Neural Networks: Old and New

Ju Sun
Computer Science & Engineering
University of Minnesota, Twin Cities

September 14, 2020

Outline

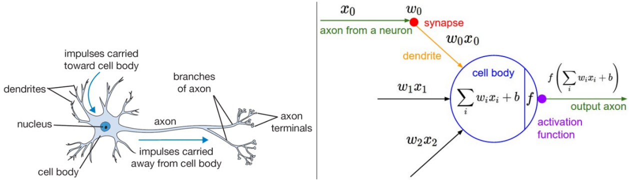
Start from neurons

Shallow to deep neural networks

A brief history of AI

Suggested reading

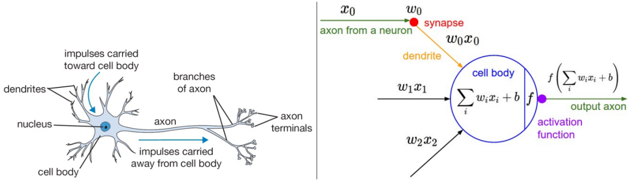
Model of biological neurons



A cartoon drawing of a biological neuron (left) and its mathematical model (right).

Credit: Stanford CS231N

Model of biological neurons



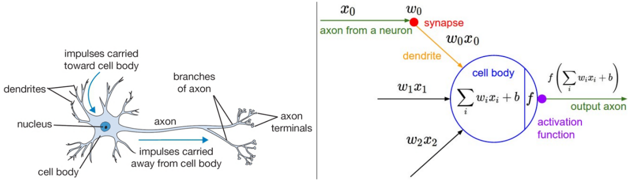
A cartoon drawing of a biological neuron (left) and its mathematical model (right).

Credit: Stanford CS231N

Biologically ...

- Each neuron receives signals from its **dendrites**

Model of biological neurons



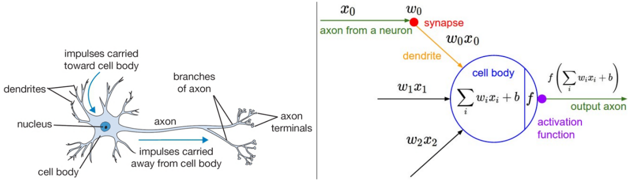
A cartoon drawing of a biological neuron (left) and its mathematical model (right).

Credit: Stanford CS231N

Biologically ...

- Each neuron receives signals from its **dendrites**
- Each neuron outputs signals via its single **axon**

Model of biological neurons



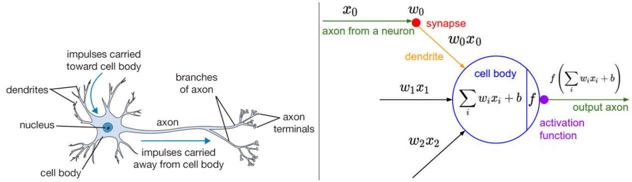
A cartoon drawing of a biological neuron (left) and its mathematical model (right).

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 **synapese** to dendrites of other neurons

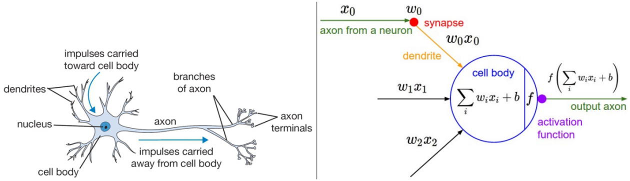
Model of biological neurons



A cartoon drawing of a biological neuron (left) and its mathematical model (right).

Credit: Stanford CS231N

Model of biological neurons



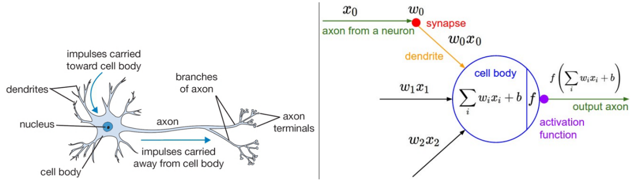
A cartoon drawing of a biological neuron (left) and its mathematical model (right).

Credit: Stanford CS231N

Mathematically ...

- Each neuron receives x_i's from its **dendrites**

Model of biological neurons



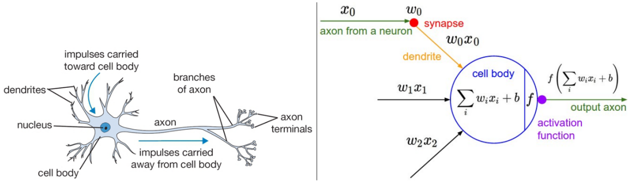
A cartoon drawing of a biological neuron (left) and its mathematical model (right).

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 $\sum_i w_i x_i$

Model of biological neurons



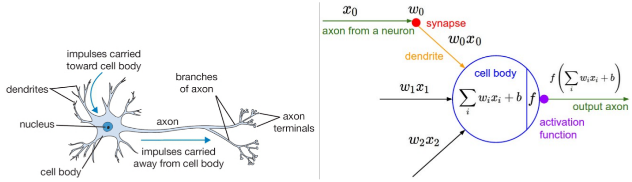
A cartoon drawing of a biological neuron (left) and its mathematical model (right).

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 $\sum_i w_ix_i$
- The neuron fires only when the combined signal is above a certain threshold: $\sum_i w_ix_i + b$

Model of biological neurons



A cartoon drawing of a biological neuron (left) and its mathematical model (right).

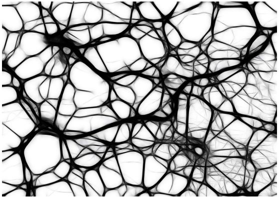
Credit: Stanford CS231N

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** f, i.e., outputting $f\left(\sum_i w_i x_i + b\right)$

Artificial neural networks

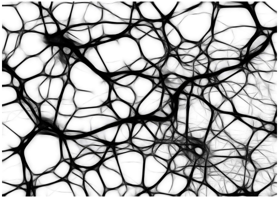
Brain neural networks



Credit: Max Pixel

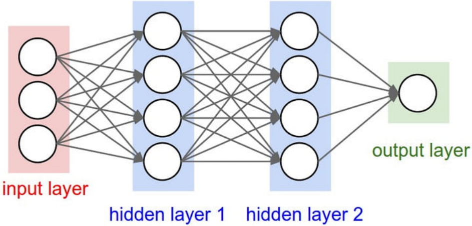
Artificial neural networks

Brain neural networks



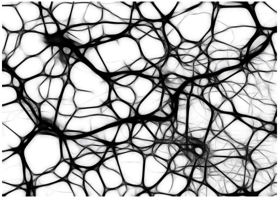
Credit: Max Pixel

Artificial neural networks



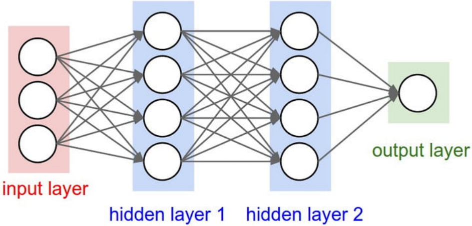
Artificial neural networks

Brain neural networks



Credit: Max Pixel

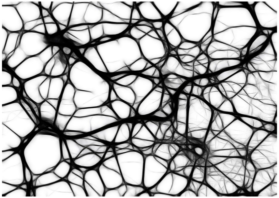
Artificial neural networks



Why called **artificial**?

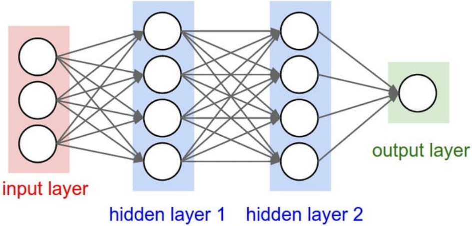
Artificial neural networks

Brain neural networks



Credit: Max Pixel

Artificial neural networks

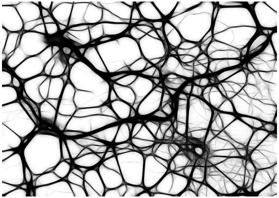


Why called **artificial**?

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

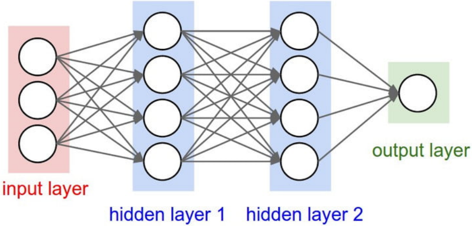
Artificial neural networks

Brain neural networks



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.

Outline

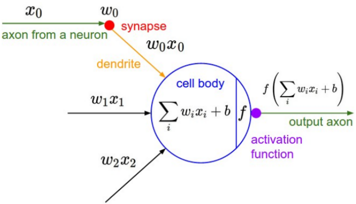
Start from neurons

Shallow to deep neural networks

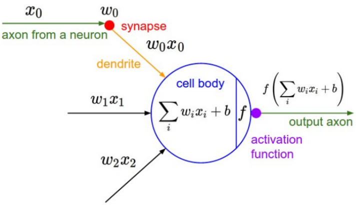
A brief history of AI

Suggested reading

Artificial neurons

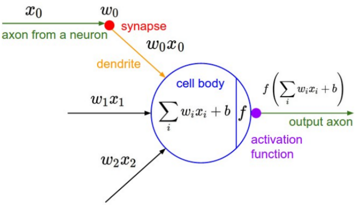


Artificial neurons



$$f\left(\sum_i w_i x_i + b\right) = f\left(\boldsymbol{w}^\intercal \boldsymbol{x} + b\right)$$

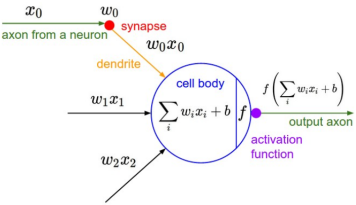
Artificial neurons



$$f\left(\sum_i w_i x_i + b\right) = f\left(\boldsymbol{w}^\intercal \boldsymbol{x} + b\right)$$

We shall use σ instead of f henceforth.

Artificial neurons

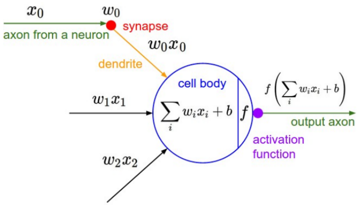


$$f\left(\sum_i w_i x_i + b\right) = f\left(\boldsymbol{w}^\intercal \boldsymbol{x} + b\right)$$

We shall use σ instead of f henceforth.

Examples of activation function σ

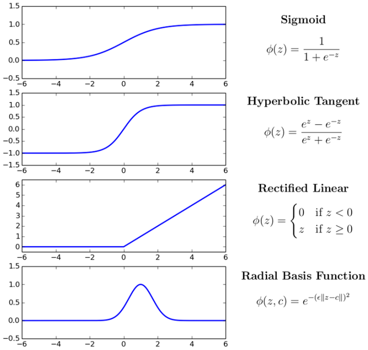
Artificial neurons



$$f\left(\sum_i w_i x_i + b\right) = f\left(\boldsymbol{w}^\top \boldsymbol{x} + b\right)$$

We shall use σ instead of f henceforth.

Examples of activation function σ



Credit: [Hughes and Correll, 2016]

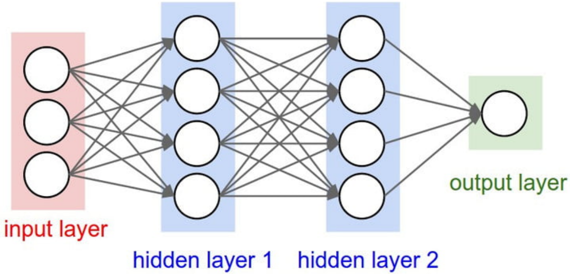
Neural networks

One neuron: $\sigma\left(\boldsymbol{w}^{\intercal}\boldsymbol{x}+b\right)$

Neural networks

One neuron: $\sigma\left(\boldsymbol{w}^{\intercal}\boldsymbol{x}+b\right)$

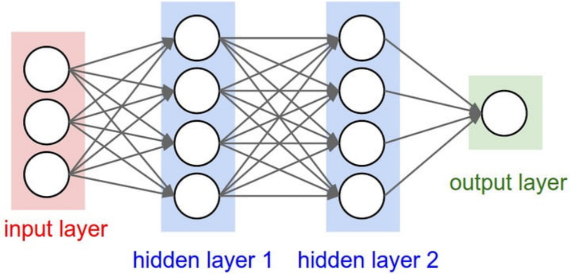
Neural networks (NN): **structured** organization of artificial neurons



Neural networks

One neuron: $\sigma\left(\boldsymbol{w}^{\intercal}\boldsymbol{x}+b\right)$

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

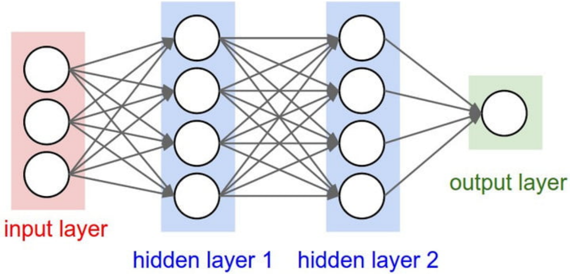


$\boldsymbol{w}$'s and b's are unknown and need to be learned

Neural networks

One neuron: $\sigma\left(\boldsymbol{w}^\intercal \boldsymbol{x} + b\right)$

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



$\boldsymbol{w}$'s and b's are unknown and need to be learned

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

A typical setup

Supervised Learning

– Gather training data $(\boldsymbol{x}_1, \boldsymbol{y}_1), \ldots, (\boldsymbol{x}_n, \boldsymbol{y}_n)$

A typical setup

Supervised Learning

- Gather training data $(\boldsymbol{x}_1, \boldsymbol{y}_1), \ldots, (\boldsymbol{x}_n, \boldsymbol{y}_n)$
- Choose a family of functions, e.g., $\mathcal{H}$, so that there is $f \in \mathcal{H}$ to ensure $\boldsymbol{y}_i \approx f(\boldsymbol{x}_i)$ for all i

A typical setup

Supervised Learning

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

A typical setup

Supervised Learning

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

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

A typical setup

Supervised Learning

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

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

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

A typical setup

Supervised Learning from NN viewpoint

- Gather training data $(\boldsymbol{x}_1, \boldsymbol{y}_1), \ldots, (\boldsymbol{x}_n, \boldsymbol{y}_n)$

A typical setup

Supervised Learning from NN viewpoint

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

$$\boldsymbol{y}_i \approx \{\mathsf{NN}(\boldsymbol{w}_1, \ldots, \boldsymbol{w}_k, b_1, \ldots, b_k)\}(\boldsymbol{x}_i) \quad \forall i$$

A typical setup

Supervised Learning from NN viewpoint

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

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

- Set up a loss function ℓ to measure the approximation quality

A typical setup

Supervised Learning from NN viewpoint

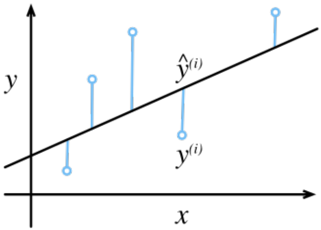
- Gather training data $(\boldsymbol{x}_1, \boldsymbol{y}_1), \dots, (\boldsymbol{x}_n, \boldsymbol{y}_n)$
- Choose a NN with k neurons, so that there is a group of weights, e.g., $(\boldsymbol{w}_1, \dots, \boldsymbol{w}_k, b_1, \dots, b_k)$, to ensure

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

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

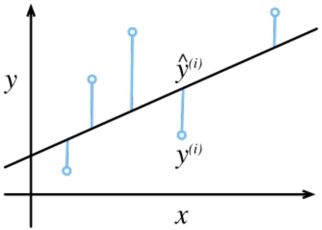
$$\min_{\boldsymbol{w}'s, b's} \frac{1}{n} \sum_{i=1}^{n} \ell\left[\boldsymbol{y}_i, \{\mathsf{NN}(\boldsymbol{w}_1, \dots, \boldsymbol{w}_k, b_1, \dots, b_k)\}(\boldsymbol{x}_i)\right]$$

Linear regression



Credit: D2L

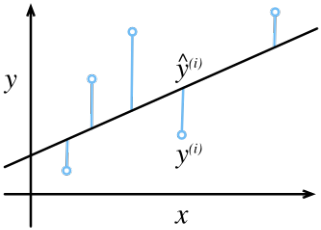
Linear regression



Credit: D2L

– Data: $(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_n, y_n)$, $\boldsymbol{x}_i \in \mathbb{R}^d$

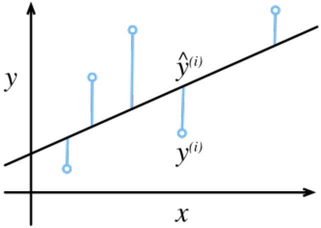
Linear regression



Credit: D2L

- Data: $(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_n, y_n)$, $\boldsymbol{x}_i \in \mathbb{R}^d$
- Model: $y_i \approx \boldsymbol{w}^\intercal \boldsymbol{x}_i + b$

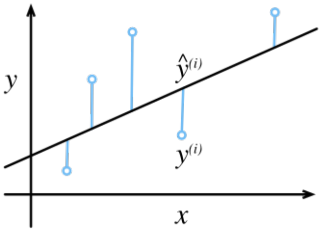
Linear regression



Credit: D2L

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

Linear regression

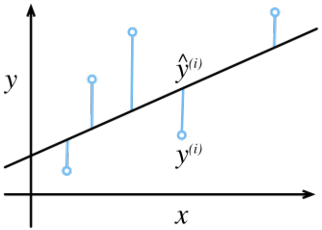


Credit: D2L

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

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

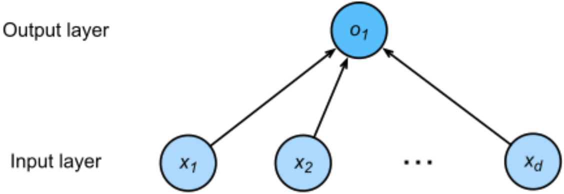
Linear regression



Credit: D2L

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

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



Credit: D2L

σ is the identity function

Perceptron

Frank Rosenblatt

(1928–1971)

Perceptron

Frank Rosenblatt

(1928–1971)

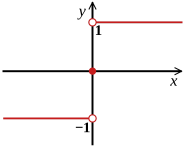
- Data: $(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_n, y_n)$, $\boldsymbol{x}_i \in \mathbb{R}^d$, $y_i \in \{+1, -1\}$

Perceptron

Frank Rosenblatt

(1928–1971)

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

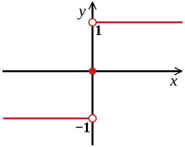


Perceptron

Frank Rosenblatt

(1928–1971)

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



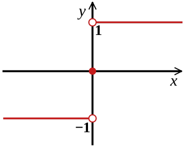
- Loss: $\mathbf{1}\{y \neq \hat{y}\}$

Perceptron

Frank Rosenblatt

(1928–1971)

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

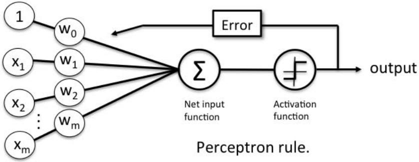


- Loss: $\mathbf{1}\{y \neq \hat{y}\}$
- Optimization:

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

Perceptron

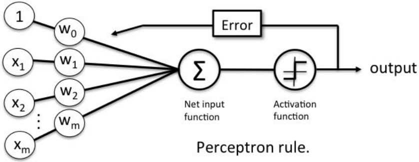
Perceptron is a single artificial neuron for **binary classification**



Perceptron rule.

Perceptron

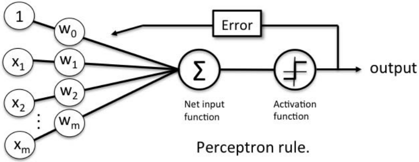
Perceptron is a single artificial neuron for **binary classification**



dominated early AI (50's – 70's)

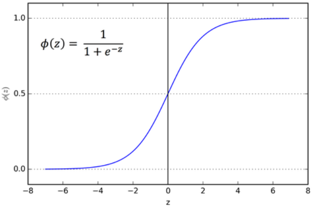
Perceptron

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



dominated early AI (50's – 70's)

Logistic regression is similar but with **sigmod** activiation



Softmax regression

- Data: $(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_n, y_n)$, $\boldsymbol{x}_i \in \mathbb{R}^d$, $y_i \in \{L_1, \ldots, L_p\}$, i.e., multiclass classification problem

Softmax regression

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

$$L_k \Longrightarrow [\underbrace{0, \ldots, 0,}_{k-1\,0's}, 1, \underbrace{0, \ldots, 0}_{n-k\,0's}]^\intercal$$

So: $y_i \Longrightarrow \boldsymbol{y}_i$

Softmax regression

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

$$L_k \Longrightarrow [\underbrace{0, \ldots, 0,}_{k-1\, 0's}, 1, \underbrace{0, \ldots, 0}_{n-k\, 0's}]^\intercal$$

So: $y_i \Longrightarrow \boldsymbol{y}_i$

- Model: $\boldsymbol{y}_i \approx \sigma\left(\boldsymbol{W}^\intercal \boldsymbol{x}_i + \boldsymbol{b}\right)$, here σ is the softmax function

Softmax regression

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

$$L_k \Longrightarrow [\underbrace{0, \ldots, 0,}_{k-1\ 0's}, 1, \underbrace{0, \ldots, 0}_{n-k\ 0's}]^\intercal$$

So: $y_i \Longrightarrow \boldsymbol{y}_i$

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

$$\boldsymbol{z} \mapsto \left[\frac{e^{z_1}}{\sum_j e^{z_j}}, \ldots, \frac{e^{z_p}}{\sum_j e^{z_j}}\right]^\intercal.$$

Softmax regression

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

$$L_k \Longrightarrow [\underbrace{0, \ldots, 0,}_{k-1\,0's}, 1, \underbrace{0, \ldots, 0}_{n-k\,0's}]^\intercal$$

So: $y_i \Longrightarrow \boldsymbol{y}_i$

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

$$\boldsymbol{z} \mapsto \left[\frac{e^{z_1}}{\sum_j e^{z_j}}, \ldots, \frac{e^{z_p}}{\sum_j e^{z_j}}\right]^\intercal .$$

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

Softmax regression

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

$$L_k \Longrightarrow [\underbrace{0, \dots, 0,}_{k-1\, 0's}, 1, \underbrace{0, \dots, 0}_{n-k\, 0's}]^\intercal$$

  So: $y_i \Longrightarrow \boldsymbol{y}_i$

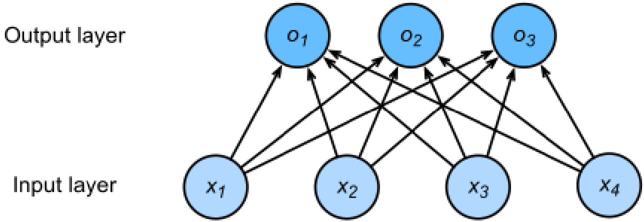
- Model: $\boldsymbol{y}_i \approx \sigma\left(\boldsymbol{W}^\intercal \boldsymbol{x}_i + \boldsymbol{b}\right)$, here σ is the softmax function (maps vectors to vectors): for $\boldsymbol{z} \in \mathbb{R}^p$,

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

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

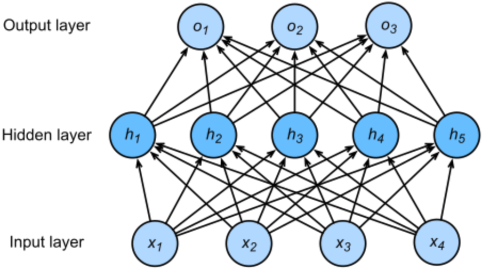
Softmax regression

… for multiclass classification



Credit: D2L

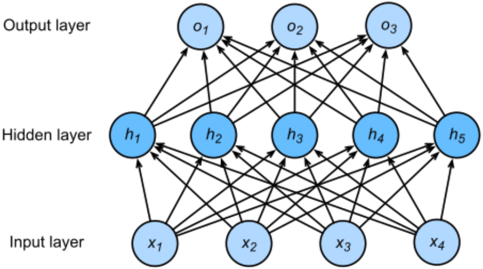
Multilayer perceptrons



Credit: D2L

Model: $\boldsymbol{y}_i \approx \sigma_2\left(\boldsymbol{W}_2^\intercal \sigma_1(\boldsymbol{W}_1^\intercal \boldsymbol{x} + \boldsymbol{b}_1) + \boldsymbol{b}_2\right)$

Multilayer perceptrons

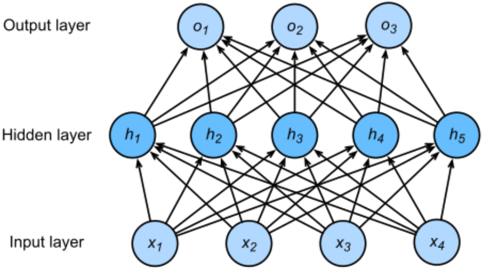


Credit: D2L

Model: $\boldsymbol{y}_i \approx \sigma_2 \left(\boldsymbol{W}_2^\intercal \sigma_1 (\boldsymbol{W}_1^\intercal \boldsymbol{x} + \boldsymbol{b}_1) + \boldsymbol{b}_2 \right)$

Also called **feedforward networks** or **fully-connected networks**

Multilayer perceptrons



Credit: D2L

Model: $\boldsymbol{y}_i \approx \sigma_2\left(\boldsymbol{W}_2^\intercal \sigma_1(\boldsymbol{W}_1^\intercal \boldsymbol{x} + \boldsymbol{b}_1) + \boldsymbol{b}_2\right)$

Also called **feedforward networks** or **fully-connected networks**

Modern NNs: many hidden layers (deep), refined connection structure and/or activations

They're all (shallow) NNs

- Linear regression
- Perception and Logistic regression
- Softmax regression
- Multilayer perceptron (feedforward NNs)

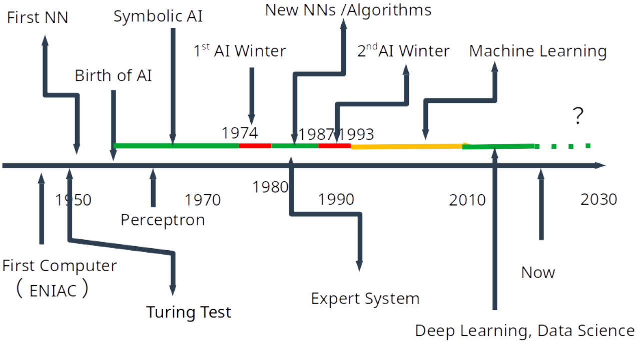
They're all (shallow) NNs

- 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].

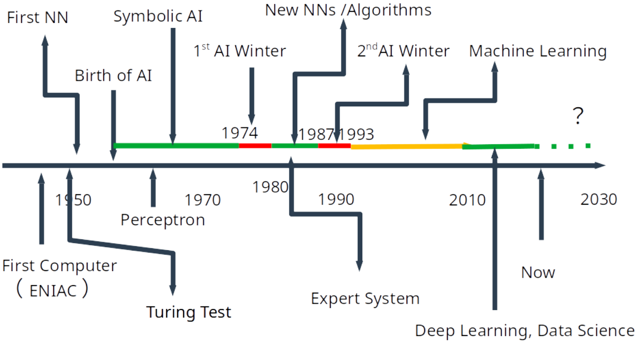
Outline

Birth of AI



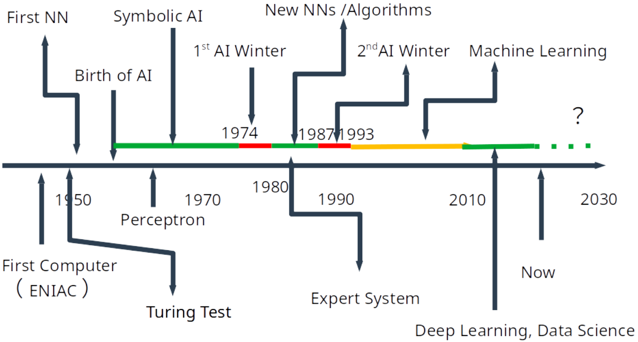
First NN
Symbolic AI
New NNs /Algorithms
1st AI Winter
2nd AI Winter
Machine Learning
Birth of AI
?
1974
1987
1993
1950
1970
1980
1990
2010
2030
Perceptron
First Computer
(ENIAC)
Turing Test
Expert System
Now
Deep Learning, Data Science

Birth of AI



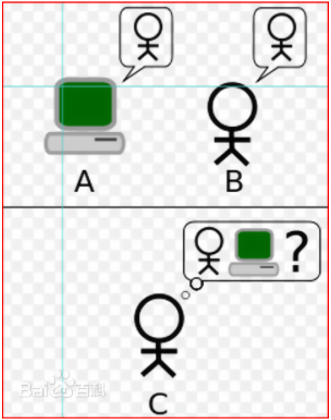
– Crucial precursors: first computer, Turing test

Birth of AI

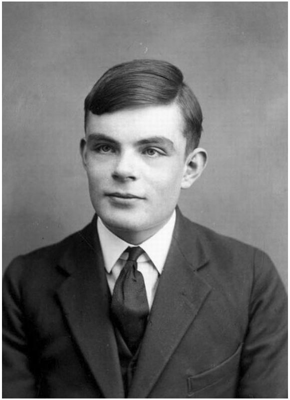


- Crucial precursors: first computer, Turing test
- 1956: Dartmouth Artificial Intelligence Summer Research Project — Birth of AI

Turing test

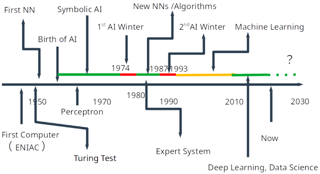


Turing Test

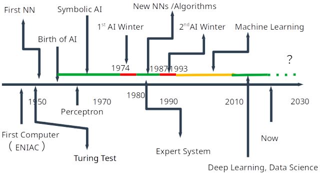


Alan Turing (1912–1954)

First golden age

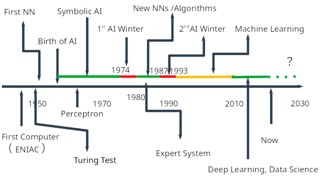


First golden age

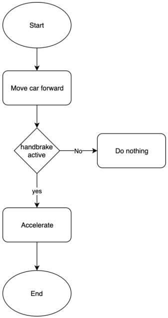


Symbolic AI: based on rules and logic

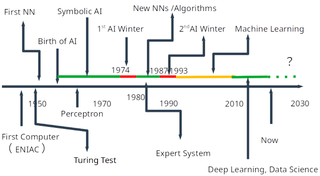
First golden age



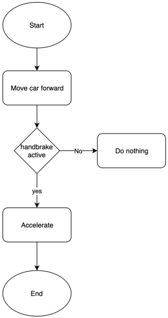
Symbolic AI: based on rules and logic



First golden age

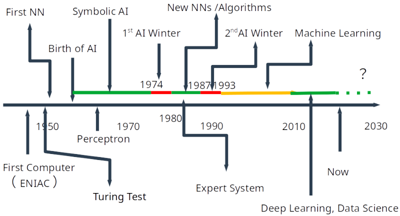


Symbolic AI: based on rules and logic

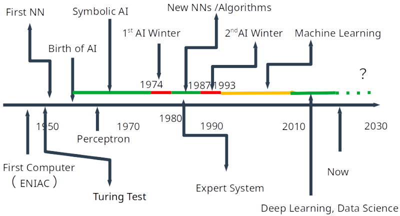


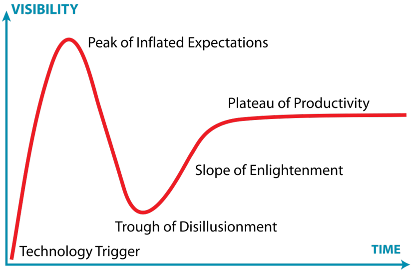
rules for recognizing dogs?

First AI winter



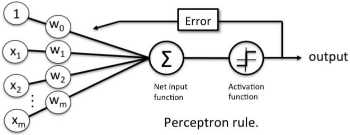
First AI winter





Gartner hype cycle

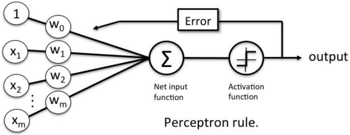
Perceptron



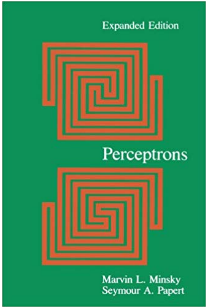
Perceptron rule.

invented 1962

Perceptron



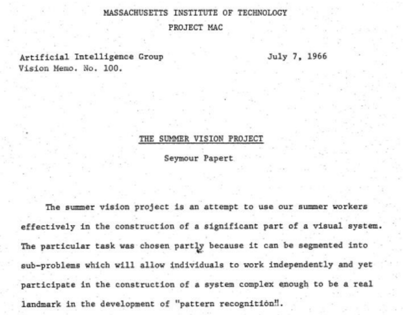
invented 1962



written in 1969, end of Perceptron era

Marvin Minsky (1927–2016)

Birth of computer vision


MASSACHUSETTS INSTITUTE OF TECHNOLOGY
PROJECT MAC

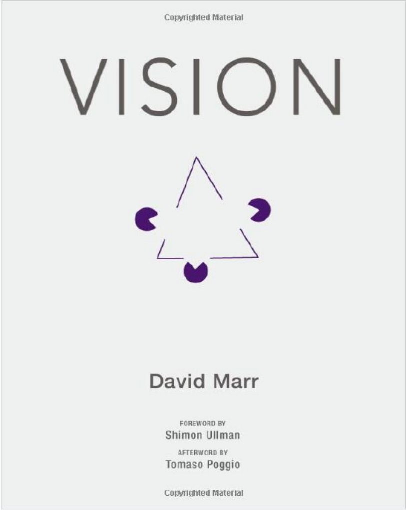
Artificial Intelligence Group July 7, 1966
Vision Memo. No. 100.

THE SUMMER VISION PROJECT

Seymour Papert

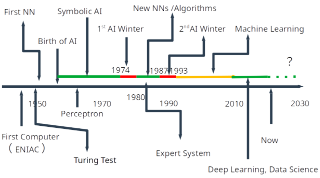
The summer vision project is an attempt to use our summer workers effectively in the construction of a significant part of a visual system. The particular task was chosen partly because it can be segmented into sub-problems which will allow individuals to work independently and yet participate in the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

1966

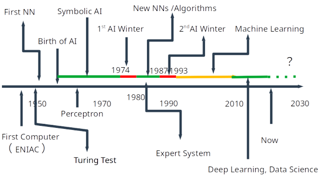


around 1980

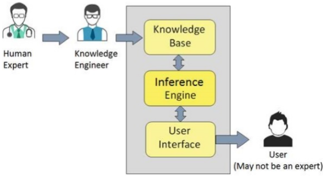
Second golden age



Second golden age



expert system



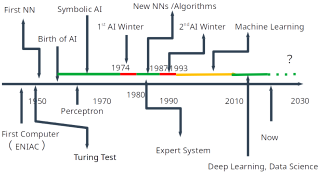
Can we build comprehensive knowledge bases and know all rules?

Big bang in DNNs

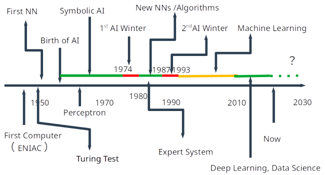
Key ingredients of DL have been in place for 25-30 years:

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
CTF	*DARPA etc*	*mid 1980's*

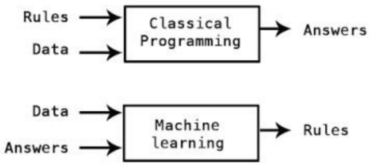
After 2nd AI winter



After 2nd AI winter



Machine learning takes over ...



Golden age of Machine learning

Starting 1990's

- Support vector machines (SVM)
- Adaboost
- Decision trees and random forests
- Deep learning
- ...

Outline

Suggested reading

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

References i

[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*.