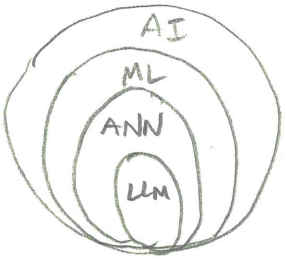
Lecture 9.5 - Coding and "A.I."

I. Neural networks and Large Language models

* Caveat: I am not a machine learning or artificial intelligence expert. I am learning alongside you here.

* Let us define some terms:

- Artificial Intelligence (AI)
- Machine Learning (ML)
- Artificial Neural Networks / Deep Learning. (ANN)
- Large Language Model. (LLM)



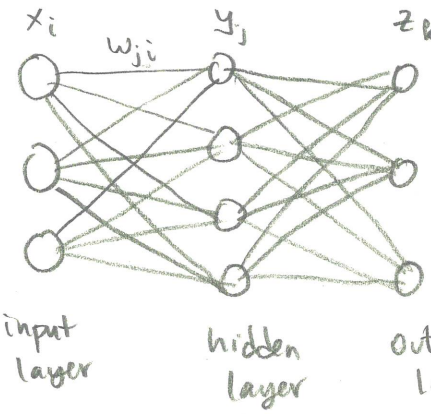
- Artificial Intelligence: A field of study, mainly in computer science, concerned with how machines & software can be made "intelligent", ie. acquire and apply knowledge and skills.

- Machine learning: A problem within AI, where the goal is the creation of an algorithm based on data / experience, rather than explicit coding. In ML the computer can improve its performance on a task over time.

- Supervised learning - a human labels data that guides the ML process (classification, regression)
- Unsupervised learning - no human intervention needed to guide ML process / pattern recognition.

Artificial
- ^Neural Networks / Deep learning : An algorithm for a universal approximator* inspired by a simplified model of the brain.

* of a function or algorithm



x_i ○ : a node (inspired by a neuron)

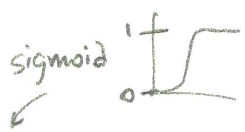
w_i —— : connection with a weight (inspired by a synapse)

$$y_j = \sum_i w_{ji} x_i \quad \text{(linear)}$$

or

$$y_j = \left[1 + \exp\left(-\sum_i w_{ji} x_i\right)\right]^{-1} \quad \text{(non-linear)}$$

sigmoid



- One uses an optimization algorithm to find the best weights w_{ji} for the output to match training data. This is the "learning" process.

- Deep learning involves the use of many hidden layers between the output and input. This expands the ability of the ANN to approximate the desired function.

- Two key aspects of an ANN model are: (1) the architecture of the model (weights, how many, how connected) and (2) the data used to determine the weights

- Large Language model: A type of ANN trained to take text as inputs and predict a sequence of text that follows as the output.

  - Text is "encoded" as numbers. These are called tokens.
  - Generative Pre-trained Transformer (GPT) models use an architecture called a "transformer"
  - GPT is trained on much of the text on the internet (Common Crawl data). → 500×10^9 tokens in dataset.
  - GPT-3 has 175×10^9 parameters (weights) and 96 layers.
  - ChatGPT has also been subsequently modified (fine-tuned) using reinforcement learning using human feedback (RLHF) to make the model align more w/ human values.

* key takeaways about LLMs like ChatGPT

- LLMs are a sophisticated mathematical model for predicting text based on a prompt.
- LLMs are trained on data from the internet.
- LLMs do not make perfect predictions. They can "hallucinate" or predict text that seems crazy to a human user.
- LLMs are cutting edge algorithms, and we are still learning a lot about them. They can be very powerful, but there is no guarantee of 100% accuracy.

II. Using AI tools for coding

A. What tools currently exist for coding?

* I see two types right now (as of Feb 2024)

(1) Web-based chat tools: ChatGPT, Google Gemini, Microsoft Copilot (chat bots)

(2) Autocompletes inside an IDE. Github copilot, Microsoft copilot inside visual studio.

* comments on these tools (again, as of Feb 2024)

chat GPT - v3.5 is free, v4.0 requires subscription ($20/mo)
Not connected to the internet
Has APIs to make extensions

Google Gemini - was Google Bard, not based on open AI's GPT. Not connected to internet search (yet).
Free chatbot "Advanced" version in Beta for $20/mo.
Trained by Google (based on Pathways language model, PaLM). Claims higher accuracy (in training & data set).

Microsoft Copilot - Based on GPT v4.0. Includes search and gives some references.
Free chatbot, $30/mo. subscription to integrate with Microsoft 365 and VS code.

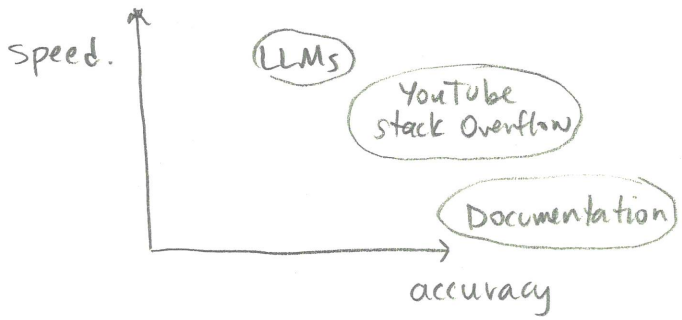
⛤ There are many other tools out there being developed and being used. This landscape is changing quickly.

Github Copilot - $10 per month code autocomplete.
Not a chatbot.

* There are many other useful sources beyond AI!

- Internet search
- Documentation ← authoritative!
- Stack Overflow ← human experts!
- Online courses, textbooks ← pedagogy! human experts!
- YouTube ← pedagogy! human experts!

- see my references page for some really good resources.



* My cautionary message:

  - Learning is not the same thing as getting your homework done.

    (true north vs. magnetic north)

  - learning takes effort. It is a physical and spiritual process. No substitutes.

  - Tools can greatly enhance your effectiveness, but then you need the tool. You compensate. This is ok. Even desirable at times. Choose wisely how you will specialize

B. Prompt Engineering.

* LLMs (e.g. Chat GPT) take tokens (words) as inputs. When they run, they take your entire chat history as an input to produce the next output (up to some limit)

  - My research suggests GPT 3.5 token limit is 2048 or about 4 single spaced pages of text (≈2000 words).

- This means you can use a conversation to engineer the output you want from the LLM. This is called "prompt engineering"!

* Here are some prompt engineering strategies that others have suggested:

- Chain-of-thought prompting. (CoT)
  - go step by step. give it time to "think"
  - ask it to provide its reasoning.
- provide a reference text (digitized notes or article)
- split task into smaller subtasks (higher error rate on more complex task)
- use multiple tools (if it stinks at math, use something else for that part)
- Evaluate answers to known questions first.
- Provide examples
- write clear instructions. It can't read your mind!

III. Examples & Activity

* Recursion
* Plot of normal distribution

} Examples. Do in class with the students.

* lessons :

- Being vague is curse
- ChatGPT can make mistakes
  (CDF less than 99% not more than)
- It can use things you don't know about
  → Honesty as an engineer.
  You should not present work you do not understand to others
  This includes homework.
  → You should know what the code is doing.
- It works well for simple problems
- It can really help you do things faster.
- It can really help you learn new commands or features or faster ways of doing things.