Lecture 4

Multi-layer Perceptron

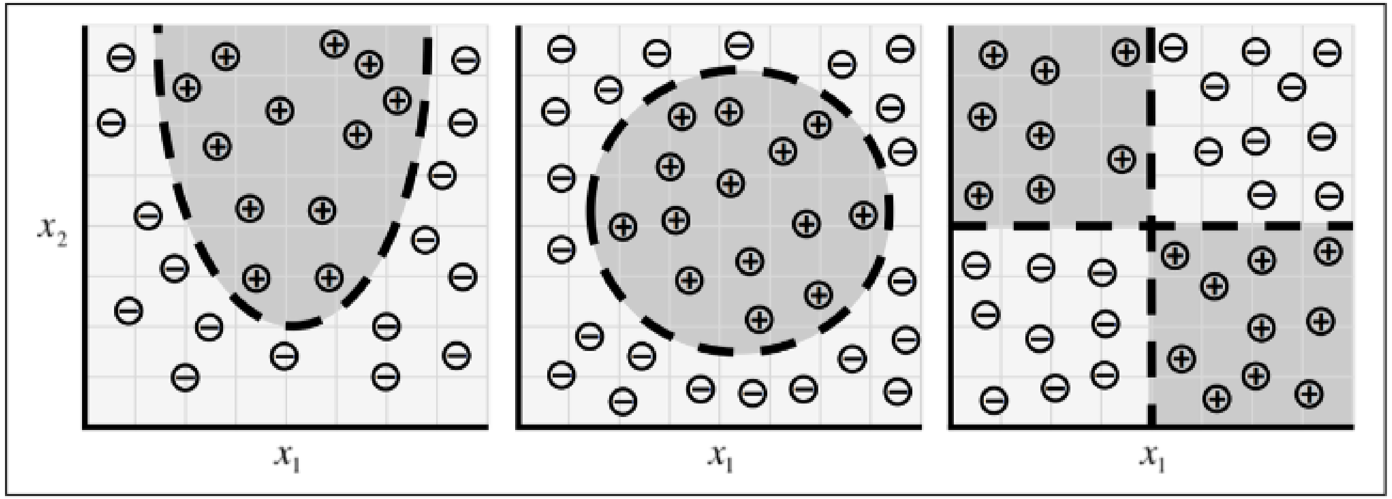
CSE465: Pattern Recognition and Neural Network
Sec: 3
Faculty: Silvia Ahmed (SvA)
Summer 2025

Today’s Topic

1. Multi layer perceptron
    a) Problems with Perceptron
    b) MLP Notation
    c) Forward Propagation
    d) Loss function
    e) Back Propagation

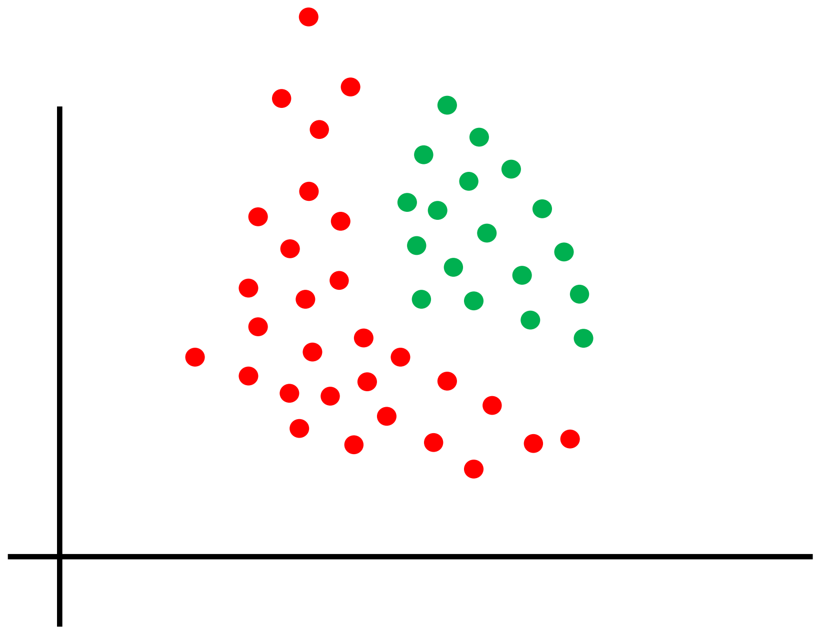
Limitation of a Perceptron

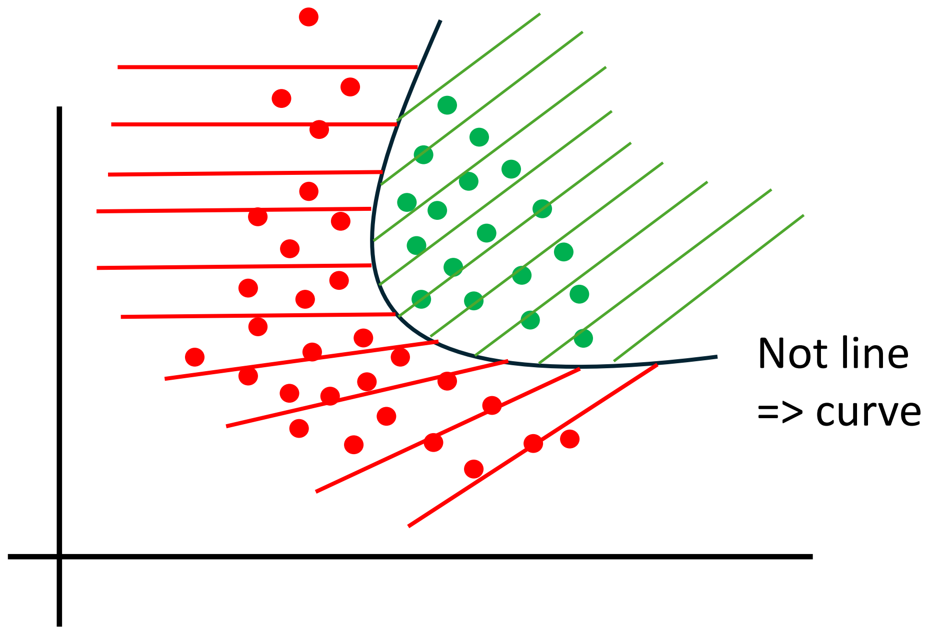
- Works only with linear or “sort-of” linear data



- Solution:
  - Multi-layer Perceptron

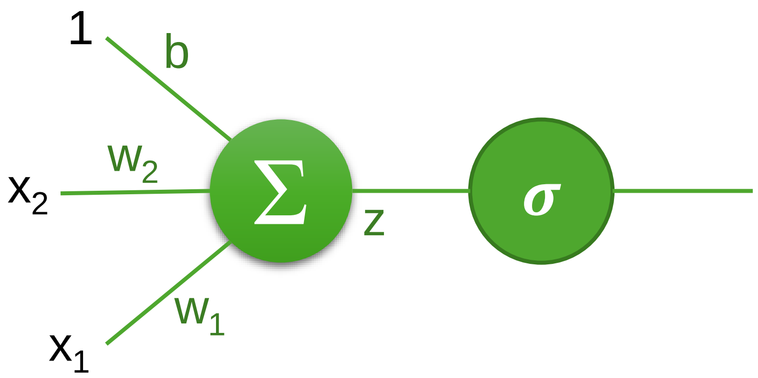
The problem

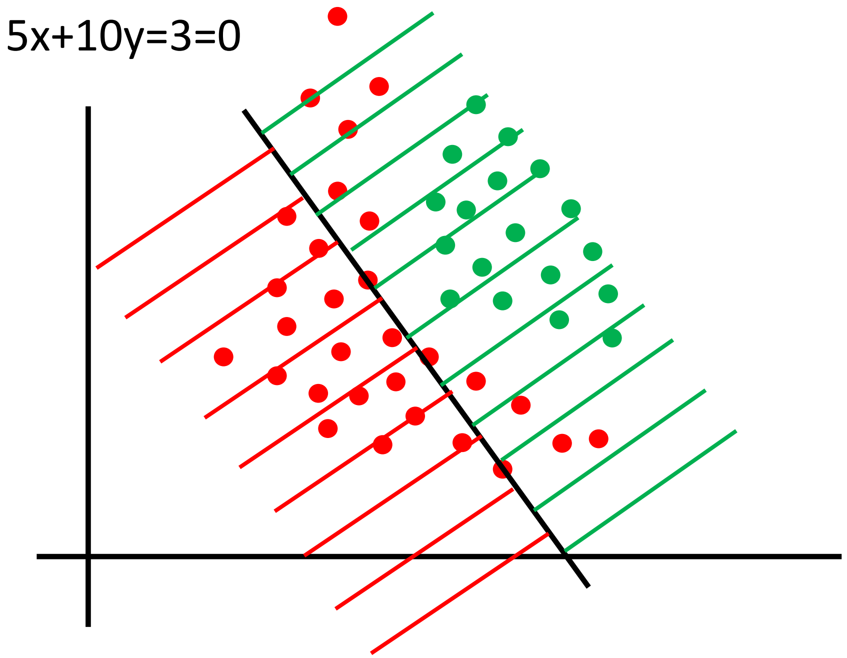




- Perceptron doesn’t work on a dataset like this, as perceptron ends up with a line. We need a curve as the decision boundary.
- Challenge: we can only use Perceptron(s) to build such a system.

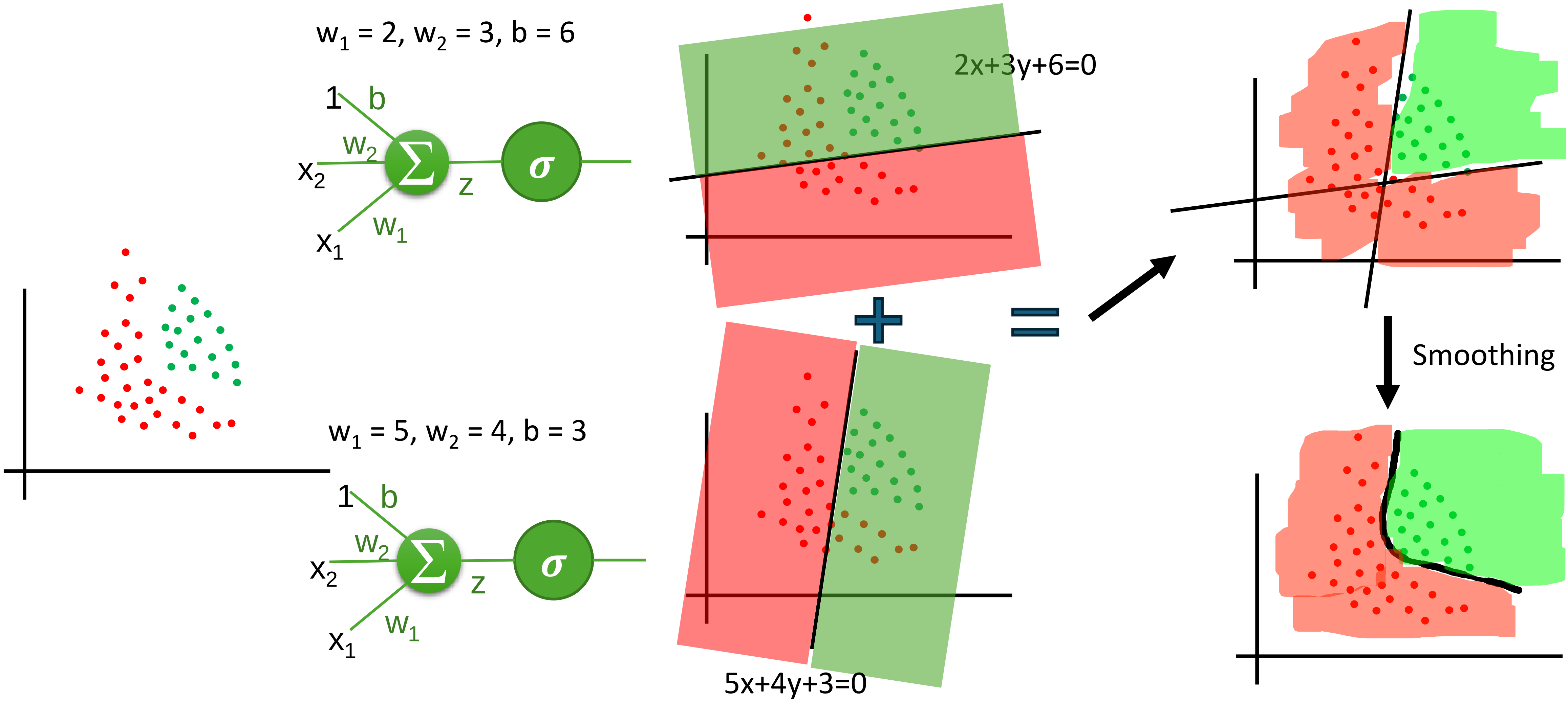
Perceptron with Sigmoid



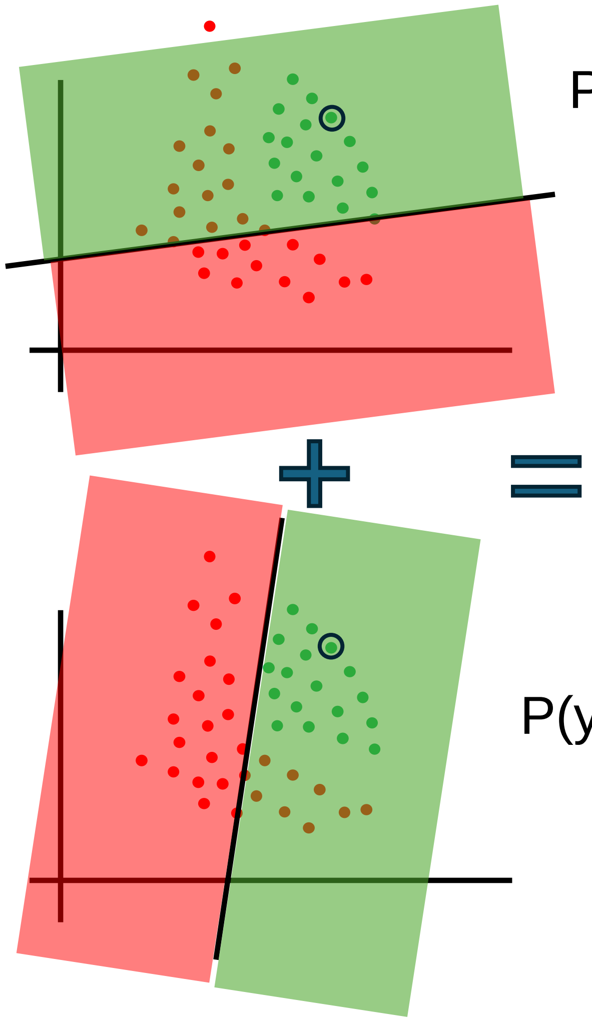


CGPA, x_1	IQ, x_2	z	$\sigma = \frac{1}{1+e^{-z}}$
3.7	87	5 x 3.7 + 10 x 87 + 3 = 891.5	0

MLP Intuition

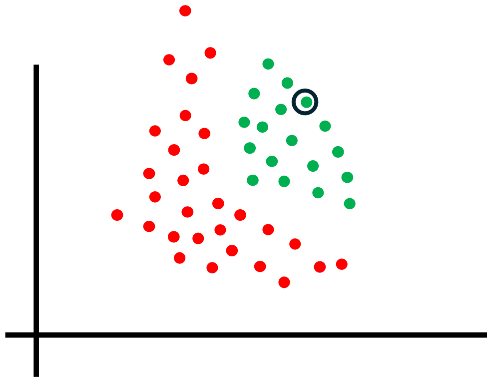


MLP: Mathematics

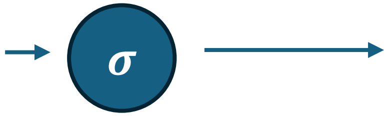


P(y) = 0.7

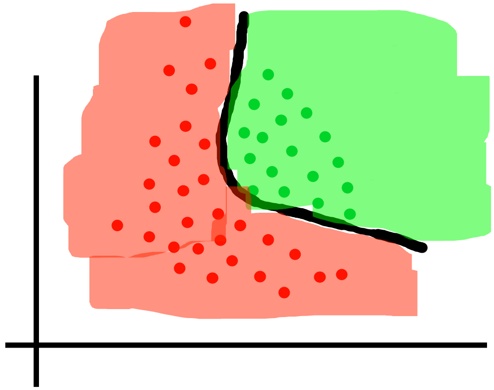
P(y) = 0.8



$0.7 + 0.8 = 1.5$ (!)



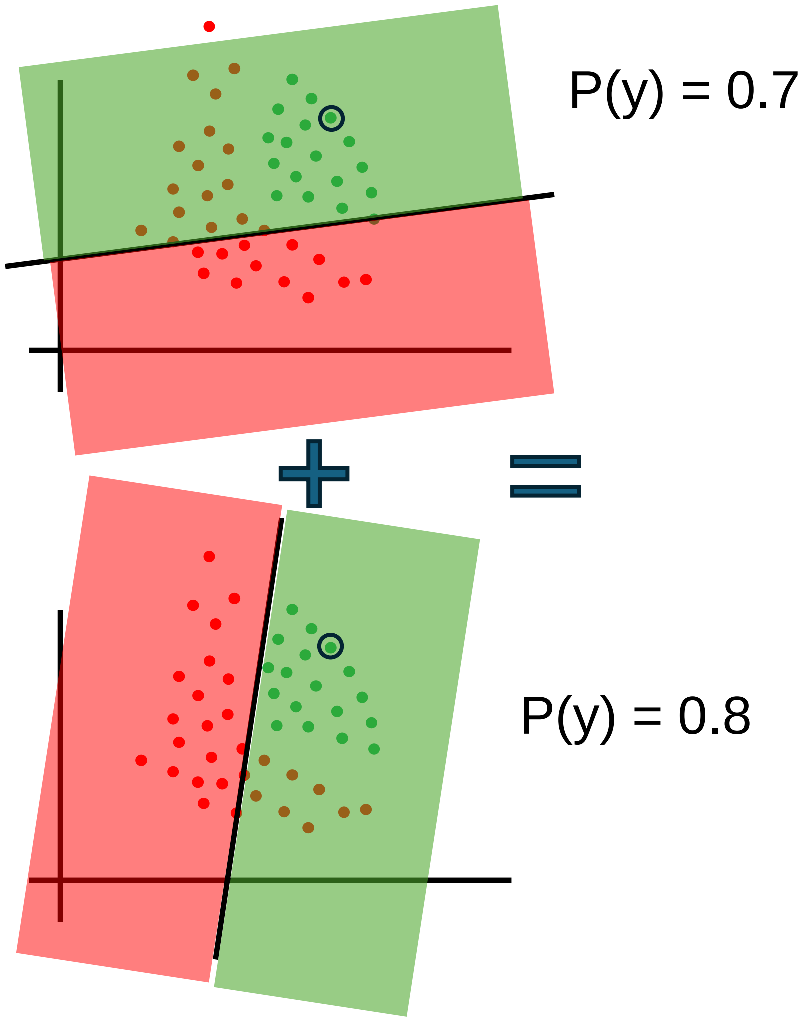
$$\frac{1}{1+e^{-1.5}} = 0.82$$



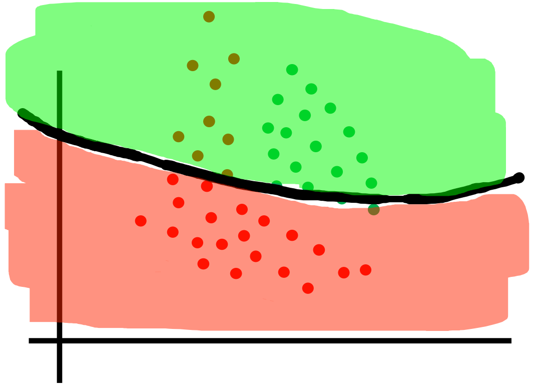
Therefore,

- First take addition of perceptrons.
- Then apply sigmoid as activation.

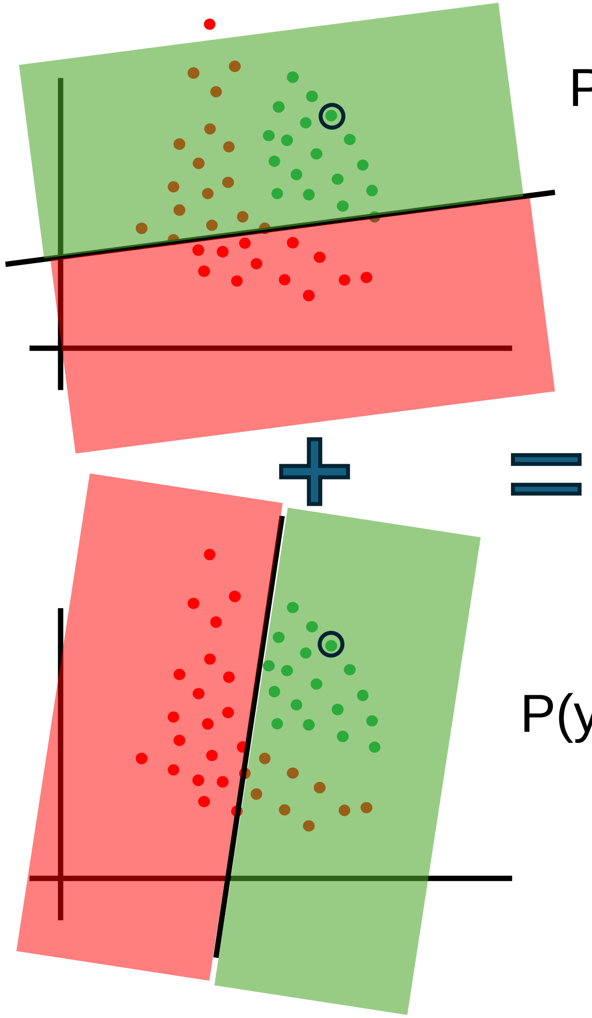
MLP: Mathematics (further enhancements)



Suppose we want the effect of Perceptron-1 more

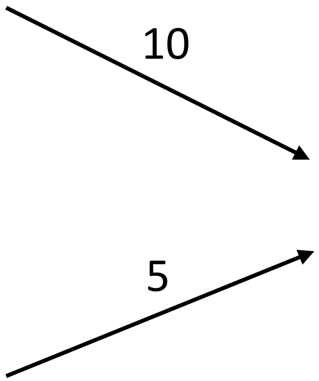


MLP: Mathematics (further enhancements)

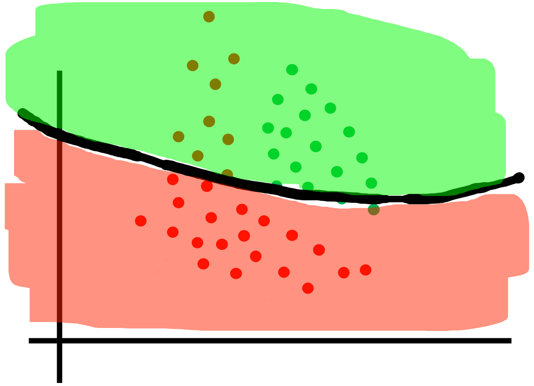


P(y) = 0.7

P(y) = 0.8



Suppose we want the effect of Perceptron-1 more



Instead of normal sum, we have to do weighted sum.

0.7 x 10 + 0.8 x 5 = z

↓

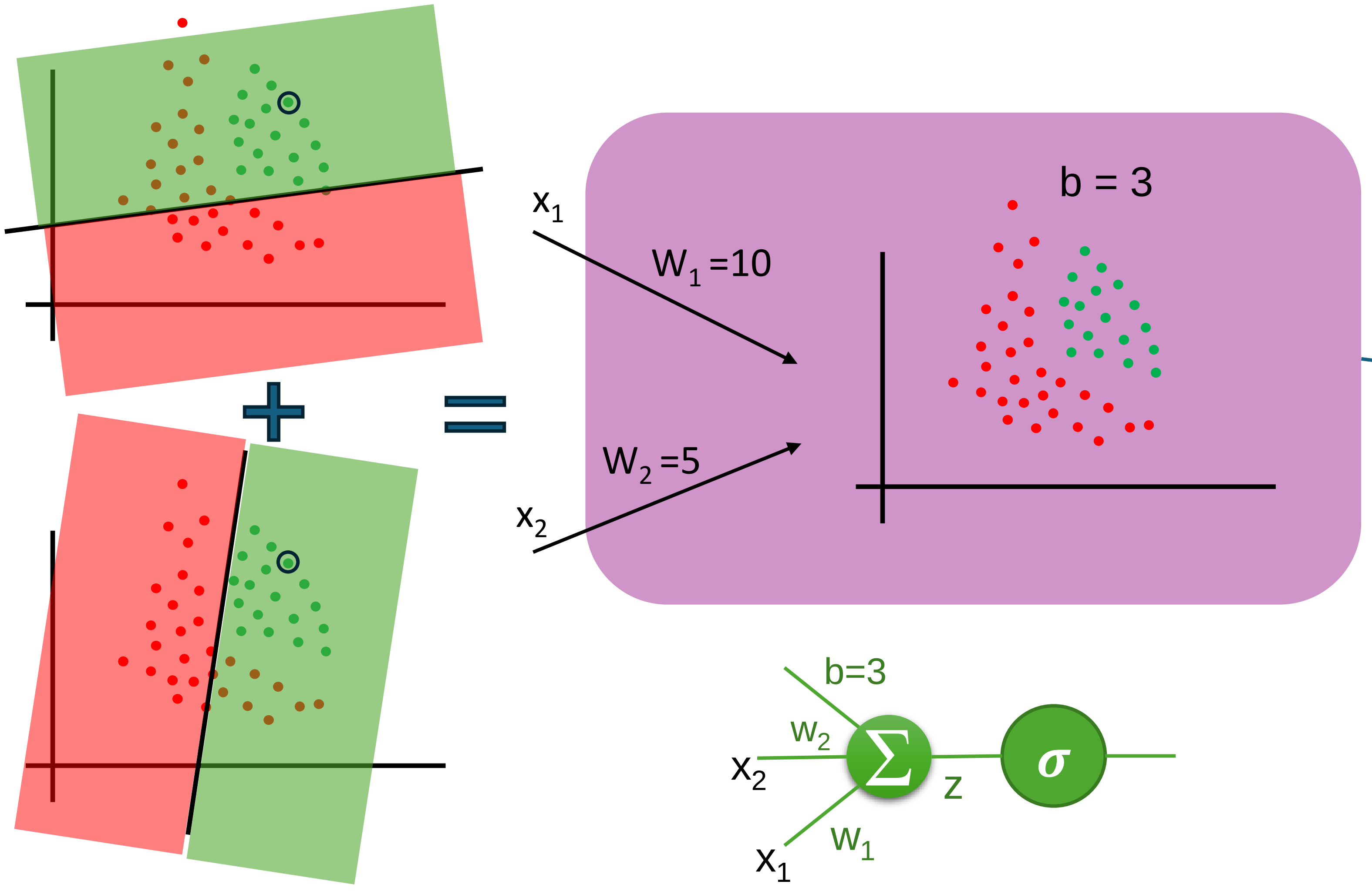
$\sigma(z)$

↓

P(y)

You can even add some “bias”

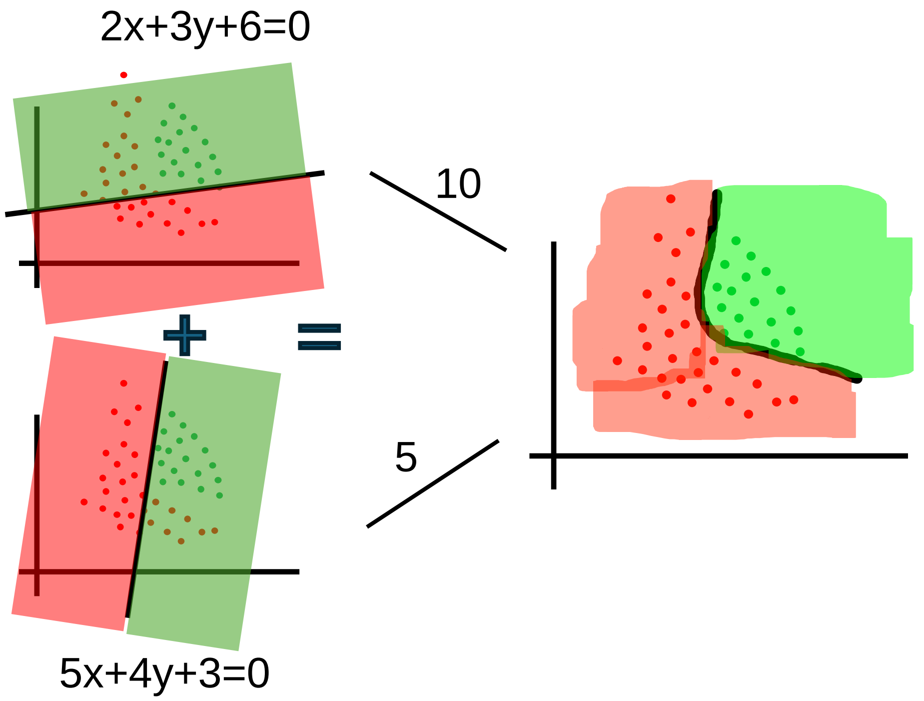
MLP: Mathematics (further enhancements)

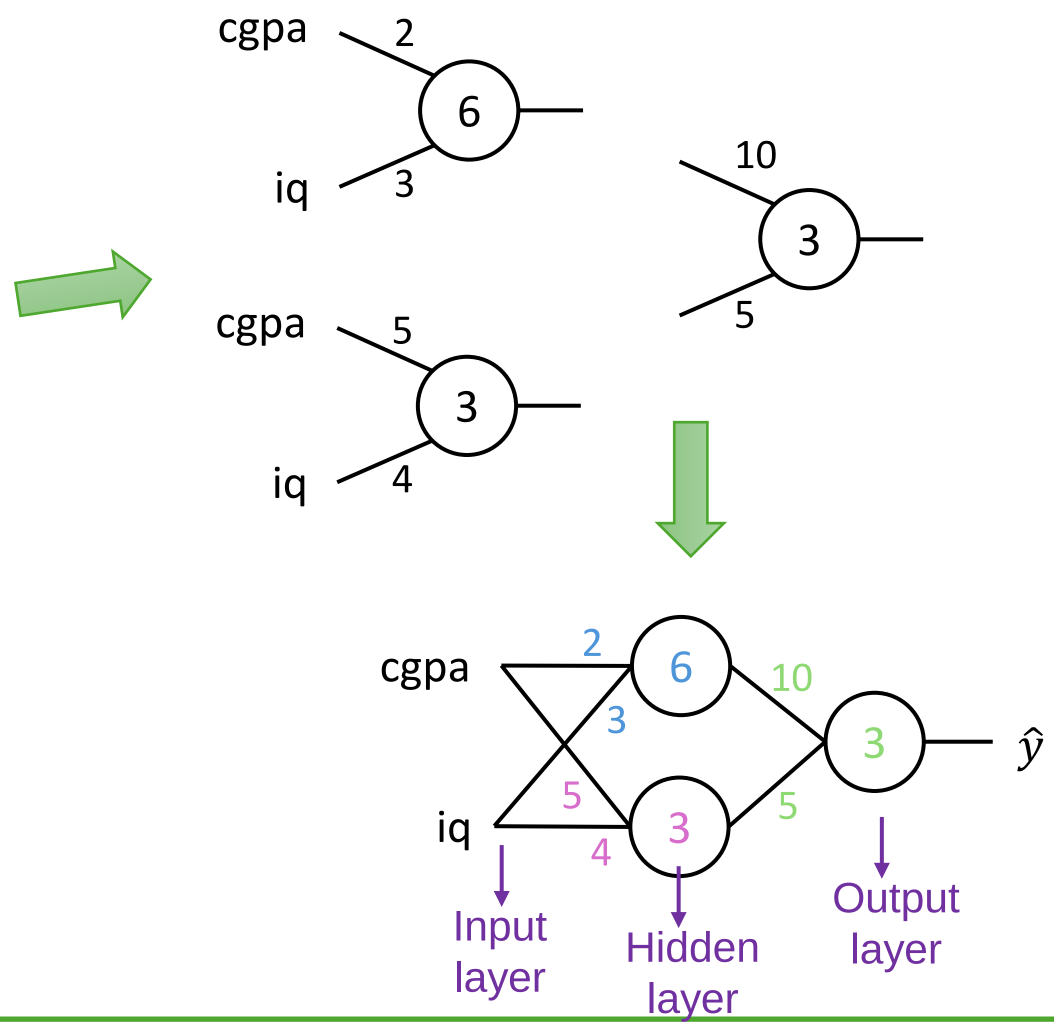


So the entire thing works as a Perceptron, where the inputs are the outputs of the 2 perceptrons.

That is, it is a multi-layered perceptron

MLP: Formation

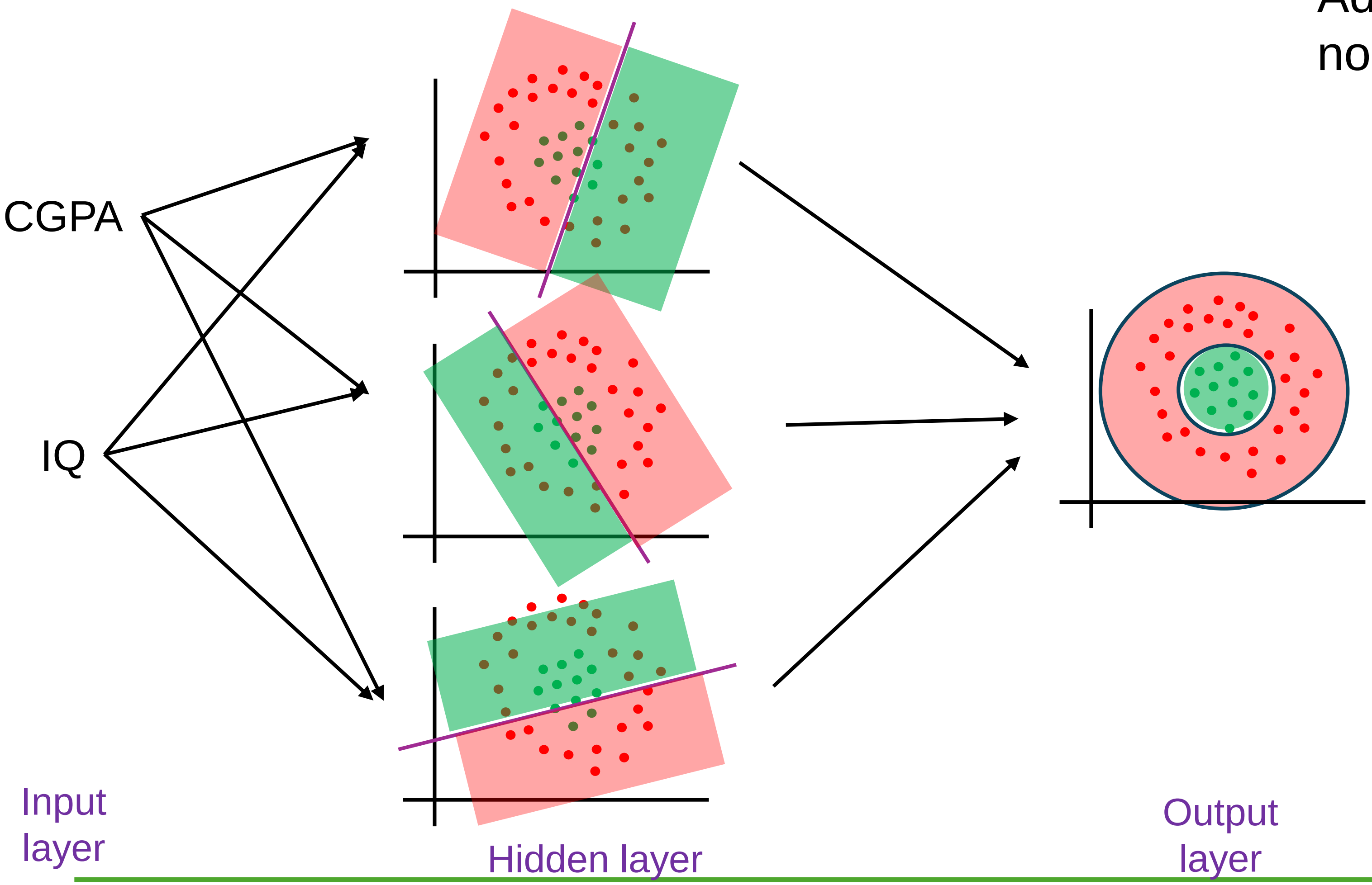




MLP: Formation with more flexibility

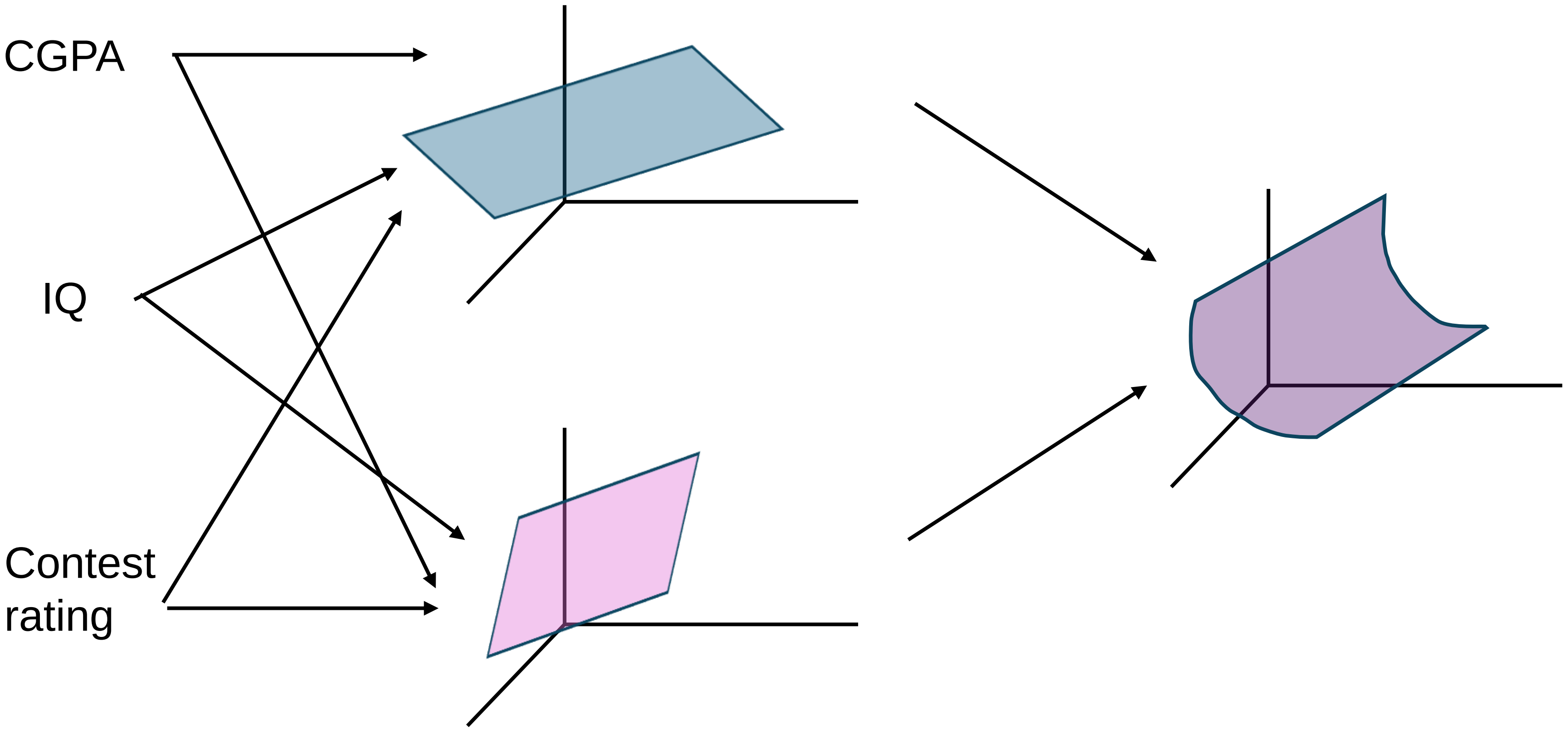
- Adding nodes in hidden layer

Additional and more complex non-linearity can be captured



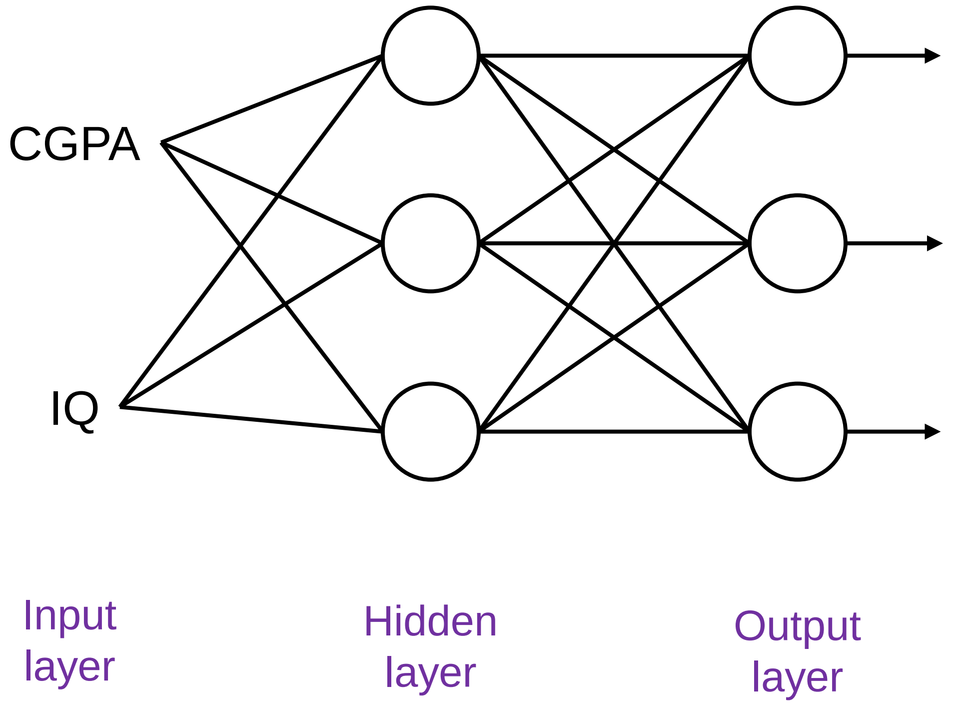
MLP: Formation with more flexibility

- Adding nodes in input layer (with extra features)



MLP: Formation with more flexibility

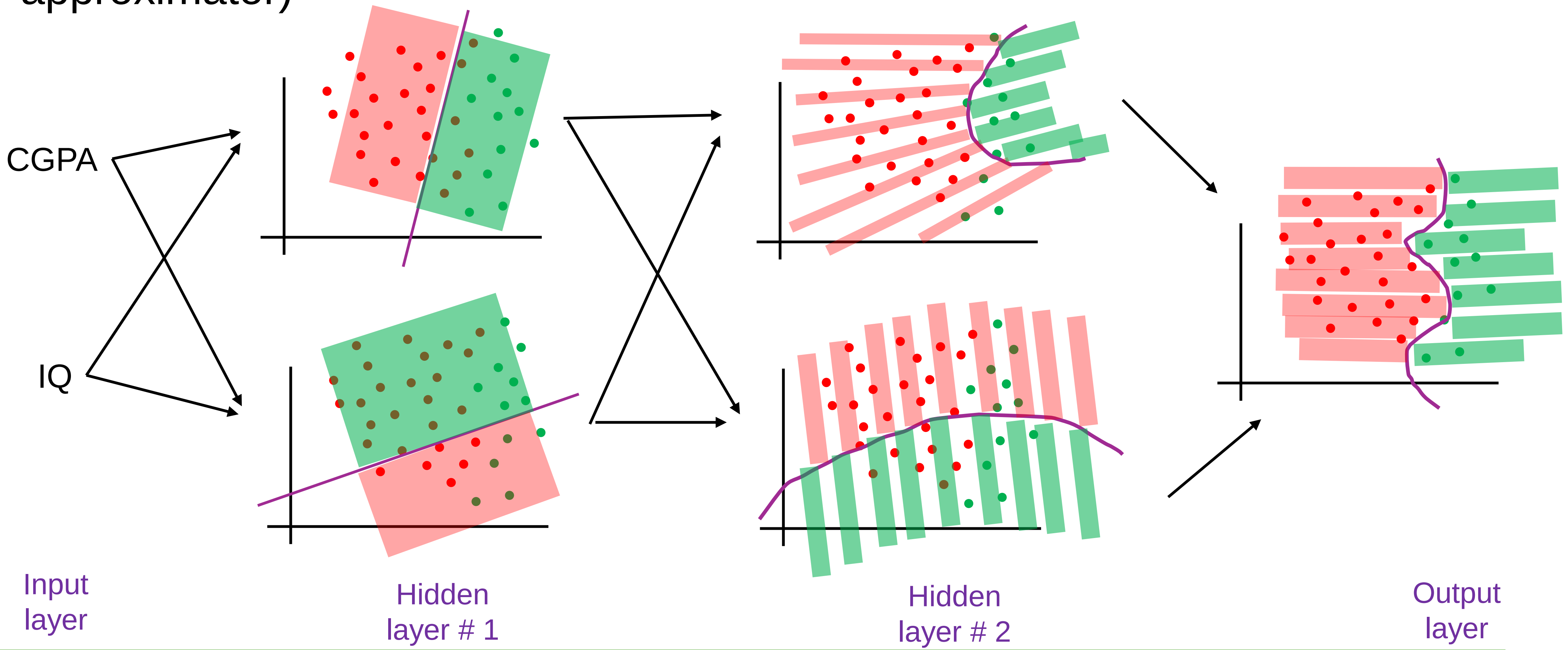
- Adding nodes in output layer



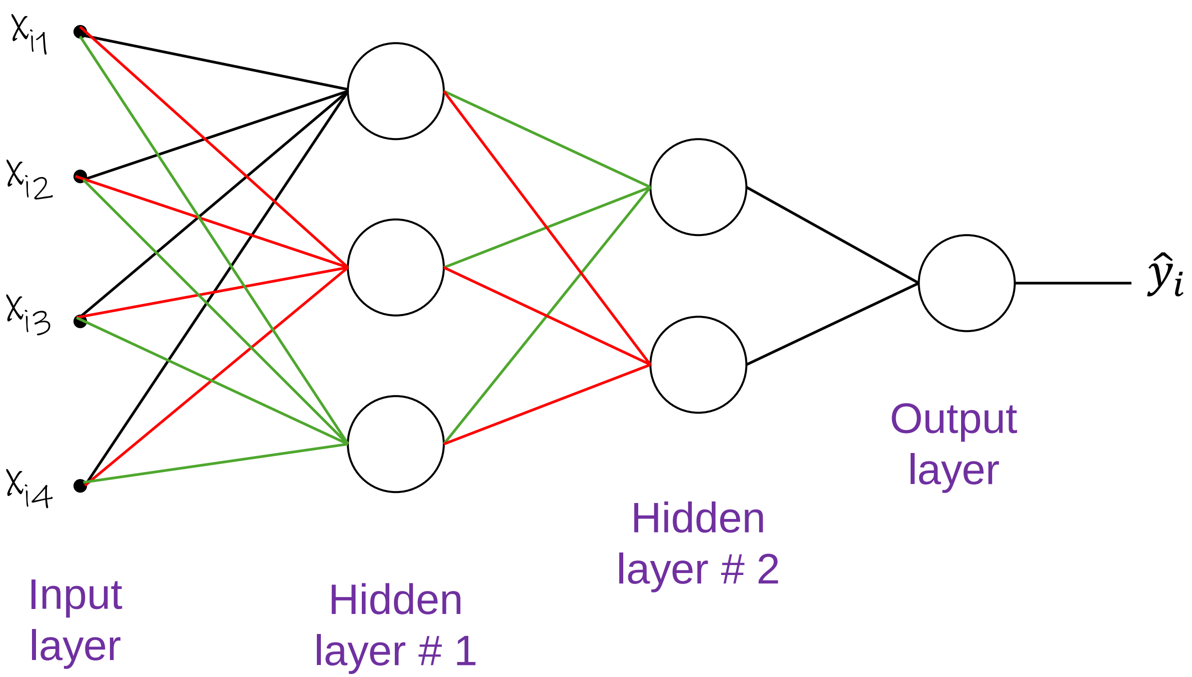
Used in multi-class classification.
E.g. Object classification

MLP: Formation with more flexibility

- Adding more hidden layers (Deep Neural Network, a.k.a. Universal Function approximator)



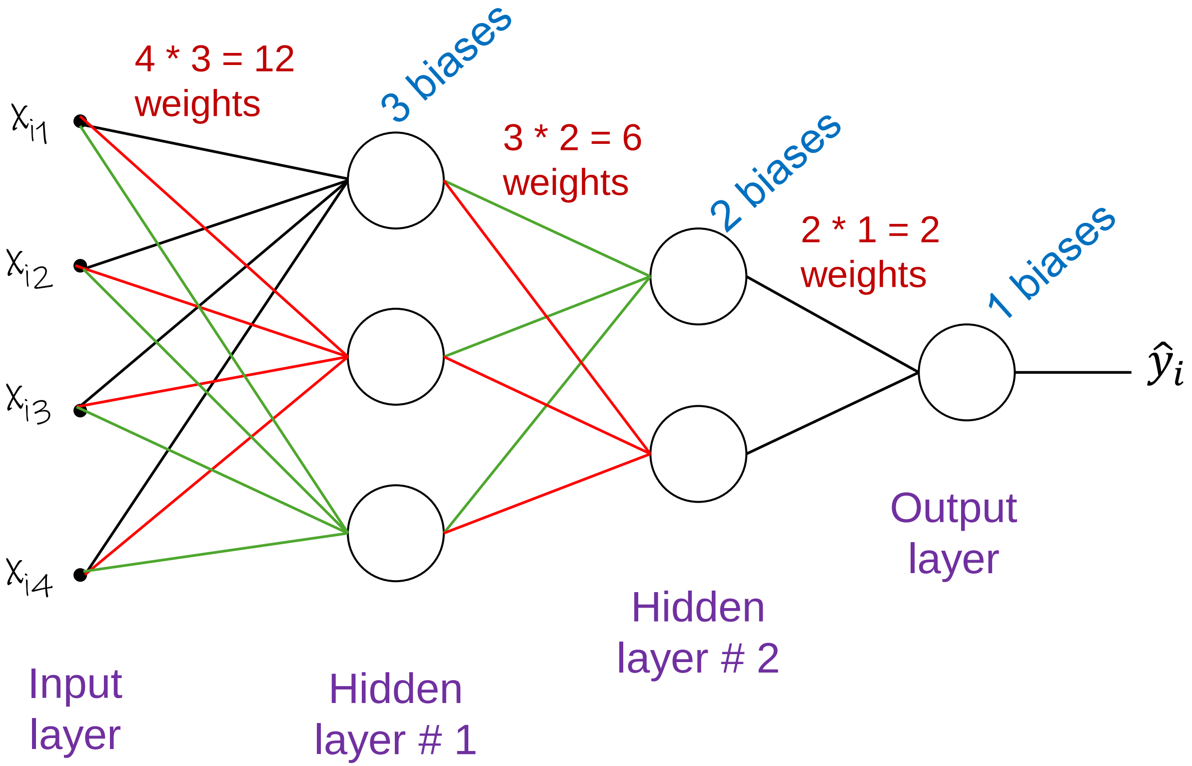
MLP Notation



Q1: What is the total number of trainable parameters for a neural network?

- Data → {m x n}
- m → No. of rows (samples)
- n → number of input features
- Each row is denoted by x_i

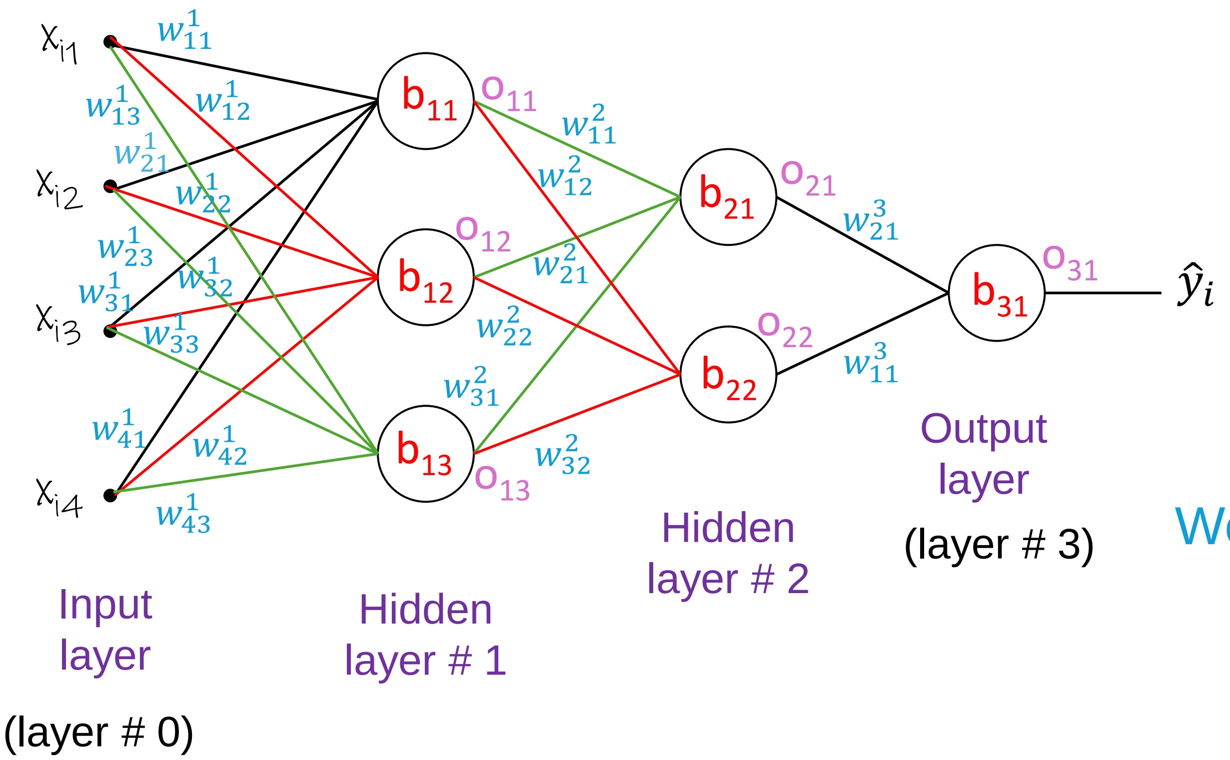
MLP Notation



Q1: What is the total number of trainable parameters for a neural network?

$$\text{Total parameters} = \overbrace{(12 + 3)}^{\text{Layer 1}} + \overbrace{(6 + 2)}^{\text{Layer 2}} + \overbrace{(2 + 1)}^{\text{Layer 3}}$$
$$= 26$$

MLP Notation

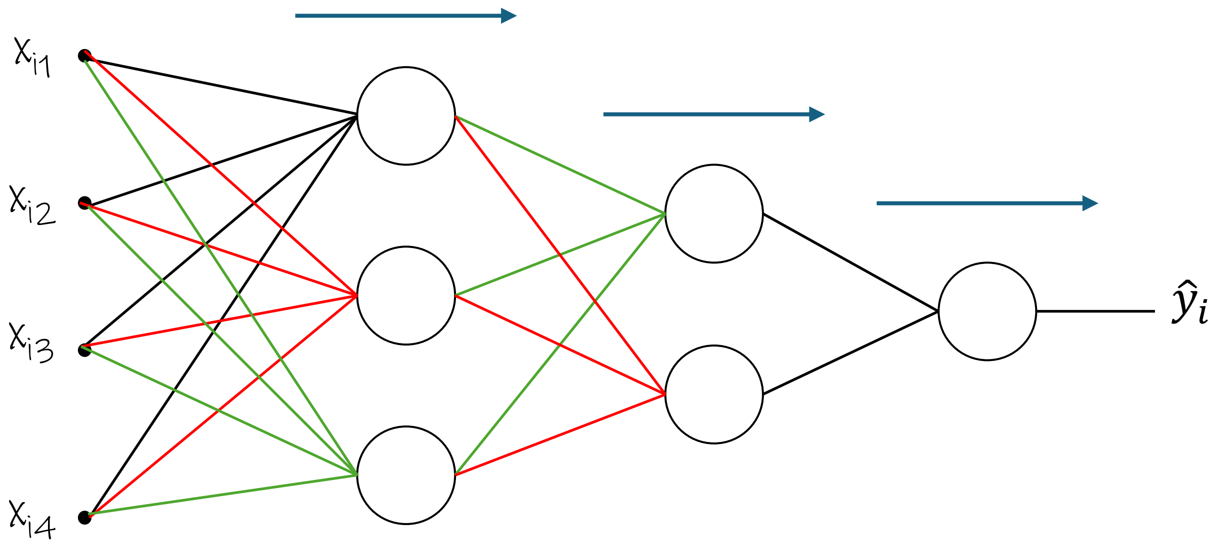


Bias ➔ b_{ij} ➔ i : layer no.
j : node no.

Output ➔ o_{ij} ➔ i : layer no.
j : node no.

Weights ➔ w_{ij}^{k} ➔ k : layer no. it's going into
i : node no. in the source layer
j : node no. in the destination layer

Forward Propagation



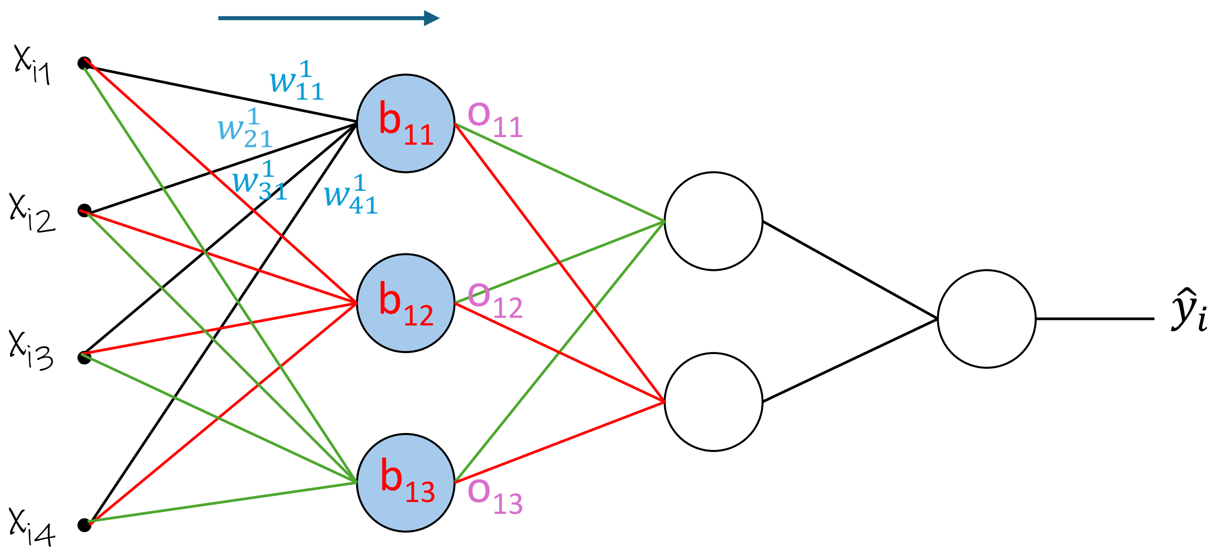
Starting from the input, calculations are done layer-by-layer to finally calculate the prediction, $\hat{y}_i$

Forward Propagation: layer 1

$$z_{11} = w_{11}^1 x_{i1} + w_{21}^1 x_{i2} + w_{31}^1 x_{i3} + w_{41}^1 x_{i4} + b_{11}$$
$$o_{11} = \sigma(z_{11})$$

$$z_{12} = w_{12}^1 x_{i1} + w_{22}^1 x_{i2} + w_{32}^1 x_{i3} + w_{42}^1 x_{i4} + b_{12}$$
$$o_{12} = \sigma(z_{12})$$

$$z_{13} = w_{13}^1 x_{i1} + w_{23}^1 x_{i2} + w_{33}^1 x_{i3} + w_{43}^1 x_{i4} + b_{13}$$
$$o_{13} = \sigma(z_{13})$$



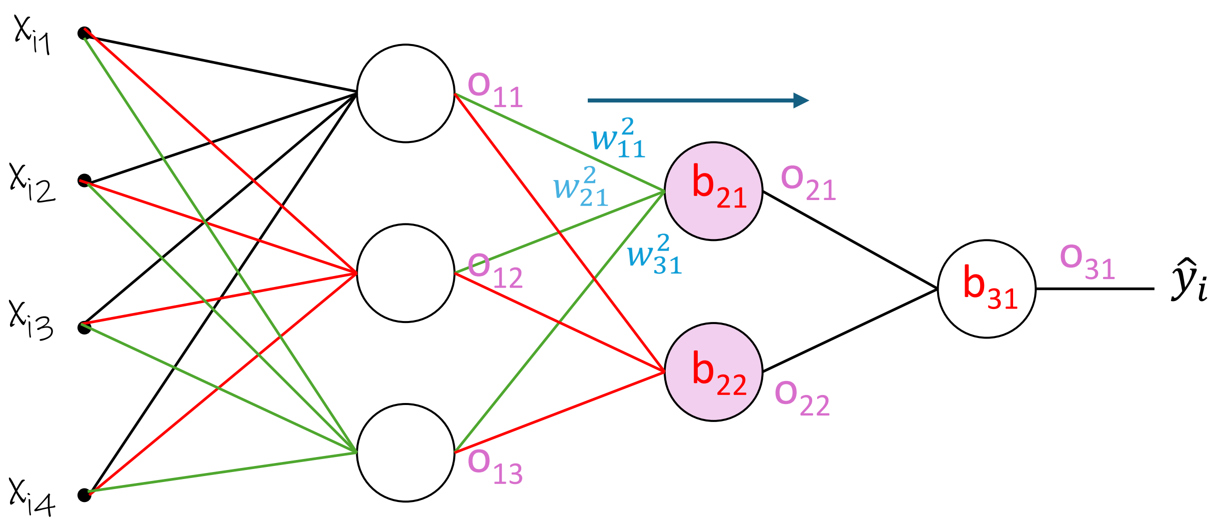
Fortunately, this can be done easily using matrix calculation

$$\begin{bmatrix} w_{11}^1 & w_{12}^1 & w_{13}^1 \\ w_{21}^1 & w_{22}^1 & w_{23}^1 \\ w_{31}^1 & w_{32}^1 & w_{33}^1 \\ w_{41}^1 & w_{42}^1 & w_{43}^1 \end{bmatrix}^T \cdot \begin{bmatrix} x_{i1} \\ x_{i2} \\ x_{i3} \\ x_{i4} \end{bmatrix} + \begin{bmatrix} b_{11} \\ b_{12} \\ b_{13} \end{bmatrix} = \begin{bmatrix} z_{11} \\ z_{12} \\ z_{13} \end{bmatrix} \Rightarrow \sigma\left(\begin{bmatrix} z_{11} \\ z_{12} \\ z_{13} \end{bmatrix}\right) = \begin{bmatrix} o_{11} \\ o_{12} \\ o_{13} \end{bmatrix}$$

Forward Propagation: layer 3 / Output layer

$$z_{21} = w_{11}^2 o_{11} + w_{21}^2 o_{12} + w_{31}^2 o_{13} + b_{21}$$
$$o_{21} = \sigma(z_{21})$$

$$z_{22} = w_{12}^2 o_{11} + w_{22}^2 o_{12} + w_{32}^2 o_{13} + b_{22}$$
$$o_{22} = \sigma(z_{22})$$

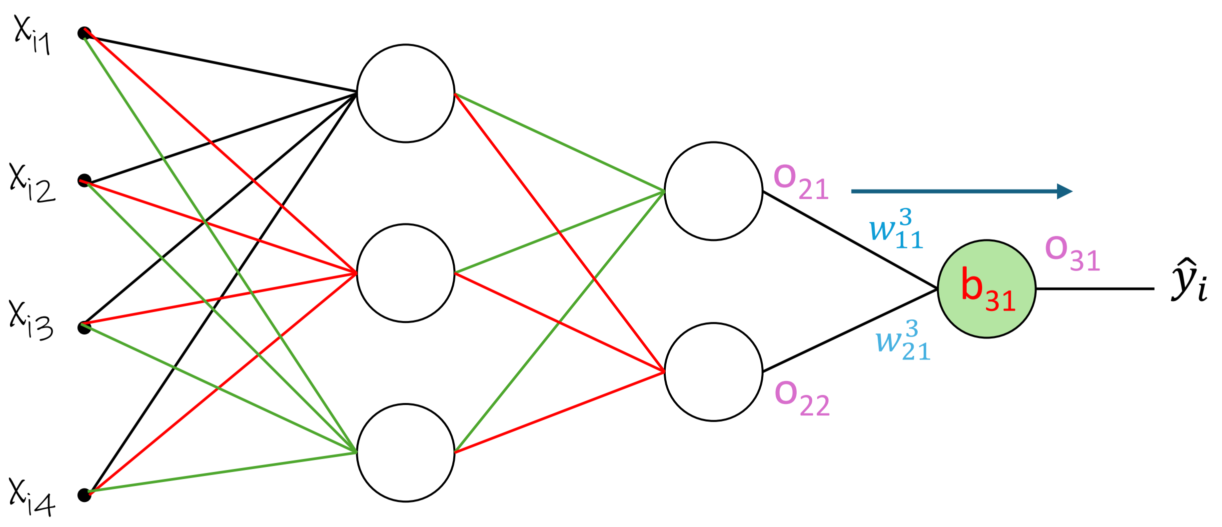


Fortunately, this can be done easily using matrix calculation

$$\begin{bmatrix} w_{11}^2 & w_{12}^2 \\ w_{21}^2 & w_{22}^2 \\ w_{31}^2 & w_{32}^2 \end{bmatrix}^T \cdot \begin{bmatrix} o_{11} \\ o_{12} \\ o_{13} \end{bmatrix} + \begin{bmatrix} b_{21} \\ b_{22} \end{bmatrix} = \begin{bmatrix} z_{21} \\ z_{22} \end{bmatrix} \quad \Longrightarrow \quad \sigma\left(\begin{bmatrix} z_{21} \\ z_{22} \end{bmatrix}\right) = \begin{bmatrix} o_{21} \\ o_{22} \end{bmatrix}$$

Forward Propagation: layer 2

$$z_{31} = w_{11}^3 o_{21} + w_{21}^3 o_{22} + b_{31}$$
$$\hat{y}_i = o_{31} = \sigma(z_{31})$$



Fortunately, this can be done easily using matrix calculation

$$\begin{bmatrix} w_{11}^3 \\ w_{21}^3 \end{bmatrix}^T \cdot \begin{bmatrix} o_{21} \\ o_{22} \end{bmatrix} + b_{31} = z_{31} \quad \Rightarrow \quad \sigma(z_{31}) = o_{31} = \hat{y}_i$$

Reference and further reading

1. “Deep Learning”, Ian Goodfellow, et al.
2. Pramoditha, Rukshan. “The Concept of Artificial Neurons (Perceptrons) in Neural Networks.” *Medium*, Towards Data Science, 29 Dec. 2021, towardsdatascience.com/the-concept-of-artificial-neurons-perceptrons-in-neural-networks-fab22249cbfc. Accessed 21 Jan. 2025.