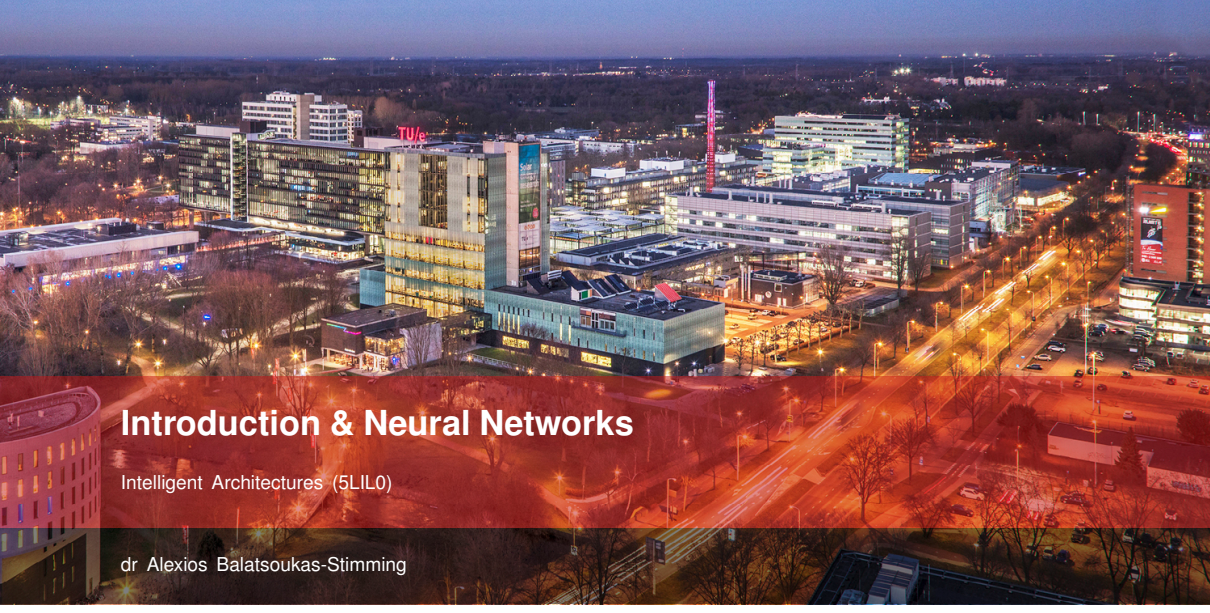


Introduction & Neural Networks

Intelligent Architectures (5LIL0)

dr Alexios Balatsoukas-Stimming

Department of Electrical Engineering, Electronic Systems Group

TU/e EINDHOVEN UNIVERSITY OF TECHNOLOGY

Intelligent Architectures (5LIL0)

Efficient deep learning algorithms and hardware architectures.

“Efficient...”

Efficiency

“Efficient...”

Efficiency

The ratio of useful output O to total input I:

$$E = \frac{O}{I}$$

“Efficient...”

Efficiency

The ratio of useful output O to total input I:

$$E = \frac{O}{I}$$

Efficiency examples:

“Efficient...”

Efficiency

The ratio of useful output O to total input I:

$$E = \frac{O}{I}$$

Efficiency examples:

1. Automotive: km/L
2. Lighting: lm/W
3. Communications: bps/Hz
4. Computing: pJ/operation

Is Improved Efficiency Always Beneficial?

Jevons Paradox (William S. Jevons, 1865)

Increases in efficiency with which a resource is used tend to increase the rate of consumption of that resource due to increasing demand.

Is Improved Efficiency Always Beneficial?

Jevons Paradox (William S. Jevons, 1865)

Increases in efficiency with which a resource is used tend to increase the rate of consumption of that resource due to increasing demand.

Examples:

1. Coal use (Jevons' original observation)

Is Improved Efficiency Always Beneficial?

Jevons Paradox (William S. Jevons, 1865)

Increases in efficiency with which a resource is used tend to increase the rate of consumption of that resource due to increasing demand.

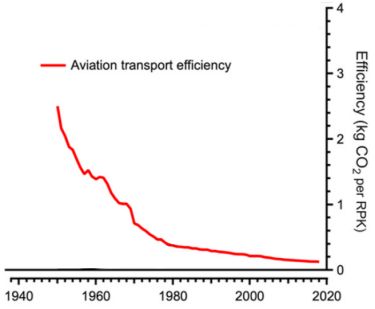
Examples:

1. Coal use (Jevons' original observation)
2. LED lights

Is Improved Efficiency Always Beneficial?

Jevons Paradox (William S. Jevons, 1865)

Increases in efficiency with which a resource is used tend to increase the rate of consumption of that resource due to increasing demand.



Examples:

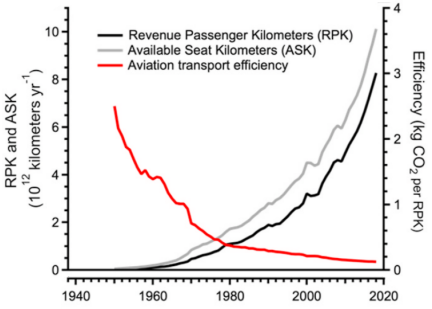
1. Coal use (Jevons' original observation)
2. LED lights
3. Air travel [1]

[1] D. S. Lee *et al.*, "The contribution of global aviation to anthropogenic climate forcing for 2000 to 2018," 2021.

Is Improved Efficiency Always Beneficial?

Jevons Paradox (William S. Jevons, 1865)

Increases in efficiency with which a resource is used tend to increase the rate of consumption of that resource due to increasing demand.



Examples:

1. Coal use (Jevons' original observation)
2. LED lights
3. Air travel [1]

[1] D. S. Lee *et al.*, "The contribution of global aviation to anthropogenic climate forcing for 2000 to 2018," 2021.

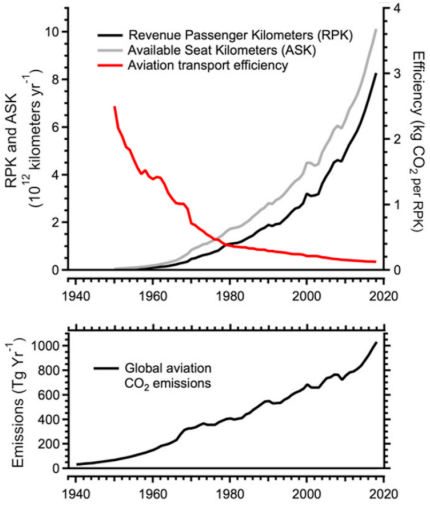
Is Improved Efficiency Always Beneficial?

Jevons Paradox (William S. Jevons, 1865)

Increases in efficiency with which a resource is used tend to increase the rate of consumption of that resource due to increasing demand.

Examples:

1. Coal use (Jevons' original observation)
2. LED lights
3. Air travel [1]



[1] D. S. Lee *et al.*, "The contribution of global aviation to anthropogenic climate forcing for 2000 to 2018," 2021.

Is Improved Efficiency Always Beneficial?

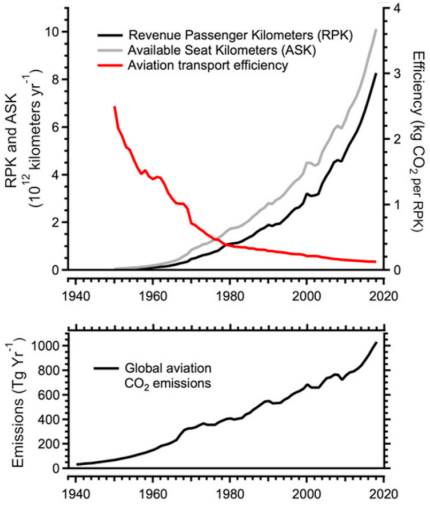
Jevons Paradox (William S. Jevons, 1865)

Increases in efficiency with which a resource is used tend to increase the rate of consumption of that resource due to increasing demand.

Examples:

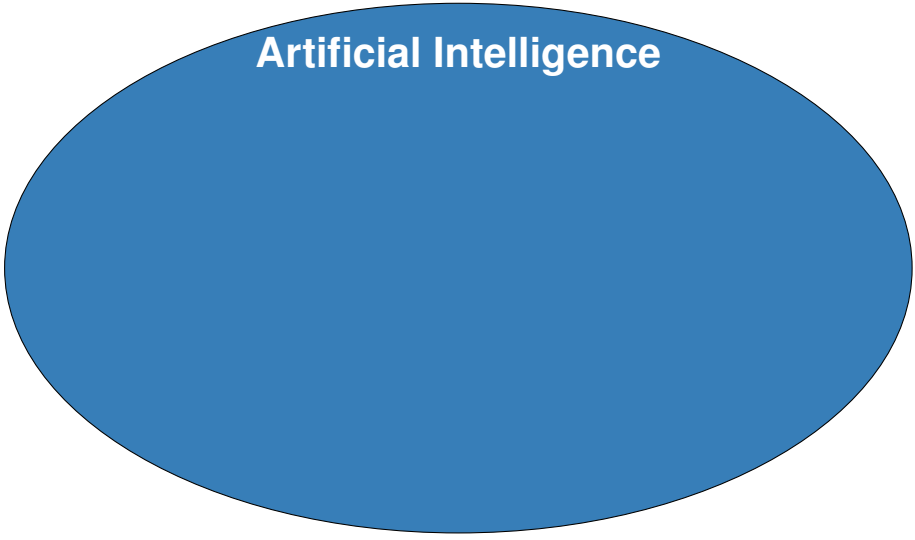
1. Coal use (Jevons' original observation)
2. LED lights
3. Air travel [1]

So: Don't look at energy efficiency in isolation!

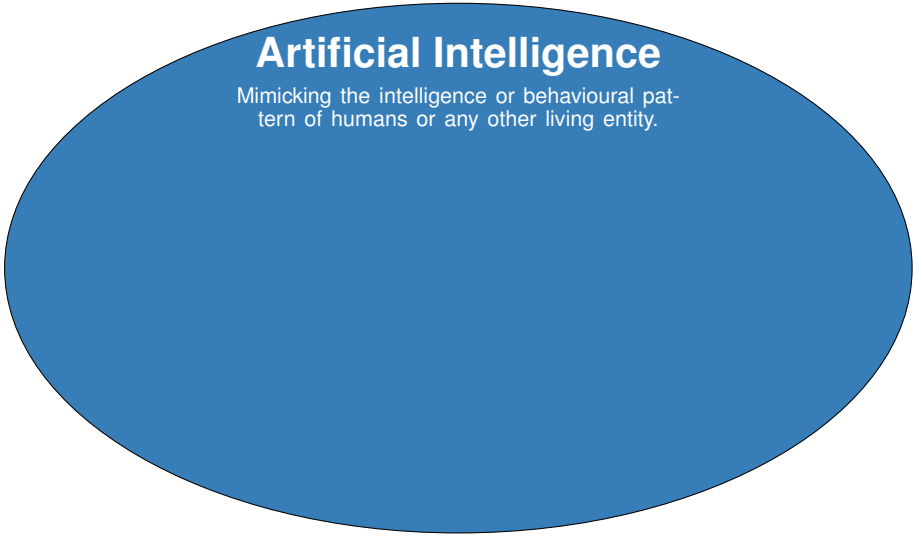


[1] D. S. Lee *et al.*, "The contribution of global aviation to anthropogenic climate forcing for 2000 to 2018," 2021.

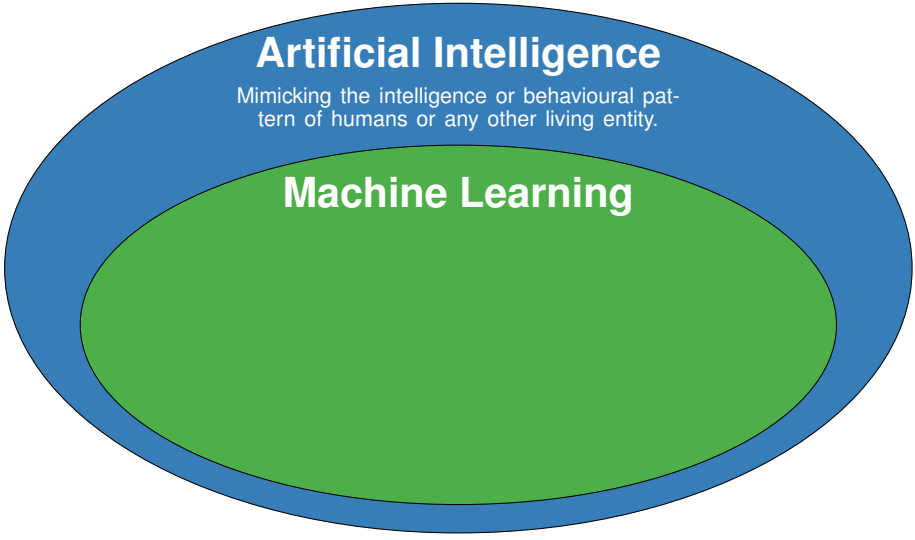
“...deep learning algorithms...”



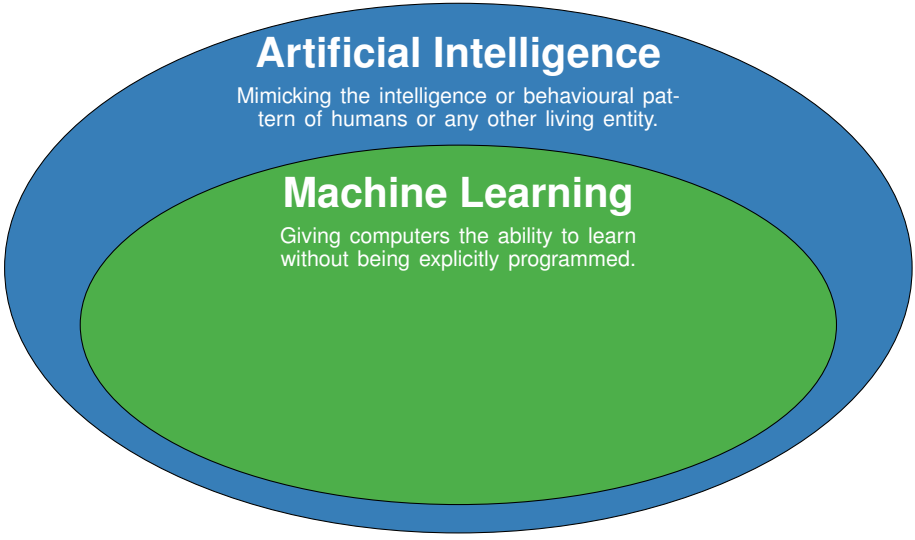
“...deep learning algorithms...”



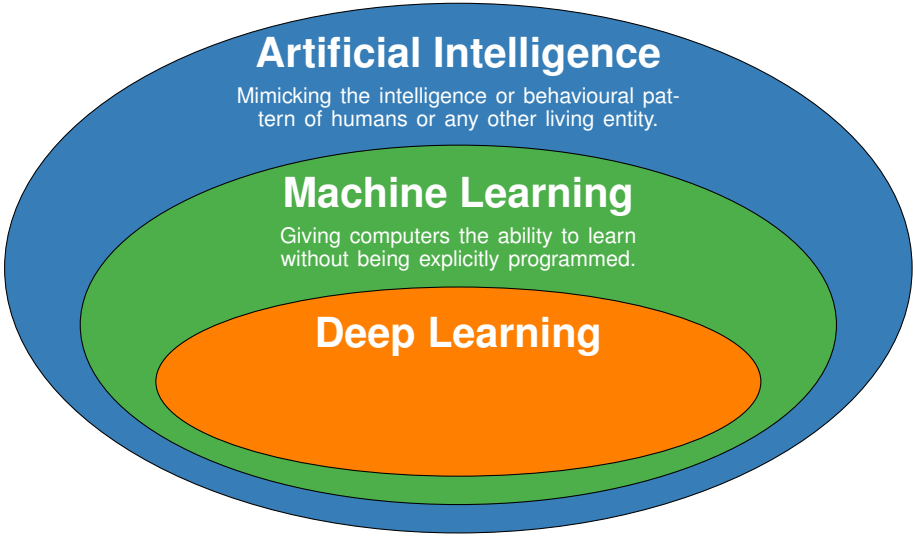
“...deep learning algorithms...”



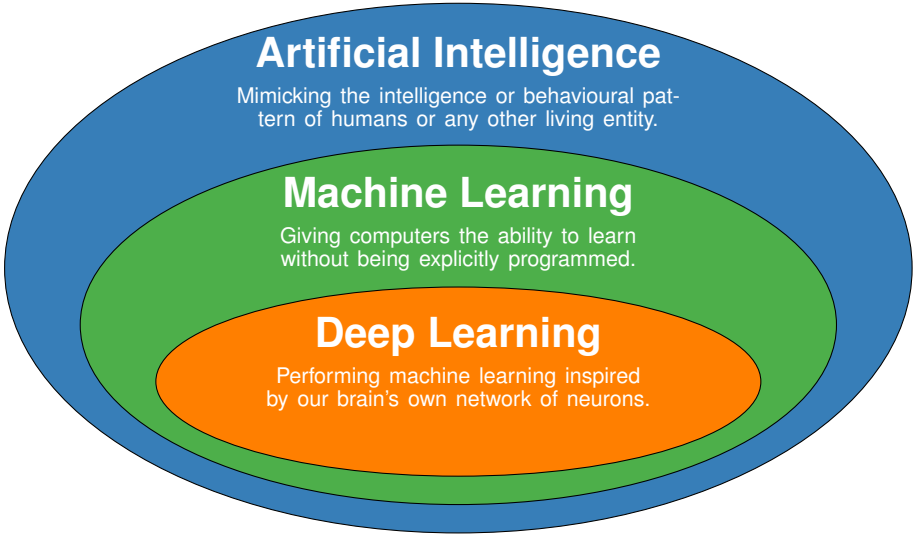
“...deep learning algorithms...”



“...deep learning algorithms...”



“...deep learning algorithms...”



Types of Learning

1. **Supervised learning:** learns from labeled datasets.
   - **Example:** Camera-based object detection for self-driving cars.

Types of Learning

1. **Supervised learning:** learns from labeled datasets.
   - **Example:** Camera-based object detection for self-driving cars.

2. **Unsupervised learning:** discovers patterns in unlabeled datasets.
   - **Example:** Network intrusion detection via anomalies.

Types of Learning

1. **Supervised learning:** learns from labeled datasets.
   - **Example:** Camera-based object detection for self-driving cars.

2. **Unsupervised learning:** discovers patterns in unlabeled datasets.
   - **Example:** Network intrusion detection via anomalies.

3. **Reinforcement learning:** learns to act based on feedback.
   - **Example:** Learning how to play Starcraft.

Types of Learning

1. **Supervised learning:** learns from labeled datasets.
   - **Example:** Camera-based object detection for self-driving cars.

2. **Unsupervised learning:** discovers patterns in unlabeled datasets.
   - **Example:** Network intrusion detection via anomalies.

3. **Reinforcement learning:** learns to act based on feedback.
   - **Example:** Learning how to play Starcraft.

In practice, a combination of approaches is often used.

For example, ChatGPT uses both supervised learning and reinforcement learning.

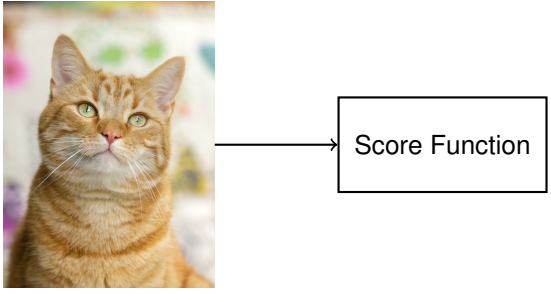
Learning Systems

Basic components of a (supervised) learning system:

Learning Systems

Basic components of a (supervised) learning system:

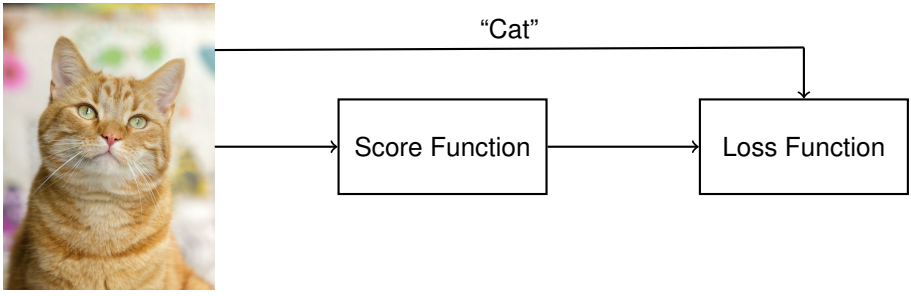
1. **Score function**



Learning Systems

Basic components of a (supervised) learning system:

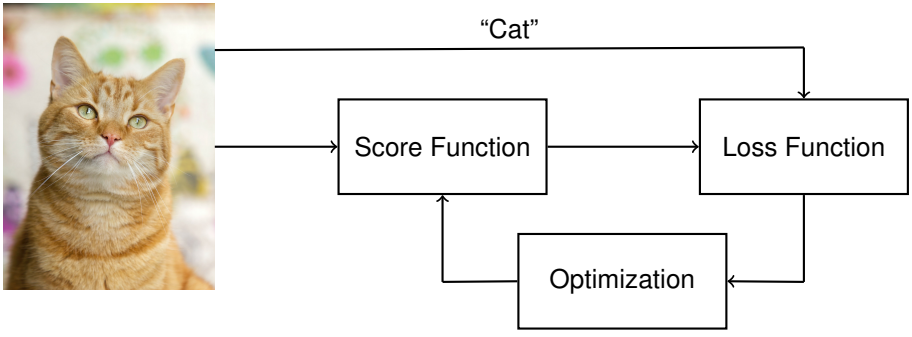
1. **Score function**
2. **Loss function**



Learning Systems

Basic components of a (supervised) learning system:

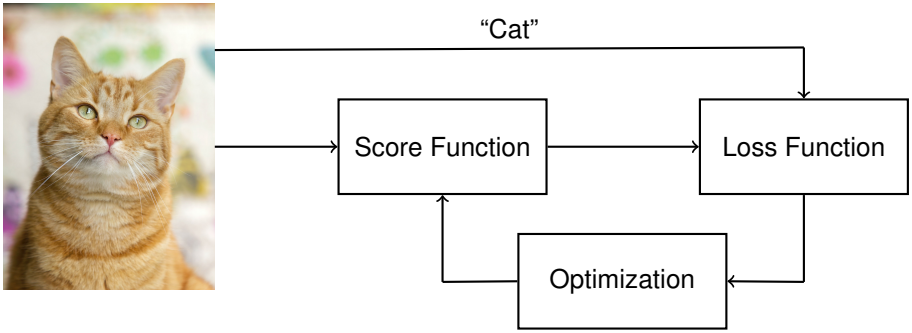
1. **Score function**
2. **Loss function**
3. **Optimization**



Learning Systems

Basic components of a (supervised) learning system:

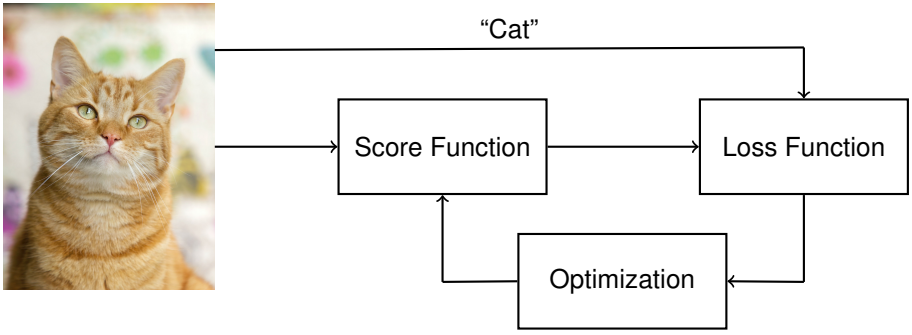
1. **Score function:** decision trees, support vector machines, neural networks, ...
2. **Loss function**
3. **Optimization**



Learning Systems

Basic components of a (supervised) learning system:

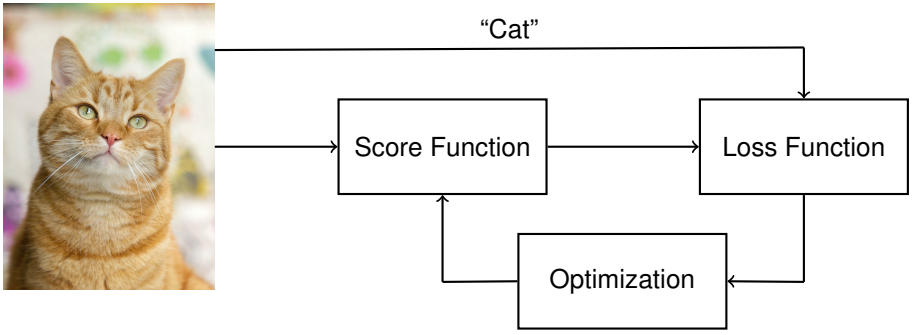
1. **Score function:** decision trees, support vector machines, neural networks, ...
2. **Loss function**
3. **Optimization**



Learning Systems

Basic components of a (supervised) learning system:

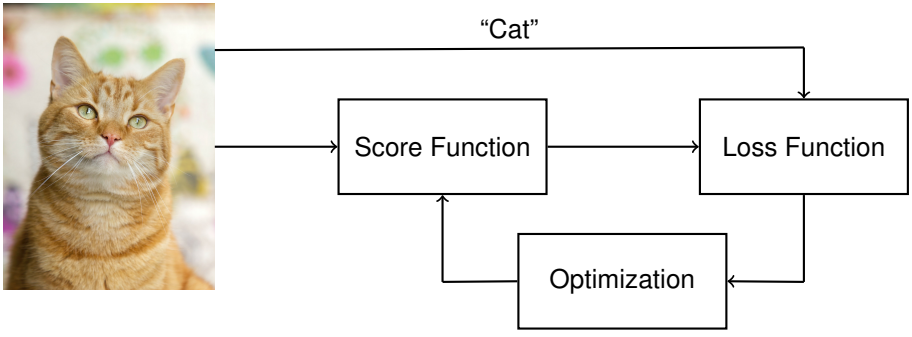
1. **Score function:** decision trees, support vector machines, neural networks, ...
2. **Loss function:** mean squared-error, binary cross-entropy, ...
3. **Optimization**



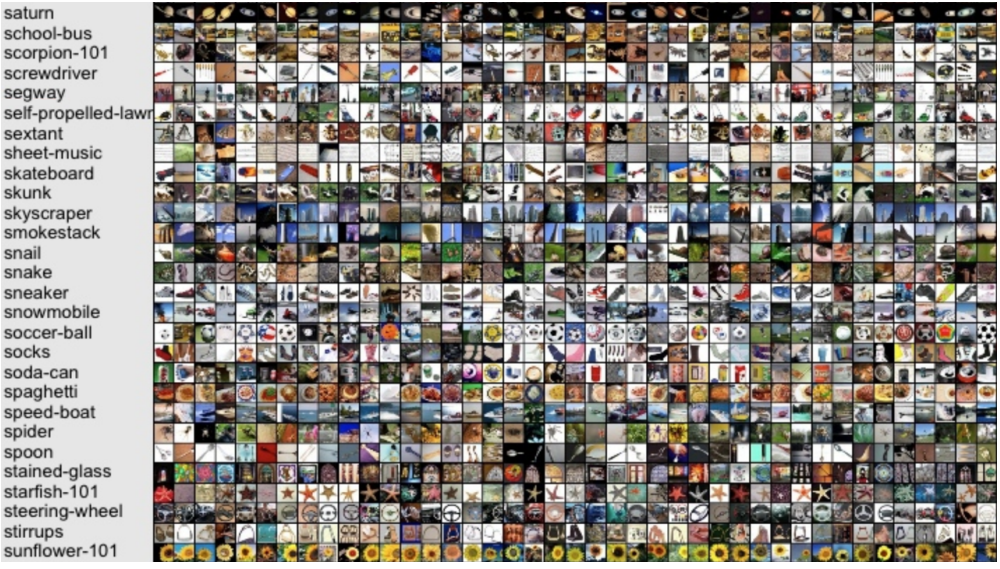
Learning Systems

Basic components of a (supervised) learning system:

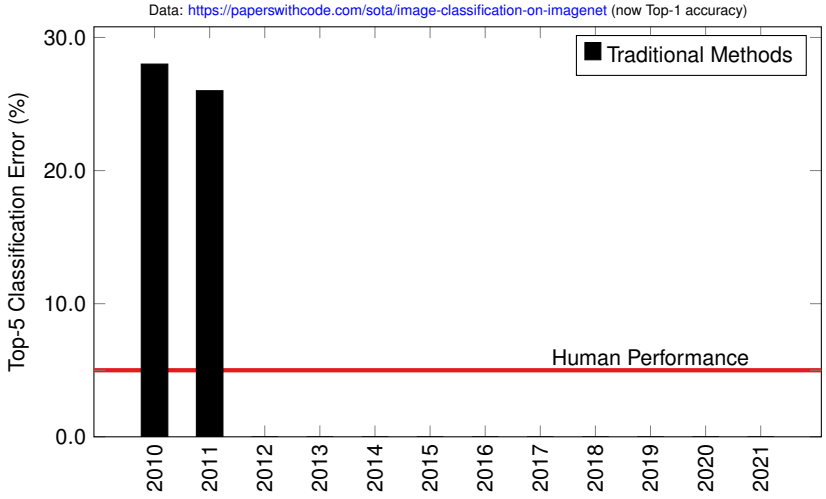
1. **Score function:** decision trees, support vector machines, neural networks, ...
2. **Loss function:** mean squared-error, binary cross-entropy, ...
3. **Optimization:** stochastic gradient descent, batch gradient descent, ...



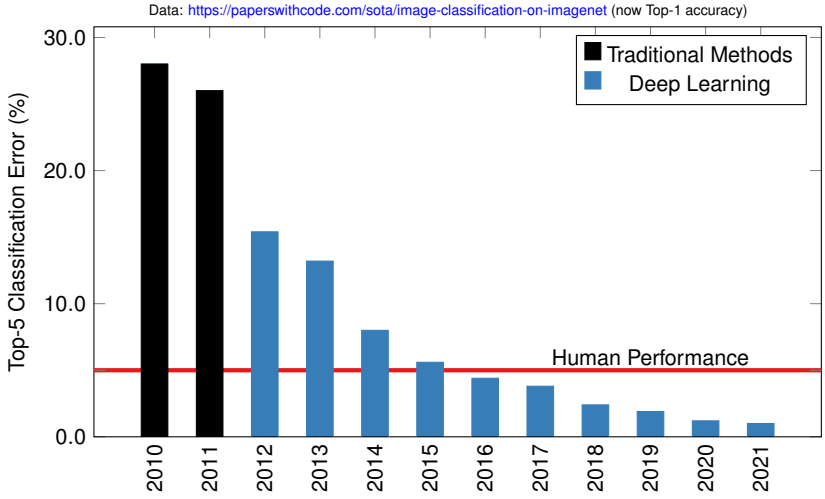
The ImageNet Challenge



The ImageNet Challenge

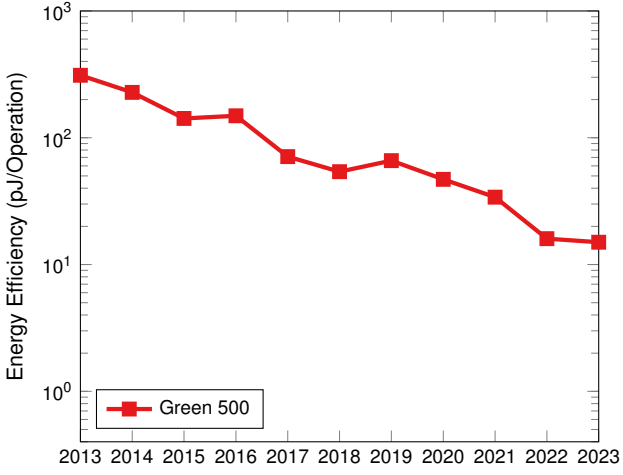


The ImageNet Challenge



“...and hardware architectures.”

Data: https://top500.org/lists/green500/, https://en.wikipedia.org/wiki/Tensor_Processing_Unit

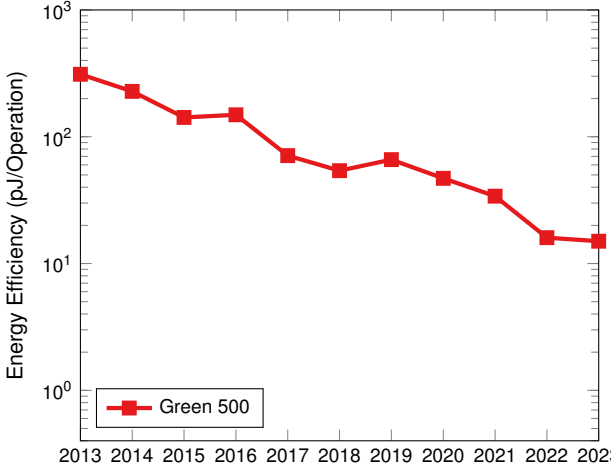


OLCF at ORNL, CC BY 2.0, Wikimedia

The Frontier supercomputer.

“...and hardware architectures.”

Data: https://top500.org/lists/green500/, https://en.wikipedia.org/wiki/Tensor_Processing_Unit



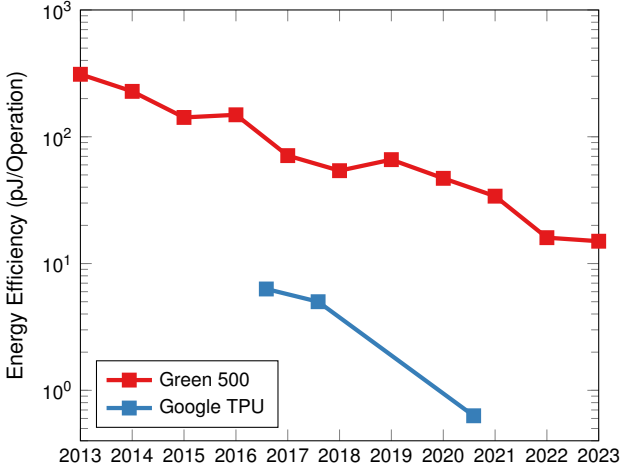
OLCF at ORNL, CC BY 2.0, Wikimedia

The Frontier supercomputer.

Energy efficiency of brain: 10^{-8} pJ/Operation!!!

“...and hardware architectures.”

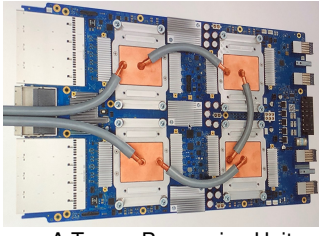
Data: https://top500.org/lists/green500/, https://en.wikipedia.org/wiki/Tensor_Processing_Unit



Energy efficiency of brain: 10^{-8} pJ/Operation!!!

OLCF at ORNL, CC BY 2.0, Wikimedia

The Frontier supercomputer.



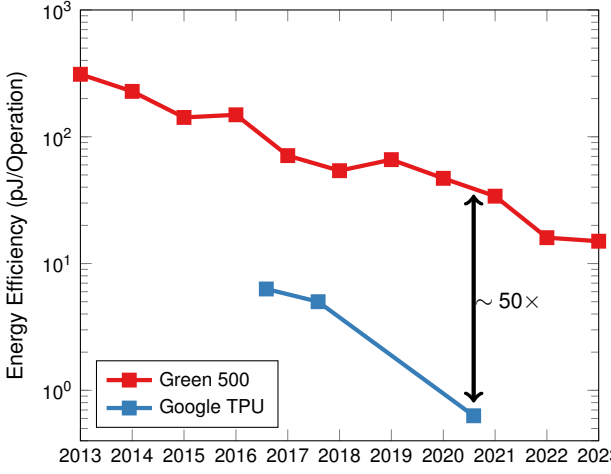
Zinskauf, CC BY-SA 4.0, Wikimedia

A Tensor Processing Unit.

TU/e

“...and hardware architectures.”

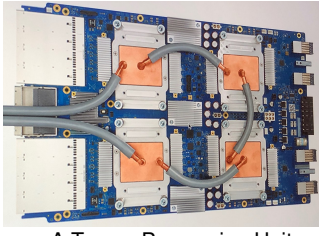
Data: https://top500.org/lists/green500/, https://en.wikipedia.org/wiki/Tensor_Processing_Unit



Energy efficiency of brain: 10^{-8} pJ/Operation!!!

OLCF at ORNL, CC BY 2.0, Wikimedia

The Frontier supercomputer.



Zinskauf, CC BY-SA 4.0, Wikimedia

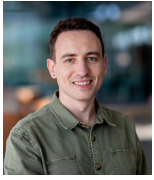
A Tensor Processing Unit.

TU/e

Course Team

Core Team

dr Alexios Balatsoukas-Stimming
Responsible lecturer



dr Federico Corradi
Co-lecturer

Zhanbo Shen
Teaching assistant

Lorenzo Pes
Teaching assistant

Guest Lecturers

Prof. Henk Corporaal
TU/e Emeritus

dr Maurice Peemen
Thermo Fischer Scientific

dr Willem Sandberg
NXP

Learning Objectives & Prerequisites

At the end of this course, you will be able to:

1. Implement deep learning methods, including inference and learning, both from scratch and using modern frameworks.

Learning Objectives & Prerequisites

At the end of this course, you will be able to:

1. Implement deep learning methods, including inference and learning, both from scratch and using modern frameworks.
2. Design deep neural network (DNN) architectures, with a focus on state-of-the-art convolutional neural networks (CNNs).

Learning Objectives & Prerequisites

At the end of this course, you will be able to:

1. Implement deep learning methods, including inference and learning, both from scratch and using modern frameworks.
2. Design deep neural network (DNN) architectures, with a focus on state-of-the-art convolutional neural networks (CNNs).
3. Optimize DNNs using various methods (e.g., quantization, pruning) and explain the trade-offs of these methods.

Learning Objectives & Prerequisites

At the end of this course, you will be able to:

1. Implement deep learning methods, including inference and learning, both from scratch and using modern frameworks.
2. Design deep neural network (DNN) architectures, with a focus on state-of-the-art convolutional neural networks (CNNs).
3. Optimize DNNs using various methods (e.g., quantization, pruning) and explain the trade-offs of these methods.
4. Evaluate special processing architectures and dedicated hardware for efficient deep learning.

Learning Objectives & Prerequisites

At the end of this course, you will be able to:

1. Implement deep learning methods, including inference and learning, both from scratch and using modern frameworks.
2. Design deep neural network (DNN) architectures, with a focus on state-of-the-art convolutional neural networks (CNNs).
3. Optimize DNNs using various methods (e.g., quantization, pruning) and explain the trade-offs of these methods.
4. Evaluate special processing architectures and dedicated hardware for efficient deep learning.

Prerequisites:

1. Basic linear algebra.
2. Basic differential calculus.
3. Very good programming skills (ideally in Python).

Course Organization

- **Lectures:** Tuesdays 10:45-12:30 & Fridays 15:30-17:15
- **TA hours:** Tuesdays 12:30-13:30 (Flux 4.076) & Thursdays 12:30-13:30 (Flux 4.078)

Course Organization

- **Lectures:** Tuesdays 10:45-12:30 & Fridays 15:30-17:15
- **TA hours:** Tuesdays 12:30-13:30 (Flux 4.076) & Thursdays 12:30-13:30 (Flux 4.078)

- **Assessment:**
  - 70% written final exam (includes material from lectures & assignments)
  - 30% assignments (group code submission and **individual single-attempt** quizzes)
  - Minimum of 5 in each component.

Course Organization

- **Lectures:** Tuesdays 10:45-12:30 & Fridays 15:30-17:15
- **TA hours:** Tuesdays 12:30-13:30 (Flux 4.076) & Thursdays 12:30-13:30 (Flux 4.078)

- **Assessment:**
  - 70% written final exam (includes material from lectures & assignments)
  - 30% assignments (group code submission and **individual single-attempt** quizzes)
  - Minimum of 5 in each component.

- **Assignments:**
  - **Assignment 1 (Basics):** Neural network inference and learning.
  - **Assignment 2 (Optimizations):** Quantization, pruning, neural architecture search.
  - **Assignment 3 (Platforms):** Hardware accelerators.

Course Organization

- **Lectures:** Tuesdays 10:45-12:30 & Fridays 15:30-17:15
- **TA hours:** Tuesdays 12:30-13:30 (Flux 4.076) & Thursdays 12:30-13:30 (Flux 4.078)

- **Assessment:**
  - 70% written final exam (includes material from lectures & assignments)
  - 30% assignments (group code submission and **individual single-attempt** quizzes)
  - Minimum of 5 in each component.

- **Assignments:**
  - **Assignment 1 (Basics):** Neural network inference and learning.
  - **Assignment 2 (Optimizations):** Quantization, pruning, neural architecture search.
  - **Assignment 3 (Platforms):** Hardware accelerators.

- **Course material:** Canvas.

Use of Generative AI

Not allowed

Using AI to **create content** like images, source code, video, text or any other kind.

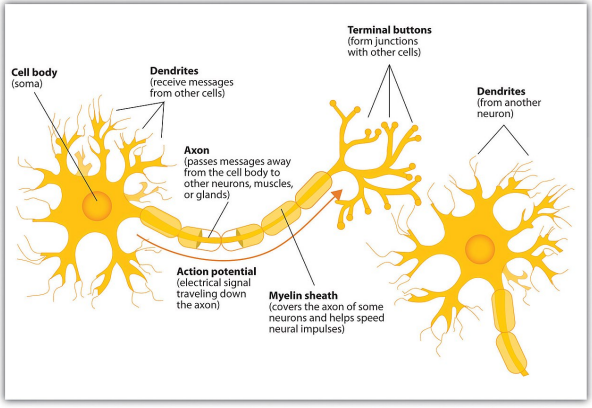
Allowed

Use of AI-powered tools to improve **your own** content that operate at the level of at most a single sentence (e.g. spelling checkers) or a single line of code (as provided by many IDEs).

Time For A Break

> "*Our civilization is first and foremost a civilization of means; in the reality of modern life, the means, it would seem, are more important than the ends.*"
>
> – Jacques Ellul, "The Technological Society," 1954.

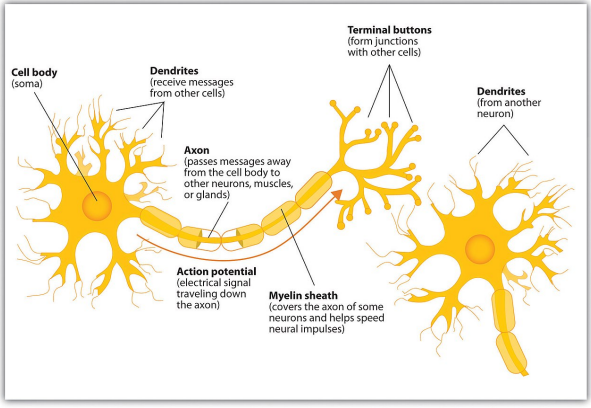
Our Brain



The basic computational unit of the brain is a **neuron**.

- Neurons receive input signals from **dendrites** and produce output signals along their **axon**, which interact with the dendrites of other neurons via **synaptic weights**.

Our Brain

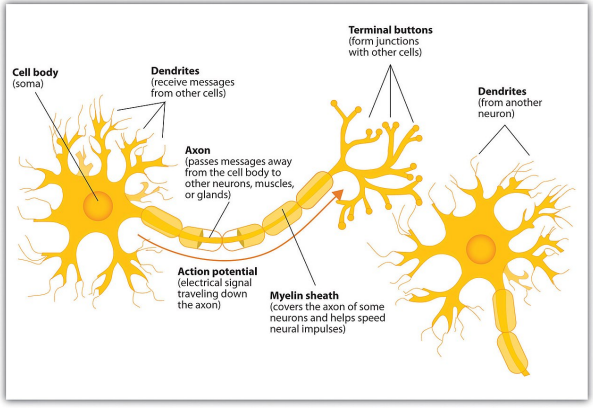


Jennifer Walinga, CC BY-SA 4.0, via Wikimedia Commons

The basic computational unit of the brain is a **neuron**.

- Neurons receive input signals from **dendrites** and produce output signals along their **axon**, which interact with the dendrites of other neurons via **synaptic weights**.
- About 80 billion neurons.
- About 10^{14} to 10^{15} synapses.

Our Brain



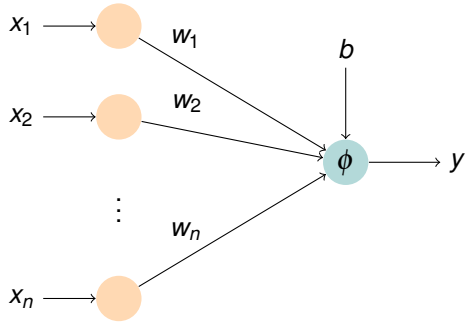
Jennifer Walinga, CC BY-SA 4.0, via Wikimedia Commons

The basic computational unit of the brain is a **neuron**.

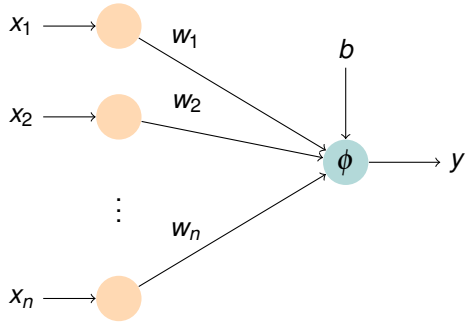
- Neurons receive input signals from **dendrites** and produce output signals along their **axon**, which interact with the dendrites of other neurons via **synaptic weights**.
- About 80 billion neurons.
- About 10^{14} to 10^{15} synapses.

The synaptic weights are **learnable** and they control the influence between neurons.

An Artificial Neuron

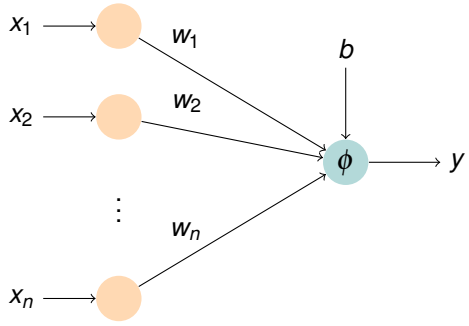


An Artificial Neuron



$$y = \phi\left(\sum_{i=1}^{n} w_i x_i + b\right) = \phi\left(\mathbf{w}^\top \mathbf{x} + b\right)$$

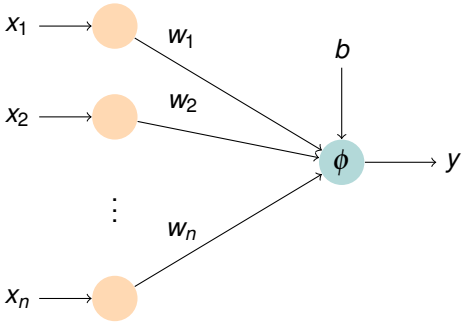
An Artificial Neuron



- **Input** nodes correspond to dendrites.

$$y = \phi\left(\sum_{i=1}^{n} w_i x_i + b\right) = \phi\left(\mathbf{w}^\top \mathbf{x} + b\right)$$

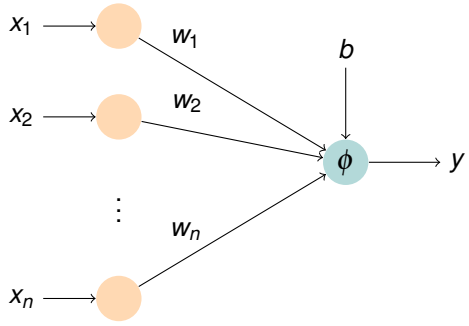
An Artificial Neuron



- **Input** nodes correspond to dendrites.
- **Weights** w_i correspond to synaptic weights.

$$y = \phi\left(\sum_{i=1}^{n} w_i x_i + b\right) = \phi\left(\mathbf{w}^\top \mathbf{x} + b\right)$$

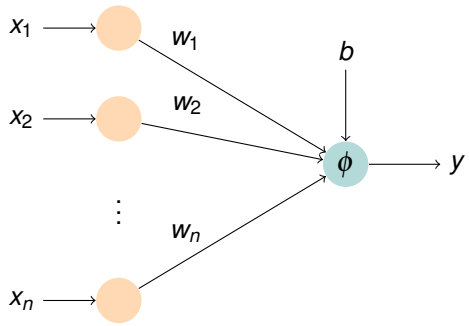
An Artificial Neuron



$$y = \phi\left(\sum_{i=1}^{n} w_i x_i + b\right) = \phi\left(\mathbf{w}^\top \mathbf{x} + b\right)$$

- **Input** nodes correspond to dendrites.
- **Weights** w_i correspond to synaptic weights.
- **Hidden** node corresponds to soma-axon interaction:
  1. Incoming potentials and bias b are added up.
  2. A non-linear function $\phi(\cdot)$ is applied.

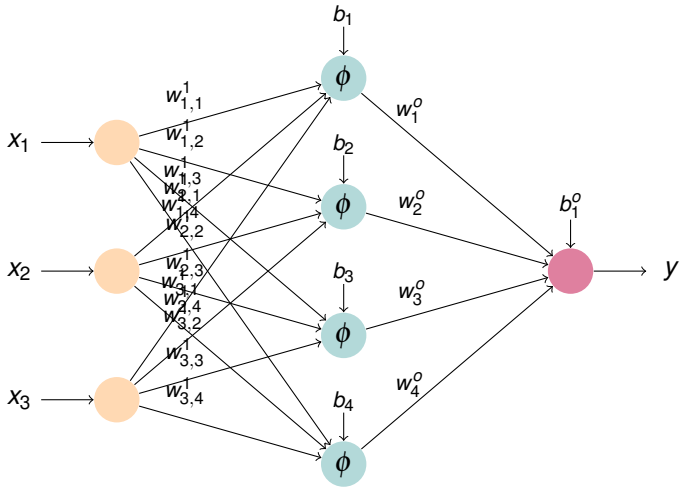
An Artificial Neuron



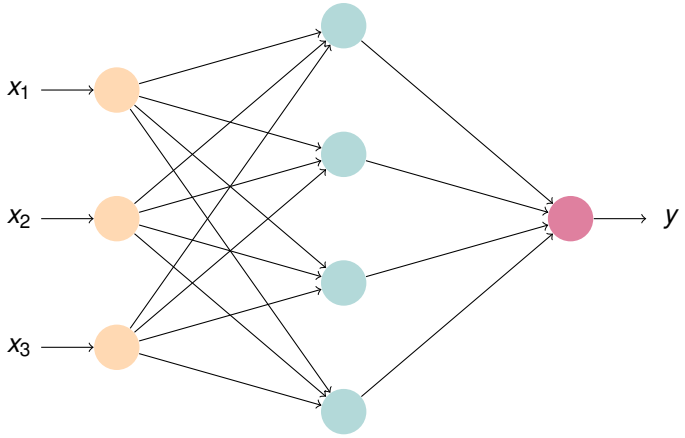
$$y = \phi\left(\sum_{i=1}^{n} w_i x_i + b\right) = \phi\left(\mathbf{w}^\top \mathbf{x} + b\right)$$

- **Input** nodes correspond to dendrites.
- **Weights** w_i correspond to synaptic weights.
- **Hidden** node corresponds to soma-axon interaction:
  1. Incoming potentials and bias b are added up.
  2. A non-linear function $\phi(\cdot)$ is applied.
- **Output** y corresponds to action potential on axon.

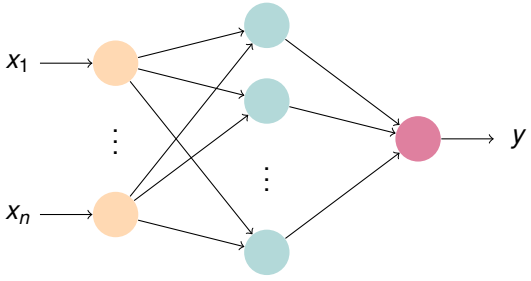
A Single-Layer Single-Output Neural Network



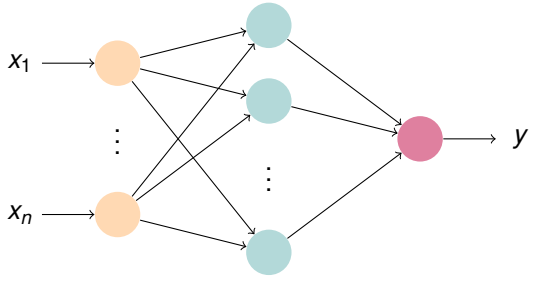
A Single-Layer Single-Output Neural Network



A Single-Layer Single-Output Neural Network

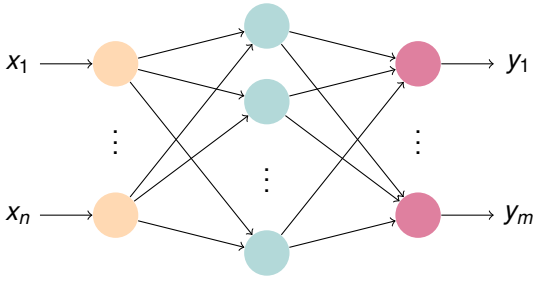


A Single-Layer Single-Output Neural Network



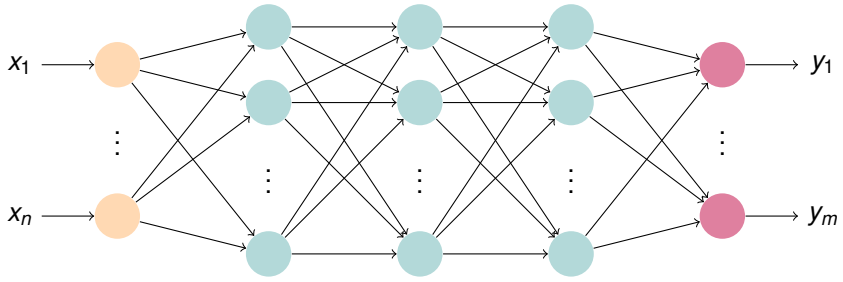
$$y = \left(\mathbf{w}^{[2]}\right)^{\top} \phi\left(\mathbf{W}^{[1]}\mathbf{x} + \mathbf{b}^{[1]}\right) + b^{[2]}$$

A Single-Layer Multi-Output Neural Network



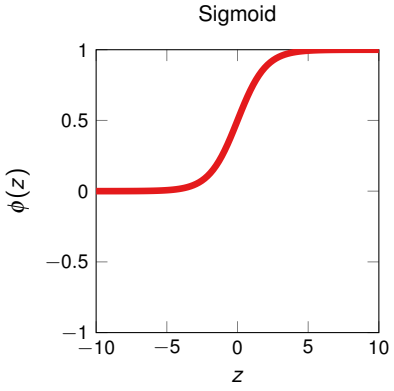
$$\mathbf{y} = \mathbf{W}^{[2]}\phi\left(\mathbf{W}^{[1]}\mathbf{x} + \mathbf{b}^{[1]}\right) + \mathbf{b}^{[2]}$$

A Multi-Layer Multi-Output Neural Network



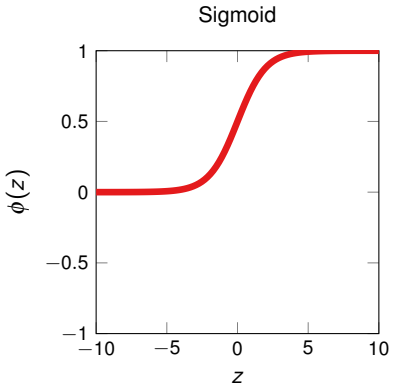
Let $f^{[i]}(\mathbf{z}) = \phi\left(\mathbf{W}^{[i]}\mathbf{z} + \mathbf{b}^{[i]}\right)$. Then, we have $\mathbf{y} = f^{[4]}\left(f^{[3]}\left(f^{[2]}\left(f^{[1]}(\mathbf{x})\right)\right)\right)$

Traditional Activation Functions

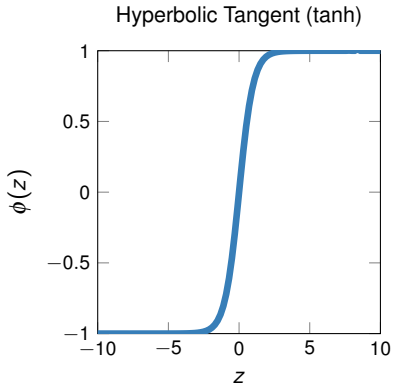


$$\phi(z) = \frac{1}{1+e^{-z}} = \sigma(z)$$

Traditional Activation Functions

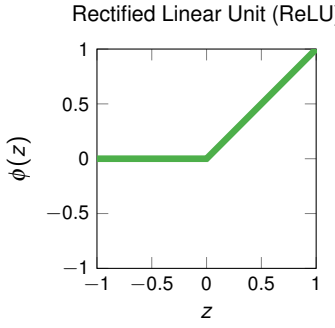


$$\phi(z) = \frac{1}{1+e^{-z}} = \sigma(z)$$



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

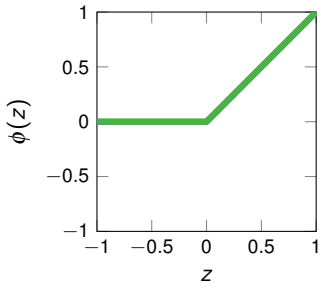
Modern Activation Functions



$$\phi(z) = \max(0, z)$$

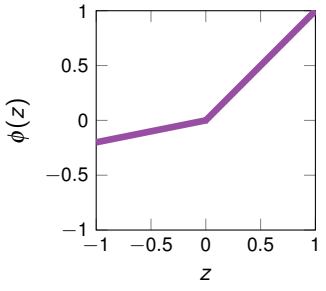
Modern Activation Functions

Rectified Linear Unit (ReLU)



$$\phi(z) = \max(0, z)$$

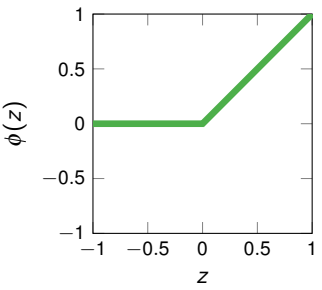
Leaky ReLU ($\alpha = 0.2$)



$$\phi(z) = \max(\alpha z, z)$$

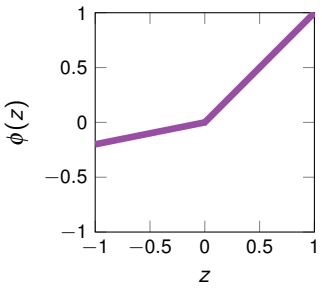
Modern Activation Functions

Rectified Linear Unit (ReLU)



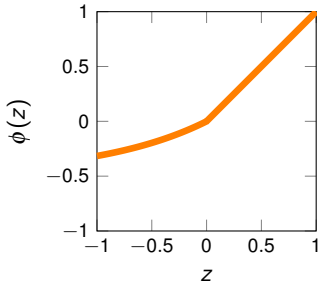
$$\phi(z) = \max(0, z)$$

Leaky ReLU ($\alpha = 0.2$)



$$\phi(z) = \max(\alpha z, z)$$

Exponential Linear Unit (ELU, $\alpha = 0.5$)



$$\phi(z) = \begin{cases} \alpha(e^z - 1), & z < 0, \\ z, & z \geq 0. \end{cases}$$

Final Layer Activation Function

- The final layer activation function is chosen **depending on the task**.

Final Layer Activation Function

- The final layer activation function is chosen **depending on the task**.
- For **regression**, typically a linear activation function (i.e., $\phi(z) = z$).

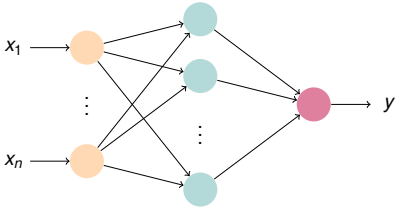
Final Layer Activation Function

- The final layer activation function is chosen **depending on the task**.
- For **regression**, typically a linear activation function (i.e., $\phi(z) = z$).
- For **classification**:

Final Layer Activation Function

- The final layer activation function is chosen **depending on the task**.
- For **regression**, typically a linear activation function (i.e., $\phi(z) = z$).
- For **classification**:

Binary Classification

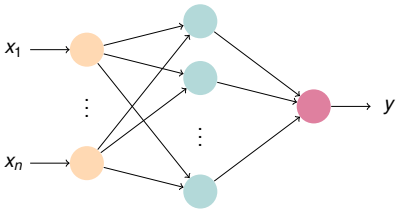


$$y = \text{sigmoid}(z)$$

Final Layer Activation Function

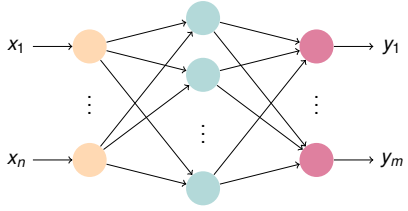
- The final layer activation function is chosen **depending on the task**.
- For **regression**, typically a linear activation function (i.e., $\phi(z) = z$).
- For **classification**:

Binary Classification



$$y = \text{sigmoid}(z)$$

Multi-class classification



$$y_i = \text{softmax}(\mathbf{z}) = \frac{e^{z_i}}{\sum_{k=1}^{m} e^{z_k}}$$

Universal Approximation Theorem

Theorem 1. *Let σ be any continuous discriminatory function. Then finite sums of the form*

$$G(x) = \sum_{j=1}^{N} \alpha_j \sigma(y_j^T x + \theta_j) \tag{2}$$

are dense in $C(I_n)$. In other words, given any $f \in C(I_n)$ and $\varepsilon > 0$, there is a sum, $G(x)$, of the above form, for which

$$|G(x) - f(x)| < \varepsilon \qquad \text{for all} \quad x \in I_n.$$

G. Cybenko "Approximation by superpositions of a sigmoidal function," Mathematics of Control, Signals and Systems, 1989.

Universal Approximation Theorem

Theorem 1. *Let σ be any continuous discriminatory function. Then finite sums of the form*

$$G(x) = \sum_{j=1}^{N} \alpha_j \sigma(y_j^T x + \theta_j) \qquad (2)$$

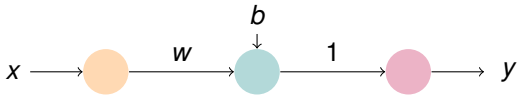
are dense in $C(I_n)$. In other words, given any $f \in C(I_n)$ and $\varepsilon > 0$, there is a sum, $G(x)$, of the above form, for which

$$|G(x) - f(x)| < \varepsilon \qquad \text{for all} \quad x \in I_n.$$

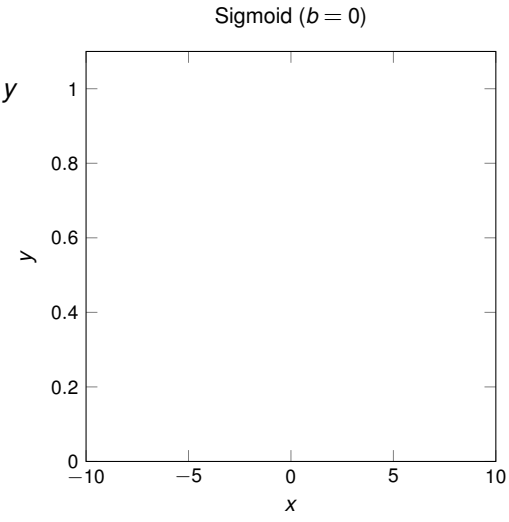
Huh?!?

G. Cybenko "Approximation by superpositions of a sigmoidal function," Mathematics of Control, Signals and Systems, 1989.

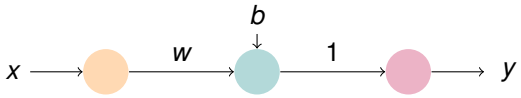
Neural Network as a Step Function



$$y = \sigma(w \cdot x + b) = \frac{1}{1 + e^{-(w \cdot x + b)}}$$

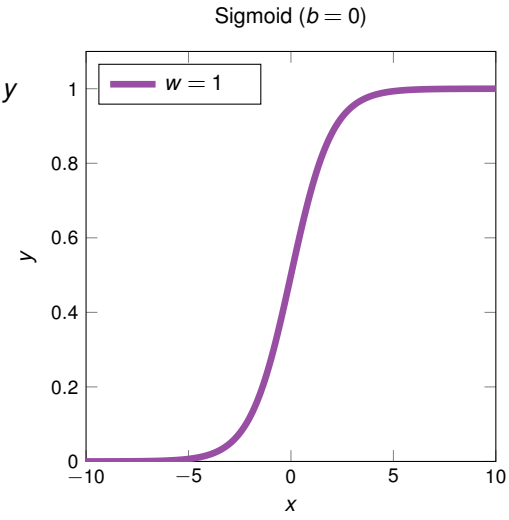


Neural Network as a Step Function

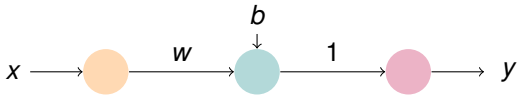


$$y = \sigma(w \cdot x + b) = \frac{1}{1 + e^{-(w \cdot x + b)}}$$

1. Assume $b = 0$ and vary w

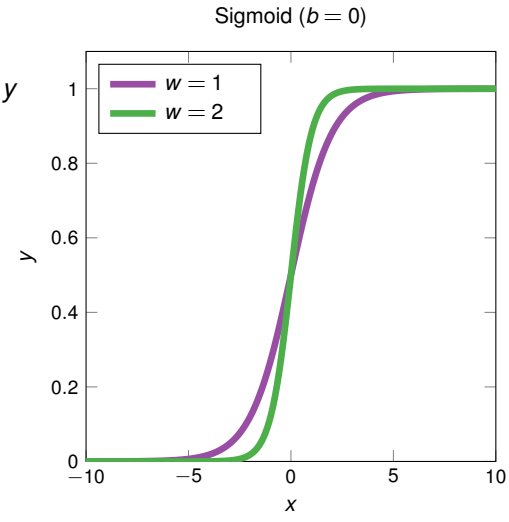


Neural Network as a Step Function

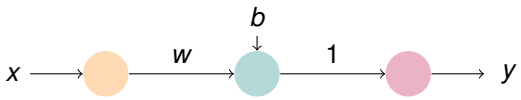


$$y = \sigma(w \cdot x + b) = \frac{1}{1 + e^{-(w \cdot x + b)}}$$

1. Assume $b = 0$ and vary w

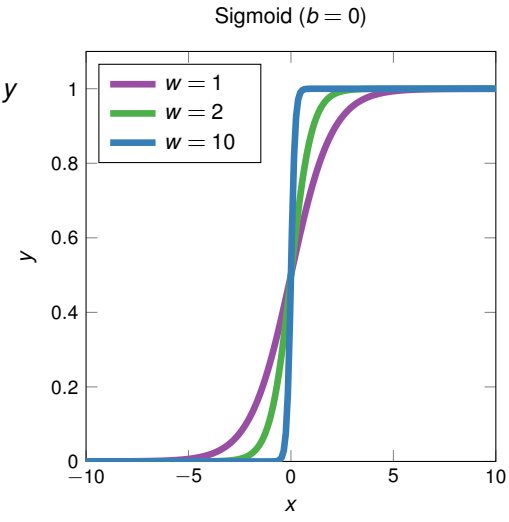


Neural Network as a Step Function

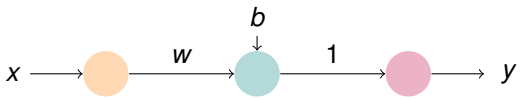


$$y = \sigma(w \cdot x + b) = \frac{1}{1 + e^{-(w \cdot x + b)}}$$

1. Assume $b = 0$ and vary w

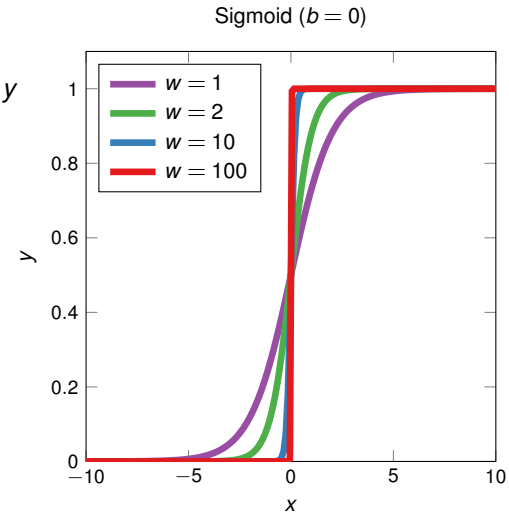


Neural Network as a Step Function

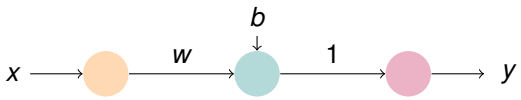


$$y = \sigma(w \cdot x + b) = \frac{1}{1 + e^{-(w \cdot x + b)}}$$

1. Assume $b = 0$ and vary w

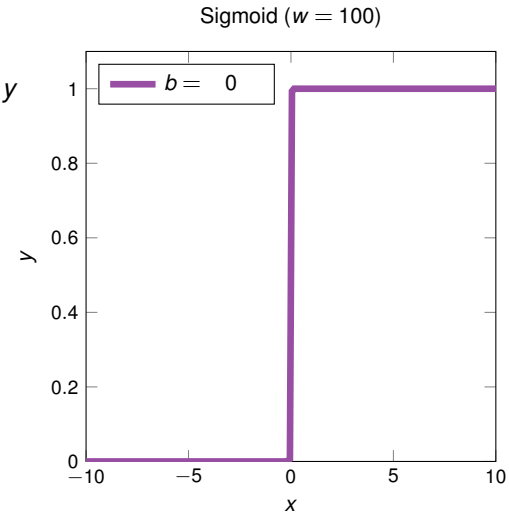


Neural Network as a Step Function

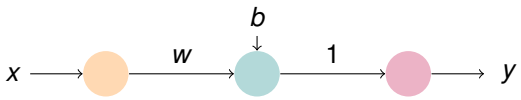


$$y = \sigma(w \cdot x + b) = \frac{1}{1 + e^{-(w \cdot x + b)}}$$

1. Assume $b = 0$ and vary w
2. Assume $w = 100$ and vary b

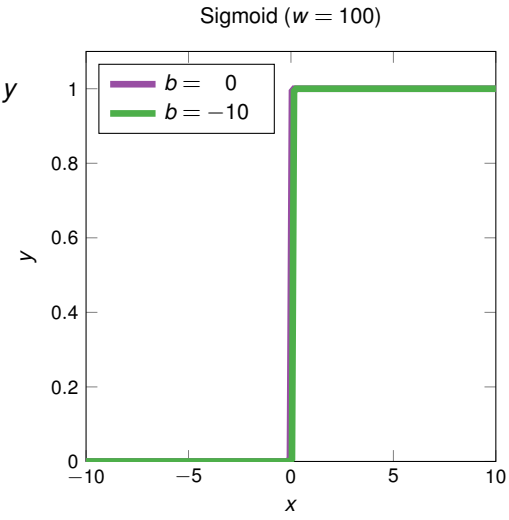


Neural Network as a Step Function

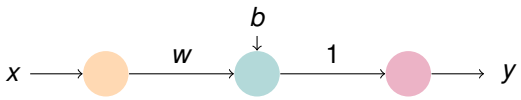


$$y = \sigma(w \cdot x + b) = \frac{1}{1 + e^{-(w \cdot x + b)}}$$

1. Assume $b = 0$ and vary w
2. Assume $w = 100$ and vary b

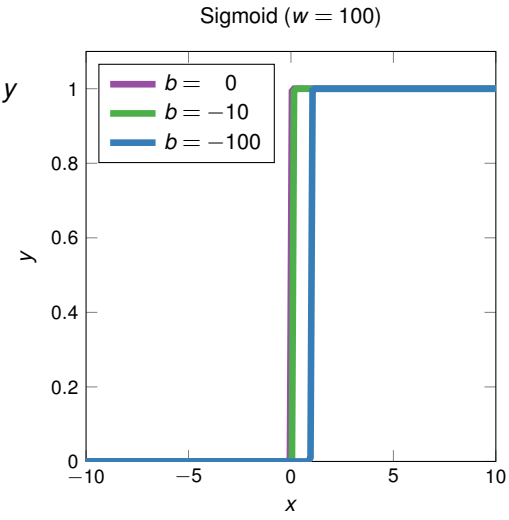


Neural Network as a Step Function

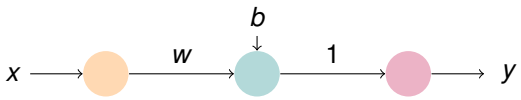


$$y = \sigma(w \cdot x + b) = \frac{1}{1 + e^{-(w \cdot x + b)}}$$

1. Assume $b = 0$ and vary w
2. Assume $w = 100$ and vary b

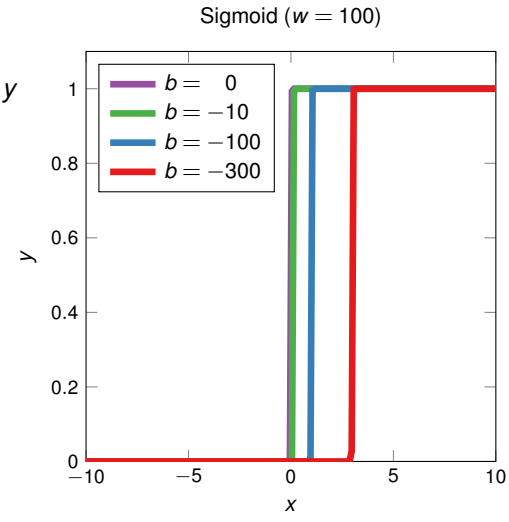


Neural Network as a Step Function

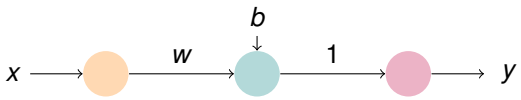


$$y = \sigma(w \cdot x + b) = \frac{1}{1 + e^{-(w \cdot x + b)}}$$

1. Assume $b = 0$ and vary w
2. Assume $w = 100$ and vary b



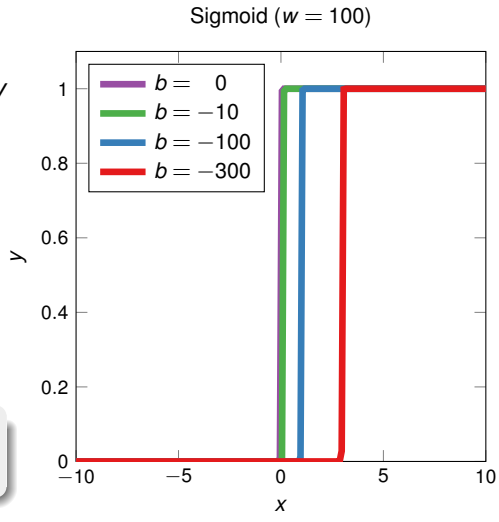
Neural Network as a Step Function



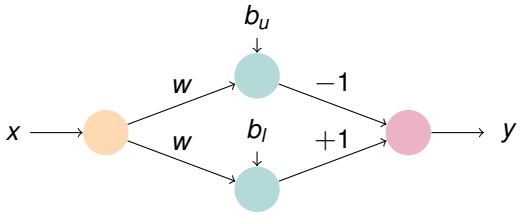
$$y = \sigma(w \cdot x + b) = \frac{1}{1 + e^{-(w \cdot x + b)}}$$

1. Assume $b = 0$ and vary w
2. Assume $w = 100$ and vary b

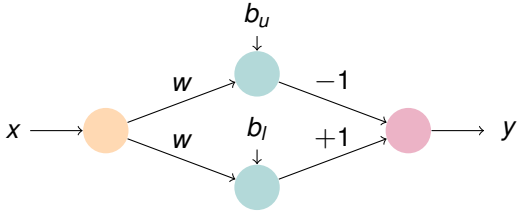
In general, $y = \sigma(wx + b)$ approximates a **step function** at $x = -\frac{b}{w}$.



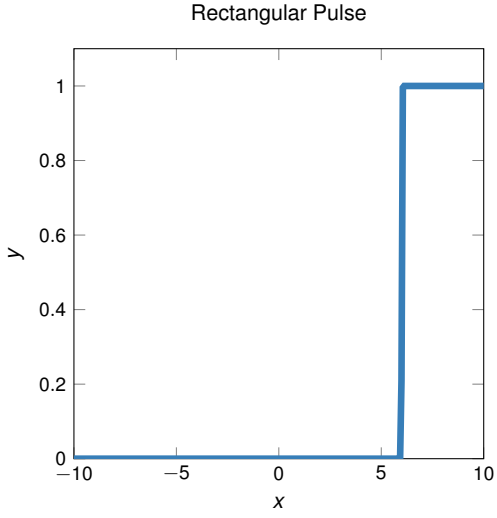
Neural Network as a Rectangular Function



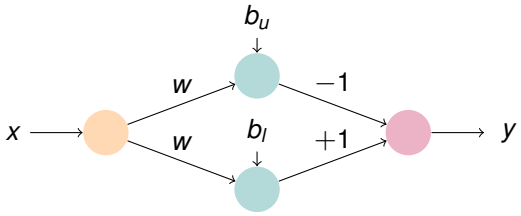
Neural Network as a Rectangular Function



$$y_u = \sigma(100x - 600)$$

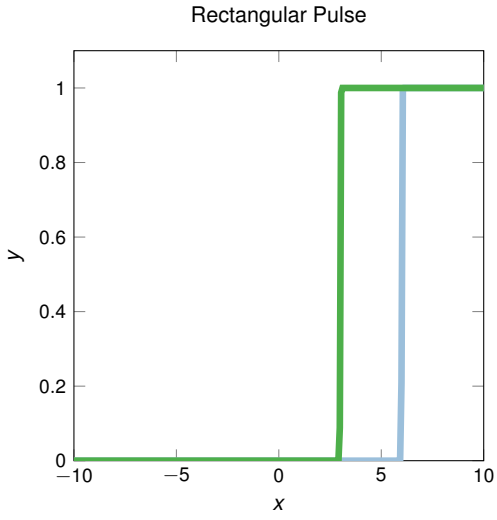


Neural Network as a Rectangular Function

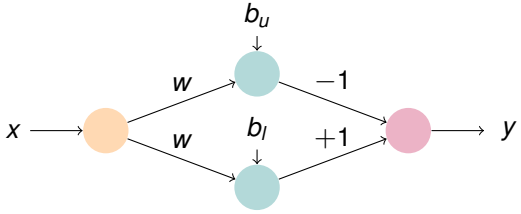


$$y_u = \sigma(100x - 600)$$

$$y_l = \sigma(100x - 300)$$



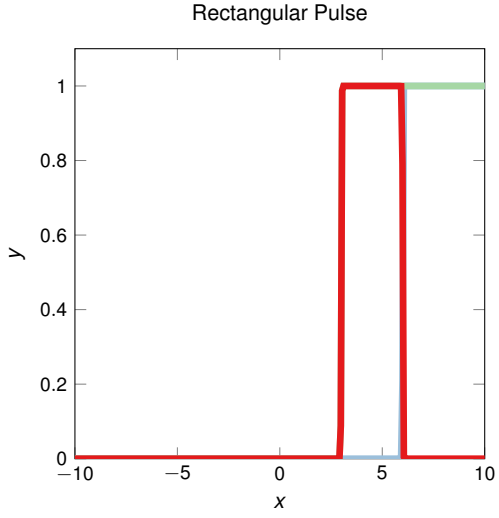
Neural Network as a Rectangular Function



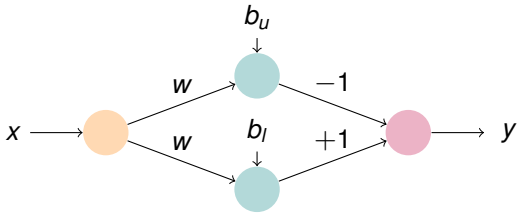
$$y_u = \sigma(100x - 600)$$

$$y_l = \sigma(100x - 300)$$

$$y = y_l - y_u = \sigma(100x - 300) - \sigma(100x - 600)$$



Neural Network as a Rectangular Function

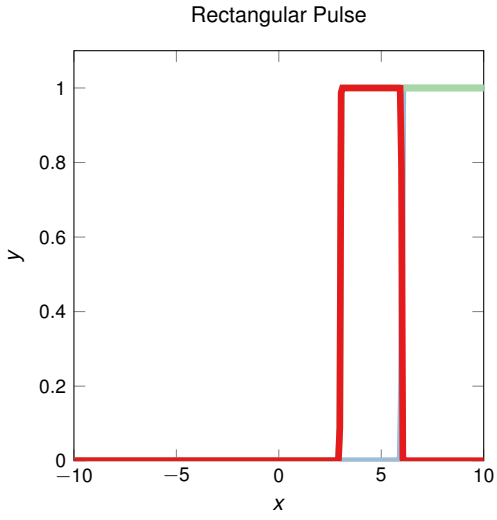


$$y_u = \sigma(100x - 600)$$

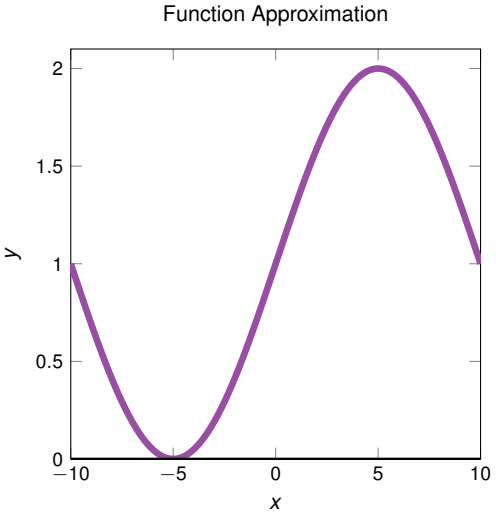
$$y_l = \sigma(100x - 300)$$

$$y = y_l - y_u = \sigma(100x - 300) - \sigma(100x - 600)$$

In general, $y = \sigma(wx + b_l) - \sigma(wx + b_u)$ approximates a **rectangular function** from $x_l = -\frac{b_l}{w}$ to $x_u = -\frac{b_u}{w}$.



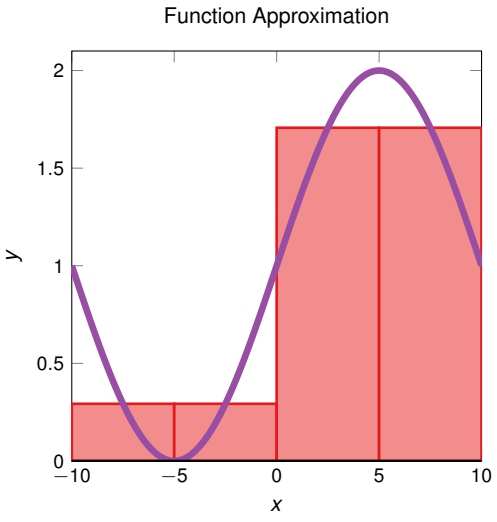
Neural Network as a Universal Approximator



Neural Network as a Universal Approximator

Generalization:

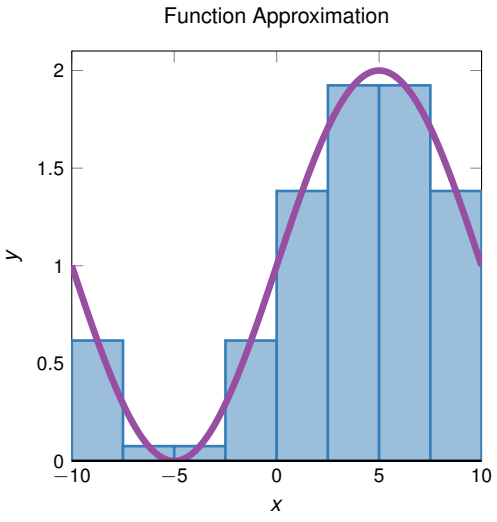
$$y = \sum_{j=1}^{4} \alpha_j \left(\sigma\left(wx - b_{l,j}\right) - \sigma\left(wx - b_{u,j}\right) \right)$$



Neural Network as a Universal Approximator

Generalization:

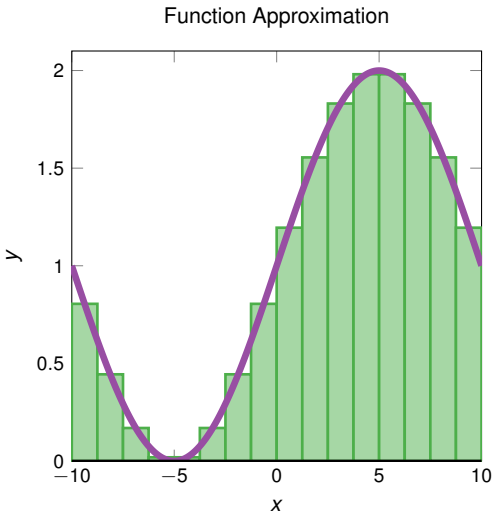
$$y = \sum_{j=1}^{8} \alpha_j \left(\sigma\left(wx - b_{l,j}\right) - \sigma\left(wx - b_{u,j}\right) \right)$$



Neural Network as a Universal Approximator

Generalization:

$$y = \sum_{j=1}^{16} \alpha_j \left(\sigma\left(wx - b_{l,j}\right) - \sigma\left(wx - b_{u,j}\right) \right)$$



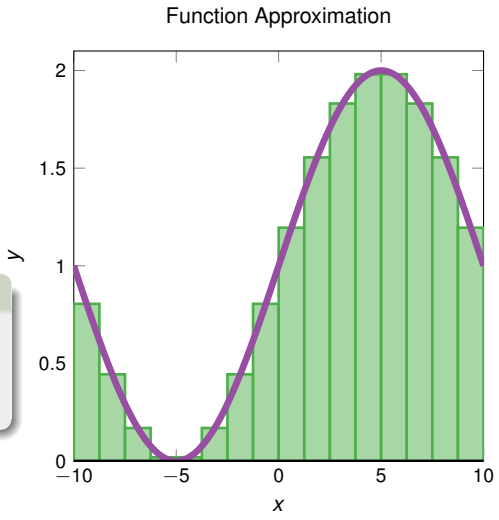
Neural Network as a Universal Approximator

Generalization:

$$y = \sum_{j=1}^{16} \alpha_j \left(\sigma\left(wx - b_{l,j}\right) - \sigma\left(wx - b_{u,j}\right) \right)$$

Result

We can use N rectangular functions constructed from $2N$ neurons to **approximate any function arbitrarily well**!



Conclusion

Summary:

1. Neural networks (NNs) have a long history, but recently became extremely popular.
2. NNs consist of artificial neurons that are brain-inspired.
3. NNs perform matrix-vector multiplications, vector additions, and activation functions.
4. NNs are universal function approximators.

Conclusion

Summary:

1. Neural networks (NNs) have a long history, but recently became extremely popular.
2. NNs consist of artificial neurons that are brain-inspired.
3. NNs perform matrix-vector multiplications, vector additions, and activation functions.
4. NNs are universal function approximators.

Next time:

- **NN training:** cost functions, (improved) gradient descent, backpropagation, weight initialization, data normalization.