

Neural Network Basics

Weizi Li

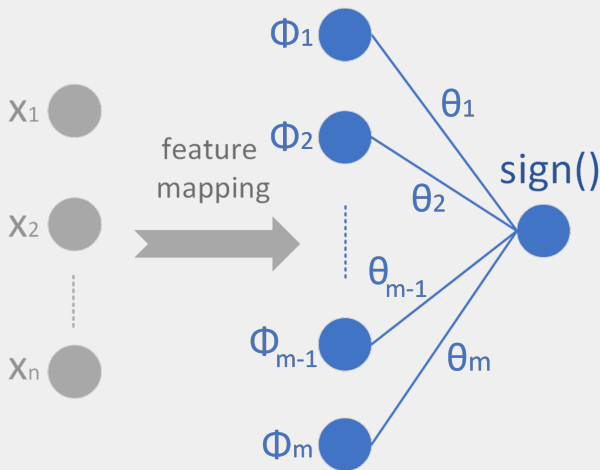
Department of Computer Science
University of Memphis



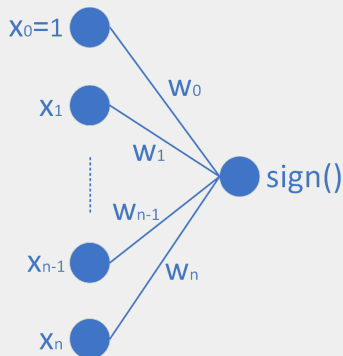
- Recall non-linear classifier via feature mapping:

$$h(x; \theta, \theta_0) = \textit{sign}(\theta \phi(x))$$

- $x = [x_1, x_2, \dots, x_n]$
- $\phi(x) = [\phi_1(x), \phi_2(x), \dots, \phi_m(x)]$
- $\theta = [\theta_1, \theta_2, \dots, \theta_m]$



- x can be raw data but more commonly, features extracted from raw data



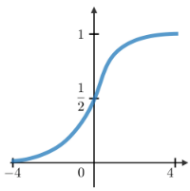
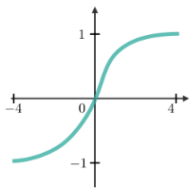
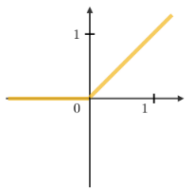
- If $\sum_{i=0}^n x_i w_i > 0$, x is $+$, otherwise x is $-$

- A simple **neural network** that delivers a **linear classifier** for binary classification
- Guarantee to solve any binary classification task if data are linearly separable

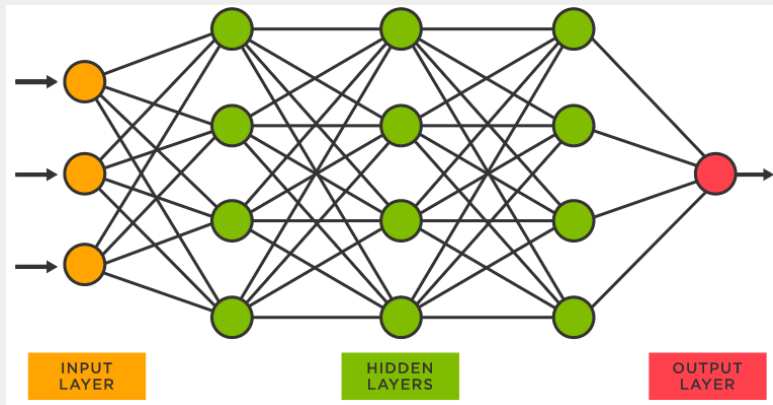
Algorithm 1 Perceptron Learning Algorithm

Require: $\{(X^{(i)}, y^{(i)})\}_{i=1, \dots, n}; T$
 for $t = 1, \dots, T$ **do**
 for $i = 1, \dots, n$ **do**
 if $y^{(i)}(wX^{(i)} + w_0) \leq 0$ **then**
 $w_0 = w_0 + y^{(i)}$
 $w = w + y^{(i)}X^{(i)}$
 end if
 end for
 end for
 return w_0, w

- Use a non-linear activation function instead of simple $\text{sign}()$
- $g(z = \sum_{i=0}^n x_i w_i)$: Sigmoid, Tanh, ReLU, etc.

Sigmoid	Tanh	ReLU
$g(z) = \frac{1}{1 + e^{-z}}$	$g(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$	$g(z) = \max(0, z)$
		

- Neural networks with **multiple hidden layers**; backbone for “deep learning”



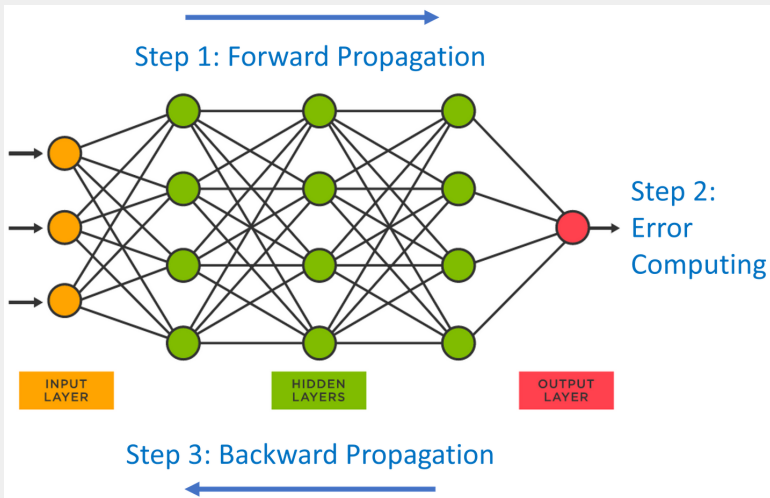
- Number of hidden layers
- Number of units of each layer
- Activation function in the units (usually the same function is used for the entire network)
- Ideal architecture is guided by monitoring validation error

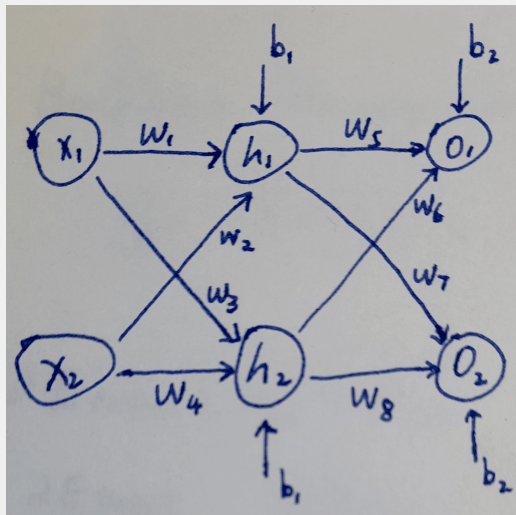
- Playground ([link](#))

- Fully-connected: all units of a layer connect to all units of the next layer
- Convolutional neural networks are **not** fully connected

- Feedforward: connections among units do not form a cycle
 - ▶ Fully-connected, feedforward neural networks are also called multilayer perceptron (MLP)
- Recurrent: connections among units form a cycle

- The concepts of “multiple layers”, “connected”, and “activation function” are loosely from neuroscience
- Modern neural networks are guided by mathematical and engineering principles
- Better to think modern neural networks as **function approximators to achieve statistical generalizability**





- Activation function (h_1, h_2, o_1, o_2): Sigmoid
- Input: $x_1 = 0.05, x_2 = 0.1$
- Weights: $w_1 = 0.15, w_2 = 0.2, w_3 = 0.25, w_4 = 0.3, w_5 = 0.4, w_6 = 0.45, w_7 = 0.5, w_8 = 0.55$
- Bias: $b_1 = 0.35, b_2 = 0.6$
- Label: $y_1 = 0.01, y_2 = 0.99$
- Learning rate: $\alpha = 0.5$
- Loss function: $\frac{1}{2}(y - \hat{y})^2, y = [y_1, y_2], \hat{y} = [\hat{y}_1, \hat{y}_2]$

- Try to solve this example yourself! The answers are provided below.
- Input and output of h_1 : $IN_{h_1} = 0.3775$, $OUT_{h_1} = 0.5933$
- Input and output of h_2 : $IN_{h_2} = 0.3925$, $OUT_{h_2} = 0.5969$
- Input and output of o_1 : $IN_{o_1} = 1.1059$, $OUT_{o_1} = 0.7514$
- Input and output of o_2 : $IN_{o_2} = 1.2249$, $OUT_{o_2} = 0.7729$

- Error at o_1 : $E_{o_1} = \frac{1}{2}(y_1 - OUT_{o_1})^2 = 0.2748$
- Error at o_2 : $E_{o_2} = \frac{1}{2}(y_2 - OUT_{o_2})^2 = 0.0236$
- Total error: $E_{total} = E_{o_1} + E_{o_2} = 0.2984$

- An example: $w_5 = w_5 - \alpha \frac{\partial E_{total}}{\partial w_5} = 0.3589$
- Use one training data point to update all parameters (i.e., from w_1 to w_8); then move on to the next training data point
- Epoch: going through the entire training set