Advanced Machine Learning (Semester 1 2023)

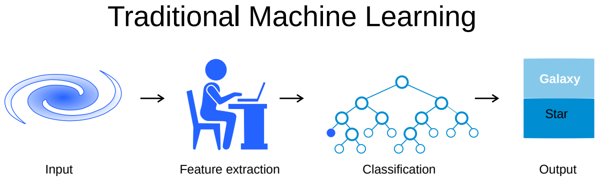
Artificial Neural Networks (I)

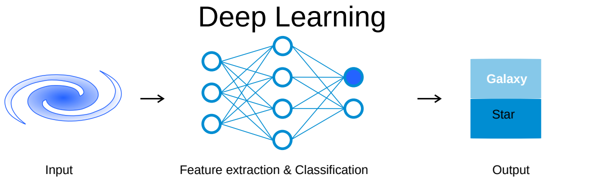
Nina Hernitschek
Centro de Astronomía CITEVA
Universidad de Antofagasta

April 17, 2023

Motivation

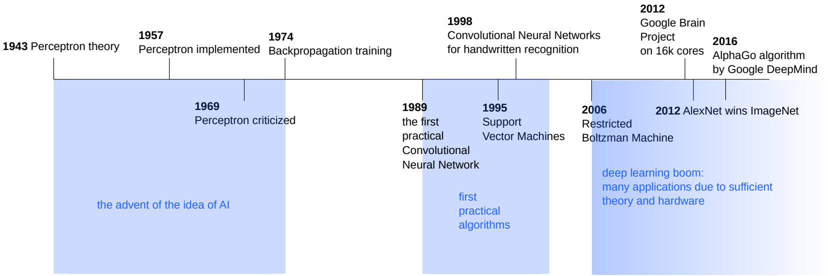
Neural Networks are mimicing processes in the human brain to **solve complex data-driven problems**. They are the functional unit of Deep Learning.
The input data is processed through different **layers of artificial neurons** producing the desired output. From speech and face recognition to astronomy and robotics, Neural Networks are used in many domains.





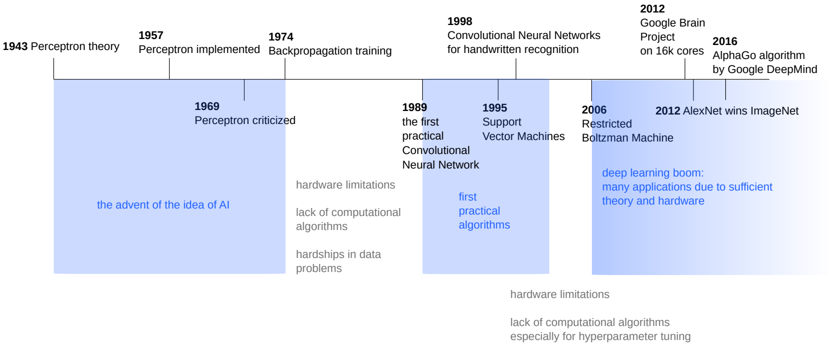
Motivation

a brief history of Neural Networks:



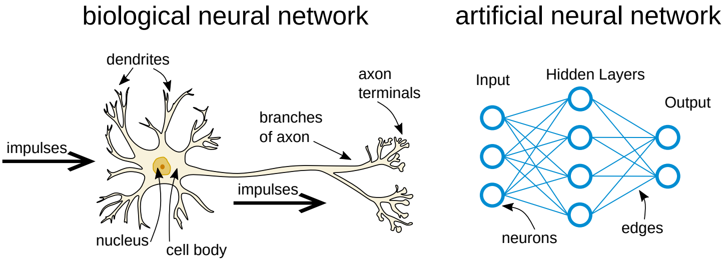
Motivation

a brief history of Neural Networks:



Neural Networks - The Idea

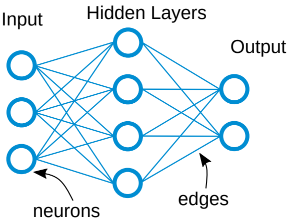
Artificial neural networks (ANNs), usually simply called neural networks (NNs), are information processing paradigms **inspired by biological neural networks**.



Neural Networks - The Idea

Artificial neural networks (ANNs), usually simply called neural networks (NNs), are information processing paradigms **inspired by biological neural networks**.

artificial neural network



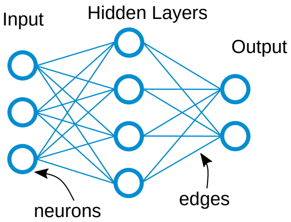
applications:

- pattern recognition
- predictive modeling
- robotics

Neural Networks - The Idea

Artificial neural networks (ANNs), usually simply called neural networks (NNs), are information processing paradigms **inspired by biological neural networks**.

artificial neural network

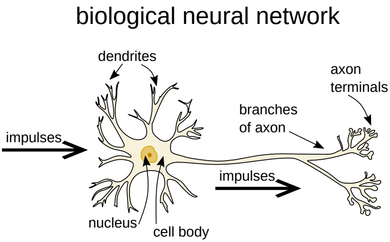


applications:

- pattern recognition
- predictive modeling
- robotics

⇨ training, self-learning via experience to derive conclusions from complex data sets

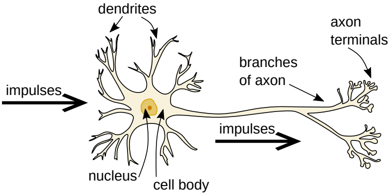
Components of Neural Networks



A biological neuron is an electrically excitable cell that receives, processes, and transmits information through electrical and chemical signals. The signal propagates from neuron to neuron by means of an axon, which connects them. The **connection** takes place between the axon terminals of the emitting neuron and the dendrites of the receiving neuron, in a structure called synapse.

Components of Neural Networks

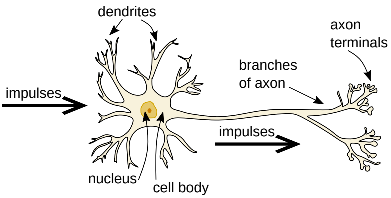
biological neural network



When a neuron is at rest, i.e. it is not receiving any signal, it is subject to a resting electric potential, which on average is around -70 millivolt and it is related to the difference between electric charge inside and outside the neuron. When a cell sends an electric impulse, an electric potential related to it propagates through the axon. It is called action potential. The action potential propagates through the axon.

Components of Neural Networks

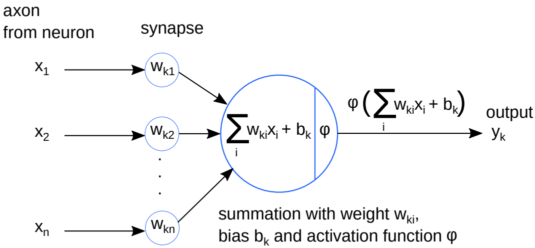
biological neural network



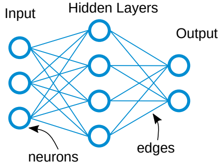
Eventually, the action potential reaches the neuron, where the electric tension rapidly changes. If its value gets bigger than a certain **threshold**, the so-called firing process is triggered, consisting in the emission of a signal from the neuron itself. There is **no partial firing**, that is, when the electric potential reaches the threshold, the neuron fires an electric signal whose intensity is independent of the received one.

Components of Neural Networks

simple mathematical model of a **neuron** k:

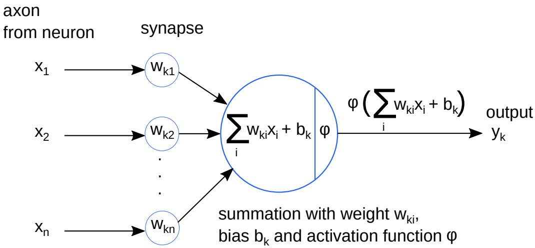


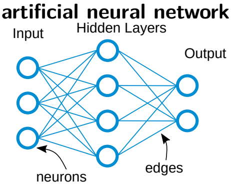
artificial neural network



Components of Neural Networks

simple mathematical model of a **neuron** k:

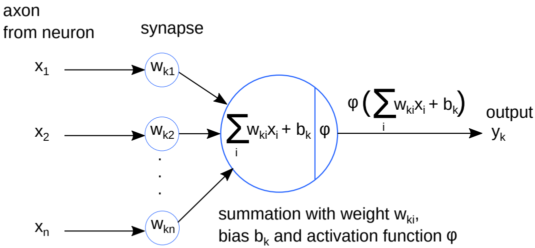




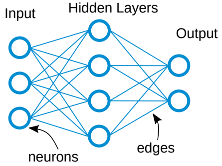
Each artificial neuron (node) has 1 to n inputs (the n-dimensional **input vector**) and produces a **single output** which can be sent 1 to m other neurons.

Components of Neural Networks

simple mathematical model of a **neuron** k:



artificial neural network



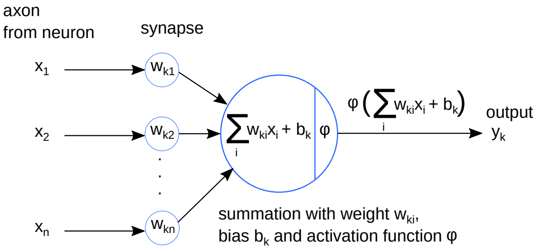
Each neuron k computes their **ouput as weighted sums** of their inputs x_i:

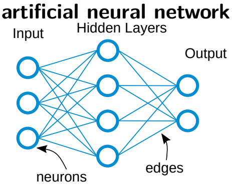
$$v_k = \sum_i w_{ki} x_i \; + b_k = \mathbf{w}_k \cdot \mathbf{x} + b_k$$

The **bias** b is the equivalent of the resting electric potential in the biological neuron. If the sum of the received action potentials reaches b, the neuron fires.

Components of Neural Networks

simple mathematical model of a **neuron** k:





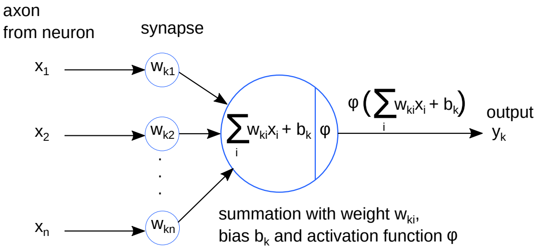
Each neuron k computes their **ouput as weighted sums** of their inputs x_i:

$$v_k = \sum_i w_{ki} x_i \ + b_k = \mathbf{w}_k \cdot \mathbf{x} + b_k$$

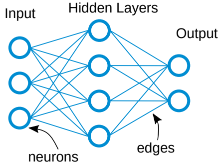
The multiplicative factor, representing the thickness of the myelin sheath covering the axon, is expressed by the **weight $\mathbf{w}_k$**. As we consider a neuron k receiving at each time-step n impulses through n different axons, it is a

Components of Neural Networks

simple mathematical model of a **neuron** k:

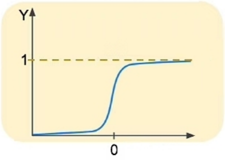


artificial neural network



Activation function* φ controls the amplitude of the output to be within $[0, 1]$:

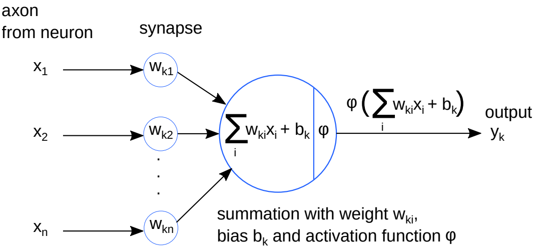
$$y_k = \varphi(v)$$



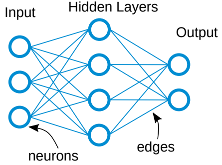
*more on this later

Components of Neural Networks

simple mathematical model of a **neuron** k:



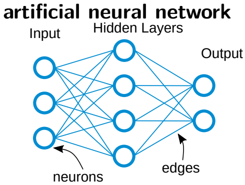
artificial neural network



To recap: a neuron k receives an n-dimensional input $\mathbf{x}$, which is weighted with the weight vector $\mathbf{w}$. The neuron fires a signal with an intensity given by the activation function φ which controls the amplitude of the output to be within $[0, 1]$.

Components of Neural Networks

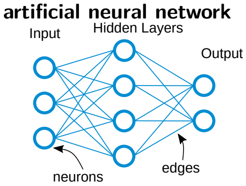
the architecture of a neural network:



Neurons are typically organized into multiple **layers**.

Components of Neural Networks

the architecture of a neural network:



Neurons are typically organized into multiple **layers**.

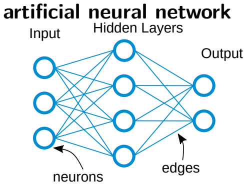
Neurons of one layer connect via **edges** only to neurons of the immediately preceding and immediately following layers.

The layer that receives external data is the **input layer**.

Neuron **inputs** can be external data or outputs of other neurons.

Components of Neural Networks

the architecture of a neural network:



Neurons are typically organized into multiple **layers**.

Neurons of one layer connect via **edges** only to neurons of the immediately preceding and immediately following layers.

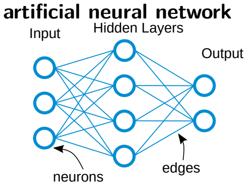
The layer that receives external data is the **input layer**.

Neuron **inputs** can be external data or outputs of other neurons.

The outputs of the **final output neurons** of the neural net accomplish the task, such as recognizing an object in an image.

Components of Neural Networks

the architecture of a neural network:



Neurons are typically organized into multiple **layers**.

Neurons of one layer connect via **edges** only to neurons of the immediately preceding and immediately following layers.

The layer that receives external data is the **input layer**.

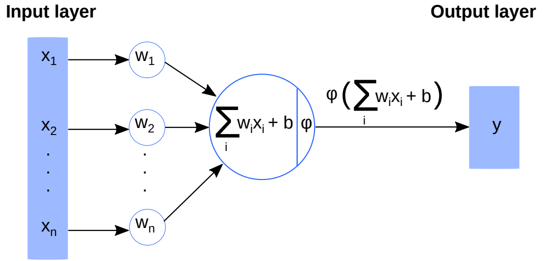
Neuron **inputs** can be external data or outputs of other neurons.

The outputs of the **final output neurons** of the neural net accomplish the task, such as recognizing an object in an image.

a note on the **activation function**:
Usually all hidden layers use the same activation function. However, the output layer will typically use a different activation function from the hidden layers. The choice depends on the goal or type of prediction made by the model.

The Perceptron

The (single-layer) perceptron is the **simplest** and oldest neural network architecture, invented by McCulloch and Pitts (1943) and impemented by the psychologist Frank Rosenblatt (1957).



- a single neuron with inputs x_i, output y
- adjustable (trainable) weights w_i
- Heaviside activation function: $\varphi(\mathbf{w}\cdot\mathbf{x}+b) = H(\mathbf{w}\cdot\mathbf{x}+b) = \begin{cases} 1, & \text{if } \mathbf{w}\cdot\mathbf{x}+b \geq 0 \\ 0, & \text{otherwise.} \end{cases}$
- training sets of $D = \{x_{ij}, \hat{y}_j\}$ where j is the sample number and $\hat{y}_j$ the desired output for that sample
- in training, weights w_i are updated to minimize loss $\sum_j (y_j - \hat{y}_j)^2$

The Perceptron - Mathematical Representation

The Perceptron is mathematically represented as implementing the following function:

$$y = \varphi(w_1 x_1 + w_2 x_2 + \ldots + w_n x_n + b) = \varphi(\mathbf{w} \cdot \mathbf{x} + b)$$

with

$$\varphi(\mathbf{w} \cdot \mathbf{x} + b) = \begin{cases} 1, & \text{if } \mathbf{w} \cdot \mathbf{x} + b \geq 0 \\ 0, & \text{otherwise.} \end{cases}$$

where $\mathbf{w}$ is a vector of real-valued weights, $\mathbf{w} \cdot \mathbf{x}$ is the dot product $\sum_{i=1}^{n} w_i x_i$, where n is the input vector dimension, and b is the bias.

For a particular choice of $\mathbf{w}$ and b, the output y only depends on the input vector $\mathbf{x}$.

The Perceptron - Mathematical Representation

The Perceptron is mathematically represented as implementing the following function:

$$y = \varphi(w_1 x_1 + w_2 x_2 + \ldots + w_n x_n + b) = \varphi(\mathbf{w} \cdot \mathbf{x} + b)$$

with

$$\varphi(\mathbf{w} \cdot \mathbf{x} + b) = \begin{cases} 1, & \text{if } \mathbf{w} \cdot \mathbf{x} + b \geq 0 \\ 0, & \text{otherwise.} \end{cases}$$

where $\mathbf{w}$ is a vector of real-valued weights, $\mathbf{w} \cdot \mathbf{x}$ is the dot product $\sum_{i=1}^{n} w_i x_i$, where n is the input vector dimension, and b is the bias.

For a particular choice of $\mathbf{w}$ and b, the output y only depends on the input vector $\mathbf{x}$.

? What is the role of the bias?

The Perceptron - Mathematical Representation

You may notice the similarity of the Perceptron with the canonical form of a **linear function**. If we remove the activation function and consider only one input for clarity, the equations would be the same:

$$f(x) = ax + b \mathrel{\widehat{=}} w_i x_i + b$$

The Perceptron - Mathematical Representation

You may notice the similarity of the Perceptron with the canonical form of a **linear function**. If we remove the activation function and consider only one input for clarity, the equations would be the same:

$$f(x) = ax + b \mathrel{\widehat{=}} w_i x_i + b$$

$\Rightarrow$ the bias is the b component of a linear function

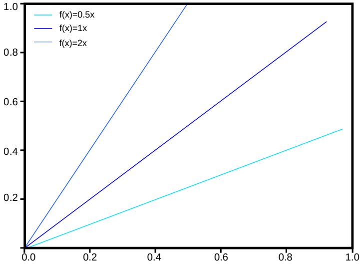
The Perceptron - Mathematical Representation

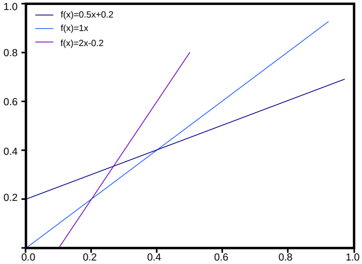
You may notice the similarity of the Perceptron with the canonical form of a **linear function**. If we remove the activation function and consider only one input for clarity, the equations would be the same:

$$f(x) = ax + b \mathrel{\widehat{=}} w_i x_i + b$$

$\Rightarrow$ the bias is the b component of a linear function

Without a bias ($b = 0$), the function will always pass through the origin [0,0]. With introducing a b, the function cuts the y-axis.





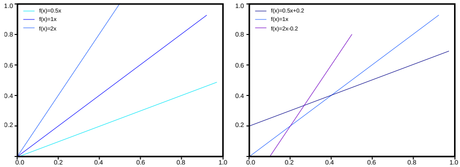
The Perceptron - Mathematical Representation

You may notice the similarity of the Perceptron with the canonical form of a **linear function**. If we remove the activation function and consider only one input for clarity, the equations would be the same:

$$f(x) = ax + b \mathrel{\widehat{=}} w_i x_i + b$$

$\Rightarrow$ the bias is the b component of a linear function

Without a bias ($b = 0$), the function will always pass through the origin [0,0]. With introducing a b, the function cuts the y-axis.

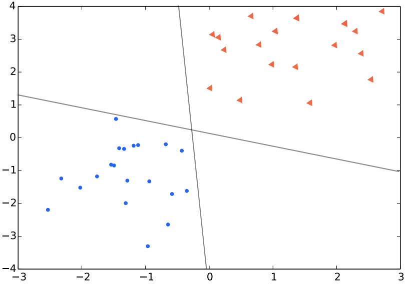


Without b we are losing **flexibility** when classifying distributions.

we will see more on this in our second `Jupyter` notebook

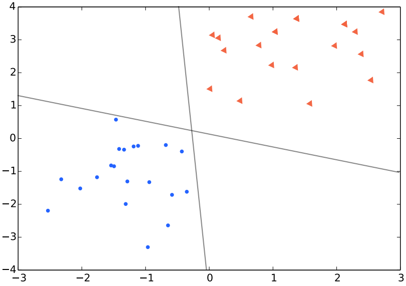
The Perceptron - Limitations, Convergence

Single-layer perceptrons are **linear classifiers**: only capable of learning (classifying) linearly separable data sets.



The Perceptron - Limitations, Convergence

Single-layer perceptrons are **linear classifiers**: only capable of learning (classifying) linearly separable data sets.

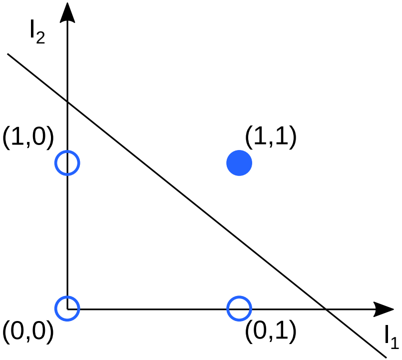


If the training set is not linearly separable, no approximate solution will be gradually approached under the standard learning algorithm - instead, learning will fail completely.

If the training set is linearly separable, then the perceptron is guaranteed to converge. There is an upper bound on the number of times the perceptron will adjust its weights during the training (Novikoff (1962)).

Perceptrons can **implement Logic Gates** like AND, OR, or NAND

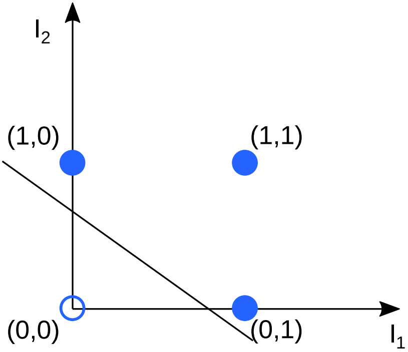
AND		
I_1	I_2	out
0	0	0
0	1	0
1	0	0
1	1	1



The Perceptron - Limitations, Convergence

Perceptrons can **implement Logic Gates** like AND, OR, or NAND

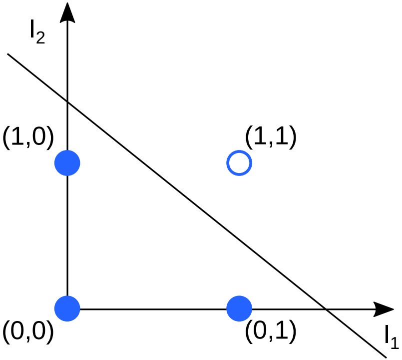
OR		
I_1	I_2	out
0	0	0
0	1	1
1	0	1
1	1	1



The Perceptron - Limitations, Convergence

Perceptrons can **implement Logic Gates** like AND, OR, or NAND

NAND		
I_1	I_2	**out**
0	0	1
0	1	1
1	0	1
1	1	0

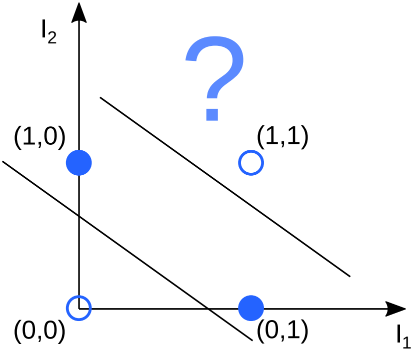


we will see how this works in our second `Jupyter` notebook

The Perceptron - Limitations, Convergence

Perceptrons can **implement Logic Gates** like AND, OR, or NAND
but not XOR

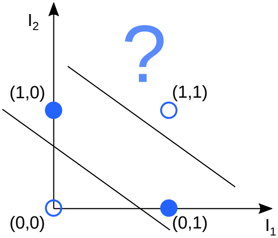
XOR		
$\mathbf{I_1}$	$\mathbf{I_2}$	**out**
0	0	0
0	1	1
1	0	1
1	1	0



The Perceptron - Limitations, Convergence

Perceptrons can **implement Logic Gates** like AND, OR, or NAND
but not XOR

XOR		
I_1	I_2	out
0	0	0
0	1	1
1	0	1
1	1	0

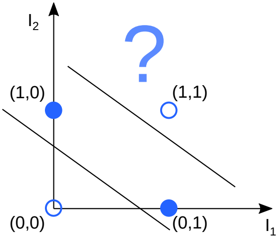


Since the XOR function is not linearly separable, it really is impossible for a single hyperplane to separate it. Single perceptrons can only learn linearly separable problems such as boolean AND problem.

The Perceptron - Limitations, Convergence

Perceptrons can **implement Logic Gates** like AND, OR, or NAND
but not XOR

XOR		
I_1	I_2	out
0	0	0
0	1	1
1	0	1
1	1	0

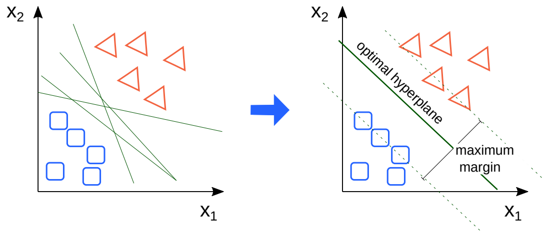


Perceptron units can be combined to form bigger Artificial Neural Network architectures.

The Perceptron - Limitations, Convergence

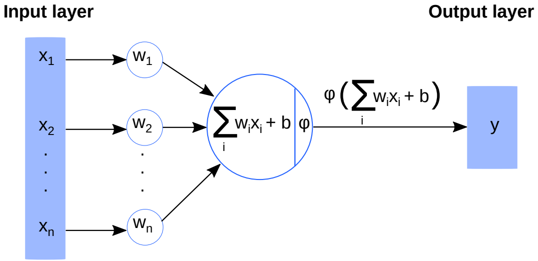
A **second layer** of perceptrons is sufficient to solve a lot of otherwise non-separable problems. More nodes can create more dividing lines, but those lines must somehow be combined to form more complex classifications.

While the perceptron algorithm is guaranteed to converge on some solution in the case of a linearly separable training set, it may still pick any solution and problems may admit many solutions of varying quality. The perceptron of optimal stability, nowadays better known as the **linear support-vector machine**, was designed to solve this problem (Krauth and Mezard, 1987).



Neural Network Architectures (I)

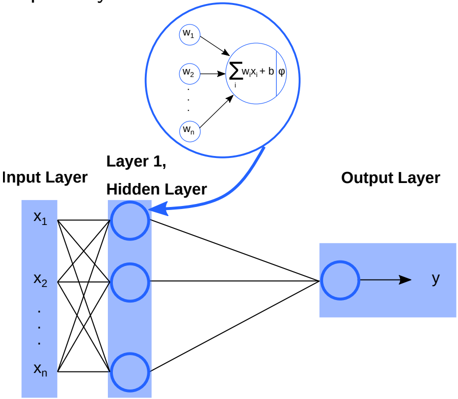
The Perceptron forms a **single-layer neural network**.



Perceptron units can be combined in **layers** to form bigger Artificial Neural Network architectures.

Neural Network Architectures (I)

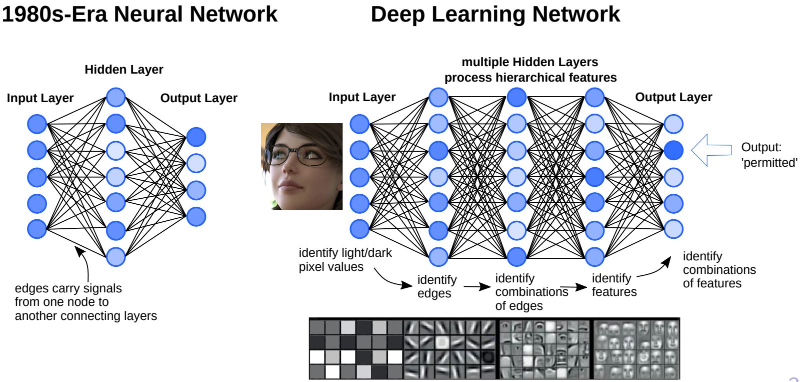
The **2-Layer Perceptron** consists of only an Input Layer, a Hidden Layer and an Output Layer.



By convention, the input layer is considered as the zero-th layer, which is why this Perceptron is considered two-layered. The Hidden Layer's computed activations are hidden from sight, as only the values from the Input Layer and Output Layer are observed.

Neural Network Architectures (I)

In contrast to **shallow neural networks** like the Perceptron (input, output, at most one hidden layer), modern **Deep-Learning Networks** can have dozens to hundreds of layers. Each layer trains on a distinct set of features based on the previous layer's output with increasing complexity since they aggregate and recombine features from the previous layer.

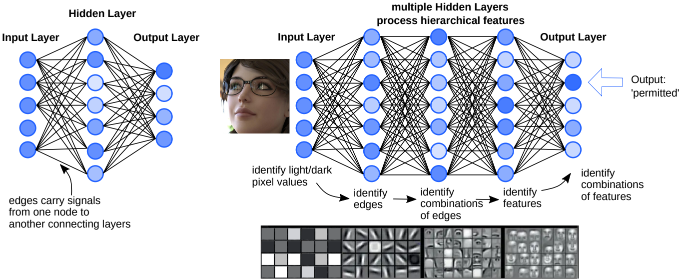


Neural Network Architectures (I)

In contrast to **shallow neural networks** like the Perceptron (input, output, at most one hidden layer), modern **Deep-Learning Networks** can have dozens to hundreds of layers. Each layer trains on a distinct set of features based on the previous layer's output with increasing complexity since they aggregate and recombine features from the previous layer.

feature hierarchy

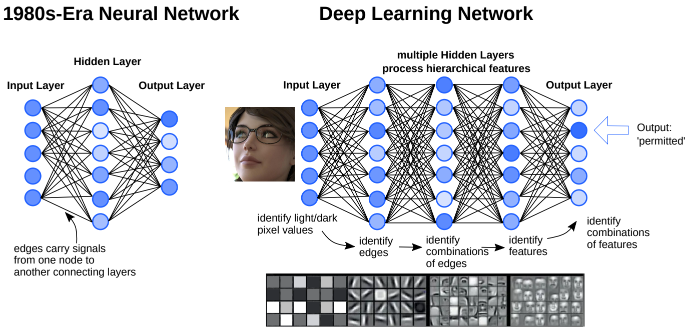
1980s-Era Neural Network **Deep Learning Network**



Neural Network Architectures (I)

Unlike most traditional machine-learning algorithms, deep-learning networks perform **automatic feature extraction** without human intervention.

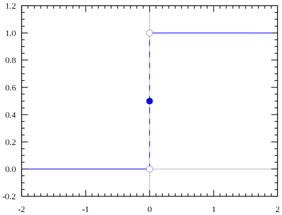
⇨ ideal to structure large data sets with hard to define features (e.g.: images, videos, sound)



An Outlook: Learning in Neural Networks

What about this series of perceptrons stacked in a row and piled in different layers? How does the model **learn**?

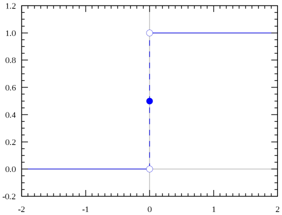
with the perceptron, the output is just a step function: 0 or 1 due to the Heaviside activation function

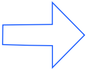


An Outlook: Learning in Neural Networks

What about this series of perceptrons stacked in a row and piled in different layers? How does the model **learn**?

with the perceptron, the output is just a step function: 0 or 1 due to the Heaviside activation function



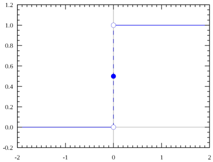


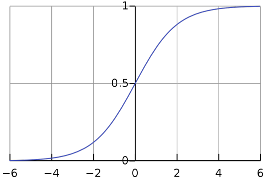
this makes a network of perceptrons hard to train, as **improvements cannot be made incrementally**

An Outlook: Learning in Neural Networks

What about this series of perceptrons stacked in a row and piled in different layers? How does the model **learn**?

with the perceptron, the output is just a step function: 0 or 1 due to the Heaviside activation function





instead: **different activation functions** can be used providing a **gradual transition between 0 and 1**
a common activation function: the sigmoid function

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