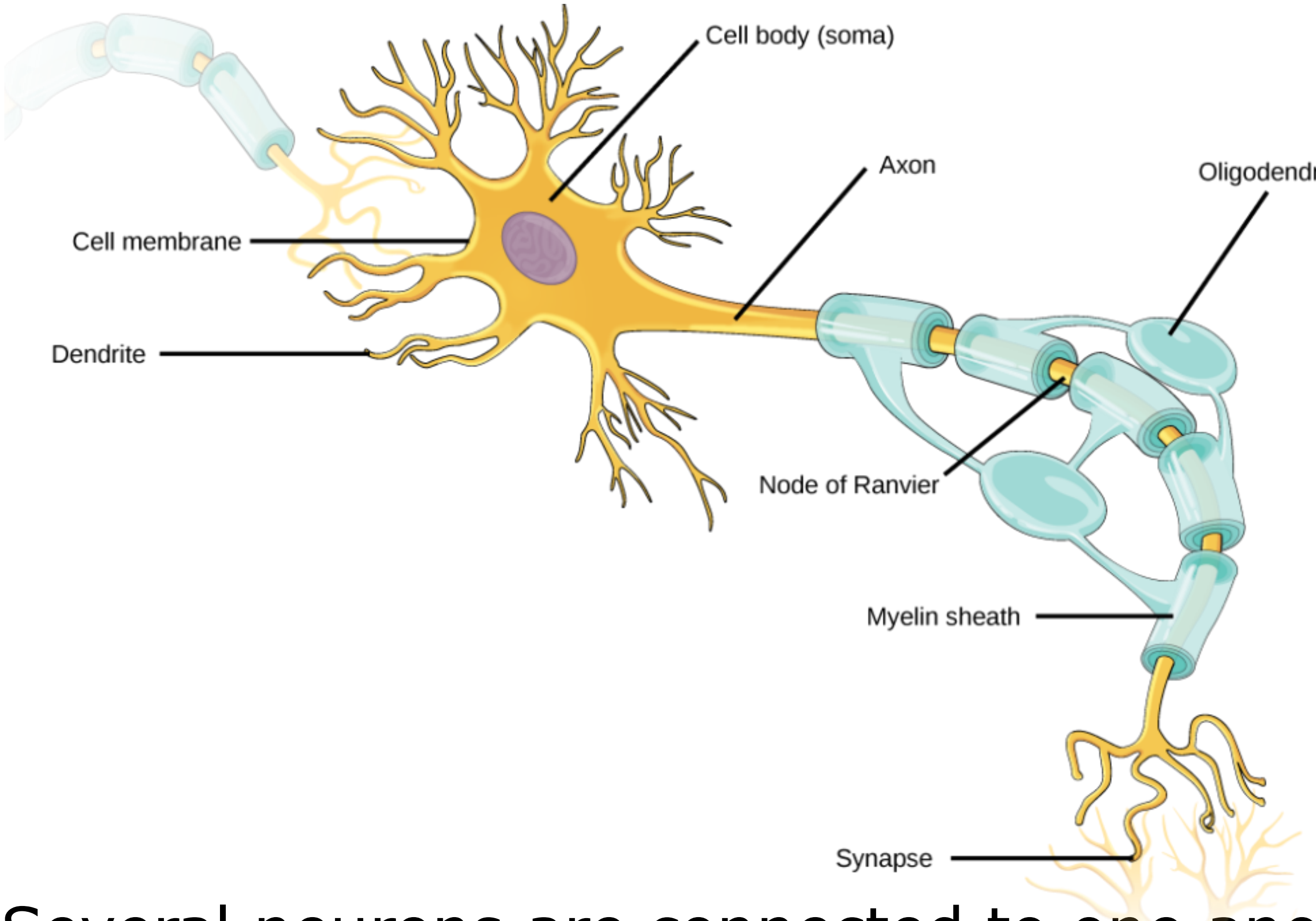
Introduction to Deep Learning

Basics of Artificial Neural Networks

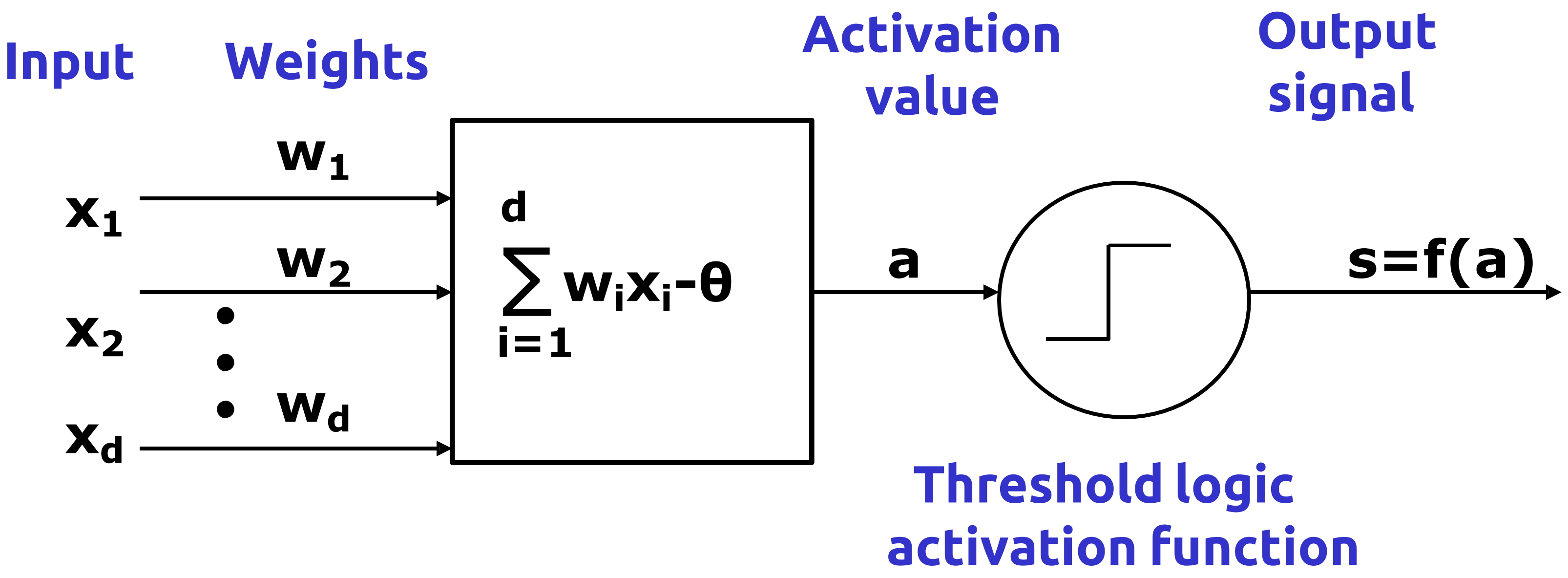
Biological Neural Networks



- Several neurons are connected to one another to form a neural network or a layer of a neural network.

Feed Forward Neural Networks (FFNN)

Neuron with Threshold Logic Activation Function



- McCulloch-Pitts Neuron

Linearly Separable Classes

- Regions of two classes are separable by a linear surface (line, plane or hyperplane)
- Perceptron model that uses a single MuCulloch-Pitts neuron can be trained using the perceptron learning algorithm

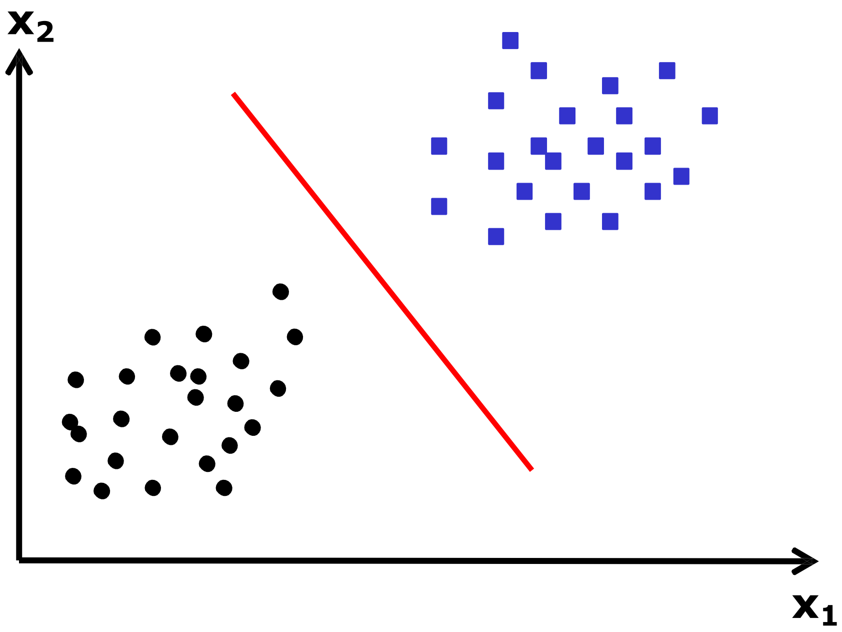
Decision surface in a 2-dimensional space is a line:

$$w_1 x_1 + w_2 x_2 + w_0 = 0$$

$$x_2 = -\frac{w_1}{w_2} x_1 - \frac{w_0}{w_2}$$

Decision surface in a d-dimensional space is a hyperplane:

$$\sum_{i=0}^{d} w_i x_i = \mathrm{w}^t \mathrm{x} = 0$$



Perceptron Learning

- Perceptron learning rule for weight update:

  At Step m, choose a training example **x**(m).

  $$\boldsymbol{w}(m+1) = \boldsymbol{w}(m) + \eta\, \boldsymbol{x}(m), \text{for} \quad \boldsymbol{w}(m)^t \boldsymbol{x}(m) \le 0 \;\text{ and }\; \boldsymbol{x}(m) \in C_1$$

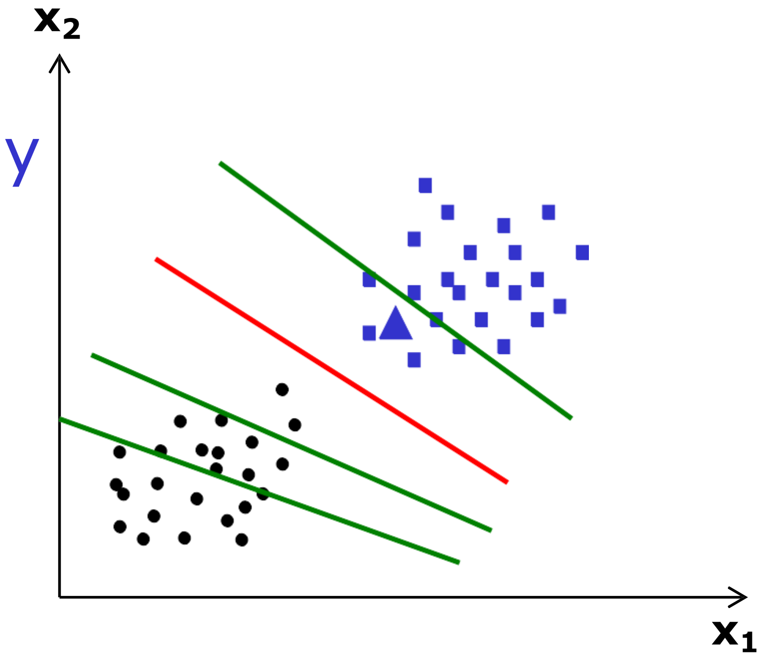
  $$\boldsymbol{w}(m+1) = \boldsymbol{w}(m) - \eta\, \boldsymbol{x}(m), \text{for} \quad \boldsymbol{w}(m)^t \boldsymbol{x}(m) > 0 \;\text{ and }\; \boldsymbol{x}(m) \in C_2$$

  Here η is the learning rate parameter.

- Classification of a test pattern **x** using the weights **w** obtained by training the model :

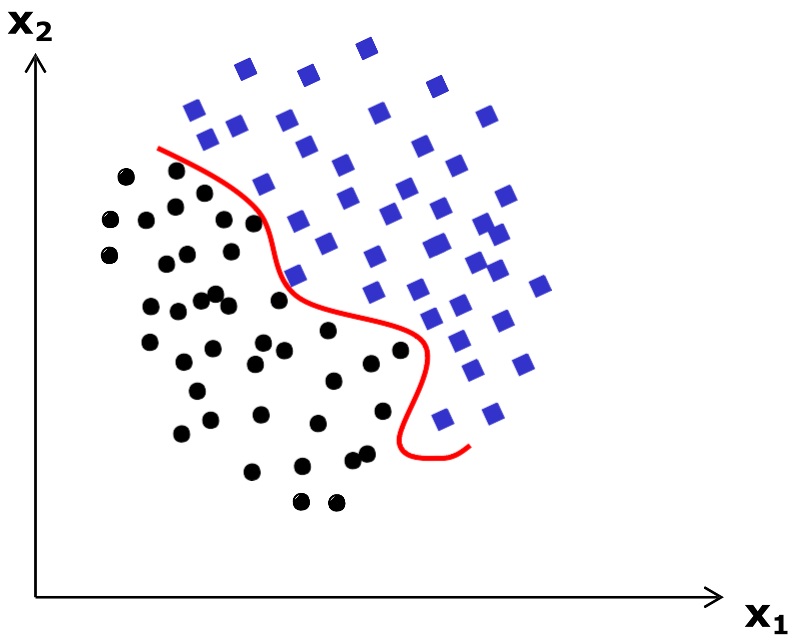
  If $\mathbf{w}^t\mathbf{x} > 0$ then **x** is assigned to C_1

  If $\mathbf{w}^t\mathbf{x} \le 0$ then **x** is assigned to C_2

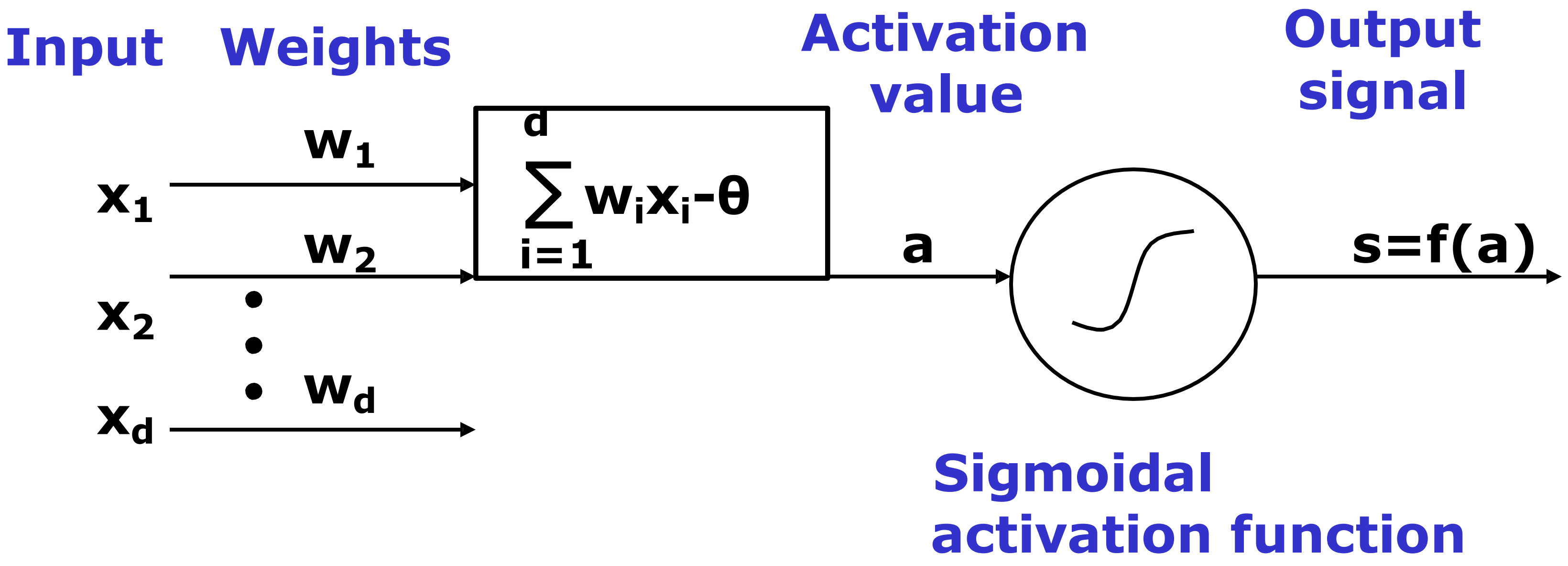


Hard Problems

- Nonlinearly separable classes



Neuron with Continuous Activation Function



Sigmoidal Neuron

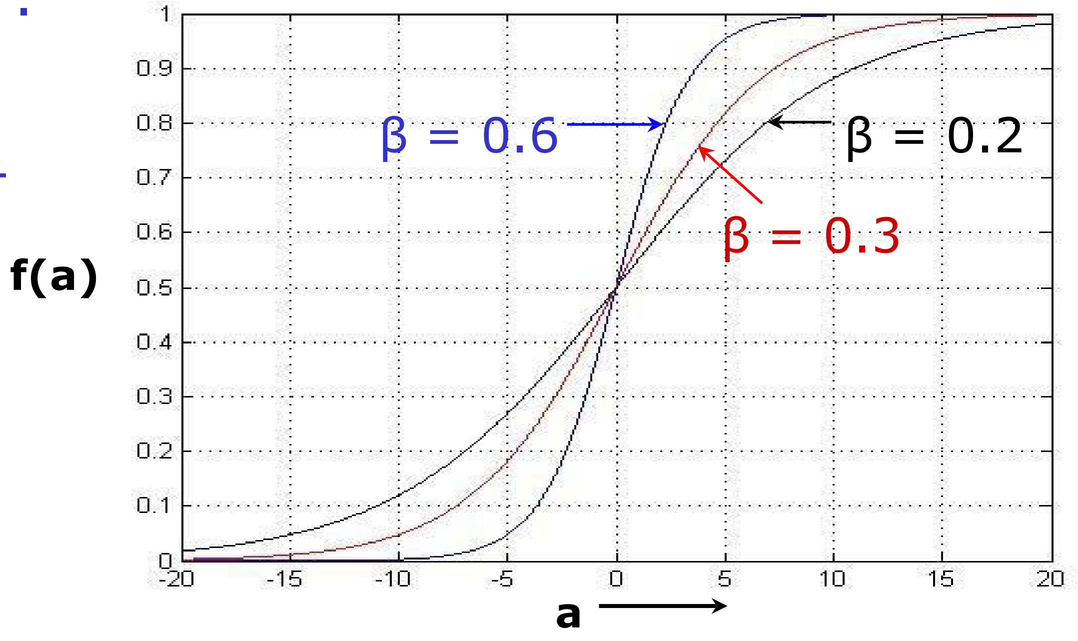
Sigmoidal Activation Functions

Logistic function:

Range of output is 0.0 to 1

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

$$\frac{df(a)}{da} = \beta f(a)\left(1 - f(a)\right)$$



Hyperbolic tangent function:

Range of output is -1.0 to 1.0

$$f(a) = \tanh(\beta a)$$

$$\frac{df(a)}{da} = \beta\left(1 - f^2(a)\right)$$

Logistic function and Hyperbolic tangent functions can be used in hidden layers and output layer

Softmax Activation Function

Softmax activation function:

- Used in the output layer only
- Used for multi-class pattern classification tasks
- Outputs of softmax neurons are in the range 0.0 to 1.0, and add up to 1.0
- Outputs of softmax neurons can be interpreted as probabilities for classes

- Let K be the number of classes
- Let a_j , $j = 1, 2, \ldots, K$, be the activation value for the node of class j.
- Then the output of kth node is given by

$$s_k = \frac{e^{a}}{\sum_{j=1}^{K} e^{a_j}}$$

- The following can be derived:

$$\frac{\partial s}{\partial a} = s_k (1 - s_k)$$

$$\frac{\partial s_k}{\partial a_j} = -s_k s_j$$

Rectilinear Activation Function

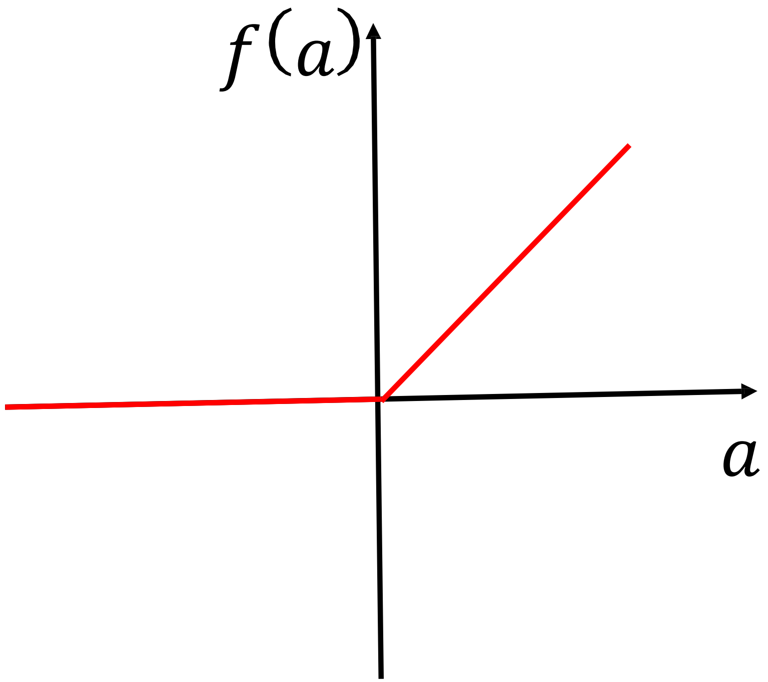
Rectilinear activation function:

- **Used in the hidden layers only (mainly in convolutional neural networks)**
- **Can not be used in the output layer**
- **Nodes with rectilinear activation function are called as Rectilinear Units (ReLU)**

- **No saturating region in the activation function**

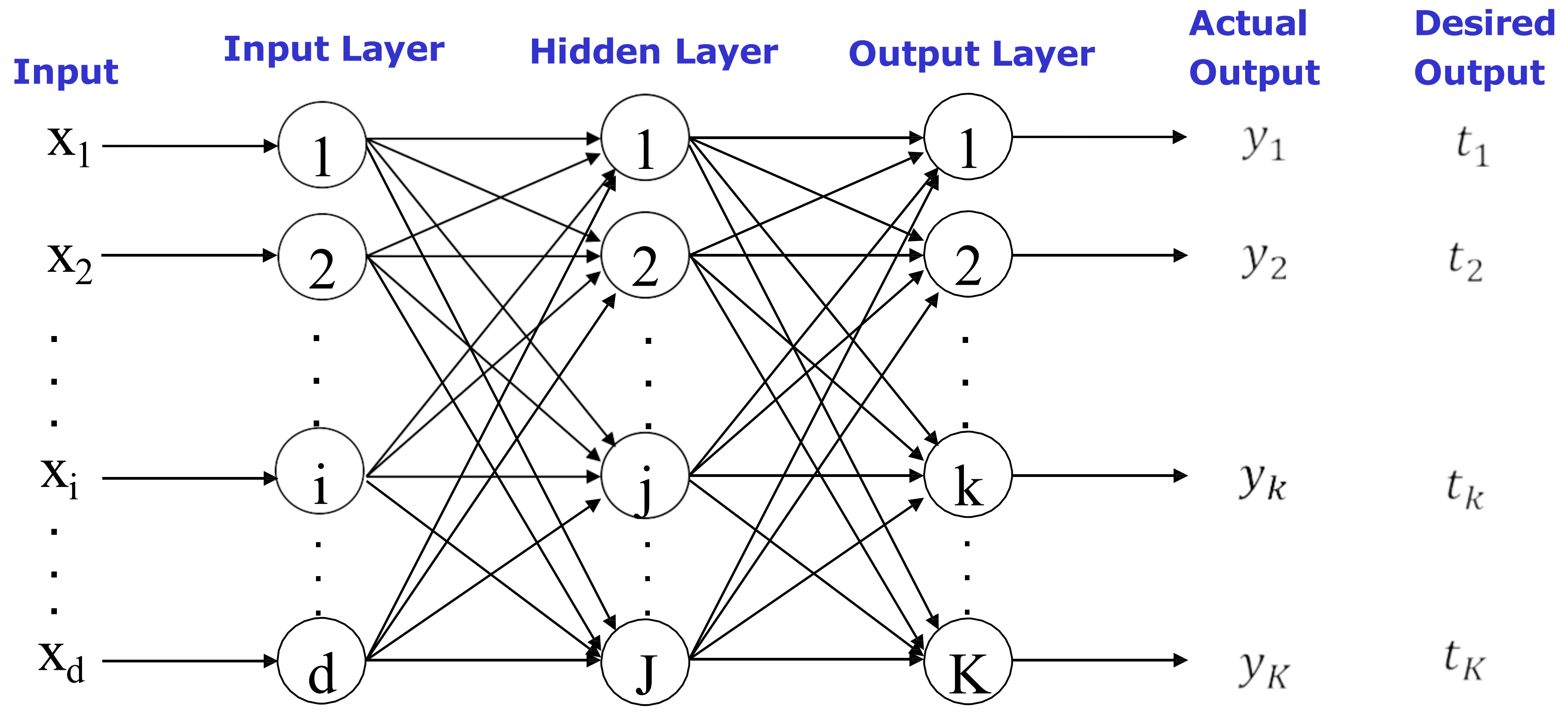
$$f(a) = \max(0, a)$$

$$\frac{df(a)}{da} = 1, \text{for } a > 0$$

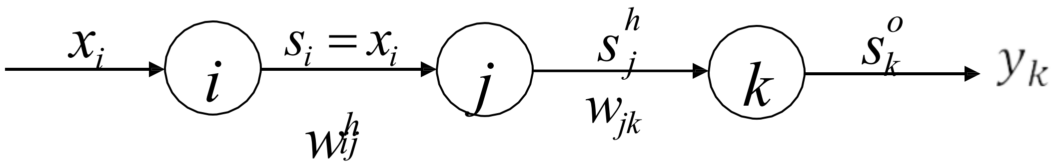


Multilayer Feedforward Neural Network

- **Architecture:**
  - Input layer ---- Linear neurons
  - One or more hidden layers ---- Nonlinear neurons
  - Output layer ---- Linear or nonlinear neurons



Computations in Forward Pass



$$s_j^h = f_j^h(a_j^h)$$

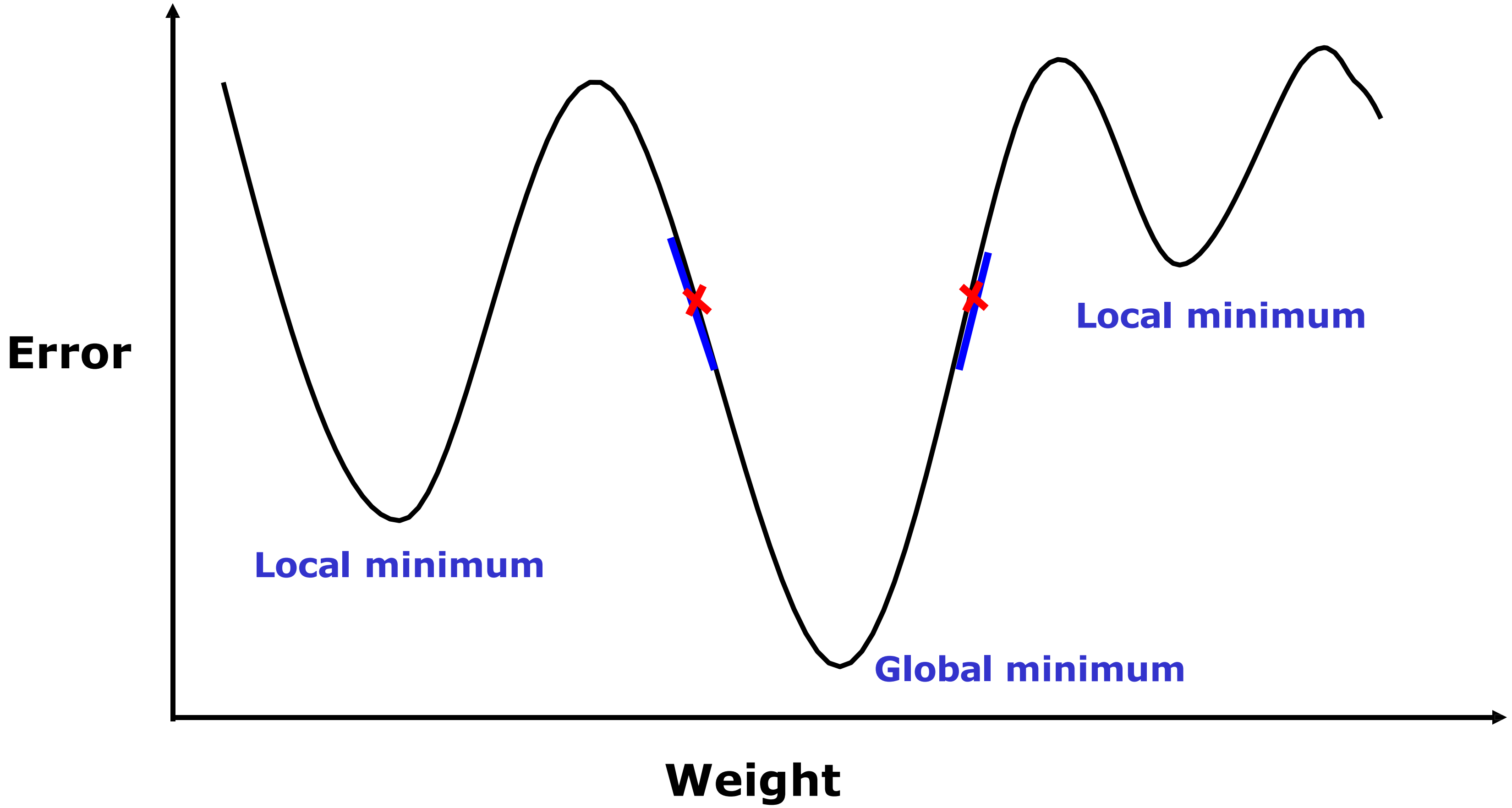
$$a_j^h = \sum_{i=1}^{d} w_{ij}^h s_i - \theta_j^h, \text{ where } s_i = x_i$$

$$s_k^o = f_k^o(a_k^o)$$

$$a_k^o = \sum_{j=1}^{J} w_{jk} s_j^h - \theta_k^o$$

$$s_k^o = f_k^o\left[\sum_{j=1}^{J} w_{jk} f_j^h\left(\sum_{i=1}^{d} w_{ij}^h s_i - \theta_j^h\right) - \theta_k^o\right]$$

Gradient Descent Method



Backpropagation Learning

- Gradient descent method
- Error backpropagation algorithm
- Forward computation:
  - Innerproduct computation
  - Activation function computation
- Backward operation:
  - Error calculation and propagation
  - Modification of weights
- Given a set of N input–output pairs

$$(\boldsymbol{x}_n, \boldsymbol{t}_n),\ n=1,2,\dots.N.$$

- Instantaneous error for the nth pattern:

$$\tilde{E}_n = \frac{1}{2}\sum_{k=1}^{K}(t_{nk} - y_k)^2$$ **Sum-of-squared errors**

Modes of Learning

- Pattern Mode :
  - Stochastic Gradient Descent Method
  - Weights are updated after the presentation of each example.

$$\Delta w(m) = -\eta \frac{\partial \tilde{E}(m)}{\partial w}$$

  - Epoch: Presentation of all the examples once.
- Batch Mode:
  - Weights are updated after the presentation of all the examples once.

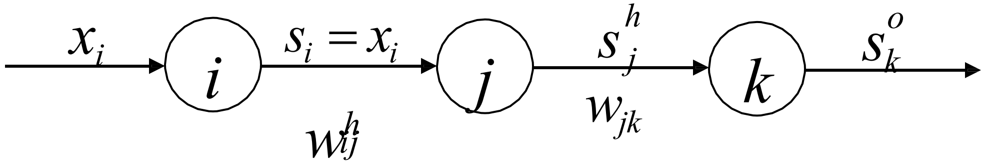
$$\Delta w(m) = -\eta \frac{\partial E_{av}(m)}{\partial w}$$

$$\text{where } E_{av} = \frac{1}{N} \sum_{n=1}^{N} \tilde{E}_n$$

Modes of Learning

- **Mini-Batch Mode:**
  - The training dataset of N examples is divided into M subsets, with approximately same number of examples in each subset.
  - Weights are updated after the presentation of examples in the subset (mini-batch)
  - It is empirically shown that the mini-batch mode is more effective than the pattern mode and batch mode.

Backpropagation Learning (contd.)



- **Change in weight at output layer is given by**

$$\Delta w_{jk}(m) = -\eta \frac{\partial \tilde{E}(m)}{\partial w_{jk}}$$

$$\Delta w_{jk}(m) = \eta \, \delta_k^o s_j^h$$

It can be shown that these two are equal

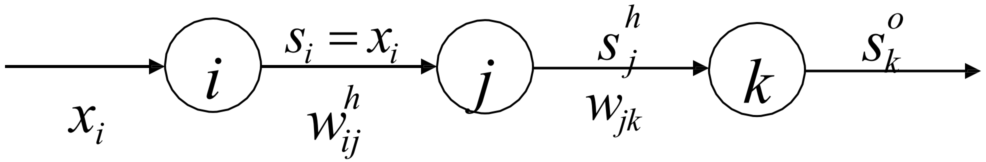
$$\delta_k^o = -\frac{\partial \tilde{E}(m)}{\partial a_k^o} = (t_k - s_k^o) \frac{d f_k^o(a_k^o)}{d a_k^o}$$

Activation value

Target

δ_k^o is called the Local Gradient. It is the negative of the derivative of instantaneous error with respect to activation value of target node of the connection.

Backpropagation Learning (contd.)



- **Change in weight at hidden layer is given by**

$$\Delta w_{ij}^h(m) = -\eta \frac{\partial \tilde{E}(m)}{\partial w_{ij}^h}$$

$$\Delta w_{ij}^h(m) = \eta\, \delta_j^h s_i$$

$$\delta_j^h = -\frac{\partial \tilde{E}(m)}{\partial a_j^h} = \left(\sum_{k=1}^K \delta_k^o\, w_{jk}\right) \frac{df_j^h\left(a_j^h\right)}{da_j^h}$$

$$\Delta w_{ij}^h(m) = \eta\, \delta_j^h s_i = \eta\left(\sum_{k=1}^K \delta_k^o\, w_{jk}\right) \frac{df_j^h\left(a_j^h\right)}{da_j^h} \quad s_i$$

$$\Delta w_{ij}^h(m) = \eta \left(\sum_{k=1}^{K} (t_k - s_k^o) \frac{df_k^o(a_k^o)}{da_k^o} w_{jk} \right) \frac{df_j^h(a_j^h)}{da_j^h} s_i$$

Practical Considerations

- Stopping Criterion:
  - Threshold on average error
  - Threshold on average gradient
- Number of Weights:
  - Depends on number of input nodes, output nodes, hidden nodes
- Number of Hidden Nodes
  - Cross-validation method
- Data Requirements
- Limitations:
  - Slow convergence
  - Local minima problem

Feedforward Neural Networks: Summary

- Perceptrons, with threshold logic function as activation function, are suitable for pattern classification tasks that involve linearly separable classes.

- Multilayer feedforward neural networks, are suitable for pattern classification tasks that involve nonlinearly separable classes.
  - Complexity of the model to be used for a given task depends on
    - Dimension of the input pattern vector
    - Number of classes
    - Shapes of the decision surfaces to be formed
  - Architecture of the model is empirically determined
  - Large number of training examples are required when the complexity of the model is high
  - Local minima problem

- Multilayer feedforward neural network with one or two hidden layers is now called a shallow network.
- Multilayer feedforward neural network with more than two hidden layers is called a Deep Feedforward Neural Network (DNN).