

AI & ML (DI05000051)

Complete Lecture Deck – All 45 Lectures

Milav Dabgar

Lecturer, GP Palanpur

July 22, 2026

Table of Contents I

- 1 Lecture 1.1: Introduction to Artificial Intelligence
- 2 Lecture 1.2: Applications of Artificial Intelligence
- 3 Lecture 1.3: AI Development Tools - Introduction to Python for AI
- 4 Lecture 1.4: AI Development Tools - Overview of NumPy, Pandas, Matplotlib, Scikit-learn
- 5 Lecture 1.5: Data Handling Using Python Libraries (Pandas)
- 6 Lecture 1.6: Data Visualization - Line, Bar, and Pie Charts

Table of Contents II

- 7 Lecture 1.7: Data Visualization - Scatter Plots, Histograms, and Real-World Datasets
- 8 Lecture 1.8: Ethical Considerations in AI
- 9 Lecture 2.1: Introduction to Machine Learning
- 10 Lecture 2.2: Well-Posed Machine Learning Problems
- 11 Lecture 2.3: Types of Machine Learning - Supervised Learning
- 12 Lecture 2.4: Types of Machine Learning - Unsupervised Learning and Comparison

Table of Contents III

- 13 Lecture 2.5: Supervised Learning Algorithms - k-Nearest Neighbours (KNN)
- 14 Lecture 2.6: Supervised Learning Algorithms - Decision Tree
- 15 Lecture 2.7: Supervised Learning Algorithms - Linear Regression
- 16 Lecture 2.8: Model Training and Testing
- 17 Lecture 2.9: Model Evaluation Techniques - Accuracy and Confusion Matrix
- 18 Lecture 2.10: Model Evaluation Techniques - Precision and Recall

Table of Contents IV

- 19 Lecture 3.1: Introduction to Neural Networks - Motivation and Biological Inspiration
- 20 Lecture 3.2: Applications of Neural Networks
- 21 Lecture 3.3: Structure of Artificial Neural Networks - Layers
- 22 Lecture 3.4: Structure of Artificial Neural Networks - Working, Weights and Bias
- 23 Lecture 3.5: Activation Functions - Purpose and Sigmoid Function
- 24 Lecture 3.6: Activation Functions - ReLU and Tanh

Table of Contents V

- 25 Lecture 3.7: Neural Network Architectures - Single Layer Neural Network
- 26 Lecture 3.8: Neural Network Architectures - Multi-Layer Neural Network
- 27 Lecture 3.9: Building a Simple Neural Network Model with Python Libraries
- 28 Lecture 3.10: Building a Simple Neural Network Model - Iris and Digit Recognition Datasets
- 29 Lecture 4.1: Introduction to Natural Language Processing
- 30 Lecture 4.2: NLP Processing Pipeline

Table of Contents VI

- 31 Lecture 4.3: Text Preprocessing - Introduction and Tokenization
- 32 Lecture 4.4: Text Preprocessing - Stop Word Removal, Stemming, and Lemmatization
- 33 Lecture 4.5: Text Preprocessing - POS Tagging and NLTK
- 34 Lecture 4.6: Text Representation - Bag of Words and Term Frequency
- 35 Lecture 4.7: Text Representation - TF-IDF
- 36 Lecture 4.8: Simple NLP Applications - Sentiment and Word Frequency Analysis

Table of Contents VII

- 37 Lecture 4.9: Simple NLP Applications - Keyword Extraction
- 38 Lecture 5.1: Emerging Trends in Artificial Intelligence
- 39 Lecture 5.2: Introduction to Generative AI
- 40 Lecture 5.3: Large Language Models (LLMs) - Introduction
- 41 Lecture 5.4: Large Language Models (LLMs) - Capabilities and Applications
- 42 Lecture 5.5: Prompt Engineering - Fundamentals and Basic Techniques

Table of Contents VIII

- 43 Lecture 5.6: Prompt Engineering - Types of Prompts
- 44 Lecture 5.7: Ethical and Responsible AI - Bias and Privacy
- 45 Lecture 5.8: Ethical and Responsible AI - Misuse and Course Recap

Lecture 1.1: Introduction to Artificial Intelligence

DI05000051 - AI & ML

Milav Dabgar

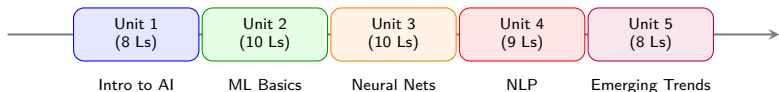
Lecturer, GP Palanpur

July 22, 2026

Course Overview

Course Overview & Today's Agenda

- Course spans 5 units across 15 weeks (45 lectures, 3 lectures/week)
- Unit 1: Introduction to AI (8) - Unit 2: ML Fundamentals (10) - Unit 3: Neural Networks (10)
- Unit 4: Natural Language Processing (9) - Unit 5: Emerging Trends in AI (8)
- Today: What is Artificial Intelligence, and why does it matter?



1.1 Introduction to Artificial Intelligence

Learning Objectives

- Define AI and explain what makes a system 'intelligent'
- Understand why AI matters today
- Separate the terms AI and ML clearly
- Connect AI theory to tools students already use

What is Artificial Intelligence?

- AI = the branch of computer science that builds systems able to perform tasks that normally require human intelligence
- Such tasks include reasoning, learning, perception, understanding language, and decision-making
- An AI system 'appears' intelligent because it can adapt its output based on input, not just follow one fixed rule
- AI is a broad umbrella field - Machine Learning is one major way to build AI



Goals of Artificial Intelligence

- Automate tasks that are repetitive, dangerous, or too large-scale for humans alone
- Enable machines to reason and make decisions from data
- Allow computers to perceive the world (see, hear, read text)
- Support human decision-making rather than fully replace it in most current applications

Automation

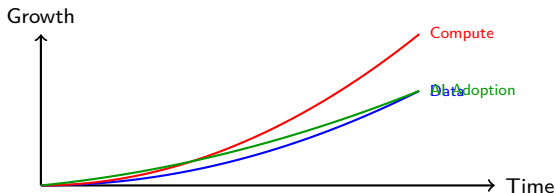
Reasoning

Perception

Decision-Support

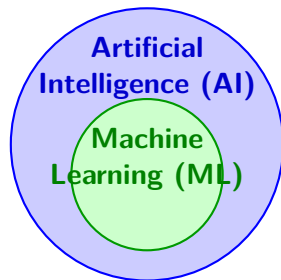
Why is AI Important Today?

- Massive growth in available data (from phones, sensors, the internet) that AI can learn from
- Increasing computing power (GPUs, cloud computing) makes training AI systems practical
- AI improves efficiency, accuracy, and personalisation across industries
- Diploma engineers with AI/ML skills are in high demand across almost every sector



Artificial Intelligence vs Machine Learning

- **AI**: the broad goal - building systems that behave intelligently (by any method, including hand-written rules)
- **ML**: a specific approach to AI - systems that learn patterns automatically from data instead of being explicitly programmed
- Every ML system is a form of AI, but not every AI system uses ML
- This course covers AI broadly in Unit 1, then focuses on ML techniques from Unit 2 onward



AI in Daily Life

AI in Daily Life: Google Maps

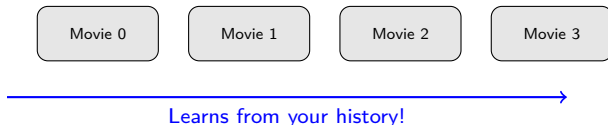
- Google Maps uses AI to predict traffic and suggest the fastest route
- It learns from historical and live location data from millions of users
- This is a real-time decision-making AI application most students already use daily



AI in Daily Life: Netflix Recommendations

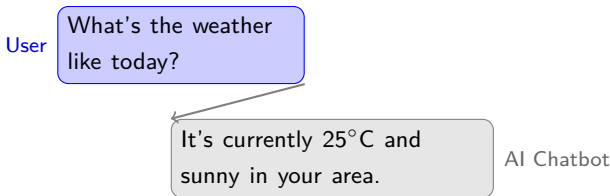
- Netflix (and YouTube, Spotify) recommend content based on your past viewing/listening behaviour
- The system learns your preferences from patterns in what you watch, skip, or rate
- This is a recommendation system - a common ML application covered conceptually across the course

Recommended for you



AI in Daily Life: Chatbots

- Chatbots (customer support bots, virtual assistants) understand typed or spoken questions and generate replies
- Modern chatbots use Natural Language Processing (NLP) to understand human language
- This is a preview of Unit 4 (NLP) and Unit 5's discussion of modern AI chat systems (LLMs)



Quick Check: Test Your Understanding

Questions for the Class

- **Q:** In one sentence, define Artificial Intelligence
- **Q:** Is Machine Learning the same as AI, or a part of AI?
- **Q:** Name one AI application you personally used this week

Summary & Preview

Key Takeaways

- AI is the broad field of building systems that perform tasks requiring human-like intelligence
- ML is one specific, learning-based approach to achieving AI - not a synonym for it
- AI is already part of daily life through maps, recommendations, and chatbots
- AI's importance is driven by growing data, growing compute power, and clear industry demand

Next Lecture Preview

Coming Up: Lecture 2

Applications of Artificial Intelligence - a tour of AI across healthcare, finance, education, manufacturing, and social media.

Real-World AI Applications →

Questions?

Lecture 1.2: Applications of Artificial Intelligence

DI05000051 - AI & ML

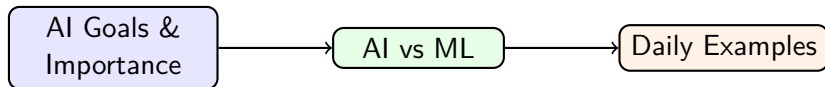
Milav Dabgar

Lecturer, GP Palanpur

July 22, 2026

Recap: Lecture 1

- Lecture 1 defined AI, explained its goals and importance, distinguished AI from ML, and showed daily-life AI examples (Maps, Netflix, chatbots).



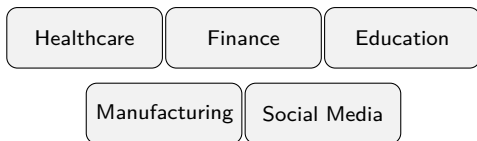
Today's Agenda

- 1 Part 1: AI in healthcare and finance
- 2 Part 2: AI in education and manufacturing/automation
- 3 Part 3: AI in social media platforms

AI in Healthcare and Finance

Why Look at Applications Industry by Industry?

- Recall Lecture 1: AI is a broad field - its real impact is best seen through concrete industry examples
- Every industry generates data; AI turns that data into predictions, automation, or personalisation
- Seeing five different industries shows AI is not limited to any one domain



AI in Healthcare

- Medical image analysis: AI helps detect abnormalities in X-rays, MRIs, and CT scans
- Disease prediction: models estimate risk of conditions like diabetes or heart disease from patient data
- Robotic-assisted surgery and AI-powered diagnostic support tools
- Drug discovery: AI accelerates identifying promising chemical compounds



- Fraud detection: models flag unusual transaction patterns in real time
- Algorithmic trading: AI systems analyse markets and execute trades at high speed
- Credit scoring: predicting loan repayment risk from applicant data
- Chatbots for customer service in banking apps

Fraud Alert Shield

Transaction flagged as suspicious!

AI in Education and Manufacturing/Automation

AI in Education

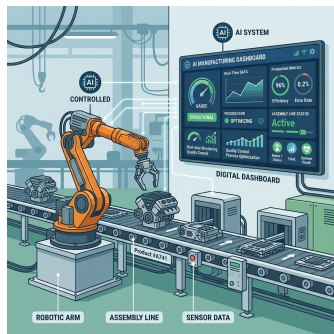
- Personalised learning platforms that adapt content difficulty to each student's pace
- Automated grading of objective and even some subjective assessments
- AI tutoring/chat assistants that answer student questions on demand
- Predicting students at risk of falling behind, so instructors can intervene early

Adaptive Quiz Interface

Next Question Ad-
justed for Difficulty...

AI in Manufacturing and Automation

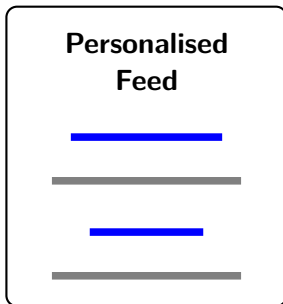
- Predictive maintenance: AI predicts machine failures before they happen, from sensor data
- Quality control: computer vision inspects products on the assembly line for defects
- Robotic automation: AI-guided robotic arms handle assembly, sorting, and packaging
- Supply chain optimisation: demand forecasting and inventory management



AI in Social Media Platforms

AI in Social Media Platforms

- Content recommendation feeds (what you see is ranked by an AI model)
- Automatic content moderation: detecting spam, hate speech, or explicit content
- Face recognition for photo tagging
- Ad targeting: matching advertisements to likely-interested users

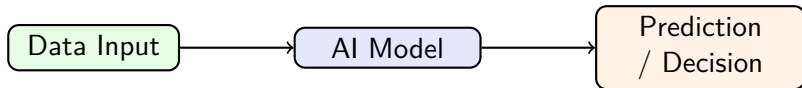


Applications at a Glance

Industry	Key AI Application
Healthcare	Diagnosis support, drug discovery
Finance	Fraud detection, algorithmic trading
Education	Personalised learning, automated grading
Manufacturing	Predictive maintenance, robotics
Social Media	Recommendation feeds, content moderation

A Common Thread Across Industries

- In every case: AI takes in data (images, transactions, text, sensor readings) and outputs a prediction or decision
- This 'data in, prediction out' pattern is the foundation of Machine Learning, covered starting Unit 2
- Ethical questions (bias, privacy) arise in every one of these applications - covered in Lecture 8



Quick Check: Match Industry to Application

- ① Q: Which industry uses AI for predictive maintenance?
- ② Q: Name one AI application in education
- ③ Q: Which industry example is most similar to Lecture 1's Netflix example?

Self-Check

Think about the answers based on the table we just reviewed!

Summary & Next Steps

Key Takeaways

- AI applications span healthcare, finance, education, manufacturing/automation, and social media
- Across all industries, the common pattern is: data in, AI model, prediction/decision out
- AI is already deeply embedded in modern industry - not just a future possibility

- **Lecture 3: AI Development Tools** - why Python has become the leading programming language for AI and Machine Learning

Coming Up Next:
Python and its ecosystem for AI/ML!

Questions?

Lecture 1.3: AI Development Tools - Introduction to Python for AI

DI05000051 - AI & ML

Milav Dabgar

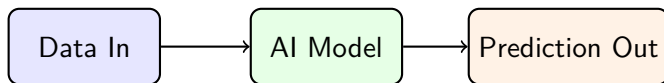
Lecturer, GP Palanpur

July 22, 2026

Recap & Agenda

Recap: Lecture 2

- Lecture 2 surveyed AI applications across five industries, all following the same data-to-prediction pattern.



Today's Agenda

- ① **Part 1:** Python as a general-purpose programming language
- ② **Part 2:** Why Python specifically suits AI/ML work
- ③ **Part 3:** How this course will use Python

Part 1: What is Python?

Why is Python Easy to Learn?

- Minimal boilerplate: a working program can be just one or two lines
- Indentation-based syntax forces readable, consistent code formatting
- Dynamic typing: no need to declare variable types explicitly
- Large, friendly community and abundant free learning resources

Python vs Verbose Language (Java)

Python

```
print("Hello AI")
```

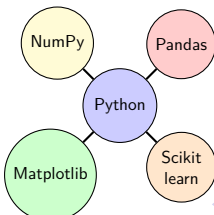
Java

```
class Main {  
    public static void main(String[] args) {  
        System.out.println("Hello AI");  
    }  
}
```

Part 2: Why Python for AI?

Why Python for AI and Machine Learning Specifically?

- Rich ecosystem of ready-made AI/ML libraries (NumPy, Pandas, Matplotlib, Scikit-learn, and more) - covered next lecture
- Readable syntax lets researchers and engineers express complex ML ideas concisely
- Strong integration with data formats (CSV, Excel, JSON, databases) commonly used in ML pipelines
- Widely adopted in both academia and industry - most AI research code and courses use Python



Python vs Traditional Languages

- For learning and rapidly building AI/ML models, Python's productivity advantage outweighs its speed disadvantage.

Language	AI/ML Suitability
Python	Concise syntax, huge ML library ecosystem, slower raw execution speed.
C/C++	Faster execution, but verbose and low-level for expressing ML ideas.
Java	Strong for large enterprise systems, but a smaller ML-specific library ecosystem.

A Simple Python Program

- Such simple building blocks (variables, functions, loops) are what larger AI programs are built from.

Simple Python Snippets

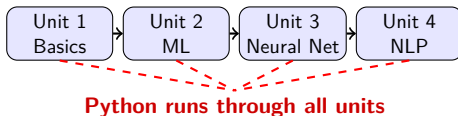
```
# Outputs text to the screen  
print("Hello , AI & ML!")
```

```
# Basic variables and arithmetic  
x = 10  
y = 20  
print(x + y)
```

Part 3: Python in This Course

Where Python Fits in This Course

- From Lecture 4 onward, we introduce specific libraries: NumPy, Pandas, Matplotlib, Scikit-learn
- Unit 2's ML algorithms (KNN, Decision Tree, Linear Regression) will be described using Scikit-learn-style workflows
- Unit 3's neural network examples and Unit 4's NLP examples also rely on Python libraries
- You are not expected to be a professional programmer - just comfortable reading and reasoning about simple Python/ML code



Summary & Next Steps

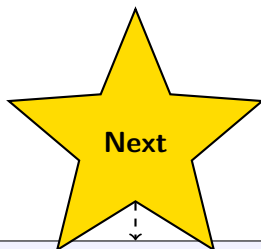
Quick Check: Python Fundamentals

Questions

- Q:** Is Python a general-purpose language or an AI-only language?
- Q:** Name one reason Python suits AI/ML development.
- Q:** What will next lecture introduce?

Key Takeaways

- Python is a general-purpose language whose readability and library ecosystem make it the AI/ML standard
- Simplicity of syntax lets learners focus on ML concepts rather than programming complexity
- This course will use Python informally throughout Units 1-4 to illustrate concepts
- Its dominance in AI/ML comes from its rich library ecosystem and concise syntax, not built-in AI features



Lecture 4: AI Development Tools Part B

Overview of NumPy, Pandas, Matplotlib,
and Scikit-learn, one library at a time

Lecture 4 completes topic 1.3 by introducing the four specific libraries that give Python its AI/ML power.

Questions?

Lecture 1.4: AI Development Tools - Overview of NumPy, Pandas, Matplotlib, Scikit-learn

DI05000051 - AI & ML

Milav Dabgar

Lecturer, GP Palanpur

July 22, 2026

Recap & Agenda

Recap: Lecture 3

- **Previous Lecture:** Lecture 3 explained why Python's general-purpose readability and ecosystem make it the language of choice for AI/ML.
- Lecture 3 covered why Python (readable syntax, huge ecosystem) became the standard language for AI/ML development.

Today's Agenda

- ① Part 1: NumPy - numerical computing
- ② Part 2: Pandas - data handling
- ③ Part 3: Matplotlib - visualization
- ④ Part 4: Scikit-learn - machine learning

1.3 AI Development Tools and Libraries

Four Libraries, Four Jobs

- NumPy: fast numerical computing with arrays and matrices
- Pandas: loading, cleaning, and organising tabular data
- Matplotlib: visualising data as charts and graphs
- Scikit-learn: building and evaluating machine learning models



NumPy: Numerical Computing

- NumPy = 'Numerical Python' - provides fast array and matrix operations
- Core object: the ndarray (n-dimensional array), far faster than plain Python lists for numeric work
- Supports operations like addition, multiplication, and statistics on entire arrays at once
- Almost every other ML library (Pandas, Scikit-learn) is built on top of NumPy arrays

1	2	3	$\xrightarrow{\times 2}$	2	4	6
4	5	6		8	10	12
7	8	9		14	16	18

Pandas: Data Handling

- Pandas provides the DataFrame - a table-like structure with labelled rows and columns (like a spreadsheet)
- Used to import data (from CSV/Excel files), clean it, and explore it
- Common operations: viewing rows, selecting columns, computing summary statistics
- This is the exact library used in Lecture 5's Data Handling topic

Index	Name	Age	Salary
0	Alice	28	70000
1	Bob	34	85000
2	Charlie	25	62000

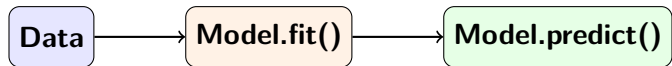
Matplotlib: Data Visualization

- Matplotlib is Python's foundational plotting library
- Creates line graphs, bar charts, pie charts, scatter plots, and histograms
- Helps humans spot patterns, trends, and outliers in data before modelling it
- This library gets two dedicated lectures (Lecture 6 and 7) covering each chart type



Scikit-learn: Machine Learning

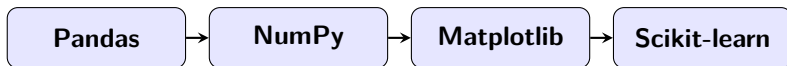
- Scikit-learn (sklearn) provides ready-to-use implementations of ML algorithms
- Includes classification algorithms (KNN, Decision Tree), regression (Linear Regression), and evaluation tools
- Consistent workflow across algorithms: fit the model on training data, then predict on new data
- Used throughout Unit 2 (classical ML) and referenced again in Unit 3



Summary & Project Workflow

How the Four Libraries Work Together

- Step 1 - Pandas: load and clean the raw dataset
- Step 2 - NumPy: perform underlying numerical computations
- Step 3 - Matplotlib: visualise the data to understand it before modelling
- Step 4 - Scikit-learn: train and evaluate a machine learning model on the data



A Day in the Life of an ML Project

- A data scientist predicting house prices might: load housing data with Pandas, compute statistics with NumPy, plot price trends with Matplotlib, and train a Linear Regression model with Scikit-learn
- This single example touches all four libraries in one realistic workflow



Quick Check: Library Roles

- **Q:** Which library would you use to load a CSV file?
- **Q:** Which library creates bar charts and histograms?
- **Q:** Which library actually trains a KNN or Linear Regression model?

Key Takeaways

- NumPy, Pandas, Matplotlib, and Scikit-learn form the core Python AI/ML toolkit
- Each library has a distinct role: numerical computation, data handling, visualization, and modelling respectively
- A typical ML project workflow moves through all four libraries in sequence
- **Overall:** A typical ML project workflow moves through all four libraries in sequence

- **Lecture 5: Data Handling Using Python Libraries** - importing datasets and performing basic operations with Pandas
- **Preview:** Lecture 5 dives deeper into Pandas specifically, covering how to import real datasets and perform basic data operations - the first hands-on step of any ML project.

Questions?

Lecture 1.5: Data Handling Using Python Libraries (Pandas)

DI05000051 - AI & ML

Milav Dabgar

Lecturer, GP Palanpur

July 22, 2026

Recap: Lecture 4

- Lecture 4 introduced all four core AI/ML libraries; Pandas was identified as the tool for data handling.

Lecture 4: NumPy, Pandas, Matplotlib, Scikit-learn

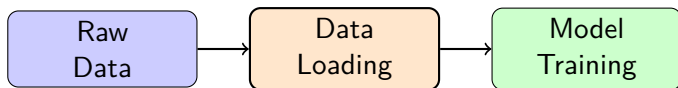


Pandas for Data Handling

Introduction & Data Pipeline

Why Data Handling Comes First

- Recall Lecture 2: every AI application follows a 'data in, prediction out' pattern
- Before any model can be trained, the data must be loaded and understood
- This is the first practical, hands-on step in any real ML project



The Pandas DataFrame

- A DataFrame is Pandas' core data structure - rows and labelled columns, just like a spreadsheet
- Each column can hold a different type of data (numbers, text, dates)
- Most real-world datasets (customer records, sensor logs, survey results) map naturally onto a DataFrame

Name	Age	Score
Alice	20	85
Bob	22	90
Charlie	21	88

Importing Datasets

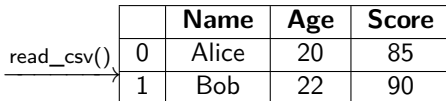
Importing a Dataset from CSV

- CSV (Comma-Separated Values) is the most common simple data file format
- In Pandas: `df = pandas.read_csv("filename.csv")`
- This single line loads the entire file into a DataFrame ready for exploration

filename.csv

```
Name, Age, Score  
Alice, 20, 85  
Bob, 22, 90
```

DataFrame



	Name	Age	Score
0	Alice	20	85
1	Bob	22	90

Importing a Dataset from Excel

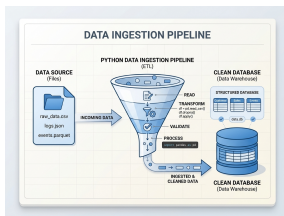
- Excel files (.xlsx) are common in business and academic settings
- In Pandas: `df = pandas.read_excel("filename.xlsx")`
- Very similar syntax to `read_csv` - Pandas hides the file-format complexity from the user

```
import pandas as pd
df = pd.read_excel("data.xlsx")
```

Viewing and Selecting Data

Viewing Data: head(), tail(), shape

- `df.head()` shows the first 5 rows - a quick sanity check after loading
- `df.tail()` shows the last 5 rows
- `df.shape` reports (number of rows, number of columns)
- These are almost always the first commands run immediately after loading any dataset



Selecting Columns

- Select one column: `df["column_name"]` → returns that column's values
- Select multiple columns: `df[["col1", "col2"]]` → returns a smaller DataFrame
- Useful when a dataset has many columns but only a few are relevant to the current task

Example

```
# Select single column
ages = df["Age"]

# Select multiple columns
subset = df[["Name", "Score"]]
```


Basic Statistical Summary

Basic Statistical Summary

- `df.describe()` generates count, mean, standard deviation, minimum, maximum, and quartiles for numeric columns
- Quickly reveals the scale and spread of the data (e.g., are values 0-1 or 0-1,000,000?)
- Helps spot obvious data issues, such as impossible values or missing data

The diagram illustrates the process of cleaning and structuring data. On the left, a 'BEFORE: MESSY SPREADSHEET' shows an 'UNORGANIZED DATA' table with various issues: missing dates, inconsistent currency symbols, duplicate rows, incorrect data, blank cells, and mismatched values. An arrow labeled 'CLEANING & STRUCTURING PROCESS' points to the right, where the 'AFTER: NEAT DATA TABLE' is shown. This table is an 'ORGANIZED DATA TABLE' with a consistent structure: ID, Name, Date, Department, and Revenue (\$).

BEFORE: MESSY SPREADSHEET

UNORGANIZED DATA

UNORGANIZED DATA				
John D.	2024-03-15	Sales: 550		
	Missing date! No date		Maria: 690	
		€		
John D.	550	03/15	\$	
Anna S.	3/10	Support		
Anna	120			
Anna		Mar-10	€	
		Blank cell	\$	
		Marketing	\$	
ID 101	2024/03/12	EUR 300	€	
		mismatched values		

AFTER: NEAT DATA TABLE

ORGANIZED DATA TABLE

ID	Name	Date	Department	Revenue (\$)
101	John Doe	2024-03-15	Sales	550.00
	Anna Smith	2024-03-10	Support	120.00
	Ursula Chen	2024-03-12	Marketing	300.00
103	Maria Garcia	2024-03-18	HR	450.00
104	Maria Garcia	2024-03-19	HR	450.00

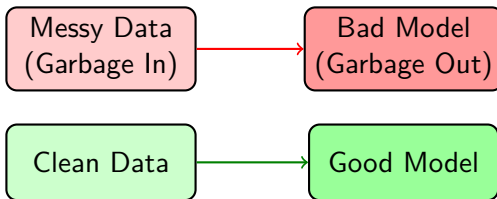
Worked Mini-Example: Student Marks Dataset

- Load a small CSV of student marks: `df = pandas.read_csv("marks.csv")`
- View it: `df.head()` shows student names and their marks in different subjects
- Select one subject: `df["Maths"]`
- Summarise: `df.describe()` reveals average, highest, and lowest marks

```
import pandas as pd
df = pd.read_csv("marks.csv")
print(df.head())
print(df["Maths"].describe())
```

Why This Matters for Machine Learning

- Every ML algorithm in Unit 2 (KNN, Decision Tree, Linear Regression) expects data already loaded and organised this way
- Poor data handling (wrong columns, unnoticed missing values) leads to poor or misleading models
- 'Garbage in, garbage out' - the quality of data handling directly affects model quality



Summary & Next Steps

Quick Check: Pandas Operations

Self-Assessment Questions

- 1 Which function loads a CSV file into a DataFrame?
- 2 How would you view just the first 5 rows of a dataset?
- 3 Which function gives you mean, min, and max of numeric columns?

Key Takeaways

- Pandas' DataFrame is the standard way to represent tabular data in Python
- Datasets can be imported from CSV or Excel with a single `read_csv()/read_excel()` call
- `head()`, column selection, and `describe()` are the essential first steps of exploring any dataset

Solid data handling is the essential first step before any modeling.

- **Lecture 6: Data Visualization Part A** - turning raw data into line graphs, bar charts, and pie charts using Matplotlib

Next up: Visualising the data we just loaded!

Questions?

Lecture 1.6: Data Visualization - Line, Bar, and Pie Charts

DI05000051 - AI & ML

Milav Dabgar

Lecturer, GP Palanpur

July 22, 2026

Recap: Lecture 5

- Lecture 5 covered importing CSV/Excel datasets with Pandas and exploring them via `head()`, column selection, and `describe()`.
- Essential prerequisite before visualising anything.

Introduction

Today's Agenda

- 1 Part 1: Line graphs
- 2 Part 2: Bar charts
- 3 Part 3: Pie charts

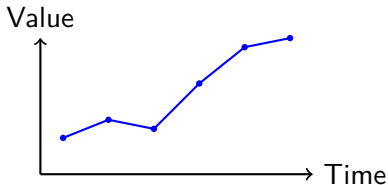
Why Visualize Data?

- Recall Lecture 4: Matplotlib is the library for turning data into charts
- Numbers in a table (from Pandas' `describe()`, Lecture 5) are hard to interpret at a glance
- A well-chosen chart reveals trends, comparisons, and proportions instantly
- Visualization is a bridge between raw data and human understanding, before any ML model is built

Line Graphs

Line Graphs

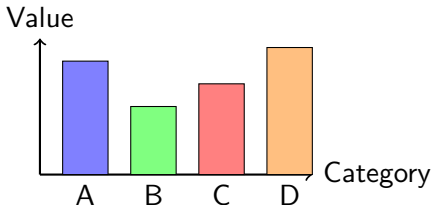
- Best for showing how a value changes over a continuous variable, especially time
- In Matplotlib: `matplotlib.pyplot.plot(x, y)` then `.show()`
- Example: plotting daily temperature over a month, or a student's marks across semesters



Bar Charts

Bar Charts

- Best for comparing quantities across distinct categories
- In Matplotlib: `matplotlib.pyplot.bar(categories, values)`
- Example: comparing average marks across different subjects, or sales across different regions



Pie Charts

Pie Charts

- Best for showing proportions of a whole (parts add up to 100%)
- In Matplotlib: `matplotlib.pyplot.pie(values, labels=categories)`
- Example: showing the percentage share of each product category in total sales

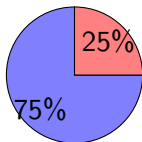


Chart Comparison & Application

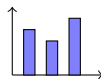
Line vs Bar vs Pie: When to Use Which

- Line graph: trend of one value over a continuous sequence (e.g., time)
- Bar chart: comparison of values across separate categories
- Pie chart: proportion of each category within a whole
- Choosing the wrong chart type is a common real-world data visualization mistake

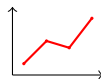
Chart Type	Best Used When	Example
Line Graph	Trends over continuous variables (time)	Temp. over a month
Bar Chart	Comparing distinct categories	Marks across subjects
Pie Chart	Proportions of a whole (adds to 100%)	Market share

Worked Example: Student Marks

- **Bar chart:** compare one student's marks across 5 subjects
- **Line graph:** track one student's average marks across 4 semesters
- **Pie chart:** show the proportion of total marks contributed by each subject



Bar: Subjects



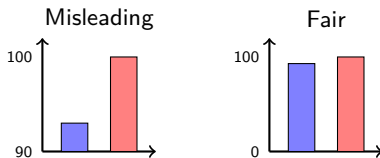
Line: Semesters



Pie: Contribution

Reading a Chart Critically

- Always check axis labels and units before interpreting a chart
- Watch for misleading scales (e.g., a bar chart not starting at zero can exaggerate differences)
- A chart is only as trustworthy as the data and scale behind it



Quick Check: Choosing Chart Types

- Q: You want to show how website visits changed each day over 30 days. Which chart?
- Q: You want to show the market share of 4 companies. Which chart?
- Q: You want to compare rainfall across 6 cities. Which chart?

Summary

Key Takeaways

- Line graphs show trends over a continuous sequence such as time
- Bar charts compare quantities across distinct categories
- Pie charts show proportions of a whole
- Choosing the right chart type is as important as knowing the plotting syntax

Next Lecture Preview

- Lecture 7: Data Visualization Part B - scatter plots, histograms, and applying charts to real-life or hypothetical datasets

Questions?

Lecture 1.7: Data Visualization - Scatter Plots, Histograms, and Real-World Datasets

DI05000051 - AI & ML

Milav Dabgar

Lecturer, GP Palanpur

July 22, 2026

Recap: Lecture 6

- Lecture 6 covered line graphs, bar charts, and pie charts, and how to choose the right chart type for a question.

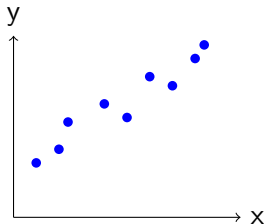
Today's Agenda

- Part 1: Scatter plots
- Part 2: Histograms
- Part 3: Applying all 5 chart types to real datasets

Scatter Plots

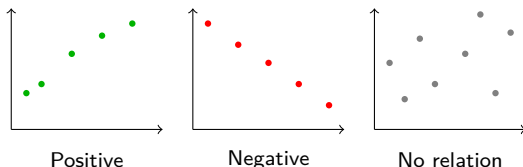
Scatter Plots

- Best for showing the relationship between two numeric variables
- In Matplotlib: `matplotlib.pyplot.scatter(x, y)`
- Each point represents one observation, plotted by its two variable values
- Example: plotting hours studied (x) vs marks obtained (y) for a class of students



Interpreting Scatter Plots

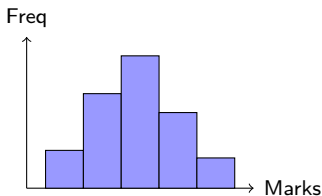
- Points trending upward together suggest a positive relationship (as one increases, so does the other)
- Points trending in opposite directions suggest a negative relationship
- A scattered cloud with no clear pattern suggests little or no relationship



Histograms

Histograms

- Best for showing the distribution (spread and frequency) of a single numeric variable
- In Matplotlib: `matplotlib.pyplot.hist(values, bins=n)`
- Data is grouped into 'bins' (ranges), and bar height shows how many values fall in each bin
- Example: distribution of marks scored by an entire class in an exam



Scatter Plot vs Histogram

- Both differ from bar charts, which compare distinct, non-numeric categories

Scatter Plot

Relationship between TWO numeric variables (x and y)



Histogram

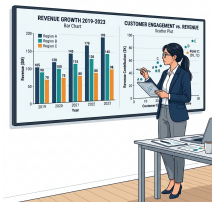
Distribution of ONE numeric variable across its range



Applying 5 Chart Types

The Complete Matplotlib Toolkit: 5 Chart Types

- **Line graph:** trend over a sequence
- **Bar chart:** compare categories
- **Pie chart:** proportions of a whole
- **Scatter plot:** relationship between two numeric variables
- **Histogram:** distribution of one numeric variable



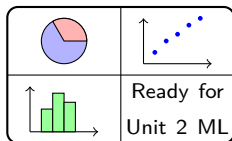
Real-Life Dataset Example: Weather Data

- **Line graph:** daily temperature over one month
- **Bar chart:** total rainfall compared across 4 cities
- **Histogram:** distribution of daily temperatures over a year (how many days were hot/mild/cold)
- **Scatter plot:** relationship between humidity and temperature



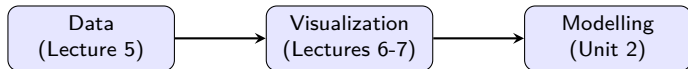
Hypothetical Dataset Example: Class Performance

- **Pie chart:** proportion of students scoring grade A, B, C, D
- **Scatter plot:** hours of study vs marks obtained
- **Histogram:** distribution of marks across the whole class
- This dataset will reappear conceptually in Unit 2



Visualization Before Modelling

- Recall Lecture 5: good data handling comes before modelling
- Similarly, good visualization comes before modelling - it helps decide which ML approach might work
- A clear positive-trend scatter plot suggests regression may work well (Unit 2)
- A clear grouping in a scatter plot suggests clustering may be relevant (Unit 2)



Review

Quick Check: Scatter vs Histogram vs Bar

- ① **Q:** You want to see if taller people tend to weigh more. Which chart?
- ② **Q:** You want to see how exam scores are distributed across a class. Which chart?
- ③ **Q:** You want to compare number of students choosing 4 different elective subjects. Which chart?

Key Takeaways

- Scatter plots reveal relationships between two numeric variables; histograms reveal the distribution of one
- Matplotlib's five core chart types (line, bar, pie, scatter, histogram) each answer a different kind of question about data
- Visualizing real or hypothetical datasets is a critical step performed before choosing and training an ML model in Unit 2

- Lecture 8: Ethical Considerations in AI - bias, privacy, and responsible use, closing out Unit 1



Unit 1 Finale!

Questions?

Lecture 1.8: Ethical Considerations in AI

DI05000051 - AI & ML

Milav Dabgar

Lecturer, GP Palanpur

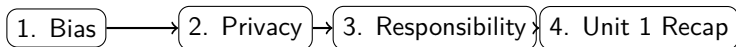
July 22, 2026

Recap: Lecture 7

- Lecture 7 completed the Matplotlib visualization toolkit and applied it to real-life and hypothetical datasets, closing the loop into Unit 2's modelling.

Today's Agenda

- Part 1: Bias in AI systems
- Part 2: Privacy issues
- Part 3: Responsible use of AI
- Part 4: Unit 1 recap



Part 1: Bias in AI systems

Why Ethics Matters in AI

- Recall Lecture 2: AI now touches healthcare, finance, education, manufacturing, and social media - decisions with real human impact
- AI systems learn from data created by humans, and human data can carry human biases
- As AI's reach grows, so does its responsibility to be fair, private, and safe



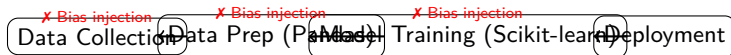
What is Bias in AI Systems?

- Bias occurs when an AI system produces systematically unfair outcomes for certain groups
- Common source: training data that under-represents or stereotypes certain groups
- Example: a hiring AI trained mostly on past resumes from one gender may unfairly favour that gender
- Example: a facial recognition system trained mostly on one skin tone performs worse on others



Where Does Bias Come From?

- Biased or unrepresentative training data (recall Lecture 5: data quality drives model quality)
- Biased labelling by humans when preparing training data
- Biased choice of which features/columns the model is trained on
- Bias can enter at any stage of the data-to-model pipeline discussed since Lecture 4



Consequences of Biased AI

- Unfair decisions in hiring, lending, or law enforcement applications
- Loss of public trust in AI systems and the organisations deploying them
- Real harm to individuals or groups who are systematically disadvantaged

Jobs/Hiring

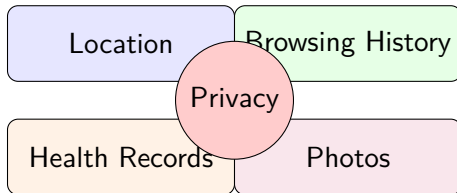
Finance/Loans

Law/Justice

Part 2: Privacy issues

Privacy Issues in AI

- AI systems often require large amounts of personal data (location, browsing history, health records, photos)
- Recall Lecture 1's Google Maps example - it uses your location data to make predictions
- Risk: data can be misused, leaked, or used beyond what a user consented to
- Risk: AI can sometimes infer sensitive information (e.g., health conditions) indirectly from unrelated data



Part 3: Responsible use of AI

Responsible Use of AI

- Use representative, diverse training data to reduce bias
- Be transparent about what data is collected and how AI decisions are made
- Keep a human involved in high-stakes decisions rather than fully automating them
- Continuously test AI systems for fairness and unintended harm after deployment

✓ Diverse
Data

✓ Trans-
parency

✓ Human
Oversight

✓ Mon-
itoring

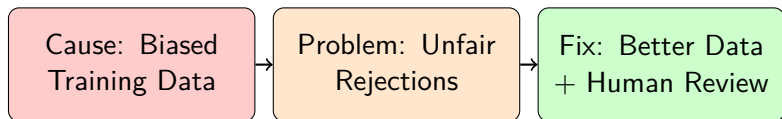
Bias vs Privacy vs Responsible Use

Concern	Core Question	Example
Bias	Does the AI treat all groups fairly?	Hiring AI favouring one gender
Privacy	Is personal data collected and used appropriately?	Medical data leaked
Responsible Use	Is the AI deployed thoughtfully, with human oversight?	Human reviewing border cases

All three are interconnected parts of building trustworthy AI.

Case Example: A Flawed Loan Approval AI

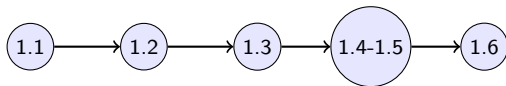
- An AI model trained mainly on past approved loans from one demographic
- Result: the model unfairly rejects qualified applicants from under-represented groups
- Fix: retrain with more representative data, and add human review for borderline/rejected cases



Part 4: Unit 1 recap

Unit 1 Recap: The Foundation We've Built

- 1.1: Defined AI and distinguished it from ML
- 1.2: Toured AI applications across 5 industries
- 1.3: Introduced Python and its 4 core AI/ML libraries
- 1.4-1.5: Practiced data handling (Pandas) and visualization (Matplotlib)
- 1.6: Considered the ethical responsibilities that come with building AI



Key Takeaways

- Bias in AI mainly stems from unrepresentative training data and can produce systematically unfair outcomes
- AI's reliance on personal data creates real privacy risks that must be actively managed
- Responsible AI requires diverse data, transparency, human oversight, and continuous monitoring
- Unit 1 established AI foundations, tools, and ethics - Unit 2 now builds the technical core of Machine Learning

Next Lecture Preview

- Lecture 9: Introduction to Machine Learning - Unit 2 begins with what ML actually is and how it differs from traditional programming

Next: Introduction to Machine Learning (Unit 2)

Questions?

Lecture 2.1: Introduction to Machine Learning

DI05000051 - AI & ML

Milav Dabgar

Lecturer, GP Palanpur

July 22, 2026

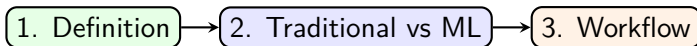
Recap: Lecture 8

- Lecture 8 closed Unit 1 with AI ethics - bias, privacy, and responsible use - and a full Unit 1 recap.

Foundation: AI Concepts and Python Tools

Today's Agenda

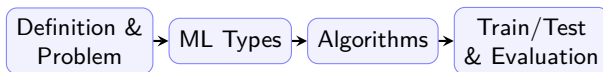
- ① **Part 1:** Definition of Machine Learning
- ② **Part 2:** Traditional programming vs ML
- ③ **Part 3:** The basic ML workflow



Definition of Machine Learning

Welcome to Unit 2: Machine Learning Fundamentals

- Recall Lecture 1: ML is one specific approach to achieving AI - systems that learn from data
- Unit 2 (10 lectures) covers ML's definition, well-posed problems, types, algorithms, training/testing, and evaluation
- This is the technical core the rest of the course (Neural Networks, NLP) builds directly upon

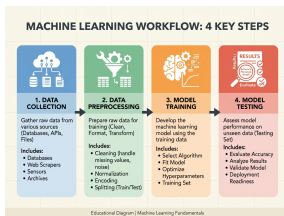


What is Machine Learning?

Machine Learning (ML)

A system's ability to improve its performance on a task by learning from data, without being explicitly programmed for every rule.

- Instead of a programmer writing exact rules, the system finds patterns in example data on its own
- The 'learning' happens during a training process, using historical/example data

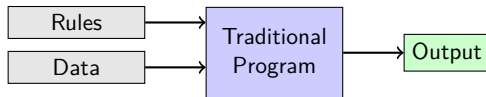


Traditional programming vs ML

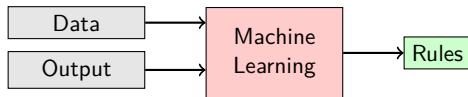
Traditional Programming vs Machine Learning

- **Traditional programming:** Programmer writes explicit rules (code) + Input data → Program computes Output
- **Machine Learning:** Input data + known Output (examples) → Learning algorithm produces the Rules (a trained model)
- In ML, the 'rules' are learned from examples rather than hand-written by a programmer

Traditional:



ML:



Concrete Example: Detecting Spam Emails

- **Traditional approach:** a programmer manually writes rules like “if email contains word FREE MONEY, mark as spam” - brittle and easy to fool
- **ML approach:** show the system thousands of emails already labelled spam/not-spam; it learns the patterns itself
- The ML approach generalises better to new, previously unseen types of spam

Rule-Based:

```
if "FREE" in email:  
    return SPAM  
if "Offer" in email:  
    return SPAM  
else:  
    return NOT_SPAM
```

ML Model:

[10,000 Emails + Labels]



Algorithm learns statistical word associations

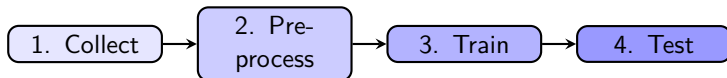


Probability of Spam: 92%

The Basic ML Workflow

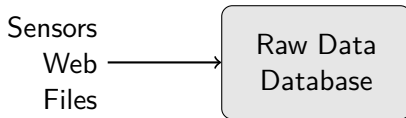
The Basic ML Workflow: Four Stages

- 1 **Data collection:** gathering relevant example data for the task
- 2 **Data preprocessing:** cleaning and preparing the data (recall Lecture 5's Pandas operations)
- 3 **Model training:** the learning algorithm finds patterns in the prepared data
- 4 **Model testing:** checking how well the trained model performs on new, unseen data



Stage 1: Data Collection

- Gathering data relevant to the task: sensor logs, customer records, images, text, etc.
- Recall Lecture 4: Pandas is commonly used to import this data (from CSV/Excel) once collected
- Data quality and quantity here directly determine how well the model can later learn



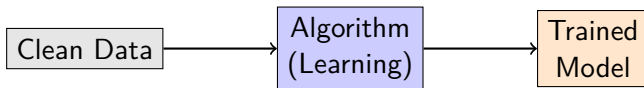
Stage 2: Data Preprocessing

- Cleaning data: handling missing values, removing errors or duplicates
- Organising data: selecting relevant columns (recall Lecture 5's column selection)
- Preparing data in a format the learning algorithm can use



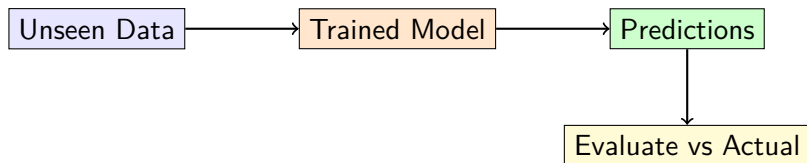
Stage 3: Model Training

- The learning algorithm examines the preprocessed data and identifies patterns
- The output of training is a 'model' - a set of learned parameters that can make predictions
- Different algorithms (KNN, Decision Tree, Linear Regression - Lectures 13-15) learn patterns differently



Stage 4: Model Testing

- The trained model is evaluated on new data it has not seen during training
- This checks whether the model actually learned generalisable patterns, not just memorised the training data
- Lecture 16 covers the training/testing dataset split in detail; Lectures 17-18 cover how to measure performance



Self-Assessment Questions

- **Q:** What is the key difference between traditional programming and ML?
- **Q:** Name the four stages of the basic ML workflow in order.
- **Q:** Which stage checks if a model generalises to new data?

Summary

Key Takeaways

- **Machine Learning** enables systems to learn patterns from data instead of following explicit hand-written rules
- **The Key Inversion:** ML derives rules from data + output examples, rather than applying pre-written rules to data
- **Basic ML Workflow:** proceeds through data collection, preprocessing, model training, and model testing

Next Up: Lecture 10

Well-Posed Machine Learning Problems

formally defining an ML problem using Task,
Experience, and Performance measure.

Questions?

Lecture 2.2: Well-Posed Machine Learning Problems

DI05000051 - AI & ML

Milav Dabgar

Lecturer, GP Palanpur

July 22, 2026

Recap: Lecture 9

- Lecture 9 defined ML, contrasted it with traditional programming, and outlined the 4-stage ML workflow (collect, preprocess, train, test).

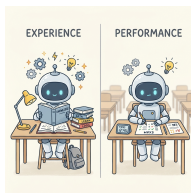
Today's Agenda

- ① Part 1: The Task-Experience-Performance framework
- ② Part 2: Worked example - spam detection
- ③ Part 3: Worked examples - movie recommendation and house price prediction

The Task-Experience-Performance Framework

What Makes an ML Problem 'Well-Posed'?

- A well-posed ML problem clearly defines exactly what the system must do and how success is measured
- Recall Lecture 9: the workflow starts with data collection - but data collection for WHAT exact goal?
- Tom Mitchell's classic framework: a program learns from Experience E , with respect to Task T , if its Performance P improves with E



Component 1: Task (T)

- The specific job the system must perform
- Examples: classify an email as spam/not-spam, recommend a movie, predict a house price
- The Task should be stated precisely enough that you know exactly what output is expected



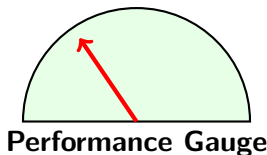
Component 2: Experience (E)

- The data the system learns from - recall Lecture 9's 'data collection' stage
- Usually a set of past examples: e.g., thousands of emails already labelled spam/not-spam
- More relevant and representative Experience generally leads to a better-performing model

Historical Records
Labelled Data

Component 3: Performance Measure (P)

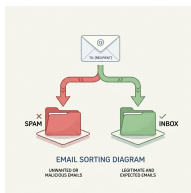
- A way to numerically measure how well the system performs the Task
- Examples: percentage of emails correctly classified, accuracy of price predictions
- Performance measures are covered in full technical detail in Lectures 17-18 (accuracy, confusion matrix, precision/recall)



Worked Examples

Well-Posed Example 1: Email Spam Detection

- Task (T): classify each incoming email as spam or not-spam
- Experience (E): a large set of emails already labelled as spam or not-spam
- Performance (P): percentage of new emails correctly classified



Well-Posed Example 2: Movie Recommendation

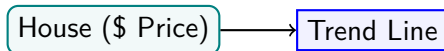
- Task (T): recommend movies a specific user is likely to enjoy
- Experience (E): the user's past viewing history and ratings, plus other users' patterns
- Performance (P): how often the user actually watches or rates highly the recommended movies

Recommendation Row

Movie A ★ Movie B ★ Movie C

Well-Posed Example 3: House Price Prediction

- Task (T): predict the selling price of a house given its features (area, location, number of rooms)
- Experience (E): historical records of houses sold, with their features and actual sale prices
- Performance (P): how close predicted prices are to actual sale prices



Three Examples Side by Side

Problem	Task (T)	Experience (E)	Performance (P)
Spam Detection	Classify email	Labelled emails	% correctly classified
Movie Recommendation	Recommend movies	Viewing/rating history	User engagement
House Price Prediction	Predict price	Historical sales data	Closeness to actual price

Summary & Practice

Why the T-E-P Framework Matters

- Forces clarity before jumping into building a model - avoids vague, poorly scoped ML projects
- Helps decide what data to collect (Experience) and how to judge success (Performance) upfront
- Sets up Unit 2's next topic: different Types of Machine Learning suited to different kinds of Tasks

✓ Task (T) defined
✓ Experience (E) defined
✓ Performance (P) defined
Ready to Start Project

Quick Check: Apply the T-E-P Framework

Discussion

- Q: For a system that predicts if a student will pass or fail, what is the Task?
- Q: What would the Experience be for that system?
- Q: How might you measure Performance for that system?

Key Takeaways

- The Task-Experience-Performance (T-E-P) framework formally defines a well-posed ML problem
- Spam detection, movie recommendation, and house price prediction are classic examples expressible via T-E-P
- Clearly defining T, E, and P before building a model is essential ML project planning practice

Next Lecture Preview

- Lecture 11 begins exploring Types of Machine Learning, starting with Supervised Learning - its definition and workflow.

➤ Supervised Learning

Questions?

Lecture 2.3: Types of Machine Learning - Supervised Learning

DI05000051 - AI & ML

Milav Dabgar

Lecturer, GP Palanpur

July 22, 2026

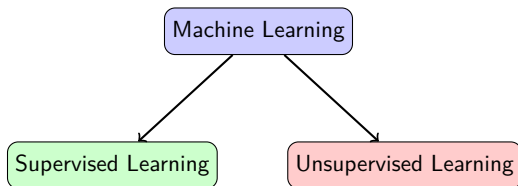
Recap: Lecture 10

- Lecture 10 introduced the T-E-P framework for well-posed ML problems, using spam detection, recommendation, and price prediction as examples.

Definition of Supervised Learning

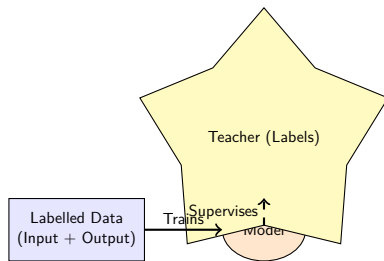
Types of Machine Learning: Overview

- ML problems can be grouped into different types based on the kind of data and feedback available
- Today and next lecture cover the two major types: Supervised Learning and Unsupervised Learning
- The type of learning shapes which algorithms are appropriate (Lectures 13-15 cover supervised algorithms specifically)



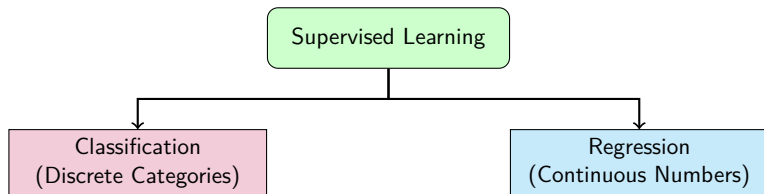
What is Supervised Learning?

- Supervised Learning = learning from labelled data, where each example has a known correct output
- Recall Lecture 10's spam example: each email is labelled 'spam' or 'not-spam' - this IS supervised learning
- The model learns a mapping from input features to the known output label
- 'Supervised' because the correct answers act like a teacher supervising the learning process



Two Broad Kinds of Supervised Learning Output

- Classification: the output is a category/class (e.g., spam or not-spam)
- Regression: the output is a continuous number (e.g., a house price)
- Both are supervised because both require labelled training data (known categories or known numeric outputs)



The Supervised Learning Workflow

The Supervised Learning Workflow

- 1 Collect labelled data: each example has both input features and a known correct output
- 2 Train a model on this labelled data - it learns the input-to-output mapping
- 3 Test the model on new, unseen inputs (recall Lecture 9's testing stage)
- 4 The model predicts an output for each new input, based on patterns learned from labels



Supervised Learning Example: Spam Detection Revisited

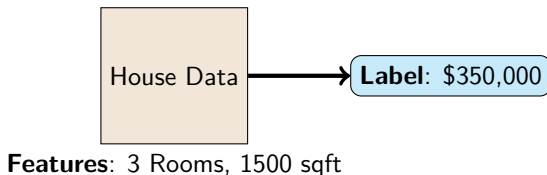
- Input features: sender address, subject line words, presence of certain phrases
- Label: spam or not-spam (known for training emails)
- This is a classification-type supervised learning problem

Inbox Example

Subject	Sender	Label
Meeting at 10 AM	boss@company.com	Not-Spam
WIN A FREE IPHONE	spammer@scam.net	Spam
Project Update	team@company.com	Not-Spam

Supervised Learning Example: House Price Prediction Revisited

- Input features: house area, location, number of rooms
- Label: actual historical sale price (a known number for training houses)
- This is a regression-type supervised learning problem



Why Labelled Data Matters

- Without correct labels, the model has nothing to check its predictions against during training
- Labelling data (e.g., marking emails as spam) is often a manual, time-consuming, human effort
- This labelling cost is a key practical limitation of supervised learning, motivating the alternative covered next lecture



Recognising Supervised Learning Problems

Quick Check: Is It Supervised Learning?

Discussion Questions

- **Q:** Predicting exam pass/fail from labelled past student records - supervised or not? Why?
- **Q:** Is the output a classification or a regression in this case?
- **Q:** What would the 'label' be in a supervised email spam problem?

Key Takeaways

- Supervised Learning uses labelled data (known correct outputs) to train a model
- The workflow is: collect labelled data, train, then predict on new unseen inputs
- Supervised learning includes classification (categorical output) and regression (numeric output) problems

Lecture 12 Preview

Lecture 12: Types of Machine Learning Part B - Unsupervised Learning, and a full comparison with Supervised Learning.

Unsupervised Learning (learning without labels) directly compares with today's Supervised Learning across input data, output, and applications.

Questions?

Lecture 2.4: Types of Machine Learning - Unsupervised Learning and Comparison

DI05000051 - AI & ML

Milav Dabgar

Lecturer, GP Palanpur

July 22, 2026

Recap & Agenda

Recap: Lecture 11

- Lecture 11 defined **Supervised Learning** (learning from labelled data).
- Covered its two main output types: **classification** and **regression**.

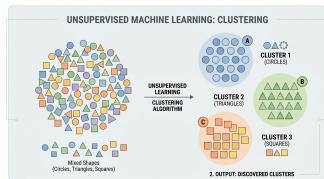
Today's Agenda

- ① **Part 1:** What is Unsupervised Learning?
- ② **Part 2:** Supervised vs Unsupervised comparison
- ③ **Part 3:** Classification, Regression, and Clustering examples

What is Unsupervised Learning?

What is Unsupervised Learning?

- **Unsupervised Learning** = learning from **UNlabelled** data.
- No known correct output is provided.
- Avoids needing labelled data at all (resolving labelling cost).
- Model finds hidden structure or patterns on its own.
- Common goal: group similar data points together (clustering).

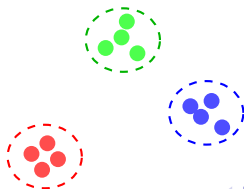


What is Clustering (Concept Only)?

Clustering

Grouping data points so that similar points are in the same group (cluster).

- No predefined categories - the algorithm discovers groupings from data's structure.
- **Example:** grouping customers by purchasing behaviour without pre-defined customer categories.



Supervised vs Unsupervised Comparison

Supervised vs Unsupervised: Input Data

Supervised Learning

Input data comes **WITH** labels (known correct outputs).

X1	X2	Label
5.1	3.5	Cat
4.9	3.0	Dog

Unsupervised Learning

Input data comes **WITHOUT** labels – just raw features.

X1	X2
5.1	3.5
4.9	3.0

This is the single defining difference between the two types.

Supervised vs Unsupervised: Output

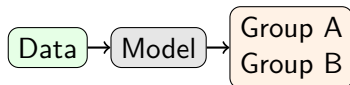
Supervised: Prediction

Output is a specific predicted label or number.



Unsupervised: Discovery

Output is a discovered structure (grouping).



Supervised vs Unsupervised: Applications

Supervised Applications:

- Spam detection
- House price prediction
- Disease diagnosis from labelled records

Unsupervised Applications:

- Customer segmentation
- Grouping similar news articles
- Anomaly/fraud pattern discovery

Note: Fraud Detection (Lecture 2 Callback)

Fraud detection can use either approach depending on whether labelled fraud examples exist!

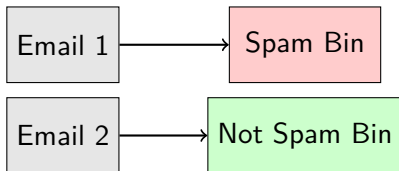
Supervised vs Unsupervised: Full Comparison Table

Aspect	Supervised Learning	Unsupervised Learning
Input Data	Labelled	Unlabelled
Output	Predicted label/value	Discovered groups/patterns
Applications	Spam detection, Price prediction	Customer segmentation, Anomaly discovery
Labelling Effort	Yes (Requires effort)	No

Classification, Regression, and Clustering

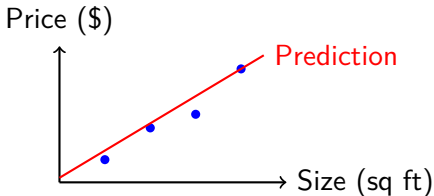
Example of Classification

- **Classification:** predicting a category/class label (a type of Supervised Learning).
- **Example:** classifying an email as spam or not-spam (recall Lectures 10-11).
- Detailed classification algorithms (KNN, Decision Tree) are covered in Lectures 13-14.



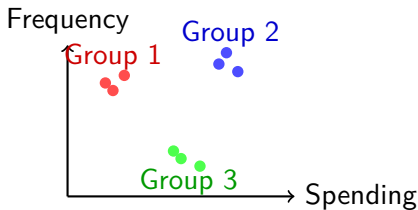
Example of Regression

- **Regression:** predicting a continuous numeric value (a type of Supervised Learning).
- **Example:** predicting a house's selling price (recall Lecture 10).
- Detailed regression algorithm (Linear Regression) is covered in Lecture 15.



Example of Clustering

- **Clustering:** grouping similar data points together (a type of Unsupervised Learning).
- **Example:** grouping customers into segments based on purchasing patterns, without predefined labels.
- Recall Lecture 7's scatter plot discussion - natural groupings visually hint at clusters.



Summary & Next Steps

Quick Check: Classify the Learning Type

- 1 **Grouping unlabelled news articles by topic similarity:**

Quick Check: Classify the Learning Type

- 1 **Grouping unlabelled news articles by topic similarity:**
Unsupervised Learning (Clustering)
- 2 **Predicting tomorrow's temperature from labelled historical weather data:**

Quick Check: Classify the Learning Type

- ① **Grouping unlabelled news articles by topic similarity:**
Unsupervised Learning (Clustering)
- ② **Predicting tomorrow's temperature from labelled historical weather data:**
Supervised Learning (Regression)
- ③ **What is the main practical advantage of unsupervised learning over supervised learning?**

Quick Check: Classify the Learning Type

- ① **Grouping unlabelled news articles by topic similarity:**
Unsupervised Learning (Clustering)
- ② **Predicting tomorrow's temperature from labelled historical weather data:**
Supervised Learning (Regression)
- ③ **What is the main practical advantage of unsupervised learning over supervised learning?**
It does not require expensive/time-consuming manual data labelling.

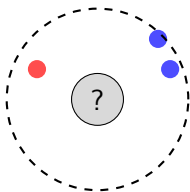
Key Takeaways

- **Unsupervised Learning** discovers structure (typically via clustering) in unlabelled data, without a known correct output.
- **Supervised and Unsupervised Learning** differ in input data (labelled vs unlabelled) and output (prediction vs discovery).
- **Classification and Regression** are supervised learning problems; **Clustering** is an unsupervised learning problem.

Next Lecture Preview

Coming Up: Lecture 13

Supervised Learning Algorithms Part A - **k-Nearest Neighbours (KNN)**, the first classification algorithm.



Questions?

Lecture 2.5: Supervised Learning Algorithms - k-Nearest Neighbours (KNN)

DI05000051 - AI & ML

Milav Dabgar

Lecturer, GP Palanpur

July 22, 2026

Recap: Lecture 12

- Lecture 12 covered Unsupervised Learning and compared it fully against Supervised Learning (input data, output, applications).

Supervised Learning
(Labelled Data)

Unsupervised Learning
(Unlabelled Data)

Today's Agenda

- Part 1: KNN's working principle
- Part 2: Effect of the value of k
- Part 3: Applications of KNN

Section 1: Introduction

Introducing Supervised Learning Algorithms

- Recall Lecture 11: Supervised Learning includes Classification (categorical output) and Regression (numeric output)
- Lectures 13-15 cover three specific algorithms: KNN and Decision Tree (classification), then Linear Regression (regression)
- All three are implemented in Scikit-learn (recall Lecture 4), following a common `fit()/predict()` pattern

KNN

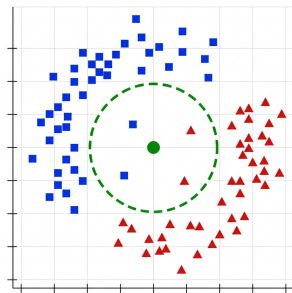
Decision Tree

Linear Regression

Section 2: KNN Working Principle

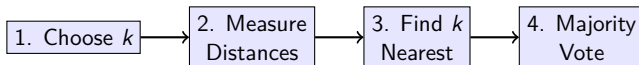
What is k-Nearest Neighbours (KNN)?

- KNN is a simple classification algorithm: to classify a new data point, look at its 'k' closest neighbours in the training data
- The new point is assigned the class that is most common among those k neighbours ('majority vote')
- Intuition: 'you are similar to the company you keep' - similar data points tend to belong to the same class



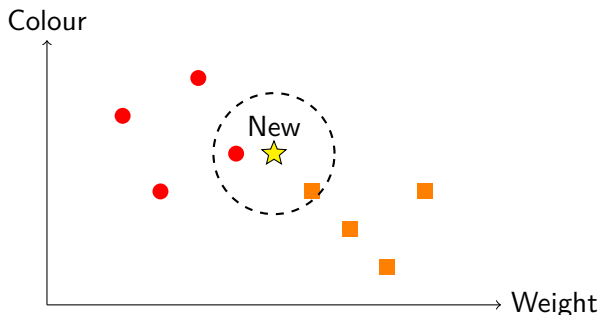
How KNN Works: Step by Step

- 1 Choose a value for k (e.g., $k = 3$)
- 2 When classifying a new point, measure its distance to all training points
- 3 Identify the k training points closest to the new point
- 4 Assign the new point the majority class among those k nearest neighbours



Worked Example: Classifying a New Data Point

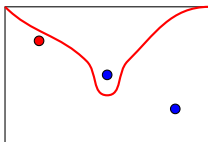
- Suppose we classify fruits as 'Apple' or 'Orange' based on weight and colour intensity
- A new fruit's 3 nearest neighbours ($k=3$) are: 2 Apples and 1 Orange
- By majority vote, the new fruit is classified as Apple



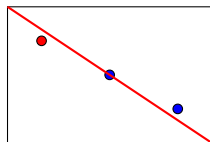
Section 3: Effect of the value of k

The Role of k : How Many Neighbours to Consider?

- Small k (e.g., $k=1$): decision based on very few neighbours - sensitive to noise/outliers
- Large k : decision based on many neighbours - smoother but may blur fine distinctions between classes
- Choosing a good k value is an important practical decision when using KNN



Small k (e.g., $k = 1$)

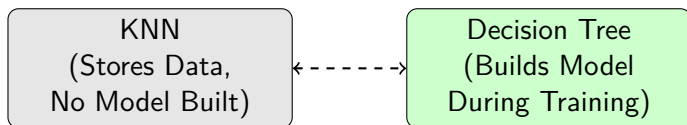


Large k (e.g., $k = 10$)

Section 4: Properties and Applications

Why KNN is Called a 'Lazy Learner'

- KNN does not build an explicit model during training - it simply stores all the training data
- All the real computation (finding nearest neighbours) happens only when a new prediction is requested
- This contrasts with Decision Tree (Lecture 14), which builds an explicit tree structure during training



Applications of KNN

- **Recommendation systems:** recommending items similar to what a user already likes
- **Handwriting/digit recognition:** classifying a handwritten digit based on similarity to known digit examples
- **Simple medical diagnosis support:** classifying a patient's condition based on similarity to past labelled cases

Recommendations

Digit
Recognition

Medical
Diagnosis

Strengths and Limitations of KNN

Strengths

- Simple to understand and implement
- No complex training phase
- Naturally handles multi-class problems

Limitations

- Can be slow for large datasets
- Sensitive to irrelevant or unscaled features

Section 5: Summary

Quick Check: KNN Understanding

- **Q:** In KNN, how is a new data point classified?
A: By majority vote among its k nearest neighbours.
- **Q:** If $k=1$, how many neighbours influence the classification?
A: Only 1 (the single closest point).
- **Q:** Why is KNN sometimes called a 'lazy learner'?
A: It doesn't build a model during training; it only stores data and computes at prediction time.

Key Takeaways

- KNN classifies a new data point by majority vote among its k nearest neighbours in the training data
- The choice of k affects sensitivity to noise (small k) vs smoothness of decision (large k)
- KNN is a 'lazy learner' with no explicit training phase, useful for recommendation and similarity-based classification

- **Lecture 14:** Supervised Learning Algorithms Part B - Decision Tree, the second classification algorithm

How does a Decision Tree build an explicit model unlike KNN?

Questions?

Lecture 2.6: Supervised Learning Algorithms - Decision Tree

DI05000051 - AI & ML

Milav Dabgar

Lecturer, GP Palanpur

July 22, 2026

Recap of KNN

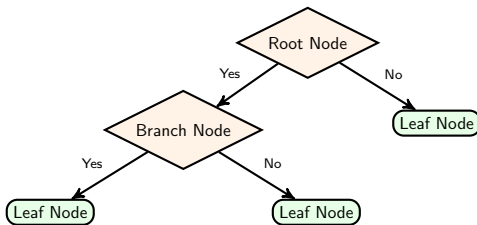
Lecture 13 covered **k-Nearest Neighbours (KNN)**:

- Classification by majority vote among the k nearest neighbours.
- A **lazy learner** with no explicit training phase.

Decision Tree Fundamentals

What is a Decision Tree?

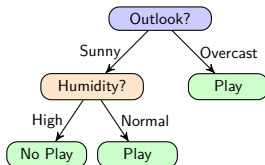
- A Decision Tree classifies data by asking a sequence of simple **yes/no** (or if-then) questions about its features.
- Structured like an **upside-down tree**: starts at a 'root' question, branches down to a final decision ('leaf').
- Unlike KNN (Lecture 13), a Decision Tree **DOES** build an explicit model structure during training.



Structure of a Decision Tree

Node Types

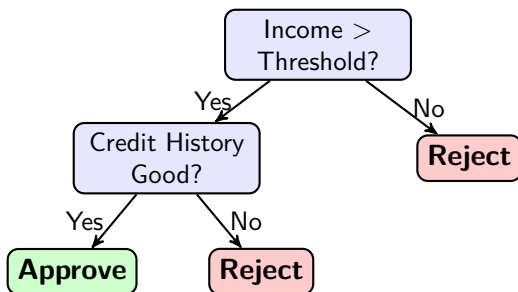
- **Root node:** the first, most informative question asked of the data.
- **Internal (branch) nodes:** further questions that split the data based on feature values.
- **Leaf nodes:** the final classification decision (the predicted class).



Working and Prediction

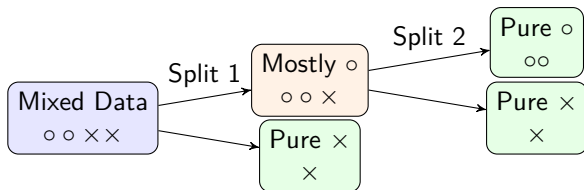
Worked Example: Loan Approval Decision Tree

- **Root question:** Is the applicant's income above a threshold?
- If yes → **branch:** Is their credit history good?
- If credit history is good → **leaf:** Approve loan. If not → **leaf:** Reject loan.
- If income is below threshold → **leaf:** Reject loan directly.



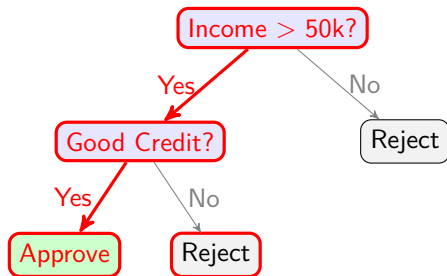
How a Decision Tree is Built (Basic Concept)

- The algorithm looks at the training data and **selects the feature that best separates the classes** at each step.
- This process repeats, creating branches, until the data is well-separated into **pure/mostly-pure groups** at the leaves.
- The tree is **built once during training** - unlike KNN, which stores raw data and computes at prediction time.



Reading a Decision Tree for Prediction

- To classify a **NEW** data point, start at the root and answer each question using that point's feature values.
- Follow the corresponding branch at each step until reaching a leaf node.
- The leaf node's class is the tree's **prediction** for that new data point.



New Applicant:

- Income: 60k
- Credit: Good

Comparison & Applications

Decision Tree vs KNN

Aspect	Decision Tree	k-Nearest Neighbours
Learning Type	Eager Learner	Lazy Learner
Training Phase	Builds an explicit tree structure during training	Stores raw training data (no explicit model built)
Prediction	Fast (simply follow branches to a leaf)	Slower (must compare distances to all stored points)
Interpretability	Output is highly interpretable (visually follow the branches)	Decision is less directly visualisable

Applications of Decision Trees

Finance



Loan approval
and credit risk
assessment.

Healthcare



Medical
diagnosis based
on sequence
of symptoms.

Business



Customer
churn pre-
diction for
services.

Strengths and Limitations of Decision Trees

Strengths

- **Easy to interpret** and explain - can be visualised as a simple flowchart.
- Handles both numeric and categorical features naturally.

Limitations

- Can become **overly complex** ('overgrown') and fit training data too specifically.
- Small changes in data can sometimes produce a very different tree.

Summary & Next Steps

Quick Check: Decision Tree Understanding

Concept Check

- 1 What are the **three types** of nodes in a decision tree?
- 2 How is a **new data point** classified using a decision tree?
- 3 Name **one advantage** of Decision Trees over KNN.

Summary

- A Decision Tree classifies data through a sequence of feature-based questions, from a root node down to a leaf decision.
- Unlike KNN, a Decision Tree builds an explicit structure during training, making predictions fast and highly interpretable.
- Decision Trees are widely used in finance and healthcare where explainable decisions matter.

Upcoming in Lecture 15

Supervised Learning Algorithms Part C

Linear Regression

The course's first **regression algorithm** - shifting from classification to predicting continuous numeric values (like house prices)!

Questions?

Lecture 2.7: Supervised Learning Algorithms - Linear Regression

DI05000051 - AI & ML

Milav Dabgar

Lecturer, GP Palanpur

July 22, 2026

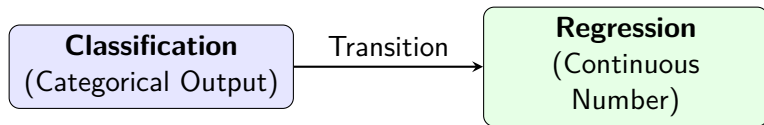
Recap: Lecture 14

- Lecture 14 covered the Decision Tree classification algorithm - an interpretable, explicit tree structure built during training for classification tasks like loan approval.
- **Key Highlight:** Lecture 14 covered Decision Trees: an interpretable classification algorithm using a root-to-leaf sequence of questions.

What is Linear Regression?

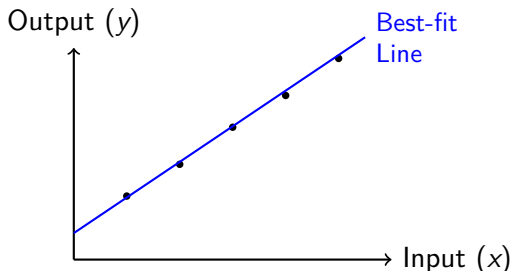
From Classification to Regression

- Recall Lecture 11: Regression predicts a continuous NUMBER, unlike classification's categorical output (KNN, Decision Tree)
- Linear Regression is the classic, simplest regression algorithm
- Recall Lecture 10's house price prediction example - this is a textbook Linear Regression problem



What is Linear Regression?

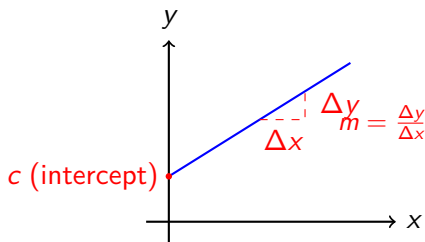
- Linear Regression models the relationship between input feature(s) and a continuous output as a straight line
- Simplest form: one input feature (e.g., house area) predicting one output (e.g., price)
- The model finds the 'best-fit' straight line through the training data points



The Regression Line and Fitting

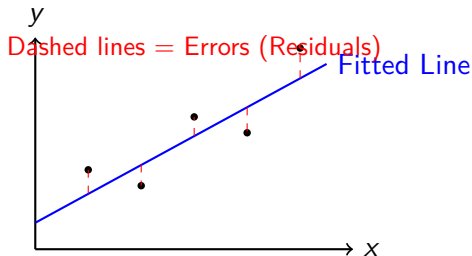
The Regression Line: $y = mx + c$

- The fitted line follows the familiar equation: $y = mx + c$
- x = input feature (e.g., house area); y = predicted output (e.g., price)
- m = slope (how much y changes per unit of x); c = intercept (value of y when $x = 0$)
- 'Training' the model means finding the best values of m and c from the training data



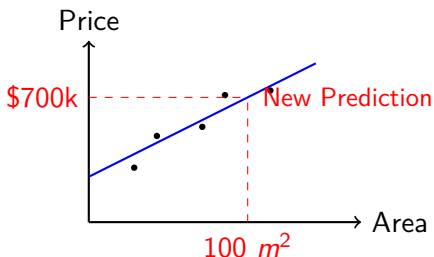
What Does 'Best Fit' Mean?

- The best-fit line minimises the overall distance between the line and all the actual data points
- Points that are far from the line represent larger prediction errors
- The training process automatically searches for the m and c that make these errors as small as possible



Worked Example: Predicting House Price

- Training data: pairs of (house area, sale price) from past sales
- The Linear Regression model fits a line: $\text{Price} = m \times \text{Area} + c$
- For a NEW house of a given area, the model predicts its price by plugging area into this line equation



Applications and Comparisons

Applications of Linear Regression

- House/property price prediction (recall Lecture 10)
- Predicting sales based on advertising spend
- Estimating a student's expected marks from hours studied
- Forecasting demand or resource requirements from historical trends

Real Estate
Price Estimation

Marketing
Sales vs Ad Spend

Education
Marks Prediction

Business
Demand Forecasting

Classification vs Regression

- Classification (KNN, Decision Tree): output is a category/class
- Regression (Linear Regression): output is a continuous number
- All three are supervised learning algorithms requiring labelled training data

Classification Algorithms	Regression Algorithm
Outputs a category (e.g., Spam)	Outputs a number (e.g., \$500k)
KNN, Decision Tree	Linear Regression
Decision boundaries separate classes	Line of best fit models the trend

Strengths and Limitations

Strengths

- Simple, fast to train, and easy to interpret
- Works well when relationship is genuinely close to linear

Limitations

- Performs poorly if the true relationship is not a straight line (non-linear)
- Sensitive to outliers

Quick Check: Understanding

Questions for Review

- 1 What type of output does Linear Regression predict?
- 2 In $y = mx + c$, what does ' m ' represent?
- 3 Name one real-world application of Linear Regression.

Key Takeaways

- Linear Regression fits a straight line ($y = mx + c$) to model the relationship between input feature(s) and a continuous output
- The 'best-fit' line minimises the overall distance between the line and the actual training data points
- Linear Regression is a widely applicable regression algorithm suited to price prediction and similar forecasting tasks

Summary of Topics Covered:

What is Linear Regression? → Fitting the Line → Applications →
Comparison to Classification

Lecture 16: Model Training and Testing

How a training dataset and testing dataset are used together to build and validate ML models

- Lecture 16 steps back from specific algorithms to cover how any of them (KNN, Decision Tree, Linear Regression) is properly trained and tested using training/testing dataset splits.

Questions?

Thank You!

Lecture 2.8: Model Training and Testing

DI05000051 - AI & ML

Milav Dabgar

Lecturer, GP Palanpur

July 22, 2026

Introduction

Recap: Lecture 15

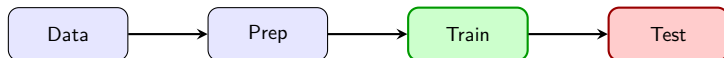
- Lectures 13-15 covered KNN, Decision Tree, and Linear Regression - three supervised learning algorithms needing proper training and testing.

Today's Agenda

- 1 Part 1: Training dataset
- 2 Part 2: Testing dataset
- 3 Part 3: The train-test split concept

Recall: Stage 4 of the ML Workflow

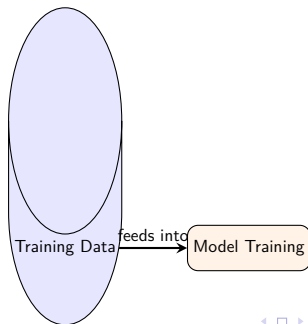
- Lecture 9 introduced 'model testing' as the final stage of the basic ML workflow
- Today we make this precise: HOW do we split data so training and testing are done properly?
- This applies to every algorithm already covered - KNN, Decision Tree, and Linear Regression alike



Training Dataset

What is a Training Dataset?

- The portion of data used to teach the model - it learns patterns from these examples
- For supervised learning (recall Lecture 11), the training dataset includes both input features and known labels
- The model's parameters (e.g., Linear Regression's m and c from Lecture 15) are determined using this data



Testing Dataset

What is a Testing Dataset?

- A separate portion of data, NOT used during training, held back to evaluate the model afterwards
- Since the model has never seen this data, it reveals how well the model generalises to new, unseen examples
- If a model performs well on training data but poorly on testing data, it likely just memorised the training examples rather than learning generalisable patterns

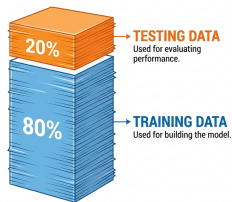
Key Concept

Testing evaluates **generalisation**, not just memorisation!

The Train-Test Split Concept

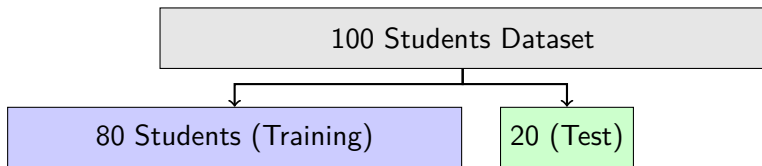
The Train-Test Split Concept

- Before training, the full dataset is randomly divided into two parts: a training set and a testing set
- A common split ratio is 80% training / 20% testing (other ratios like 70/30 are also used)
- Splitting must happen BEFORE training, so the testing portion truly remains 'unseen' by the model



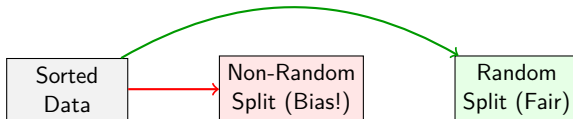
Worked Example: Splitting a Student Marks Dataset

- Suppose we have 100 students' data (hours studied vs marks obtained, recall Lecture 7's scatter plot)
- An 80/20 split gives 80 students for training the Linear Regression model, and 20 students held back for testing
- After training, we check: how close are the model's predicted marks to the actual marks of those 20 held-back students?



Why Random Splitting Matters

- The split should be done randomly, not e.g. by simply taking the first 80 rows
- A non-random split risks the training and testing sets being systematically different (e.g., if data was originally sorted by class)
- Randomisation helps ensure both sets are representative of the overall data



Train-Test Split Applies to Every Algorithm

- **KNN** (Lecture 13): trained/stored using the training set; nearest-neighbour searches use only training set points
- **Decision Tree** (Lecture 14): tree structure is built using only the training set
- **Linear Regression** (Lecture 15): the best-fit line (m, c) is computed using only the training set
- In every case, the testing set is reserved purely for evaluating performance afterwards

What Comes After Testing?

- Once a model is tested, its performance needs to be MEASURED numerically, not just judged by eye
- Lectures 17-18 introduce specific performance measures: Accuracy, Confusion Matrix, Precision, and Recall
- These measures make Lecture 10's abstract 'Performance measure (P)' concept fully concrete

Quick Check: Train-Test Split

Questions

- **Q:** Why can't we test a model on the same data it was trained on?
- **Q:** What is a common train-test split ratio?
- **Q:** Why should the split be done randomly rather than sequentially?

Summary

Key Takeaways

- A training dataset teaches the model; a testing dataset (held back, unseen) fairly evaluates it afterwards
- A common train-test split ratio is 80% training / 20% testing, performed randomly before training begins
- Every supervised algorithm (KNN, Decision Tree, Linear Regression) follows this same train-then-test discipline

- **Lecture 17:** Model Evaluation Techniques Part A - Accuracy and the Confusion Matrix, quantifying how well a classification model performs

Questions?

Lecture 2.9: Model Evaluation Techniques - Accuracy and Confusion Matrix

DI05000051 - AI & ML

Milav Dabgar

Lecturer, GP Palanpur

July 22, 2026

Recap: Lecture 16

- Lecture 16 covered the train-test split concept: models are trained on a training set and fairly evaluated on a separate, unseen testing set.

Train-Test Split



Train on Training Set → Evaluate on Unseen Testing Set

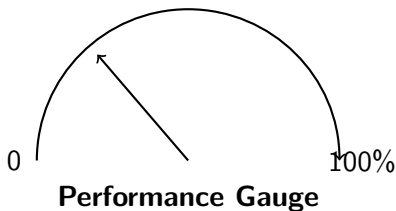
Today's Agenda

- 1 Part 1: Accuracy as a performance measure
- 2 Part 2: The Confusion Matrix
- 3 Part 3: Limitations of accuracy alone

Accuracy as a performance measure

Making 'Performance' Concrete

- Recall Lecture 10's T-E-P framework: Performance (P) needs a precise, numeric definition
- Recall Lecture 16: after testing, we compare predictions against actual known labels from the testing set
- Today's focus: two ways to measure classification performance
 - Accuracy and the Confusion Matrix



What is Accuracy?

Accuracy

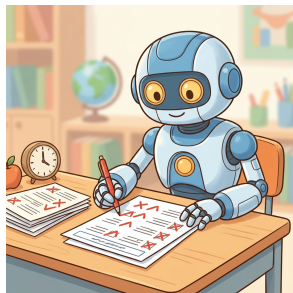
$$\text{Accuracy} = \frac{\text{number of correct predictions}}{\text{total number of predictions}}$$

- Often expressed as a percentage.
- Example: if a spam classifier correctly labels 90 out of 100 test emails (recall Lecture 10's spam example), accuracy = 90%.
- Accuracy is the simplest and most commonly cited performance measure for classification.

The Confusion Matrix

What is a Confusion Matrix?

- A table that breaks down predictions into 4 categories, comparing predicted vs actual class
- **True Positive (TP)**: correctly predicted as positive (e.g., spam)
- **True Negative (TN)**: correctly predicted as negative (e.g., not-spam)
- **False Positive (FP)**: incorrectly predicted positive (e.g., good email marked spam)
- **False Negative (FN)**: incorrectly predicted negative (e.g., spam marked not-spam)



Worked Example: A Simple Confusion Matrix

- Out of 100 test emails: 45 correctly marked spam (TP), 40 correctly marked not-spam (TN)
- 10 good emails wrongly marked spam (FP), 5 spam emails wrongly marked not-spam (FN)

		Predicted Class	
		Spam	Not Spam
Actual	Spam	45 (TP)	5 (FN)
	Not Spam	10 (FP)	40 (TN)

- Accuracy here = $(TP + TN) / \text{Total} = (45+40)/100 = 85\%$

Reading a Confusion Matrix: Simple Interpretation

- Rows typically represent actual classes; columns represent predicted classes
- The diagonal (TP and TN) represents correct predictions - larger diagonal values mean a better model
- Off-diagonal values (FP and FN) represent the two different KINDS of mistakes a model can make

Correct TP	Mistake FN
Mistake FP	Correct TN

Limitations of accuracy alone

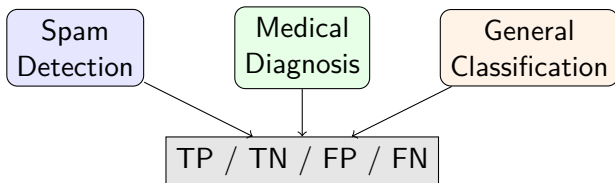
Why Accuracy Alone Can Be Misleading

- Example: a rare disease affects 1% of patients. A model that always predicts 'no disease' achieves 99% accuracy - yet it is useless!
- High accuracy can hide poor performance on an important minority class (e.g., missing all actual disease cases)
- This motivates looking beyond accuracy to Precision and Recall, covered next lecture

Warning! **99% Accuracy**
...but misses every sick patient!

Confusion Matrix Applies Across Applications

- **Spam detection:** TP=correctly caught spam, FP=good email wrongly blocked
- **Medical diagnosis:** TP=correctly diagnosed patient, FN=missed a real case - often the most costly mistake
- Any binary classification problem (KNN or Decision Tree output) can be evaluated this way



Quick Check: Accuracy and Confusion Matrix

Questions

- **Q:** If a model makes 95 correct predictions out of 100, what is its accuracy?
- **Q:** In a confusion matrix, what does a False Negative represent?
- **Q:** Why might accuracy alone be misleading for a rare disease detection problem?

Key Takeaways

- Accuracy measures the proportion of correct predictions out of all predictions made
- A Confusion Matrix breaks predictions into True Positive, True Negative, False Positive, and False Negative categories
- Accuracy alone can be misleading, especially with imbalanced classes (e.g., rare disease detection)

Lecture 18: Model Evaluation Techniques Part B

Precision and Recall, addressing accuracy's blind spots

→ Next: Precision & Recall

Questions?

Lecture 2.10: Model Evaluation Techniques - Precision and Recall

DI05000051 - AI & ML

Milav Dabgar

Lecturer, GP Palanpur

July 22, 2026

Recap & Agenda

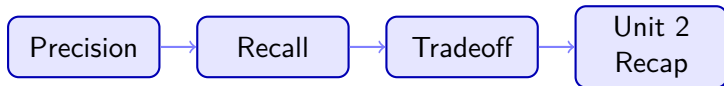
Recap: Lecture 17

- Lecture 17 introduced Accuracy and the Confusion Matrix (TP, TN, FP, FN).
- We showed how accuracy alone can be misleading on imbalanced data such as rare disease detection.

Predicted		
Actual	TP	FN
	FP	TN

Today's Agenda

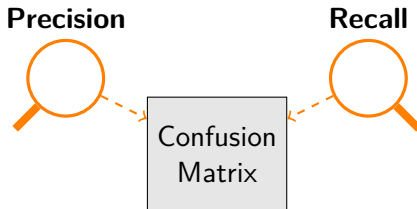
- **Part 1:** Precision
- **Part 2:** Recall
- **Part 3:** Choosing between Precision and Recall by application
- **Part 4:** Unit 2 recap



Precision and Recall

Beyond Accuracy: Two Sharper Questions

- Recall Lecture 17's disease-detection problem: accuracy alone hid the model missing real disease cases.
- Precision and Recall use the SAME confusion matrix (TP, TN, FP, FN) but ask two more specific questions.
- **Precision asks:** "Of everything predicted positive, how much was actually correct?"
- **Recall asks:** "Of everything actually positive, how much did we correctly catch?"



Precision: Formula and Meaning

Precision

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

- **Measures:** of all items the model labelled positive, what fraction were truly positive?
- High precision means few false alarms (few False Positives).
- **Example:** of all emails the spam filter flagged as spam, what percentage were truly spam?

Predicted Positive

TP 45	FN 5
FP 10	TN 40

Recall: Formula and Meaning

Recall

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

- **Measures:** of all items that were truly positive, what fraction did the model correctly catch?
- High recall means few missed cases (few False Negatives).
- **Example:** of all truly spam emails, what percentage did the filter successfully catch?

Actually Positive



TP 45	FN 5
FP 10	TN 40

Worked Example: Precision and Recall

Recall the spam matrix from Lecture 17: TP=45, TN=40, FP=10, FN=5 (Accuracy was 85%)

TP 45	FN 5
FP 10	TN 40

- **Precision** = $\frac{45}{45+10} = \frac{45}{55} \approx 82\%$
- **Recall** = $\frac{45}{45+5} = \frac{45}{50} = 90\%$

The filter is quite good at catching spam (90% recall) but occasionally flags good emails too (82% precision).

Choosing the Right Metric

When Precision Matters More

- Precision matters most when **False Positives** are costly.
- **Example:** A spam filter with low precision annoyingly blocks legitimate, important emails.
- **Example:** Recommending an unsuitable product wastes user trust and attention.

× **False Alarm!** ×

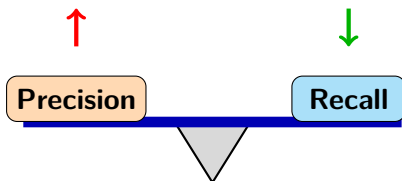
When Recall Matters More

- Recall matters most when **False Negatives** are costly.
- **Example:** Missing a real disease case can be far more dangerous than a false alarm.
- **Example:** A fraud detection system missing actual fraud cases.

✗ Missed Threat! ✗

Precision vs Recall: The Tradeoff

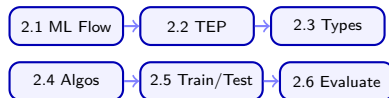
- Improving one often slightly reduces the other - there is usually a tradeoff between them.
- A model tuned to catch every possible positive case (high recall) may also flag more false alarms (lower precision).
- The right balance depends entirely on the specific application and the cost of each type of error.



Unit 2 Recap

Unit 2 Recap: The Big Picture

- **2.1:** ML vs Traditional Programming
- **2.2:** Task, Experience, Performance
- **2.3:** Supervised vs Unsupervised Learning
- **2.4:** KNN, Decision Tree, Linear Regression
- **2.5:** Training, Testing, and Splits
- **2.6:** Evaluation: Accuracy, Confusion Matrix, Precision, Recall



Summary & Next Steps

Quick Check: Precision and Recall

Questions

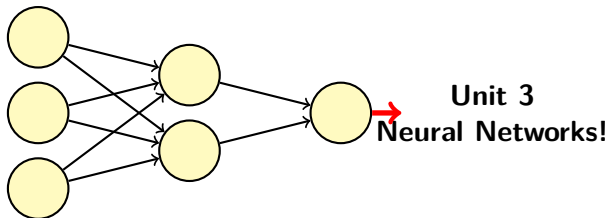
- 1 What does **Precision** measure, in plain English?
- 2 What does **Recall** measure, in plain English?
- 3 For a disease-detection system, is Recall or Precision usually more critical? Why?

Key Takeaways

- **Precision** ($TP/(TP + FP)$): Of predicted positives, how many were truly positive. Matters when false alarms are costly.
- **Recall** ($TP/(TP + FN)$): Of true positives, how many were correctly caught. Matters when missed cases are costly.
- Precision and Recall typically trade off against each other; the right balance depends on the application.
- Unit 2 built the full ML pipeline: definitions, learning types, algorithms, training/testing, and evaluation.

Next Lecture Preview

- **Lecture 19:** Introduction to Neural Networks
- Unit 3 begins with the motivation for neural networks and the analogy between biological and artificial neurons.



Questions?

Lecture 3.1: Introduction to Neural Networks - Motivation and Biological Inspiration

DI05000051 - AI & ML

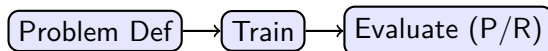
Milav Dabgar

Lecturer, GP Palanpur

July 22, 2026

Recap: Lecture 18

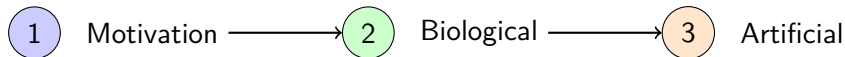
- Lecture 18 closed Unit 2 with Precision/Recall and a recap of the full classical ML pipeline built across 10 lectures.



Motivation: Why Neural Networks?

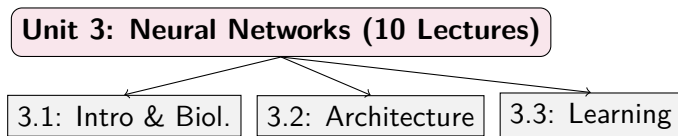
Today's Agenda

- Part 1: Why neural networks were developed
- Part 2: The biological neuron
- Part 3: The artificial neuron



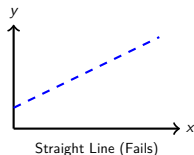
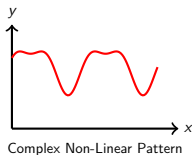
Welcome to Unit 3: Neural Networks

- Recall Unit 2's algorithms: KNN, Decision Tree, and Linear Regression are relatively simple and interpretable
- But some problems (image recognition, speech, complex patterns) are far too intricate for those simple algorithms alone
- Unit 3 (10 lectures) introduces Neural Networks - a more powerful, brain-inspired class of ML model



Motivation: Why Do We Need Neural Networks?

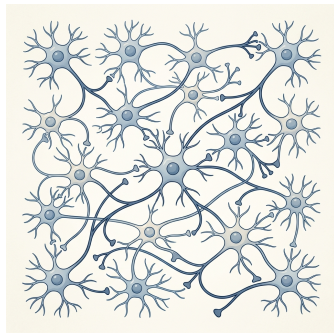
- Recall Lecture 15's Linear Regression: works well only for roughly straight-line relationships
- Recall Lecture 13-14's KNN/Decision Tree: struggle with very complex patterns in large, high-dimensional data (like raw images)
- Real-world tasks like recognising handwritten digits or understanding speech involve extremely complex, non-linear patterns
- Neural networks were designed to learn such complex patterns by loosely mimicking how the human brain processes information



The Biological Neuron

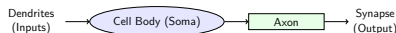
The Biological Inspiration

- The human brain contains billions of interconnected cells called neurons
- Neurons communicate by receiving signals, processing them, and passing signals on to other neurons
- This massively parallel, interconnected structure gives the brain its remarkable pattern-recognition and learning ability
- Artificial Neural Networks (ANNs) are loosely inspired by this biological structure - not an exact replica



Structure of a Biological Neuron

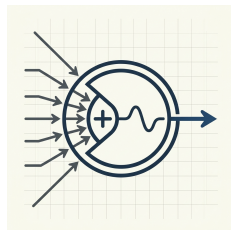
- **Dendrites:** receive incoming signals from other neurons
- **Cell body (soma):** processes/combines the incoming signals
- **Axon:** transmits an outgoing signal to other neurons if the combined signal is strong enough
- **Synapse:** the connection point where signals pass from one neuron to the next



The Artificial Neuron

What is an Artificial Neuron?

- An artificial neuron is a simple mathematical unit modelled loosely on a biological neuron
- It receives one or more numeric inputs, combines them, and produces one output
- Just like a biological neuron 'fires' only when its combined input signal is strong enough, an artificial neuron uses a similar threshold-like mechanism
- This mechanism (introduced fully in Lecture 23) is called

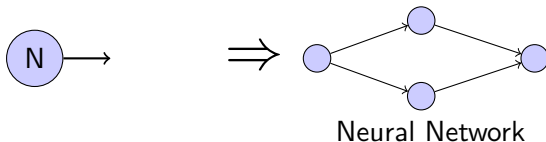


Biological Neuron vs Artificial Neuron

Biological Neuron	Artificial Neuron
Dendrites (receive signals)	Inputs (numeric values fed into the neuron)
Cell body (combines signals)	Weighted sum (inputs combined mathematically, covered in Lecture 22)
Axon firing threshold	Activation function (covered in Lecture 23)
Axon output (signal to next neuron)	Output (numeric value passed to next neuron/layer)

From One Neuron to a Network

- A single artificial neuron is simple, but connecting MANY of them together creates a powerful Neural Network
- Just as the brain's power comes from billions of interconnected neurons, an ANN's power comes from many interconnected artificial neurons
- The structure of these connections (layers) is covered in Lecture 21



Quick Check: Biological vs Artificial Neuron

Questions

- **Q:** What part of a biological neuron receives incoming signals?
- **Q:** What is the artificial-neuron equivalent of a biological neuron's dendrites?
- **Q:** Why were neural networks developed, given that Unit 2 already covered several ML algorithms?

Summary & Next Steps

Key Takeaways

- Neural networks were developed to handle complex, non-linear patterns beyond the reach of simpler algorithms like Linear Regression
- A biological neuron receives signals via dendrites, combines them in the cell body, and fires an output via the axon
- An artificial neuron loosely mirrors this: inputs, a combining mechanism, and an output governed by an activation function

Next Lecture Preview

- **Lecture 20:** Applications of Neural Networks - image recognition, speech recognition, and recommendation systems

Coming Up: Real-World Applications



Questions?

Lecture 3.2: Applications of Neural Networks

DI05000051 - AI & ML

Milav Dabgar

Lecturer, GP Palanpur

July 22, 2026

Introduction

Recap: Lecture 19

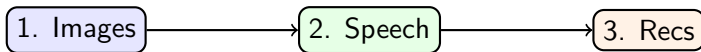
- Lecture 19 covered why neural networks were developed and how an artificial neuron loosely models a biological neuron.

Key Recap

Lecture 19 covered the motivation for neural networks and the biological-to-artificial neuron analogy: inputs, combining mechanism, activation, and output.

Today's Agenda

- Part 1: Image recognition
- Part 2: Speech recognition
- Part 3: Recommendation systems



From Single Neurons to Real Applications

- Recall Lecture 19: a single artificial neuron is simple, but networks of many neurons are extremely powerful
- Recall Lecture 2: AI already touches healthcare, finance, education, manufacturing, and social media
- Neural networks power many of the **MOST** advanced examples within those very industries

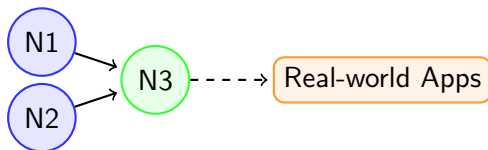
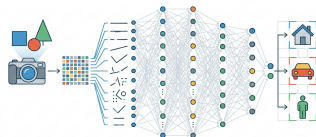


Image Recognition

Application 1: Image Recognition

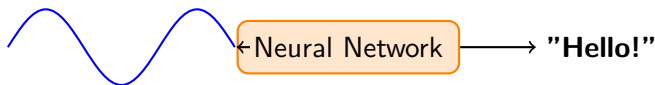
- Neural networks can classify images: is this a cat or a dog? Is this X-ray showing a tumour?
- The network takes pixel values as input and learns to recognise visual patterns (edges, shapes, textures)
- Recall Lecture 2's healthcare application - AI-assisted medical image analysis is largely powered by neural networks



Speech Recognition

Application 2: Speech Recognition

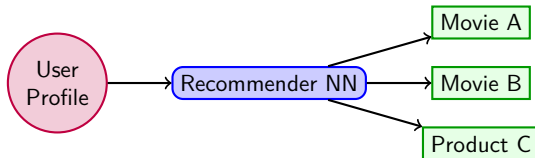
- Neural networks convert spoken audio into text (speech-to-text)
- Powers voice assistants, transcription services, and voice-controlled devices
- The network learns to recognise patterns in sound waves corresponding to words and phonemes



Recommendation Systems

Application 3: Recommendation Systems

- Neural networks can learn complex patterns in user behaviour to recommend content or products
- Recall Lecture 1's Netflix example and Lecture 13's KNN-based recommendation discussion
- Neural network-based recommenders can capture more subtle, non-linear patterns in preferences than simpler methods



Why Neural Networks Excel

Why Neural Networks Excel at These Tasks

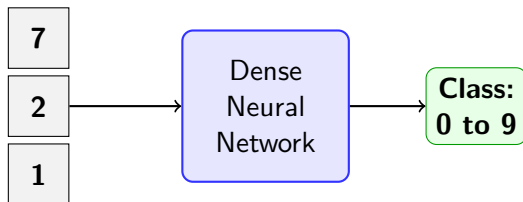
- Images, audio, and rich behavioural data all have extremely complex, high-dimensional patterns
- Recall Lecture 19: neural networks were motivated specifically to handle such complex, non-linear patterns
- Unit 2's simpler algorithms (KNN, Decision Tree, Linear Regression) generally cannot capture this level of complexity as effectively

Task Complexity	Suitable Algorithm Type
Low-dimensional, simple relationships	Linear Regression, Decision Trees
Local patterns in structured data	KNN, K-Means
High-dimensional, highly non-linear (Images, Audio)	Neural Networks

Worked Example & Beyond

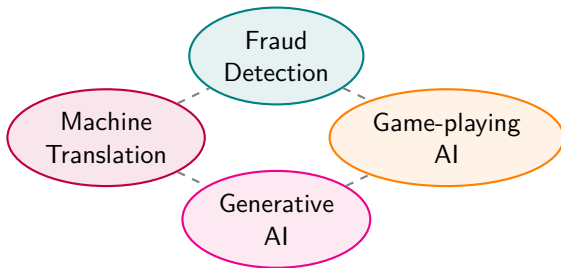
Worked Example: Digit Recognition Preview

- One classic image recognition task: recognising a handwritten digit (0-9) from an image
- This exact task will be revisited hands-on in Lecture 28 using the Digit Recognition dataset
- It combines everything from Unit 3: neurons, layers, activation functions, and network architectures



Applications Beyond the Three Highlighted

- Neural networks also power machine translation, game-playing AI, and fraud detection systems
- Recall Lecture 5.2 preview: modern Generative AI (Unit 5) is also built on advanced neural network architectures
- The three applications studied today (image, speech, recommendation) are foundational examples, not an exhaustive list



Quick Check: Matching Applications

- ① **Q:** Which application converts spoken audio into text?
- ② **Q:** Which application analyses pixel patterns to classify what an image shows?
- ③ **Q:** How does a neural network recommendation system relate to Lecture 1's Netflix example?

Summary & Next Steps

Key Takeaways

- Neural networks power image recognition (classifying visual content), speech recognition (audio-to-text), and recommendation systems (personalised suggestions)
- These applications succeed because neural networks can learn extremely complex, non-linear patterns in rich data
- Topic 3.1 (Lectures 19-20) has now covered both the motivation/biological inspiration AND the real-world applications of neural networks

- Lecture 21: Structure of Artificial Neural Networks Part A - the input, hidden, and output layers that make up an ANN

Next Up:
Lecture 21: Structure of ANNs

Questions?

Lecture 3.3: Structure of Artificial Neural Networks - Layers

DI05000051 - AI & ML

Milav Dabgar

Lecturer, GP Palanpur

July 22, 2026

Recap: Lecture 20

- Lecture 20 covered neural network applications: image recognition, speech recognition, and recommendation systems.

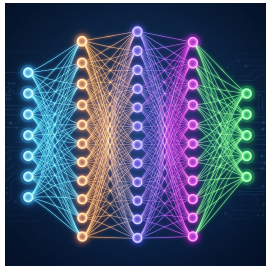
Today's Agenda

- Part 1: The Input layer
- Part 2: The Hidden layer
- Part 3: The Output layer
- Part 4: Data flow through the network

Structure of Artificial Neural Networks

From One Neuron to a Layered Network

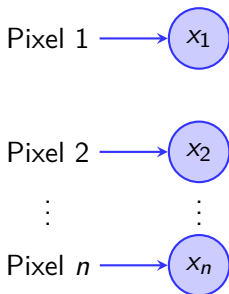
- Recall Lecture 19: a single artificial neuron takes inputs and produces an output
- A full Artificial Neural Network (ANN) organises many such neurons into layers
- Today's topic: the three types of layers found in a typical ANN



The Input Layer

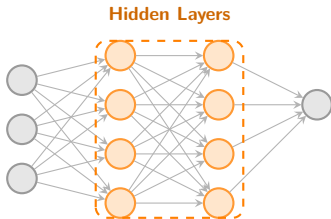
- The first layer of the network - it receives the raw input data
- Each neuron in the input layer typically corresponds to one input feature
- Example: for digit recognition (previewed in Lecture 20), each input neuron might represent one pixel value

Input Layer



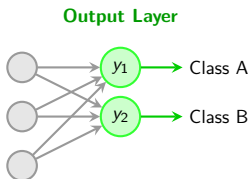
The Hidden Layer(s)

- Layer(s) between the input and output layers - not directly observed from outside the network
- This is where most of the network's pattern-detection and computation happens
- A network can have one or many hidden layers; more hidden layers generally allow learning more complex patterns (previewed further in Lecture 26)



The Output Layer

- The final layer - it produces the network's actual prediction/result
- Number of output neurons depends on the task: e.g., one neuron for a yes/no prediction, ten neurons for classifying digits 0-9
- The output layer's result is the network's final answer, comparable to a classification/regression output from Unit 2



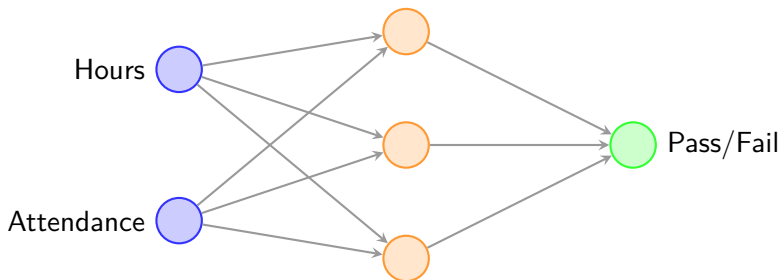
Input vs Hidden vs Output Layer

- Input layer: receives raw data, one neuron per input feature
- Hidden layer(s): perform internal computation and pattern detection, not directly visible
- Output layer: produces the final prediction/result
- Data flows in one direction: input $\rightarrow$ hidden $\rightarrow$ output (in the simplest, 'feedforward' networks)

Layer	Role	Example
Input	Receives raw data features	Pixels of an image
Hidden	Internal computation & patterns	Edges, shapes
Output	Final prediction/result	Digit '7'

Worked Example: A Simple ANN for Pass/Fail Prediction

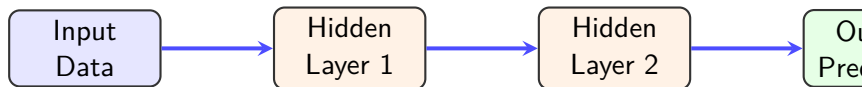
- Input layer: 2 neurons for 'hours studied' and 'attendance percentage'
- Hidden layer: a few neurons combining these inputs in more complex ways
- Output layer: 1 neuron producing a pass/fail prediction



How Data Flows Through the Network

- Input values enter through the input layer
- Each hidden layer neuron receives values from the previous layer, combines them (Lecture 22), and passes a result forward
- This continues until the output layer produces the final prediction
- This step-by-step forward movement of data is called a '**forward pass**'

Forward Pass →



Quick Check: ANN Layers

Check your understanding

- ① **Q:** Which layer receives the raw input data?
- ② **Q:** What is the role of the hidden layer(s)?
- ③ **Q:** For a network predicting one of 10 digit classes, how many output neurons might it have?

Key Takeaways

- An ANN is organised into an input layer, one or more hidden layers, and an output layer
- The input layer receives raw data; hidden layers perform internal computation; the output layer produces the final prediction
- Data flows forward through the network from input to output in a 'forward pass'

Up Next:

Lecture 22: Structure of Artificial Neural Networks Part B

The basic working of neural networks and the concept of weights and bias.

Questions?

Lecture 3.4: Structure of Artificial Neural Networks - Working, Weights and Bias

DI05000051 - AI & ML

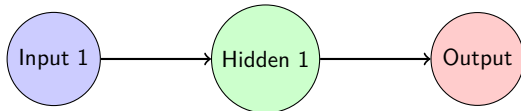
Milav Dabgar

Lecturer, GP Palanpur

July 22, 2026

Recap: Lecture 21

- Lecture 21 covered the input, hidden, and output layers of an ANN, and the forward flow of data through the network.



Introduction

Today's Agenda

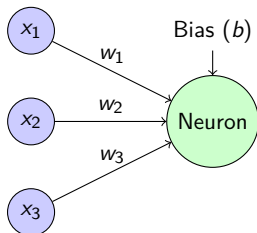
- 1 Part 1: What weights represent
- 2 Part 2: What bias represents
- 3 Part 3: How a neuron combines inputs, weights, and bias

Focus: The internal computation of a single neuron

Working of a Neuron: Weights and Bias

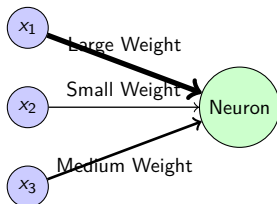
What Exactly Happens Inside a Neuron?

- Recall Lecture 21: hidden layer neurons 'combine' values from the previous layer - today we make this precise
- Every connection between neurons has an associated numeric **WEIGHT**
- Every neuron also has a numeric **BIAS** value
- Together, weights and bias determine exactly how a neuron computes its output



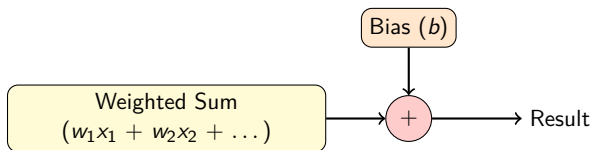
What is a Weight?

- A weight is a number attached to each input connection, representing how important that input is
- A larger weight means that input has a stronger influence on the neuron's output
- A weight close to zero means that input has very little influence
- During training, the network adjusts these weights to improve its predictions



What is a Bias?

- Bias is an extra adjustable number added to a neuron's combined input, independent of any specific input value
- It shifts the neuron's output up or down, giving the model more flexibility
- Analogy: recall Lecture 15's $y = mx + c$ - bias plays a role similar to the intercept 'c'



The Weighted Sum and Learning

The Weighted Sum: Combining Inputs

- Each input is multiplied by its weight, and all these products are added together
- The bias is then added to this sum

Weighted Sum Formula

$$\text{Weighted Sum} = (\text{input}_1 \times \text{weight}_1) + (\text{input}_2 \times \text{weight}_2) + \text{bias}$$

Worked Example: A Neuron with 2 Inputs

- **Inputs:** hours studied = 5, attendance % = 80
- **Weights:** 0.6 for hours studied, 0.02 for attendance; **Bias:** -1

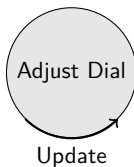
Calculation Steps

$$\begin{aligned}\text{Weighted Sum} &= (5 \times 0.6) + (80 \times 0.02) + (-1) \\ &= 3 + 1.6 - 1 \\ &= 3.6\end{aligned}$$

- This weighted sum (3.6) then passes through an **activation function** (Lecture 23) to produce the neuron's final output

How Does the Network 'Learn' Weights and Bias?

- Initially, weights and bias are set to small random values
- During training, the network compares its predictions to the correct answers
- It then adjusts weights and bias to reduce prediction errors - repeated over many training examples
- This adjustment process is what we mean when we say a neural network 'learns'



Weights and Bias vs Unit 2's Model Parameters

- Recall Lecture 15: Linear Regression learns parameters m (slope) and c (intercept)
- A neural network neuron learns similar parameters: weights (like m , one per input) and bias (like c)
- The key difference: a neural network has **MANY** such neurons, layered together, learning many weights and biases simultaneously

Linear Regression

$$y = mx + c$$

$m \rightarrow$ slope

$c \rightarrow$ intercept

Neural Network Neuron

$$z = (w_1x_1 + w_2x_2 + \dots) + b$$

$w \rightarrow$ weight

$b \rightarrow$ bias

Review and Conclusion

Quick Check: Weights and Bias

Questions

- **Q:** What does a larger weight on an input mean?
- **Q:** What role does bias play in a neuron's computation?
- **Q:** How is bias similar to something learned in Lecture 15?

Key Takeaways

- Weights represent the importance of each input; larger weights mean stronger influence on the neuron's output
- Bias is an adjustable value added to the weighted sum, similar in role to Lecture 15's intercept ' c '
- A neuron computes a weighted sum of inputs plus bias; training adjusts these weights and bias to reduce prediction errors

- **Lecture 23: Activation Functions Part A** - why activation functions are needed, and the Sigmoid function

Weighted Sum $\rightarrow$ **Activation Function** $\rightarrow$ Output

Questions?

Lecture 3.5: Activation Functions - Purpose and Sigmoid Function

DI05000051 - AI & ML

Milav Dabgar

Lecturer, GP Palanpur

July 22, 2026

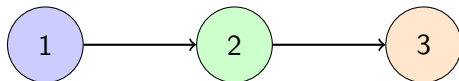
Recap: Lecture 22

- Lecture 22 covered weights (input importance) and bias, and how a neuron computes a weighted sum plus bias.

$$\text{Weighted Sum} = w_1x_1 + w_2x_2 + \cdots + b$$

Today's Agenda

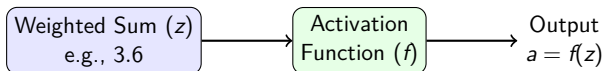
- ① **Part 1:** Why activation functions are needed
- ② **Part 2:** The Sigmoid function
- ③ **Part 3:** Interpreting the Sigmoid graph



3.3 Activation Functions

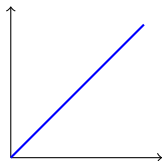
Recall: The Weighted Sum Needs One More Step

- Recall Lecture 22's worked example: weighted sum + bias = 3.6 - but what does '3.6' actually mean as an output?
- Raw weighted sums can be any number - very large, very small, negative, etc.
- An activation function transforms this raw number into a more useful, controlled output

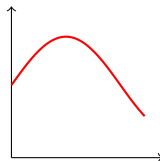


Why Are Activation Functions Necessary?

- They introduce **non-linearity**, letting the network model complex, non-linear patterns (recall Lecture 19's motivation)
- Without activation functions, stacking many layers would mathematically collapse into something no more powerful than Linear Regression (Lecture 15)
- They also often constrain the output into a useful, predictable range (e.g., between 0 and 1)



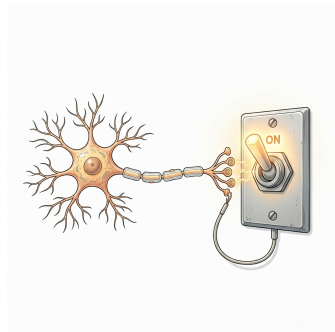
Linear



Non-linear

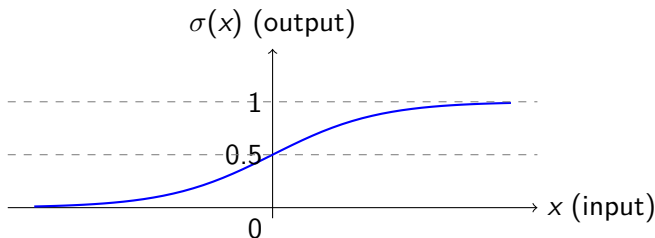
The Sigmoid Function

- One of the earliest and most well-known activation functions
- Takes any real number as input and '**squashes**' it into an output between 0 and 1
- Large positive inputs produce outputs close to 1; large negative inputs produce outputs close to 0
- Inputs near 0 produce outputs near 0.5



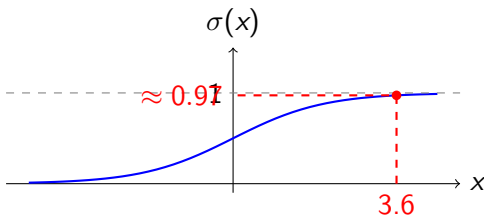
Graphical Shape of the Sigmoid Function

- Produces a smooth 'S-shaped' curve when plotted
- Flattens out near 0 for very negative inputs, and near 1 for very positive inputs
- Rises steeply through the middle, around input = 0



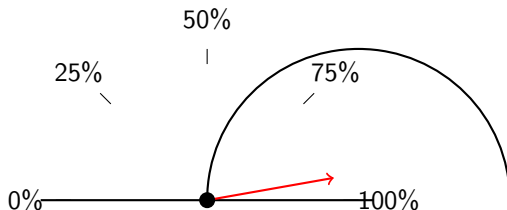
Worked Example: Applying Sigmoid to Lecture 22's Result

- Recall Lecture 22's weighted sum + bias = 3.6
- Since 3.6 is a fairly large positive number, $\text{Sigmoid}(3.6)$ is close to 1 (approximately 0.97)
- This means the neuron's final output strongly signals 'positive'/'yes' for this example (e.g., predicting 'pass')



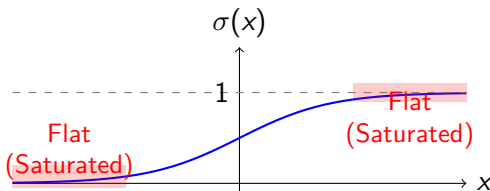
Why Sigmoid's 0-to-1 Range is Useful

- An output between 0 and 1 can be naturally interpreted as a probability
- Example: 0.97 could mean 'the network is 97% confident this student will pass'
- This makes Sigmoid especially popular in the output layer for yes/no (binary) classification problems



A Limitation of Sigmoid

- For very large positive or negative inputs, the curve becomes almost flat
- This means the function becomes very insensitive to further changes in input in those regions
- This limitation partly motivates the alternative activation functions covered next lecture: ReLU and Tanh



Quick Check: Sigmoid Function

Self-Assessment

- **Q:** What is the output range of the Sigmoid function?
- **Q:** What shape does the Sigmoid function have when graphed?
- **Q:** Why is a 0-to-1 output range useful for binary classification?

Key Takeaways

- Activation functions introduce non-linearity, letting neural networks model complex patterns beyond simple linear relationships
- The Sigmoid function squashes any input into an output between 0 and 1, useful as a probability-like signal
- Sigmoid produces a smooth S-shaped curve but saturates (flattens) for very large positive or negative inputs

- **Lecture 24:** Activation Functions Part B - ReLU and Tanh functions, with graphical representation and examples

See you in the next lecture!

Questions?

Lecture 3.6: Activation Functions - ReLU and Tanh

DI05000051 - AI & ML

Milav Dabgar

Lecturer, GP Palanpur

July 22, 2026

Recap

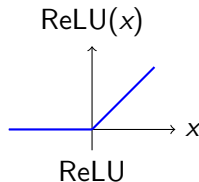
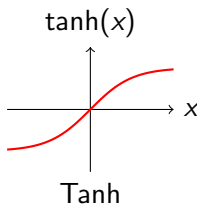
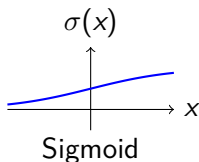
Recap: Lecture 23

- Lecture 23 covered why activation functions are needed and introduced Sigmoid, which squashes inputs into a 0-to-1 range but saturates at extremes.

The ReLU Function

Recall: Sigmoid's Saturation Limitation

- Lecture 23 ended noting Sigmoid flattens out (saturates) for very large positive or negative inputs
- This motivated the search for alternative activation functions - today we cover two widely-used ones
- ReLU and Tanh are the other two activation functions named in the syllabus alongside Sigmoid



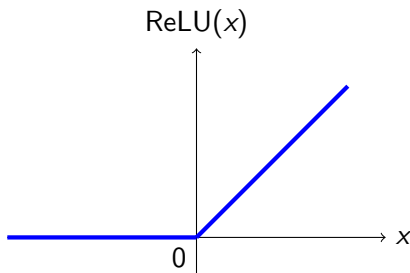
The ReLU Function (Rectified Linear Unit)

- ReLU's rule is very simple: if the input is positive, output the input unchanged; if the input is negative (or zero), output 0
- Mathematically: $\text{ReLU}(x) = \max(0, x)$
- Extremely simple and fast to compute, which is one reason for its popularity in modern neural networks



Graphical Shape of ReLU

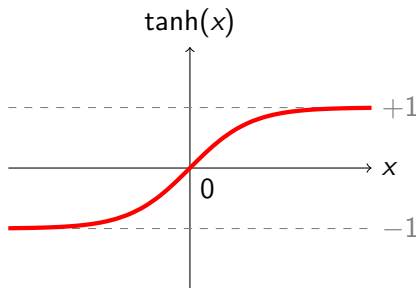
- Flat at 0 for all negative inputs (a horizontal line along the x-axis)
- Then rises in a straight diagonal line for positive inputs (output = input)
- The graph has a distinctive 'bent line' or hockey-stick shape, with the bend at input = 0



The Tanh Function

The Tanh Function (Hyperbolic Tangent)

- Similar S-shape to Sigmoid, but squashes inputs into a range of -1 to +1 instead of 0 to 1
- Large positive inputs approach +1; large negative inputs approach -1; input 0 gives output 0
- Because Tanh is centred around 0, it is often preferred over Sigmoid for hidden layers in practice



Comparing Activation Functions

Sigmoid vs ReLU vs Tanh: Side by Side

- **Sigmoid**: S-curve, output range 0 to 1, can saturate at extremes
- **Tanh**: S-curve, output range -1 to +1, centred at 0, can also saturate at extremes
- **ReLU**: bent-line shape, output range 0 to infinity, does not saturate for positive inputs, very fast to compute

Function	Shape	Output Range	Key Property
Sigmoid	S-curve	$(0, 1)$	Probabilistic output
Tanh	S-curve	$(-1, +1)$	Zero-centred
ReLU	Bent-line	$[0, \infty)$	Fast, non-saturating (for $x > 0$)

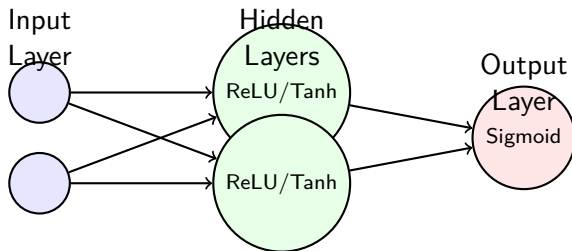
Worked Example: Same Input, Three Functions

- Take an input value of $x = 3.6$ (recall Lecture 22's worked example)
- $\text{Sigmoid}(3.6)$ is approximately 0.97 (close to 1, recall Lecture 23)
- $\text{Tanh}(3.6)$ is approximately 0.998 (close to +1)
- $\text{ReLU}(3.6) = 3.6$ (unchanged, since the input is positive)

Input	Sigmoid(x)	Tanh(x)	ReLU(x)
$x = 3.6$	~ 0.97	~ 0.998	3.6

When Is Each Activation Function Used?

- **Sigmoid**: often used in the output layer for binary (yes/no) classification, due to its probability-like 0-1 output
- **Tanh**: often used in hidden layers when a zero-centred output is preferred
- **ReLU**: very commonly used in hidden layers of modern, deeper networks due to its simplicity and speed



Quick Check: Comparing Activation Functions

- **Q:** What is the output range of ReLU for positive inputs?
- **Q:** How does Tanh's output range differ from Sigmoid's?
- **Q:** Why might ReLU be preferred in hidden layers of a large network?

Summary

Key Takeaways

- ReLU outputs $\max(0, x)$: unchanged for positive inputs, zero for negative inputs - simple and fast
- Tanh mirrors Sigmoid's S-shape but outputs a zero-centred range of -1 to +1
- Sigmoid, ReLU, and Tanh differ in shape, output range, and typical placement within a network (output vs hidden layers)

- **Lecture 25:** Neural Network Architectures Part A - the single layer neural network

Single Layer Network

Questions?

Lecture 3.7: Neural Network Architectures - Single Layer Neural Network

DI05000051 - AI & ML

Milav Dabgar

Lecturer, GP Palanpur

July 22, 2026

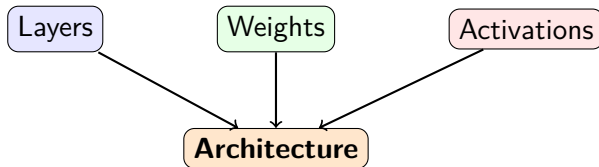
Recap: Lecture 24

- Lecture 24 compared Sigmoid, ReLU, and Tanh activation functions, completing topic 3.3.

Introduction

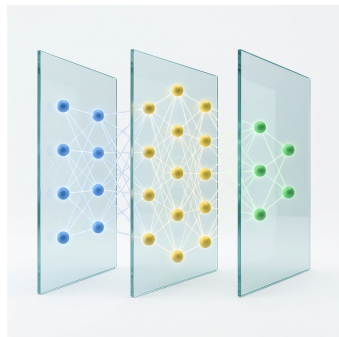
From Components to Architecture

- Recall Lectures 21-24: we now understand layers, weights, bias, and activation functions individually
- Today we look at how these pieces combine into complete network 'architectures'
- The simplest possible architecture is the single layer neural network



What is a Single Layer Neural Network?

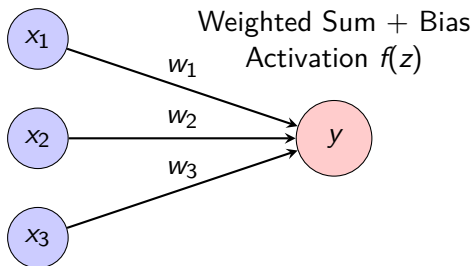
- Consists of just an input layer directly connected to an output layer - NO hidden layer in between
- Each output neuron computes a weighted sum of the inputs, plus bias, then applies an activation function (recall Lectures 22-23)
- Also historically known as a 'Perceptron' - one of the earliest neural network models



Architecture & Decision Making

How a Single Layer Network Makes a Decision

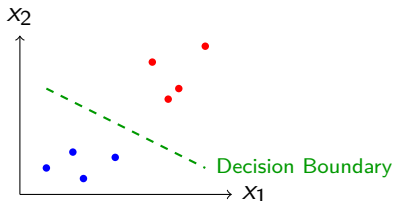
- Inputs are fed directly into the output neuron(s)
- Each output neuron computes: weighted sum of inputs + bias, then applies an activation function (recall Lecture 23's Sigmoid, or Lecture 24's ReLU/Tanh)
- The result is the network's final prediction - there is no intermediate hidden-layer processing



Capabilities & Limitations

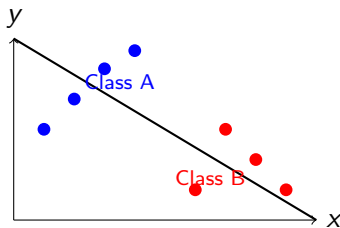
What Can a Single Layer Network Learn?

- Can solve problems where the classes/categories can be separated by a single straight line (or flat plane) - 'linearly separable' problems
- Recall Lecture 15's Linear Regression: a single layer network with no activation (or a simple activation) is conceptually similar in power
- Cannot solve more complex problems where the classes cannot be separated by a single straight line



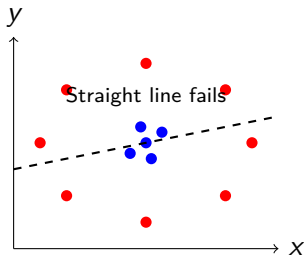
Example: A Solvable Problem for a Single Layer Network

- Classifying points as 'left of a line' vs 'right of a line' on a 2D scatter plot (recall Lecture 7's scatter plots)
- Since one straight line perfectly separates the two classes, a single layer network can learn this easily



Example: A Problem Too Hard for a Single Layer Network

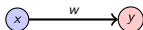
- Some problems require a more complex, curved or non-linear boundary to separate classes (e.g., one class surrounding another)
- A single layer network cannot represent such a boundary - it can only draw a straight line/plane
- This exact limitation is why multi-layer networks (covered next lecture) were developed



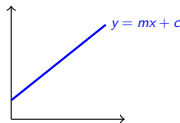
Single Layer Network vs Unit 2 Algorithms

- Conceptually similar in power to Linear Regression (Lecture 15) for numeric prediction, or a simple linear classifier for classification
- Both are limited to relatively simple, roughly linear relationships
- This connects Unit 3 back to Unit 2 - neural networks did not appear from nowhere, they build on the same underlying ideas

Single Layer NN



Linear Regression



Quick Check: Single Layer Networks

Review Questions

- **Q:** Does a single layer neural network have a hidden layer?
- **Q:** What kind of problems can a single layer network solve well?
- **Q:** What is another historical name for a single layer network?

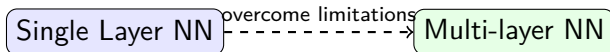
Summary

Key Takeaways

- A single layer neural network has only an input layer directly connected to an output layer (also called a Perceptron)
- It can only solve linearly separable problems, similar in power to Unit 2's simpler linear models
- Its inability to solve non-linearly-separable problems motivates the multi-layer architecture covered next

Lecture 26: Neural Network Architectures Part B

Multi-layer neural networks, their applications, and simple working.



Any Questions?

Lecture 3.8: Neural Network Architectures - Multi-Layer Neural Network

DI05000051 - AI & ML

Milav Dabgar

Lecturer, GP Palanpur

July 22, 2026

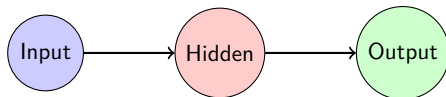
Recap: Lecture 25

- Lecture 25 covered single layer neural networks.
- Input connected directly to output.
- Capable of solving **only linearly separable problems**.

Introduction to Multi-Layer Networks

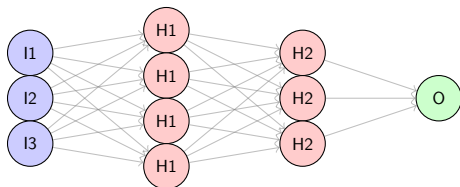
Solving Lecture 25's Limitation

- Recall Lecture 25: a single layer network cannot solve problems that are NOT linearly separable.
- Adding one or more hidden layers (recall Lecture 21) between input and output solves this exact limitation.
- A network with at least one hidden layer is called a **multi-layer neural network**.



What is a Multi-Layer Neural Network?

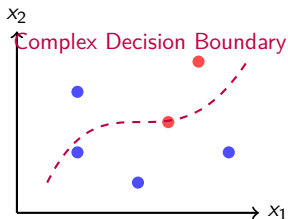
- Contains an input layer, one or more hidden layers, and an output layer (recall Lecture 21's three layer types).
- Sometimes called a **Multi-Layer Perceptron (MLP)**.
- Each hidden layer applies weights, bias, and an activation function (recall Lectures 22-24), passing results to the next layer.



Power of Hidden Layers

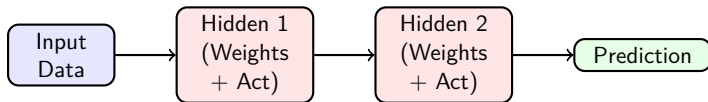
Why Hidden Layers Add Power

- Each hidden layer can learn to detect increasingly complex patterns, building on the patterns detected by the previous layer.
- Recall Lecture 23's non-linearity discussion: stacking layers WITH activation functions lets the network model **curved, complex decision boundaries**.
- This is exactly what solves Lecture 25's 'not linearly separable' problem.



Simple Working of a Multi-Layer Network

- 1 Input layer receives the raw data (recall Lecture 21).
- 2 Each hidden layer computes weighted sums + bias, applies an activation function, and passes results forward (recall Lecture 22-24).
- 3 This repeats through however many hidden layers exist.
- 4 The output layer produces the final prediction.



Applications & Comparison

Applications of Multi-Layer Neural Networks

- **Image recognition** (recall Lecture 20): detecting complex visual patterns requires multiple hidden layers.
- **Speech recognition** (recall Lecture 20): recognising complex audio patterns.
- **Recommendation systems** (recall Lecture 20): capturing subtle, non-linear user preference patterns.

Requires Hidden Layers!

Image
Recognition

Speech
Recognition

Recommendation
Systems

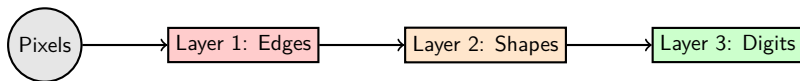
Single vs Multi-Layer Neural Network

Aspect	Single Layer (Lec 25)	Multi-Layer (Today)
Structure	Input directly to output	Input, hidden layer(s), output
Capabilities	Only linearly separable problems	Complex, non-linearly separable problems
Complexity	Simpler and faster	More computation, more data needed

Worked Example & Summary

Worked Example: Digit Recognition Needs Multiple Layers

- Recognising handwritten digits (previewed in Lecture 20, detailed in Lecture 28) requires detecting **edges**, then **shapes**, then **whole digit patterns**.
- Each of these levels of pattern is naturally handled by a different hidden layer.
- A single layer network could not capture this hierarchy of increasingly complex visual patterns.



Quick Check: Single vs Multi-Layer

Questions

- 1 What is the key structural difference between a single layer and a multi-layer network?
- 2 Why can multi-layer networks solve problems that single layer networks cannot?
- 3 Name one application that requires a multi-layer network.

Key Takeaways

- A multi-layer neural network (Multi-Layer Perceptron) adds one or more hidden layers between the input and output layers.
- Hidden layers, combined with non-linear activation functions, let the network solve complex, non-linearly separable problems.
- Multi-layer networks are essential for complex applications like image recognition, speech recognition, and recommendation systems.

Lecture 27 Preview

Building a Simple Neural Network Model - moving from theory to practice, showing how to build a simple neural network model using Python libraries like Scikit-learn or Keras for a simple classification example.

Questions?

Lecture 3.9: Building a Simple Neural Network Model with Python Libraries

DI05000051 - AI & ML

Milav Dabgar

Lecturer, GP Palanpur

July 22, 2026

Recap: Lecture 26

- Lecture 26 completed neural network architecture theory: multi-layer networks solve complex, non-linearly separable problems via hidden layers.

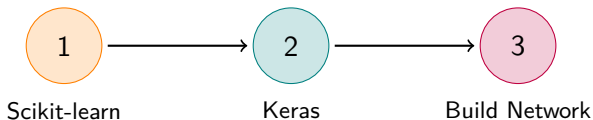
NN Architecture Theory



Solve Complex Non-linear Problem

Today's Agenda

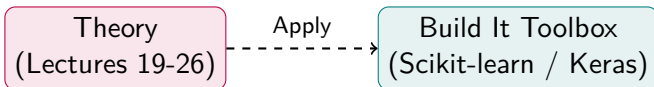
- ① Part 1: Scikit-learn for neural networks
- ② Part 2: Keras - a basic introduction
- ③ Part 3: Steps to build a simple classification neural network



Section: From Theory to Practice

From Theory to Practice

- Recall Lectures 19-26: we've covered neurons, layers, weights/bias, activation functions, and architectures conceptually
- Recall Lecture 4: Scikit-learn is the library already used for KNN, Decision Tree, and Linear Regression (Unit 2)
- Today: how the SAME practical tools let us build neural networks too



Section: Using Libraries

Using Scikit-learn for Neural Networks

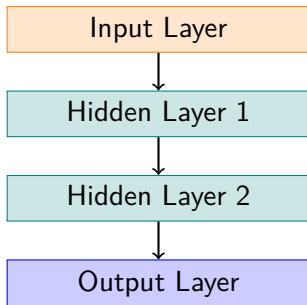
- Scikit-learn (recall Lecture 4) includes a neural network module (e.g., `MLPClassifier` - Multi-Layer Perceptron Classifier)
- Follows the SAME familiar pattern as Unit 2's algorithms: create the model, then call `fit()` to train, then `predict()` to classify new data
- Good choice for simple, small-scale neural network examples in an introductory course

```
graph LR; A[MLPClassifier()] --> B[.fit(X_train, y_train)]; B --> C[.predict(X_test)]
```

`MLPClassifier()` → `.fit(X_train, y_train)` → `.predict(X_test)`

Using Keras (Basic Introduction)

- Keras is a higher-level library specifically designed for building neural networks, often used with TensorFlow
- Allows building networks layer by layer, closely matching the Input-Hidden-Output structure from Lecture 21
- More flexible than Scikit-learn's `MLPClassifier` for complex, deep, or custom network designs



Scikit-learn vs Keras for Neural Networks

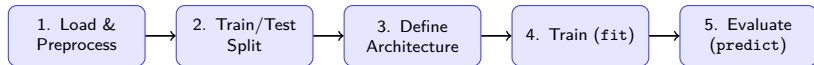
- Both ultimately implement the same underlying concepts: layers, weights, bias, and activation functions (Lectures 21-24)

Library	Best Suited For
Scikit-learn	Simpler syntax, good for small/simple neural networks, consistent with Unit 2's workflow
Keras	More powerful and flexible, better suited for larger, more complex, or deep networks

Section: Practical Steps

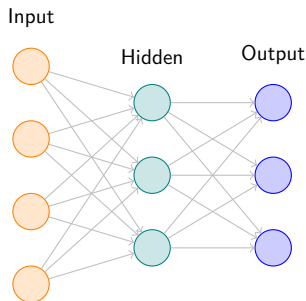
Basic Steps to Build a Classification Neural Network

- 1 Load and preprocess data (recall Lecture 5's Pandas operations)
- 2 Split into training and testing sets (recall Lecture 16's train-test split)
- 3 Define the network architecture: number of hidden layers and neurons (recall Lectures 21, 25-26)
- 4 Train the network (`fit`) on the training data, then evaluate (`predict/score`) on the testing data (recall Lectures 17-18's evaluation measures)



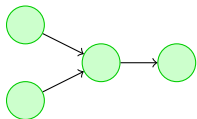
Worked Example: A Simple Classification Network

- Task: classify flowers into species based on measurements (previewed here, detailed with the Iris dataset next lecture)
- Input layer: one neuron per measurement feature
- One small hidden layer with a few neurons and a ReLU or Sigmoid activation (recall Lectures 23-24)
- Output layer: one neuron per flower species, indicating the predicted class

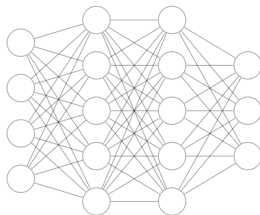


Why Start Simple?

- Small networks with few neurons/layers are faster to train and easier to understand
- They are a good starting point before tackling large-scale problems like full image recognition (recall Lecture 20)
- This course's goal is conceptual mastery of how the pieces fit together, not building production-scale AI systems



Start Here



Large Scale AI

Self-Assessment Questions

- **Q:** Name one Python library that can build a simple neural network for classification.
- **Q:** What are the basic steps to build and train a classification neural network?
- **Q:** How does this practical workflow relate to Unit 2's train-test split (Lecture 16)?

Key Takeaways

- Scikit-learn's `MLPClassifier` and the Keras library are two Python tools for building neural networks
- Building a classification neural network follows a workflow: load/preprocess data, split train/test, define architecture, train, evaluate
- This practical process combines nearly every concept covered so far in Units 2 and 3



New!

Lecture 28

Building a Simple Neural Network Model
*A walkthrough using the Iris dataset
and the Digit recognition dataset*

Questions?

Lecture 3.10: Building a Simple Neural Network Model - Iris and Digit Recognition Datasets

DI05000051 - AI & ML

Milav Dabgar

Lecturer, GP Palanpur

July 22, 2026

Recap & Agenda

Recap: Lecture 27

- Lecture 27 covered the practical workflow for building a classification neural network using Scikit-learn or Keras.

Previously Covered Workflow

Prepare Data → Train-Test Split → Define Architecture → Train → Evaluate

Today's Agenda

- 1 Part 1: The Iris dataset walkthrough
- 2 Part 2: The Digit recognition dataset walkthrough
- 3 Part 3: Unit 3 recap

Two Classic Datasets

Two Classic Example Datasets

- Recall Lecture 27's preview: today we walk through two of the most famous beginner ML/neural network datasets
- **The Iris dataset:** classifying flower species from simple numeric measurements
- **The Digit recognition dataset:** classifying handwritten digit images (recall Lecture 20's preview)

Iris Dataset

150 numeric samples
3 species

Digit Dataset

Image pixels
10 digits (0-9)

The Iris Dataset Walkthrough

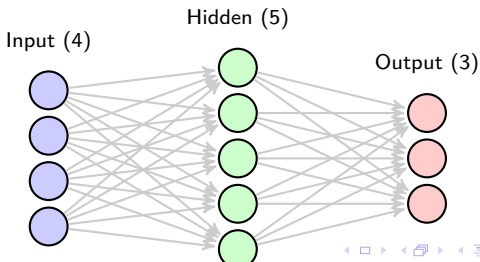
The Iris Dataset: Overview

- A classic dataset of 150 flower samples across 3 species of Iris flower
- Each sample has 4 numeric features: sepal length, sepal width, petal length, petal width
- Task: classify each flower sample into one of the 3 species based on these 4 measurements



Building a Simple Neural Network for Iris Classification

- **Input layer:** 4 neurons (one per measurement feature - recall Lecture 21)
- **Hidden layer:** A small hidden layer (recall Lectures 25-26) to capture relationships between measurements
- **Output layer:** 3 neurons (one per species), the highest-output neuron indicates the predicted species
- Trained using a train-test split (recall Lecture 16) on the 150 labelled samples



The Digit Recognition Dataset Walkthrough

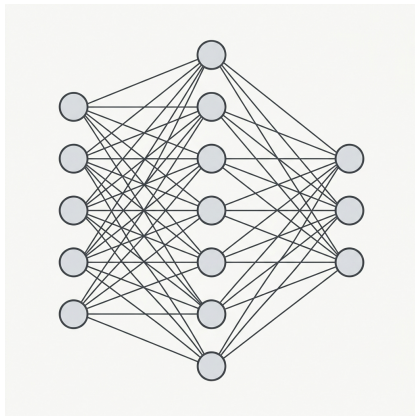
The Digit Recognition Dataset: Overview

- A classic dataset of small images of handwritten digits (0 through 9)
- Each image is represented as a grid of pixel brightness values (recall Lecture 21's input-layer-per-pixel idea)
- Task: classify each image into the correct digit class (0-9) - recall this exact example was previewed in Lecture 20



Building a Simple Neural Network for Digit Recognition

- **Input layer:** one neuron per pixel in the digit image (recall Lecture 21's input layer concept)
- **Hidden layers:** One or more (recall Lecture 26) to detect increasingly complex visual patterns - edges, then shapes, then whole digits
- **Output layer:** 10 neurons (one per digit class 0-9), highest-output neuron is the prediction



Comparisons & Evaluations

Iris Dataset vs Digit Recognition Dataset

Aspect	Iris Dataset	Digit Recognition Dataset
Features	4 numeric features	Many pixel features
Classes	3 classes	10 classes
Complexity	Small dataset, simple patterns	Complex visual patterns
Architecture	Single hidden layer or very small network works well	Benefits more from multiple hidden layers

Both follow the **exact same overall workflow** from Lecture 27: prepare data, split, define architecture, train, evaluate.

Evaluating These Neural Network Models

- After training, we evaluate using the **SAME measures** from Lecture 17-18: Accuracy, Confusion Matrix, Precision, Recall
- Example: a well-trained Iris classifier might achieve well over 90% accuracy, since the dataset is small and clean
- Example: a well-trained Digit recognition model is evaluated similarly across all 10 digit classes

Sample Iris Confusion Matrix

Actual Class	48	0	2
	1	46	3
	0	1	49
	Predicted Class		

Unit 3 Recap

Unit 3 Recap: The Complete Neural Network Journey

- **3.1:** Motivation, biological inspiration, and applications of neural networks
- **3.2:** Structure - layers, weights, and bias
- **3.3:** Activation functions - Sigmoid, ReLU, Tanh
- **3.4:** Architectures - single layer and multi-layer networks
- **3.5:** Practical implementation using Scikit-learn/Keras, with the Iris and Digit recognition datasets

Quick Check: Iris and Digit Recognition

Questions

- **Q:** How many input features does the Iris dataset have, and how many output classes?
- **Q:** Why might the Digit recognition task benefit from more hidden layers than the Iris task?
- **Q:** Which evaluation measures from Unit 2 apply to a neural network's classification results?

Summary & Next Steps

Key Takeaways

- The Iris dataset (4 features, 3 classes) is a classic simple example for a small classification neural network
- The Digit recognition dataset (pixel inputs, 10 classes) is a classic example benefiting from multiple hidden layers
- Unit 3 progressed from neuron motivation and biology, through structure and activation functions, to architectures and hands-on practice

Lecture 29

Introduction to Natural Language Processing - Unit 4 begins with what NLP is and why it matters in AI

Questions?

Thank you for your attention!

Lecture 4.1: Introduction to Natural Language Processing

DI05000051 - AI & ML

Milav Dabgar

Lecturer, GP Palanpur

July 22, 2026

Recap

- Lecture 28 closed Unit 3 with Iris and Digit recognition dataset walkthroughs and a full Unit 3 recap.

Unit 3 Recap
Neural Networks: Motivation to Implementation

Today's Agenda

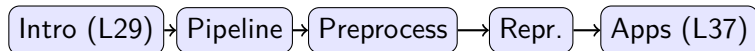
- Part 1: What is NLP?
- Part 2: Why NLP matters in AI
- Part 3: Major NLP applications

- 1 What is NLP?
- 2 Why NLP matters in AI
- 3 Major NLP applications

What is NLP?

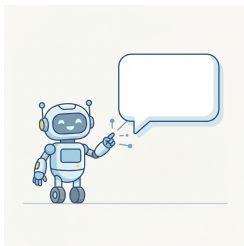
Welcome to Unit 4: Natural Language Processing

- Recall Lecture 20's preview: speech recognition converts audio to text - Unit 4 covers what happens with that text next
- Unit 4 (9 lectures) covers NLP definition, pipeline, preprocessing, text representation, and simple applications
- Human language (text and speech) is one of the richest, most challenging kinds of data for AI to process



What is Natural Language Processing?

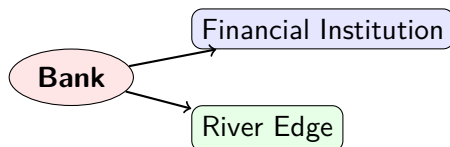
- NLP = the branch of AI focused on enabling computers to understand, interpret, and generate human language
- 'Natural language' means everyday human language (English, Hindi, Gujarati, etc.), as opposed to formal programming languages
- Recall Lecture 1's chatbot example - that is a direct, everyday application of NLP



Why NLP matters in AI

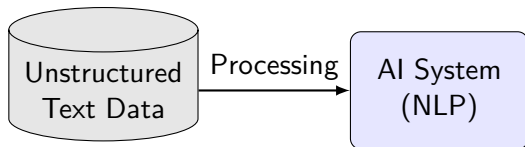
Why is NLP Challenging?

- Human language is ambiguous: the same word can mean different things in different contexts
- Language has enormous variability: slang, grammar variations, sarcasm, multiple languages
- Unlike numeric data (recall Unit 2's algorithms), raw text is not naturally numeric - it must be converted (Lecture 34-35's topic)



Why is NLP Important in AI?

- Language is the most natural way humans communicate - NLP lets AI systems interact with people directly
- Enables automating huge volumes of text-based work: customer support, translation, content analysis
- A large fraction of the world's information exists as unstructured text (documents, social media, reviews)



Major NLP applications

NLP Application: Chatbots

- Chatbots understand user text/voice input and generate relevant, natural-sounding replies
- Used in customer support, virtual assistants, and educational tutoring (recall Lecture 2's education application)
- This exact chatbot example was first introduced all the way back in Lecture 1

User

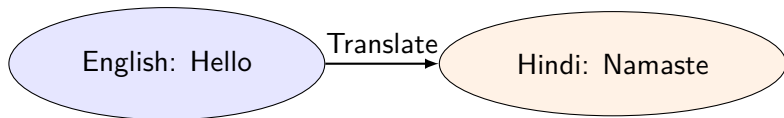
How can I reset my password?

Chatbot (AI)

I can help with that.
Please click the link sent to your email.

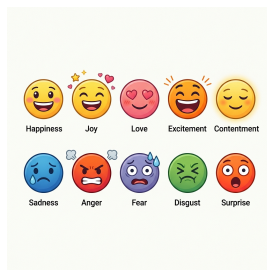
NLP Application: Machine Translation

- Automatically translates text from one language to another (e.g., English to Hindi)
- Requires understanding grammar, meaning, and context in both languages
- Powers tools like translation apps and multilingual customer support systems



NLP Application: Sentiment Analysis

- Automatically determines whether a piece of text expresses positive, negative, or neutral sentiment
- Used to analyse product reviews, social media posts, and customer feedback
- This exact application returns in full detail in Lecture 36 as a hands-on NLP task



NLP Application: Voice Assistants

- Voice assistants (e.g., smartphone or smart-speaker assistants) combine speech recognition (Lecture 20) with NLP understanding
- They interpret spoken commands/questions and generate helpful spoken or text responses
- A great example of Units 3 and 4 working together: neural networks for speech-to-text, then NLP for understanding



Self-Assessment

- **Q:** What does NLP stand for, and what is its goal?
- **Q:** Name one NLP application that analyses whether text is positive or negative.
- **Q:** How do voice assistants combine ideas from Unit 3 and Unit 4?

Summary

Key Takeaways

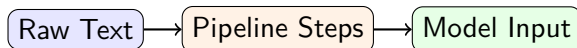
- NLP is the branch of AI enabling computers to understand, interpret, and generate human language
- NLP is challenging because language is ambiguous, variable, and not naturally numeric
- Major NLP applications include chatbots, machine translation, sentiment analysis, and voice assistants

NLP unlocks unstructured text for AI

Language → Processing →
Understanding → Action

Next Lecture Preview

- Lecture 30: The NLP Processing Pipeline - the basic steps involved in any NLP task, from text collection to model building



Questions?

Lecture 4.2: NLP Processing Pipeline

DI05000051 - AI & ML

Milav Dabgar

Lecturer, GP Palanpur

July 22, 2026

Recap: Lecture 29

- Lecture 29 defined NLP and covered its major applications: chatbots, machine translation, sentiment analysis, and voice assistants.

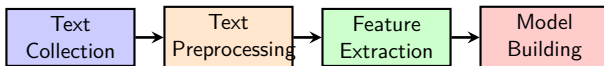
Today's Agenda

- Part 1: Text collection
- Part 2: Text preprocessing
- Part 3: Feature extraction
- Part 4: Model building

NLP Processing Pipeline

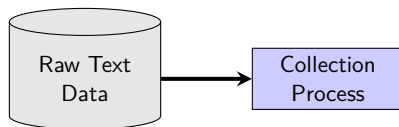
Behind Every NLP Application: A Common Pipeline

- Recall Lecture 29's applications - each one (chatbots, translation, sentiment analysis) follows the same general processing steps
- Recall Lecture 9's general ML workflow (collect, preprocess, train, test) - NLP has an analogous, text-specific pipeline
- Today's four steps: text collection, text preprocessing, feature extraction, model building



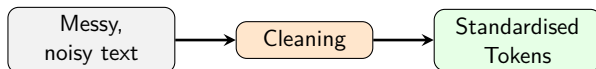
Step 1: Text Collection

- Gathering raw text data relevant to the NLP task
- Sources: customer reviews, social media posts, chat logs, documents, transcribed speech (recall Lecture 20/29)
- Similar in spirit to Lecture 9's 'data collection' stage, but specifically for text



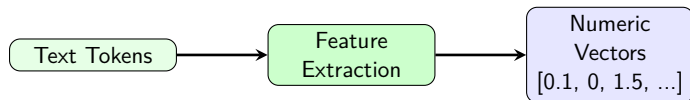
Step 2: Text Preprocessing

- Cleaning and standardising raw text before further processing
- Includes tokenization, removing unnecessary words, and normalising word forms
- This entire step is explored in much greater depth across Lectures 31-33 (topic 4.3)



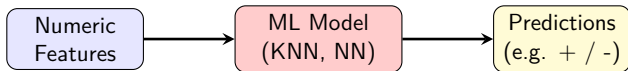
Step 3: Feature Extraction

- Converting cleaned text into a numeric form that a machine learning model can actually use
- Recall Unit 2/3's algorithms all operate on numbers - text must be transformed into numbers first
- This step is explored in depth in Lectures 34-35 (topic 4.4): Bag of Words, TF, and TF-IDF



Step 4: Model Building

- Using the extracted numeric features to train a model for the specific NLP task
- Could reuse Unit 2's classification algorithms (e.g., classifying sentiment as positive/negative) or Unit 3's neural networks
- This closes the loop: NLP is not an entirely separate skill, but an application domain for concepts already learned



Comparisons & Examples

NLP Pipeline vs General ML Workflow

- **General ML (Lecture 9):** Data collection → Preprocessing → Model training → Model testing
- **NLP pipeline (today):** Text collection → Text preprocessing → Feature extraction → Model building
- NLP's 'feature extraction' step is a text-specific addition, needed because raw text isn't naturally numeric

Workflow	Step 1	Step 2	Step 3	Step 4
General ML	Data Collection	Preprocessing	Model Training & Testing	
NLP	Text Collection	Text Preprocessing	Feature Extraction	Model Building

Worked Example: Sentiment Analysis Pipeline

- **Text collection:** gather product reviews (recall Lecture 29's sentiment analysis application)
- **Text preprocessing:** clean and tokenize the review text (Lectures 31-33)
- **Feature extraction:** convert cleaned text into numeric features like word frequencies (Lectures 34-35)
- **Model building:** train a classifier to predict positive/negative sentiment (Lecture 36)

Why Order Matters in the Pipeline

- Feature extraction requires clean, preprocessed text - skipping preprocessing leads to noisy, unreliable features
- Model building requires numeric features - skipping feature extraction means models cannot process the data at all
- Just like Unit 2's workflow (Lecture 9), each pipeline stage depends on the previous stage being done well

Quick Check: The NLP Pipeline

- **Q:** List the four basic steps of the NLP processing pipeline in order
- **Q:** Which pipeline step converts text into numbers?
- **Q:** How does the NLP pipeline relate to Lecture 9's general ML workflow?

Summary

Key Takeaways

- The NLP pipeline has four steps: text collection, text preprocessing, feature extraction, and model building
- This pipeline mirrors Unit 2's general ML workflow, with an added feature extraction step to convert text into numbers
- Each stage depends on the previous one being done well, just like the general ML workflow

- **Lecture 31:** Text Preprocessing Part A - the purpose of preprocessing and tokenization (splitting text into words and sentences)

Questions?

Lecture 4.3: Text Preprocessing - Introduction and Tokenization

DI05000051 - AI & ML

Milav Dabgar

Lecturer, GP Palanpur

July 22, 2026

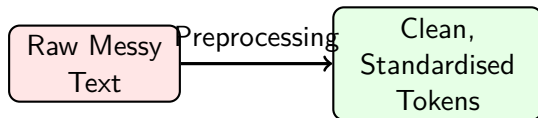
Recap: Lecture 30

- Lecture 30 introduced the NLP pipeline's four steps, flagging that text preprocessing (step 2) would be covered in depth across 3 lectures.

Why Preprocessing is Needed

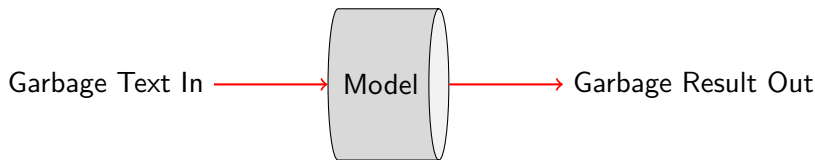
Why Does Text Need Preprocessing?

- Recall Lecture 30: preprocessing is Step 2 of the NLP pipeline, cleaning raw text before feature extraction
- Raw text is messy: inconsistent capitalisation, punctuation, filler words, and varying word forms
- Preprocessing standardises text so that later steps (feature extraction, Lecture 34-35) work reliably



The Purpose of Preprocessing: An Analogy

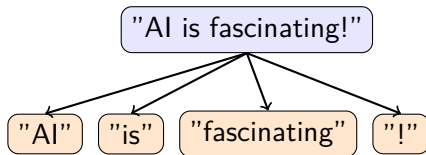
- Recall Lecture 5's Pandas data cleaning discussion - preprocessing is the text equivalent of cleaning a numeric dataset
- Just as 'garbage in, garbage out' applied to ML models (Lecture 5, Lecture 9), it applies equally to NLP models
- Well-preprocessed text leads to more accurate, reliable feature extraction and modelling



Word Tokenization

What is Tokenization?

- Tokenization = breaking text into smaller units called 'tokens' (usually words or sentences)
- It is usually the very first preprocessing step applied to raw text
- Without tokenization, a computer sees text as one long string of characters, not meaningful units



Word Tokenization

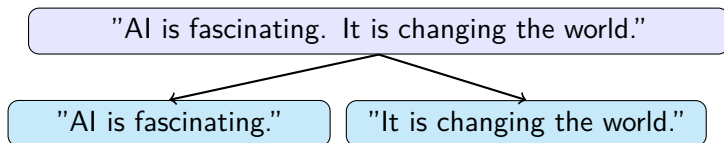
- Splits a piece of text into individual words (and sometimes punctuation marks)
- Example: "AI is fascinating!" becomes tokens: ["AI", "is", "fascinating", "!"]
- Word tokenization is the foundation for almost all further text processing steps

AI is fascinating !

Sentence Tokenization

Sentence Tokenization

- Splits a larger piece of text (a paragraph or document) into individual sentences
- Example: "AI is fascinating. It is changing the world." becomes 2 sentence tokens
- Useful when a task needs to process or analyse text sentence by sentence, rather than as one big block



Word Tokenization vs Sentence Tokenization

- Both types of tokenization can be applied together, depending on the task's needs

Word Tokenization	Sentence Tokenization
Splits text into individual words	Splits text into individual sentences
Used for word-level analysis (e.g., word frequency, Lecture 36)	Used for sentence-level analysis (e.g., processing one review sentence at a time)
Output: ["AI", "is", "fun"]	Output: ["AI is fun.", "Try it out."]

Worked Example: Tokenizing a Product Review

Original text: "Great product. Fast delivery! Highly recommend."

Sentence Tokens:

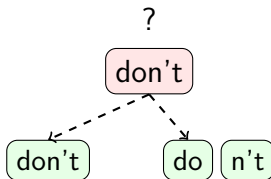
- "Great product."
- "Fast delivery!"
- "Highly recommend."

Word Tokens (1st sentence):

- "Great"
- "product"
- "."

Challenges in Tokenization

- Punctuation handling: should "don't" be one token or split into "do" and "n't"?
- Different languages and scripts may require different tokenization rules
- Real-world tokenizers (like those in the NLTK library, covered in Lecture 33) handle many such edge cases automatically



Quick Check: Tokenization

Questions:

- 1 What is tokenization, and why is it usually the first preprocessing step?
- 2 Tokenize this sentence into words: "Machine learning is powerful."
- 3 What is the difference between word tokenization and sentence tokenization?

Key Takeaways

- Text preprocessing cleans and standardises raw text, applying the same 'garbage in, garbage out' logic as numeric data cleaning
- Tokenization breaks text into smaller units - words or sentences - and is usually the first preprocessing step
- Word tokenization supports word-level analysis; sentence tokenization supports sentence-level analysis

Next Lecture Preview

- Lecture 32: Text Preprocessing Part B - stop word removal, stemming, and lemmatization

Next Up: Refining tokens further!

Questions?

Lecture 4.4: Text Preprocessing - Stop Word Removal, Stemming, and Lemmatization

DI05000051 - AI & ML

Milav Dabgar

Lecturer, GP Palanpur

July 22, 2026

Recap: Lecture 31

- Lecture 31 covered tokenization: splitting text into word tokens and sentence tokens as the first preprocessing step.

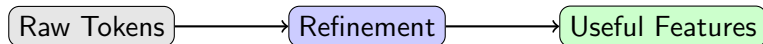
Raw Text $\rightarrow$ Tokens

Today's Agenda

- ① **Part 1:** Stop word removal
- ② **Part 2:** Stemming
- ③ **Part 3:** Lemmatization

After Tokenization: Refining the Tokens

- Recall Lecture 31: tokenization produces individual word tokens from raw text
- Not all tokens are equally useful - some are common filler words; others need to be standardised to a base form
- Today's three techniques operate on tokens to make them more useful for feature extraction (Lectures 34-35)



Stop Word Removal

What Are Stop Words?

- Stop words are extremely common words that usually carry little unique meaning on their own
- Examples: "the", "is", "a", "and", "of", "in"
- These words appear in almost every sentence, so they rarely help distinguish one piece of text from another

of is
the and
a in

Stop Word Removal

- The process of removing stop words from a token list, keeping only the more meaningful words
- Reduces noise and dataset size before feature extraction, often improving model performance and speed

Example

Original: "The product is great and fast"

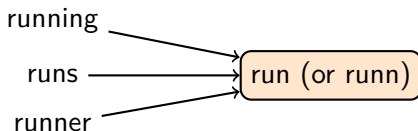


After removal: ["product", "great", "fast"]

Stemming and Lemmatization

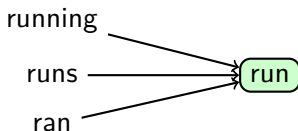
What is Stemming?

- Stemming reduces a word to its 'stem' or root form by simply chopping off common word endings
- Example: "running", "runs", "runner" might all be stemmed to "run" or a similar rough root
- Fast and rule-based, but can sometimes produce a root that isn't a real dictionary word



What is Lemmatization?

- Lemmatization also reduces a word to its base form, but uses proper linguistic/dictionary knowledge to do so correctly
- Example: "running", "runs", "ran" are all lemmatized to "run"
 - a valid dictionary word
- More accurate than stemming, but typically slower since it needs to know word grammar/context



Comparison & Application

Stemming vs Lemmatization

- Both aim to group different forms of the same word together, just with different levels of accuracy and speed

Aspect	Stemming	Lemmatization
Approach	Fast, rule-based	Uses linguistic knowledge
Speed	Fast	Slower
Output	May produce non-dictionary roots	Always valid dictionary words

Worked Example: Full Refinement Pipeline

Original tokens: ["The", "product", "is", "running", "great"]



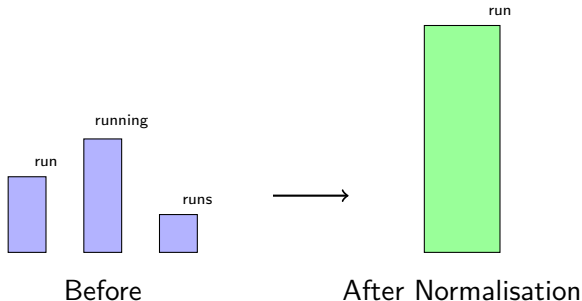
Stop word removal: ["product", "running", "great"]



Stemming / Lemmatization: ["product", "run", "great"]

Why These Techniques Matter for Feature Extraction

- Recall Lecture 30: feature extraction (Lectures 34-35) converts text into numeric features, often based on word counts
- Without these techniques, "run", "running", and "runs" would be counted as three separate, unrelated words
- Stop word removal, stemming, and lemmatization group and focus these counts on meaningful words



Quick Check: Preprocessing Techniques

- ① **Q:** Why are common words like 'the' and 'is' often removed before feature extraction?
- ② **Q:** What is the key difference between stemming and lemmatization?
- ③ **Q:** Stem or lemmatize (informally) the words: 'studies', 'studying', 'studied'

Summary

Key Takeaways

- Stop word removal eliminates common, low-meaning words like 'the' and 'is' from token lists
- Stemming crudely chops words to a root form; lemmatization uses linguistic knowledge to produce a valid dictionary base form
- These techniques group different forms of the same word, improving later feature extraction

Lecture 33

Text Preprocessing Part C - Part-of-Speech (POS) tagging and using the NLTK library in Python

Questions?

Lecture 4.5: Text Preprocessing - POS Tagging and NLTK

DI05000051 - AI & ML

Milav Dabgar

Lecturer, GP Palanpur

July 22, 2026

Recap: Lecture 32

- Lecture 32 covered stop word removal, stemming, and lemmatization for refining word tokens.

Previous Lecture Recap

Lecture 32 covered stop word removal, stemming, and lemmatization - techniques that filter and normalise word tokens after tokenization.

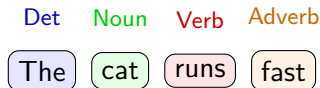
Today's Agenda

- Part 1: What is POS tagging?
- Part 2: Common POS tags with examples
- Part 3: The NLTK library in Python

Part 1: What is POS tagging?

One More Preprocessing Technique: POS Tagging

- Recall Lectures 31-32: tokenization, stop word removal, stemming, and lemmatization all operate on individual words
- Part-of-Speech (POS) tagging adds grammatical information: what ROLE does each word play in a sentence?
- This is the final preprocessing technique named in the syllabus, completing topic 4.3



What is Part-of-Speech (POS) Tagging?

- POS tagging assigns a grammatical category (noun, verb, adjective, etc.) to each word in a sentence
- Example: "The cat runs fast" → The (determiner), cat (noun), runs (verb), fast (adverb)
- Helps a system understand sentence structure and word roles, not just the words themselves

Part 2: Common POS tags with examples

Common POS Tags

- Noun (N): names a person, place, thing, or idea (e.g., "cat", "product")
- Verb (V): describes an action or state (e.g., "runs", "is")
- Adjective (ADJ): describes a noun (e.g., "great", "fast")
- Adverb (ADV): describes a verb, adjective, or another adverb (e.g., "quickly", "very")

Noun (N)

e.g., cat

Verb (V)

e.g., runs

Adjective (ADJ)

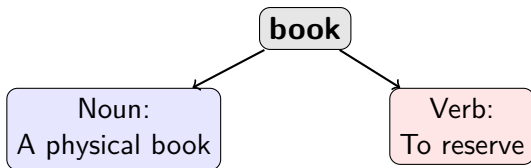
e.g., fast

Adverb (ADV)

e.g., quickly

Why POS Tagging is Useful

- Helps distinguish word meaning by role - e.g., "book" as a noun (a book) vs a verb (to book a ticket)
- Improves the quality of later feature extraction (Lectures 34-35) by understanding word context
- Useful in more advanced NLP applications like grammar checking and information extraction



Part 3: The NLTK library in Python

Introducing the NLTK Library

- NLTK = Natural Language Toolkit, a widely used Python library for NLP tasks
- Provides ready-made functions for tokenization, stop word removal, stemming, lemmatization, and POS tagging
- Recall Lecture 4's library overview - NLTK plays the same specialised role for NLP that Scikit-learn plays for classical ML

NLTK
Natural Language Toolkit



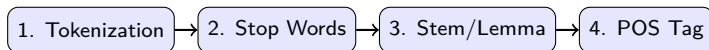
`word_tokenize` | `stopwords` | `PorterStemmer` | `pos_tag`

Worked Example: NLTK Performing Preprocessing

- `nltk.word_tokenize(text)` → performs word tokenization (recall Lecture 31)
- `nltk.corpus.stopwords` → provides a ready-made list of stop words for removal (recall Lecture 32)
- `nltk.stem.PorterStemmer()` → performs stemming (recall Lecture 32)
- `nltk.pos_tag(tokens)` → performs POS tagging (today's topic)

The Complete Text Preprocessing Toolkit

- 1. Tokenization (word/sentence) - Lecture 31
- 2. Stop word removal - Lecture 32
- 3. Stemming or Lemmatization - Lecture 32
- 4. POS tagging - today
- All available through the NLTK library in Python



What Comes After Preprocessing?

- Recall Lecture 30's NLP pipeline: after preprocessing comes feature extraction
- Preprocessed, POS-tagged, cleaned tokens are now ready to be converted into numeric features
- Lectures 34-35 cover exactly this: Bag of Words, Term Frequency, and TF-IDF



Quick Check: POS Tagging and NLTK

- **Q:** What grammatical categories are commonly identified by POS tagging?
- **Q:** Give one NLTK function and the preprocessing technique it performs
- **Q:** List, in order, the four text preprocessing techniques covered across Lectures 31-33

Summary

Key Takeaways

- POS tagging assigns grammatical roles (noun, verb, adjective, adverb) to words, clarifying their meaning and function
- The NLTK library provides ready-made Python functions for tokenization, stop word removal, stemming/lemmatization, and POS tagging
- Text preprocessing (Lectures 31-33) transforms raw text into clean, tagged tokens ready for feature extraction

- Lecture 34: Text Representation Part A - Bag of Words (BoW) and Term Frequency (TF), converting text into numeric features

Questions?

Lecture 4.6: Text Representation - Bag of Words and Term Frequency

DI05000051 - AI & ML

Milav Dabgar

Lecturer, GP Palanpur

July 22, 2026

Recap: Lecture 33

- Lecture 33 completed text preprocessing (tokenization, stop word removal, stemming/lemmatization, POS tagging) using the NLTK library.

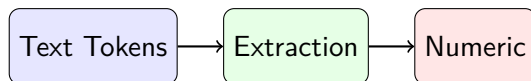
Today's Agenda

- 1 Part 1: Why convert text to numbers?
- 2 Part 2: Bag of Words (BoW)
- 3 Part 3: Term Frequency (TF)

Why Convert Text into Numbers?

Why Convert Text into Numbers?

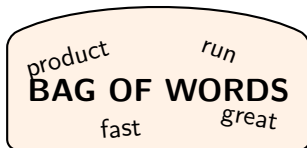
- Recall Lecture 30's NLP pipeline: feature extraction is the step that converts preprocessed text into numeric form
- Recall Unit 2/3: every ML algorithm (KNN, Decision Tree, Linear Regression, Neural Networks) works on NUMBERS, not raw words
- Text representation techniques bridge this gap between clean text tokens (Lectures 31-33) and numeric ML models



Bag of Words (BoW)

What is Bag of Words (BoW)?

- BoW represents a document by counting how many times each word appears, ignoring word order and grammar
- Each unique word across all documents becomes a 'feature' (column); each document becomes a row of word counts
- 'Bag' reflects that word order is thrown away - only the presence/count of words matters



Worked Example: Building a Bag of Words

Documents:

- Document 1 (recall Lecture 32's example): "product run great"
- Document 2: "product great fast"

Vocabulary (unique words): [product, run, great, fast]

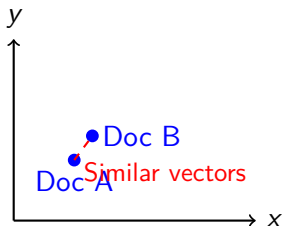
Document	product	run	great	fast
Doc 1	1	1	1	0
Doc 2	1	0	1	1

Vectors:

- Doc 1 vector: [1, 1, 1, 0]
- Doc 2 vector: [1, 0, 1, 1]

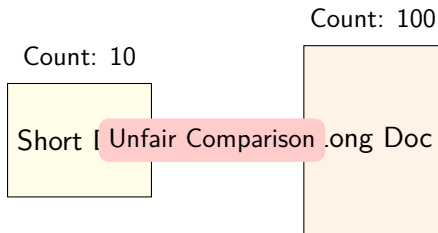
What Does a BoW Vector Represent?

- Each document becomes a row of numbers (a vector) - one number per vocabulary word
- Similar documents (sharing many words) produce similar-looking vectors
- This numeric vector can now be fed into Unit 2's classifiers (e.g., KNN, recall Lecture 13) or Unit 3's neural networks



Limitation of Plain Word Counts

- A word appearing many times in a document isn't necessarily always more 'important'
- Very common words (even after stop word removal, Lecture 32) can still dominate raw counts in longer documents
- This motivates normalising counts, which is exactly what Term Frequency (TF) does



Term Frequency (TF)

What is Term Frequency (TF)?

- TF measures how frequently a word appears in a document, **RELATIVE** to the document's total word count
- This normalises for document length, so a word's importance isn't just due to a longer document having more words overall

TF Formula

$$\text{TF}(\text{word}) = \frac{\text{number of times word appears in document}}{\text{total number of words in document}}$$

Worked Example: Computing Term Frequency

- **Document:** "product run great product" (4 words total)
- **Target Word:** "product"
- Count of "product" = 2

$$\text{TF}(\text{"product"}) = \frac{2}{4} = 0.5$$

Bag of Words vs Term Frequency

Aspect	BoW	TF
Measurement	Raw word counts	Normalised word counts
Doc Length	Favors longer docs	Fairer measure
Similarities	Both ignore word order and grammar - focus purely on WHICH words appear and HOW OFTEN	

Quick Check: BoW and TF

- ① **Q:** What does a Bag of Words representation ignore about the original text?
- ② **Q:** Why does Term Frequency divide by the total word count in a document?
- ③ **Q:** For a 10-word document where 'AI' appears twice, what is $TF('AI')$?

Key Takeaways

- Text representation converts preprocessed text into numeric features usable by ML algorithms from Units 2-3
- Bag of Words represents documents as vectors of raw word counts across a shared vocabulary, ignoring word order
- Term Frequency normalises word counts by document length, giving a fairer measure of word importance

Lecture 35: Text Representation Part B - TF-IDF

Lecture 35 completes text representation with TF-IDF, which further refines Term Frequency by also considering how rare or common a word is across all documents.

Questions?

Lecture 4.7: Text Representation - TF-IDF

DI05000051 - AI & ML

Milav Dabgar

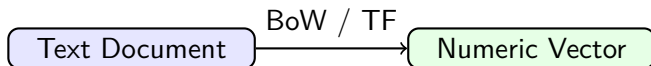
Lecturer, GP Palanpur

July 22, 2026

Recap

Recap: Lecture 34

- Lecture 34 covered Bag of Words and Term Frequency, converting text into numeric word-count-based features.



Today's Agenda

Today's Agenda

- 1 Part 1: The limitation TF alone still has
- 2 Part 2: Inverse Document Frequency (IDF)
- 3 Part 3: TF-IDF and its applications

A Remaining Limitation of TF

A Remaining Limitation of TF

- Recall Lecture 34: TF measures how often a word appears within ONE document
- But some words (even after stop word removal, Lecture 32) are common across almost ALL documents, and thus not very distinctive
- Example: in a set of product reviews, the word "product" might appear frequently in EVERY review - not helpful for telling reviews apart

Doc 1: **product**

Doc 2: **product**

Doc 3: **product**

The word "product" is everywhere, diluting its usefulness!

What is Inverse Document Frequency (IDF)?

What is Inverse Document Frequency (IDF)?

- IDF measures how RARE or common a word is across the entire collection of documents (not just one document)
- Words that appear in many documents get a LOW IDF score (less distinctive)
- Words that appear in only a few documents get a HIGH IDF score (more distinctive)

Rare Word

Appears in few docs

HIGH IDF

Common Word

Appears in many docs

LOW IDF

What is TF-IDF?

What is TF-IDF?

- $\text{TF-IDF} = \text{Term Frequency} \times \text{Inverse Document Frequency}$
- Combines HOW OFTEN a word appears in a document (TF, recall Lecture 34) with HOW RARE it is across all documents (IDF)
- A word gets a high TF-IDF score if it appears often in THIS document but rarely in OTHER documents - i.e., it is distinctive

The TF-IDF Formula

$$\text{TF-IDF} = \text{TF} \times \text{IDF}$$

Worked Example: TF-IDF in Product Reviews

- Word "product" appears in nearly every review $\rightarrow$ low IDF $\rightarrow$ low TF-IDF, even if its TF is high
- Word "waterproof" appears often in only a few specific reviews $\rightarrow$ high IDF $\rightarrow$ high TF-IDF in those reviews
- TF-IDF correctly highlights "waterproof" as more distinctive/informative than "product" for those specific reviews

Word	TF (local)	IDF (global)	TF-IDF (combined)
product	High	Low	Low
waterproof	High	High	High

Comparisons and Applications

BoW vs TF vs TF-IDF

- **BoW (Lecture 34):** raw word count per document, ignoring document length and word rarity
- **TF (Lecture 34):** word count normalised by document length, still ignoring word rarity across documents
- **TF-IDF (today):** combines document-level frequency with cross-document rarity, highlighting distinctive words

Technique	What it does
BoW	Counts words (ignores length & rarity)
TF	Normalizes by length (ignores rarity)
TF-IDF	Highlights distinctive words (considers length & rarity)

Applications of TF-IDF

- **Search engines:** ranking documents by how relevant/distinctive their word matches are to a search query
- **Document classification:** identifying which distinctive words characterise different document categories
- **Keyword extraction:** highlighting a document's most distinctive words - previewed further in Lecture 37

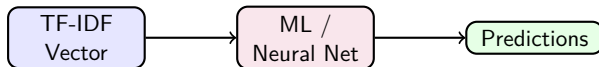
Search Engines

Document
Classification

Keyword Ex-
traction

TF-IDF Feeding into ML Models

- Just like BoW vectors (Lecture 34), TF-IDF scores form a numeric vector representation of each document
- This TF-IDF vector can be used as input to Unit 2's classifiers or Unit 3's neural networks
- TF-IDF often performs better than plain BoW because it emphasises more meaningful, distinctive words



Summary

Quick Check: TF-IDF

- **Q:** What does a HIGH IDF value indicate about a word?
- **Q:** Why might TF-IDF give the word 'product' a low score across many product reviews?
- **Q:** Name one application where TF-IDF is commonly used



Key Takeaways

- Inverse Document Frequency (IDF) measures how rare a word is across an entire document collection
- $\text{TF-IDF} = \text{TF} \times \text{IDF}$, highlighting words that are frequent in one document but rare across others - i.e., distinctive words
- TF-IDF is widely used in search engines, document classification, and keyword extraction

Next Lecture Preview

- **Lecture 36:** Simple NLP Applications Part A - sentiment analysis of text and word frequency analysis, applying everything learned so far

Coming up: Sentiment Analysis & Word Frequency Analysis!

Questions?

Thank you!

Lecture 4.8: Simple NLP Applications - Sentiment and Word Frequency Analysis

DI05000051 - AI & ML

Milav Dabgar

Lecturer, GP Palanpur

July 22, 2026

Recap: Lecture 35

- Lecture 35 completed text representation techniques with TF-IDF, highlighting distinctive words across a document collection.

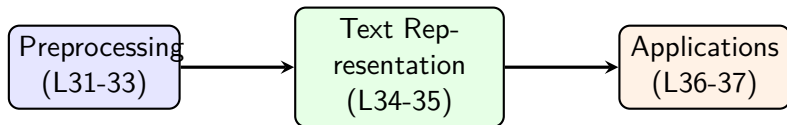
Today's Agenda

- 1 Part 1: Sentiment analysis of text
- 2 Part 2: Word frequency analysis
- 3 Part 3: Putting the full pipeline together

From Building Blocks to Complete Applications

From Building Blocks to Complete Applications

- Recall Lectures 29-35: we now have the full toolkit - NLP applications (29), the pipeline (30), preprocessing (31-33), and text representation (34-35)
- Lectures 36-37 apply this complete toolkit to build simple, real NLP applications end to end
- Today: sentiment analysis (previewed in Lecture 29) and word frequency analysis



Sentiment Analysis of Text

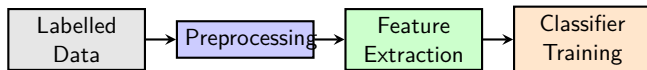
Sentiment Analysis of Text: Recap and Deep Dive

- Recall Lecture 29's preview: sentiment analysis determines whether text expresses positive, negative, or neutral sentiment
- Example use case: automatically analysing product reviews (recall Lectures 31-35's running example) for overall customer satisfaction
- A classic supervised classification problem (recall Lecture 11): input = text features, output = sentiment label



Building a Sentiment Analysis Pipeline

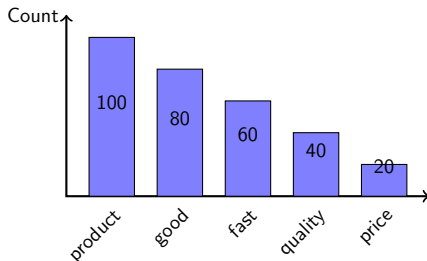
- 1 **Collect labelled text data:** reviews already tagged positive/negative (recall Lecture 30's pipeline)
- 2 **Preprocess the text:** tokenize, remove stop words, stem/lemmatize (recall Lectures 31-33)
- 3 **Extract features:** convert text to numeric form using BoW, TF, or TF-IDF (recall Lectures 34-35)
- 4 **Train a classifier:** e.g., KNN, Decision Tree, or a neural network (recall Units 2-3) to predict sentiment



Word Frequency Analysis

What is Word Frequency Analysis?

- Word frequency analysis counts how often each word appears across a piece of text or a collection of documents
- Directly related to Lecture 34's Bag of Words and Term Frequency concepts, but used here as an analysis tool rather than just a modelling input
- Reveals the most commonly discussed topics or terms within a body of text



Worked Example: Word Frequency in Customer Reviews

- Collect and preprocess a set of reviews (tokenize, remove stop words - recall Lectures 31-32)
- Count remaining word occurrences across all reviews
- Visualise using a bar chart (recall Lecture 6): most frequent words might be "great", "fast", "quality", "price"

Word Frequency Analysis: Applications

- Identifying trending topics on social media (recall Lecture 2's social media application)
- Summarising the main themes in a large set of documents or customer feedback
- Complementing sentiment analysis by showing WHICH words drive positive or negative sentiment

analysis
social
topics
trending
feedback
summary
them

Bringing it Together

Sentiment Analysis vs Word Frequency Analysis

- Both rely on the same preprocessing and text representation foundations from Lectures 31-35

Task	Type of Analysis	Output
Sentiment Analysis	Supervised Learning / Predictive	Positive / Negative / Neutral
Word Frequency Analysis	Exploratory / Descriptive	Word counts / Rankings

- Sentiment analysis: classifies text into a sentiment category - a supervised learning task
- Word frequency analysis: counts and ranks word occurrences - a descriptive analysis, not necessarily a prediction task

Quick Check: Sentiment and Word Frequency Analysis

Review Questions

- **Q:** Is sentiment analysis a supervised or unsupervised learning task? Why?
- **Q:** What preprocessing steps would you apply before counting word frequencies?
- **Q:** How might word frequency analysis complement sentiment analysis?

Summary

Key Takeaways

- Sentiment analysis is a supervised classification task predicting whether text is positive, negative, or neutral
- Word frequency analysis counts and ranks word occurrences to reveal common themes in text
- Both applications integrate preprocessing and text representation techniques from earlier in Unit 4

Next Lecture Preview

- Lecture 37 completes Unit 4 with keyword extraction from text using Python libraries, closing out the NLP unit before Unit 5's emerging AI trends.

Next up:
Keyword Extraction



Any Questions?

Lecture 4.9: Simple NLP Applications - Keyword Extraction

DI05000051 - AI & ML

Milav Dabgar

Lecturer, GP Palanpur

July 22, 2026

Recap: Lecture 36

- Lecture 36 covered sentiment analysis and word frequency analysis, both built on Unit 4's preprocessing and text representation foundations.

What is Keyword Extraction?

What is Keyword Extraction?

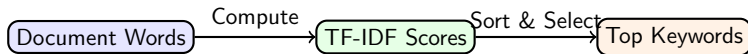
- Keyword extraction automatically identifies the most important or representative words/phrases in a piece of text
- Goal: summarise a document's main topics without reading the whole thing
- Recall Lecture 35's TF-IDF preview - keyword extraction is one of TF-IDF's most natural applications



Using TF-IDF for Keyword Extraction

Using TF-IDF for Keyword Extraction

- Recall Lecture 35: TF-IDF gives high scores to words that are frequent in a document but rare across the whole collection
- To extract keywords: compute TF-IDF scores for all words in a document, then pick the words with the highest scores
- These high-scoring words are, by definition, the ones that best distinguish this document from others



Worked Example: Extracting Keywords from a Review

- Recall Lecture 35's review example: "waterproof" had a high TF-IDF score, while "product" had a low one
- Sorting all words in the review by TF-IDF score and taking the top 3-5 gives a good keyword summary
- Example output keywords for a review: ["waterproof", "battery", "lightweight"] - immediately informative

Word	TF-IDF Score
waterproof	0.85
battery	0.72
lightweight	0.64
camera	0.45
product	0.05

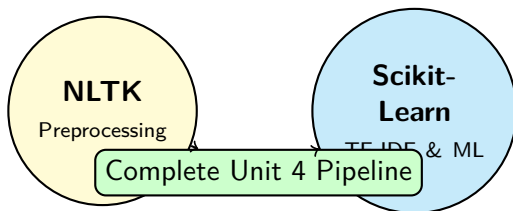
Other Approaches to Keyword Extraction

- Simple frequency-based extraction: just picking the most frequent meaningful words (recall Lecture 36's word frequency analysis)
- POS-based filtering: keeping mainly nouns and adjectives (recall Lecture 33's POS tagging), since keywords are often nouns/adjectives
- TF-IDF-based extraction (today's focus) is usually more effective since it accounts for distinctiveness, not just raw frequency

Python Libraries & Applications

Python Libraries for These NLP Applications

- NLTK (recall Lecture 33): tokenization, stop words, stemming/lemmatization, POS tagging
- Scikit-learn (recall Lecture 4): includes TF-IDF vectorisation tools and classifiers for sentiment analysis
- Together, these libraries implement the ENTIRE Unit 4 pipeline: preprocessing through to simple applications



Applications of Keyword Extraction

- Automatic tagging/categorising of articles, documents, or support tickets
- Summarising customer feedback themes at scale (recall Lecture 2's applications across industries)
- Improving search engines by indexing documents by their extracted keywords

Article Tagging

Feedback
Summary

Search
Indexing

Unit 4 Recap & Summary

Unit 4 Recap: The Complete NLP Journey

- 4.1: What is NLP, and its major applications
- 4.2: The NLP processing pipeline (collection, preprocessing, feature extraction, model building)
- 4.3: Text preprocessing - tokenization, stop words, stemming/lemmatization, POS tagging
- 4.4: Text representation - Bag of Words, Term Frequency, TF-IDF
- 4.5: Simple applications - sentiment analysis, word frequency analysis, keyword extraction

Quick Check: Keyword Extraction and Unit 4 Recap

Questions

- 1 How does TF-IDF help identify keywords in a document?
- 2 Name one Python library used for text preprocessing and one used for TF-IDF/classification.
- 3 List Unit 4's five topics in order.

Key Takeaways

- Keyword extraction identifies a document's most representative words, often using TF-IDF scores
- NLTK and Scikit-learn together implement the complete Unit 4 pipeline, from preprocessing to simple applications
- Unit 4 progressed from NLP's definition and applications, through the pipeline, preprocessing, and text representation, to hands-on applications

Lecture 38: Emerging Trends in Artificial Intelligence

- Unit 5 begins with an overview of modern AI developments like Generative AI and autonomous systems

Questions?

Lecture 5.1: Emerging Trends in Artificial Intelligence

DI05000051 - AI & ML

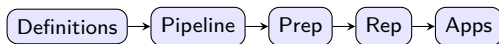
Milav Dabgar

Lecturer, GP Palanpur

July 22, 2026

Recap: Lecture 37

- Lecture 37 closed Unit 4 with keyword extraction and a recap of NLP's full journey: definitions, pipeline, preprocessing, representation, and applications.



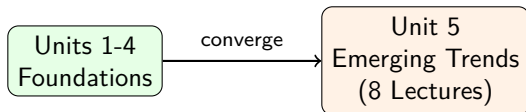
Today's Agenda

- Part 1: Generative AI, AI-powered assistants, autonomous systems
- Part 2: AI in healthcare, finance, and education revisited
- Part 3: Short real-world case studies

Introduction to Unit 5

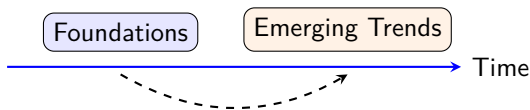
Welcome to Unit 5: Emerging Trends in AI

- Recall Lecture 1: this course began by defining AI and ML broadly - Unit 5 (8 lectures) closes the loop with today's cutting edge
- Units 1-4 built the foundations: AI basics, classical ML, neural networks, and NLP
- Unit 5 covers Generative AI, LLMs, prompt engineering, and AI ethics - the most current and fast-moving parts of the field



What Makes These Trends 'Emerging'?

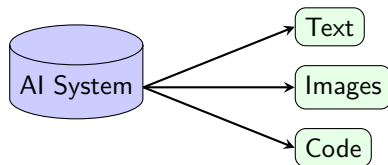
- They represent recent, rapidly evolving developments built on neural networks (recall Unit 3) and NLP (recall Unit 4)
- Many became widely accessible to the public only in the last few years
- Understanding them requires the foundation already built in this course - they are not a separate, unrelated topic



Emerging Trends in AI

Emerging Trend: Generative AI

- AI systems that CREATE new content (text, images, code) rather than only classifying or predicting
- A major shift from Unit 2/3's classification/regression outputs to actually generating original content
- Covered in full detail next lecture (Lecture 39)



Emerging Trend: AI-Powered Assistants

- Modern virtual assistants and AI chatbots capable of much richer, more natural conversations than earlier chatbots (recall Lecture 1/29)
- Built on advances in neural networks (Unit 3) and NLP (Unit 4), particularly Large Language Models (covered in Lecture 40-41)
- Can answer questions, write text, summarise documents, and hold multi-turn conversations

User: Can you summarise this doc?

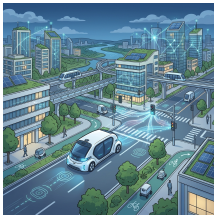
Assistant: Here is the summary...

User: Explain the second point.

Assistant: Sure, the second point...

Emerging Trend: Autonomous Systems

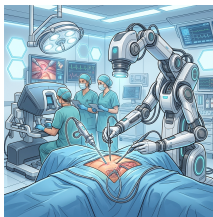
- Systems that can operate and make decisions with minimal human intervention
- Examples: self-driving vehicles, autonomous drones, automated manufacturing robots (recall Lecture 2's manufacturing application)
- Rely heavily on neural networks (Unit 3) for perception (e.g., image recognition, Lecture 20) and decision-making



Revisiting Key Applications & Case Studies

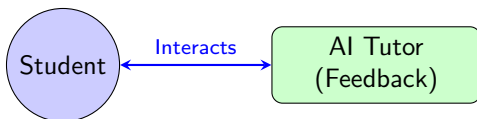
AI in Healthcare, Finance, and Education: What's New?

- Recall Lecture 2's applications - emerging trends push these further with more powerful, general-purpose AI models
- Healthcare: AI models now assist in more complex diagnostic reasoning, not just pattern-matching in images
- Finance and Education: increasingly personalised, conversational AI tools supporting decisions and learning



Short Case Study: AI Writing Assistants in Education

- Modern AI writing/tutoring assistants can explain concepts, generate practice questions, and give personalised feedback
- Recall Lecture 2's 'AI in education' application - this is a direct evolution of personalised learning platforms
- Raises new questions about academic integrity and responsible use - previewed for Lecture 44-45's ethics discussion



Summary & Next Lecture

Quick Check: Emerging AI Trends

Discussion Questions

- **Q:** Name three emerging trends covered today.
- **Q:** How do autonomous systems relate to Unit 3's neural network applications?
- **Q:** Which Unit 2 concept (classification/regression) contrasts most with Generative AI's approach?

Key Takeaways

- Emerging AI trends include Generative AI, advanced AI-powered assistants, and autonomous systems
- These trends build directly on neural networks (Unit 3) and NLP (Unit 4) foundations already covered
- Healthcare, finance, and education (recall Lecture 2) are being reshaped further by these more powerful AI capabilities

Lecture 39: Introduction to Generative AI

The concept, its difference from traditional AI, and its major application types (text, image, and code generation).

Questions?

Lecture 5.2: Introduction to Generative AI

DI05000051 - AI & ML

Milav Dabgar

Lecturer, GP Palanpur

July 22, 2026

Recap: Lecture 38

- Lecture 38 surveyed emerging AI trends broadly, including a first introduction to Generative AI, AI assistants, and autonomous systems.

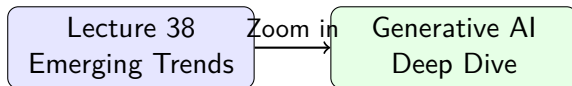
Today's Agenda

- ① Part 1: The concept of Generative AI
- ② Part 2: Traditional AI vs Generative AI
- ③ Part 3: Text, image, and code generation examples
- ④ Part 4: Industry applications

Part 1: The Concept of Generative AI

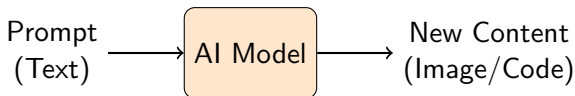
Recall: Generative AI Was Previewed Last Lecture

- Lecture 38 introduced Generative AI as an emerging trend that CREATES content rather than just classifying/predicting
- Today we explore this concept in full depth, as required by topic 5.2
- This directly extends everything learned about neural networks (Unit 3) to a new, generative purpose



What is Generative AI?

- Generative AI refers to AI systems that generate new, original content - text, images, audio, code, and more
- Rather than only recognising existing patterns (classification) or predicting existing values (regression), it produces NEW output
- Built on advanced neural network architectures (recall Unit 3) trained on massive datasets



Part 2: Traditional AI vs Generative AI

Traditional AI vs Generative AI

Traditional AI (predict/classify)	Generative AI (create/generate)
Given input, predict a label or a number (fixed category)	Given input, produce entirely NEW content
Example: Label an email as spam/not-spam	Example: WRITE a new email from scratch
Focus: Analysis and prediction	Focus: Creation and synthesis

Part 3: Text, Image, and Code Generation Examples

Example: Text Generation

- Generative AI systems can write essays, stories, emails, summaries, and answers to questions
- Recall Lecture 29-37's NLP techniques - text generation is the natural, more advanced extension of text-based NLP
- Powers modern AI chat assistants (recall Lecture 38, detailed further in Lecture 40-41's LLM discussion)



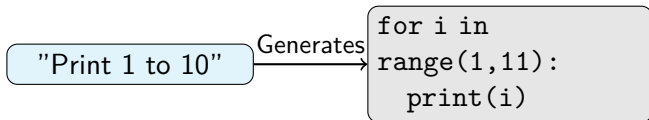
Example: Image Generation

- Generative AI systems can create entirely new images from text descriptions or other images
- Example: generating a picture of "a futuristic city at sunset" purely from a text description
- Builds on neural network image-processing concepts (recall Lecture 20's image recognition, now applied in reverse - to CREATE images)



Example: Code Generation

- Generative AI systems can write, complete, or explain programming code from natural language descriptions
- Example: describing a task in plain English (recall Lecture 3's Python introduction) and receiving working code as output
- Useful for accelerating software development and helping beginners learn programming



Part 4: Industry Applications

Applications of Generative AI in Industry

- Content creation: marketing copy, articles, social media posts (recall Lecture 2's social media application)
- Software development: accelerating coding tasks and prototyping
- Design: generating initial concepts for graphics, products, or architecture
- Education: generating practice questions, explanations, and personalised study material (recall Lecture 38's case study)

Marketing

Software

Design

Education

Questions

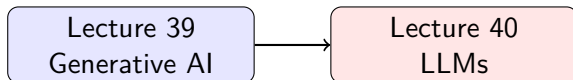
- Q: What is the key difference between traditional AI and Generative AI?
- Q: Name the three types of generative AI examples covered today
- Q: How does image generation relate to Lecture 20's image recognition, but in reverse?

Key Takeaways

- Generative AI creates new content (text, images, code) rather than only classifying or predicting existing categories/values
- It contrasts with traditional AI's classification/regression approach (Units 2-3) by generating original output
- Generative AI is already applied across content creation, software development, design, and education

Next Lecture Preview

- Lecture 40: Large Language Models (LLMs) Part A - what LLMs are and examples of LLM-based systems



Questions?

Lecture 5.3: Large Language Models (LLMs) - Introduction

DI05000051 - AI & ML

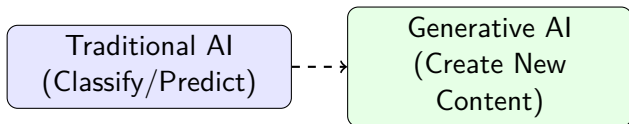
Milav Dabgar

Lecturer, GP Palanpur

July 22, 2026

Recap: Lecture 39

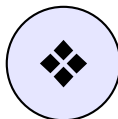
- Lecture 39 introduced Generative AI: creating new content, contrasted with traditional AI's classification/regression, across text, image, and code examples.



What is a Large Language Model?

Zooming into Text Generation: LLMs

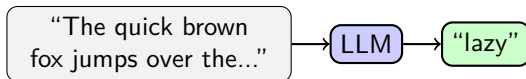
- Recall Lecture 39's text generation example - Large Language Models (LLMs) are the specific technology behind it
- LLMs are the engine behind most modern AI-powered assistants (recall Lecture 38)
- Today: what exactly is an LLM, and what makes it 'large'?



Large Language Model (LLM)

What is a Large Language Model (LLM)?

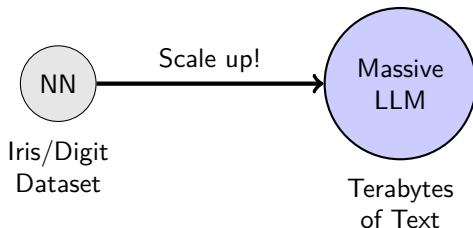
- An LLM is a neural network (recall Unit 3) trained on enormous amounts of text data to understand and generate human language
- It predicts the most likely next word (or sequence of words) given some input text
- By repeating this next-word prediction, it can generate entire coherent sentences, paragraphs, or answers



What Makes an LLM 'Large'?

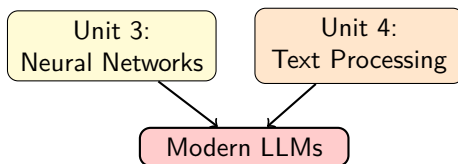
What Makes an LLM 'Large'?

- Trained on massive amounts of text data - far more than any dataset discussed earlier in this course (recall Lecture 28's Iris/Digit datasets, by contrast, are tiny)
- Built from neural networks with a huge number of layers and neurons (recall Unit 3's architecture concepts, scaled up enormously)
- This scale allows LLMs to capture very broad, nuanced patterns in language



How LLMs Build on Earlier Course Concepts

- Neurons, layers, weights, and activation functions (Units 3) are still the underlying building blocks, just at a vastly larger scale
- Text preprocessing and representation ideas (Unit 4) relate to how LLMs process input text, though LLMs use more advanced internal representations
- LLMs are best understood as a natural, large-scale evolution of concepts already covered in this course



Examples of LLM-Based Systems

Examples of LLM-Based Systems

- AI chat assistants that can answer questions, write text, and hold conversations (recall Lecture 38)
- AI-powered writing and coding assistants (recall Lecture 39's text/code generation examples)
- Search and question-answering systems that generate direct natural-language answers

Chat Assistants

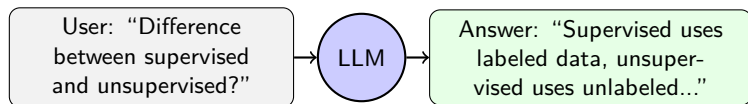
Writing Assistants

Coding Assistants

Q&A Systems

Worked Example: An LLM Answering a Question

- User asks: “What is the difference between supervised and unsupervised learning?”
- The LLM generates a fluent, relevant answer by predicting likely words/sentences based on patterns learned from its training data
- Notice: the LLM could correctly answer this because Lecture 12’s exact distinction appears widely in its training text



Quick Check: LLM Basics

Questions

- 1 In simple terms, what does an LLM do when generating text?
- 2 What makes a Large Language Model 'large'?
- 3 Name one type of LLM-based system.

Key Takeaways

- A Large Language Model (LLM) is a neural network trained on massive text data to predict and generate language, word by word
- LLMs are 'large' due to their enormous training data and very large neural network architectures
- LLM-based systems include AI chat assistants, writing/coding assistants, and natural-language question-answering systems

Lecture 41: Large Language Models (LLMs) Part B

Capabilities of modern AI chat systems and their applications in education, business, and software development.

Questions?

Lecture 5.4: Large Language Models (LLMs) - Capabilities and Applications

DI05000051 - AI & ML

Milav Dabgar

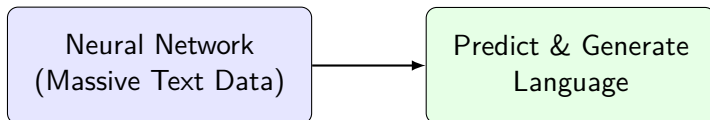
Lecturer, GP Palanpur

July 22, 2026

Recap & Agenda

Recap: Lecture 40

- Lecture 40 introduced LLMs as neural networks trained on massive text data to predict and generate language, and gave examples of LLM-based systems.



Today's Agenda

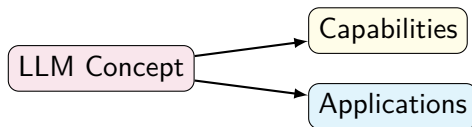
- Part 1: Core capabilities of AI chat systems
- Part 2: Applications in education
- Part 3: Applications in business and software development

- 1 Core Capabilities
- 2 Education Apps
- 3 Business & Dev Apps

Core Capabilities of AI Chat Systems

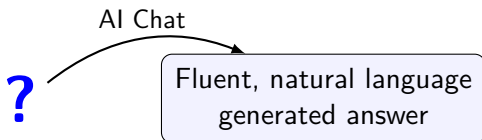
From What LLMs Are to What They Can Do

- Recall Lecture 40: LLMs generate text by predicting likely words based on massive training data
- Today we explore the practical **CAPABILITIES** this enables in modern AI chat systems
- Then we look at concrete applications across education, business, and software development



Capability: Answering Questions

- Modern AI chat systems can answer factual, conceptual, and even open-ended questions in natural language
- Recall Lecture 40's worked example: explaining the difference between supervised and unsupervised learning
- Answers are generated fluently, but their accuracy still depends on the quality and coverage of training data



Capability: Writing Code

- AI chat systems can write, explain, and debug programming code from natural language descriptions
- Recall Lecture 39's code generation example and Lecture 3's introduction to Python
- Useful for learning, prototyping, and speeding up software development tasks

"Write a Python function..."



```
def my_func():
```

Capability: Generating Text

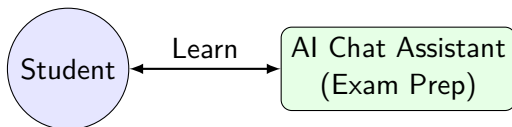
- AI chat systems can write essays, summaries, emails, stories, and explanations on demand
- Recall Lecture 39's text generation example - this is that same capability, now embedded in an interactive chat system
- Can adapt tone, length, and style based on how the request (prompt) is phrased - previewed for Lecture 42's prompt engineering



Applications of LLMs

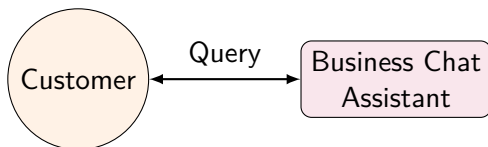
Application: Education

- Personalised tutoring: explaining concepts at a student's own level and pace (recall Lecture 2's education application)
- Generating practice questions, summaries, and study guides
- Recall Lecture 38's case study on AI writing assistants in education - this is exactly that capability in action



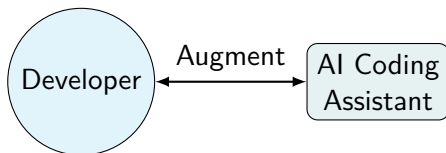
Application: Business

- Customer support chatbots that resolve queries more naturally and effectively than older rule-based bots (recall Lecture 1's original chatbot example)
- Drafting business documents, reports, and marketing content (recall Lecture 39's text generation)
- Summarising large volumes of business documents or customer feedback quickly



Application: Software Development

- Code completion and generation, reducing repetitive coding work (recall today's 'writing code' capability)
- Explaining unfamiliar code and helping debug errors
- Assisting with documentation and test-case generation



Knowledge Check & Summary

Quick Check: LLM Capabilities and Applications

- Q: Name the three core capabilities of modern AI chat systems covered today
- Q: How does LLM-based tutoring relate to Lecture 2's education application?
- Q: What earlier course lecture's chatbot example is now dramatically more capable thanks to LLMs?

Discuss your answers!

Key Takeaways

- Modern AI chat systems built on LLMs can answer questions, write code, and generate text fluently
- LLMs are transforming education (tutoring), business (customer support, content), and software development (code assistance)
- These capabilities depend heavily on HOW a request is phrased - the subject of prompt engineering, covered next

Next Lecture Preview

- **Lecture 42:** Prompt Engineering Part A - what a prompt is, why prompt design matters, and basic prompt techniques

See you next time!

Questions?

Lecture 5.5: Prompt Engineering - Fundamentals and Basic Techniques

DI05000051 - AI & ML

Milav Dabgar

Lecturer, GP Palanpur

July 22, 2026

Recap: Lecture 41

- Lecture 41 covered LLM capabilities and applications across education, business, and software development.

Today's Agenda

- Part 1: What is a prompt?
- Part 2: Why prompt design matters
- Part 3: Clear instructions, role prompting, step-by-step prompting

1. What is a prompt?

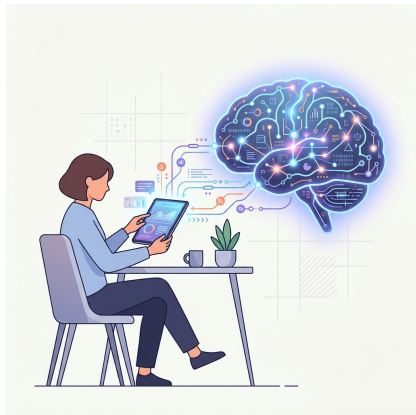
2. Why prompt design matters

3. Basic Techniques

What is a Prompt?

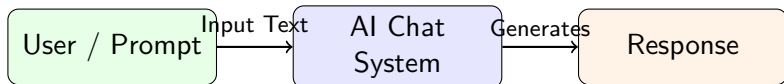
Recall: Output Depends on How You Ask

- Lecture 41 ended noting LLM output quality depends heavily on how a request is phrased
- Today's topic, Prompt Engineering, is the practical skill of phrasing requests effectively
- This is a genuinely useful, hands-on skill for using any of Lecture 40-41's AI chat systems well



What is a Prompt?

- A prompt is the input text (instruction, question, or request) given to an LLM or AI chat system
- Recall Lecture 40: the LLM generates its response by predicting likely words based on this prompt and its training
- The prompt is the **ONLY** way a user communicates their intent to the AI system



Why Prompt Design Matters

Why Does Prompt Design Matter?

- A vague or unclear prompt often produces a vague, generic, or unhelpful response
- A well-designed prompt clearly conveys the task, context, and desired output format
- Recall Lecture 10's well-posed ML problem idea (Task, Experience, Performance) - a good prompt similarly needs to clearly specify the TASK

Vague Prompt

"Tell me about AI" →
Generic result

Clear Prompt

"Explain AI vs ML in 3 points" → Focused result

Basic Prompt Techniques

Basic Technique 1: Clear Instructions

- State exactly what you want, including relevant context, constraints, and desired format
- Recall Lecture 1's AI vs ML distinction - this clear-instruction example directly reuses that exact concept

Vague

"Tell me about AI."

Likely Output:

Long, unstructured essay about everything related to AI.

Clear

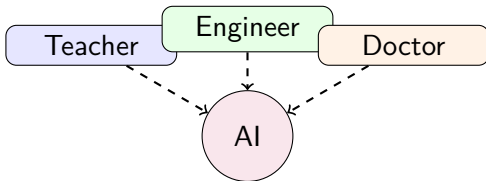
"Explain the difference between AI and ML in 3 simple bullet points for a beginner."

Likely Output:

3 concise, easy-to-read bullet points.

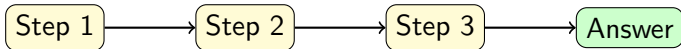
Basic Technique 2: Role Prompting

- Asking the AI to respond AS IF it were a specific role or persona
- Example: "Act as an experienced Python teacher and explain loops to a beginner" (recall Lecture 3's Python introduction)
- Role prompting can shape the tone, depth, and style of the response



Basic Technique 3: Step-by-Step Prompting

- Asking the AI to work through a problem step by step, rather than jumping straight to a final answer
- Especially useful for reasoning-heavy or multi-step tasks (e.g., solving a calculation, explaining a process)
- This technique will reappear as 'Chain-of-thought prompting', one of the specific prompt types covered next lecture



Worked Example: Improving a Prompt

Poor Prompt

"Explain neural networks."

Better Prompt

"Act as a teacher (*Role*). **Explain, step by step** (*Step-by-step*), what a neural network's input, hidden, and output layers do (recall Lecture 21), **in simple language for a diploma student** (*Clear Instruction*)."

- The better prompt combines clear instructions + role prompting + step-by-step prompting all at once

Quick Check: Basic Prompt Techniques

Questions

- 1 What is a prompt?
- 2 Rewrite this vague prompt using clear instructions:
"Tell me about ML."
- 3 What is role prompting, and give one example.

Key Takeaways

- A prompt is the input text given to an LLM/AI chat system, and it is the sole channel for conveying user intent
- Well-designed prompts produce far better results than vague ones - prompt design is a genuinely practical skill
- Basic techniques include clear instructions, role prompting, and step-by-step prompting

- **Lecture 43:** Prompt Engineering Part B - the different types of prompts, from zero-shot to chain-of-thought and beyond

→ Types of Prompts ←

Questions?

Lecture 5.6: Prompt Engineering - Types of Prompts

DI05000051 - AI & ML

Milav Dabgar

Lecturer, GP Palanpur

July 22, 2026

Recap: Lecture 42

- Lecture 42 covered prompt basics and three techniques: clear instructions, role prompting, and step-by-step prompting.

Clear Instructions

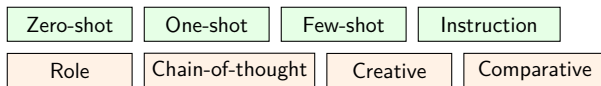
Role Prompting

Step-by-step

Zero-shot, One-shot, Few-shot Prompting

From Techniques to Formal Prompt Types

- Recall Lecture 42's three basic techniques - today we name and formalise 8 specific, recognised prompt TYPES
- Many of these types build directly on Lecture 42's techniques, just with more precise names and structures
- Knowing these types helps you choose the right approach for a given task deliberately



Zero-Shot Prompting

- Asking the AI to perform a task with NO examples given - just the instruction itself
- Works well when the task is common enough that the LLM has learned it broadly from its training data (recall Lecture 40)

Example

"Classify this review as positive or negative: 'The battery life is excellent.'"

One-Shot Prompting

- Providing exactly ONE example alongside the instruction, to show the AI the expected format/style
- Useful when the desired output format needs a little more clarification than zero-shot alone provides

Example

Positive: 'Great battery life.'

Now classify: 'The screen cracked on day one.'

Few-Shot Prompting

- Providing SEVERAL examples (typically 2-5) alongside the instruction
- More examples generally help the AI understand the exact pattern or format expected, at the cost of a longer prompt

Example

Showing 3 labelled reviews (positive/negative) before asking the AI to classify a new one.

Zero-Shot vs One-Shot vs Few-Shot

Type	# Examples	Best Used When
Zero-shot	0	Relies entirely on the AI's general training knowledge
One-shot	1	Clarifies format/style with minimal extra prompt length
Few-shot	2-5	Gives the clearest guidance, but requires a longer prompt

Instruction, Role, and Chain-of-Thought

Instruction-Based and Role-Based Prompting

- **Instruction-based:** giving a direct, explicit command (recall Lecture 42's 'clear instructions' technique)
- **Role-based:** asking the AI to respond as a specific persona (recall Lecture 42's 'role prompting' technique, e.g., 'Act as a Python teacher')
- Both are formal names for techniques already introduced in Lecture 42

Command

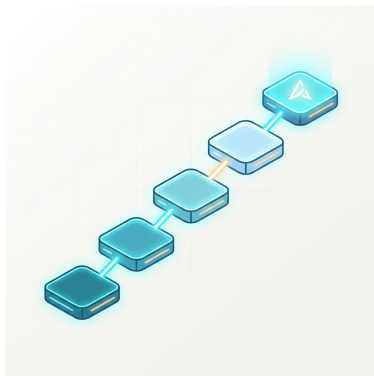
"Summarize this..."

Persona

"Act as a teacher..."

Step-by-Step (Chain-of-Thought) Prompting

- The formal name for Lecture 42's 'step-by-step prompting' technique
- Asks the AI to reason through a problem in explicit steps before giving a final answer
- Especially valuable for multi-step reasoning tasks, such as working through a calculation



Creative and Comparative Prompting

Creative Prompting

- Encourages the AI to generate original, imaginative content rather than a strictly factual or structured answer
- Useful for brainstorming, content creation, and exploring creative possibilities

Example

"Write a short story about a robot learning to paint" (recall Lecture 39's text generation example)

Comparative Prompting

- Asks the AI to compare two or more things, options, or approaches
- Useful for decision-making, study/revision, and understanding trade-offs between options

Example

"Compare KNN and Decision Tree for a classification task"
(recall Lectures 13-14)



All 8 Prompt Types at a Glance

Category	Prompt Types
By Examples	Zero-shot, One-shot, Few-shot
By Framing	Instruction-based, Role-based
By Structure	Step-by-step (Chain-of-thought)
By Output	Creative, Comparative

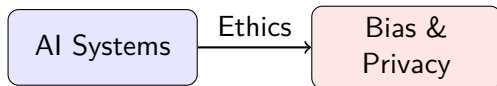
Quick Check: Types of Prompts

- 1 What is the difference between zero-shot and few-shot prompting?
- 2 Which prompt type would you use to ask an AI to compare two ML algorithms?
- 3 Which prompt type asks the AI to show its reasoning step by step?

Key Takeaways

- Zero-shot, one-shot, and few-shot prompting differ by how many examples are provided in the prompt
- Instruction-based, role-based, and chain-of-thought prompting formalise Lecture 42's basic techniques under precise names
- Creative prompting encourages imaginative output; comparative prompting requests structured comparisons

- **Lecture 44:** Ethical and Responsible AI Part A - bias in AI systems and data privacy concerns



Questions?

Thank You!

Lecture 5.7: Ethical and Responsible AI - Bias and Privacy

DI05000051 - AI & ML

Milav Dabgar

Lecturer, GP Palanpur

July 22, 2026

Recap: Lecture 43

- Lecture 43 completed prompt engineering's 8 formal prompt types, from zero-shot to comparative prompting.

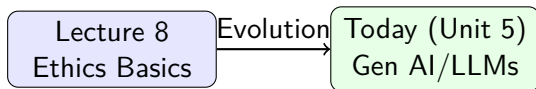
Today's Agenda

- ① Part 1: Bias in AI systems, revisited for modern AI
- ② Part 2: Data privacy concerns with Generative AI and LLMs
- ③ Part 3: Why these concerns matter more as AI grows more powerful

Bias in AI Systems Revisited

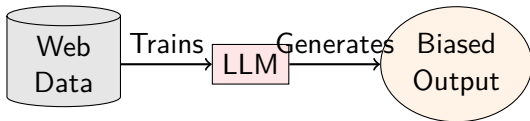
Returning to Ethics, Now with Full Course Context

- **Recall Lecture 8:** we first covered bias, privacy, and responsible AI use back in Unit 1
- **Since then,** we've learned about neural networks (Unit 3), NLP (Unit 4), and now Generative AI/LLMs (Unit 5)
- **Today revisits bias and privacy** with this much deeper technical understanding of HOW modern AI actually works



Bias in AI Systems: A Deeper Look

- **Recall Lecture 8:** bias arises mainly from unrepresentative or unfair training data
- **Recall Lecture 40:** LLMs are trained on massive amounts of text data scraped from many sources, including biased or unbalanced content
- Because LLMs learn patterns from this training data, they can reproduce or amplify stereotypes and imbalances present in it



Example: Bias in Generative AI Output

- An **image generation system** (recall Lecture 39) might, unless carefully designed, generate stereotypical depictions for certain professions or roles
- A **text-generating LLM** might produce responses that subtly reflect biases present in its training text
- These issues are actively studied and mitigated by AI developers, but not perfectly solved

Review & Mitigation

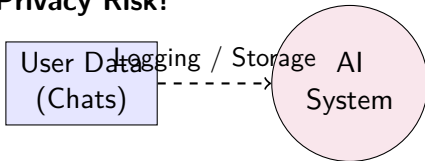
Outputs must be monitored and algorithms fine-tuned (e.g., RLHF) to align models and reduce biased generations.

Data Privacy Concerns with Generative AI

Data Privacy Concerns: A Deeper Look

- **Recall Lecture 8:** AI's reliance on personal data creates privacy risks
- **LLMs (Lecture 40)** are trained on vast datasets that may include publicly available text - raising questions about consent and data ownership
- When users chat with **AI assistants (Lecture 41)**, their conversations may be stored or used to further improve the system

Privacy Risk!



Privacy Considerations When Using AI Chat Systems

- Avoid sharing sensitive personal or confidential information in prompts to AI chat systems (recall Lecture 42's prompt engineering)
- Be aware that conversations may be logged or reviewed for system improvement, depending on the platform
- Organisations (recall Lecture 2's finance/healthcare applications) must be especially careful with sensitive data in AI tools

Actionable Advice

Always assume that what you type into a public AI chat system could potentially be reviewed or used in future training data.

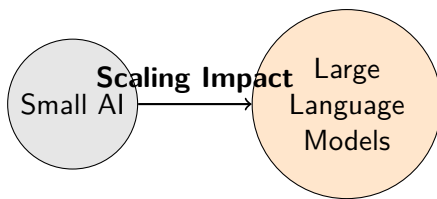
Why Scale Matters for Ethics

Bias and Privacy: Then vs Now

Lecture 8 Context	Generative AI / LLM Context
Classification/regression models (Units 2-3)	Generative AI/LLMs (Unit 5)
Trained on structured data	Trained on massive, less controlled text data
Core Lesson Unchanged	
AI reflects the data and processes used to build it	
Now happening at a much larger scale	

Why These Concerns Matter More Now

- **Recall Lecture 38-41:** modern AI systems (LLMs, chat assistants) are far more widely used and influential than earlier, narrower AI models
- Wider reach means bias or privacy failures can affect far more people, far more visibly
- This motivates continued attention to responsible AI use, discussed further in Lecture 45



Quick Check: Bias and Privacy in Modern AI

- **Q:** How can an LLM's training data (recall Lecture 40) lead to biased output?
- **Q:** What privacy precaution should you take when using an AI chat system?
- **Q:** How do today's bias/privacy concerns relate to Lecture 8's original discussion?

Key Takeaways

- **Bias Scaling:** LLMs can reproduce or amplify biases present in their massive training data, extending Lecture 8's foundational bias discussion
- **Privacy Concerns:** Generative AI and AI chat systems raise new privacy questions around training data sourcing and conversation logging
- **Magnified Impact:** These concerns grow more important as AI systems become more powerful and widely deployed

- **Lecture 45: Ethical and Responsible AI Part B**
 - Misuse of AI-generated content
 - The importance of ethical AI development
 - A full course recap

Up Next: The Final Lecture!

Questions?

Lecture 5.8: Ethical and Responsible AI - Misuse and Course Recap

DI05000051 - AI & ML

Milav Dabgar

Lecturer, GP Palanpur

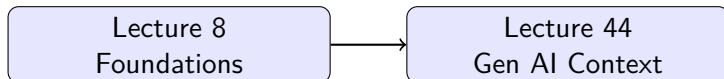
July 22, 2026

Recap of Previous Lecture

- Lecture 44 covered bias in AI systems and data privacy concerns in the context of modern Generative AI and LLMs, extending Lecture 8's foundational ethics discussion.

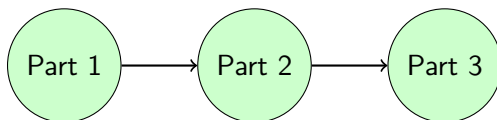
Recap: Bias and Privacy in Modern AI

- Lecture 44 covered how LLMs can reproduce training-data bias, and new privacy concerns around AI chat systems
- Both extended Lecture 8's foundational bias/privacy discussion to today's Generative AI/LLM era



Today's Agenda: Final Lecture of the Course

- Part 1: Misuse of AI-Generated Content
- Part 2: Importance of Ethical AI Development
- Part 3: Full Course Recap (all 5 units)



Misuse of AI-Generated Content

Part 1: What Counts as Misuse of AI-Generated Content?

- Recall Lecture 39's Generative AI: text, image, and code generation can be misused as well as used well
- Misuse includes: creating misleading or fake content (deepfakes, fabricated text) presented as real
- Also includes: plagiarism (submitting AI-generated work as entirely one's own) and generating harmful or deceptive material

X Warning: AI-Generated Misinformation

Example: Misinformation via Generative AI

- Recall Lecture 39's image generation example - the same technology that creates a nice image of a city can create a convincing but fake photo of a real event
- Recall Lecture 40-41's text generation - the same technology that writes a helpful essay can generate convincing false information at scale
- The concern is not the technology itself, but how it can be intentionally misused

BREAKING NEWS

Fabricated Headline Generated by AI

✗ Flagged as Misinformation

Example: Academic and Professional Misuse

- Submitting AI-generated writing or code as one's own original work without disclosure
- Recall Lecture 38's education case study - this is exactly the risk hinted at there
- Educational and professional institutions are actively developing policies on acceptable AI use

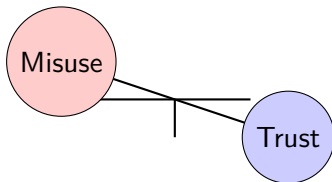
AI-Assisted
Properly Disclosed

VS

Undisclosed
Academic Misconduct

Consequences of Misuse

- Erodes public trust in both AI systems and in genuine human-created content
- Can cause real harm: reputational damage, spread of false information, unfair academic/professional outcomes
- Makes it harder to tell reliable information from fabricated content across society



Ethical AI Development

Part 2: Importance of Ethical AI Development

- Recall Lecture 8's responsible AI principles: diverse data, transparency, human oversight, ongoing monitoring
- For Generative AI/LLMs specifically: adding safeguards against generating harmful, false, or misleading content
- Ethical AI development is an ongoing responsibility, not a one-time checklist

Responsible AI Checklist

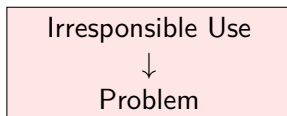
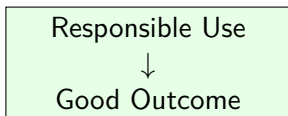
- ✓ Diverse Data
- ✓ Transparency
- ✓ Human Oversight
- ✓ Content Safe-guards

Principles for Ethical AI Development Going Forward

- Transparency: clearly disclosing when content is AI-generated
- Accountability: developers and organisations taking responsibility for how their AI systems are used
- Human oversight: keeping humans involved in reviewing high-stakes or sensitive AI-generated content
- Continuous evaluation: regularly checking AI systems for bias, misuse potential, and unintended harms

Case Example: Responsible Use of an AI Assistant

- A student uses an AI assistant (recall Lecture 41) to help brainstorm and draft an assignment
- **Responsible practice:** reviewing, verifying, and taking ownership of the final content; disclosing AI assistance where required
- **Irresponsible practice:** submitting unreviewed AI output directly as one's own original work



Full Course Recap

Unit 5 Recap: The Big Picture

- 5.1 Emerging trends in AI: Generative AI, AI assistants, autonomous systems
- 5.2 Introduction to Generative AI: concept, comparison with traditional AI
- 5.3 Large Language Models: what they are, their capabilities and applications
- 5.4 Prompt Engineering: fundamentals, basic techniques, and 8 formal prompt types
- 5.5 Ethical and Responsible AI: bias, privacy, misuse, and ethical development

FULL COURSE RECAP: Unit 1

- Unit 1 - Introduction to Artificial Intelligence
- Defined AI, distinguished it from ML, and toured AI applications across healthcare, finance, education, manufacturing, and social media
- Introduced Python and its core AI/ML libraries (NumPy, Pandas, Matplotlib, Scikit-learn)
- Covered data handling and visualization, and closed with AI ethics: bias, privacy, responsible use

FULL COURSE RECAP: Unit 2

- Unit 2 - Machine Learning Fundamentals
- Defined ML, contrasted it with traditional programming, and introduced well-posed problems (Task, Experience, Performance)
- Covered Supervised vs Unsupervised Learning, then three algorithms: KNN, Decision Tree, and Linear Regression
- Covered training/testing datasets and evaluation via Accuracy, Confusion Matrix, Precision, and Recall

FULL COURSE RECAP: Unit 3

- Unit 3 - Neural Networks
- Covered the motivation for neural networks, the biological-to-artificial neuron analogy, and applications (image, speech, recommendation)
- Covered ANN structure (layers, weights, bias) and activation functions (Sigmoid, ReLU, Tanh)
- Covered single vs multi-layer architectures, and built simple classification models using the Iris and Digit recognition datasets

FULL COURSE RECAP: Unit 4

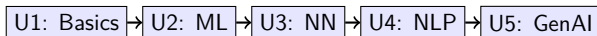
- Unit 4 - Natural Language Processing
- Covered NLP's definition, importance, and applications (chatbots, translation, sentiment analysis, voice assistants)
- Covered the NLP pipeline and text preprocessing (tokenization, stop words, stemming/lemmatization, POS tagging)
- Covered text representation (Bag of Words, TF, TF-IDF) and applications (sentiment analysis, word frequency, keyword extraction)

FULL COURSE RECAP: Unit 5

- Unit 5 - Emerging Trends in AI
- Covered emerging trends, Generative AI, and Large Language Models (LLMs)
- Covered prompt engineering: fundamentals, basic techniques, and 8 formal prompt types
- Covered ethical and responsible AI: bias, privacy, misuse, and the importance of ethical development

The Complete AI & ML Picture

- From foundations and tools (Unit 1) to classical ML algorithms (Unit 2) to neural networks (Unit 3) to language understanding (Unit 4) to today's cutting edge and its responsibilities (Unit 5)
- Each unit built directly on the concepts of the previous one - data handling, then modelling, then more powerful models, then a new data type (text), then today's most advanced systems
- Together, these 5 units form a complete foundation for understanding, applying, and responsibly using AI and Machine Learning



Course Key Takeaways

- AI is the broad goal of building intelligent systems; ML achieves this by learning patterns from data (Unit 1)
- Machine learning follows a disciplined pipeline: define the problem, choose a learning type and algorithm, train, test, and evaluate (Unit 2)
- Neural networks extend this pipeline with brain-inspired architectures capable of learning highly complex patterns (Unit 3)
- NLP applies these ideas to human language, and Generative AI/LLMs represent the field's most powerful and fast-moving frontier (Units 4-5)
- With this power comes responsibility: bias, privacy, and misuse must be actively managed throughout the AI lifecycle

Congratulations and Course Closing

- You have completed all 45 lectures of AI & ML (DI05000051)
- You now understand AI foundations, classical ML, neural networks, NLP, and today's emerging AI trends - along with the responsibility that comes with them
- This foundation prepares you for further study, hands-on projects, and careers across the rapidly growing AI/ML industry

✓ Course Successfully Completed!

Summary

Summary: Key Takeaways

- Misuse of AI-generated content includes misinformation, deepfakes, and undisclosed plagiarism, all of which erode trust and cause real harm
- Ethical AI development requires transparency, accountability, human oversight, and continuous evaluation, extending Lecture 8's responsible-AI principles
- The course progressed from AI/ML foundations (Unit 1) through classical ML (Unit 2), neural networks (Unit 3), and NLP (Unit 4) to emerging trends and ethics (Unit 5)
- Every concept across the 5 units connects into one coherent picture of how AI systems are built, trained, evaluated, applied to language, advanced through Generative AI, and used responsibly

Questions?