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

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

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

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

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

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

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.*

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

Review

Task: Given your education, predict your life expectancy

Review

Task: Given your education, predict your life expectancy

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

Review

Task: Given your education, predict your life expectancy

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

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

Review

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

Review

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

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

Review

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 θ such that

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

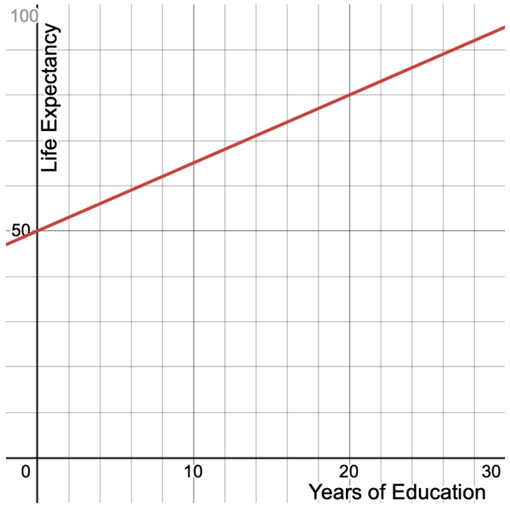
Review

Started with a linear function f

Review

Started with a linear function f

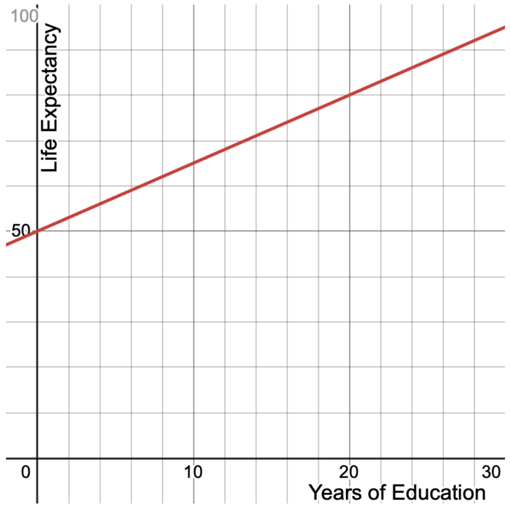
$$f(x, \boldsymbol{\theta}) = f\left(x, \begin{bmatrix} \theta_1 \\ \theta_0 \end{bmatrix}\right) = \theta_1 x + \theta_0$$



Review

Started with a linear function f

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

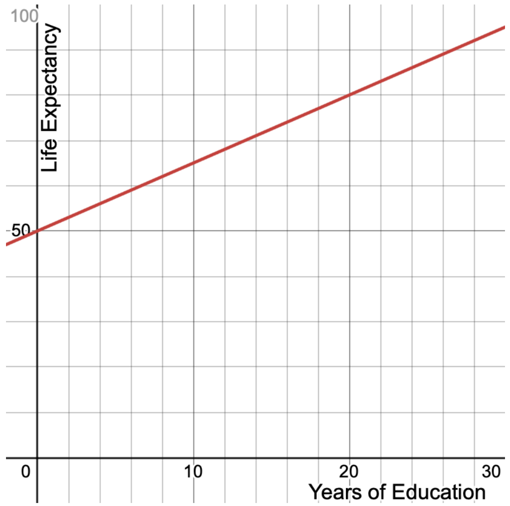


Then, we derived the square error function

Review

Started with a linear function f

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



Then, we derived the square error function

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

Review

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

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

Review

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

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

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

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

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

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

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

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

$$\boldsymbol{x} = \begin{bmatrix} x_{[1]} & x_{[2]} & \cdots & x_{[n]} \end{bmatrix}^\top, \boldsymbol{y} = \begin{bmatrix} y_{[1]} & y_{[2]} & \cdots & y_{[n]} \end{bmatrix}^\top$$

Review

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

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

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

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

$$\boldsymbol{x} = \begin{bmatrix} x_{[1]} & x_{[2]} & \cdots & x_{[n]} \end{bmatrix}^\top, \boldsymbol{y} = \begin{bmatrix} y_{[1]} & y_{[2]} & \cdots & y_{[n]} \end{bmatrix}^\top$$

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

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

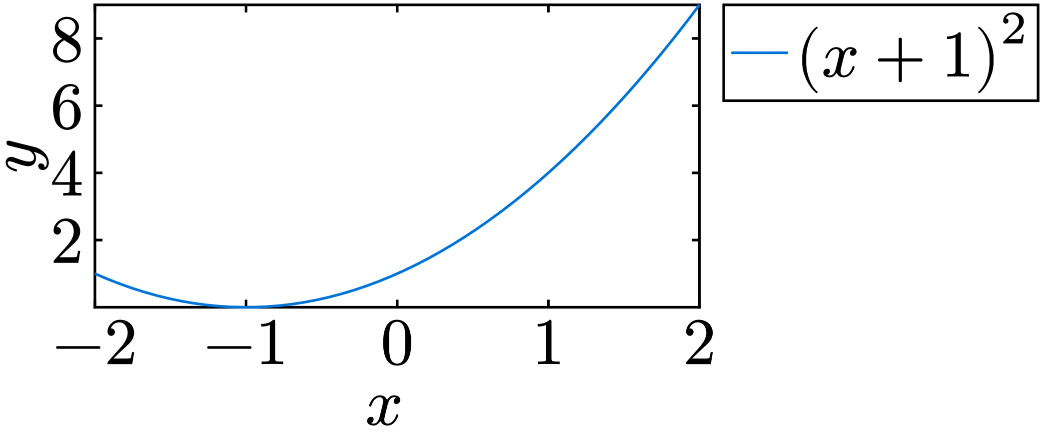
We introduced the $\arg\min$ operator

Review

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

We introduced the arg min operator

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

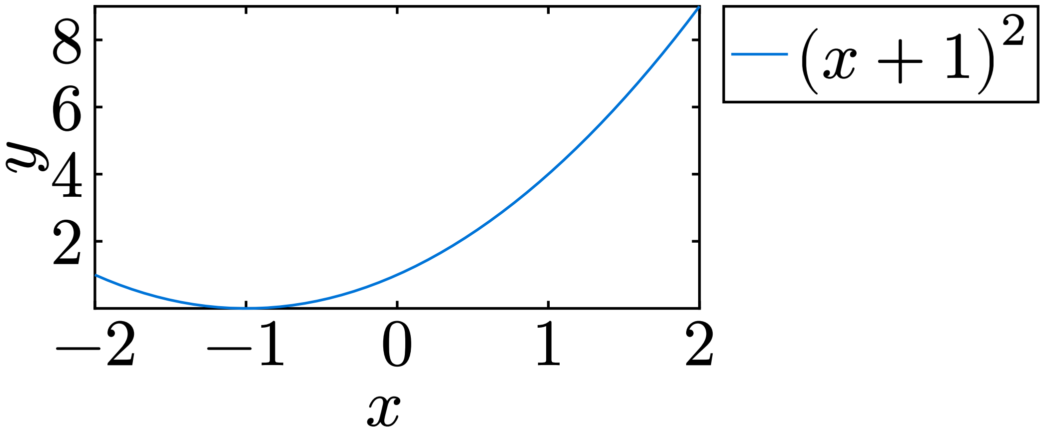


Review

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

Review

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

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

$$= \arg\min_{\boldsymbol{\theta}} \sum_{i=1}^{n} \left(f\left(x_{[i]}, \boldsymbol{\theta}\right) - y_{[i]}\right)^2$$

Review

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

Review

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

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

Review

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

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

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

Review

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

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

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

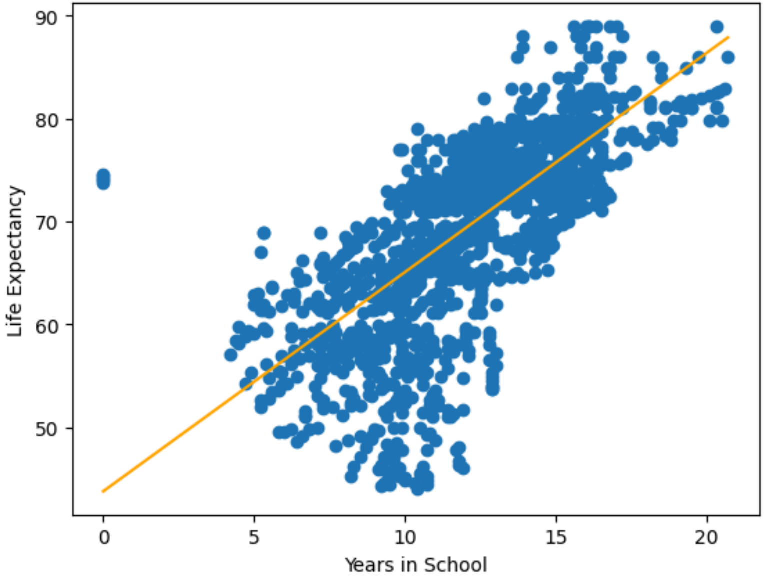
$$\boldsymbol{\theta} = \left(\boldsymbol{X}_D^\top \boldsymbol{X}_D\right)^{-1} \boldsymbol{X}_D^\top \boldsymbol{y}$$

Review

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

Review

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



Review

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

Review

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

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

Review

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

Review

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

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

Review

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

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

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

Review

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

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

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

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

Review

Finally, we discussed overfitting

Review

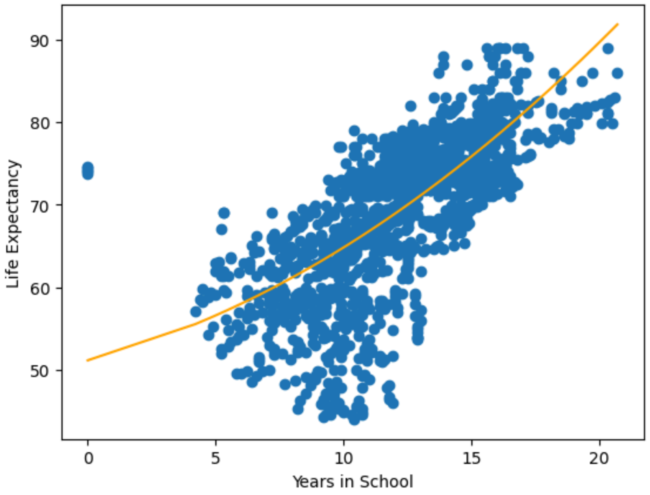
Finally, we discussed overfitting

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

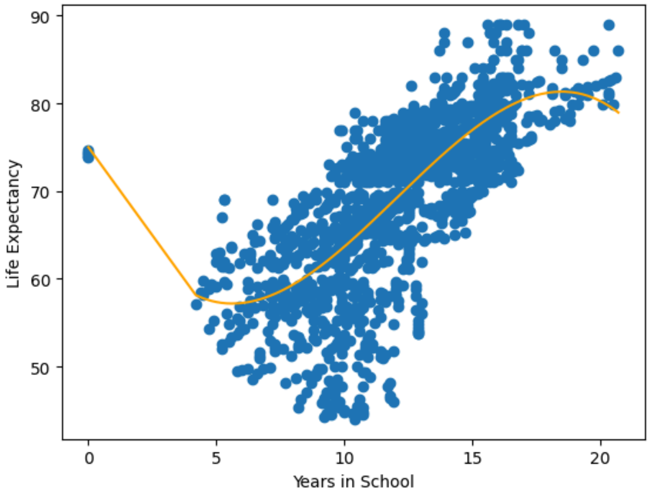
Review

Finally, we discussed overfitting

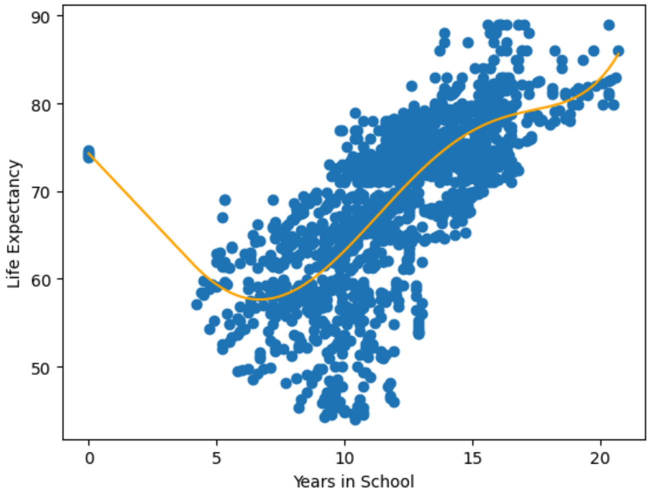
$$f(x, \boldsymbol{\theta}) = \theta_m x^m + \theta_{m-1} x^{m-1}, ..., \theta_1 x^1 + \theta_0$$



$m = 2$



$m = 3$



$m = 5$

Review

We care about **generalization** in machine learning

Review

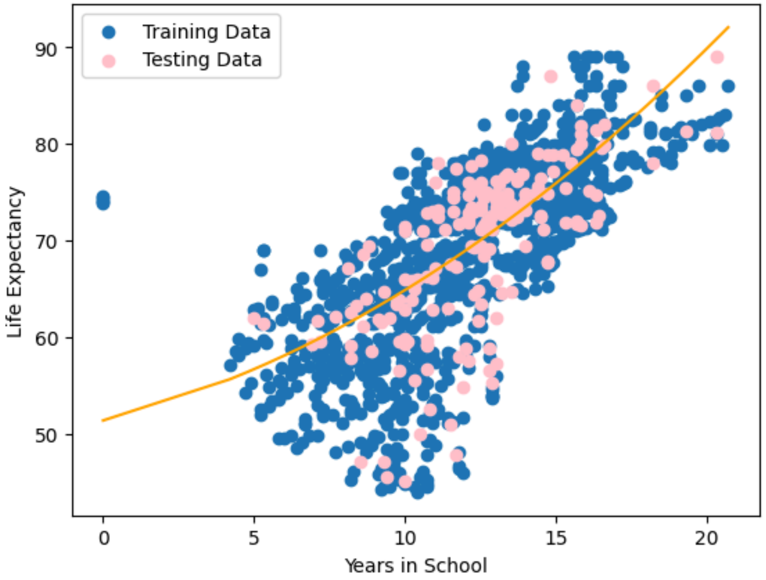
We care about **generalization** in machine learning

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

Review

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

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

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

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

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

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

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

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

The solution is the same as before

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

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

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

Limitations of Linear Regression

So far, we have seen:

One-dimensional polynomial functions

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

Multi-dimensional linear functions

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

Limitations of Linear Regression

So far, we have seen:

One-dimensional polynomial functions

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

Multi-dimensional linear functions

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

Combine them to create multi-dimensional polynomial functions

Limitations of Linear Regression

So far, we have seen:

One-dimensional polynomial functions

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

Multi-dimensional linear functions

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

Combine them to create multi-dimensional polynomial functions

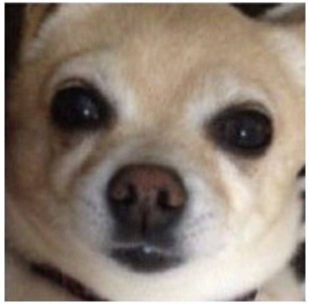
Limitations of Linear Regression

Let us do an example

Limitations of Linear Regression

Let us do an example

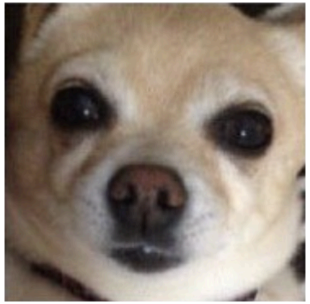
Task: predict how many ♥ a photo gets on social media



Limitations of Linear Regression

Let us do an example

Task: predict how many ♥ a photo gets on social media

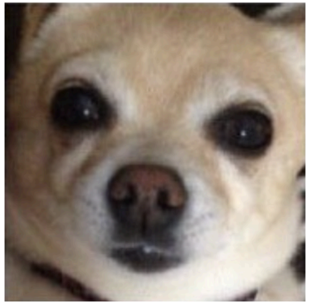


$$f : X \times \Theta \mapsto Y; \quad X : \text{Image}, \quad Y : \text{Number of } ♥$$

Limitations of Linear Regression

Let us do an example

Task: predict how many ♥ a photo gets on social media



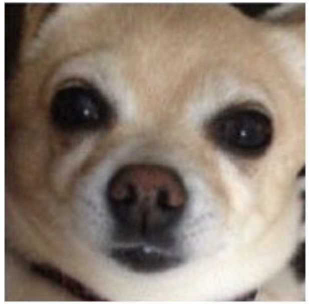
$$f : X \times \Theta \mapsto Y; \quad X : \text{Image}, \quad Y : \text{Number of } ♥$$

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

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

$$\boldsymbol{X}_D = \begin{bmatrix} \boldsymbol{x}_{D,[1]} & \cdots & \boldsymbol{x}_{D,[n]} \end{bmatrix}^\top$$

Limitations of Linear Regression

$$\boldsymbol{X}_D = \begin{bmatrix} \boldsymbol{x}_{D,[1]} & \cdots & \boldsymbol{x}_{D,[n]} \end{bmatrix}^\top$$

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

$$\begin{bmatrix} \underbrace{x_{[i],d_x}^m x_{[i],d_x-1}^m \ldots x_{[i],1}^m}_{(d_x \Rightarrow 1, x^m)} & \underbrace{x_{[i],d_x}^m x_{[i],d_x-1}^m \ldots x_{[i],2}^m}_{(d_x \Rightarrow 2, x^m)} & \cdots & \underbrace{x_{[i],d_x}^{m-1} x_{[i],d_x-1}^{m-1} \ldots x_{[i],1}^m}_{(d_x \Rightarrow 1, x^{m-1})} & \cdots \end{bmatrix}$$

Question: How many columns in this matrix?

Limitations of Linear Regression

$$\boldsymbol{X}_D = \begin{bmatrix} \boldsymbol{x}_{D,[1]} & \cdots & \boldsymbol{x}_{D,[n]} \end{bmatrix}^\top$$

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

$$\begin{bmatrix} \underbrace{x^m_{[i],d_x} x^m_{[i],d_x-1} \dots x^m_{[i],1}}_{(d_x \Rightarrow 1, x^m)} & \underbrace{x^m_{[i],d_x} x^m_{[i],d_x-1} \dots x^m_{[i],2}}_{(d_x \Rightarrow 2, x^m)} & \cdots & \underbrace{x^{m-1}_{[i],d_x} x^{m-1}_{[i],d_x-1} \dots x^m_{[i],1}}_{(d_x \Rightarrow 1, x^{m-1})} & \cdots \end{bmatrix}$$

Question: How many columns in this matrix?

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

Limitations of Linear Regression

$$\boldsymbol{X}_D = \begin{bmatrix} \boldsymbol{x}_{D,[1]} & \cdots & \boldsymbol{x}_{D,[n]} \end{bmatrix}^\top$$

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

$$\begin{bmatrix} \underbrace{x_{[i],d_x}^m x_{[i],d_x-1}^m \dots x_{[i],1}^m}_{(d_x \Rightarrow 1, x^m)} & \underbrace{x_{[i],d_x}^m x_{[i],d_x-1}^m \dots x_{[i],2}^m}_{(d_x \Rightarrow 2, x^m)} & \cdots & \underbrace{x_{[i],d_x}^{m-1} x_{[i],d_x-1}^{m-1} \dots x_{[i],1}^m}_{(d_x \Rightarrow 1, x^{m-1})} & \cdots \end{bmatrix}$$

Question: How many columns in this matrix?

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

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

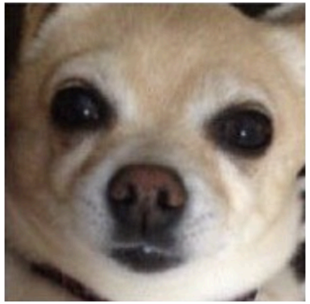
How big is 10^{96}?

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

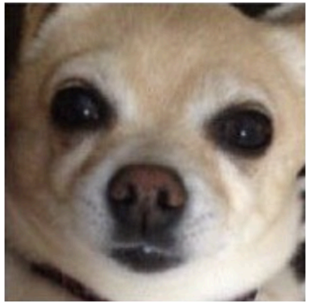
How big is 10^{96}?

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

$$\lim_{x \to \infty} \theta_m x^m + \theta_{m-1} x^{m-1} + \ldots$$

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

$$\lim_{x \to \infty} \theta_m x^m + \theta_{m-1} x^{m-1} + \dots$$

Equation of a polynomial

$$\lim_{x \to \infty} x^m \left(\theta_m + \frac{\theta_{m-1}}{x} + \dots \right)$$

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

$$\lim_{x \to \infty} \theta_m x^m + \theta_{m-1} x^{m-1} + \ldots$$

Equation of a polynomial

$$\lim_{x \to \infty} x^m \left(\theta_m + \frac{\theta_{m-1}}{x} + \ldots \right)$$

Factor out x^m

$$\lim_{x \to \infty} x^m \cdot \lim_{x \to \infty} \left(\theta_m + \frac{\theta_{m-1}}{x} + \ldots \right)$$

Split the limit (limit of products)

Limitations of Linear Regression

$$\lim_{x \to \infty} x^m \cdot \lim_{x \to \infty} \left(\theta_m + \frac{\theta_{m-1}}{x} + \ldots \right)$$

Split the limit (limit of products)

$$\left(\lim_{x \to \infty} x^m \right) \cdot (\theta_m + 0 + \ldots)$$

Evaluate right limit

Limitations of Linear Regression

$$\lim_{x \to \infty} x^m \cdot \lim_{x \to \infty} \left(\theta_m + \frac{\theta_{m-1}}{x} + \ldots \right)$$

Split the limit (limit of products)

$$\left(\lim_{x \to \infty} x^m \right) \cdot (\theta_m + 0 + \ldots)$$

Evaluate right limit

$$\theta_m \lim_{x \to \infty} x^m$$

Rewrite

Limitations of Linear Regression

$$\lim_{x \to \infty} x^m \cdot \lim_{x \to \infty} \left(\theta_m + \frac{\theta_{m-1}}{x} + \dots \right)$$

Split the limit (limit of products)

$$\left(\lim_{x \to \infty} x^m \right) \cdot (\theta_m + 0 + \dots)$$

Evaluate right limit

$$\theta_m \lim_{x \to \infty} x^m$$

Rewrite

$$\theta_m \lim_{x \to \infty} x^m = \infty$$

If $\theta_m > 0$

Limitations of Linear Regression

$\lim_{x \to \infty} x^m \cdot \lim_{x \to \infty} \left(\theta_m + \frac{\theta_{m-1}}{x} + \ldots \right)$	Split the limit (limit of products)
$\left(\lim_{x \to \infty} x^m \right) \cdot (\theta_m + 0 + \ldots)$	Evaluate right limit
$\theta_m \lim_{x \to \infty} x^m$	Rewrite
$\theta_m \lim_{x \to \infty} x^m = \infty$	If $\theta_m > 0$
$\theta_m \lim_{x \to \infty} x^m = -\infty$	If $\theta_m < 0$

Limitations of Linear Regression

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

Limitations of Linear Regression

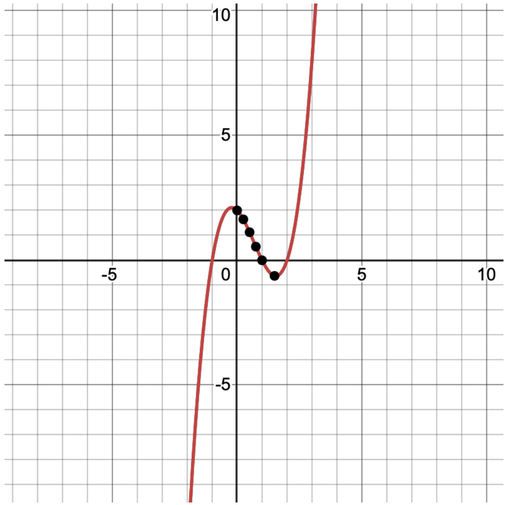
Polynomials quickly tend towards $-\infty, \infty$ outside of the support

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

Limitations of Linear Regression

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

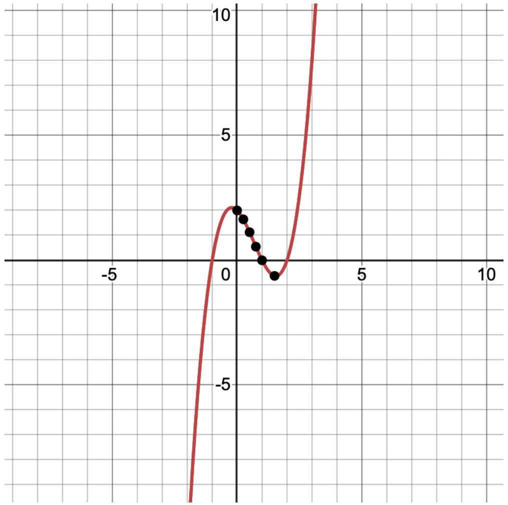
$$f(x) = x^3 - 2x^2 - x + 2$$



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

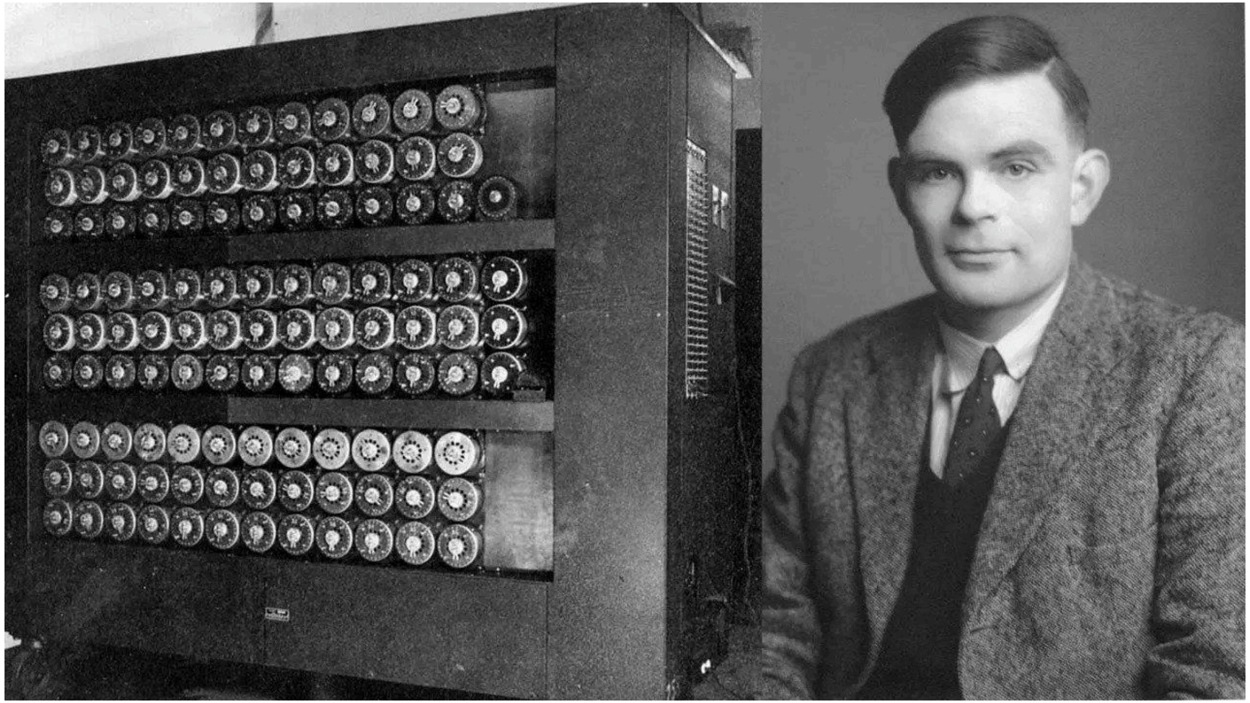
In 1939-1945, there was a World War

Militaries invested funding for research, and invented the computer

History of Neural Networks

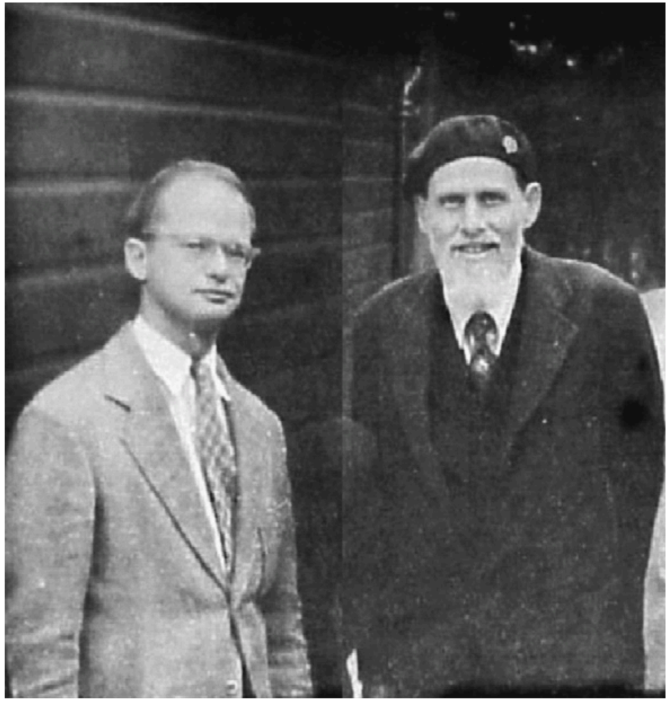
In 1939-1945, there was a World War

Militaries invested funding for research, and invented the computer

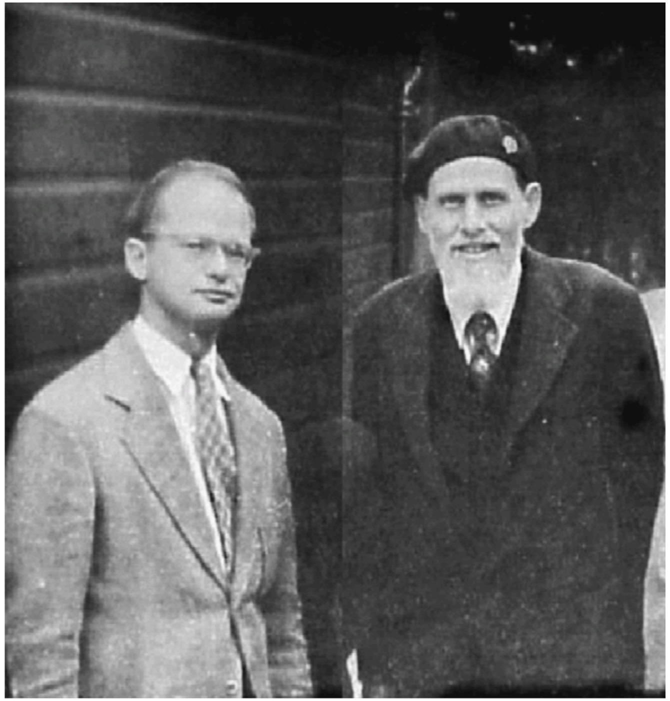


History of Neural Networks

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



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



They designed the theory for the first neural network

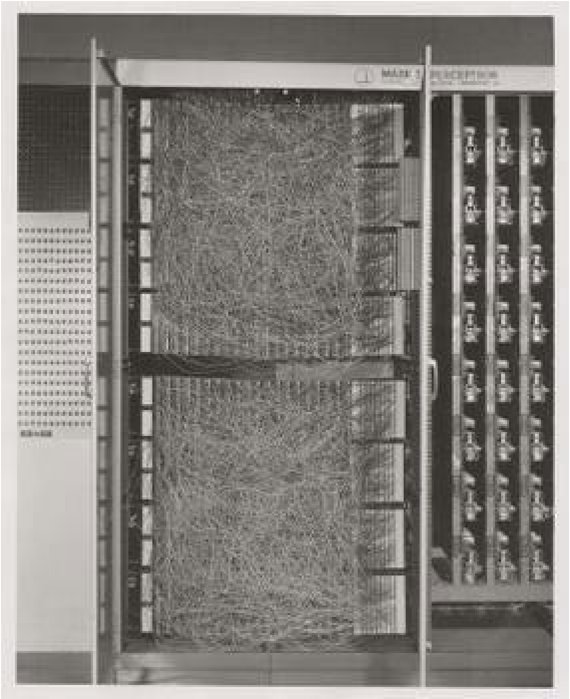
History of Neural Networks

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

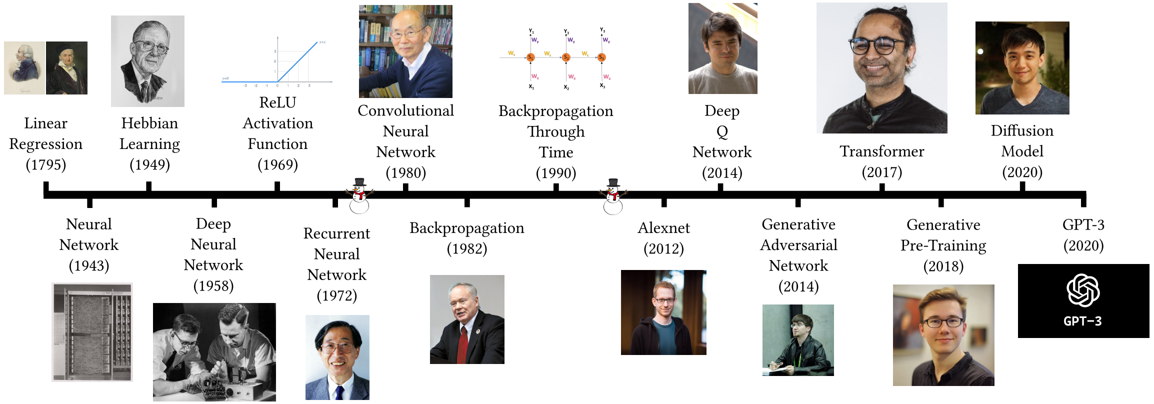


History of Neural Networks

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

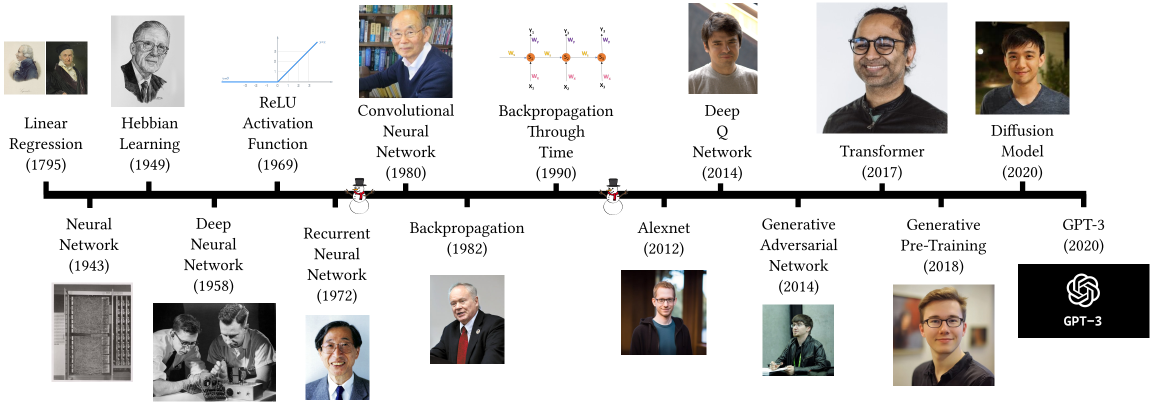
History of Neural Networks

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 $\rightarrow$ Biological neural network

History of Neural Networks

Brains and neural networks rely on **neurons**

Brain: Biological neurons $\rightarrow$ Biological neural network

Computer: Artificial neurons $\rightarrow$ Artificial neural network

History of Neural Networks

Brains and neural networks rely on **neurons**

Brain: Biological neurons $\rightarrow$ Biological neural network

Computer: Artificial neurons $\rightarrow$ Artificial neural network

First, let us review biological neurons

History of Neural Networks

Brains and neural networks rely on **neurons**

Brain: Biological neurons $\rightarrow$ Biological neural network

Computer: Artificial neurons $\rightarrow$ Artificial neural network

First, let us review biological neurons

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

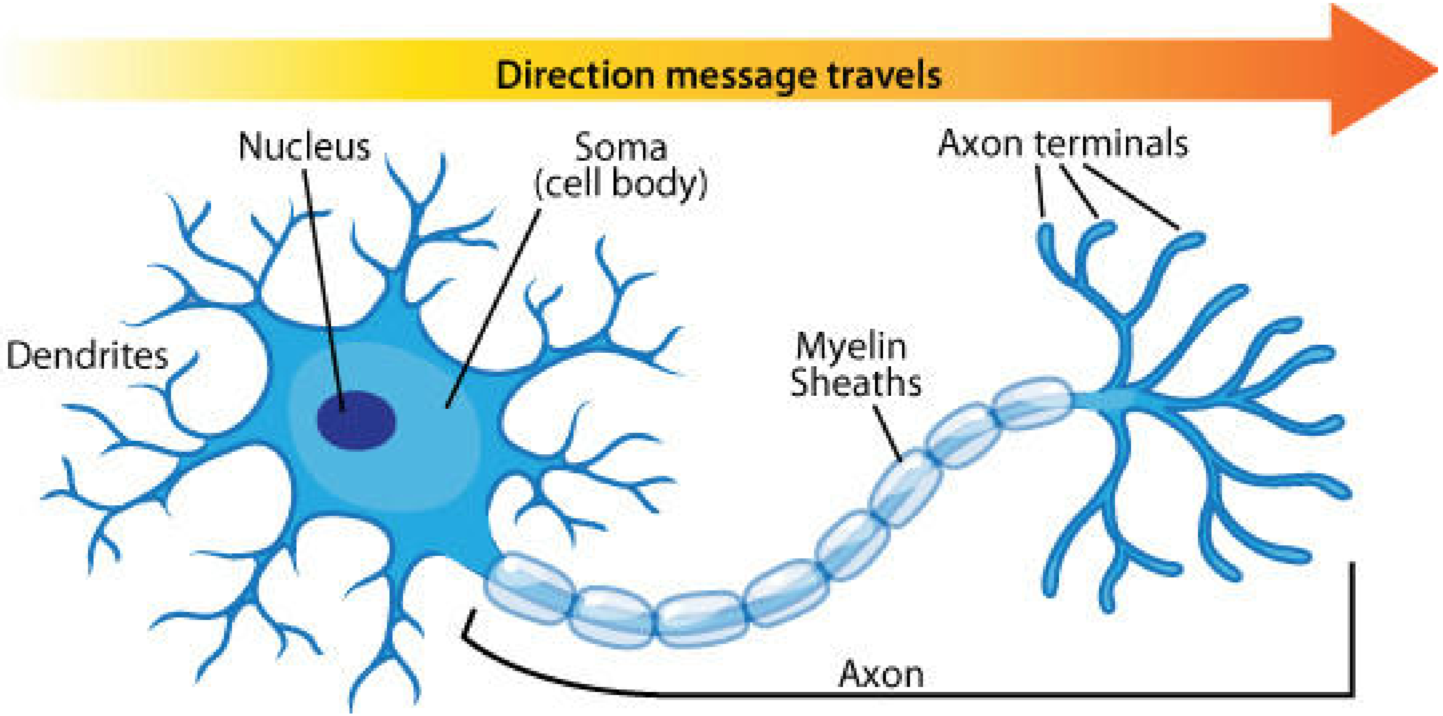
Agenda

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

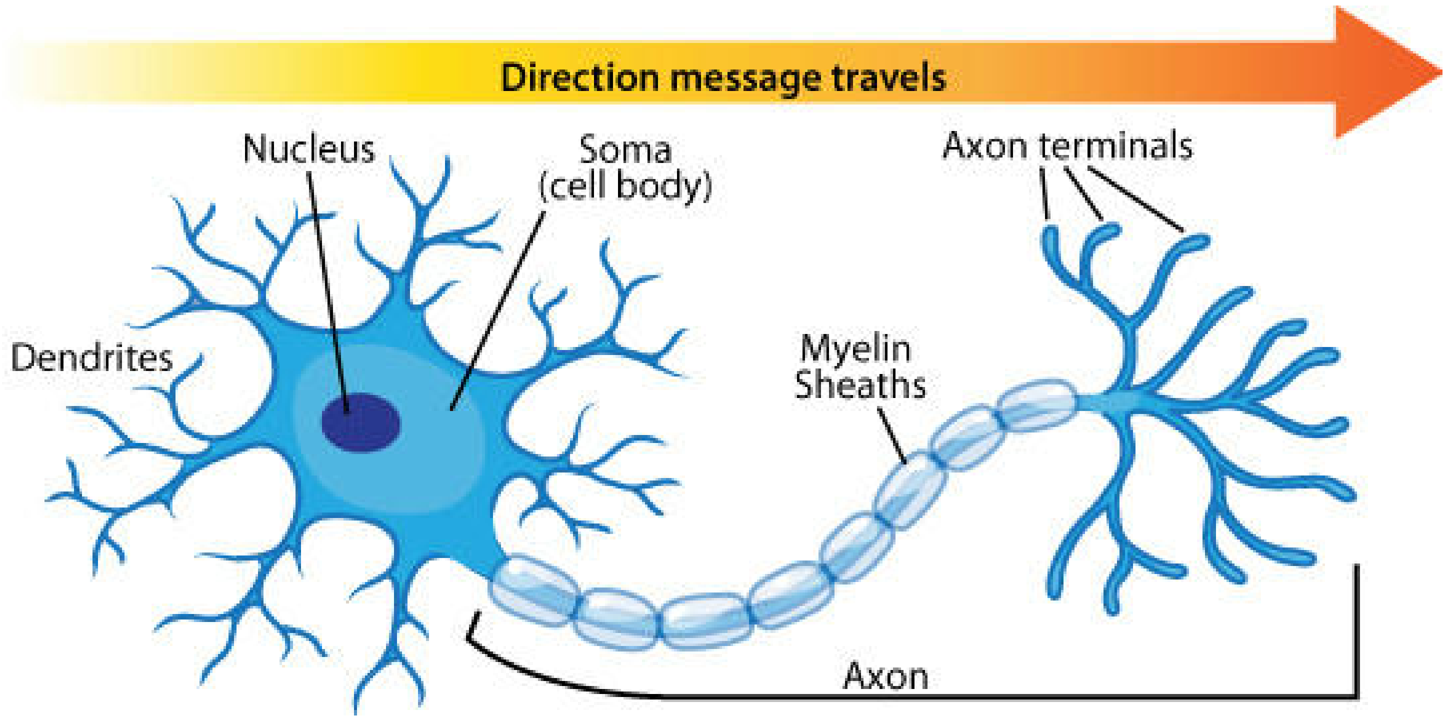
1. Review
2. Multivariate linear regression
3. Limitations of linear regression
4. History of neural networks
5. **Biological neurons**
6. Artificial neurons
7. Wide neural networks
8. Deep neural networks
9. Practical considerations

Biological Neurons



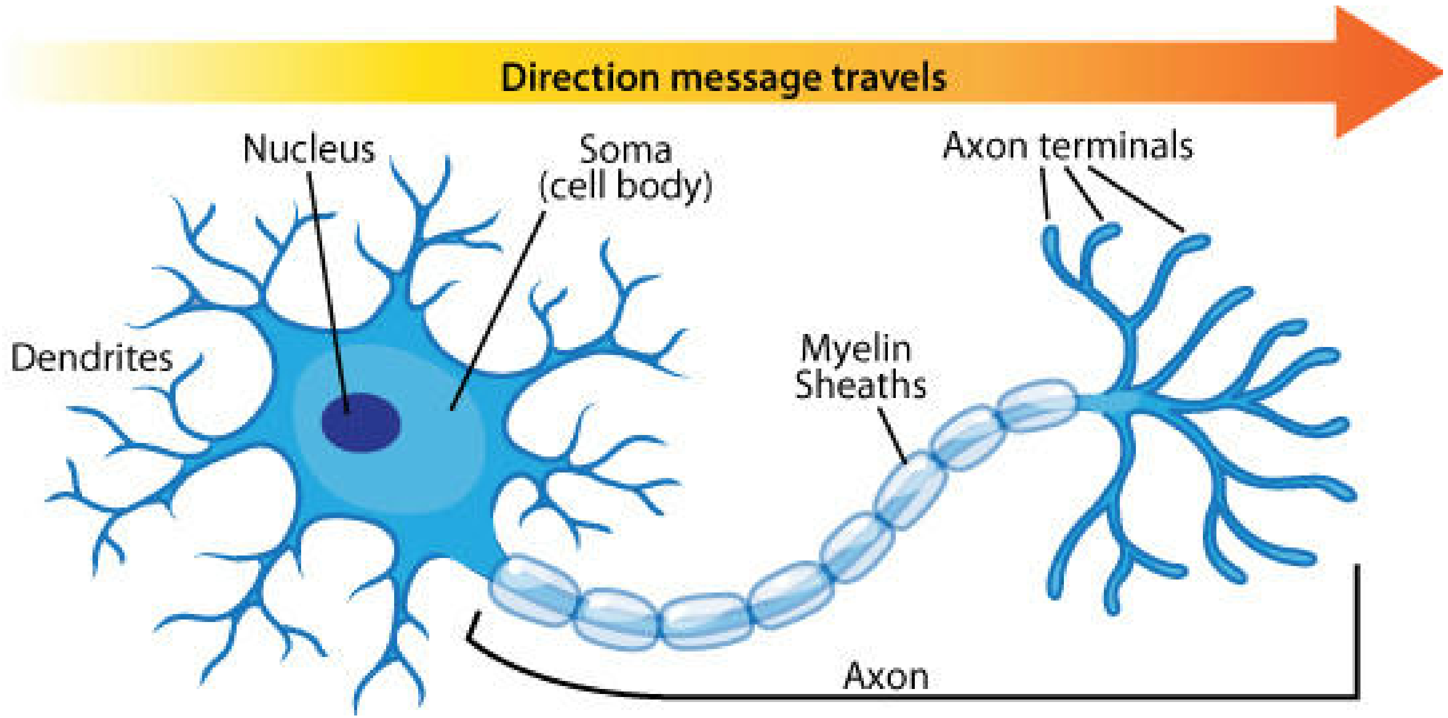
A simplified neuron consists of many parts

Biological Neurons



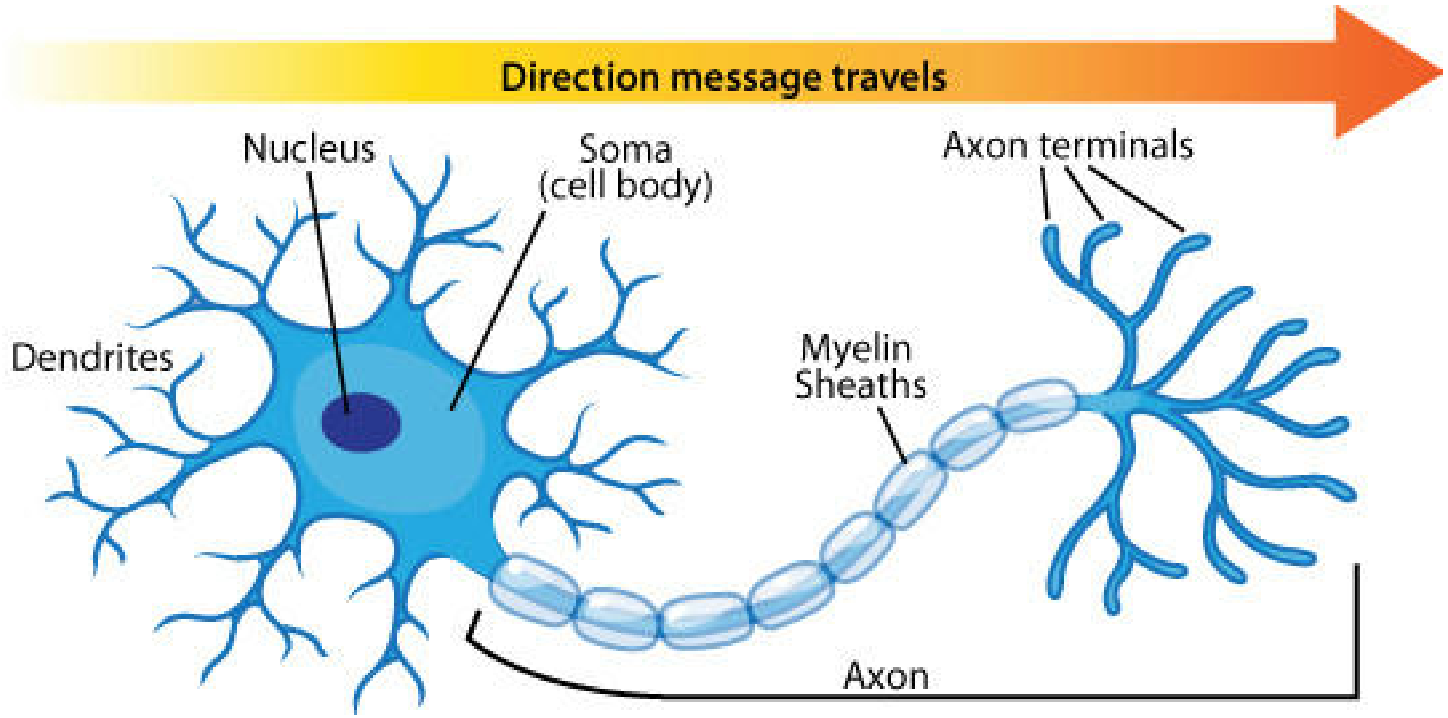
Neurons send messages based on messages received from other neurons

Biological Neurons



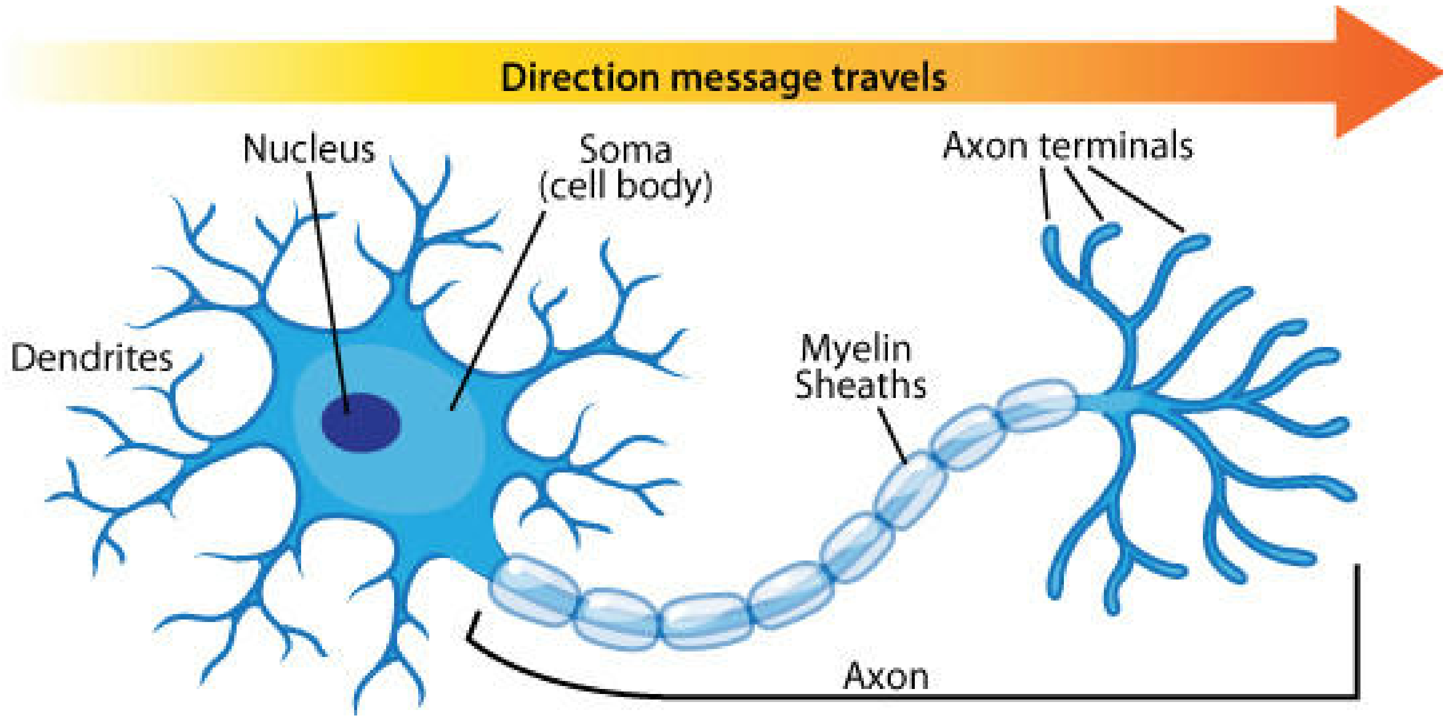
Incoming electrical signals travel along dendrites

Biological Neurons



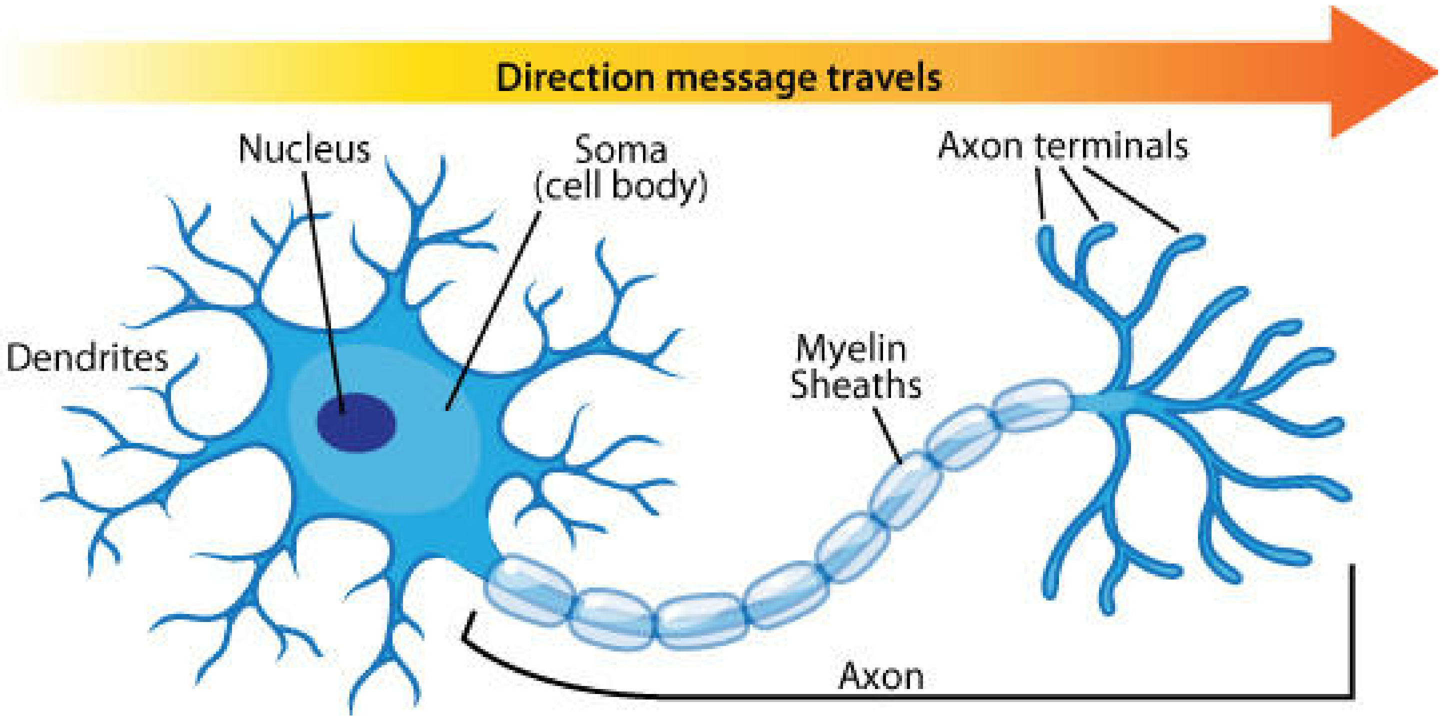
Dendrites are not all equal! Different dendrites have different diameters and structures

Biological Neurons



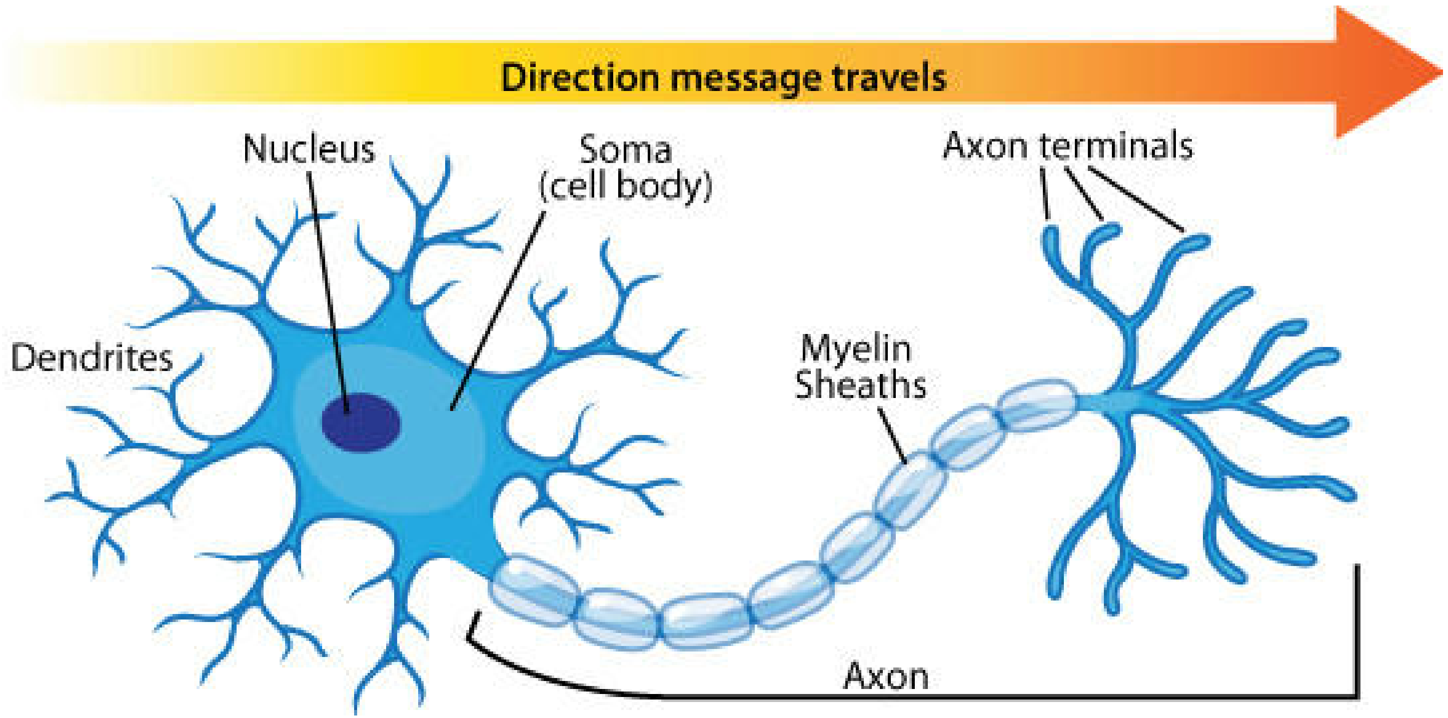
Electrical charges collect in the Soma (cell body)

Biological Neurons



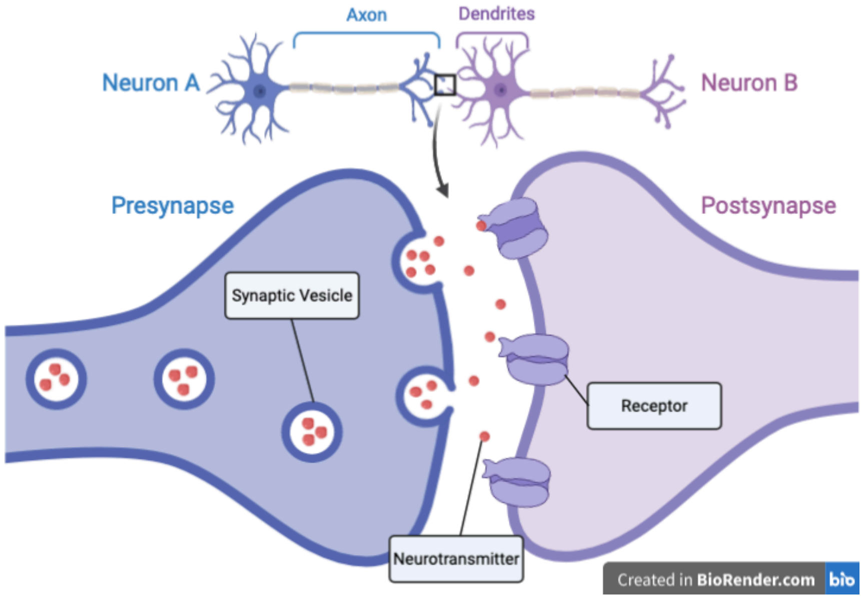
The axon outputs an electrical signal to other neurons

Biological Neurons



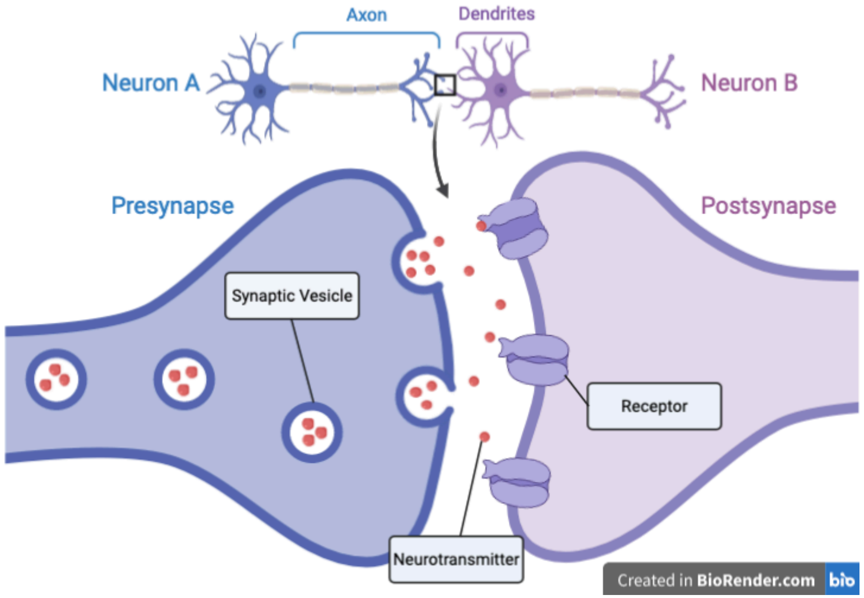
The axon terminals will connect to dendrites of other neurons through a synapse

Biological Neurons



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

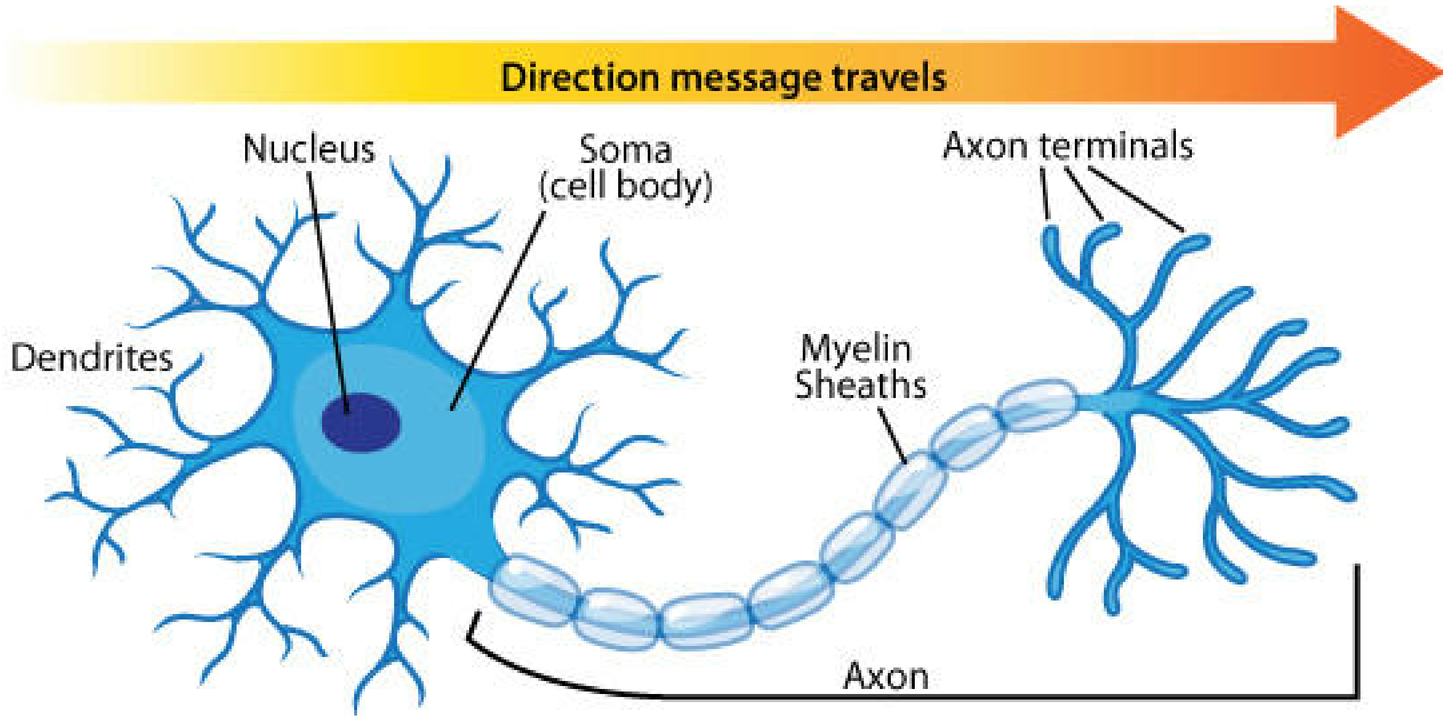
Biological Neurons



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

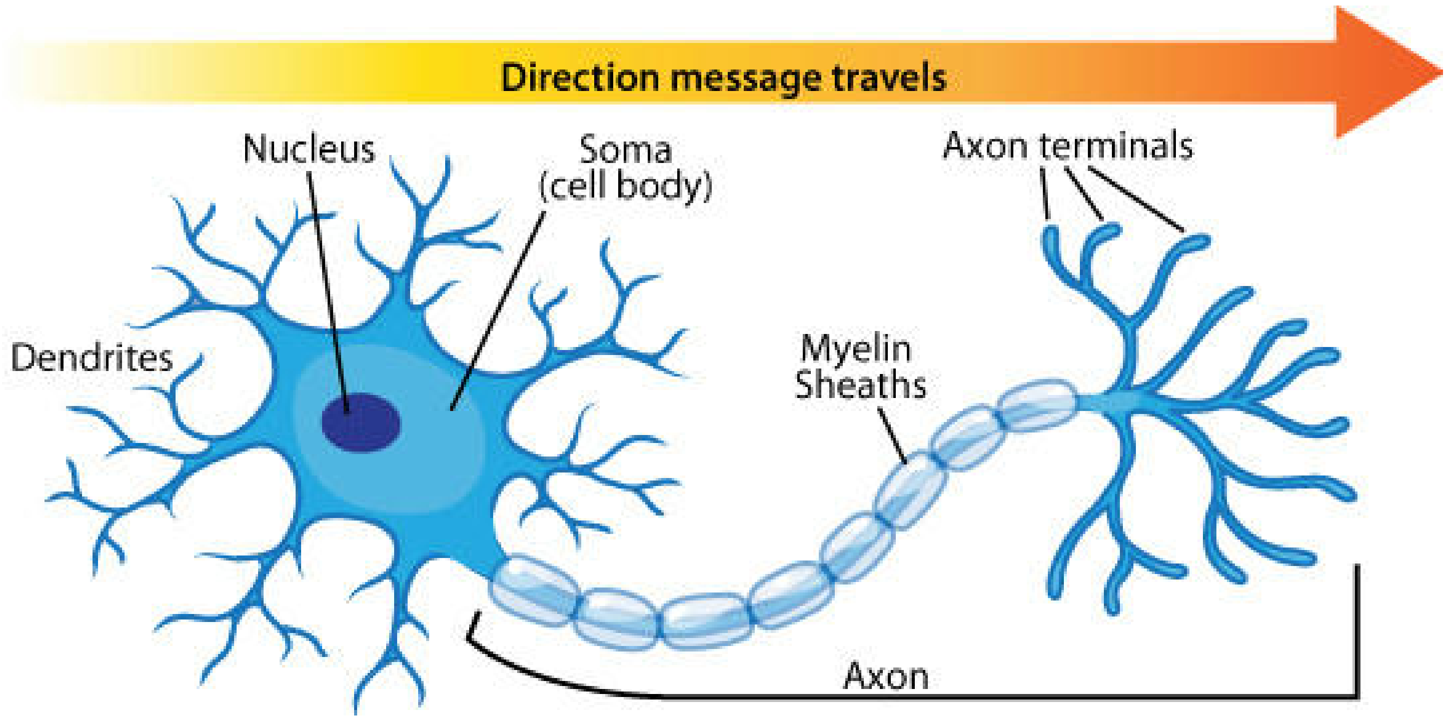
Synaptic weight determines how well a signal crosses the gap

Biological Neurons



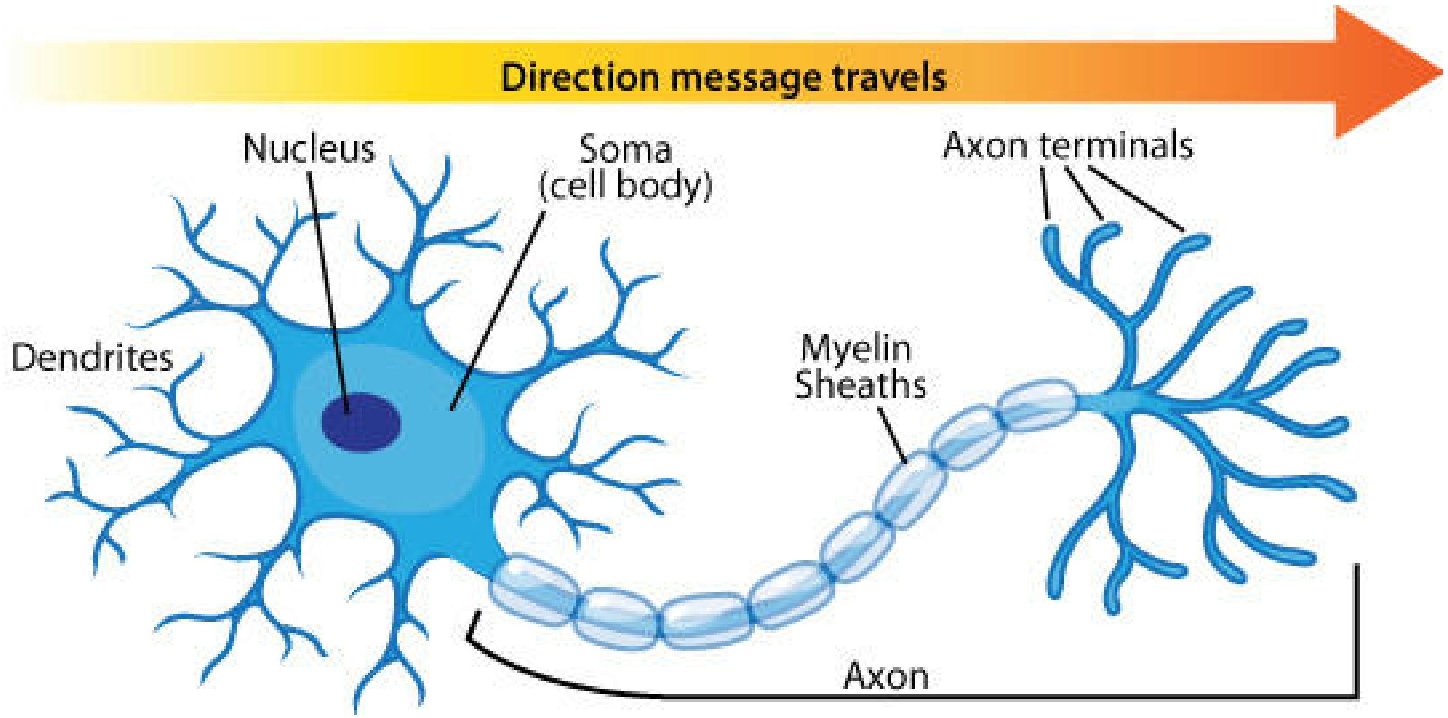
For our purposes, we can model the axon terminals, dendrites, and synapses to be one thing

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

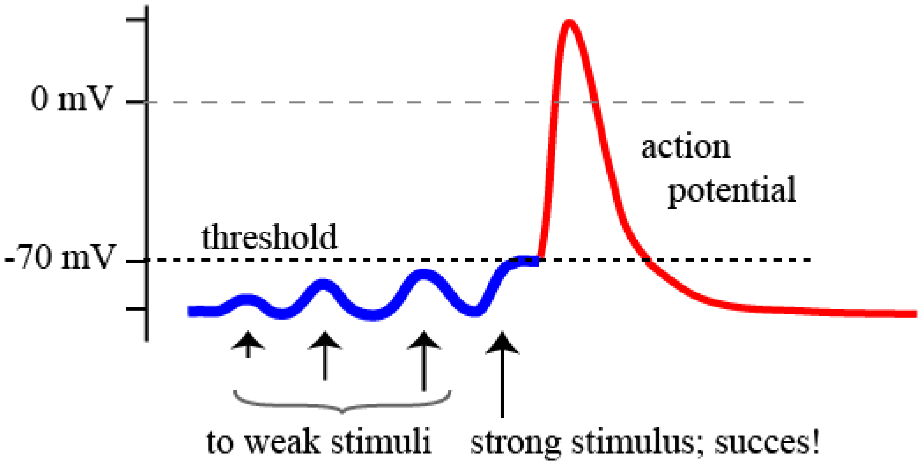
Biological Neurons

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

Biological Neurons

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

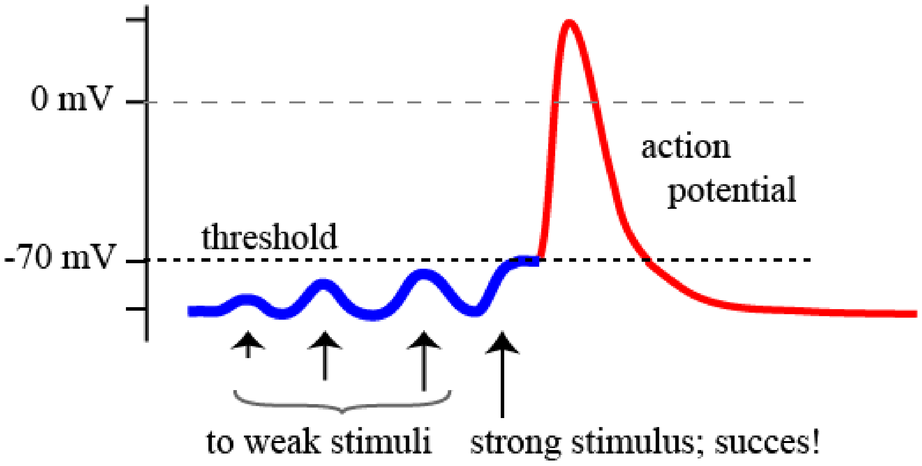
Incoming impulses (via dendrites) change the electric potential of the neuron



Biological Neurons

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

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

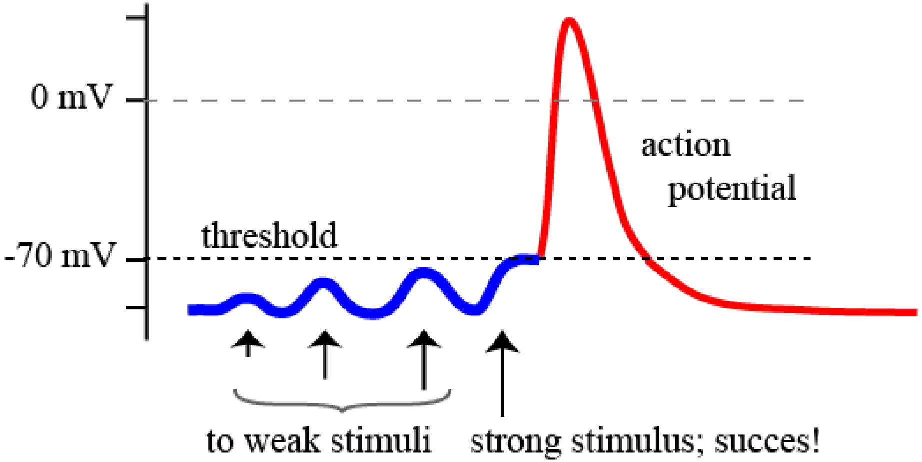


In a parallel circuit, we can sum voltages together

Biological Neurons

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

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

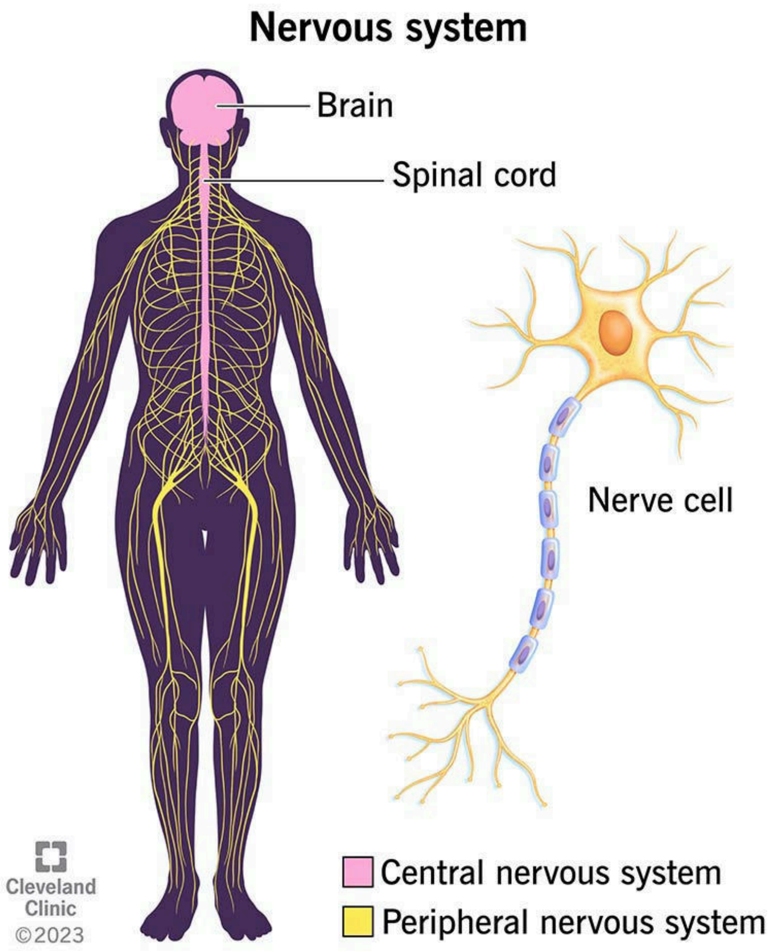


In a parallel circuit, we can sum voltages together

Many active dendrites will add together and trigger an impulse

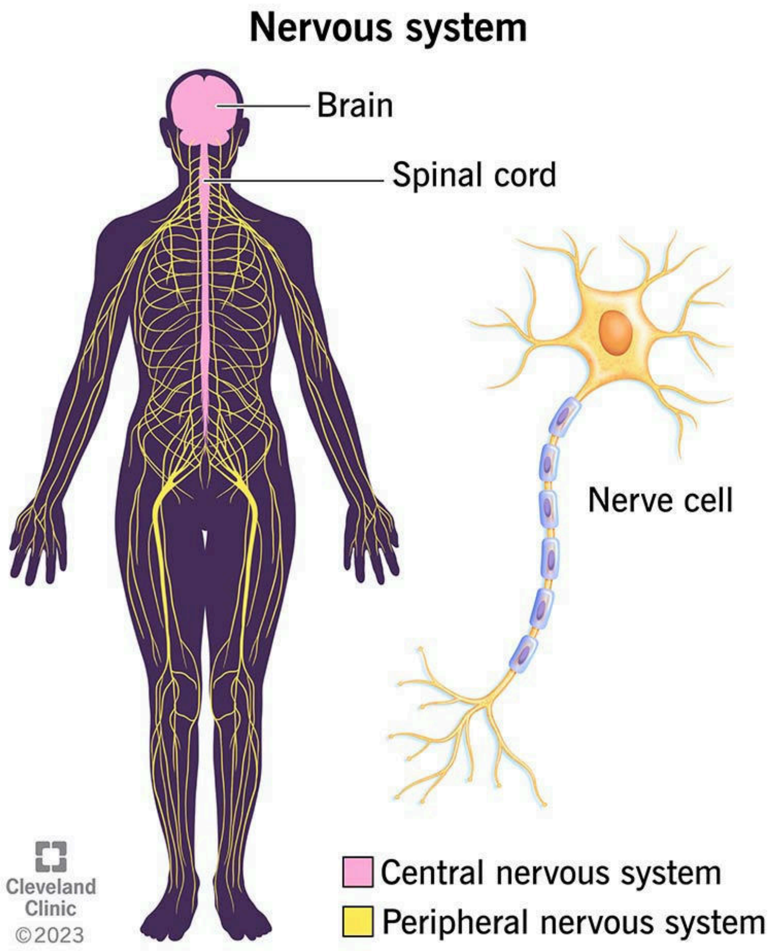
Biological Neurons

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



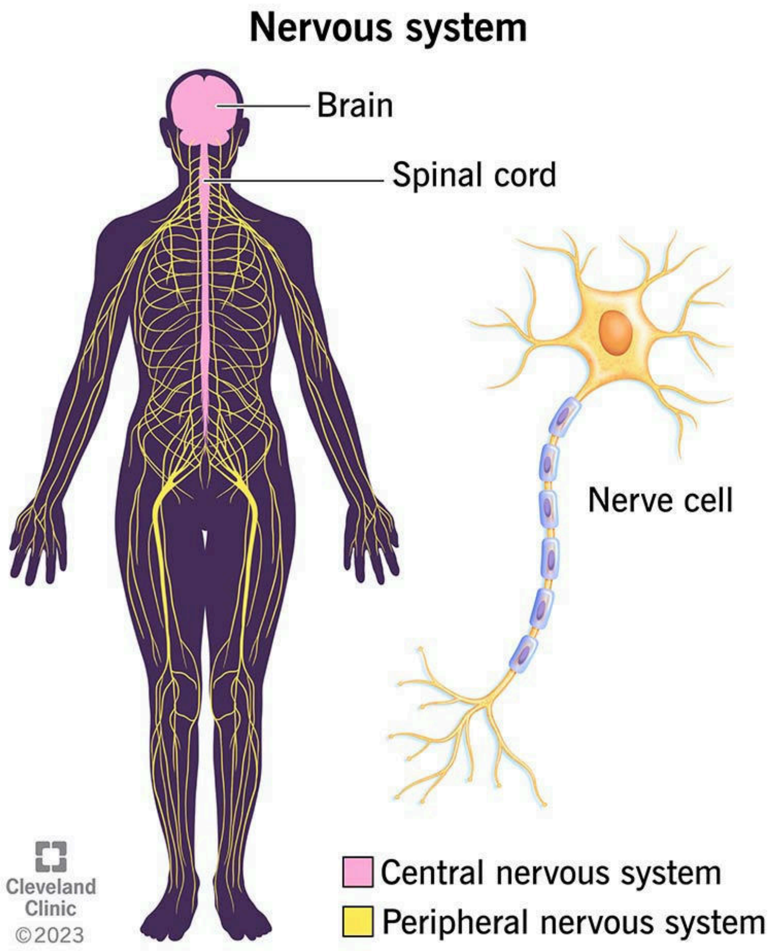
Biological Neurons

When the signal reaches the brain,
we will think



Biological Neurons

After thinking, we will take action



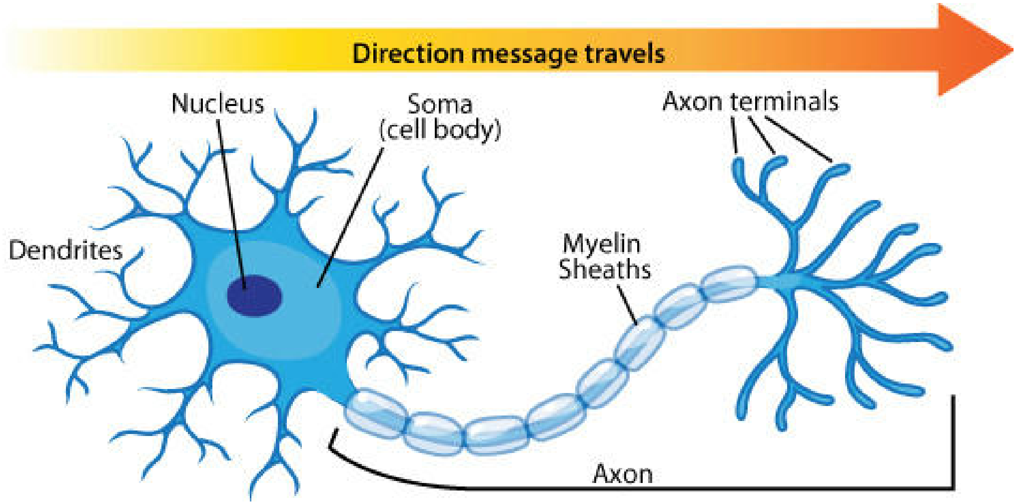
Agenda

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

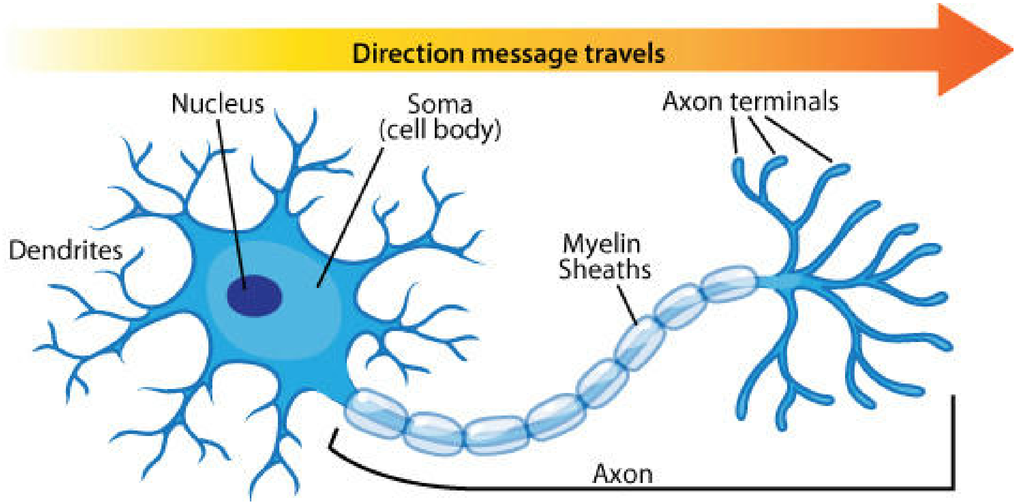
Agenda

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

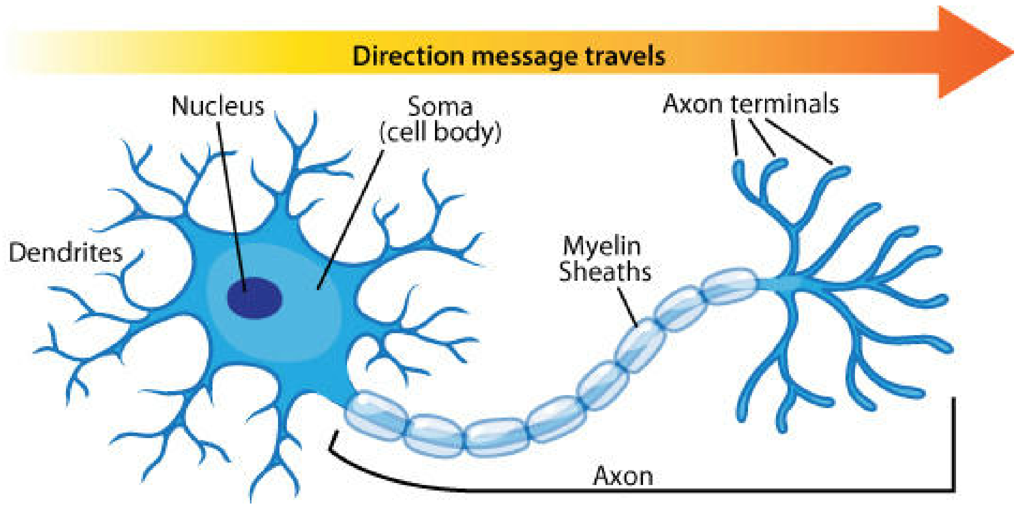


Artificial Neurons



Question: How could we write a neuron as a function? $f : \_\_ \mapsto \_\_$

Artificial Neurons



Question: How could we write a neuron as a function? $\quad f : \_\_ \mapsto \_\_$

Answer:

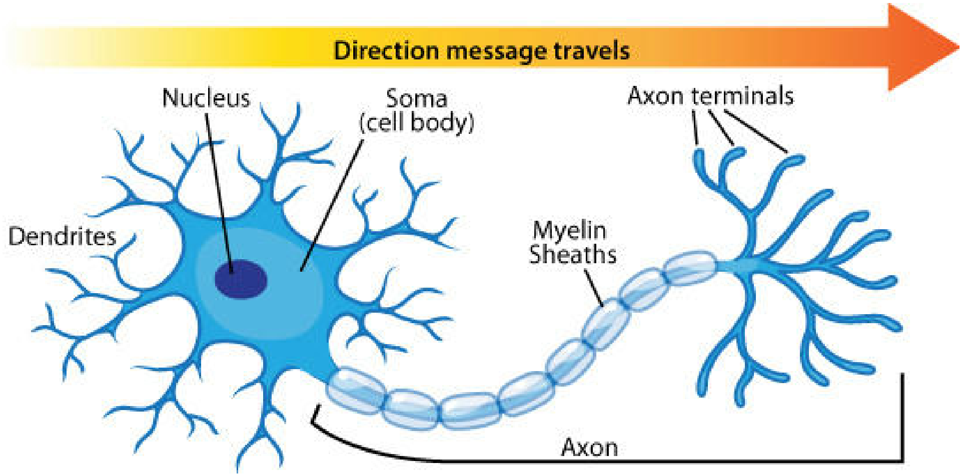
$$f : \underbrace{\mathbb{R}^{d_x}}_{\text{Dendrite voltages}} \times \underbrace{\mathbb{R}^{d_x}}_{\text{Synaptic weight}} \mapsto \underbrace{\mathbb{R}}_{\text{Axon voltage}}$$

Artificial Neurons

Let us implement an artifical neuron as a function

Artificial Neurons

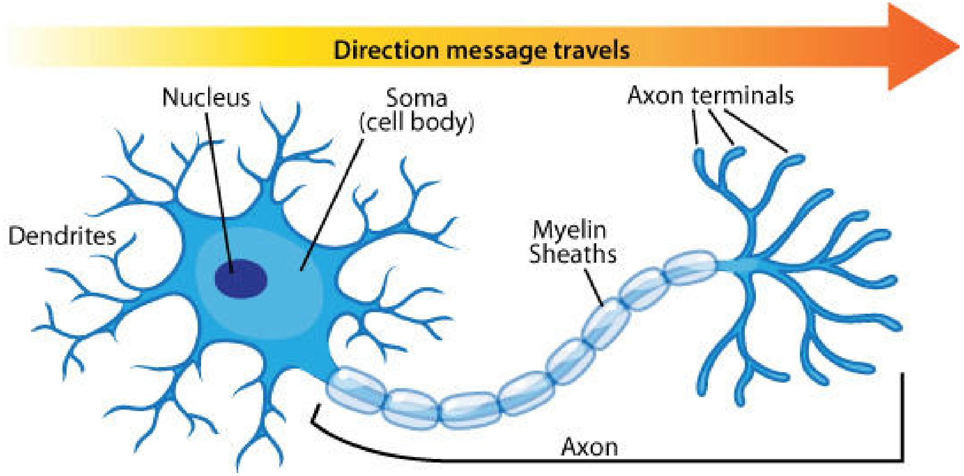
Let us implement an artifical neuron as a function



Neuron has a structure of dendrites with synaptic weights

Artificial Neurons

Let us implement an artifical neuron as a function



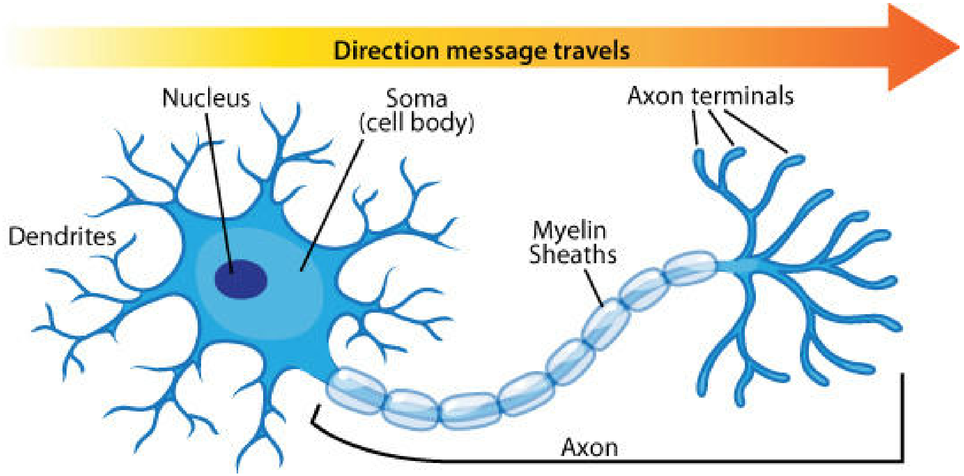
Neuron has a structure of dendrites with synaptic weights

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

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

Artificial Neurons

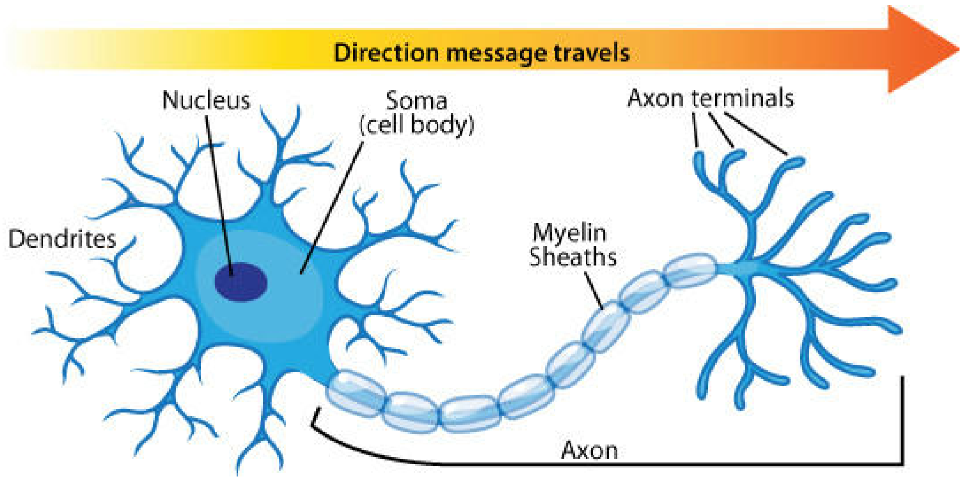
Let us implement an artifical neuron as a function



Each incoming dendrite has some voltage potential

Artificial Neurons

Let us implement an artifical neuron as a function



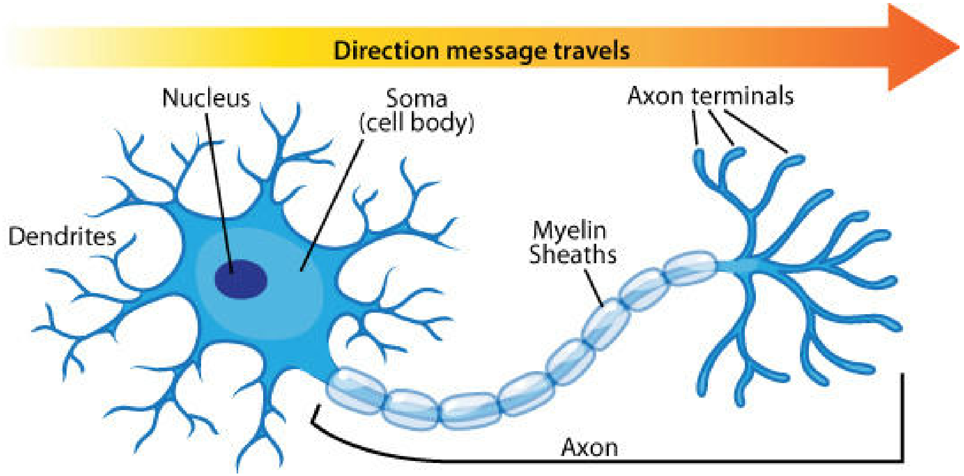
Each incoming dendrite has some voltage potential

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

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

Artificial Neurons

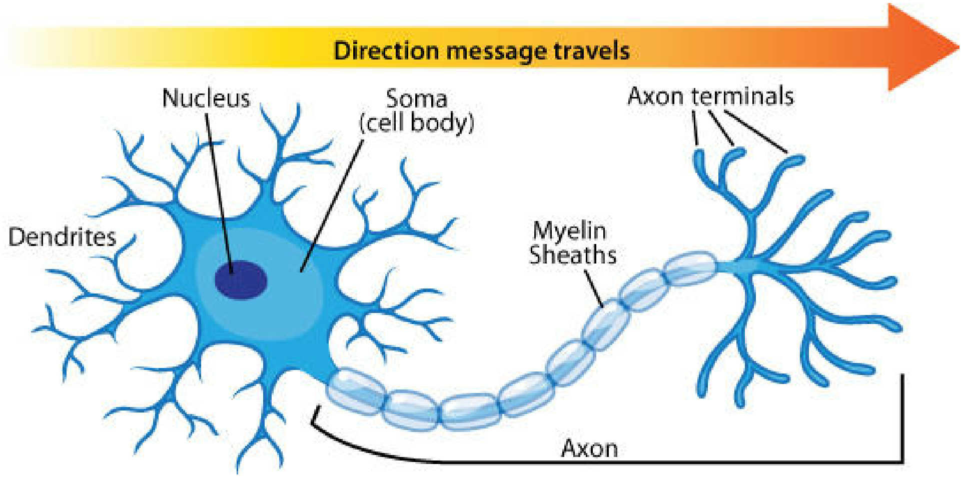
Let us implement an artifical neuron as a function



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

Artificial Neurons

Let us implement an artifical neuron as a function



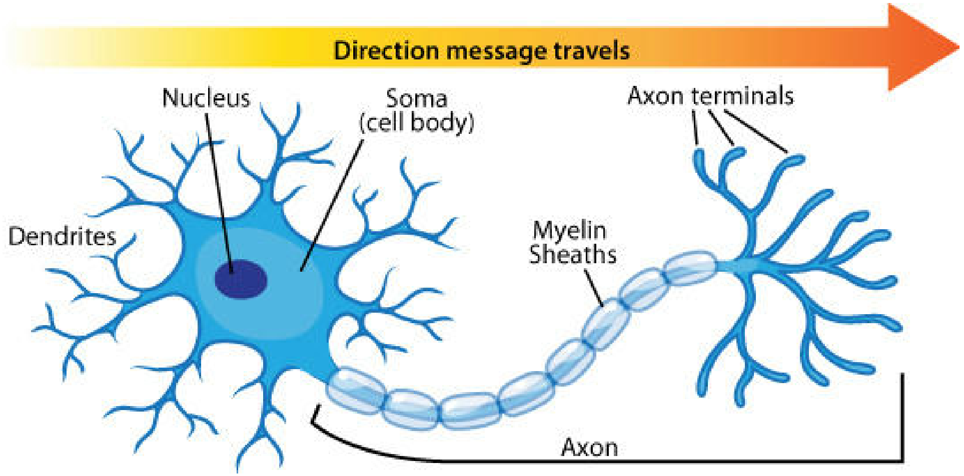
Voltage potentials sum together to give us the voltage in the cell body

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

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

Artificial Neurons

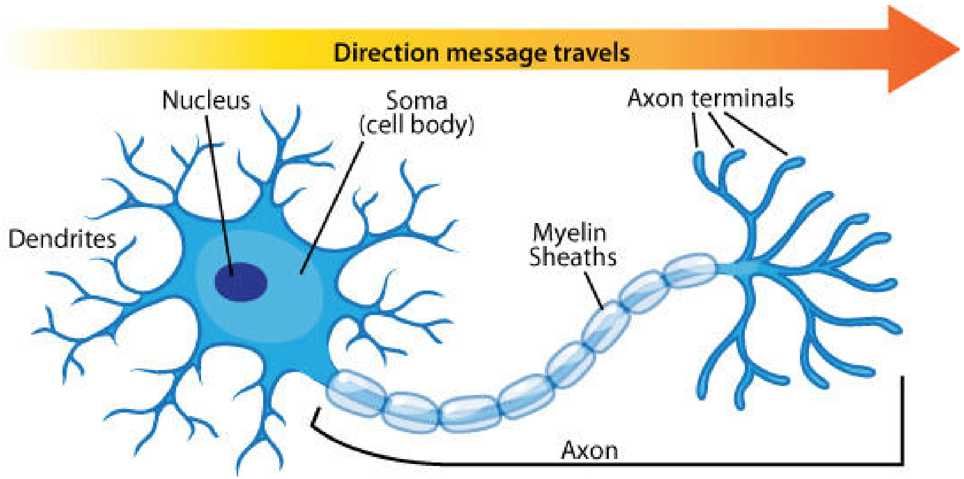
Let us implement an artifical neuron as a function



The axon fires only if the voltage is over a threshold

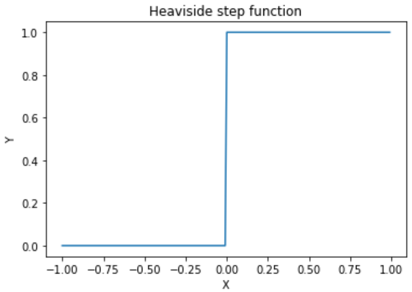
Artificial Neurons

Let us implement an artifical neuron as a function



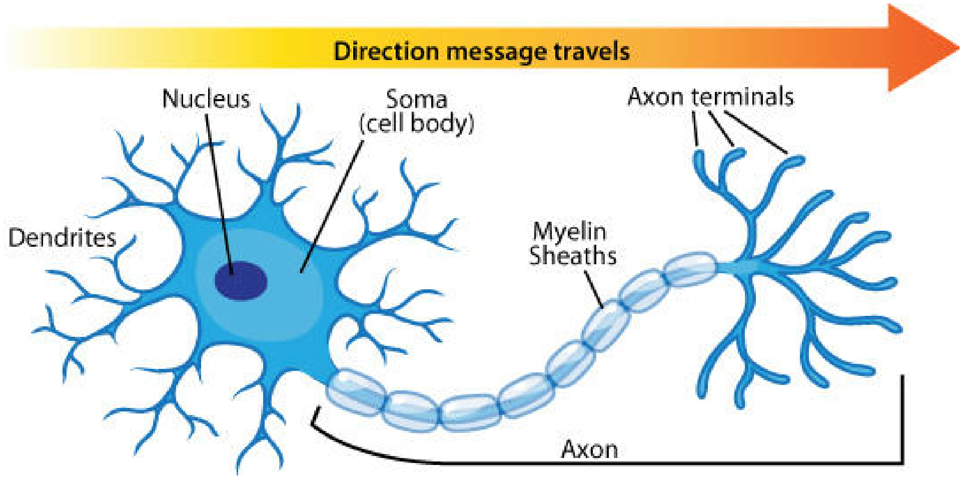
The axon fires only if the voltage is over a threshold

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



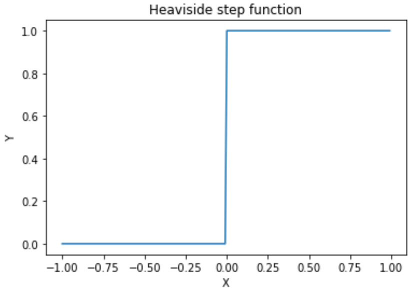
Artificial Neurons

Let us implement an artifical neuron as a function



The axon fires only if the voltage is over a threshold

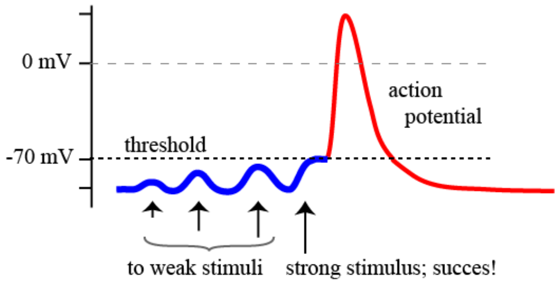
$$\sigma(x) = H(x) =$$

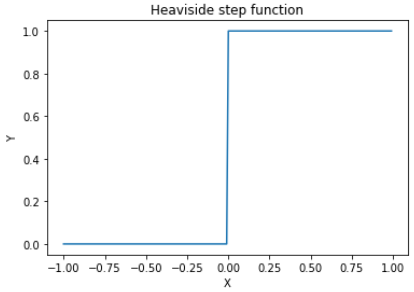


$$f\left(\begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix}, \begin{bmatrix} \theta_1 \\ \vdots \\ \theta_n \end{bmatrix}\right) = \sigma\left(\sum_{j=1}^{d_x} \theta_j x_j\right)$$

Artificial Neurons

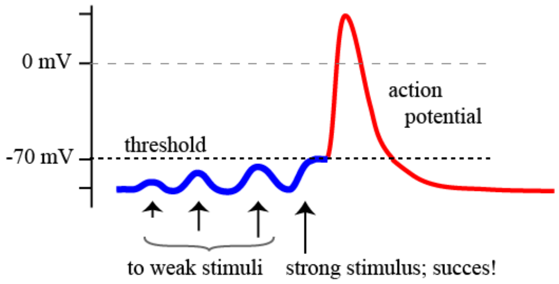
Maybe we want to vary the activation threshold

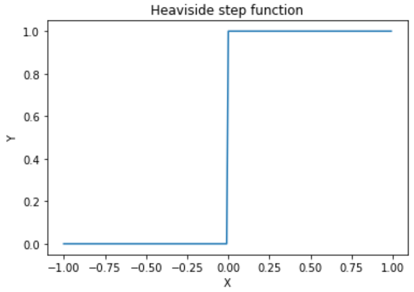




Artificial Neurons

Maybe we want to vary the activation threshold

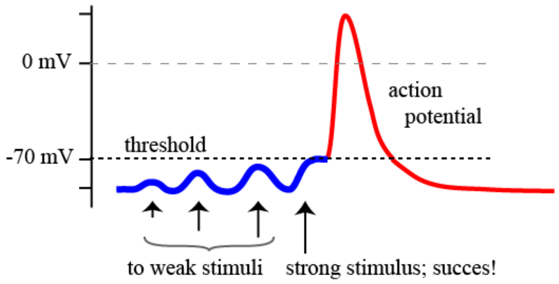


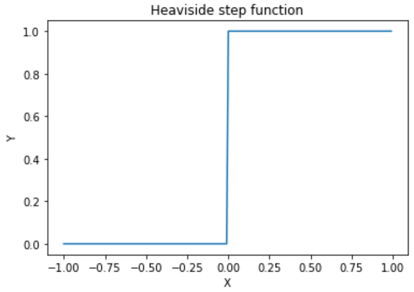


$$f\left(\begin{bmatrix} 1 \\ x_1 \\ \vdots \\ x_{d_x} \end{bmatrix}, \begin{bmatrix} \theta_0 \\ \theta_1 \\ \vdots \\ \theta_{d_x} \end{bmatrix}\right) = \sigma\left(\theta_0 + \sum_{j=1}^{d_x} \theta_j x_j\right) = \sigma\left(\sum_{j=0}^{d_x} \theta_j x_j\right)$$

Artificial Neurons

Maybe we want to vary the activation threshold





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

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

Artificial Neurons

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

Artificial Neurons

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

This is the artificial neuron!

Artificial Neurons

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

This is the artificial neuron!

Let us write out the full equation for a neuron

Artificial Neurons

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

This is the artificial neuron!

Let us write out the full equation for a neuron

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

Artificial Neurons

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

This is the artificial neuron!

Let us write out the full equation for a neuron

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

Question: Does this look familiar to anyone?

Artificial Neurons

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

This is the artificial neuron!

Let us write out the full equation for a neuron

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

Question: Does this look familiar to anyone?

Answer: Inside σ is the multivariate linear model!

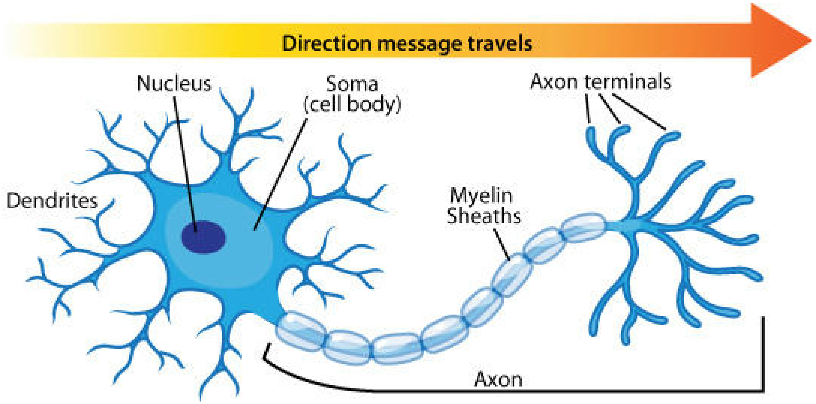
$$f(\boldsymbol{x}, \boldsymbol{\theta}) = \theta_{d_x} x_{d_x} + \theta_{d_x - 1} x_{d_x - 1} + \ldots + \theta_0 1$$

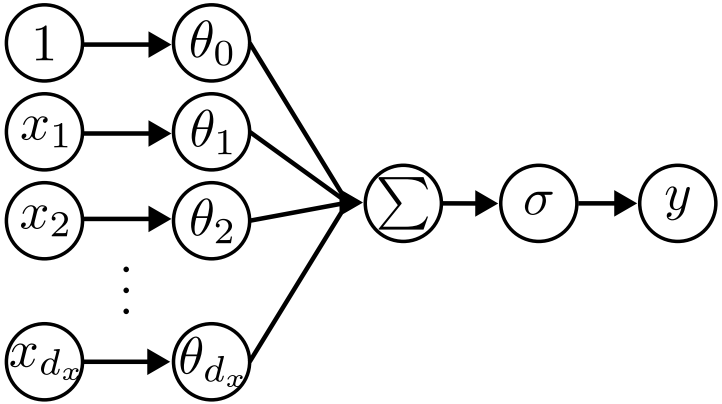
Artificial Neurons

We model a neuron using a linear model and activation function

Artificial Neurons

We model a neuron using a linear model and activation function





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

Artificial Neurons

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

Artificial Neurons

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

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

Artificial Neurons

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

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

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

Artificial Neurons

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

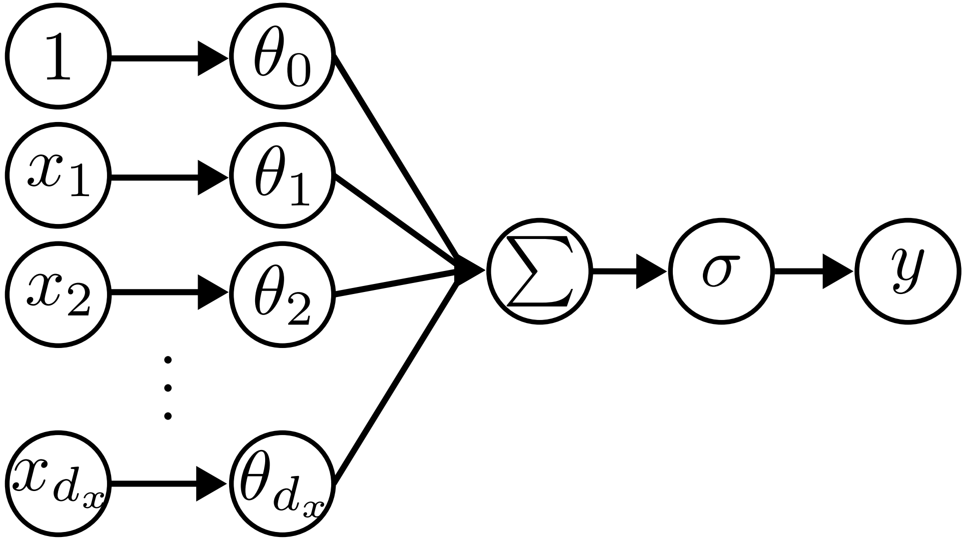
Sometimes, we will write $\boldsymbol{\theta}$ as a bias and weight $b, \boldsymbol{w}$

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

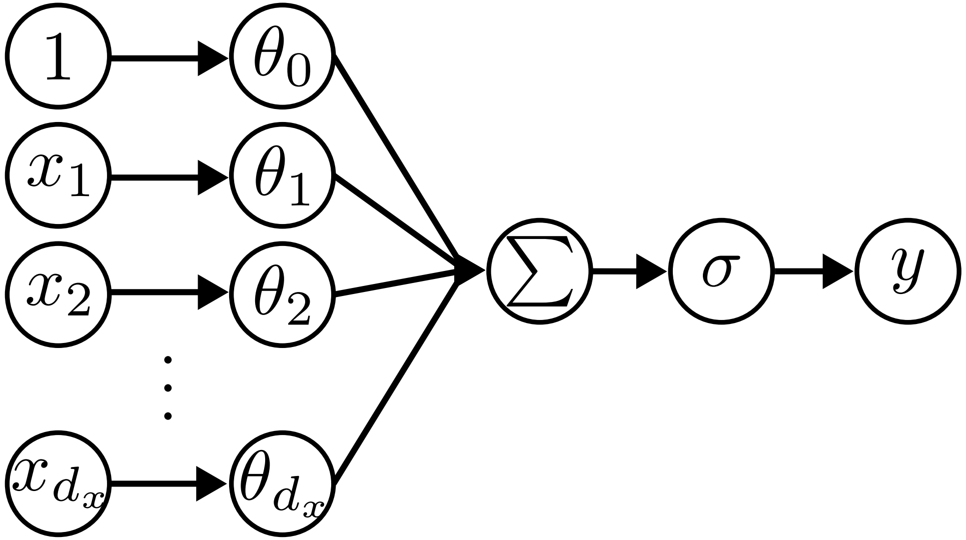
$$f\left(\boldsymbol{x}, \begin{bmatrix} b \\ \boldsymbol{w} \end{bmatrix}\right) = b + \boldsymbol{w}^\top \boldsymbol{x}$$

Relax

Artificial Neurons

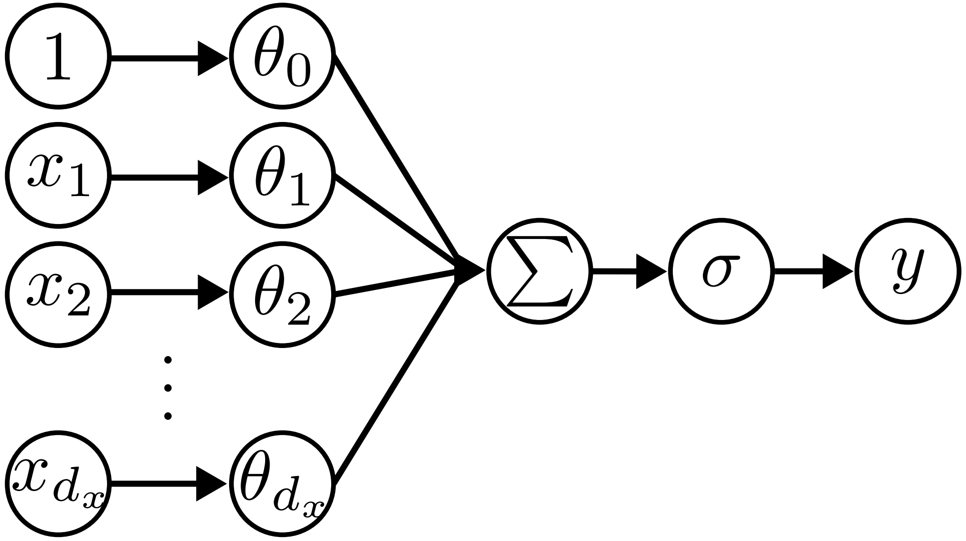


Artificial Neurons



In machine learning, we represent functions

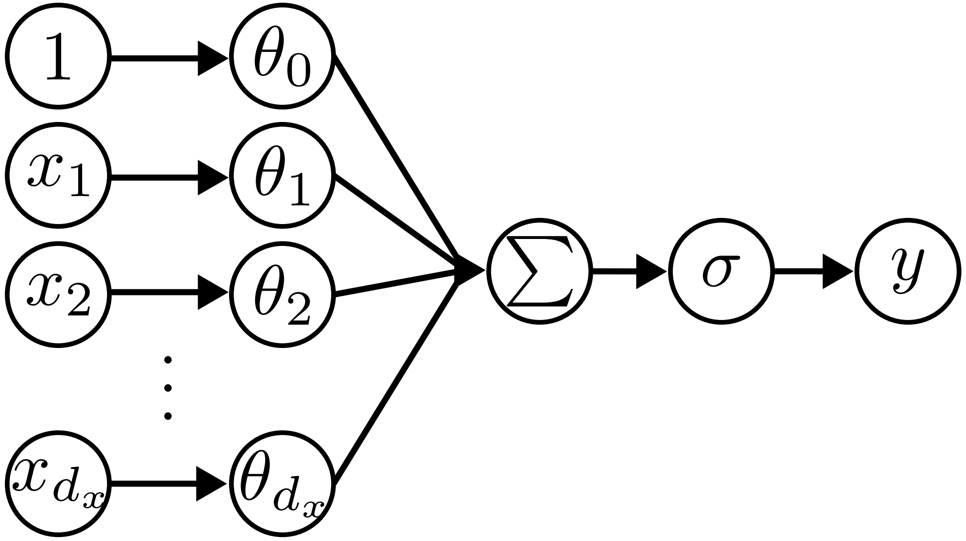
Artificial Neurons



In machine learning, we represent functions

What kinds of functions can our neuron represent?

Artificial Neurons

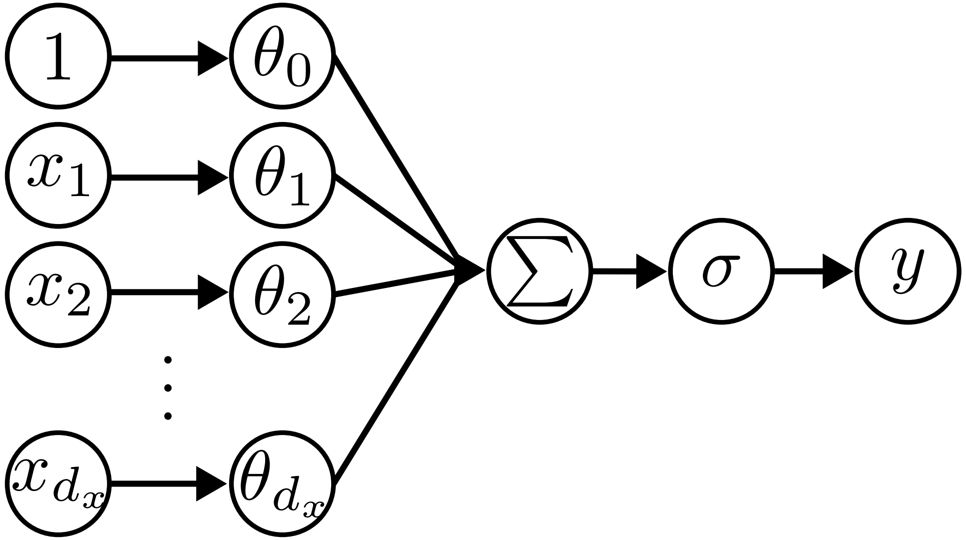


In machine learning, we represent functions

What kinds of functions can our neuron represent?

Let us consider some **boolean** functions

Artificial Neurons



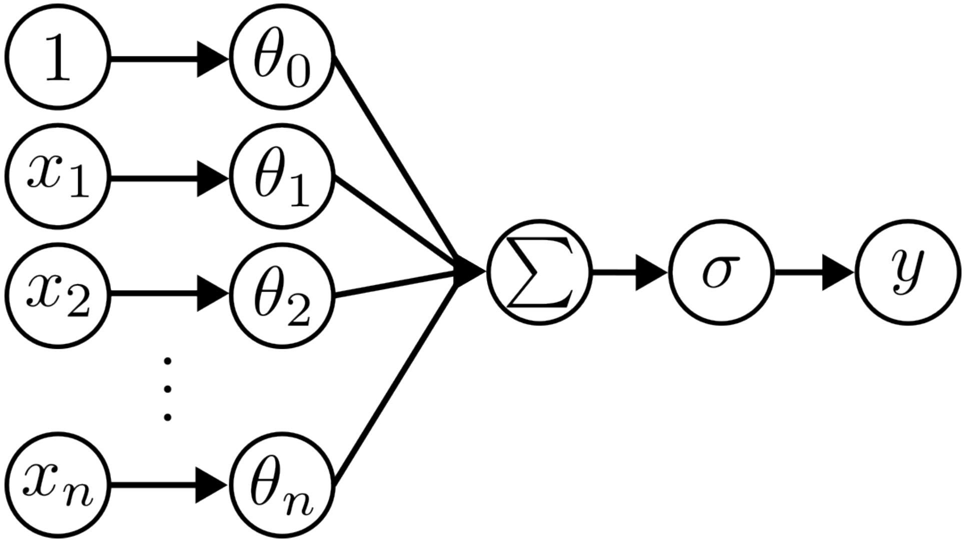
In machine learning, we represent functions

What kinds of functions can our neuron represent?

Let us consider some **boolean** functions

Let us start with a logical AND function

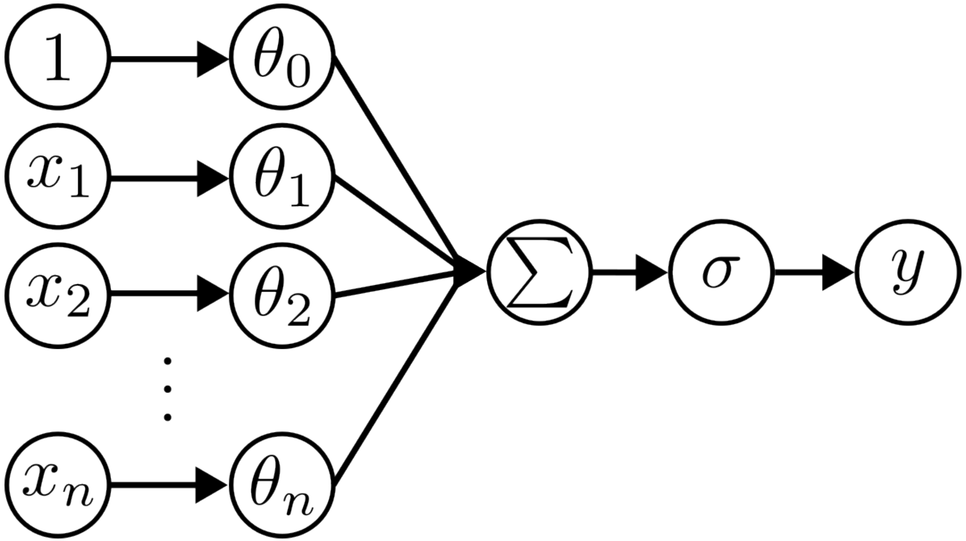
Artificial Neurons



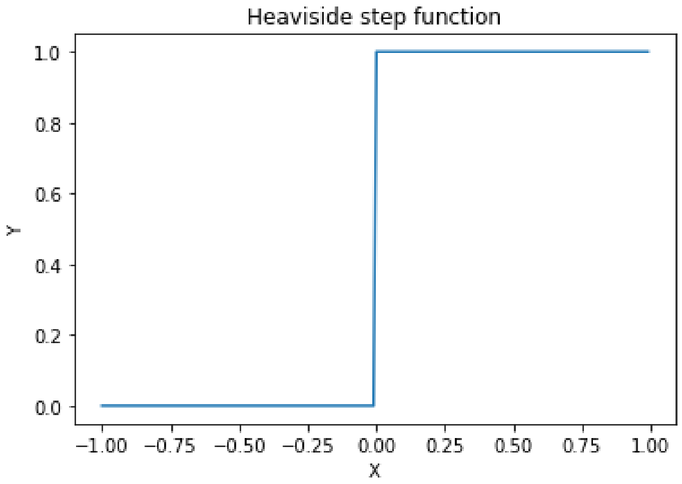
Review: Activation function

(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

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

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

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

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

$$\boldsymbol{\theta} = [\theta_0 \;\; \theta_1 \;\; \theta_2]^\top = [0 \;\; 1 \;\; 1]^\top$$

Artificial Neurons

Implement OR using an artificial neuron

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

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

x_1	x_2	y	$f(x_1, x_2, \boldsymbol{\theta})$	$\hat{y}$
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

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

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

Artificial Neurons

Implement XOR using an artificial neuron

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

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

x_1	x_2	y	$f(x_1, x_2, \boldsymbol{\theta})$	$\hat{y}$
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?

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

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 $\sigma(\text{linear function})$

Artificial Neurons

Why can't we represent XOR using a neuron?

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

We can only represent $\sigma(\text{linear function})$

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 $\sigma(\text{linear function})$

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 $\sigma(\text{linear function})$

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 $\rightarrow$ Biological neural network

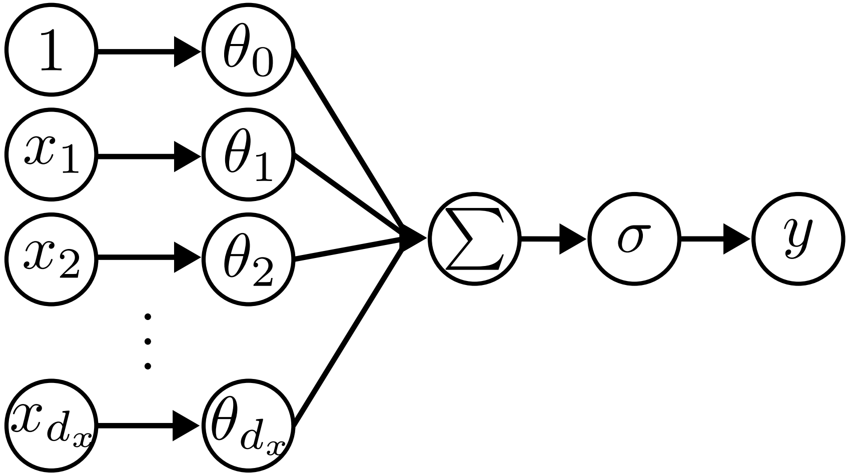
Artificial Neurons

Brain: Biological neurons $\rightarrow$ Biological neural network

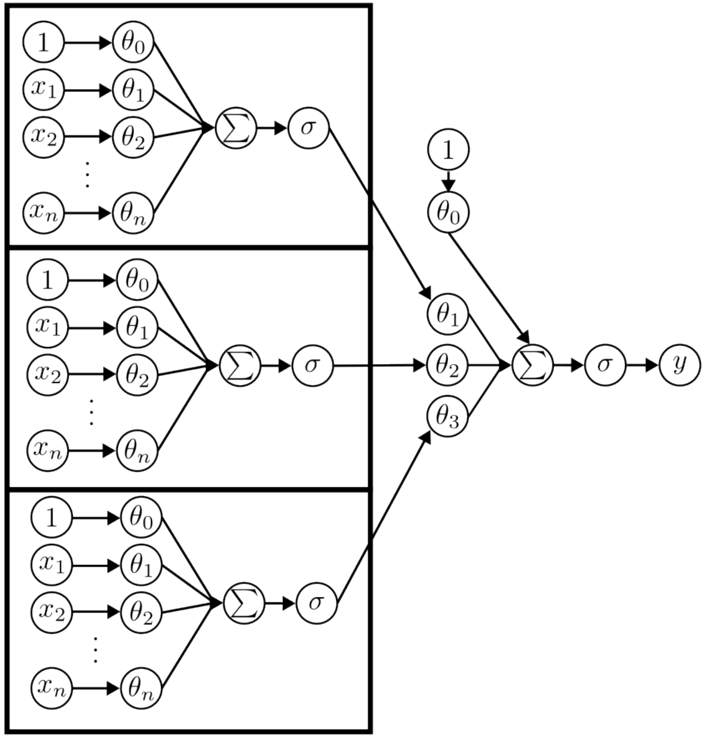
Computer: Artificial neurons $\rightarrow$ Artificial neural network

Artificial Neurons

Connect artificial neurons into a network

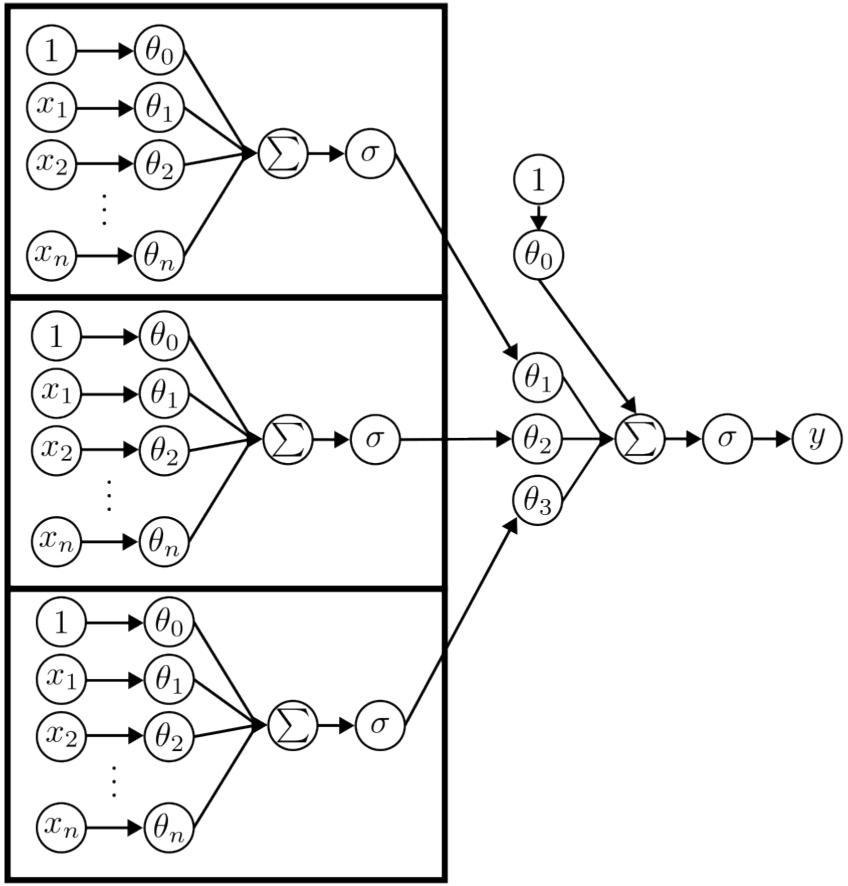


Neuron



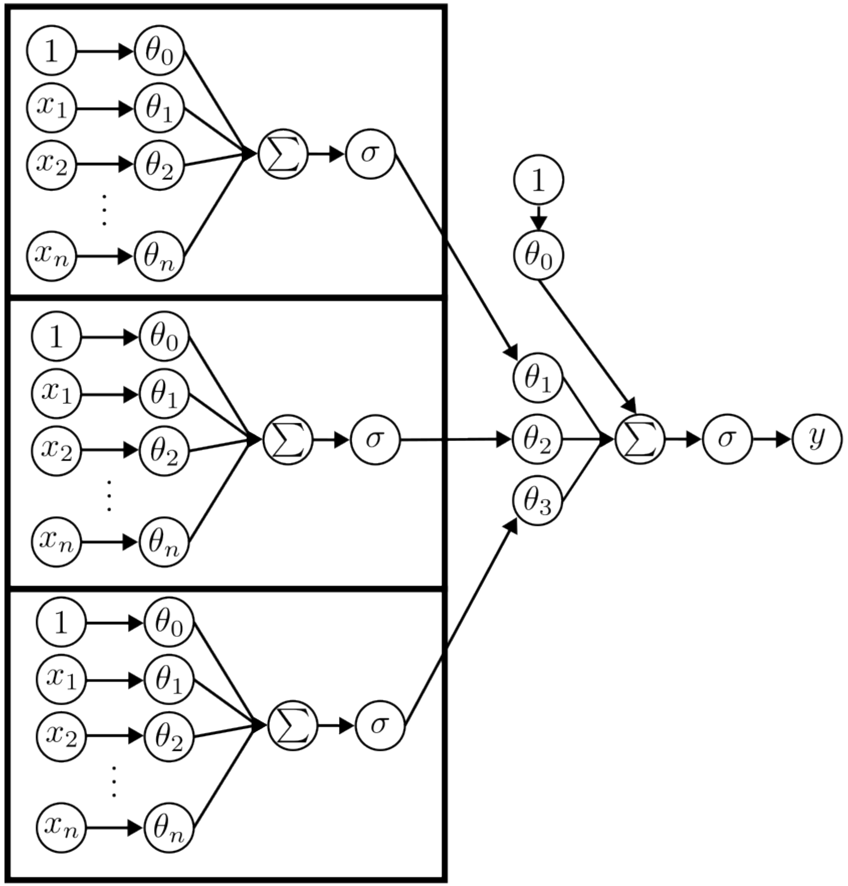
Neural Network

Artificial Neurons



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

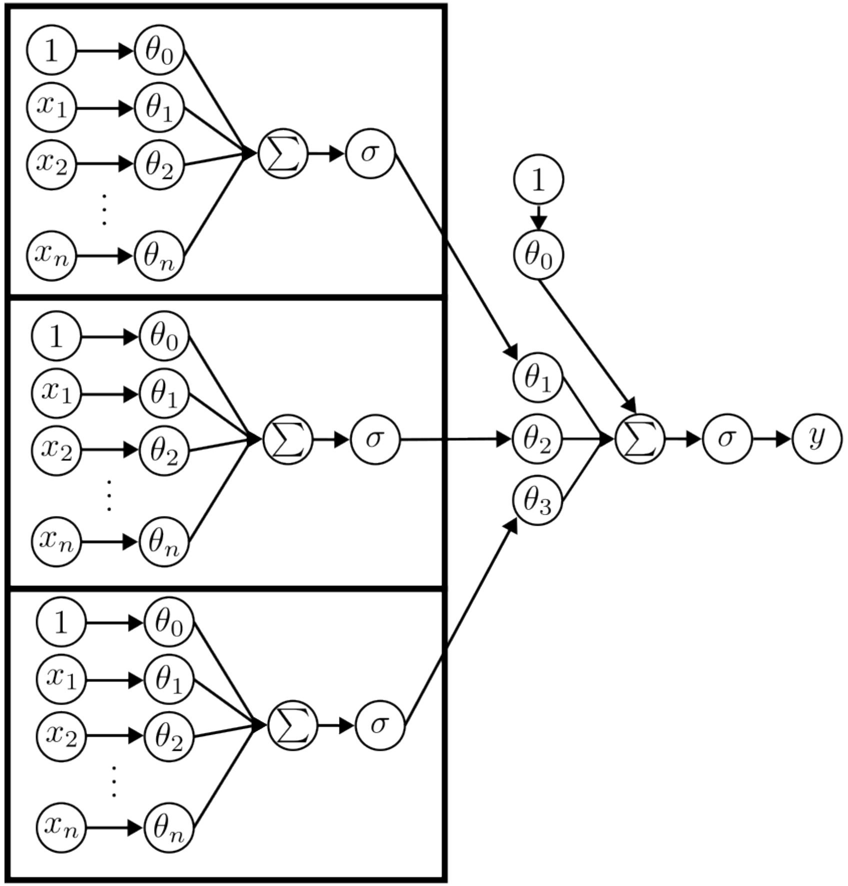
Artificial Neurons



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

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

Artificial Neurons

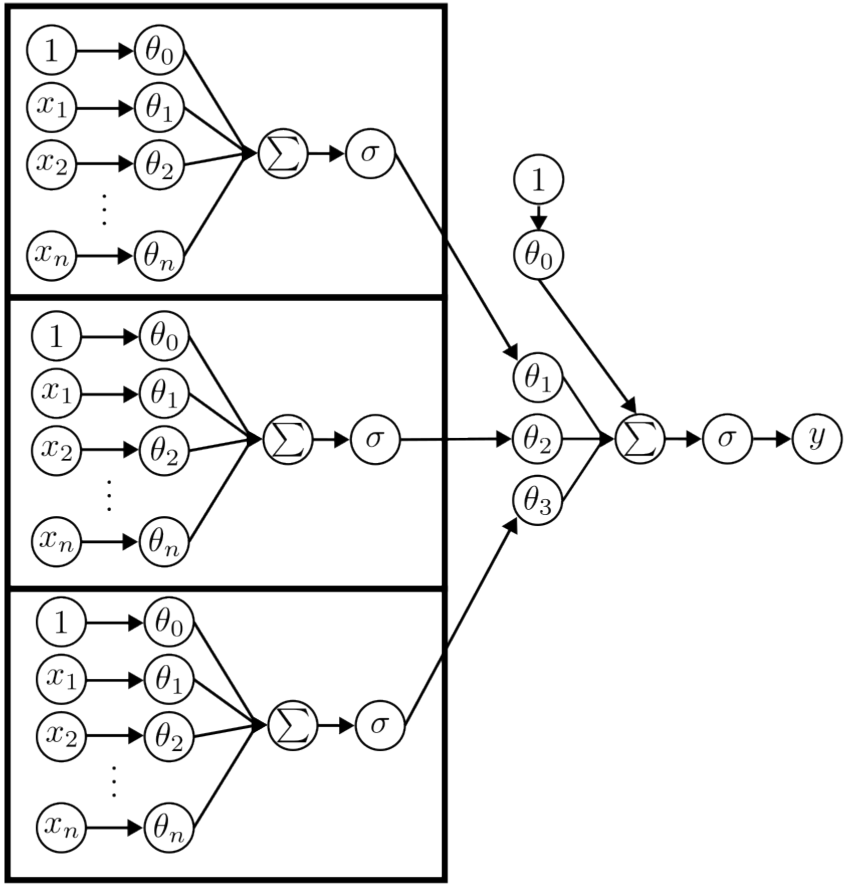


Adding neurons in **parallel** creates a **wide** neural network

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:

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

Wide Neural Networks

For a single neuron:

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

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

For a wide network:

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

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

For a wide network:

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

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

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

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

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

$$\boldsymbol{\theta} = \begin{bmatrix} \boldsymbol{\theta}_1 & \boldsymbol{\theta}_2 & \ldots & \boldsymbol{\theta}_\ell \end{bmatrix}^\top = \begin{bmatrix} \boldsymbol{\varphi} & \boldsymbol{\psi} & \ldots & \boldsymbol{\xi} \end{bmatrix}^\top$$

Deep Neural Networks

A wide network:

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

Deep Neural Networks

A wide network:

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

A deep network has many internal functions

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

Deep Neural Networks

A wide network:

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

A deep network has many internal functions

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

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

Deep Neural Networks

Written another way

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

Deep Neural Networks

Written another way

$$\boldsymbol{z}_1 = f_1(\boldsymbol{x}, \boldsymbol{\varphi}) = \boldsymbol{\varphi}^\top \overline{\boldsymbol{x}}$$

$$\boldsymbol{z}_2 = f_2(\boldsymbol{z}_{\mathbf{1}}, \boldsymbol{\psi}) = \boldsymbol{\psi}^\top \overline{\boldsymbol{z}}_1$$

Deep Neural Networks

Written another way

$$\boldsymbol{z}_1 = f_1(\boldsymbol{x}, \boldsymbol{\varphi}) = \boldsymbol{\varphi}^\top \overline{\boldsymbol{x}}$$

$$\boldsymbol{z}_2 = f_2(\boldsymbol{z_1}, \boldsymbol{\psi}) = \boldsymbol{\psi}^\top \overline{\boldsymbol{z}}_1$$

$$\vdots$$

Deep Neural Networks

Written another way

$$\boldsymbol{z}_1 = f_1(\boldsymbol{x}, \boldsymbol{\varphi}) = \boldsymbol{\varphi}^\top \overline{\boldsymbol{x}}$$

$$\boldsymbol{z}_2 = f_2(\boldsymbol{z}_{\mathbf{1}}, \boldsymbol{\psi}) = \boldsymbol{\psi}^\top \overline{\boldsymbol{z}}_1$$

$$\vdots$$

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

We call each function a **layer**

Deep Neural Networks

Written another way

$$\boldsymbol{z}_1 = f_1(\boldsymbol{x}, \boldsymbol{\varphi}) = \boldsymbol{\varphi}^\top \overline{\boldsymbol{x}}$$

$$\boldsymbol{z}_2 = f_2(\boldsymbol{z}_{\mathbf{1}}, \boldsymbol{\psi}) = \boldsymbol{\psi}^\top \overline{\boldsymbol{z}}_1$$

$$\vdots$$

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

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

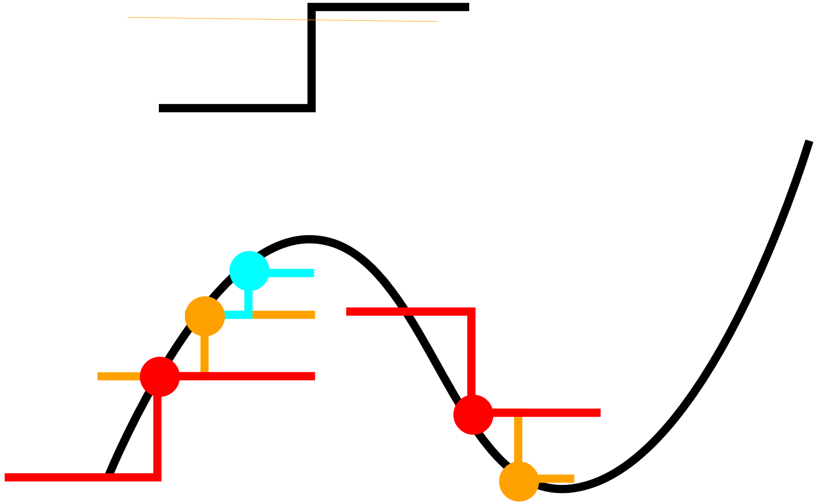
What functions can we represent using a deep neural network?

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?

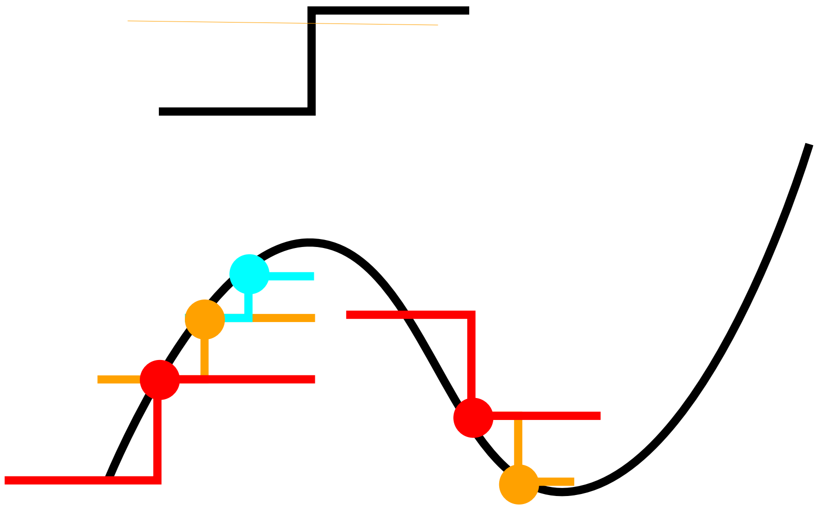
Proof Sketch: Approximate a continuous function $g : \mathbb{R} \mapsto \mathbb{R}$ using a linear combination of Heaviside functions



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



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

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 ε

Deep Neural Networks

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

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

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

Deep Neural Networks

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

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

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

Making the network deeper or wider decreases ε

Deep Neural Networks

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

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

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

Making the network deeper or wider decreases ε

Very powerful finding! The basis of deep learning.

Deep Neural Networks

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

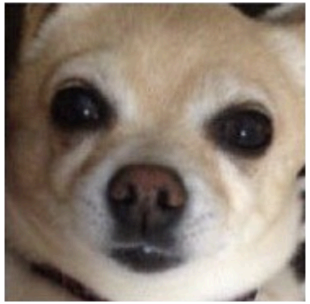
It can approximate **any** continuous function $g(x)$ to precision ε

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

Making the network deeper or wider decreases ε

<u>Very powerful finding! The basis of deep learning.</u>

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

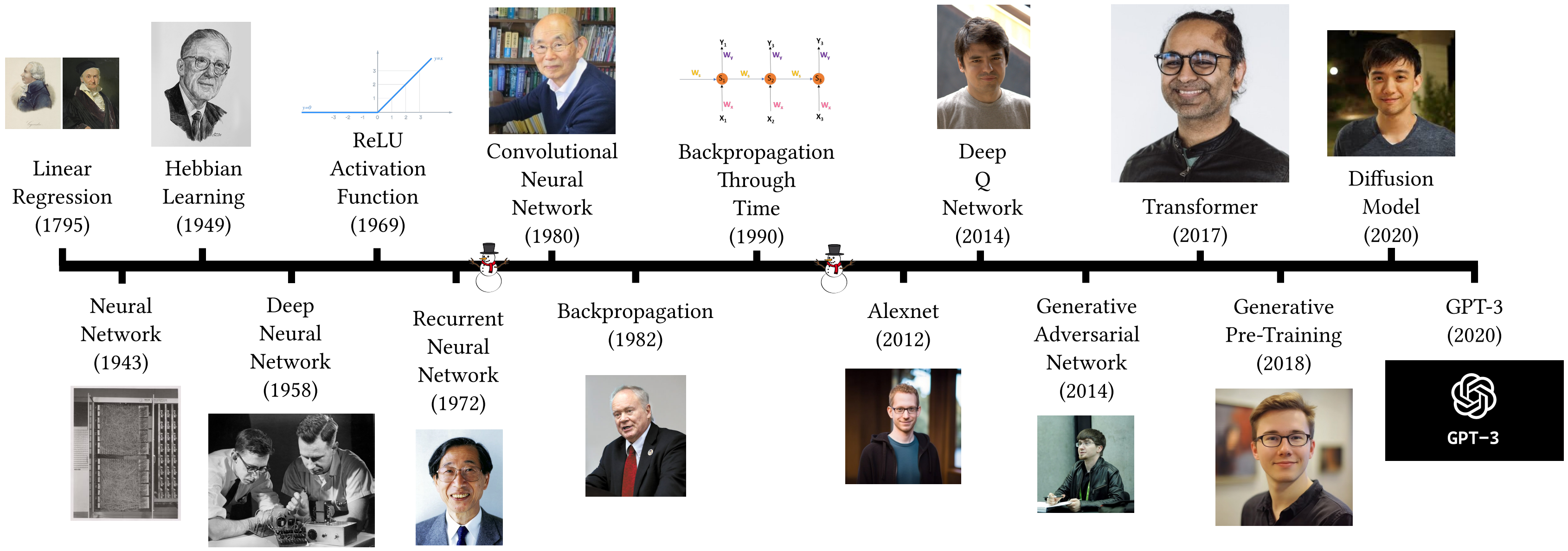
We call wide neural networks **perceptrons**

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

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

Many layers makes a deep neural network

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

$$\boldsymbol{z}_2 = f\left(\boldsymbol{z}_1, \begin{bmatrix} \boldsymbol{b}_2 \\ \boldsymbol{W}_2 \end{bmatrix}\right)$$

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

Practical Considerations

Let us create a wide neural network in colab! 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