The Beginnings of AI and ML

- 1950: Alan Turing -- “Computing Machinery and Intelligence”
  - Introduced what would come to be known as the “Turing Test”
  - Can interrogator distinguish between computer and human?
  - If not, then we might infer that the “machine thinks”

- 1956: Dartmouth AI Conference
  - 1955: John McCarthy coins term “Artificial Intelligence”

- 1959: Arthur Samuel develops learning checkers program
  - Evaluation function of board with learned weights
  - Learning based on data from professional players and playing against itself
  - Program was eventually able to beat Samuel
Models Can Be Complicated

- Binomial or Normal distribution have well-defined parameters to estimate
- Sometimes, you don’t have a well-known distribution
  - E.g., want to estimate the probability of the next word given a set of words that came before, such as in a Large Language Model
- Need to learn a more complicated model
What is Machine Learning?

• Many different forms of “Machine Learning”
  ▪ We focus on the problem of prediction

• Want to make a prediction based on observations
  ▪ Vector $\mathbf{X}$ of $m$ observed variables: $<X_1, X_2, \ldots, X_m>$
    ▪ $X_1, X_2, \ldots, X_m$ are called “input features/variables”
    ▪ Also called “independent variables,” but this can be misleading!
      • $X_1, X_2, \ldots, X_m$ need not be (and usually are not) independent
  ▪ Based on observed $\mathbf{X}$, want to predict unseen variable $Y$
    ▪ $Y$ called “output feature/variable” (or the “dependent variable”)
  ▪ Seek to “learn” a function $g(\mathbf{X})$ to predict $Y$: $\hat{Y} = g(\mathbf{X})$
    ▪ When $Y$ is discrete, prediction of $Y$ is called “classification”
    ▪ When $Y$ is continuous, prediction of $Y$ is called “regression”
Training a Learning Machine

• Set-up of the *supervised* learning task
  ▪ We are given set of $N$ “training” *instances*
    o Each training instance is pair: ($<x_1, x_2, \ldots, x_m>$, $y$)
    o Training instances are *previously* observed data
    o Gives the output value $y$ associated with each observed vector of input values $<x_1, x_2, \ldots, x_m>$
  ▪ Learning: use training data to specify $g(X)$
    o Generally, first select a parametric form for $g(X)$
    o Then, estimate parameters of model $g(X)$ using training data
    o For regression, usually want $g(X)$ that minimizes $E[(Y - g(X))^2]$
      • Mean squared error (MSE) “loss” function. (Others exist.)
    o For classification, generally best choice of $g(X) = \arg \max_y \hat{P}(Y \mid X)$
The Machine Learning Process

- **Training data**: set of $N$ pre-classified data instances
  - $N$ training pairs: $(x^{(1)}, y^{(1)})$, $(x^{(2)}, y^{(2)})$, ..., $(x^{(N)}, y^{(N)})$
    - Use superscripts to denote $i$-th training instance
- **Learning algorithm**: method for determining $g(X)$
  - Given a new input observation of $X = <X_1, X_2, ..., X_m>$
  - Use $g(X)$ to compute a corresponding output (prediction)
  - When prediction is discrete, we call $g(X)$ a “classifier” and call the output the predicted “class” of the input
A Grounding Example: Linear Regression

• Predict real value $Y$ based on observing variable $X$
  - Assume model is linear: $\hat{Y} = g(X) = aX + b$
  - Training data
    - Each vector $X$ has one observed variable: $<X_1>$ (just call it $X$)
    - $Y$ is continuous output variable
    - Given $N$ training pairs: $(<x>^{(1)},y^{(1)}), (<x>^{(2)},y^{(2)}), \ldots, (<x>^{(N)},y^{(N)})$
      - Use superscripts to denote $i$-th training instance
  - Determine $a$ and $b$ minimizing $E[(Y - g(X))^2]$
    - Take partial derivatives, set to 0, solve simultaneous equations
    - Thankfully, we won’t do that right now
Motivation for Artificial Neurons

- A neuron
- An artificial neuron
- Formalized

\[ \begin{align*}
&\text{Dendrites} \\
&\text{Axon} \\
&\text{Synapses} \\
&\text{Neuron scheme}
\end{align*}\]

\[ \begin{align*}
\text{dendrites} & \quad \text{neuron} \\
\text{axon} & \quad \text{formalized}
\end{align*}\]

\[ \begin{align*}
X_1 & \quad w_1 \\
X_2 & \quad w_2 \\
X_3 & \quad w_3 \\
X_4 & \quad w_4 \\
& \quad y
\end{align*}\]
Perceptron Learning Algorithm

Compute $S = \sum_{i=0}^{n} X_i \cdot w_i$

If $S > 0$, set $Q = 1$, else $Q = 0$ \textit{(Q is prediction)}

if $(Q \neq y)$ \{ 
    \text{if (Q = 1) \{}
        For all weights $w_i$ (where $i = 0$ to $n$)
        \[ w_i = w_i - X_i \]
    \} else {
        For all weights $w_i$ (where $i = 0$ to $n$)
        \[ w_i = w_i + X_i \]
    }
Learning Linearly Separable Functions

- Consider function: \( y = x_1 \) and \( x_2 \)
  - Note: \( y = 1 \) iff both \( x_1 \) and \( x_2 = 1 \)

- Can draw a line that successfully separates all the \( y = 1 \) points (blue) from the \( y = 0 \) points (red)
Data Often Not Linearly Separable

- Many data sets/functions are not linearly separable
  - Consider function: \( y = x_1 \text{ XOR } x_2 \)
  - Note: \( y = 1 \) iff one of either \( x_1 \) or \( x_2 = 1 \)
  - Not possible to draw a line that successfully separates all the \( y = 1 \) points (blue) from the \( y = 0 \) points (red)
Network of Neurons for XOR

XOR

X1 & X2

X1 or X2

XOR

X1

X2
Biological Basis for Neural Networks

- A neuron

- Your brain

Actually, it’s probably someone else’s brain
Neural Networks

- Neural network:

  ![Diagram of a neural network with input nodes $x_1$, $x_2$, $x_3$, $x_4$ and output nodes.]

  - But how do we learn all the weights in the network?
  - 1986: Back-propagation algorithm
    - Due to Rumelhart, Hinton and Williams
    - Overcomes limitations of Perceptron
    - Shows general learning mechanism
  - With enough nodes, can approximate *any* function
The Deep Learning Revolution

2010’s: Deep Learning
- Essentially, neural networks with many nodes/layers and billions of parameters
  - Represent enormously complicated functions
- Have led to impressive (human beating) results on a number of tasks

A sampling of systems related to deep learning
- Convolutional neural network (e.g., LeNet for digit recognition)
- Recurrent neural networks (sequences, e.g. translation)
- Large scale image recognition tasks (e.g., ImageNet)
- Generative adversarial networks (e.g., text to image generation)
- Transformers and large language models (GPT, LLaMA, BARD, etc.)
Large Language Model: GPT-3

- GPT-3 (Generative Pre-trained Transformer 3)
  - Deep learning (175 billion parameters) model developed by OpenAI
  - Predictive language model: predict next word given previous text
  - Give it a short prompt to generate text

Example:
For my child's lunch, I made a peanut ______
For my child's lunch, I made a peanut butter ______
For my child's lunch, I made a peanut butter and ______
For my child's lunch, I made a peanut butter and jelly ______
For my child's lunch, I made a peanut butter and jelly sandwich.