Machine Learning Basics

COMP9312_24T2

UNSW SYDNEY

About this topic

- Introduce basic knowledge about machine learning
- You need them to understand graph neural networks
- Concepts in this topic would not be in assignments/exam

Machine Learning ≈ Looking for Function

Speech Recognition

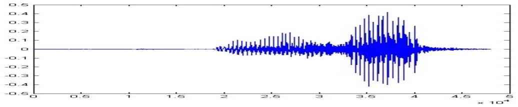
$f($  $) =$ "Hello World"

Image Recognition

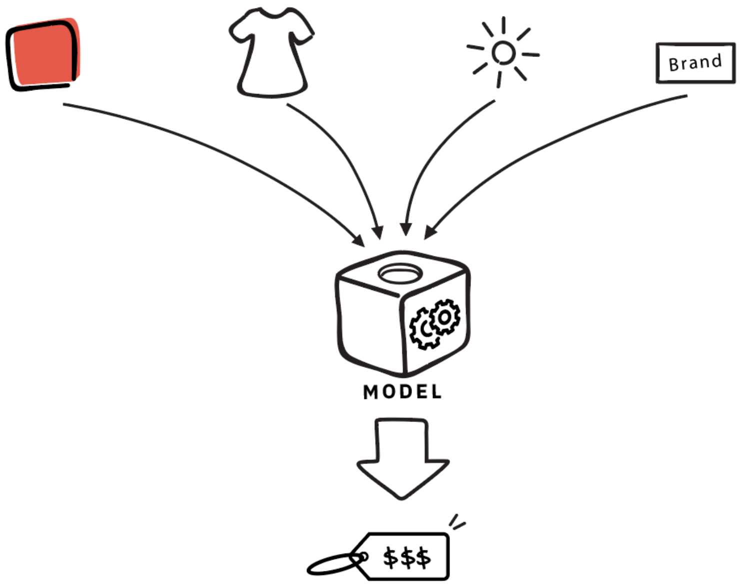
$f($  $) =$ "Cat"

ChatGPT

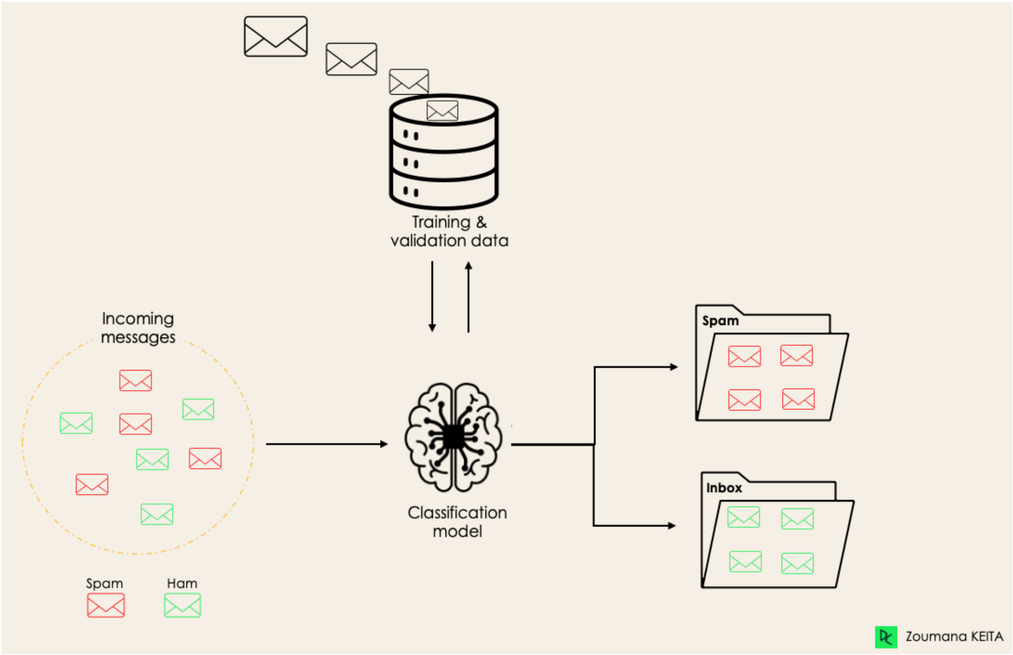
$f($ write a solution for my assignment 1 $) =$ "..."

Two types of ML function

Regression

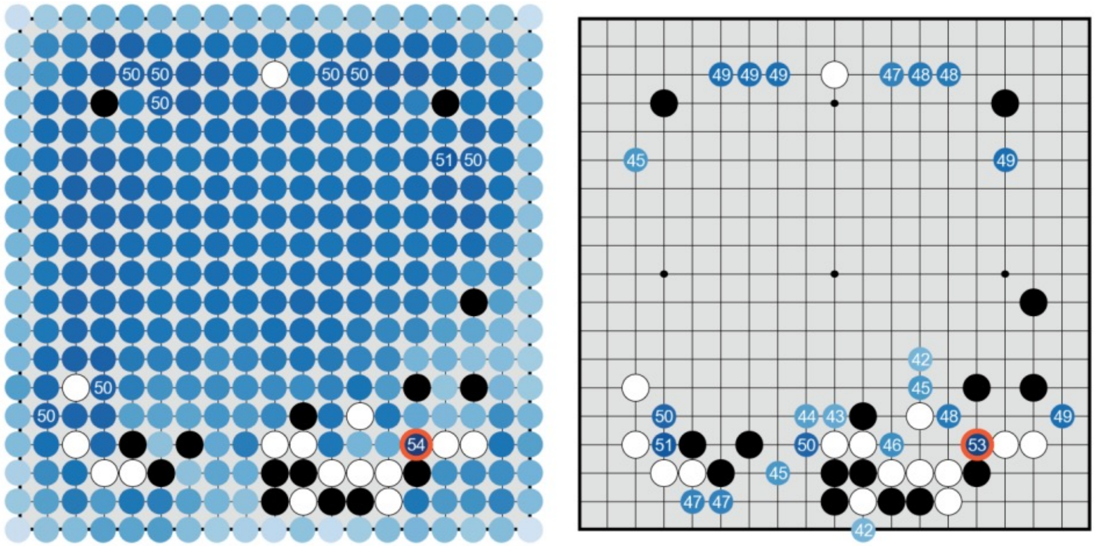


Classification



Some Classification Tasks

AlphaGo



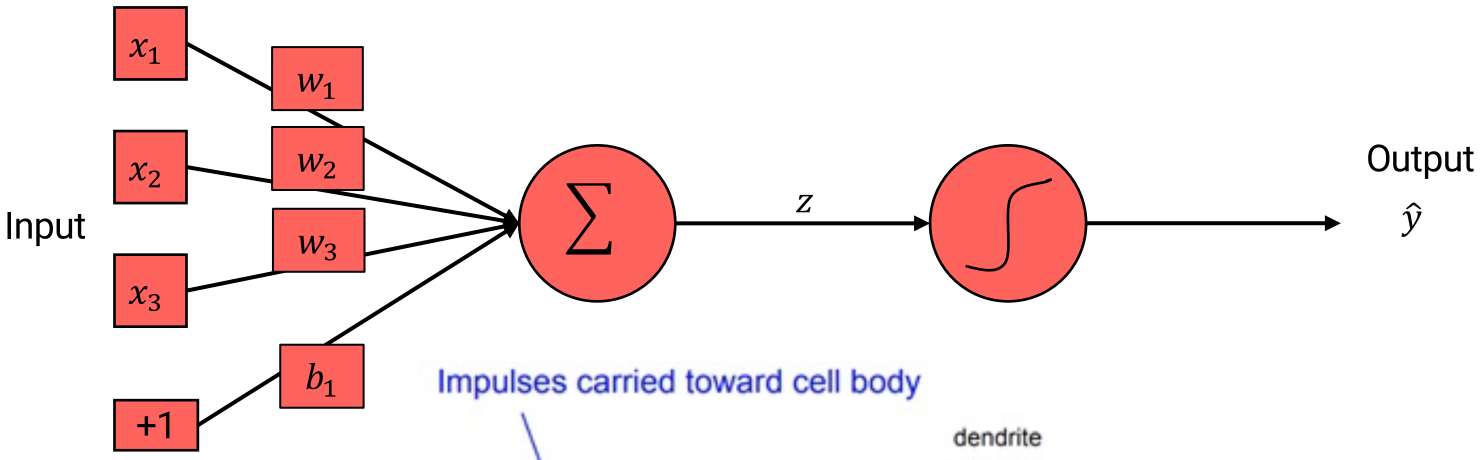
ChatGPT

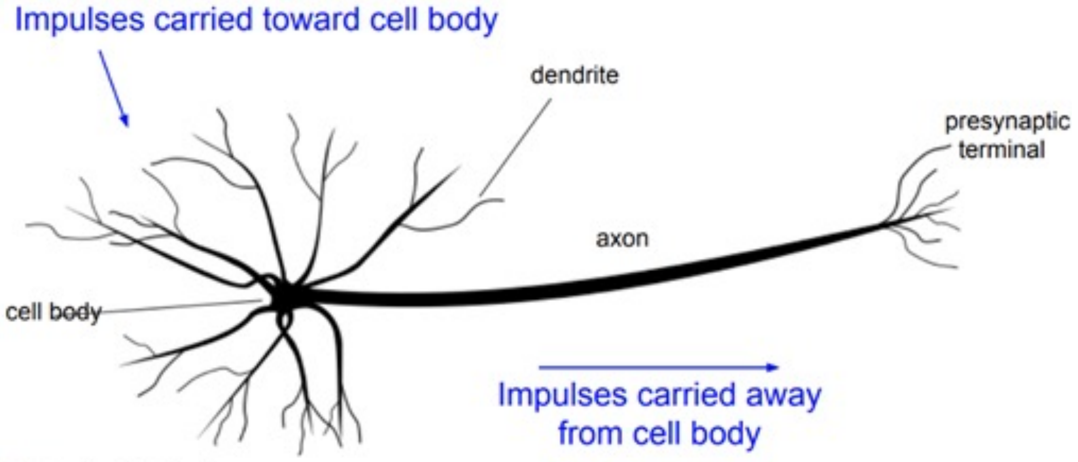
Machine Learning

- Algorithms that improve automatically through experience.

- The algorithm has a (large) number of parameters whose values need to be learned from the data.

Neurons and Perceptron

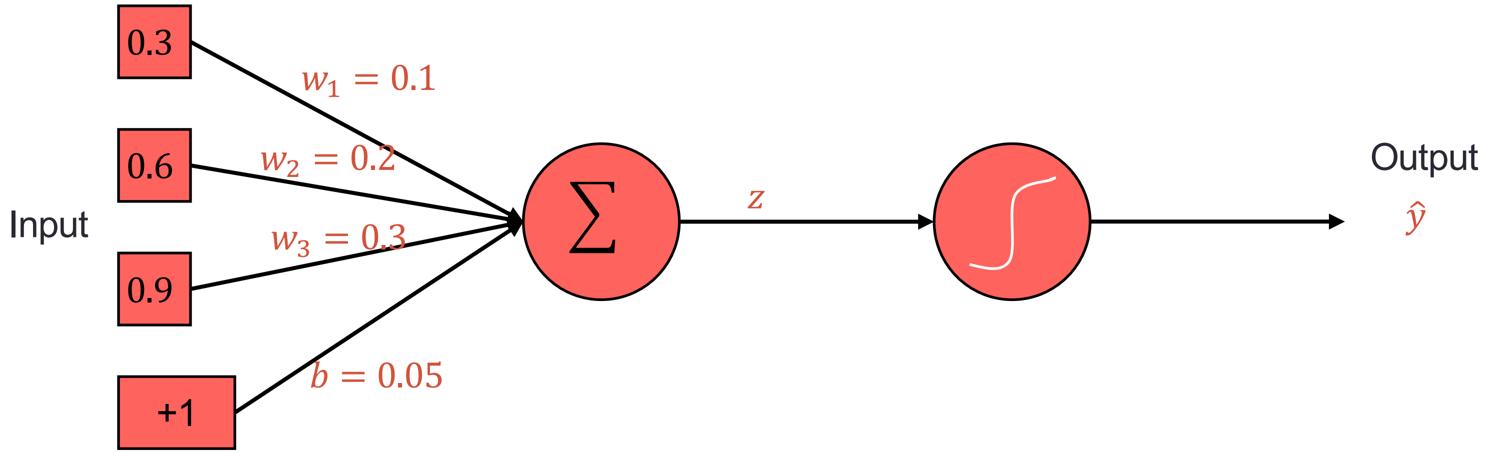




What does a Perceptron do? (1)

Suppose a NN initialized to weight w be (0.1, 0.2, 0.3) & bias $b = 0.05$

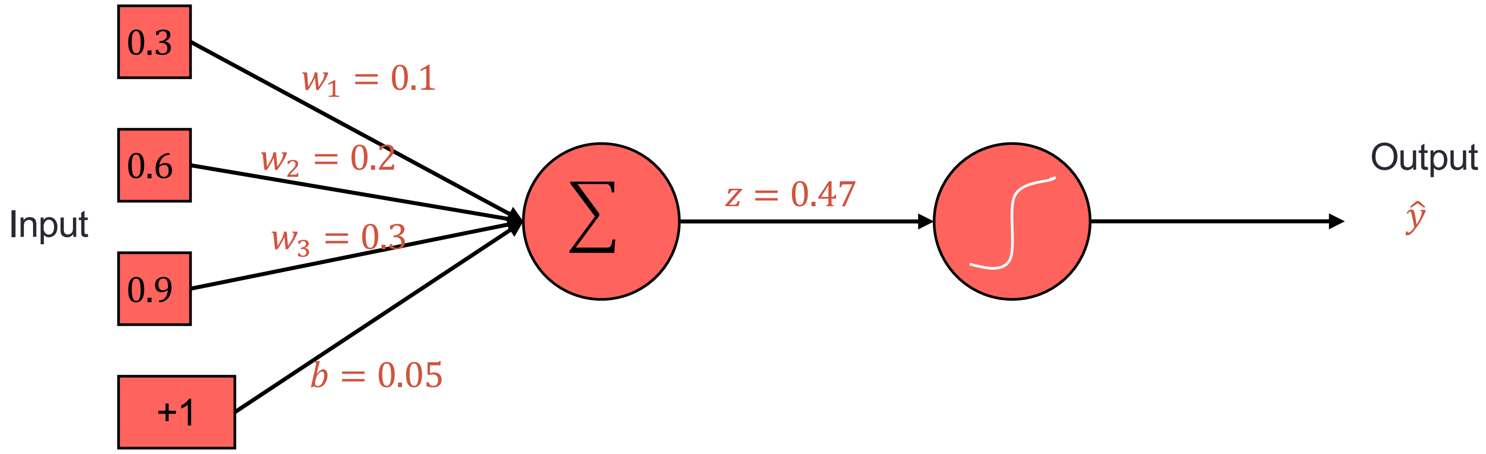
Step0: Take an input x (0.3, 0.6, 0.9)



What does a Perceptron do? (2)

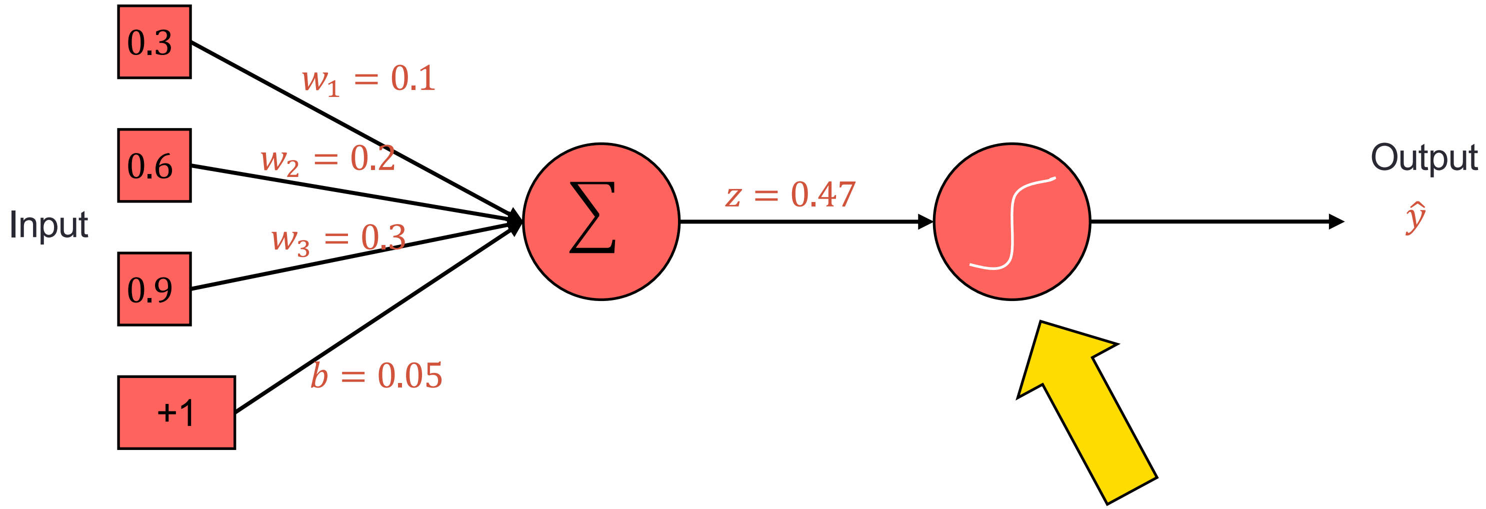
Step1: Calculate a weighted sum

$$z = w^T x + b;\ z = 0.1 \times 0.3 + 0.2 \times 0.6 + 0.3 \times 0.9 + 0.05 = 0.47$$



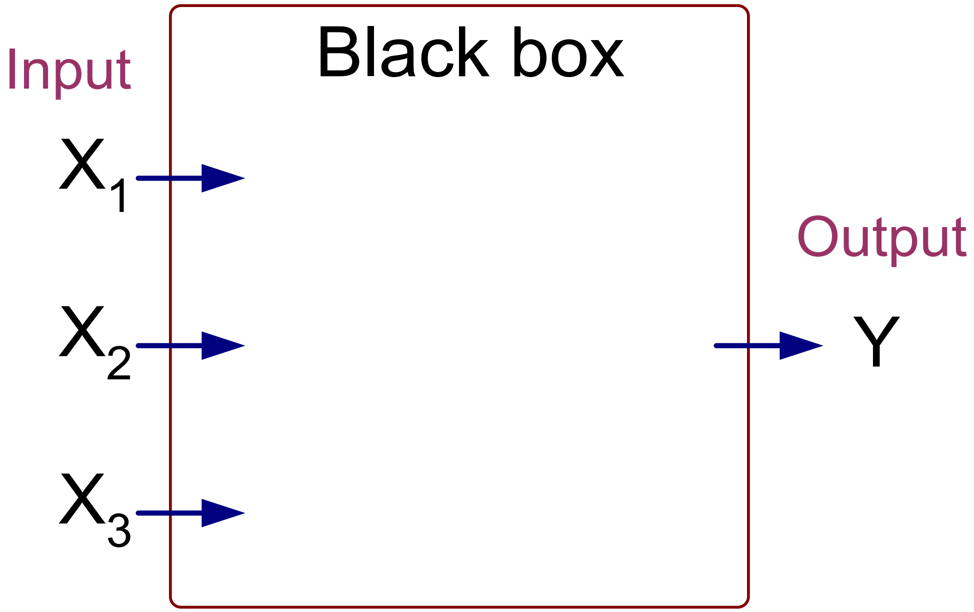
What does a Perceptron do? (3)

Step2: Apply an activation function



Neural Networks

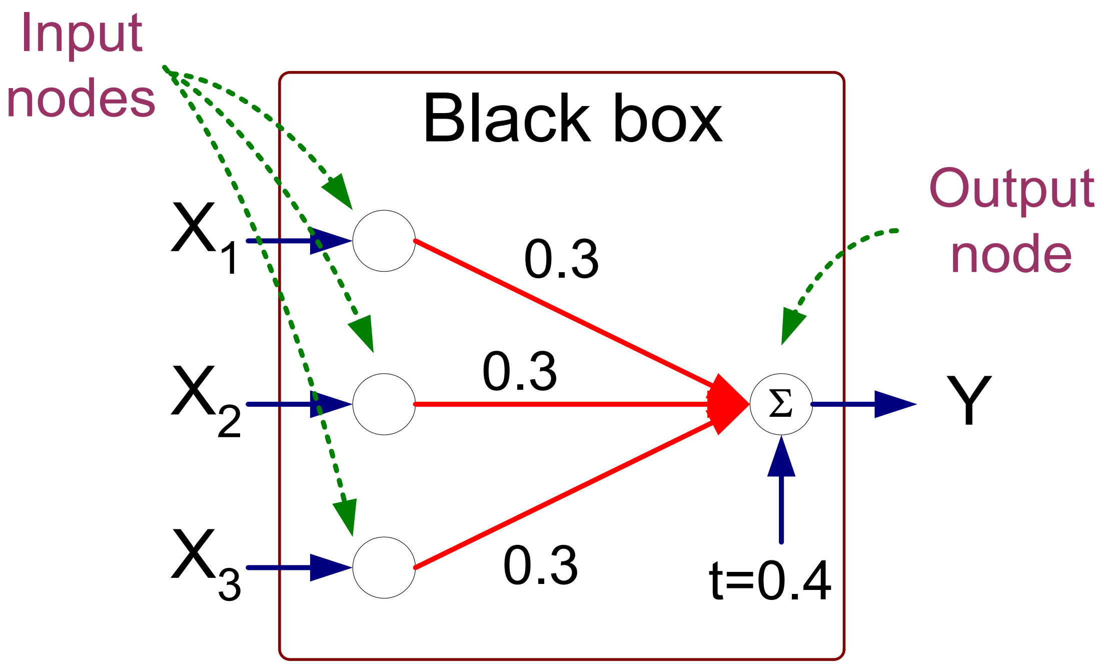
X_1	X_2	X_3	Y
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1
0	0	1	0
0	1	0	0
0	1	1	1
0	0	0	0



Output Y is 1 if at least two of the three inputs are equal to 1.

Neural Networks (cont)

X_1	X_2	X_3	Y
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1
0	0	1	0
0	1	0	0
0	1	1	1
0	0	0	0

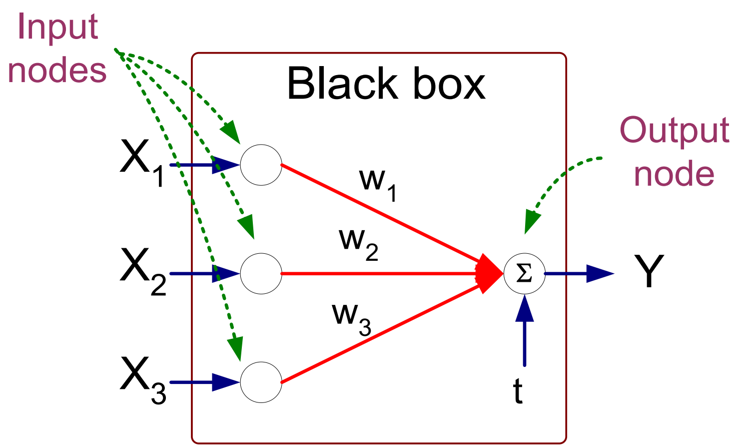


$$Y = I(0.3X_1 + 0.3X_2 + 0.3X_3 - 0.4 > 0)$$

$$\text{where } I(z) = \begin{cases} 1 & \text{if } z \text{ is true} \\ 0 & \text{otherwise} \end{cases}$$

Neural Networks (cont)

- Model is an assembly of inter-connected nodes and weighted links
- Output node sums up each of its input value according to the weights of its links
- Compare output node against some threshold t
- The sign function (activation function) outputs a value +1 if its argument is positive and -1 otherwise.



Perceptron Model

$$Y = I(\sum_i w_i X_i - t) \quad \text{or}$$

$$Y = sign(\sum_i w_i X_i - t)$$

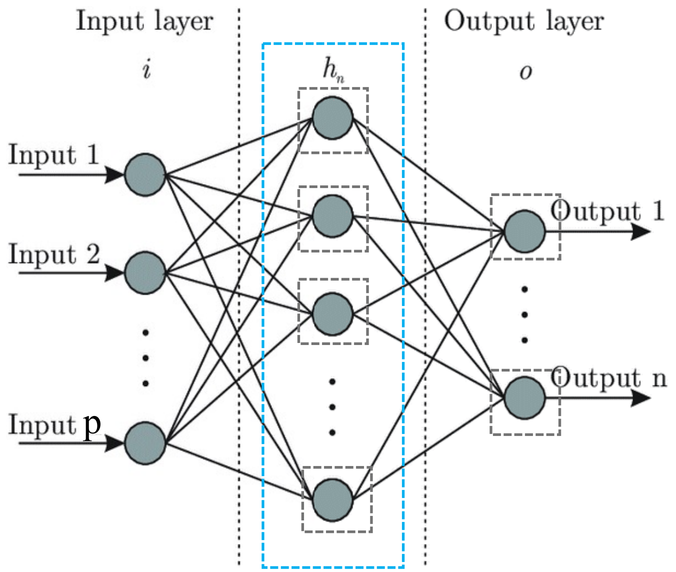
Increase Expressive Power

From Perceptrons to NN

- Perceptrons are a basic unit of a neural network.
- 1-layered neural network on the right

Structure:

- Input layer, output layer,
- Middle are hidden layers.



A Case Study

What is the final mark of a student in 9312? (a regression problem)

$$f(\text{student's mark for COMP9024}) = \;??$$

Where does the machine learn from?

Marks of many previous students for 9024 and 9312 (Supervised Learning)

Other types: unsupervised learning (NLP), semi-supervised learning, …

How to get the function

1. Tell the machine what to learn (Parameters)

2. Tell the machine how to evaluate the function (Loss Function)

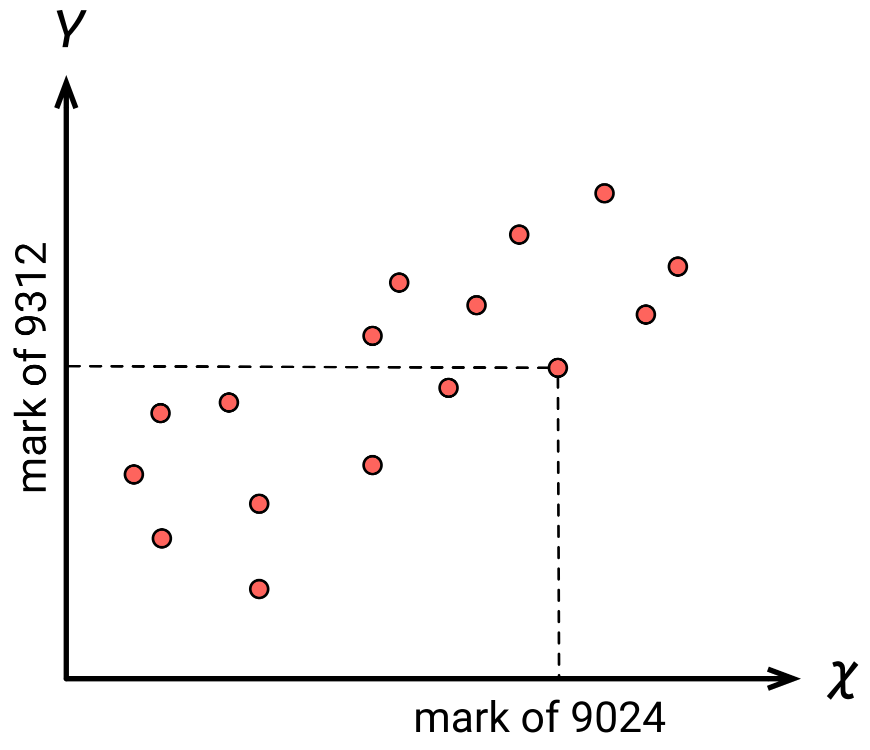
3. Wait … (Training)

Step 1 - Parameters

1. Tell the machine what to learn (Parameters)

A linear function based on domain knowledge.

$$y = \underbrace{b}_{\text{bias}} + \overbrace{w}^{\text{weight}}\underbrace{x_1}_{\text{feature}}$$



training data

w and ***b*** are unknown parameters to learn.

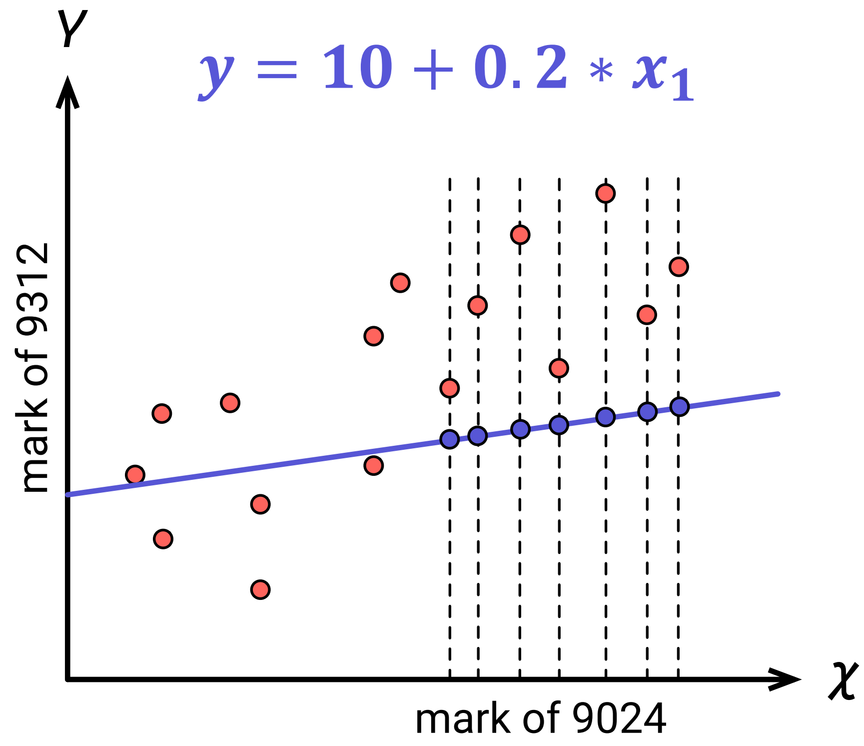
Step 2 – Loss Function

2. Tell the machine how to evaluate the function (Loss Function)

$\hat{y}$ is the label (real value)

y is the estimation

How good is a value / a function?



Loss: $L = \frac{1}{N}\sum_{n} e_n$

$e = |y - \hat{y}|$ L is mean absolute error (MAE)

$e = (y - \hat{y})^2$ L is mean square error (MSE)

Step 2 – Loss Function

2. Tell the machine how to evaluate the function (Loss Function)

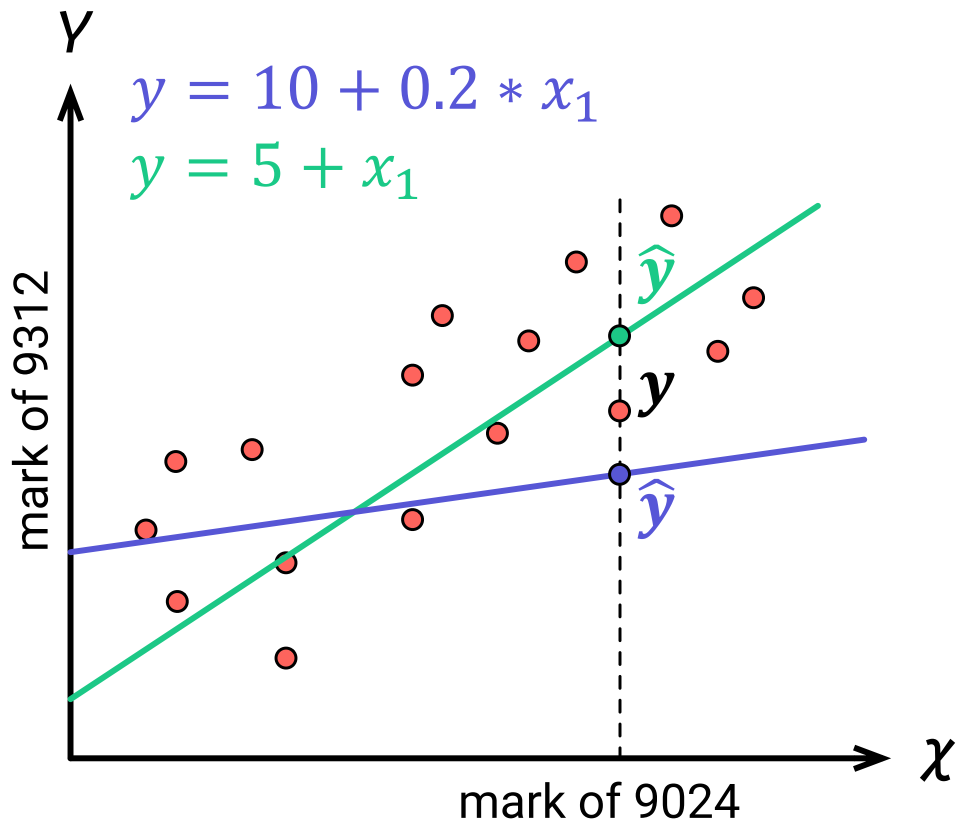
$$y = b + wx_1$$

We aim to find a good w and b
to minimize the loss function.

Loss: $L = \frac{1}{N}\sum_n e_n$

$e = |y - \hat{y}|$ L is mean absolute error (MAE)

$e = (y - \hat{y})^2$ L is mean square error (MSE)



Step 3 - Training

How to get good parameters?

$$y = b + wx_1$$

Gradient Descent.

Done by the toolkit (e.g., pytorch…).

Gradient Descent

Optional

$$y = b + wx_1 \qquad w^*, b^* = arg \min_{w,b} L$$

1. pick a random w^0
2. compute gradient $\frac{\partial L}{\partial w}|_{w=w^0}$

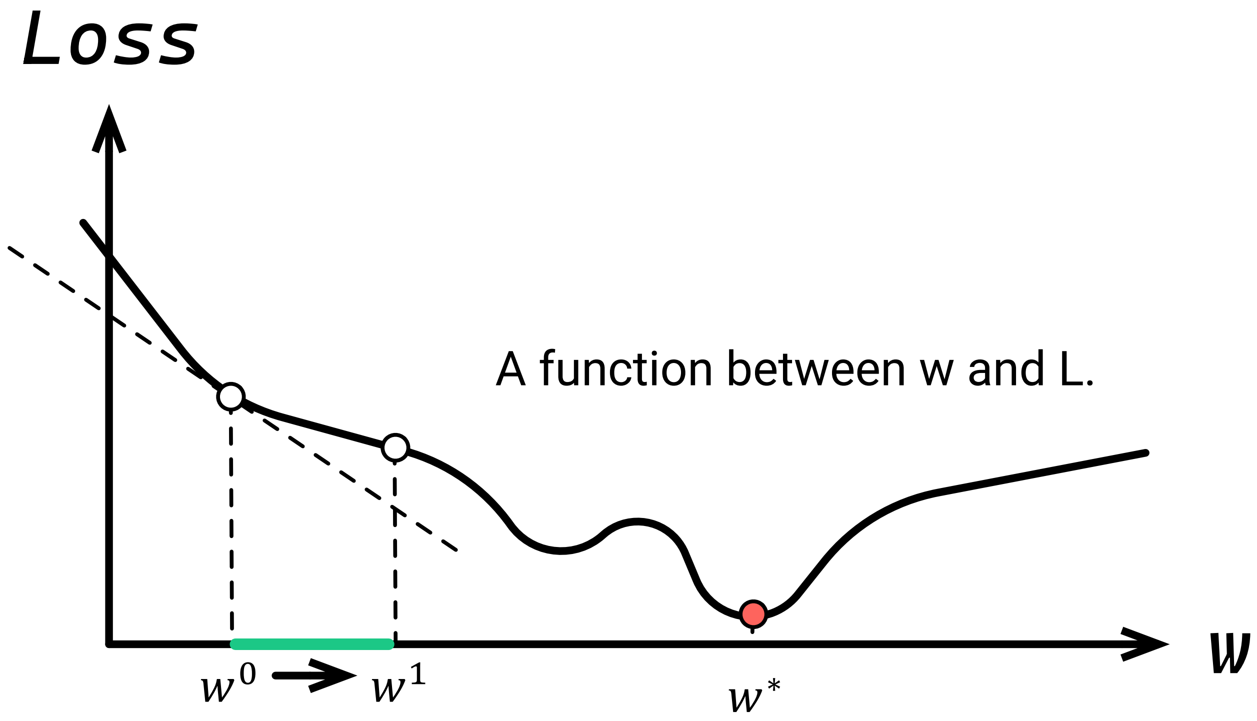
Negative -> increase w^0

Positive -> decrease w^0

3. Update w based on a hyperparameter η

$$w^1 \leftarrow w^0 - \eta \frac{\partial L}{\partial w}|_{w=w^0}$$

4. Update w iteratively.



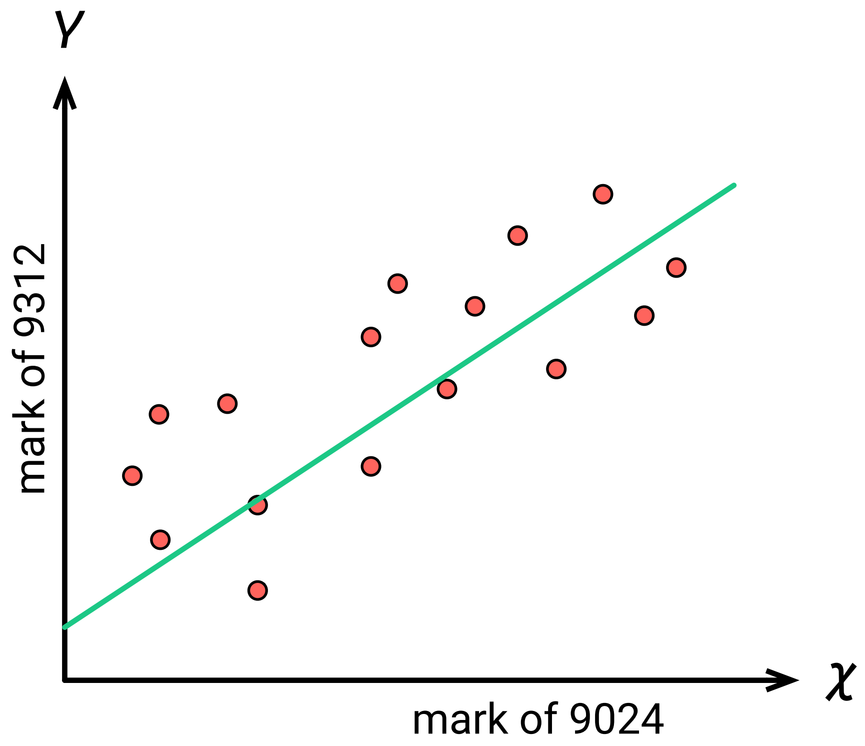
η is also called learning rate.

Evaluate the function

Now we have a good linear function to predict the mark.

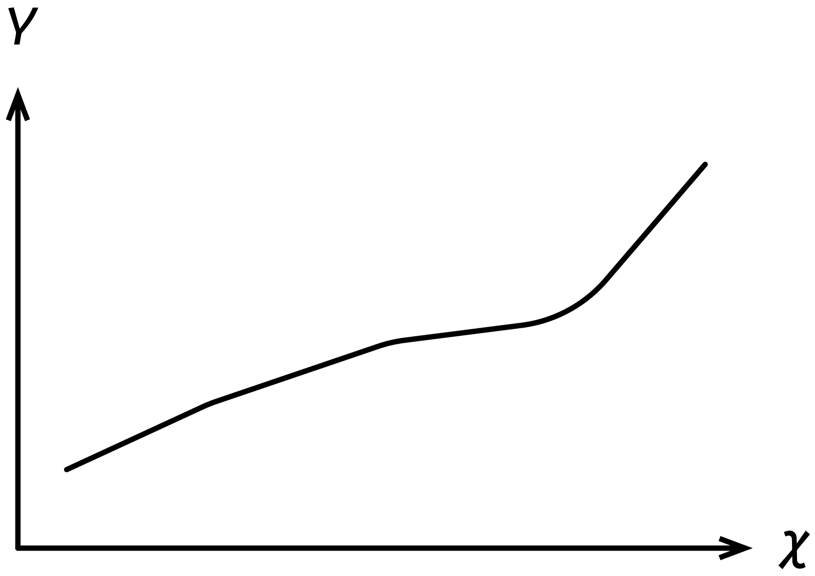
$f\left(\text{student's mark for COMP9024}\right) = \ ??$

Are linear models good enough?

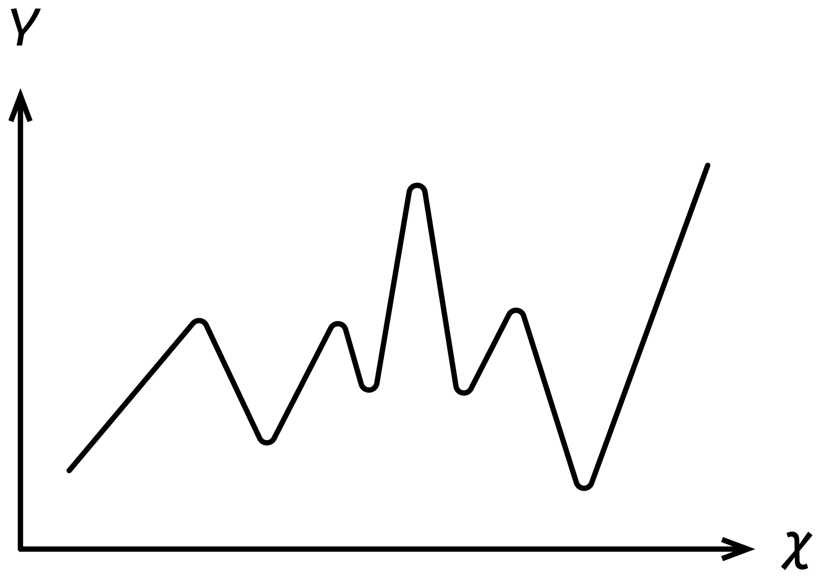


Beyond Linear Models

Real problems are much more sophisticated.



This is what you expect.



Real scenarios may be terrible …

How to get sophisticated functions

Combine simple functions in two ways:

$f_1(x)+f_2(x)$ $f_1(f_2(x))$

Combine linear functions? The result is still a linear function~

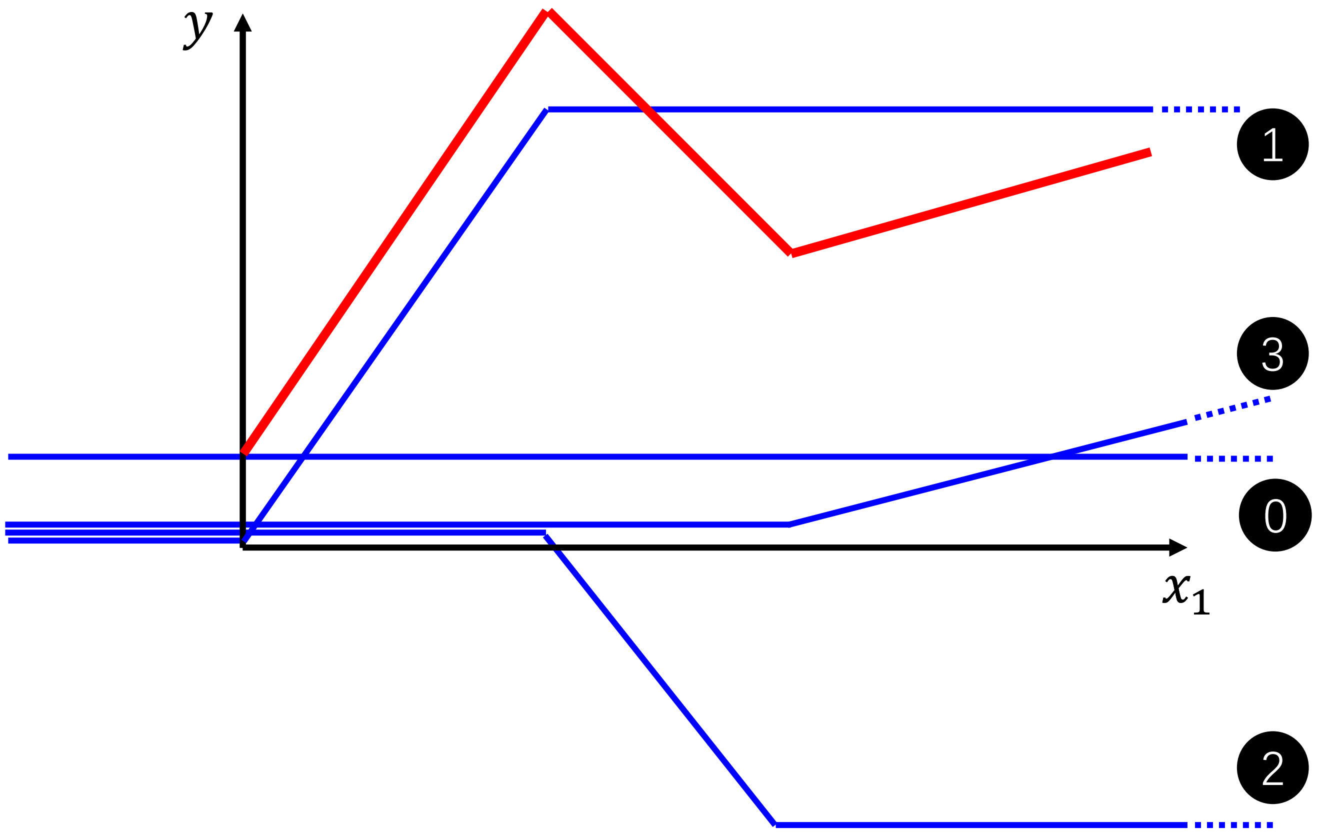
y=3x+1 y=5x+2 -> y=3(5x+2)+1=15x+7

Activation functions are required:

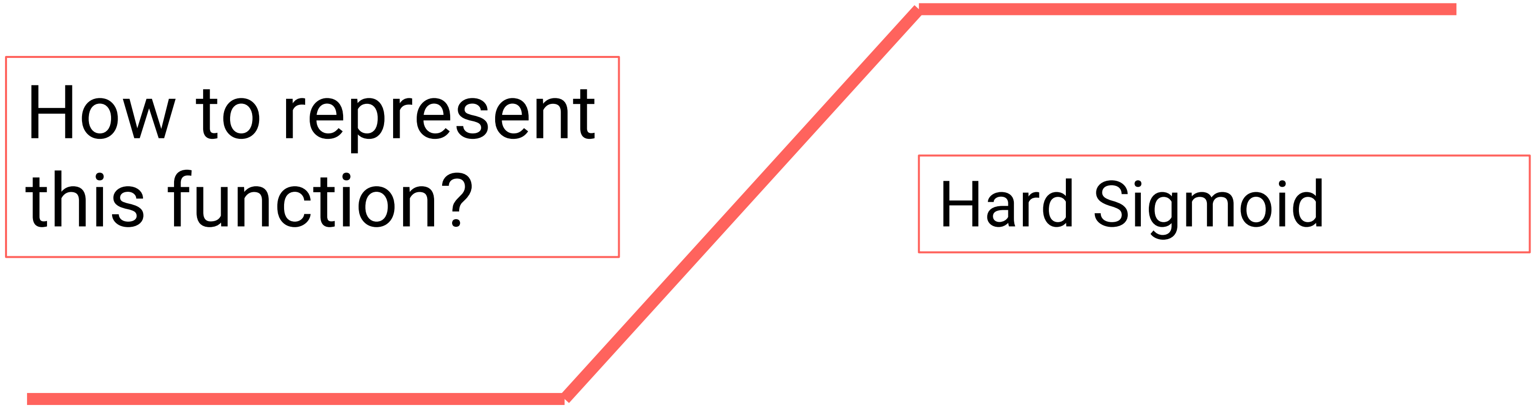
- Sigmoid
- Relu

Sigmoid Function

red curve = constant + sum of a set of



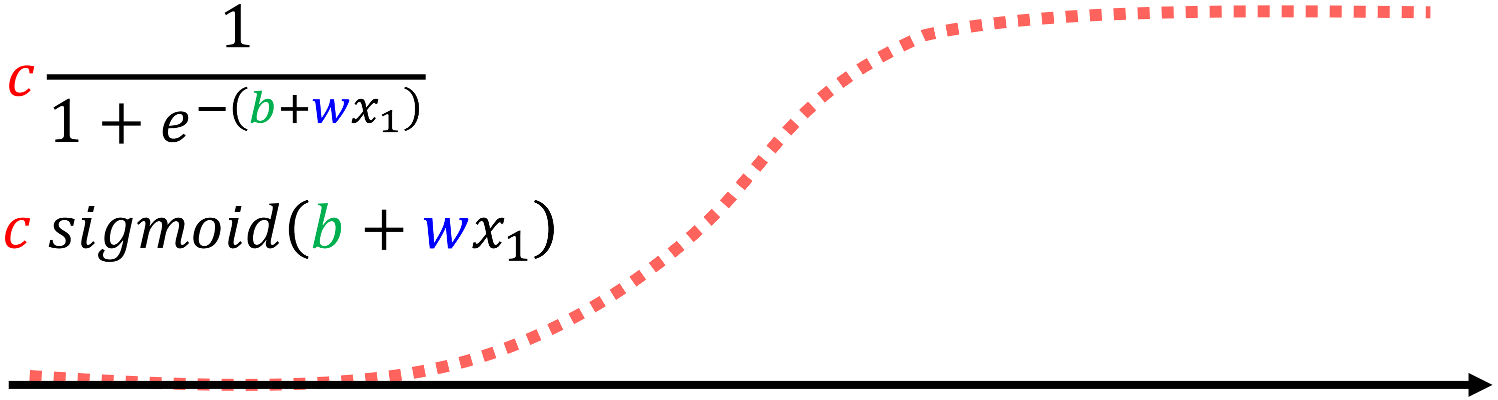
Sigmoid Function



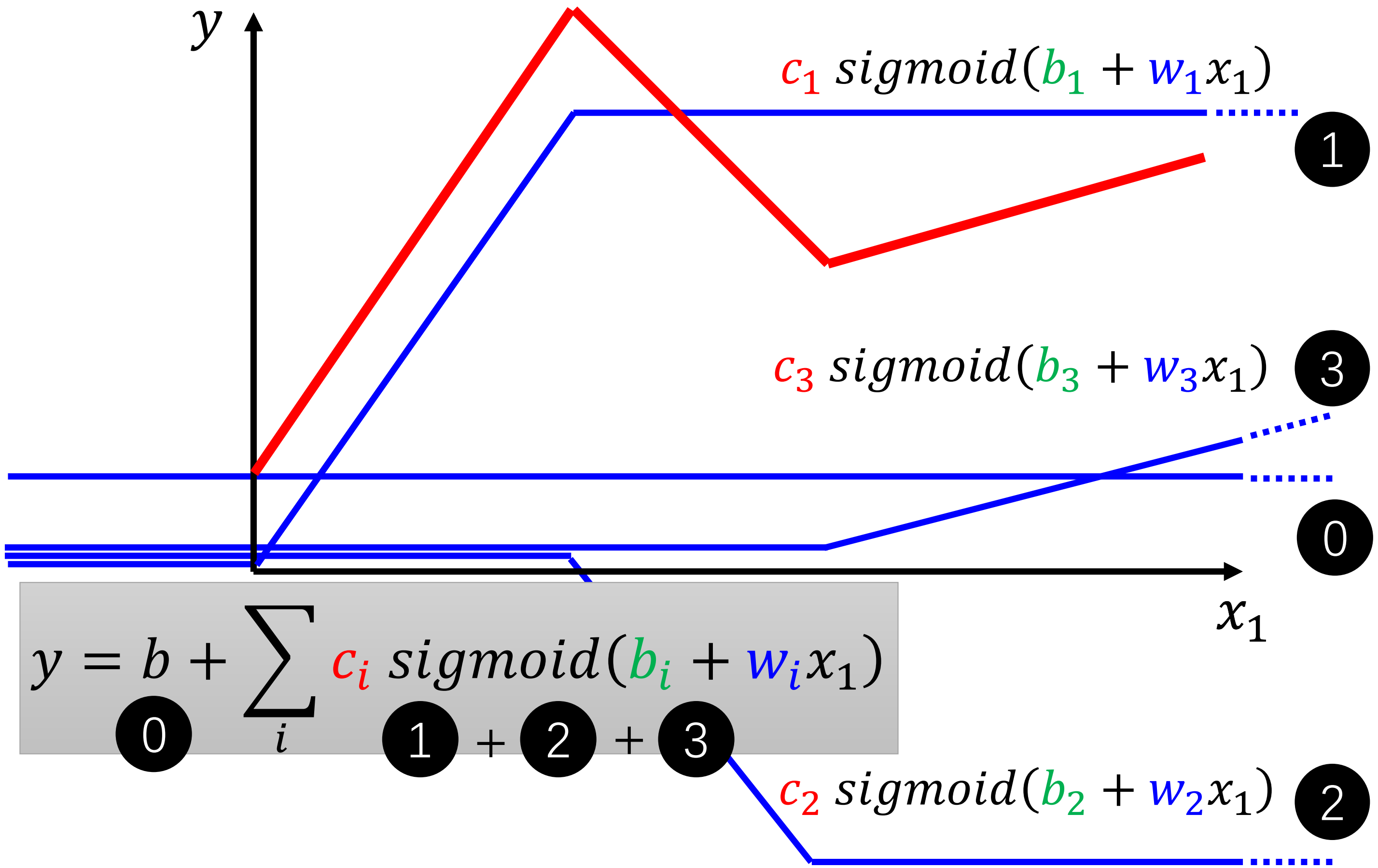
Sigmoid Function

$$y = c\frac{1}{1+e^{-(b+wx_1)}}$$

$$= c\ sigmoid(b + wx_1)$$



Combining Sigmoid Functions

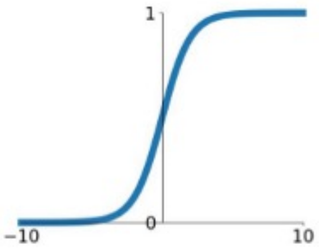


$$y = \underbrace{b}_{0} + \underbrace{\sum_i c_i\ sigmoid(b_i + w_i x_1)}_{1\ +\ 2\ +\ 3}$$

Other activation functions

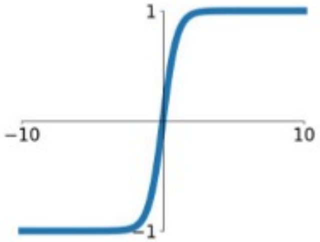
Sigmoid

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



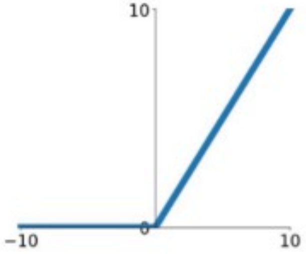
tanh

$\tanh(x)$



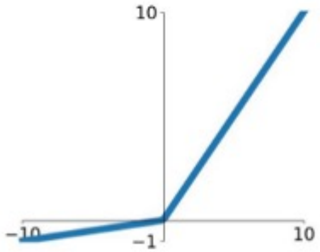
ReLU

$\max(0, x)$



Leaky ReLU

$\max(0.1x, x)$

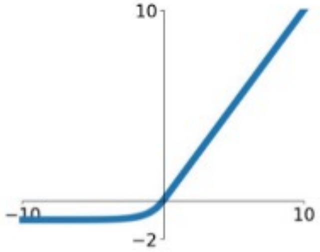


Maxout

$\max(w_1^T x + b_1, w_2^T x + b_2)$

ELU

$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$



Sigmoid and ReLU

$$y = b + \sum_{i} c_i \, \underline{sigmoid}\left(b_i + \sum_{j} w_{ij} x_j\right)$$

$$y = b + \sum_{2i} c_i \, \underline{max}\left(0, b_i + \sum_{j} w_{ij} x_j\right)$$

New Model

Combine i Sigmoid functions

$$y = \underline{b + wx_1} \quad \rightarrow \quad y = b + \sum_i c_i \, sigmoid(\underline{b_i + w_i x_1})$$

New Model

Combine i Sigmoid functions

$$y = \underline{b + wx_1} \rightarrow y = b + \sum_i c_i \; sigmoid(\underline{b_i + w_i x_1})$$

Combine j features

$$y = \underline{b + \sum_j w_j x_j} \rightarrow \boxed{y = b + \sum_i c_i \; sigmoid\left(\underline{b_i + \sum_j w_{ij} x_j}\right)}$$

New Model

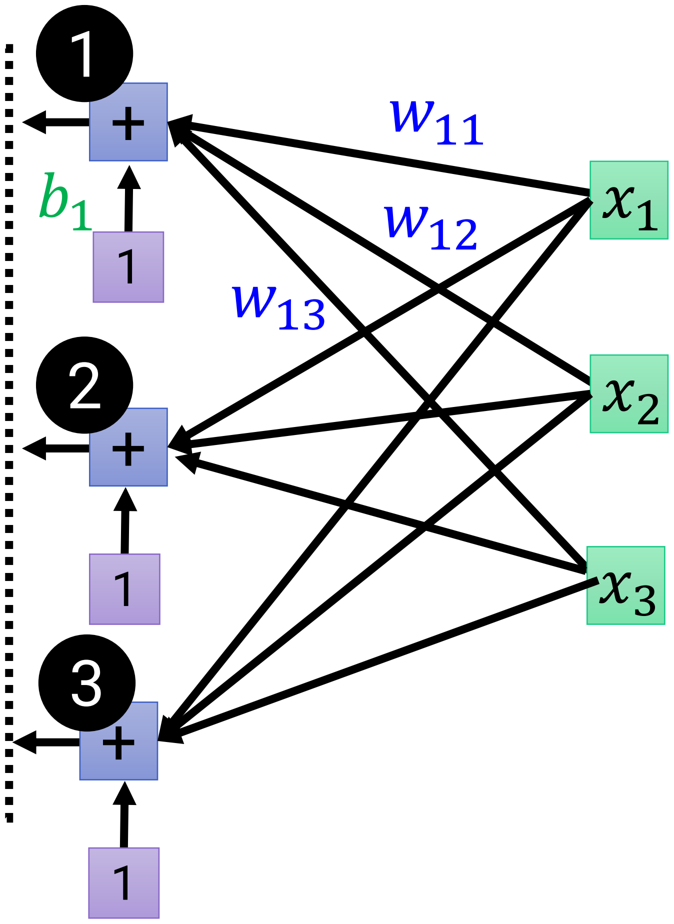
$$y = b + \sum_i c_i \, sigmoid\boxed{\left(b_i + \sum_j w_{ij} x_j\right)}$$

$$r_1 = b_1 + w_{11}x_1 + w_{12}x_2 + w_{13}x_3$$

w_{ij}: weight for x_j for i-th sigmoid

$$r_2 = b_2 + w_{21}x_1 + w_{22}x_2 + w_{23}x_3$$

$$r_3 = b_3 + w_{31}x_1 + w_{32}x_2 + w_{33}x_3$$

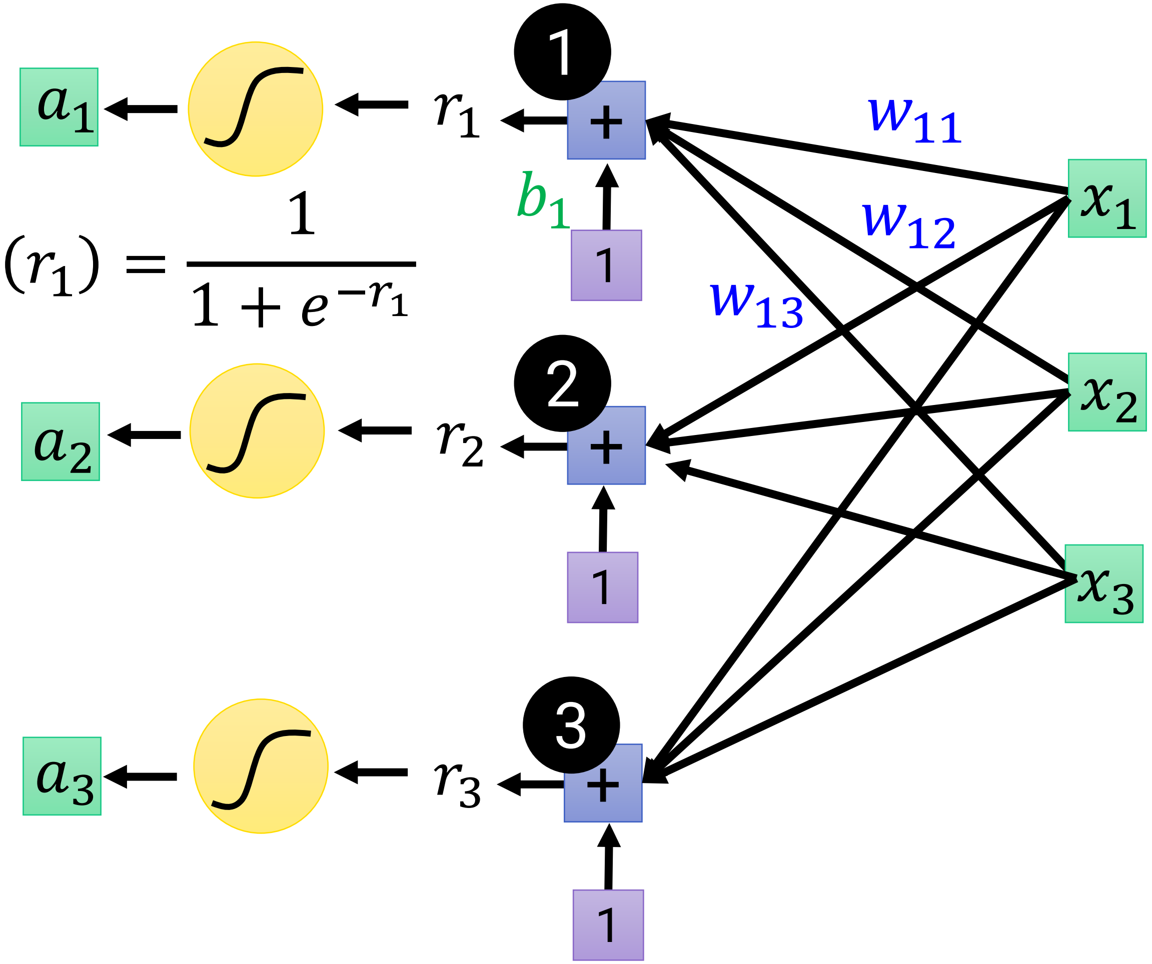


j is #features

i is #sigmoid

New Model

$$y = b + \sum_i c_i \, \boxed{sigmoid\left(b_i + \sum_j w_{ij} x_j\right)}$$



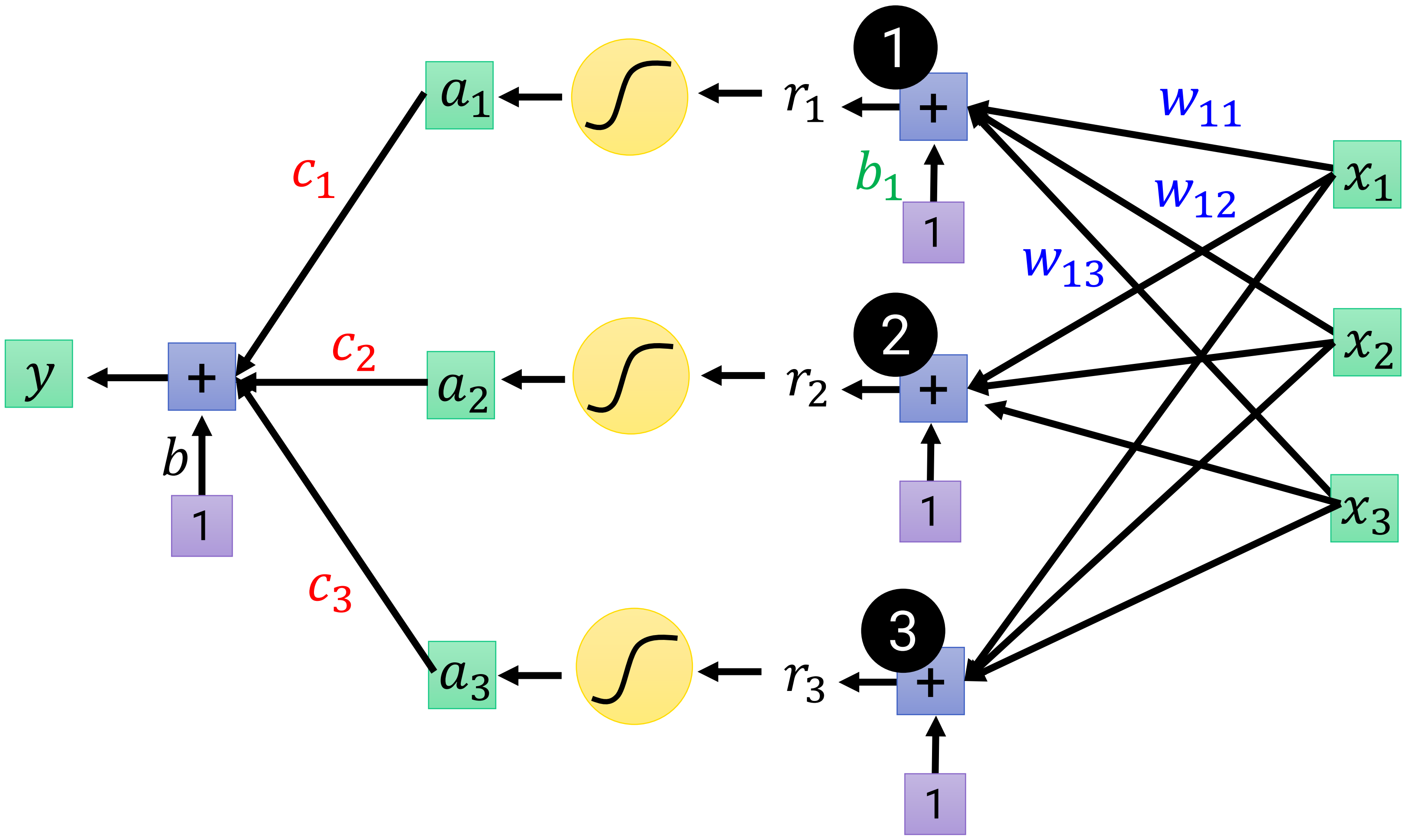
$$a_1 = sigmoid(r_1) = \frac{1}{1 + e^{-r_1}}$$

j is #features

i is #sigmoid

New Model

$$y = \boxed{b + \sum_i c_i \; sigmoid\left(b_i + \sum_j w_{ij} x_j\right)}$$

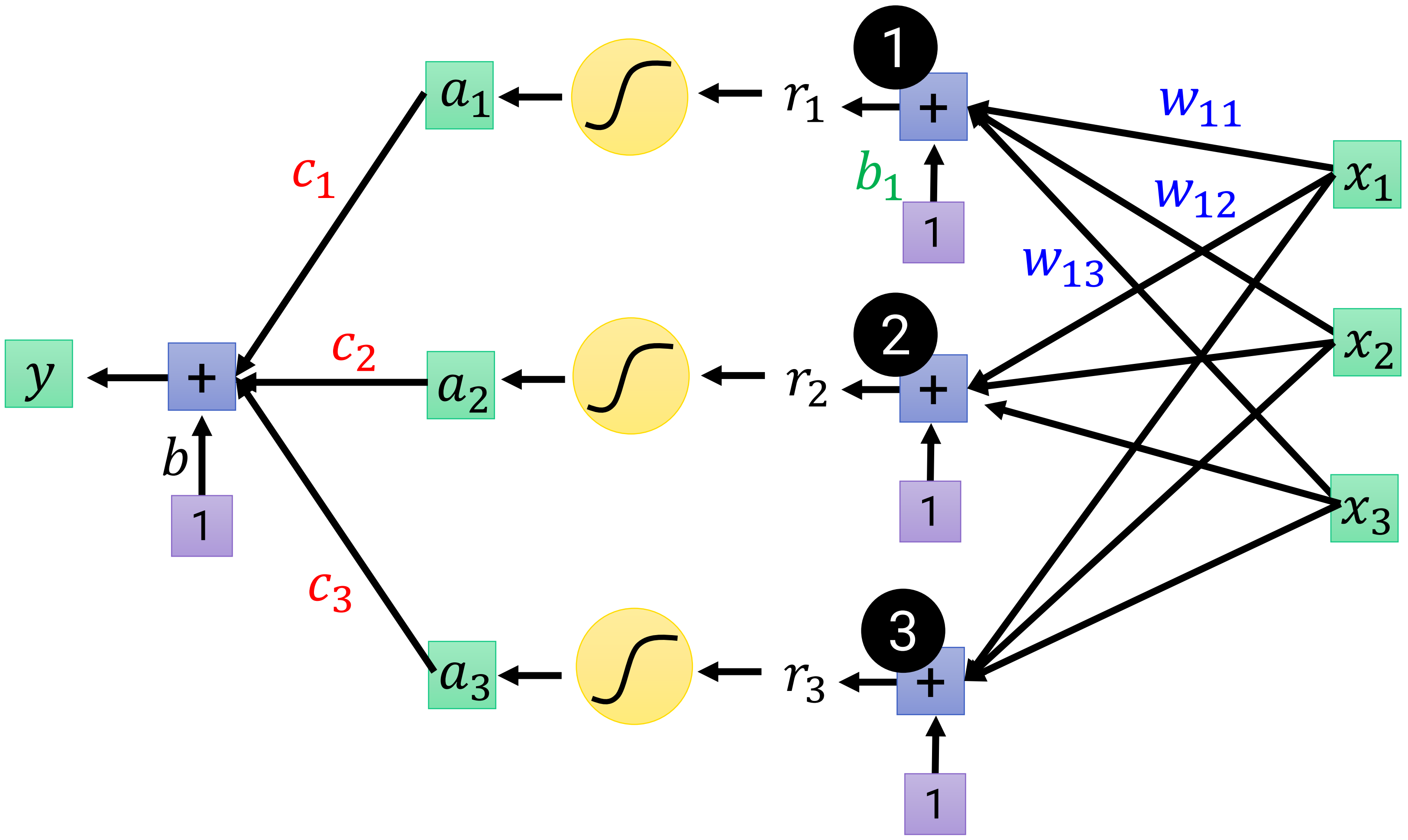


j is #features

i is #sigmoid

Matrix Style

$$y = b + \boldsymbol{c}^T \sigma(\boldsymbol{b} + W\boldsymbol{x})$$



Minimize Loss

feature

$$y = b + \boldsymbol{c}^T \sigma(\boldsymbol{b} + W \boldsymbol{x})$$

e

label $\hat{y}$

Parameters to learn

$$\text{Loss:} \quad L = \frac{1}{N} \sum_n e_n$$

Multiple parameters

Optional

$$\boldsymbol{\theta}^* = arg \min_{\boldsymbol{\theta}} L \qquad \boldsymbol{\theta} = \begin{bmatrix} \theta_1 \\ \theta_2 \\ \theta_3 \\ \vdots \end{bmatrix}$$

(Randomly) Pick initial values $\boldsymbol{\theta}^0$

$$\boldsymbol{g} = \begin{bmatrix} \frac{\partial L}{\partial \theta_1}|_{\boldsymbol{\theta}=\boldsymbol{\theta}^0} \\ \frac{\partial L}{\partial \theta_2}|_{\boldsymbol{\theta}=\boldsymbol{\theta}^0} \\ \vdots \end{bmatrix}$$

gradient

$$\begin{bmatrix} \theta_1^1 \\ \theta_2^1 \\ \vdots \end{bmatrix} \leftarrow \begin{bmatrix} \theta_1^0 \\ \theta_2^0 \\ \vdots \end{bmatrix} - \begin{bmatrix} \eta \frac{\partial L}{\partial \theta_1}|_{\boldsymbol{\theta}=\boldsymbol{\theta}^0} \\ \eta \frac{\partial L}{\partial \theta_2}|_{\boldsymbol{\theta}=\boldsymbol{\theta}^0} \\ \vdots \end{bmatrix}$$

$$\boldsymbol{g} = \nabla L(\boldsymbol{\theta}^0) \qquad \boldsymbol{\theta}^1 \leftarrow \boldsymbol{\theta}^0 - \eta \boldsymbol{g}$$

Compute the Loss for all data?

$$\boldsymbol{\theta}^* = arg \min_{\boldsymbol{\theta}} L \qquad \text{Loss: } L = \frac{1}{N}\sum_n e_n$$

- (Randomly) Pick initial values $\boldsymbol{\theta}^0$
  - Compute gradient $\boldsymbol{g} = \nabla L(\boldsymbol{\theta}^0)$

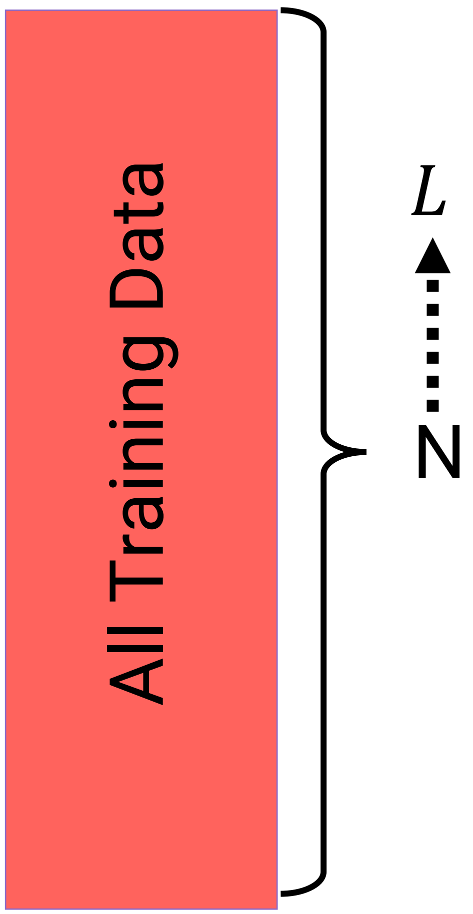
    $$\boldsymbol{\theta}^1 \leftarrow \boldsymbol{\theta}^0 - \eta \boldsymbol{g}$$

  - Compute gradient $\boldsymbol{g} = \nabla L(\boldsymbol{\theta}^1)$

    $$\boldsymbol{\theta}^2 \leftarrow \boldsymbol{\theta}^1 - \eta \boldsymbol{g}$$

    $$\cdots$$
- Repeat a set of iterations

Inefficient & ...



Compute the Loss for all data?

$\boldsymbol{\theta}^* = arg \min_{\boldsymbol{\theta}} L$ Loss: $L = \frac{1}{N}\sum_n e_n$

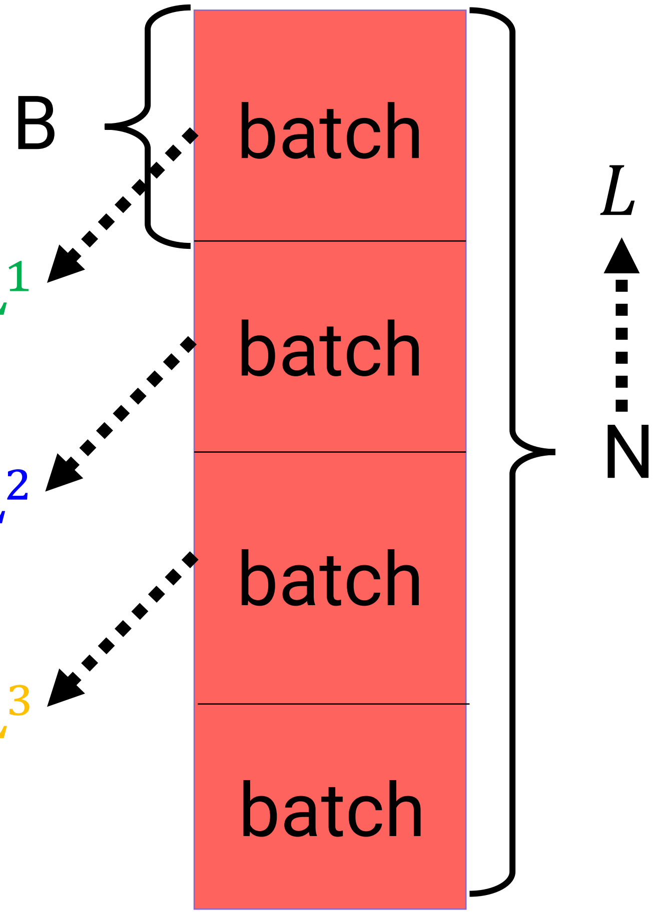
- (Randomly) Pick initial values $\boldsymbol{\theta}^0$
- Compute gradient $\boldsymbol{g} = \nabla L^1(\boldsymbol{\theta}^0)$ L^1

  update $\boldsymbol{\theta}^1 \leftarrow \boldsymbol{\theta}^0 - \eta \boldsymbol{g}$
- Compute gradient $\boldsymbol{g} = \nabla L^2(\boldsymbol{\theta}^1)$ L^2

  update $\boldsymbol{\theta}^2 \leftarrow \boldsymbol{\theta}^1 - \eta \boldsymbol{g}$
- Compute gradient $\boldsymbol{g} = \nabla L^3(\boldsymbol{\theta}^2)$ L^3

  update $\boldsymbol{\theta}^3 \leftarrow \boldsymbol{\theta}^2 - \eta \boldsymbol{g}$

1 **epoch** = see all the batches once

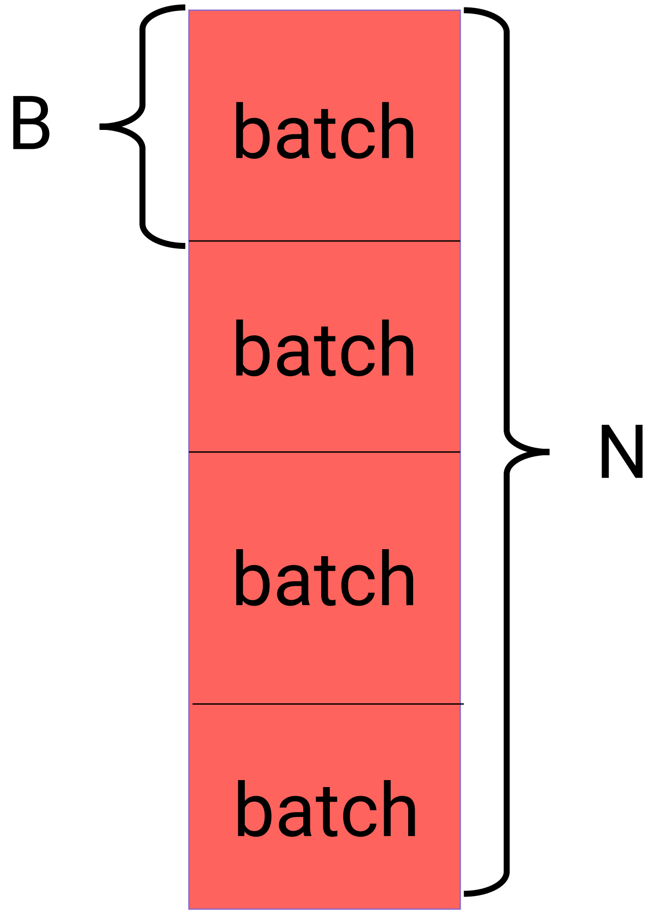


Quiz

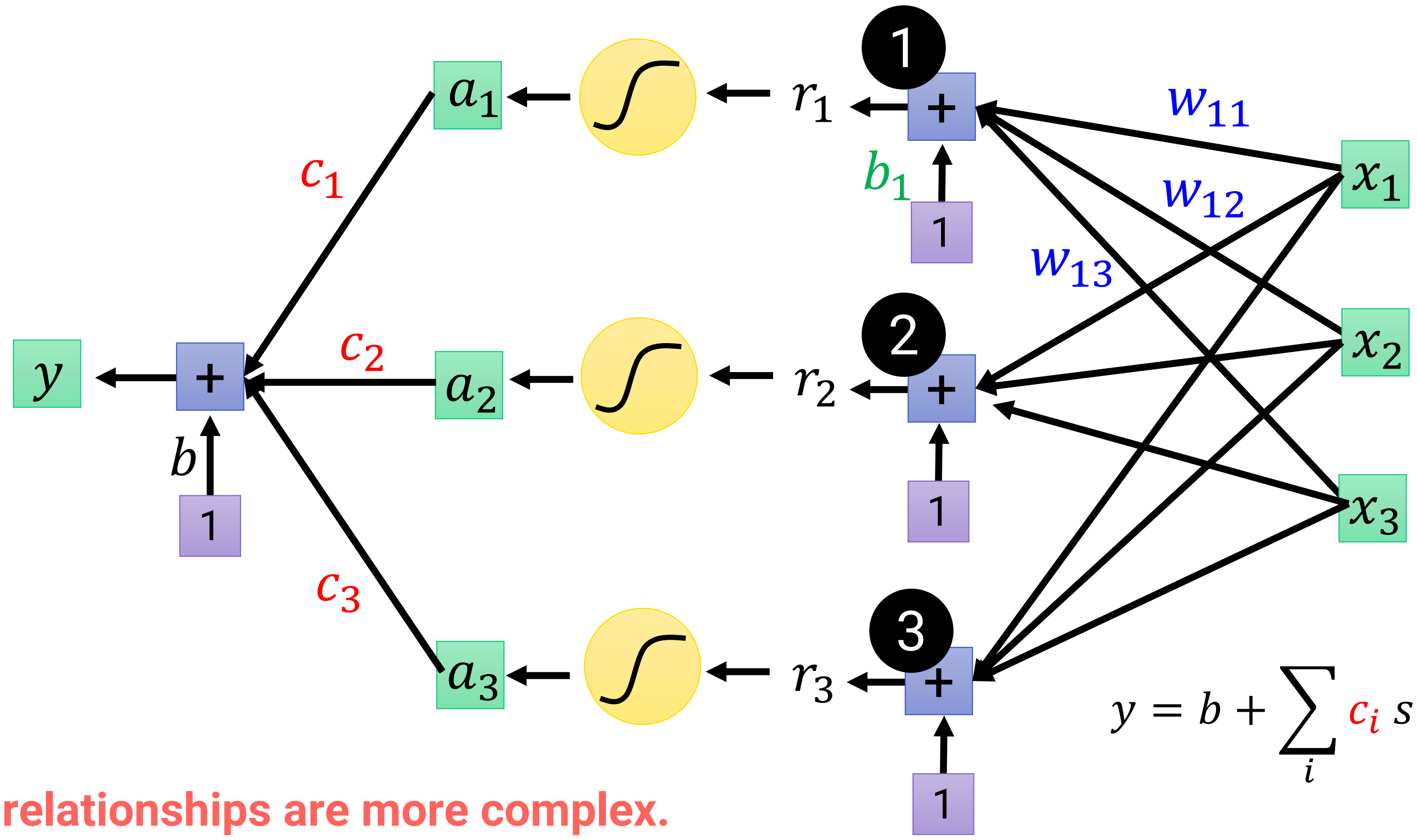
- 10,000 examples (N = 10,000)
- Batch size is 10 (B = 10)

How many update in **1 epoch**?

1,000 updates



Machine Learning to Deep Learning

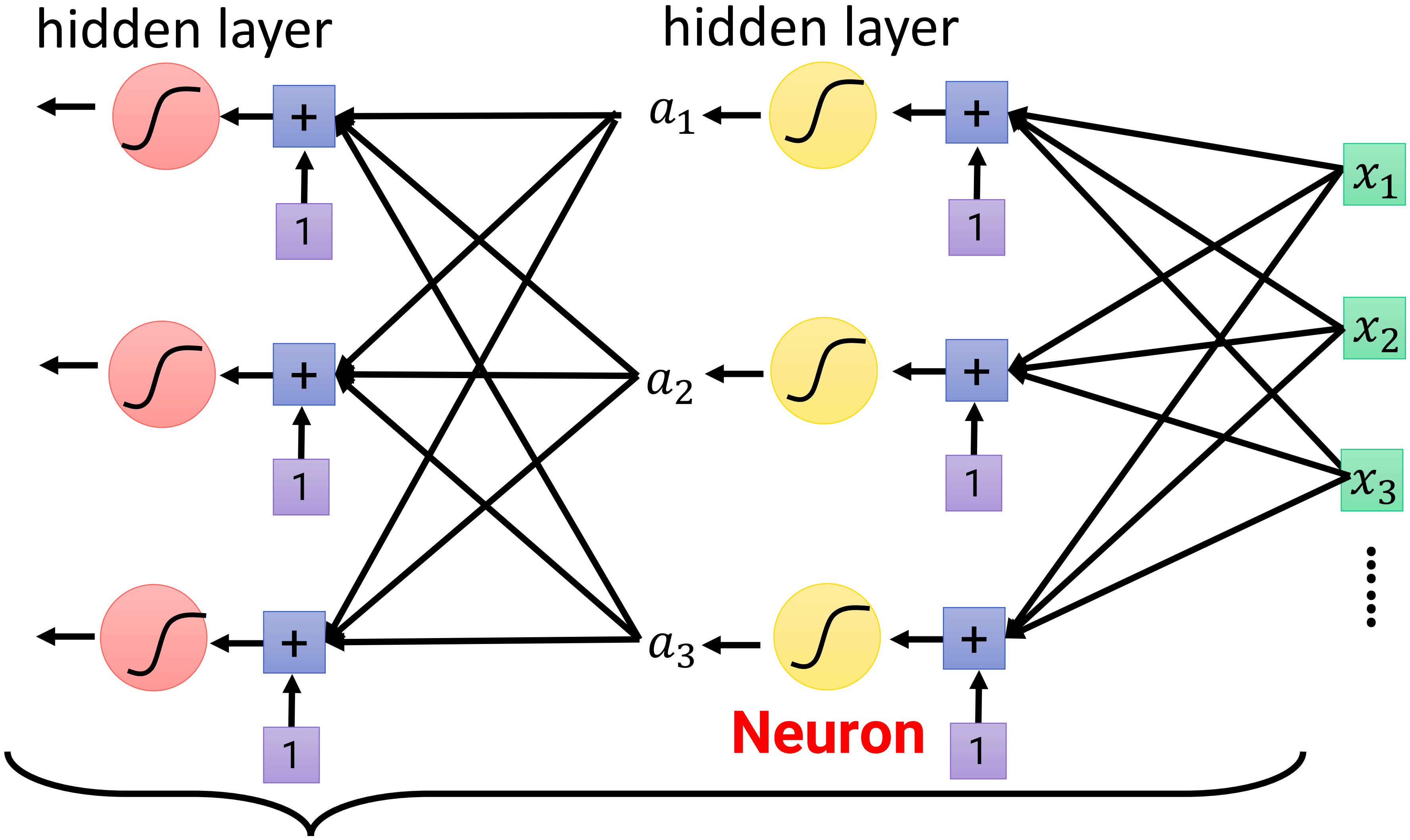


j is #features

i is #sigmoid

$$y = b + \sum_i c_i \; sigmoid\left(b_i + \sum_j w_{ij} x_j\right)$$

Real relationships are more complex.

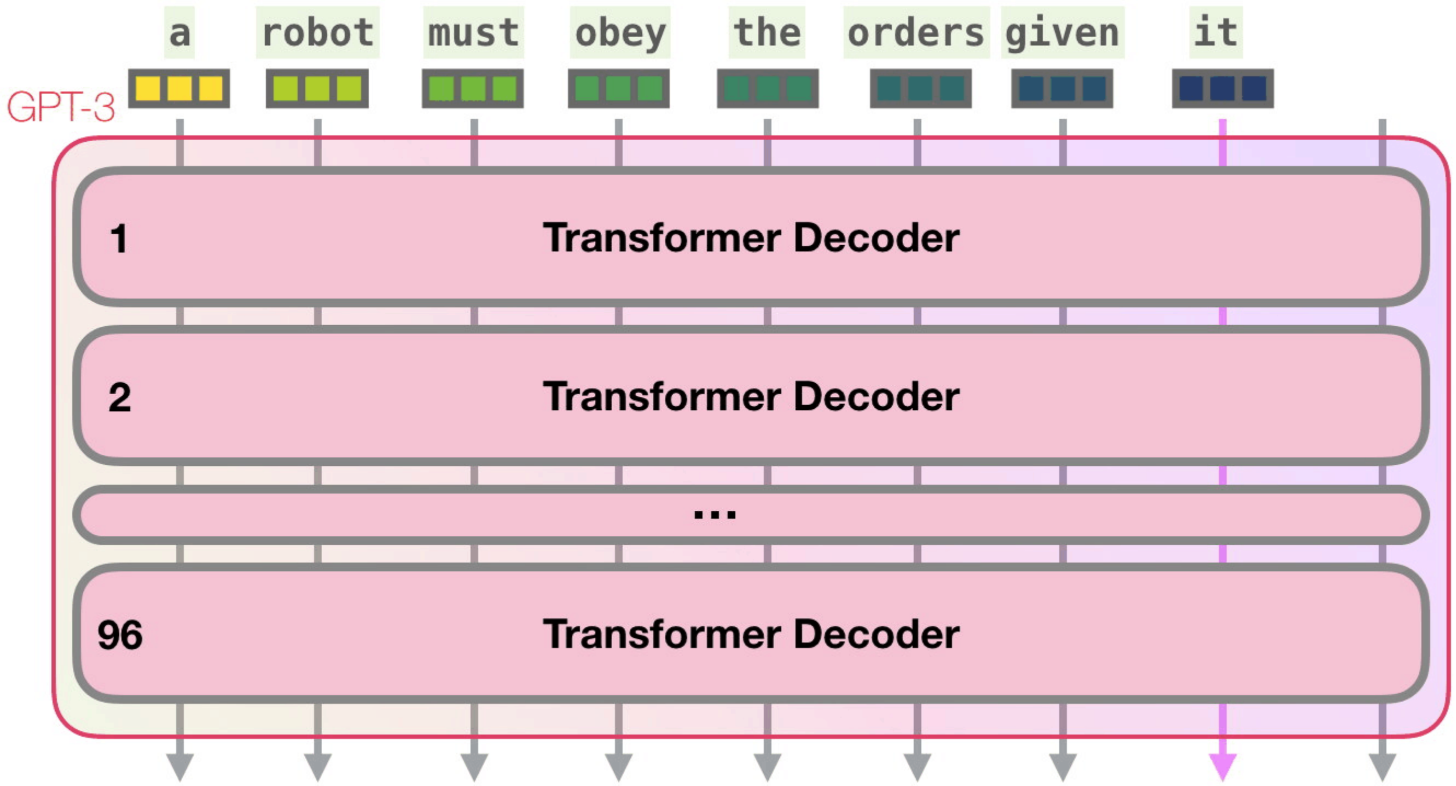


Neural Network

Deep means many layers

#Layers in GPT-3

https://jalammar.github.io/how-gpt3-works-visualizations-animations/

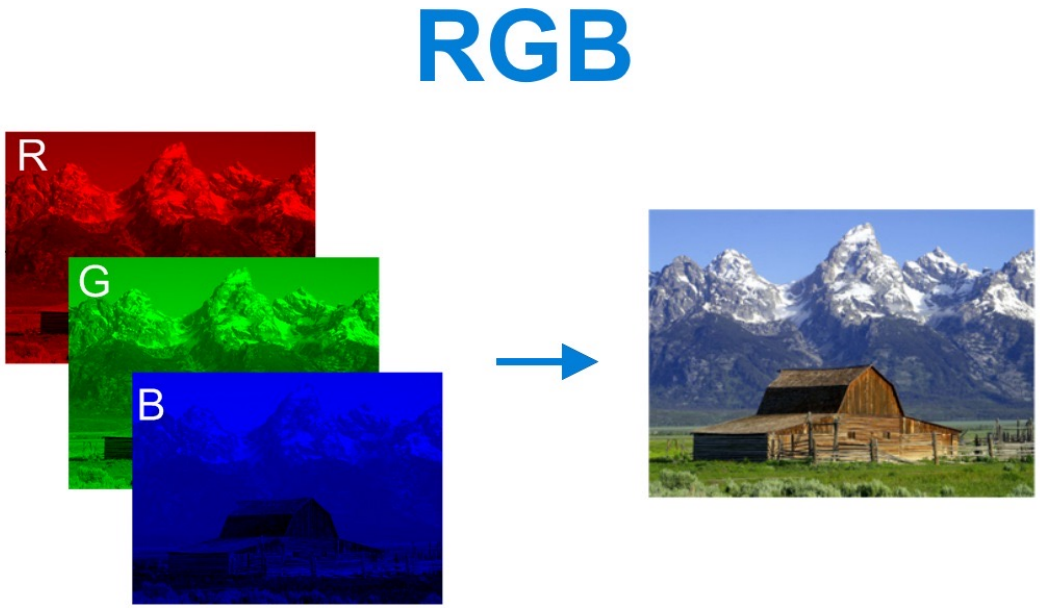


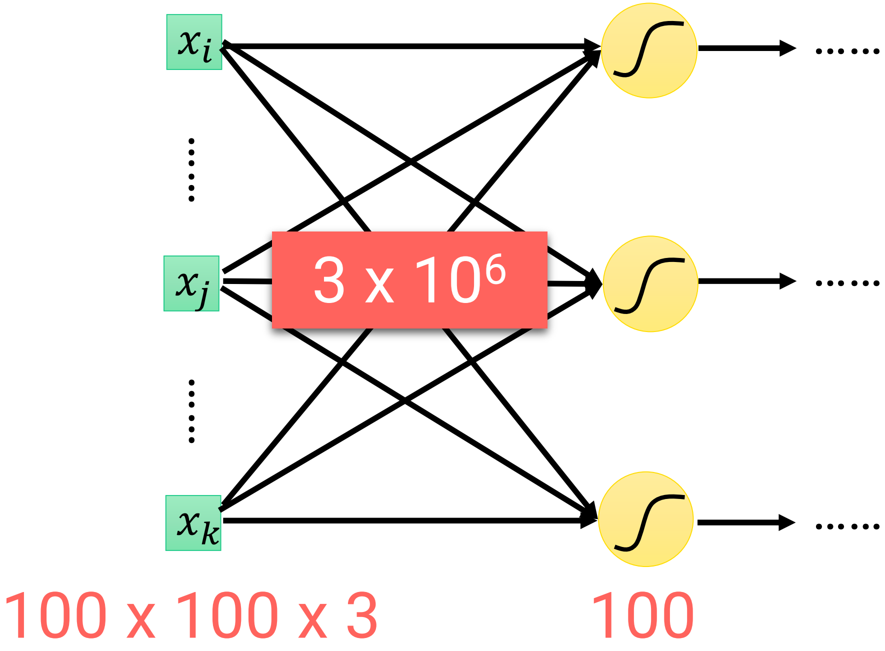
generative pre-trained transformer

Deep vs Wide

DL in real applications

Domain knowledge -> customized model (neural network)

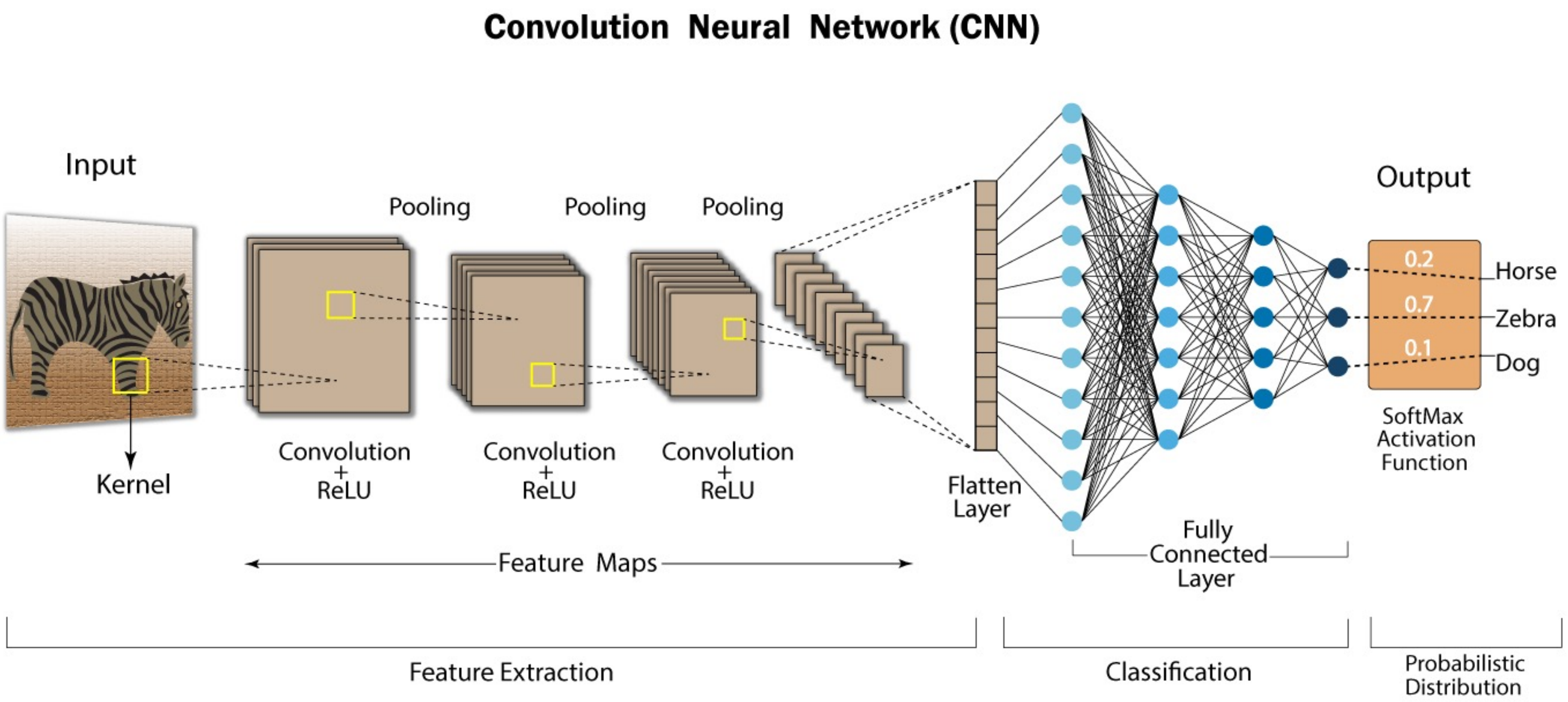




Assume we have an image with 100 pixels
A color can be represented by (256*256*256)

DL in real applications

Domain knowledge -> customized model (neural network)



Learning Outcome

Understand the basic idea of ML and DL

Learn more details in **COMP9444**