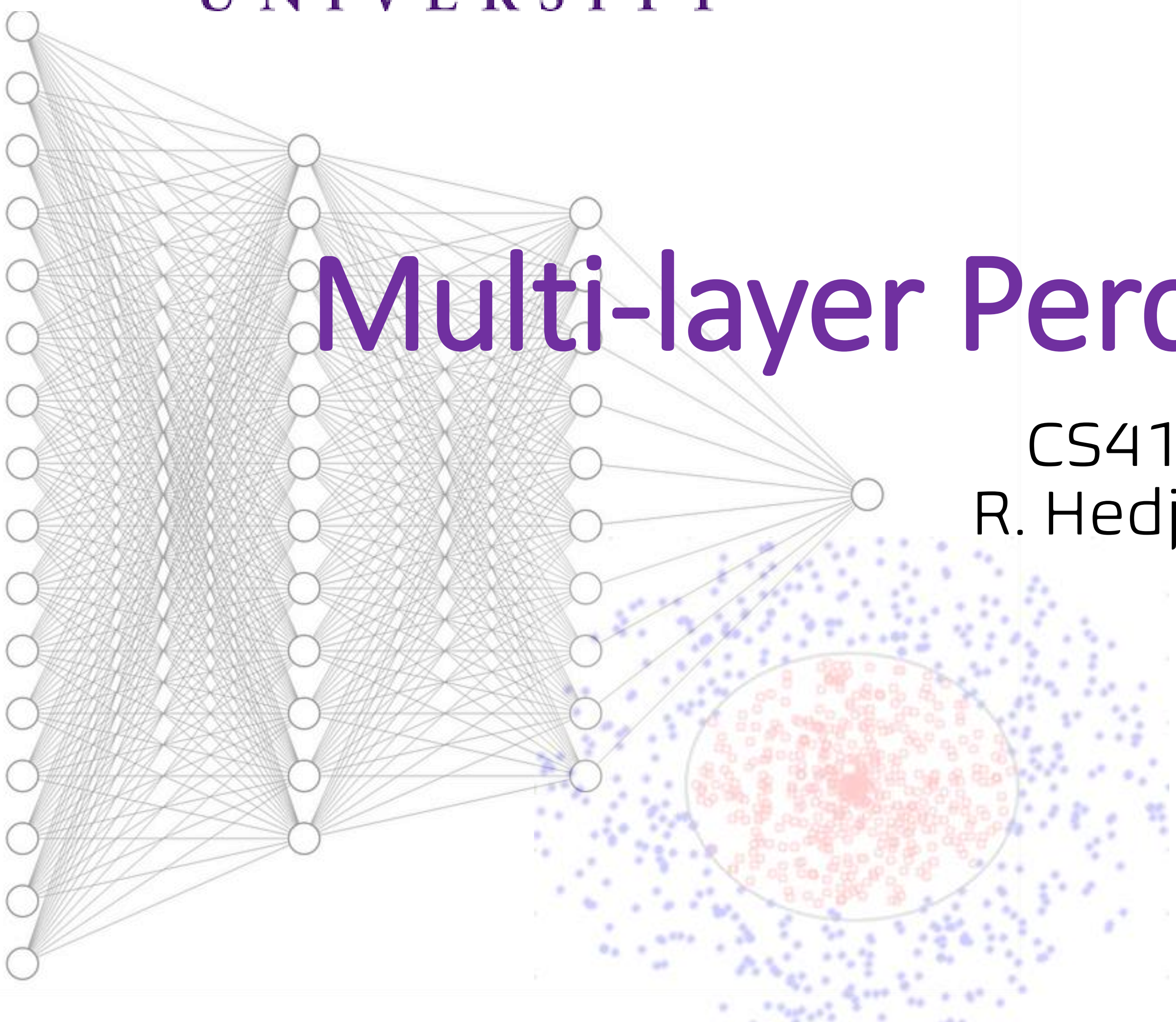


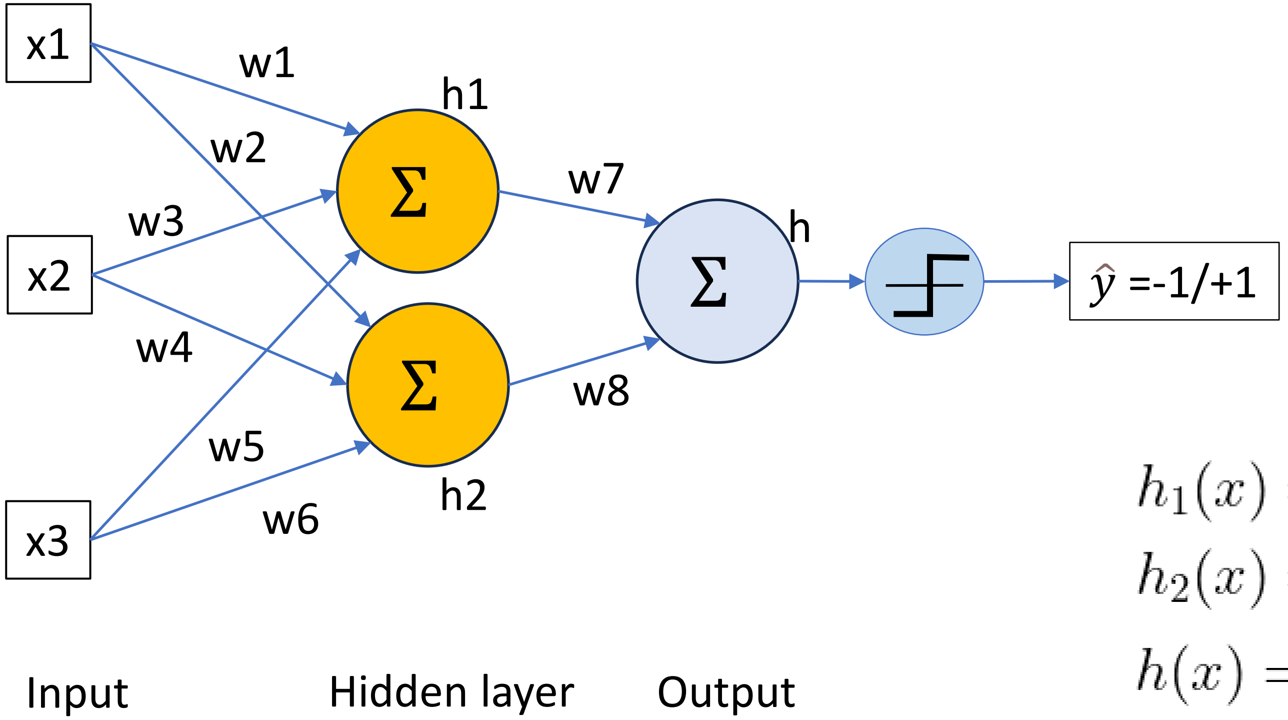
Multi-layer Perceptron (MLP)

CS417

R. Hedjam

Multi-layers perceptron

- MLP contains at least one hidden layer connecting the input to the output.



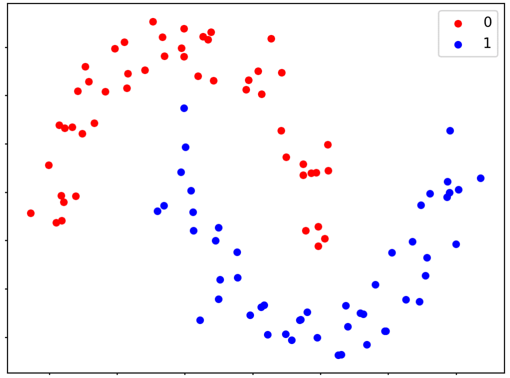
$$h_1(x) = w_1x_1 + w_3x_2 + w_5x_3$$

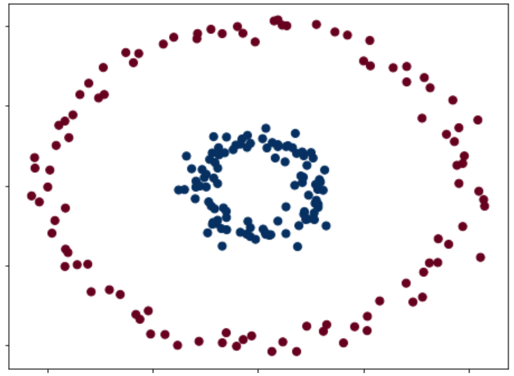
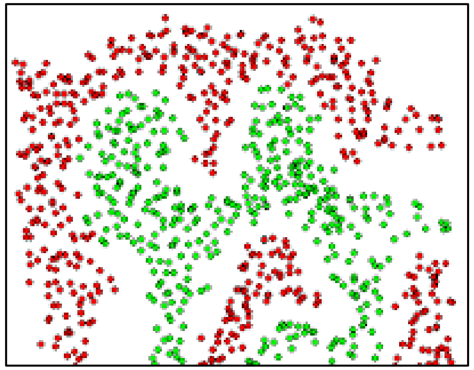
$$h_2(x) = w_2x_1 + w_4x_2 + w_6x_3$$

$$h(x) = w_7h_1 + w_8h_2$$

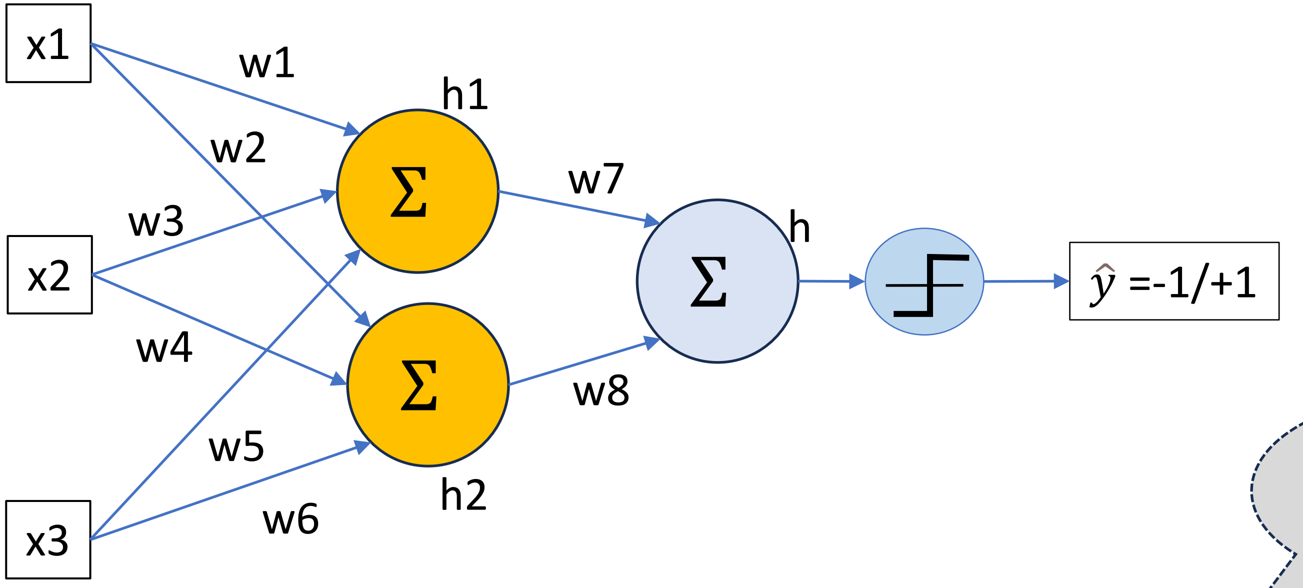
Multi-layers perceptron

- Does the MLP solve non-linear problems?
- Non-linear problems are those for which the data from different classes cannot be separated by a linear boundary decision.
- Examples:



Multi-layers perceptron



Always linear

$$h_1(x) = w_1 x_1 + w_3 x_2 + w_5 x_3$$

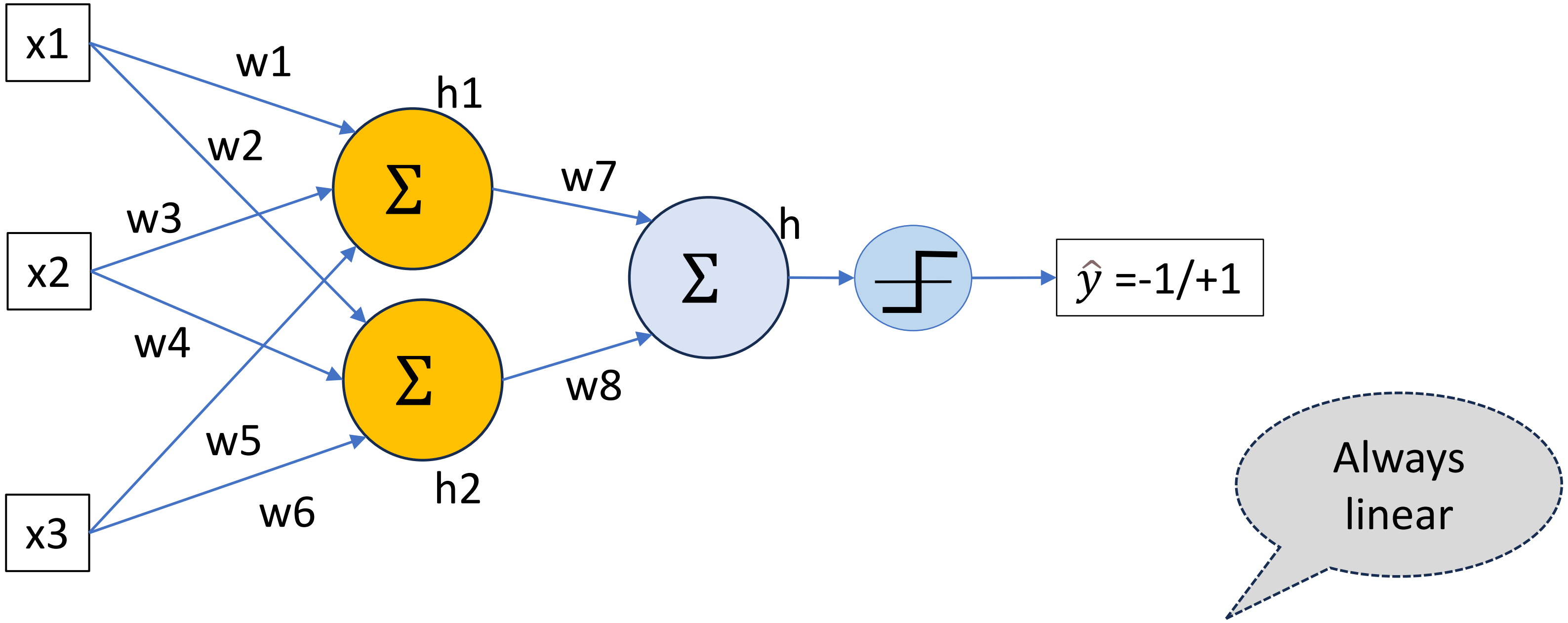
$$h_2(x) = w_2 x_1 + w_4 x_2 + w_6 x_3$$

$$h(x) = w_7 h_1 + w_8 h_2$$

$$\Rightarrow$$

$$h(x) = w_7 w_1 x_1 + w_7 w_3 x_2 + w_7 w_5 x_3 + w_8 w_2 x_1 + w_8 w_4 x_2 + w_8 w_6 x_3$$

Multi-layers perceptron



$$h_1(x) = w_1x_1 + w_3x_2 + w_5x_3$$
$$h_2(x) = w_2x_1 + w_4x_2 + w_6x_3$$
$$h(x) = w_7h_1 + w_8h_2$$

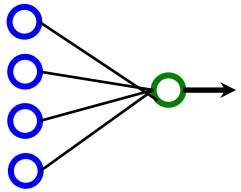
$\Rightarrow$

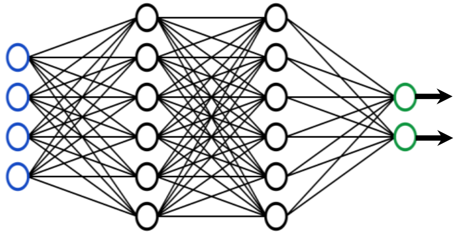
$$h(x) = w_7w_1x_1 + w_7w_3x_2 + w_7w_5x_3 + w_8w_2x_1 + w_8w_4x_2 + w_8w_6x_3$$

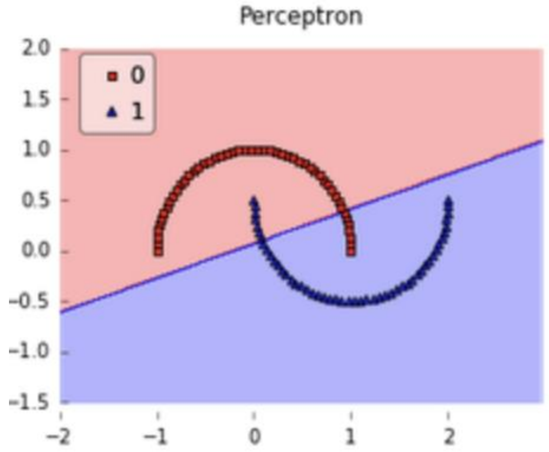
MLP with Step-wise or linear activation functions can't solve non-linear problems.

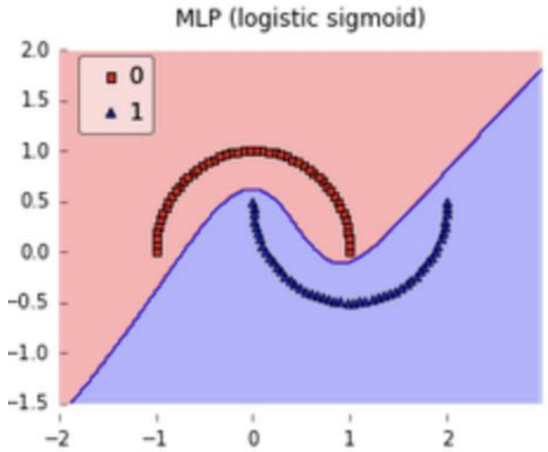
Perceptron vs MLP

- By connecting the artificial neurons in the network through non-linear activation functions and multi layers, we can create complex, non-linear decision boundaries that allow us to tackle problems where the different classes are not linearly separable.







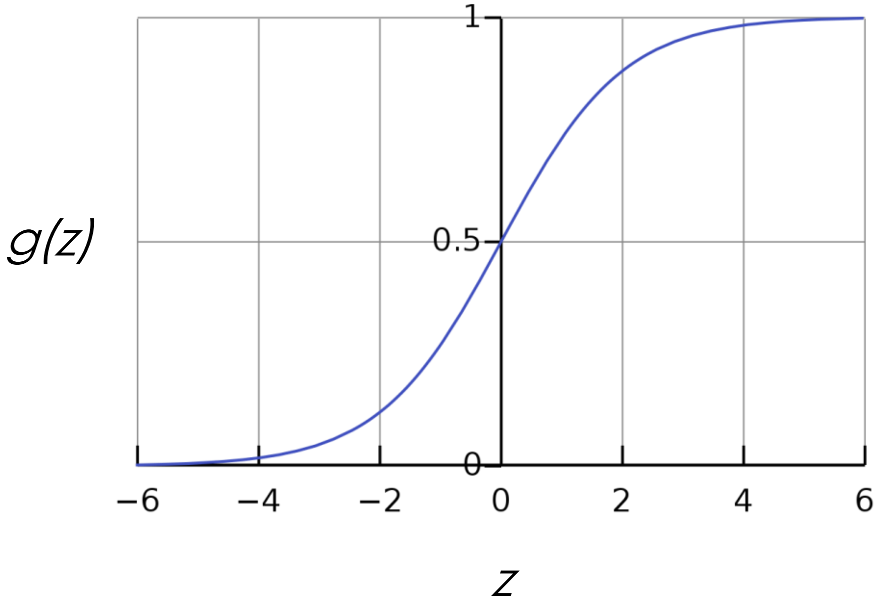


Deal with non-linearity – Sigmoid function

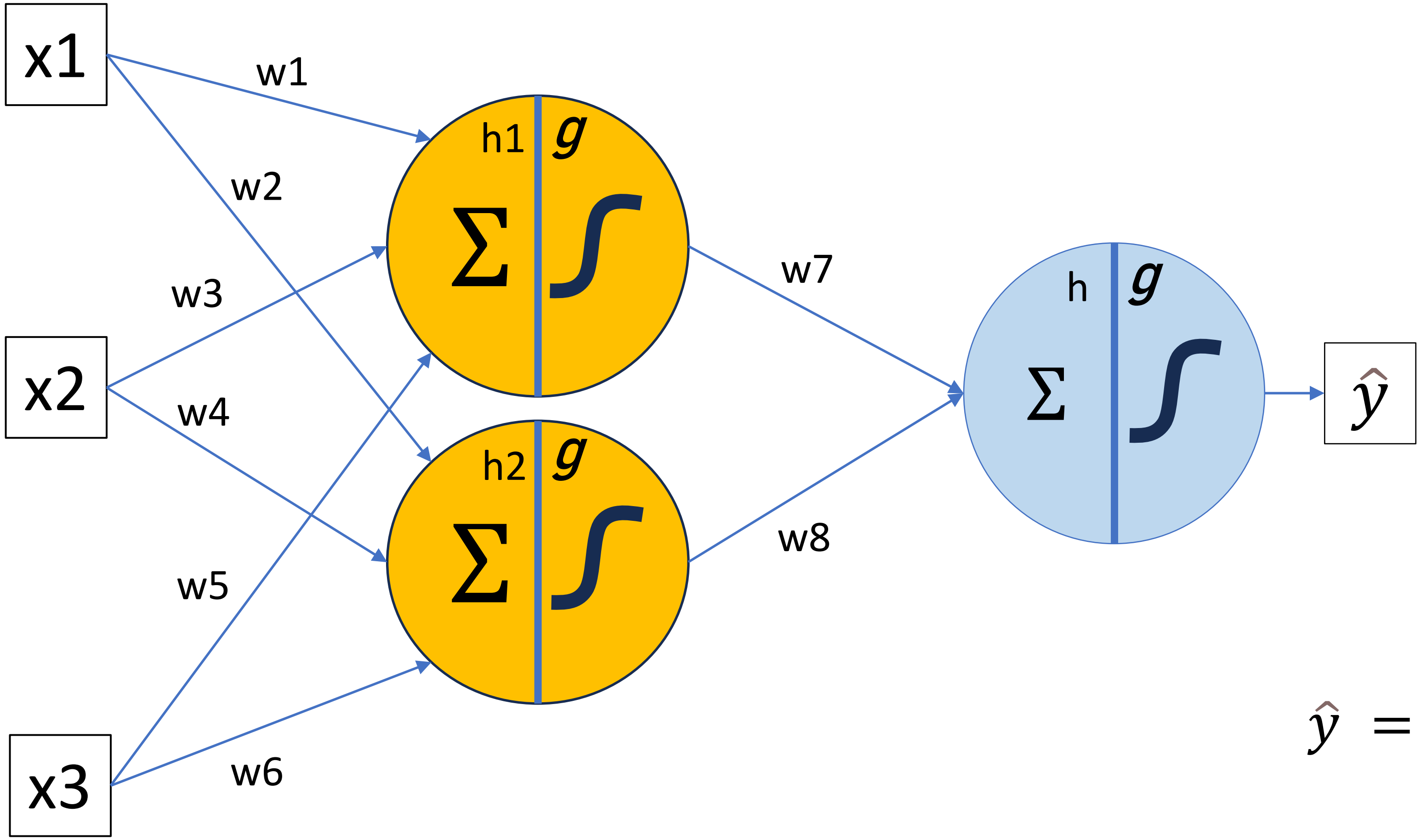
$$g(z) = \frac{1}{1 + e^{-z}}$$

$$\text{let } z = h(\mathbf{x}) = \mathbf{w}^\top \mathbf{x} + w_0$$

$$\begin{aligned} g(h(\mathbf{x})) &= \frac{1}{1 + e^{-h(\mathbf{x})}} \\ &= \frac{1}{1 + e^{-(\mathbf{w}^\top \mathbf{x} + w_0)}} \end{aligned}$$

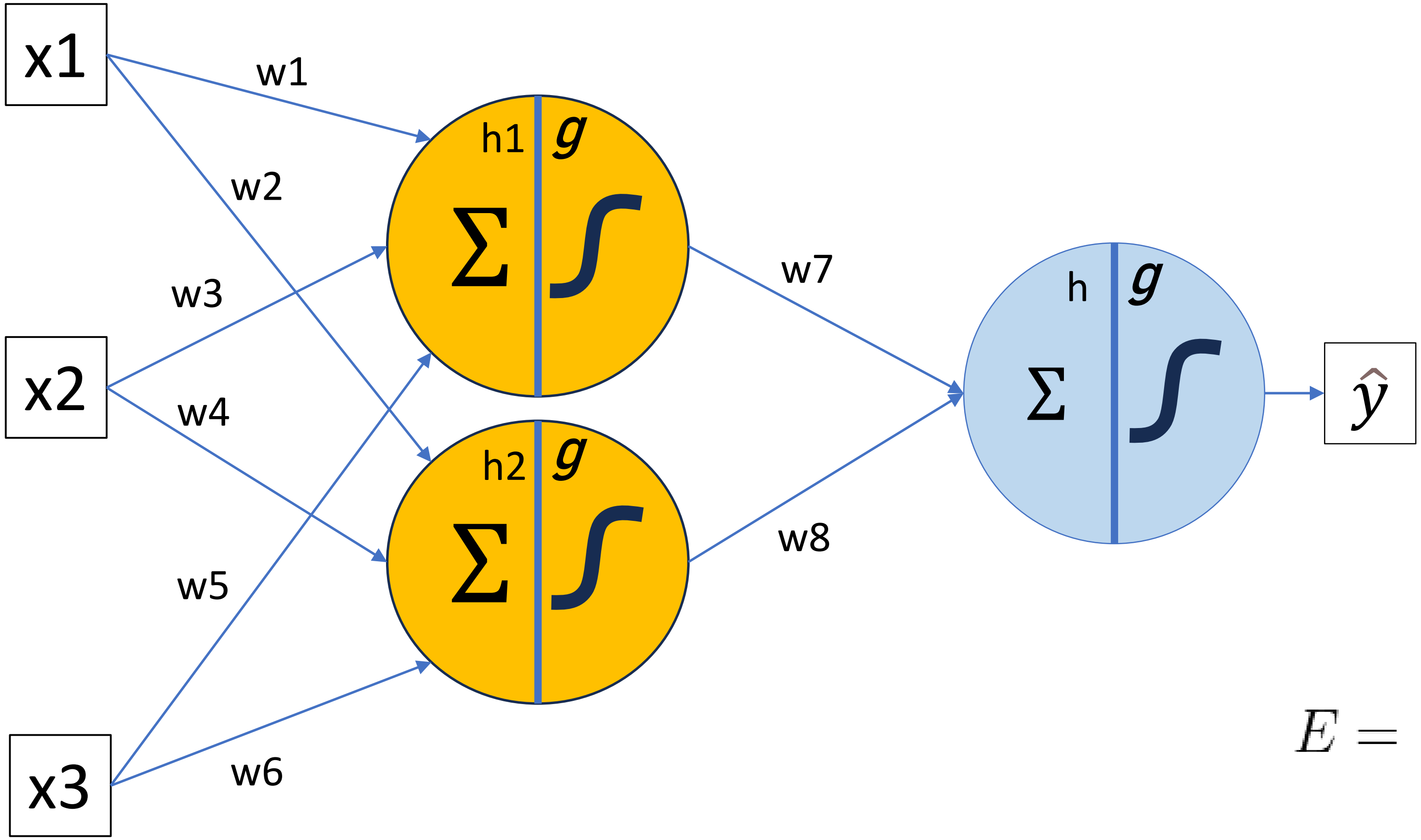


MLP with Sigmoid activation function



$$\hat{y} = \begin{cases} -1, & g(h) < 0.5 \\ +1, & g(h) \geq 0.5 \end{cases}$$

MLP – Prediction Error

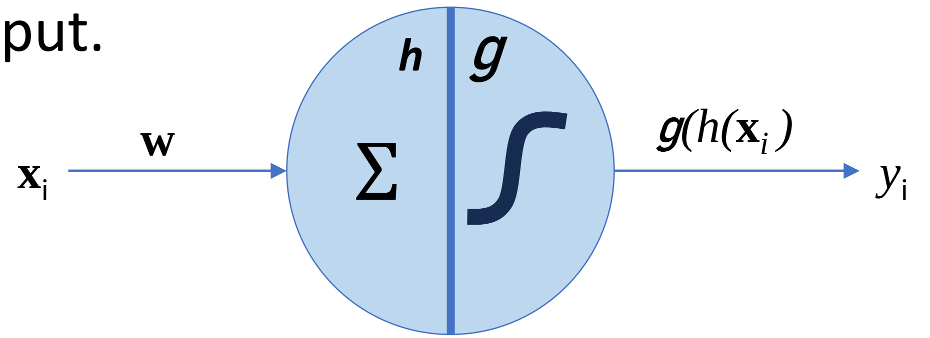


$$E = \frac{1}{2}\sum_{i=1}^{N}\Big(g(h(\mathbf{x}_i)) - y_i\Big)^2$$

Error: MLP with Sigmoid activation function

- Error between the predicted output and the true output.

$$E = \frac{1}{2}\sum_{i=1}^{N}\Big(g(h(\mathbf{x}_i)) - y_i\Big)^2$$



- How does the change in **w** affect the change in ***E***?

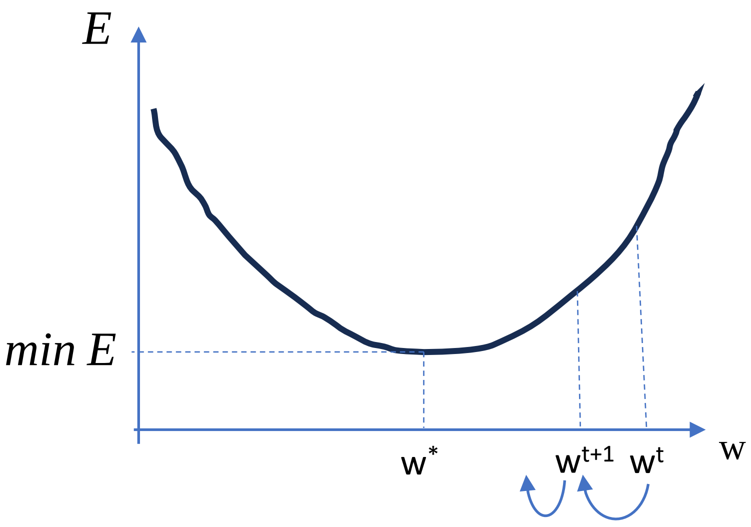
  → Compute the derivative of E with respect to w.

$$\frac{\partial E(\mathbf{w})}{\partial \mathbf{w}} = \frac{\partial E(\mathbf{w})}{\partial g(h(\mathbf{x}))}.\frac{\partial g(h(\mathbf{x}))}{\partial \mathbf{w}}$$

$$\frac{\partial E(\mathbf{w})}{\partial g(h(\mathbf{x}))} = g(h(x)) - y$$

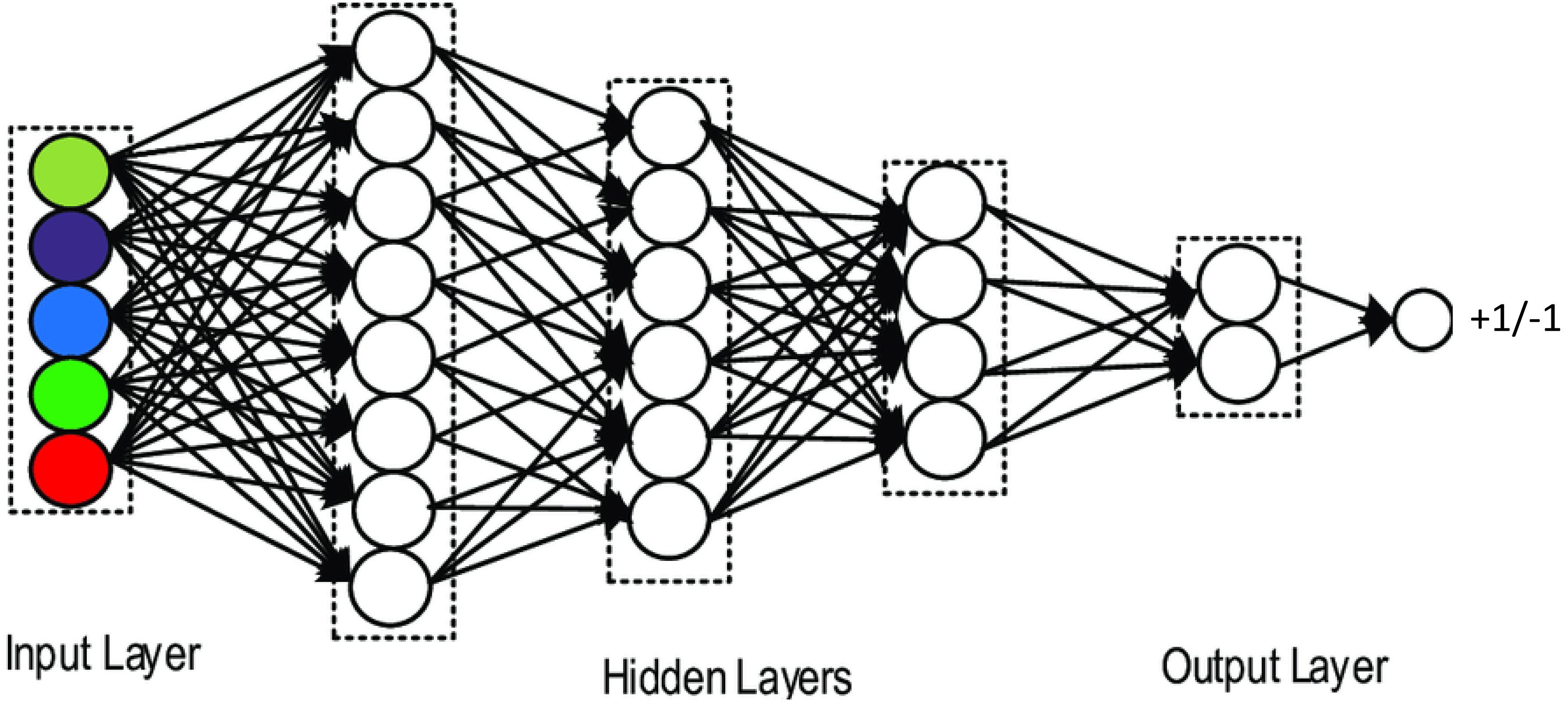
$$\frac{\partial g(h(\mathbf{x}))}{\partial \mathbf{w}} = \frac{1}{1+e^{-(\mathbf{w}^\top\mathbf{x}+w_0)}}.(1 - \frac{1}{1+e^{-(\mathbf{w}^\top\mathbf{x}+w_0)}})$$

$$\mathbf{w}^{t+1} = \mathbf{w}^t - \eta\frac{\partial E(\mathbf{w}^t)}{\partial \mathbf{w}^t}$$ (Updating weights)



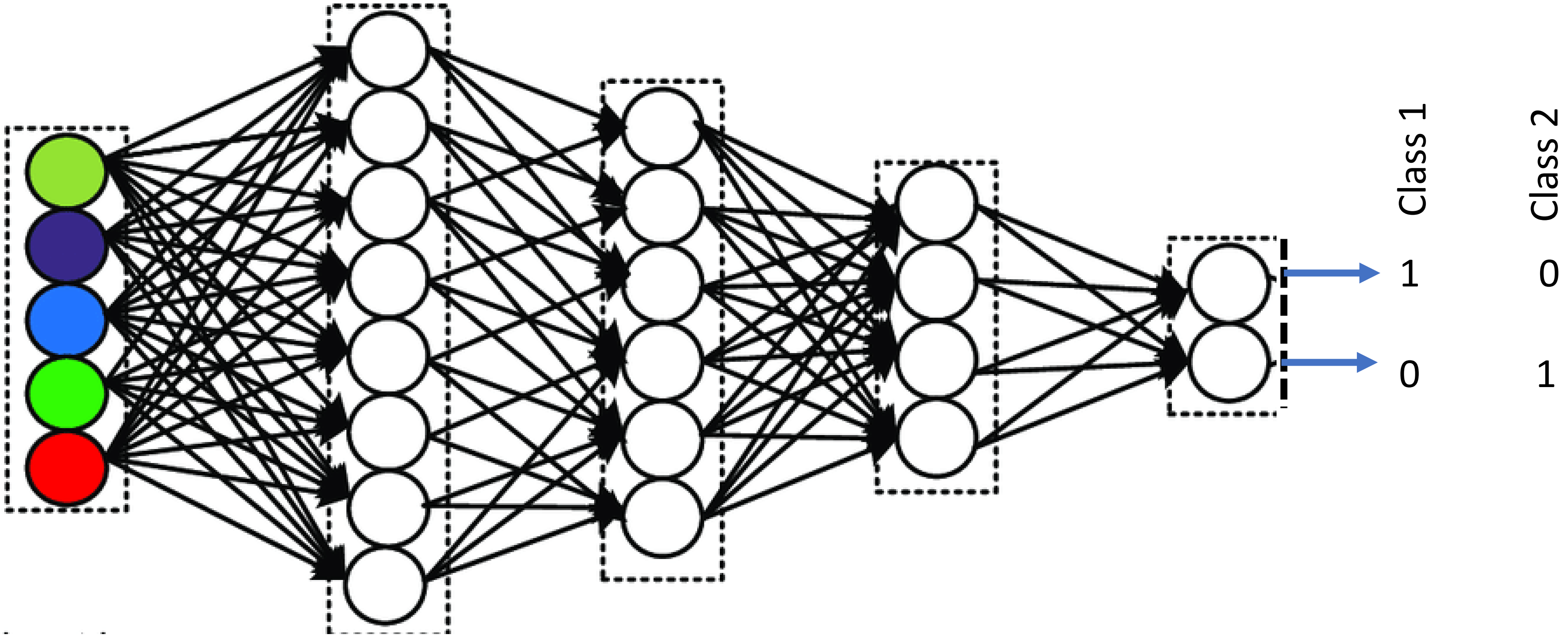
MLP – binary classification

- Predict either +1 or -1



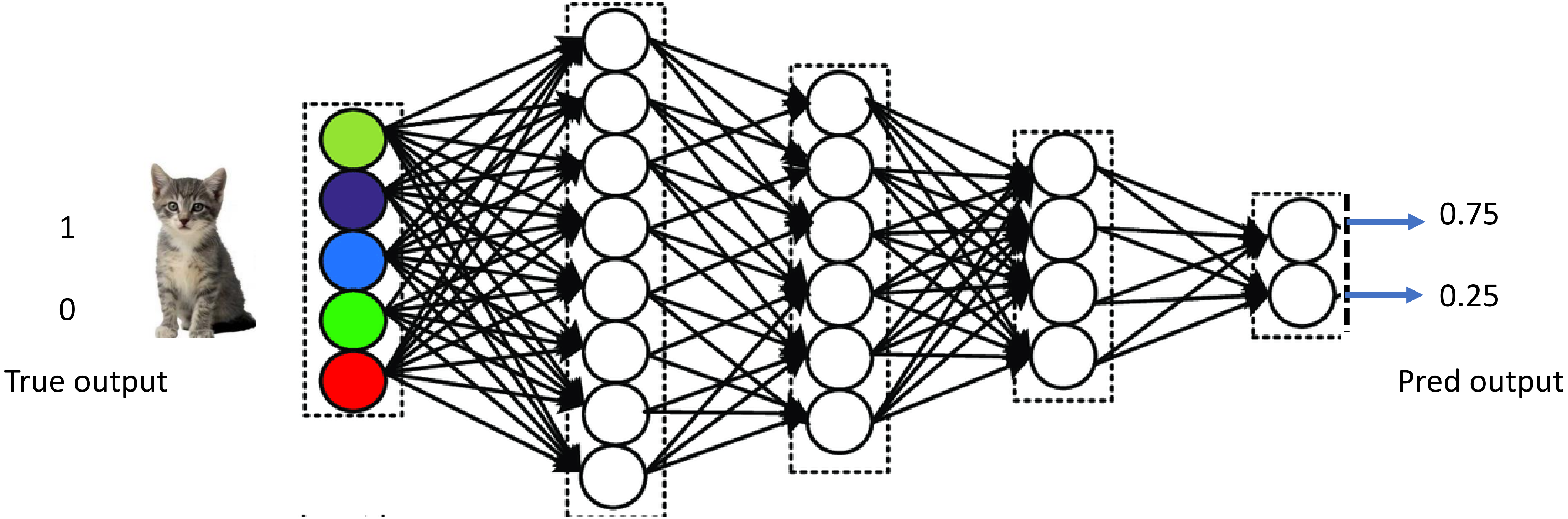
DOI:10.3390/su11051449

MLP – binary classification – hot encoding



DOI:10.3390/su11051449

Example

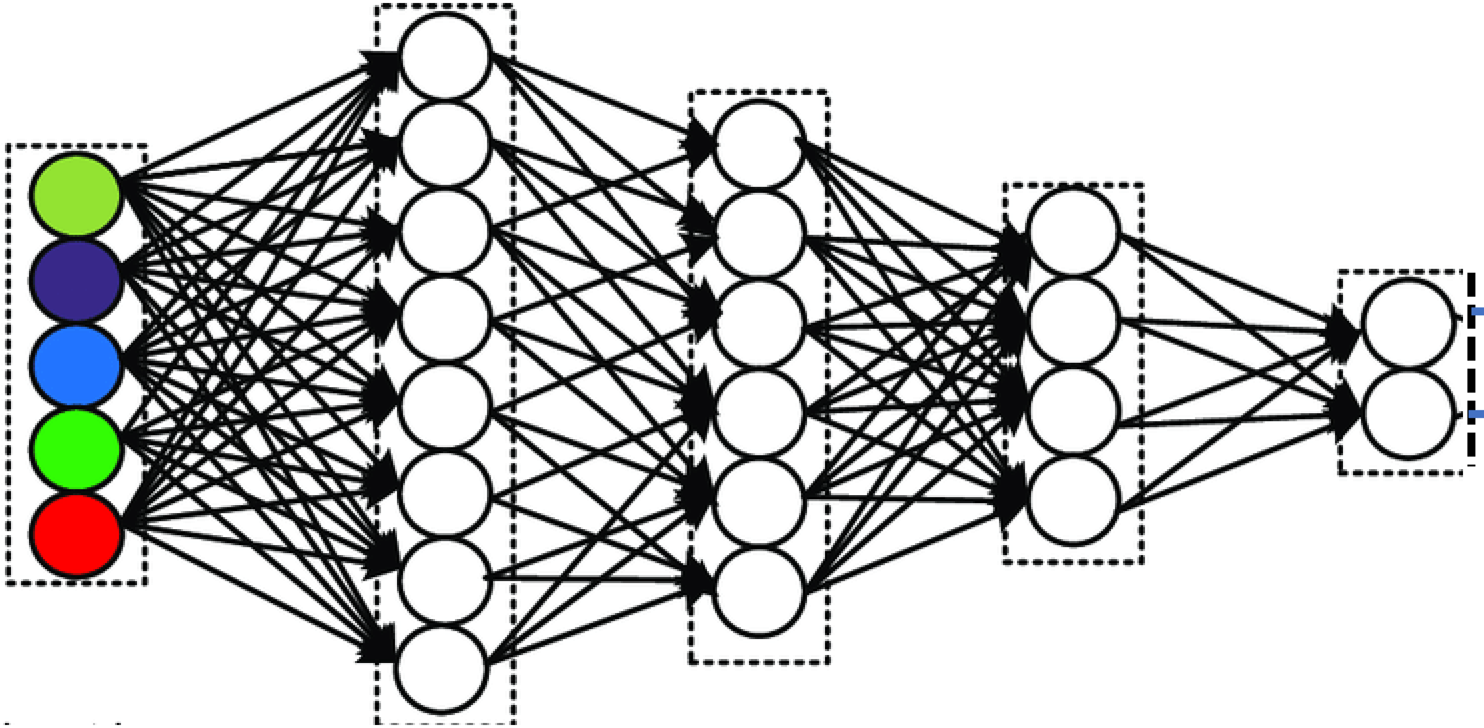


Example

0

1

True output

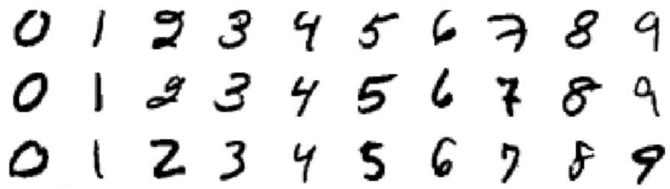


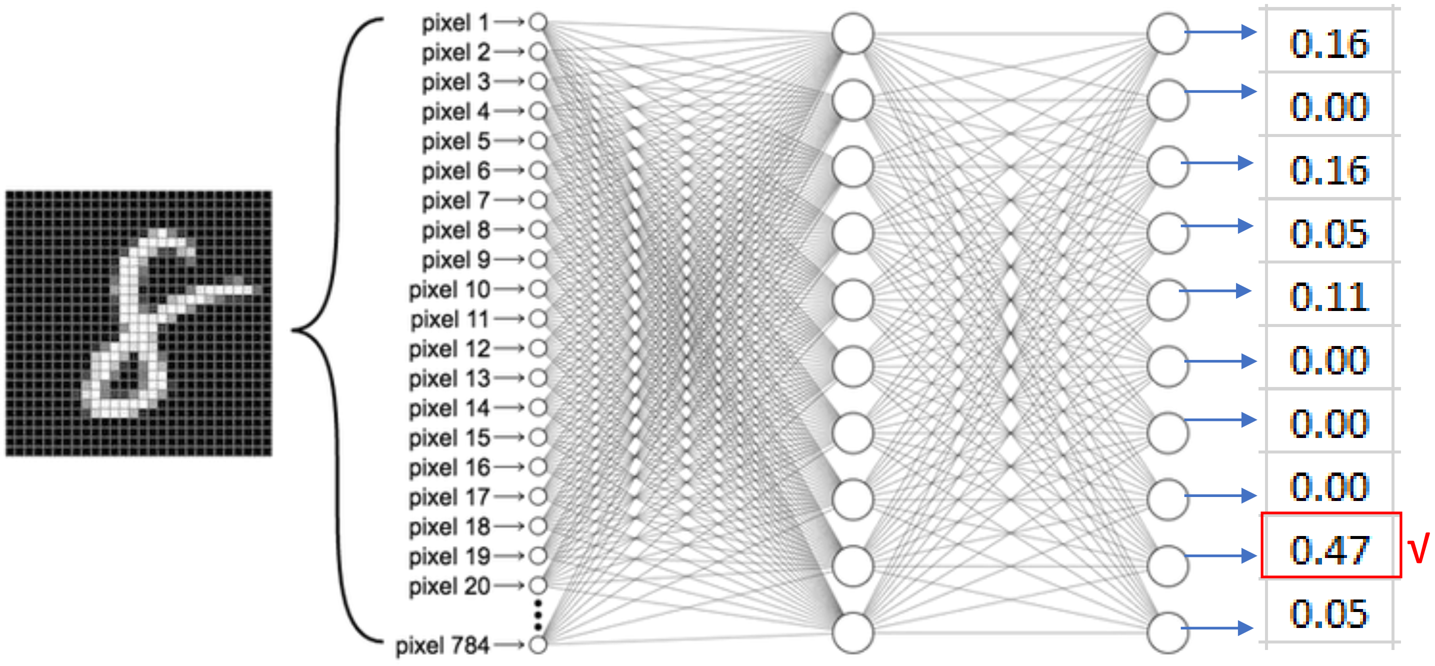
0.20

0.80

Pred output

Example – Digit classification



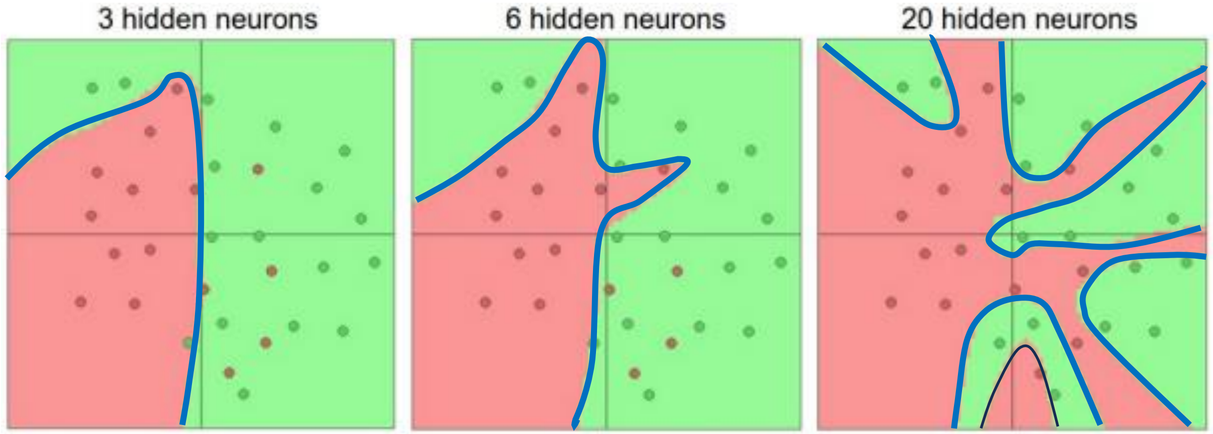


0	1	2	3	4	5	6	7	8	9	10
1	0	0	0	0	0	0	0	0	0	1
0	1	0	0	0	0	0	0	0	0	0
0	0	1	0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	0	0	0	0
0	0	0	0	1	0	0	0	0	0	0
0	0	0	0	0	1	0	0	0	0	0
0	0	0	0	0	0	1	0	0	0	0
0	0	0	0	0	0	0	1	0	0	0
0	0	0	0	0	0	0	0	1	0	0
0	0	0	0	0	0	0	0	0	1	0

https://deeplearningfromscratch.wordpress.com/

Universal approximation theory

- A Neural network with at least one hidden layer and sigmoid activation functions can approximate any function (Cybenco).



From Ethan Fetaya et al.

A visual demo

http://playground.tensorflow.org/