection.
- Dimensionality reduction (PCA, t-SNE, UMAP).

Combining Datasets

Why Combine Datasets?

- To enrich data by adding more features or samples.

Common Techniques:

- Merging: Combines datasets based on common keys (e.g., primary keys in databases).
- Concatenation: Stacks datasets either row-wise or column-wise.
- Joining: Combines datasets similar to SQL joins (e.g., inner, outer, left, right joins).

Combining Datasets

Initial Data Frames to merge, **a** and **b**:

	a		b	
	C1	C2	C1	C3
0	A	x	0	11
1	B	y	1	12
2	C	z	2	13

left	<table border="1"><thead><tr><th></th><th>C1</th><th>C2</th><th>C3</th></tr></thead><tbody><tr><td>0</td><td>A</td><td>x</td><td>11</td></tr><tr><td>1</td><td>B</td><td>y</td><td>12</td></tr><tr><td>2</td><td>C</td><td>z</td><td>NaN</td></tr></tbody></table> <p>Start with a and join the matching rows of "C1" from b to a</p> 		C1	C2	C3	0	A	x	11	1	B	y	12	2	C	z	NaN				
	C1	C2	C3																		
0	A	x	11																		
1	B	y	12																		
2	C	z	NaN																		
right	<table border="1"><thead><tr><th></th><th>C1</th><th>C2</th><th>C3</th></tr></thead><tbody><tr><td>0</td><td>A</td><td>x</td><td>11</td></tr><tr><td>1</td><td>B</td><td>y</td><td>12</td></tr><tr><td>2</td><td>D</td><td>NaN</td><td>13</td></tr></tbody></table> <p>Start with b and join the matching rows of "C1" from a to b</p> 		C1	C2	C3	0	A	x	11	1	B	y	12	2	D	NaN	13				
	C1	C2	C3																		
0	A	x	11																		
1	B	y	12																		
2	D	NaN	13																		
inner	<table border="1"><thead><tr><th></th><th>C1</th><th>C2</th><th>C3</th></tr></thead><tbody><tr><td>0</td><td>A</td><td>x</td><td>11</td></tr><tr><td>1</td><td>B</td><td>y</td><td>12</td></tr></tbody></table> <p>Only keep matching rows "C1" in both a and b</p> 		C1	C2	C3	0	A	x	11	1	B	y	12								
	C1	C2	C3																		
0	A	x	11																		
1	B	y	12																		
outer	<table border="1"><thead><tr><th></th><th>C1</th><th>C2</th><th>C3</th></tr></thead><tbody><tr><td>0</td><td>A</td><td>x</td><td>11</td></tr><tr><td>1</td><td>B</td><td>y</td><td>12</td></tr><tr><td>2</td><td>C</td><td>z</td><td>NaN</td></tr><tr><td>3</td><td>D</td><td>NaN</td><td>13</td></tr></tbody></table> <p>Keep ALL values and ALL rows from both a and b</p> 		C1	C2	C3	0	A	x	11	1	B	y	12	2	C	z	NaN	3	D	NaN	13
	C1	C2	C3																		
0	A	x	11																		
1	B	y	12																		
2	C	z	NaN																		
3	D	NaN	13																		

Source: Combining datasets, from Duke MIDS Practical Data Science course IDS 720 by Kyle Bradbury and Nick Eubank.

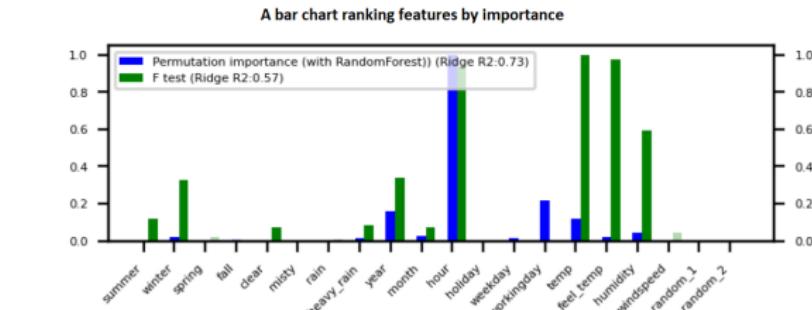
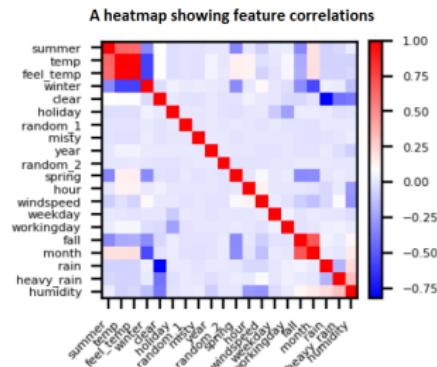
What is Feature Selection?

- The process of selecting the most relevant features for the analysis.
- Reduces dimensionality, improves interpretability, and enhances model performance.

Methods for Feature Selection

Common Methods:

- **Correlation:** Identifies features highly correlated with the target variable.
- **Variance Threshold:** Removes features with low variance.
- **Feature Importance:** Uses algorithms like Random Forest to rank feature importance.



Dimensionality Reduction

What is Dimensionality Reduction?

- Reducing the number of features while preserving important information.

Why is it Important?

- Reduces computational complexity.
- Improves model performance by removing noise.
- Visualizes high-dimensional data.

Common Techniques:

- PCA (Principal Component Analysis).
- t-SNE (t-Distributed Stochastic Neighbor Embedding).
- UMAP (Uniform Manifold Approximation and Projection).

Methods for Dimensionality Reduction

- **Principal Component Analysis (PCA):**

- Projects data onto a lower-dimensional space.
- Retains maximum variance.

- **t-SNE:**

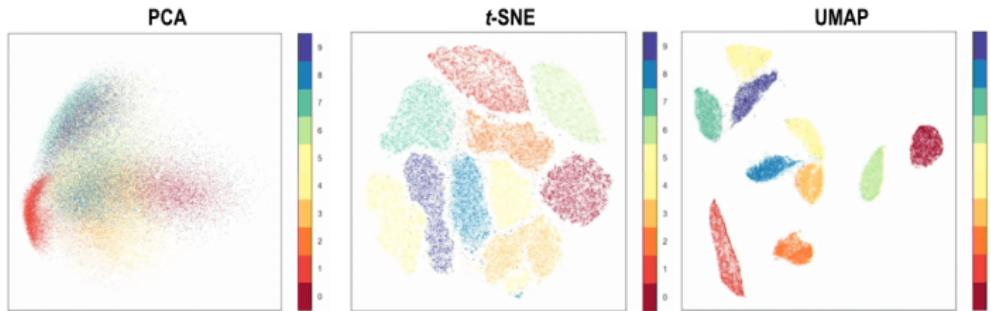
- Non-linear dimensionality reduction technique.
- Focuses on preserving local relationships in data.

- **UMAP:**

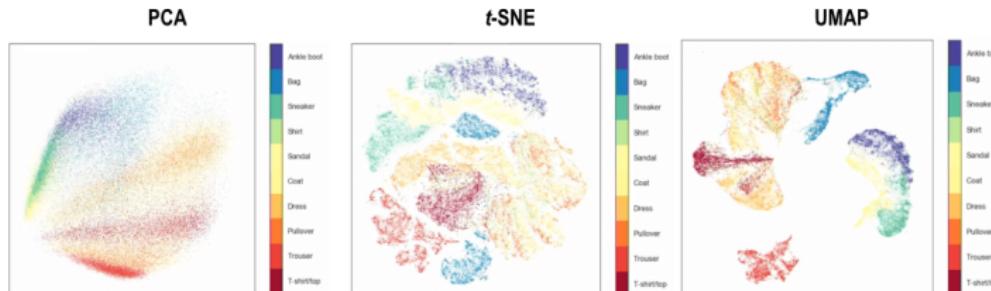
- Preserves local and global data structures.
- Suitable for clustering and visualization.

Dimensionality Reduction

MNIST Digits



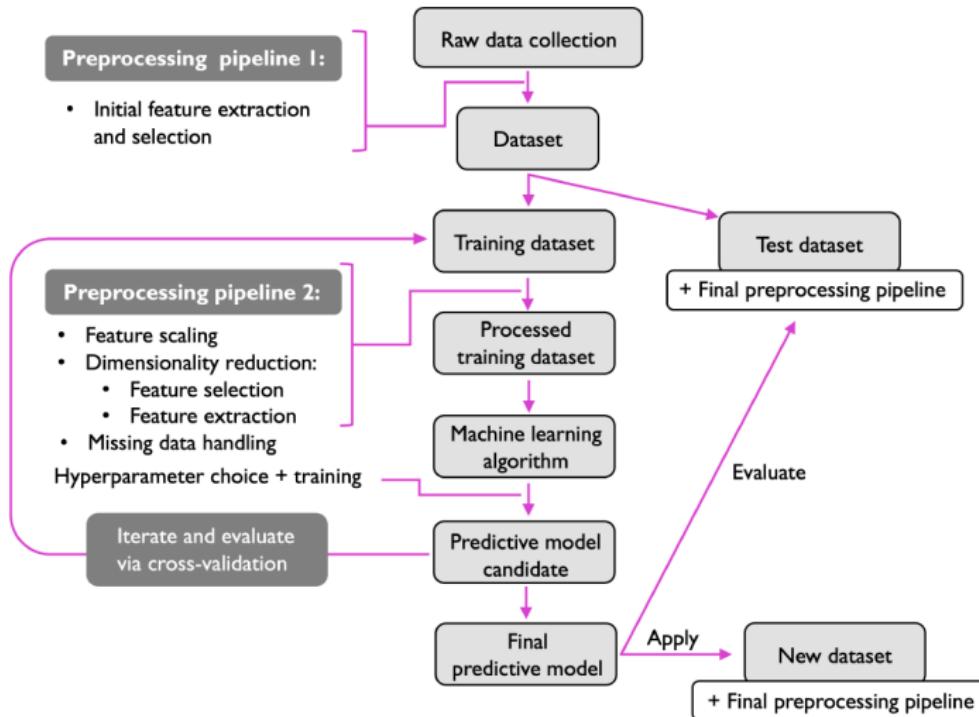
Fashion MNIST



Source: <https://meta.caspershire.net/umap/>

The Complete ML Workflow

The Complete ML Workflow



Source: STAT 451 – Introduction to Machine Learning and Statistical Pattern Classification by Sebastian Raschka

COMP4131: Data Modelling and Analysis

Lecture 3: Data Visualisation

Kian Ming Lim

University of Nottingham Ningbo China

kian-ming.lim@nottingham.edu.cn

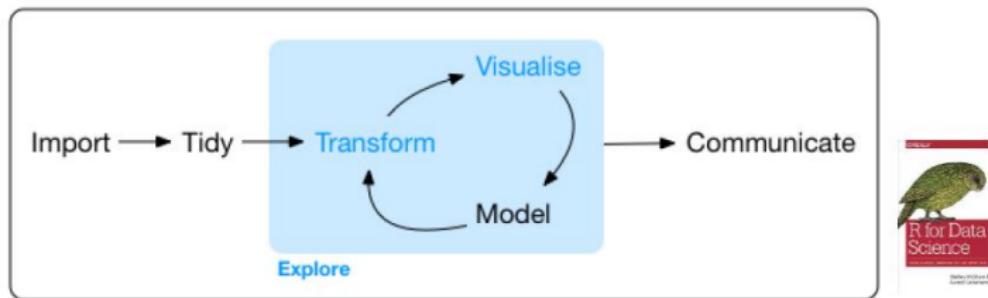
February 26, 2025

Outline

- ① Introduction to Data Visualisation
- ② Types of Data Visualisations
- ③ Principles of Effective Data Visualisation
- ④ Design and Aesthetics in Data Visualisation
- ⑤ Data Storytelling

Recap - Data Modelling and Analysis Pipeline

- We have discussed:
 - The pipeline of data modelling and analysis
 - Data wrangling and pre-processing



From: R for Data Science, Wickham and Grolemund

Learning Outcomes

By the end of this lecture, you will be able to:

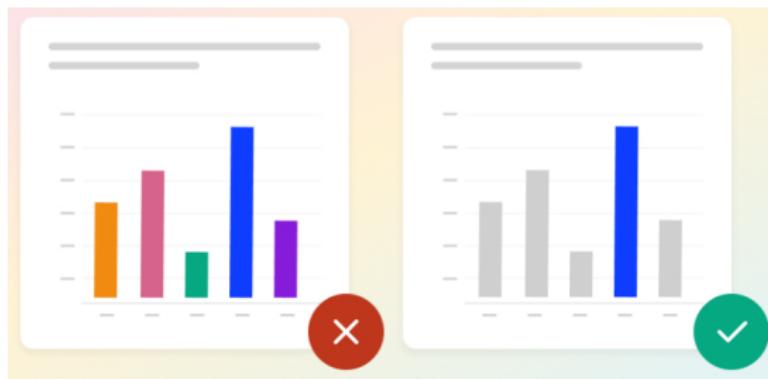
- Understand the importance of data visualisation in data analysis and decision-making.
- Identify the key principles of effective data visualisation.
- Choose the right chart types for different types of data and analysis goals.
- Apply design and storytelling techniques to create compelling visualisations.

Introduction to Data Visualisation

What is Data Visualisation?

Definition:

- Data visualisation is the graphical representation of data to communicate information clearly and effectively.
- Helps in identifying patterns, trends, and outliers.
- Enables better decision-making by presenting data in an understandable format.

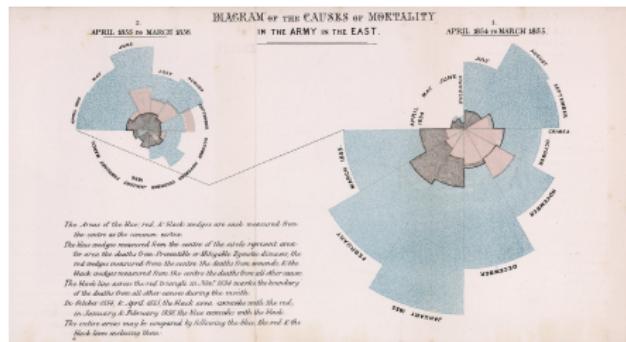


Source:

<https://www.polymersearch.com/blog/10-good-and-bad-examples-of-data-visualization>

Historical Context of Data Visualisation

- Florence Nightingale's Coxcomb chart (1858): Visualized mortality data during the Crimean War.



Source: Florence Nightingale: Pioneer of Data Visualization

- John Snow's cholera map (1854): Identified the source of a cholera outbreak in London.



Source: Vintage John Snow Cholera Map of London 1854

Key Goals of Data Visualisation

Data visualisation serves three primary goals:

- **Exploration**

- Discover patterns, trends, and relationships in data.
- Identify outliers and anomalies.

- **Analysis**

- Understand data distributions and correlations.
- Extract actionable insights from data.

- **Communication**

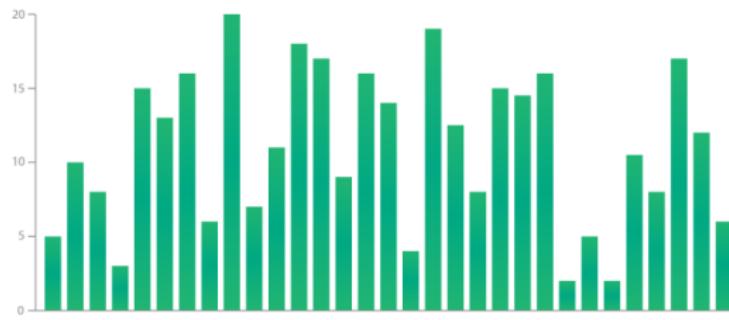
- Present findings clearly and effectively.
- Tell a compelling story with data.
- Support decision-making for stakeholders.

Types of Data Visualisations

Basic Charts: Bar Charts

Bar Charts:

- Compare categories or groups.
- Use horizontal or vertical bars.

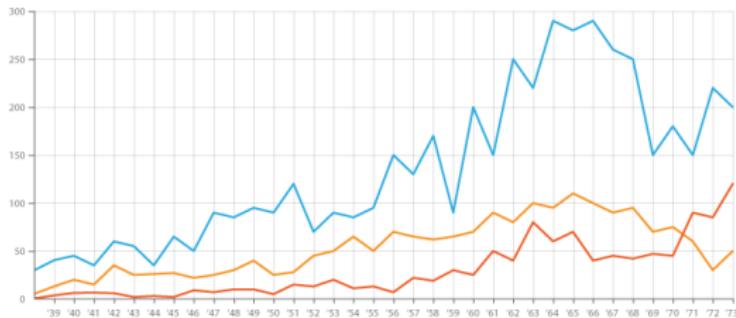


Source: Data Viz Catalogue - Bar Chart

Basic Charts: Line Charts

Line Charts:

- This chart is used to display quantitative values over a continuous interval or time period.
- A line chart is most frequently used to show trends and analyse how the data has changed over time.



Source: Data Viz Catalogue - Line Graph

Basic Charts: Pie Charts

Pie Charts:

- Pie charts help show proportions and percentages between categories, by dividing a circle into proportional segments (use sparingly).
- Each arc length represents a proportion of each category, while the full circle represents the total sum of all the data, equal to 100%.

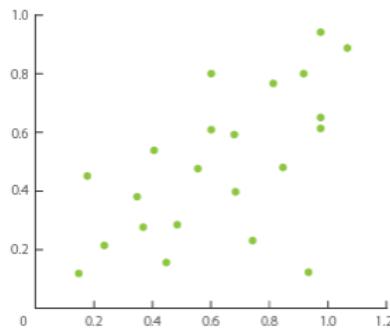
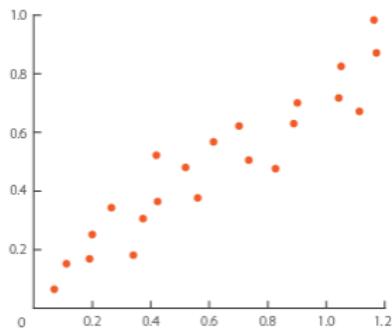


Source: Data Viz Catalogue - Pie Chart

Advanced Charts: Scatter Plots

Scatter Plots:

- Reveal relationships between two variables.
- Identify correlations or clusters.

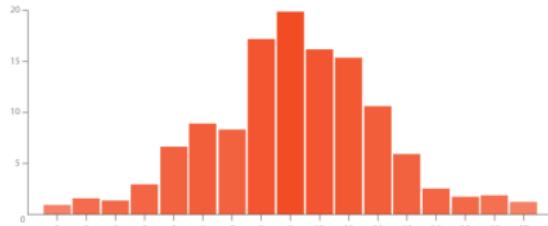


Source: Data Viz Catalogue - Scatter Plot

Advanced Charts: Histograms

Histograms:

- A histogram visualises the distribution of data over a continuous interval.
- Each bar in a histogram represents the tabulated frequency at each interval/bin.
- Histograms help give an estimate as to where values are concentrated, what the extremes are and whether there are any gaps or unusual values.

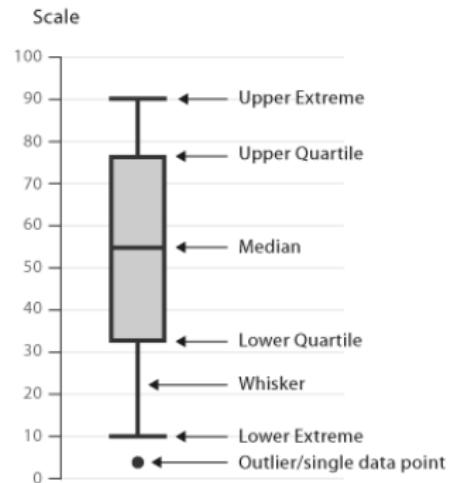


Source: Data Viz Catalogue - Histogram

Advanced Charts: Box Plots

Box Plots:

- Summarize data distribution (median, quartiles, outliers).
- The lines extending parallel from the boxes are known as the “whiskers”, which are used to indicate variability outside the upper and lower quartiles.
- Outliers are sometimes plotted as individual dots that are in-line with whiskers.

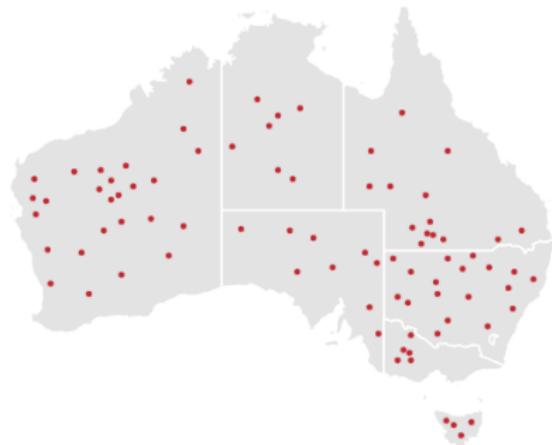


Source: Data Viz Catalogue -
Boxplot

Geospatial Visualisations: Maps

Maps:

- Display data with geographic context.
- A way of detecting spatial patterns or the distribution of data over a geographical region.
- Examples: Point maps, heatmaps.



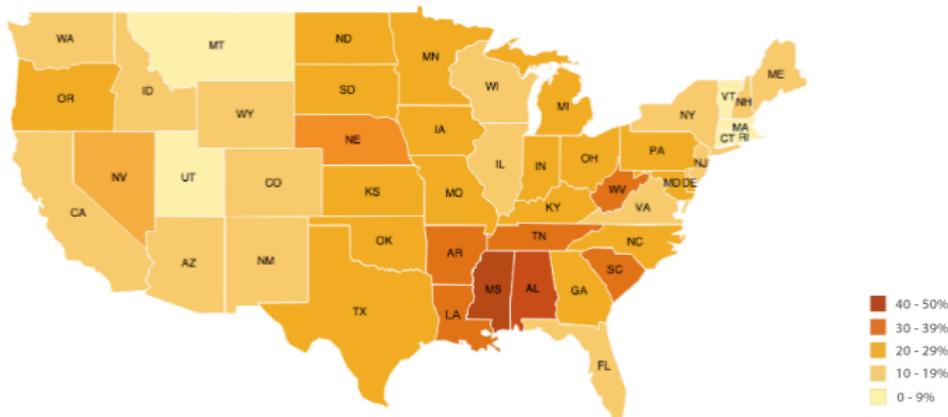
Source: Data Viz Catalogue - Dot Map

Source: Johns Hopkins University's COVID-19 dashboard.

Geospatial Visualisations: Choropleth Maps

Choropleth Maps:

- Use shading or patterns to represent data values.
- Examples: Election results, population density.

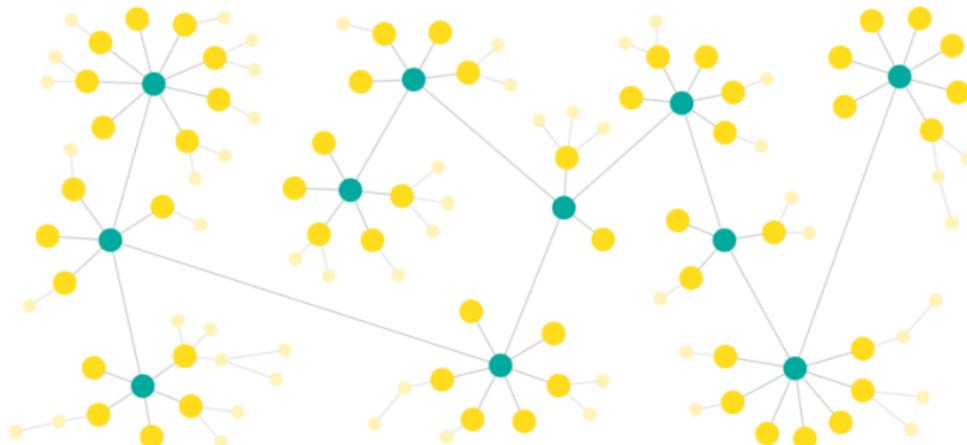


Source: Data Viz Catalogue - Choropleth Map

Network Graphs

Network Graphs:

- Visualise relationships between entities.
- Examples: Social networks, transportation networks.

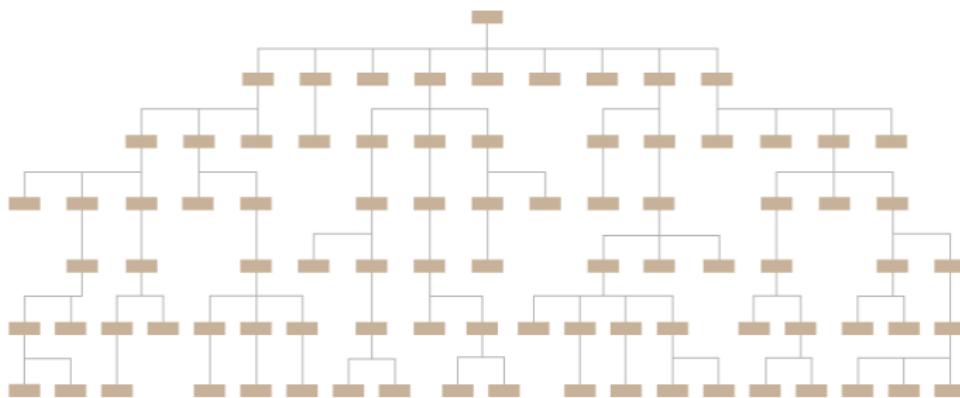


Source: From Data to Viz - Network Graph

Tree Maps

Tree Maps:

- Display hierarchical data as nested rectangles.
- Examples: File system structures, budget allocations.

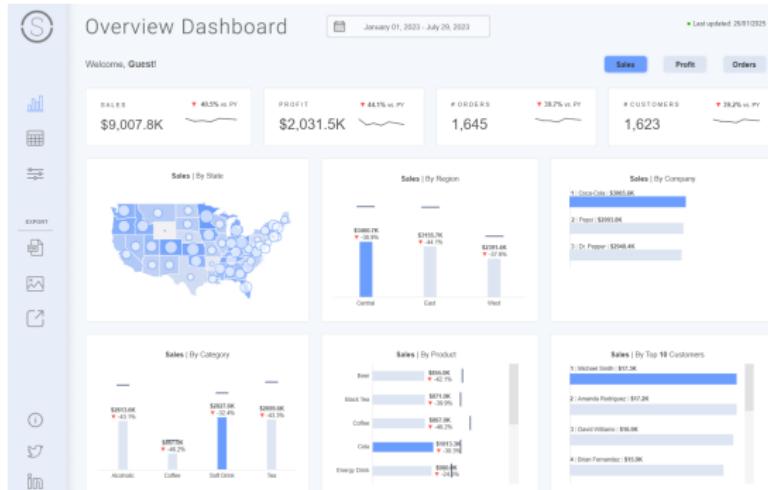


Source: Data Viz Catalogue - Tree Map

Interactive Visualisations: Dashboards

Dashboards:

- Combine multiple visualisations into a single interface.
- Examples: Business intelligence dashboards.



Source: Tableau - Data Dashboards

Principles of Effective Data Visualisation

Overview of Principles

Key Principles:

- **Clarity:** Ensure the visualisation is easy to understand.
- **Accuracy:** Represent data truthfully without distortion.
- **Efficiency:** Convey information quickly and effectively.
- **Aesthetics:** Make the visualisation visually appealing.

Choosing the Right Chart Type

Match the chart to the data:

- **Bar charts:** Compare categories.
- **Line charts:** Show trends over time.
- **Scatter plots:** Reveal relationships between variables.
- **Pie charts:** Display proportions (use sparingly).
- **Heatmaps:** Visualise density or correlations.

Importance of Color Theory

Color best practices:

- Use color to highlight, not distract.
- Choose colorblind-friendly palettes (e.g., ColorBrewer).
- Avoid using too many colors in a single chart.
- Use consistent colors for the same categories across visualisations.



Source: Data Visualization: Design Considerations
Useful link: [ColorBrewer](#)

Ethical Considerations in Data Visualisation

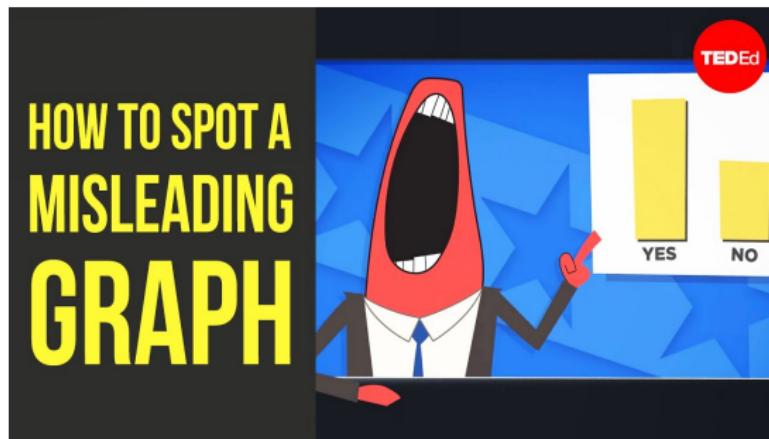
Ethical guidelines:

- Avoid manipulating data to mislead.
- Be transparent about data sources and methods.
- Respect privacy and confidentiality.
- Ensure accessibility for all audiences.

Avoiding Chart Junk and Misleading Visualisations

Common pitfalls:

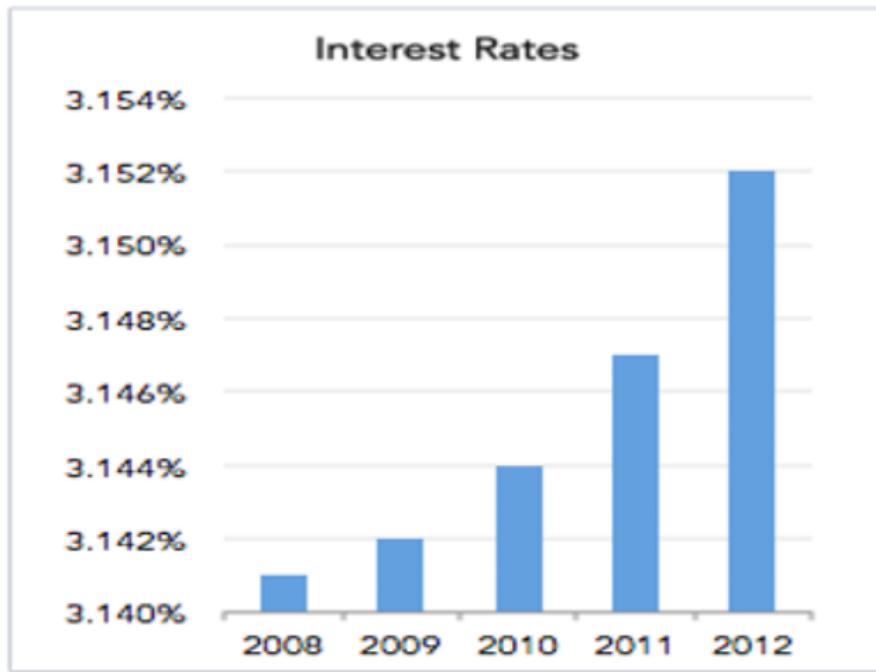
- **Chart junk:** Unnecessary decorations (e.g., 3D effects, excessive gridlines).
- **Misleading scales:** Truncated axes or inconsistent intervals.
- **Overloading:** Too much information in one chart.



Video: How to spot a misleading graph - Lea Gaslowitz

ACTIVITY . THINK-PAIR-SHARE

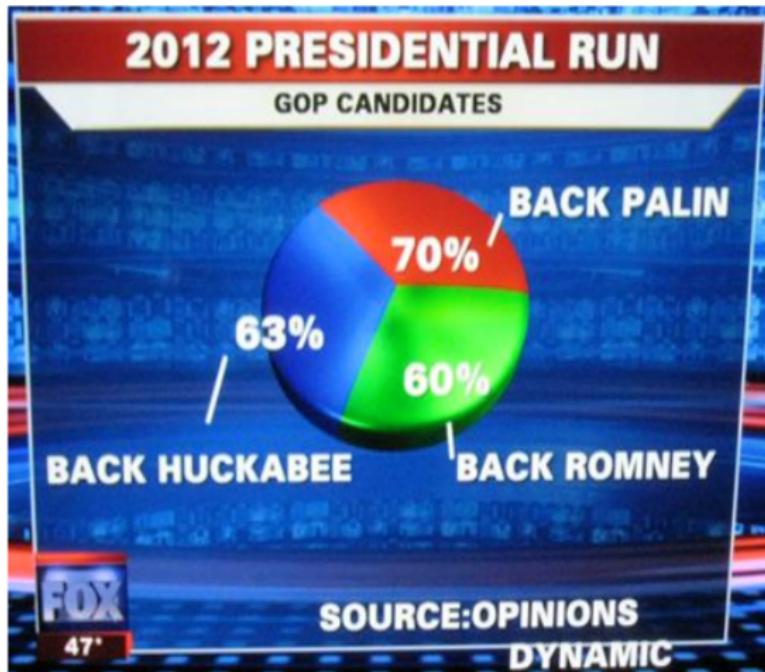
- What's wrong with this visualisation?



Source: CMSC320 - Introduction to Data Science

ACTIVITY . THINK-PAIR-SHARE

- What's wrong with this visualisation?



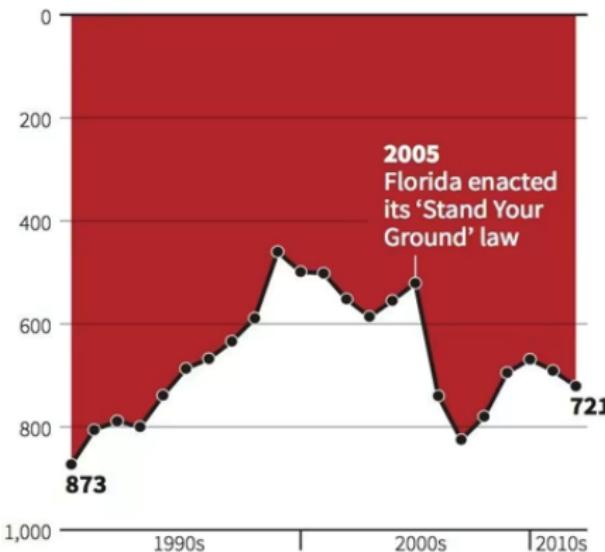
Source: CMSC320 - Introduction to Data Science

ACTIVITY . THINK-PAIR-SHARE

- What's wrong with this visualisation?

Gun deaths in Florida

Number of murders committed using firearms



Source: Florida Department of Law Enforcement

C. Chan 16/02/2014

REUTERS



Source: CMSC320 - Introduction to Data Science

Data Visualisation

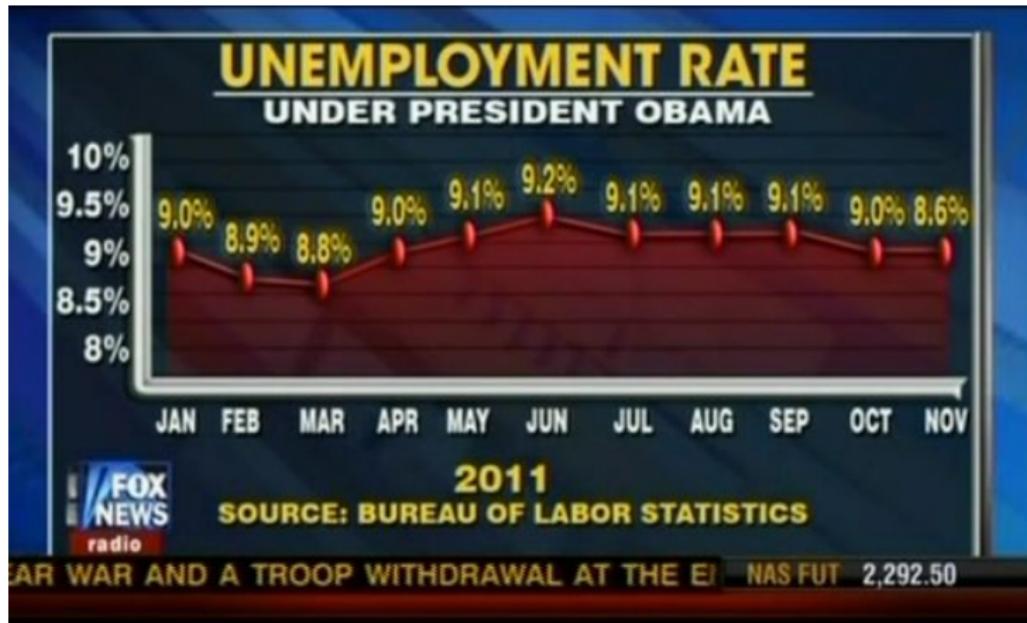
February 26, 2025

Kian Ming Lim (UNNC)

29 / 57

ACTIVITY . THINK-PAIR-SHARE

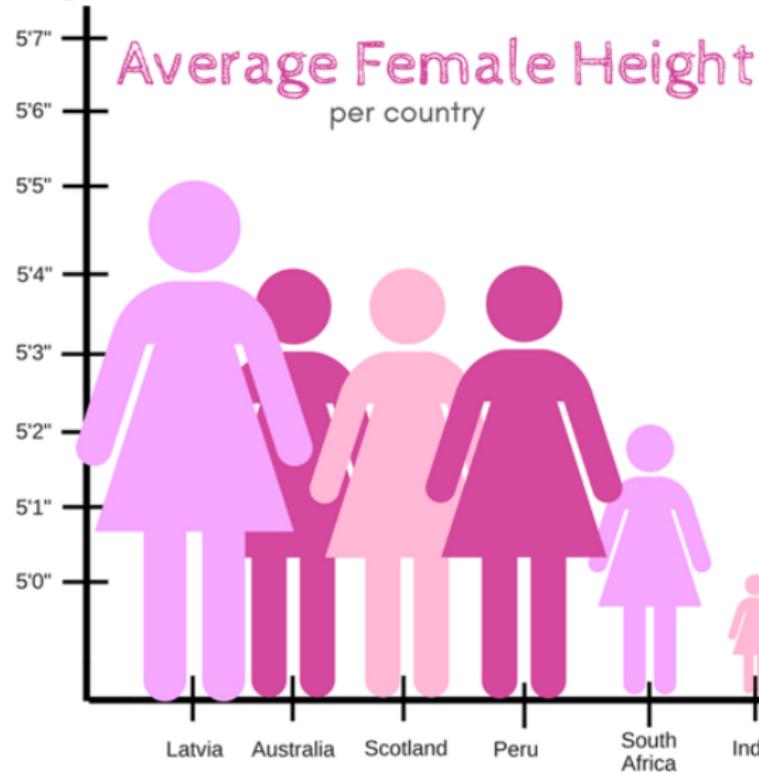
- What's wrong with this visualisation?



Source: CMSC320 - Introduction to Data Science

ACTIVITY . THINK-PAIR-SHARE

- What's wrong with this visualisation?



Source: CMSC320 - Introduction to Data Science

Takeaway

Be careful when designing visualisations, and **be extra careful** when interpreting graphs created by others.

Design and Aesthetics in Data Visualisation

Importance of Design in Visualisation

Why Design Matters:

- Enhances clarity and understanding.
- Makes visualisations more engaging and memorable.
- Builds trust and credibility with the audience.

Gestalt Principles in Visualisation

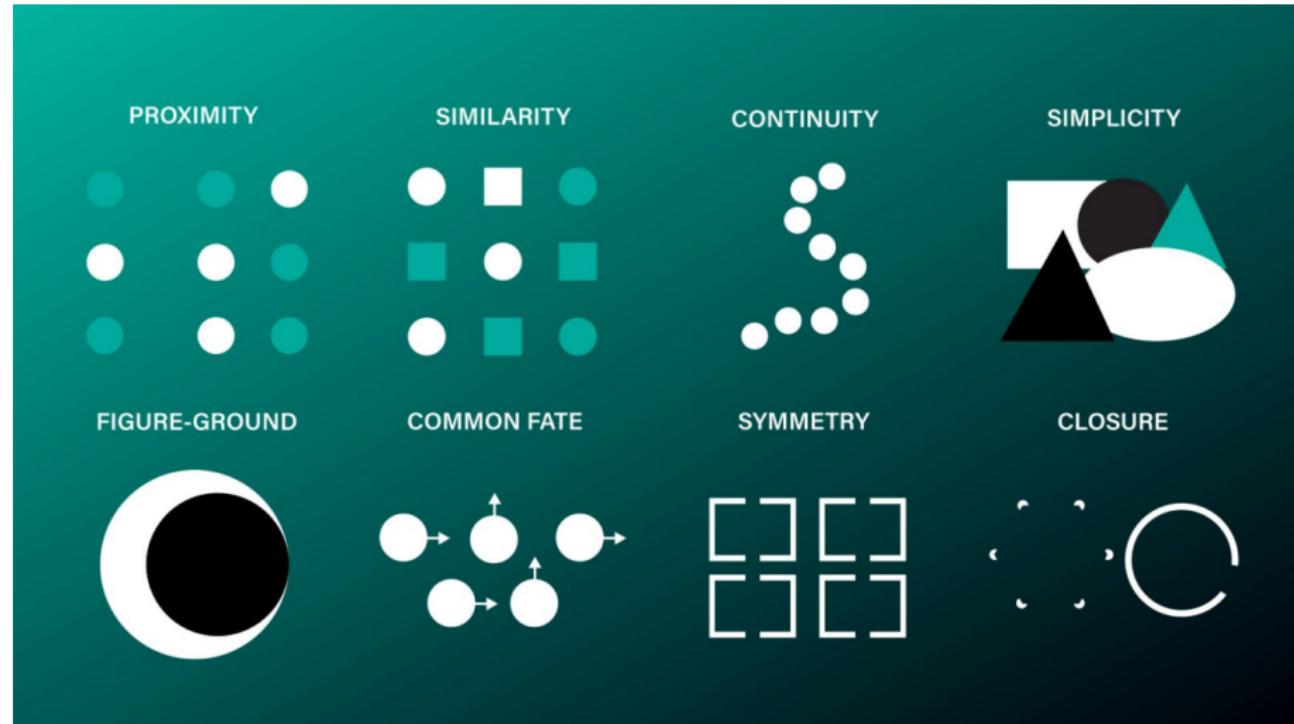
What are Gestalt Principles?

- A set of principles that describe how humans naturally organize visual elements into groups or unified wholes.
- These principles help designers create clear, intuitive, and effective visualisations.

Key Principles:

- **Proximity**
- **Similarity**
- **Continuity**
- **Closure**
- **Figure/Ground**
- **Common Fate**
- **Symmetry and Order**
- **Prägnanz (Simplicity)**

Gestalt Principles in Visualisation

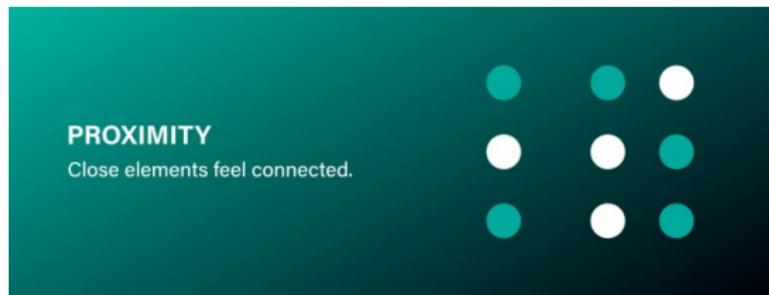


Source: Enhancing User Experience with Gestalt Principles

Gestalt Principle: Proximity

Proximity:

- Elements that are close to each other are perceived as related.
- Use spacing to group related items and separate unrelated ones.

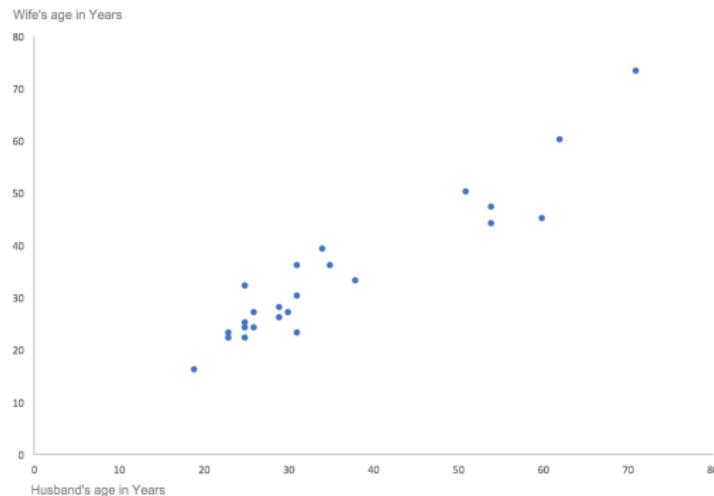


Source: Enhancing User Experience with Gestalt Principles

Gestalt Principle: Proximity

Proximity:

- In this chart, we see 2 groups or clusters of dots, though there are no visible markings of a group.

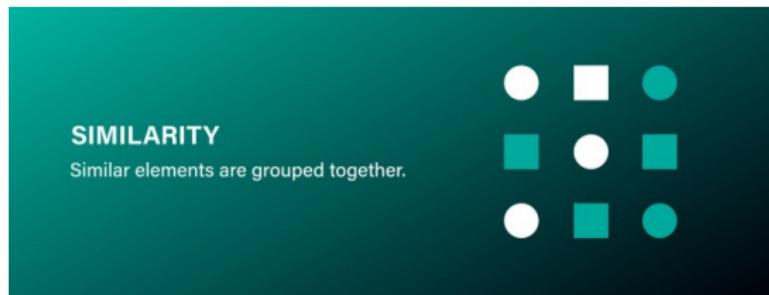


Source: Gestalt Laws Applied to Data Visualization

Gestalt Principle: Similarity

Similarity:

- Elements that look similar (e.g., color, shape, size) are perceived as related.
- Use consistent styles to group similar items.

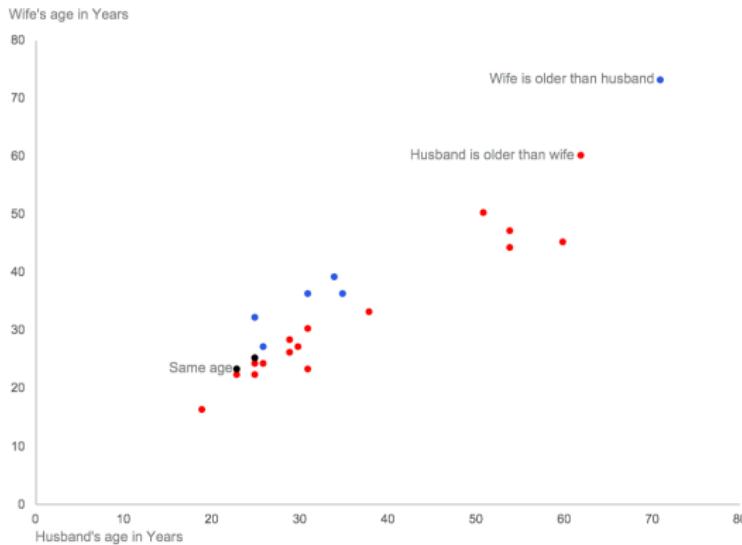


Source: Enhancing User Experience with Gestalt Principles

Gestalt Principle: Similarity

Similarity:

- We see 3 groups in this chart – dots of colour Red, Blue and Black.
Though the Red dots do not appear close together, we see them as a group since they are all Red.

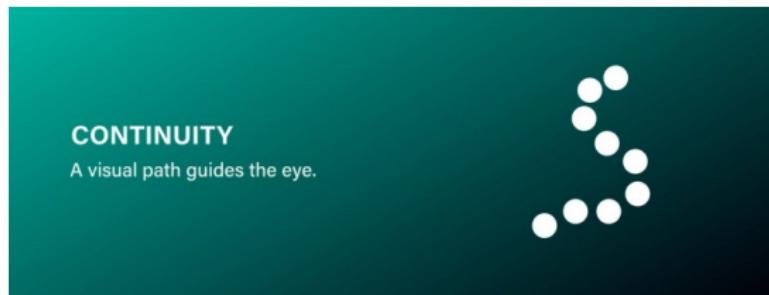


Source: Gestalt Laws Applied to Data Visualization

Gestalt Principle: Continuity

Continuity:

- Elements arranged in a line or curve are perceived as related.
- Use alignment to guide the viewer's eye through the visualisation.

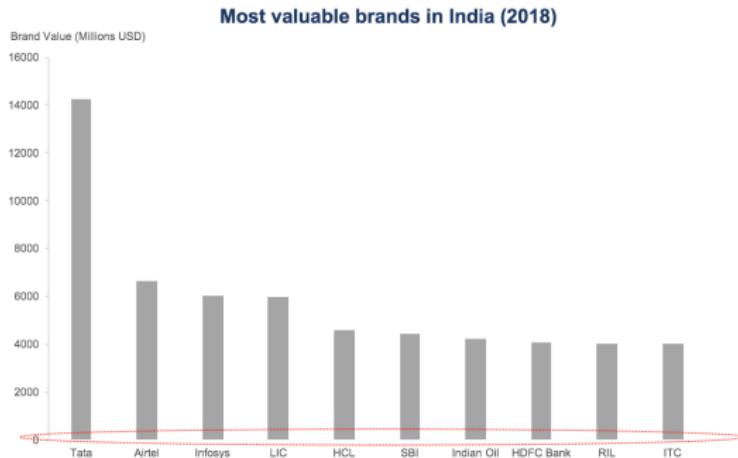


Source: Enhancing User Experience with Gestalt Principles

Gestalt Principle: Continuity

Continuity:

- If you look closely, we haven't used the X-Axis line in this chart. But we see these bars as sharing a common baseline due to the law of continuity.

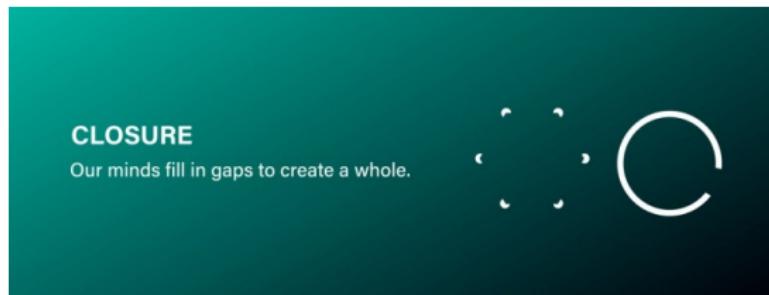


Source: Gestalt Laws Applied to Data Visualization

Gestalt Principle: Closure

Closure:

- The mind fills in missing parts of a shape or design.
- Use partial shapes to imply completeness.

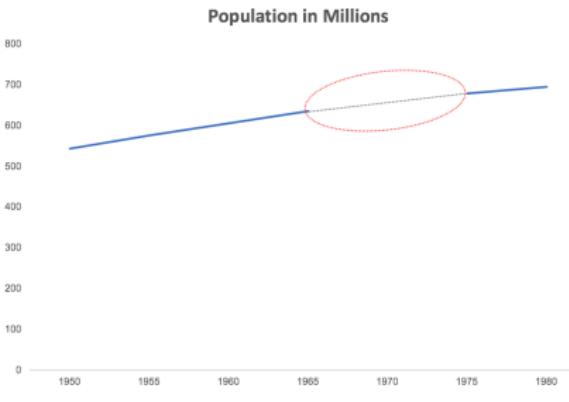
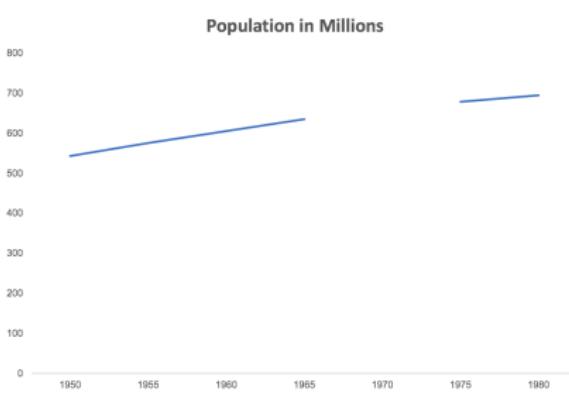


Source: Enhancing User Experience with Gestalt Principles

Gestalt Principle: Closure

Closure:

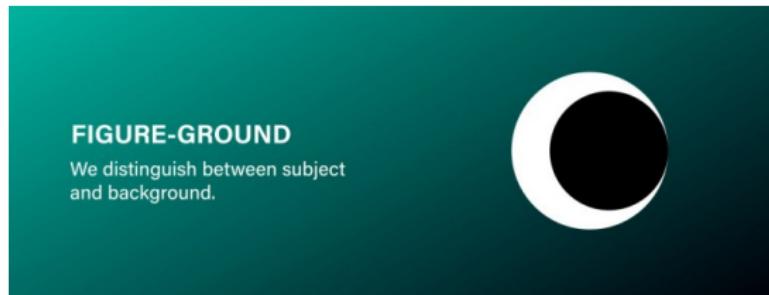
- Here is a chart with some missing data for the year 1970. When we look at this, our minds automatically imagine a line connecting the 2 broken lines.
- This is one of the pitfalls of the law of closure. We should be careful when showing graphs with breaks because our minds tend to form complete shapes even if the shape is incomplete.



Gestalt Principle: Figure/Ground

Figure/Ground:

- The mind separates elements into foreground (figure) and background (ground).
- Use contrast to distinguish important elements from the background.

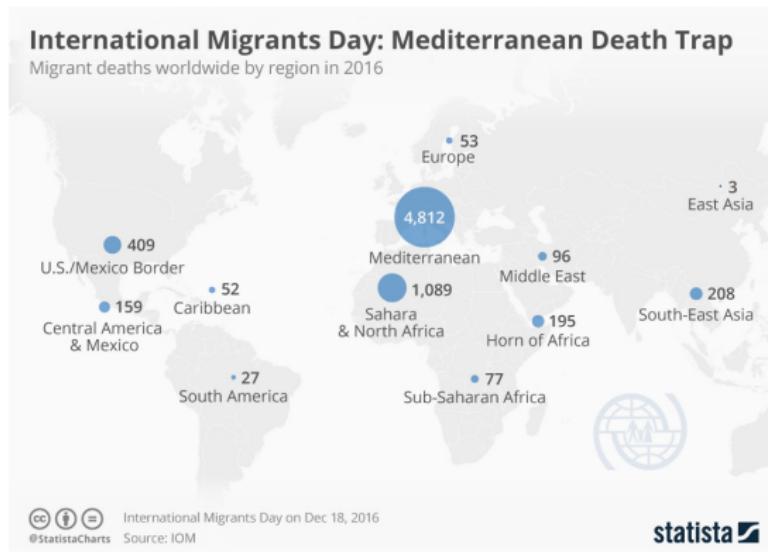


Source: Enhancing User Experience with Gestalt Principles

Gestalt Principle: Figure/Ground

Figure/Ground:

- The blue bubbles form the figure in the chart, they are in the forefront and capture our attention. The map in the background comes into focus next.

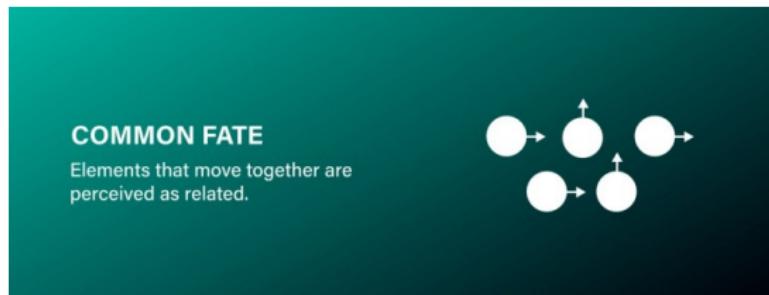


Source: Gestalt Laws Applied to Data Visualization

Gestalt Principle: Common Fate

Common Fate:

- Elements that move in the same direction are perceived as related.
- Use motion or directional cues to group elements.

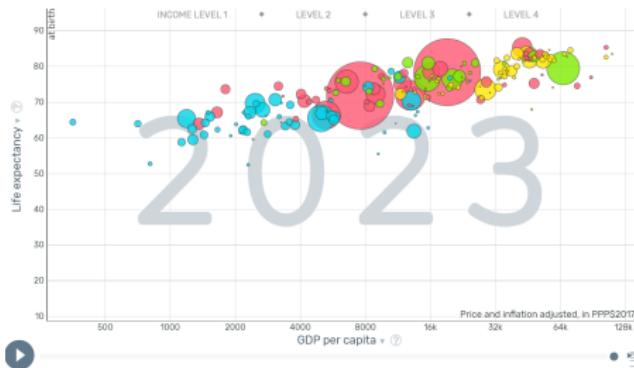


Source: Enhancing User Experience with Gestalt Principles

Gestalt Principle: Common Fate

Common Fate:

- We perceive elements moving together in the same speed and/or direction as belonging to a group.
- In this chart from Gapminder, we can see the bubbles move right and upward. We perceive this as an improvement in Life Expectancy with increase in GDP. The elements that move in the same direction are perceived to have the same fate.



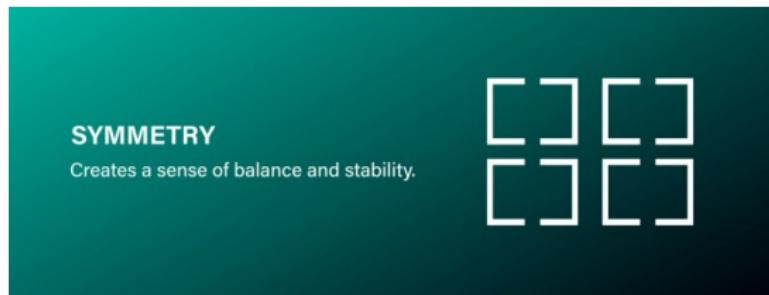
Link: [Gapminder](#)

Source: Gestalt Laws Applied to Data Visualization

Gestalt Principle: Symmetry and Order

Symmetry and Order:

- The mind perceives symmetrical elements as part of the same group.
- Use symmetry to create balance and organization.



Source: Enhancing User Experience with Gestalt Principles

Gestalt Principle: Symmetry and Order

Symmetry and Order:

- The Law of Symmetry states that visual elements that are symmetrical to each other tend to be perceived as a unified group.
- This principle is not used frequently in data visualization; however, it might be compelling when creating “Before and After” visuals or comparing two similar groups of elements.

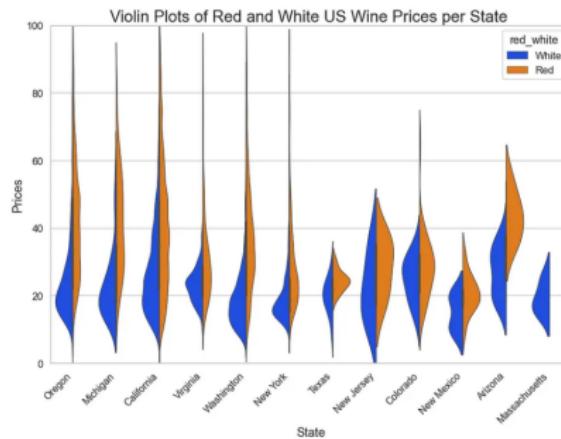


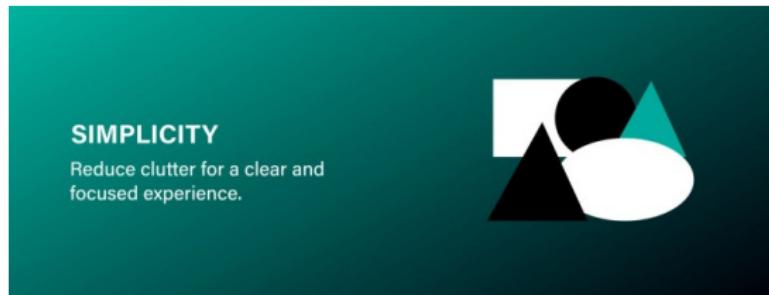
Illustration of Gestalt Symmetry principle, image by Author

Source: The Psychology behind Data Visualization Techniques

Gestalt Principle: Prägnanz (Simplicity)

Prägnanz (Simplicity):

- The mind prefers simple, clear, and stable interpretations of complex shapes.
- Avoid unnecessary complexity in visualizations.

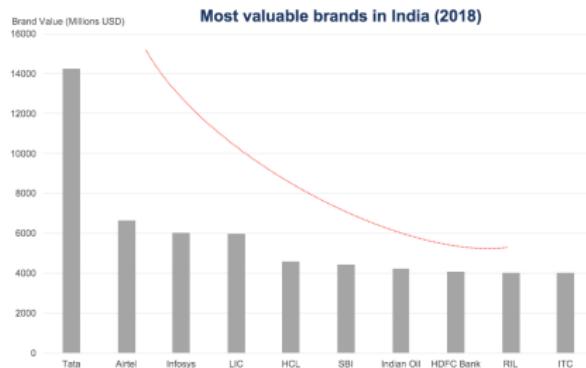
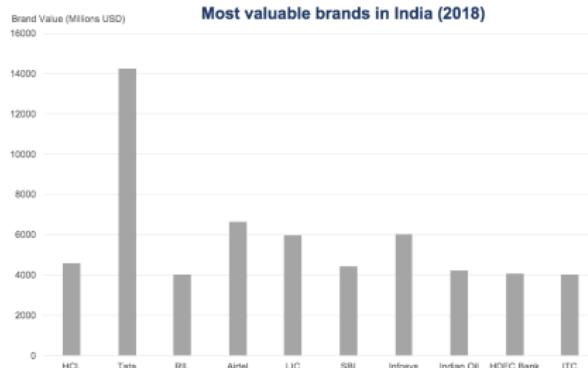


Source: Enhancing User Experience with Gestalt Principles

Gestalt Principle: Prägnanz (Simplicity)

Prägnanz (Simplicity):

- Prägnanz is a German word that means “pithiness” or “concise and meaningful”.
- Our eyes tend to find simplicity in complex shapes, preventing us from being overwhelmed by information overload.
- When the same chart is sorted in order, we find much more clarity in it.



Source: Gestalt Laws Applied to Data Visualization



Data Storytelling

Introduction to Data Storytelling

What is Data Storytelling?

- The art of using data, visuals, and narrative to communicate insights effectively.
- Combines data analysis, visualization, and storytelling techniques.

Why is it important?

- Helps audiences understand complex data.
- Drives decision-making and action.

Elements of Data Storytelling

Three Key Elements:

- **Data:** Accurate and relevant data to support the story.
- **Visuals:** Clear and engaging visualisations.
- **Narrative:** A compelling storyline to connect the data and visuals.

Storytelling with Data

Key elements:

- Start with a clear question or problem.
- Use data to build a narrative.
- Highlight key insights and conclusions.
- End with actionable recommendations.



Source: Hans Rosling's TED Talk

Useful link: [Gapminder](#)

Resources

- **Useful link 1:** From Data to Viz
- **Useful link 2:** The Data Visualisation Catalogue
- David McCandless' TED Talk: [The Beauty of Data Visualization](#)
- A video from Alberto Cairo: [A Brief History of Data Visualization](#)

COMP4131: Data Modelling and Analysis

Lecture 4: Analysis and Modelling

Kian Ming Lim

University of Nottingham Ningbo China

kian-ming.lim@nottingham.edu.cn

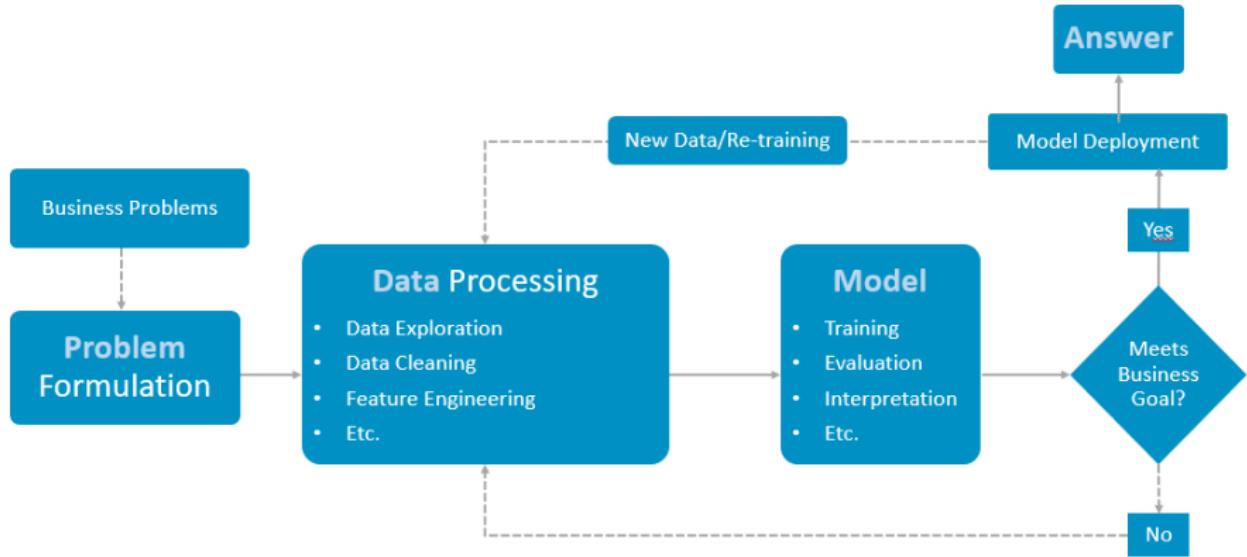
March 6, 2025

Outline

- 1 Introduction to Analysis and Modelling
- 2 Data
- 3 Data Analysis
- 4 Data Modelling
- 5 Model Evaluation and Validation

Introduction to Analysis and Modelling

The Data Pipeline



Problem ➤ Data ➤ Model ➤ Answer

© AWS Machine Learning



Definition of Data Analysis and Modelling

Data Analysis: The process of examining, cleaning, transforming, and interpreting data to extract useful insights and support decision-making.

Data Modelling: The creation of abstract representations (models) to describe, analyze, and predict real-world phenomena.

Relationship Between Analysis and Modelling:

- Data analysis provides the foundation for modelling by identifying patterns, trends, and relationships in the data.
- Modelling uses these insights to create predictive or descriptive models.

Research Questions and Negotiation

- The analysis start with the needs and objectives from stakeholders.
- You need to understand the requirements of the analysis:
 - Who is it for (at what level are you pitching it)? Professors, manager?
 - What resources do you have? Existing dataset/new data?
 - How much time is available?
 - How will the analysis be validated?
 - What level of uncertainty is acceptable?
 - Does it need to be reproducible? (Less obvious than you think!)
- Those things need to be negotiated with the stakeholders
 - Do not assume stakeholders know what they need.

Characteristics of a Good Analysis Question

The SPAIN Characteristics:

- **S**pecific:
 - "Does eating vegetables improve health?" (Too general)
 - "Does increasing vegetable consumption improve blood sugar in diabetics?" (Specific)
- **P**lausible:
 - "Does eating vegetables increase your bench press?" (No plausible link)
 - "Does your credit limit influence your salary?" (Causality link is reversed)
- **A**nswerable:
 - Is it possible to collect the data to answer the question?
 - Does the data even exist?
- **I**nteresting to a Specific Audience:
 - Who is your audience? (Researcher, boss, friends)
- **N**ovel:
 - Has someone else recently answered it?
 - Nobody likes to do work for no reason.

The Win-Win Strategy of Formulating Questions

- Do not get trapped into questions which are conditionally interesting
- The goal of a question is to learn something.
- Start with undirected questions, then follow with directional hypotheses.
- Example:
 - **Boring:** "Does algorithm X outperform algorithm Y?"
 - **Interesting:** "What is the interaction of algorithms X and Y and parameters A and B?"
- Sometimes, you may only have conditionally interesting questions, but it's worth checking if that's the case.

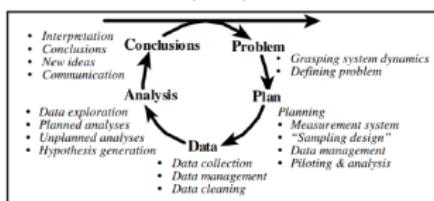
ACTIVITY . THINK-PAIR-SHARE

- Pair yourself with the person next to you and make up a good question you could answer from the dataset we collected in class.

ID	CHARACTERISTICS OF A GOOD QUESTION
Start time	Specific
Completion time	Plausible
Email	Answerable
Name	Interesting
I love	Novel
Age Band	
Do you love Marmite	
Main reason for taking DMA	
I am optimistic about the future of the world	
My favourite movie of all time is	
Today I am feeling	
My favourite way to relax after a stressful day	
On average, how many hours of sleep do you get on a weekday night?	
How many sweets do you think there are in this tin?	

Many attempts to formalise data analysis

(a) DIMENSION 1 : THE INVESTIGATIVE CYCLE



(b) DIMENSION 2 : TYPES OF THINKING

GENERAL TYPES

- Strategic
 - planning, anticipating problems
 - awareness of practical constraints
- Seeking Explanations
- Modelling
 - construction followed by use
- Applying Techniques
 - following precedents
 - recognition and use of archetypes
 - use of problem solving tools

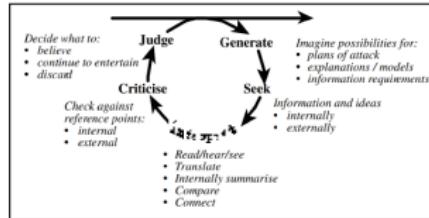
TYPES FUNDAMENTAL TO STATISTICAL THINKING (Foundations)

- Recognition of need for data
- Transmutation
 - (Changing representations to engender understanding)
 - capturing "measures" from real system
 - changing data representations
 - communicating messages in data
- Consideration of variation
 - noticing and acknowledging
 - measuring and modelling for the purposes of prediction, explanation, or control
 - explaining and dealing with investigative strategies
- Reasoning with statistical models
 - aggregate-based reasoning
- Integrating the statistical and contextual
 - information, knowledge, conceptions

(d) DIMENSION 4 : DISPOSITIONS

- Scepticism
- Imagination
- Curiosity and awareness
 - observant, noticing
- Openness
 - to ideas that challenge preconceptions
- A propensity to seek deeper meaning
- Being Logical
- Engagement
- Perseverance

(c) DIMENSION 3 : THE INTERROGATIVE CYCLE



Wild, C. J., & Pfannkuch, M. (1999). Statistical thinking in empirical enquiry. *International statistical review*, 67(3), 223-248.

The Epicycle of Analysis

Main Activities of Data Analysis:

- Stating the question
- Exploring and analysing the data
- Building a model
- Interpreting the results
- Communicating the results

Sub-Activities for Each:

- Setting expectations
- Collecting data/information and comparing it to expectations
- Revising expectations

Source: Peng, R.D., & Matsui, E. (2015). *The Art of Data Science: A Guide for Anyone Who Works with Data.*

Data

Describing the Data

- **Stevens' four level scale:**

- Nominal
- Ordinal
- Interval
- Ratio

- **Mosteller and Tukey's scale:**

- Names
- Grades
- Ranks
- Fractions
- Counts
- Amounts
- Balances

- **Chrisman's expanded scale:**

- Nominal
- Gradation of membership
- Ordinal
- Interval
- Log-interval
- Extensive ratio
- Cyclical ratio
- Derived ratio
- Counts
- Absolute

Describing the Data

- Nominal
 - Names, labels, categories, etc.
 - Unordered and distinct
 - Relevant statistics: **mode**
 - Bar charts as far as the eye can see
- Ordinal
 - Scales, ratings, etc.
 - Ordered
 - Distinct (no overlap)
 - Relevant statistics: **median, percentile**
- Interval
 - Numeric
 - Ordered, but differences are meaningful
 - Distinct, ordered and **additive**
 - Interval data has **no meaningful 0**
 - Relevant statistics: **mean, standard deviation, correlation, regression**
- Ratio
 - Numeric
 - Ordered and can be used with any arithmetic operator
 - Distinct, ordered, additive, multiplicative
 - Ratio data **has a meaningful, absolute 0**
 - **All statistics apply**

ACTIVITY . THINK-PAIR-SHARE

- Pair yourself with the person next to you and look at the data we collected in our survey. What type of attributes are those?

ID	154	
Start time	2/15/22 15:11:58	Nominal?
Completion time	2/15/22 15:13:18	Ordinal?
Email	anonymous	Interval?
Name		Ratio?
I love	Both	
Age Band	31 to 40	
Do you love Marmite	Yes	
Main reason for taking DMA	Increase knowledge of subject area	
I am optimistic about the future of the world	Agree	
My favourite movie of all time is	None of the above	
Today I am feeling:	Happy	
My favourite way to relax after a stressful day is:	Cooking	
On average, how many hours of sleep do you get on a weekday night?	7	
How many sweets do you think there are in this tin?	75	

Tricky ones:

- Date / time?
- Age range?
- Amount of sleep?
- Number of sweets in the tin?

Describing Data

- We use numbers and drawings to get a sense of a large dataset.
- The process is called **Transnumeration**.
- **Location:**
 - What's the most common value?
- **Variability/Dispersion:**
 - How spread out is the data? What's a deviant value?
- **Shape:**
 - How is the data distributed?
- **Relationship:**
 - How are two variables linked?

Transnumerative Techniques

Technique	Description	Example
Sorting	The data are sorted on some criterion. No new variables arise.	The data are sorted by hours of exercise, from lowest to highest.
Grouping	The data are grouped according to some criterion. This creates a new variable. This may involve the change variable type transnumeration beforehand.	A new variable "level of consumption" is created using the fast food data, with values "low" (0-1 fast food meals/week), "medium" (2-3 meals/week), and "high" (t4 fast food meals/week).
Subset selection	A subset of the data is selected for further transnumeration.	Data associated with "low" and "high" levels of consumption are considered ("medium" is not).
Change variable type	A numerical variable is thought of in categorical terms or a categorical variable is thought of in numerical or ordinal terms.	Favourite activity (a categorical variable) can be given ordinal status, by ordering activities from most to least active.
Frequency calculation	The frequencies of occurrence of values of a categorical variable are determined. Creates new variable.	The numbers of people in each of the "level of consumption" categories are determined.
Proportion calculation	Proportions (e.g., fractions) are determined in relation to a whole. This creates a new variable	The percentage of people in each of the activity categories is determined.
Graphing	Some or all of the variables in the data (in their current form) are graphed or tabulated.	A scatter graph of hours of exercise v number of fast food meals consumed is constructed.
Summary statistics	A measure of central tendency (e.g., mean), variability, shape, correlation, etc. is determined for a variable. May create new variable.	The average number of fast food meals consumed per week is determined, or parameters of line of best fit between number of hours of exercises and fast food meals consumed.

Partially reproduced and adapted from: Chick, Helen. "Tools for transnumeration: Early stages in the art of data representation."

Summary Statistics - Central Tendency

- **Mode:**

- Most frequent attribute in the sample.
- Not necessarily unique.

- **Median:**

- Middle value in a sorted dataset.
- If n is odd: Median = $x_{\frac{n+1}{2}}$.
- If n is even: Median = $\frac{x_{\frac{n}{2}} + x_{\frac{n}{2}+1}}{2}$.

- **Arithmetic Mean:**

- $\bar{x} = \frac{\sum_{i=1}^n x_i}{n}$.

Summary Statistics - Variability

Range:

- Difference between minimum and maximum value.
- Allows you to quickly see the bounds of the data.

Inter-Quartile Range (IQR):

- The p -quantile of data is the value relative to which the fraction p of the data is smaller.
- The $\frac{1}{2}$ -quantile value is the median.
- The p -inter-quartile range is the difference between the $(1 - p)$ -quantile value and the p -quantile value.
- It is common to use $p = \frac{1}{4}$ (inter-quartile range).

- Average absolute deviation

$$d_{\bar{x}} = \frac{1}{n} \sum_{i=1}^n |x_i - \bar{x}|$$

- Variance

$$s^2 = \frac{1}{n-1} \sum_{i=1}^n (x_i - \bar{x})^2$$

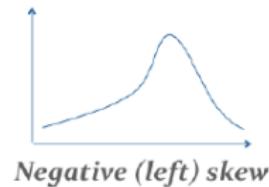
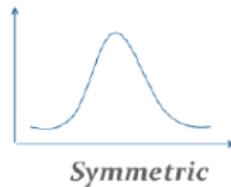
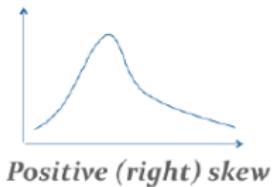
- Standard deviation

$$s = \sqrt{s^2} = \sqrt{\frac{1}{n-1} \sum_{i=1}^n (x_i - \bar{x})^2}$$

Summary Statistics - Shape

Skewness:

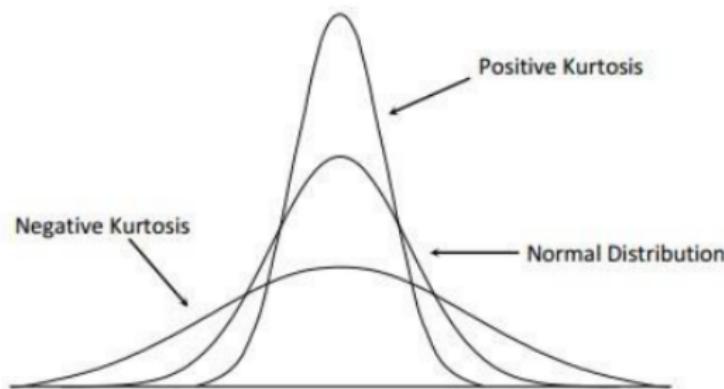
- Measures the asymmetry of the probability distribution of a real-valued random variable about its mean.
- **Positive (Right) Skew:** Tail is longer on the right.
- **Symmetric:** No skew.
- **Negative (Left) Skew:** Tail is longer on the left.



Summary Statistics - Shape

Kurtosis:

- Measures how peaked a distribution is.
- **Mesokurtic:** Looks like a Gaussian (normal) distribution.
- **Platykurtic:** Wider and flatter than a Gaussian (negative kurtosis).
- **Leptokurtic:** Thinner and peakier than a Gaussian (positive kurtosis).
- Measure is usually computed relative to the Gaussian distribution.



Summary Statistics - Relationships

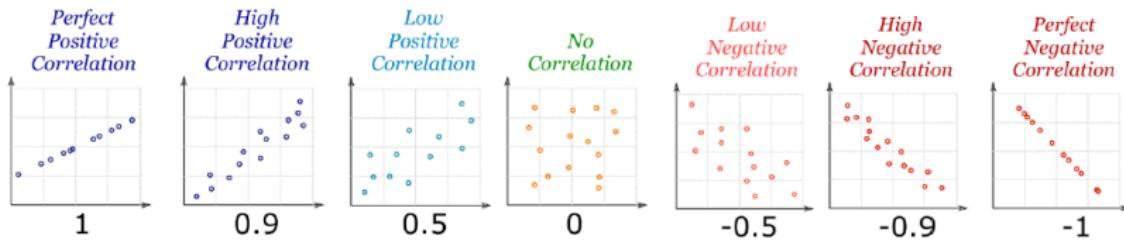
Pearson Correlation Coefficient:

- Measures the strength and direction of the linear relationship between two variables.
- Ranges between $[-1, +1]$:
 - -1 : Perfect negative linear relationship.
 - 0 : No linear relationship.
 - $+1$: Perfect positive linear relationship.

- Formula:

$$r = \frac{\sum_{i=1}^n (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^n (x_i - \bar{x})^2 \sum_{i=1}^n (y_i - \bar{y})^2}}$$

- Many other correlation measures exist for different use cases.



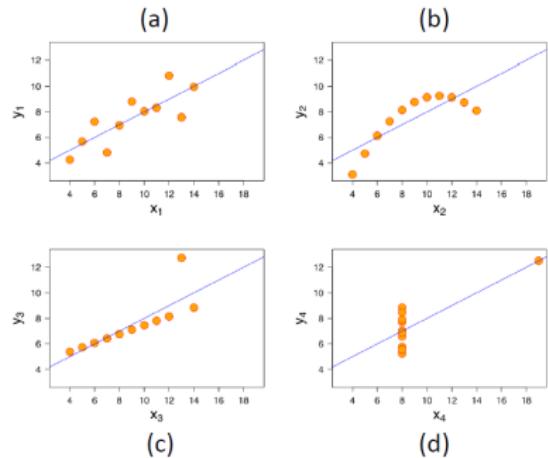
The Power of a Good Plot

- Don't overestimate numbers.
- Which plot (a, b, c, or d) is represented by this table?

Property	Value
Mean of x	9
Sample variance of x	11
Mean of y	7.50
Sample variance of y	4.125
Correlation(x,y)	0.816
Linear regression line	$y = 3 + 0.5x$

→ Anscombe's quartet

Anscombe, F. J. (1973). Graphs in statistical analysis. *The american statistician*, 27(1), 17-21.



Diagnosing Data Problems

Check for Boundary Violations:

- **Impossible Values:**
 - If a field contains "Age," negative values are impossible.
- **Outliers:**
 - Different from impossible values!
 - Example: An "Age" value of 185.
- Knowing how the data is generated helps in identifying faulty data.

Check the Packaging:

- Look at the first few rows and the last few rows. Does it all look as expected?
- Check the dimensions of your dataset (number of observations, number of columns). Does it all match up?

Diagnosing Data Problems - Imputation

Imputation:

- Replacement of missing data with substituted values.
 - **Unit Imputation:**
 - Impute a data point (e.g., remove a person with missing Age).
 - **Item Imputation:**
 - Impute a value of a data point (e.g., replace missing Age value).
- **Hot Deck Imputation:**
 - Replace value by value of previous observation.
- **Cold Deck Imputation:**
 - Replace value by value randomly sampled from an external dataset.
- **Mean Substitution:**
 - Replace value by central value of all other cases (mean/mode/median).
- **Regression Imputation:**
 - Use a basic machine learning model to predict the missing value.
- **Multiple Imputations:**
 - Sometimes you need more than one method! A lot of design choices go into imputation.

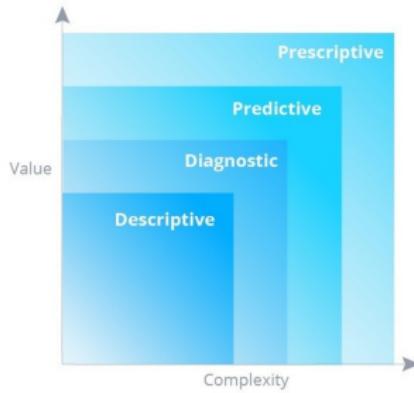
Data Analysis

Overview of Data Analysis Techniques

Purpose: To transform raw data into actionable insights.

Types of Data Analysis:

- Descriptive Analysis
- Diagnostic Analysis
- Predictive Analysis
- Prescriptive Analysis
- Exploratory Data Analysis (EDA)



Descriptive analytics addresses the issue of what happened.

Diagnostic analytics answers the question of why something happened.

Predictive analytics describes what is likely to happen.

Prescriptive analytics prescribes what step to take to avoid a future problem.

Source: Things You Should Know About Types Of Data Analysis

Descriptive Analysis

Definition:

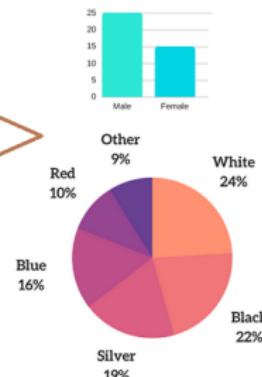
- Summarizes historical data to understand what has happened.

Examples:

- Mean, median, mode.
- Sales reports, customer demographics.

	A	B	C	D
1	Respondent Number	Age	Gender	Favorite Car Color
2	1	22	M	White
3	2	37	F	Silver
4	3	45	F	Black
5	4	62	F	Gray
6	5	28	M	Red
7	6	45	M	Green
8	7	88	F	Brown
9	8	61	M	White
10	9	95	M	Black
11	10	27	M	White
12	11	39	F	Green
13	12	43	M	Brown
14	13	55	F	Black
15	14	59	F	White

RAW DATA



Descriptive Statistics

Source: Descriptive Statistics Examples, Types and Definition

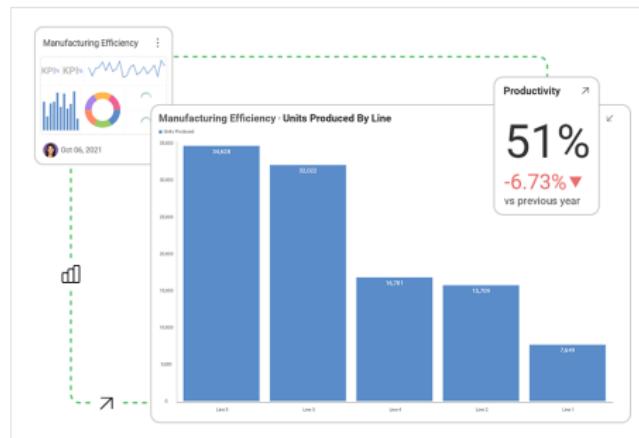
Diagnostic Analysis

Definition:

- Identifies the causes of past events or behaviors.

Examples:

- Root cause analysis.
- Correlation analysis.



Source: Types of Data Analysis, Benefits & Examples

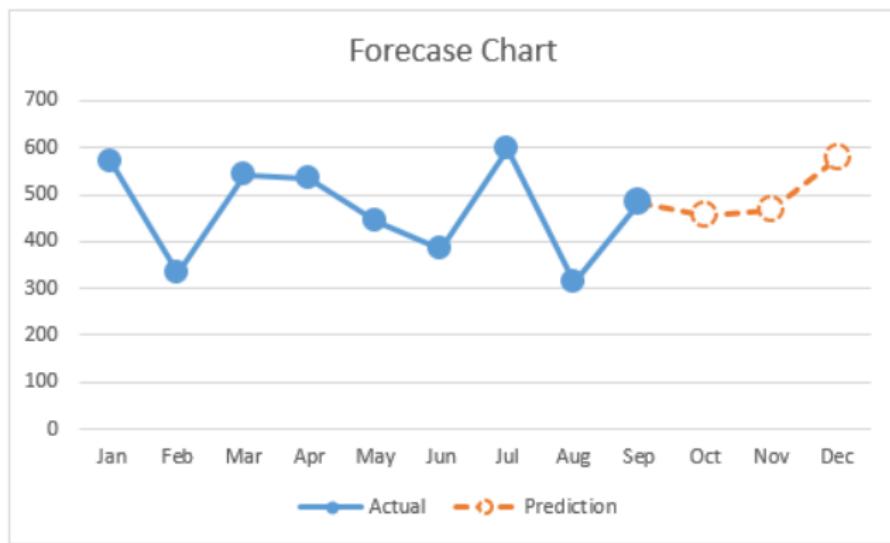
Predictive Analysis

Definition:

- Uses historical data to predict future outcomes.

Examples:

- Sales forecasting, customer churn prediction.



Source: Forecast Chart.



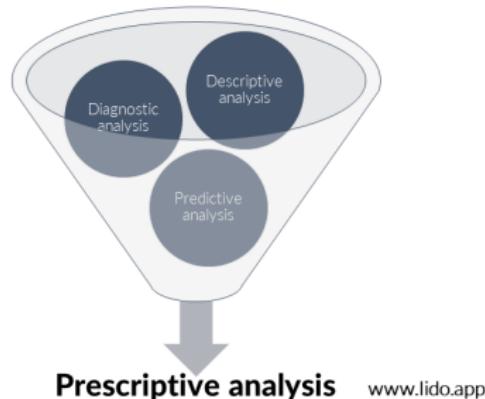
Prescriptive Analysis

Definition:

- Recommends actions based on data insights.

Examples:

- Optimization models, decision trees.



Source: Data Analysis 101

Source: Types of Data Analysis, Benefits & Examples

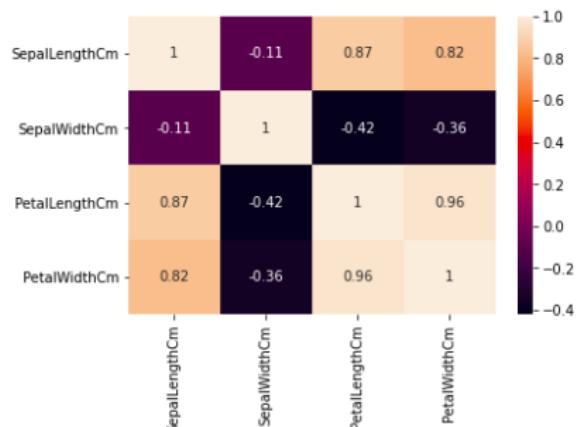
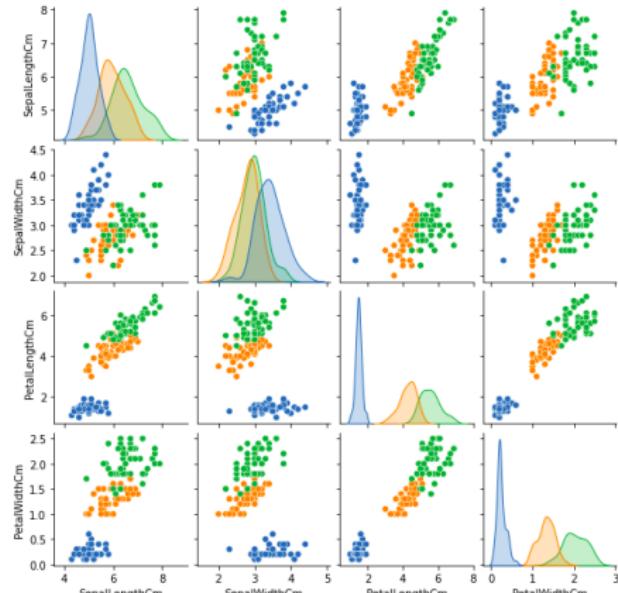
Exploratory Data Analysis (EDA)

Definition:

- Investigates data to discover patterns, trends, and anomalies.

Examples:

- Pairplots, heatmaps, summary statistics.



Source: Exploratory Data Analysis on Iris

Data Modelling

What is a Model?

Definition:

- A simplified representation of a real-world system.

Purpose:

- To understand, predict, or simulate real-world phenomena.

Types of Models

- 1. Statistical Models:** Uses probability and statistical techniques to make inferences from data.
 - Examples: Regression models, time-series models.
- 2. Machine Learning Models:** Uses data-driven algorithms to learn patterns and make predictions.
 - Examples: Decision Trees, Neural Networks.
- 3. Simulation Models:** Mimics real-world processes using computational techniques.
 - Examples: Monte Carlo simulations, agent-based models.

Introduction to Statistical Models

Definition: A statistical model is a mathematical representation of relationships between variables in data.

- Used for inference, prediction, and hypothesis testing.
- Helps in understanding patterns and making data-driven decisions.
- Examples: Regression models, time-series models, probabilistic models.

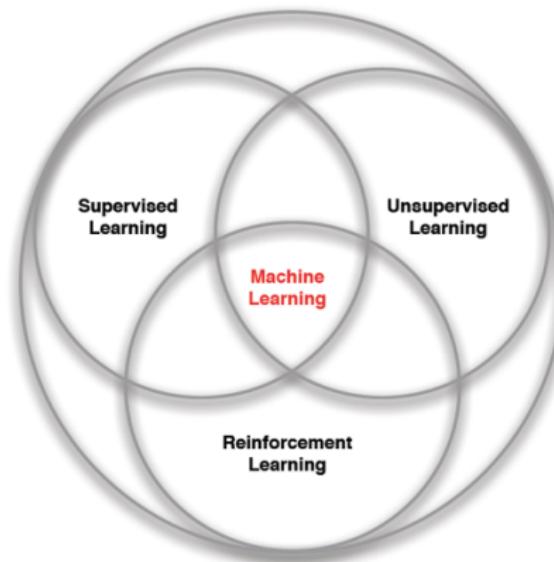
Introduction to Machine Learning Models

Definition:

- Models that learn patterns from data.

Types:

- Supervised learning, unsupervised learning, reinforcement learning.

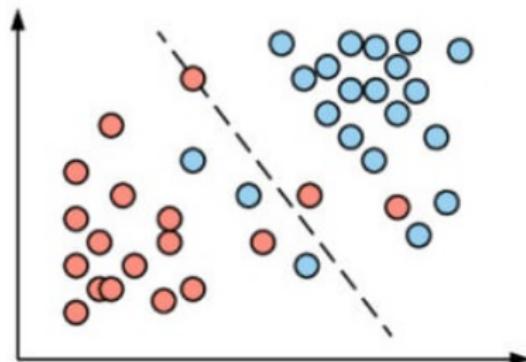


Source: [Link](#)

Supervised Learning

Definition: A machine learning approach where the model is trained using labeled data.

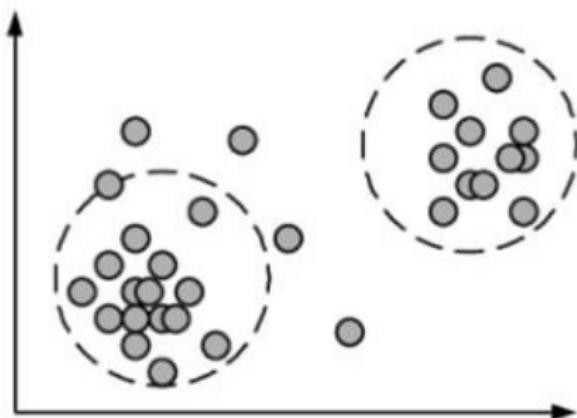
- Input (features) and output (labels) are known.
- Data: (x, y) x is data, y is label
- Learn a function to map $x \rightarrow y$
- Classification, regression, object detection, semantic segmentation, image captioning, etc.
- Examples: Linear regression, logistic regression, decision trees, support vector machines, and many more.



Unsupervised Learning

Definition: A machine learning approach where the model learns patterns without labeled data.

- Identifies hidden structures and relationships in data.
- ✗ Just data, no labels!
- Examples: K-means clustering, hierarchical clustering, principal component analysis (PCA), and many more.

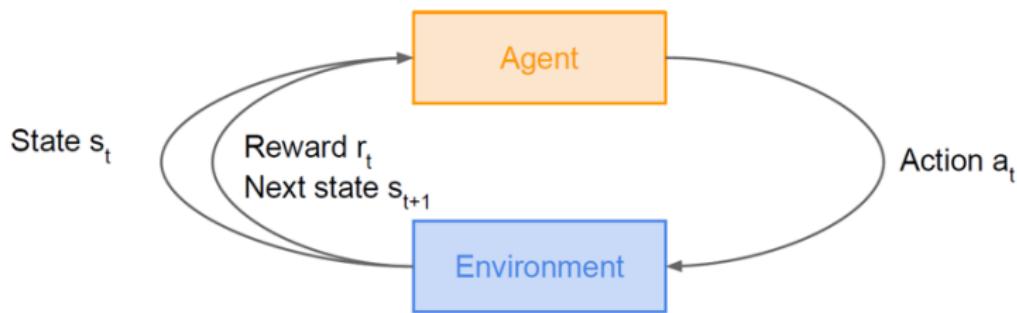


Source: SuperAnnotate

Reinforcement Learning

Definition: A learning paradigm where an agent learns by interacting with an environment and receiving rewards or penalties.

- Used in robotics, gaming, and autonomous systems.
- Learn how to take actions in order to maximize reward.
- Key components: Agent, Environment, Reward, Policy.



Source: Reinforcement Learning

Simulation Models

Definition: A model that uses computational techniques to simulate real-world processes and assess outcomes.

- Example: Monte Carlo simulation for risk assessment.
- Used in finance, healthcare, and engineering.



Source: An Introduction and Step-by-Step Guide to Monte Carlo Simulations

Model Evaluation and Validation

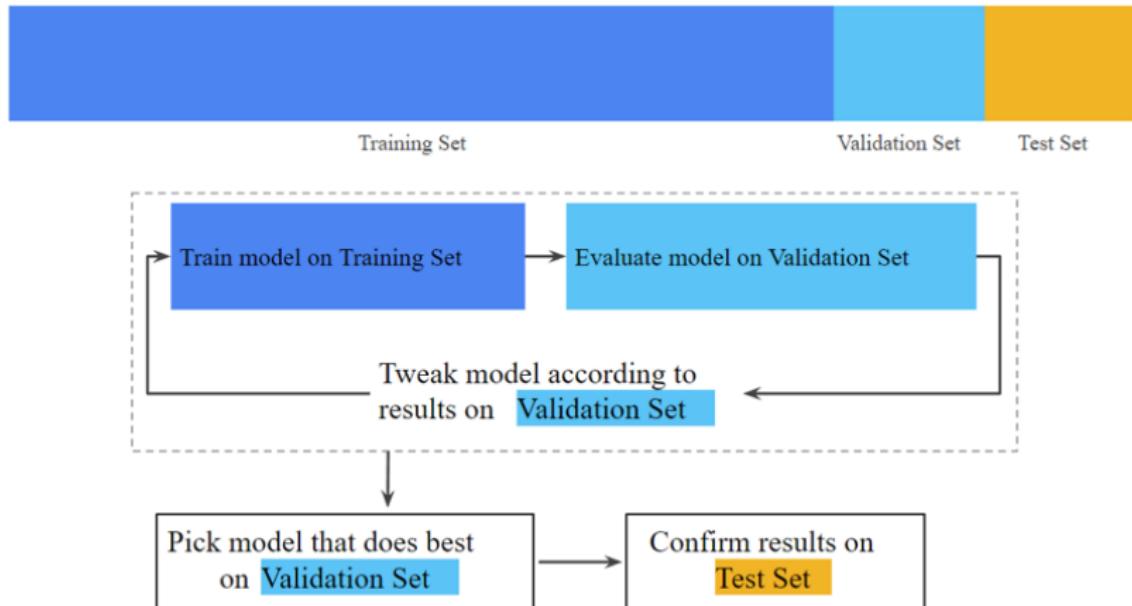
Importance of Model Evaluation

Why Model Evaluation Matters:

- Ensures model reliability and robustness.
- Helps detect overfitting and underfitting.
- Assists in selecting the best model for a given task.
- Provides insights into model performance on unseen data.

Train-Test Split/Holdout Method

Train-Test Split: A simple validation technique where data is split into training, validation, and testing sets.



Source: Pi.Exchange

Cross-Validation

Cross-Validation: A more robust method where the dataset is divided into multiple folds.

- **k -Fold Cross-Validation:** Data is split into k subsets, and each subset is used for testing once.
- Helps improve model generalization.

Iteration 1	Test	Train	Train	Train	Train
Iteration 2	Train	Test	Train	Train	Train
Iteration 3	Train	Train	Test	Train	Train
Iteration 4	Train	Train	Train	Test	Train
Iteration 5	Train	Train	Train	Train	Test

Source: Medium

Model Evaluation Metrics

Key Metrics for Regression:

- R^2 (Coefficient of Determination)
- RMSE (Root Mean Squared Error)
- MAE (Mean Absolute Error)

Key Metrics for Classification:

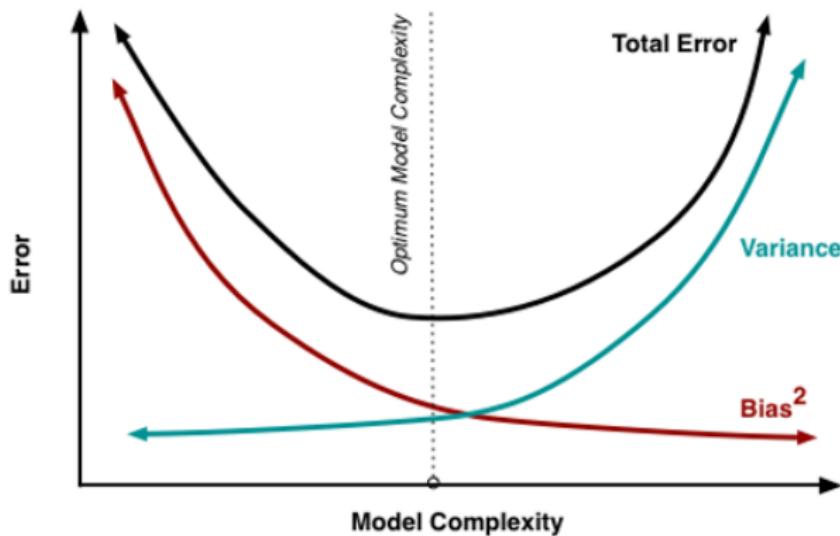
- Accuracy
- Precision
- Recall
- F1-Score
- Confusion Matrix
- Receiver Operating Characteristic (ROC) Curve
- Area Under the Curve (AUC)

Bias-Variance Tradeoff

Bias: Error due to overly simplistic assumptions.

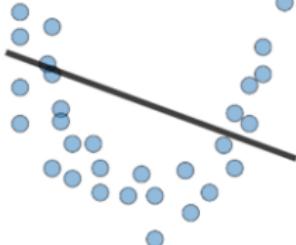
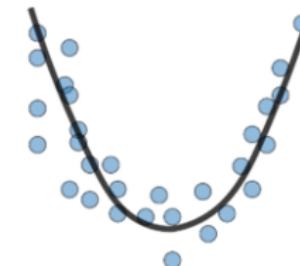
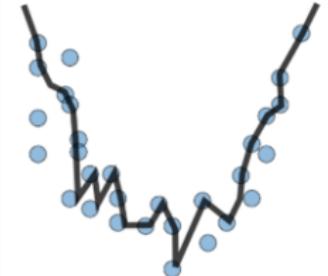
Variance: Error due to excessive sensitivity to training data.

- High bias leads to underfitting.
- High variance leads to overfitting.
- The goal is to find an optimal balance between bias and variance.



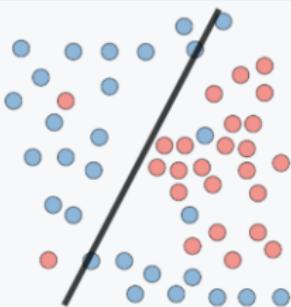
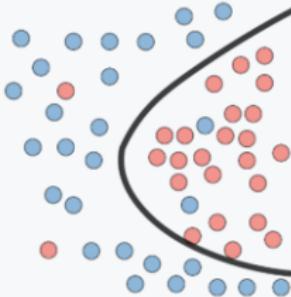
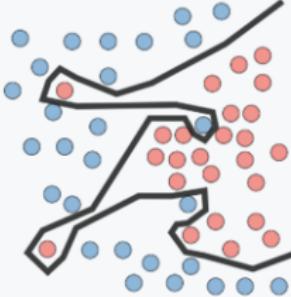
Source: Dataquest

Underfitting vs Overfitting

	Underfitting	Just right	Overfitting
Symptoms	<ul style="list-style-type: none">• High training error• Training error close to test error• High bias	<ul style="list-style-type: none">• Training error slightly lower than test error	<ul style="list-style-type: none">• Very low training error• Training error much lower than test error• High variance
Regression illustration			

Source: Machine Learning tips and tricks cheatsheet

Underfitting vs Overfitting

	Underfitting	Just right	Overfitting
Symptoms	<ul style="list-style-type: none">• High training error• Training error close to test error• High bias	<ul style="list-style-type: none">• Training error slightly lower than test error	<ul style="list-style-type: none">• Very low training error• Training error much lower than test error• High variance
Classification illustration			

Source: Machine Learning tips and tricks cheatsheet

Underfitting vs Overfitting

	Underfitting	Just right	Overfitting
Symptoms	<ul style="list-style-type: none">• High training error• Training error close to test error• High bias	<ul style="list-style-type: none">• Training error slightly lower than test error	<ul style="list-style-type: none">• Very low training error• Training error much lower than test error• High variance
Deep learning illustration			

Source: Machine Learning tips and tricks cheatsheet

Underfitting vs Overfitting

	Underfitting	Just right	Overfitting
Symptoms	<ul style="list-style-type: none">• High training error• Training error close to test error• High bias	<ul style="list-style-type: none">• Training error slightly lower than test error	<ul style="list-style-type: none">• Very low training error• Training error much lower than test error• High variance
Possible remedies	<ul style="list-style-type: none">• Complexify model• Add more features• Train longer		<ul style="list-style-type: none">• Perform regularization• Get more data

Source: Machine Learning tips and tricks cheatsheet

Characteristics of a Good Model

- **Accuracy:** Produces results that match real-world observations.
- **Generalizability:** Works well with unseen data.
- **Interpretability:** Can be easily understood and explained.
- **Efficiency:** Uses computational resources effectively.
- **Scalability:** Performs well with increasing data.

- **Performance Metrics:** Accuracy, RMSE, Precision, Recall.
- **Interpretability:** How easily the model's decisions can be explained.
- **Computational Efficiency:** Resource usage and execution time.
- **Generalization Ability:** How well the model performs on unseen data.
- **Data Availability:** The amount and quality of data required for training.

COMP4131: Data Modelling and Analysis

Lecture 5: Unsupervised Learning

Kian Ming Lim

University of Nottingham Ningbo China

kian-ming.lim@nottingham.edu.cn

March 12, 2025

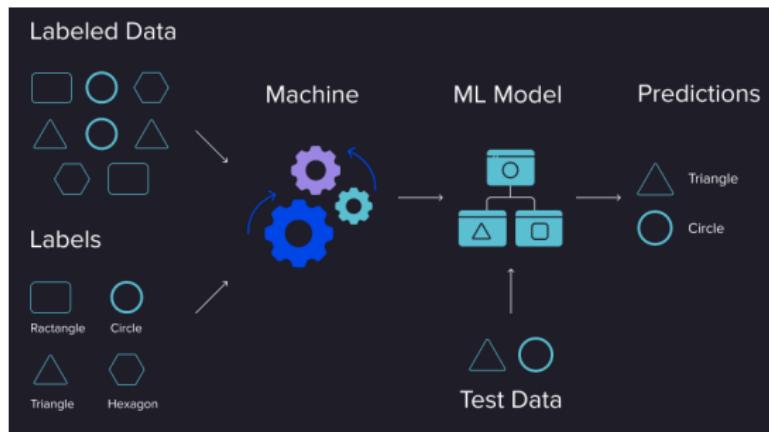
Outline

- 1 Introduction to Unsupervised Learning
- 2 Clustering
- 3 Dimensionality Reduction

Introduction to Unsupervised Learning

What is Supervised Learning?

- **Definition:** A type of machine learning where the model learns from labeled data to predict outcomes.
- **Key Concepts:**
 - Explicit supervision with labeled data (input-output pairs).
 - The algorithm learns a mapping from inputs to outputs.
 - Commonly used for classification and regression tasks.



Source: Eleks

What is Unsupervised Learning?

- **Definition:** A type of machine learning where the model learns patterns from unlabeled data.
- **Key Concepts:**
 - No explicit supervision.
 - The algorithm identifies hidden structures or patterns.
 - Commonly used for clustering and dimensionality reduction.



Source: Eleks

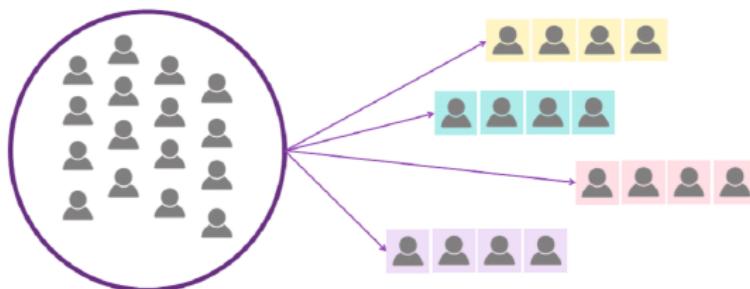
Supervised vs. Unsupervised Learning

Aspect	Supervised Learning	Unsupervised Learning
Data	Labeled	Unlabeled
Goal	Predict outcomes	Discover patterns
Examples	Classification, Regression	Clustering, Dimensionality Reduction

Table: Comparison of Supervised and Unsupervised Learning

Applications of Unsupervised Learning

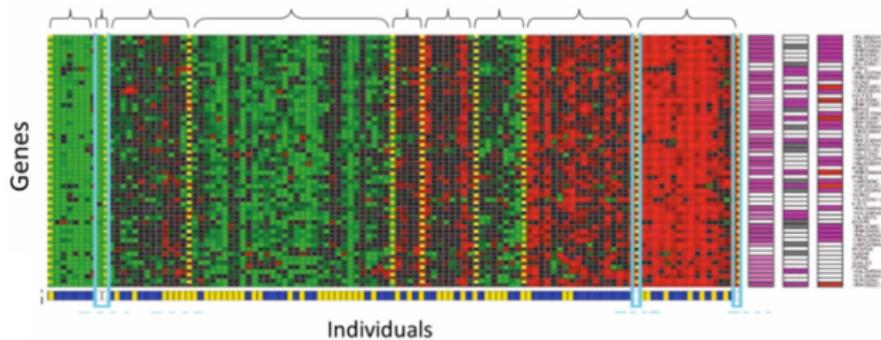
- **Customer Segmentation:** Grouping customers based on purchasing behavior.
- **Anomaly Detection:** Identifying unusual patterns in data (e.g., fraud detection).
- **Image Compression:** Reducing the size of images using clustering.
- **Recommendation Systems:** Suggesting products based on user behavior.



Source: Aspire System, Arun Lakshmanan

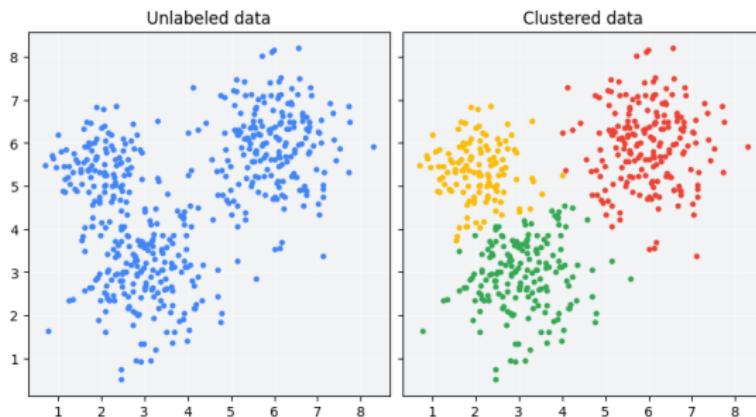
Applications of Unsupervised Learning

- **Google News:** Groups news stories into cohesive groups.
- **Genomics Microarray Data:** Cluster individual's genes into types of people.
- **Organize Computer Clusters:** Identify potential weak spots or distribute workload effectively.
- **Social network analysis:** Uncover valuable insights from the structure and dynamics of social networks.



Types of Unsupervised Learning

- **Clustering:** Grouping similar data points (e.g., K-Means, Hierarchical Clustering).
- **Dimensionality Reduction:** Reducing the number of features (e.g., PCA, t-SNE).
- **Association Rule Learning:** Discovering relationships between variables (e.g., Apriori Algorithm).

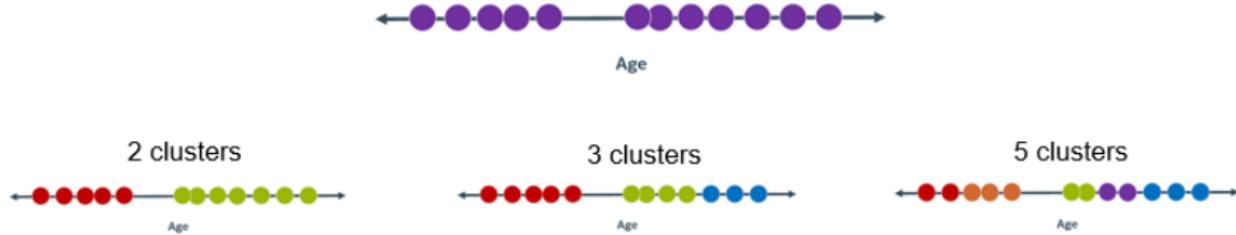


Source: Machine Learning Crash Course, Google

Clustering

What is Clustering?

- **Definition:** Grouping similar data points together based on their features.
- **Intuition:**
 - Unsupervised learning technique.
 - No predefined labels; the algorithm identifies natural groupings.
 - Used for exploratory data analysis and pattern discovery.



Types of Clustering

- **Partitional Clustering:** Divides data into non-overlapping subsets (e.g., K-Means).
- **Hierarchical Clustering:** Builds a tree of clusters (e.g., Agglomerative, Divisive).
- **Density-Based Clustering:** Groups data based on density (e.g., DBSCAN).
- **Model-Based Clustering:** Assumes data is generated from a mixture of distributions (e.g., Gaussian Mixture Models).

K-Means Clustering

- **Definition:** A type of unsupervised learning used for unlabeled data.
- **Goal:** Find groups in the data, with the number of groups represented by K .
- **Algorithm:**
 - Works iteratively to assign each data point to one of K groups.
 - Clusters data points based on feature similarity.
- **Results:**
 - Centroids of the K clusters (used to label new data).
 - Labels for the training data (each point assigned to a single cluster).

K-means Clustering: Problem and Objective

Given a dataset $\mathcal{X} = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n\}$ where each $\mathbf{x}_i \in \mathbb{R}^d$, the goal of K-means is to partition the dataset into k clusters $\mathcal{C} = \{C_1, C_2, \dots, C_k\}$ such that the within-cluster sum of squares is minimized.

Objective Function:

$$J(\mathcal{C}, \boldsymbol{\mu}) = \sum_{i=1}^k \sum_{\mathbf{x} \in C_i} \|\mathbf{x} - \boldsymbol{\mu}_i\|^2$$

where:

- $\boldsymbol{\mu}_i$ is the centroid of cluster C_i .
- $\|\mathbf{x} - \boldsymbol{\mu}_i\|^2$ is the squared Euclidean distance between a data point \mathbf{x} and the centroid $\boldsymbol{\mu}_i$.

K-means Clustering: Algorithm Steps

The K-means algorithm proceeds as follows:

- ① **Initialization:** Randomly initialize k cluster centroids $\mu_1, \mu_2, \dots, \mu_k$.
- ② **Assignment Step:** Assign each data point \mathbf{x}_j to the nearest centroid:

$$C_i = \{\mathbf{x}_j : \|\mathbf{x}_j - \mu_i\|^2 \leq \|\mathbf{x}_j - \mu_l\|^2 \forall l \neq i\}$$

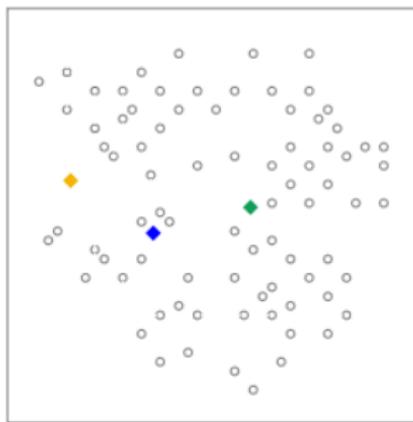
- ③ **Update Step:** Recompute the centroids as the mean of all points assigned to each cluster:

$$\mu_i = \frac{1}{|C_i|} \sum_{\mathbf{x} \in C_i} \mathbf{x}$$

- ④ **Repeat:** Repeat the assignment and update steps until convergence (i.e., when the centroids no longer change significantly or a maximum number of iterations is reached).

K-Means Algorithm - Example

1. Provide an initial guess for K . For this example, we choose $K = 3$
2. Randomly choose K centroids.

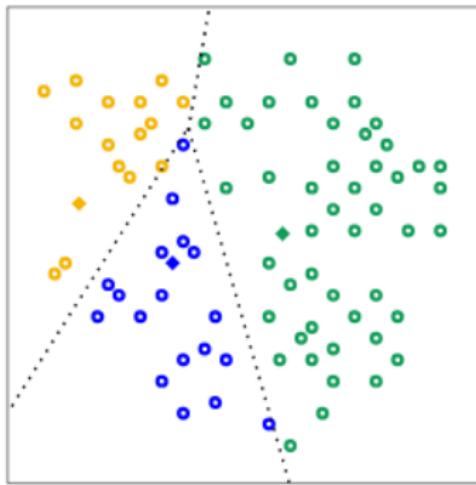


k-means at initialization

Source: Machine Learning Crash Course, Google

K-Means Algorithm - Example

3. Assign each point to the nearest centroid to get K initial clusters.

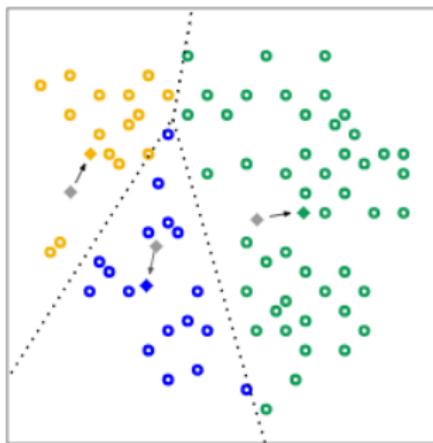


Initial clusters

Source: Machine Learning Crash Course, Google

K-Means Algorithm - Example

4. For each cluster, calculate a new centroid by taking the mean position of all points in the cluster. The arrows in the figure show the change in centroid positions.

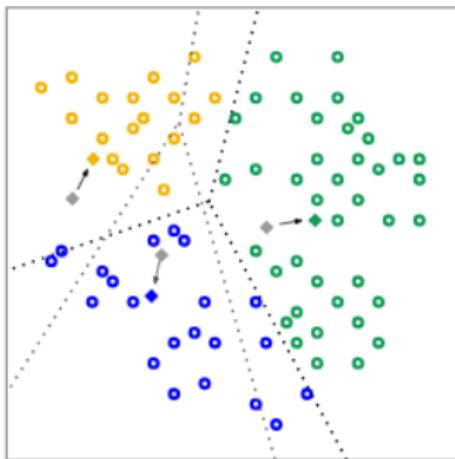


Recomputed centroids

Source: Machine Learning Crash Course, Google

K-Means Algorithm - Example

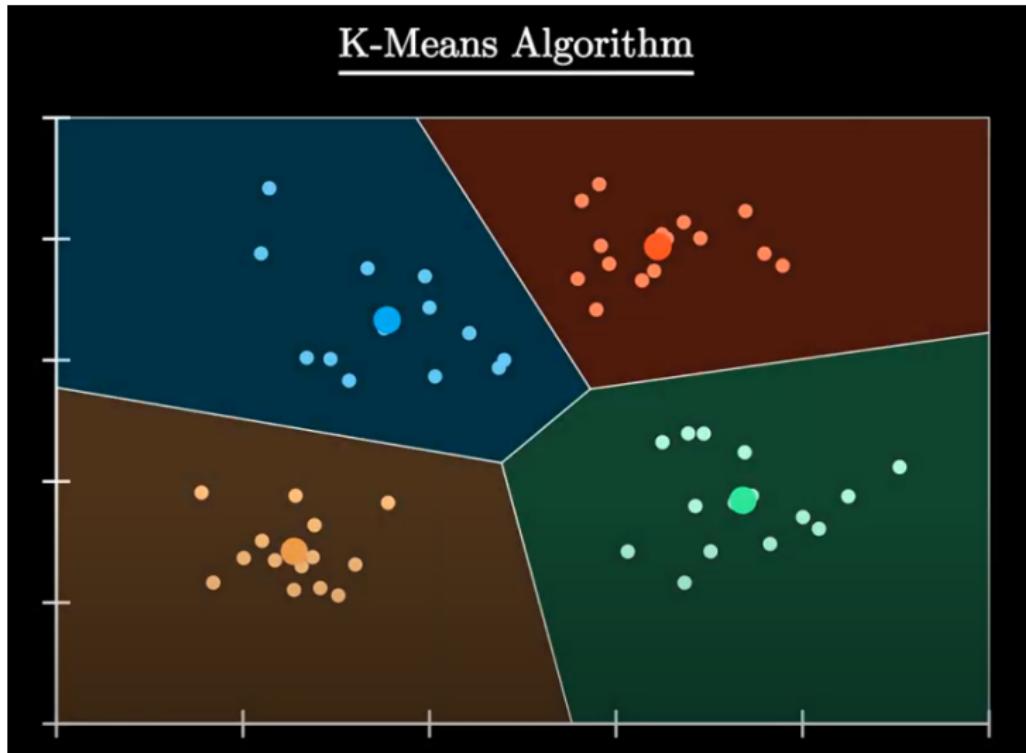
5. Reassign each point to the nearest new centroid.
6. Repeat steps 4 and 5, recalculating centroids and cluster membership, until points no longer change clusters.



Clusters after reassignment.

Source: Machine Learning Crash Course, Google

K-Means Algorithm Visualization

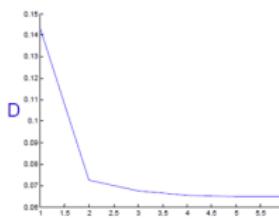
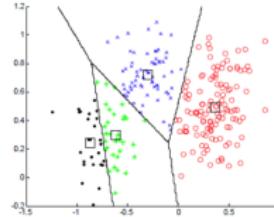
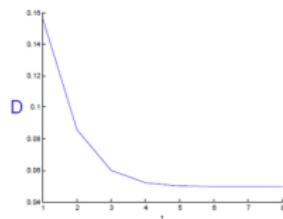
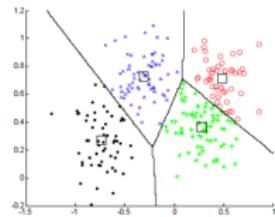
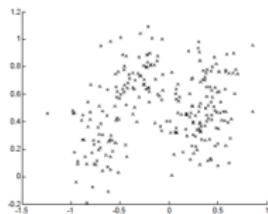


Video: K-Means Algorithm Visualization



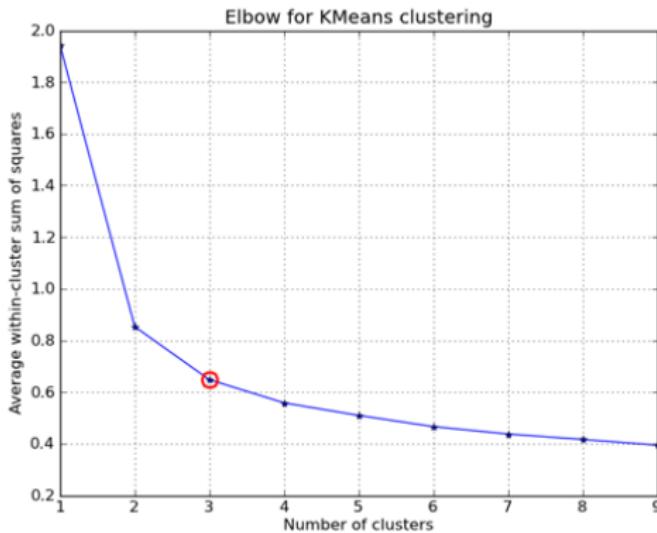
Limitations of K-Means

- Sensitive to initial centroid positions.
- Requires the number of clusters to be specified in advance.
- Struggles with non-spherical clusters and outliers.
- Not suitable for clusters of varying densities.



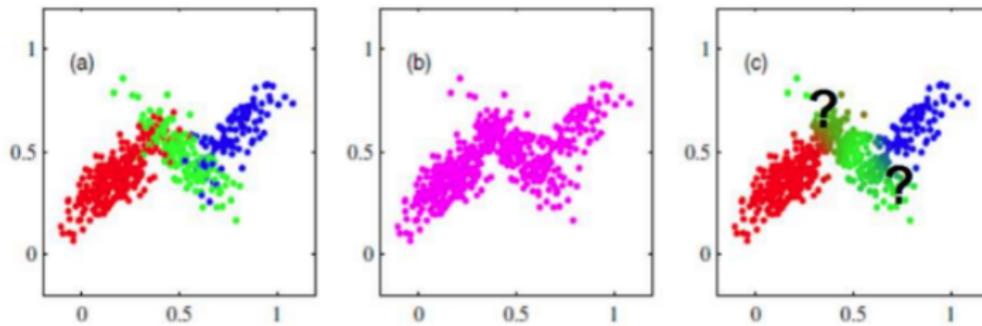
Choosing the Right Number of Clusters

- **Elbow Method:** Plot the sum of squared errors (SSE) vs. the number of clusters and look for the "elbow" point.
- **Silhouette Score:** Measures how similar a data point is to its own cluster compared to other clusters (range: -1 to 1).



Hard vs Soft Assignments

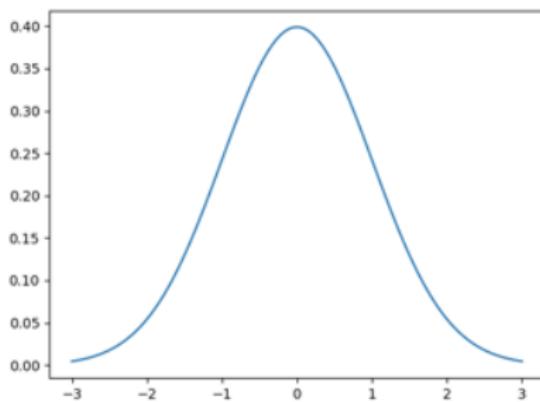
- In K-means, there is a hard assignment of feature vectors to a cluster.
- However, for feature vectors near the boundary this may be a poor representation.
- Instead, we can consider a soft-assignment, where the strength of the assignment depends on distance.
- To address these problems, we use soft clustering - Gaussian Mixture Model.
- Soft clustering is a form of clustering where observations may belong to multiple clusters.



Gaussian Distribution

- Gaussian / normal distribution / bell curve / probability density function (pdf). The function that describes the normal distribution is the following:

$$\mathcal{N}(x; \mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x - \mu)^2}{2\sigma^2}}$$



Multivariate Gaussians

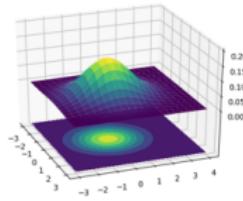
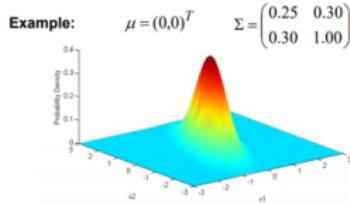
- For d dimensions, the Gaussian distribution of a vector $x = (x_1, x_2, \dots, x_d)^T$ is defined by:

- Formula:**

$$N(x|\mu, \Sigma) = \frac{1}{(2\pi)^{d/2} \sqrt{|\Sigma|}} \exp\left(-\frac{1}{2}(x - \mu)^T \Sigma^{-1} (x - \mu)\right)$$

- Parameters:**

- μ : Mean vector of the Gaussian.
- Σ : Covariance matrix of the Gaussian.



Introduction to Gaussian Mixture Model (GMM)

- GMM is a probabilistic model for clustering based on Gaussian distributions.
- Unlike K-Means, it assigns a probability to each data point belonging to every clusters.
- Uses the **Expectation-Maximization (EM)** algorithm for parameter estimation.
- **Key Concepts:**
 - Each cluster is represented by a Gaussian distribution.
 - The algorithm estimates the parameters of these distributions.

Expectation-Maximization (EM) Algorithm

Steps:

- ① Initialize Gaussian parameters (means, variances, weights).
- ② Expectation Step (E-Step): Compute probabilities for each data point.
- ③ Maximization Step (M-Step): Update parameters to maximize likelihood.
- ④ Repeat until convergence.

Expectation-Maximization (EM) Algorithm

- **Expectation Step:** Probabilistic assignment of data points to clusters.
- After initializing k random Gaussian models:
 - Calculate $E[z_{i,j}]$, the probability that x_i belongs to cluster j .
 - z_i : A vector of probabilities for x_i belonging to each cluster.
- **Formula:**

$$E[z_{i,j}] = \frac{P(x = x_i | \mu = \mu_j)}{\sum_{m=1}^k P(x = x_i | \mu = \mu_m)}$$

- **Gaussian Probability:**

$$P(x = x_i | \mu = \mu_j) = \frac{1}{\sqrt{2\pi\sigma_j^2}} e^{-\frac{(x_i - \mu_j)^2}{2\sigma_j^2}}$$

Expectation-Maximization (EM) Algorithm

- **Maximization Step:** Recalculate Gaussian models using weights from the Expectation Step.
- **Update Mean:**

$$\mu_j = \frac{\sum_i E[z_{i,j}] x_i}{\sum_i E[z_{i,j}]}$$

- **Update Covariance:**

$$\sigma_j = \frac{\sum_i E[z_{i,j}] (x_i - \mu_j) (x_i - \mu_k)}{\sum_i E[z_{i,j}]}$$

- **Key Idea:**
 - Use the probabilistic assignments $E[z_{i,j}]$ to update the parameters of the Gaussian models.
 - Iteratively refine the model to better fit the data.

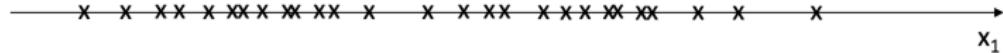
Expectation-Maximization (EM) Algorithm

Defining a stopping criterion:

- With k-means clustering, we iteratively recalculated means and reassigned observations until convergence, where observations stopped moving between clusters.
- However, since we're now dealing with soft clustering that involves continuous probabilities, we can't rely on this same type of convergence.
- Instead, we'll set a stopping criterion to end the iterative cycle. The cycle will stop once the observation probabilities stop changing by more than some threshold.

GMM - Example

- Suppose we have data for a set of observations with one feature, x_1 .
- We can use this data to build a Gaussian model for each class. This would allow us to calculate the probability of a new observation belonging to each class.

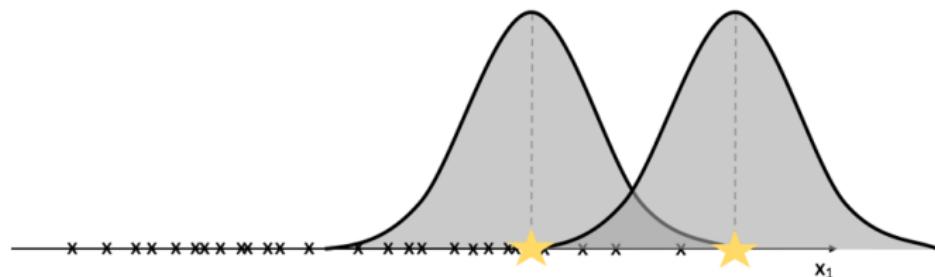


Source: Jeremy Jordan

GMM - Example

- Since we don't know which class each observation belongs to, we don't have an easy way to build multiple Gaussian models to partition the data.
- Let's start with a random initialization of our Gaussian models and then iteratively optimize the attributes.

Random Initialization

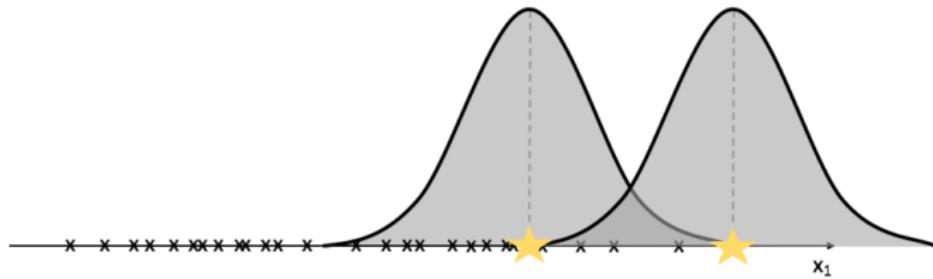


Source: Jeremy Jordan

GMM - Example

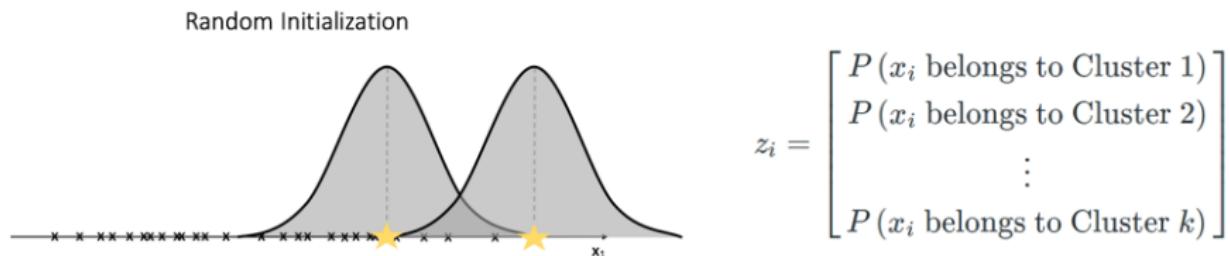
- After initializing two random Gaussian distributions, we will compute the likelihood of each observation under both Gaussian models. This likelihood is calculated using the probability density function of the Gaussian distribution.
- We will then recalculate the parameters of the Gaussian models. When calculating the mean and variance for each Gaussian, we will use all of the observations. However, each observation will be weighted by the likelihood of it belonging to the given model.

Random Initialization



GMM - Example

- Rather than assigning observations to a class, we'll assume there's a possibility that it can belong to any of the k clusters.
- Thus, for any given observation we'll calculate the probability of it belonging to each of the k clusters. Let's attach a hidden variable to each observation which contains a vector, z_i , to store these probabilities.

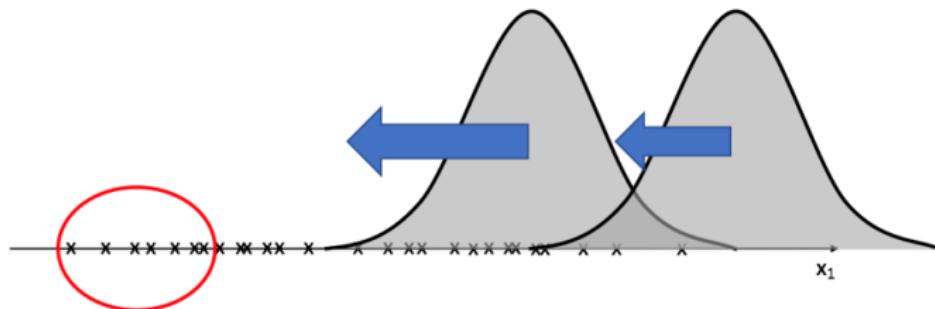


Source: Jeremy Jordan

GMM - Example

- After using z_i to calculate the weighted parameters (mean and variance) for each cluster, we update the Gaussian models.
- You'll see that the leftmost data (circled below) will pull both models in its direction, but it will have a larger effect on the left Gaussian since the observations will have a higher weight for the model that it's closest to.

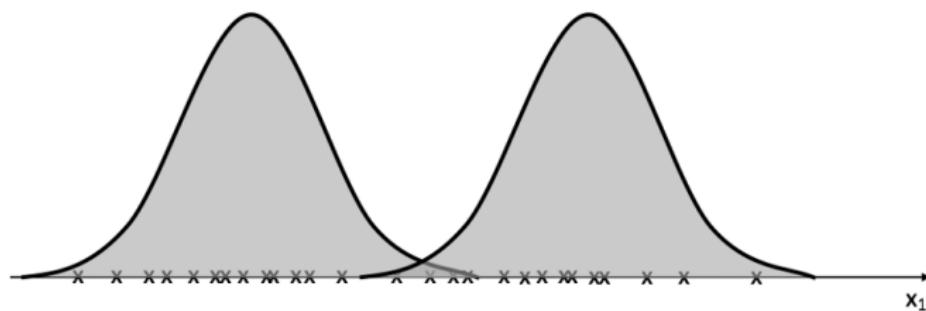
Update Gaussian models



Source: Jeremy Jordan

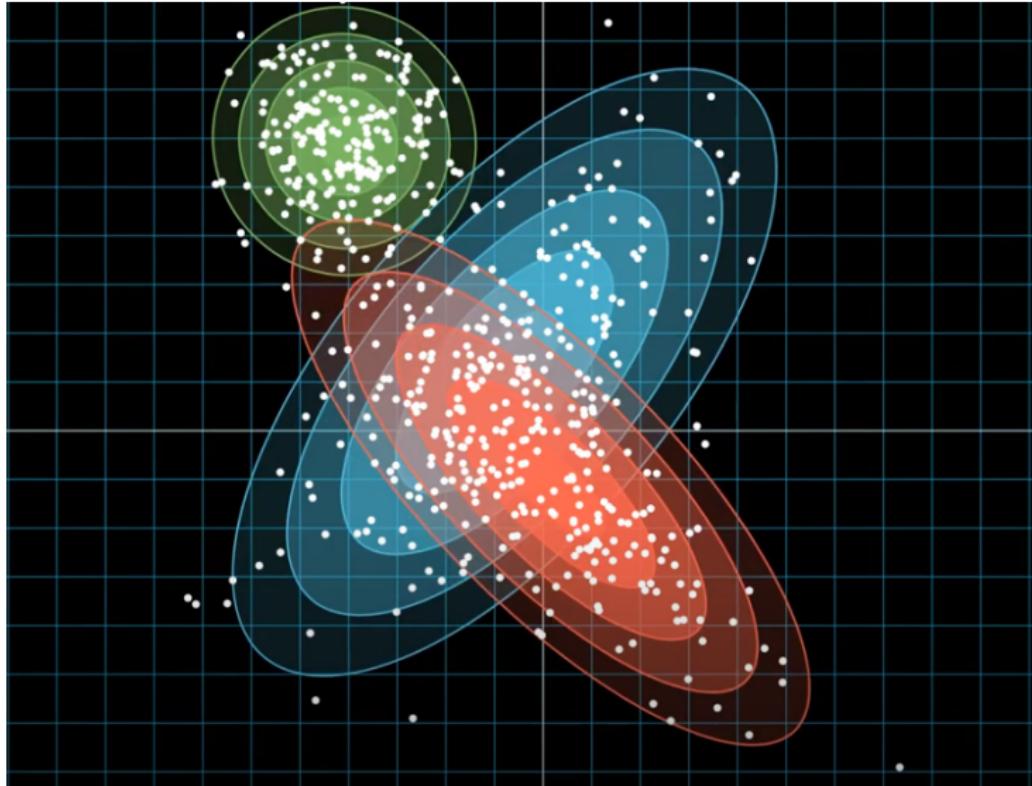
GMM - Example

- We'll continue this cycle of recalculating z_i and then using it to update the Gaussians until we "converge" on optimal clusters.
- The example visualizes a univariate Gaussian example, we can extend this logic to accommodate multivariate datasets.



Source: Jeremy Jordan

GMM Visualization



Video: GMM Visualization

GMM vs K-Means

- **K-Means:**

- Hard assignment (each point belongs to one cluster).
- Assumes spherical clusters of equal size.

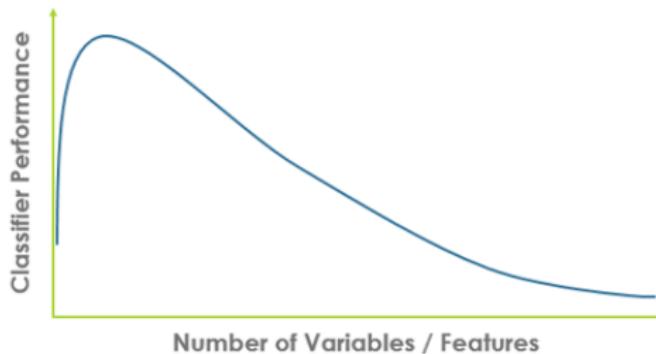
- **GMM:**

- Soft assignment (probabilistic membership in clusters).
- Can model clusters of different shapes and sizes.

Dimensionality Reduction

Curse of Dimensionality

- **Explanation:** As the number of dimensions increases, data becomes sparse, and distance metrics lose meaning.
- **Examples:**
 - High computational cost.
 - Overfitting in machine learning models.
 - Difficulty in visualizing and interpreting data.

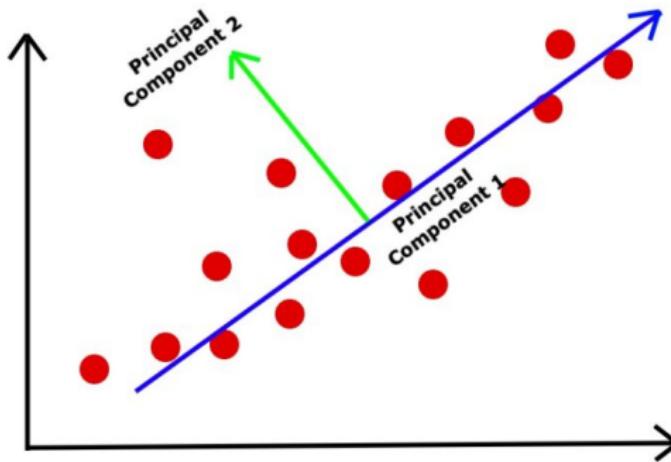


What is Dimensionality Reduction?

- **Definition:** A technique to reduce the number of features (dimensions) in a dataset while preserving its structure.
- **Motivation:**
 - Improves computational efficiency.
 - Reduces noise and redundancy.
 - Enables visualization of high-dimensional data.

Principal Component Analysis (PCA)

- **Theory:** A linear dimensionality reduction technique that projects data onto orthogonal axes (principal components).



Source: INFOARYAN

PCA Algorithm: Steps 1–2

Given a dataset $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n\}$ where each $\mathbf{x}_i \in \mathbb{R}^d$, the goal of PCA is to find a lower-dimensional representation of the data while preserving as much variance as possible.

① Standardize the Data

- Compute the mean of the dataset:

$$\bar{\mathbf{x}} = \frac{1}{N} \sum_{i=1}^N \mathbf{x}_i$$

- Center the data:

$$\mathbf{X}_{centered} = \mathbf{X} - \bar{\mathbf{x}}$$

② Compute the Covariance Matrix

- Covariance matrix captures the variance and correlation between features:

$$\mathbf{C} = \frac{1}{N-1} \mathbf{X}_{centered}^\top \mathbf{X}_{centered}$$

PCA Algorithm: Steps 3–4

③ Compute the Eigenvalues and Eigenvectors

- Solve the eigenvalue problem:

$$\mathbf{C}\mathbf{v}_i = \lambda_i \mathbf{v}_i$$

- λ_i is the eigenvalue, \mathbf{v}_i is the eigenvector.

④ Sort Eigenvectors by Eigenvalues

- Order eigenvectors by decreasing eigenvalues:

$$\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_D$$

- The eigenvectors corresponding to the largest eigenvalues capture the most variance.

PCA Algorithm: Steps 5–6

⑤ Select the Top K Eigenvectors

- Form a projection matrix \mathbf{W} :

$$\mathbf{W} = [\mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_K]$$

- K is the number of principal components you choose to keep.

⑥ Project the Data onto the New Subspace

- Transform the data into the new K -dimensional subspace:

$$\mathbf{Z} = \mathbf{X}_{centered} \mathbf{W}$$

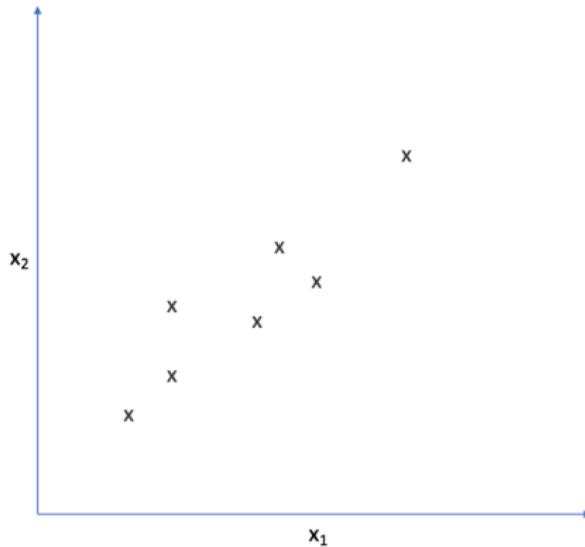
- \mathbf{Z} is the reduced representation of the original data.

PCA Algorithm Summary

- ① Standardize the data
- ② Compute the covariance matrix
- ③ Calculate eigenvalues and eigenvectors
- ④ Sort eigenvectors by eigenvalues
- ⑤ Select top K eigenvectors
- ⑥ Project data onto new subspace

Principal Components Analysis (PCA)

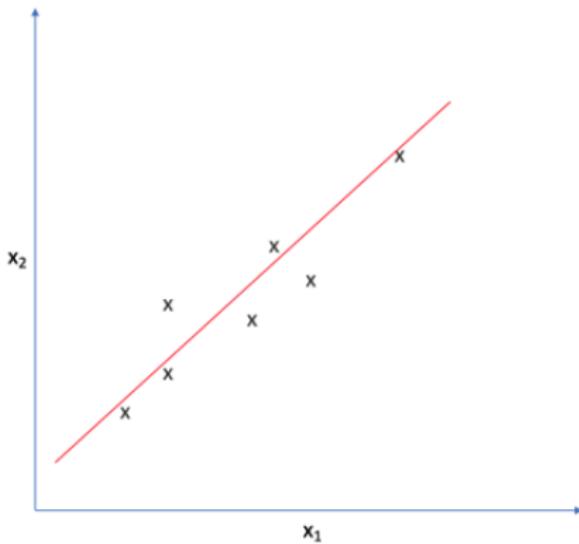
- Intuition: Let's look at the following two-dimensional feature-space.



Source: Jeremy Jordan

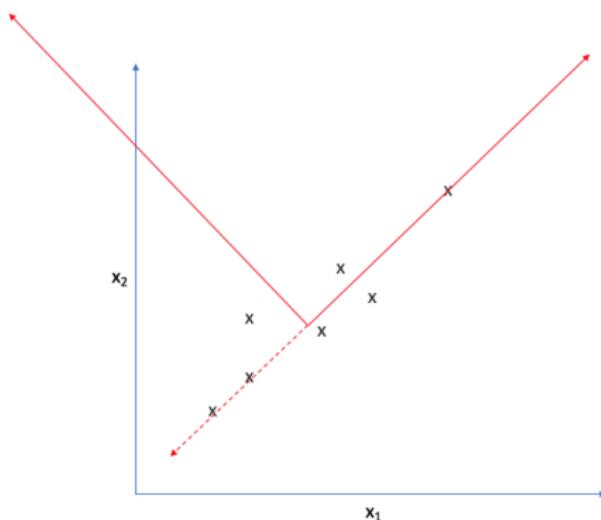
Principal Components Analysis (PCA)

- It appears that most of the points on this scatterplot lie along the following line. This suggests some degree of correlation between x_1 and x_2 .



Principal Components Analysis (PCA)

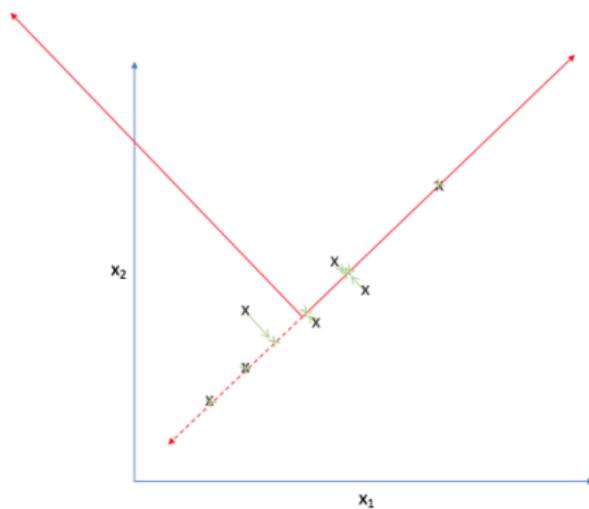
- As such, we could reorient the axes to be centered on the data and parallel to the line above.



Source: Jeremy Jordan

Principal Components Analysis (PCA)

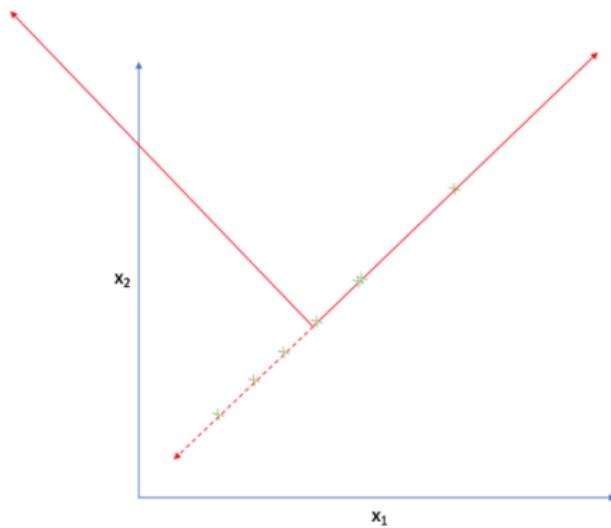
- Strictly speaking, we're still dealing with two dimensions. However, let's project each observation onto the primary axis.



Source: Jeremy Jordan

Principal Components Analysis (PCA)

- Now, every observation lies on the primary axis.



Source: Jeremy Jordan

Principal Components Analysis (PCA)

- We just compressed a two dimensional dataset into one dimension by translating and rotating our axes! After this transformation, we only really have one relevant dimension and thus we can discard the second axis.



Source: Jeremy Jordan

Principal Components Analysis (PCA)

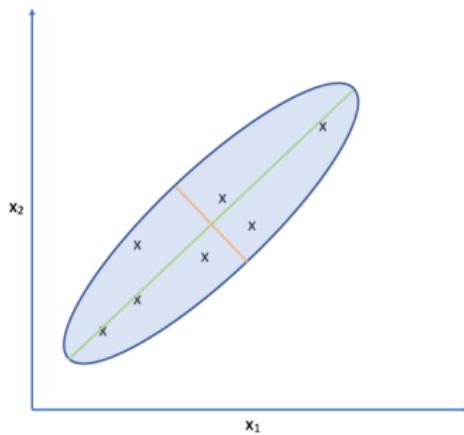
- Comparing the original observations with our new projections, we can see that it's not an exact representation of our data. However, one could argue it does capture the essence of our data - not exact, but enough to be meaningful.



Source: Jeremy Jordan

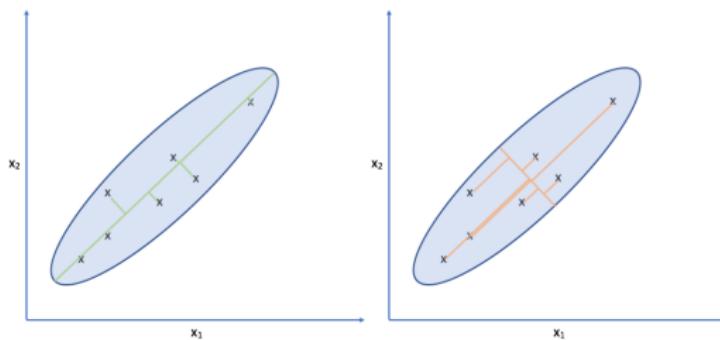
Principal Components Analysis (PCA)

- We can view the cumulative distance between our observations and projected points as a measure of information loss. Thus we'd like to orient the axes in a manner which minimizes this. How do we do this? That's where variance comes into play.
- Another example: look at the spread of the data along the green and orange directions. Notice that there's a much more deviation along the green direction than there is along the orange direction.



Principal Components Analysis (PCA)

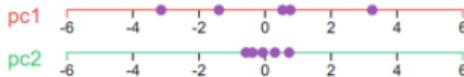
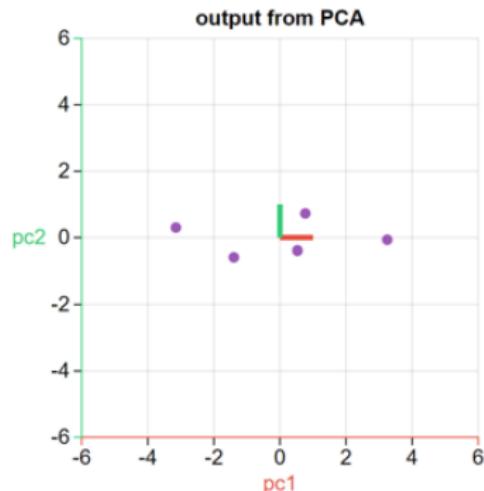
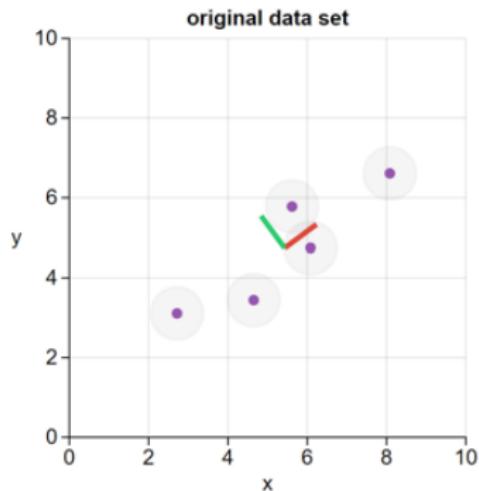
- Projecting our observations onto the orange vector requires us to move much further than we would need to for projecting onto the green vector. It turns out, the vector which is capable of capturing the maximum variance of the data minimizes the distance required to move our observations as projection onto the vector.



Source: Jeremy Jordan

Principal Components Analysis (PCA)

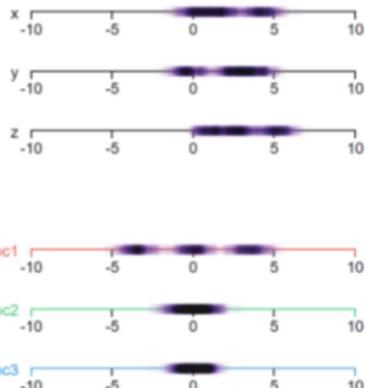
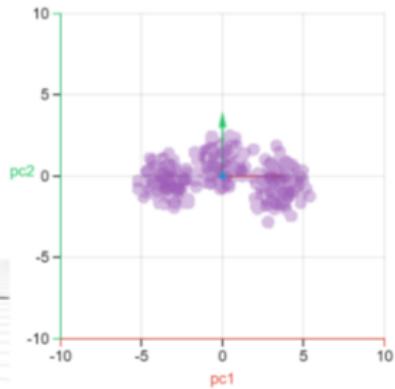
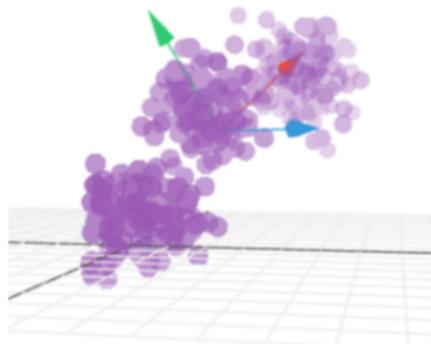
2D example



Source: Principal Component Analysis Explained Visually

Principal Components Analysis (PCA)

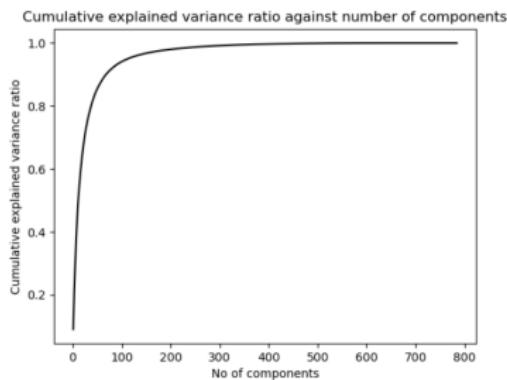
3D example



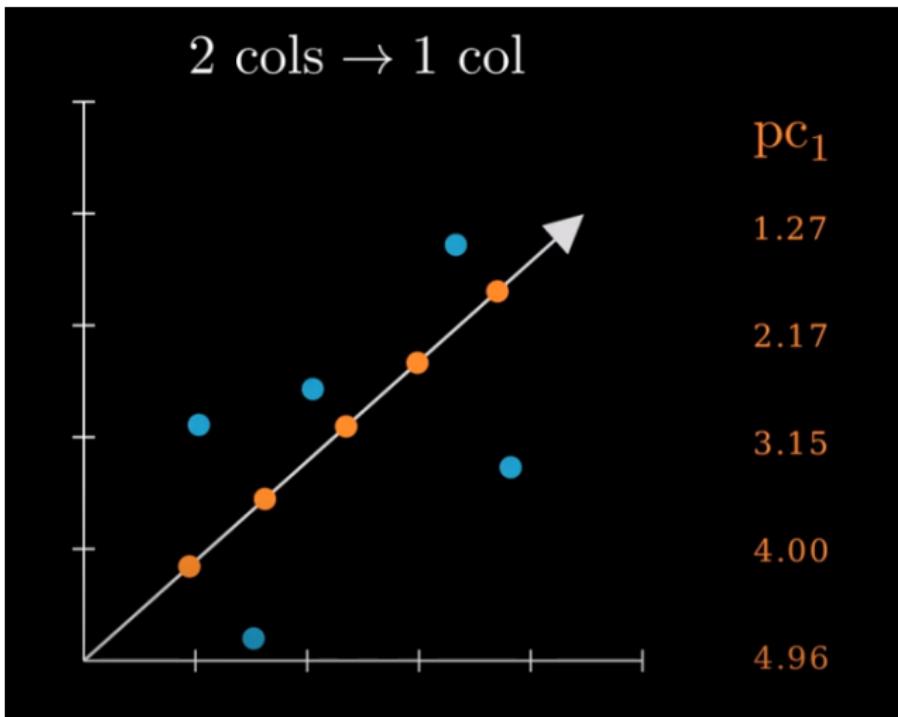
Source: Principal Component Analysis Explained Visually

Explained Variance Ratio

- **Definition:** The proportion of variance explained by each principal component. It determines how much information is retained in each principal component and helps in choosing the optimal number of components.
- **How to Choose:**
 - Plot the cumulative explained variance ratio.
 - Select the number of components that explain a significant portion of the variance (e.g., 95%).



PCA Visualization



Video: PCA Visualization

Limitations of PCA

- **Linearity Assumption:** PCA assumes linear relationships between variables.
- **Sensitivity to Scaling:** Requires standardized data.
- **Loss of Information:** May discard important variance in lower components.

COMP4131: Data Modelling and Analysis

Lecture 6: Linear Regression and Logistic Regression

Daokun Zhang

University of Nottingham Ningbo China

daokun.zhang@nottingham.edu.cn

March 24, 2025

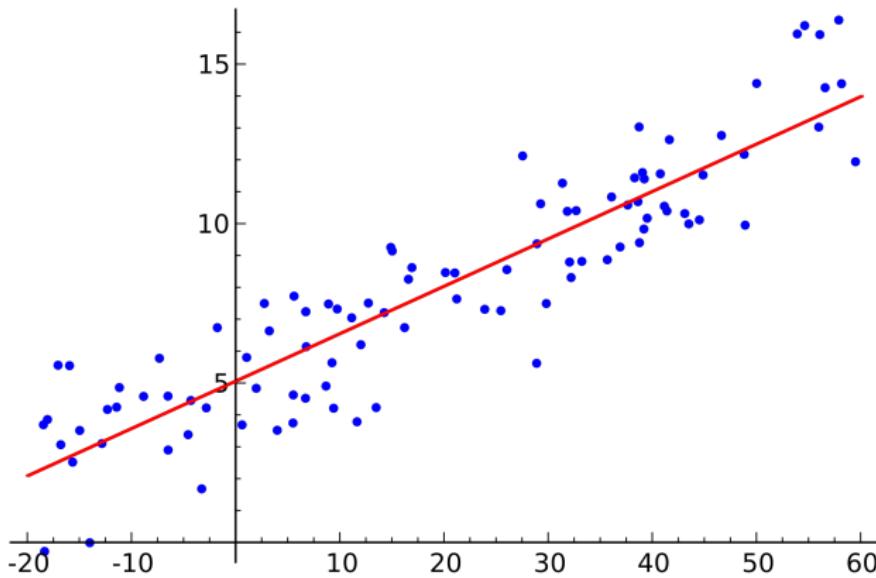
Overview

1 Linear Regression

2 Logistic Regression

Linear Regression

A Simple Example of Linear Regression



$$y = kx + b$$

Linear Regression with Multiple Input Variables

Given the input sample $\mathbf{x} = (x_1, x_2, \dots, x_D)^T$ with D feature dimensions, we can predict the target value $y \in \mathbb{R}$ as

$$y(\mathbf{x}, \mathbf{w}) = w_0 + w_1 x_1 + \dots + w_D x_D,$$

where $\mathbf{w} = (w_0, w_1, \dots, w_D)^T$ are the parameters.

Here $y(\mathbf{x}, \mathbf{w})$ is

- a linear function of the parameters w_0, w_1, \dots, w_D , and
- a linear function of the input variables x_1, x_2, \dots, x_D .

Linear Basis Function Models

We can extend the linear regression model to the linear basis function models by considering linear combinations of fixed nonlinear functions of the input variables, of the form

$$y(\mathbf{x}, \mathbf{w}) = w_0 + \sum_{j=1}^{M-1} w_j \phi_j(\mathbf{x}),$$

where $\phi_j(\mathbf{x})$ are the basis functions, and M is the number of parameters.

To be more concise, we define an additional dummy 'basis function' $\phi_0(\mathbf{x}) = 1$ for the *bias* parameter w_0 so that

$$y(\mathbf{x}, \mathbf{w}) = \sum_{j=0}^{M-1} w_j \phi_j(\mathbf{x}) = \mathbf{w}^T \phi(\mathbf{x})$$

where $\mathbf{w} = (w_0, \dots, w_{M-1})^T$ and $\phi(\mathbf{x}) = [\phi_0(\mathbf{x}), \dots, \phi_{M-1}(\mathbf{x})]^T$.

Examples of Basis Functions

- Feature Pre-processing: impute/remove missing values, feature normalization
- Feature Selection
- Dimension Reduction
- Identity Basis Function: $\phi(\mathbf{x}) = \mathbf{x}$
- Polynomial Basis Function: $\phi_j(x) = x^j$
- Gaussian Basis Function: $\phi_j(x) = \exp\left\{-\frac{(x-\mu_j)^2}{2s^2}\right\}$
- Sigmoidal Basis Function: $\phi_j(x) = \sigma\left(\frac{x-\mu_j}{s}\right)$ with $\sigma(a) = \frac{1}{1+\exp(-a)}$
- Fourier Basis Function
- ...

Maximum Likelihood and Least Squares

Now consider a data set of inputs $\mathbf{X} = \{\mathbf{x}_1, \dots, \mathbf{x}_N\}$ with corresponding target values $\mathbf{t} = \{t_1, \dots, t_N\}$, to find the best parameter \mathbf{w} , we need to minimize the sum-of-squares error function:

$$\begin{aligned} E_D(\mathbf{w}) &= \frac{1}{2} \sum_{n=1}^N \{t_n - y(\mathbf{x}, \mathbf{w})\}^2 \\ &= \frac{1}{2} \sum_{n=1}^N \{t_n - \mathbf{w}^T \phi(\mathbf{x}_n)\}^2. \end{aligned}$$

Question: Why the square error is used to design the objective function?

Maximum Likelihood and Least Squares

We assume that the target variable t is given by a deterministic function $y(\mathbf{x}, \mathbf{w})$ with additive Gaussian noise so that

$$t = y(\mathbf{x}, \mathbf{w}) + \epsilon,$$

where ϵ is a zero mean Gaussian random variable with precision (inverse variance) β . Thus we can write

$$p(t|\mathbf{x}, \mathbf{w}, \beta) = \mathcal{N}(t|y(\mathbf{x}, \mathbf{w}), \beta^{-1}),$$

where

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

Maximum Likelihood and Least Squares

For the given data set $\mathbf{X} = \{\mathbf{x}_1, \dots, \mathbf{x}_N\}$ and $\mathbf{t} = \{t_1, \dots, t_N\}$, the joint likelihood function can be expressed as

$$p(\mathbf{t}|\mathbf{X}, \mathbf{w}, \beta) = \prod_{n=1}^N \mathcal{N}(t_n | \mathbf{w}^T \phi(\mathbf{x}_n), \beta^{-1}).$$

Taking the logarithm of the likelihood function, and making use of the standard form for the univariate Gaussian, we have

$$\begin{aligned}\ln p(\mathbf{t}|\mathbf{w}, \beta) &= \sum_{n=1}^N \ln \mathcal{N}(t_n | \mathbf{w}^T \phi(\mathbf{x}_n), \beta^{-1}) \\ &= \frac{N}{2} \ln \beta - \frac{N}{2} \ln(2\pi) - \frac{\beta}{2} \sum_{n=1}^N \{t_n - \mathbf{w}^T \phi(\mathbf{x}_n)\}^2 \\ &= \frac{N}{2} \ln \beta - \frac{N}{2} \ln(2\pi) - \beta E_D(\mathbf{w}).\end{aligned}$$

Maximum Likelihood and Least Squares

The gradient of the sum-of-squares error function $E_D(\mathbf{w})$ takes the form

$$\nabla E_D(\mathbf{w}) = - \sum_{n=1}^N \{ t_n - \mathbf{w}^T \phi(\mathbf{x}_n) \} \phi(\mathbf{x}_n)^T.$$

Setting this gradient to zero gives

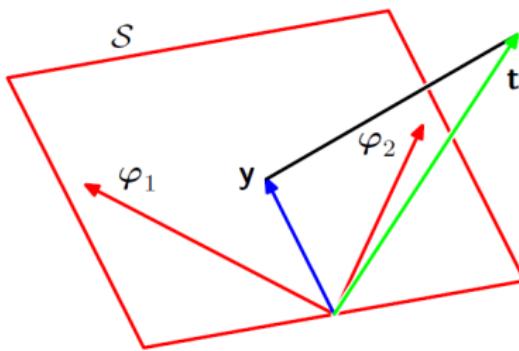
$$0 = - \sum_{n=1}^N t_n \phi(\mathbf{x}_n)^T + \mathbf{w}^T \left(\sum_{n=1}^N \phi(\mathbf{x}_n) \phi(\mathbf{x}_n)^T \right).$$

Solving for \mathbf{w} we obtain

$$\mathbf{w}^* = (\Phi^T \Phi)^{-1} \Phi^T \mathbf{t},$$

which are known as the *normal equations* for the least squares problem.
Here $\Phi = [\phi(\mathbf{x}_1), \dots, \phi(\mathbf{x}_N)]^T$ is an $N \times M$ matrix.

Geometry Interpretation of Least Squares



- consider $\mathbf{t} = \{t_1, \dots, t_N\}$ is a vector in an N -dimensional space
- define $\varphi_j = \{\phi_j(\mathbf{x}_1), \dots, \phi_j(\mathbf{x}_N)\}$ as the j th column vector of Φ
- $\{\varphi_0, \dots, \varphi_{M-1}\}$ span a subspace \mathcal{S} of dimensionality M , if $M < N$
- the least-squares solution corresponds to the choice of $\mathbf{y} = \{y(\mathbf{x}_1, w), \dots, y(\mathbf{x}_N, w)\}$ that lies in subspace \mathcal{S} and that is closest to \mathbf{t} , i.e., the projection of \mathbf{t} onto \mathcal{S}

Regularized Least Squares

To avoid over-fitting, we consider adding a regularization term to an error function, so that the total error function to be minimized takes the form

$$E(\mathbf{w}) = E_D(\mathbf{w}) + \lambda E_W(\mathbf{w}),$$

where λ is the regularization coefficient, $E_D(\mathbf{w})$ is the data-dependent error, and $E_W(\mathbf{w})$ is the regularization term:

$$E_D(\mathbf{w}) = \frac{1}{2} \sum_{n=1}^N \{ t_n - \mathbf{w}^T \phi(\mathbf{x}_n) \}^2,$$

$$E_W(\mathbf{w}) = \frac{1}{2} \mathbf{w}^T \mathbf{w}.$$

Then the total error function becomes

$$E(\mathbf{w}) = \frac{1}{2} \sum_{n=1}^N \{ t_n - \mathbf{w}^T \phi(\mathbf{x}_n) \}^2 + \frac{\lambda}{2} \mathbf{w}^T \mathbf{w}.$$

Regularized Least Squares

The gradient of the new error function $E(\mathbf{w})$ takes the form

$$\nabla E(\mathbf{w}) = - \sum_{n=1}^N \{t_n - \mathbf{w}^T \phi(\mathbf{x}_n)\} \phi(\mathbf{x}_n)^T + \lambda \mathbf{w}.$$

Setting this gradient to zero gives

$$0 = - \sum_{n=1}^N t_n \phi(\mathbf{x}_n)^T + \mathbf{w}^T \left(\sum_{n=1}^N \phi(\mathbf{x}_n) \phi(\mathbf{x}_n)^T \right) + \lambda \mathbf{w}^T.$$

Solving for \mathbf{w} we obtain

$$\mathbf{w}^* = (\lambda \mathbf{I} + \Phi^T \Phi)^{-1} \Phi^T \mathbf{t},$$

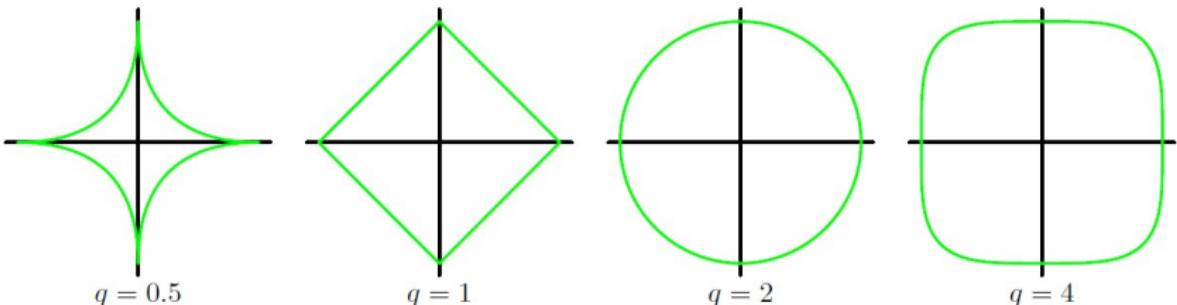
where \mathbf{I} is the $M \times M$ identity matrix.

Regularized Least Squares

A more general regularizer is sometimes used, for which the regularized error takes the form

$$E(\mathbf{w}) = \frac{1}{2} \sum_{n=1}^N \{ t_n - \mathbf{w}^T \phi(\mathbf{x}_n) \}^2 + \frac{\lambda}{2} \sum_{j=1}^M |w_j|^q,$$

where $q = 2$ corresponds to the **quadratic regularizer**, and the case of $q = 1$ known as the **lasso regularizer**.

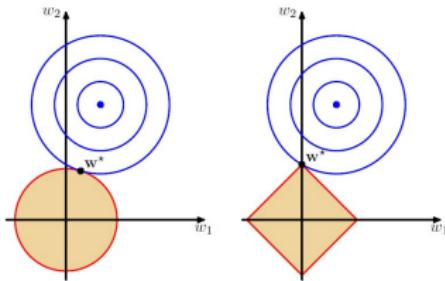


Contours of the regularization term for various values of the parameter q .

Regularized Least Squares

Minimizing the regularized error $E(\mathbf{w})$ is equivalent to

$$\min_{\mathbf{w}} \underbrace{\frac{1}{2} \sum_{n=1}^N \{t_n - \mathbf{w}^T \phi(\mathbf{x}_n)\}^2}_{E_D(\mathbf{w})}, \text{ s.t. } \sum_{j=1}^M |w_j|^q \leq \eta.$$



The contours of the unregularized error function (blue) along with the constraint region for the quadratic regularizer $q = 2$ on the left and the lasso regularizer $q = 1$ on the right, in which the optimum value for the parameter vector \mathbf{w} is denoted by \mathbf{w}^* . The lasso gives a sparse solution in which $w_1^* = 0$.

The Bias-Variance Decomposition

Given the joint distribution $p(\mathbf{x}, t)$ of the input \mathbf{x} and target value t , we can determine the specific estimate $y(\mathbf{x})$ of the value t for each input \mathbf{x} (a mapping function from \mathbf{x} to t), as the function that minimizes the expected squared loss

$$\mathbb{E}_{(\mathbf{x}, t)}[L] = \iint \{y(\mathbf{x}) - t\}^2 p(\mathbf{x}, t) d\mathbf{x} dt.$$

The optimal $y(\mathbf{x})$ should be

$$y^*(\mathbf{x}) = h(\mathbf{x}) = \int t p(t|\mathbf{x}) dt = \mathbb{E}_t[t|\mathbf{x}].$$

For a given estimate function $y(\mathbf{x})$, the expected squared loss can be expressed as

$$\begin{aligned}\mathbb{E}_{(\mathbf{x}, t)}[L] &= \iint \{y(\mathbf{x}) - h(\mathbf{x}) + h(\mathbf{x}) - t\}^2 p(\mathbf{x}, t) d\mathbf{x} dt \\ &= \int \{y(\mathbf{x}) - h(\mathbf{x})\}^2 p(\mathbf{x}) d\mathbf{x} + \iint \{h(\mathbf{x}) - t\}^2 p(\mathbf{x}, t) d\mathbf{x} dt.\end{aligned}$$

The Bias-Variance Decomposition

In general, the prediction function $y(\mathbf{x})$ is learned from a given data set \mathcal{D} , then we use $y(\mathbf{x}; \mathcal{D})$ to denote $y(\mathbf{x})$, and write the squared difference between $y(\mathbf{x})$ and $h(\mathbf{x})$ as

$$\{y(\mathbf{x}; \mathcal{D}) - h(\mathbf{x})\}^2.$$

If we add and subtract the quantity $\mathbb{E}_{\mathcal{D}}[y(\mathbf{x}; \mathcal{D})]$ inside the braces, and then expand, we obtain

$$\begin{aligned}& \{y(\mathbf{x}; \mathcal{D}) - \mathbb{E}_{\mathcal{D}}[y(\mathbf{x}; \mathcal{D})] + \mathbb{E}_{\mathcal{D}}[y(\mathbf{x}; \mathcal{D})] - h(\mathbf{x})\}^2 \\&= \{y(\mathbf{x}; \mathcal{D}) - \mathbb{E}_{\mathcal{D}}[y(\mathbf{x}; \mathcal{D})]\}^2 + \{\mathbb{E}_{\mathcal{D}}[y(\mathbf{x}; \mathcal{D})] - h(\mathbf{x})\}^2 \\&\quad + 2 \{y(\mathbf{x}; \mathcal{D}) - \mathbb{E}_{\mathcal{D}}[y(\mathbf{x}; \mathcal{D})]\} \{\mathbb{E}_{\mathcal{D}}[y(\mathbf{x}; \mathcal{D})] - h(\mathbf{x})\}.\end{aligned}$$

We now take the expectation of this expression with respect to \mathcal{D} and note that the final term will vanish, giving

The Bias-Variance Decomposition

$$\begin{aligned}\mathbb{E}_{\mathcal{D}} [\{y(\mathbf{x}; \mathcal{D}) - h(\mathbf{x})\}^2] \\ = \underbrace{\{\mathbb{E}_{\mathcal{D}}[y(\mathbf{x}; \mathcal{D})] - h(\mathbf{x})\}^2}_{\text{(bias)}^2} + \underbrace{\mathbb{E}_{\mathcal{D}} [\{y(\mathbf{x}; \mathcal{D}) - \mathbb{E}_{\mathcal{D}}[y(\mathbf{x}; \mathcal{D})]\}^2]}_{\text{variance}}.\end{aligned}$$

So far, we have considered a single input value \mathbf{x} . If we substitute this expansion back into the expected loss

$$\begin{aligned}\mathbb{E}_{\mathcal{D}} [\mathbb{E}_{(\mathbf{x}, t)}[L]] &= \mathbb{E}_{\mathcal{D}} \left[\int \{y(\mathbf{x}; \mathcal{D}) - h(\mathbf{x})\}^2 p(\mathbf{x}) d\mathbf{x} + \iint \{h(\mathbf{x}) - t\}^2 p(\mathbf{x}, t) d\mathbf{x} dt \right] \\ &= \int \mathbb{E}_{\mathcal{D}} [\{y(\mathbf{x}; \mathcal{D}) - h(\mathbf{x})\}^2] p(\mathbf{x}) d\mathbf{x} + \iint \{h(\mathbf{x}) - t\}^2 p(\mathbf{x}, t) d\mathbf{x} dt \\ &= \int \{\mathbb{E}_{\mathcal{D}}[y(\mathbf{x}; \mathcal{D})] - h(\mathbf{x})\}^2 p(\mathbf{x}) d\mathbf{x} \\ &\quad + \int \mathbb{E}_{\mathcal{D}} [\{y(\mathbf{x}; \mathcal{D}) - \mathbb{E}_{\mathcal{D}}[y(\mathbf{x}; \mathcal{D})]\}^2] p(\mathbf{x}) d\mathbf{x} + \iint \{h(\mathbf{x}) - t\}^2 p(\mathbf{x}, t) d\mathbf{x} dt.\end{aligned}$$

The Bias-Variance Decomposition

We obtain the following decomposition of the expected squared loss

$$\text{expected loss} = (\text{bias})^2 + \text{variance} + \text{noise},$$

where

$$(\text{bias})^2 = \int \{\mathbb{E}_{\mathcal{D}}[y(\mathbf{x}; \mathcal{D})] - h(\mathbf{x})\}^2 p(\mathbf{x}) d\mathbf{x},$$

$$\text{variance} = \int \mathbb{E}_{\mathcal{D}} \left[\{y(\mathbf{x}; \mathcal{D}) - \mathbb{E}_{\mathcal{D}}[y(\mathbf{x}; \mathcal{D})]\}^2 \right] p(\mathbf{x}) d\mathbf{x},$$

$$\text{noise} = \iint \{h(\mathbf{x}) - t\}^2 p(\mathbf{x}, t) d\mathbf{x} dt.$$

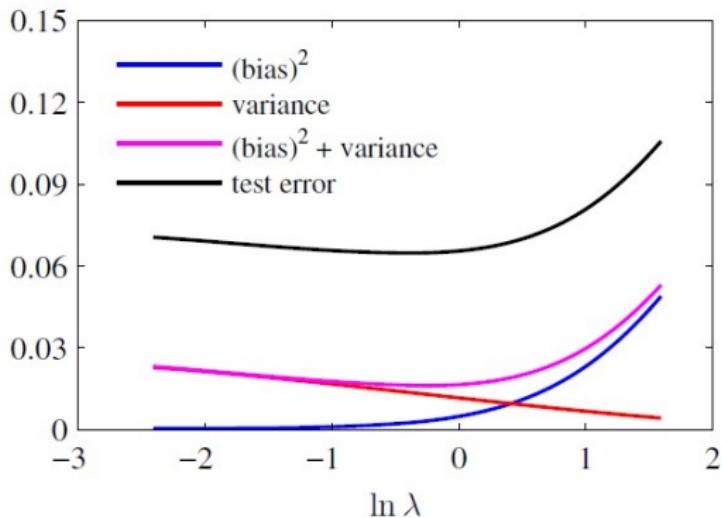
The Bias-Variance Decomposition

Given the regularized error

$$E(\mathbf{w}) = \frac{1}{2} \sum_{n=1}^N \{t_n - \mathbf{w}^T \phi(\mathbf{x}_n)\}^2 + \frac{\lambda}{2} \sum_{j=1}^M |w_j|^q,$$

- large λ : restrict the model's freedom for fitting data, leading to **high bias and low variance**
- small λ : warrant more freedom to fit data, resulting in **low bias and high variance**

The Bias-Variance Decomposition



Plot of squared bias and variance, together with their sum, as well as the test error, for a linear regression model with the sinusoidal basis function and the quadratic regularizer. The minimum value of $(\text{bias})^2 + \text{variance}$ occurs around $\ln \lambda = 0.31$, which is close to the value that gives the minimum error on the test data.

Logistic Regression

Logistic Regression for Binary Classification

Given the feature vector of an example \mathbf{x} , binary classification aims to determine its class label from the label candidate set $\{\mathcal{C}_1, \mathcal{C}_2\}$.

From probabilistic view, the class label corresponds to the label \mathcal{C}_k that maximizes the conditional probability $p(\mathcal{C}_k|\mathbf{x})$.

Suppose we can model class-conditional densities $p(\mathbf{x}|\mathcal{C}_k)$, as well as the class priors $p(\mathcal{C}_k)$, the posterior probabilities $p(\mathcal{C}_k|\mathbf{x})$ can be computed through Bayes' theorem.

$$\begin{aligned} p(\mathcal{C}_1|\mathbf{x}) &= \frac{p(\mathbf{x}|\mathcal{C}_1) p(\mathcal{C}_1)}{p(\mathbf{x}|\mathcal{C}_1) p(\mathcal{C}_1) + p(\mathbf{x}|\mathcal{C}_2) p(\mathcal{C}_2)} \\ &= \frac{1}{1 + \exp(-a)} = \sigma(a), \end{aligned}$$

where we have defined $a = \ln \frac{p(\mathbf{x}|\mathcal{C}_1)p(\mathcal{C}_1)}{p(\mathbf{x}|\mathcal{C}_2)p(\mathcal{C}_2)}$ and $\sigma(a)$ is the *sigmoid* function defined by $\sigma(a) = \frac{1}{1+\exp(-a)}$.

Logistic Regression for Binary Classification

We can use a linear basis function model to parameterize the log of probability ratio $a = \ln \frac{p(\mathbf{x}|\mathcal{C}_1)p(\mathcal{C}_1)}{p(\mathbf{x}|\mathcal{C}_2)p(\mathcal{C}_2)}$, by setting $a = \mathbf{w}^T \phi(\mathbf{x})$ so that

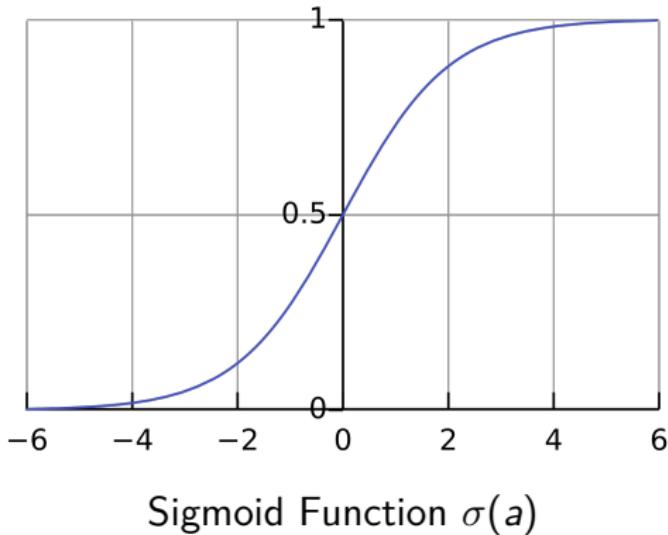
$$p(\mathcal{C}_1|\mathbf{x}) = \sigma(\mathbf{w}^T \phi(\mathbf{x})), \text{ and } p(\mathcal{C}_2|\mathbf{x}) = 1 - p(\mathcal{C}_1|\mathbf{x}).$$

Now, we get the *logistic regression* model.

Next, we use maximum likelihood to determine the parameters of the logistic regression model. To do this, we shall make use of the derivative of the sigmoid function, which can conveniently be expressed in terms of the sigmoid function

$$\frac{d\sigma}{da} = \sigma(1 - \sigma).$$

Logistic Regression for Binary Classification



Logistic Regression for Binary Classification

For a data set $\{\mathbf{x}_n, t_n\}$, where $t_n \in \{0, 1\}$, with $n = 1, \dots, N$, the likelihood function can be written as

$$p(\mathbf{t}|\mathbf{w}) = \prod_{n=1}^N \{y_n(\mathbf{w})\}^{t_n} \{1 - y_n(\mathbf{w})\}^{1-t_n},$$

where $\mathbf{t} = (t_1, \dots, t_N)^T$ and $y_n(\mathbf{w}) = p(\mathcal{C}_1|\mathbf{x}_n) = \sigma(\mathbf{w}^T \phi(\mathbf{x}_n))$. We can define an error function by taking the negative logarithm of the likelihood, which gives the *cross-entropy* error function in the form

$$\begin{aligned} E(\mathbf{w}) &= -\ln p(\mathbf{t}|\mathbf{w}) \\ &= -\sum_{n=1}^N \{t_n \ln y_n(\mathbf{w}) + [1 - t_n] \ln [1 - y_n(\mathbf{w})]\}. \end{aligned}$$

To avoid overfitting, we can also add a regularization term to the *cross-entropy* error function.

Logistic Regression for Binary Classification

Taking the gradient of the error function with respect to \mathbf{w} , we obtain

$$\nabla E(\mathbf{w}) = \sum_{n=1}^N \{y_n(\mathbf{w}) - t_n\} \phi_n(\mathbf{x}),$$

we have made use of the equation where we have made use of the equation $\frac{d\sigma}{da} = \sigma(1 - \sigma)$. We see that the contribution to the gradient from data point n is given by the 'error' $y_n(\mathbf{w}) - t_n$ between the target value and the prediction of the model, times the basis function vector $\phi_n(\mathbf{x})$.

Iterative Reweighted Least Squares

In the case of the linear regression models, whose sum-of-squares error function is a quadratic function of \mathbf{w} , we can obtain the closed-form optimal solution of \mathbf{w} by setting the gradient to zero.

For logistic regression, there is no longer a closed-form solution, due to the nonlinearity of the logistic sigmoid function.

However, the departure from a quadratic form is not substantial. To be precise, the error function is convex, and hence has a unique minimum.

Furthermore, the error function can be minimized by an efficient iterative technique based on the Newton-Raphson iterative optimization scheme:

$$\mathbf{w}^{\text{new}} = \mathbf{w}^{\text{old}} - \mathbf{H}(\mathbf{w}^{\text{old}})^{-1} \nabla E(\mathbf{w}^{\text{old}}),$$

where $\mathbf{H}(\mathbf{w}^{\text{old}})$ is the Hessian matrix at $\mathbf{w} = \mathbf{w}^{\text{old}}$, whose elements comprise the second derivatives of $E(\mathbf{w})$ with respect to the components of \mathbf{w} .

Iterative Reweighted Least Squares

The gradient and Hessian matrix of *logistic regression's cross-entropy* error function are given by

$$\nabla E(\mathbf{w}) = \sum_{n=1}^N \{y_n(\mathbf{w}) - t_n\} \phi(\mathbf{x}_n) = \Phi^T \{\mathbf{y}(\mathbf{w}) - \mathbf{t}\},$$

$$\mathbf{H} = \nabla \nabla E(\mathbf{w}) = \sum_{n=1}^N y_n(\mathbf{w}) \{1 - y_n(\mathbf{w})\} \phi(\mathbf{x}_n) \phi(\mathbf{x}_n)^T = \Phi^T \mathbf{R}(\mathbf{w}) \Phi,$$

where $\mathbf{y}(\mathbf{w}) = [y_1(\mathbf{w}), \dots, y_N(\mathbf{w})]^T$, $\Phi = [\phi(\mathbf{x}_1), \dots, \phi(\mathbf{x}_N)]^T$ is an $N \times M$ matrix, and $\mathbf{R}(\mathbf{w})$ is an $N \times N$ diagonal matrix with entries

$$R_{nn}(\mathbf{w}) = y_n(\mathbf{w}) \{1 - y_n(\mathbf{w})\}.$$

Iterative Reweighted Least Squares

The Newton-Raphson update formula for the *logistic regression* model then becomes

$$\begin{aligned}\mathbf{w}^{\text{new}} &= \mathbf{w}^{\text{old}} - \left\{ \Phi^T \mathbf{R}(\mathbf{w}^{\text{old}}) \Phi \right\}^{-1} \Phi^T \left\{ \mathbf{y}(\mathbf{w}^{\text{old}}) - \mathbf{t} \right\} \\ &= \left\{ \Phi^T \mathbf{R}(\mathbf{w}^{\text{old}}) \Phi \right\}^{-1} \left\{ \Phi^T \mathbf{R}(\mathbf{w}^{\text{old}}) \Phi \mathbf{w}^{\text{old}} - \Phi^T \left\{ \mathbf{y}(\mathbf{w}^{\text{old}}) - \mathbf{t} \right\} \right\} \\ &= \left\{ \Phi^T \mathbf{R}(\mathbf{w}^{\text{old}}) \Phi \right\}^{-1} \Phi^T \mathbf{R}(\mathbf{w}^{\text{old}}) \mathbf{z}(\mathbf{w}^{\text{old}}),\end{aligned}$$

taking the form of a set of normal equations for a weighted least-squares problem, where $\mathbf{z}(\mathbf{w}^{\text{old}})$ is an N -dimensional vector with elements

$$\mathbf{z} = \Phi \mathbf{w}^{\text{old}} - \mathbf{R}(\mathbf{w}^{\text{old}})^{-1} \left\{ \mathbf{y}(\mathbf{w}^{\text{old}}) - \mathbf{t} \right\}.$$

Logistic Regression for Multi-Class Classification

For multi-class classification with label candidate set $\{\mathcal{C}_1, \mathcal{C}_2, \dots, \mathcal{C}_K\}$, the posterior probabilities $p(\mathcal{C}_k|\mathbf{x})$ are given by a softmax transformation of linear functions of the feature variables

$$p(\mathcal{C}_k|\mathbf{x}) = y_k = \frac{\exp(a_k)}{\sum_{j=1}^K \exp(a_j)},$$

where the ‘activations’ a_k are given by

$$a_k = \mathbf{w}_k^T \phi(\mathbf{x}).$$

We consider the use of maximum likelihood to determine the parameters $\{\mathbf{w}_k\}$ of this model. To do this, we will require the derivatives of y_k with respect to all of the activations a_j . These are given by

$$\frac{\partial y_k}{\partial a_j} = y_k (I_{kj} - y_j),$$

where I_{kj} are the elements of the identity matrix, with $I_{kj} = 1$ for $k = j$ and $I_{kj} = 0$ for $k \neq j$.

Logistic Regression for Multi-Class Classification

Next we write down the likelihood function. This is most easily done using the 1-of- K coding scheme in which the target vector \mathbf{t}_n for a feature vector $\phi(\mathbf{x}_n)$ belonging to class \mathcal{C}_k is a binary vector with all elements zero except for element k , which equals one:

$$\mathbf{t}_n = (0, \dots, 1, \dots, 0)^T.$$

↑
k

The likelihood function is then given by

$$p(\mathbf{T}|\mathbf{w}_1, \dots, \mathbf{w}_K) = \prod_{n=1}^N \prod_{k=1}^K p(\mathcal{C}_k|\mathbf{x}_n)^{t_{nk}} = \prod_{n=1}^N \prod_{k=1}^K y_{nk}^{t_{nk}},$$

where $y_{nk} = p(\mathcal{C}_k|\mathbf{x}_n)$, $\mathbf{T} = [\mathbf{t}_1, \dots, \mathbf{t}_N]^T$ is an $N \times M$ matrix target variables with elements t_{nk} .

Logistic Regression for Multi-Class Classification

Taking the negative logarithm then gives of the likelihood function then gives

$$E(\mathbf{w}_1, \dots, \mathbf{w}_K) = -\ln p(\mathbf{T}|\mathbf{w}_1, \dots, \mathbf{w}_K) = -\sum_{n=1}^N \sum_{k=1}^K t_{nk} \ln y_{nk},$$

which is known as the *cross-entropy* error function for the multiclass classification problem.

We now take the gradient of the error function with respect to one of the parameter vectors \mathbf{w}_j :

$$\nabla_{\mathbf{w}_j} E(\mathbf{w}_1, \dots, \mathbf{w}_K) = \sum_{n=1}^N (y_{nj} - t_{nj}) \phi(\mathbf{x}_n).$$

Logistic Regression for Multi-Class Classification

We again appeal to the Newton-Raphson update to obtain the corresponding Iterative Reweighted Least Squares algorithm for the multiclass problem. This requires evaluation of the Hessian matrix that comprises blocks of size $M \times M$ in which block j, k is given by

$$\nabla_{\mathbf{w}_k} \nabla_{\mathbf{w}_j} E(\mathbf{w}_1, \dots, \mathbf{w}_K) = - \sum_{n=1}^N y_{nk} (I_{kj} - y_{nj}) \phi(\mathbf{x}_n) \phi(\mathbf{x}_n)^T.$$

As with the two-class problem, the Hessian matrix for the multiclass logistic regression model is positive definite and so the error function again has a unique minimum.

References

Christopher M. Bishop. **Pattern Recognition and Machine Learning.**
New York: Springer; 2006 Aug 17.

- Linear Regression: Section 3.1-3.2
- Logistic Regression: Section 4.3

The End

COMP4131: Data Modelling and Analysis

Lecture 7: Gaussian Process Regression and Classification

Daokun Zhang

University of Nottingham Ningbo China

daokun.zhang@nottingham.edu.cn

31 March, 2025

Overview

1 Gaussian Process

2 Gaussian Process Regression

3 Gaussian Process Classification

Gaussian Process

Linear Regression

Recall the probabilistic analysis of the standard linear regression model with Gaussian noise

$$f(\mathbf{x}) = \mathbf{w}^T \phi(\mathbf{x}), \quad y = f(\mathbf{x}) + \varepsilon,$$

where ε is the additive noise that follows an i.i.d Gaussian distribution with zero mean and variance σ_n^2 :

$$\varepsilon \sim \mathcal{N}(0, \sigma_n^2).$$

Given the training data $\mathbf{X} = \{\mathbf{x}_1, \dots, \mathbf{x}_N\}$ and $\mathbf{y} = \{y_1, \dots, y_N\}$, we can use maximum likelihood estimation (MLE) to find the solution of \mathbf{w} as

$$\mathbf{w}^* = (\Phi^T \Phi)^{-1} \Phi^T \mathbf{y},$$

where $\Phi = [\phi(\mathbf{x}_1), \dots, \phi(\mathbf{x}_N)]^T$ is an $N \times M$ matrix.

Bayesian Linear Regression

In the Bayesian formalism, we can specify a prior over the parameters, expressing our beliefs about the parameters before we look at the observations. We put a Gaussian prior with mean vector μ_w and covariance matrix Σ_w on the weights

$$\mathbf{w} \sim \mathcal{N}(\mu_w, \Sigma_w).$$

Due to the randomness in \mathbf{w} , $f(\mathbf{x}) = \mathbf{w}^T \phi(\mathbf{x})$ is no longer a deterministic but a **random function**. Let f_p denote the linear function value $f(\mathbf{x}_n)$ at sample \mathbf{x}_n , then f_1, \dots, f_N follow a multivariate Gaussian distribution with

$$\mathbb{E}[f_p] = \mu_w^T \phi(\mathbf{x}_p),$$

$$\text{Cov}(f_n, f_m) = \phi(\mathbf{x}_n)^T \Sigma_w \phi(\mathbf{x}_m).$$

Now we get a **Gaussian process** $f(\mathbf{x})$ with mean function $m(\mathbf{x}) = \mu_w^T \phi(\mathbf{x})$ and covariance function $k(\mathbf{x}, \mathbf{x}') = \phi(\mathbf{x})^T \Sigma_w \phi(\mathbf{x}')$.

Gaussian Process

Definition: A Gaussian process is a collection of random variables, any finite number of which have a joint Gaussian distribution.

Implications

- “a collection of random variables” means a set that has
 - a finite number of random variables
 - an infinite number of random variables
- “any finite number of which have a joint Gaussian distribution” means:
 - multiple random variables follow a joint multivariate Gaussian distribution
 - after marginalization, a single random variable follows a univariate Gaussian distribution

Gaussian Process

A Gaussian process is completely specified by its **mean function** and **covariance function**. We define mean function $m(\mathbf{x})$ and the covariance function $k(\mathbf{x}, \mathbf{x}')$ as

$$m(\mathbf{x}) = \mathbb{E}[f(\mathbf{x})], \\ k(\mathbf{x}, \mathbf{x}') = \mathbb{E} [\{f(\mathbf{x}) - m(\mathbf{x})\} \{f(\mathbf{x}') - m(\mathbf{x}')\}] ,$$

and write the Gaussian process as

$$f(\mathbf{x}) \sim \mathcal{GP}(m(\mathbf{x}), k(\mathbf{x}, \mathbf{x}')) .$$

Usually, for notational simplicity we will take the mean function to be zero, i.e., set $m(\mathbf{x}) = 0$.

Example: For Bayesian linear regression model $f(\mathbf{x}) = \mathbf{w}^T \phi(\mathbf{x})$ with prior $\mathbf{w} \sim \mathcal{N}(\mathbf{0}, \Sigma_{\mathbf{w}})$, we have the mean and covariance function

$$m(\mathbf{x}) = \mathbb{E}[f(\mathbf{x})] = \phi(\mathbf{x})^T \mathbb{E}[\mathbf{w}] = 0, \\ k(\mathbf{x}, \mathbf{x}') = \mathbb{E}[f(\mathbf{x})f(\mathbf{x}')] = \phi(\mathbf{x})^T \mathbb{E}[\mathbf{w}\mathbf{w}^T] \phi(\mathbf{x}') = \phi(\mathbf{x})^T \Sigma_{\mathbf{w}} \phi(\mathbf{x}') .$$

Kernel Trick

The covariance function $k(\mathbf{x}, \mathbf{x}')$ can be extended to any [kernel functions](#).

For all \mathbf{x} and \mathbf{x}' in the input space \mathcal{X} , the function $k: \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}$ is called a kernel function, if there exists a feature map $\varphi: \mathcal{X} \rightarrow \mathcal{V}$ that satisfies

$$k(\mathbf{x}, \mathbf{x}') = \langle \varphi(\mathbf{x}), \varphi(\mathbf{x}') \rangle_{\mathcal{V}},$$

where \mathcal{V} is a inner product space, and $\langle \cdot, \cdot \rangle_{\mathcal{V}}$ is the inner product operation defined by \mathcal{V} .

An alternative definition can be formulated by the [positive semidefinite \(PSD\)](#) property: For any points $(\mathbf{x}_1, \dots, \mathbf{x}_n)$ in \mathcal{X} , and all choices of n real-valued coefficients (c_1, \dots, c_n) , the function $k: \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}$ is a kernel function, if

$$\sum_{i=1}^n \sum_{j=1}^n k(\mathbf{x}_i, \mathbf{x}_j) c_i c_j \geq 0.$$

The spanned matrix, $\mathbf{K} \in \mathbb{R}^n \times \mathbb{R}^n$ with its ij -th entry $K_{ij} = k(\mathbf{x}_i, \mathbf{x}_j)$, is called [Gram matrix](#).

Kernel Trick

Many kernels can be chosen for various application scenarios

- Fisher kernel
- Polynomial kernel
- Radial basis function kernel (RBF)
- String kernels
- Graph kernels

The Radial basis function kernel (RBF) is also called squared-exp or Gaussian kernel, which is formulated as

$$k_{\text{SE}}(\mathbf{x}_i, \mathbf{x}_j) = \sigma_f^2 \exp\left(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|_2^2}{2\ell^2}\right).$$

Gaussian Process Regression

Prediction with Noise-free Observations

We first consider the ideal case where the observation y can be described by a Gaussian process $f(\mathbf{x}) \sim \mathcal{GP}(0, k(\mathbf{x}, \mathbf{x}'))$ in a **noise-free** manner, i.e., $y = f(x)$. For a set of test samples with size n_* , and feature matrix \mathbf{X}_* , its function value vector \mathbf{f}_* follows a multivariate Gaussian distribution

$$\mathbf{f}_* \sim \mathcal{N}(\mathbf{0}, \mathbf{K}(\mathbf{X}_*, \mathbf{X}_*)) ,$$

where $\mathbf{K}(\cdot, \cdot)$ is the covariance matrix between two collections of input feature vectors.

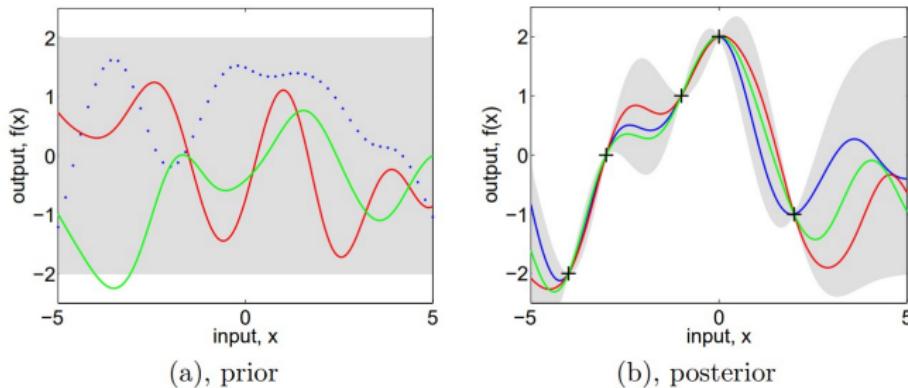
If we are given a set of training samples with size n , and feature matrix \mathbf{X} , its function value vector \mathbf{f} should follow a multivariate Gaussian distribution jointly with \mathbf{f}_*

$$\begin{bmatrix} \mathbf{f} \\ \mathbf{f}_* \end{bmatrix} \sim \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} \mathbf{K}(\mathbf{X}, \mathbf{X}) & \mathbf{K}(\mathbf{X}, \mathbf{X}_*) \\ \mathbf{K}(\mathbf{X}_*, \mathbf{X}) & \mathbf{K}(\mathbf{X}_*, \mathbf{X}_*) \end{bmatrix}\right) .$$

Prediction with Noise-free Observations

Given the observations of \mathbf{f} , the **posterior distribution** of \mathbf{f}_* is also a Gaussian distribution

$$\begin{aligned}\mathbf{f}_* | \mathbf{X}_*, \mathbf{X}, \mathbf{f} &\sim \mathcal{N} \left(\mathbf{K}(\mathbf{X}_*, \mathbf{X}) \mathbf{K}(\mathbf{X}, \mathbf{X})^{-1} \mathbf{f} \right. \\ &\quad \left. - \mathbf{K}(\mathbf{X}_*, \mathbf{X}) \mathbf{K}(\mathbf{X}, \mathbf{X})^{-1} \mathbf{K}(\mathbf{X}, \mathbf{X}_*) \right).\end{aligned}$$



The prior and posterior distributions of \mathbf{f}_* with the RBF kernel function
 $k(\mathbf{x}_p, \mathbf{x}_q) = \exp(-\frac{1}{2} \|\mathbf{x}_p - \mathbf{x}_q\|_2^2)$.

Prediction using Noisy Observations

It is typical for more realistic modelling situations that we do not have access to function values themselves, but only **noisy versions** thereof
 $y = f(\mathbf{x}) + \varepsilon$.

Assuming additive **independent identically distributed** Gaussian noise ε with zero mean and variance σ_n^2 , the prior on the noisy observations becomes another Gaussian distribution with mean $\mathbb{E}[y] = \mathbb{E}[f(\mathbf{x}) + \varepsilon] = 0$, and covariance

$$\text{cov}(y_p, y_q) = k(\mathbf{x}_p, \mathbf{x}_q) + \sigma_n^2 \delta_{pq} \text{ or } \text{cov}(\mathbf{y}) = K(\mathbf{X}, \mathbf{X}) + \sigma_n^2 \mathbf{I},$$

where δ_{pq} is a Kronecker delta symbol which is one if and only if $p = q$ and zero otherwise.

Gaussian Process Regression

We can write the joint distribution of the observed target values \mathbf{y} and the function values \mathbf{f}_* of test samples under the prior as

$$\begin{bmatrix} \mathbf{y} \\ \mathbf{f}_* \end{bmatrix} \sim \mathcal{N} \left(\mathbf{0}, \begin{bmatrix} \mathbf{K}(\mathbf{X}, \mathbf{X}) + \sigma_n^2 \mathbf{I} & \mathbf{K}(\mathbf{X}, \mathbf{X}_*) \\ \mathbf{K}(\mathbf{X}_*, \mathbf{X}) & \mathbf{K}(\mathbf{X}_*, \mathbf{X}_*) \end{bmatrix} \right).$$

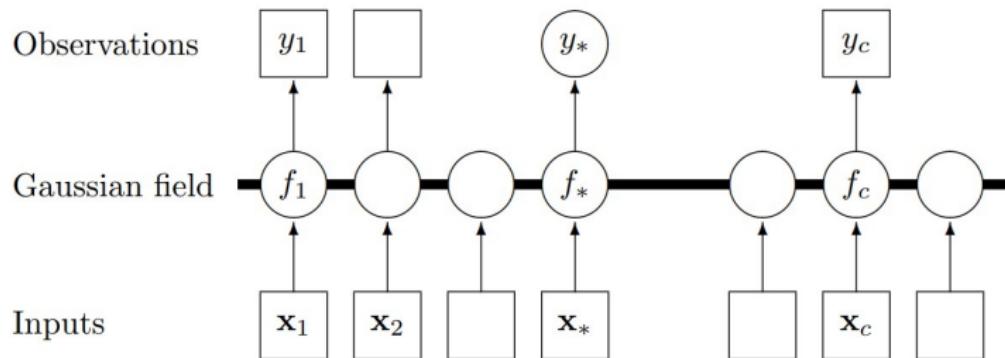
Deriving the conditional distribution, we arrive at the **key predictive equations** for Gaussian process regression

$\mathbf{f}_* | \mathbf{X}, \mathbf{y}, \mathbf{X}_* \sim \mathcal{N} (\bar{\mathbf{f}}_*, \text{cov}(\mathbf{f}_*))$, where

$$\bar{\mathbf{f}}_* \triangleq \mathbb{E} [\mathbf{f}_* | \mathbf{X}, \mathbf{y}, \mathbf{X}_*] = \mathbf{K}(\mathbf{X}_*, \mathbf{X}) [\mathbf{K}(\mathbf{X}, \mathbf{X}) + \sigma_n^2 \mathbf{I}]^{-1} \mathbf{y},$$

$$\text{cov}(\mathbf{f}_*) = \mathbf{K}(\mathbf{X}_*, \mathbf{X}_*) - \mathbf{K}(\mathbf{X}_*, \mathbf{X}) [\mathbf{K}(\mathbf{X}, \mathbf{X}) + \sigma_n^2 \mathbf{I}]^{-1} \mathbf{K}(\mathbf{X}, \mathbf{X}_*).$$

Gaussian Process Regression



Graphical model (chain graph) for a Gaussian process for regression.
Squares represent **observed variables** and circles represent **unknowns**.

Gaussian Process Regression

By using a compact form of the notation setting $\mathbf{K} = \mathbf{K}(\mathbf{X}, \mathbf{X})$ and $\mathbf{k}_* = \mathbf{K}(\mathbf{X}, \mathbf{x}_*)$ for a test sample \mathbf{x}_* , the prediction f_* for the single test sample \mathbf{x}_* has mean \bar{f}_* and variance $\mathbb{V}[f_*]$ as

$$\begin{aligned}\bar{f}_* &= \mathbf{k}_*^\top (\mathbf{K} + \sigma_n^2 \mathbf{I})^{-1} \mathbf{y}, \\ \mathbb{V}[f_*] &= k(\mathbf{x}_*, \mathbf{x}_*) - \mathbf{k}_*^\top (\mathbf{K} + \sigma_n^2 \mathbf{I})^{-1} \mathbf{k}_*.\end{aligned}$$

Another way to look at the expression of the mean \bar{f}_* is to see it as a **linear combination** of N kernel functions, each one centered on a training point, by writing

$$\bar{f}_* = \sum_{i=1}^N \alpha_i k(\mathbf{x}_i, \mathbf{x}_*),$$

where $\boldsymbol{\alpha} = (\mathbf{K} + \sigma_n^2 \mathbf{I})^{-1} \mathbf{y}$.

Gaussian Process Regression

input: X (inputs), \mathbf{y} (targets), k (covariance function), σ_n^2 (noise level),
 \mathbf{x}_* (test input)

- 2: $L := \text{cholesky}(K + \sigma_n^2 I)$
 - 3: $\boldsymbol{\alpha} := L^\top \backslash (L \backslash \mathbf{y})$
 - 4: $\bar{f}_* := \mathbf{k}_*^\top \boldsymbol{\alpha}$
 - 5: $\mathbf{v} := L \backslash \mathbf{k}_*$
 - 6: $\mathbb{V}[f_*] := k(\mathbf{x}_*, \mathbf{x}_*) - \mathbf{v}^\top \mathbf{v}$
 - 7: $\log p(\mathbf{y}|X) := -\frac{1}{2}\mathbf{y}^\top \boldsymbol{\alpha} - \sum_i \log L_{ii} - \frac{n}{2} \log 2\pi$
 - 8: **return:** \bar{f}_* (mean), $\mathbb{V}[f_*]$ (variance), $\log p(\mathbf{y}|X)$ (log marginal likelihood)
- } predictive mean
} predictive variance

Algorithm for Gaussian process regression, where the predictions (the target value's mean and variance) at the test sample \mathbf{x}_* and the **log marginal likelihood** on training set are returned.

Varying the Hyperparameters

Setting the RBF kernel $k(\mathbf{x}_p, \mathbf{x}_q) = \sigma_f^2 \exp(-\frac{1}{2\ell^2} \|\mathbf{x}_p - \mathbf{x}_q\|_2^2)$ as the covariance function, the covariance between the observations y_p and y_q becomes

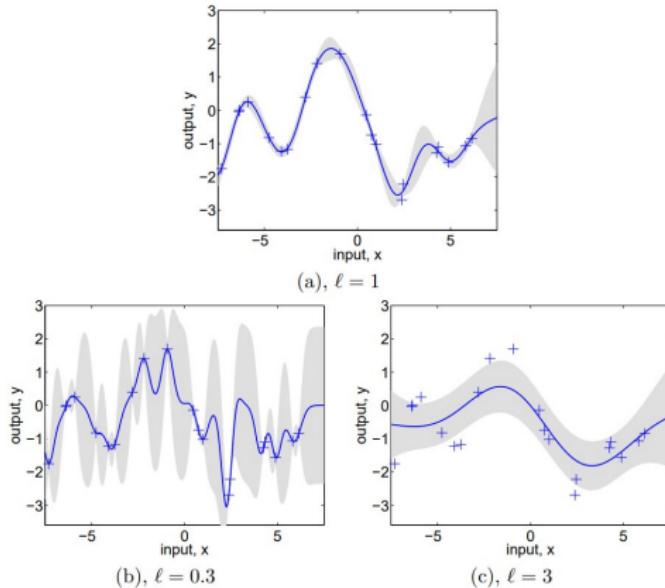
$$\text{cov}(y_p, y_q) = \sigma_f^2 \exp(-\frac{1}{2\ell^2} \|\mathbf{x}_p - \mathbf{x}_q\|_2^2) + \sigma_n^2 \delta_{pq},$$

which implies that the model relies on three hyperparameters:

- signal variance σ_f^2
- noise variance σ_n^2
- length-scale ℓ

The log marginal likelihood can be used to tune the hyperparameters.

Varying the Hyperparameters



Small length-scale $\ell = 0.3 \rightarrow$ a sharply varying function f with small noise.
Large length-scale $\ell = 3 \rightarrow$ a slowly varying function f with large noise.
Moderate length-scale $\ell = 1 \rightarrow$ exact fitting of the underlying function f .

Gaussian Process Classification

Binary Gaussian Process Classifier

For binary classification with label candidate set $\{+1, -1\}$, linear regression uses the **sigmoid** function to squash the linear regression output into the label probability $p(y = +1|\mathbf{x}) = \sigma(\mathbf{w}^T \phi(\mathbf{x}))$.

In Gaussian process, the latent function value $f(\mathbf{x})$ is used to construct the linear regression output.

Similarly, for **Gaussian process classification**, we can also construct the label probability via the sigmoid function

$$p(y = +1|f(\mathbf{x})) = \sigma(f(\mathbf{x})).$$

Different from $\mathbf{w}^T \phi(\mathbf{x})$, $f(\mathbf{x})$ is Gaussian random variable, we need to marginalize it to get the posterior probability

$$p(y = +1|\mathbf{x}) = \int p(y = +1|f)p(f|\mathbf{x})df.$$

Binary Gaussian Process Classifier

Given a training set $\{\mathbf{X}, \mathbf{y}\}$, for a new test sample \mathbf{x}_* , Gaussian process classifier predicts its label by estimating the posterior probability

$$\begin{aligned}\bar{\pi}_* &\triangleq p(y_* = +1 | \mathbf{X}, \mathbf{y}, \mathbf{x}_*) = \int p(y_* = +1 | f_*) p(f_* | \mathbf{X}, \mathbf{y}, \mathbf{x}_*) df_* \\ &= \int \sigma(f_*) p(f_* | \mathbf{X}, \mathbf{y}, \mathbf{x}_*) df_*.\end{aligned}$$

To achieve the estimation, we need first calculate the posterior distribution of the latent variable f_* corresponding to the test sample \mathbf{x}_*

$$p(f_* | \mathbf{X}, \mathbf{y}, \mathbf{x}_*) = \int p(f_* | \mathbf{X}, \mathbf{x}_*, \mathbf{f}) p(\mathbf{f} | \mathbf{X}, \mathbf{y}) d\mathbf{f},$$

where $p(\mathbf{f} | \mathbf{X}, \mathbf{y}) = p(\mathbf{y} | \mathbf{f}) p(\mathbf{f} | \mathbf{X}) / p(\mathbf{y} | \mathbf{X})$ is the posterior over the latent variables and $p(\mathbf{y} | \mathbf{f})$ is computed by applying the elementwise sigmoid function to \mathbf{f} .

Laplace Approximation for Binary GP Classifier

In the regression case, \mathbf{y} and f_* jointly follow a multivariate Gaussian distribution, the conditional probability distribution $p(f_*|\mathbf{X}, \mathbf{y}, \mathbf{x}_*)$ can be computed analytically.

However, in the classification case, due to the computational intractability of the distribution $p(\mathbf{f}|\mathbf{X}, \mathbf{y})$, the posterior distribution $p(f_*|\mathbf{X}, \mathbf{y}, \mathbf{x}_*)$ cannot be computed analytically with the intractable integral.

To solve this problem, we resort to the Laplace approximation that approximates the distribution $p(\mathbf{f}|\mathbf{X}, \mathbf{y})$ with a Gaussian distribution $q(\mathbf{f}|\mathbf{X}, \mathbf{y})$, obtained by doing a second order Taylor expansion of $\log p(\mathbf{f}|\mathbf{X}, \mathbf{y})$ around the maximum of the posterior

$$q(\mathbf{f}|\mathbf{X}, \mathbf{y}) = \mathcal{N}\left(\mathbf{f}|\hat{\mathbf{f}}, \mathbf{A}^{-1}\right) \propto \exp\left(-\frac{1}{2}(\mathbf{f} - \hat{\mathbf{f}})^\top \mathbf{A}(\mathbf{f} - \hat{\mathbf{f}})\right),$$

where $\hat{\mathbf{f}} = \operatorname{argmax}_{\mathbf{f}} p(\mathbf{f}|\mathbf{X}, \mathbf{y})$ and $\mathbf{A} = -\nabla\nabla \log p(\mathbf{f}|\mathbf{X}, \mathbf{y})|_{\mathbf{f}=\hat{\mathbf{f}}}$ is the Hessian matrix of the negative log posterior at that point.

Laplace Approximation for Binary GP Classifier

By Bayes' rule, the posterior over the latent variables $p(\mathbf{f}|\mathbf{X}, \mathbf{y})$ is given by

$$p(\mathbf{f}|\mathbf{X}, \mathbf{y}) = \frac{p(\mathbf{y}, \mathbf{f}|\mathbf{X})}{p(\mathbf{y}|\mathbf{X})} = \frac{p(\mathbf{y}|\mathbf{f})p(\mathbf{f}|\mathbf{X})}{p(\mathbf{y}|\mathbf{X})},$$

but $p(\mathbf{y}|\mathbf{X})$ is independent of \mathbf{f} , we only need to consider the un-normalized posterior $p(\mathbf{y}, \mathbf{f}|\mathbf{X}) = p(\mathbf{y}|\mathbf{f})p(\mathbf{f}|\mathbf{X})$ when maximizing w.r.t. \mathbf{f} .

Taking the logarithm on the un-normalized posterior, we get

$$\begin{aligned}\Psi(\mathbf{f}) &\triangleq \log p(\mathbf{y}|\mathbf{f}) + \log p(\mathbf{f}|\mathbf{X}) \\ &= \log p(\mathbf{y}|\mathbf{f}) - \frac{1}{2}\mathbf{f}^T \mathbf{K}^{-1} \mathbf{f} - \frac{1}{2} \log |\mathbf{K}| - \frac{n}{2} \log 2\pi,\end{aligned}$$

where $p(\mathbf{f}|\mathbf{X})$ is a Gaussian distribution $\mathcal{N}(\mathbf{0}, \mathbf{K})$, \mathbf{K} is short for the kernel matrix $\mathbf{K}(\mathbf{X}, \mathbf{X})$, and $|\mathbf{K}|$ is the determinant of the kernel matrix \mathbf{K} .

Laplace Approximation for Binary GP Classifier

Differentiating $\Psi(\mathbf{f})$ w.r.t. \mathbf{f} , we obtain

$$\nabla \Psi(\mathbf{f}) = \nabla \log p(\mathbf{y}|\mathbf{f}) - \mathbf{K}^{-1}\mathbf{f},$$

$$\nabla \nabla \Psi(\mathbf{f}) = \nabla \nabla \log p(\mathbf{y}|\mathbf{f}) - \mathbf{K}^{-1},$$

where $\nabla \nabla \log p(\mathbf{y}|\mathbf{f})$ is diagonal, since the distribution for $y_i = +1$ or -1 depends only on f_i , not on $f_{j \neq i}$.

We can apply the [Newton-Raphson](#) method to find the latent function value vector $\hat{\mathbf{f}}$ that maximizes $\Psi(\mathbf{f})$, and obtain the approximated probability distribution as

$$q(\mathbf{f}|\mathbf{X}, \mathbf{y}) = \mathcal{N}\left(\hat{\mathbf{f}}, (\mathbf{K}^{-1} + \mathbf{W})^{-1}\right),$$

where $\mathbf{W} \triangleq -\nabla \nabla \log p(\mathbf{y}|\mathbf{f})|_{\mathbf{f}=\hat{\mathbf{f}}}$.

Laplace Approximation for Binary GP Classifier

With $q(\mathbf{f}|\mathbf{X}, \mathbf{y})$ approximating $p(\mathbf{f}|\mathbf{X}, \mathbf{y})$, we can get the approximation for the posterior $p(f_*|\mathbf{X}, \mathbf{y}, \mathbf{x}_*)$ as

$$q(f_*|\mathbf{X}, \mathbf{y}, \mathbf{x}_*) = \int p(f_*|\mathbf{X}, \mathbf{x}_*, \mathbf{f}) q(\mathbf{f}|\mathbf{X}, \mathbf{y}) d\mathbf{f},$$

where both $p(f_*|\mathbf{X}, \mathbf{x}_*, \mathbf{f})$ and $q(\mathbf{f}|\mathbf{X}, \mathbf{y})$ are Gaussian distributions. It can be proved that $q(f_*|\mathbf{X}, \mathbf{y}, \mathbf{x}_*)$ is also a Gaussian distribution with mean and variance as

$$\mathbb{E}_q [f_*|\mathbf{X}, \mathbf{y}, \mathbf{x}_*] = \mathbf{k}_*^T \mathbf{K}^{-1} \hat{\mathbf{f}},$$

$$\mathbb{V}_q [f_*|\mathbf{X}, \mathbf{y}, \mathbf{x}_*] = k(\mathbf{x}_*, \mathbf{x}_*) - \mathbf{k}_*^T (\mathbf{K} + \mathbf{W}^{-1})^{-1} \mathbf{k}_*.$$

Finally, the label probability for the test sample \mathbf{x}_* can be computed as

$$\bar{\pi}_* \triangleq p(y_* = +1|\mathbf{X}, \mathbf{y}, \mathbf{x}_*) := \int \sigma(f_*) q(f_*|\mathbf{X}, \mathbf{y}, \mathbf{x}_*) df_*.$$

Numeric integration techniques are generally used to calculate the one-dimensional integral.

Binary Gaussian Process Classifier

```
input:  $K$  (covariance matrix),  $\mathbf{y}$  ( $\pm 1$  targets),  $p(\mathbf{y}|\mathbf{f})$  (likelihood function)
2:  $\mathbf{f} := \mathbf{0}$                                      initialization
   repeat
     4:  $W := -\nabla\nabla \log p(\mathbf{y}|\mathbf{f})$ 
         $L := \text{cholesky}(I + W^{\frac{1}{2}} K W^{\frac{1}{2}})$ 
     6:  $\mathbf{b} := W\mathbf{f} + \nabla \log p(\mathbf{y}|\mathbf{f})$ 
         $\mathbf{a} := \mathbf{b} - W^{\frac{1}{2}} L^\top \backslash (L \backslash (W^{\frac{1}{2}} K \mathbf{b}))$ 
     8:  $\mathbf{f} := K\mathbf{a}$ 
   until convergence
10:  $\log q(\mathbf{y}|X, \theta) := -\frac{1}{2}\mathbf{a}^\top \mathbf{f} + \log p(\mathbf{y}|\mathbf{f}) - \sum_i \log L_{ii}$ 
return:  $\mathbf{f} := \mathbf{f}$  (post. mode),  $\log q(\mathbf{y}|X, \theta)$  (approx. log marg. likelihood)
```

Training algorithm for the binary Gaussian process classifier with Laplace approximation. The mode $\hat{\mathbf{f}}$ for the posterior probability distribution $p(\mathbf{f}|\mathbf{X}, \mathbf{y})$ is returned, as well as the **approximated log marginal likelihood**, which can be used to tune hyperparameters.

Binary Gaussian Process Classifier

input: $\hat{\mathbf{f}}$ (mode), X (inputs), \mathbf{y} (± 1 targets), k (covariance function),
 $p(\mathbf{y}|\mathbf{f})$ (likelihood function), \mathbf{x}_* test input

- 2: $W := -\nabla \nabla \log p(\mathbf{y}|\hat{\mathbf{f}})$
 $L := \text{cholesky}(I + W^{\frac{1}{2}} K W^{\frac{1}{2}})$
- 4: $\bar{f}_* := \mathbf{k}(\mathbf{x}_*)^\top \nabla \log p(\mathbf{y}|\hat{\mathbf{f}})$
 $\mathbf{v} := L \backslash (W^{\frac{1}{2}} \mathbf{k}(\mathbf{x}_*))$
- 6: $\mathbb{V}[f_*] := k(\mathbf{x}_*, \mathbf{x}_*) - \mathbf{v}^\top \mathbf{v}$
 $\bar{\pi}_* := \int \sigma(z) \mathcal{N}(z|\bar{f}_*, \mathbb{V}[f_*]) dz$
- 8: **return:** $\bar{\pi}_*$ (predictive class probability (for class 1))

Prediction algorithm for the binary Gaussian classifier with Laplace approximation.

Multi-Class Gaussian Process Classifier

For the multi-class classification problem, where the class label y belongs to the label candidate set $\{1, \dots, C\}$ with size C , we need to construct a Gaussian process for each class $c \in \{1, \dots, C\}$, with the latent function value $f^c(\mathbf{x})$ evaluated at \mathbf{x} .

The conditional label probability $p(y = c | f^1(\mathbf{x}), \dots, f^C(\mathbf{x}))$ can be constructed by the softmax function

$$p(y = c | f^1(\mathbf{x}), \dots, f^C(\mathbf{x})) = \frac{\exp(f^c(\mathbf{x}))}{\sum_{c'=1}^C \exp(f^{c'}(\mathbf{x}))}.$$

Following the same procedure as the binary classification case, given a test sample \mathbf{x}_* , we can estimate the posterior probabilities $p(y_* = c | \mathbf{X}, \mathbf{y}, \mathbf{x}_*)$ to predict \mathbf{x}_* 's class label.

References

Williams CK, Rasmussen CE. **Gaussian Processes for Machine Learning**. Cambridge, MA: MIT press; 2006.

- Section 1 - Section 3

The End

COMP4131: Data Modelling and Analysis

Lecture 8: Naive Bayes and K Nearest Neighbors

Daokun Zhang

University of Nottingham Ningbo China

daokun.zhang@nottingham.edu.cn

April 7, 2025

Overview

1 Naive Bayes Classifiers

2 K Nearest Neighbors Classifier

Naive Bayes Classifiers

Naive Bayes Classification

Suppose we have a set of training samples $\mathcal{T}_r = \{(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_n, y_n)\}$, where each sample is described by a m -dimensional feature vector $\mathbf{x} = (x_1, x_2, \dots, x_m)$, and a categorical label $y \in \mathcal{Y} = \{1, 2, \dots, C\}$, with C being the class number.

For a new test sample with feature vector \mathbf{x}^* , we can predict its class label as

$$y^* = \arg \max_{y \in \mathcal{Y}} \mathbb{P}(y | \mathbf{x}^*),$$

where

$$\mathbb{P}(y | \mathbf{x}^*) = \frac{\mathbb{P}(\mathbf{x}^*, y)}{\mathbb{P}(\mathbf{x}^*)} = \frac{\mathbb{P}(\mathbf{x}^* | y) \mathbb{P}(y)}{\mathbb{P}(\mathbf{x}^*)}.$$

As $\mathbb{P}(\mathbf{x}^*)$ is a constant with regard to y , then we have

$$y^* = \arg \max_{y \in \mathcal{Y}} \mathbb{P}(\mathbf{x}^*, y) = \arg \max_{y \in \mathcal{Y}} \mathbb{P}(\mathbf{x}^* | y) \mathbb{P}(y).$$

Naive Bayes Classification

The task becomes to estimate the conditional probability $\mathbb{P}(\mathbf{x}^*|y)$ and prior probability $\mathbb{P}(y)$.

The prior probability can be estimated by counting the number of training samples having the targeting class labels

$$\mathbb{P}(y = c) = \frac{|\{(\mathbf{x}, y) \in \mathcal{T}_r : y = c\}|}{n},$$

where $|\cdot|$ is the size of a set and $\{(\mathbf{x}, y) \in \mathcal{T}_r : y = c\}$ is the set of samples with class label $y = c$.

To calculate the conditional probability $\mathbb{P}(\mathbf{x}^*|y)$, we can leverage the conditional independence assumption

$$\mathbb{P}(\mathbf{x}^*|y) = \mathbb{P}(x_1^*|y) \cdot \mathbb{P}(x_2^*|y) \cdots \mathbb{P}(x_m^*|y),$$

where $\mathbb{P}(x_j^*|y)$ with $1 \leq j \leq m$ is the occurrence probability of feature value x_j^* conditioned on the class label y .

Naive Bayes Classification

For different data types, $\mathbb{P}(x_j^*|y)$ can be estimated in different ways.

For categorical data, where x_j^* takes one of the pre-defined values $\{f_1, f_2 \dots, f_{n_j}\}$, we assume $x_j^*|y = c$ follows a Categorical distribution

$$x_j^*|y = c \sim \mathbf{Cat}(n_j, \mathbf{p}),$$

where $\mathbf{p} = (p_1, p_2, \dots, p_{n_j})$ with $p_k = \mathbb{P}(x_j^* = f_k|y = c)$ and $\sum_{k=1}^{n_j} p_k = 1$. That is to say

$$\mathbb{P}(x_j^* = f_k|y = c) = \prod_{k=1}^{n_j} p_k^{\delta(x_j^*, f_k)},$$

where $\delta(x_j^*, f_k)$ is a Kronecker delta function, whose value is equal to 1 if $x_j^* = f_k$ and 0 otherwise.

Naive Bayes Classification

The parameter p_k ($1 \leq k \leq n_j$) can be estimated by the maximum likelihood estimation (MLE) method, for which the log likelihood is formulated as

$$\begin{aligned}\log L(\mathbf{p}) &= \log \prod_{(\mathbf{x},y) \in \mathcal{T}_r : y=c} \mathbb{P}(x_j | y=c) \\ &= \log \prod_{(\mathbf{x},y) \in \mathcal{T}_r : y=c} \left\{ \prod_{s=1}^{n_j} p_s^{\delta(x_j, f_s)} \right\} \\ &= \sum_{(\mathbf{x},y) \in \mathcal{T}_r : y=c} \left\{ \sum_{s=1}^{n_j} \delta(x_j, f_s) \log p_s \right\}\end{aligned}$$

subject to the constraint

$$\sum_{s=1}^{n_j} p_s = 1.$$

Naive Bayes Classification

The Lagrangian function with the constraint than has the following form

$$\begin{aligned}\mathcal{L}(\mathbf{p}) &= \log L(\mathbf{p}) + \lambda \left(\sum_{s=1}^{n_j} p_s - 1 \right) \\ &= \sum_{(\mathbf{x}, y) \in \mathcal{T}_r : y=c} \left\{ \sum_{s=1}^{n_j} \delta(x_j, f_s) \log p_s \right\} + \lambda \left(\sum_{s=1}^{n_j} p_s - 1 \right).\end{aligned}$$

The derivative of $\mathcal{L}(\mathbf{p})$ w.r.t p_k is

$$\frac{\partial \mathcal{L}(\mathbf{p})}{\partial p_k} = \sum_{(\mathbf{x}, y) \in \mathcal{T}_r : y=c} \frac{\delta(x_j, f_k)}{p_k} + \lambda.$$

Set the derivative to zero, we have

$$p_k = - \sum_{(\mathbf{x}, y) \in \mathcal{T}_r : y=c} \frac{\delta(x_j, f_k)}{\lambda}.$$

Naive Bayes Classification

Using the property $\sum_{s=1}^{n_j} p_s = 1$,

$$\lambda = - \sum_{s=1}^{n_j} \sum_{(\mathbf{x}, y) \in \mathcal{T}_r : y=c} \delta(x_j, f_s).$$

Then

$$p_k = \frac{\sum_{(\mathbf{x}, y) \in \mathcal{T}_r : y=c} \delta(x_j, f_k)}{\sum_{s=1}^{n_j} \sum_{(\mathbf{x}, y) \in \mathcal{T}_r : y=c} \delta(x_j, f_s)}$$
$$= \frac{|\{(\mathbf{x}, y) \in \mathcal{T}_r : y=c, x_j = f_k\}|}{|\{(\mathbf{x}, y) \in \mathcal{T}_r : y=c\}|}.$$

Naive Bayes Classification

Then we can get the estimation for the probability $\mathbb{P}(x_j = f_k | y = c)$ as

$$\mathbb{P}(x_j = f_k | y = c) = \frac{|\{(x, y) \in \mathcal{T}_r : y = c, x_j = f_k\}|}{|\{(x, y) \in \mathcal{T}_r : y = c\}|},$$

and the prior probability $\mathbb{P}(y = c)$ as

$$\mathbb{P}(y = c) = \frac{|\{(x, y) \in \mathcal{T}_r : y = c\}|}{n}.$$

$|\{(x, y) \in \mathcal{T}_r : y = c, x_j = f_k\}|$ is the number of training samples with label y taking value c and attribute x_j taking value f_k , which can also be represented as **Count**($y = c, x_j = f_k$).

$|\{(x, y) \in \mathcal{T}_r : y = c\}|$ is the number of training samples with label y taking value c , which can also be represented as **Count**($y = c$).

Naive Bayes Classification

To avoid the case that $\mathbb{P}(x_j = f_k | y = c) = 0$ with the number of supported training samples equal 0, we adopt the Laplace smoothing to adjust the probability estimation

$$\mathbb{P}(x_j = f_k | y = c) = \frac{|\{(x, y) \in \mathcal{T}_r : y = c, x_j = f_k\}| + \alpha}{|\{(x, y) \in \mathcal{T}_r : y = c\}| + n_j \alpha}.$$

where the “pseudocount” $\alpha > 0$ is the smoothing parameter.

An Alternative Way to Parameter Estimation

It looks like the probability $\mathbb{P}(x_j = f_k | y = c)$ can be estimated in a more straightforward way

1. $|\{(x, y) \in \mathcal{T}_r : y = c, x_j = f_k\}|$ is the occurrence times of the event $\{y = c, x_j = f_k\}$ in training data
2. $|\{(x, y) \in \mathcal{T}_r : y = c\}|$ is the occurrence times of the event $\{y = c\}$ in training data
3. According to the [classical definition of probability](#), we can directly calculate the (unsmoothed) probability as

$$\mathbb{P}(x_j = f_k | y = c) = \frac{|\{(x, y) \in \mathcal{T}_r : y = c, x_j = f_k\}|}{|\{(x, y) \in \mathcal{T}_r : y = c\}|}.$$

So, why shall we make an assumption on the distribution of attribute values and use MLE to estimate the distribution parameters?

According to the difference in data type, the Categorical Naive Bayes can be extended into the following algorithm variants by **modeling the distribution of attribute values**

- **Gaussian Naive Bayes**

- x_j takes continuous values
- assume $x_j|y$ follows a Gaussian distribution

- **Bernoulli Naive Bayes**

- x_j takes binary values $\{0, 1\}$, indicating the occurrence status of a word in a text sample
- assume $x_j|y$ follows a Bernoulli distribution

- **Multinomial Naive Bayes**

- x_j takes non-negative integer values $\{0, 1, 2, 3, \dots\}$, indicating the occurrence times of a word in a text sample
- assume $x_1, x_2, \dots, x_m|y$ jointly follow a Multinomial distribution

Categorical Naive Bayes: Example

Suppose we are given a set of training samples, where each sample includes the weather condition record of a day as attributes, and the decision to play tennis or not as a binary class label. Please use [Categorical Naive Bayes](#) to predict whether we shall play tennis or not for a new day.

PlayTennis: training examples

Day	Outlook	Temperature	Humidity	Wind	PlayTennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

Categorical Naive Bayes: Example

From the collected training data, we can first summarize the categorical value set for each attribute/label as

- **Attribute:** Outlook $\in \{\text{Sunny, Rain, Overcast}\}$
- **Attribute:** Temperature $\in \{\text{Hot, Cool, Mild}\}$
- **Attribute:** Humidity $\in \{\text{High, Normal}\}$
- **Attribute:** Wind $\in \{\text{Strong, Weak}\}$
- **Label:** PlayTennis $\in \{\text{Yes, No}\}$

Categorical Naive Bayes: Example

We can count the occurrence times of the interested events as

- **Count(PlayTennis = Yes) = ?**
- **Count(PlayTennis = No) = ?**
- **Count(Outlook = Sunny, PlayTennis = Yes) = ?**
- **Count(Outlook = Rain, PlayTennis = Yes) = ?**
- **Count(Outlook = Overcast, PlayTennis = Yes) = ?**
- **Count(Outlook = Sunny, PlayTennis = No) = ?**
- **Count(Outlook = Rain, PlayTennis = No) = ?**
- **Count(Outlook = Overcast, PlayTennis = No) = ?**
- **Count(Temperature = Hot, PlayTennis = Yes) = ?**
- **Count(Temperature = Cool, PlayTennis = Yes) = ?**
- **Count(Temperature = Mild, PlayTennis = Yes) = ?**

Categorical Naive Bayes: Example

- **Count**(Temperature = Hot, PlayTennis = No) = ?
- **Count**(Temperature = Cool, PlayTennis = No) = ?
- **Count**(Temperature = Mild, PlayTennis = No) = ?
- **Count**(Humidity = High, PlayTennis = Yes) = ?
- **Count**(Humidity = Normal, PlayTennis = Yes) = ?
- **Count**(Humidity = High, PlayTennis = No) = ?
- **Count**(Humidity = Normal, PlayTennis = No) = ?
- **Count**(Wind = Strong, PlayTennis = Yes) = ?
- **Count**(Wind = Weak, PlayTennis = Yes) = ?
- **Count**(Wind = Strong, PlayTennis = No) = ?
- **Count**(Wind = Weak, PlayTennis = No) = ?

Categorical Naive Bayes: Example

We can calculate the interested probability value as (without the use of Laplace smoothing)

- $\mathbb{P}(\text{PlayTennis} = \text{Yes}) = ?$
- $\mathbb{P}(\text{PlayTennis} = \text{No}) = ?$
- $\mathbb{P}(\text{Outlook} = \text{Sunny} \mid \text{PlayTennis} = \text{Yes}) = ?$
- $\mathbb{P}(\text{Outlook} = \text{Rain} \mid \text{PlayTennis} = \text{Yes}) = ?$
- $\mathbb{P}(\text{Outlook} = \text{Overcast} \mid \text{PlayTennis} = \text{Yes}) = ?$
- $\mathbb{P}(\text{Outlook} = \text{Sunny} \mid \text{PlayTennis} = \text{No}) = ?$
- $\mathbb{P}(\text{Outlook} = \text{Rain} \mid \text{PlayTennis} = \text{No}) = ?$
- $\mathbb{P}(\text{Outlook} = \text{Overcast} \mid \text{PlayTennis} = \text{No}) = ?$
- $\mathbb{P}(\text{Temperature} = \text{Hot} \mid \text{PlayTennis} = \text{Yes}) = ?$
- $\mathbb{P}(\text{Temperature} = \text{Cool} \mid \text{PlayTennis} = \text{Yes}) = ?$
- $\mathbb{P}(\text{Temperature} = \text{Mild} \mid \text{PlayTennis} = \text{Yes}) = ?$

Categorical Naive Bayes: Example

- $\mathbb{P}(\text{Temperature} = \text{Hot} \mid \text{PlayTennis} = \text{No}) = ?$
- $\mathbb{P}(\text{Temperature} = \text{Cool} \mid \text{PlayTennis} = \text{No}) = ?$
- $\mathbb{P}(\text{Temperature} = \text{Mild} \mid \text{PlayTennis} = \text{No}) = ?$
- $\mathbb{P}(\text{Humidity} = \text{High} \mid \text{PlayTennis} = \text{Yes}) = ?$
- $\mathbb{P}(\text{Humidity} = \text{Normal} \mid \text{PlayTennis} = \text{Yes}) = ?$
- $\mathbb{P}(\text{Humidity} = \text{High} \mid \text{PlayTennis} = \text{No}) = ?$
- $\mathbb{P}(\text{Humidity} = \text{Normal} \mid \text{PlayTennis} = \text{No}) = ?$
- $\mathbb{P}(\text{Wind} = \text{Strong} \mid \text{PlayTennis} = \text{Yes}) = ?$
- $\mathbb{P}(\text{Wind} = \text{Weak} \mid \text{PlayTennis} = \text{Yes}) = ?$
- $\mathbb{P}(\text{Wind} = \text{Strong} \mid \text{PlayTennis} = \text{No}) = ?$
- $\mathbb{P}(\text{Wind} = \text{Weak} \mid \text{PlayTennis} = \text{No}) = ?$

Categorical Naive Bayes: Example

Make prediction for a new day

$$\mathbf{x}^* = (\text{Outlook} = \text{Sunny}, \text{Temperature} = \text{Cool}, \text{Humidity} = \text{High}, \text{Wind} = \text{Strong})$$

We can calculate the joint probability $\mathbb{P}(\mathbf{x}^*, \text{PlayTennis})$ as

$$\begin{aligned}\mathbb{P}(\mathbf{x}^*, \text{PlayTennis} = \text{Yes}) &= \mathbb{P}(\text{Outlook} = \text{Sunny} \mid \text{PlayTennis} = \text{Yes}) \cdot \\ &\quad \mathbb{P}(\text{Temperature} = \text{Cool} \mid \text{PlayTennis} = \text{Yes}) \cdot \\ &\quad \mathbb{P}(\text{Humidity} = \text{High} \mid \text{PlayTennis} = \text{Yes}) \cdot \\ &\quad \mathbb{P}(\text{Wind} = \text{Strong} \mid \text{PlayTennis} = \text{Yes}) \cdot \\ &\quad \mathbb{P}(\text{PlayTennis} = \text{Yes}) = ?\end{aligned}$$

$$\begin{aligned}\mathbb{P}(\mathbf{x}^*, \text{PlayTennis} = \text{No}) &= \mathbb{P}(\text{Outlook} = \text{Sunny} \mid \text{PlayTennis} = \text{No}) \cdot \\ &\quad \mathbb{P}(\text{Temperature} = \text{Cool} \mid \text{PlayTennis} = \text{No}) \cdot \\ &\quad \mathbb{P}(\text{Humidity} = \text{High} \mid \text{PlayTennis} = \text{No}) \cdot \\ &\quad \mathbb{P}(\text{Wind} = \text{Strong} \mid \text{PlayTennis} = \text{No}) \cdot \\ &\quad \mathbb{P}(\text{PlayTennis} = \text{No}) = ?\end{aligned}$$

By comparing $\mathbb{P}(\mathbf{x}^*, \text{PlayTennis} = \text{Yes})$ and $\mathbb{P}(\mathbf{x}^*, \text{PlayTennis} = \text{No})$, the label of \mathbf{x}^* is predicted as "PlayTennis = ?".

Pros of Naive Bayes

- Simple and easy to implement
- Fast and scalable
- Works well with high-dimensional data
- Handles missing data well
- Robust to irrelevant features
- Effective for text classification
- Suitable for multi-class problems
- Provides probabilistic outputs
- Low memory consumption
- Good baseline classifier
- Suitable for binary and categorical data

Cons of Naive Bayes

- Assumes independence between features
- Can be overly simplistic
- Relatively poor performance with continuous data
- Sensitive to zero frequency problem
- Limited expressiveness for complex problems
- Not ideal for large feature spaces with sparse data
- Requires large amounts of data for accurate probabilities
- Struggles with highly imbalanced data

K Nearest Neighbors Classifier

K Nearest Neighbors Classifier

K Nearest Neighbors (KNN) classifier is a non-parametric/lazy classifier, which does not require to train a parametric model from the training data.

KNN classifier classifies unlabeled examples by assigning them the class of the most similar labeled examples.

Formally, given a set of training samples $\mathcal{T}_r = \{(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_n, y_n)\}$, where each sample is described by a m -dimensional continuous feature vector \mathbf{x} and a class label y . For a new test example \mathbf{x}^* , KNN determines its label probability as

$$\mathbb{P}(y^* = c | \mathbf{x}^*) = \frac{|\{(\mathbf{x}, y) \in \mathcal{T}_r : y = c, d(\mathbf{x}^*, \mathbf{x}) \text{ ranks smallest-}K\}|}{K},$$

where $d(\mathbf{x}^*, \mathbf{x})$ is the distance between \mathbf{x}^* and \mathbf{x} .

Distance Metric

For the two samples \mathbf{x}^* and \mathbf{x} , there are various choices to measure their distance

- Euclidean distance

$$d(\mathbf{x}^*, \mathbf{x}) = \|\mathbf{x}^* - \mathbf{x}\|_2 = \sqrt{\sum_{j=1}^m (x_j^* - x_j)^2}$$

- Manhattan distance

$$d(\mathbf{x}^*, \mathbf{x}) = \|\mathbf{x}^* - \mathbf{x}\|_1 = \sum_{j=1}^m |x_j^* - x_j|$$

- Cosine distance

$$d(\mathbf{x}^*, \mathbf{x}) = 1 - \cos(\mathbf{x}^*, \mathbf{x}) = 1 - \frac{\sum_{j=1}^m x_j^* x_j}{\sqrt{\sum_{j=1}^m x_j^{*2}} \sqrt{\sum_{j=1}^m x_j^2}}$$

Radius Neighbors Classifier

KNN is only suitable to the case where neighboring samples are uniformly distributed, so that we can use the majority voting by assigning equal weights to the neighboring label references.

For the non-uniform case, the KNN variants, [Radius Neighbors](#) and [Weighted KNN](#), would be better choices.

Radius Neighbors Classifier classifies unlabeled test samples by referring to the class of their neighboring samples within a radius in the training set.

The label probability $\mathbb{P}(y^* = c|\mathbf{x}^*)$ for the test sample is estimated as

$$\mathbb{P}(y^* = c|\mathbf{x}^*) = \frac{|\{(\mathbf{x}, y) \in \mathcal{T}_r : y = c, d(\mathbf{x}^*, \mathbf{x}) \leq r\}|}{|\{(\mathbf{x}, y) \in \mathcal{T}_r : d(\mathbf{x}^*, \mathbf{x}) \leq r\}|},$$

where r is the pre-specified radius parameter.

Weighted KNN

By defining the K Nearest Neighbors of \mathbf{x}^* in the training set as

$$\mathcal{N}(\mathbf{x}^*) = \{(\mathbf{x}, y) \in \mathcal{T}_r : d(\mathbf{x}^*, \mathbf{x}) \text{ ranks smallest-}K\},$$

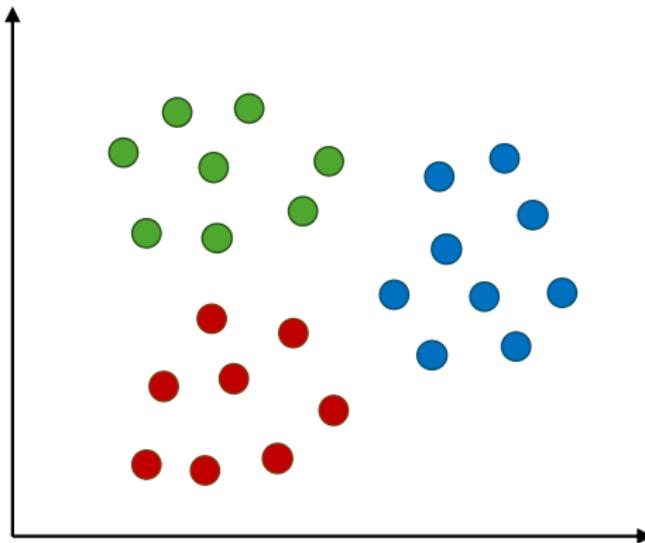
Weighted KNN puts more weights on the neighboring samples closer to the test example \mathbf{x}^* for estimating the probability $\mathbb{P}(y^* = c | \mathbf{x}^*)$

$$\mathbb{P}(y^* = c | \mathbf{x}^*) = \frac{\sum_{(\mathbf{x}, y) \in \mathcal{N}(\mathbf{x}^*)} w_{\mathbf{x}} \cdot \delta(y, c)}{\sum_{(\mathbf{x}, y) \in \mathcal{N}(\mathbf{x}^*)} w_{\mathbf{x}}},$$

where $\delta(y, c)$ is the Kronecker delta function which is equal to 1 if $y = c$ and 0 otherwise, and $w_{\mathbf{x}}$ is the weight assigned to the neighboring sample \mathbf{x} , which can be the inverse of the distance between \mathbf{x} and \mathbf{x}^*

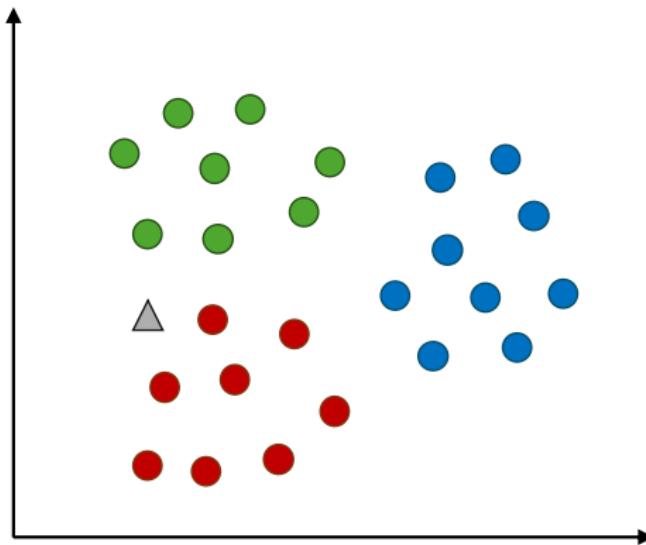
$$w_{\mathbf{x}} = \frac{1}{d(\mathbf{x}^*, \mathbf{x})}.$$

KNN: Example



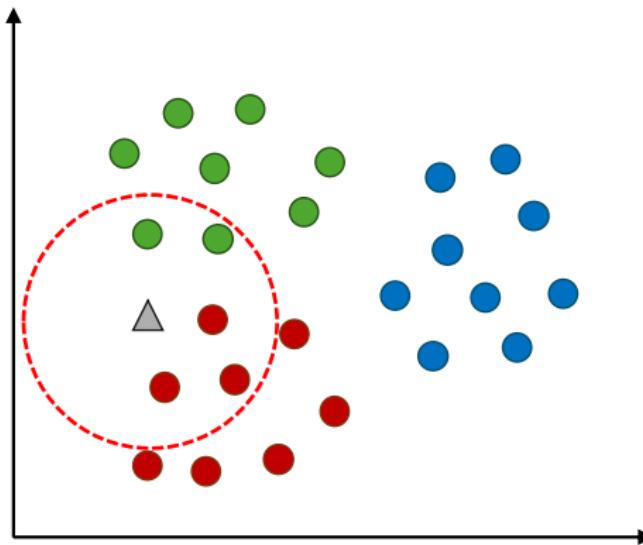
We are given a set of labeled data points in three classes $\{\text{Red}, \text{Green}, \text{Blue}\}$ as training samples.

KNN: Example



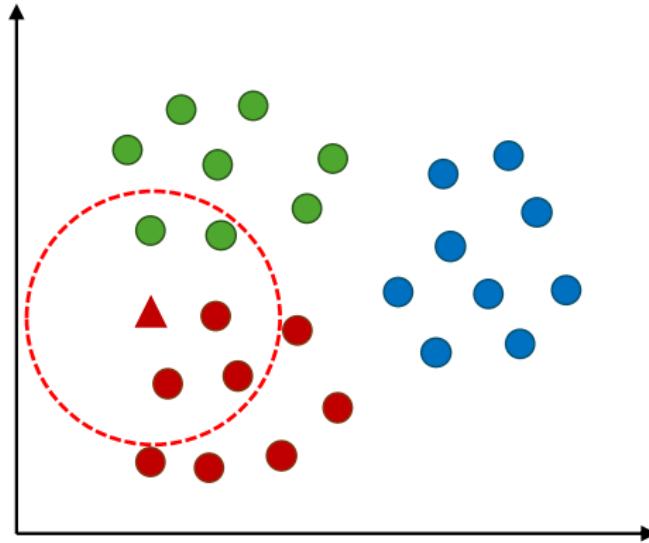
Predict the label of the unlabeled data point represented by the grey triangle.

KNN: Example



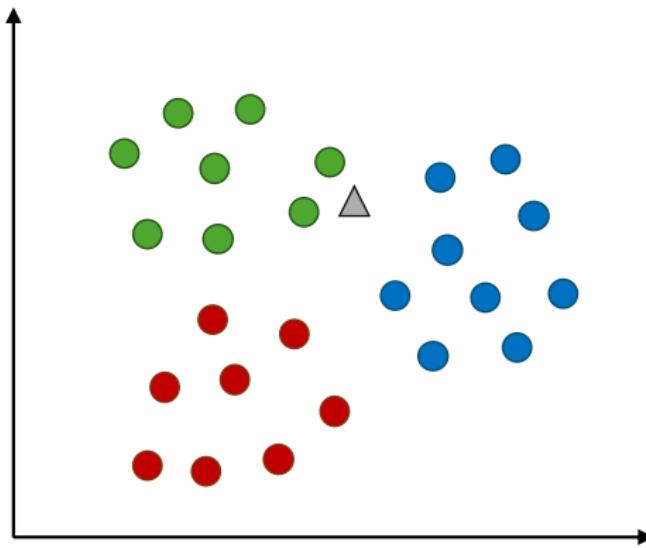
We collect its 5 nearest neighbors with three **Red** data points and two **Green** data points.

KNN: Example



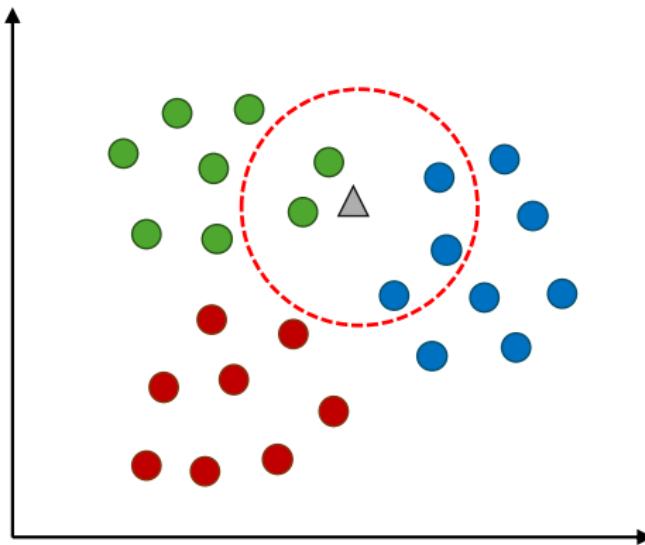
Predict the class of the test data point as **Red**.

KNN: Example



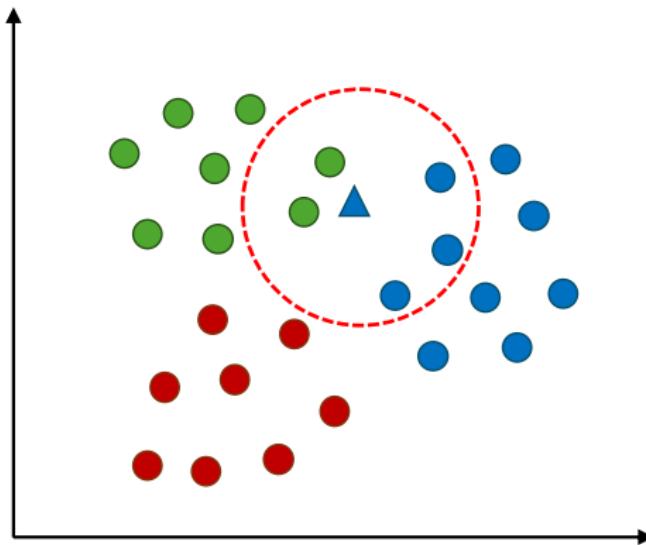
KNN cannot work well for the data points whose K nearest neighbors are not uniformly distributed.

KNN: Example



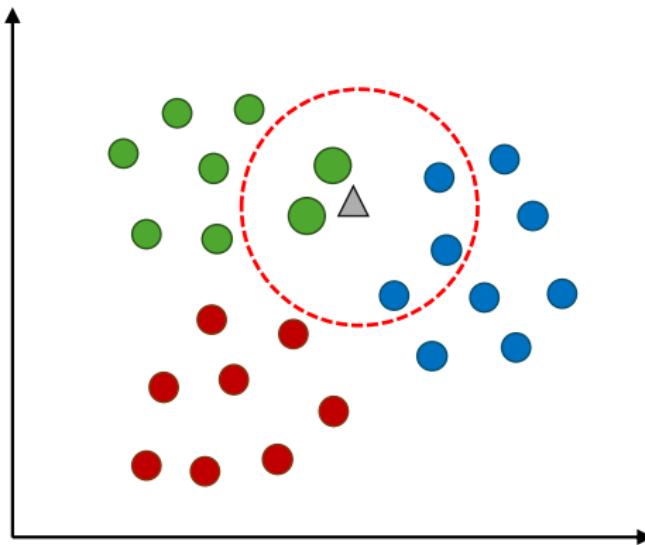
We collect the test data point's 5 nearest neighbors with three **Blue** data points and two **Green** data points.

KNN: Example



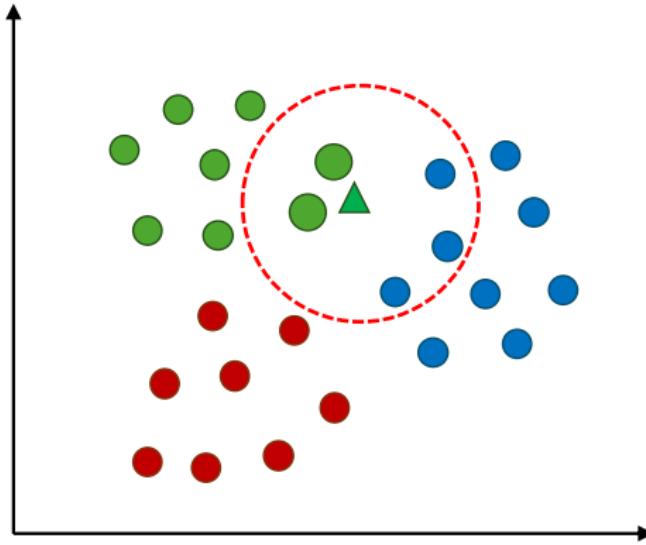
In this case, KNN would unfairly predict the class of the test data point as **Blue**.

Weighted KNN: Example



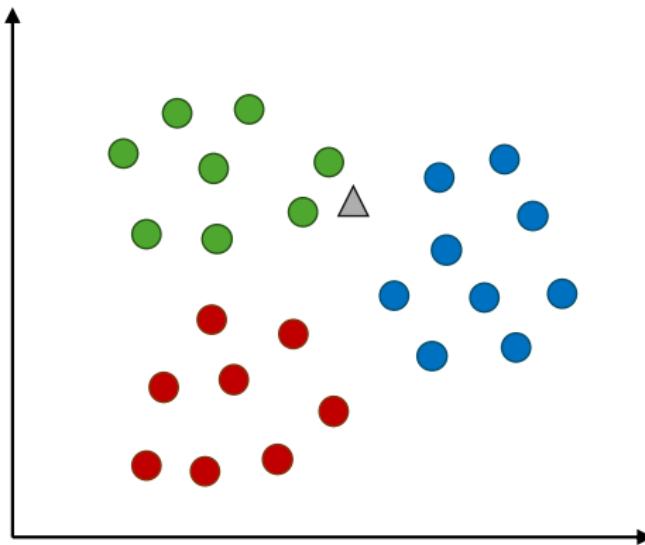
Weighted KNN increases the weights of the closer **Green** neighbors.

Weighted KNN: Example



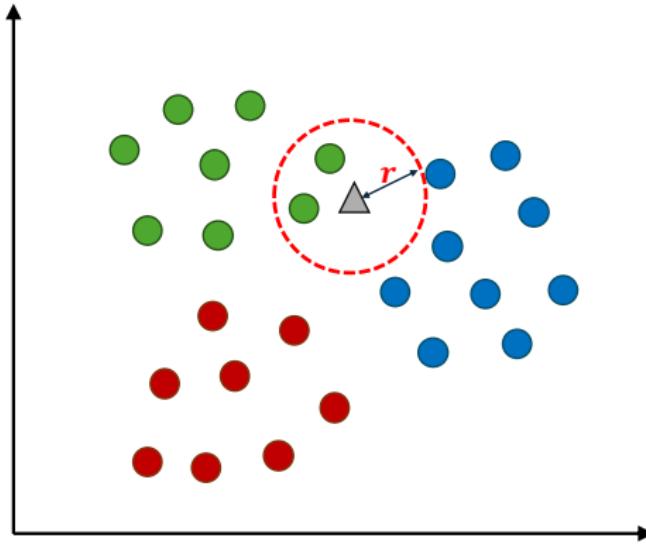
The class of the test data point is then predicted as **Green** fairly.

Radius Neighbors: Example



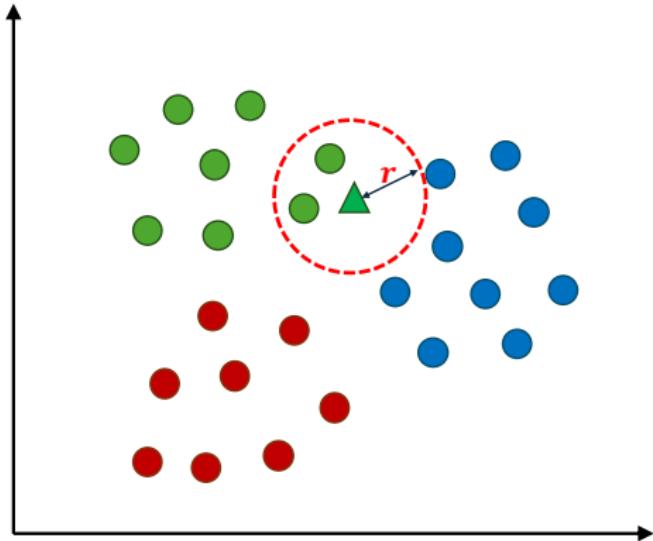
Radius Neighbors Classifier collects neighbors within a fixed radius.

Radius Neighbors: Example



With radius r , two **Green** neighbors are selected.

Radius Neighbors: Example



The class of the test data point is then predicted as **Green** fairly.

KNN: Example

“PlayTennis” prediction with normalized daily weather condition attributes:
training samples

Day	Temperature	Humidity	Wind	PlayTennis
D1	0.8	0.9	0.2	No
D2	0.7	0.8	0.8	No
D3	0.6	0.7	0.3	Yes
D4	0.5	0.9	0.2	Yes
D5	0.2	0.5	0.2	Yes
D6	0.3	0.4	0.7	No
D7	0.5	0.6	0.5	Yes
D8	0.4	0.7	0.2	No
D9	0.1	0.6	0.2	Yes
D10	0.5	0.4	0.1	Yes
D11	0.6	0.4	0.8	Yes
D12	0.4	0.9	0.6	Yes
D13	0.8	0.6	0.1	Yes
D14	0.5	0.8	0.9	No

KNN: Example

Using KNN to predict whether to play tennis for a new day

$D^* = (\text{Temperature}=0.2, \text{Humidity} = 0.8, \text{Wind} = 0.8)$, by setting K to 5 and leveraging Euclidean distance as the distance metric.

First, calculate the distance between the new day D^* and all the training days as

- $d(D^*, D1) = ?$
- $d(D^*, D2) = ?$
- $d(D^*, D3) = ?$
- $d(D^*, D4) = ?$
- $d(D^*, D5) = ?$
- $d(D^*, D6) = ?$
- $d(D^*, D7) = ?$
- $d(D^*, D8) = ?$
- $d(D^*, D9) = ?$
- $d(D^*, D10) = ?$
- $d(D^*, D11) = ?$
- $d(D^*, D12) = ?$
- $d(D^*, D13) = ?$
- $d(D^*, D14) = ?$

KNN: Example

- The 5 nearest neighbors of D^* is ?
- The number of neighboring days with class “PlayTennis = Yes” is ?
- The number of neighboring days with class “PlayTennis = No” is ?
- So, the class of D^* is predicted as “PlayTennis = ?”.

Pros and Cons of KNN

Pros:

- Simple implementation
- No training phase
- Adaptable to new data
- Effective for small datasets

Cons:

- High memory requirements
- Computationally expensive for large datasets
- Sensitive to irrelevant features
- Struggles with imbalanced classes

The End

COMP4131: Data Modelling and Analysis

Lecture 9: Decision Tree and Random Forest

Daokun Zhang

University of Nottingham Ningbo China

daokun.zhang@nottingham.edu.cn

April 14, 2025

Overview

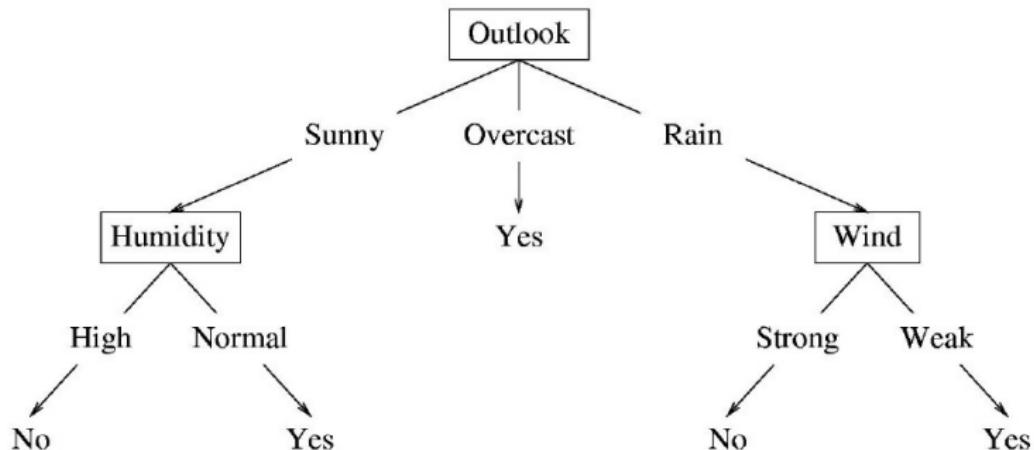
1 Decision Tree

2 Random Forest

Decision Tree

Classification with Decision Tree: An Overview

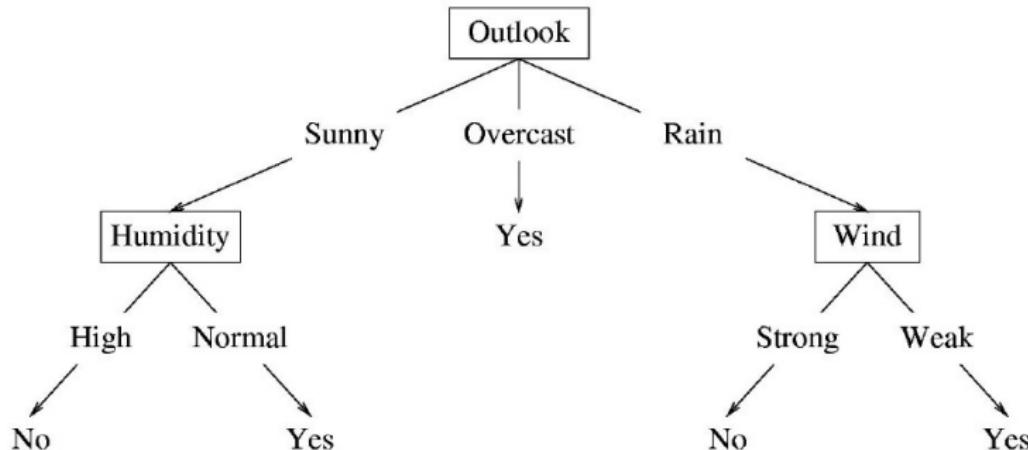
- A possible decision tree for the data:



- What prediction would we make for the new data (Outlook = Sunny, Temperature = Cool, Humidity = High, Wind = Strong)?

Classification with Decision Tree: An Overview

- A possible decision tree for the data:



- Each internal node: test one attribute, like “Outlook”.
- Each branch from a node: select one value for the attribute, like “Outlook = Humidity”.
- Each leaf node: predict the class label, like “PlayTennis = No”.

The History of Decision Tree



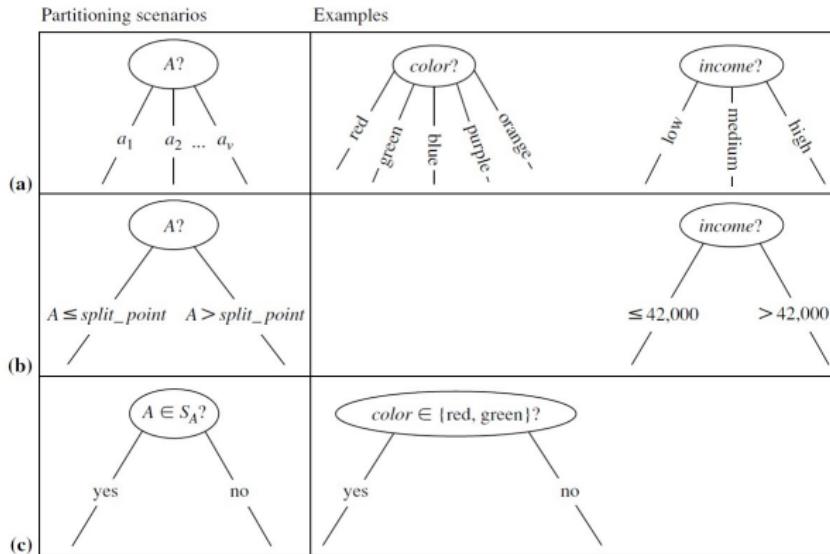
John Ross Quinlan

- ID3
 - During the late 1970s and early 1980s, **John Ross Quinlan**, developed a decision tree algorithm known as ID3 (Iterative Dichotomiser).
- C4.5
 - Quinlan later presented C4.5 (as successor of ID3), which then served as a competitive baseline to benchmark new machine learning algorithms.
- CART
 - In 1984, a group of statisticians published the book Classification and Regression Trees (CART), which described the generation of binary decision trees.

Decision Tree: Core Idea

- In ID3, C4.5, and CART, decision trees are constructed in a **top-down recursive divide-and-conquer** manner.
- The decision tree algorithm starts with a set of training samples including attributes and associated labels.
- The training set is recursively partitioned into smaller subsets as the tree is being built.
- Each partition requires to select an attribute and define a partition rule according to the choice of attribute values.
- **Greedy algorithms** are generally used to do iterative attribute selection and training set partition.

Decision Tree: Data Partition Rules



- (a): Branch according to the value of the categorical-valued attribute A .
(b): Branch using a “split_point” for the continuous-valued attribute A .
(c): Branch with a value subset for the categorical-valued attribute A .
Different from scenario (a), scenarios (b) and (c) construct a binary tree.



Decision Tree Algorithm for Categorical-valued Data

```
1: Input: training dataset  $\mathcal{D}$ , attribute_list;  
2: create a node  $N$ ;  
3: if all samples in  $\mathcal{D}$  have the same class  $C$  then  
4:     return  $N$  as a leaf node labeled with the class  $C$ ;  
5: if attribute_list is empty then  
6:     return  $N$  as a leaf node labeled with the majority class in  $\mathcal{D}$ ;  
7: splitting_attribute  $\leftarrow$  select an attribute from attribute_list;  
8: attribute_list  $\leftarrow$  attribute_list – splitting_attribute;  
9: splitting_criterion  $\leftarrow$  split  $\mathcal{D}$  according to samples' categorical values at  
   splitting_attribute;  
10: for each outcome  $j$  of splitting_criterion do  
11:     let  $\mathcal{D}_j$  be the subset of samples in  $\mathcal{D}$  satisfying outcome  $j$ ;  
12:     if  $\mathcal{D}_j$  is empty then  
13:         attach a leaf labeled with the majority class in  $\mathcal{D}$  to node  $N$ ;  
14:     else  
15:         go to step 1 to generate a decision tree with input  $(\mathcal{D}_j, \textit{attribute\_list})$ ;  
16:         attach the generated decision tree to node  $N$ ;  
17: return  $N$ .
```

Attribute Selection Measure: Information Gain

- ID3 uses **information gain** as its attribute selection measure.
- The attribute with the highest information gain is chosen as the splitting attribute for current node N .
- The highest information gain means that after partition, the information required to classify samples is minimized.
- The expected information needed to classify a sample in \mathcal{D} is given by

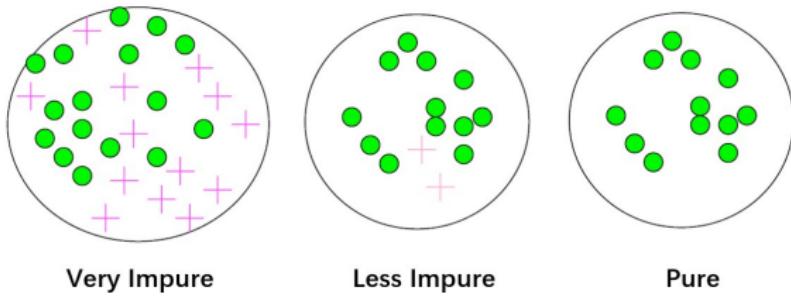
$$\text{Info}(\mathcal{D}) = - \sum_{i=1}^m p_i \log_2(p_i),$$

where m is the number of class, p_i is the probability that an arbitrary sample in \mathcal{D} belongs to class C_i and is estimated by $|\mathcal{C}_{i,\mathcal{D}}|/|\mathcal{D}|$, with $\mathcal{C}_{i,\mathcal{D}}$ being the set of samples in \mathcal{D} having class C_i .

- $\text{Info}(\mathcal{D})$ is just the average amount of information needed to identify the class label of a sample in \mathcal{D} , which is encoded in bits with a log function with base 2.

Attribute Selection Measure: Information Gain

- Suppose we are to partition the tuples in \mathcal{D} on some categorical attribute A having v distinct values, (a_1, a_2, \dots, a_v) , as observed from the training data.
- Then we can split \mathcal{D} into v subsets $\{\mathcal{D}_1, \mathcal{D}_2, \dots, \mathcal{D}_v\}$ where \mathcal{D}_j contains those samples in \mathcal{D} that have the attribute value $A = a_j$.
- These subsets would correspond to the branches grown from node N .
- Ideally, we would like this partitioning to produce an exact classification of the samples in \mathcal{D} . That is, we would like for each subset to be pure.



Attribute Selection Measure: Information Gain

- How much information would we still need (after the partitioning) to arrive at an exact classification? This amount is measured by

$$\text{Info}_A(\mathcal{D}) = \sum_{j=1}^v \frac{|\mathcal{D}_j|}{|\mathcal{D}|} \times \text{Info}(\mathcal{D}_j).$$

- The term $|\mathcal{D}_j|/|\mathcal{D}|$ acts as the weight of the j th partition.
- $\text{Info}_A(\mathcal{D})$ is the expected information required to classify a sample from \mathcal{D} after the partitioning by attribute A .
- The smaller the expected information (still) required, the greater the purity of the subsets.
- Information gain is defined as the difference between the original information requirement and the new requirement (i.e., obtained after partitioning on A). That is,

$$\text{Gain}(A) = \text{Info}(\mathcal{D}) - \text{Info}_A(\mathcal{D}).$$

Attribute Selection Measure: Information Gain

- $\text{Gain}(A)$ tells us how much would be gained by branching on A .
- It is the expected reduction in the information requirement caused by knowing the value of A .
- The attribute A with the highest information gain, $\text{Gain}(A)$, is chosen as the splitting attribute at node N .
- This is equivalent to saying that we want to partition on the attribute A that would do the “best classification”, so that the amount of information still required to finish classifying the samples is minimal (i.e., minimum $\text{Info}_A(\mathcal{D})$).

Attribute Selection Measure: Information Gain

Class-Labeled Training Sample from the *AllElectronics* Customer Database

RID	age	income	student	credit_rating	Class: buys_computer
1	youth	high	no	fair	no
2	youth	high	no	excellent	no
3	middle_aged	high	no	fair	yes
4	senior	medium	no	fair	yes
5	senior	low	yes	fair	yes
6	senior	low	yes	excellent	no
7	middle_aged	low	yes	excellent	yes
8	youth	medium	no	fair	no
9	youth	low	yes	fair	yes
10	senior	medium	yes	fair	yes
11	youth	medium	yes	excellent	yes
12	middle_aged	medium	no	excellent	yes
13	middle_aged	high	yes	fair	yes
14	senior	medium	no	excellent	no

Attribute Selection Measure: Information Gain

- We first compute the expected information needed to classify a sample in \mathcal{D} :

$$\text{Info}(\mathcal{D}) = -\frac{9}{14} \log_2 \left(\frac{9}{14} \right) - \frac{5}{14} \log_2 \left(\frac{5}{14} \right) = 0.940 \text{ bits.}$$

- The expected information needed to classify a sample in \mathcal{D} if the samples are partitioned according to *age* is

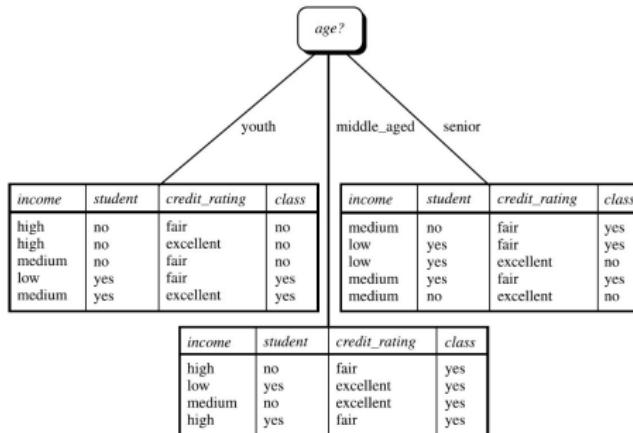
$$\begin{aligned}\text{Info}_{\text{age}}(\mathcal{D}) &= \frac{5}{14} \times \left(-\frac{2}{5} \log_2 \frac{2}{5} - \frac{3}{5} \log_2 \frac{3}{5} \right) + \frac{4}{14} \times \left(-\frac{4}{4} \log_2 \frac{4}{4} \right) \\ &\quad + \frac{5}{14} \times \left(-\frac{3}{5} \log_2 \frac{3}{5} - \frac{2}{5} \log_2 \frac{2}{5} \right) = 0.694 \text{ bits.}\end{aligned}$$

- Hence, the gain in information from such a partitioning would be

$$\text{Gain}(\text{age}) = \text{Info}(\mathcal{D}) - \text{Info}_{\text{age}}(\mathcal{D}) = 0.940 - 0.694 = 0.246 \text{ bits.}$$

Attribute Selection Measure: Information Gain

- Similarly, we can calculate the information gain for other attributes as
 - $\text{Gain}(\text{income}) = 0.029$ bits.
 - $\text{Gain}(\text{student}) = 0.151$ bits.
 - $\text{Gain}(\text{credit_rating}) = 0.048$ bits.
- Because age has the highest information gain among the attributes, it is selected as the splitting attribute.
- Node N is labeled with age , and branches are grown for each of the attribute's values.



Attribute Selection Measure: Information Gain

- But how can we compute the information gain of an attribute that is **continuous-valued**, unlike in the example?
- For such a scenario, we must determine the “best” *split_point* for the continuous-valued attribute A , where the *split_point* is a threshold on A .
- We first sort the values of A observed in the training set in increasing order, also denoted as (a_1, a_2, \dots, a_v) .
- Typically, the midpoint between each pair of adjacent values, a_i and a_{i+1} , is considered as a possible *split_point*, $\frac{a_i + a_{i+1}}{2}$.
- For each possible *split_point* for A , we evaluate $\text{Info}_A(\mathcal{D})$, where the number of split subsets is two.
- We can denote \mathcal{D}_1 as the set of samples in \mathcal{D} satisfying $A \leq \text{split_point}$, \mathcal{D}_2 as the set of samples in \mathcal{D} satisfying $A > \text{split_point}$.

Attribute Selection Measure: Gain Ratio

- The information gain measure is biased toward **splits with many outcomes**. That is, it prefers to select attributes having a large number of values.
- For the *AllElectronics* Customer Database, if we choose the record Id (*RID*) as an attribute, the expected information needed to classify a sample in \mathcal{D} if the samples are partitioned according to *RID* is

$$\text{Info}_{\text{RID}}(\mathcal{D}) = \frac{1}{14} \times \left(-\frac{1}{14} \log_2 \frac{1}{14} \right) + \cdots + \frac{1}{14} \times \left(-\frac{1}{14} \log_2 \frac{1}{14} \right) = 0 \text{ bits.}$$

- The information gain for the partition at *RID* is

$$\text{Gain}(\text{RID}) = \text{Info}(\mathcal{D}) - \text{Info}_{\text{RID}}(\mathcal{D}) = 0.940 - 0 = 0.940 \text{ bits} > \text{Gain}(\text{age}).$$

- The *RID* attribute would be selected to partition \mathcal{D} . However, this partition is meaningless, as test samples have different record IDs, making the trained decision tree unable to generalize.

Attribute Selection Measure: Gain Ratio

- C4.5, a successor of ID3, uses an extension to information gain known as **gain ratio**, which attempts to overcome this bias, by **normalizing information gain** with a “**split information**” value defined analogously with $\text{Info}(\mathcal{D})$ as

$$\text{SplitInfo}_A(\mathcal{D}) = - \sum_{j=1}^v \frac{|\mathcal{D}_j|}{|\mathcal{D}|} \times \log_2 \left(\frac{|\mathcal{D}_j|}{|\mathcal{D}|} \right),$$

which would be large if the number of split subsets v is large.

- The gain ratio is defined as

$$\text{GainRatio}(A) = \frac{\text{Gain}(A)}{\text{SplitInfo}_A(\mathcal{D})}.$$

- The attribute with the maximum gain ratio is selected as the splitting attribute.

Attribute Selection Measure: Gini Index

- CART uses **Gini index** to measures the impurity of sample set \mathcal{D} ,

$$\text{Gini}(\mathcal{D}) = 1 - \sum_{i=1}^m p_i^2,$$

where p_i is the probability that a sample in \mathcal{D} belongs to class C_i and is estimated by $|\mathcal{C}_{i,\mathcal{D}}|/|\mathcal{D}|$, with $\mathcal{C}_{i,\mathcal{D}}$ being the set of samples in \mathcal{D} having class C_i . The sum is computed over m classes.

- The Gini index considers a binary split for each attribute.
 - For the case A is a categorical-valued attribute having v distinct values, (a_1, a_2, \dots, a_v) , each subset, \mathcal{S}_A , can be considered as a binary test for attribute A of the form “ $A \in \mathcal{S}_A$ ”?
 - For continuous-valued attributes, each possible split-point must be considered to construct a binary split “ $A \leq \text{split_point}$ ” and “ $A > \text{split_point}$ ”.

Attribute Selection Measure: Gini Index

- When considering a binary split, we compute a weighted sum of the impurity of each resulting partition.
- For example, if a binary split on A partitions \mathcal{D} into \mathcal{D}_1 and \mathcal{D}_2 , the Gini index of \mathcal{D} given that partitioning is

$$\text{Gini}_A(\mathcal{D}) = \frac{|\mathcal{D}_1|}{|\mathcal{D}|} \text{Gini}(\mathcal{D}_1) + \frac{|\mathcal{D}_2|}{|\mathcal{D}|} \text{Gini}(\mathcal{D}_2).$$

- The reduction in impurity/Gini index that would be incurred by a binary split on a categorical- or continuous-valued attribute A is

$$\Delta \text{Gini}(A) = \text{Gini}(\mathcal{D}) - \text{Gini}_A(\mathcal{D}).$$

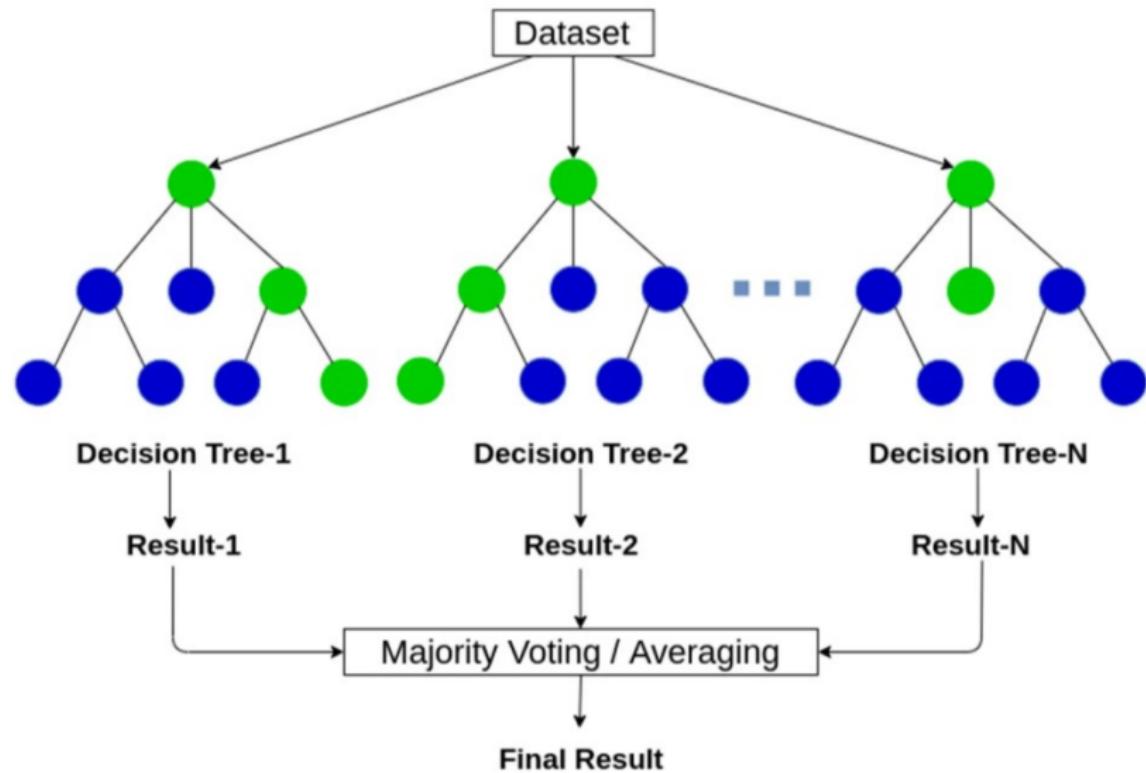
- The attribute that maximizes the reduction in impurity/Gini index is selected as the splitting attribute.
- This attribute and either its splitting subset (for a categorical-valued splitting attribute) or split-point (for a continuous-valued splitting attribute) together form the splitting criterion.

Decision Tree Classification and Regression

- Through iteratively selecting an attribute to split the set of training samples at each node, we can construct a complete tree.
- The Information Gain, Gain Ratio and Gini Index measures are designed to select splitting attributes for the classification case, and finally the **majority class of the samples** in each leaf node is assigned as the class of the leaf node.
- For the regression case, **the reduction in the mean squared error of average prediction** can be used to select splitting attributes, and finally the **average target value of the samples** in each leaf node is assigned as the target value of the leaf node.

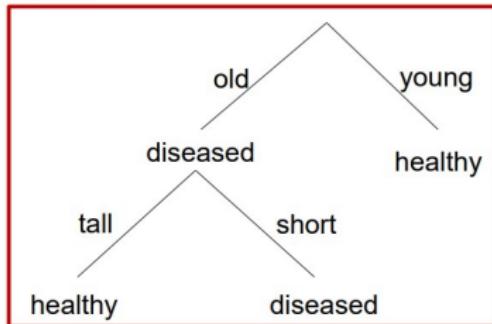
Random Forest

Intuition of Random Forest

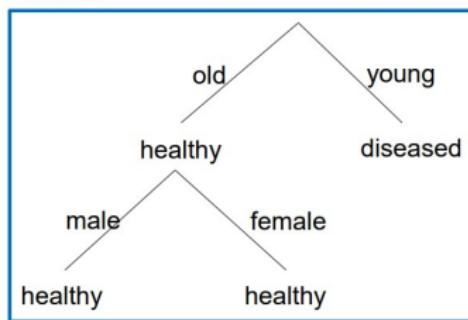


Random Forest Classification

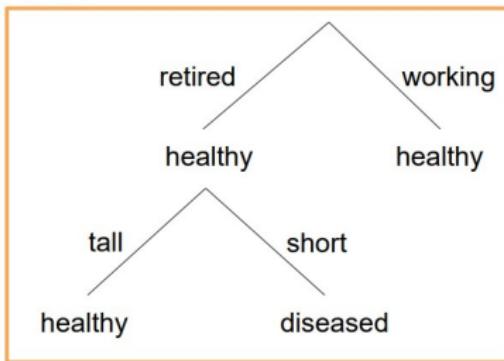
Tree 1



Tree 2



Tree 3



- New sample:
(old, retired, male, short)
- Tree predictions:
 - Tree 1: diseased
 - Tree 2: healthy
 - Tree 3: diseased
- Majority rule: **diseased**

Bagging

The training algorithm for random forests applies the general technique of **bootstrap aggregating**, or **bagging**, to tree learners. Given a training set $X = (\mathbf{x}_1, \dots, \mathbf{x}_n)$ with responses $Y = (y_1, \dots, y_n)$, bagging repeatedly (B times) selects a random sample with replacement of the training set and fits trees to these samples:

For $i = 1, \dots, B$:

1. Sample, with replacement, n training examples from X, Y ; call these X_i, Y_i .
2. Train a classification or regression tree f_i on X_i, Y_i .

After training, predictions for unseen samples \mathbf{x}' can be made by averaging the predictions from all the individual regression trees on \mathbf{x}' :

$$\hat{f}(\mathbf{x}') = \frac{1}{B} \sum_{i=1}^B f_i(\mathbf{x}'),$$

or by taking the majority vote in the case of classification trees.



Bagging Effects on Variance Reduction

For the regression case, recall the decomposition of the mean squared error:

$$\text{Mean Squared Error} = \text{Bias}^2 + \text{Variance} + \text{Noise}.$$

If trees are sufficiently deep, they have very small bias.

Bagging can reduce the variance of a single tree:

- For a given test example \mathbf{x}' , we can compare the variance of the bagging prediction $\hat{f}(\mathbf{x}')$ and a single tree's prediction $f_i(\mathbf{x}')$.
- Suppose under the randomness of the training set sampling,
 - the variance of $f_i(\mathbf{x}')$ is

$$\text{Var}[f_i(\mathbf{x}')] = \sigma^2,$$

- the co-variance between $f_i(\mathbf{x}')$ and $f_j(\mathbf{x}')$ with $i \neq j$ is

$$\text{Cov}[f_i(\mathbf{x}'), f_j(\mathbf{x}')] = \rho\sigma^2,$$

where ρ is the Pearson correlation coefficient between $f_i(\mathbf{x}')$ and $f_j(\mathbf{x}')$.



Bagging Effects on Variance Reduction

- We can calculate the variance of the bagging prediction $\hat{f}(\mathbf{x}')$ as

$$\begin{aligned}\text{Var} \left[\hat{f}(\mathbf{x}') \right] &= \text{Var} \left[\frac{1}{B} \sum_{i=1}^B f_i(\mathbf{x}') \right] \\ &= \frac{1}{B^2} \sum_{i=1}^B \sum_{j=1}^B \text{Cov}[f_i(\mathbf{x}'), f_j(\mathbf{x}')] \\ &= \frac{1}{B^2} \sum_{i=1}^B \left\{ \sum_{j=1, j \neq i}^B \text{Cov}[f_i(\mathbf{x}'), f_j(\mathbf{x}')] + \text{Var}[f_i(\mathbf{x}')] \right\} \\ &= \frac{1}{B^2} \sum_{i=1}^B [(B-1)\rho\sigma^2 + \sigma^2] \\ &= \frac{(B-1)\rho\sigma^2}{B} + \frac{\sigma^2}{B} \\ &= \rho\sigma^2 + \sigma^2 \frac{1-\rho}{B}.\end{aligned}$$

Bagging Effects on Variance Reduction

- From the identity $\text{Var} [\hat{f}(\mathbf{x}')]=\rho\sigma^2+\sigma^2\frac{1-\rho}{B}$, we can see that if $B \geq 2$, $\text{Var} [\hat{f}(\mathbf{x}')]<\sigma^2=\text{Var}[f_i(\mathbf{x}')]$.
- However, the reduction of $\text{Var} [\hat{f}(\mathbf{x}')]$ would be very small, if ρ is relatively large.
- From the identity $\text{Var} [\hat{f}(\mathbf{x}')]=\frac{(B-1)\rho\sigma^2}{B}+\frac{\sigma^2}{B}$, we can find that if we want to reduce $\text{Var} [\hat{f}(\mathbf{x}')]$ (irrespective of B), we need reduce the Pearson correlation coefficient ρ .
- Random Forest uses random feature selection (feature bagging) to de-correlate individual trees.

Random Forest

For $i = 1, \dots, B$:

1. Sample, with replacement, n training examples from X, Y ; call these X_i, Y_i .
2. Grow a classification or regression tree f_i on X_i, Y_i with **feature bagging**:
 - 2.1 At each partitioning, **randomly select a subset of attributes** from all attributes;
 - 2.2 Find the best attribute/split point from the attribute subset to do the split;

After training, predictions for unseen samples \mathbf{x}' can be made by averaging the predictions from all the individual regression trees on \mathbf{x}' :

$$\hat{f}(\mathbf{x}') = \frac{1}{B} \sum_{i=1}^B f_i(\mathbf{x}'),$$

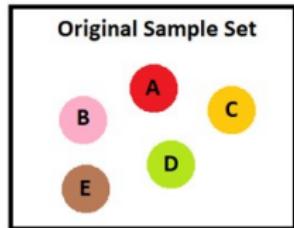
or by taking the majority vote in the case of classification trees.

Random Forest Evaluation: Out-Of-Bag (OOB) Error

Random Forest use a bag (random subset) of samples to train each individual tree. The samples not in the set are called **Out-Of-Bag (OOB) samples**. As an evaluation metric of random forest, we can calculate the OOB error in the following procedure:

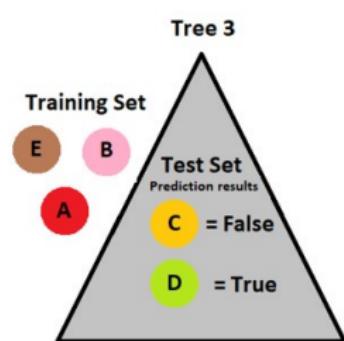
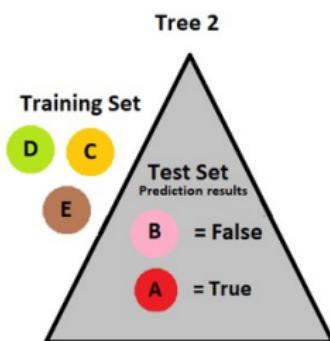
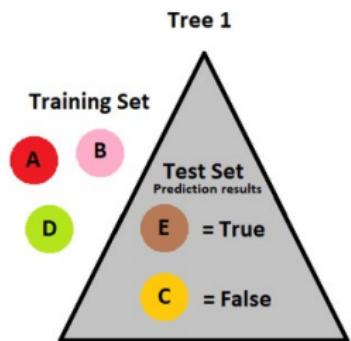
1. For each OOB sample, find all trees that are not trained by the OOB sample.
2. Take the majority vote of these trees' prediction for the OOB sample, or average these trees' prediction for regression case, compare the ensemble prediction to the true value of the OOB sample.
3. Calculate the OOB error as the averaged prediction error of all OOB samples.

Random Forest Evaluation: OOB Error



Samples	Majority Vote	Ground Truth
A	True	False
B	False	True
C	False	True
D	True	True
E	True	True

2/5 Out of Bag Error



Random Forest Property: Attribute Importance Evaluation

With Random Forest, the importance of an attribute can be evaluated by the **permutation importance**. Taking the classification case as an example, the permutation importance for an attribute is calculated as

1. Record the prediction accuracy on the OOB samples for each tree.
2. Randomly permute the data for attribute a_j in the OOB samples, then record the accuracy again.
3. The decrease in accuracy as a result of this permuting is averaged over all trees, and is used as a measure of the importance of attribute a_j in the random forest.

The attribute importance evaluation can be used to do feature selection, remove irrelevant/noisy features, increase the generalization ability of the trained machine learning models.

References

- Decision Tree:
 - Jiawei Han, Micheline Kamber, and Jian Pei. "**Data Mining: Concepts and Techniques.**" Third Edition. (2013) ([Chapter 8.2 Decision Tree Induction](#))
- Random Forest:
 - https://en.wikipedia.org/wiki/Random_forest
 - https://en.wikipedia.org/wiki/Out-of-bag_error

The End

COMP4131: Data Modelling and Analysis

Lecture 10: Support Vector Machines and Kernel Methods

Daokun Zhang

University of Nottingham Ningbo China

daokun.zhang@nottingham.edu.cn

April 21, 2025

FAQs for Coursework 2

- Q1: Should I compare my solutions with the state-of-the-art baselines?
- A1: Comparison with the state-of-the-art is encouraged but not compulsory.
- Q2: Should I work on the classification and regression tasks simultaneously?
- A2: Please focus on only one task. You can choose one of the three tasks: classification, regression and clustering.
- Q3: Can I choose a method that has not been covered by our module?
- A3: Yes. The solution methods are not limited to what we have learned. We encourage you to explore more advanced machine learning techniques to solve your problem.

FAQs for Coursework 2

- Q4: How many solution methods should I compare?
- A4: It's expected that more than 3 methods should be investigated. Generally, the more the better, but enough words and space have to be used for describing data pre-processing details, justifying the reason for choosing the used methods, showcasing and analyzing the experimental results.
- Q5: What data scale is expected?
- A5: Generally, a dataset with more than 2000 samples is favored. If the dataset you identified contains too many samples, like in million or billion scales, you can do some down-sampling to reduce the scale.
- Q6: Can I use image or textual data?
- A6: No. Please use the tabular data.

FAQs for Coursework 2

- Q7: Can I re-use the datasets of our lab sessions?
- A7: No. Please identify some new datasets.
- Q8: Can I use the same dataset with some other student?
- A8: No. Please be different to avoid the suspicion of plagiarism.

Overview

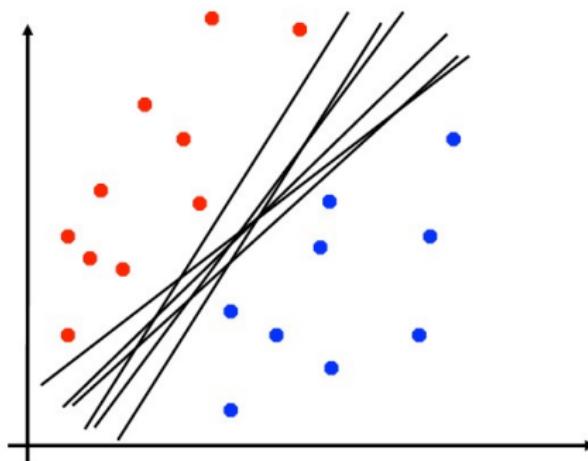
1 Support Vector Machines

2 Kernel Methods

Support Vector Machines

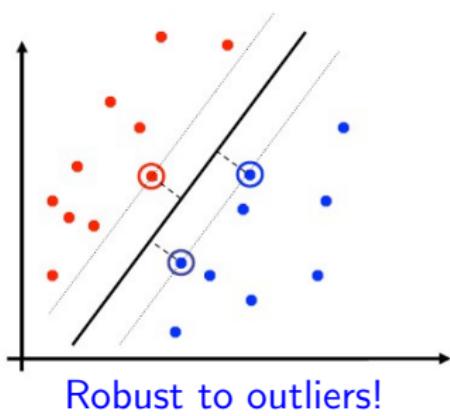
Linear Separators

- If training data is linearly separable, we can find some **linear separator** to distinguish two classes of examples.
- Which of these is **optimal**?



Support Vector Machines (SVMs)

- SVMs (Vapnik, 1990's) choose the linear separator with the largest margin.



Vladimir Vapnik

- Good according to intuition, theory, practice.
- SVMs became famous when, using images as input, it gave accuracy comparable to neural-network with hand-designed features in a handwriting recognition task.

Geometry of Linear Separators

In a high-dimensional Euclidean space, a linear separator is determined by a **hyperplane** that can be specified as the set of points given by

$$\mathbf{p} = \mathbf{a} + s\mathbf{u} + t\mathbf{v}, \quad s, t \in \mathbb{R},$$

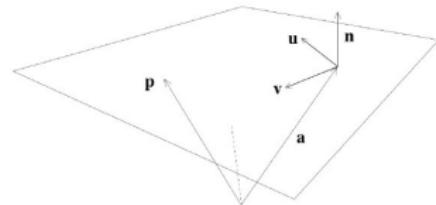
where \mathbf{a} is a vector from origin to a point in the plane, and \mathbf{u} and \mathbf{v} denote two non-parallel directions in the plane.

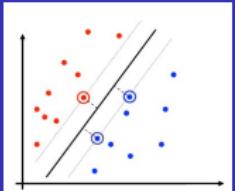
Alternatively, the points can be specified as

$$(\mathbf{p} - \mathbf{a}) \cdot \mathbf{n} = 0 \Leftrightarrow \mathbf{p} \cdot \mathbf{n} = \mathbf{a} \cdot \mathbf{n},$$

where \mathbf{n} is the **norm vector**, which is also noted as \mathbf{w} , and the dot product $\mathbf{a} \cdot \mathbf{n}$ can be treated as a scalar to be specified, denoted as the **offset** b .

So, the hyperplane can be represented as $\mathbf{w}^T \mathbf{x} + b = 0$.





Hard-SVM

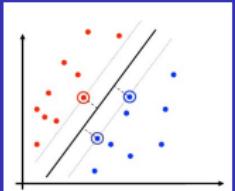
Suppose we are given a set of training samples $\{(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_N, y_N)\}$ that are linearly separable, the distance between the data point \mathbf{x}_n with class label $y_n \in \{+1, -1\}$ and the hyperplane $\mathbf{w}^T \mathbf{x} + b = 0$ is

$$\frac{|\mathbf{w}^T \mathbf{x}_n + b|}{\|\mathbf{w}\|},$$

where $\|\cdot\|$ is the 2-norm of a vector.

As we are only interested in the linear separators for which all data points are correctly classified, implying that $\mathbf{w}^T \mathbf{x}_n + b \geq 0$ for $y_n = +1$ and $\mathbf{w}^T \mathbf{x}_n + b < 0$ for $y_n = -1$, i.e., $y_n(\mathbf{w}^T \mathbf{x}_n + b) \geq 0$, the distance between the data point \mathbf{x}_n and the hyperplane can be rewritten as

$$\frac{y_n(\mathbf{w}^T \mathbf{x}_n + b)}{\|\mathbf{w}\|}.$$



Hard-SVM

The margin between the hyperplane and all training samples is determined by the data points closest to the hyperplane, whose value is calculated as

$$\min_n \frac{y_n(\mathbf{w}^T \mathbf{x}_n + b)}{\|\mathbf{w}\|}.$$

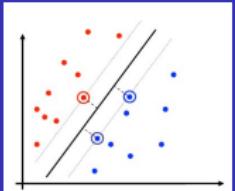
The hyperplane corresponding to the largest margin can be specified as

$$\arg \max_{\mathbf{w}, b} \left\{ \frac{1}{\|\mathbf{w}\|} \min_n [y_n (\mathbf{w}^T \mathbf{x}_n + b)] \right\},$$

where we have taken the factor $1/\|\mathbf{w}\|$ outside the optimization over n because \mathbf{w} does not depend on n .

Direct solution of this optimization problem would be very complex, and so we shall convert it into an equivalent problem that is much easier to solve.





Hard-SVM

We note that if we make the rescaling $\mathbf{w} \rightarrow \kappa\mathbf{w}$ and $b \rightarrow \kappa b$, the distance between any point \mathbf{x}_n to the hyperplane, given by $y_n(\mathbf{w}^T \mathbf{x}_n + b)/\|\mathbf{w}\|$, remains unchanged. We can use this freedom to set

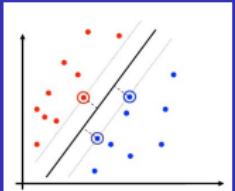
$$y_n (\mathbf{w}^T \mathbf{x}_n + b) = 1$$

for the point that is closest to the hyperplane. In this case, all data points will satisfy the constraints

$$y_n (\mathbf{w}^T \mathbf{x}_n + b) \geq 1, \quad n = 1, \dots, N.$$

In the case of data points for which the equality holds, the constraints are said to be **active**, whereas for the remainder they are said to be **inactive**.

By definition, there will always be **at least one active constraint**, because there will always be a closest point, and once the margin has been maximized there will be **at least two active constraints**.



Hard-SVM

The optimization problem then simply requires that we maximize $\|\mathbf{w}\|^{-1}$, which is equivalent to minimizing $\|\mathbf{w}\|^2$, and so we have to solve the optimization problem

$$\min_{\mathbf{w}, b} \frac{1}{2} \|\mathbf{w}\|^2 \quad \text{s.t.} \quad y_n (\mathbf{w}^T \mathbf{x}_n + b) \geq 1, \quad n = 1, \dots, N.$$

The factor of $1/2$ is included for mathematical convenience. This is a quadratic programming problem in which we are trying to minimize a quadratic function subject to a set of linear inequality constraints.

Soft-SVM

The Hard-SVM formulation assumes that the training set is linearly separable, which is a rather strong assumption.

Soft-SVM can be viewed as a relaxation of the Hard-SVM rule that can be applied even if the training set is not linearly separable.

The optimization problem for Hard-SVM enforces the hard constraints $y_n (\mathbf{w}^T \mathbf{x}_n + b) \geq 1$ for all n .

A natural relaxation is to allow the constraint to be violated for some of the examples in the training set.

This can be modeled by introducing nonnegative slack variables, ξ_1, \dots, ξ_N , and replacing each constraint $y_n (\mathbf{w}^T \mathbf{x}_n + b) \geq 1$ by the constraint $y_n (\mathbf{w}^T \mathbf{x}_n + b) \geq 1 - \xi_n$.

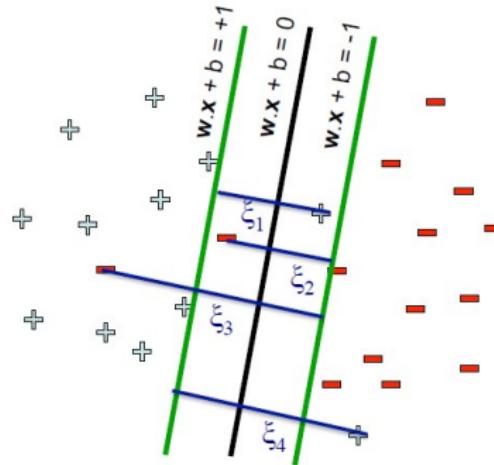
That is, ξ_n measures by how much the constraint $y_n (\mathbf{w}^T \mathbf{x}_n + b) \geq 1$ is being violated.

Soft-SVM

Soft-SVM jointly minimizes the norm of \mathbf{w} (corresponding to the margin) and the average of ξ_n (corresponding to the violations of the constraints).

The tradeoff between the two terms is controlled by a parameter C . The Soft-SVM optimization problem can be formulated as

$$\begin{aligned} \min_{\mathbf{w}, b, \xi} \quad & \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{n=1}^N \xi_n \\ \text{s.t.} \quad & y_n (\mathbf{w}^T \mathbf{x}_n + b) \geq 1 - \xi_n \\ & \text{and } \xi_n \geq 0, \\ & n = 1, 2, \dots, N. \end{aligned}$$



Hinge Loss

For the constraint $y_n (\mathbf{w}^T \mathbf{x}_n + b) \geq 1 - \xi_n$, if $y_n (\mathbf{w}^T \mathbf{x}_n + b) \geq 1$ is satisfied, the minimization over ξ_n with the constraint $\xi_n \geq 0$ would make $\xi_n = 0$; otherwise, $\xi_n = 1 - y_n (\mathbf{w}^T \mathbf{x}_n + b)$. In this sense, ξ_n can be expressed as

$$\xi_n = \max\{0, 1 - y(\mathbf{w}^T \mathbf{x}_n + b)\}.$$

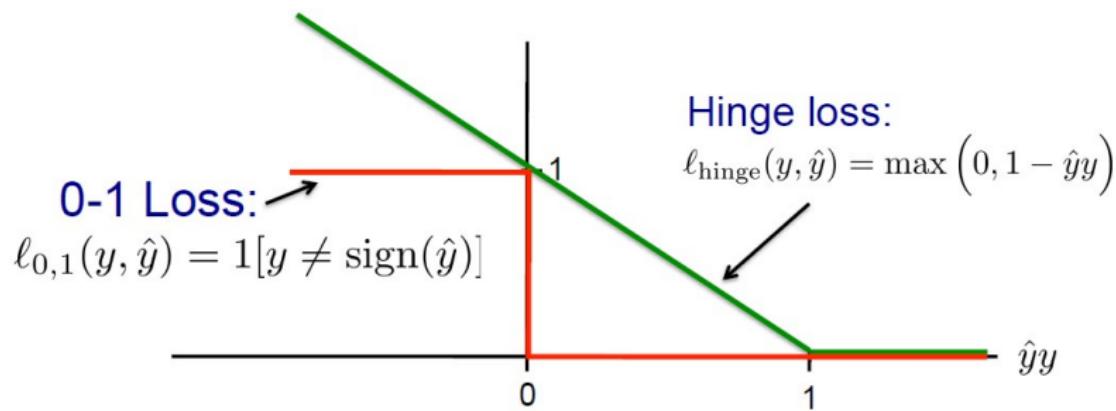
Then the Soft-SVM optimization problem can be re-formulated as

$$\min_{\mathbf{w}, b} \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{n=1}^N \max\{0, 1 - y(\mathbf{w}^T \mathbf{x}_n + b)\}.$$

By denoting $\hat{y}_n = \mathbf{w}^T \mathbf{x}_n + b$, ξ_n defines the hinge loss to measure the discrepancy between the prediction \hat{y}_n and the ground truth y_n :

$$\ell_{\text{hinge}}(y_n, \hat{y}_n) = \max(0, 1 - \hat{y}_n y_n).$$

Hinge Loss



Hinge loss upper bounds 0-1 loss!

It is the tightest *convex* upper bound on the 0-1 loss.

Dual Form for Hard-SVM

Given the Hard-SVM's primal optimization problem

$$\min_{\mathbf{w}, b} \frac{1}{2} \|\mathbf{w}\|^2 \quad s.t. \quad y_n (\mathbf{w}^T \mathbf{x}_n + b) \geq 1, \quad n = 1, \dots, N,$$

which might be difficult to solve. We can derive its equivalent dual form, by starting with constructing a function

$$\begin{aligned} g(\mathbf{w}, b) &= \max_{\alpha \in \mathbb{R}^N : \alpha \geq \mathbf{0}} \sum_{n=1}^N \alpha_n \{1 - y_n (\mathbf{w}^T \mathbf{x}_n + b)\} \\ &= \begin{cases} 0 & \text{if } \forall n, y_n (\mathbf{w}^T \mathbf{x}_n + b) \geq 1 \\ \infty & \text{otherwise} \end{cases}. \end{aligned}$$

Dual Form for Hard-SVM

The Hard-SVM's primal optimization problem can be re-formulated as

$$\min_{\mathbf{w}, b} \left\{ \frac{1}{2} \|\mathbf{w}\|^2 + g(\mathbf{w}, b) \right\}, \text{ i.e.,}$$

$$\min_{\mathbf{w}, b} \max_{\alpha \in \mathbb{R}^N : \alpha \geq \mathbf{0}} \left\{ \frac{1}{2} \|\mathbf{w}\|^2 + \sum_{n=1}^N \alpha_n \{1 - y_n (\mathbf{w}^T \mathbf{x}_n + b)\} \right\}.$$

Now suppose that we flip the order of min and max in the above equation. This can only decrease the objective value, and we have

$$\begin{aligned} & \min_{\mathbf{w}, b} \max_{\alpha \in \mathbb{R}^N : \alpha \geq \mathbf{0}} \left\{ \frac{1}{2} \|\mathbf{w}\|^2 + \sum_{n=1}^N \alpha_n \{1 - y_n (\mathbf{w}^T \mathbf{x}_n + b)\} \right\} \\ & \geq \max_{\alpha \in \mathbb{R}^N : \alpha \geq \mathbf{0}} \min_{\mathbf{w}, b} \left\{ \frac{1}{2} \|\mathbf{w}\|^2 + \sum_{n=1}^N \alpha_n \{1 - y_n (\mathbf{w}^T \mathbf{x}_n + b)\} \right\}. \end{aligned}$$

Dual Form for Hard-SVM

The inequality (\geq) is called **weak duality**. It turns out that in our case, **strong duality** also holds; namely, the inequality holds with equality.

Therefore, the dual problem is

$$\max_{\alpha \in \mathbb{R}^N: \alpha \geq 0} \min_{\mathbf{w}, b} \left\{ \frac{1}{2} \|\mathbf{w}\|^2 + \sum_{n=1}^N \alpha_n \{1 - y_n (\mathbf{w}^T \mathbf{x}_n + b)\} \right\}.$$

We can simplify the dual problem by noting that once α is fixed, the optimization problem with respect to \mathbf{w} is unconstrained and the objective is differentiable; thus, at the optimum, the gradient equals zero:

$$\mathbf{w} - \sum_{n=1}^N \alpha_n y_n \mathbf{x}_n = 0 \Rightarrow \mathbf{w} = \sum_{n=1}^N \alpha_n y_n \mathbf{x}_n.$$

Dual Form for Hard-SVM

However, as $\frac{1}{2}\|\mathbf{w}\|^2 + \sum_{n=1}^N \alpha_n \{1 - y_n (\mathbf{w}^\top \mathbf{x}_n + b)\}$ is a linear function of b with coefficient $-\sum_{n=1}^N \alpha_n y_n$, its minimum value would be $-\infty$ for all α except for the α satisfying

$$\sum_{n=1}^N \alpha_n y_n = 0,$$

which is a necessary condition to make sure the maximization dual problem has a valid optimal solution.

The Hard-SVM's dual optimization problem can be finally formulated as

$$\begin{aligned} & \max_{\alpha \in \mathbb{R}^N} \left\{ -\frac{1}{2} \sum_{n=1}^N \sum_{m=1}^N \alpha_n \alpha_m y_n y_m \mathbf{x}_n^\top \mathbf{x}_m + \sum_{n=1}^N \alpha_n \right\} \\ & \text{s.t. } \alpha_n \geq 0 \text{ for } n = 1, 2, \dots, N \text{ and } \sum_{n=1}^N \alpha_n y_n = 0. \end{aligned}$$

Dual Form for Hard-SVM

Once we find the optimal solution α^* to the dual optimization problem, how shall we convert it to the optimal solution \mathbf{w}^* and b^* to the primal problem?

The optimal \mathbf{w}^* is straightforward with $\mathbf{w}^* = \sum_{n=1}^N \alpha_n^* y_n \mathbf{x}_n$.

The optimal b^* is a bit difficult to be derived. At the optimal solution with \mathbf{w}^* and α^* having been specified, b^* should be the optimal solution to the optimization problem

$$\min_b \left\{ \frac{1}{2} \|\mathbf{w}^*\|^2 + \sum_{n=1}^N \alpha_n^* \left\{ 1 - y_n (\mathbf{w}^{*T} \mathbf{x}_n + b) \right\} \right\}.$$

If $\alpha_n^* > 0$ for only one n , the value of $\alpha_n^* \left\{ 1 - y_n (\mathbf{w}^{*T} \mathbf{x}_n + b) \right\}$ should be minimized with the optimal b^* .

As it is impossible for $1 - y_n (\mathbf{w}^{*T} \mathbf{x}_n + b^*) < 0$ (otherwise α_n^* would be zero for the optimality), b^* should satisfy $1 - y_n (\mathbf{w}^{*T} \mathbf{x}_n + b^*) = 0$.

Dual Form for Hard-SVM

More rigorously, it can be proved that

1. There exists $\alpha_n^* > 0$, and
2. $1 - y_n (\mathbf{w}^{*T} \mathbf{x}_n + b^*) = 0$ for all $\alpha_n^* > 0$ (KKT condition).

The data point \mathbf{x}_n with $\alpha_n > 0$ are termed as **support vectors**. Multiplying the left and right sides of the equations $1 - y_n (\mathbf{w}^{*T} \mathbf{x}_n + b^*) = 0$ by y_n , we can solve b^* as

$$\begin{aligned} b^* &= \frac{\sum_{n=1}^N \mathbb{I}(\alpha_n^* > 0) \cdot (y_n - \mathbf{w}^{*T} \mathbf{x}_n)}{\sum_{n=1}^N \mathbb{I}(\alpha_n^* > 0)} \\ &= \frac{\sum_{n=1}^N \mathbb{I}(\alpha_n^* > 0) \cdot \left(y_n - \sum_{m=1}^N \alpha_m^* y_m \mathbf{x}_m^T \mathbf{x}_n \right)}{\sum_{n=1}^N \mathbb{I}(\alpha_n^* > 0)}, \end{aligned}$$

where $\mathbb{I}(\alpha_n^* > 0)$ is an indicator function, whose value is 1 if $\alpha_n^* > 0$, and 0 otherwise.

Dual Form for Hard-SVM

Given the feature vector of a new test sample \mathbf{x} , its label \hat{y} can be predicted as

$$\hat{y} = \begin{cases} +1 & \text{if } \mathbf{w}^{*T}\mathbf{x} + b^* \geq 0 \\ -1 & \text{if } \mathbf{w}^{*T}\mathbf{x} + b^* < 0 \end{cases}.$$

Equivalently,

$$\hat{y} = \begin{cases} +1 & \text{if } \sum_{n=1}^N \alpha_n^* y_n \mathbf{x}_n^T \mathbf{x} + b^* \geq 0 \\ -1 & \text{if } \sum_{n=1}^N \alpha_n^* y_n \mathbf{x}_n^T \mathbf{x} + b^* < 0 \end{cases}.$$

Dual Form for Soft-SVM

Given the Soft-SVM's primal optimization problem

$$\begin{aligned} & \min_{\mathbf{w}, b, \xi} \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{n=1}^N \xi_n \\ \text{s.t. } & y_n (\mathbf{w}^T \mathbf{x}_n + b) \geq 1 - \xi_n \text{ and } \xi_n \geq 0, \\ & n = 1, 2, \dots, N. \end{aligned}$$

We can also derive its equivalent dual form, by starting with constructing a function

$$\begin{aligned} g(\mathbf{w}, b, \xi) &= \max_{\alpha \geq 0, \mu \geq 0} \left\{ \sum_{n=1}^N \alpha_n \{1 - \xi_n - y_n (\mathbf{w}^T \mathbf{x}_n + b)\} - \sum_{n=1}^N \mu_n \xi_n \right\} \\ &= \begin{cases} 0 & \text{if } \forall n, y_n (\mathbf{w}^T \mathbf{x}_n + b) \geq 1 - \xi_n \text{ and } \xi_n \geq 0 \\ \infty & \text{otherwise} \end{cases}. \end{aligned}$$

Dual Form for Soft-SVM

The Soft-SVM's primal optimization problem can be re-formulated as

$$\min_{\mathbf{w}, b, \xi} \left\{ \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{n=1}^N \xi_n + g(\mathbf{w}, b, \xi) \right\}, \text{ i.e.,}$$

$$\min_{\mathbf{w}, b, \xi} \max_{\alpha \geq 0, \mu \geq 0} \left\{ \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{n=1}^N \xi_n + \sum_{n=1}^N \alpha_n \{1 - \xi_n - y_n (\mathbf{w}^T \mathbf{x}_n + b)\} - \sum_{n=1}^N \mu_n \xi_n \right\}.$$

Now suppose that we flip the order of min and max in the above equation. This can only decrease the objective value, and we have

$$\min_{\mathbf{w}, b, \xi} \max_{\alpha \geq 0, \mu \geq 0} \left\{ \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{n=1}^N \xi_n + \sum_{n=1}^N \alpha_n \{1 - \xi_n - y_n (\mathbf{w}^T \mathbf{x}_n + b)\} - \sum_{n=1}^N \mu_n \xi_n \right\}$$

\geq

$$\max_{\alpha \geq 0, \mu \geq 0} \min_{\mathbf{w}, b, \xi} \left\{ \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{n=1}^N \xi_n + \sum_{n=1}^N \alpha_n \{1 - \xi_n - y_n (\mathbf{w}^T \mathbf{x}_n + b)\} - \sum_{n=1}^N \mu_n \xi_n \right\}.$$

Dual Form for Soft-SVM

The inequality (\geq) is called **weak duality**. It turns out that in our case, **strong duality** also holds; namely, the inequality holds with equality. Therefore, the dual problem is

$$\max_{\alpha \geq 0, \mu \geq 0} \min_{\mathbf{w}, b, \xi} \left\{ \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{n=1}^N \xi_n + \sum_{n=1}^N \alpha_n \left\{ 1 - \xi_n - y_n (\mathbf{w}^T \mathbf{x}_n + b) \right\} - \sum_{n=1}^N \mu_n \xi_n \right\}.$$

We can simplify the dual problem by noting that once α and μ are fixed, the optimization problem with respect to \mathbf{w} is unconstrained and the objective is differentiable; thus, at the optimum, the gradient equals zero:

$$\mathbf{w} - \sum_{n=1}^N \alpha_n y_n \mathbf{x}_n = 0 \Rightarrow \mathbf{w} = \sum_{n=1}^N \alpha_n y_n \mathbf{x}_n.$$

Dual Form for Soft-SVM

$\left\{ \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{n=1}^N \xi_n + \sum_{n=1}^N \alpha_n \{1 - \xi_n - y_n (\mathbf{w}^\top \mathbf{x}_n + b)\} - \sum_{n=1}^N \mu_n \xi_n \right\}$ is a linear function of b with coefficient $-\sum_{n=1}^N \alpha_n y_n$, whose minimum value would be $-\infty$ for all α except for the α satisfying

$$\sum_{n=1}^N \alpha_n y_n = 0.$$

$\left\{ \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{n=1}^N \xi_n + \sum_{n=1}^N \alpha_n \{1 - \xi_n - y_n (\mathbf{w}^\top \mathbf{x}_n + b)\} - \sum_{n=1}^N \mu_n \xi_n \right\}$ is also a linear function of ξ_n with coefficient $C - \alpha_n - \mu_n$, whose minimum value would be $-\infty$ for all choices of α_n and μ_n except for the choice of α_n and μ_n satisfying

$$C - \alpha_n - \mu_n = 0.$$

Dual Form for Soft-SVM

The Soft-SVM's dual optimization problem can be finally formulated as

$$\begin{aligned} & \max_{\alpha, \mu \in \mathbb{R}^N} \left\{ -\frac{1}{2} \sum_{n=1}^N \sum_{m=1}^N \alpha_n \alpha_m y_n y_m \mathbf{x}_n^T \mathbf{x}_m + \sum_{n=1}^N \alpha_n \right\} \\ & \text{s.t. } \alpha_n \geq 0 \text{ for } n = 1, 2, \dots, N \text{ and } \sum_{n=1}^N \alpha_n y_n = 0, \\ & \quad \mu_n \geq 0 \text{ and } C - \alpha_n - \mu_n = 0 \text{ for } n = 1, 2, \dots, N. \end{aligned}$$

Equivalently, a concise formulation is

$$\begin{aligned} & \max_{\alpha \in \mathbb{R}^N} \left\{ -\frac{1}{2} \sum_{n=1}^N \sum_{m=1}^N \alpha_n \alpha_m y_n y_m \mathbf{x}_n^T \mathbf{x}_m + \sum_{n=1}^N \alpha_n \right\} \\ & \text{s.t. } C \geq \alpha_n \geq 0 \text{ for } n = 1, 2, \dots, N \text{ and } \sum_{n=1}^N \alpha_n y_n = 0. \end{aligned}$$

Dual Form for Soft-SVM

Given the optimal solution α^* to the Soft-SVM's dual optimization problem, we can recover the optimal solution \mathbf{w}^* and b^* to the primal problem as

$$\mathbf{w}^* = \sum_{n=1}^N \alpha_n^* y_n \mathbf{x}_n,$$
$$b^* = \frac{\sum_{n=1}^N \mathbb{I}(C > \alpha_n^* > 0) \cdot \left(y_n - \sum_{m=1}^N \alpha_m^* y_m \mathbf{x}_m^\top \mathbf{x}_n \right)}{\sum_{n=1}^N \mathbb{I}(C > \alpha_n^* > 0)},$$

where $\mathbb{I}(C > \alpha_n^* > 0)$ is an indicator function, whose value is 1 if $C > \alpha_n^* > 0$, and 0 otherwise.

Dual Form for Soft-SVM

Given the feature vector of a new test sample \mathbf{x} , its label \hat{y} can be predicted as

$$\hat{y} = \begin{cases} +1 & \text{if } \mathbf{w}^{*T}\mathbf{x} + b^* \geq 0 \\ -1 & \text{if } \mathbf{w}^{*T}\mathbf{x} + b^* < 0 \end{cases}.$$

Equivalently,

$$\hat{y} = \begin{cases} +1 & \text{if } \sum_{n=1}^N \alpha_n^* y_n \mathbf{x}_n^T \mathbf{x} + b^* \geq 0 \\ -1 & \text{if } \sum_{n=1}^N \alpha_n^* y_n \mathbf{x}_n^T \mathbf{x} + b^* < 0 \end{cases}.$$

Kernel Methods

Kernel SVM

Recall the Soft-SVM's dual optimization problem formulation

$$\begin{aligned} \max_{\alpha \in \mathbb{R}^N} & \left\{ -\frac{1}{2} \sum_{n=1}^N \sum_{m=1}^N \alpha_n \alpha_m y_n y_m \mathbf{x}_n^T \mathbf{x}_m + \sum_{n=1}^N \alpha_n \right\} \\ \text{s.t. } & C \geq \alpha_n \geq 0 \text{ for } n = 1, 2, \dots, N \text{ and } \sum_{n=1}^N \alpha_n y_n = 0, \end{aligned}$$

where the training samples' feature vectors \mathbf{x}_n could be replaced by the transformed feature vectors $\phi(\mathbf{x}_n)$ through a non-linear basis function $\phi(\cdot)$:

$$\begin{aligned} \max_{\alpha \in \mathbb{R}^N} & \left\{ -\frac{1}{2} \sum_{n=1}^N \sum_{m=1}^N \alpha_n \alpha_m y_n y_m \phi(\mathbf{x}_n)^T \phi(\mathbf{x}_m) + \sum_{n=1}^N \alpha_n \right\} \\ \text{s.t. } & C \geq \alpha_n \geq 0 \text{ for } n = 1, 2, \dots, N \text{ and } \sum_{n=1}^N \alpha_n y_n = 0. \end{aligned}$$

Kernel Trick

The dot product $\phi(\mathbf{x}_n)^T \phi(\mathbf{x}_m)$ can be extended to any [kernel functions](#).

For all \mathbf{x} and \mathbf{x}' in the input space \mathcal{X} , the function $k : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}$ is called a kernel function, if there exists a feature map $\varphi : \mathcal{X} \rightarrow \mathcal{V}$ that satisfies

$$k(\mathbf{x}, \mathbf{x}') = \langle \varphi(\mathbf{x}), \varphi(\mathbf{x}') \rangle_{\mathcal{V}},$$

where \mathcal{V} is a inner product space, and $\langle \cdot, \cdot \rangle_{\mathcal{V}}$ is the inner product operation defined by \mathcal{V} .

An alternative definition can be formulated by the [positive semidefinite \(PSD\)](#) property: For any points $(\mathbf{x}_1, \dots, \mathbf{x}_N)$ in \mathcal{X} , and all choices of n real-valued coefficients (c_1, \dots, c_N) , the function $k : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}$ is a kernel function, if

$$\sum_{n=1}^N \sum_{m=1}^N k(\mathbf{x}_n, \mathbf{x}_m) c_n c_m \geq 0.$$

The spanned matrix, $\mathbf{K} \in \mathbb{R}^N \times \mathbb{R}^N$ with its nm -th entry $K_{nm} = k(\mathbf{x}_n, \mathbf{x}_m)$, is called [Gram matrix](#).

Kernel Trick

Many kernels can be chosen for various application scenarios

- Fisher kernel
- Polynomial kernel
- Radial basis function kernel (RBF)
- String kernels
- Graph kernels

The Radial basis function kernel (RBF) is also called squared-exp or Gaussian kernel, which is formulated as

$$k_{\text{SE}}(\mathbf{x}_n, \mathbf{x}_m) = \sigma_f^2 \exp\left(-\frac{\|\mathbf{x}_n - \mathbf{x}_m\|_2^2}{2\ell^2}\right).$$

Kernel SVM

The kernel version of Soft-SVM's dual optimization problem is

$$\begin{aligned} \max_{\alpha \in \mathbb{R}^N} & \left\{ -\frac{1}{2} \sum_{n=1}^N \sum_{m=1}^N \alpha_n \alpha_m y_n y_m k(\mathbf{x}_n, \mathbf{x}_m) + \sum_{n=1}^N \alpha_n \right\} \\ \text{s.t. } & C \geq \alpha_n \geq 0 \text{ for } n = 1, 2, \dots, N \text{ and } \sum_{n=1}^N \alpha_n y_n = 0. \end{aligned}$$

Kernel SVM

With the optimal solution α^* , for a new test sample with feature vector \mathbf{x} , it labeled can be predicted as

$$\hat{y} = \begin{cases} +1 & \text{if } \sum_{n=1}^N \alpha_n^* y_n k(\mathbf{x}, \mathbf{x}_n) + b^* \geq 0 \\ -1 & \text{if } \sum_{n=1}^N \alpha_n^* y_n k(\mathbf{x}, \mathbf{x}_n) + b^* < 0 \end{cases},$$

where

$$b^* = \frac{\sum_{n=1}^N \mathbb{I}(C > \alpha_n^* > 0) \cdot \left\{ y_n - \sum_{m=1}^N \alpha_m^* y_m k(\mathbf{x}_m, \mathbf{x}_n) \right\}}{\sum_{n=1}^N \mathbb{I}(C > \alpha_n^* > 0)}.$$

References

- Christopher M. Bishop. **Pattern Recognition and Machine Learning**. New York: Springer; 2006 Aug 17. ([Chapter 7.1 Maximum Margin Classifiers](#))
- Shai Shalev-Shwartz, and Shai Ben-David. **Understanding Machine Learning: From Theory to Algorithms**. Cambridge university press, 2014. ([Chapter 15 Support Vector Machines](#))
- https://en.wikipedia.org/wiki/Support_vector_machine

The End