AI in Society and Public Services

Session 2: The Evolution of Machine Learning Models

Mário Antunes

January 26, 2026

Universidade de Aveiro

Table of Contents i

AI in Society and Public Services

Session 2: The Evolution of Machine Learning Models

From Simple Algorithms to Transformers

Duration: 3 Hours

Instructor: Mário Antunes

Part 1: The Learning Paradigms

1. Supervised Learning

“Learning with a Teacher”

- **Definition:** The model is trained on **Labeled Data**. It has input data (X) and the correct answers (Y).
- **The Goal:** Map inputs to outputs so accurately that it can predict Y for new, unseen X.
- **Analogy:** A student using flashcards with the answers written on the back. They guess, check the answer, and correct their mental model.
- **Examples:**
  - Spam Classification (Input: Email $\rightarrow$ Label: Spam/Not Spam).
  - House Price Prediction (Input: Specs $\rightarrow$ Label: Price).

2. Unsupervised Learning

“Learning by Discovery”

- **Definition:** The model is given **Unlabeled Data**. It has only inputs (X) and no answers.
- **The Goal:** Discover hidden structures, patterns, or groupings within the data.
- **Analogy:** A librarian receives a box of books with no titles or genres. They must organize them onto shelves based on which books “look similar” or share content.
- **Examples:**
  - Customer Segmentation (Grouping citizens by usage patterns).
  - Anomaly Detection (Spotting unusual credit card transactions).

3. Reinforcement Learning (RL)

“Learning by Trial and Error”

- **Definition:** An **Agent** interacts with an **Environment** and receives **Rewards** or **Penalties**.
- **The Goal:** Maximize the cumulative reward over time.
- **Analogy:** Training a dog. You don’t tell the dog *how* to sit (mechanics of muscle movement); you give a treat (reward) when it happens to sit, and ignore it (no reward) when it doesn’t.
- **Examples:**
  - Robotics (Learning to walk).
  - Game Playing (AlphaGo, Chess engines).
  - RLHF (Reinforcement Learning from Human Feedback) in ChatGPT.

Part 2: Shallow vs. Deep Models

Complexity and Architecture

- **Model Capacity:**
  - *Low Capacity:* Can only learn simple, linear relationships (e.g., "If Price goes up, Sales go down").
  - *High Capacity:* Can capture nuance, exceptions, and non-linear interactions (e.g., understanding sarcasm or recognizing a face).
- **The Architecture Metaphor:**
  - **Shallow (Flat):** Input $\rightarrow$ Rule $\rightarrow$ Output. Like a reflex action.
  - **Deep (Hierarchical):** Input $\rightarrow$ Layer 1 $\rightarrow$ Layer 2 $\rightarrow$ ... $\rightarrow$ Output. Like deep contemplation.
- **The Trade-off:**
  - **Simplicity** offers speed and transparency (Occam's Razor).
  - **Complexity** offers raw power and perception.

Defining the Architectures i

Shallow Models (Traditional ML)

- **Structure:** Simple algorithms that usually perform a direct mapping or a simple hierarchy.
- **Feature Engineering:** Requires humans to manually define what is important (e.g., “Is the email in all caps?”).
- **Examples:** Linear Regression, Decision Trees, SVM, Naive Bayes.

Defining the Architectures ii

Deep Models (Deep Learning)

- **Structure:** Artificial Neural Networks with many hidden layers (hence “Deep”).
- **Feature Learning:** The model automatically figures out what features are important (e.g., edges $\rightarrow$ shapes $\rightarrow$ faces).
- **Examples:** CNNs (Vision), RNNs/LSTMs, Transformers (LLMs).

Comparison: Shallow vs. Deep

Feature	Shallow Learning	Deep Learning
Data Requirements	Works well with small datasets.	Needs massive amounts of data.
Hardware	Runs on standard CPUs.	Requires GPUs/TPUs (Parallel compute).
Training Time	Seconds to Minutes.	Days to Months.
Interpretability	**White Box:** Easy to explain (e.g., "rejected because income < X").	**Black Box:** Hard to explain ("Neuron 405 activated").
Performance	Plateaus as data increases.	Scales with data (Scaling Laws).
Use Case	Tabular data (Excel), simple classifications.	Perceptual tasks (Vision, Audio, Text).

When to Use Which?

Use Shallow Models If:

1. You have structured/tabular data (SQL, Excel).
2. Explainability is legally required (e.g., denying a loan).
3. Compute resources are limited.
4. You have a small dataset (< 10k rows).

Use Deep Models If:

1. You are dealing with unstructured data (Images, Text, Voice).
2. State-of-the-art accuracy is the only goal.
3. You have massive datasets and GPU access.

Part 3: Learning as Optimization

The Landscape of Loss

“The Hiker in the Fog”

- **The Loss Function:** A mathematical formula that calculates “How wrong is the model?” (Error).
- **The Goal:** Find the lowest point in the landscape (Minimum Error).
- **The Challenge:** The model is a hiker in a thick fog. It cannot see the bottom; it can only feel the slope of the ground under its feet.

Gradient Descent

The Compass

- **Mechanism:**
  1. Calculate the **Gradient** (the slope/derivative) of the error with respect to the parameters.
  2. Take a small step in the opposite direction (downhill).
  3. Repeat until the slope is zero (bottom of the valley).
- **Learning Rate:** How big of a step to take.
  - *Too small:* Takes forever.
  - *Too big:* You might jump over the valley and miss the bottom.

Backpropagation

“The Blame Game”

Used specifically in Neural Networks to calculate the Gradient.

1. **Forward Pass:** Data goes in, prediction comes out. Error is calculated.
2. **Backward Pass:** The error is sent *backwards* through the network.
3. **Credit Assignment:** We calculate how much *each specific neuron* contributed to the error.
4. **Update:** Adjust the connection weights of the “guilty” neurons to reduce error next time.

Alternative Optimizations i

1. **Evolutionary Algorithms (Population-Based)**
   - **Blind Optimization:** Does not use gradients (calculus).
   - **Mechanism:**
     1. Create a population of 100 random models.
     2. Test them all.
     3. Kill the worst 50%.
     4. **Mutate** and **Breed** the survivors to create a new generation.
   - **Use Case:** When the problem has no smooth math solution (e.g., designing an antenna shape).

Alternative Optimizations ii

2. **Decision Tree Construction (Greedy Search)**
    - Does not use Gradient Descent.
    - **Mechanism:** Uses **Information Gain** (Entropy/Gini Impurity).
    - *Question:* "Which Yes/No question splits this data into the cleanest groups?"

Part 4: The Neural Network Revolution

The Artificial Neuron (Perceptron)

- **Biological Inspiration:**
  - **Dendrites:** Inputs (Data).
  - **Axon:** Output (Prediction).
  - **Synapse Strength:** Weights (w).
- **The Formula:**

$$y = \text{Activation}(\sum(inputs \times weights) + bias)$$

- **Activation Function (ReLU/Sigmoid):** Adds non-linearity. This is the “spark” that decides if the neuron fires or not.

Why Neural Networks Changed Everything i

- **Universal Function Approximators:** Mathematically proven to be able to mimic *any* function given enough neurons.
- **The Impact:**
  - **Computer Vision:** Convolutional Neural Networks (CNNs) allowed machines to "see" (edges $\rightarrow$ textures $\rightarrow$ objects).
  - **NLP:** Recurrent Neural Networks (RNNs) allowed machines to handle sequences (time).

Why Neural Networks Changed Everything i

- **Limitations:**
  - **Data Hungry:** Requires millions of examples.
  - **Catastrophic Forgetting:** Often forgets old tasks when learning new ones.
  - **The Vanishing Gradient:** In deep networks, the learning signal can fade away before reaching the early layers (solved later by ResNets and Transformers).

Part 5: The Era of LLMs

The Transformer Architecture (2017)

“Attention Is All You Need”

Before 2017, AI read text linearly (left-to-right). It often forgot the beginning of a long sentence by the time it reached the end.

The Transformer Solution:

- **Parallelization:** Reads the entire sentence at once.
- **Self-Attention Mechanism:** Allows the model to weigh the relationship between *every word* and *every other word* simultaneously.
    - *Example:* In “The animal didn’t cross the street because **it** was too tired,” Attention links “**it**” strongly to “**animal**”, not “street.”

What is a “Foundation Model”?

A paradigm shift in AI economics and engineering.

1. **Pre-Training (The Expensive Part):**
   - Train a massive model on “the whole internet” (TB of text).
   - **Task:** Self-Supervised Learning (Next-token prediction).
   - *Result:* A model that understands general language, logic, and world facts.
2. **Fine-Tuning (The Cheaper Part):**
   - Take the Foundation Model and train it slightly on a specific task (e.g., Medical coding, Legal analysis).
   - *Analogy:* It is easier to teach a doctor how to use a specific hospital software than to teach a software engineer how to be a doctor.

Inside the Brain of an LLM: Embeddings

Converting Language to Math

Computers cannot understand words; they understand numbers. **Embeddings** map words to vectors (lists of numbers) in a high-dimensional geometric space.

- **Semantic Proximity:** Words with similar meanings are mathematically close in this space.
  - *Distance:*
    $Distance(Cat, Dog) < Distance(Cat, Car).$
- **Vector Arithmetic:**

$$\vec{King} - \vec{Man} + \vec{Woman} \approx \vec{Queen}$$

$$\vec{Paris} - \vec{France} + \vec{Portugal} \approx \vec{Lisbon}$$

From Input to Output: The Process

1. **Tokenization:**
   - Text is chopped into chunks (tokens).
   - "Thinking" $\longrightarrow$ `[Think, ing]`. Roughly 0.75 words per token.
2. **Embedding:**
   - Tokens are converted to vectors (e.g., 12,288 dimensions for GPT-3).
3. **Transformer Layers (Attention):**
   - The model processes the context, understanding syntax and nuance.
4. **Probabilistic Generation:**
   - The model outputs a probability distribution for the *next* token.
   - It samples from this list (Temperature controls randomness).

Chain of Thought (CoT)

LLMs are probabilistic, not logical. They struggle with multi-step math or logic puzzles if forced to answer immediately.

- **Standard Prompting:**
  - *Q:* “Roger has 5 balls. He buys 2 cans of tennis balls. Each can has 3 balls. How many balls does he have?”
  - *A:* “11.” (Often wrong, guesses based on numbers seen).
- **Chain of Thought:**
  - *Prompt:* “Let’s think step by step.”
  - *Model output:* “Roger started with 5. 2 cans * 3 balls = 6 new balls. 5 + 6 = 11.”
  - **Mechanism:** Generating the intermediate steps puts the necessary numbers into the **Context Window**, allowing the attention mechanism to “attend” to the intermediate values to solve the final step.

Overcoming Limitations: RAG vs. CAG i

How do we stop LLMs from hallucinating facts? We give them the “Right Answers” along with the question.

Overcoming Limitations: RAG vs. CAG ii

1. **RAG (Retrieval-Augmented Generation):**
   - **The Mechanism:**
     1. *User asks:* "What is the policy on remote work?"
     2. *Retriever:* Searches a Vector Database (using Embeddings) to find the specific paragraphs in your PDF manual that mention "remote work."
     3. *Generator:* Feeds those paragraphs to the LLM.
     4. *Result:* The LLM answers using **only** that retrieved text.
   - **Analogy:** Taking an exam with access to a library. You run to the shelf, grab the specific book, read the page, and write the answer.
   - **Pros:** Scales to infinite data (TB of documents).

Overcoming Limitations: RAG vs. CAG iii

2. **CAG (Cache/Context-Augmented Generation):**
   - **The Mechanism:**
     - Instead of searching, we **pre-load** the *entire* knowledge base (e.g., the whole 500-page manual) into the model's massive Context Window (e.g., Gemini 1.5's 1M token window).
     - The model holds the whole dataset in its active "short-term memory" (KV Cache).
   - **Analogy:** Taking an exam with the textbook already open on your desk. No running to the library; instant access.
   - **Pros:** Faster (no retrieval step), better global reasoning (connects dots across the whole file).
   - **Cons:** Limited by context size (currently ~1-2 million tokens).

Part 6: The Frontier

Multimodality

Beyond Text

Current models (GPT-4o, Gemini 1.5) are **Native Multimodal**.

- **Not just glue:** They are not separate models glued together (e.g., speech-to-text $\rightarrow$ LLM $\rightarrow$ text-to-speech).
- **Native Training:** The model is trained on tokens representing text, image patches, and audio waves *simultaneously*.
- **Capability:** The model can "hear" tone of voice (sarcasm) or "see" spatial relationships in images directly.

Agentic Models

From “Chatbot” to “Employee”

- **Chatbot (Passive):**
  - User: “Book me a flight.”
  - Bot: “I cannot browse the web. Here is how you can do it.”
- **Agent (Active):**
  - **Perception:** Reads email, sees calendar.
  - **Reasoning:** “I need to find a flight that fits this gap in the calendar.”
  - **Tool Use:** Uses an API to search Expedia.
  - **Action:** Books the flight and sends the confirmation.

The Loop: Observation $\rightarrow$ Thought $\rightarrow$ Action $\rightarrow$ Observation (Did it work?)

Q&A