AI Foundations & Learning Paradigms

The Algorithmic View: The "Brain"

Focuses on the core mathematical model, such as a neural network or a decision tree. This "brain" is essentially a function mapping inputs to outputs, primarily concerned with optimization and loss minimisation. It defines the underlying mechanics of how AI learns.

The System View: The Integrated Agent

Emphasises a holistic, agent-oriented perspective. An AI System is an agent that perceives its environment via sensors and acts upon it through actuators to achieve specific goals. This view, popularised by Russell and Norvig, integrates the algorithm within an interactive architecture, like a self-driving car's comprehensive control systems.

Practical Implication: *The distinction is crucial. Researchers often advance the algorithmic frontier, while engineers build robust, safe, and integrated systems. Algorithms are not deployed directly; systems are.*

Core Components of an Intelligent System

Deconstructing the AI System view, we identify four key cognitive functions essential for any intelligent agent:

01

Perception

Interpreting raw sensory data (e.g., pixels, sound waves, text strings) to construct a structured internal representation. This involves tasks such as object identification in images or speech-to-text conversion.

02

Reasoning

Drawing logical inferences from perceived information. This function relies on logic, sophisticated knowledge representation, and often probabilistic judgment, enabling the system to deduce consequences or make predictions based on observed data.

03

Learning

The cornerstone of modern AI, this is the process of automatically improving performance on a given task through accumulated experience and data, as opposed to relying on explicit, rigid programming rules.

04

Decision-Making & Action

Selecting optimal actions from available alternatives to maximise a predefined utility or reward function. This is where the AI agent translates its understanding and learning into tangible interactions within its environment.

The Learning Paradigms

We begin with the two most fundamental paradigms, distinguished by the presence or absence of labelled data. These form the basis for most commercial AI applications.

Supervised Learning: Learning with a Guide

Core Concept: The algorithm is trained on a dataset comprising input-output pairs (X, y), with the objective of learning a mapping function `f: X -> y` that generalises effectively to unseen data.

- **Classification:** Predicting a discrete class label (e.g., "spam" or "not spam").
- **Regression:** Predicting a continuous numerical value (e.g., house prices).

Technical Mechanics: The model generates predictions, compares them against true labels using a **loss function** (e.g., Cross-Entropy, Mean Squared Error), and iteratively adjusts its internal parameters (e.g., neural network weights) via **optimization algorithms** (e.g., Gradient Descent) to minimise this loss.

Unsupervised Learning: Finding Hidden Structure

Core Concept: The algorithm receives only input data X, without any corresponding labels. The primary goal is to uncover inherent structure, patterns, or relationships within the data itself.

- **Clustering:** Grouping similar data points together (e.g., customer segmentation).
- **Dimensionality Reduction:** Projecting high-dimensional data into a lower-dimensional space while preserving essential structure (e.g., PCA for visualisation).
- **Anomaly Detection:** Identifying rare events that deviate significantly from the majority of the data (e.g., fraud detection).

The Learning Paradigms

Modern AI problems frequently necessitate more nuanced approaches that combine existing ideas or address sequential decision-making challenges.

Semi-Supervised Learning

Core Concept: *Leverages a small amount of labelled data and a large quantity of unlabelled data to refine decision boundaries. Ideal for scenarios where data labelling is costly but raw data is abundant (e.g., medical imaging).*

Transfer Learning

Core Concept: *Reuses a model developed for a source task as a starting point for a target task. This approach, which significantly reduces training time and data requirements, is the de facto standard for tasks with limited data.*

Reinforcement Learning

Core Concept: *An agent learns optimal behaviour through trial-and-error interactions with a dynamic environment, aiming to maximise cumulative reward over time. Applied in game playing, robotics, and autonomous driving.*

Datasets: Types and Preprocessing

The quality of your data is the most critical determinant of your model's performance. This section covers the raw material and its initial refinement.

Data Types: Understanding Your Medium

- ***Structured Data:*** *Highly organised, typically tabular, found in relational databases. Examples include sales records and sensor readings.*
- ***Unstructured Data:*** *The vast majority of global data, lacking a predefined model. Examples: text documents, images, and audio files.*
- ***Time-Series Data:*** *Data points indexed in time order, requiring specialised handling for temporal dependencies, such as stock prices or ECG signals.*

Data Preprocessing: The Crucial Art of Cleaning

- ***Handling Missing Values:*** *Strategies include deletion, imputation (filling with mean, median), or flagging missingness.*
- ***Normalization/Standardization:*** *Scaling numerical features to a common range or distribution, crucial for models sensitive to feature scales.*
- ***Handling Categorical Data:*** *Converting text categories into numerical formats (covered in subsequent slides).*
- ***Outlier Detection & Treatment:*** *Identifying and managing anomalous data points that could skew the model, using statistical methods or domain knowledge.*

Critical Point: *"Garbage In, Garbage Out" (GIGO). No algorithm can extract meaningful patterns from poorly prepared data. Preprocessing is a foundational step, not an afterthought.*

Datasets: The Train/Val/Test Split

This slide details the strategic partitioning of data, essential for building models that generalise effectively to new, unseen data—the ultimate goal of machine learning.

The Problem of Overfitting

A model that merely memorises the training data performs poorly on new data. It exhibits high **variance** and fails to **generalise**.

The Solution: Rigorous Data Partitioning

1

Training Set (~70%)

Data used to train the model's parameters (e.g., weights in a neural network), enabling the model to learn underlying patterns.

2

Validation Set (~15%)

Data used to tune hyperparameters (e.g., learning rate) and for model selection during development, serving as a crucial proxy for the test set.

3

Test Set (~15%)

Data used **only once** for the final, unbiased evaluation of the model's performance. It must remain completely untouched throughout the training and tuning processes.

Best Practices & Advanced Techniques

- **Stratified Sampling:** For classification, ensures each split maintains the same proportion of class labels as the full dataset.
- **Cross-Validation:** A robust technique involving multiple training/validation splits to obtain a more reliable estimate of model performance, particularly useful with smaller datasets.

The Sacred Rule: *The* ***test set is sacrosanct****. Any information leak from the test set into the training process invalidates results and creates a false sense of performance.*

Encoding: Classical Methods for Categorical Data

Before feeding data to an algorithm, we must convert categorical (non-numeric) variables into a numerical format. The choice of method has significant implications for model performance.

Label Encoding:

Mechanism: Assigns a unique integer to each category (e.g., "Red"=0, "Blue"=1, "Green"=2).

- **Advantage:** Simple to implement and avoids increasing dataset dimensionality.
- **Critical Disadvantage:** Introduces a false sense of **ordinality**. Models may interpret encoded values as having an order (0 < 1 < 2), which is meaningless for nominal data like colours. This can severely mislead models that assume higher numbers imply "more" of something (e.g., linear models).

When to Use: Only for tree-based models (e.g., Decision Trees, Random Forests) or genuinely ordinal data (e.g., "Low"=0, "Medium"=1, "High"=2).

One-Hot Encoding:

Mechanism: Creates new binary columns for each category. For a given data point, the column corresponding to its category is set to 1, and all others to 0 (e.g., "Red" = \[1, 0, 0\]).

- **Advantage:** Eliminates the problem of false ordinality, making it a safe choice for most algorithms.
- **Disadvantage: The Curse of Dimensionality.** High-cardinality categorical variables generate many new features, leading to computationally expensive training and sparse data.

Encoding: Advanced Learned Representations

For complex data like text or high-cardinality categories, classical methods are often insufficient. We require learned, dense representations that capture semantic meaning and relationships.

Embeddings: The Power of Learned Representations

Core Concept: A learned mapping from discrete objects (e.g., words, categories) to dense vectors of real numbers within a continuous, lower-dimensional space.

Key Property: Semantic Similarity. The geometric distance and direction between these vectors inherently capture semantic relationships. For instance, semantically similar items (e.g., "king" and "queen") will have closely related vector representations.

How are Embeddings Learned?

- **Task-Based Learning:** Embeddings are typically learned as an integral part of a larger model. For example, within a neural network designed for product recommendation, an embedding layer learns to represent each product ID as a vector that optimises the prediction of user preferences.
- **Pre-trained Embeddings:** Models are trained on vast, general-purpose datasets to generate universal embeddings, which can then be transferred and fine-tuned for specific tasks.
  - **Word2Vec/GloVe (for Text):** These algorithms learn word embeddings by predicting words from their contexts (or vice-versa) within extensive text corpora (e.g., Wikipedia). The resulting vectors exhibit remarkable algebraic properties, such as `king - man + woman ≈ queen`.
  - **Image/Graph Embeddings:** Similar techniques are employed to create vector representations for complex entities like images or nodes within a graph structure, capturing their inherent features and relationships.