

# Neural Networks

CISC 7026: Introduction to Deep Learning

University of Macau

# Notation Change

**Notation change:** Previously  $x_i, y_i$  referred to data  $i$

# Notation Change

**Notation change:** Previously  $x_i, y_i$  referred to data  $i$

Moving forward, I will differentiate between **data** indices  $x_{[i]}$  and other indices  $x_i$

# Notation Change

**Notation change:** Previously  $x_i, y_i$  referred to data  $i$

Moving forward, I will differentiate between **data** indices  $x_{[i]}$  and other indices  $x_i$

$$\mathbf{X}_D = \begin{bmatrix} x_{[1]} \\ \vdots \\ x_{[n]} \end{bmatrix} = \begin{bmatrix} x_{[1],1} & x_{[1],2} & \cdots \\ \vdots & \vdots & \vdots \\ x_{[n],1} & x_{[n],2} & \cdots \end{bmatrix}$$

# Notation Change

**Notation change:** Previously  $x_i, y_i$  referred to data  $i$

Moving forward, I will differentiate between **data** indices  $x_{[i]}$  and other indices  $x_i$

$$\mathbf{X}_D = \begin{bmatrix} x_{[1]} \\ \vdots \\ x_{[n]} \end{bmatrix} = \begin{bmatrix} x_{[1],1} & x_{[1],2} & \cdots \\ \vdots & \vdots & \vdots \\ x_{[n],1} & x_{[n],2} & \cdots \end{bmatrix}$$

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \end{bmatrix}, \quad \mathbf{X} = \begin{bmatrix} x_{1,1} & \cdots & x_{1,n} \\ \vdots & \ddots & \vdots \\ x_{m,1} & \cdots & x_{m,n} \end{bmatrix}$$

# Agenda

1. Review
2. Multivariate linear regression
3. Limitations of linear regression
4. History of neural networks
5. Biological neurons
6. Artificial neurons
7. Wide neural networks
8. Deep neural networks
9. Practical considerations

# Agenda

1. Review
2. Multivariate linear regression
3. Limitations of linear regression
4. History of neural networks
5. Biological neurons
6. Artificial neurons
7. Wide neural networks
8. Deep neural networks
9. Practical considerations

# Review

Since you are very educated, we focused on how education affects life expectancy

# Review

Since you are very educated, we focused on how education affects life expectancy

Studies show a causal effect of education on health

Since you are very educated, we focused on how education affects life expectancy

Studies show a causal effect of education on health

- *The causal effects of education on health outcomes in the UK Biobank.*  
Davies et al. *Nature Human Behaviour*.

# Review

Since you are very educated, we focused on how education affects life expectancy

Studies show a causal effect of education on health

- *The causal effects of education on health outcomes in the UK Biobank.*  
Davies et al. *Nature Human Behaviour*.
- By staying in school, you are likely to live longer

# Review

Since you are very educated, we focused on how education affects life expectancy

Studies show a causal effect of education on health

- *The causal effects of education on health outcomes in the UK Biobank.*  
Davies et al. *Nature Human Behaviour*.
- By staying in school, you are likely to live longer
- Being rich also helps, but education alone has a **causal** relationship with life expectancy

**Task:** Given your education, predict your life expectancy

**Task:** Given your education, predict your life expectancy

$X \in \mathbb{R}_+$  : Years in school

**Task:** Given your education, predict your life expectancy

$X \in \mathbb{R}_+$  : Years in school

$Y \in \mathbb{R}_+$  : Age of death

**Task:** Given your education, predict your life expectancy

$X \in \mathbb{R}_+$  : Years in school

$Y \in \mathbb{R}_+$  : Age of death

$\Theta \in \mathbb{R}^2$  : Parameters

**Task:** Given your education, predict your life expectancy

$X \in \mathbb{R}_+$  : Years in school

$Y \in \mathbb{R}_+$  : Age of death

$\Theta \in \mathbb{R}^2$  : Parameters

$$f : X \times \Theta \mapsto Y$$

**Task:** Given your education, predict your life expectancy

$X \in \mathbb{R}_+$  : Years in school

$Y \in \mathbb{R}_+$  : Age of death

$\Theta \in \mathbb{R}^2$  : Parameters

$$f : X \times \Theta \mapsto Y$$

**Approach:** Learn the parameters  $\theta$  such that

$$f(x, \theta) = y; \quad x \in X, y \in Y$$

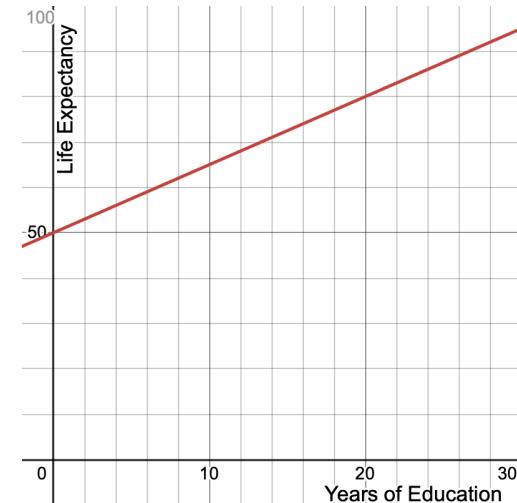
# Review

Started with a linear function  $f$

# Review

Started with a linear function  $f$

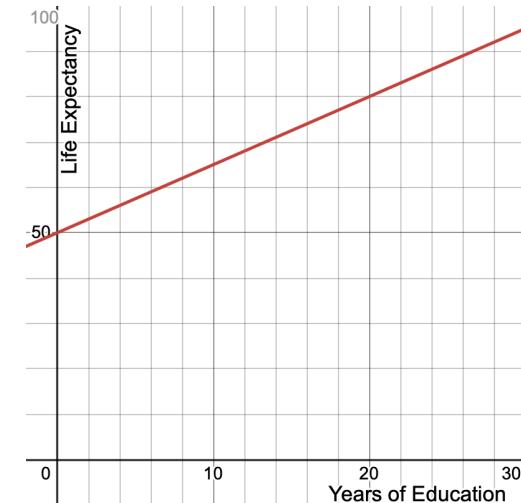
$$f(x, \theta) = f\left(x, \begin{bmatrix} \theta_1 \\ \theta_0 \end{bmatrix}\right) = \theta_1 x + \theta_0$$



# Review

Started with a linear function  $f$

$$f(x, \theta) = f\left(x, \begin{bmatrix} \theta_1 \\ \theta_0 \end{bmatrix}\right) = \theta_1 x + \theta_0$$

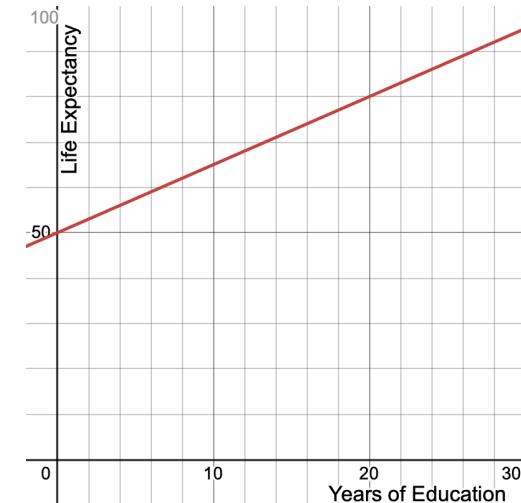


Then, we derived the square error function

# Review

Started with a linear function  $f$

$$f(x, \theta) = f\left(x, \begin{bmatrix} \theta_1 \\ \theta_0 \end{bmatrix}\right) = \theta_1 x + \theta_0$$



Then, we derived the square error function

$$\text{error}(f(x, \theta), y) = (f(x, \theta) - y)^2$$

# Review

We wrote the loss function for a single datapoint  $x_{[i]}, y_{[i]}$  using the square error

$$\mathcal{L}(x_{[i]}, y_{[i]}, \theta) = \text{error}\left(f(x_{[i]}, \theta), y_{[i]}\right) = \left(f(x_{[i]}, \theta) - y_{[i]}\right)^2$$

# Review

We wrote the loss function for a single datapoint  $x_{[i]}, y_{[i]}$  using the square error

$$\mathcal{L}(x_{[i]}, y_{[i]}, \theta) = \text{error}\left(f(x_{[i]}, \theta), y_{[i]}\right) = \left(f(x_{[i]}, \theta) - y_{[i]}\right)^2$$

But we wanted to learn a model over **all** the data, not a single datapoint

# Review

We wrote the loss function for a single datapoint  $x_{[i]}, y_{[i]}$  using the square error

$$\mathcal{L}(x_{[i]}, y_{[i]}, \theta) = \text{error}\left(f(x_{[i]}, \theta), y_{[i]}\right) = \left(f(x_{[i]}, \theta) - y_{[i]}\right)^2$$

But we wanted to learn a model over **all** the data, not a single datapoint

We wanted to make **new** predictions, to **generalize**

# Review

We wrote the loss function for a single datapoint  $x_{[i]}, y_{[i]}$  using the square error

$$\mathcal{L}(x_{[i]}, y_{[i]}, \theta) = \text{error}\left(f(x_{[i]}, \theta), y_{[i]}\right) = \left(f(x_{[i]}, \theta) - y_{[i]}\right)^2$$

But we wanted to learn a model over **all** the data, not a single datapoint

We wanted to make **new** predictions, to **generalize**

$$\mathbf{x} = [x_{[1]} \ x_{[2]} \ \dots \ x_{[n]}]^\top, \mathbf{y} = [y_{[1]} \ y_{[2]} \ \dots \ y_{[n]}]^\top$$

# Review

We wrote the loss function for a single datapoint  $x_{[i]}, y_{[i]}$  using the square error

$$\mathcal{L}(x_{[i]}, y_{[i]}, \theta) = \text{error}\left(f(x_{[i]}, \theta), y_{[i]}\right) = \left(f(x_{[i]}, \theta) - y_{[i]}\right)^2$$

But we wanted to learn a model over **all** the data, not a single datapoint

We wanted to make **new** predictions, to **generalize**

$$\mathbf{x} = [x_{[1]} \ x_{[2]} \ \dots \ x_{[n]}]^\top, \mathbf{y} = [y_{[1]} \ y_{[2]} \ \dots \ y_{[n]}]^\top$$

$$\mathcal{L}(\mathbf{x}, \mathbf{y}, \theta) = \sum_{i=1}^n \text{error}\left(f(x_{[i]}, \theta), y_{[i]}\right) = \sum_{i=1}^n \left(f(x_{[i]}, \theta) - y_{[i]}\right)^2$$

# Review

Our objective was to find the parameters that minimized the loss function over the dataset

# Review

Our objective was to find the parameters that minimized the loss function over the dataset

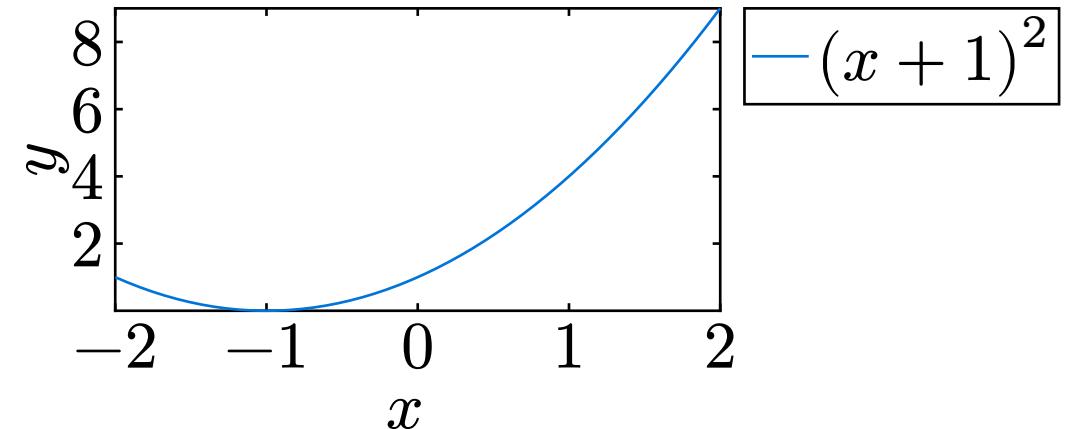
We introduced the arg min operator

# Review

Our objective was to find the parameters that minimized the loss function over the dataset

We introduced the arg min operator

$$f(x) = (x + 1)^2$$

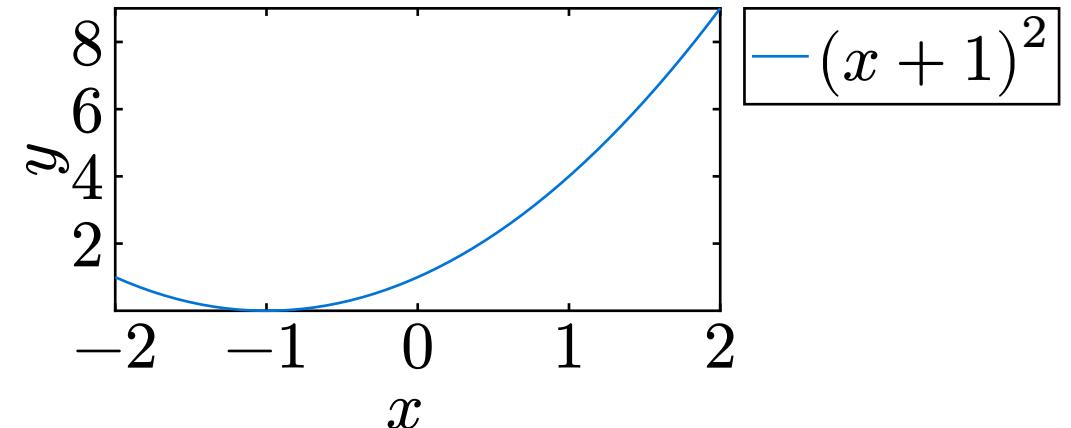


# Review

Our objective was to find the parameters that minimized the loss function over the dataset

We introduced the arg min operator

$$f(x) = (x + 1)^2$$



$$\arg \min_x f(x) = -1$$

# Review

With the  $\arg \min$  operator, we formally wrote our optimization objective

With the  $\arg \min$  operator, we formally wrote our optimization objective

$$\begin{aligned}\arg \min_{\theta} \mathcal{L}(x, y, \theta) &= \arg \min_{\theta} \sum_{i=1}^n \text{error}\left(f\left(x_{[i]}, \theta\right), y_{[i]}\right) \\ &= \arg \min_{\theta} \sum_{i=1}^n \left(f\left(x_{[i]}, \theta\right) - y_{[i]}\right)^2\end{aligned}$$

# Review

We defined the design matrix  $X_D$

We defined the design matrix  $\mathbf{X}_D$

$$\mathbf{X}_D = [\mathbf{x} \ 1] = \begin{bmatrix} x_{[1]} & 1 \\ x_{[2]} & 1 \\ \vdots & \vdots \\ x_{[n]} & 1 \end{bmatrix}$$

We defined the design matrix  $\mathbf{X}_D$

$$\mathbf{X}_D = [\mathbf{x} \ 1] = \begin{bmatrix} x_{[1]} & 1 \\ x_{[2]} & 1 \\ \vdots & \vdots \\ x_{[n]} & 1 \end{bmatrix}$$

We use the design matrix to find an **analytical** solution to the optimization objective

We defined the design matrix  $\mathbf{X}_D$

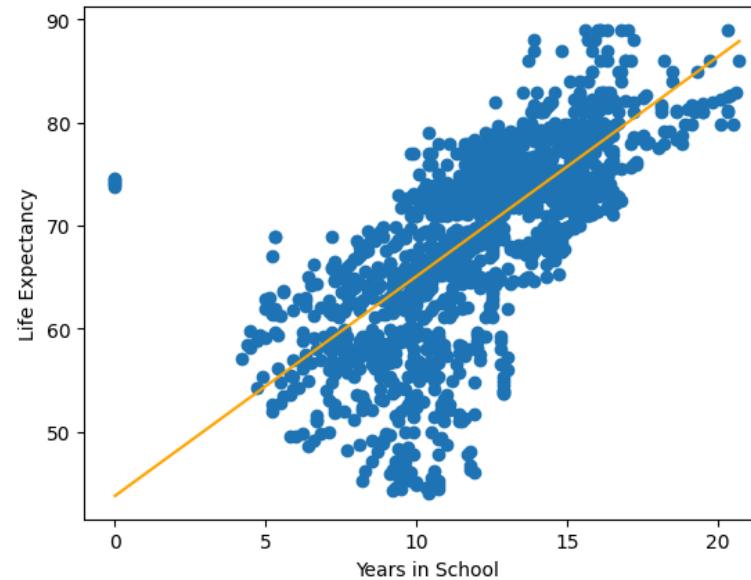
$$\mathbf{X}_D = [\mathbf{x} \ 1] = \begin{bmatrix} x_{[1]} & 1 \\ x_{[2]} & 1 \\ \vdots & \vdots \\ x_{[n]} & 1 \end{bmatrix}$$

We use the design matrix to find an **analytical** solution to the optimization objective

$$\boldsymbol{\theta} = (\mathbf{X}_D^\top \mathbf{X}_D)^{-1} \mathbf{X}_D^\top \mathbf{y}$$

With this analytical solution, we were able to learn a linear model

With this analytical solution, we were able to learn a linear model



Then, we used a trick to extend linear regression to nonlinear models

Then, we used a trick to extend linear regression to nonlinear models

$$\mathbf{X}_D = \begin{bmatrix} x_{[1]} & 1 \\ x_{[2]} & 1 \\ \vdots & \vdots \\ x_{[n]} & 1 \end{bmatrix} \Rightarrow \mathbf{X}_D = \begin{bmatrix} \log(1 + x_{[1]}) & 1 \\ \log(1 + x_{[2]}) & 1 \\ \vdots & \vdots \\ \log(1 + x_{[n]}) & 1 \end{bmatrix}$$

We extended to polynomials, which are **universal function approximators**

# Review

We extended to polynomials, which are **universal function approximators**

$$X_D = \begin{bmatrix} x_{[1]} & 1 \\ x_{[2]} & 1 \\ \vdots & \vdots \\ x_{[n]} & 1 \end{bmatrix} \Rightarrow X_D = \begin{bmatrix} x_{[1]}^m & x_{[1]}^{m-1} & \dots & x_{[1]} & 1 \\ x_{[2]}^m & x_{[2]}^{m-1} & \dots & x_{[2]} & 1 \\ \vdots & \vdots & \ddots & & \\ x_{[n]}^m & x_{[n]}^{m-1} & \dots & x_{[n]} & 1 \end{bmatrix}$$

# Review

We extended to polynomials, which are **universal function approximators**

$$X_D = \begin{bmatrix} x_{[1]} & 1 \\ x_{[2]} & 1 \\ \vdots & \vdots \\ x_{[n]} & 1 \end{bmatrix} \Rightarrow X_D = \begin{bmatrix} x_{[1]}^m & x_{[1]}^{m-1} & \dots & x_{[1]} & 1 \\ x_{[2]}^m & x_{[2]}^{m-1} & \dots & x_{[2]} & 1 \\ \vdots & \vdots & \ddots & & \\ x_{[n]}^m & x_{[n]}^{m-1} & \dots & x_{[n]} & 1 \end{bmatrix}$$

$$f : X \times \Theta \mapsto \mathbb{R}$$

We extended to polynomials, which are **universal function approximators**

$$X_D = \begin{bmatrix} x_{[1]} & 1 \\ x_{[2]} & 1 \\ \vdots & \vdots \\ x_{[n]} & 1 \end{bmatrix} \Rightarrow X_D = \begin{bmatrix} x_{[1]}^m & x_{[1]}^{m-1} & \dots & x_{[1]} & 1 \\ x_{[2]}^m & x_{[2]}^{m-1} & \dots & x_{[2]} & 1 \\ \vdots & \vdots & \ddots & & \\ x_{[n]}^m & x_{[n]}^{m-1} & \dots & x_{[n]} & 1 \end{bmatrix}$$

$$f : X \times \Theta \mapsto \mathbb{R}$$

$$\Theta \in \mathbb{R}^2 \Rightarrow \Theta \in \mathbb{R}^{m+1}$$

Finally, we discussed overfitting

# Review

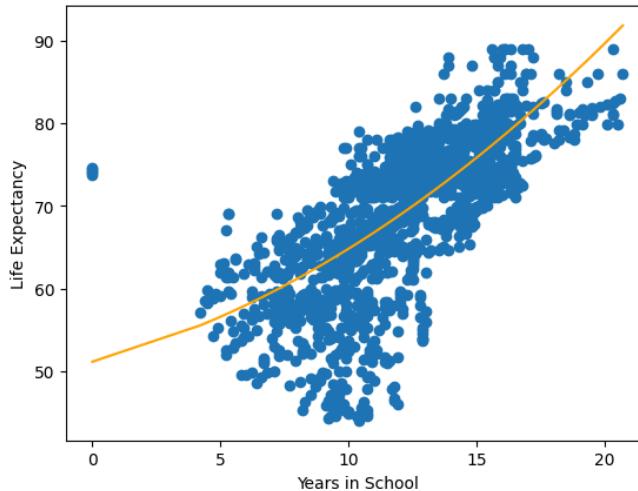
Finally, we discussed overfitting

$$f(x, \theta) = \theta_m x^m + \theta_{m-1} x^{m-1}, \dots, \theta_1 x^1 + \theta_0$$

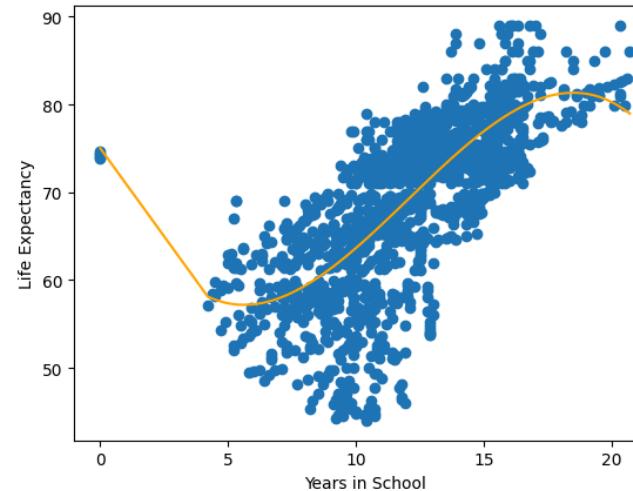
# Review

Finally, we discussed overfitting

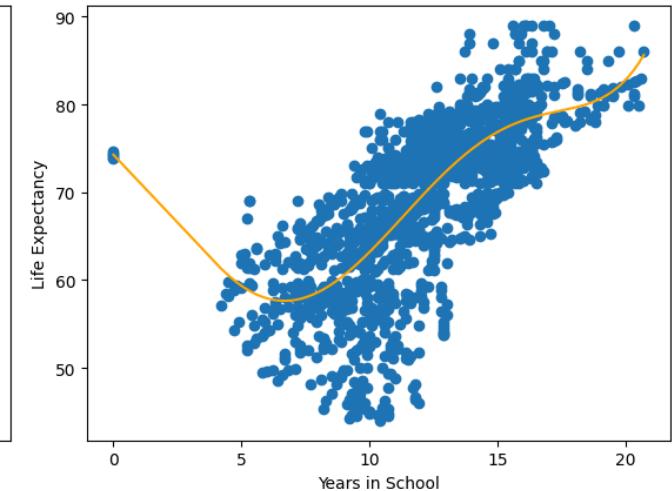
$$f(x, \theta) = \theta_m x^m + \theta_{m-1} x^{m-1}, \dots, \theta_1 x^1 + \theta_0$$



$$m = 2$$



$$m = 3$$



$$m = 5$$

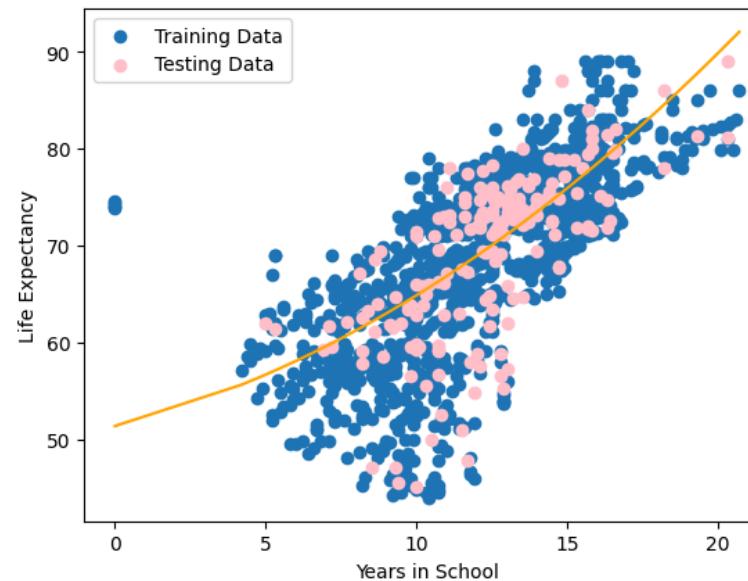
We care about **generalization** in machine learning

We care about **generalization** in machine learning

So we should always split our dataset into a training dataset and a testing dataset

We care about **generalization** in machine learning

So we should always split our dataset into a training dataset and a testing dataset



1. Review
2. Multivariate linear regression
3. Limitations of linear regression
4. History of neural networks
5. Biological neurons
6. Artificial neurons
7. Wide neural networks
8. Deep neural networks
9. Practical considerations

1. Review
2. **Multivariate linear regression**
3. Limitations of linear regression
4. History of neural networks
5. Biological neurons
6. Artificial neurons
7. Wide neural networks
8. Deep neural networks
9. Practical considerations

Last time, we assumed a single-input system

Last time, we assumed a single-input system

Years of education:  $X \in \mathbb{R}$

Last time, we assumed a single-input system

Years of education:  $X \in \mathbb{R}$

But sometimes we want to consider multiple input dimensions

Last time, we assumed a single-input system

Years of education:  $X \in \mathbb{R}$

But sometimes we want to consider multiple input dimensions

Years of education, BMI, GDP:  $X \in \mathbb{R}^3$

Last time, we assumed a single-input system

Years of education:  $X \in \mathbb{R}$

But sometimes we want to consider multiple input dimensions

Years of education, BMI, GDP:  $X \in \mathbb{R}^3$

We can solve these problems using linear regression too

For multivariate problems, we will define the input dimension as  $d_x$

For multivariate problems, we will define the input dimension as  $d_x$

$$\boldsymbol{x} \in X; \quad X \in \mathbb{R}^{d_x}$$

For multivariate problems, we will define the input dimension as  $d_x$

$$\boldsymbol{x} \in X; \quad X \in \mathbb{R}^{d_x}$$

We will write the vectors as

$$\boldsymbol{x}_{[i]} = \begin{bmatrix} x_{[i],1} \\ x_{[i],2} \\ \vdots \\ x_{[i],d_x} \end{bmatrix}$$

For multivariate problems, we will define the input dimension as  $d_x$

$$\boldsymbol{x} \in X; \quad X \in \mathbb{R}^{d_x}$$

We will write the vectors as

$$\boldsymbol{x}_{[i]} = \begin{bmatrix} x_{[i],1} \\ x_{[i],2} \\ \vdots \\ x_{[i],d_x} \end{bmatrix}$$

$x_{[i],1}$  refers to the first dimension of training data  $i$

The design matrix for a **multivariate** linear system is

$$\mathbf{X}_D = \begin{bmatrix} x_{[1],d_x} & x_{[1],d_x-1} & \cdots & x_{[1],1} & 1 \\ x_{[2],d_x} & x_{[2],d_x-1} & \cdots & x_{[2],1} & 1 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ x_{[n],d_x} & x_{[n],d_x-1} & \cdots & x_{[n],1} & 1 \end{bmatrix}$$

The design matrix for a **multivariate** linear system is

$$\mathbf{X}_D = \begin{bmatrix} x_{[1],d_x} & x_{[1],d_x-1} & \cdots & x_{[1],1} & 1 \\ x_{[2],d_x} & x_{[2],d_x-1} & \cdots & x_{[2],1} & 1 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ x_{[n],d_x} & x_{[n],d_x-1} & \cdots & x_{[n],1} & 1 \end{bmatrix}$$

Remember  $x_{[n],d_x}$  refers to dimension  $d_x$  of training data  $n$

The design matrix for a **multivariate** linear system is

$$\mathbf{X}_D = \begin{bmatrix} x_{[1],d_x} & x_{[1],d_x-1} & \cdots & x_{[1],1} & 1 \\ x_{[2],d_x} & x_{[2],d_x-1} & \cdots & x_{[2],1} & 1 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ x_{[n],d_x} & x_{[n],d_x-1} & \cdots & x_{[n],1} & 1 \end{bmatrix}$$

Remember  $x_{[n],d_x}$  refers to dimension  $d_x$  of training data  $n$

The solution is the same as before

$$\boldsymbol{\theta} = (\mathbf{X}_D^\top \mathbf{X}_D)^{-1} \mathbf{X}_D^\top \mathbf{y}$$

# Agenda

1. Review
2. **Multivariate linear regression**
3. Limitations of linear regression
4. History of neural networks
5. Biological neurons
6. Artificial neurons
7. Wide neural networks
8. Deep neural networks
9. Practical considerations

# Agenda

1. Review
2. Multivariate linear regression
3. **Limitations of linear regression**
4. History of neural networks
5. Biological neurons
6. Artificial neurons
7. Wide neural networks
8. Deep neural networks
9. Practical considerations

# Limitations of Linear Regression

Linear models are useful for certain problems

# Limitations of Linear Regression

Linear models are useful for certain problems

1. Analytical solution

# Limitations of Linear Regression

Linear models are useful for certain problems

1. Analytical solution
2. Low data requirement

# Limitations of Linear Regression

Linear models are useful for certain problems

1. Analytical solution
2. Low data requirement

Issues arise with other problems

# Limitations of Linear Regression

Linear models are useful for certain problems

1. Analytical solution
2. Low data requirement

Issues arise with other problems

1. Poor scalability

# Limitations of Linear Regression

Linear models are useful for certain problems

1. Analytical solution
2. Low data requirement

Issues arise with other problems

1. Poor scalability
2. Polynomials do not generalize well

# Limitations of Linear Regression

Issues arise with other problems

1. **Poor scalability**
2. Polynomials do not generalize well

# Limitations of Linear Regression

So far, we have seen:

# Limitations of Linear Regression

So far, we have seen:

One-dimensional polynomial  
functions

$$X_D = \begin{bmatrix} x_{[1]}^m & x_{[1]}^{m-1} & \dots & x_{[1]} & 1 \\ x_{[2]}^m & x_{[2]}^{m-1} & \dots & x_{[2]} & 1 \\ \vdots & \vdots & \ddots & & \\ x_{[n]}^m & x_{[n]}^{m-1} & \dots & x_{[n]} & 1 \end{bmatrix}$$

# Limitations of Linear Regression

So far, we have seen:

One-dimensional polynomial  
functions

$$\mathbf{X}_D = \begin{bmatrix} x_{[1]}^m & x_{[1]}^{m-1} & \dots & x_{[1]} & 1 \\ x_{[2]}^m & x_{[2]}^{m-1} & \dots & x_{[2]} & 1 \\ \vdots & \vdots & \ddots & & \\ x_{[n]}^m & x_{[n]}^{m-1} & \dots & x_{[n]} & 1 \end{bmatrix}$$

Multi-dimensional linear functions

$$\mathbf{X}_D = \begin{bmatrix} x_{[1],d_x} & x_{[1],d_x-1} & \dots & 1 \\ x_{[2],d_x} & x_{[2],d_x-1} & \dots & 1 \\ \vdots & \vdots & \ddots & \vdots \\ x_{[n],d_x} & x_{[n],d_x-1} & \dots & 1 \end{bmatrix}$$

# Limitations of Linear Regression

So far, we have seen:

One-dimensional polynomial  
functions

$$\mathbf{X}_D = \begin{bmatrix} x_{[1]}^m & x_{[1]}^{m-1} & \dots & x_{[1]} & 1 \\ x_{[2]}^m & x_{[2]}^{m-1} & \dots & x_{[2]} & 1 \\ \vdots & \vdots & \ddots & & \\ x_{[n]}^m & x_{[n]}^{m-1} & \dots & x_{[n]} & 1 \end{bmatrix}$$

Multi-dimensional linear functions

$$\mathbf{X}_D = \begin{bmatrix} x_{[1],d_x} & x_{[1],d_x-1} & \dots & 1 \\ x_{[2],d_x} & x_{[2],d_x-1} & \dots & 1 \\ \vdots & \vdots & \ddots & \vdots \\ x_{[n],d_x} & x_{[n],d_x-1} & \dots & 1 \end{bmatrix}$$

Combine them to create multi-dimensional polynomial functions

# Limitations of Linear Regression

So far, we have seen:

One-dimensional polynomial  
functions

$$\mathbf{X}_D = \begin{bmatrix} x_{[1]}^m & x_{[1]}^{m-1} & \dots & x_{[1]} & 1 \\ x_{[2]}^m & x_{[2]}^{m-1} & \dots & x_{[2]} & 1 \\ \vdots & \vdots & \ddots & & \\ x_{[n]}^m & x_{[n]}^{m-1} & \dots & x_{[n]} & 1 \end{bmatrix}$$

Multi-dimensional linear functions

$$\mathbf{X}_D = \begin{bmatrix} x_{[1],d_x} & x_{[1],d_x-1} & \dots & 1 \\ x_{[2],d_x} & x_{[2],d_x-1} & \dots & 1 \\ \vdots & \vdots & \ddots & \vdots \\ x_{[n],d_x} & x_{[n],d_x-1} & \dots & 1 \end{bmatrix}$$

Combine them to create multi-dimensional polynomial functions

# Limitations of Linear Regression

Let us do an example

# Limitations of Linear Regression

Let us do an example

**Task:** predict how many ❤ a photo gets on social media



# Limitations of Linear Regression

Let us do an example

**Task:** predict how many ❤ a photo gets on social media



$$f : X \times \Theta \mapsto Y; \quad X : \text{Image}, \quad Y : \text{Number of ❤}$$

# Limitations of Linear Regression

Let us do an example

**Task:** predict how many ❤ a photo gets on social media



$f : X \times \Theta \mapsto Y; \quad X : \text{Image}, \quad Y : \text{Number of ❤}$

$$X \in \mathbb{Z}_+^{256 \times 256} = \mathbb{Z}_+^{65536}; \quad Y \in \mathbb{Z}_+$$

# Limitations of Linear Regression

Let us do an example

**Task:** predict how many ❤ a photo gets on social media



$$f : X \times \Theta \mapsto Y; \quad X : \text{Image}, \quad Y : \text{Number of ❤}$$

$$X \in \mathbb{Z}_+^{256 \times 256} = \mathbb{Z}_+^{65536}; \quad Y \in \mathbb{Z}_+$$

Highly nonlinear task, use a polynomial with order  $m = 20$

# Limitations of Linear Regression

$$\mathbf{X}_D = [\mathbf{x}_{D,[1]} \ \cdots \ \mathbf{x}_{D,[n]}]^\top$$

# Limitations of Linear Regression

$$\mathbf{X}_D = [\mathbf{x}_{D,[1]} \ \dots \ \mathbf{x}_{D,[n]}]^\top$$

$$\mathbf{x}_{D,[i]} =$$

$$\left[ \underbrace{\mathbf{x}_{[i],d_x}^m \mathbf{x}_{[i],d_x-1}^m \dots \mathbf{x}_{[i],1}^m}_{(d_x \Rightarrow 1, x^m)} \ \underbrace{\mathbf{x}_{[i],d_x}^m \mathbf{x}_{[i],d_x-1}^m \dots \mathbf{x}_{[i],2}^m}_{(d_x \Rightarrow 2, x^m)} \ \dots \ \underbrace{\mathbf{x}_{[i],d_x}^{m-1} \mathbf{x}_{[i],d_x-1}^{m-1} \dots \mathbf{x}_{[i],1}^m}_{(d_x \Rightarrow 1, x^{m-1})} \ \dots \right]$$

**Question:** How many columns in this matrix?

# Limitations of Linear Regression

$$\mathbf{X}_D = [\mathbf{x}_{D,[1]} \ \dots \ \mathbf{x}_{D,[n]}]^\top$$

$$\mathbf{x}_{D,[i]} =$$

$$\left[ \underbrace{\mathbf{x}_{[i],d_x}^m \mathbf{x}_{[i],d_x-1}^m \dots \mathbf{x}_{[i],1}^m}_{(d_x \Rightarrow 1, x^m)} \ \underbrace{\mathbf{x}_{[i],d_x}^m \mathbf{x}_{[i],d_x-1}^m \dots \mathbf{x}_{[i],2}^m}_{(d_x \Rightarrow 2, x^m)} \ \dots \ \underbrace{\mathbf{x}_{[i],d_x}^{m-1} \mathbf{x}_{[i],d_x-1}^{m-1} \dots \mathbf{x}_{[i],1}^m}_{(d_x \Rightarrow 1, x^{m-1})} \ \dots \right]$$

**Question:** How many columns in this matrix?

**Hint:**  $d_x = 2, m = 3: x^3 + y^3 + x^2y + y^2x + xy + x + y + 1$

# Limitations of Linear Regression

$$\mathbf{X}_D = [\mathbf{x}_{D,[1]} \ \dots \ \mathbf{x}_{D,[n]}]^\top$$

$$\mathbf{x}_{D,[i]} =$$

$$\left[ \underbrace{\mathbf{x}_{[i],d_x}^m \mathbf{x}_{[i],d_x-1}^m \dots \mathbf{x}_{[i],1}^m}_{(d_x \Rightarrow 1, x^m)} \ \underbrace{\mathbf{x}_{[i],d_x}^m \mathbf{x}_{[i],d_x-1}^m \dots \mathbf{x}_{[i],2}^m}_{(d_x \Rightarrow 2, x^m)} \ \dots \ \underbrace{\mathbf{x}_{[i],d_x}^{m-1} \mathbf{x}_{[i],d_x-1}^{m-1} \dots \mathbf{x}_{[i],1}^m}_{(d_x \Rightarrow 1, x^{m-1})} \ \dots \right]$$

**Question:** How many columns in this matrix?

**Hint:**  $d_x = 2, m = 3: x^3 + y^3 + x^2y + y^2x + xy + x + y + 1$

**Answer:**  $(d_x)^m = 65536^{20} + 1 \approx 10^{96}$

# Limitations of Linear Regression

How big is  $10^{96}$ ?

# Limitations of Linear Regression

How big is  $10^{96}$ ?

**Question:** How many atoms are there in the universe?

# Limitations of Linear Regression

How big is  $10^{96}$ ?

**Question:** How many atoms are there in the universe?

**Answer:**  $10^{82}$

# Limitations of Linear Regression

How big is  $10^{96}$ ?

**Question:** How many atoms are there in the universe?

**Answer:**  $10^{82}$

There is not enough matter in the universe to represent one row

# Limitations of Linear Regression

How big is  $10^{96}$ ?

**Question:** How many atoms are there in the universe?

**Answer:**  $10^{82}$

There is not enough matter in the universe to represent one row

We cannot predict how many ❤  
the picture will get



# Limitations of Linear Regression

How big is  $10^{96}$ ?

**Question:** How many atoms are there in the universe?

**Answer:**  $10^{82}$

There is not enough matter in the universe to represent one row

We cannot predict how many ❤  
the picture will get



Polynomial regression does not scale to large inputs

# Limitations of Linear Regression

Issues arise with other problems

1. **Poor scalability**
2. Polynomials do not generalize well

# Limitations of Linear Regression

Issues arise with other problems

1. Poor scalability
2. **Polynomials do not generalize well**

# Limitations of Linear Regression

What happens to polynomials outside of the support (dataset)?

# Limitations of Linear Regression

What happens to polynomials outside of the support (dataset)?

Take the limit of polynomials to see their behavior

# Limitations of Linear Regression

What happens to polynomials outside of the support (dataset)?

Take the limit of polynomials to see their behavior

$$\lim_{x \rightarrow \infty} \theta_m x^m + \theta_{m-1} x^{m-1} + \dots \quad \text{Equation of a polynomial}$$

# Limitations of Linear Regression

What happens to polynomials outside of the support (dataset)?

Take the limit of polynomials to see their behavior

$$\lim_{x \rightarrow \infty} \theta_m x^m + \theta_{m-1} x^{m-1} + \dots \quad \text{Equation of a polynomial}$$

$$\lim_{x \rightarrow \infty} x^m \left( \theta_m + \frac{\theta_{m-1}}{x} + \dots \right) \quad \text{Factor out } x^m$$

# Limitations of Linear Regression

What happens to polynomials outside of the support (dataset)?

Take the limit of polynomials to see their behavior

$$\lim_{x \rightarrow \infty} \theta_m x^m + \theta_{m-1} x^{m-1} + \dots \quad \text{Equation of a polynomial}$$

$$\lim_{x \rightarrow \infty} x^m \left( \theta_m + \frac{\theta_{m-1}}{x} + \dots \right) \quad \text{Factor out } x^m$$

$$\lim_{x \rightarrow \infty} x^m \cdot \lim_{x \rightarrow \infty} \left( \theta_m + \frac{\theta_{m-1}}{x} + \dots \right) \quad \text{Split the limit (limit of products)}$$

# Limitations of Linear Regression

$$\lim_{x \rightarrow \infty} x^m \cdot \lim_{x \rightarrow \infty} \left( \theta_m + \frac{\theta_{m-1}}{x} + \dots \right) \quad \text{Split the limit (limit of products)}$$

$$\left( \lim_{x \rightarrow \infty} x^m \right) \cdot (\theta_m + 0 + \dots) \quad \text{Evaluate right limit}$$

# Limitations of Linear Regression

$$\lim_{x \rightarrow \infty} x^m \cdot \lim_{x \rightarrow \infty} \left( \theta_m + \frac{\theta_{m-1}}{x} + \dots \right) \quad \text{Split the limit (limit of products)}$$

$$\left( \lim_{x \rightarrow \infty} x^m \right) \cdot (\theta_m + 0 + \dots) \quad \text{Evaluate right limit}$$

$$\theta_m \lim_{x \rightarrow \infty} x^m \quad \text{Rewrite}$$

# Limitations of Linear Regression

$$\lim_{x \rightarrow \infty} x^m \cdot \lim_{x \rightarrow \infty} \left( \theta_m + \frac{\theta_{m-1}}{x} + \dots \right) \quad \text{Split the limit (limit of products)}$$

$$\left( \lim_{x \rightarrow \infty} x^m \right) \cdot (\theta_m + 0 + \dots) \quad \text{Evaluate right limit}$$

$$\theta_m \lim_{x \rightarrow \infty} x^m \quad \text{Rewrite}$$

$$\theta_m \lim_{x \rightarrow \infty} x^m = \infty \quad \text{If } \theta_m > 0$$

# Limitations of Linear Regression

$$\lim_{x \rightarrow \infty} x^m \cdot \lim_{x \rightarrow \infty} \left( \theta_m + \frac{\theta_{m-1}}{x} + \dots \right) \quad \text{Split the limit (limit of products)}$$

$$\left( \lim_{x \rightarrow \infty} x^m \right) \cdot (\theta_m + 0 + \dots) \quad \text{Evaluate right limit}$$

$$\theta_m \lim_{x \rightarrow \infty} x^m \quad \text{Rewrite}$$

$$\theta_m \lim_{x \rightarrow \infty} x^m = \infty \quad \text{If } \theta_m > 0$$

$$\theta_m \lim_{x \rightarrow \infty} x^m = -\infty \quad \text{If } \theta_m < 0$$

# Limitations of Linear Regression

Polynomials quickly tend towards  $-\infty, \infty$  outside of the support

# Limitations of Linear Regression

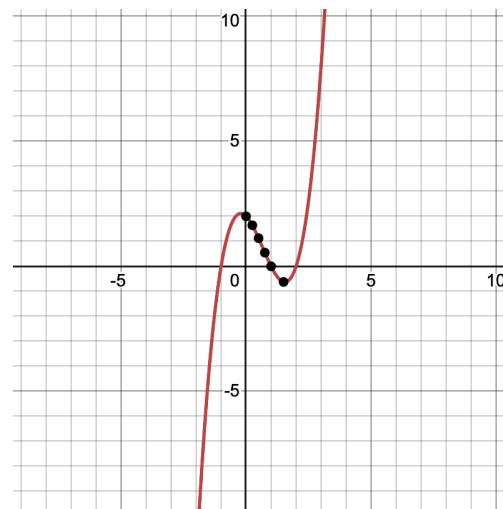
Polynomials quickly tend towards  $-\infty, \infty$  outside of the support

$$f(x) = x^3 - 2x^2 - x + 2$$

# Limitations of Linear Regression

Polynomials quickly tend towards  $-\infty, \infty$  outside of the support

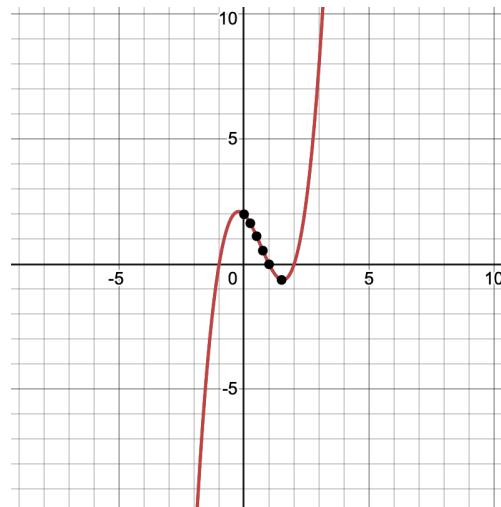
$$f(x) = x^3 - 2x^2 - x + 2$$



# Limitations of Linear Regression

Polynomials quickly tend towards  $-\infty, \infty$  outside of the support

$$f(x) = x^3 - 2x^2 - x + 2$$



Remember, to predict new data we want our functions to generalize

# Limitations of Linear Regression

Linear regression has issues

# Limitations of Linear Regression

Linear regression has issues

1. Poor scalability

# Limitations of Linear Regression

Linear regression has issues

1. Poor scalability
2. Polynomials do not generalize well

# Limitations of Linear Regression

We can use neural networks as an alternative to linear regression

# Limitations of Linear Regression

We can use neural networks as an alternative to linear regression

Neural network benefits:

# Limitations of Linear Regression

We can use neural networks as an alternative to linear regression

Neural network benefits:

1. Scale to large inputs

# Limitations of Linear Regression

We can use neural networks as an alternative to linear regression

Neural network benefits:

1. Scale to large inputs
2. Slightly better generalization

# Limitations of Linear Regression

We can use neural networks as an alternative to linear regression

Neural network benefits:

1. Scale to large inputs
2. Slightly better generalization

Drawbacks:

# Limitations of Linear Regression

We can use neural networks as an alternative to linear regression

Neural network benefits:

1. Scale to large inputs
2. Slightly better generalization

Drawbacks:

1. No analytical solution

# Limitations of Linear Regression

We can use neural networks as an alternative to linear regression

Neural network benefits:

1. Scale to large inputs
2. Slightly better generalization

Drawbacks:

1. No analytical solution
2. High data requirement

# Agenda

1. Review
2. Multivariate linear regression
3. **Limitations of linear regression**
4. History of neural networks
5. Biological neurons
6. Artificial neurons
7. Wide neural networks
8. Deep neural networks
9. Practical considerations

# Agenda

1. Review
2. Multivariate linear regression
3. Limitations of linear regression
4. **History of neural networks**
5. Biological neurons
6. Artificial neurons
7. Wide neural networks
8. Deep neural networks
9. Practical considerations

# History of Neural Networks

In 1939-1945, there was a World War

# History of Neural Networks

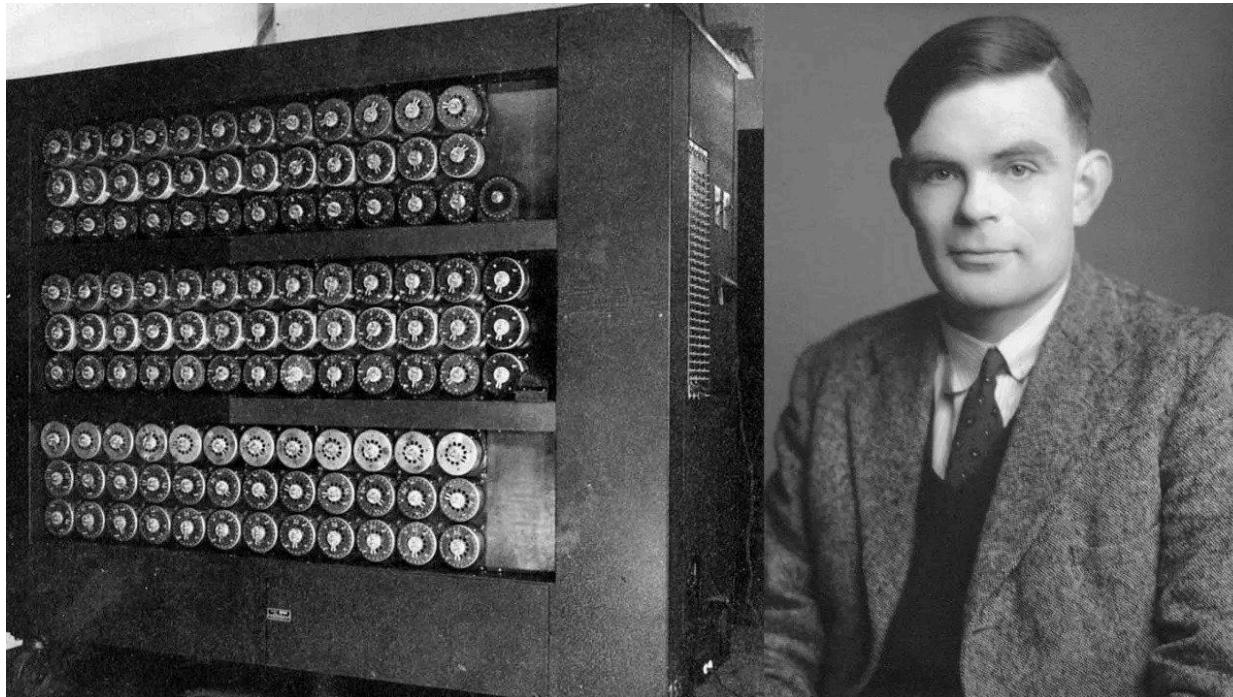
In 1939-1945, there was a World War

Militaries invested funding for research, and invented the computer

# History of Neural Networks

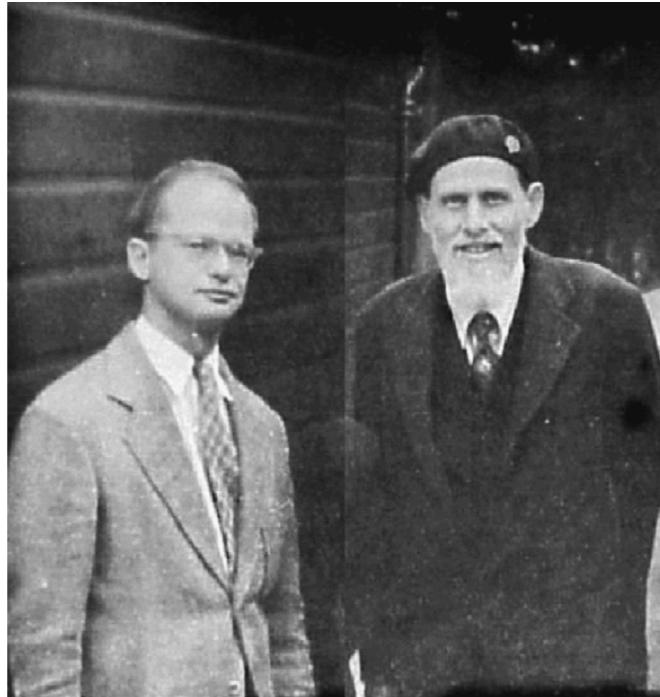
In 1939-1945, there was a World War

Militaries invested funding for research, and invented the computer



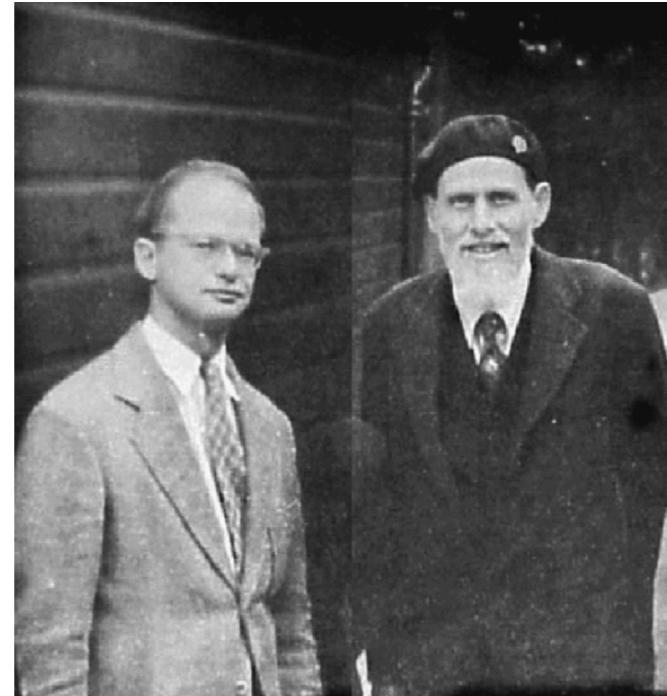
# History of Neural Networks

Meanwhile, a neuroscientist and mathematician (McCullough and Pitts) were trying to understand the human brain



# History of Neural Networks

Meanwhile, a neuroscientist and mathematician (McCullough and Pitts) were trying to understand the human brain



They designed the theory for the first neural network

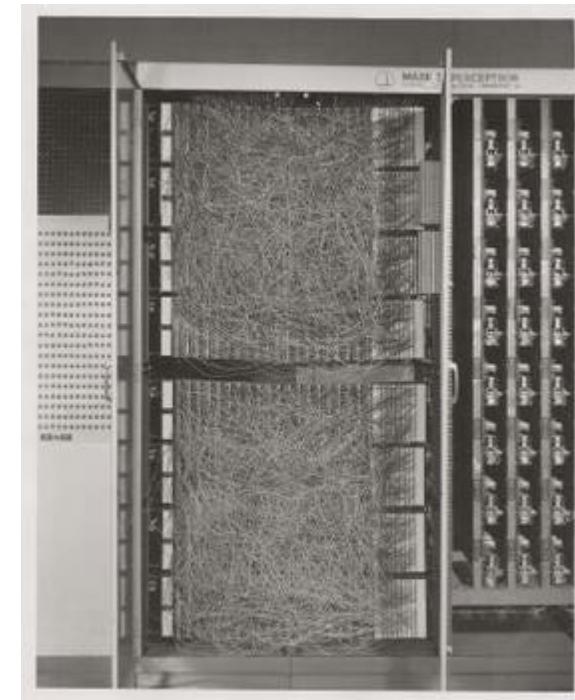
# History of Neural Networks

Rosenblatt implemented this neural network theory on a computer a few years later

# History of Neural Networks

Rosenblatt implemented this neural network theory on a computer a few years later

At the time, computers were very slow and expensive

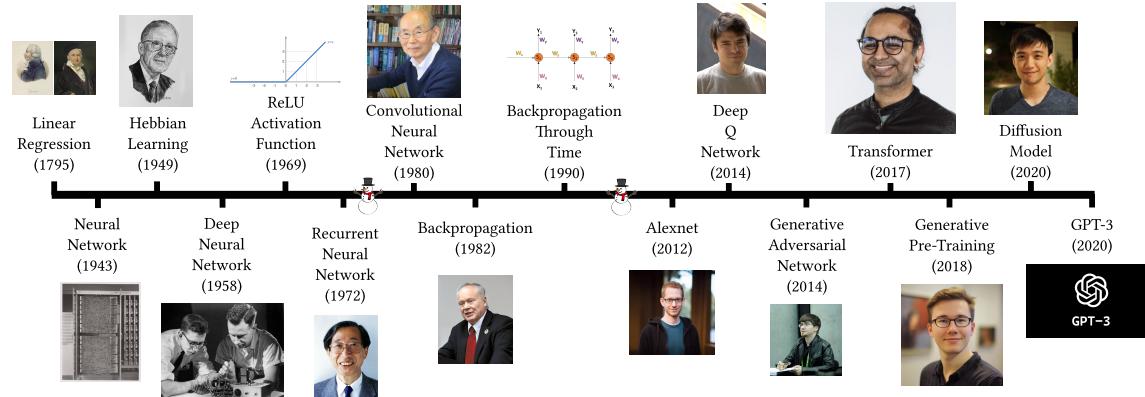


# History of Neural Networks

Through advances in theory and hardware, neural networks became slightly better

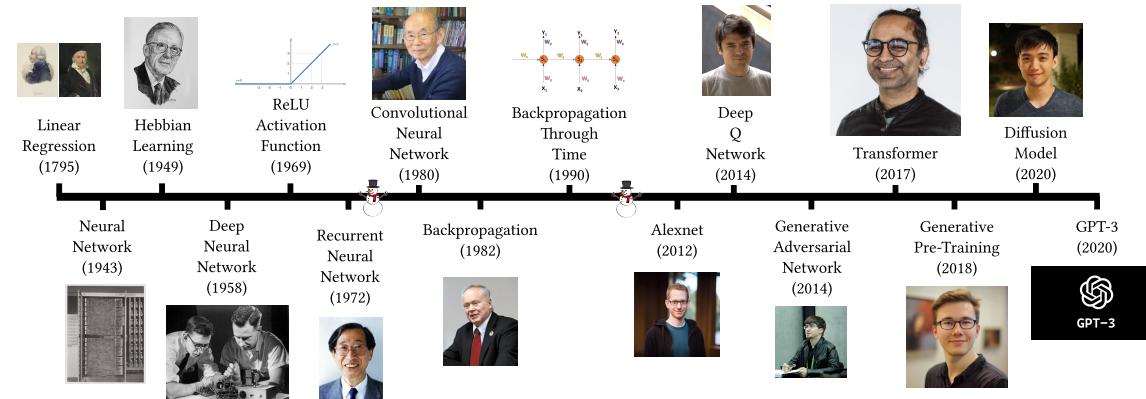
# History of Neural Networks

Through advances in theory and hardware, neural networks became slightly better



# History of Neural Networks

Through advances in theory and hardware, neural networks became slightly better



Around 2012, these improvements culminated in neural networks that perform like humans

# History of Neural Networks

So what is a neural network?

# History of Neural Networks

So what is a neural network?

It is a function, inspired by how the brain works

# History of Neural Networks

So what is a neural network?

It is a function, inspired by how the brain works

$$f : X \times \Theta \mapsto Y$$

# History of Neural Networks

Brains and neural networks rely on **neurons**

# History of Neural Networks

Brains and neural networks rely on **neurons**

**Brain:** Biological neurons → Biological neural network

# History of Neural Networks

Brains and neural networks rely on **neurons**

**Brain:** Biological neurons → Biological neural network

**Computer:** Artificial neurons → Artificial neural network

# History of Neural Networks

Brains and neural networks rely on **neurons**

**Brain:** Biological neurons → Biological neural network

**Computer:** Artificial neurons → Artificial neural network

First, let us review biological neurons

# History of Neural Networks

Brains and neural networks rely on **neurons**

**Brain:** Biological neurons → Biological neural network

**Computer:** Artificial neurons → Artificial neural network

First, let us review biological neurons

**Note:** I am not a neuroscientist! I may make simplifications or errors with biology

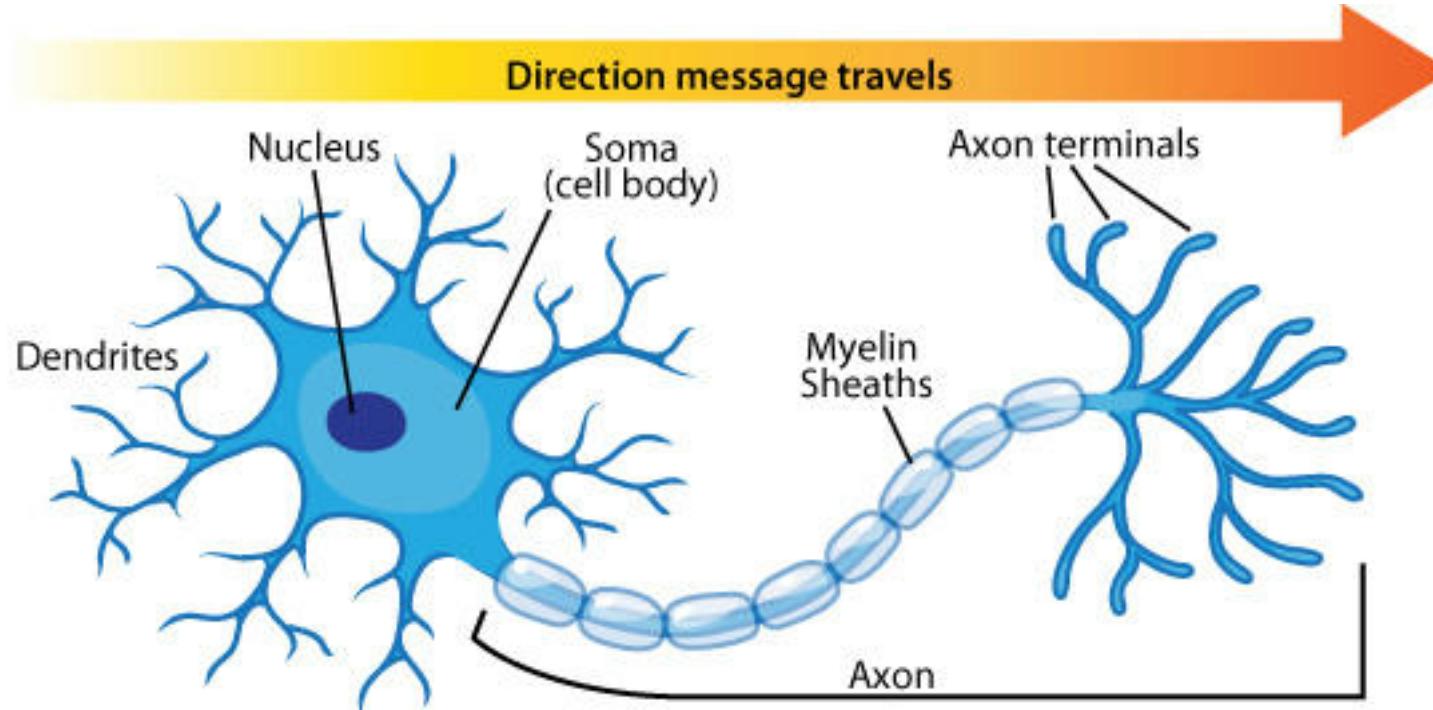
# Agenda

1. Review
2. Multivariate linear regression
3. Limitations of linear regression
4. **History of neural networks**
5. Biological neurons
6. Artificial neurons
7. Wide neural networks
8. Deep neural networks
9. Practical considerations

# Agenda

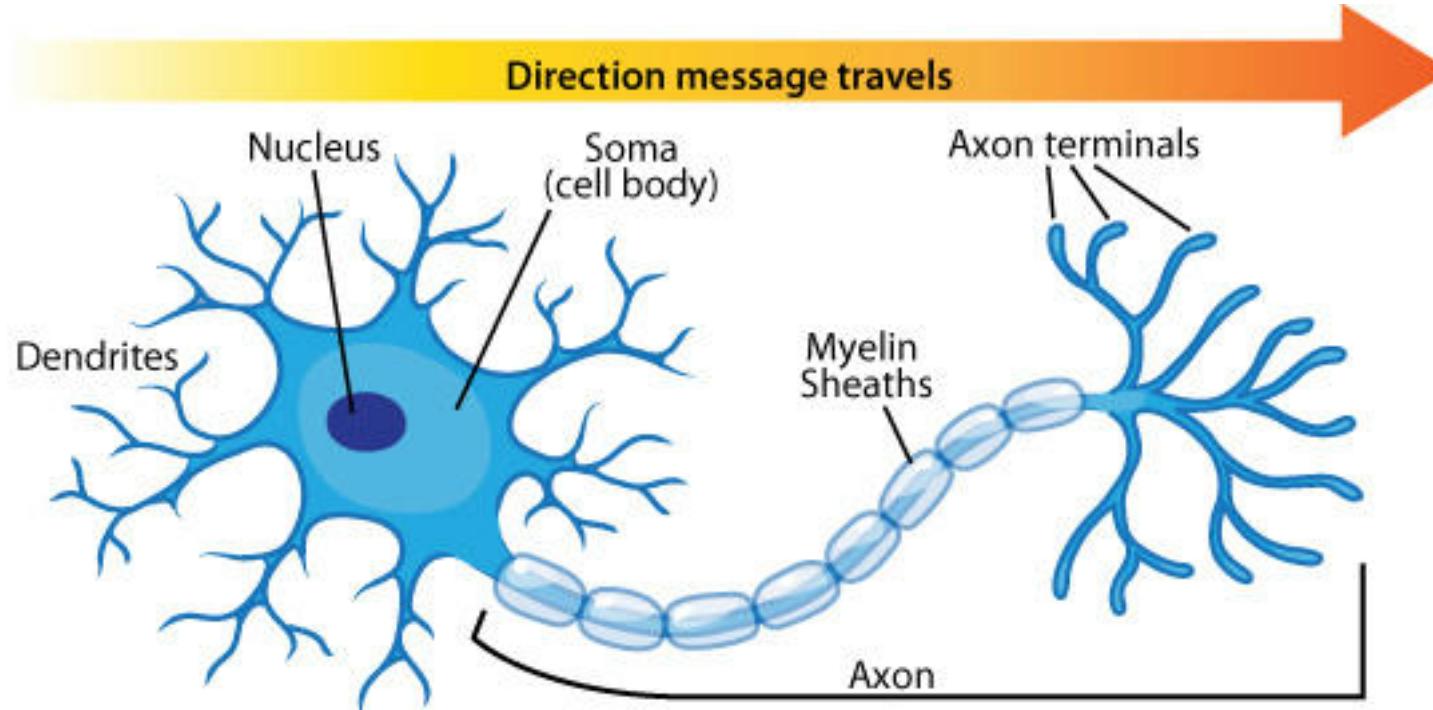
1. Review
2. Multivariate linear regression
3. Limitations of linear regression
4. History of neural networks
5. **Biological neurons**
6. Artificial neurons
7. Wide neural networks
8. Deep neural networks
9. Practical considerations

# Biological Neurons



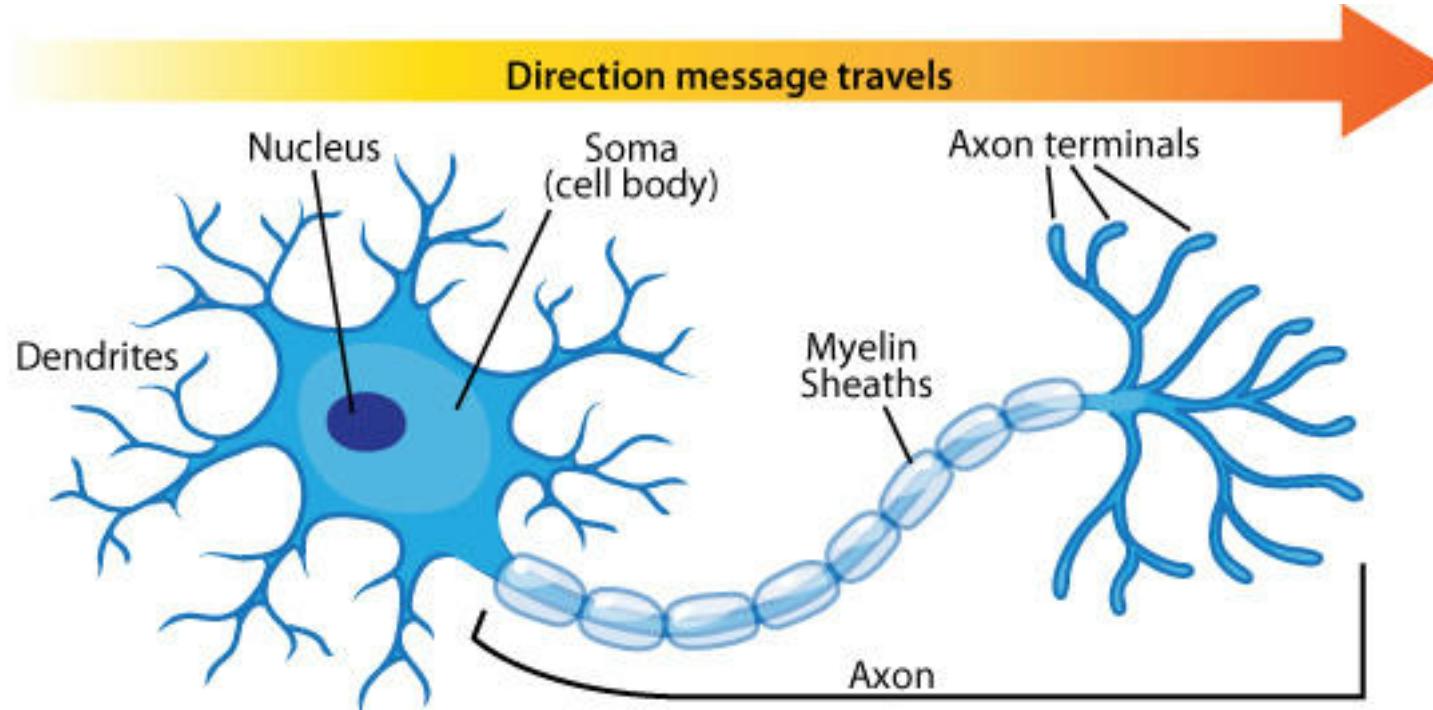
A simplified neuron consists of many parts

# Biological Neurons



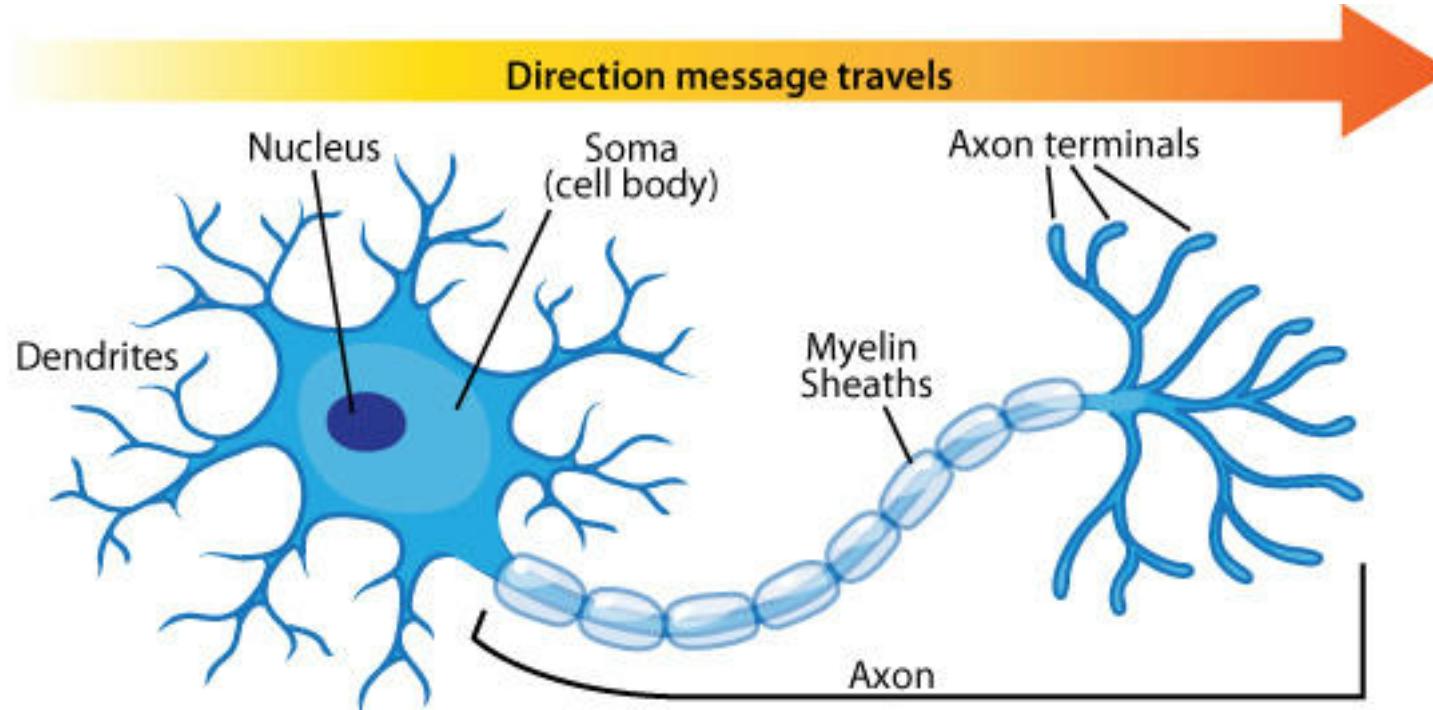
Neurons send messages based on messages received from other neurons

# Biological Neurons



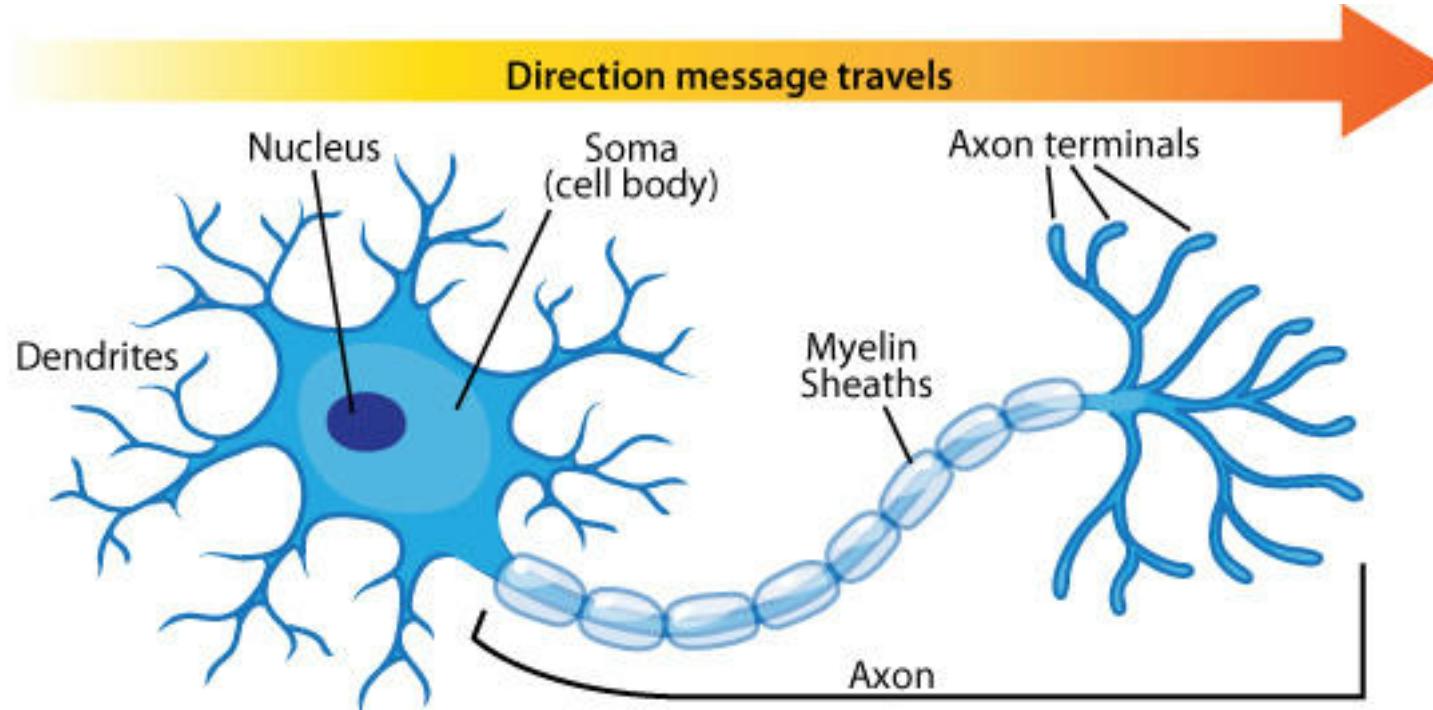
Incoming electrical signals travel along dendrites

# Biological Neurons



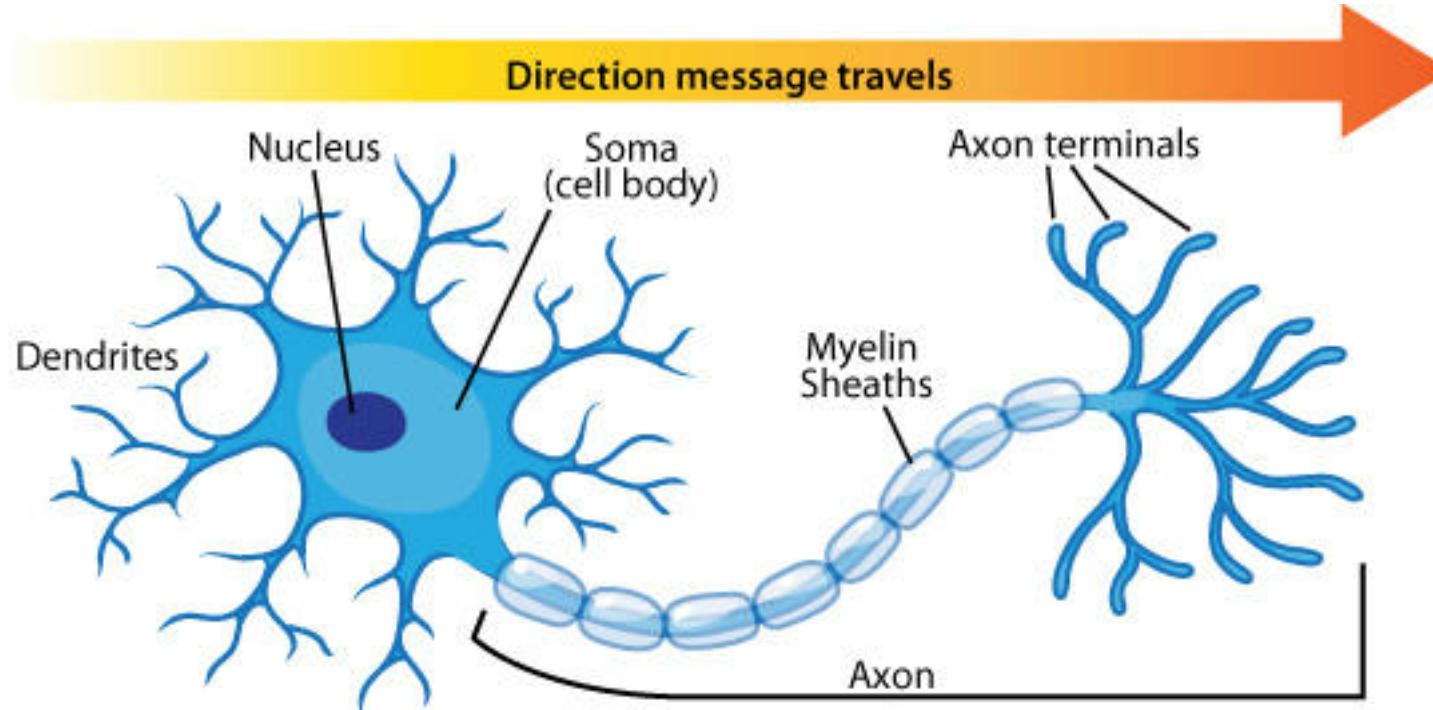
Dendrites are not all equal! Different dendrites have different diameters and structures

# Biological Neurons



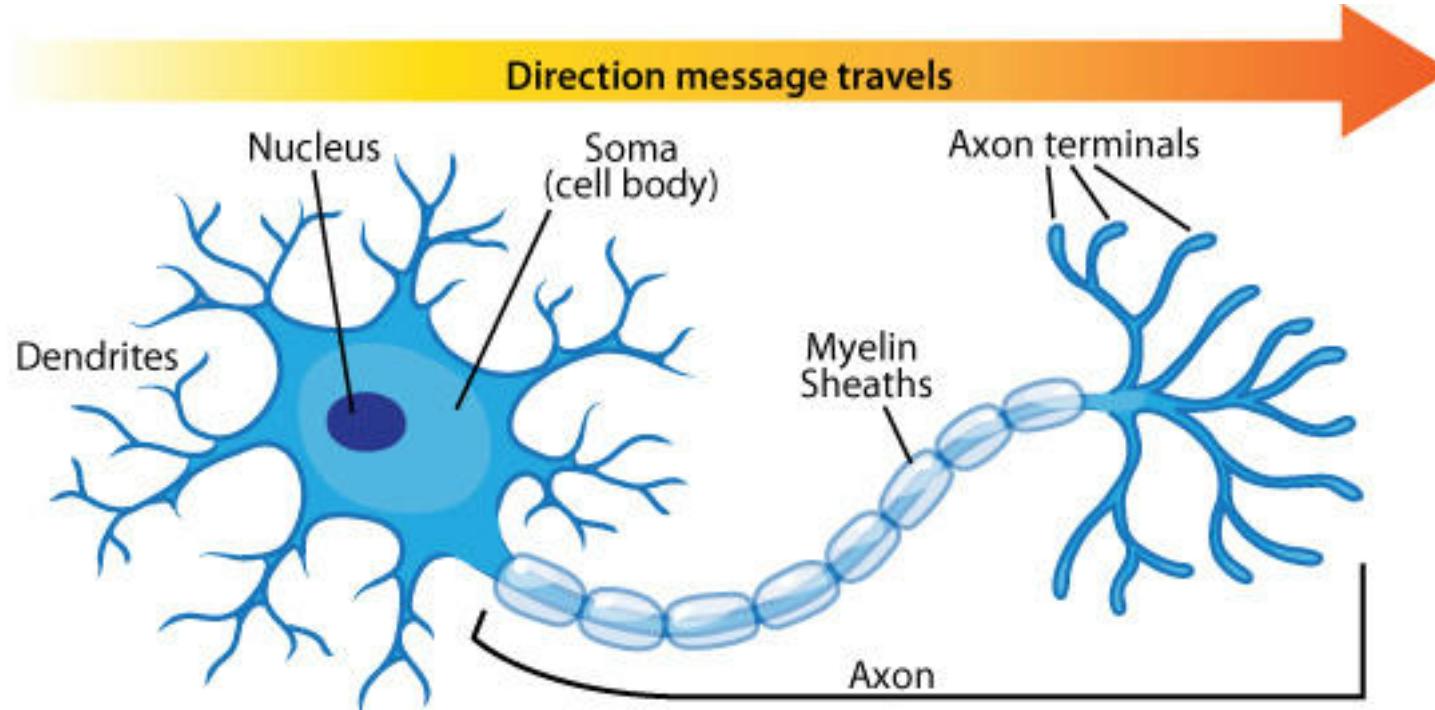
Electrical charges collect in the Soma (cell body)

# Biological Neurons



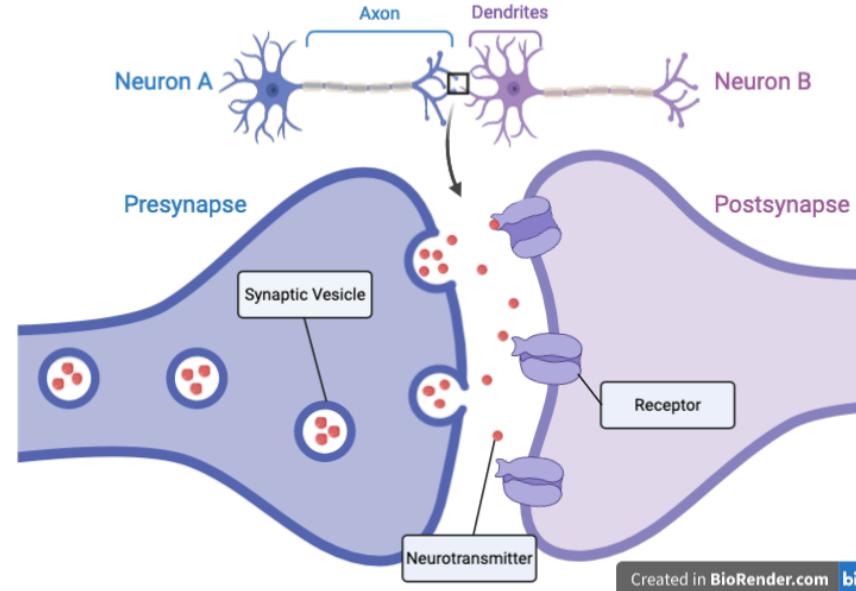
The axon outputs an electrical signal to other neurons

# Biological Neurons



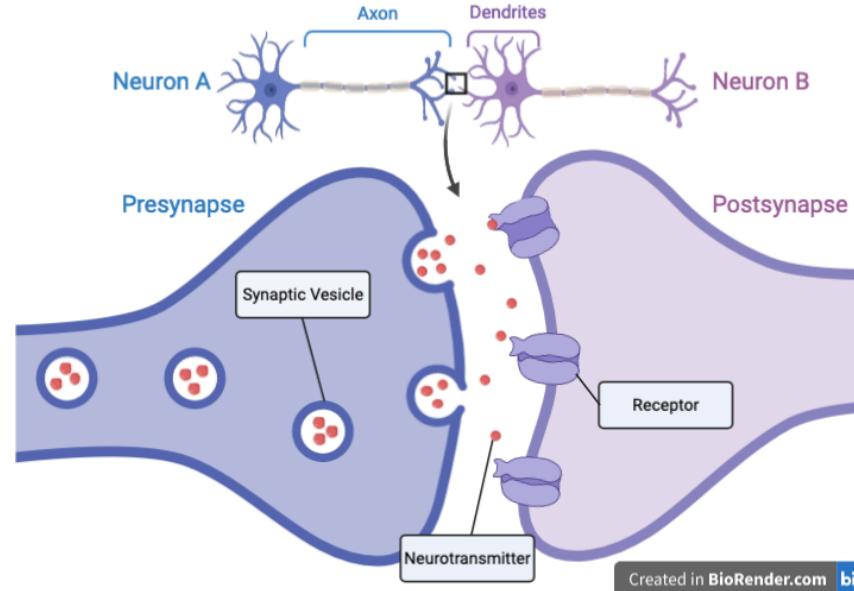
The axon terminals will connect to dendrites of other neurons through a synapse

# Biological Neurons



The synapse converts electrical signal, to chemical signal, back to electrical signal

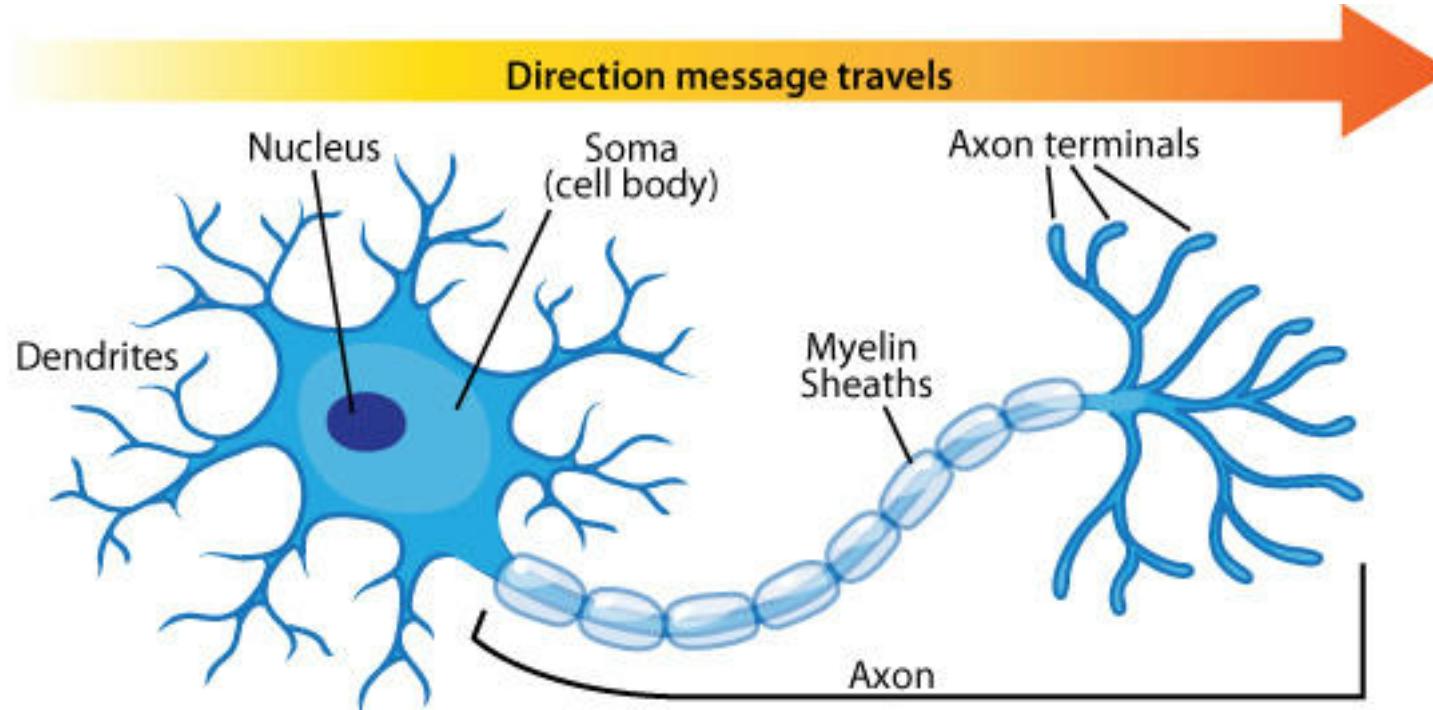
# Biological Neurons



The synapse converts electrical signal, to chemical signal, back to electrical signal

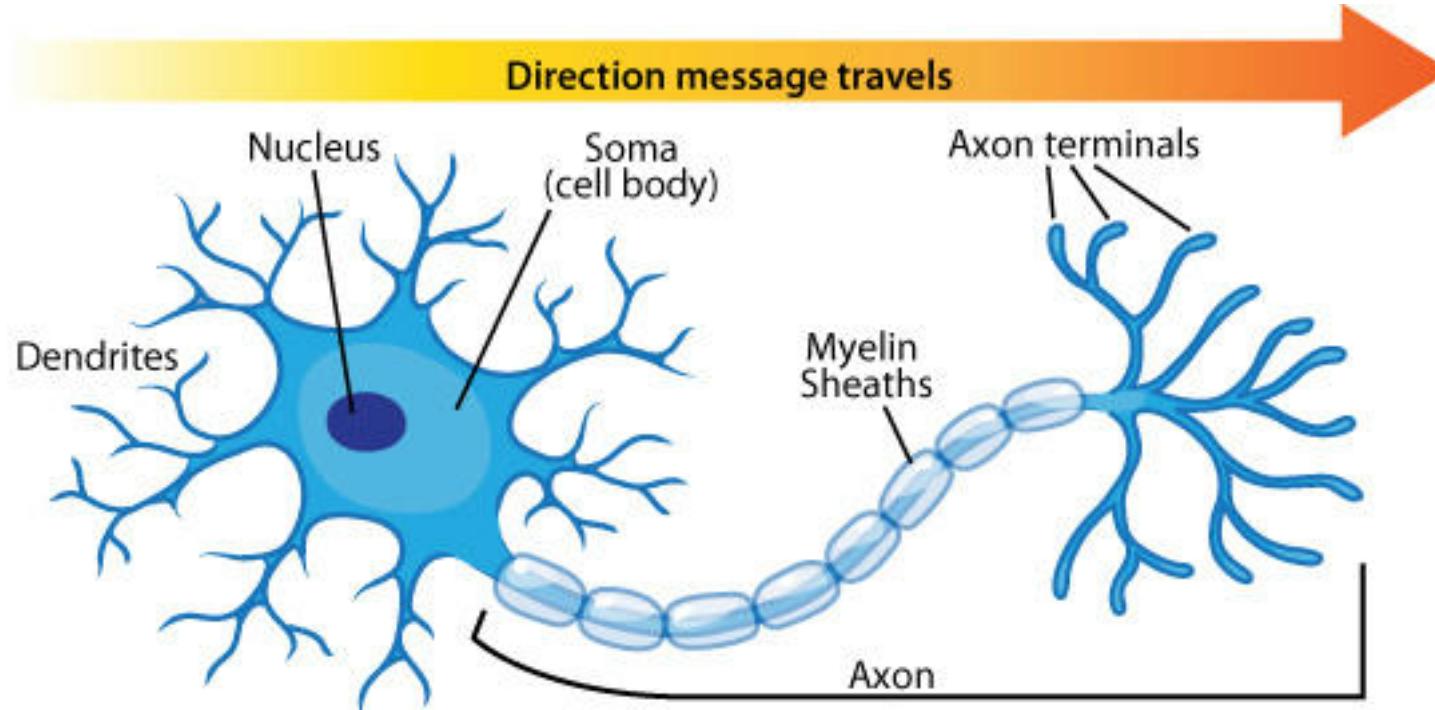
Synaptic weight determines how well a signal crosses the gap

# Biological Neurons



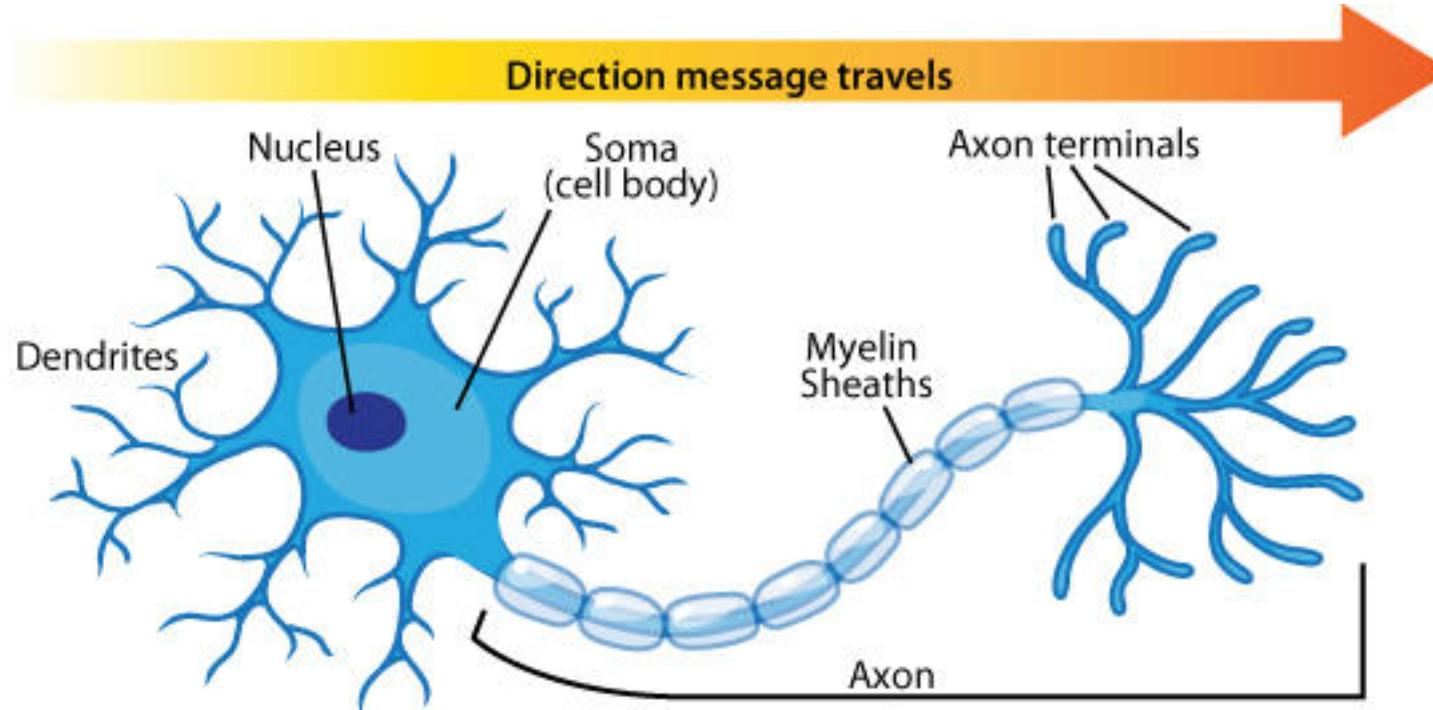
For our purposes, we can model the axon terminals, dendrites, and synapses to be one thing

# Biological Neurons



The neuron takes many inputs, and produces a single output

# Biological Neurons



The neuron will only output a signal down the axon (“fire”) at certain times

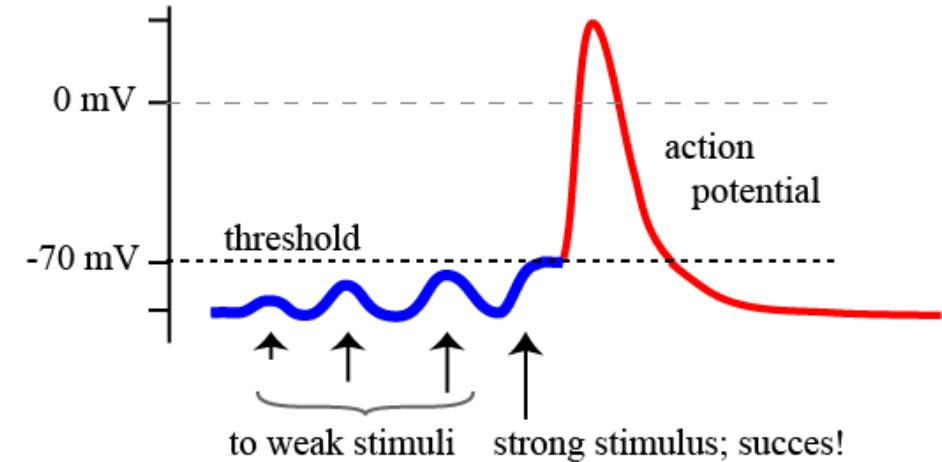
# Biological Neurons

How does a neuron decide to send an impulse (“fire”)?

# Biological Neurons

How does a neuron decide to send an impulse (“fire”)?

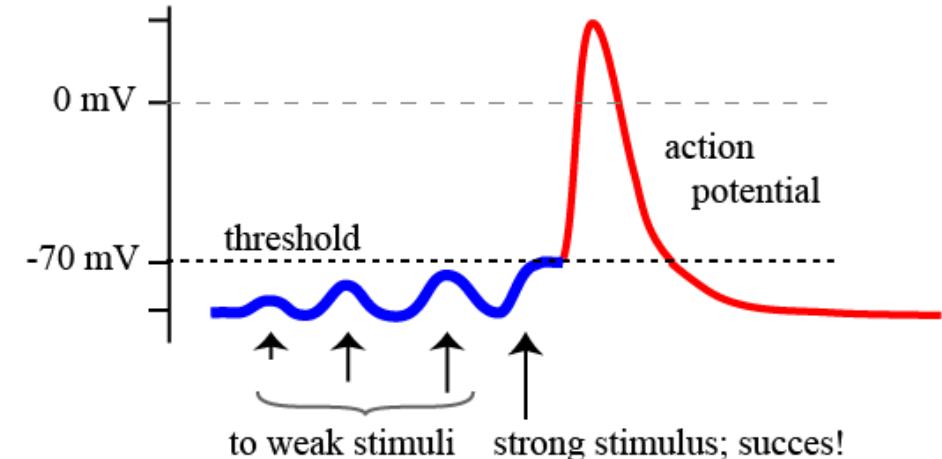
Incoming impulses (via dendrites)  
change the electric potential of the  
neuron



# Biological Neurons

How does a neuron decide to send an impulse (“fire”)?

Incoming impulses (via dendrites)  
change the electric potential of the  
neuron

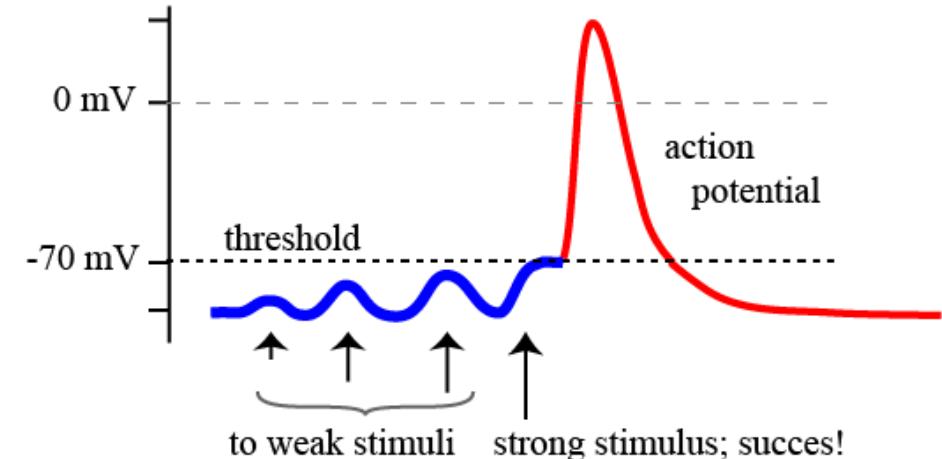


In a parallel circuit, we can sum voltages together

# Biological Neurons

How does a neuron decide to send an impulse (“fire”)?

Incoming impulses (via dendrites)  
change the electric potential of the  
neuron

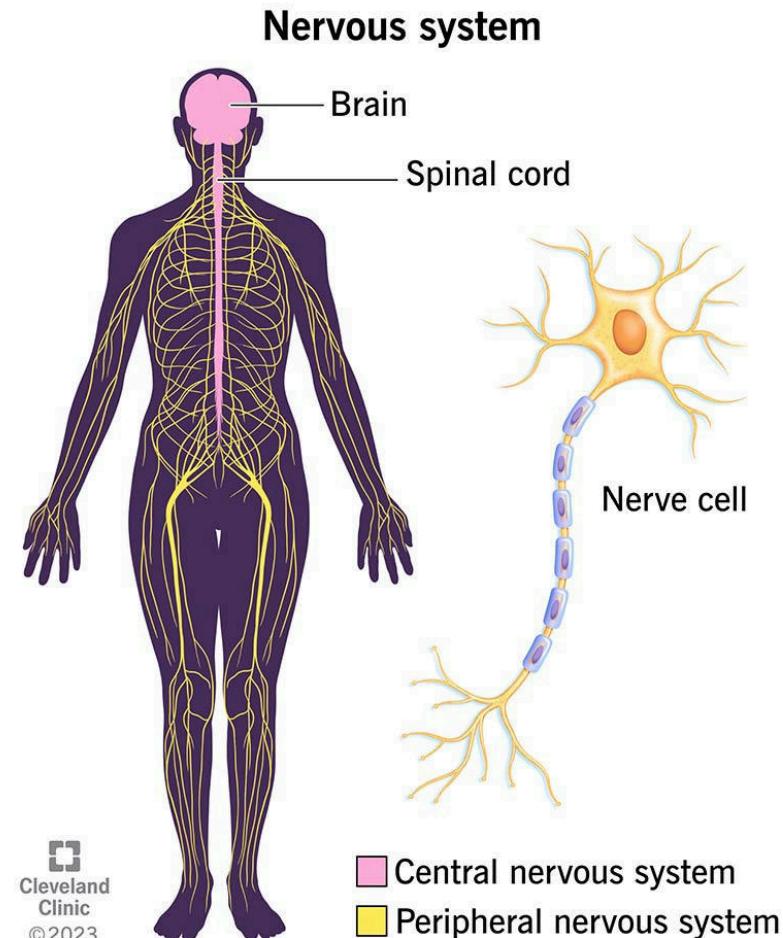


In a parallel circuit, we can sum voltages together

Many active dendrites will add together and trigger an impulse

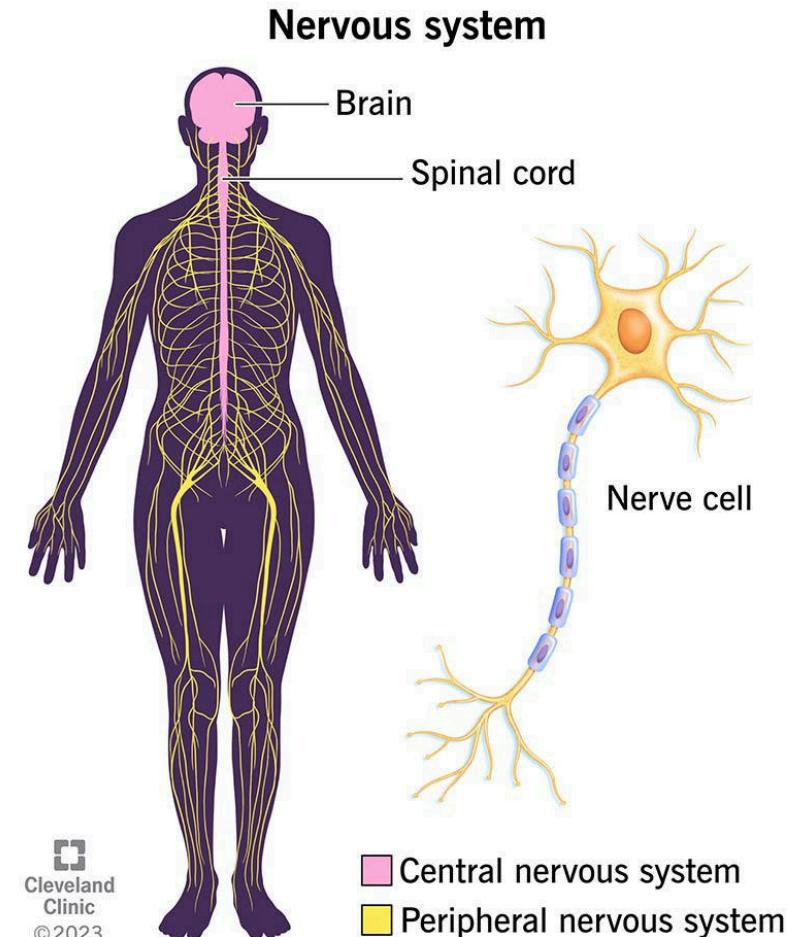
# Biological Neurons

Pain triggers initial nerve impulse,  
starts a chain reaction into the  
brain



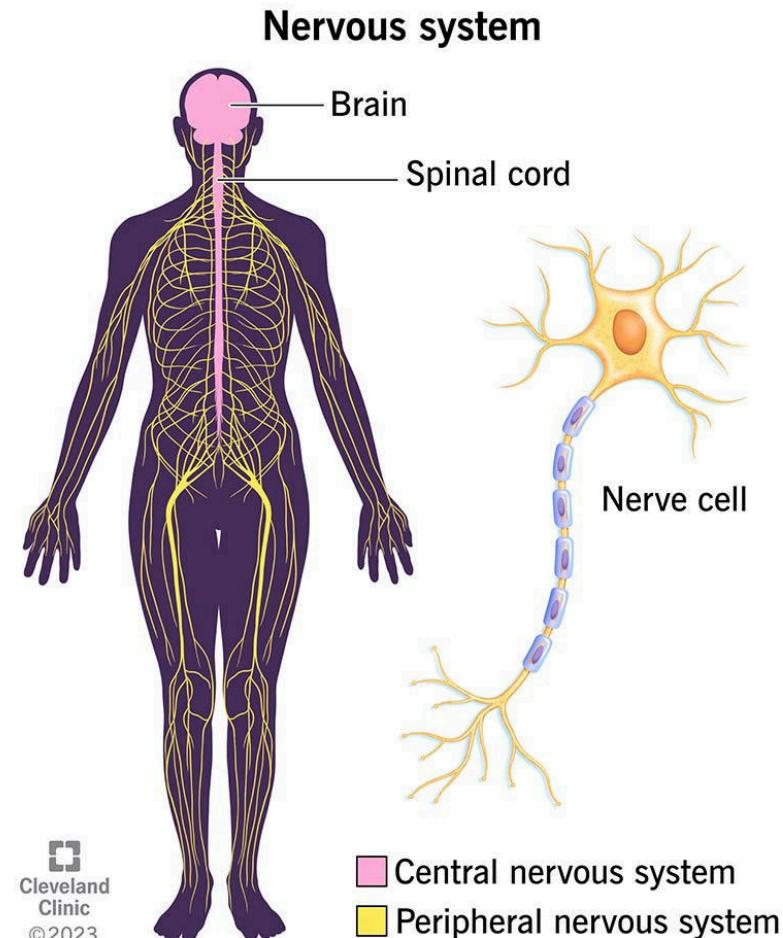
# Biological Neurons

When the signal reaches the brain,  
we will think



# Biological Neurons

After thinking, we will take action



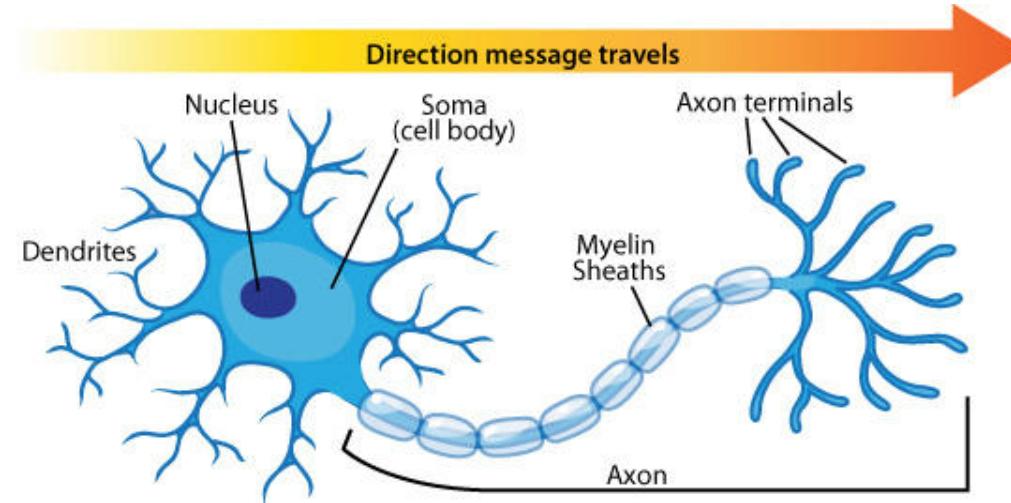
# Agenda

1. Review
2. Multivariate linear regression
3. Limitations of linear regression
4. History of neural networks
5. **Biological neurons**
6. Artificial neurons
7. Wide neural networks
8. Deep neural networks
9. Practical considerations

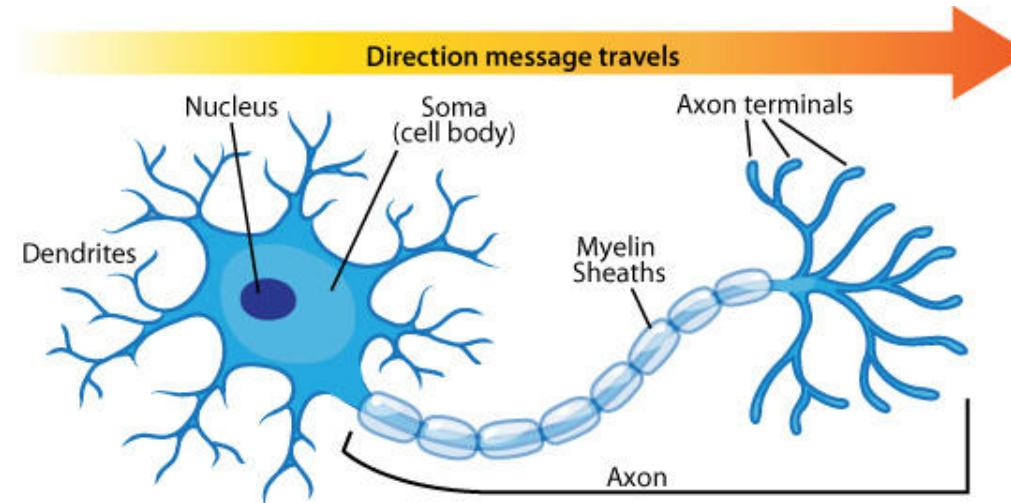
# Agenda

1. Review
2. Multivariate linear regression
3. Limitations of linear regression
4. History of neural networks
5. Biological neurons
6. **Artificial neurons**
7. Wide neural networks
8. Deep neural networks
9. Practical considerations

# Artificial Neurons



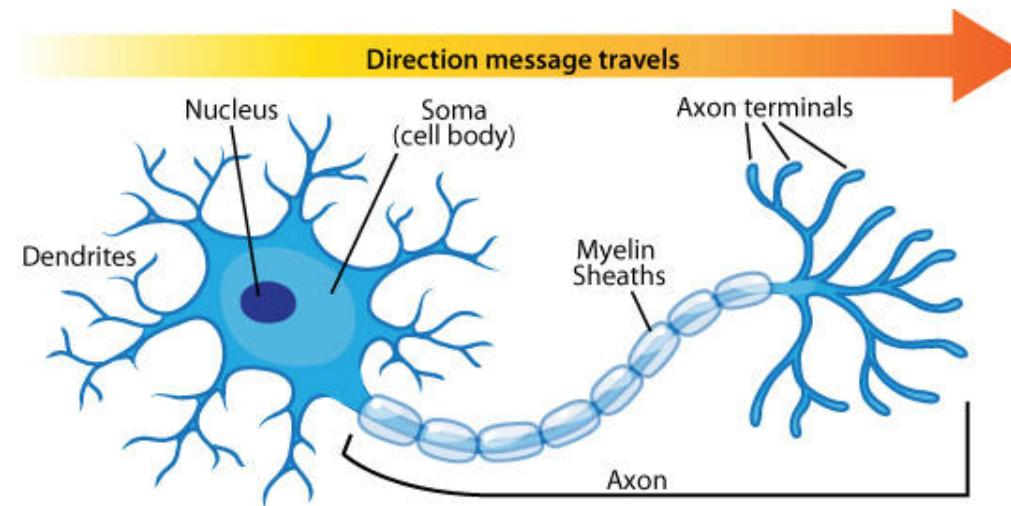
# Artificial Neurons



**Question:** How could we write a neuron as a function?

$$f : \underline{\quad} \mapsto \underline{\quad}$$

# Artificial Neurons



**Question:** How could we write a neuron as a function?

$$f : \underline{\quad} \mapsto \underline{\quad}$$

**Answer:**

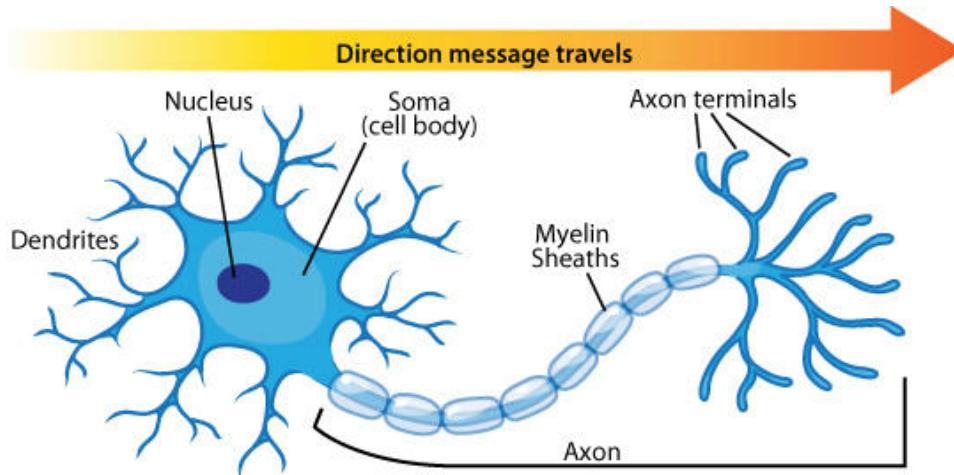
$$f : \underbrace{\mathbb{R}^{d_x}}_{\text{Dendrite voltages}} \times \underbrace{\mathbb{R}^{d_x}}_{\text{Synaptic weight}} \mapsto \underbrace{\mathbb{R}}_{\text{Axon voltage}}$$

# Artificial Neurons

Let us implement an artifical neuron as a function

# Artificial Neurons

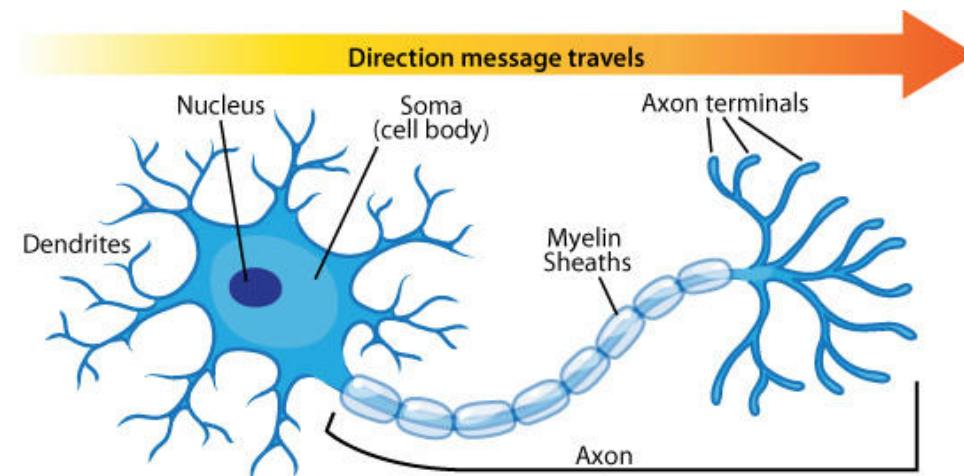
Let us implement an artifical neuron as a function



Neuron has a structure of dendrites with synaptic weights

# Artificial Neurons

Let us implement an artifical neuron as a function



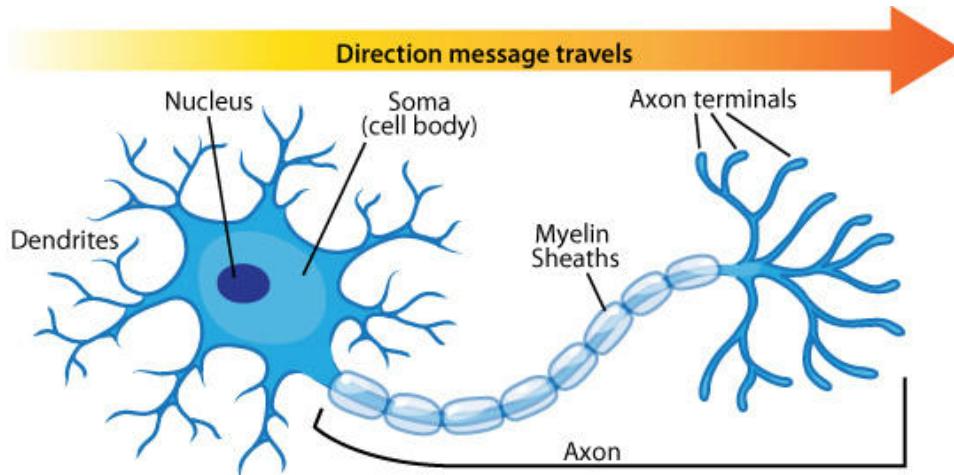
Neuron has a structure of dendrites with synaptic weights

$$f \left( \begin{bmatrix} \theta_1 \\ \theta_2 \\ \vdots \\ \theta_{d_x} \end{bmatrix} \right)$$

$$f(\boldsymbol{\theta})$$

# Artificial Neurons

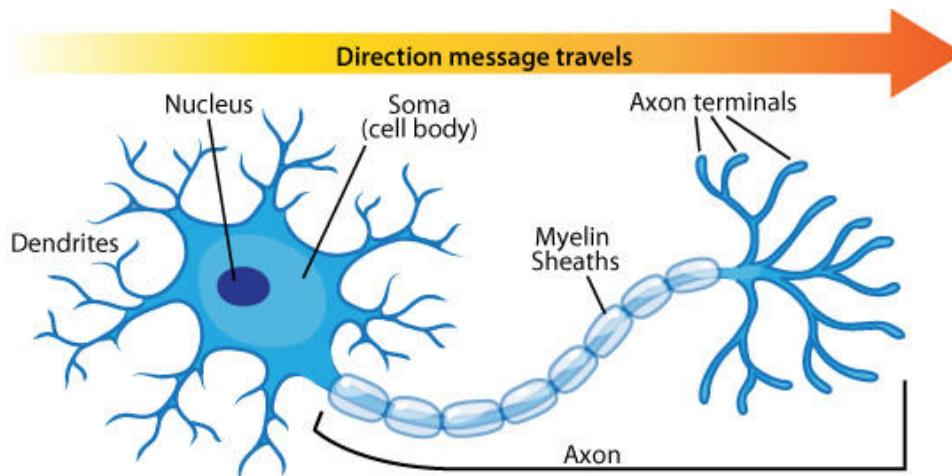
Let us implement an artifical neuron as a function



Each incoming dendrite has some voltage potential

# Artificial Neurons

Let us implement an artifical neuron as a function

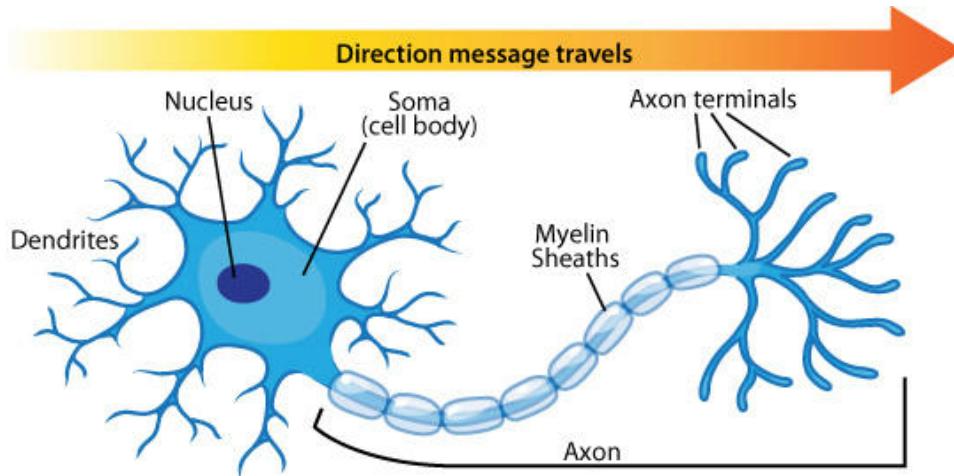


Each incoming dendrite has some voltage potential

$$f \left( \begin{bmatrix} x_1 \\ \vdots \\ x_{d_x} \end{bmatrix}, \begin{bmatrix} \theta_1 \\ \vdots \\ \theta_{d_x} \end{bmatrix} \right)$$
$$f(\mathbf{x}, \boldsymbol{\theta})$$

# Artificial Neurons

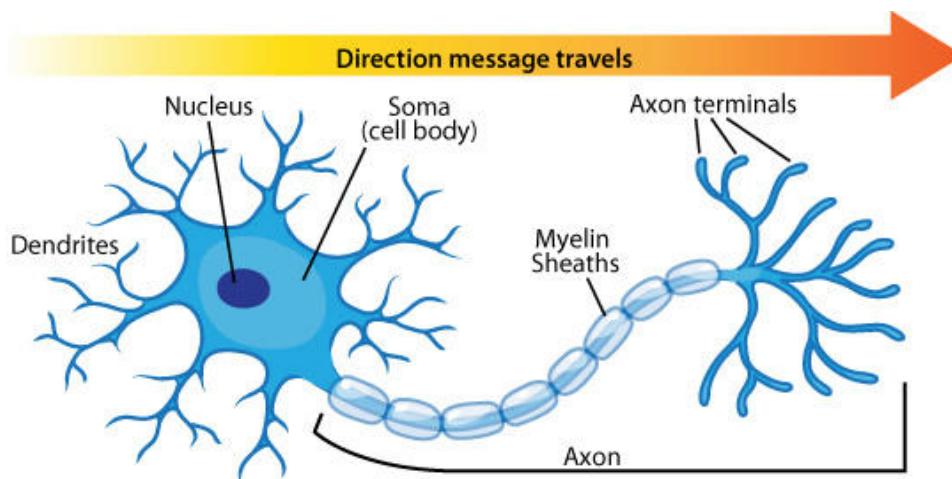
Let us implement an artificial neuron as a function



Voltage potentials sum together to give us the voltage in the cell body

# Artificial Neurons

Let us implement an artificial neuron as a function



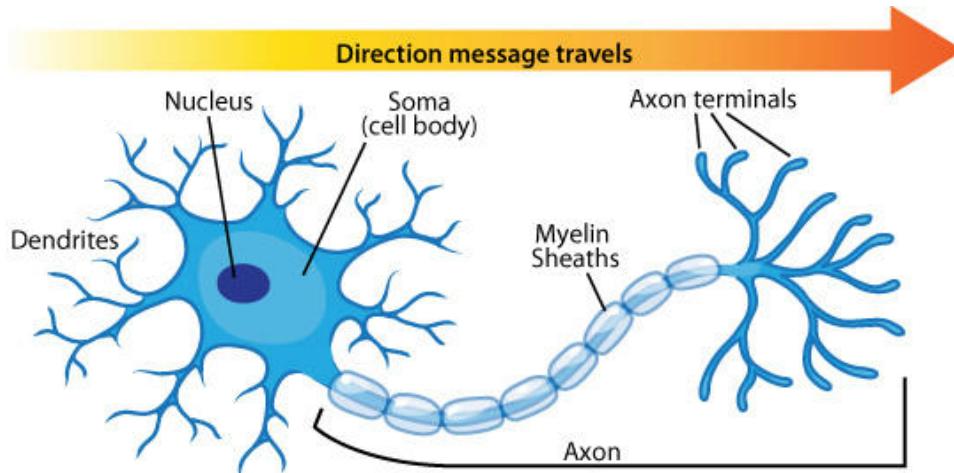
Voltage potentials sum together to give us the voltage in the cell body

$$f \left( \begin{bmatrix} x_1 \\ \vdots \\ x_{d_x} \end{bmatrix}, \begin{bmatrix} \theta_1 \\ \vdots \\ \theta_{d_x} \end{bmatrix} \right) = \sum_{j=1}^{d_x} \theta_j x_j$$

$$f(\mathbf{x}, \boldsymbol{\theta}) = \boldsymbol{\theta}^\top \mathbf{x}$$

# Artificial Neurons

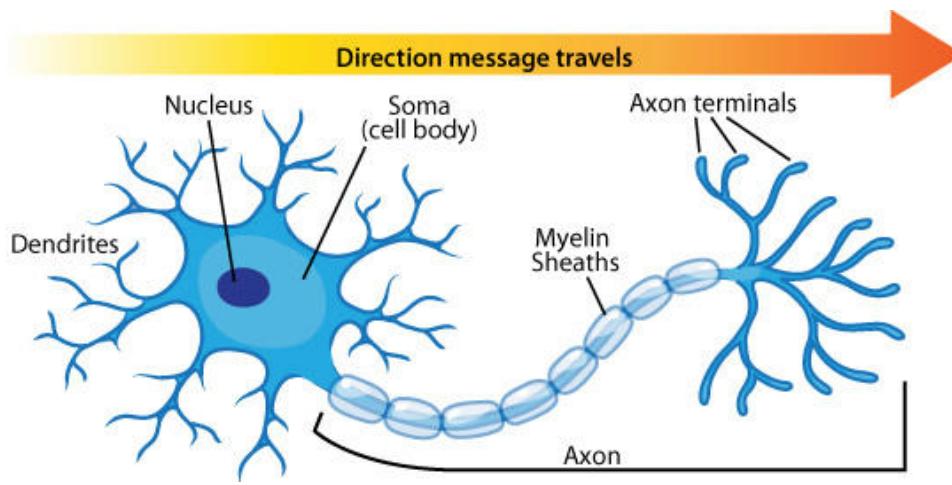
Let us implement an artifical neuron as a function



The axon fires only if the voltage  
is over a threshold

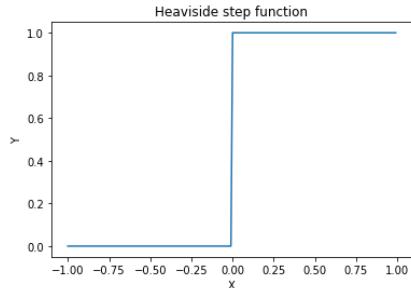
# Artificial Neurons

Let us implement an artifical neuron as a function



The axon fires only if the voltage  
is over a threshold

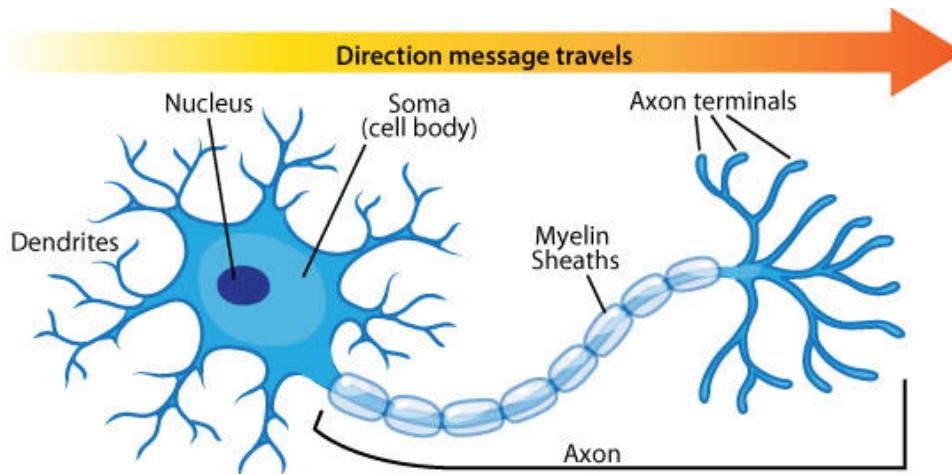
$$\sigma(x) = H(x) =$$



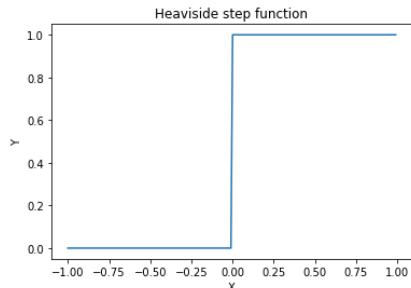
# Artificial Neurons

Let us implement an artifical neuron as a function

The axon fires only if the voltage  
is over a threshold



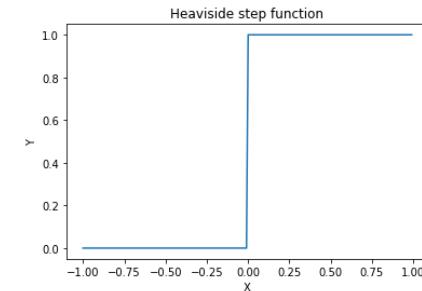
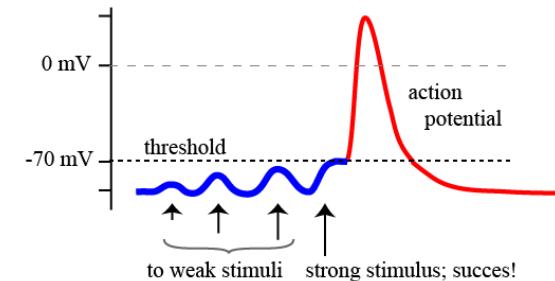
$$\sigma(x) = H(x) =$$



$$f\left(\begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix}, \begin{bmatrix} \theta_1 \\ \vdots \\ \theta_n \end{bmatrix}\right) = \sigma\left(\sum_{j=1}^{d_x} \theta_j x_j\right)$$

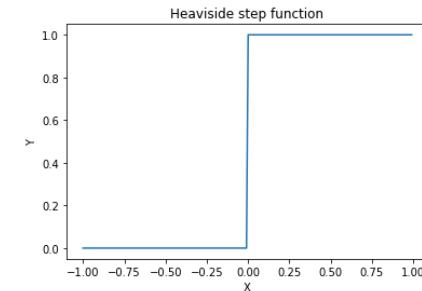
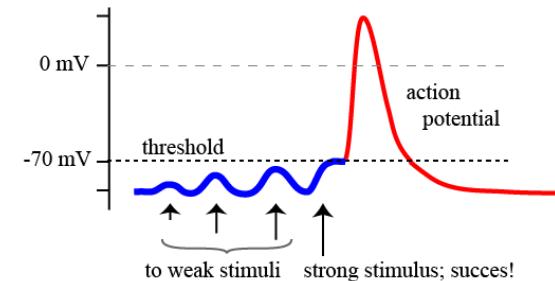
# Artificial Neurons

Maybe we want to vary the activation threshold



# Artificial Neurons

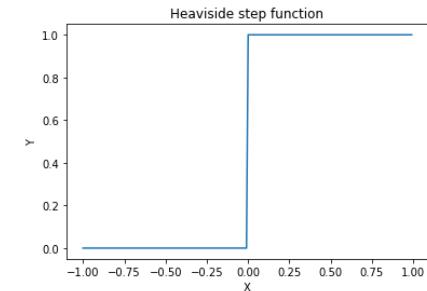
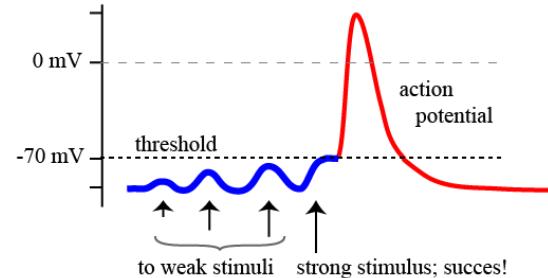
Maybe we want to vary the activation threshold



$$f \left( \begin{bmatrix} 1 \\ x_1 \\ \vdots \\ x_{d_x} \end{bmatrix}, \begin{bmatrix} \theta_0 \\ \theta_1 \\ \vdots \\ \theta_{d_x} \end{bmatrix} \right) = \sigma \left( \theta_0 + \sum_{j=1}^{d_x} \theta_j x_j \right) = \sigma \left( \sum_{j=0}^{d_x} \theta_j x_j \right)$$

# Artificial Neurons

Maybe we want to vary the activation threshold



$$f\left(\begin{bmatrix} 1 \\ x_1 \\ \vdots \\ x_{d_x} \end{bmatrix}, \begin{bmatrix} \theta_0 \\ \theta_1 \\ \vdots \\ \theta_{d_x} \end{bmatrix}\right) = \sigma\left(\theta_0 + \sum_{j=1}^{d_x} \theta_j x_j\right) = \sigma\left(\sum_{j=0}^{d_x} \theta_j x_j\right)$$

$$\bar{x} = \begin{bmatrix} 1 \\ x \end{bmatrix}, \quad f(x, \theta) = \sigma(\theta^\top \bar{x})$$

# Artificial Neurons

$$f(\boldsymbol{x}, \boldsymbol{\theta}) = \sigma(\boldsymbol{\theta}^\top \overline{\boldsymbol{x}})$$

# Artificial Neurons

$$f(\boldsymbol{x}, \boldsymbol{\theta}) = \sigma(\boldsymbol{\theta}^\top \overline{\boldsymbol{x}})$$

This is the artificial neuron!

# Artificial Neurons

$$f(\mathbf{x}, \boldsymbol{\theta}) = \sigma(\boldsymbol{\theta}^\top \mathbf{x})$$

This is the artificial neuron!

Let us write out the full equation for a neuron

# Artificial Neurons

$$f(\mathbf{x}, \boldsymbol{\theta}) = \sigma(\boldsymbol{\theta}^\top \overline{\mathbf{x}})$$

This is the artificial neuron!

Let us write out the full equation for a neuron

$$f(\mathbf{x}, \boldsymbol{\theta}) = \sigma\left(\theta_0 1 + \theta_1 x_1 + \dots + \theta_{d_x} x_{d_x}\right)$$

# Artificial Neurons

$$f(\mathbf{x}, \boldsymbol{\theta}) = \sigma(\boldsymbol{\theta}^\top \overline{\mathbf{x}})$$

This is the artificial neuron!

Let us write out the full equation for a neuron

$$f(\mathbf{x}, \boldsymbol{\theta}) = \sigma\left(\theta_0 1 + \theta_1 x_1 + \dots + \theta_{d_x} x_{d_x}\right)$$

**Question:** Does this look familiar to anyone?

# Artificial Neurons

$$f(\mathbf{x}, \boldsymbol{\theta}) = \sigma(\boldsymbol{\theta}^\top \overline{\mathbf{x}})$$

This is the artificial neuron!

Let us write out the full equation for a neuron

$$f(\mathbf{x}, \boldsymbol{\theta}) = \sigma\left(\theta_0 1 + \theta_1 x_1 + \dots + \theta_{d_x} x_{d_x}\right)$$

**Question:** Does this look familiar to anyone?

**Answer:** Inside  $\sigma$  is the multivariate linear model!

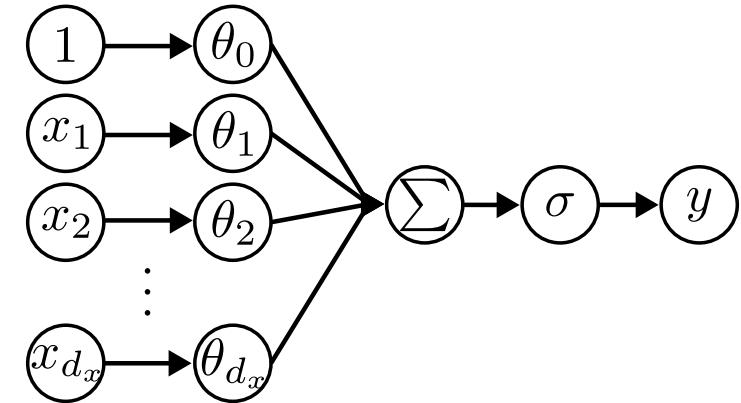
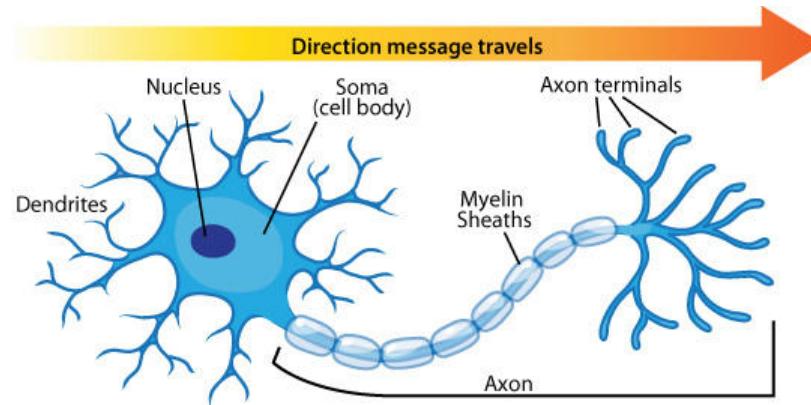
$$f(\mathbf{x}, \boldsymbol{\theta}) = \theta_{d_x} x_{d_x} + \theta_{d_x-1} x_{d_x-1} + \dots + \theta_0 1$$

# Artificial Neurons

We model a neuron using a linear model and activation function

# Artificial Neurons

We model a neuron using a linear model and activation function



$$f(\mathbf{x}, \boldsymbol{\theta}) = \sigma(\boldsymbol{\theta}^\top \mathbf{x})$$

# Artificial Neurons

$$f(\boldsymbol{x}, \boldsymbol{\theta}) = \sigma(\boldsymbol{\theta}^\top \overline{\boldsymbol{x}})$$

# Artificial Neurons

$$f(\mathbf{x}, \boldsymbol{\theta}) = \sigma(\boldsymbol{\theta}^\top \mathbf{\bar{x}})$$

Sometimes, we will write  $\boldsymbol{\theta}$  as a bias and weight  $b, \mathbf{w}$

# Artificial Neurons

$$f(\mathbf{x}, \boldsymbol{\theta}) = \sigma(\boldsymbol{\theta}^\top \overline{\mathbf{x}})$$

Sometimes, we will write  $\boldsymbol{\theta}$  as a bias and weight  $b, \mathbf{w}$

$$\boldsymbol{\theta} = \begin{bmatrix} b \\ \mathbf{w} \end{bmatrix}; \quad \begin{bmatrix} \theta_0 \\ \theta_1 \\ \vdots \\ \theta_{d_x} \end{bmatrix} = \begin{bmatrix} b \\ w_1 \\ \vdots \\ w_{d_x} \end{bmatrix}$$

# Artificial Neurons

$$f(\mathbf{x}, \boldsymbol{\theta}) = \sigma(\boldsymbol{\theta}^\top \overline{\mathbf{x}})$$

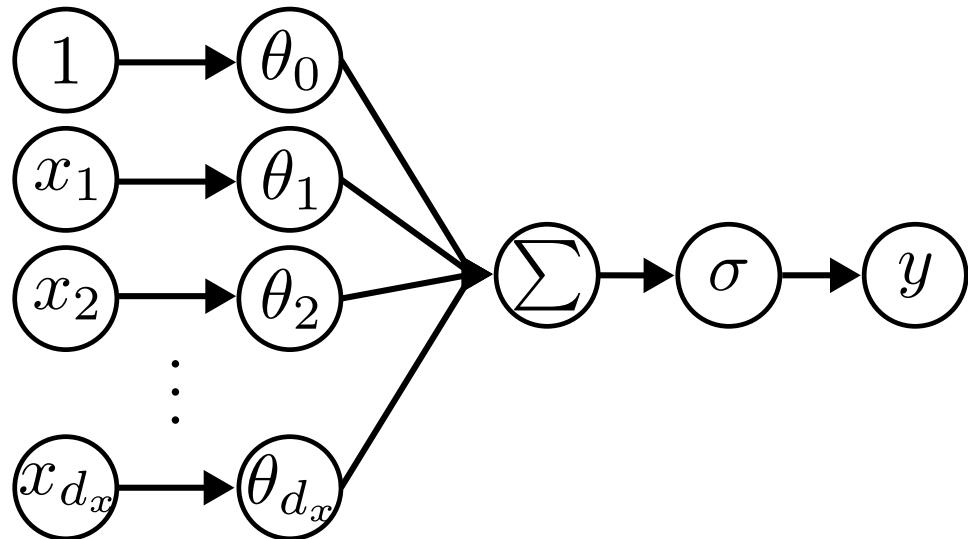
Sometimes, we will write  $\boldsymbol{\theta}$  as a bias and weight  $b, \mathbf{w}$

$$\boldsymbol{\theta} = \begin{bmatrix} b \\ \mathbf{w} \end{bmatrix}; \quad \begin{bmatrix} \theta_0 \\ \theta_1 \\ \vdots \\ \theta_{d_x} \end{bmatrix} = \begin{bmatrix} b \\ w_1 \\ \vdots \\ w_{d_x} \end{bmatrix}$$

$$f\left(\mathbf{x}, \begin{bmatrix} b \\ \mathbf{w} \end{bmatrix}\right) = b + \mathbf{w}^\top \mathbf{x}$$

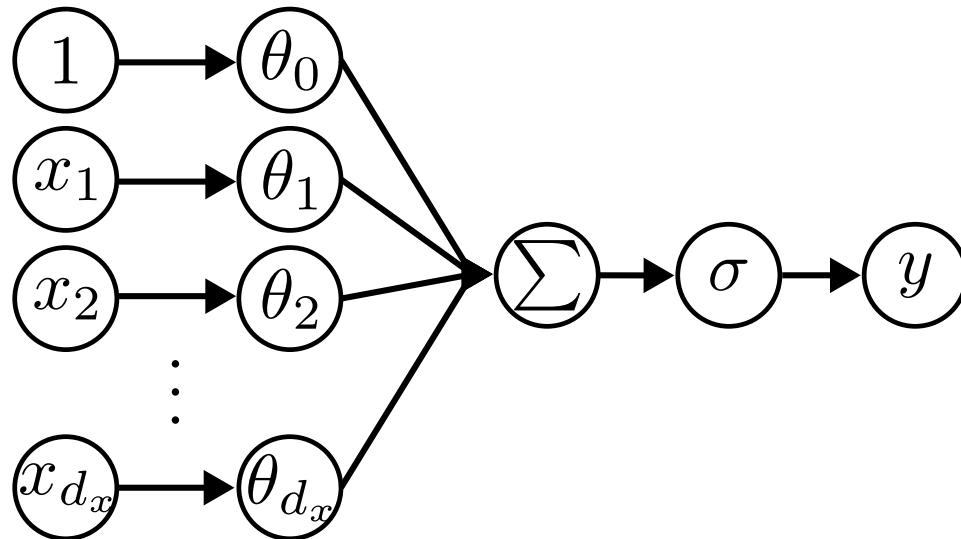
Relax

# Artificial Neurons

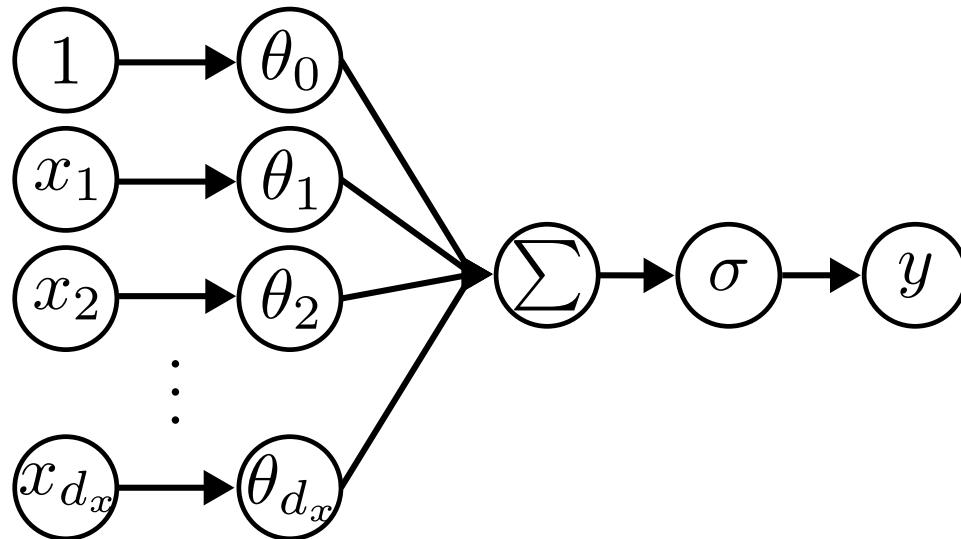


# Artificial Neurons

In machine learning, we represent functions



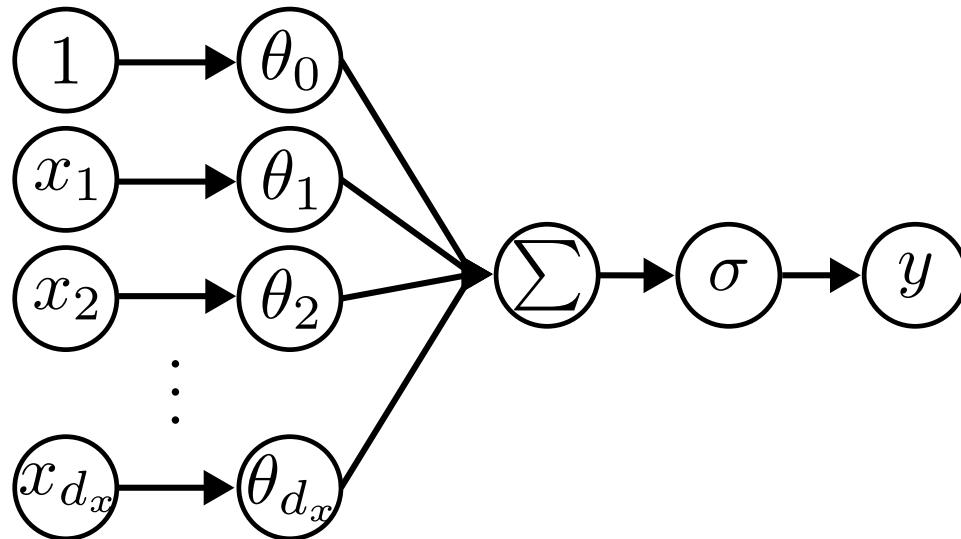
# Artificial Neurons



In machine learning, we represent functions

What kinds of functions can our neuron represent?

# Artificial Neurons

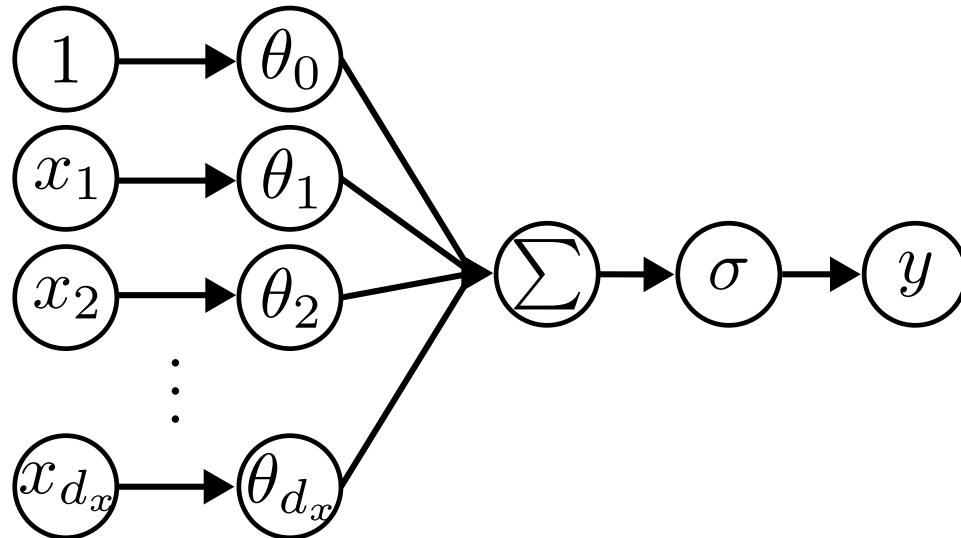


In machine learning, we represent functions

What kinds of functions can our neuron represent?

Let us consider some **boolean** functions

# Artificial Neurons



In machine learning, we represent functions

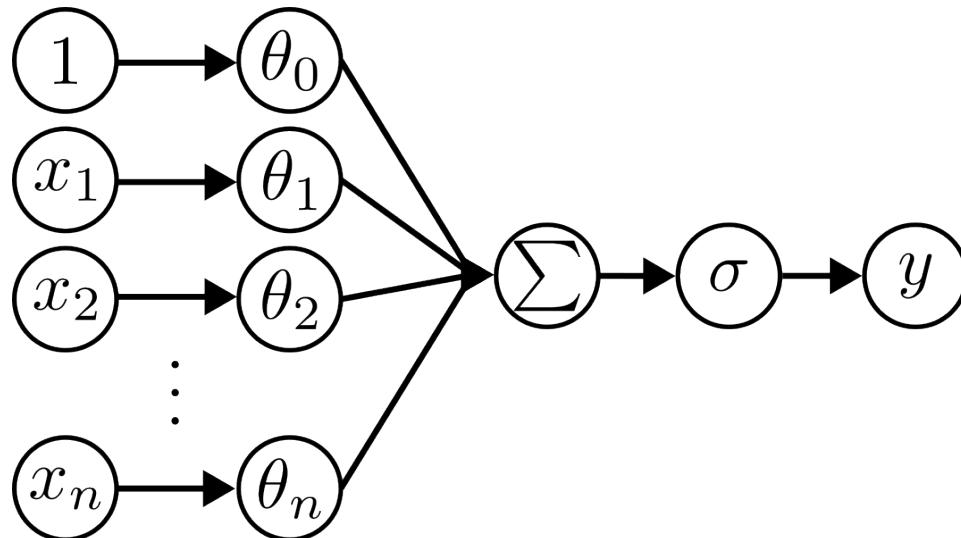
What kinds of functions can our neuron represent?

Let us consider some **boolean** functions

Let us start with a logical AND function

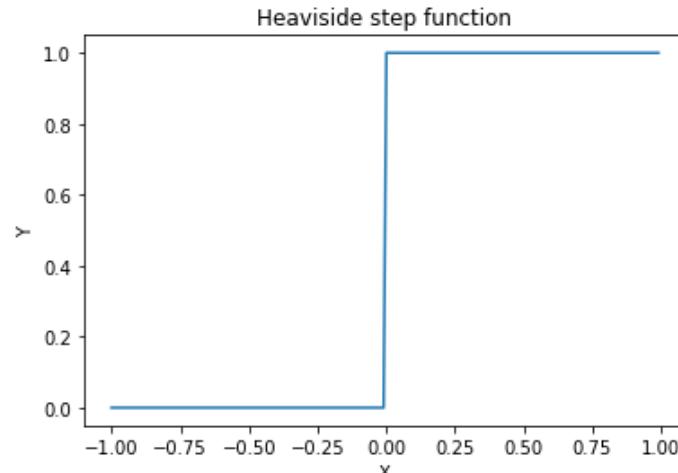
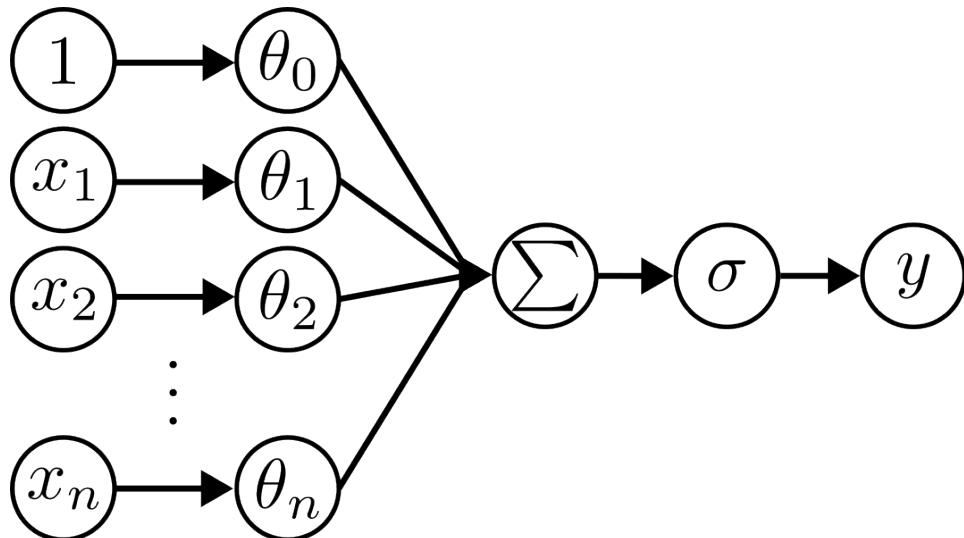
# Artificial Neurons

**Review:** Activation function  
(Heaviside step function)



# Artificial Neurons

**Review:** Activation function  
(Heaviside step function)



$$\sigma(x) = H(x) = \begin{cases} 1 & \text{if } x > 0 \\ 0 & \text{if } x \leq 0 \end{cases}$$

# Artificial Neurons

Implement AND using an artificial neuron


# Artificial Neurons

Implement AND using an artificial neuron

$$f\left(\begin{bmatrix}x_1 & x_2\end{bmatrix}^\top, \begin{bmatrix}\theta_0 & \theta_1 & \theta_2\end{bmatrix}^\top\right) = \sigma(\theta_0 1 + \theta_1 x_1 + \theta_2 x_2)$$


# Artificial Neurons

Implement AND using an artificial neuron

$$f\left(\begin{bmatrix}x_1 & x_2\end{bmatrix}^\top, \begin{bmatrix}\theta_0 & \theta_1 & \theta_2\end{bmatrix}^\top\right) = \sigma(\theta_0 1 + \theta_1 x_1 + \theta_2 x_2)$$

$$\boldsymbol{\theta} = \begin{bmatrix}\theta_0 & \theta_1 & \theta_2\end{bmatrix}^\top = \begin{bmatrix}-1 & 1 & 1\end{bmatrix}^\top$$


# Artificial Neurons

Implement AND using an artificial neuron

$$f([x_1 \ x_2]^\top, [\theta_0 \ \theta_1 \ \theta_2]^\top) = \sigma(\theta_0 1 + \theta_1 x_1 + \theta_2 x_2)$$

$$\boldsymbol{\theta} = [\theta_0 \ \theta_1 \ \theta_2]^\top = [-1 \ 1 \ 1]^\top$$

$x_1$	$x_2$	$y$	$f(x_1, x_2, \boldsymbol{\theta})$	$\hat{y}$
0	0	0	$\sigma(-1 \cdot 1 + 1 \cdot 0 + 1 \cdot 0) = \sigma(-1)$	0
0	1	0	$\sigma(-1 \cdot 1 + 1 \cdot 0 + 1 \cdot 1) = \sigma(0)$	0
1	0	0	$\sigma(-1 \cdot 1 + 1 \cdot 1 + 1 \cdot 0) = \sigma(0)$	0
1	1	1	$\sigma(-1 \cdot 1 + 1 \cdot 1 + 1 \cdot 1) = \sigma(1)$	1

# Artificial Neurons

Implement OR using an artificial neuron


# Artificial Neurons

Implement OR using an artificial neuron

$$f\left(\begin{bmatrix}x_1 & x_2\end{bmatrix}^\top, \begin{bmatrix}\theta_0 & \theta_1 & \theta_2\end{bmatrix}^\top\right) = \sigma(\theta_0 1 + \theta_1 x_1 + \theta_2 x_2)$$


# Artificial Neurons

Implement OR using an artificial neuron

$$f\left(\begin{bmatrix}x_1 & x_2\end{bmatrix}^\top, \begin{bmatrix}\theta_0 & \theta_1 & \theta_2\end{bmatrix}^\top\right) = \sigma(\theta_0 1 + \theta_1 x_1 + \theta_2 x_2)$$

$$\theta = \begin{bmatrix}\theta_0 & \theta_1 & \theta_2\end{bmatrix}^\top = \begin{bmatrix}0 & 1 & 1\end{bmatrix}^\top$$


# Artificial Neurons

Implement OR using an artificial neuron

$$f([x_1 \ x_2]^\top, [\theta_0 \ \theta_1 \ \theta_2]^\top) = \sigma(\theta_0 1 + \theta_1 x_1 + \theta_2 x_2)$$

$$\theta = [\theta_0 \ \theta_1 \ \theta_2]^\top = [0 \ 1 \ 1]^\top$$

$x_1$	$x_2$	$y$	$f(x_1, x_2, \theta)$	$\hat{y}$
0	0	0	$\sigma(1 \cdot 0 + 1 \cdot 0 + 1 \cdot 0) = \sigma(0)$	0
0	1	0	$\sigma(1 \cdot 0 + 1 \cdot 1 + 1 \cdot 0) = \sigma(1)$	1
1	0	1	$\sigma(1 \cdot 0 + 1 \cdot 0 + 1 \cdot 1) = \sigma(1)$	1
1	1	1	$\sigma(1 \cdot 0 + 1 \cdot 1 + 1 \cdot 1) = \sigma(2)$	1

# Artificial Neurons

Implement XOR using an artificial neuron


# Artificial Neurons

Implement XOR using an artificial neuron

$$f\left(\begin{bmatrix}x_1 & x_2\end{bmatrix}^\top, \begin{bmatrix}\theta_0 & \theta_1 & \theta_2\end{bmatrix}^\top\right) = \sigma(\theta_0 1 + \theta_1 x_2 + \theta_2 x_2)$$


# Artificial Neurons

Implement XOR using an artificial neuron

$$f\left(\begin{bmatrix}x_1 & x_2\end{bmatrix}^\top, \begin{bmatrix}\theta_0 & \theta_1 & \theta_2\end{bmatrix}^\top\right) = \sigma(\theta_0 1 + \theta_1 x_2 + \theta_2 x_2)$$

$$\theta = \begin{bmatrix}\theta_0 & \theta_1 & \theta_2\end{bmatrix}^\top = \begin{bmatrix}? & ? & ?\end{bmatrix}^\top$$


# Artificial Neurons

Implement XOR using an artificial neuron

$$f([x_1 \ x_2]^\top, [\theta_0 \ \theta_1 \ \theta_2]^\top) = \sigma(\theta_0 1 + \theta_1 x_2 + \theta_2 x_2)$$

$$\theta = [\theta_0 \ \theta_1 \ \theta_2]^\top = [? \ ? \ ?]^\top$$

$x_1$	$x_2$	$y$	$f(x_1, x_2, \theta)$	$\hat{y}$
0	0	0	This is IMPOSSIBLE!	
0	1	1		
1	0	1		
1	1	0		

# Artificial Neurons

Why can't we represent XOR using a neuron?

# Artificial Neurons

Why can't we represent XOR using a neuron?

$$f\left(\begin{bmatrix}x_1 & x_2\end{bmatrix}^\top, \begin{bmatrix}\theta_0 & \theta_1 & \theta_2\end{bmatrix}^\top\right) = \sigma(1\theta_0 + x_1\theta_1 + x_2\theta_2)$$

# Artificial Neurons

Why can't we represent XOR using a neuron?

$$f\left(\begin{bmatrix}x_1 & x_2\end{bmatrix}^\top, \begin{bmatrix}\theta_0 & \theta_1 & \theta_2\end{bmatrix}^\top\right) = \sigma(1\theta_0 + x_1\theta_1 + x_2\theta_2)$$

We can only represent  $\sigma$ (linear function)

# Artificial Neurons

Why can't we represent XOR using a neuron?

$$f\left(\begin{bmatrix}x_1 & x_2\end{bmatrix}^\top, \begin{bmatrix}\theta_0 & \theta_1 & \theta_2\end{bmatrix}^\top\right) = \sigma(1\theta_0 + x_1\theta_1 + x_2\theta_2)$$

We can only represent  $\sigma$ (linear function)

XOR is not a linear combination of  $x_1, x_2$ !

# Artificial Neurons

Why can't we represent XOR using a neuron?

$$f\left(\begin{bmatrix}x_1 & x_2\end{bmatrix}^\top, \begin{bmatrix}\theta_0 & \theta_1 & \theta_2\end{bmatrix}^\top\right) = \sigma(1\theta_0 + x_1\theta_1 + x_2\theta_2)$$

We can only represent  $\sigma$ (linear function)

XOR is not a linear combination of  $x_1, x_2$ !

We want to represent any function, not just linear functions

# Artificial Neurons

Why can't we represent XOR using a neuron?

$$f\left(\begin{bmatrix}x_1 & x_2\end{bmatrix}^\top, \begin{bmatrix}\theta_0 & \theta_1 & \theta_2\end{bmatrix}^\top\right) = \sigma(1\theta_0 + x_1\theta_1 + x_2\theta_2)$$

We can only represent  $\sigma$ (linear function)

XOR is not a linear combination of  $x_1, x_2$ !

We want to represent any function, not just linear functions

Let us think back to biology, maybe it has an answer

# Artificial Neurons

**Brain:** Biological neurons → Biological neural network

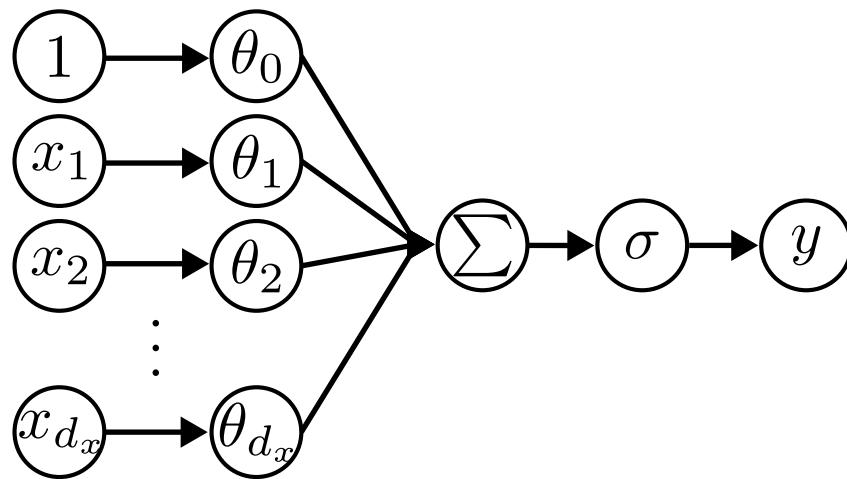
# Artificial Neurons

**Brain:** Biological neurons → Biological neural network

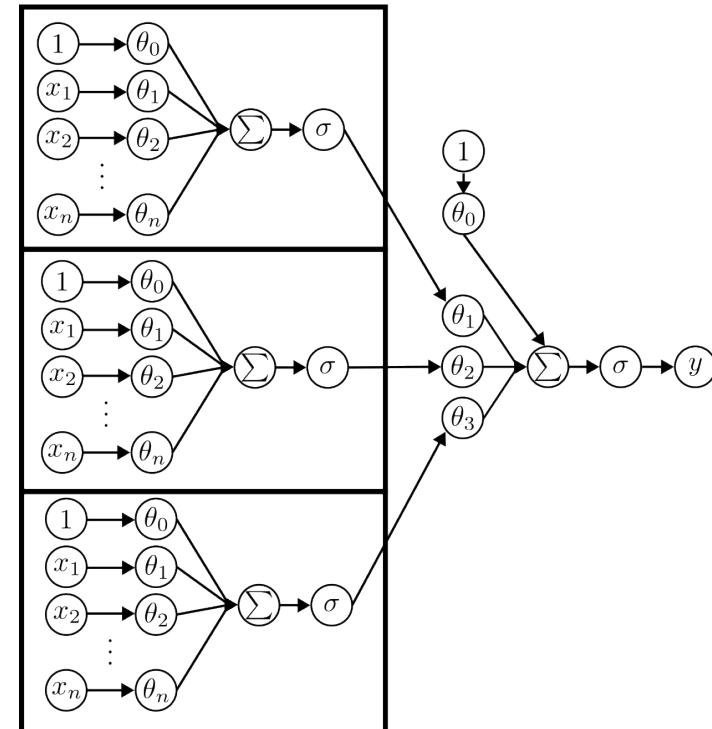
**Computer:** Artificial neurons → Artificial neural network

# Artificial Neurons

Connect artificial neurons into a network

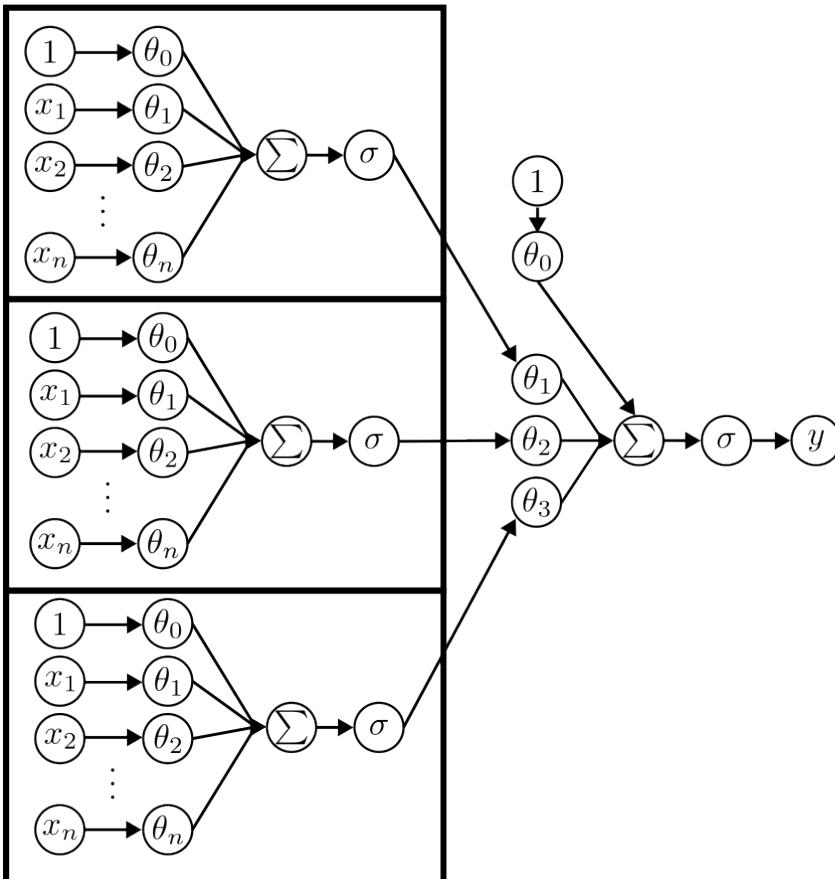


Neuron



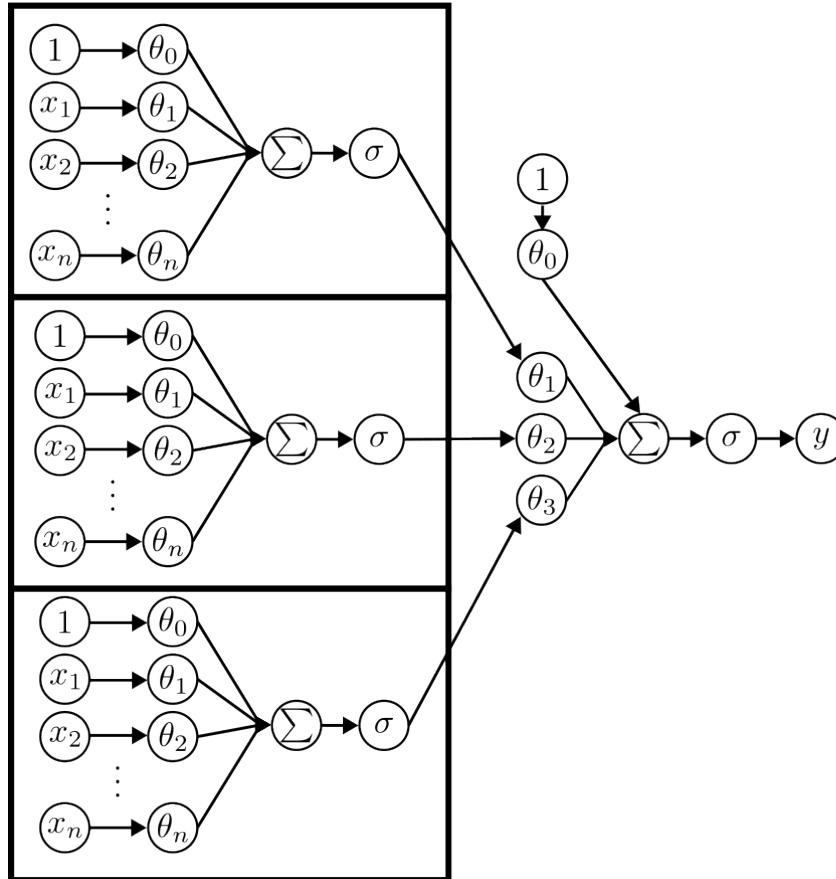
Neural Network

# Artificial Neurons



Adding neurons in **parallel** creates a **wide** neural network

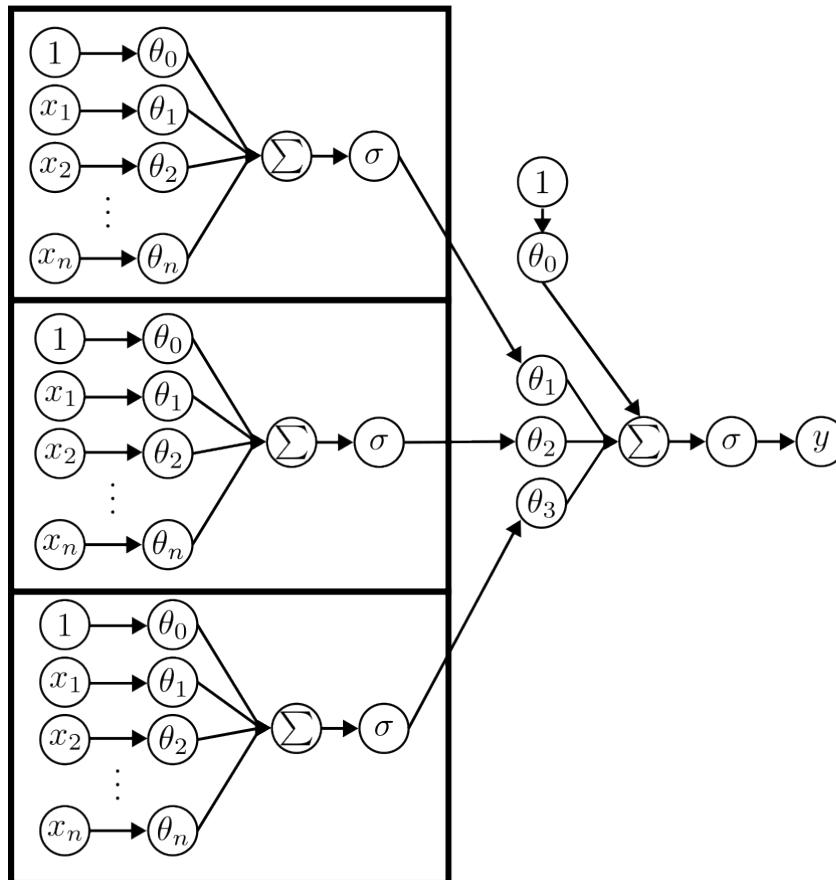
# Artificial Neurons



Adding neurons in **parallel** creates a **wide** neural network

Adding neurons in **series** creates a **deep** neural network

# Artificial Neurons

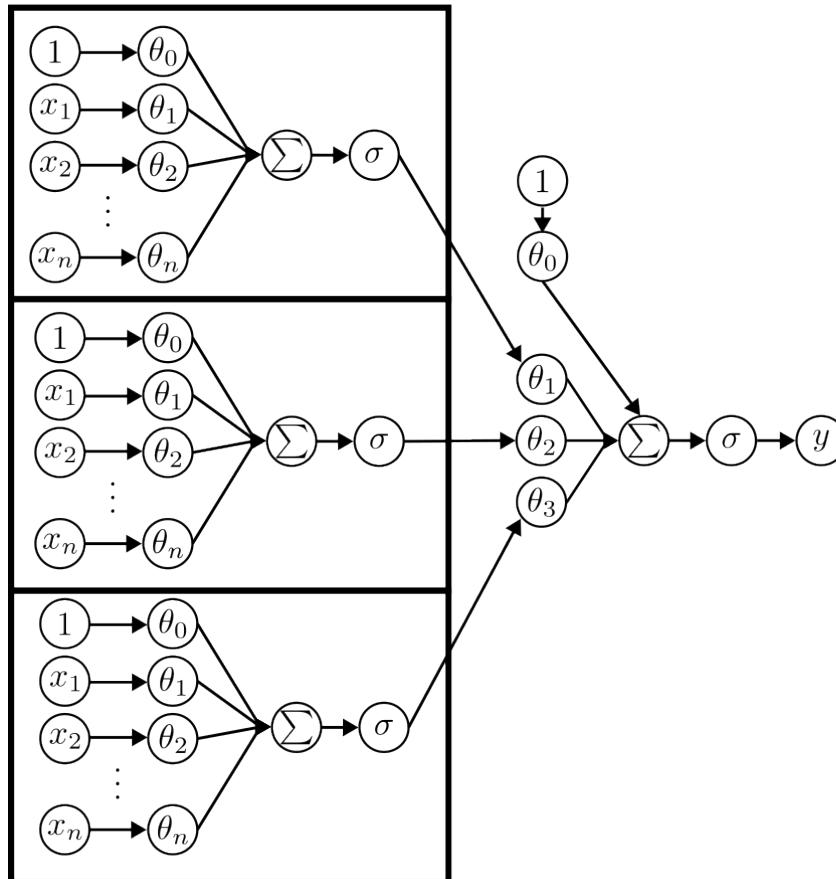


Adding neurons in **parallel** creates a **wide** neural network

Adding neurons in **series** creates a **deep** neural network

Today's powerful neural networks are both **wide** and **deep**

# Artificial Neurons



Adding neurons in **parallel** creates a **wide** neural network

Adding neurons in **series** creates a **deep** neural network

Today's powerful neural networks are both **wide** and **deep**

Let us try to implement XOR using a wide and deep neural network

# Agenda

1. Review
2. Multivariate linear regression
3. Limitations of linear regression
4. History of neural networks
5. Biological neurons
6. **Artificial neurons**
7. Wide neural networks
8. Deep neural networks
9. Practical considerations

# Agenda

1. Review
2. Multivariate linear regression
3. Limitations of linear regression
4. History of neural networks
5. Biological neurons
6. Artificial neurons
7. **Wide neural networks**
8. Deep neural networks
9. Practical considerations

# Wide Neural Networks

How do we express a **wide** neural network mathematically?

# Wide Neural Networks

How do we express a **wide** neural network mathematically?

A single neuron:

$$f : \mathbb{R}^{d_x} \times \Theta \mapsto \mathbb{R}$$

$$\Theta \in \mathbb{R}^{d_x+1}$$

# Wide Neural Networks

How do we express a **wide** neural network mathematically?

A single neuron:

$$f : \mathbb{R}^{d_x} \times \Theta \mapsto \mathbb{R}$$

$$\Theta \in \mathbb{R}^{d_x+1}$$

$d_y$  neurons (wide):

$$f : \mathbb{R}^{d_x} \times \Theta \mapsto \mathbb{R}^{d_y}$$

$$\Theta \in \mathbb{R}^{(d_x+1) \times d_y}$$

# Wide Neural Networks

For a single neuron:

$$f\left(\begin{bmatrix} x_1 \\ \vdots \\ x_{d_x} \end{bmatrix}, \begin{bmatrix} \theta_0 \\ \theta_1 \\ \vdots \\ \theta_{d_x} \end{bmatrix}\right) = \sigma\left(\sum_{i=0}^{d_x} \theta_i \bar{x}_i\right)$$

# Wide Neural Networks

For a single neuron:

$$f\left(\begin{bmatrix} x_1 \\ \vdots \\ x_{d_x} \end{bmatrix}, \begin{bmatrix} \theta_0 \\ \theta_1 \\ \vdots \\ \theta_{d_x} \end{bmatrix}\right) = \sigma\left(\sum_{i=0}^{d_x} \theta_i \bar{x}_i\right)$$

$$f(\mathbf{x}, \boldsymbol{\theta}) = \sigma(b + \mathbf{w}^\top \mathbf{x})$$

For a wide network:

$$f \left( \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_{d_x} \end{bmatrix}, \begin{bmatrix} \theta_{0,1} & \theta_{0,2} & \dots & \theta_{0,d_y} \\ \theta_{1,1} & \theta_{1,2} & \dots & \theta_{1,d_y} \\ \vdots & \vdots & \ddots & \vdots \\ \theta_{d_x,1} & \theta_{d_x,2} & \dots & \theta_{d_x,d_y} \end{bmatrix} \right) = \begin{bmatrix} \sigma\left(\sum_{i=0}^{d_x} \theta_{i,1} \bar{x}_i\right) \\ \sigma\left(\sum_{i=0}^{d_x} \theta_{i,2} \bar{x}_i\right) \\ \vdots \\ \sigma\left(\sum_{i=0}^{d_x} \theta_{i,d_y} \bar{x}_i\right) \end{bmatrix}$$

$$f(\mathbf{x}, \boldsymbol{\theta}) = \sigma(\boldsymbol{\theta}^\top \bar{\mathbf{x}}); \quad \boldsymbol{\theta}^\top \in \mathbb{R}^{d_y \times (d_x + 1)}$$

For a wide network:

$$f \left( \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_{d_x} \end{bmatrix}, \begin{bmatrix} \theta_{0,1} & \theta_{0,2} & \dots & \theta_{0,d_y} \\ \theta_{1,1} & \theta_{1,2} & \dots & \theta_{1,d_y} \\ \vdots & \vdots & \ddots & \vdots \\ \theta_{d_x,1} & \theta_{d_x,2} & \dots & \theta_{d_x,d_y} \end{bmatrix} \right) = \begin{bmatrix} \sigma\left(\sum_{i=0}^{d_x} \theta_{i,1} \bar{x}_i\right) \\ \sigma\left(\sum_{i=0}^{d_x} \theta_{i,2} \bar{x}_i\right) \\ \vdots \\ \sigma\left(\sum_{i=0}^{d_x} \theta_{i,d_y} \bar{x}_i\right) \end{bmatrix}$$

$$f(\mathbf{x}, \boldsymbol{\theta}) = \sigma(\boldsymbol{\theta}^\top \bar{\mathbf{x}}); \quad \boldsymbol{\theta}^\top \in \mathbb{R}^{d_y \times (d_x + 1)}$$

$$f\left(\mathbf{x}, \begin{bmatrix} \mathbf{b} \\ \mathbf{W} \end{bmatrix}\right) = \sigma(\mathbf{b} + \mathbf{W}^\top \mathbf{x}); \quad \mathbf{b} \in \mathbb{R}^{d_y}, \mathbf{W} \in \mathbb{R}^{d_x \times d_y}$$

1. Review
2. Multivariate linear regression
3. Limitations of linear regression
4. History of neural networks
5. Biological neurons
6. Artificial neurons
7. **Wide neural networks**
8. Deep neural networks
9. Practical considerations

1. Review
2. Multivariate linear regression
3. Limitations of linear regression
4. History of neural networks
5. Biological neurons
6. Artificial neurons
7. Wide neural networks
8. **Deep neural networks**
9. Practical considerations

# Deep Neural Networks

How do we express a **deep** neural network mathematically?

# Deep Neural Networks

How do we express a **deep** neural network mathematically?

A wide network and deep network have a similar function signature:

$$f : \mathbb{R}^{d_x} \times \Theta \mapsto \mathbb{R}^{d_y}$$

# Deep Neural Networks

How do we express a **deep** neural network mathematically?

A wide network and deep network have a similar function signature:

$$f : \mathbb{R}^{d_x} \times \Theta \mapsto \mathbb{R}^{d_y}$$

But the parameters change!

Wide:  $\Theta \in \mathbb{R}^{(d_x+1) \times d_y}$

# Deep Neural Networks

How do we express a **deep** neural network mathematically?

A wide network and deep network have a similar function signature:

$$f : \mathbb{R}^{d_x} \times \Theta \mapsto \mathbb{R}^{d_y}$$

But the parameters change!

Wide:  $\Theta \in \mathbb{R}^{(d_x+1) \times d_y}$

Deep:  $\Theta \in \mathbb{R}^{(d_x+1) \times d_h} \times \mathbb{R}^{(d_h+1) \times d_h} \times \dots \times \mathbb{R}^{(d_h+1) \times d_y}$

# Deep Neural Networks

How do we express a **deep** neural network mathematically?

A wide network and deep network have a similar function signature:

$$f : \mathbb{R}^{d_x} \times \Theta \mapsto \mathbb{R}^{d_y}$$

But the parameters change!

Wide:  $\Theta \in \mathbb{R}^{(d_x+1) \times d_y}$

Deep:  $\Theta \in \mathbb{R}^{(d_x+1) \times d_h} \times \mathbb{R}^{(d_h+1) \times d_h} \times \dots \times \mathbb{R}^{(d_h+1) \times d_y}$

$$\boldsymbol{\theta} = [\boldsymbol{\theta}_1 \ \boldsymbol{\theta}_2 \ \dots \ \boldsymbol{\theta}_\ell]^\top = [\boldsymbol{\varphi} \ \boldsymbol{\psi} \ \dots \ \boldsymbol{\xi}]^\top$$

# Deep Neural Networks

A wide network:

$$f(\mathbf{x}, \theta) = \theta^\top \overline{\mathbf{x}}$$

# Deep Neural Networks

A wide network:

$$f(\mathbf{x}, \boldsymbol{\theta}) = \boldsymbol{\theta}^\top \overline{\mathbf{x}}$$

A deep network has many internal functions

$$f_1(\mathbf{x}, \boldsymbol{\varphi}) = \boldsymbol{\varphi}^\top \overline{\mathbf{x}} \quad f_2(\mathbf{x}, \boldsymbol{\psi}) = \boldsymbol{\psi}^\top \overline{\mathbf{x}} \quad \dots \quad f_\ell(\mathbf{x}, \boldsymbol{\xi}) = \boldsymbol{\xi}^\top \overline{\mathbf{x}}$$

# Deep Neural Networks

A wide network:

$$f(\mathbf{x}, \boldsymbol{\theta}) = \boldsymbol{\theta}^\top \overline{\mathbf{x}}$$

A deep network has many internal functions

$$f_1(\mathbf{x}, \boldsymbol{\varphi}) = \boldsymbol{\varphi}^\top \overline{\mathbf{x}} \quad f_2(\mathbf{x}, \boldsymbol{\psi}) = \boldsymbol{\psi}^\top \overline{\mathbf{x}} \quad \dots \quad f_\ell(\mathbf{x}, \boldsymbol{\xi}) = \boldsymbol{\xi}^\top \overline{\mathbf{x}}$$

$$f(\mathbf{x}, \boldsymbol{\theta}) = f_\ell(\dots f_2(f_1(\mathbf{x}, \boldsymbol{\varphi}), \boldsymbol{\psi}) \dots \boldsymbol{\xi})$$

# Deep Neural Networks

Written another way

$$z_1 = f_1(\mathbf{x}, \boldsymbol{\varphi}) = \boldsymbol{\varphi}^\top \overline{\mathbf{x}}$$

# Deep Neural Networks

Written another way

$$z_1 = f_1(\mathbf{x}, \boldsymbol{\varphi}) = \boldsymbol{\varphi}^\top \overline{\mathbf{x}}$$

$$z_2 = f_2(z_1, \boldsymbol{\psi}) = \boldsymbol{\psi}^\top \overline{z}_1$$

# Deep Neural Networks

Written another way

$$z_1 = f_1(\mathbf{x}, \boldsymbol{\varphi}) = \boldsymbol{\varphi}^\top \overline{\mathbf{x}}$$

$$z_2 = f_2(z_1, \boldsymbol{\psi}) = \boldsymbol{\psi}^\top \overline{z}_1$$

⋮

# Deep Neural Networks

Written another way

$$z_1 = f_1(x, \varphi) = \varphi^\top \bar{x}$$

$$z_2 = f_2(z_1, \psi) = \psi^\top \bar{z}_1$$

⋮

$$y = f_\ell(x, \xi) = \xi^\top \bar{z}_{\ell-1}$$

We call each function a **layer**

# Deep Neural Networks

Written another way

$$z_1 = f_1(x, \varphi) = \varphi^\top \bar{x}$$

$$z_2 = f_2(z_1, \psi) = \psi^\top \bar{z}_1$$

⋮

$$y = f_\ell(x, \xi) = \xi^\top \bar{z}_{\ell-1}$$

We call each function a **layer**

A deep neural network is made of many layers

# Deep Neural Networks

What functions can we represent using a deep neural network?

# Deep Neural Networks

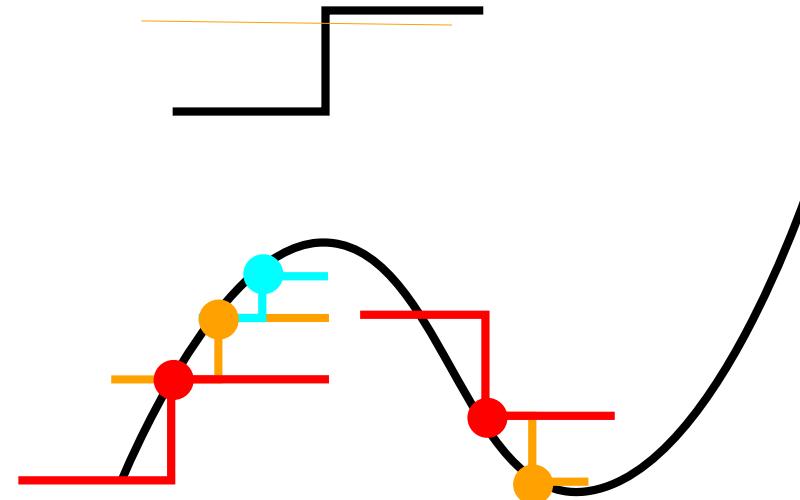
What functions can we represent using a deep neural network?

**Proof Sketch:** Approximate a continuous function  $g : \mathbb{R} \mapsto \mathbb{R}$  using a linear combination of Heaviside functions

# Deep Neural Networks

What functions can we represent using a deep neural network?

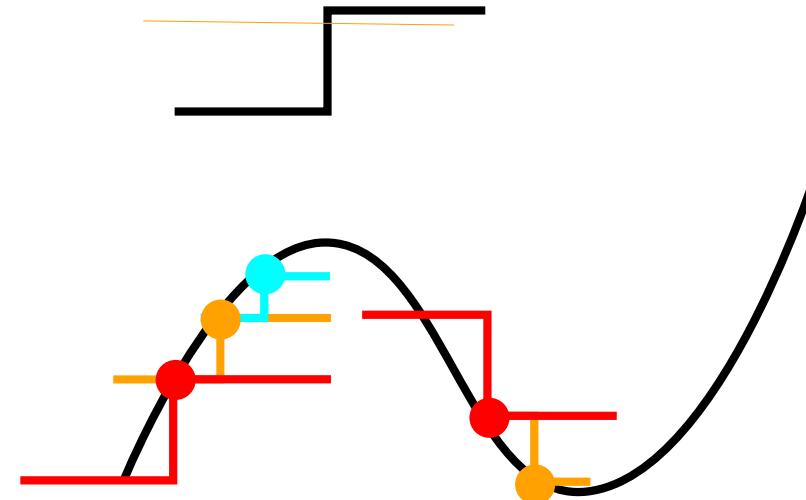
**Proof Sketch:** Approximate a continuous function  $g : \mathbb{R} \mapsto \mathbb{R}$  using a linear combination of Heaviside functions



# Deep Neural Networks

What functions can we represent using a deep neural network?

**Proof Sketch:** Approximate a continuous function  $g : \mathbb{R} \mapsto \mathbb{R}$  using a linear combination of Heaviside functions



$$\exists (\theta \in \mathbb{R}^{1 \times d_h}, \varphi \in \mathbb{R}^{(d_h+1) \times d_1}) \text{ such that } \lim_{d_h \rightarrow \infty} [\varphi^\top \sigma(\overline{\theta^\top \bar{x}})]$$

# Deep Neural Networks

A deep neural network is a **universal function approximator**

# Deep Neural Networks

A deep neural network is a **universal function approximator**

It can approximate **any** continuous function  $g(x)$  to precision  $\varepsilon$

# Deep Neural Networks

A deep neural network is a **universal function approximator**

It can approximate **any** continuous function  $g(x)$  to precision  $\varepsilon$

$$| g(\mathbf{x}) - f(\mathbf{x}, \boldsymbol{\theta}) | < \varepsilon$$

# Deep Neural Networks

A deep neural network is a **universal function approximator**

It can approximate **any** continuous function  $g(x)$  to precision  $\varepsilon$

$$| g(\mathbf{x}) - f(\mathbf{x}, \boldsymbol{\theta}) | < \varepsilon$$

Making the network deeper or wider decreases  $\varepsilon$

# Deep Neural Networks

A deep neural network is a **universal function approximator**

It can approximate **any** continuous function  $g(x)$  to precision  $\varepsilon$

$$| g(x) - f(x, \theta) | < \varepsilon$$

Making the network deeper or wider decreases  $\varepsilon$

Very powerful finding! The basis of deep learning.

# Deep Neural Networks

A deep neural network is a **universal function approximator**

It can approximate **any** continuous function  $g(x)$  to precision  $\varepsilon$

$$| g(x) - f(x, \theta) | < \varepsilon$$

Making the network deeper or wider decreases  $\varepsilon$

Very powerful finding! The basis of deep learning.

**Task:** predict how many ❤ a photo gets on social media



# Agenda

1. Review
2. Multivariate linear regression
3. Limitations of linear regression
4. History of neural networks
5. Biological neurons
6. Artificial neurons
7. Wide neural networks
8. **Deep neural networks**
9. Practical considerations

# Agenda

1. Review
2. Multivariate linear regression
3. Limitations of linear regression
4. History of neural networks
5. Biological neurons
6. Artificial neurons
7. Wide neural networks
8. Deep neural networks
9. **Practical considerations**

# Practical Considerations

We call wide neural networks **perceptrons**

# Practical Considerations

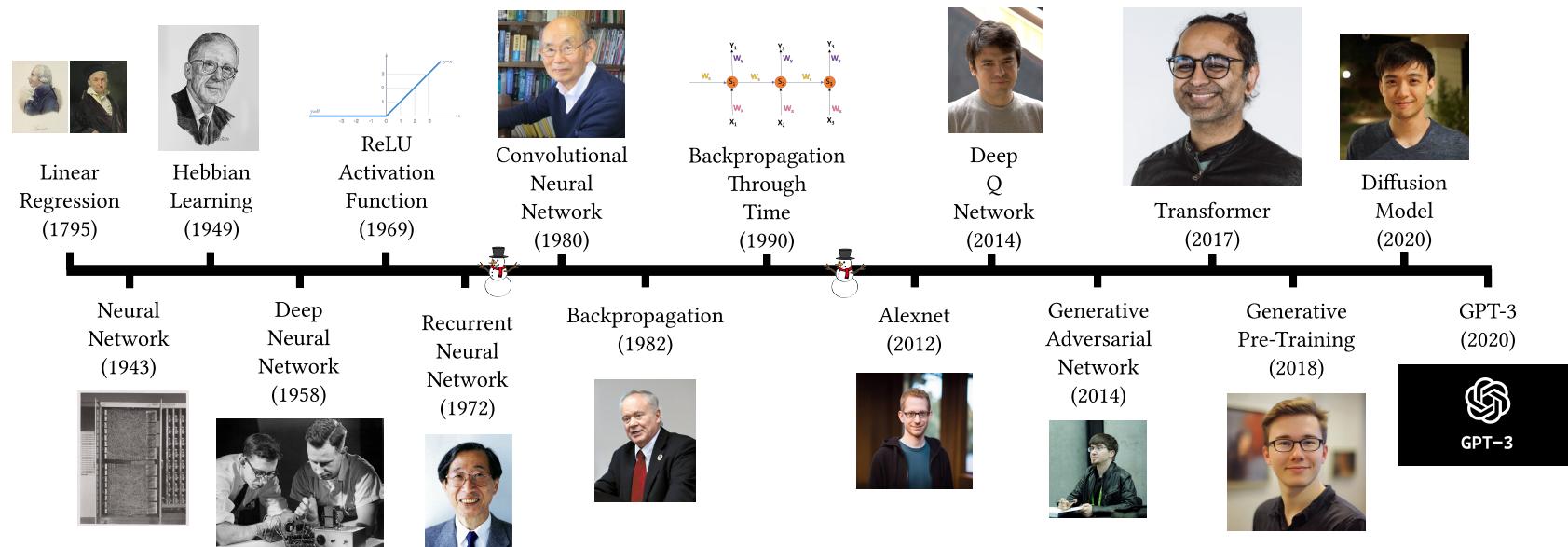
We call wide neural networks **perceptrons**

We call deep neural networks **multi-layer perceptrons (MLP)**

# Practical Considerations

We call wide neural networks **perceptrons**

We call deep neural networks **multi-layer perceptrons (MLP)**



# Practical Considerations

All the models we examine in this course will use MLPs

# Practical Considerations

All the models we examine in this course will use MLPs

- Recurrent neural networks

# Practical Considerations

All the models we examine in this course will use MLPs

- Recurrent neural networks
- Graph neural networks

# Practical Considerations

All the models we examine in this course will use MLPs

- Recurrent neural networks
- Graph neural networks
- Transformers

# Practical Considerations

All the models we examine in this course will use MLPs

- Recurrent neural networks
- Graph neural networks
- Transformers
- Chatbots

# Practical Considerations

All the models we examine in this course will use MLPs

- Recurrent neural networks
- Graph neural networks
- Transformers
- Chatbots

It is very important to understand MLPs!

# Practical Considerations

All the models we examine in this course will use MLPs

- Recurrent neural networks
- Graph neural networks
- Transformers
- Chatbots

It is very important to understand MLPs!

I will explain them again very simply

# Practical Considerations

A **layer** is a linear operation and an activation function

$$f\left(x, \begin{bmatrix} \mathbf{b} \\ \mathbf{W} \end{bmatrix}\right) = \sigma(\mathbf{b} + \mathbf{W}^\top x)$$

$$z_1 = f\left(x, \begin{bmatrix} \mathbf{b}_1 \\ \mathbf{W}_1 \end{bmatrix}\right)$$

$$z_2 = f\left(z_1, \begin{bmatrix} \mathbf{b}_2 \\ \mathbf{W}_2 \end{bmatrix}\right)$$

$$y = f\left(z_2, \begin{bmatrix} \mathbf{b}_2 \\ \mathbf{W}_2 \end{bmatrix}\right)$$

Many layers makes a deep neural network

# Practical Considerations

Let us create a wide neural network in colab! [https://colab.research.google.com/drive/1bLtf3QY-yROIif\\_EoQSU1WS7svd0q8j7?usp=sharing](https://colab.research.google.com/drive/1bLtf3QY-yROIif_EoQSU1WS7svd0q8j7?usp=sharing)

# Practical Considerations

Linear regression:

# Practical Considerations

Linear regression:

+ Analytical solution

# Practical Considerations

Linear regression:

- + Analytical solution
- + Low data requirement

# Practical Considerations

Linear regression:

- + Analytical solution
- + Low data requirement
- Poor scalability

# Practical Considerations

Linear regression:

- + Analytical solution
- + Low data requirement
- Poor scalability
- Poor polynomials generalization

# Practical Considerations

Linear regression:

- + Analytical solution
- + Low data requirement
- Poor scalability
- Poor polynomials generalization

Neural networks:

# Practical Considerations

Linear regression:

- + Analytical solution
- + Low data requirement
- Poor scalability
- Poor polynomials generalization

Neural networks:

- No analytical solution

# Practical Considerations

Linear regression:

- + Analytical solution
- + Low data requirement
- Poor scalability
- Poor polynomials generalization

Neural networks:

- No analytical solution
- High data requirement

# Practical Considerations

Linear regression:

- + Analytical solution
- + Low data requirement
- Poor scalability
- Poor polynomials generalization

Neural networks:

- No analytical solution
- High data requirement
- + Scale to large inputs

# Practical Considerations

Linear regression:

- + Analytical solution
- + Low data requirement
- Poor scalability
- Poor polynomials generalization

Neural networks:

- No analytical solution
- High data requirement
- + Scale to large inputs
- + Slightly better generalization

1. Review
2. Multivariate linear regression
3. Limitations of linear regression
4. History of neural networks
5. Biological neurons
6. Artificial neurons
7. Wide neural networks
8. Deep neural networks
9. Practical considerations

# Conclusion

There might be a quiz next time

# Conclusion

There might be a quiz next time

Always bring paper and a pen

# Conclusion

There might be a quiz next time

Always bring paper and a pen

You should be able to write a neural network layer mathematically

# Conclusion

There might be a quiz next time

Always bring paper and a pen

You should be able to write a neural network layer mathematically

You should also know the shapes of  $\theta, x, \bar{x}, y$