Neural Networks 1 - Artificial Neurons

18NES1 - Lecture 2, Summer semester 2024/25

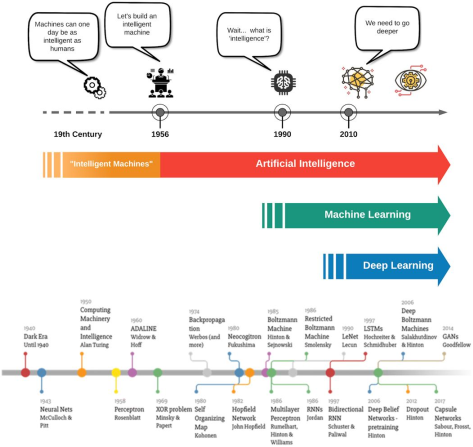
Zuzana Petříčková

February 24, 2025

What We Covered Last Time

- Introduction to Artificial Intelligence and Machine Learning
- Fundamental Concepts of Machine Learning
- Brief History of Artificial Neural Networks

https://www.codesofinterest.com/p/build-deeper.html



Mourtzis, Dimitris & Angelopoulos, John. (2020). An intelligent framework for modelling and simulation of artificial neural networks (ANNs) based on augmented reality. International Journal of Advanced Manufacturing Technology. 111. 10.1007/s00170-020-06192-y.

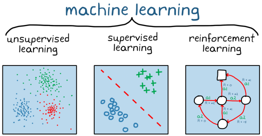
Review — Machine Learning

Core Principle:

- The system "builds itself," meaning it learns from data (training dataset) or previous experiences.

Learning Paradigms:

- **Supervised Learning**
  - Training dataset in the form of [*input, expected output*]
- **Unsupervised Learning (Self-Organization)**
  - Training dataset in the form of [*input*]
- **Reinforcement Learning**
  - The program learns an optimal strategy based on previous experiences.

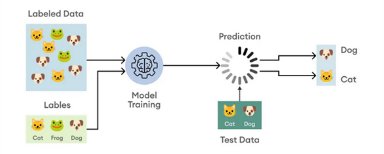


Source: https://www.mathworks.com/discovery/reinforcement-learning.html

Review — Machine Learning

Supervised Learning - Task Types

- **Classification:** Predicting a discrete class (category)



- **Regression:** Predicting a numerical value (e.g., price, temperature, handwriting slant, etc.)

- **Structured Data Learning** (e.g., natural language sentences, molecular structures, etc.)

Review — Machine Learning

Typical Machine Learning Workflow

- **Data Preprocessing**
  - Transforming raw data into a format suitable for machine learning models.
  - Example: Feature selection.
- **Model Selection and Development**
  - Choosing the right type of model (depends on the problem).
  - Selecting a specific model within the chosen type (optimizing parameters).
- **Model Evaluation** — Best performed on unseen (test) data.

Review — Brief History of Artificial Neural Networks

Development Progressed in Waves:

- The field experienced cycles of rapid progress and high expectations, followed by periods of disappointment and stagnation.

Key Milestones:

- **1940 – 1960:** Theoretical foundations.
- **1960 – 1970:** First boom — the "single neuron era."
- **1970 – 1980:** First "AI Winter" for neural networks.
- **1980 – 1990:** Second boom — the era of "shallow" neural networks.
- **1990 – 2000:** Gradual stabilization of the field.
- **2000 – 2010:** Second "AI Winter" for neural networks.
- **2010 – Present:** Third boom — the era of "deep" neural networks.

Today's Lecture: A Single Neuron

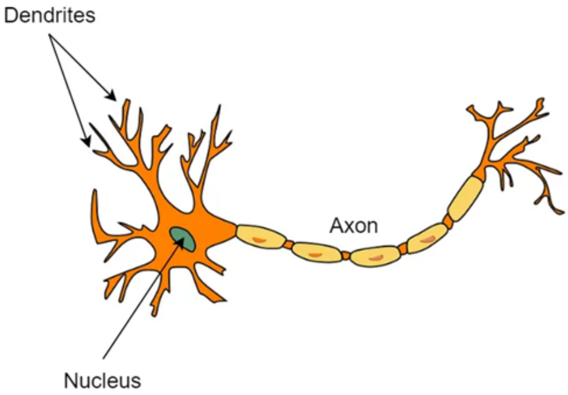
Today:

1. From Biological to Artificial Neurons
2. The Earliest Artificial Neuron Models:
   - McCulloch-Pitts Neuron (1943)
   - Perceptron (Rosenblatt, 1955)
3. Perceptron and Logical Function Representation
4. Perceptron Network — Logical Threshold Circuit
5. Geometric Interpretation of the Perceptron and Linear Separability

Next Week:

1. Perceptron Learning Algorithms

Biological Neuron Model



Biological Neuron

- Fundamental building block of biological neural networks.
- The output depends on inputs received by the neuron and how they are processed within the neuron's body.

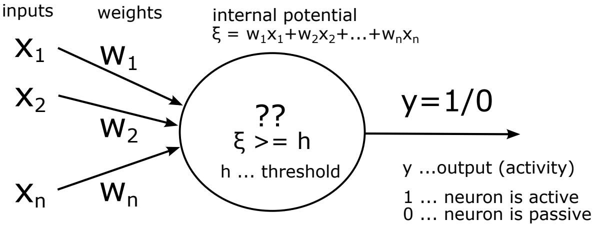
Biological Neural Network

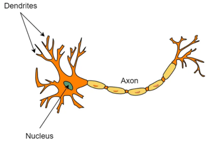
- Neurons are interconnected to form networks.
  - Axons connect to dendrites of other neurons via synapses.
  - New synapses are formed throughout life $\rightarrow$ this is essential for **learning and memory**.

Memory Mechanisms

- **Short-Term Memory Mechanism**
  - Based on cyclic circulation of neural signals within neural networks.
  - Information retention lasts approximately 30 seconds.
- **Medium-Term Memory Mechanism**
  - Based on synaptic modifications and changes in neuron weights.
  - The hippocampus plays a crucial role.
  - Information retention ranges from hours to days.
- **Long-Term Memory Mechanism**
  - Long-term synaptic modifications rely on proteins in neuronal nuclei.
  - Information can be retained for a lifetime.

From Biological to Artificial Neurons





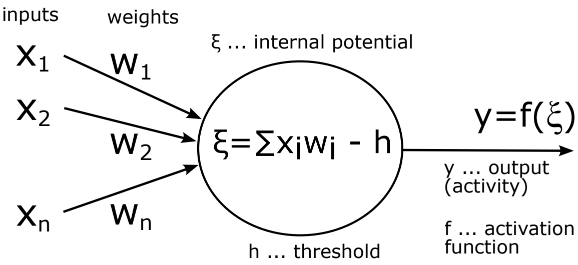
Neuron Model:

- Inputs: $x_1, x_2, \dots, x_n$ (binary values: 0 or 1).
- Input weights: $w_1, w_2, \dots, w_n$.
- Threshold: h.
- Internal potential: weighted sum of inputs

$$\xi = w_1 x_1 + w_2 x_2 + \cdots + w_n x_n$$

- Output: 1 or 0 (depending on whether the internal potential exceeds the threshold or not).

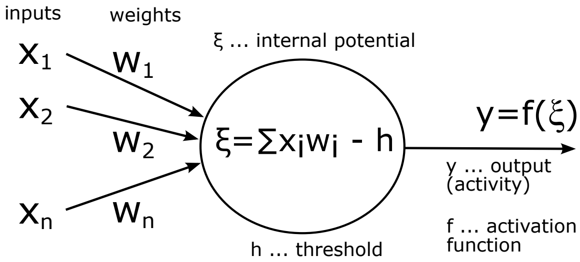
Mathematical Model of a Neuron — Formal Definition



- **Neuron Parameters:**
  - Weight vector: $\vec{w} = (w_1, ..., w_n) \in \mathbb{R}^n$.
  - Threshold (bias): $h \in \mathbb{R}$.
  - Activation function: $f : \mathbb{R} \to \mathbb{R}$.
- Given an input $\vec{x} \in \mathbb{R}^n$, the neuron computes an output $y \in \mathbb{R}$ as:

$$y = f_{\vec{w},h}(\vec{x})$$

Mathematical Model of a Neuron — Formal Definition



Neuron Output Calculation:

- Internal potential:

$$\xi = \sum_{i=1}^{n} w_i x_i - h = \vec{w} \cdot \vec{x}^T - h$$

- Output:

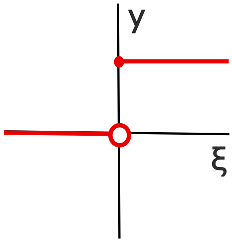
$$y = f(\xi)$$

Mathematical Model of a Neuron — Formal Definition

- The terminology used originates from early neuron models:
  - **Perceptron** (Rosenblatt, 1955).
  - **McCulloch-Pitts Neuron** (1943).
- Both models utilized a **step activation function**.

Step Activation Function:

- If $\sum_{i=1}^{n} w_i x_i \geq h$, i.e.,
  $\xi = \sum_{i=1}^{n} w_i x_i - h \geq 0$
  then the neuron is **active** ($f(\xi) = 1$).
- If $\sum_{i=1}^{n} w_i x_i < h$, i.e.,
  $\xi = \sum_{i=1}^{n} w_i x_i - h < 0$
  then the neuron is **passive** ($f(\xi) = 0$).



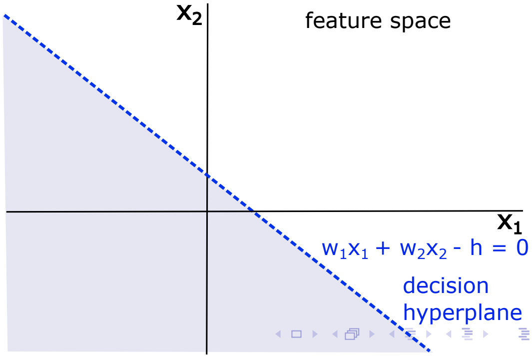
Mathematical Model of a Neuron

Geometric Interpretation of an Artificial Neuron

- The neuron's inputs can be represented as points in an n-dimensional Euclidean space (input/feature space).
- Setting the neuron's internal potential to $\xi = 0$ results in the equation of a **decision hyperplane (decision boundary)** .

$$\xi = w_1 x_1 + w_2 x_2 - h = 0$$

$$x_2 = -\frac{w_1}{w_2} x_1 + \frac{h}{w_2}$$



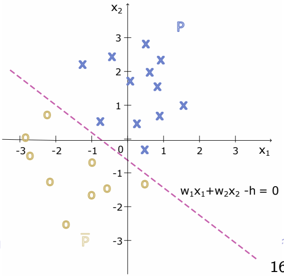
Perceptron (Rosenblatt, 1955) and McCulloch-Pitts Neuron (1943)

The perceptron can be used as a **linear classifier**, separating patterns into two classes (P and $\overline{P}$).

- **why linear?:** The boundary between the two classes is a hyperplane: $w_1 x_1 + w_2 x_2 + \cdots + w_n x_n - h = 0$ which represents a point, a line, or a plane depending on the number of input dimensions.

Step Activation Function:

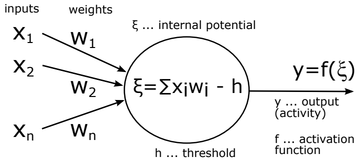
- If $\sum_{i=1}^{n} w_i x_i \geq h$, i.e., $\xi \geq 0, f(\xi) = 1$ ... the neuron is **active** (class P).
- If $\sum_{i=1}^{n} w_i x_i < h$, i.e., $\xi < 0, f(\xi) = 0$ ... the neuron is **passive** (class $\overline{P}$).



McCulloch-Pitts Neurons (1943)

Binary Variant:

- Binary inputs: $x_i \in \{0, 1\}$
- Binary outputs: $y \in \{0, 1\}$
- Weights: $w_i \in \{-1, 1\}$
- Step activation function



Bipolar Variant:

- Bipolar inputs: $x_i \in \{-1, 1\}$
- Bipolar outputs: $y \in \{-1, 1\}$
- Weights: $w_i \in \{-1, 1\}$
- Step activation function

Application: Representation of logical functions (AND, OR, NOT, etc.) $\rightarrow$ We will explore this in a moment.

Major drawback:

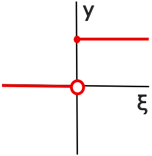
- The model had no learning algorithm. (Later solved using **Hebbian learning**.)

Perceptron (Rosenblatt, 1955)

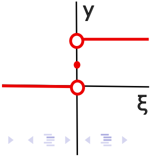
- **Real-valued inputs**: $x_i \in \mathbb{R}$
- **Real-valued weights and thresholds**: $w_i \in \mathbb{R}$
- **Outputs:**
  - Binary variant: $y \in \{0, 1\}$
  - Bipolar variant: $y \in \{-1, 1\}$

Step Activation Function Variants for Binary Perceptron:

$$f(\xi) = \begin{cases} 1, & \text{if } \xi \geq 0 \quad \text{(active neuron)} \\ 0, & \text{if } \xi < 0 \quad \text{(passive neuron)} \end{cases}$$



$$f(\xi) = \begin{cases} 1, & \text{if } \xi > 0 \quad \text{(active neuron)} \\ 0.5, & \text{if } \xi = 0 \quad \text{(neutral neuron)} \\ 0, & \text{if } \xi < 0 \quad \text{(passive neuron)} \end{cases}$$



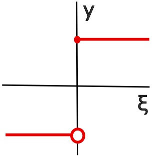
→ Also known as the **signum function** (signum).

Perceptron (Rosenblatt, 1955)

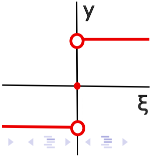
- **Real-valued inputs**: $x_i \in \mathbb{R}$
- **Real-valued weights and thresholds**: $w_i \in \mathbb{R}$
- **Outputs:**
    - Binary variant: $y \in \{0, 1\}$
    - Bipolar variant: $y \in \{-1, 1\}$

Step Activation Function Variants for Bipolar Perceptron:

$$f(\xi) = \begin{cases} 1, & \text{if } \xi \geq 0 \quad \text{(active neuron)} \\ -1, & \text{if } \xi < 0 \quad \text{(passive neuron)} \end{cases}$$

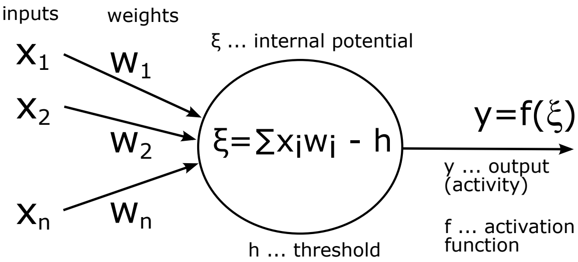


$$f(\xi) = \begin{cases} 1, & \text{if } \xi > 0 \quad \text{(active neuron)} \\ 0, & \text{if } \xi = 0 \quad \text{(neutral neuron)} \\ -1, & \text{if } \xi < 0 \quad \text{(passive neuron)} \end{cases}$$



$\rightarrow$ Also known as the **sign** function (sign).

Mathematical Model of a Neuron — Original Definition



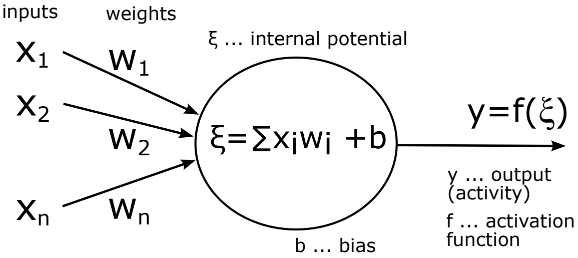
Classical Definition: Threshold h

- Internal potential:

$$\xi = \sum_{i=1}^{n} w_i x_i - h = \vec{w} \cdot \vec{x}^T - h$$

- Output: $y = f(\xi)$

Mathematical Model of a Neuron — Modern Definition



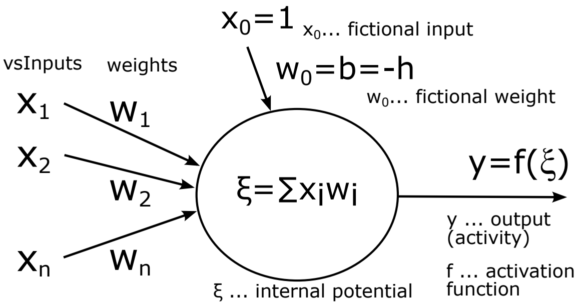
Alternative Definition: Threshold h → Bias b

- Internal potential:

$$\xi = \sum_{i=1}^{n} w_i x_i + b = \vec{w} \cdot \vec{x}^T + b$$

- Output: $y = f(\xi)$

Mathematical Model of a Neuron — Matrix Definition



Alternative Definition: Introducing a Fictional Bias Input

- Extended feature space ... $\vec{x} = (x_0 = 1, x_1, ..., x_n)$
- Extended weight vector ... $\vec{w} = (w_0 = b = -h, w_1, ..., w_n)$
- Internal potential ... $\xi = \sum_{i=0}^{n} w_i x_i = \vec{w} \cdot \vec{x}^T$
- Output: $y = f(\xi)$

Neural Networks 1 — Lecture 2: Artificial Neuron

Perceptron and Logical Function Representation

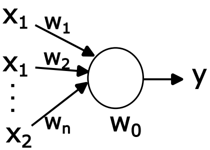
A perceptron can implement basic logical functions:

- NOT (negation)
- ID (identity)
- AND (conjunction)
- OR (disjunction)

$\rightarrow$ Using perceptrons, we can build a **logical threshold circuit**, allowing the representation of any Boolean function.

Perceptron and Logical Function Representation

We consider the following perceptron model:



$$\xi = \vec{w} \cdot \vec{x}^T = \sum_{i=0}^{n} w_i x_i$$

$$= w_0 + \sum_{i=1}^{n} w_i x_i$$

Binary Perceptron

- Inputs: $x_i \in \{0, 1\}$
- Outputs: $y \in \{0, 0.5, 1\}$
- $y = f(\xi) = signum(\xi)$

$$signum(\xi) = \begin{cases} 1, & \text{if } \xi > 0 \\ 0.5, & \text{if } \xi = 0 \\ 0, & \text{if } \xi < 0 \end{cases}$$

Bipolar Perceptron

- Inputs: $x_i \in \{-1, +1\}$
- Outputs: $y \in \{-1, 0, +1\}$
- $y = f(\xi) = sign(\xi)$

$$sign(\xi) = \begin{cases} 1, & \text{if } \xi > 0 \\ 0, & \text{if } \xi = 0 \\ -1, & \text{if } \xi < 0 \end{cases}$$

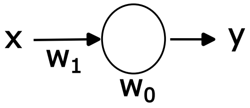
Logical Functions — NOT (Negation)

Bipolar Model

x	$y = \neg x$
-1	1
1	-1

Binary Model

x	$y = \neg x$
0	1
1	0



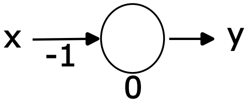
$$y = sign(w_0 + w_1 x)$$

- How should we choose w_0 and w_1 for the bipolar model?
- And for the binary model?

Logical Functions — NOT (Negation)

Bipolar Model

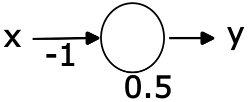
x	$y = \neg x$
-1	1
1	-1



$$y = sign(w_0 + w_1 x) = sign(-x)$$

Binary Model

x	$y = \neg x$
0	1
1	0

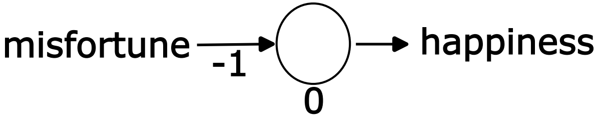


$$y = signum(w_0 + w_1 x) = signum(0.5 - x)$$

Question: Can you think of alternative solutions?

Logical Functions — NOT (Negation)

Example: Happiness $= \neg$ Misfortune



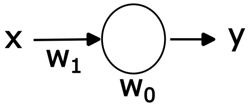
Logical Functions — ID (Identity)

Bipolar Model

x	$y = x$
-1	-1
1	1

Binary Model

x	$y = x$
0	0
1	1



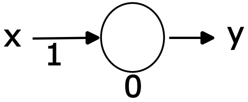
$$y = sign(w_0 + w_1 x)$$

- How do we choose w_0 and w_1 for the bipolar model?
- And for the binary model?

Logical Functions — ID (Identity)

Bipolar Model

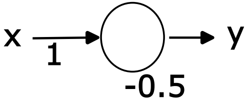
x	$y = x$
-1	-1
1	1



$$y = sign(w_0 + w_1 x) = sign(x)$$

Binary Model

x	$y = x$
0	0
1	1



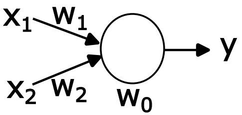
$$y = signum(w_0 + w_1 x) = signum(-0.5 + x)$$

$\rightarrow$ Nothing to solve here.

Logical Functions — AND (Conjunction)

Bipolar Model

x_1	x_2	$y = x_1 \wedge x_2$
-1	-1	-1
-1	+1	-1
+1	-1	-1
+1	+1	+1



Binary Model

x_1	x_2	$y = x_1 \wedge x_2$
0	0	0
0	1	0
1	0	0
1	1	1

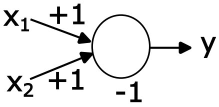
$$y = sign(w_0 + w_1 x_1 + w_2 x_2)$$

- How do we choose w_0, w_1, and w_2 for the bipolar model?
- And for the binary model?

Logical Functions — AND (Conjunction)

Bipolar Model

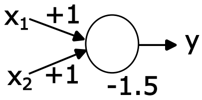
x_1	x_2	$y = x_1 \wedge x_2$
-1	-1	-1
-1	+1	-1
+1	-1	-1
+1	+1	+1



$$y = sign(w_0 + w_1 x_1 + w_2 x_2)$$
$$= sign(-1 + x_1 + x_2)$$

Binary Model

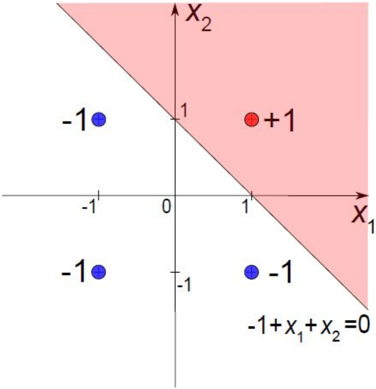
x_1	x_2	$y = x_1 \wedge x_2$
0	0	0
0	1	0
1	0	0
1	1	1



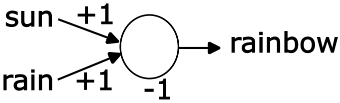
$$y = signum(w_0 + w_1 x_1 + w_2 x_2)$$
$$= signum(-1.5 + x_1 + x_2)$$

Question: Could there be other solutions?

Logical Functions — AND (Conjunction)



Example: Rainbow $=$ Sun $\wedge$ Rain

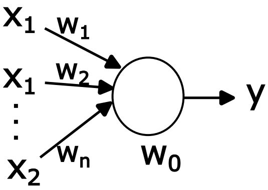


(Bipolar Model)

Logical Functions — AND (Conjunction)

How do we set the weights in the general case?

$$y = x_1 \wedge x_2 \wedge \cdots \wedge x_n$$

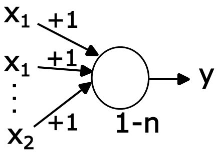


$$y = sign\left(w_0 + \sum_{i=1}^{n} w_i x_i\right)$$

Logical Functions — AND (Conjunction)

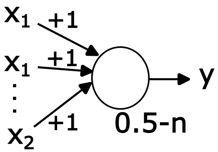
$$y = x_1 \wedge x_2 \wedge \cdots \wedge x_n$$

Bipolar Model



$$y = sign\left(w_0 + \sum_{i=1}^{n} w_i x_i\right)$$
$$= sign\left(1 - n + \sum_{i=1}^{n} x_i\right)$$

Binary Model

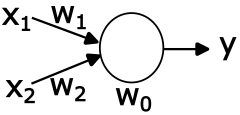


$$y = signum\left(w_0 + \sum_{i=1}^{n} w_i x_i\right)$$
$$= signum\left(0.5 - n + \sum_{i=1}^{n} x_i\right)$$

Logical Functions — OR (Disjunction)

Bipolar Model

x_1	x_2	$y = x_1 \vee x_2$
-1	-1	-1
-1	+1	+1
+1	-1	+1
+1	+1	+1



Binary Model

x_1	x_2	$y = x_1 \vee x_2$
0	0	0
0	1	1
1	0	1
1	1	1

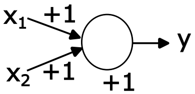
$$y = sign(w_0 + w_1 x_1 + w_2 x_2)$$

- How do we choose w_0, w_1, and w_2 for the bipolar model?
- And for the binary model?

Logical Functions — OR (Disjunction)

Bipolar Model

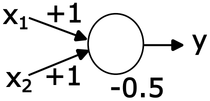
x_1	x_2	$y = x_1 \vee x_2$
-1	-1	-1
-1	$+1$	$+1$
$+1$	-1	$+1$
$+1$	$+1$	$+1$



$$
\begin{aligned}
y &= sign(w_0 + w_1x_1 + w_2x_2) \\
&= sign(1 + x_1 + x_2)
\end{aligned}
$$

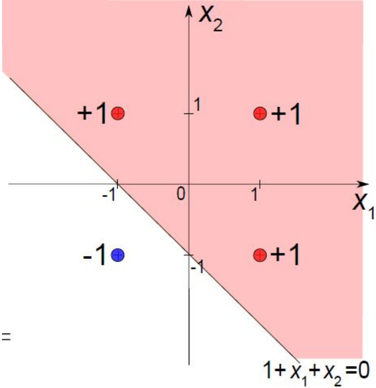
Binary Model

x_1	x_2	$y = x_1 \vee x_2$
0	0	0
0	1	1
1	0	1
1	1	1

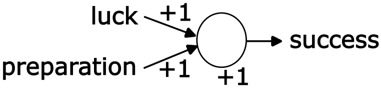


$$
\begin{aligned}
y &= signum(w_0 + w_1x_1 + w_2x_2) \\
&= signum(-0.5 + x_1 + x_2)
\end{aligned}
$$

Logical Functions — OR (Disjunction)



Example: success $=$ luck $\vee$ preparation

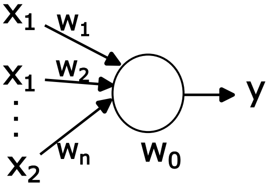


(Bipolar Model)

Logical Functions — OR (Disjunction)

How do we set the weights in the general case?

$y = x_1 \vee x_2 \vee \cdots \vee x_n$

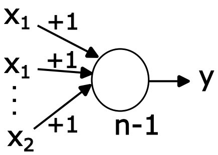


$y = sign(w_0 + \sum_{i=1}^{n} w_i x_i)$

Logical Functions — OR (Disjunction)

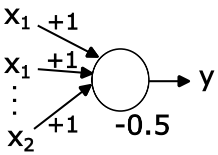
$y = x_1 \vee x_2 \vee \cdots \vee x_n$

Bipolar Model



$$y = sign(w_0 + \sum_{i=1}^{n} w_i x_i)$$
$$= sign(n - 1 + \sum_{i=1}^{n} x_i)$$

Binary Model



$$y = signum(w_0 + \sum_{i=1}^{n} w_i x_i)$$
$$= signum(-0.5 + \sum_{i=1}^{n} x_i)$$

Logical Functions — Exclusive OR (XOR)

Bipolar Model

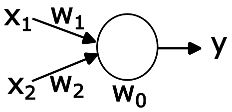
x_1	x_2	$y = x_1 \oplus x_2$
-1	-1	-1
-1	+1	+1
+1	-1	+1
+1	+1	-1

Binary Model

x_1	x_2	$y = x_1 \oplus x_2$
0	0	0
0	1	1
1	0	1
1	1	0

Example:

peaceful home = cat $\oplus$ rabbit



$$y = sign(w_0 + w_1 x_1 + w_2 x_2)$$

- How do we choose w_0, w_1, and w_2 for the bipolar model?
- And for the binary model?

Logical Functions — Exclusive OR (XOR)

Bipolar Model

x_1	x_2	$y = x_1 \oplus x_2$
-1	-1	-1
-1	+1	+1
+1	-1	+1
+1	+1	-1

$$1 \ldots w_0 - w_1 - w_2 < 0$$
$$2 \ldots w_0 - w_1 + w_2 > 0$$
$$3 \ldots w_0 + w_1 - w_2 > 0$$
$$4 \ldots w_0 + w_1 + w_2 < 0$$

$$sign(w_0 - w_1 - w_2) = -1$$
$$sign(w_0 - w_1 + w_2) = +1$$
$$sign(w_0 + w_1 - w_2) = +1$$
$$sign(w_0 + w_1 + w_2) = -1$$

Adding 1st and 4th rows, and 2nd and 3rd rows:

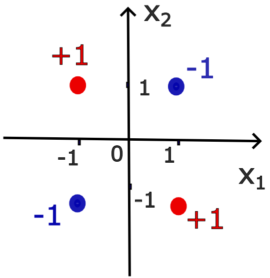
$$2w_0 < 0$$
$$2w_0 > 0$$

$\rightarrow$ Contradiction

Conclusion: A single perceptron cannot represent the XOR function.

Logical Functions — Exclusive OR (XOR)

- XOR cannot be implemented using a single perceptron:



A perceptron cannot implement all logical functions:

- However, by combining perceptrons for NOT, ID, AND, and OR in a structured way (into a neural network, specifically a logical threshold circuit), we can construct more complex logical functions.

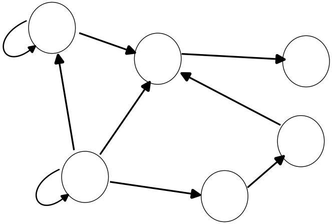
Neural Networks 1 - Lecture 3: Artificial Neuron

Neural Network

- A neural network consists of neurons that are interconnected by edges.
- The output of one neuron can serve as an input to one or more other neurons.

Neural Network Architecture (Topology)

- Represented as a directed graph, where neurons correspond to nodes and synaptic connections correspond to edges.



Neural Network

Output Neurons

- Their outputs together form the final output of the neural network.
- **Typically:** no outgoing edges to other neurons.

Input Neurons

- Their inputs are the input patterns.
- **Typically:** no incoming edges from other neurons.

Network Output (Response)

- Defined by the activities of the output neurons.

Neural Network

Definition: A neural network is a six-tuple (N, C, I, O, w, t):

- N is a finite non-empty set of neurons.
- $C \subseteq N \times N$ is a non-empty set of directed connections (edges) between neurons.
- $I \subseteq N$ is a non-empty set of input neurons.
- $O \subseteq N$ is a non-empty set of output neurons.
- $w : C \to R$ is a weight function.
- $t : N \to R$ is a bias function.

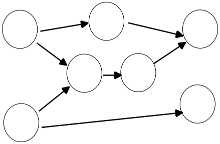
Neural Network Configuration

- Defined by the weights of all edges and the biases of all neurons.

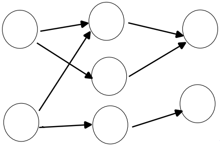
Neural Network

Neural Network Architecture

- **Cyclic, recurrent** networks: allow feedback connections.
- **Acyclic, feedforward** networks: all connections go in the same direction (i.e., the graph can be topologically ordered).



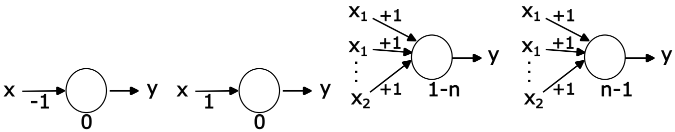
- **Hierarchical (layered)** networks: divided into layers, with connections only between neurons in consecutive layers.



Neural Networks 1 - Lecture 3: Artificial Neuron

Logical Threshold Circuit

- Using perceptrons for NOT, ID, AND, and OR, we can construct a neural network representing more complex logical functions.



- AND represents the **intersection** of convex regions, while OR represents their **union**.

Questions: How does the logical function implemented by a perceptron changes if:

- All weights (including the bias) are multiplied by a positive number?
- All weights (including the bias) are multiplied by a negative number (e.g., -1)?
- Some weights are multiplied by -1?

Logical Threshold Circuit - Examples

Example 1: Crop Yield Prediction

$$\text{yield} = ((\text{warm} \wedge \text{rain}) \vee (\text{warm} \wedge \text{irrigation})) \wedge \text{fertilizer} \wedge \neg\text{pests}$$

1. Design a perceptron network for this logical function using basic logical operations.
   - How many inputs, outputs, neurons, and layers does it require?
2. Design a perceptron network with a hierarchical (layered) architecture.
3. Minimize this logical function and design a neural network for the simplified version.
4. Can this logical function be represented by a single perceptron?

Logical Threshold Circuit - Examples

Example 2: Majority Circuit

$$y = (x_1 \wedge x_2) \vee (x_1 \wedge x_3) \vee (x_2 \wedge x_3)$$

1. Design a neural network for the majority circuit using basic logical operations.
2. Can it be represented by a single perceptron?
3. What would the solution look like for the general case (for arbitrary n)?

Perceptron Network as a Logical Threshold Circuit

Theorem:
Every logical formula can be expressed in **Disjunctive Normal Form (DNF)**, i.e., as a disjunction of conjunctions of atoms, where atoms are variables or their negations.

- Disjunction: $F = K_1 \vee K_2 \vee ... \vee K_n$
- Conjunction: $K_i = A_{i1} \wedge A_{i2} \wedge ... \wedge A_{in_i}$
- Atoms: $A_{ij} = L$ or $A_{ij} = \neg L$

Example: $y = (x_2 \wedge x_4) \vee \neg x_1 \vee (x_2 \wedge \neg x_3)$
Consequence:
Every logical function can be represented by a perceptron-based neural network.

- **Question:** Design a schematic of such a perceptron neural network. How many layers will it have?

Perceptron Network as a Logical Threshold Circuit

Similarly:
Every logical formula can be expressed in **Conjunctive Normal Form (CNF)**, i.e., as a conjunction of disjunctions of atoms, where atoms are variables or their negations.

- Conjunction: $F = D_1 \wedge D_2 \wedge \ldots \wedge D_n$
- Disjunction: $D_i = A_{i1} \vee A_{i2} \vee \ldots \vee A_{in_i}$
- Atoms: $A_{ij} = L$ or $A_{ij} = \neg L$

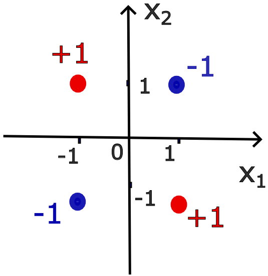
Example: $y = (x_2 \vee x_4) \wedge \neg x_1 \wedge (x_2 \vee \neg x_3)$
Implementation using a Perceptron Network: Analogous to DNF

- AND represents the **intersection** of convex regions, while OR represents their **union**.

Logical Threshold Circuit - Examples

Example 3: Exclusive OR (XOR)

- XOR cannot be implemented using a single perceptron.



- However, XOR can be implemented using a perceptron network.

Logical Threshold Circuit - Examples

Example 3: Exclusive OR (XOR) - Optional Homework for Next Time

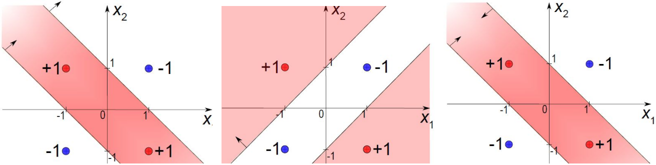
1. XOR can be represented using basic logical operations (AND, OR, NOT) in different ways. Can you design multiple representations?
2. Design the smallest possible neural network that can represent XOR. How many neurons does it contain?

x_1	x_2	$y = x_1 \otimes x_2$
-1	-1	-1
-1	+1	+1
+1	-1	+1
+1	+1	-1

Logical Threshold Circuit - Examples

Example 3: Exclusive OR (XOR) - Optional Homework for Next Time

1. XOR can be represented using basic logical operations (AND, OR, NOT) in different ways. Can you design multiple representations?
2. Design the smallest possible neural network that can represent XOR. How many neurons does it contain?

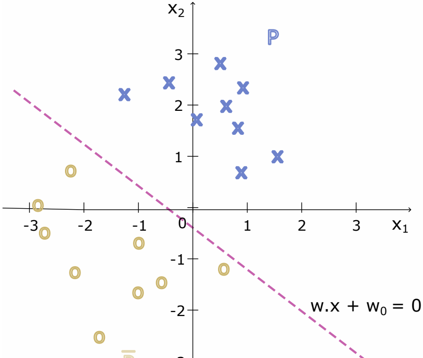


Neural Networks 1 - Lecture 3: Artificial Neuron

Linear Separability

→ The perceptron can function as a **linear classifier**, categorizing patterns into two classes (here P, $\overline{P}$) using a **decision hyperplane**:

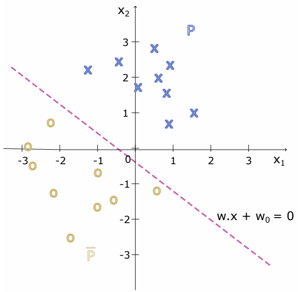
$$\vec{w}\vec{x} + w_0 = \sum_{i=1}^{n} w_i x_i + w_0 = 0$$



Linear Separability

Definition:
Two sets, P and $\overline{P}$, are **linearly separable** in an n-dimensional space if there exist real numbers $w_0, w_1, \ldots, w_n$ such that:
For each point $\vec{x} \in P$: $\sum_{i=1}^{n} w_i x_i + w_0 > 0$ For each point $\vec{x} \in \overline{P}$: $\sum_{i=1}^{n} w_i x_i + w_0 < 0$

Why was this concept introduced?
→ Researchers investigated which functions could be implemented by a perceptron (or, more generally, a linear classifier) and which could not.



Linear Separability in Boolean Space for $n = 2$

- There are a total of $2^4 = 16$ logical functions, out of which **14 are linearly separable**:
  - 6 simple logical functions:

$$0, 1, A, B, \neg A, \neg B$$

    $\rightarrow$ trivial cases
  - 8 variations of conjunction and disjunction:

$$A \wedge B, A \wedge \neg B, \neg A \wedge B, \neg A \wedge \neg B$$

$$A \vee B, A \vee \neg B, \neg A \vee B, \neg A \vee \neg B$$

    $\rightarrow$ slightly more complex cases
- **2 functions are not linearly separable** and thus cannot be realized using a perceptron:
  - $A \otimes B$ (XOR) and $A \Leftrightarrow B$ (Equivalence)

Linear Separability in General Boolean Space

For a general Boolean space:

- $n = 2$ … 14 out of $2^4 = 16$ logical functions are linearly separable.
- $n = 3$ … 104 out of $2^8 = 256$ functions are separable.
- $n = 4$ … 1882 out of $2^{16} = 65536$ functions are separable.
- n general … ??

$\rightarrow$ The number of functions that cannot be represented by a perceptron is significant, and their proportion increases with the dimensionality of the feature (input) space.

How can we address this?

- Instead of using a single neuron, we can use a **perceptron network**.
- We can extend the feature space by adding additional variables.