

# Neural Networks: Old and New

---

**Ju Sun**

Computer Science & Engineering  
University of Minnesota, Twin Cities

January 23, 2025

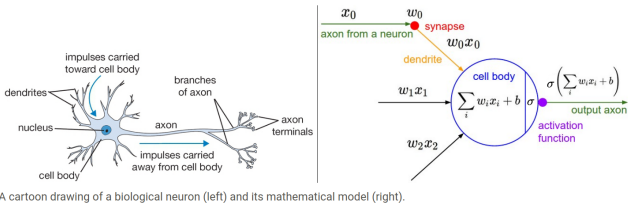
Start from neurons

Shallow to deep neural networks

A brief history of AI

Suggested reading

# Model of biological neurons

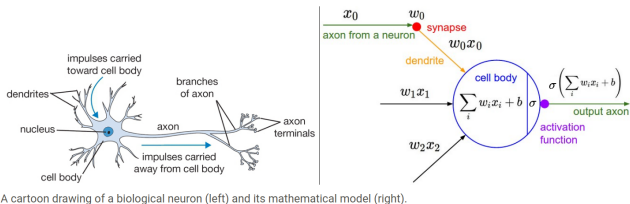


Credit: Stanford CS231N

Biologically ...

- Each neuron receives signals from its **dendrites**
- Each neuron outputs signals via its single **axon**
- The axon branches out and connects via **synapse** to dendrites of other neurons

# Model of biological neurons



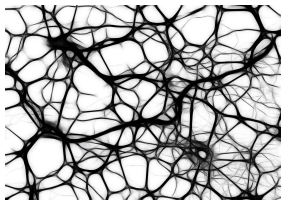
Credit: Stanford CS231N

Mathematically ...

- Each neuron receives  $x_i$ 's from its **dendrites**
- $x_i$ 's weighted by  $w_i$ 's (synaptic strengths) and summed  $\sum_i w_i x_i$
- The neuron fires only when the combined signal is above a certain threshold:  $\sum_i w_i x_i + b$
- Fire rate is modeled by an **activation function**  $\sigma$ , i.e., outputting  $\sigma(\sum_i w_i x_i + b)$

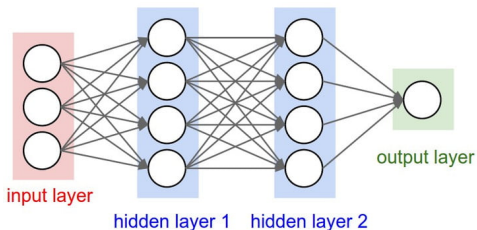
# Artificial neural networks

## Brain neural networks



~ 86-billion neurons (Credit: Max Pixel)

## Artificial neural networks



## Why called **artificial**?

- (Over-)simplification on neural level
- (Over-)simplification on connection level

In this course, neural networks are always artificial.

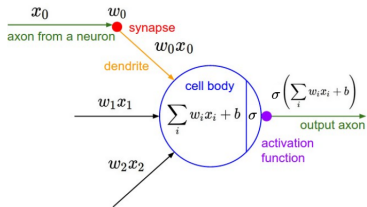
Start from neurons

Shallow to deep neural networks

A brief history of AI

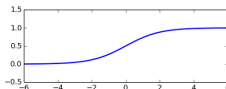
Suggested reading

# Artificial neurons



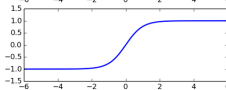
$$\sigma \left( \sum_i w_i x_i + b \right) = \sigma (\mathbf{w}^T \mathbf{x} + b)$$

## Examples of activation function $\sigma$



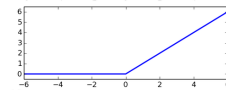
Sigmoid

$$\phi(z) = \frac{1}{1 + e^{-z}}$$



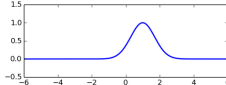
Hyperbolic Tangent

$$\phi(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$$



Rectified Linear

$$\phi(z) = \begin{cases} 0 & \text{if } z < 0 \\ z & \text{if } z \geq 0 \end{cases}$$



Radial Basis Function

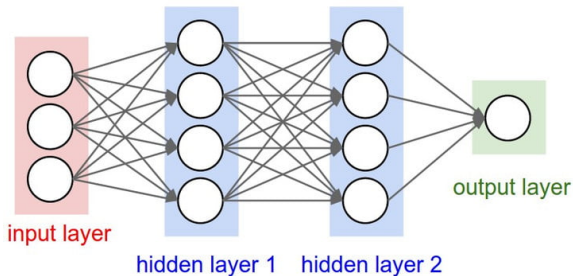
$$\phi(z, c) = e^{-c(\|z - c\|)^2}$$

Credit: [Hughes and Correll, 2016]

# Neural networks

One neuron:  $\sigma(\mathbf{w}^\top \mathbf{x} + b)$

Neural networks (NN): **structured** organization of artificial neurons



$w$ 's and  $b$ 's are unknown and need to be learned

Many models in machine learning **are** neural networks



## Supervised Learning

- Gather training data  $(\mathbf{x}_1, \mathbf{y}_1), \dots, (\mathbf{x}_n, \mathbf{y}_n)$
- Choose a family of functions, e.g.,  $\mathcal{H}$ , so that there is  $f \in \mathcal{H}$  to ensure  $\mathbf{y}_i \approx f(\mathbf{x}_i)$  for all  $i$
- Set up a loss function  $\ell$  to measure the approximation quality
- Find an  $f \in \mathcal{H}$  to minimize the average loss

$$\min_{f \in \mathcal{H}} \frac{1}{n} \sum_{i=1}^n \ell(\mathbf{y}_i, f(\mathbf{x}_i))$$

... known as **empirical risk minimization** (ERM) framework in learning theory

## Supervised Learning from NN viewpoint

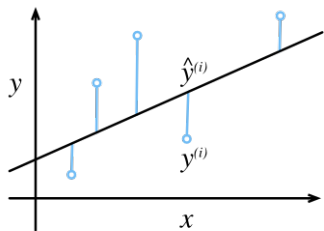
- Gather training data  $(\mathbf{x}_1, \mathbf{y}_1), \dots, (\mathbf{x}_n, \mathbf{y}_n)$
- Choose a NN with  $k$  neurons, so that there is a group of weights, e.g.,  $(\mathbf{w}_1, \dots, \mathbf{w}_k, b_1, \dots, b_k)$ , to ensure

$$\mathbf{y}_i \approx \{\text{NN}(\mathbf{w}_1, \dots, \mathbf{w}_k, b_1, \dots, b_k)\}(\mathbf{x}_i) \quad \forall i$$

- Set up a loss function  $\ell$  to measure the approximation quality
- Find weights  $(\mathbf{w}_1, \dots, \mathbf{w}_k, b_1, \dots, b_k)$  to minimize the average loss

$$\min_{\mathbf{w}'s, b's} \frac{1}{n} \sum_{i=1}^n \ell[\mathbf{y}_i, \{\text{NN}(\mathbf{w}_1, \dots, \mathbf{w}_k, b_1, \dots, b_k)\}(\mathbf{x}_i)]$$

# Linear regression



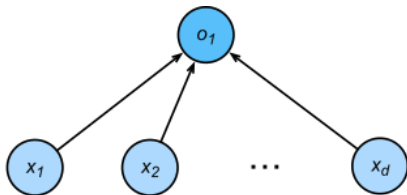
Credit: D2L

- Data:  $(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_n, y_n), \mathbf{x}_i \in \mathbb{R}^d$
- Model:  $y_i \approx \mathbf{w}^\top \mathbf{x}_i + b$
- Loss:  $\|y - \hat{y}\|_2^2$
- Optimization:

$$\min_{\mathbf{w}, b} \frac{1}{n} \sum_{i=1}^n \|y_i - (\mathbf{w}^\top \mathbf{x}_i + b)\|_2^2$$

Output layer

Input layer



Credit: D2L

$\sigma$  is the identity function

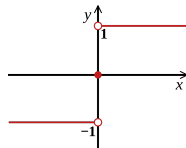
# Perceptron



**Frank Rosenblatt**

(1928–1971)

- Data:  $(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_n, y_n)$ ,  
 $\mathbf{x}_i \in \mathbb{R}^d$ ,  $y_i \in \{+1, -1\}$
- Model:  $y_i \approx \sigma(\mathbf{w}^\top \mathbf{x}_i + b)$ ,  $\sigma$  sign function

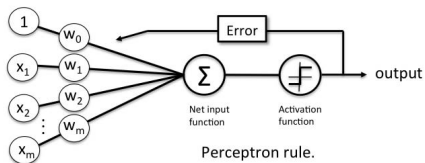


- Loss:  $\mathbf{1}\{y \neq \hat{y}\}$
- Optimization:

$$\min_{\mathbf{w}, b} \frac{1}{n} \sum_{i=1}^n \mathbf{1}\{y_i \neq \sigma(\mathbf{w}^\top \mathbf{x}_i + b)\}$$

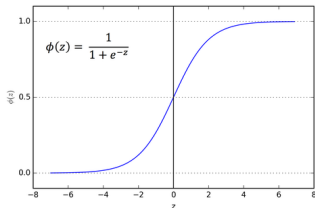
# Perceptron

Perceptron is a single artificial neuron for **binary classification**



dominated early AI (50's – 60's)

**Logistic regression** is similar but with **sigmoid** activation



# Softmax regression

- Data:  $(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_n, y_n)$ ,  $\mathbf{x}_i \in \mathbb{R}^d$ ,  $y_i \in \{L_1, \dots, L_p\}$ , i.e., multiclass classification problem
- Data preprocessing: labels into vectors via **one-hot encoding**

$$L_k \implies \underbrace{[0, \dots, 0]}_{k-1 \text{ 0's}}, 1, \underbrace{[0, \dots, 0]}_{p-k \text{ 0's}}^\top$$

So:  $y_i \implies \mathbf{y}_i$

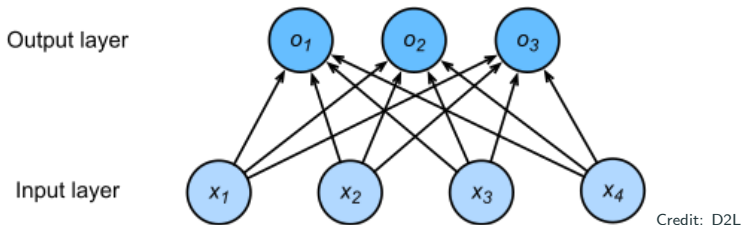
- Model:  $\mathbf{y}_i \approx \sigma(\mathbf{W}^\top \mathbf{x}_i + \mathbf{b})$ , here  $\sigma$  is the softmax function (**maps vectors to vectors**): for  $\mathbf{z} \in \mathbb{R}^p$ ,

$$\mathbf{z} \mapsto \left[ \frac{e^{z_1}}{\sum_j e^{z_j}}, \dots, \frac{e^{z_p}}{\sum_j e^{z_j}} \right]^\top.$$

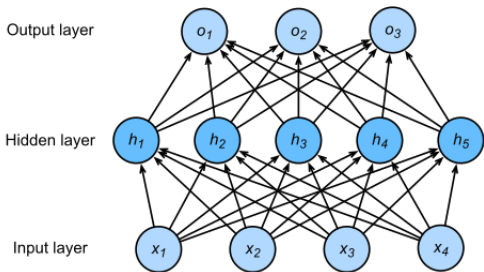
- Loss: **cross-entropy loss**  $-\sum_j y_j \log \hat{y}_j$
- Optimization ...

# Softmax regression

... for multiclass classification



# Multilayer perceptrons (MLP)



Credit: D2L

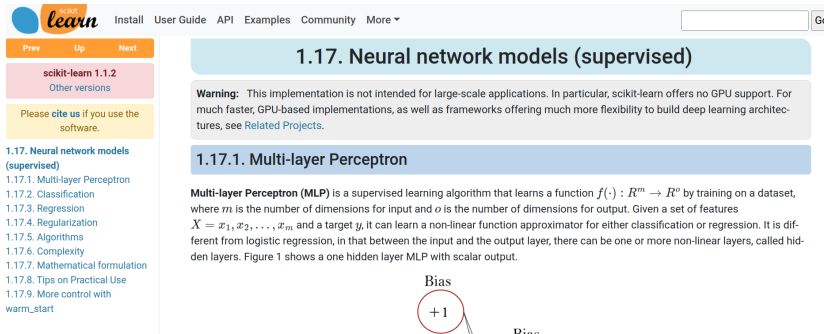
$$\text{Model: } \mathbf{y}_i \approx \sigma_2(\mathbf{W}_2^T \sigma_1(\mathbf{W}_1^T \mathbf{x} + \mathbf{b}_1) + \mathbf{b}_2)$$

Also called **fully-connected networks**

Modern NNs:

- many hidden layers: deep neural networks (DNNs)
- refined/structured connection and/or activations (convolutional/recurrent/graph/... NNs)





The screenshot shows the scikit-learn documentation interface. At the top left is the scikit-learn logo. To its right are navigation links: Install, User Guide, API, Examples, Community, and More. A search bar is on the far right. Below the navigation is a sidebar with a 'Prev Up Next' menu, the current page title 'scikit-learn 1.1.2', and a note to cite the software. The main content area has a light blue header for '1.17. Neural network models (supervised)'. Below this is a 'Warning' box stating that the implementation is not for large-scale applications and lacks GPU support. The sub-section '1.17.1. Multi-layer Perceptron' is highlighted. The text defines an MLP as a supervised learning algorithm that learns a function  $f(\cdot) : R^m \rightarrow R^o$ . It notes that  $m$  is the number of input dimensions and  $o$  is the number of output dimensions. It describes the input as a set of features  $X = x_1, x_2, \dots, x_m$  and a target  $y$ , and explains that the model can learn a non-linear function approximator. It distinguishes MLP from logistic regression by noting that MLP can have one or more non-linear hidden layers. A diagram of a single hidden layer MLP with scalar output is mentioned, with a callout showing a bias node labeled '+1' circled in red and labeled 'Bias'.

[https://scikit-learn.org/stable/modules/neural\\_networks\\_supervised.html](https://scikit-learn.org/stable/modules/neural_networks_supervised.html)

## They're all (shallow) NNs

- Linear regression
- Perception and Logistic regression
- Softmax regression
- Multilayer perceptron (feedforward NNs)
- Support vector machines (SVM)
- PCA (autoencoder)
- Matrix factorization

see, e.g., Chapter 2 of [[Aggarwal, 2018](#)].

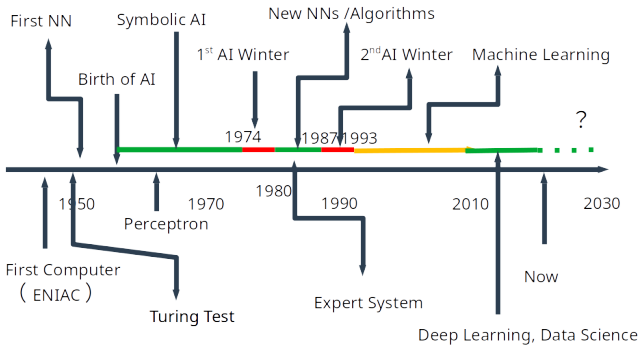
Start from neurons

Shallow to deep neural networks

A brief history of AI

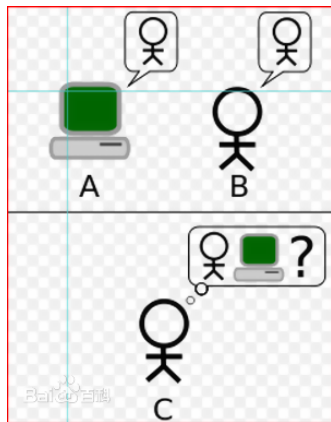
Suggested reading

# Birth of AI

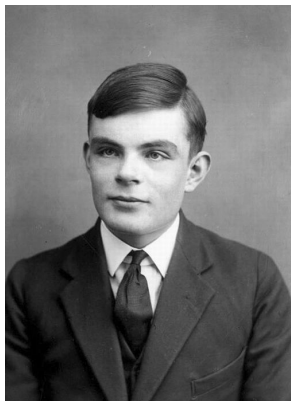


- Crucial precursors: first computer, Turing test
- 1956: Dartmouth Artificial Intelligence Summer Research Project — Birth of AI

# Turing test

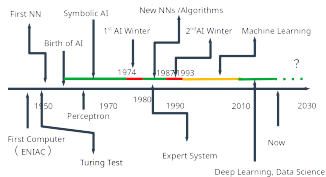


Turing Test

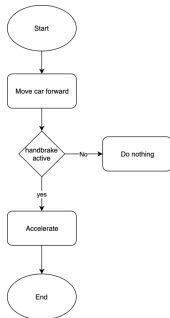


Alan Turing (1912–1954)

# First golden age

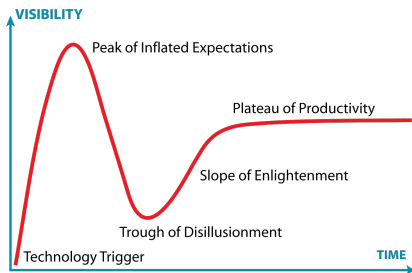
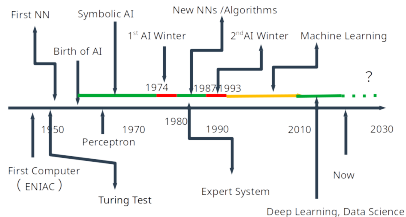


## Symbolic AI: modeling general logic and reasoning



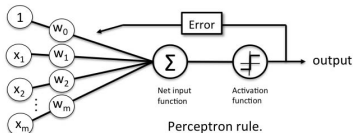
rules for recognizing dogs?

# First AI winter

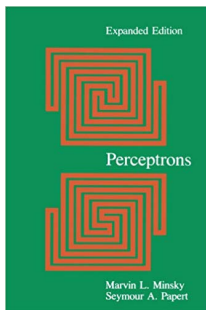


Gartner hype cycle

# Perceptron



invented 1962



written in 1969, end of  
Perceptron era



Marvin Minsky (1927–2016)



# Birth of computer vision

MASSACHUSETTS INSTITUTE OF TECHNOLOGY  
PROJECT MAC

Artificial Intelligence Group  
Vision Memo. No. 100.

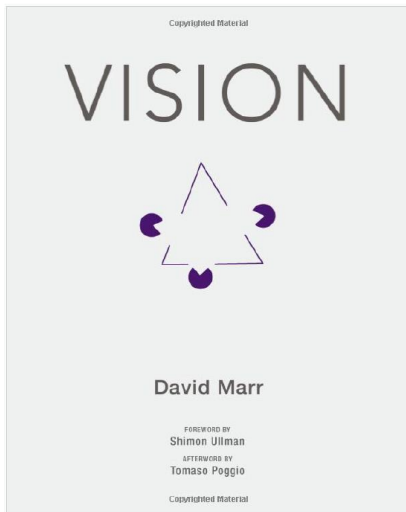
July 7, 1966

## THE SUMMER VISION PROJECT

Seymour Papert

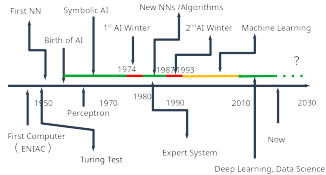
The summer vision project is an attempt to use our summer workers effectively in the construction of a significant part of a visual system. The particular task was chosen partly because it can be segmented into sub-problems which will allow individuals to work independently and yet participate in the construction of a system complex enough to be a real landmark in the development of "pattern recognition".

1966

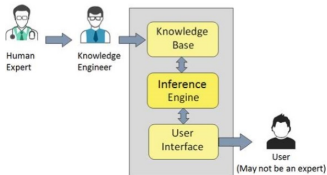


around 1980

# Second golden age



## expert system—building in domain-specific knowledge

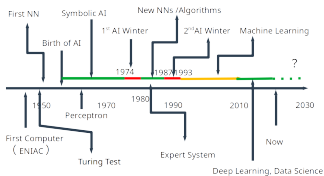


Can we build comprehensive knowledge bases and know all rules?

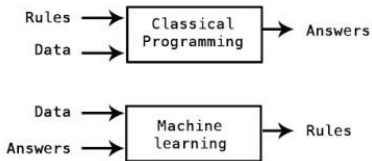
Key ingredients of DL have been in place for 25-30 years:

Landmark	Emblem	Epoch
Neocognitron	Fukushima	1980
CNN	Le Cun	mid 1980s'
Backprop	Hinton	mid 1980's
SGD	Le Cun, Bengio etc	mid 1990's
Various	Schmidhuber	mid 1980's
<i>CTF</i>	<i>DARPA etc</i>	<i>mid 1980's</i>

# After 2nd AI winter



Machine learning takes over ...



rules learned from data, or **data-driven**

Starting 1990's

Support vector machines (SVM)

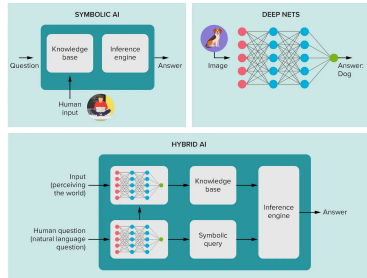
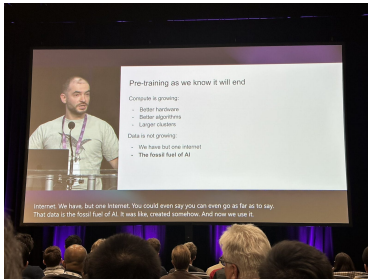
Adaboost

Decision trees and random forests

Deep learning (2010's)

...

# What's next?



Start from neurons

Shallow to deep neural networks

A brief history of AI

Suggested reading

- Chap 2, Neural Networks and Deep Learning.
- Chap 3–4, Dive into Deep Learning.
- Chap 1, Deep Learning with Python.



- [Aggarwal, 2018] Aggarwal, C. C. (2018). **Neural Networks and Deep Learning**. Springer International Publishing.
- [Hughes and Correll, 2016] Hughes, D. and Correll, N. (2016). **Distributed machine learning in materials that couple sensing, actuation, computation and communication**. *arXiv:1606.03508*.