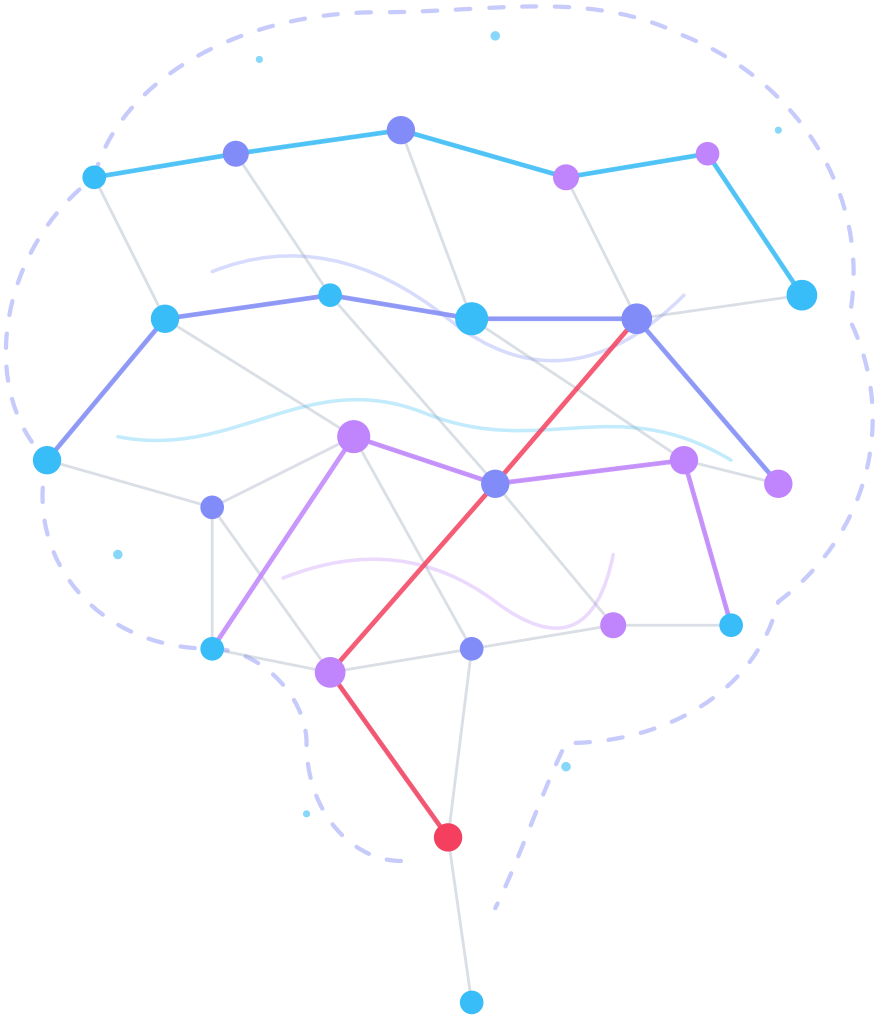
Machine Learning

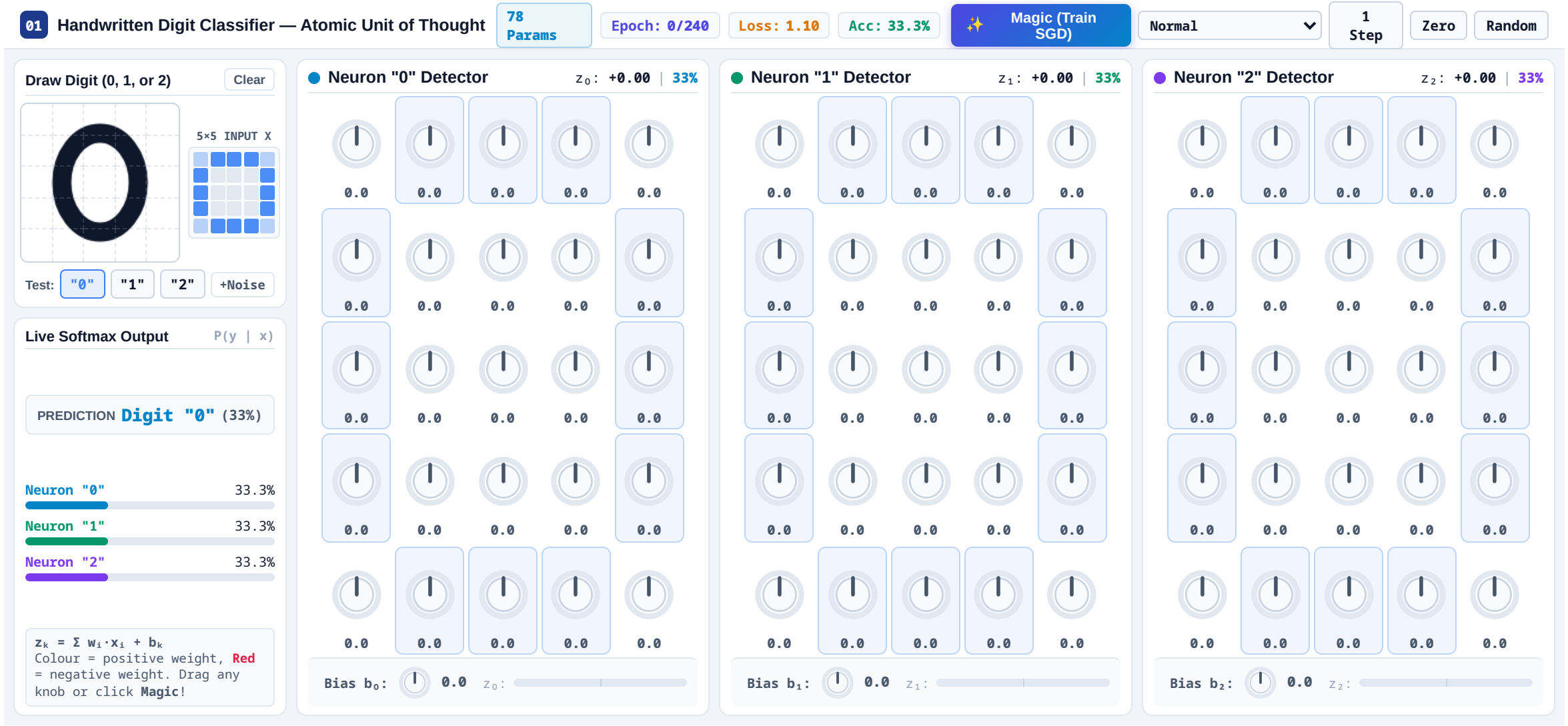
Imperial College London

Nuri Cingillioglu

Part 1: Atomic Unit of Thought



Opening Act: Mini-MNIST



Can machines think?

> I propose to consider the question, "Can machines think?" This should begin with definitions of the meaning of the terms "machine" and "think." Alan Turing (1912-1954)

COMPUTING MACHINERY AND INTELLIGENCE

Turing, A.M. (1950). Computing machinery and intelligence. Mind, 59, 433-460.



Many ways to think about intelligence

- Optimisation Theory
- Statistics
- Probabilistic Modelling
- Information Theory
- Cognitive Science
- Neuroscience
- Operations Research
- Biology

This module asks the question: How to build an **artificial brain**? We will touch on these different fields of study.

- We will have good moments.
- We will have bad times.
- And there will Agly (deliberate typo).

My take on this course

ML is a mess. I will try my best to guide you through it.

- All slides are hand-written (probably last time ever)
- All diagrams and demos are AI generated.

My background

- MEng, PhD at Imperial. Been around for a while...
- Neuro-symbolic AI, natural intelligence, and AGI.
- Previous layovers at Google DeepMind, Google.
- Passionate about teaching and innovation.

https://www.doc.ic.ac.uk/~nuric/
https://www.linkedin.com/in/nuric17/



Broadly speaking

- **Supervised Learning:** Given input $\mathbf{x}$ and output $\mathbf{y}$, we want to learn a function f such that $f(\mathbf{x}) = \mathbf{y}$
  - e.g. regression, classification
- **Unsupervised Learning:** Given input $\mathbf{x}$, we want to learn a function f such that $f(\mathbf{x})$ reveals some structure in $\mathbf{x}$
  - e.g. clustering, dimensionality reduction
- **Reinforcement Learning:** Given an environment, maximise some reward signal.

> What about self-supervised learning such as next token prediction?

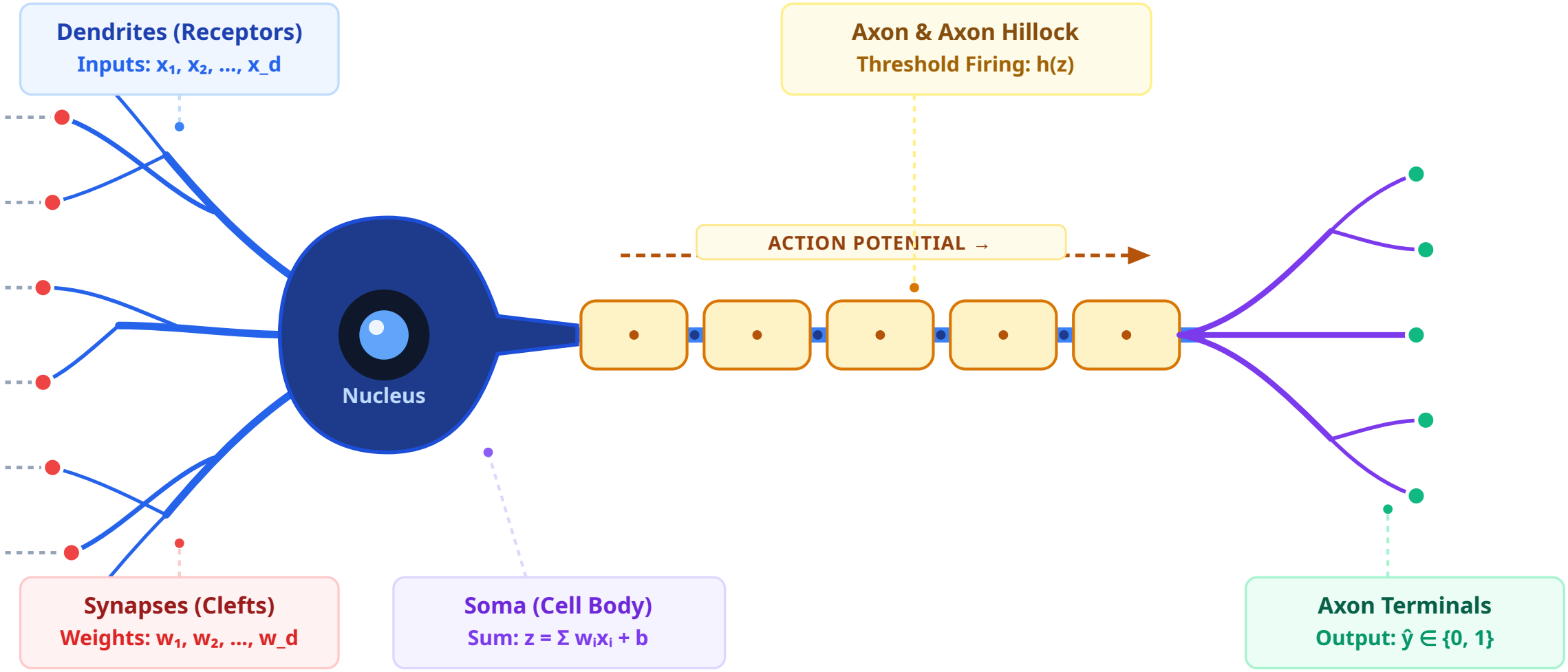
Cool AI examples

Normally we show many cool examples of AI but you already know them. Ask an LLM to generate cool demos for you!

- Image generation, classification
- Text generation, reasoning, coding assistants
- AlphaGo, AlphaFold, AlphaStar, AlphaZero, ~~AlphaNuri~~...

Atomic Unit of Thought

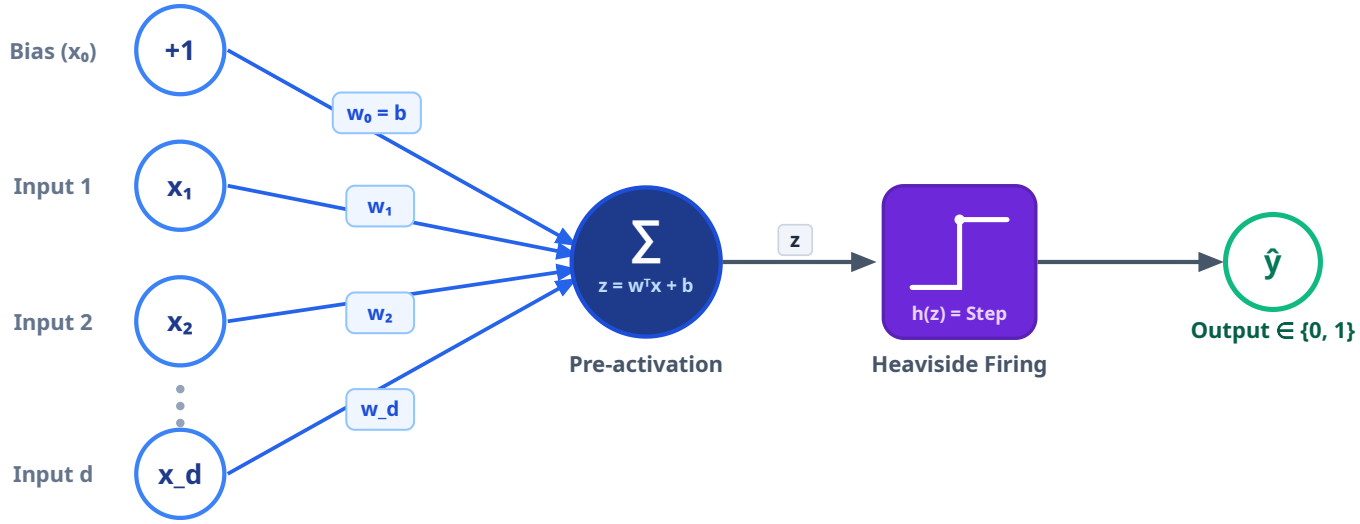
The biological neuron is a complex system of dendrites, soma, axon, and synapses.



Biological vs. Artificial Neuron

Biological Component	Computational Equivalent	Role in Information Processing
Dendrites	Inputs $\mathbf{x} = [x_1, \ldots, x_d]^T$	Receive incoming signals from input features or upstream neurons
Synapses	Weights $\mathbf{w} = [w_1, \ldots, w_d]^T$	Modulate input strength (excitatory > 0 / inhibitory < 0, simplified)
Soma (Cell Body)	Pre-activation $z = \mathbf{w}^T\mathbf{x} + b$	Spatially and temporally integrates weighted inputs
Axon Hillock	Activation function $h(z)$	Evaluates all-or-none firing threshold (action potential)
Axon & Terminals	Output prediction $\hat{y}$	Transmits output signal to downstream units

The Perceptron (1958)



Rosenblatt, F. (1958). "The perceptron: A probabilistic model for information storage and organisation in the brain". Psychological Review. 65 (6): 386–408. https://doi.org/10.1037%2Fh0042519



Perceptron

We'll start our journey in 1958 with the following equation:

$$f(x) = h(\mathbf{w}^T\mathbf{x} + b) = h(\sum_{i=1}^{n} w_i x_i + b) = h(\mathbf{w}^T[\mathbf{x}, 1])$$

where h is the activation function, $\mathbf{w}$ is the weight vector, $\mathbf{x}$ is the input vector, and b is the bias. We can also absorb b into the weight vector by adding an extra 1 to $\mathbf{x}$.

In 1958, the activation function was the Heaviside step function:

$$h(z) = \begin{cases} 1 & z \geq 0 \\ 0 & z < 0 \end{cases}$$

Let there be DATA

We organise data into a matrix $\mathbf{X} \in \mathbb{R}^{n \times d}$ and a vector $\mathbf{y} \in \mathbb{R}^n$

For example, consider logical AND:

$$\mathbf{X} = \begin{bmatrix} 0 & 0 \\ 0 & 1 \\ 1 & 0 \\ 1 & 1 \end{bmatrix}, \quad \mathbf{y} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

then a single data point is $\mathbf{X}_0 = \mathbf{x} = [0, 0]$ and $y = 0$. Usually, capital bold is matrix, lower case bold is vector and lower case is scalar.

> It is difficult to find a single convention to write these equations and they vary between textbooks. My advice $\rightarrow$ just go with it.

Let there be learning

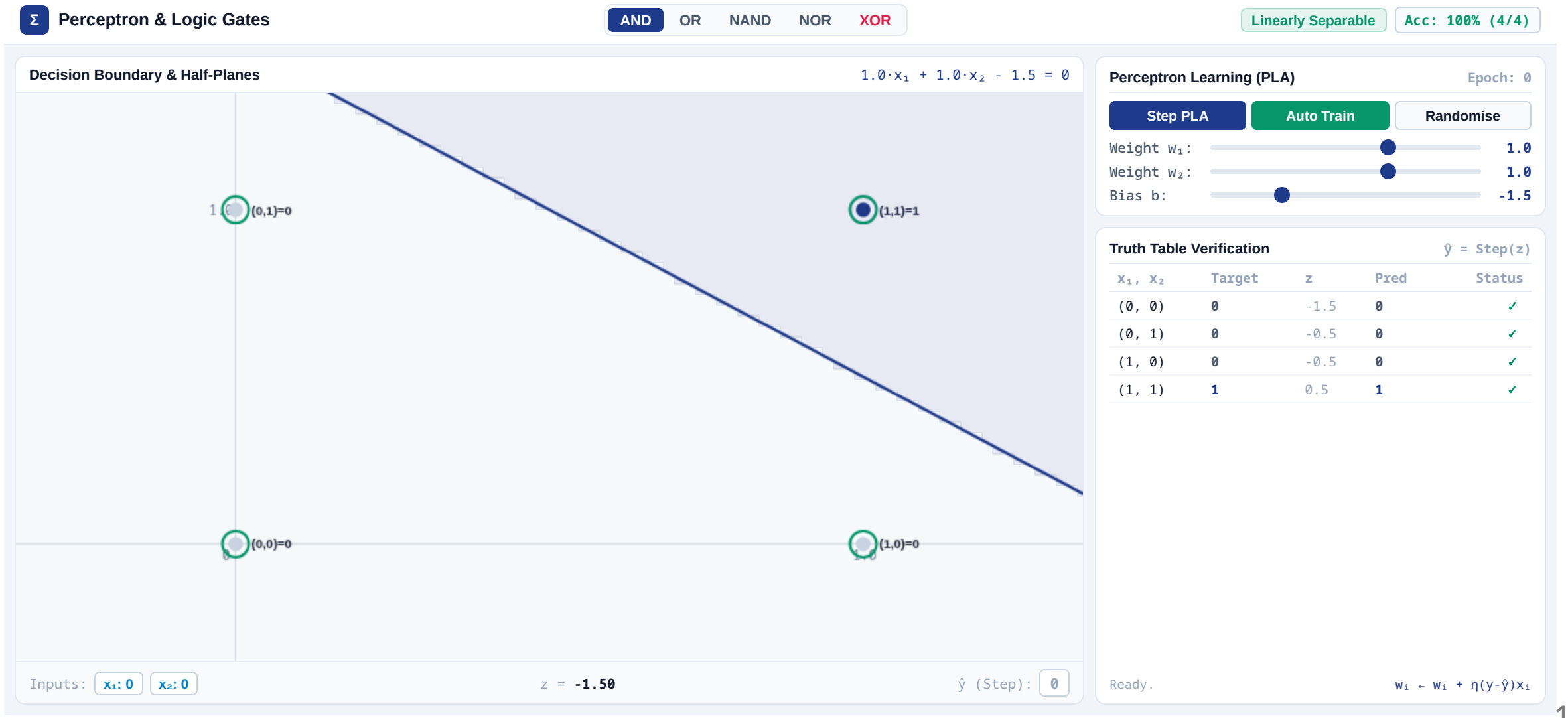
$$\mathbf{w} \leftarrow \mathbf{w} + \eta(y - \hat{y})\mathbf{x}$$

where η is the learning rate, $y \in \{0, 1\}$ is the true label, and $\hat{y} = f(\mathbf{x}) \in \{0, 1\}$ is the predicted label.

Error $(y - \hat{y})$	**Prediction State**	**Weight Update Action**
$+1$	False Negative ($y = 1, \hat{y} = 0$)	$\mathbf{w} \leftarrow \mathbf{w} + \eta\mathbf{x}$ (pulls $\mathbf{w}$ towards $\mathbf{x}$)
-1	False Positive ($y = 0, \hat{y} = 1$)	$\mathbf{w} \leftarrow \mathbf{w} - \eta\mathbf{x}$ (pushes $\mathbf{w}$ away from $\mathbf{x}$)
0	Correct classification	No change to $\mathbf{w}$

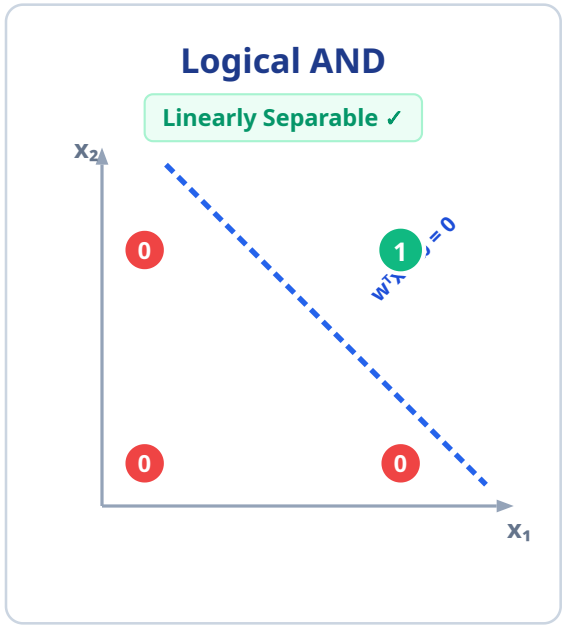
> Very underwhelming to say machine learning is just an equation we update. If this upsets you, be prepared for disappointment.

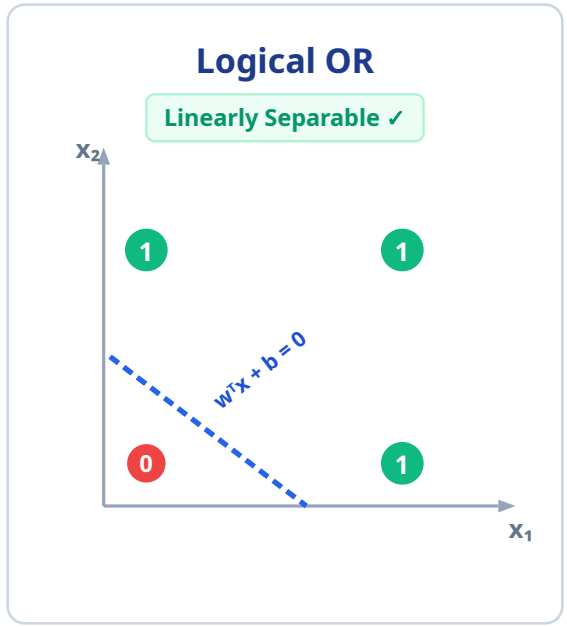
Learning in action

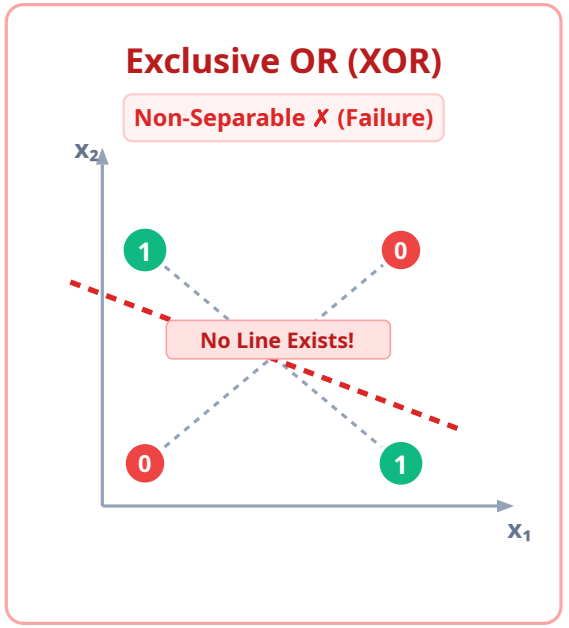


What about XOR?

$$\mathbf{X} = \begin{bmatrix} 0 & 0 \\ 0 & 1 \\ 1 & 0 \\ 1 & 1 \end{bmatrix}, \quad \mathbf{y} = \begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \end{bmatrix}$$





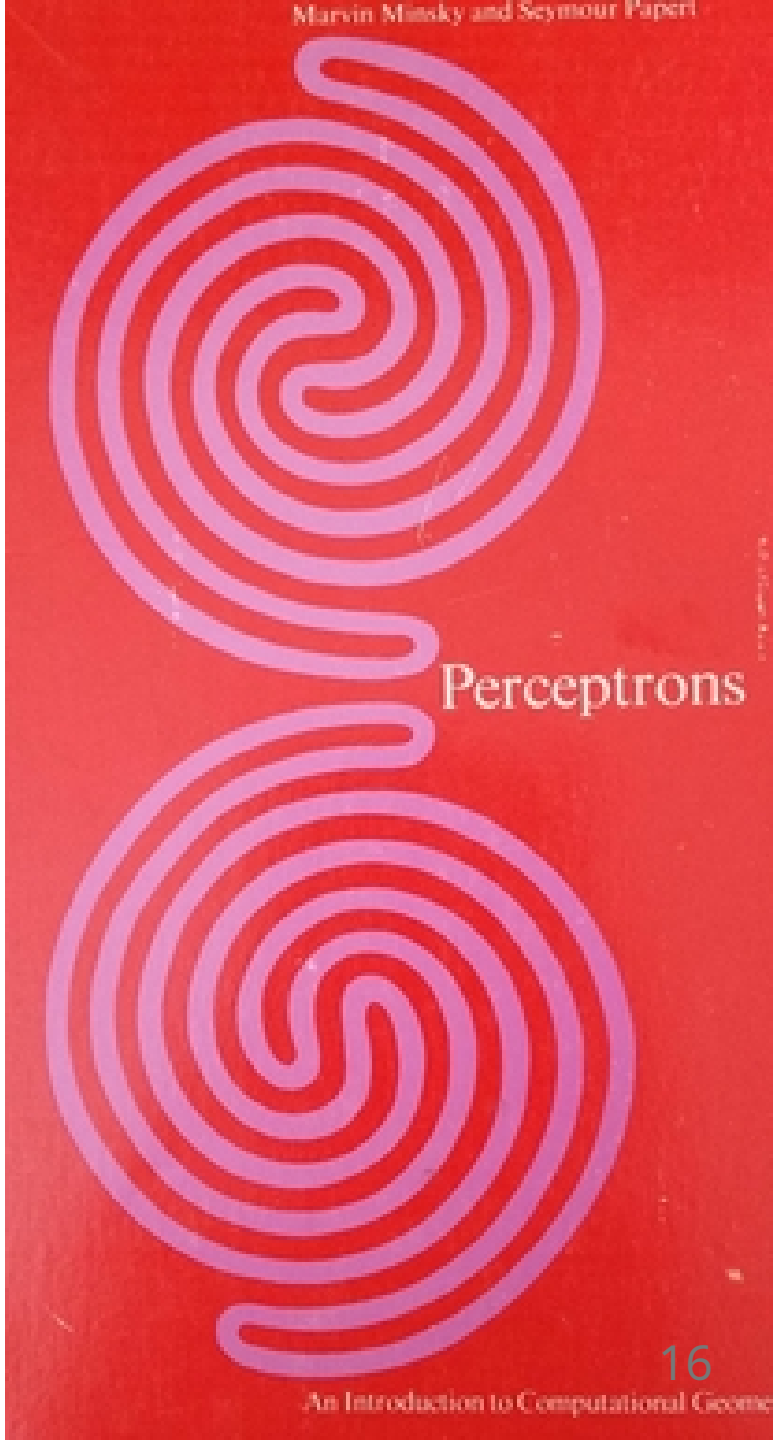


Minsky & Papert, 1969

What could they have possibly done to start the AI winter?

Perceptrons: An Introduction to Computational Geometry By: Marvin Minsky, Seymour A. Papert





XOR Contradiction: 1. Setup

x_1	x_2	$\mathrm{XOR}(x_1, x_2)$
0	0	**0**
0	1	**1**
1	0	**1**
1	1	**0**

Perceptron Classifier

A single-layer perceptron classifies via an affine hyperplane threshold:

$$y = \begin{cases} 1 & \text{if } w_1 x_1 + w_2 x_2 + b > 0 \\ 0 & \text{if } w_1 x_1 + w_2 x_2 + b \leq 0 \end{cases}$$

XOR Contradiction: 2. Four Inequalities

Assume there exist real parameters $w_1, w_2, b \in \mathbb{R}$ such that the perceptron correctly classifies all 4 XOR inputs:

Input (x_1, x_2)	Target y	Condition	Inequality Label
$(0,0)$	0	$w_1(0) + w_2(0) + b \leq 0$	$b \leq 0$ --- (1)
$(0,1)$	1	$w_1(0) + w_2(1) + b > 0$	$w_2 + b > 0$ --- (2)
$(1,0)$	1	$w_1(1) + w_2(0) + b > 0$	$w_1 + b > 0$ --- (3)
$(1,1)$	0	$w_1(1) + w_2(1) + b \leq 0$	$w_1 + w_2 + b \leq 0$ --- (4)

3. Deriving the Contradiction

Summing Class 1 ($y = 1$)

Add inequalities **(2)** and **(3)**:

$$(w_2 + b) + (w_1 + b) > 0 + 0$$

$$(w_1 + w_2 + b) + b > 0 \quad \text{--- (A)}$$

Summing Class 0 ($y = 0$)

Add inequalities **(4)** and **(1)**:

$$(w_1 + w_2 + b) + b \leq 0 + 0$$

$$(w_1 + w_2 + b) + b \leq 0 \quad \text{--- (B)}$$

> **Hint:** Notice that the left-hand sides of **(A)** and **(B)** are identical!

4. Contradiction

Comparing (A) and (B)

Define $S = (w_1 + w_2 + b) + b$. The simultaneous system requires:

$$0 < S \leq 0$$
$$\implies 0 < 0$$

Conclusion

A real scalar cannot be strictly positive and non-positive simultaneously:

$$\therefore \text{No such parameters } (w_1, w_2, b) \text{ exist. } \blacksquare$$

A single-layer perceptron **cannot** compute the XOR function!

Show me the code: Perceptron

`sklearn` is a popular library for machine learning in Python. All the classic non-deep learning algorithms you might ever use or don't use are in there.

```
from sklearn.linear_model import Perceptron

# Logical AND truth table
X = [[0, 0], [0, 1], [1, 0], [1, 1]]
y = [0, 0, 0, 1]

clf = Perceptron()
clf.fit(X, y)

print("Predictions:", clf.predict(X))  # [0, 0, 0, 1]
print("Accuracy:   ", clf.score(X, y))    # 1.0
```

k-Nearest Neighbours

Would you believe that the simplest non-trivial algorithm can solve XOR? Easy.

1. Neighbourhood Identification

Given training set $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^{N}$ and query point $\mathbf{x}$, find the k closest instances $\mathcal{N}_k(\mathbf{x})$:

$$\mathcal{N}_k(\mathbf{x}) = \arg \min_{\substack{\mathcal{S} \subseteq \{1,\ldots,N\} \\ |\mathcal{S}|=k}} \sum_{i \in \mathcal{S}} d(\mathbf{x}, \mathbf{x}_i)$$

For $k = 1$ (closest example):

$$i^* = \arg\min_i d(\mathbf{x}, \mathbf{x}_i)$$

2. Decision Rule

Majority Voting (Classification):

$$\hat{y} = \arg\max_{c \in \mathcal{Y}} \sum_{i \in \mathcal{N}_k(\mathbf{x})} \mathbb{I}(y_i = c)$$

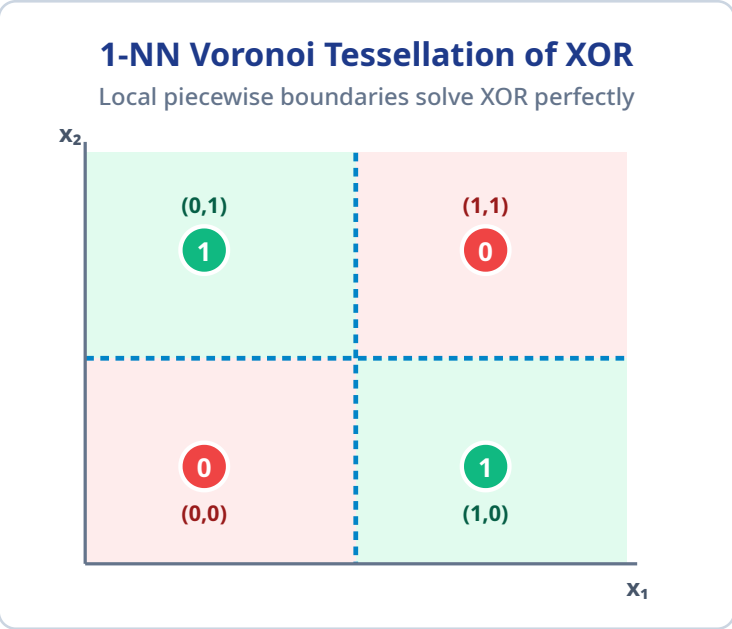
Class Probability Estimate:

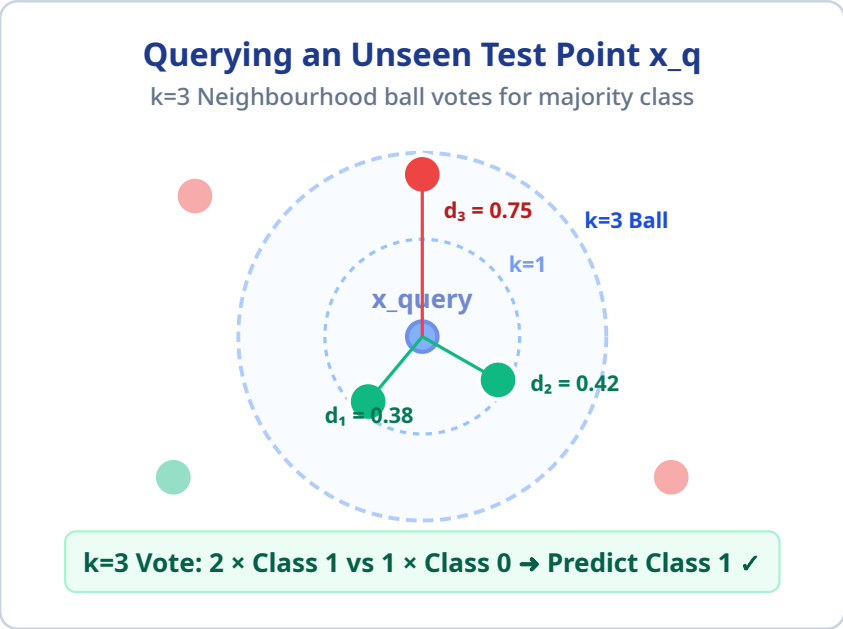
$$P(y = c \mid \mathbf{x}) = \frac{1}{k} \sum_{i \in \mathcal{N}_k(\mathbf{x})} \mathbb{I}(y_i = c)$$

Mean Value (Regression):

$$\hat{y} = \frac{1}{k} \sum_{i \in \mathcal{N}_k(\mathbf{x})} y_i$$

k-Nearest Neighbours: Geometric Intuition





- **1-NN Voronoi Tessellation:** Partitions the space into convex polyhedra. Each cell shares the label of its generating point.
- **Solves XOR naturally:** Non-linear decision boundaries emerge from local piecewise linear facets!

What does "near" mean?

The notion of similarity or distance is key to many machine learning algorithms.

Let me repeat that: **It is very very important to be able to measure similarity of things.**

You have probably seen different distance metrics before:

- Manhattan distance: $D(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^{d} |x_i - y_i|$
- Euclidean distance: $D(\mathbf{x}, \mathbf{y}) = \sqrt{\sum_{i=1}^{d} (x_i - y_i)^2}$
- Chebyshev distance: $D(\mathbf{x}, \mathbf{y}) = \max_i |x_i - y_i|$

> It is very very important to be able to measure **similarity** of things.

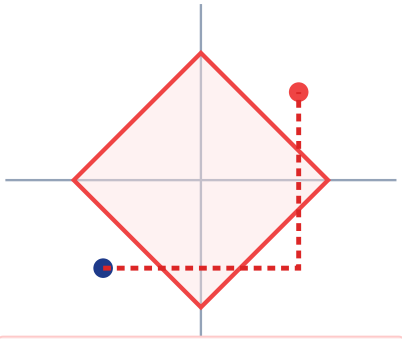
Minkowski Distance: p-norms

Minkowski Distance (L_p norm): General family parameterised by $p \geq 1$:

$$D(\mathbf{x}, \mathbf{y}) = \|\mathbf{x} - \mathbf{y}\|_p = \left(\sum_{i=1}^{d} |x_i - y_i|^p \right)^{\frac{1}{p}}$$

L_1: Manhattan (Taxicab)

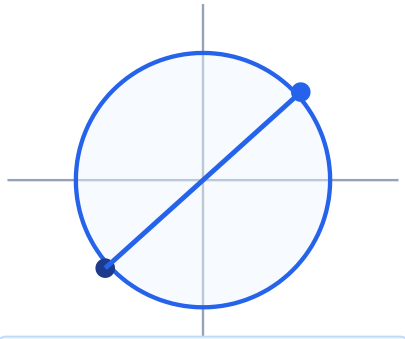
$d_1(x, x') = \Sigma\ |x_i - x'_i|$



Unit Ball: Diamond (Corners)

L_2: Euclidean (Standard)

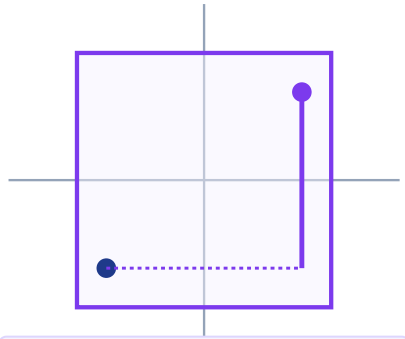
$d_2(x, x') = \sqrt{(\Sigma\ (x_i - x'_i)^2)}$



Unit Ball: Isotropic Circle

L_∞: Chebyshev (Max Norm)

$d_\infty(x, x') = \max_i\ |x_i - x'_i|$



Unit Ball: Axis-Aligned Square

Show me the code: k-Nearest Neighbours

Unlike the Perceptron, 1-NN solves non-linearly separable **XOR** easy peasy:

```
from sklearn.neighbors import KNeighborsClassifier

# Non-linearly separable XOR
X = [[0, 0], [0, 1], [1, 0], [1, 1]]
y = [0, 1, 1, 0]

knn = KNeighborsClassifier(n_neighbors=1)
knn.fit(X, y)

print("Predictions:", knn.predict(X))  # [0, 1, 1, 0]
print("Accuracy:   ", knn.score(X, y))    # 1.0
```

Instance-Based vs. Model-Based Learning

Compare where N is number of data points and d is input / model dimensionality:

Model-Based

Example: Perceptron, Neural Networks

- **Training ($\mathcal{O}(N \cdot d)$):** Learns a compact model $\mathbf{w}, b$ from data.
- **Inference ($\mathcal{O}(d)$):** Fast; evaluates $f(\mathbf{x})$. Training data can be discarded.
- **Memory ($\mathcal{O}(d)$):** Fixed, independent of dataset size N.
- **Inductive Bias:** Strong model assumptions (e.g. linear boundary).

Instance-Based

Example: k-Nearest Neighbours

- **Training ($\mathcal{O}(1)$):** Stores training instances $\mathcal{D}$ in memory.
- **Inference ($\mathcal{O}(N \cdot d)$):** Scans N examples at query time.
- **Memory ($\mathcal{O}(N \cdot d)$):** Grows linearly with dataset size N.
- **Inductive Bias:** Weak, local assumptions (nearby points).

Why Model-Based?

Everything we will look at will almost exclusively be model-based. **WHY?**

Because it generalises patterns from data into a compact mathematical function, allowing for much faster predictions and lower memory usage at scale.

- *After all, we don't carry a copy of the world to live, i.e. to identify a chair I don't need to recall every chair I've seen in my life*.

What we mean by **model** is some fittable set of parameters $\mathbf{w}$ or θ that map from a representation of data to a prediction.

> But how does your brain learn? There seems to be some memorisation, e.g. for the exam?

Pillars of Machine Learning

- **Model:** $f(x; \theta)$ where θ are the learnable parameters like weights and biases.
- **Data:** $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^N$ or $\mathcal{D} = \{(x_i)\}_{i=1}^N$ for unlabelled data.
- **Task:** Objective, Loss, Cost: something like $\min_\theta \sum_{i=1}^N L(y_i, f(x_i; \theta))$

The idea is to improve model parameters θ using $\mathcal{D}$ such that we improve on the task. In other words, *machine learning*.

> If your task is to estimate human lifespan using data that contains smokers who live longer than non-smokers, machine will learn ...

> If the brain is the model, and the world is the data, what was the task you were improving on all this time???

Importance of Data

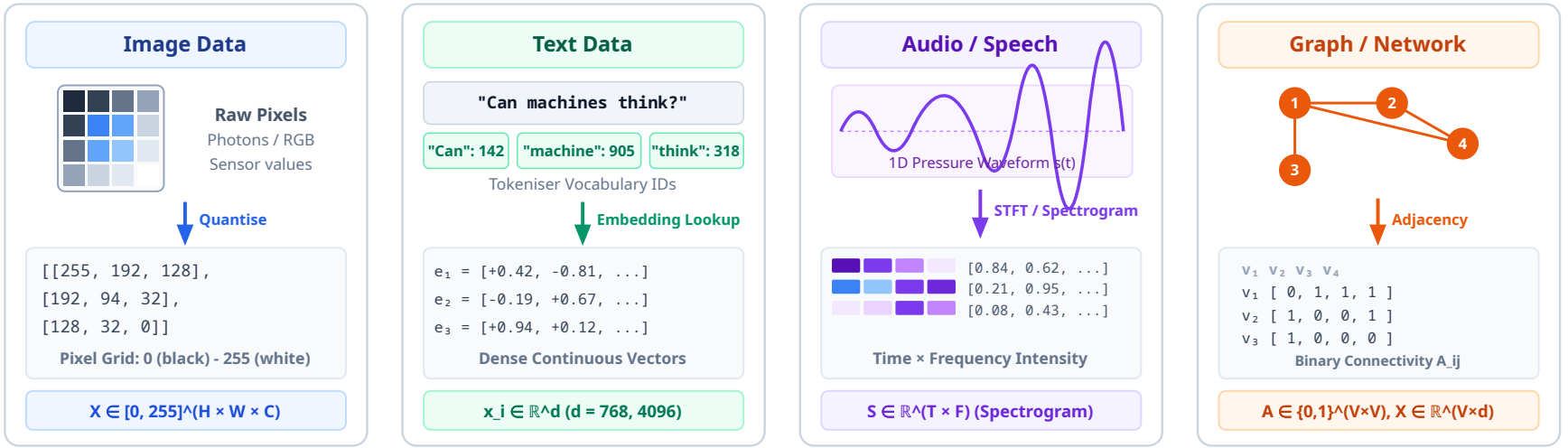
Data is everything.

- **Numerical Data**: tables, vectors, tensors, audio waveforms, pixel values, sensor readings
- **Text Data**: natural language documents, sentences, words, tokens
- **Image Data**: photographs, medical scans, satellite imagery
- **Graph Data**: social networks, molecule structures, citation networks
- **Time Series Data**: stock prices, sensor readings, weather measurements
- **Video Data**: sequences of images with temporal dependencies
- **Tabular Data**: rows and columns, spreadsheets, CSV files
- **Data Data**: data and more data, internet scale data

Everything becomes a number

All data at the end becomes a number a machine can process:

- Images are turned into pixels with values between 0 and 255 (or other values)
- Text are turned into tokens which are turned into embeddings (vectors of numbers)
- Audio are turned into spectrograms or other numerical representations
- Graphs are turned into adjacency matrices or other numerical representations
- etc.



Two Types of Data

We mainly worry about two types of data:

- **Continuous Data**: real numbers, real-valued data that can take any value within a given range. These are things we usually *measure*.
  - e.g. height, weight, temperature, price, stock value, sensor stuff
- **Discrete Data**: integer values, data that can only take specific values. These are things we usually *count*.
  - e.g. tokens, word counts, user IDs, ratings (1-5), on/off states

> By applying thresholds one can turn continuous data into discrete data, e.g. age groups.

MNIST



Modified National Institute of Standards and Technology database (CC BY-SA 4.0, Josef Steppan)

Data Addiction

Academic Benchmarks

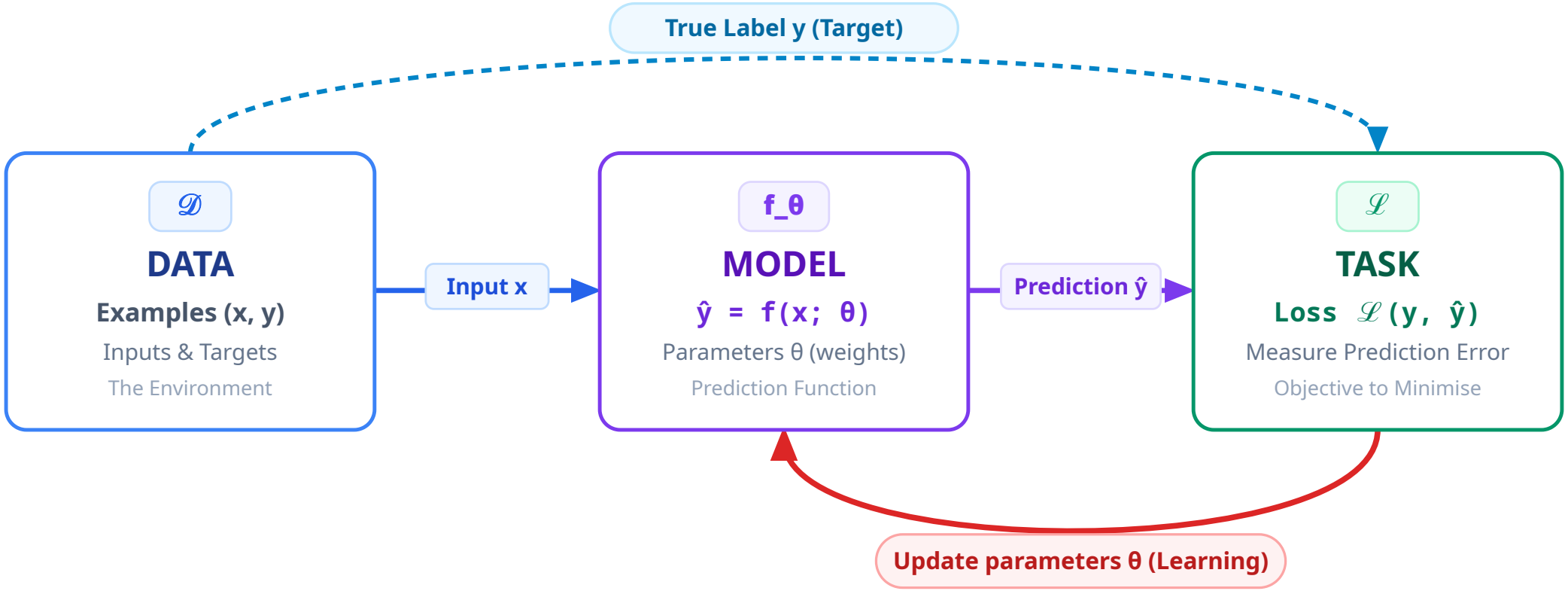
- **MNIST (1998, Vision):** $70{,}000$ greyscale digits (28×28) ~ 50 MB
- **CIFAR-10 (2009, Vision):** $60{,}000$ colour images (32×32) ~ 160 MB
- **ImageNet (2012, Vision):** 1.4M images across $1{,}000$ classes ~ 150 GB
- **LibriSpeech (2015, Audio):** $1{,}000$ hrs English speech ~ 60 GB

Web-Scale & Foundation Era

- **English Wikipedia (Text):** High-quality text dump (4.5B+ words) ~ 22 GB
- **The Pile (2020, Text/Code):** Curated diverse corpus ~ 825 GB
- **LAION-5B (2022, Multimodal):** 5.85B image-text pairs ~ 240 TB
- **Common Crawl (Web):** Petabyte-scale monthly crawls $\sim 10+$ PB (15T+ tokens)

> From a floppy disk (50 MB) to the entire open web ($10+$ PB), data hunger has grown by over $200{,}000{,}000\times$!

This is it



We will explore this landscape in the following weeks. So stay tuned to this radio channel!