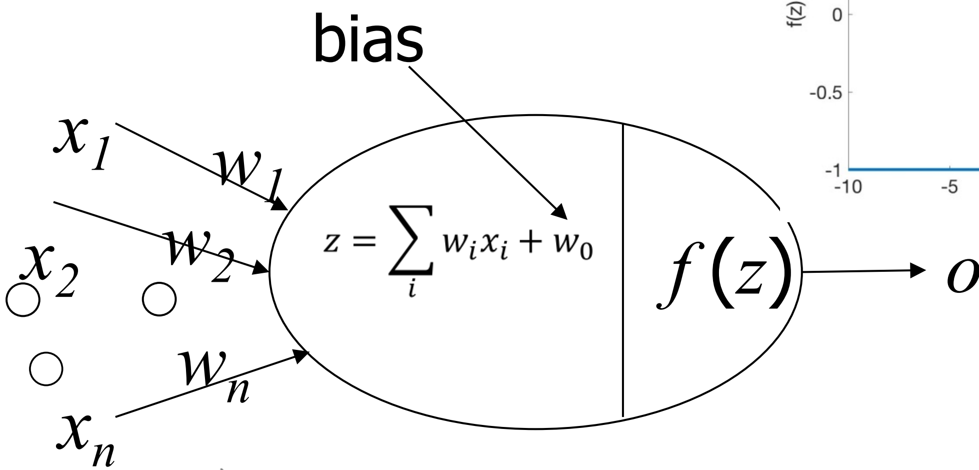
Artificial Neural Network

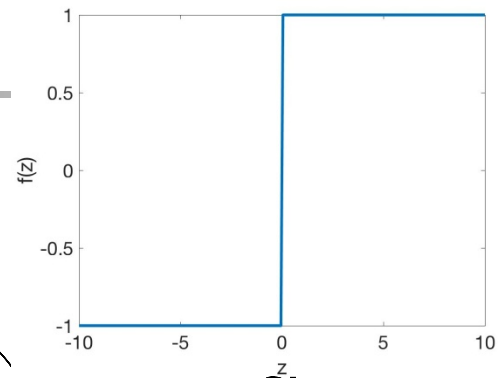
Jayanta Mukhopadhyay
Dept. of Computer Science and Engg.

Books

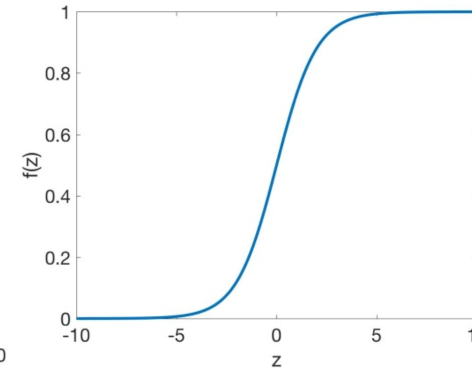
- Chapter 6 of “Pattern Classification” by R.O. Duda, P. E. Hart and D. G. Stork
- Chapter 11 of “Introduction to Machine Learning” by Ethem Alpaydin.
- Chapter 4 of “Machine learning” by Tom M. Mitchel.

Perceptron modelling a neuron



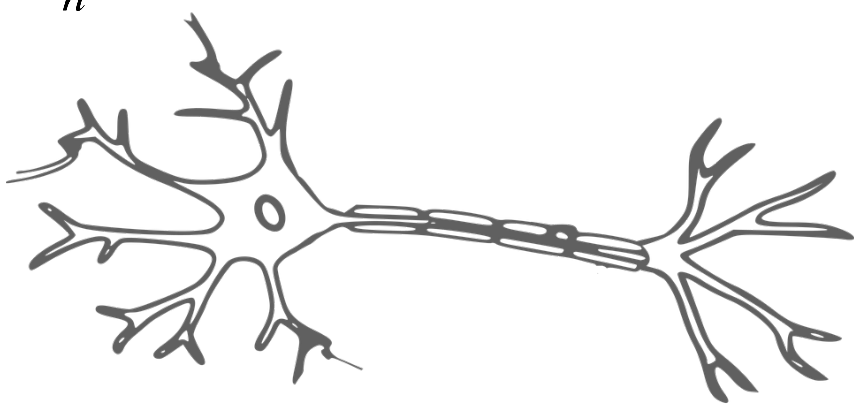


Signum



Logistic / Sigmoid

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

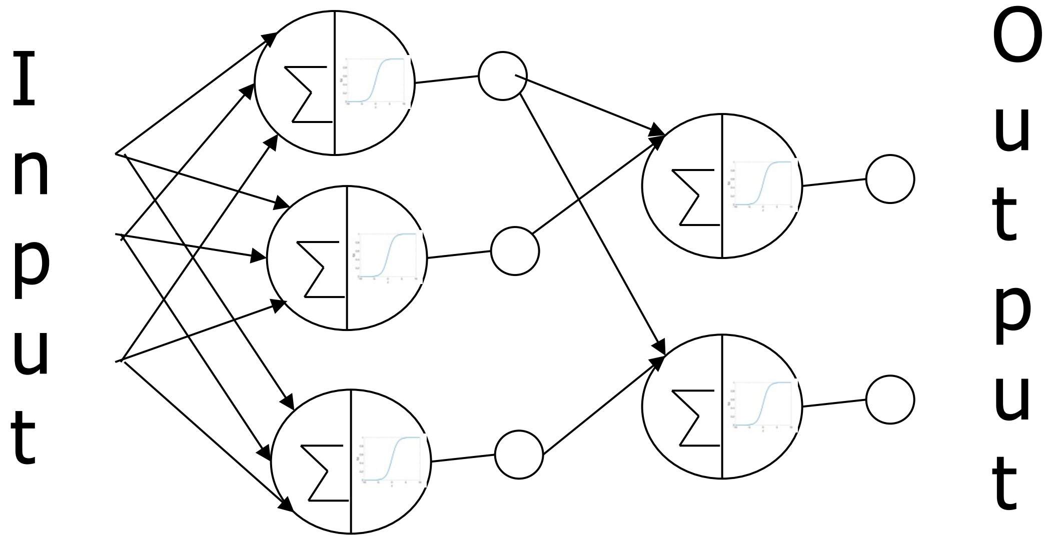


A network of perceptrons provides a powerful model describing input / output relations.

https://pixabay.com/vectors/brain-neuron-nerves-cell-science-2022398/

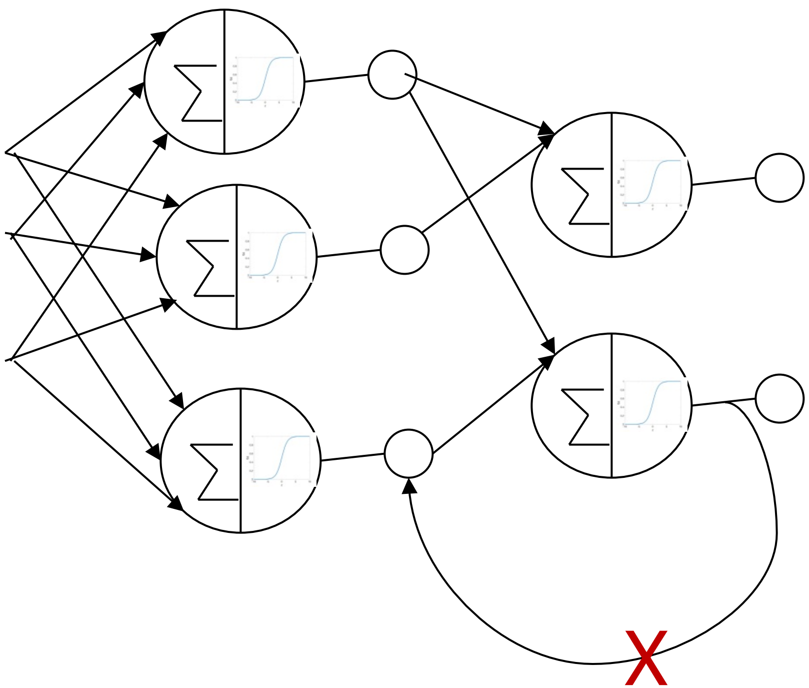
Artificial Neural Network

- A network of perceptrons.
  - Input: A vector
  - Output: A vector / A scalar

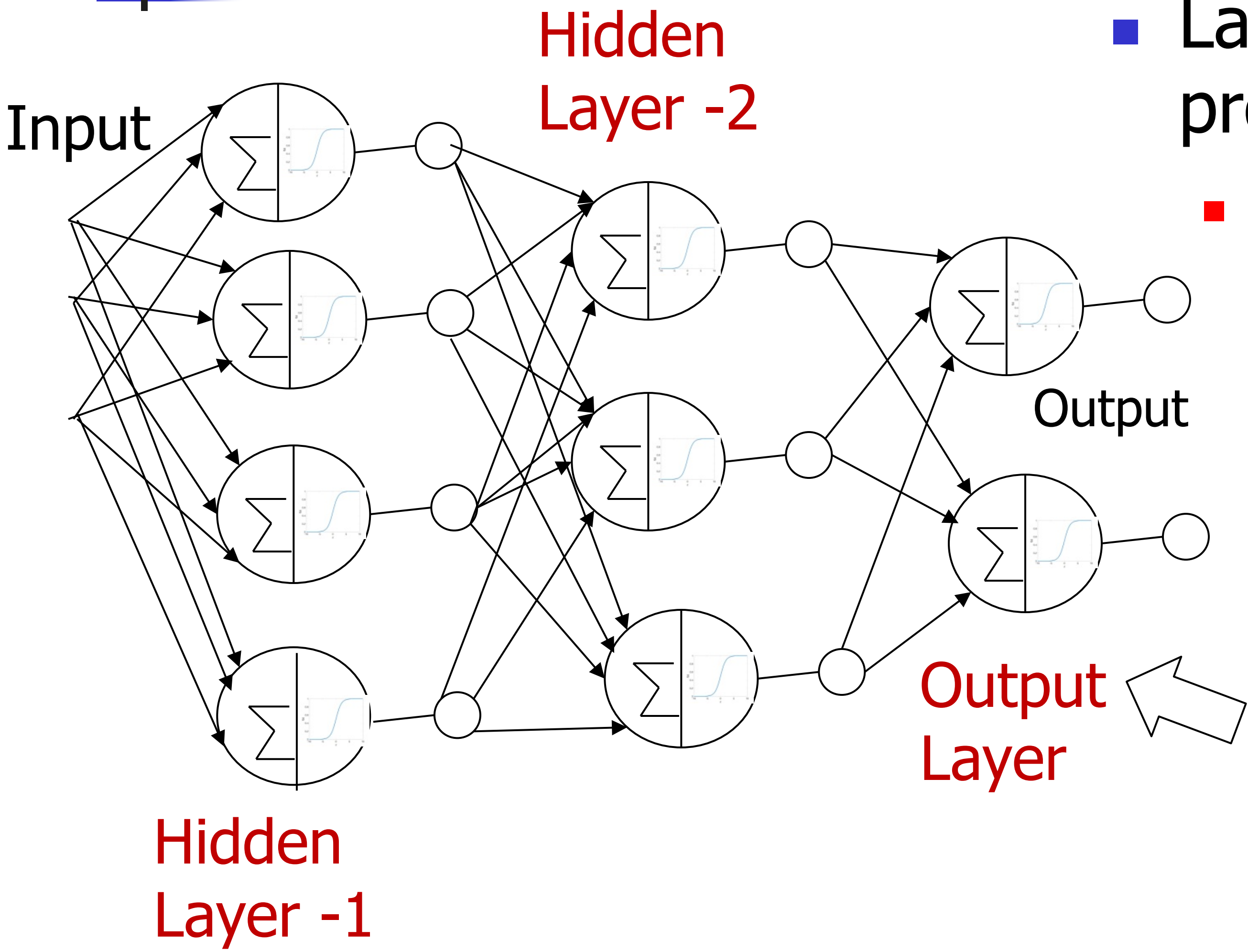


Feed-forward Network

- No feed back or loop in the network.



Multilayered feed-forward Network



- Layer-wise processing
  - i th layer takes input from (i-1)th layer and forwards its output to the input of next layer.

Fully connected (FC) feed-forward network.

Mathematical description of the model

- Let j th neuron of i th layer be $ne_j^{(i)}$.
- Its corresponding weights
  - $W_j^{(i)} = (w_{j1}^{(i)}, w_{j2}^{(i)}, \ldots, w_{jn\_(i-1)}^{(i)})$
  - Bias: $w_{j0}^{(i)}$
  - $n\_(i-1)$: Dimension of input to the neuron
  - n_i: Dimension of output at i th layer
- Output of the neuron:

$$y_j^{(i)} = f\left(W_j^{(i)^T} X^{(i-1)} + w_{j0}^{(i)}\right)$$

Mathematical description of the model

- Output of j th neuron in i th layer:

$$y_j^{(i)} = f\left({W_j^{(i)}}^T X^{(i-1)} + w_{j0}^{(i)}\right)$$

- Input output relation in i th layer

$$Z^{(i)} = \underbrace{\begin{bmatrix} {W_1^{(i)}}^T \\ {W_2^{(i)}}^T \\ \vdots \\ {W_{n\_i}^{(i)}}^T \end{bmatrix}}_{\mathbf{W}^{(i)}} X^{(i-1)} + \underbrace{\begin{bmatrix} w_{10}^{(i)} \\ w_{20}^{(i)} \\ \vdots \\ w_{n\_i0}^{(i)} \end{bmatrix}}_{\mathbf{b}^{(i)}}$$

Input output relation

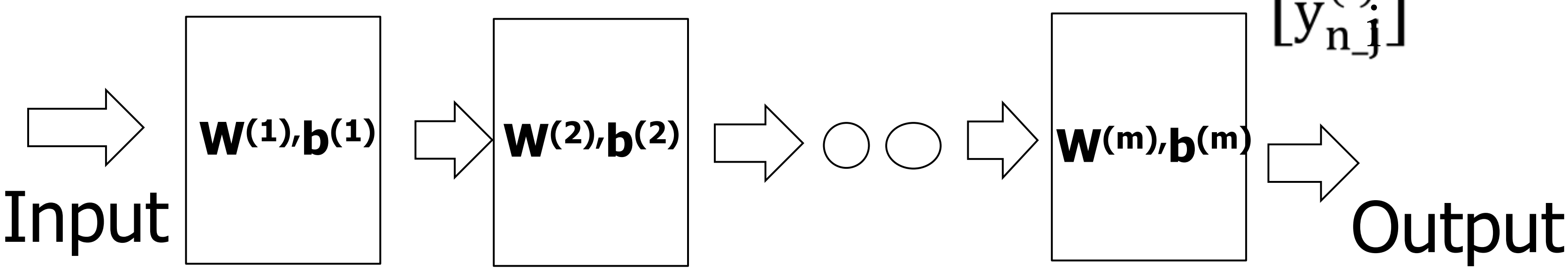
- Output of j th neuron in i th layer:

$$y_j^{(i)} = f\left(W_j^{(i)^T} X^{(i-1)} + w_{j0}^{(i)}\right)$$

$$Z^{(i)} = \overset{\mathbf{W}^{(i)}}{\begin{bmatrix} W_1^{(i)^T} \\ W_2^{(i)^T} \\ \vdots \\ W_{n\_i}^{(i)^T} \end{bmatrix}} X^{(i-1)} + \overset{\mathbf{b}^{(i)}}{\begin{bmatrix} w_{10}^{(i)} \\ w_{20}^{(i)} \\ \vdots \\ w_{n\_i0}^{(i)} \end{bmatrix}}$$

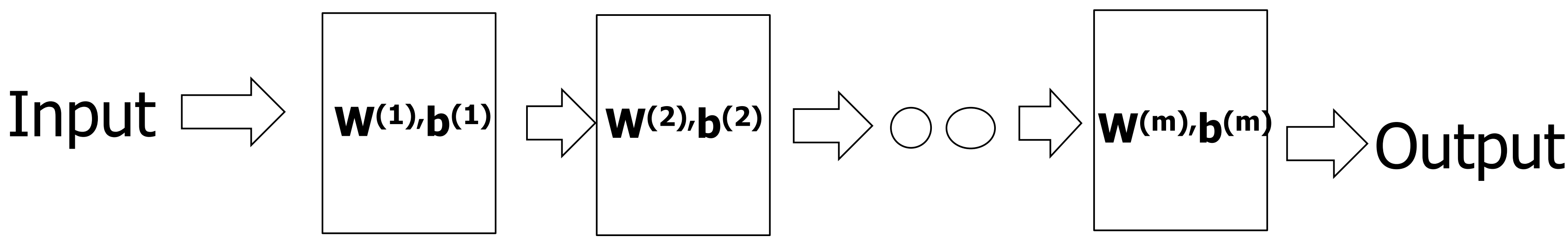
- Input output relation in i th layer

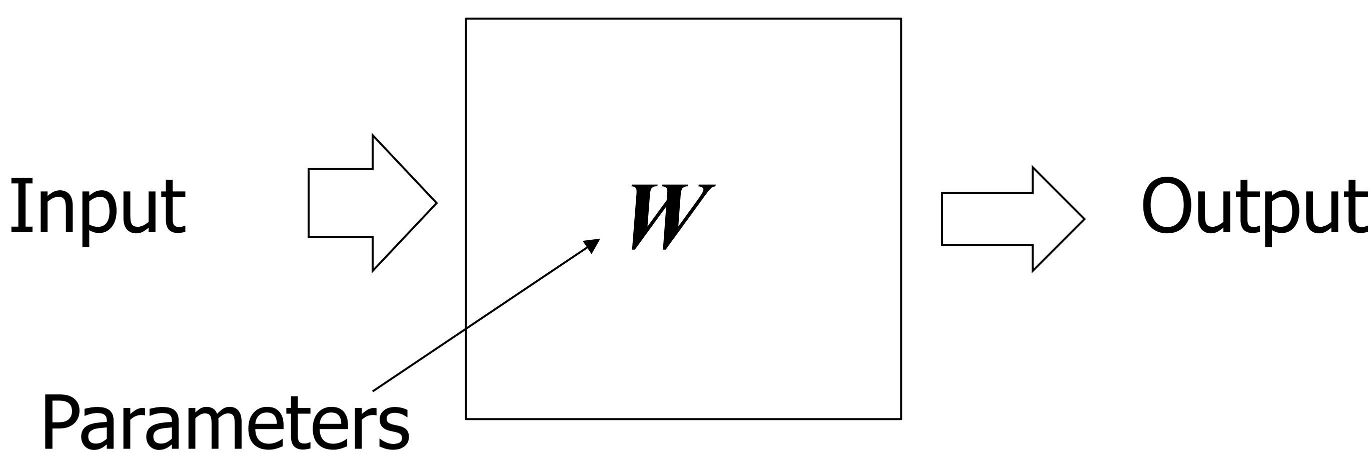
$$Y^{(i)} = \mathbf{f}\left(\mathbf{W}^{(i)} X^{(i-1)} + \mathbf{b}^{(i)}\right) \equiv \begin{bmatrix} y_1^{(i)} \\ y_2^{(i)} \\ \vdots \\ y_{n\_i}^{(i)} \end{bmatrix}$$



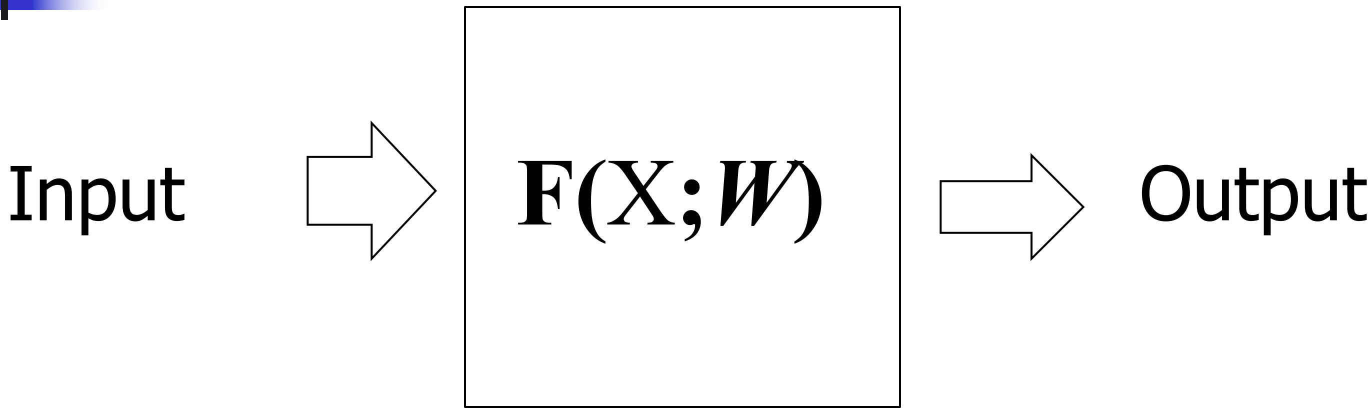
Input output relation

$$Y^{(i-1)} = \mathbf{f}\left(\mathbf{W}^{(i)}X^{(i-1)} + \mathbf{b}^{(i)}\right)$$





Optimization problem



Given $\{(X_i, O_i)\}$, i=1,2,…,N, find ***W*** such that it produces O_i given input X_i for all i.

Minimize:

$$J_n(\boldsymbol{W}) = \frac{1}{N}\sum_{i=1}^{N} \|O_i - F(X_i;\boldsymbol{W})\|^2$$

Apply the same gradient descent procedure to obtain the solution.

Optimization problem

Input $\Rightarrow$ $\boxed{\mathbf{F}(X;\boldsymbol{W})}$ $\Rightarrow$ Output

Training samples:
$\{(X_i\ ,\ O_i)\}$, i=1,2,…,N

Minimize:
$$J_n(\boldsymbol{W}) = \frac{1}{N}\sum_{i=1}^{N} \|O_i - F(X_i;\boldsymbol{W})\|^2$$

Apply the same gradient descent procedure to obtain the solution.

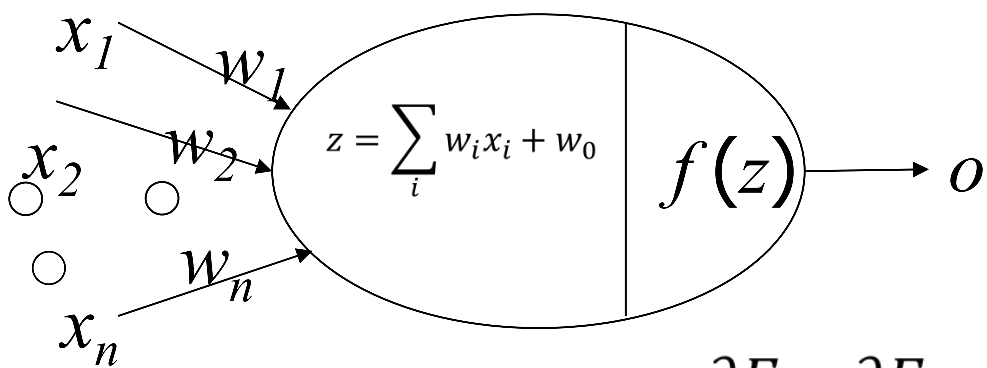
1. Start with an initial $\boldsymbol{W}_0$.
2. Update $\boldsymbol{W}$ iteratively.

$$\boldsymbol{W_i} = \boldsymbol{W_{i-1}} + \eta(i)\sum_{k}\big(O_k - F(X_k;\ \boldsymbol{W_{i-1}})\big)\,\nabla F(X_k;\boldsymbol{W_{i-1}})$$

Stochastic gradient descent:

$$\boldsymbol{W_i} = \boldsymbol{W_{i-1}} + \eta(i)(O_k - F(X_k;\boldsymbol{W_{i-1}}))\,\nabla F(X_k;\boldsymbol{W_{i-1}})$$

Chain rule of computing gradient of a single neuron



Target response: t

Error:
$E=(t\text{-}o)^2$

$$\nabla(W) = \left(\frac{\partial E}{\partial w_0}, \frac{\partial E}{\partial w_1}, \dots, \frac{\partial E}{\partial w_n}\right)$$

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

$$f'(z) = \frac{e^{-z}}{(1+e^{-z})^2}$$

$$\frac{1}{1+e^{-z}}\left(1 - \frac{1}{1+e^{-z}}\right)$$

$$f(z)(1\text{-}f(z))$$

$$\frac{\partial E}{\partial w_i} = \underbrace{\frac{\partial E}{\partial o}}_{-2(t-o)} \underbrace{\frac{\partial o}{\partial z}}_{f'(z)} \underbrace{\frac{\partial z}{\partial w_i}}_{x_i}$$

$$\frac{\partial E}{\partial w_i} = -2(t-o)f'(z)x_i$$

Analytical method!
Computed given the functional values.

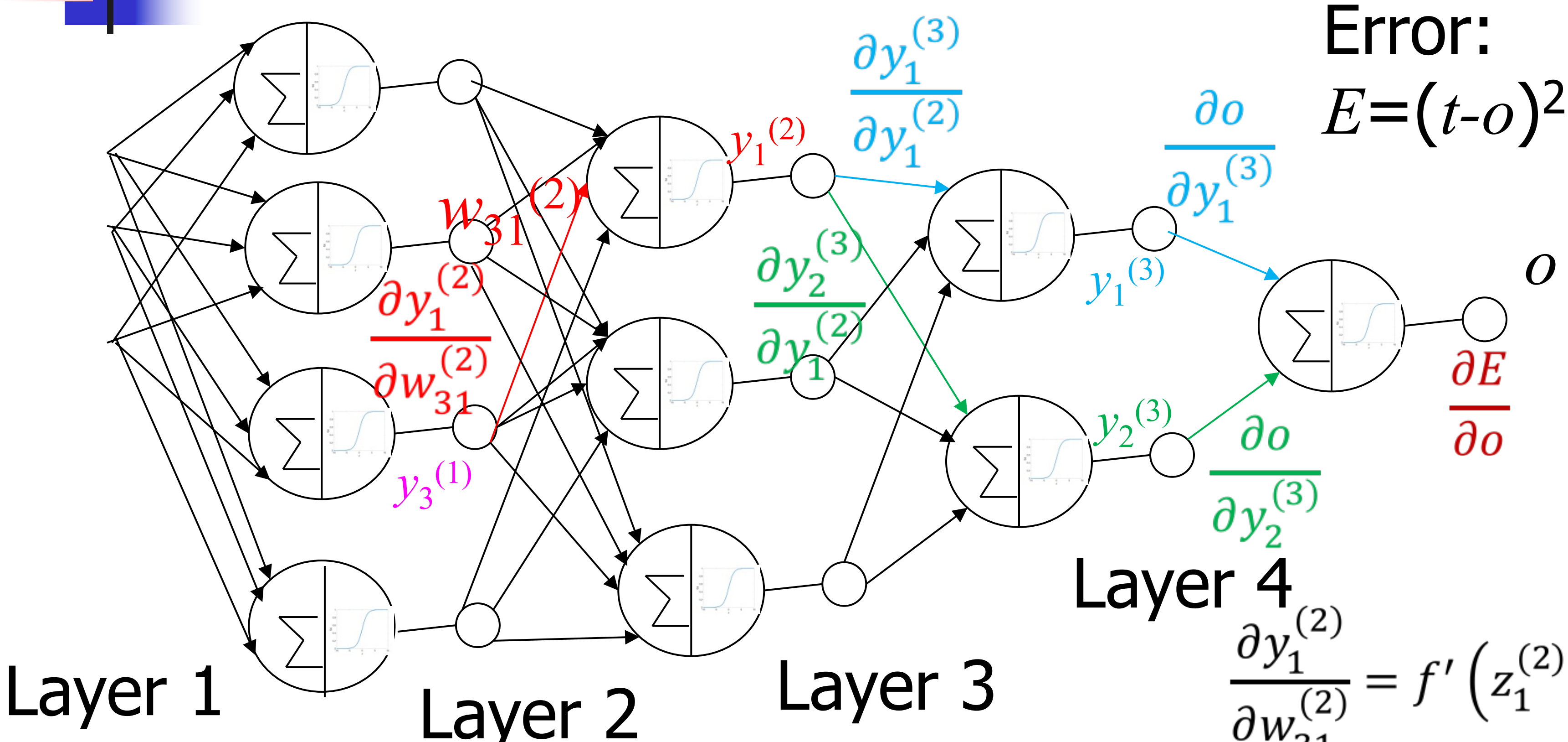
$$\frac{\partial E}{\partial x_i} = \frac{\partial E}{\partial o}\frac{\partial o}{\partial z}\frac{\partial z}{\partial x_i}$$

$$\frac{\partial E}{\partial x_i} = -2(t-o)f'(z)w_i$$

Computing gradient: Back propagation method

- For multi-layered feed forward network.
- Apply chain rule.
  - From output to toward input.
  - From output layer to toward input layer.
  - Compute partial derivatives of weights at (i-1)th layer from the i th layer.

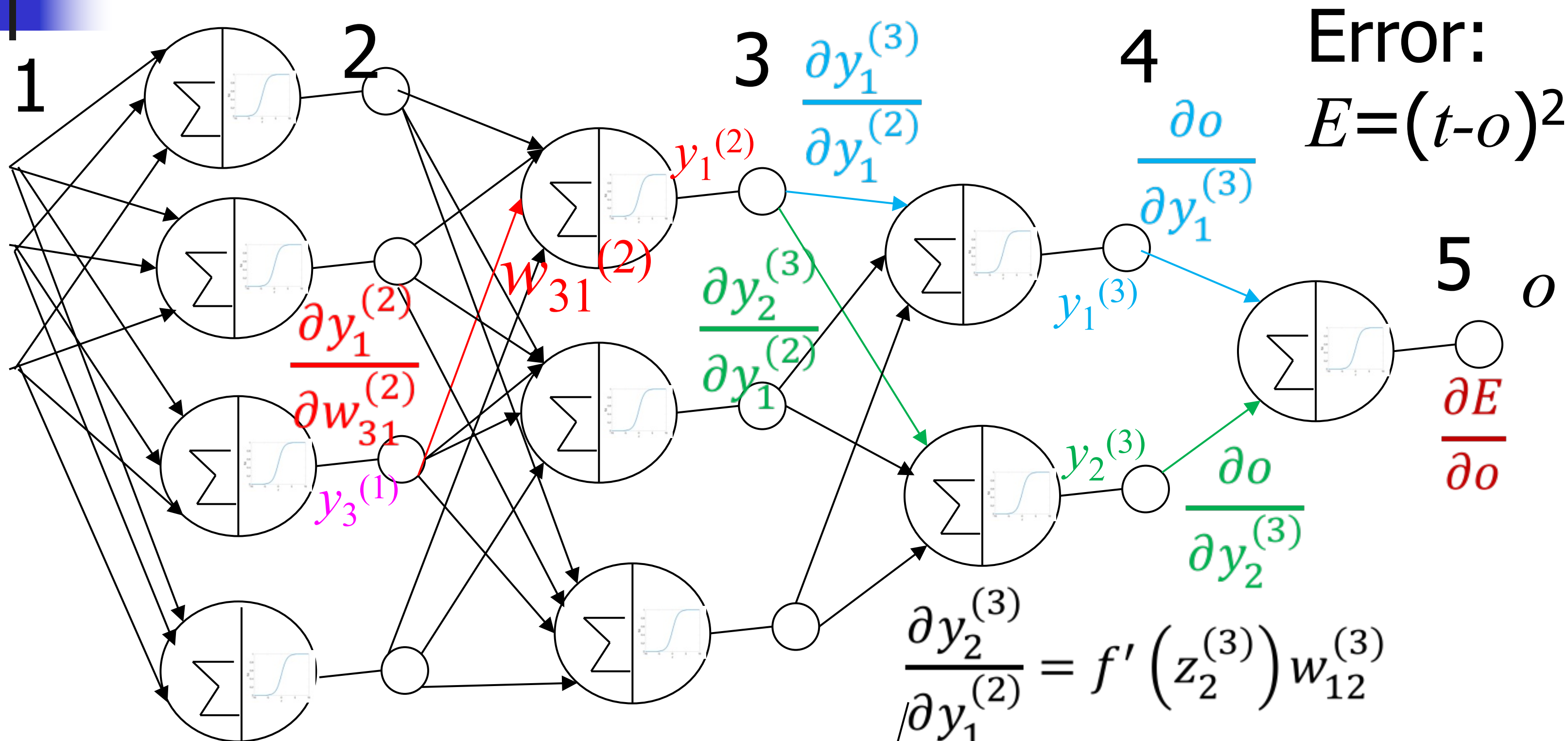
Back propagation: Concept



$$\frac{\partial y_1^{(2)}}{\partial w_{31}^{(2)}} = f'\left(z_1^{(2)}\right) y_4^{(1)}$$

$$\Delta w_{31}^{(2)} = \frac{\partial E}{\partial w_{31}^{(2)}} = \frac{\partial E}{\partial o}\left(\frac{\partial o}{\partial y_1^{(3)}}\frac{\partial y_1^{(3)}}{\partial y_1^{(2)}} + \frac{\partial o}{\partial y_2^{(3)}}\frac{\partial y_2^{(3)}}{\partial y_1^{(2)}}\right)\frac{\partial y_1^{(2)}}{\partial w_{31}^{(2)}}$$

Back propagation: Concept

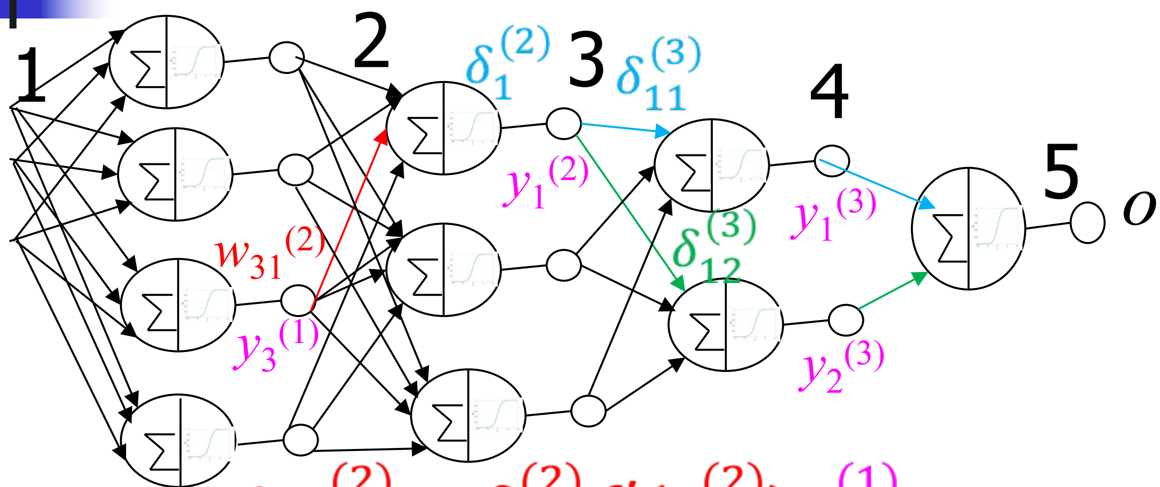


$$\frac{\partial y_2^{(3)}}{\partial y_1^{(2)}} = f'\left(z_2^{(3)}\right) w_{12}^{(3)}$$

$$\Delta w_{31}^{(2)} = \frac{\partial E}{\partial w_{31}^{(2)}} = \frac{\partial E}{\partial o}\left(\frac{\partial o}{\partial y_1^{(3)}}\frac{\partial y_1^{(3)}}{\partial y_1^{(2)}} + \frac{\partial o}{\partial y_2^{(3)}}\frac{\partial y_2^{(3)}}{\partial y_1^{(2)}}\right)\frac{\partial y_1^{(2)}}{\partial w_{31}^{(2)}}$$

$$\frac{\partial y_1^{(2)}}{\partial w_{31}^{(2)}} = f'\left(z_1^{(2)}\right) y_4^{(1)}$$

Back propagation: Delta rule



Error:
$E=(t\text{-}o)^2$

$$\frac{\partial y_2^{(3)}}{\partial y_1^{(2)}} = f'\left(z_2^{(3)}\right) w_{12}^{(3)}$$

$$\frac{\partial y_1^{(3)}}{\partial y_1^{(2)}} = f'(z_1^{(3)}) w_{11}^{(3)}$$

$$\Delta w_{31}^{(2)} = \delta_1^{(2)} f'(z_1^{(2)}) y_3^{(1)}$$

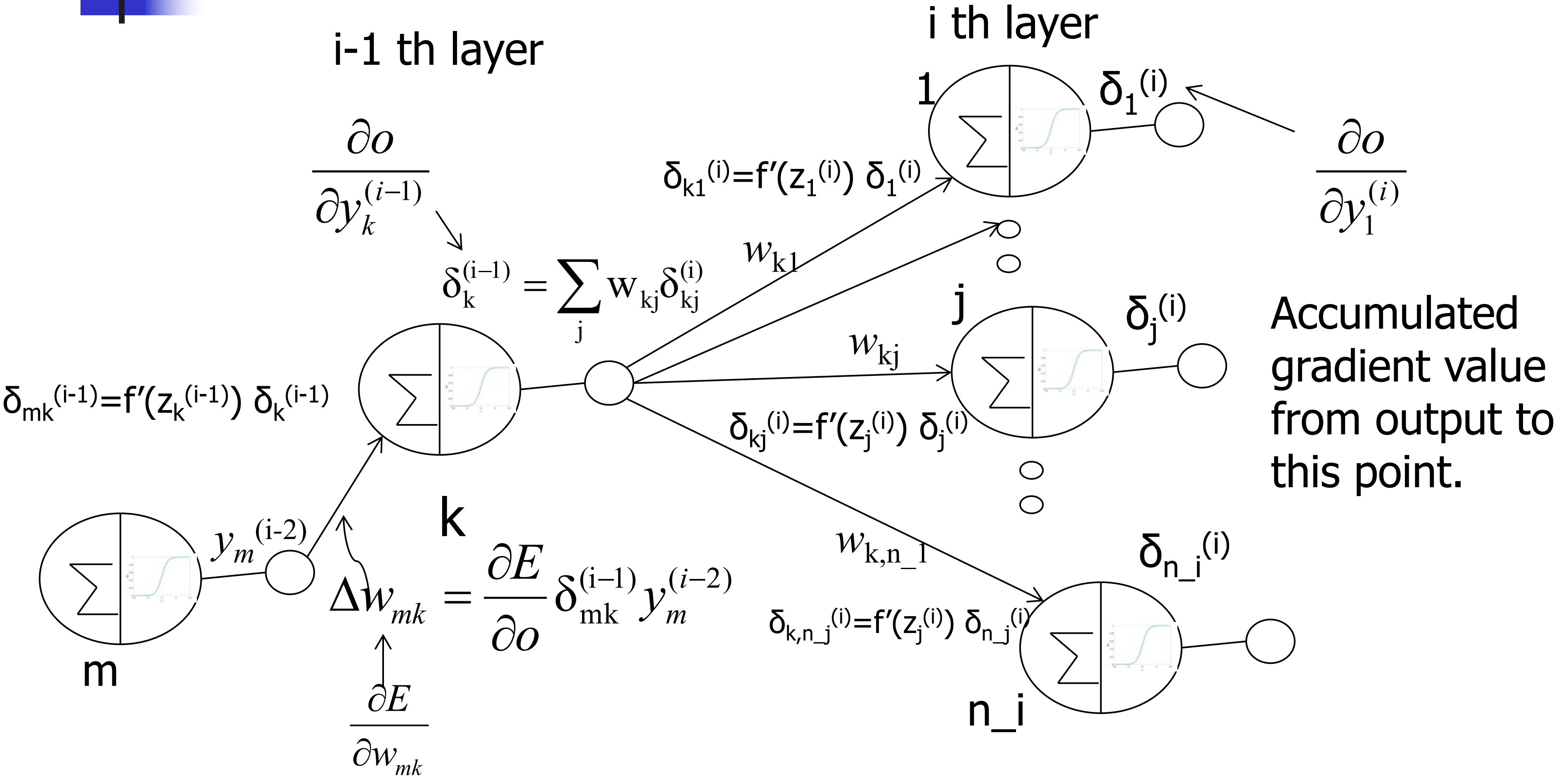
$$\Delta w_{31}^{(2)} = \frac{\partial E}{\partial w_{31}^{(2)}} = \frac{\partial E}{\partial o}\left(\frac{\partial o}{\partial y_1^{(3)}}\frac{\partial y_1^{(3)}}{\partial y_1^{(2)}} + \frac{\partial o}{\partial y_2^{(3)}}\frac{\partial y_2^{(3)}}{\partial y_1^{(2)}}\right)\frac{\partial y_1^{(2)}}{\partial w_{31}^{(2)}}$$

$$\Delta w_{31}^{(2)} = \frac{\partial E}{\partial w_{31}^{(2)}} = \frac{\partial E}{\partial o}\left(\underbrace{\frac{\partial o}{\partial y_1^{(3)}} f'\left(z_1^{(3)}\right)}_{\delta_{11}^{(3)}} w_{11}^{(3)} + \underbrace{\frac{\partial o}{\partial y_2^{(3)}} f'\left(z_2^{(3)}\right)}_{\delta_{12}^{(3)}} w_{12}^{(3)}\right)\frac{\partial y_1^{(2)}}{\partial w_{31}^{(2)}}$$

Delta rule:

$$\delta_1^{(2)} = \delta_{11}^{(3)} w_{11}^{(3)} + \delta_{12}^{(3)} w_{12}^{(3)}$$

Back propagation: Delta rule



ANN training

- Initialize $W^{(0)}$.
- For each training sample (x_i, o_i) do
  - Compute functional values of each neuron in the forward pass.
  - Update weights of each link starting from the output layer using back propagation.
  - Continue till it converges.

Improving convergence

- Momentum
  - Gradients may change abruptly in consecutive iteration. To avoid we may use running average of weight updates to be added with the gradient.

$$\Delta w_i^{(t)} = \alpha \Delta w_i^{(t-1)} - \eta \frac{\partial E^{(t)}}{\partial w_i}$$

Usually ranges between 0.5 to 1.0

$$\Delta\eta = \begin{cases} +a & if\ E^{t+T} < E^t \\ -b\eta & Otherwise \end{cases}$$

  - Adaptive learning rate:
    - We increase learning rate at constant steps if error decreases, else decrease it geometrically.

Summary

- A perceptron models a neuron.
- A network of perceptron nodes provide a powerful model for classifying linearly non-separable classes as well.
- A multilayer feed forward network of neurons can be trained using back propagation algorithm (to compute gradients of error w.r.t weights).
- Can be used for regression as well.

Thank you!