Neural Network

Tozammel Hossain

Data Science & Analytics
University of Missouri

Artificial Neural Network (ANN)

- **Basic Idea: A complex non-linear function can be learned as a composition of simple processing units**
- **ANN is a collection of simple processing units (nodes) that are connected by directed links (edges)**
  - Every node receives signals from incoming edges, performs computations, and transmits signals to outgoing edges
  - Analogous to *human brain* where nodes are neurons and signals are electrical impulses
  - Weight of an edge determines the strength of connection between the nodes
- **Simplest ANN: Perceptron (single neuron)**

These slides are adapted from: Introduction to Data Mining by Tan et al.

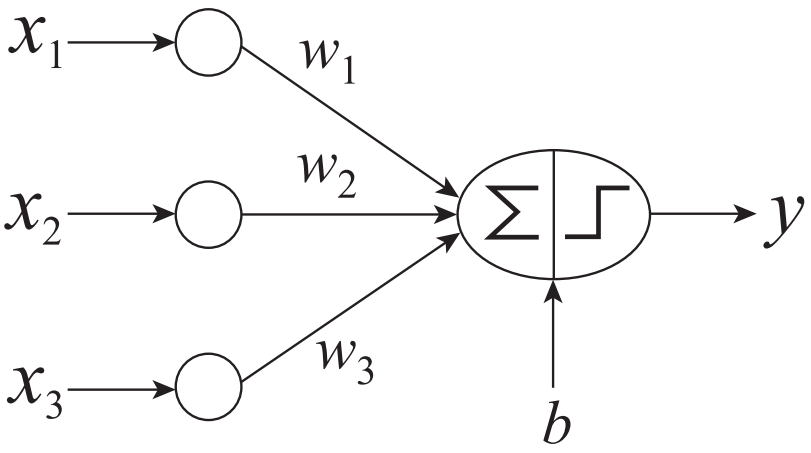
ANN vs Human Brain

- **ANN or simply NN are inspired by biological neuronal networks**
- **A real biological neuron, or a nerve cell, comprises dendrites, a cell body, and an axon that leads to synaptic terminals**
- **A neuron transmits information via electrochemical signals**
- **When there is enough concentration of ions at the dendrites of a neuron it generates an electric pulse along its axon called an action potential, which in turn activates the synaptic terminals, releasing more ions and thus causing the information to flow to dendrites of other neurons.**

ANN vs Human Brain

- **A human brain has on the order of 100 billion neurons**
  - Each neuron having between 1,000 to 10,000 connections to other neurons
- **ANN are comprised of abstract neurons that try to mimic real neurons at a very high level**
- **ANN can be described via a weighted directed graph**
  - Each node represents a neuron
  - Each directed edge represents a synaptic to dendritic connection between two nodes
  - The weight of the edge denotes the synaptic strength

Basic Architecture of Perceptron



3 features, 1 target

$$y = \begin{cases} 1, & \text{if } \mathbf{w}^T\mathbf{x} + b > 0. \\ -1, & \text{otherwise.} \end{cases}$$

$$\tilde{\mathbf{w}} = (\mathbf{w}^T \; b)^T \qquad \tilde{\mathbf{x}} = (\mathbf{x}^T \; 1)^T$$

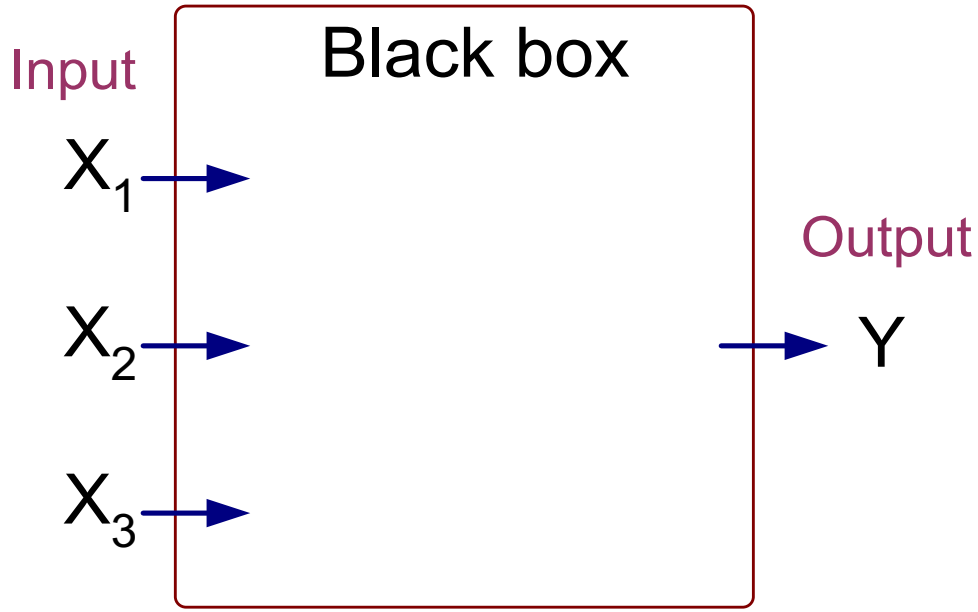
$$\hat{y} = sign(\tilde{\mathbf{w}}^T\tilde{\mathbf{x}})$$

Activation Function

- Learns linear decision boundaries
- Related to logistic regression (activation function is sign instead of sigmoid)

Perceptron Example

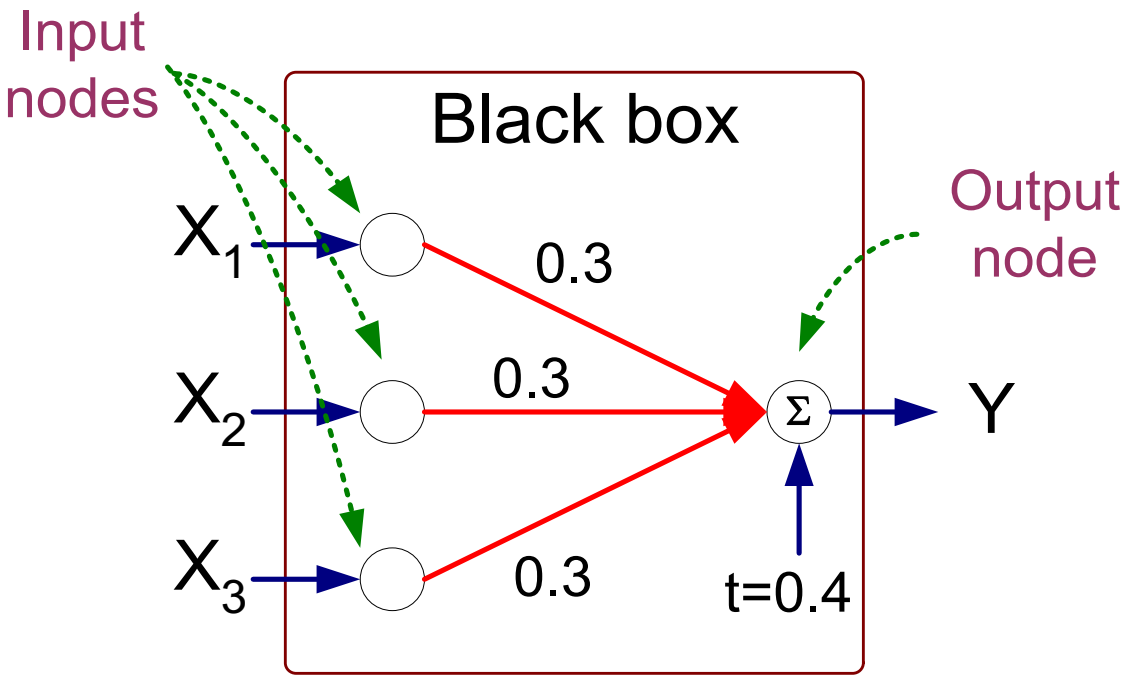
X_1	X_2	X_3	Y
1	0	0	-1
1	0	1	1
1	1	0	1
1	1	1	1
0	0	1	-1
0	1	0	-1
0	1	1	1
0	0	0	-1



Output Y is 1 if at least two of the three inputs are equal to 1.

Perceptron Example

X_1	X_2	X_3	Y
1	0	0	-1
1	0	1	1
1	1	0	1
1	1	1	1
0	0	1	-1
0	1	0	-1
0	1	1	1
0	0	0	-1



$$Y = sign(0.3X_1 + 0.3X_2 + 0.3X_3 - 0.4)$$

$$\text{where } sign(x) = \begin{cases} 1 & \text{if } x \geq 0 \\ -1 & \text{if } x < 0 \end{cases}$$

Perceptron Learning Rule

- **Initialize the weights (w_0, w_1, …, w_d)**
- **Repeat**
  - For each training example (x_i, y_i)
    - Compute $\hat{y}_i$
    - Update the weights:

$$w_j^{(k+1)} = w_j^{(k)} + \lambda\left(y_i - \hat{y}_i^{(k)}\right)x_{ij}$$

- **Until stopping condition is met**
- **k: iteration number;** λ**: learning rate**

Perceptron Learning Rule

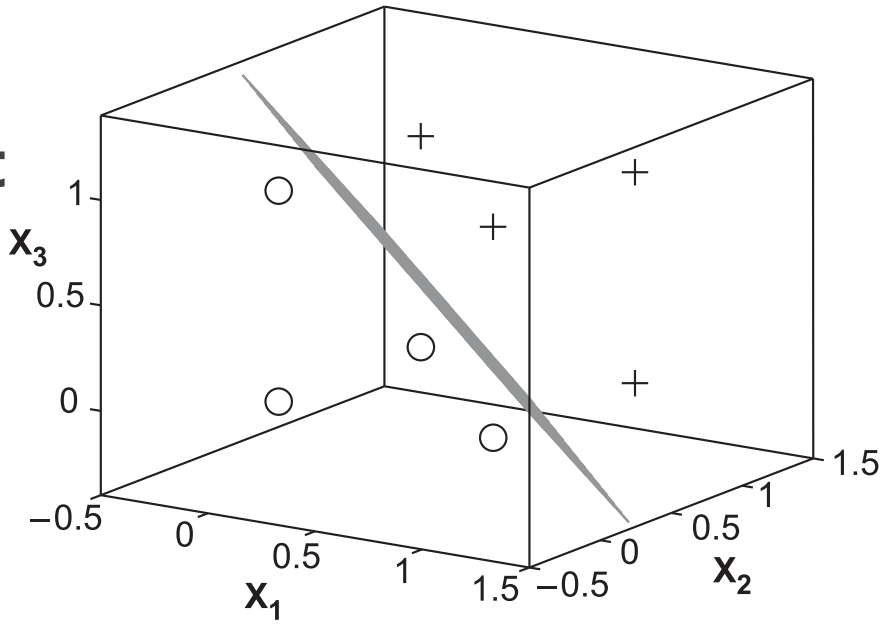
- **Weight update formula:**

$$w_j^{(k+1)} = w_j^{(k)} + \lambda\left(y_i - \hat{y}_i^{(k)}\right)x_{ij}$$

- **Intuition:**
  - Update weight based on error: e = $(y_i - \hat{y}_i)$
    - If y = $\hat{y}$, e=0: no update needed
    - If y > $\hat{y}$, e=2: weight must be increased (assuming Xij is positive) so that $\hat{y}$ will increase
    - If y < $\hat{y}$, e=-2: weight must be decreased (assuming Xij is positive) so that $\hat{y}$ will decrease

Perceptron Learning

- **Since y is a linear combination of input variables, decision boundary is linear**

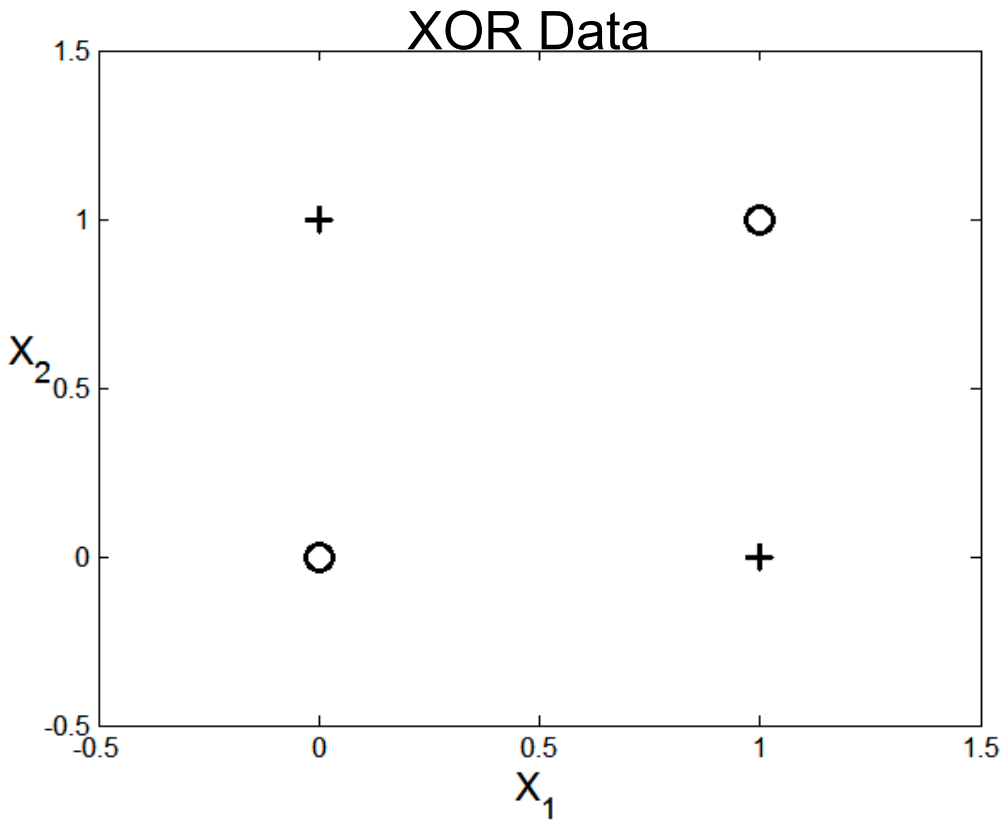


Nonlinearly Separable Data

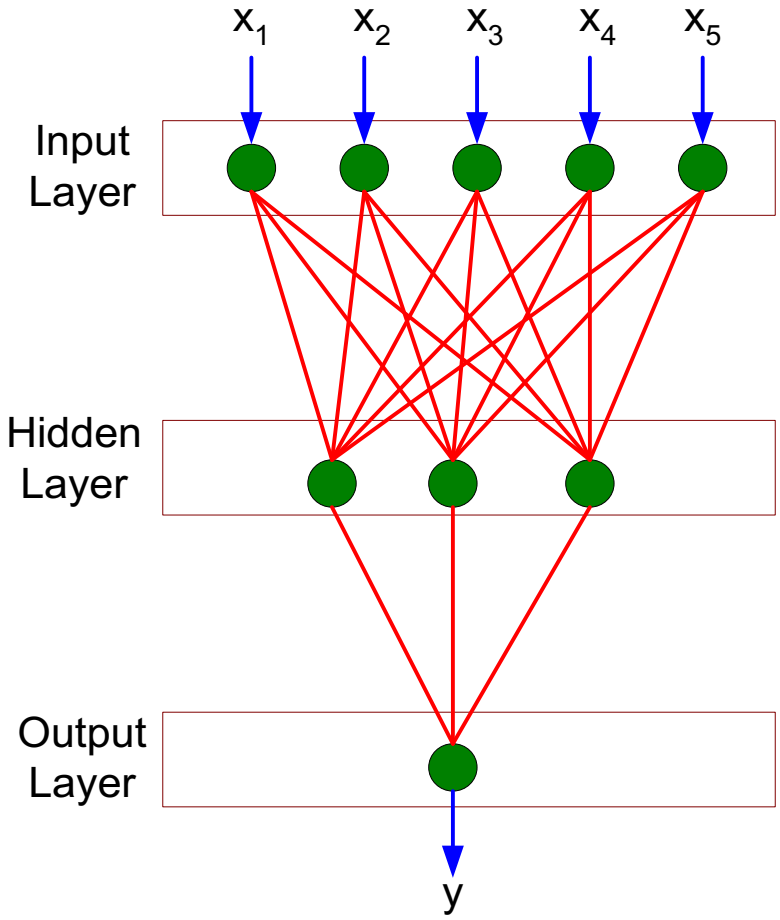
For nonlinearly separable problems, perceptron learning algorithm will fail because no linear hyperplane can separate the data perfectly

$$y = x_1 \oplus x_2$$

x_1	x_2	y
0	0	-1
1	0	1
0	1	1
1	1	-1



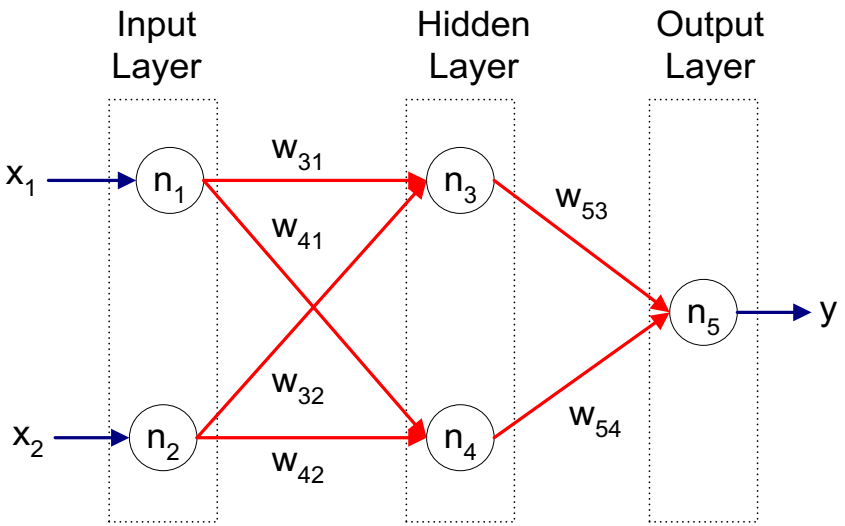
Multi-layer Neural Network

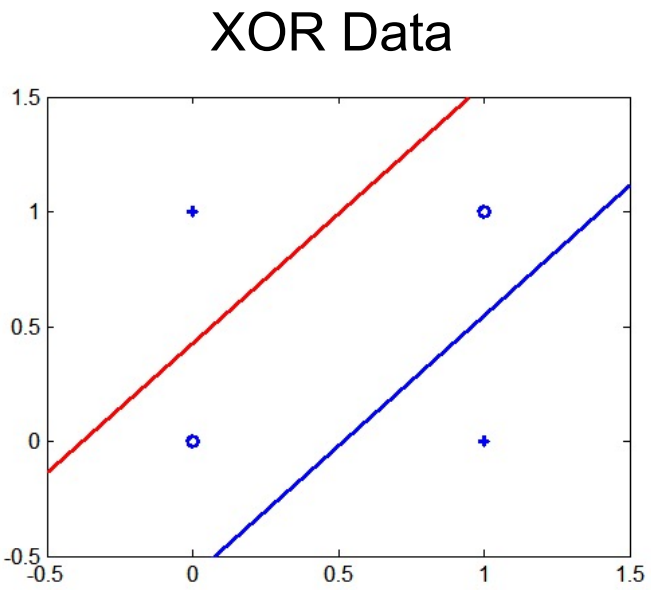


- More than one *hidden layer* of computing nodes
- Every node in a hidden layer operates on activations from preceding layer and transmits activations forward to nodes of next layer
- Also referred to as "feedforward neural networks"

Multi-layer Neural Network

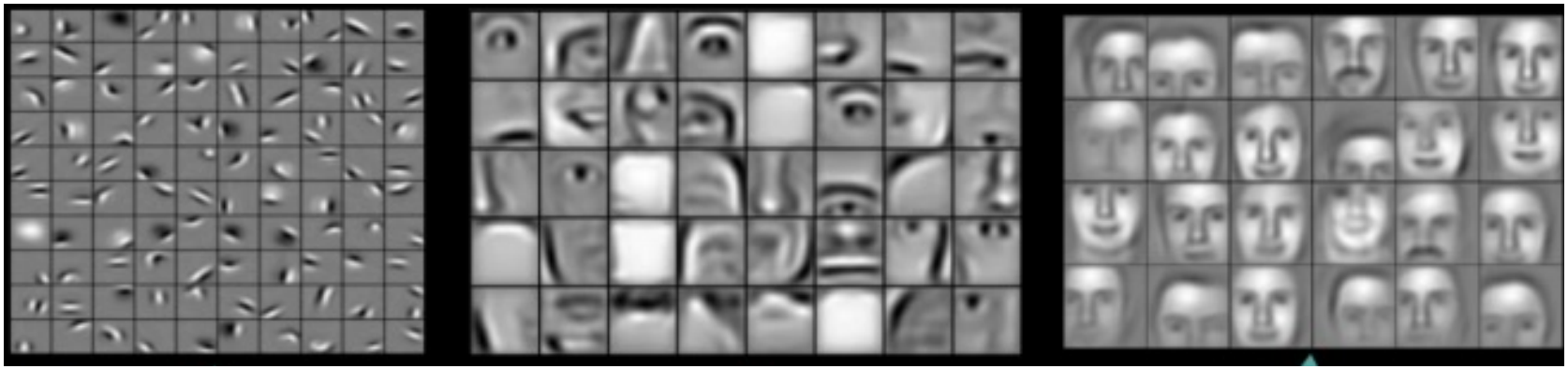
- **Multi-layer neural networks with at least one hidden layer can solve any type of classification task involving nonlinear decision surfaces**





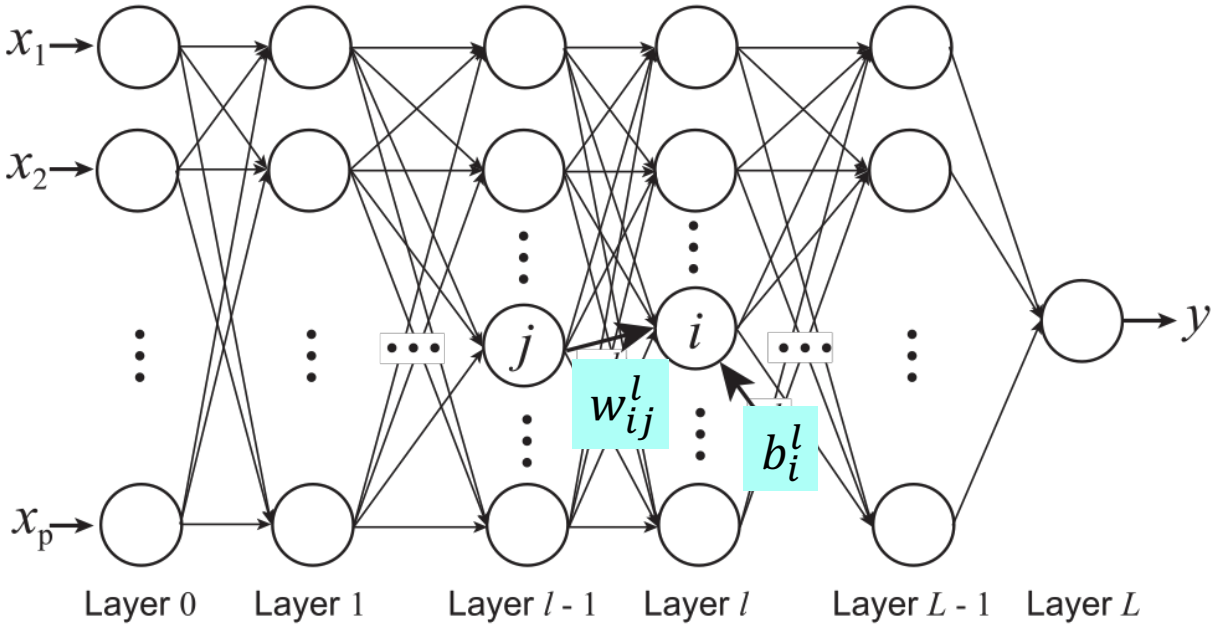
Why Multiple Hidden Layers?

- **Activations at hidden layers can be viewed as features extracted as functions of inputs**
- **Every hidden layer represents a level of abstraction**
  - *Complex features are compositions of simpler features*



- **Number of layers is known as depth of ANN**
  - *Deeper networks express complex hierarchy of features*

Multi-Layer Network Architecture



$$a_i^l = f(z_i^l) = f\Big(\sum_j w_{ij}^l a_j^{l-1} + b_i^l\Big)$$

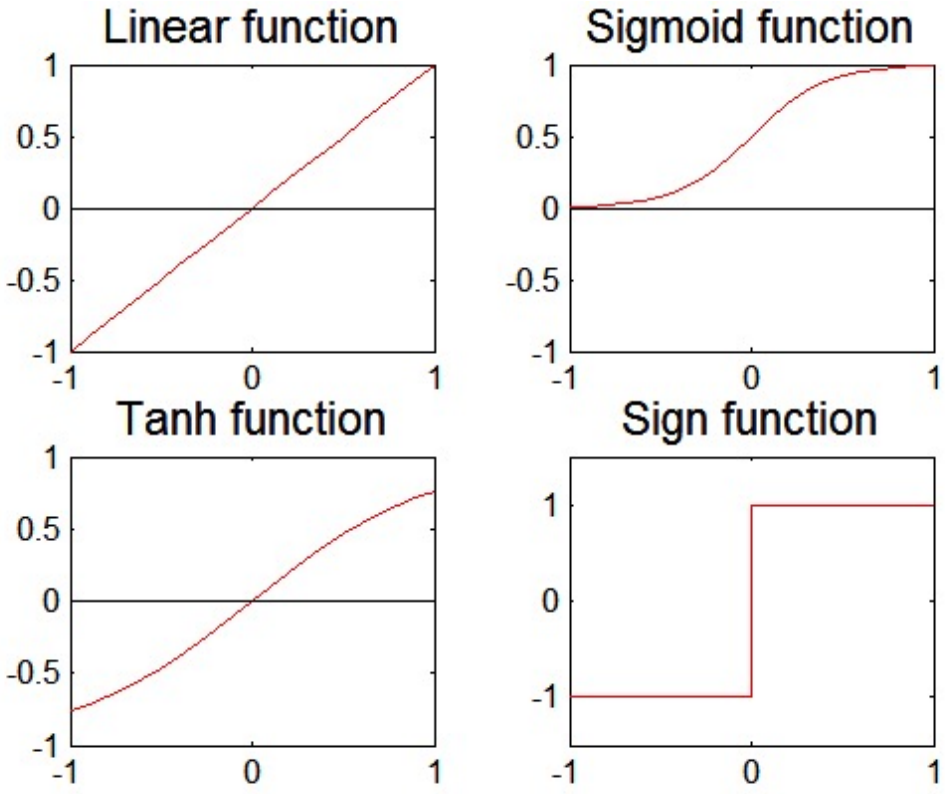
Activation value at node i at layer l

Activation Function

Linear Predictor

Activation Functions

$$a_i^l = f(z_i^l) = f\Big(\sum_j w_{ij}^l a_j^{l-1} + b_i^l\Big)$$



$$a_i^l = \sigma(z_i^l) = \frac{1}{1+e^{-z_i^l}}.$$

$$\frac{\partial a_i^l}{\partial z_i^l} = \frac{\partial\ \sigma(z_i^l)}{\partial z_i^l} = a_i^l(1-a_i^l)$$

Design Issues in ANN

- **Number of nodes in input layer**
  - One input node per **binary/continuous** attribute
  - k or $\log_2$ k nodes for each **categorical** attribute with k values
- **Number of nodes in output layer**
  - One output for binary class problem
  - k or $\log_2$ k nodes for k-class problem
- **Number of hidden layers and nodes per layer**
- **Initial weights and biases**
- **Learning rate, max. number of epochs, mini-batch size for mini-batch SGD, …**

Characteristics of ANN

- **Multilayer ANN are universal approximators but could suffer from overfitting if the network is too large**
  - Naturally represents a hierarchy of features at multiple levels of abstractions
- **Gradient descent may converge to local minimum**
- **Model building is computing intensive, but testing is fast**
- **Can handle redundant and irrelevant attributes because weights are automatically learnt for all attributes**
- **Sensitive to noise in training data**
  - This issue can be addressed by incorporating model complexity in the loss function
- **Difficult to handle missing attributes**

Deep Learning Trends

- **Training deep neural networks (more than 5-10 layers) could only be possible in recent times with:**
  - Faster computing resources (GPU)
  - Larger labeled training sets
- **Algorithmic Improvements in Deep Learning**
  - Responsive activation functions (e.g., RELU)
  - Regularization (e.g., Dropout)
  - Supervised pre-training
  - Unsupervised pre-training (auto-encoders)
- **Specialized ANN Architectures:**
  - Convolutional Neural Networks (for image data)
  - Recurrent Neural Networks (for sequence data)
  - Residual Networks (with skip connections)
- **Generative Models: Generative Adversarial Networks**