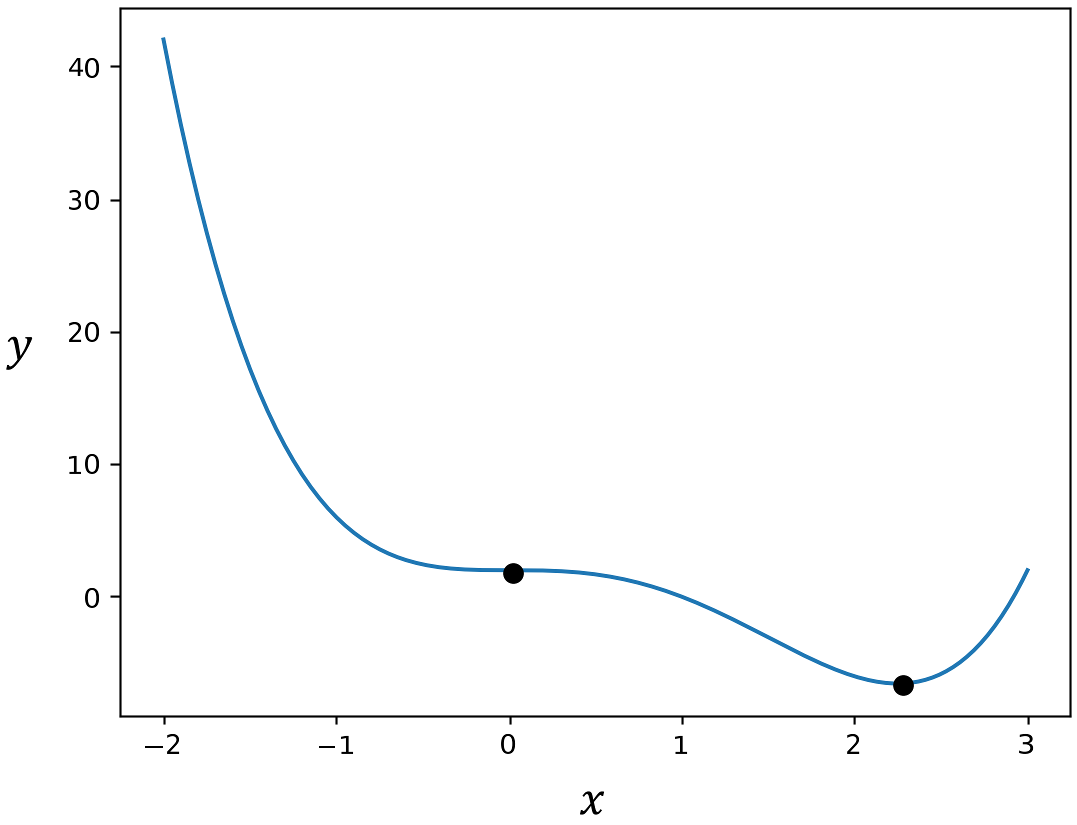
CLIMATE 405
Lecture 10 – Backpropagation and Multilayer Perceptron

Mohammed Ombadi
Assistant Professor
Climate and Space Sciences and Engineering
University of Michigan

Optimization

Optimization: selecting the best value of variables that will minimize (or maximize) a specific function.

$$y = f(x) = x^4 - 3x^3 + 2$$



For the function on the left, we can find the optimum values of x that minimize the function **analytically**.

$$f'(x) = 4x^3 - 9x^2$$

$$0 = 4x^3 - 9x^2$$

$$4x^3 = 9x^2$$

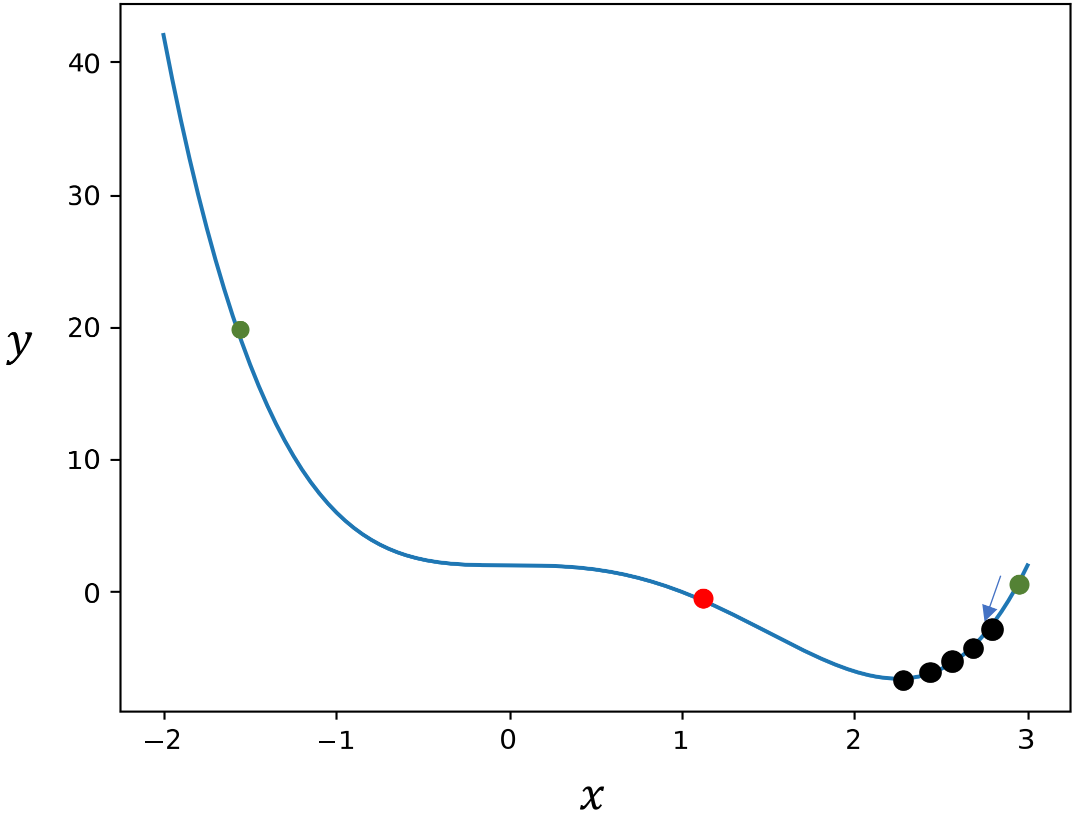
$$\frac{x^3}{x^2} = x = \frac{9}{4} = \mathbf{2.25}$$

$x = 0$ is a local minimum.

Gradient Descent

Gradient Descent: an algorithm that uses the gradient (derivative of a function) to reach a global (local) minimum.

$$y = f(x) = x^4 - 3x^3 + 2$$



Gradient Descent

<u>Inputs:</u> x_init, learning rate, num_iterations

<u>Algorithm steps:</u>

x = x_init

for i in range(num_iterations):

  calculate $f'(x)$

  x = x – [learning rate*$f'(x)$]

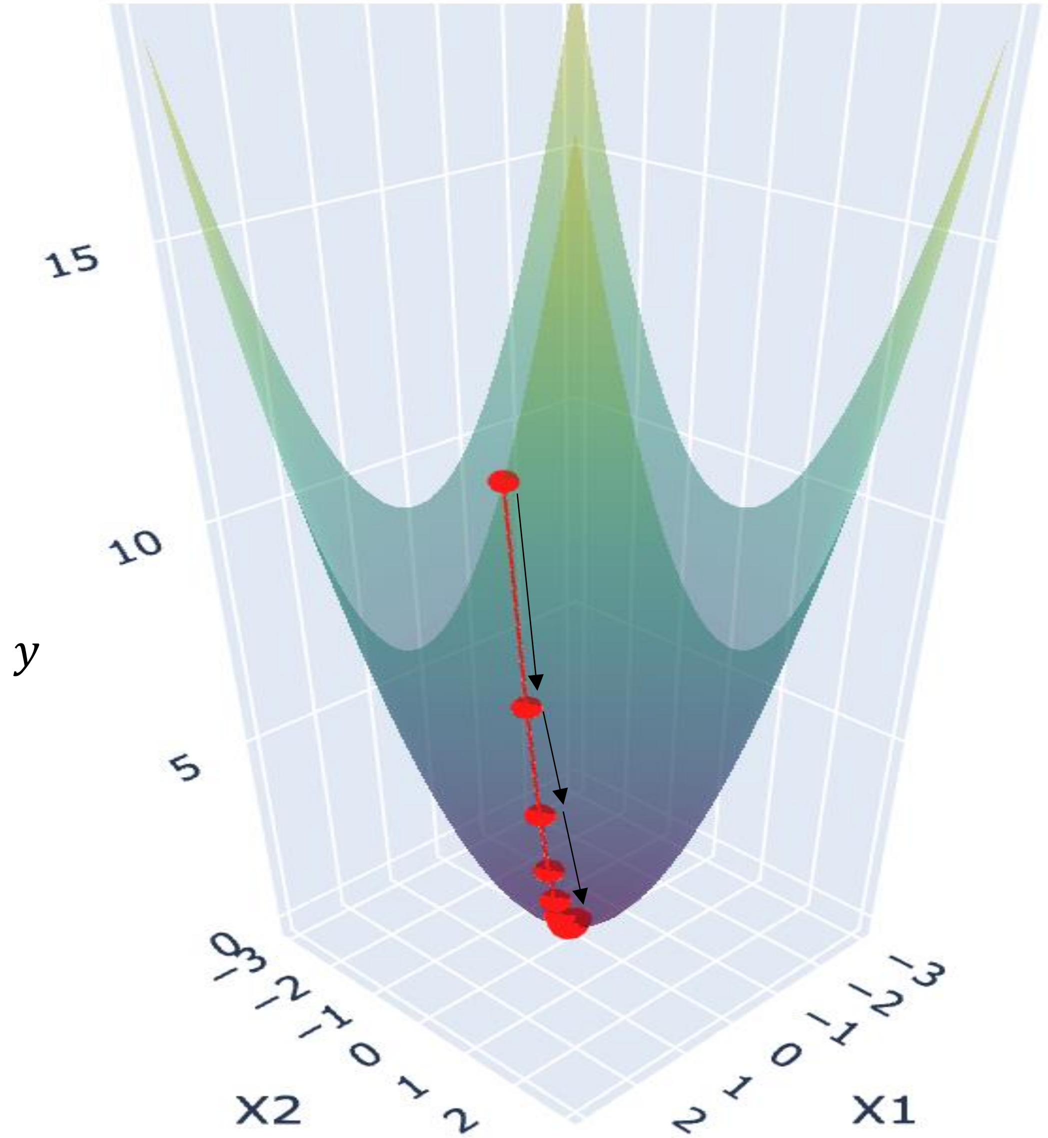
x_opt = x_new

In this case, the optimization problem is:

$$\arg\min_{x \in \mathbb{R}} f(x)$$

Gradient Descent in 3D

$y = f(x_1\, x_2) = x_1^2 + x_2^2$



In this case, the optimization problem is:

$$\arg\min_{x_1, x_2 \in \mathbb{R}} f(x_1, x_2)$$

In the context of machine learning, the optimization problem is:

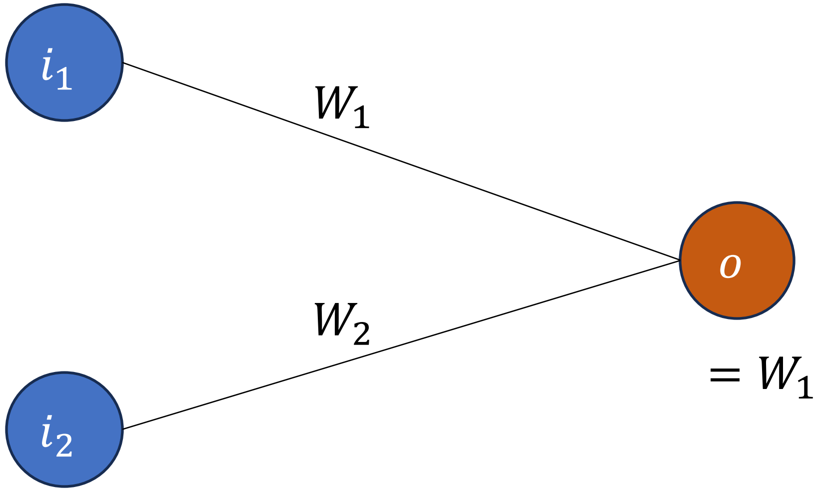
$$\arg\min_{W, b \in \mathbb{R}} f^*(W, b, x, y)$$

Where $f^*(W, b, X, y) = \sum[y - f(W, b, X)]^2$

Architecture of a Neural Network: Single Layer Perceptron

Input Layer

Output Layer

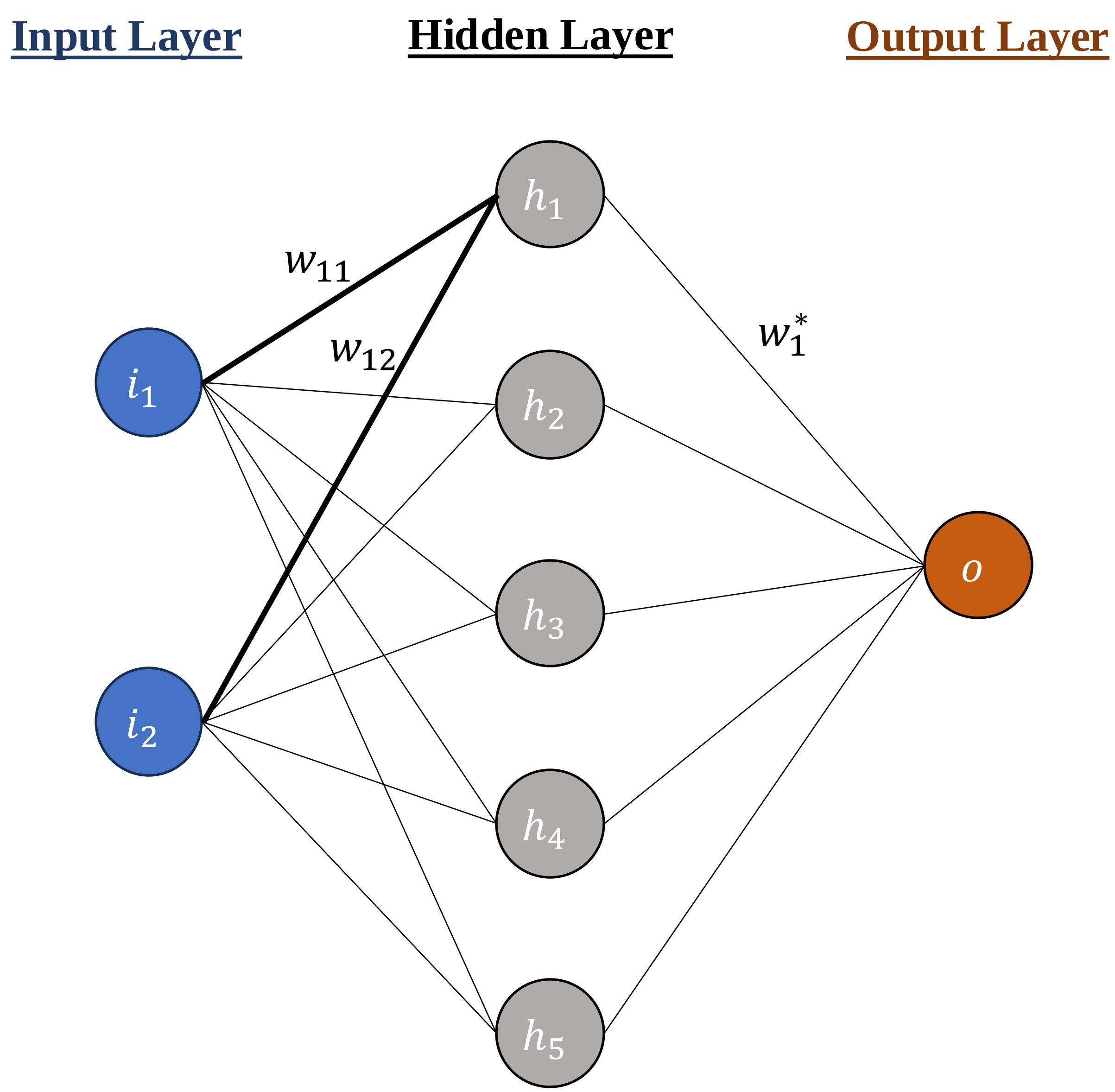


$= W_1 * i_1 + W_2 * i_2 + b$

In matrix format:

$$[o] = \begin{bmatrix} w_1 & w_2 \end{bmatrix} \begin{bmatrix} i_1 \\ i_2 \end{bmatrix} + b$$

Architecture of a Neural Network: Multilayer Perceptron (MLP)

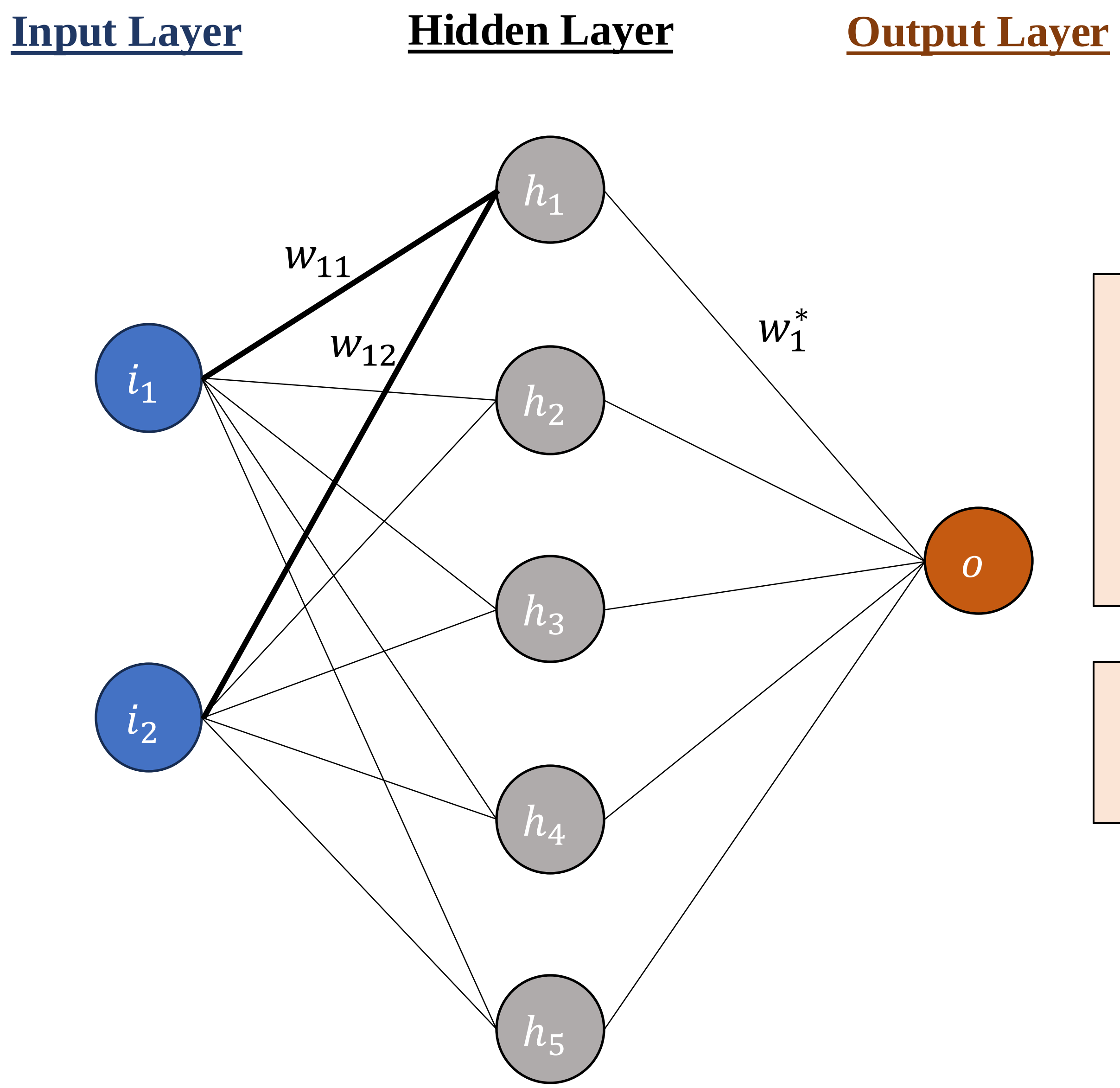


In matrix format:

$$\begin{bmatrix} h_1 \\ h_2 \\ h_3 \\ h_4 \\ h_5 \end{bmatrix} = \begin{bmatrix} w_{11} & w_{12} \\ w_{21} & w_{22} \\ w_{31} & w_{32} \\ w_{41} & w_{42} \\ w_{51} & w_{52} \end{bmatrix} \begin{bmatrix} i_1 \\ i_2 \end{bmatrix} + \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \end{bmatrix}$$

$$[o] = \begin{bmatrix} w_1^* & w_2^* & w_3^* & w_4^* & w_5^* \end{bmatrix} \begin{bmatrix} h_1 \\ h_2 \\ h_3 \\ h_4 \\ h_5 \end{bmatrix} + b^*$$

MLP: Number of Parameters

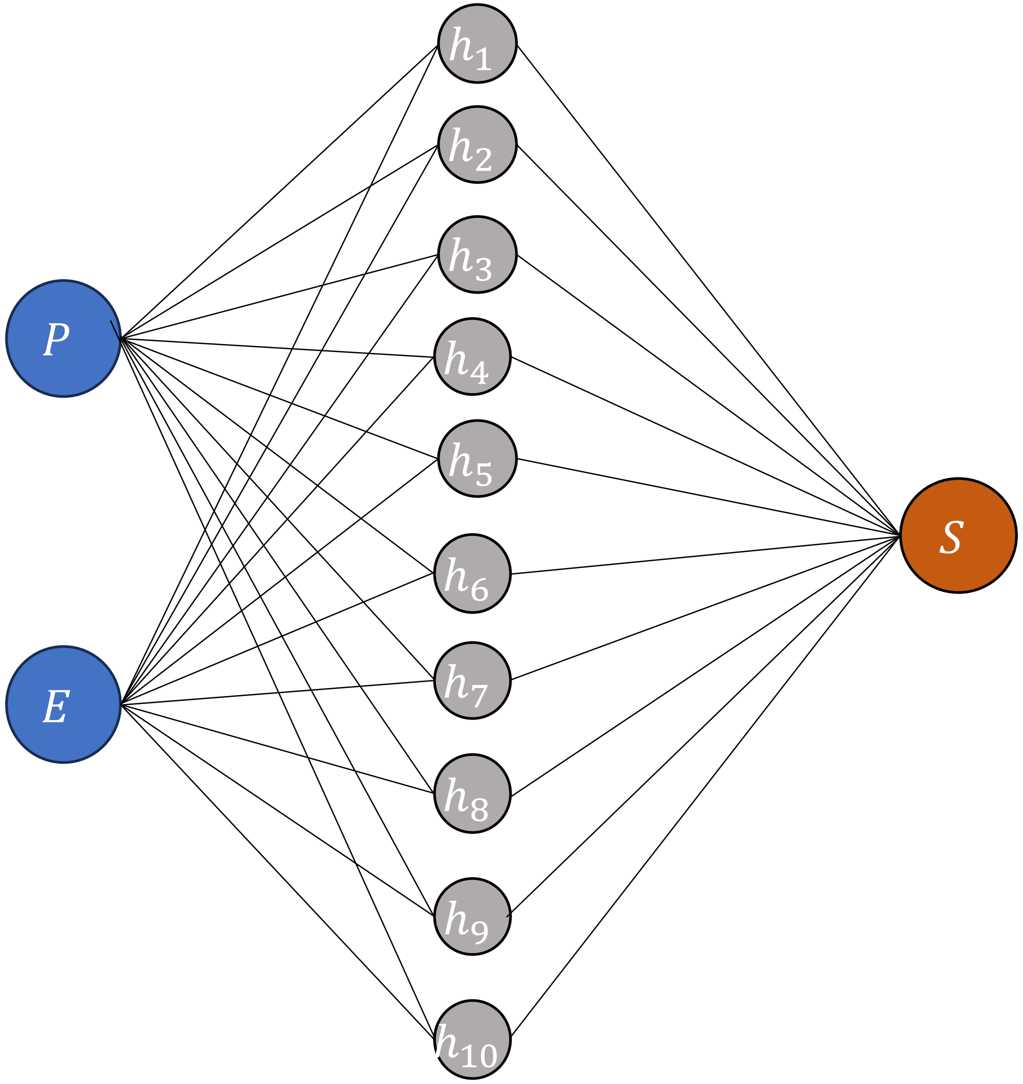


$$\begin{aligned} &Number\ of\ parameters \\ &= \left(\#Inputs \times \#Nodes_{hidden-layer}\right) \\ &+ \left(\#Nodes_{hidden-layer} \times \#\,Outputs\right) \\ &+ \#\mathrm{Nodes}_{hidden-layer} + \#Outputs \end{aligned}$$

$$\begin{aligned} &Number\ of\ parameters \\ &= (2 \times 5) + (5 \times 1) + 5 + 1 = 21 \end{aligned}$$

MLPs for different applications (Quiz – Lecture 9)

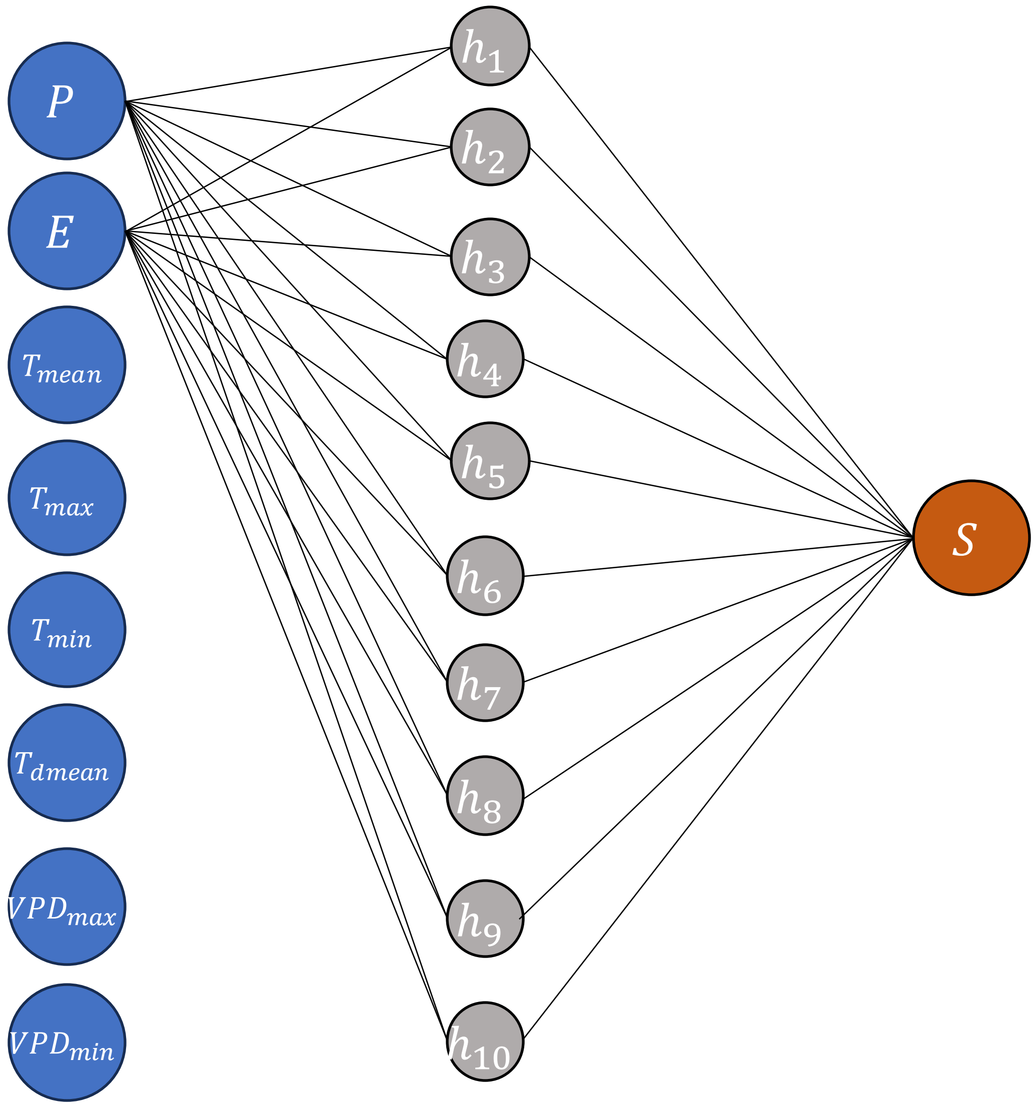
partial_correlation_data_annual.csv



$$Number\ of\ parameters$$
$$= (2 \times\ 10) + (10 \times\ 1) + 10 + 1 = 41$$

MLPs for different applications (Quiz – Lecture 9)

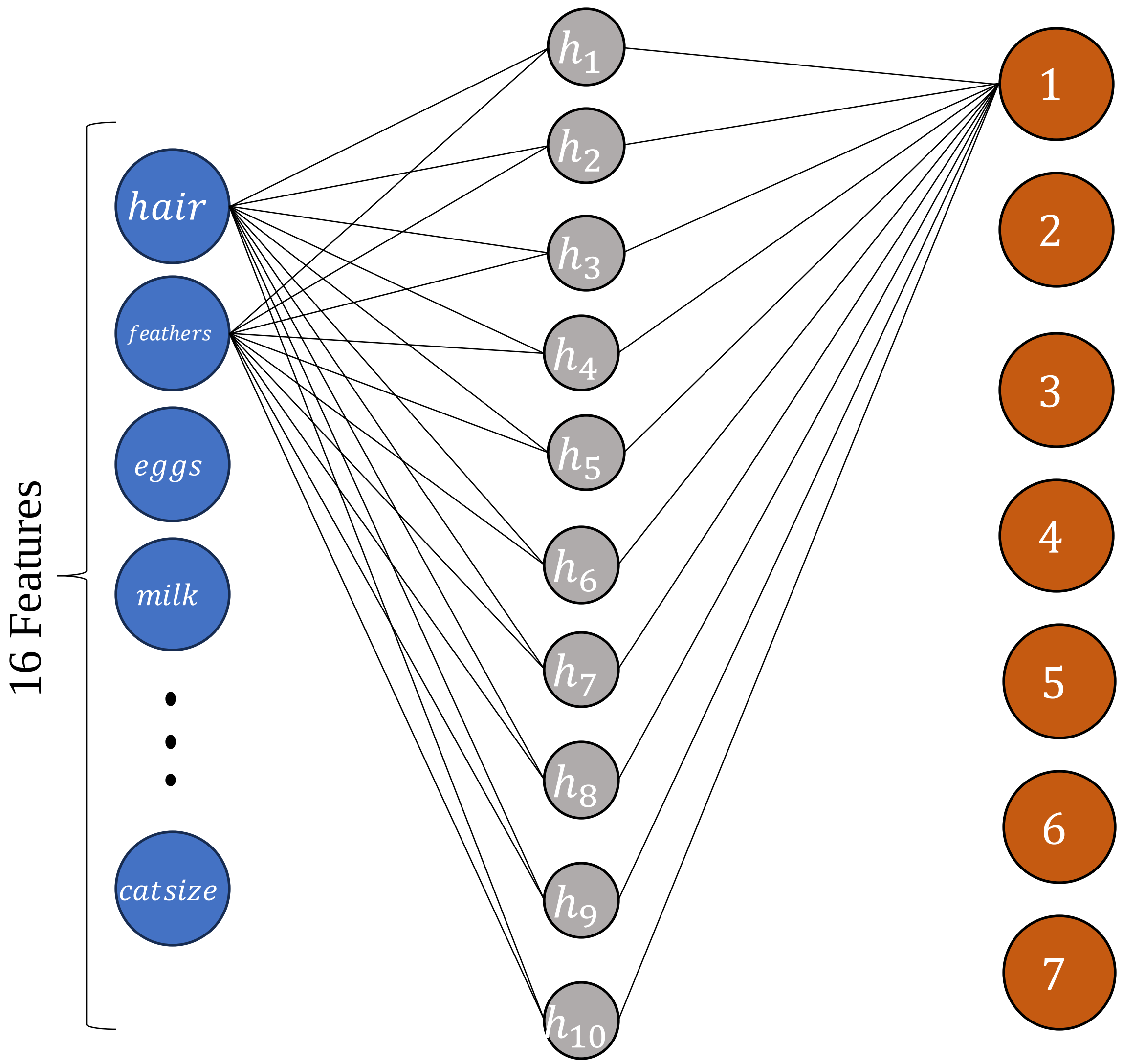
monthly_meteo_streamflow.csv



$$Number\ of\ parameters = (8\times\ 10) + (10\times\ 1) + 10 + 1 = 101$$

MLPs for different applications (Quiz – Lecture 9)

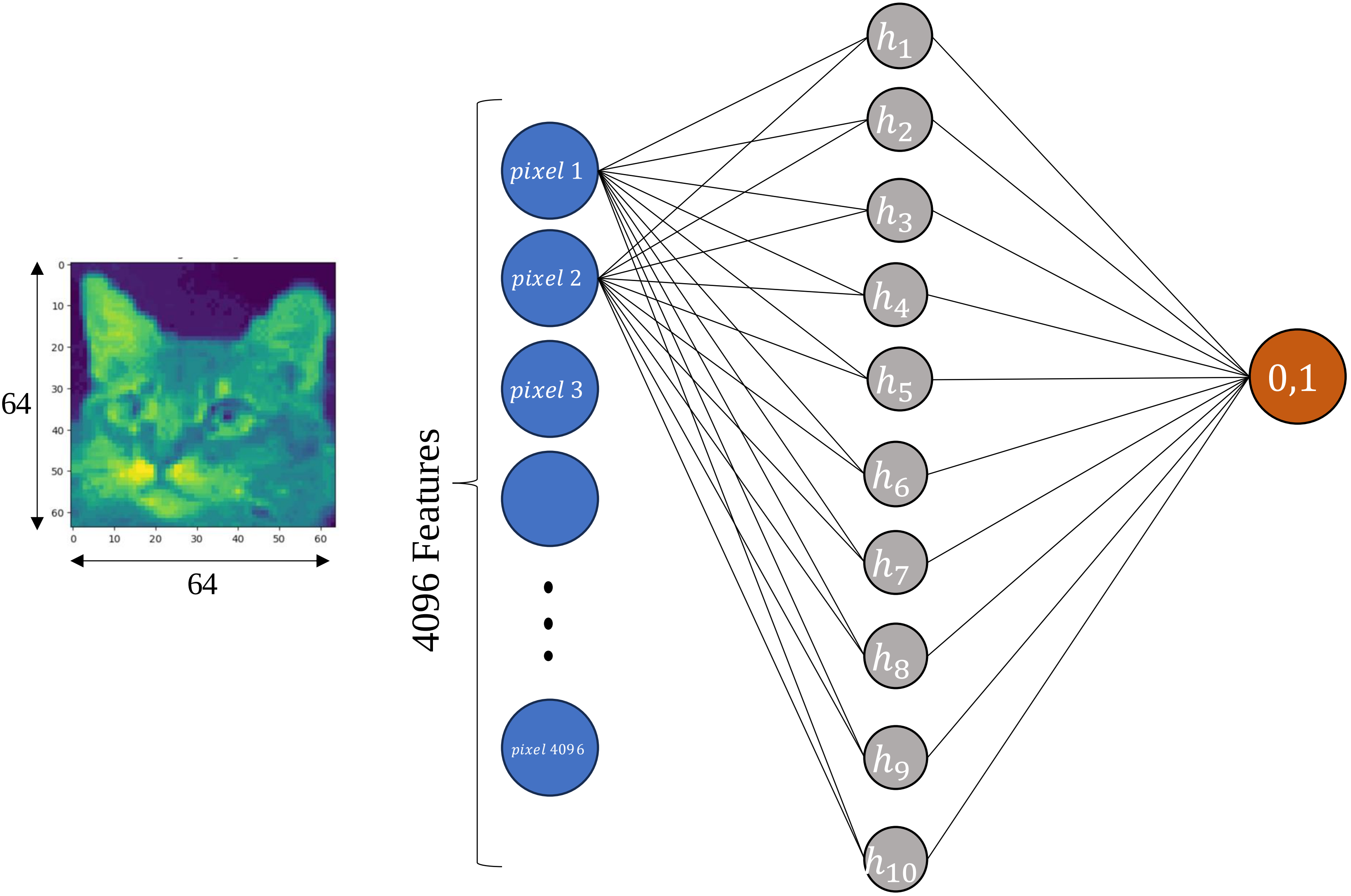
zoo_data.csv



$$Number\ of\ parameters = (16\times\ 10) + (10\times\ 7) + 10 + 7 = 247$$

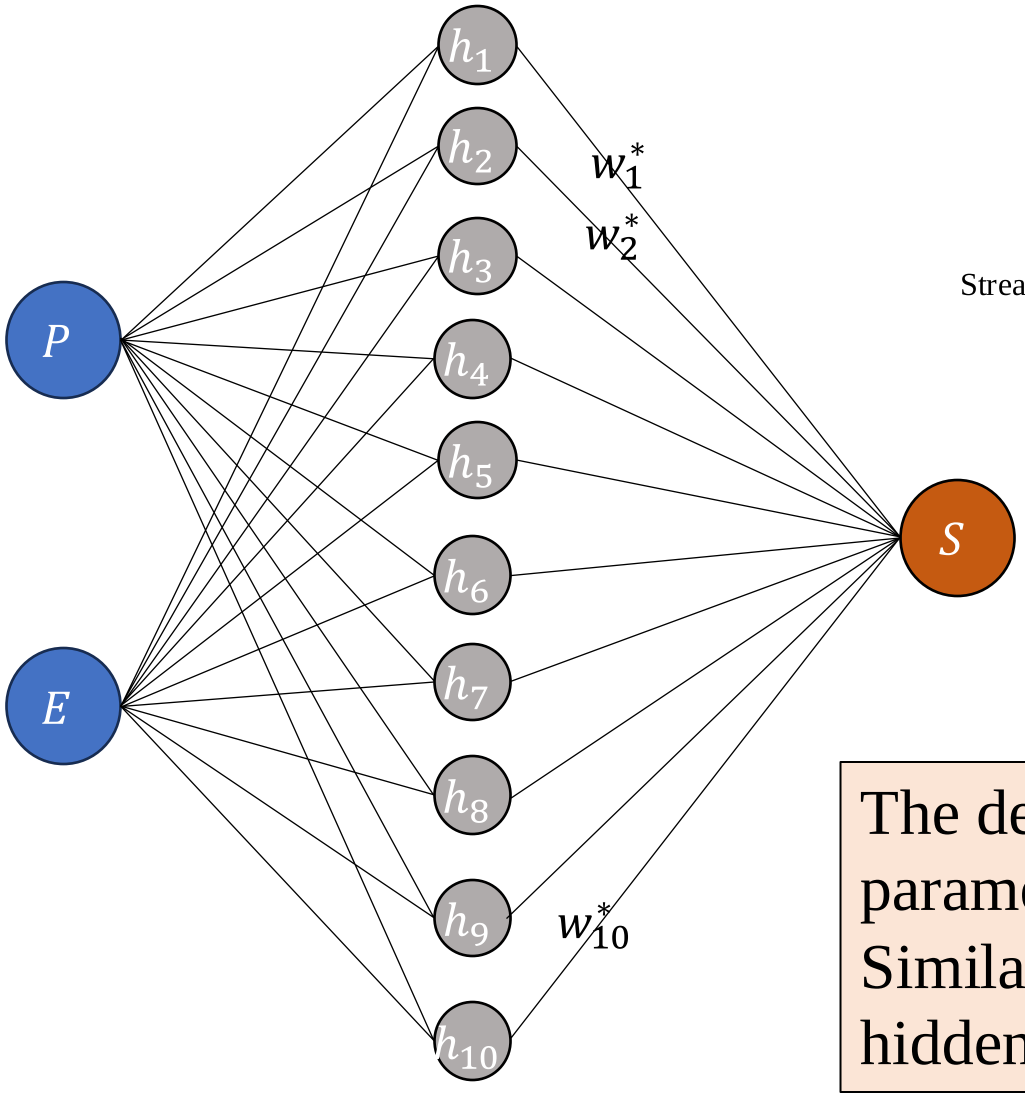
MLPs for different applications (Quiz – Lecture 9)

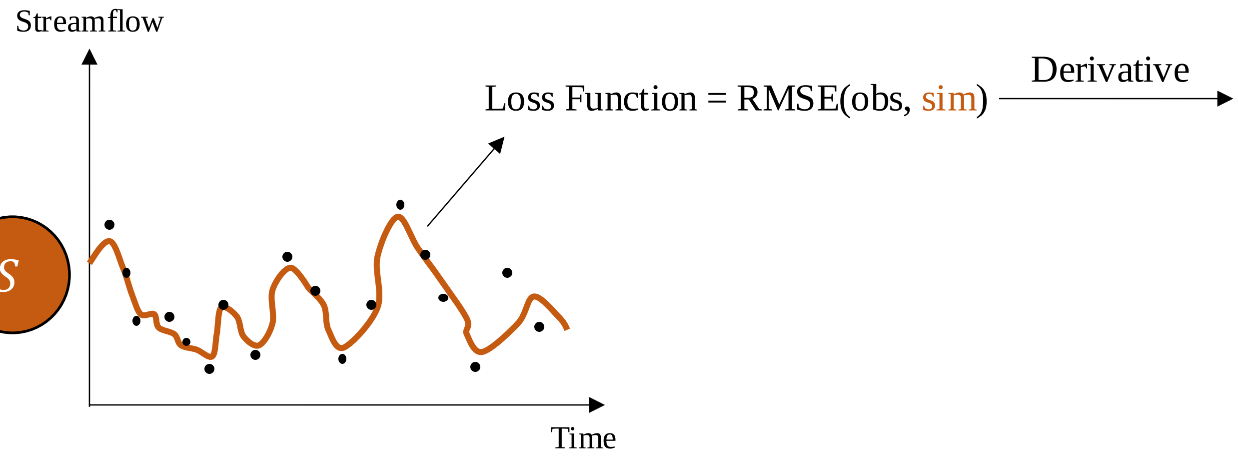
partial_correlation_data_annual.csv



$$Number\ of\ parameters = (4096 * 10) + (10 \times\ 1) + 10 + 1 = 40{,}981$$

Training of ML Models: Backpropagation



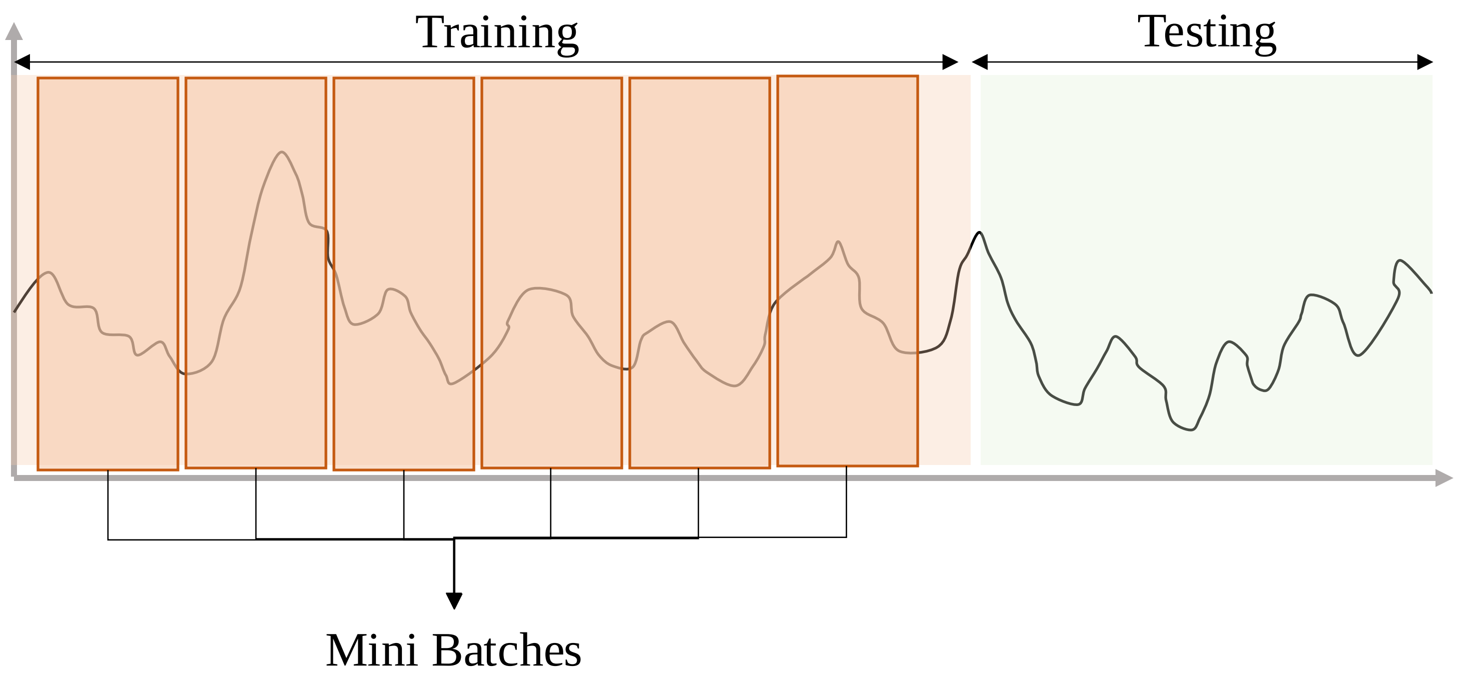


$$\begin{bmatrix} \frac{\partial RMSE}{\partial w_1^*} \\ \frac{\partial RMSE}{\partial w_2^*} \\ \frac{\partial RMSE}{\partial w_3^*} \\ \frac{\partial RMSE}{\partial w_4^*} \\ \frac{\partial RMSE}{\partial w_5^*} \\ \frac{\partial RMSE}{\partial w_6^*} \\ \vdots \\ \frac{\partial RMSE}{\partial w_{10}^*} \end{bmatrix}$$

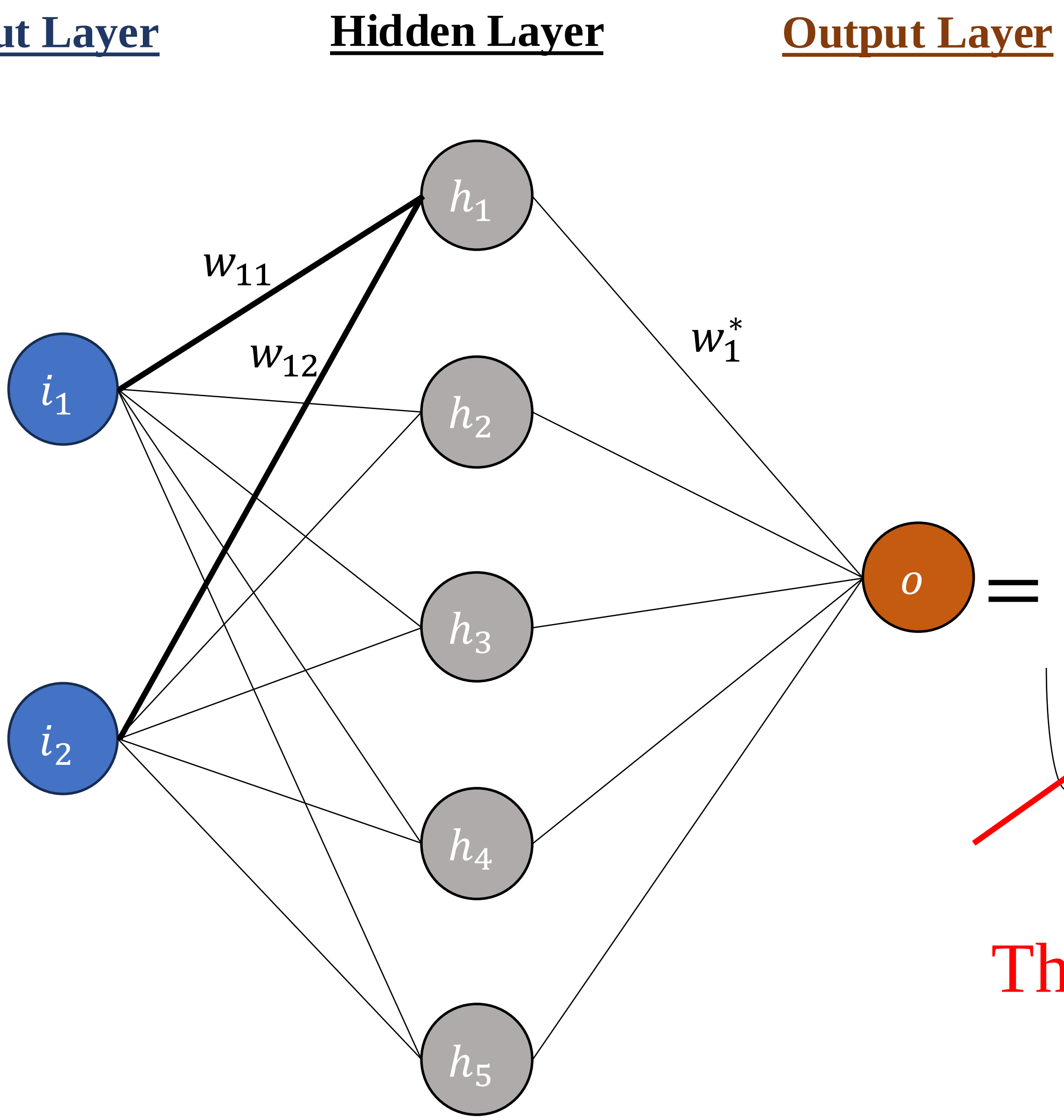
The derivative is used to update the values of the parameters from the hidden to the output layer. Similarly, the weights from the input to the hidden layer are also updated.

Training of ML Models: Stochastic Gradient Descent

Stochastic Gradient Descent: a variant of the stochastic gradient optimization algorithm, where the gradient is calculated from *a subset of the training data* instead of the entire training dataset.



Activation Functions



$$= \underbrace{\left(\sum_{i=1}^{n} w_i^* \; h_i + b\right)}$$

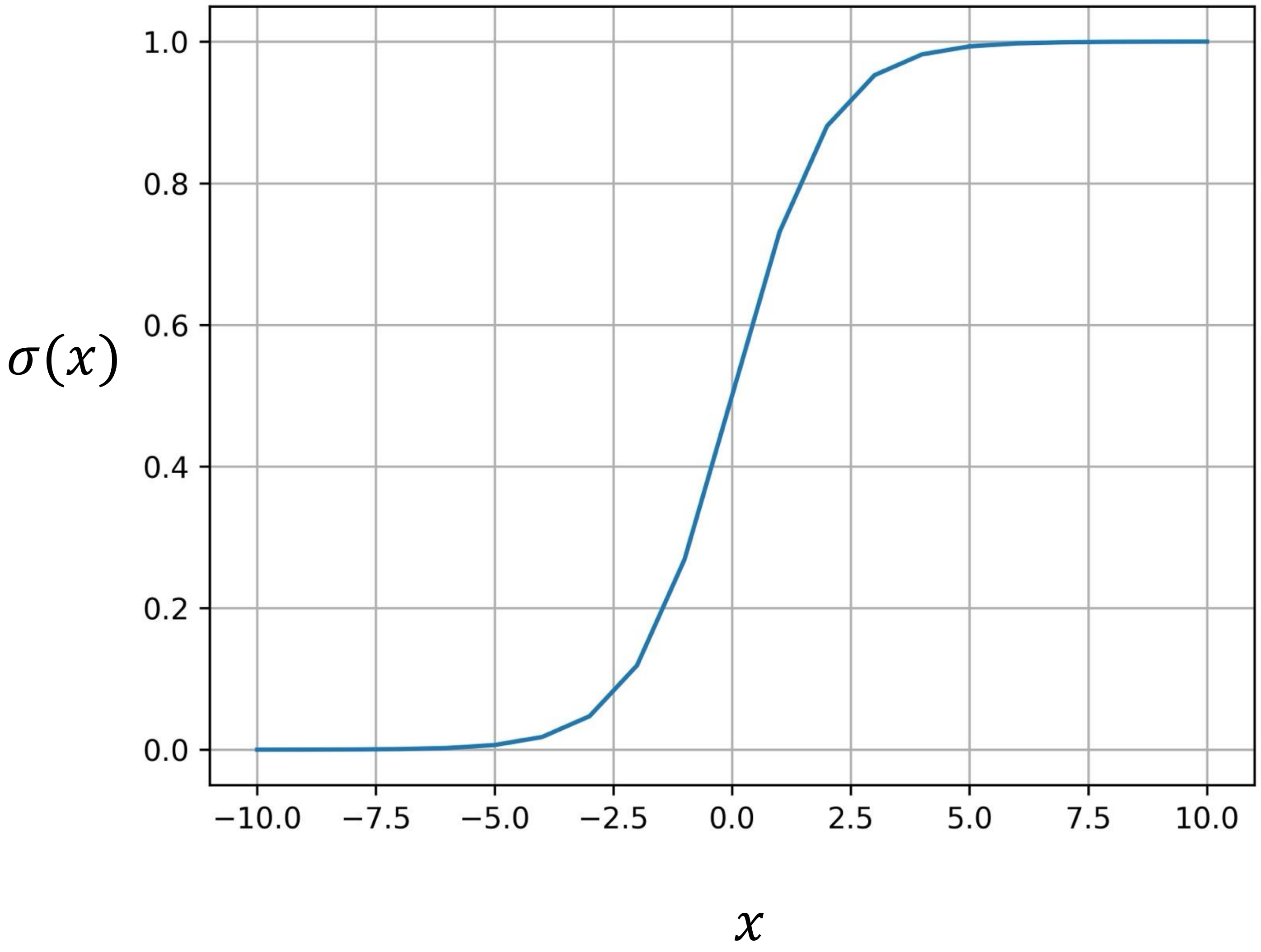
This Sum can be a very large number!

$$= f\left(\sum_{i=1}^{n} w_i^* \; h_i + b\right)$$

$$f \equiv activation\ function$$

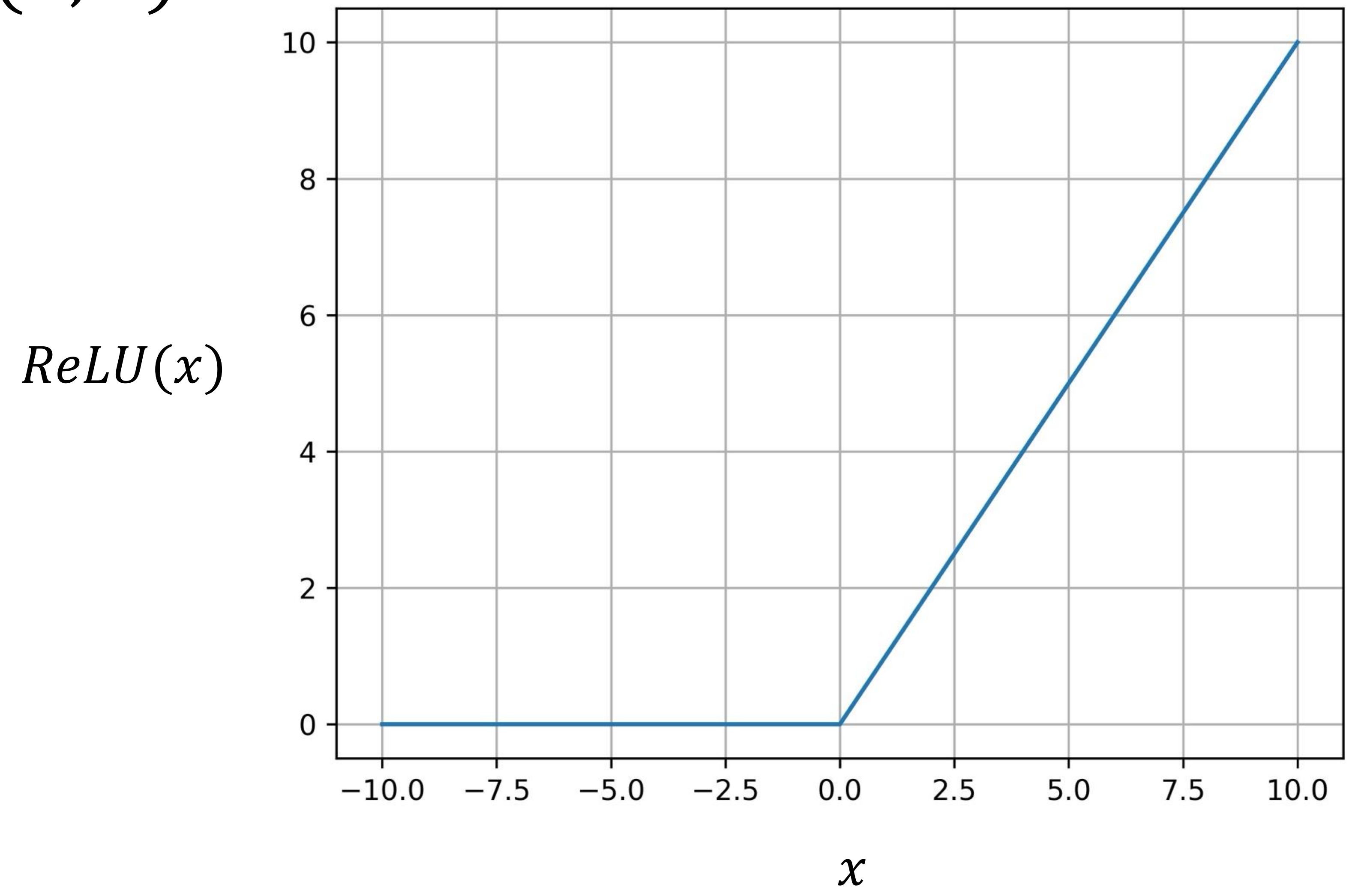
Activation Function: Sigmoid

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



Activation Function: ReLU

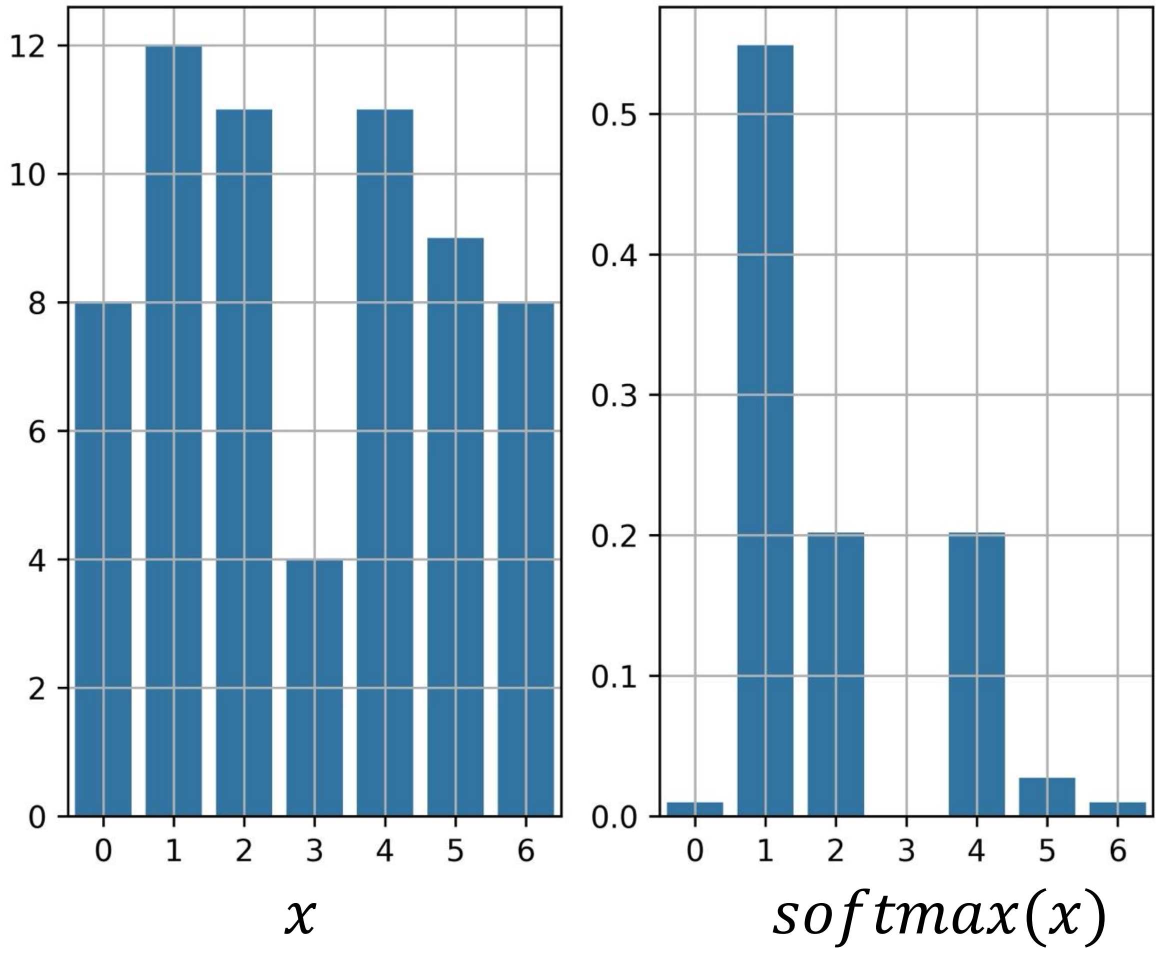
$$\mathrm{ReLU}(x) = \max(0, x)$$



Activation Function: Softmax

For an input vector of $x = [x_1, x_2, \ldots, x_n]$

$$softmax(x_i) = \frac{e^{x_i}}{\sum_{i=0}^{n} e^{x_i}}$$



Revisiting Multi-class classification