Introduction to Deep Learning (and to the Course)

Week1

What is Artificial Intelligence?

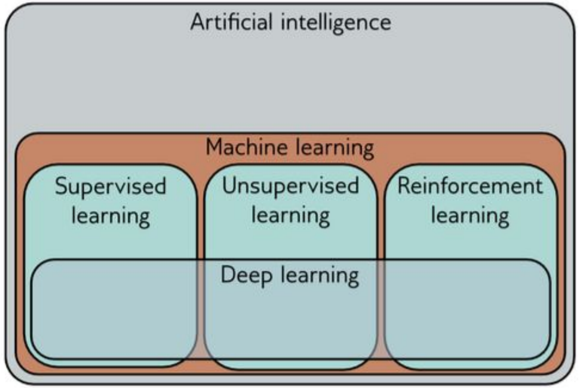
- Any technique that enables computer to mimic humans.

What is Artificial Intelligence?

- Any technique that enables computer to mimic humans.

What is Machine Learning?

- Ability to learn without explicitly programmed
- Data-driven computational approach

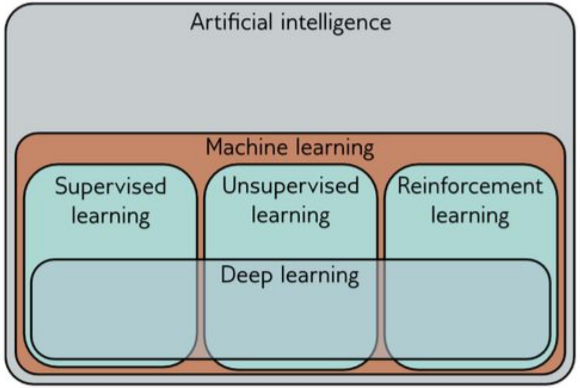


What is Artificial Intelligence?

- Any technique that enables computer to mimic humans.

What is Machine Learning?

- Ability to learn without explicitly programmed
- Data-driven computational approach



What is Deep Learning?

- Extracting patterns from data using neural networks.

Why bother?

- DL revolutionize so many different fields.

Why bother?

- DL revolutionize so many different fields.
- It has a lot of use cases:
  - Generating images/videos from text
  - Audio utilities
  - It has ability to code like humans
  - It can help and assist experts in many fields like medical sciences, natural sciences, social sciences

Why bother?

- DL revolutionize so many different fields.
- It has a lot of use cases:
    - Generating images/videos from text
    - Audio utilities
    - It has ability to code like humans
    - It can help and assist experts in many fields like medical sciences, natural sciences, social sciences
- No hand-crafted features anymore(!?)

Why bother?

- DL revolutionize so many different fields.
- It has a lot of use cases:
  - Generating images/videos from text
  - Audio utilities
  - It has ability to code like humans
  - It can help and assist experts in many fields like medical sciences, natural sciences, social sciences
- No hand-crafted features anymore(!?)
- It is developed rapidly (is it?)

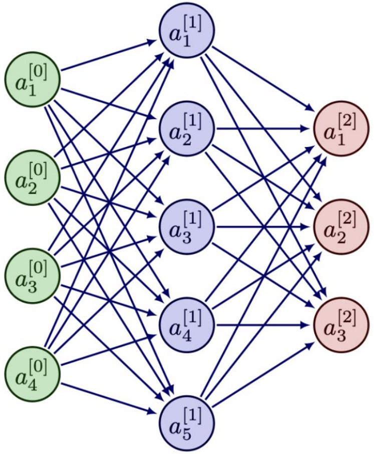
	LeNet-5	AlexNet
YEAR	1998	2012
TRAINING DATA	· MNIST Dataset · 60k training examples · 10 classes	· ImageNet Dataset (ILSVRC) · 1.2M training examples · 1000 classes
TRAINING COMPUTE	· Pentium II CPU · ~0.27 GFLOPs	· Dual Nvidia GTX 580 · 3162 GFLOPs
ALGORITHM	· ~60k Parameters · 5 Layers · Sigmoid Activation Function	· ~60M Parameters · 8 Layers · ReLU Activation Function · Dropout

	LeNet-5	AlexNet	GPT-3	GPT-4
YEAR	1998	2012	2020	2023
TRAINING DATA	· MNIST Dataset · 60k training examples · 10 classes	· ImageNet Dataset (ILSVRC) · 1.2M training examples · 1000 classes	· Common Crawl, WebText, Wikipedia, others · ~500B training tokens · ~100k unique tokens	· ~13T training tokens*
TRAINING COMPUTE	· Pentium II CPU · ~0.27 GFLOPs	· Dual Nvidia GTX 580 · 3162 GFLOPs	· 10,000 Nvidia V100 GPUs · 1+ ExaFLOPs	· 25,000 Nvidia A100s GPUs* · ~4+ ExaFLOPs
ALGORITHM	· ~60k Parameters · 5 Layers · Sigmoid Activation Function	· ~60M Parameters · 8 Layers · ReLU Activation Function · Dropout	· 175B Parameters · 96 Layers · Transformers	· 1T+ Parameters* · *120 Layers · Transformers

Introduction to the Course Content

Fully Connected Neural Networks

$$\begin{pmatrix} w_1 & w_2 & w_3 \\ w_4 & w_5 & w_6 \\ w_7 & w_8 & w_9 \end{pmatrix} \text{sigmoid}\left(\begin{pmatrix} w_1 & w_2 & w_3 \\ w_4 & w_5 & w_6 \\ w_7 & w_8 & w_9 \end{pmatrix} \text{sigmoid}\left(\left(\begin{pmatrix} w_1 & w_2 & w_3 \\ w_4 & w_5 & w_6 \\ w_7 & w_8 & w_9 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} + b_1 \right)\right) + b_2 \right) + b_3$$



Convolutional Neural Networks

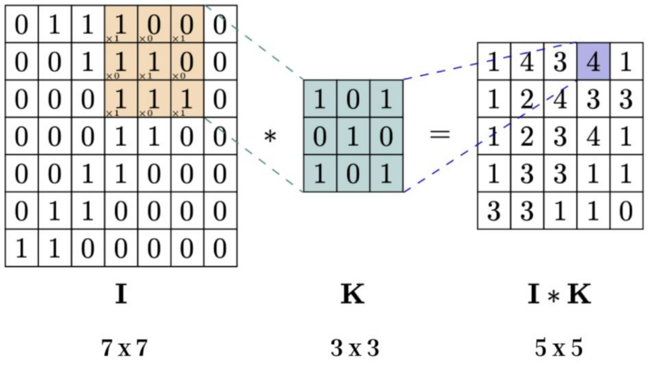


Figure 1: Convolution Operation [11]

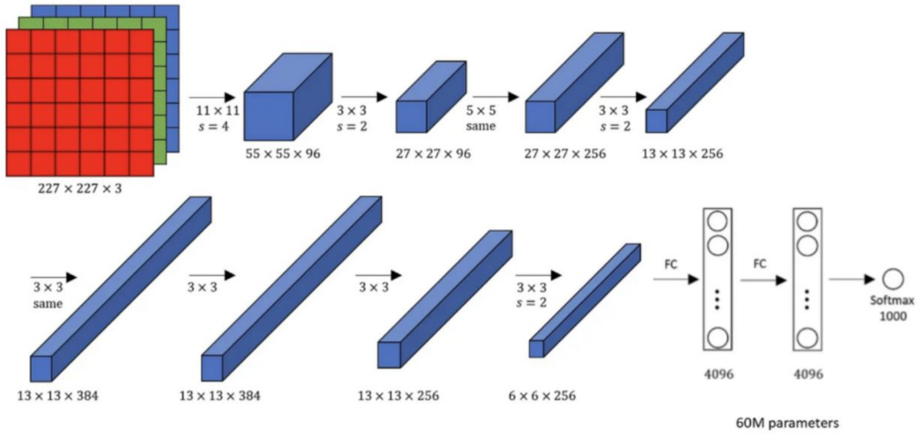


Figure 9: AlexNet Architecture [1]

Recurrent Neural Networks

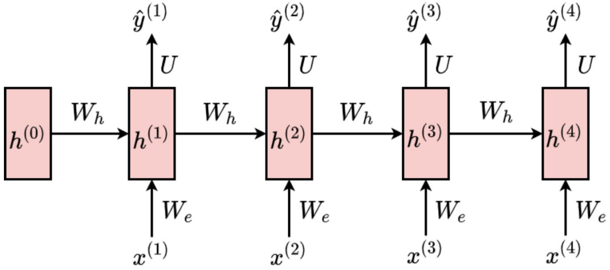


Figure 15: Representing forward equation of RNNs.

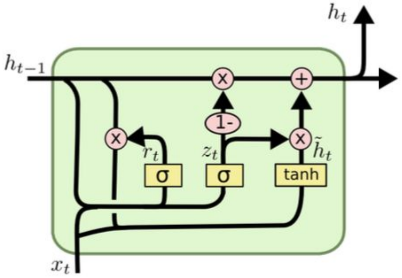


Figure 32: Gated Recurrent Unit.

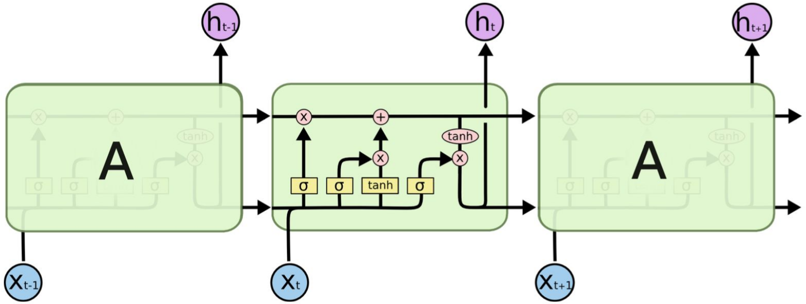


Figure 26: Long short-term memory [40].

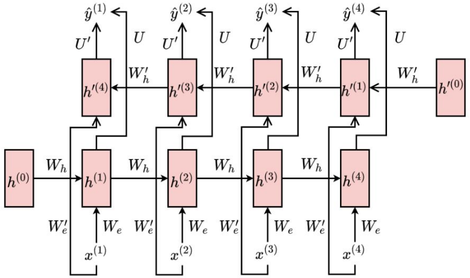
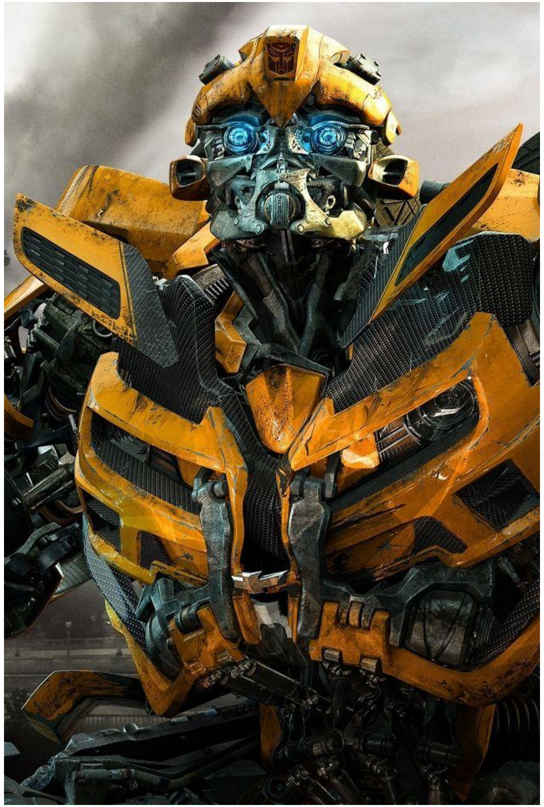


Figure 20: Bidirectional Recurrent Neural Network

Transformers



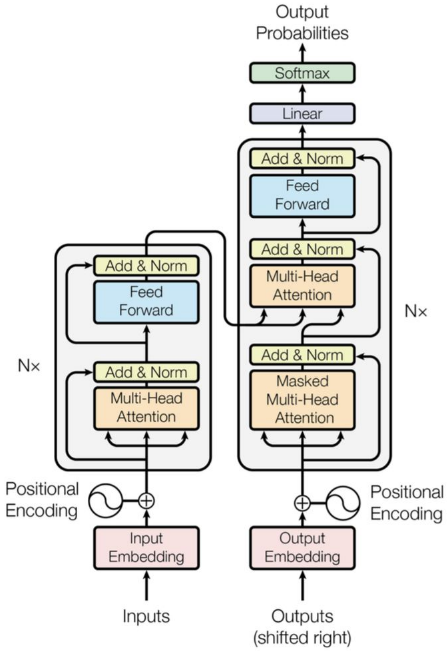
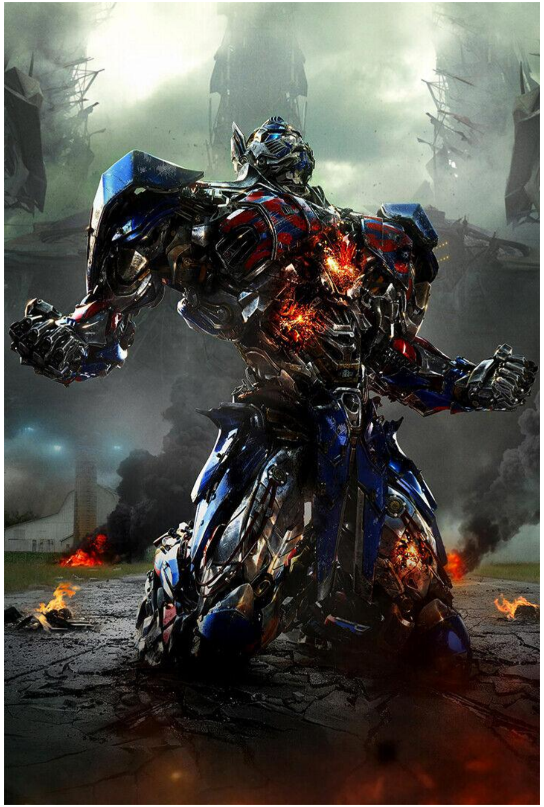


Figure 1: Transformers architecture[31]



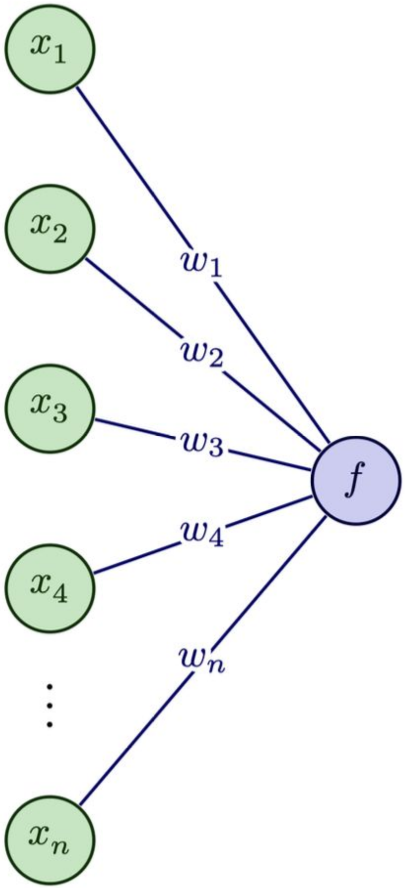
Introduction to Neural Networks

Fully Connected Layers

Here are the main things you need to know:

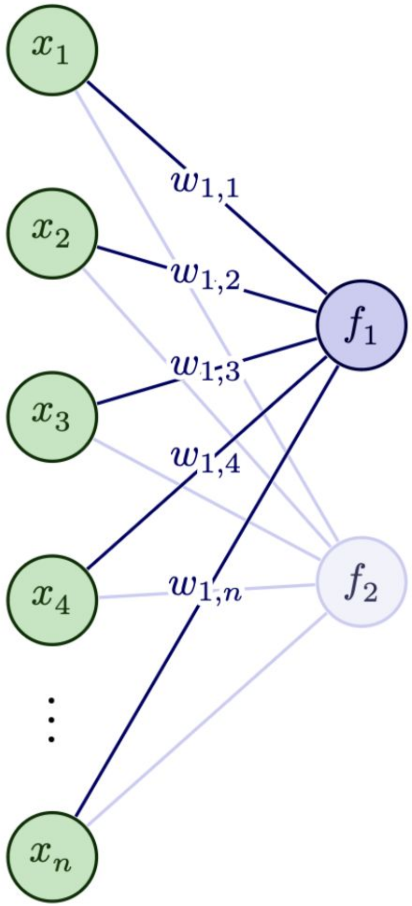
- Forward Propagation
- Backward Propagation
- Activation Functions (Non-linearity)

Neuron



$$f(x_1, ..., x_n) = \sum_{i=1}^{n} w_i x_i + b$$

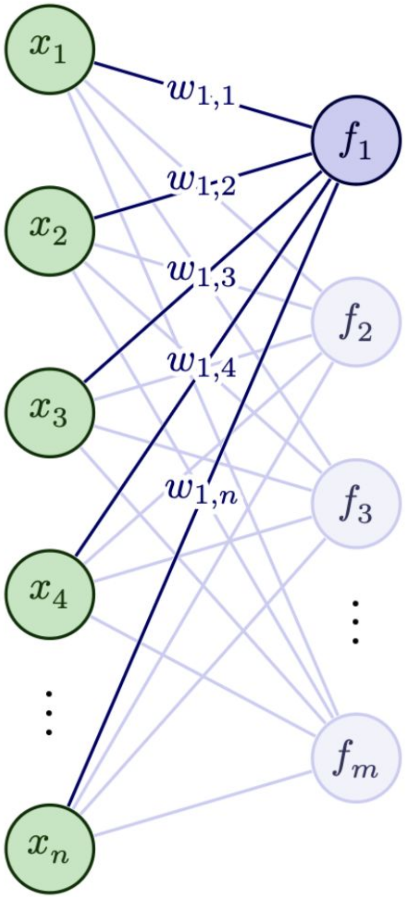
Neuron(s)



$$f_1(x_1, \ldots, x_n) = \sum_{i=1}^{n} w_{1,i} x_i$$

$$f_2(x_1, \ldots, x_n) = \sum_{i=1}^{n} w_{2,i} x_i$$

Bunch of Neurons (A Layer)



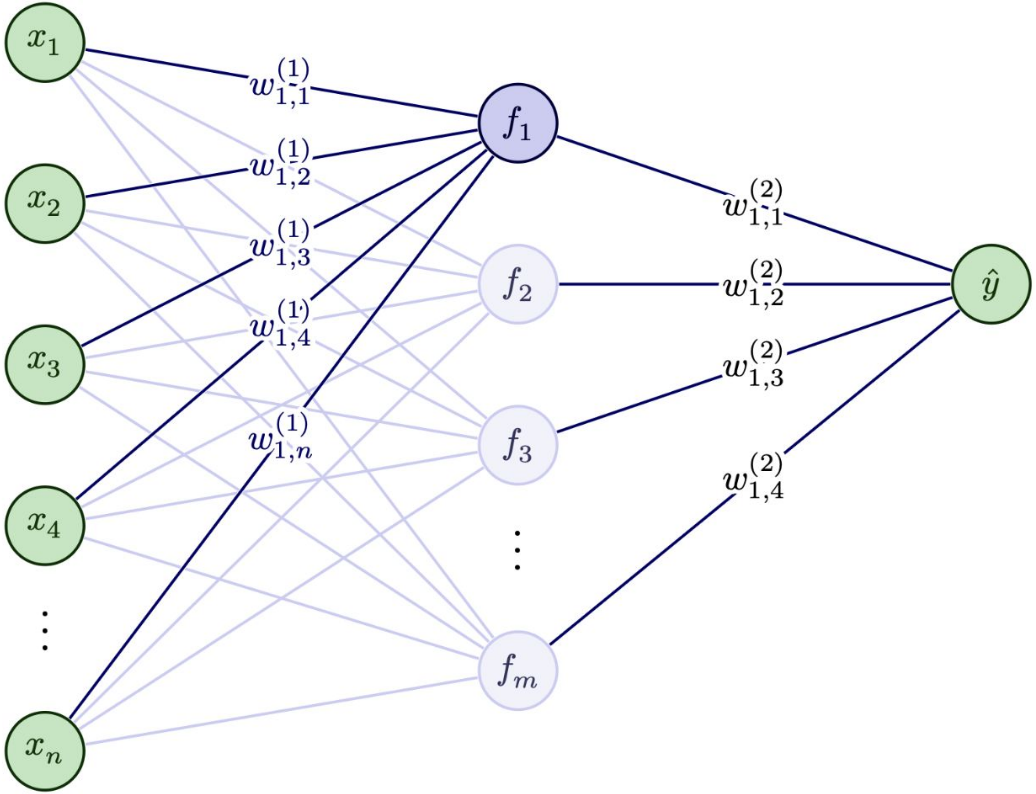
$$f_1 = \sum_{i=1}^{n} w_{1,i} x_i$$

$$f_2 = \sum_{i=1}^{n} w_{2,i} x_i$$

$$\vdots$$

$$f_m = \sum_{i=1}^{n} w_{m,i} x_i$$

A Neural Network (A function from R^{n} to R)

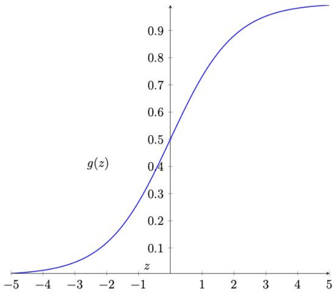


$$f_j = \sum_{i=1}^{n} w_{j,i}^{(1)} x_i$$

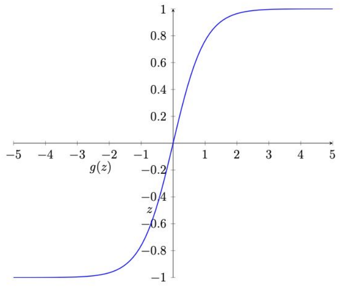
and

$$\hat{y} = \sum_{i=1}^{m} w_{1,i}^{(2)} f_i$$

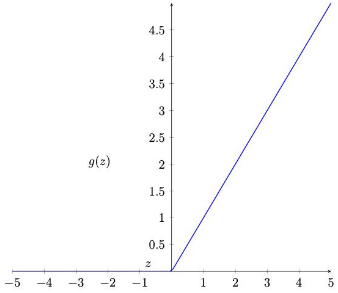
Activation Functions



(a) Sigmoid $\rightarrow \frac{1}{1+e^{-z}}$



(b) Tanh $\rightarrow \frac{e^z-e^{-z}}{e^z+e^{-z}}$



(c) ReLU $\rightarrow \max(0, z)$

Figure 5: Activation Functions

Forward Propagation

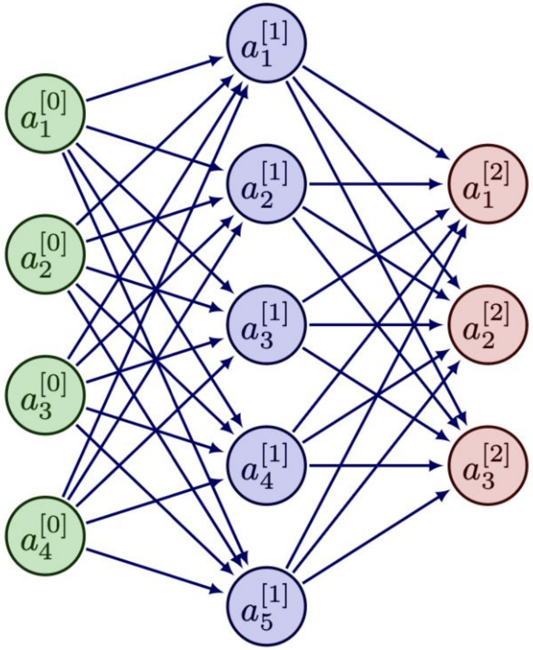


Figure 6: A shallow neural network

Forward Propagation

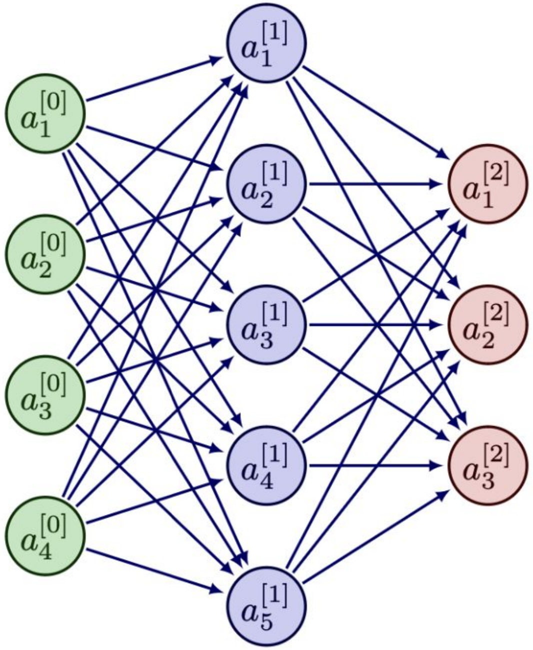


Figure 6: A shallow neural network

$$\mathbf{x} = \mathbf{a}^{[0]}$$

Forward Propagation

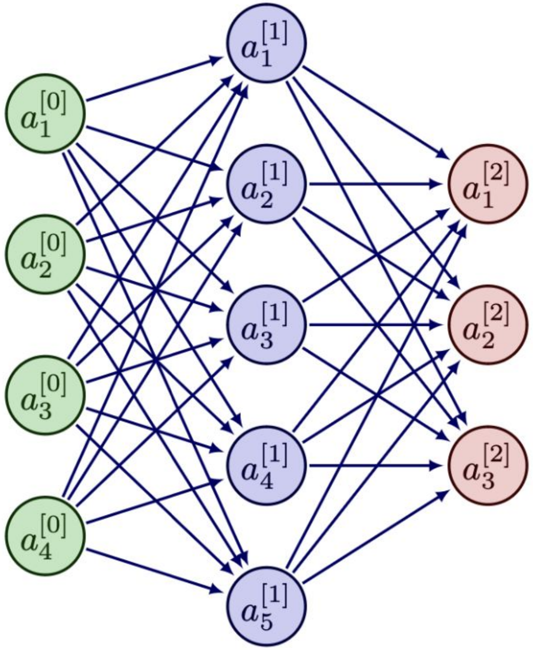


Figure 6: A shallow neural network

$$\mathbf{x} = \mathbf{a}^{[0]}$$

$$\mathbf{z}^{[1]} = \mathbf{W}^{[1]}\mathbf{a}^{[0]} + \mathbf{b}^{[1]}$$

Forward Propagation

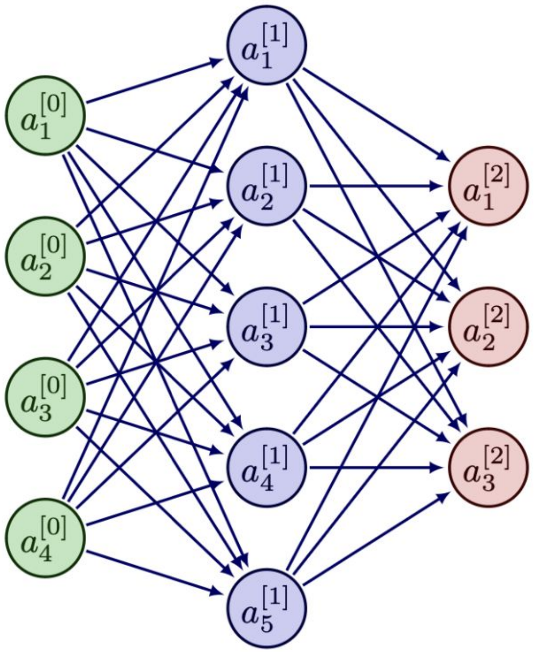


Figure 6: A shallow neural network

$$\mathbf{x} = \mathbf{a}^{[0]}$$
$$\mathbf{z}^{[1]} = \mathbf{W}^{[1]}\mathbf{a}^{[0]} + \mathbf{b}^{[1]}$$
$$\mathbf{a}^{[1]} = \sigma(\mathbf{z}^{[1]})$$

Forward Propagation

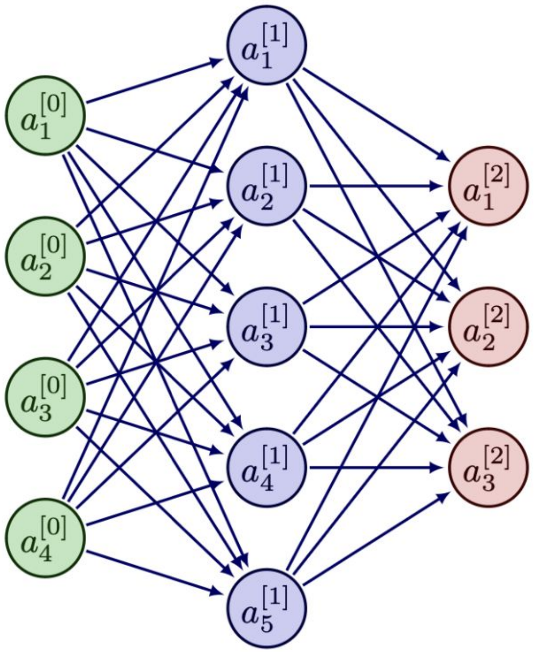


Figure 6: A shallow neural network

$$\mathbf{x} = \mathbf{a}^{[0]}$$
$$\mathbf{z}^{[1]} = \mathbf{W}^{[1]}\mathbf{a}^{[0]} + \mathbf{b}^{[1]}$$
$$\mathbf{a}^{[1]} = \sigma(\mathbf{z}^{[1]})$$
$$\mathbf{z}^{[2]} = \mathbf{W}^{[2]}\mathbf{a}^{[1]} + \mathbf{b}^{[2]}$$

Forward Propagation

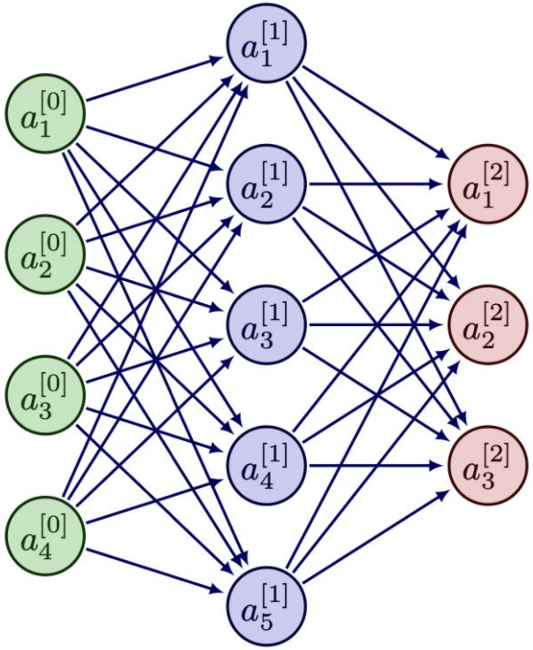


Figure 6: A shallow neural network

$$\mathbf{x} = \mathbf{a}^{[0]}$$

$$\mathbf{z}^{[1]} = \mathbf{W}^{[1]}\mathbf{a}^{[0]} + \mathbf{b}^{[1]}$$

$$\mathbf{a}^{[1]} = \sigma(\mathbf{z}^{[1]})$$

$$\mathbf{z}^{[2]} = \mathbf{W}^{[2]}\mathbf{a}^{[1]} + \mathbf{b}^{[2]}$$

$$\mathbf{a}^{[2]} = \sigma(\mathbf{z}^{[2]})$$

Forward Propagation

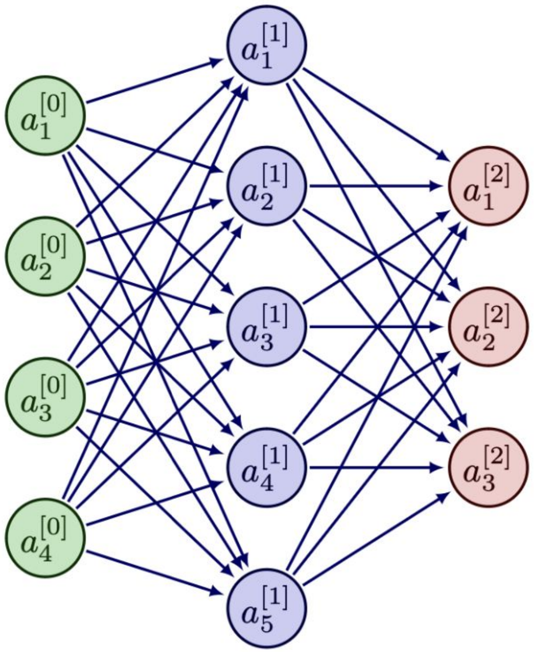


Figure 6: A shallow neural network

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} a_1^{[0]} \\ a_2^{[0]} \\ a_3^{[0]} \\ a_4^{[0]} \end{bmatrix}$$

Forward Propagation

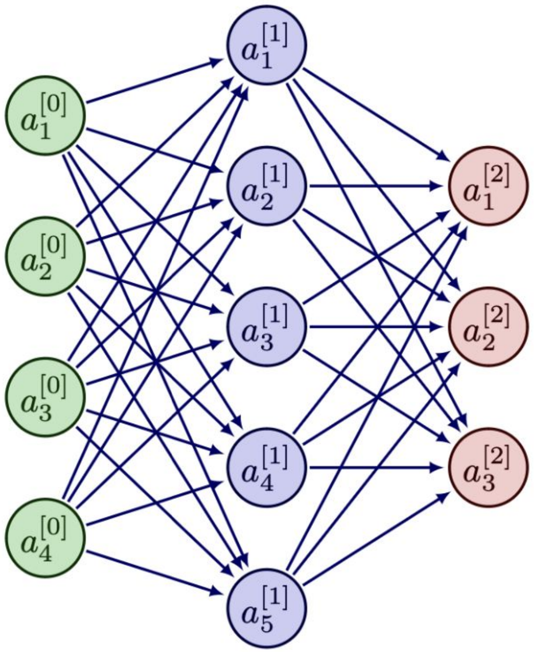


Figure 6: A shallow neural network

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} a_1^{[0]} \\ a_2^{[0]} \\ a_3^{[0]} \\ a_4^{[0]} \end{bmatrix} \Rightarrow \mathbf{x} = \mathbf{a}^{[0]}$$

Forward Propagation

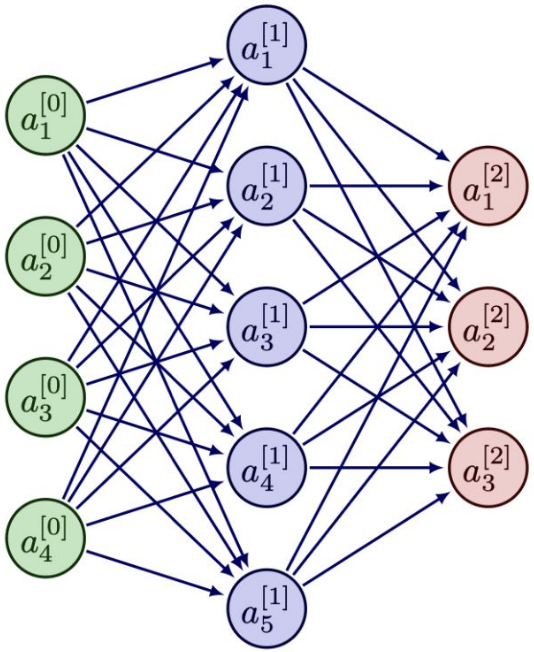


Figure 6: A shallow neural network

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} a_1^{[0]} \\ a_2^{[0]} \\ a_3^{[0]} \\ a_4^{[0]} \end{bmatrix}$$

$$\begin{bmatrix} z_1^{[1]} \\ z_2^{[1]} \\ z_3^{[1]} \\ z_4^{[1]} \end{bmatrix} = \begin{bmatrix} w_{1,1}^{[1]} & w_{1,2}^{[1]} & w_{1,2}^{[1]} & w_{1,2}^{[1]} \\ w_{2,1}^{[1]} & w_{2,2}^{[1]} & w_{2,3}^{[1]} & w_{2,4}^{[1]} \\ w_{3,1}^{[1]} & w_{3,2}^{[1]} & w_{3,3}^{[1]} & w_{3,4}^{[1]} \\ w_{4,1}^{[1]} & w_{4,2}^{[1]} & w_{4,3}^{[1]} & w_{4,4}^{[1]} \\ w_{5,1}^{[1]} & w_{5,2}^{[1]} & w_{5,3}^{[1]} & w_{5,4}^{[1]} \end{bmatrix} \begin{bmatrix} a_1^{[0]} \\ a_2^{[0]} \\ a_3^{[0]} \\ a_4^{[0]} \end{bmatrix} + \begin{bmatrix} b_1^{[1]} \\ b_2^{[1]} \\ b_3^{[1]} \\ b_4^{[1]} \\ b_5^{[1]} \end{bmatrix}$$

Forward Propagation

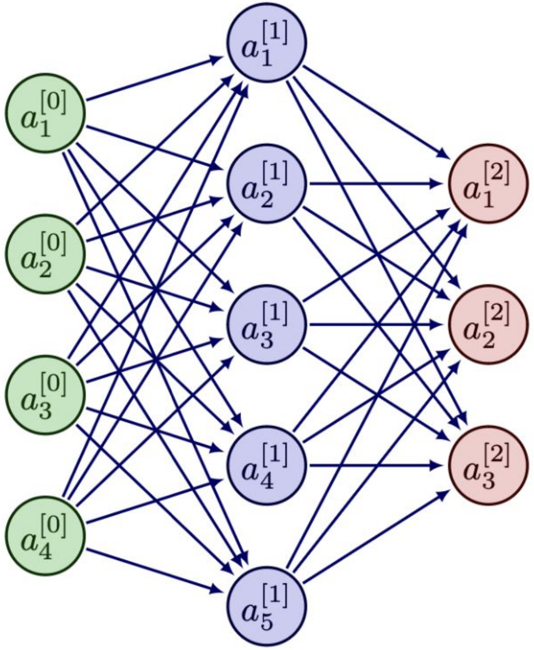


Figure 6: A shallow neural network

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} a_1^{[0]} \\ a_2^{[0]} \\ a_3^{[0]} \\ a_4^{[0]} \end{bmatrix}$$

$$\begin{bmatrix} z_1^{[1]} \\ z_2^{[1]} \\ z_3^{[1]} \\ z_4^{[1]} \end{bmatrix} = \begin{bmatrix} w_{1,1}^{[1]} & w_{1,2}^{[1]} & w_{1,2}^{[1]} & w_{1,2}^{[1]} \\ w_{2,1}^{[1]} & w_{2,2}^{[1]} & w_{2,3}^{[1]} & w_{2,4}^{[1]} \\ w_{3,1}^{[1]} & w_{3,2}^{[1]} & w_{3,3}^{[1]} & w_{3,4}^{[1]} \\ w_{4,1}^{[1]} & w_{4,2}^{[1]} & w_{4,3}^{[1]} & w_{4,4}^{[1]} \\ w_{5,1}^{[1]} & w_{5,2}^{[1]} & w_{5,3}^{[1]} & w_{5,4}^{[1]} \end{bmatrix} \begin{bmatrix} a_1^{[0]} \\ a_2^{[0]} \\ a_3^{[0]} \\ a_4^{[0]} \end{bmatrix} + \begin{bmatrix} b_1^{[1]} \\ b_2^{[1]} \\ b_3^{[1]} \\ b_4^{[1]} \\ b_5^{[1]} \end{bmatrix}$$

$$\mathbf{z}^{[1]} = \mathbf{W}^{[1]}\mathbf{a}^{[0]} + \mathbf{b}^{[1]}$$

Forward Propagation

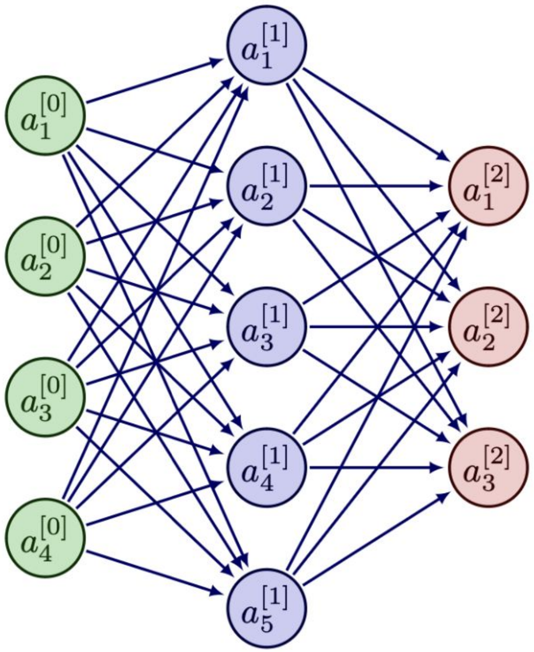


Figure 6: A shallow neural network

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} a_1^{[0]} \\ a_2^{[0]} \\ a_3^{[0]} \\ a_4^{[0]} \end{bmatrix}$$

$$\begin{bmatrix} z_1^{[1]} \\ z_2^{[1]} \\ z_3^{[1]} \\ z_4^{[1]} \end{bmatrix} = \begin{bmatrix} w_{1,1}^{[1]} & w_{1,2}^{[1]} & w_{1,2}^{[1]} & w_{1,2}^{[1]} \\ w_{2,1}^{[1]} & w_{2,2}^{[1]} & w_{2,3}^{[1]} & w_{2,4}^{[1]} \\ w_{3,1}^{[1]} & w_{3,2}^{[1]} & w_{3,3}^{[1]} & w_{3,4}^{[1]} \\ w_{4,1}^{[1]} & w_{4,2}^{[1]} & w_{4,3}^{[1]} & w_{4,4}^{[1]} \\ w_{5,1}^{[1]} & w_{5,2}^{[1]} & w_{5,3}^{[1]} & w_{5,4}^{[1]} \end{bmatrix} \begin{bmatrix} a_1^{[0]} \\ a_2^{[0]} \\ a_3^{[0]} \\ a_4^{[0]} \end{bmatrix} + \begin{bmatrix} b_1^{[1]} \\ b_2^{[1]} \\ b_3^{[1]} \\ b_4^{[1]} \\ b_5^{[1]} \end{bmatrix}$$

$$\begin{bmatrix} a_1^{[1]} \\ a_2^{[1]} \\ a_3^{[1]} \\ a_4^{[1]} \end{bmatrix} = \sigma\left(\begin{bmatrix} z_1^{[1]} \\ z_2^{[1]} \\ z_3^{[1]} \\ z_4^{[1]} \end{bmatrix} \right) = \begin{bmatrix} \sigma(z_1^{[1]}) \\ \sigma(z_2^{[1]}) \\ \sigma(z_3^{[1]}) \\ \sigma(z_4^{[1]}) \end{bmatrix}$$

Forward Propagation

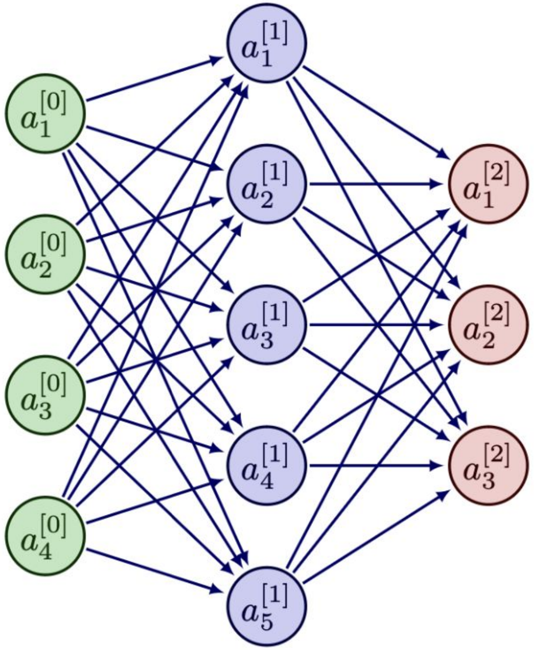


Figure 6: A shallow neural network

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} a_1^{[0]} \\ a_2^{[0]} \\ a_3^{[0]} \\ a_4^{[0]} \end{bmatrix}$$

$$\begin{bmatrix} z_1^{[1]} \\ z_2^{[1]} \\ z_3^{[1]} \\ z_4^{[1]} \end{bmatrix} = \begin{bmatrix} w_{1,1}^{[1]} & w_{1,2}^{[1]} & w_{1,2}^{[1]} & w_{1,2}^{[1]} \\ w_{2,1}^{[1]} & w_{2,2}^{[1]} & w_{2,3}^{[1]} & w_{2,4}^{[1]} \\ w_{3,1}^{[1]} & w_{3,2}^{[1]} & w_{3,3}^{[1]} & w_{3,4}^{[1]} \\ w_{4,1}^{[1]} & w_{4,2}^{[1]} & w_{4,3}^{[1]} & w_{4,4}^{[1]} \\ w_{5,1}^{[1]} & w_{5,2}^{[1]} & w_{5,3}^{[1]} & w_{5,4}^{[1]} \end{bmatrix} \begin{bmatrix} a_1^{[0]} \\ a_2^{[0]} \\ a_3^{[0]} \\ a_4^{[0]} \end{bmatrix} + \begin{bmatrix} b_1^{[1]} \\ b_2^{[1]} \\ b_3^{[1]} \\ b_4^{[1]} \\ b_5^{[1]} \end{bmatrix}$$

$$\begin{bmatrix} a_1^{[1]} \\ a_2^{[1]} \\ a_3^{[1]} \\ a_4^{[1]} \end{bmatrix} = \sigma\left(\begin{bmatrix} z_1^{[1]} \\ z_2^{[1]} \\ z_3^{[1]} \\ z_4^{[1]} \end{bmatrix} \right) = \begin{bmatrix} \sigma(z_1^{[1]}) \\ \sigma(z_2^{[1]}) \\ \sigma(z_3^{[1]}) \\ \sigma(z_4^{[1]}) \end{bmatrix} \Rightarrow \mathbf{a}^{[1]} = \sigma(\mathbf{z}^{[1]})$$

Forward Propagation

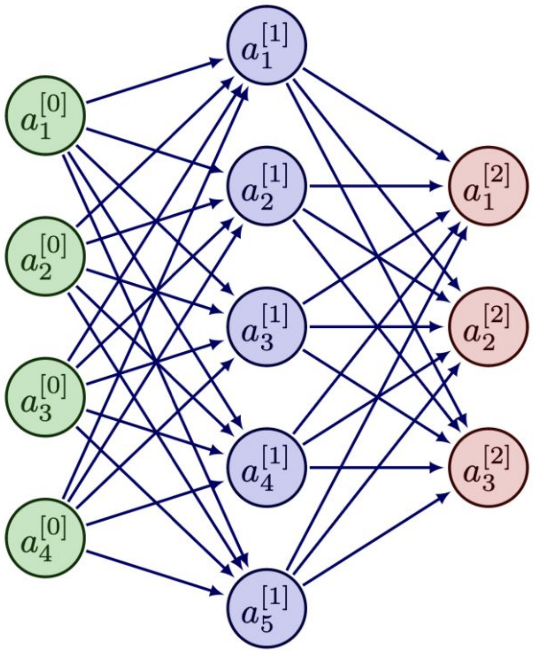


Figure 6: A shallow neural network

In this setting, $\mathbf{W}^{[1]}$ is a matrix with shape 5×4. $\mathbf{W}^{[2]}$ is a matrix with shape 3×5. $\mathbf{z}^{[1]}$ is a vector with shape 5×1, $\mathbf{a}^{[1]}$ has the same shape as $\mathbf{z}^{[1]}$ and $\mathbf{b}^{[2]}$ has the shape 3×1. Other shapes left to the curious readers as an exercise. It is clear that the shape of weight matrices are directly related to the number of neuron between layers.

Forward Propagation (General Formula)

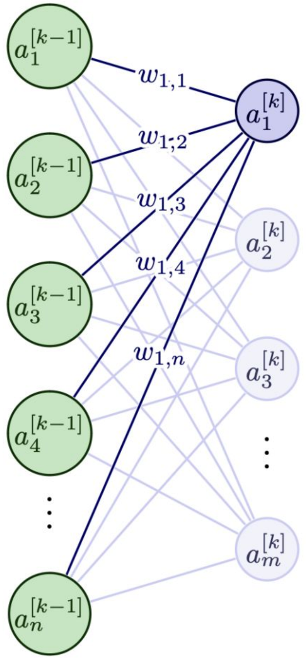


Figure 7: Forward propagation from layer $k-1$ to layer k

Forward Propagation (General Formula)

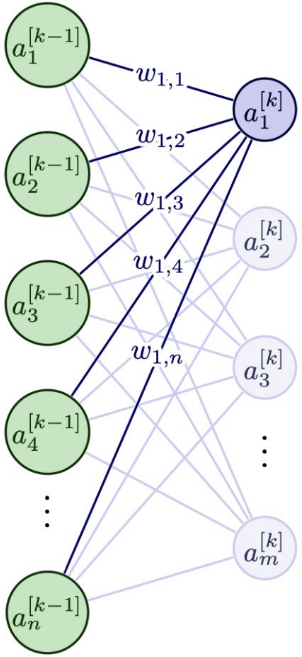


Figure 7: Forward propagation from layer $k-1$ to layer k

$$\begin{bmatrix} a_1^{[k]} \\ a_2^{[k]} \\ \vdots \\ a_m^{[k]} \end{bmatrix} = \sigma\left(\begin{bmatrix} w_{1,1} & w_{1,2} & \cdots & w_{1,n} \\ w_{2,1} & w_{2,2} & \cdots & w_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ w_{m,1} & w_{m,2} & \cdots & w_{m,n} \end{bmatrix} \begin{bmatrix} a_1^{[k-1]} \\ a_2^{[k-1]} \\ \vdots \\ a_n^{[k-1]} \end{bmatrix} + \begin{bmatrix} b_1^{[k]} \\ b_2^{[k]} \\ \vdots \\ b_m^{[k]} \end{bmatrix} \right)$$

Forward Propagation (General Formula)

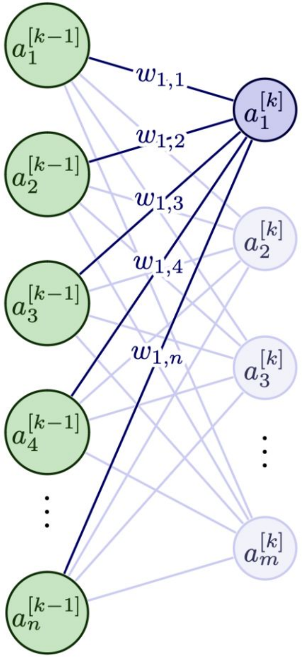


Figure 7: Forward propagation from layer $k-1$ to layer k

$$\begin{bmatrix} a_1^{[k]} \\ a_2^{[k]} \\ \vdots \\ a_m^{[k]} \end{bmatrix} = \sigma\left(\begin{bmatrix} w_{1,1} & w_{1,2} & \cdots & w_{1,n} \\ w_{2,1} & w_{2,2} & \cdots & w_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ w_{m,1} & w_{m,2} & \cdots & w_{m,n} \end{bmatrix} \begin{bmatrix} a_1^{[k-1]} \\ a_2^{[k-1]} \\ \vdots \\ a_n^{[k-1]} \end{bmatrix} + \begin{bmatrix} b_1^{[k]} \\ b_2^{[k]} \\ \vdots \\ b_m^{[k]} \end{bmatrix} \right)$$

$$\underset{m\times 1}{\mathbf{a}^{[k]}} = \sigma\left(\underset{m\times n}{\mathbf{W}^{[k]}} \underset{n\times 1}{\mathbf{a}^{[k-1]}} + \underset{m\times 1}{\mathbf{b}^{[k]}} \right)$$

Backward Propagation

- It is a gradient-estimation method for neural networks

Backward Propagation

- It is a gradient-estimation method for neural networks
- It allows us to compute gradients of a function EFFICIENTLY.

Backward Propagation

- It is a gradient-estimation method for neural networks
- It allows us to compute gradients of a function EFFICIENTLY.
- It is proposed in a paper called “Learning representations by back-propagating errors” in 1986 by Rumelhart and Hinton

Backward Propagation

- It is a gradient-estimation method for neural networks
- It allows us to compute gradients of a function EFFICIENTLY.
- It is proposed in a paper called “Learning representations by back-propagating errors” in 1986 by Rumelhart and Hinton
- Paper recommendation https://arxiv.org/pdf/2301.09977

Backward Propagation

$$\mathcal{L}_\theta(y_i, \hat{y}_i) = (y_i - \hat{y}_i)^2.$$

Backward Propagation

$$\mathcal{L}_\theta(y_i, \hat{y}_i) = (y_i - \hat{y}_i)^2.$$

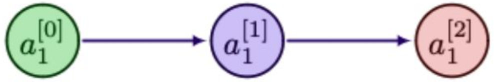


Figure 8: A neural network with one neurons in all layers

Backward Propagation

$$\mathcal{L}_\theta(y_i, \hat{y}_i) = (y_i - \hat{y}_i)^2.$$

Figure 8: A neural network with one neurons in all layers

$$
\begin{aligned}
z^{[1]} &= w^{[1]}a^{[0]} + b^{[1]} \\
a^{[1]} &= \sigma(z^{[1]}) \\
z^{[2]} &= w^{[2]}a^{[1]} + b^{[2]} \\
a^{[2]} &= \sigma(z^{[2]})
\end{aligned}
$$

Backward Propagation

Figure 8: A neural network with one neurons in all layers

$$
\begin{aligned}
z^{[1]} &= w^{[1]}a^{[0]} + b^{[1]} \\
a^{[1]} &= \sigma(z^{[1]}) \\
z^{[2]} &= w^{[2]}a^{[1]} + b^{[2]} \\
a^{[2]} &= \sigma(z^{[2]})
\end{aligned}
$$

Backward Propagation

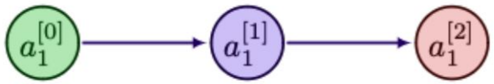


Figure 8: A neural network with one neurons in all layers

$$
\begin{aligned}
z^{[1]} &= w^{[1]}a^{[0]} + b^{[1]} \\
a^{[1]} &= \sigma(z^{[1]}) \\
z^{[2]} &= w^{[2]}a^{[1]} + b^{[2]} \\
a^{[2]} &= \sigma(z^{[2]})
\end{aligned}
$$

$$\mathcal{L}(y, a^{[2]}) = (y - a^{[2]})^2$$

Backward Propagation

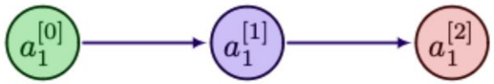


Figure 8: A neural network with one neurons in all layers

$$
\begin{aligned}
z^{[1]} &= w^{[1]}a^{[0]} + b^{[1]} \\
a^{[1]} &= \sigma(z^{[1]}) \\
z^{[2]} &= w^{[2]}a^{[1]} + b^{[2]} \\
a^{[2]} &= \sigma(z^{[2]})
\end{aligned}
$$

$$\mathcal{L}(y, a^{[2]}) = (y - a^{[2]})^2$$

$$\frac{\partial \mathcal{L}(y,a^{[2]})}{\partial w^{[2]}}, \frac{\partial \mathcal{L}(y,a^{[2]})}{\partial b^{[2]}}, \frac{\partial \mathcal{L}(y,a^{[2]})}{\partial w^{[1]}}, \frac{\partial \mathcal{L}(y,a^{[2]})}{\partial b^{[1]}}.$$

Backward Propagation

$$
\begin{aligned}
\mathcal{L}(y, a^{[2]}) &= (y - a^{[2]})^2 \\
&= (y - \sigma(z^{[2]})^2 \\
&= (y - (w^{[2]}a^{[1]} + b^{[2]}))^2
\end{aligned}
$$

Backward Propagation

$$\begin{aligned}\mathcal{L}(y, a^{[2]}) &= (y - a^{[2]})^2 \\ &= (y - \sigma(z^{[2]})^2 \\ &= (y - (w^{[2]}a^{[1]} + b^{[2]}))^2\end{aligned}$$

Compute gradient for w^{2}

$$\frac{\partial((y - (w^{[2]}a^{[1]} + b^{[2]}))^2)}{\partial w^{[2]}} =$$

Backward Propagation

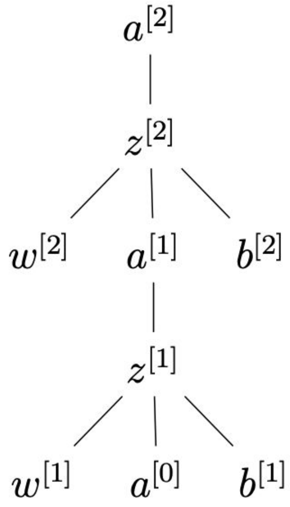
$$\begin{aligned}\mathcal{L}(y, a^{[2]}) &= (y - a^{[2]})^2 \\ &= (y - \sigma(z^{[2]})^2 \\ &= (y - (w^{[2]}a^{[1]} + b^{[2]}))^2\end{aligned}$$

Compute gradient for w^{2}

$$\frac{\partial((y - (w^{[2]}a^{[1]} + b^{[2]}))^2)}{\partial w^{[2]}} = 2(y - \sigma(z^{[2]})) \cdot (\sigma' \cdot (w^{[2]}a^{[1]} + b^{[2]}) \cdot a^{[1]})$$

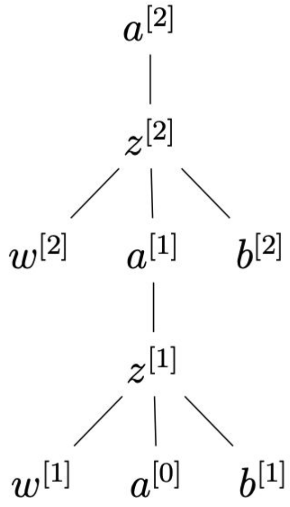
Backward Propagation

$$\frac{\partial((y-(w^{[2]}a^{[1]}+b^{[2]}))^2)}{\partial w^{[2]}} = 2(y-\sigma(z^{[2]}))\cdot(\sigma'\cdot(w^{[2]}a^{[1]}+b^{[2]})\cdot a^{[1]})$$



Backward Propagation

$$\frac{\partial((y-(w^{[2]}a^{[1]}+b^{[2]}))^2)}{\partial w^{[2]}} = 2(y-\sigma(z^{[2]}))\cdot(\sigma'\cdot(w^{[2]}a^{[1]}+b^{[2]})\cdot a^{[1]})$$



$$\frac{\partial\mathcal{L}(y,a^{[2]})}{\partial w^{[2]}} = \underbrace{\frac{\partial\mathcal{L}(y,a^{[2]})}{\partial a^{[2]}}}_{2(y-a^{[2]})}\cdot\underbrace{\frac{\partial a^{[2]}}{\partial z^{[2]}}}_{\sigma'(z^{[2]})}\cdot\underbrace{\frac{\partial z^{[2]}}{\partial w^{[2]}}}_{a^{[1]}}$$

Bonus Content

Definition 1. *An **artificial neuron** with weights* $w_1, \ldots, w_n \in R$*, bias* $b \in R$*, and non-linear activation function* $\rho : R \to R$ *is defined as the function* $f : R^n \to R$ *given by*

$$f\,(x_1, \ldots, x_n) = \rho\left(\sum_{i=1}^{n} x_i w_i + b\right) = \rho(\langle x, w\rangle - b)$$

where $\boldsymbol{w} = (w_1, \ldots, w_n)$ *and* $\boldsymbol{x} = (x_1, \ldots, x_n)$. *(Kutyniok 2022)*

Definition 2. *Let $d \in N$ be the dimension of the input layer, L the number of layers, $N_0 := d, N_\ell, \ell = 1, \ldots, L$, the dimensions of the hidden and last layer, $\rho : R \to R$ a (non-linear) activation function, and, for $\ell = 1, \ldots, L$, let T_ℓ be the affine transformations*

$$T_\ell : R^{N_{\ell-1}} \to R^{N_\ell}, \quad T_\ell x = W^{(\ell)} x + b^{(\ell)}$$

with $W^{(\ell)} \in R^{N_\ell \times N_{\ell-1}}$ being the weight matrices and $b^{(\ell)} \in R^{N_\ell}$ the bias vectors of the ℓ th layer. Then $\Phi : R^d \to R^{N_L}$, given by

$$\Phi(x) = T_L \rho \left(T_{L-1} \rho \left(\ldots \rho \left(T_1(x)\right)\right)\right), \quad x \in R^d$$

is called (deep) neural network of depth L. (Kutyniok 2022)

Neural Networks as Universal Approximators

Theorem (Universal function approximation)[5]

Let $\sigma \in C(\mathbb{R}, \mathbb{R})$ be a non-polynomial activation function. For every $n, m \in \mathbb{N}$, every compact subset $K \subseteq \mathbb{R}^n$, every function $f \in C(K, \mathbb{R}^m)$ and $\epsilon > 0$, there exist $k \in \mathbb{N}$, $\mathbf{A} \in \mathbb{R}^{k \times n}$, $\mathbf{b} \in \mathbb{R}^k$, and $\mathbf{C} \in \mathbb{R}^{m \times k}$ such that

$$\sup_{x \in K} \| f(x) - g(x) \| < \epsilon,$$

where $g(x) = \mathbf{C}\sigma(\mathbf{A}x + \mathbf{b})$.

Questions you have?