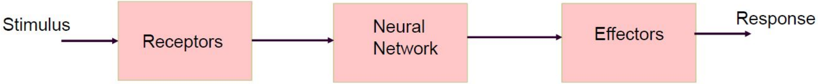
Neural Networks
CS-477 Computer Vision

Dr. Mohsin Kamal
Associate Professor
dr.mohsinkamal@seecs.edu.pk

School of Electrical Engineering and Computer Science (SEECS)
National University of Sciences and Technology (NUST), Pakistan

Motivation:

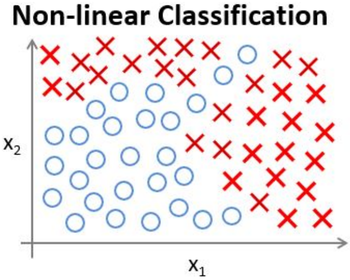


- The brain is a complex, nonlinear and distributed computer having neurons as its basic information processing units (different than traditional computers).
- The brain has the ability to perform several tasks such as pattern recognition, perception and motor control very well, despite being slow in information processing.
- Therefore, the motivation is to mimic the functioning neurons and neural networks *in-silico* so as to build machines that have very high capabilities.

Artificial Neural Networks - History

- 1943: computational model for neural networks (McCulloch & Pitts)
- 1958: Perceptron was created (Rosenblatt)
- 1969: was shown that Perceptrons were not powerful (Minsky & Papert)
- 1986: multi-layer perceptrons (Rumelhart & McClelland)
- 1990: IEEE Transactions on Neural Networks
- 2010– : Deep learning, wide number of applications

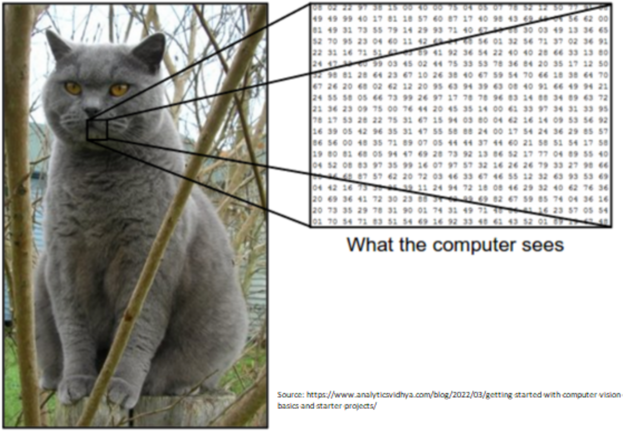
Consider a supervised learning classification problem...
If you apply the logistic machine learning algorithm, then



$$g(\theta_0 + \theta_1 x_1 + \theta_2 x_2 \\ +\theta_3 x_1 x_2 + \theta_4 x_1^2 x_2 \\ +\theta_5 x_1^3 x_2 + \theta_6 x_1 x_2^2 + \ldots)$$

- If you have more features, e.g, x_1 = size, x_2 = # of bedrooms, x_3 = # of rooms, x_4 = age $\cdots$, x_{100} = # of floors and for example, you want to predict that what are the odds that the house will be sold out in next six months?
- Here, we came up with $n = 100$.
- If we have to include all the 2-order polynomials, then we will get ≈ 5000 features having complexity $O(n^2)$.
- $x_1^2, x_1x_2, x_1x_3, \cdots, x_2^2, x_2x_3, \cdots$
- The features can be reduced by choosing only squared polynomials i.e., $x_1^2, x_2^2, x_3^2, \cdots$
- This does not provide a good fit of hypothesis.
- If 3-order or cubic polynomials are included with the squared ones then having $n = 100$, we can have 170000 cubic features which will make it reach to $O(n^3)$.

For many machine learning problems, *n* will be very large. For example



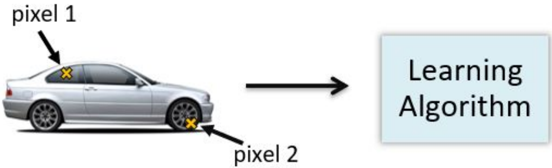
Source: https://www.analyticsvidhya.com/blog/2022/03/getting started with computer vision basics and starter projects/

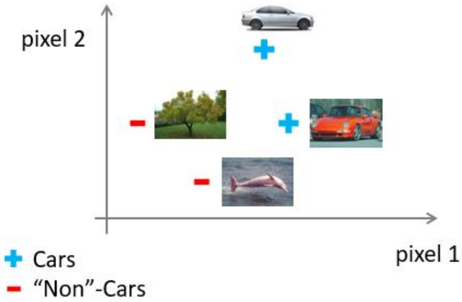
Computer Vision: Car detection

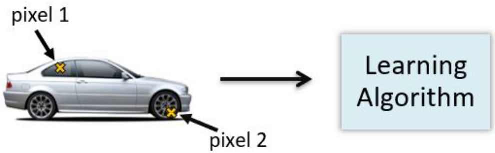
Testing:

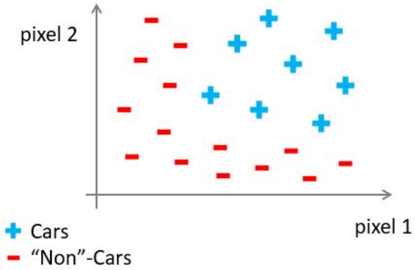
What is this?

Lets examine why we need non-linear hypothesis:









50 x 50 pixel images→ 2500 pixels
$n = 2500$ (7500 if RGB)

$$x = \begin{bmatrix} \text{pixel 1 intensity} \\ \text{pixel 2 intensity} \\ \vdots \\ \text{pixel 2500 intensity} \end{bmatrix}$$

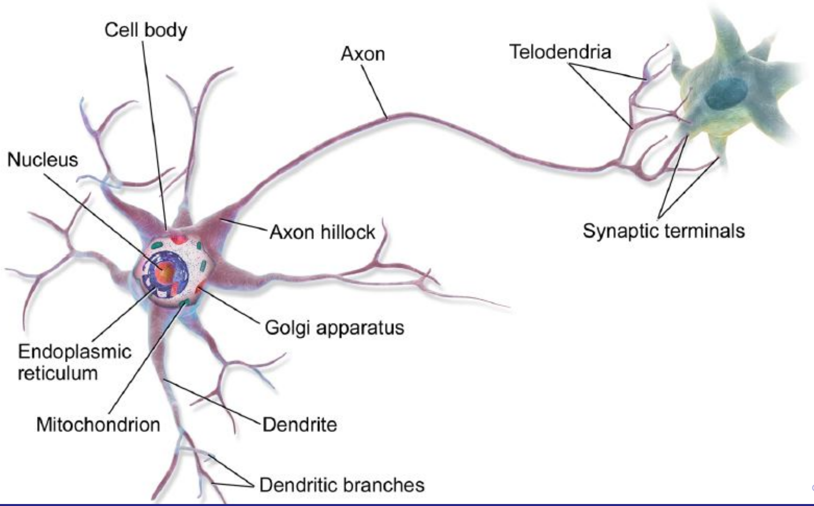
Quadratic features ($x_i \times x_j$): ≈3 million features

x will be having values from 0-255 if considering grey scale image and 7500 if RGB.

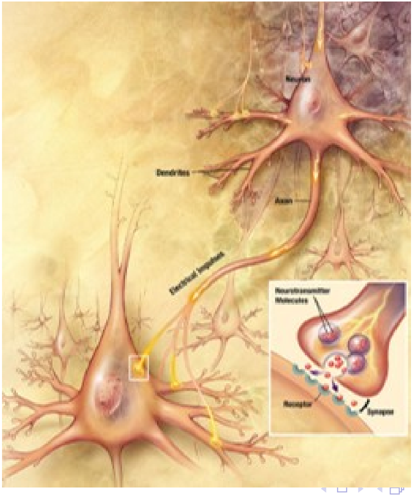
Neural Networks

- **Origins:** Algorithms that try to mimic the brain.
- Was very widely used in 80s and early 90s; popularity diminished in late 90s.
- Recent resurgence: State-of-the-art technique for many applications

Neuron in the brain



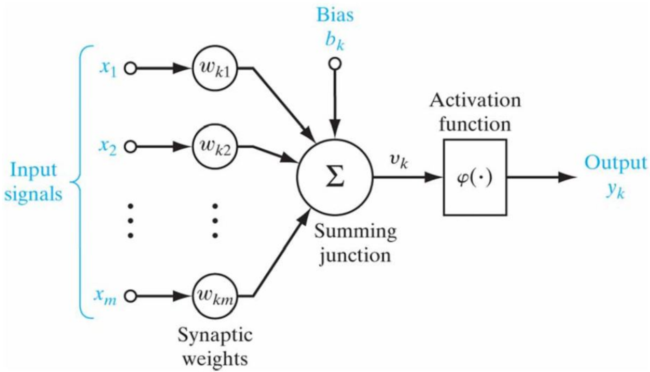
Neuron in the brain



NEURON:

- On the average, humans have 86 billion neurons in their brain and approximately 100 trillion synapses. There are 16 billion neurons in the forebrain.
- Neuron consists of a cell body, dendrites and an axon
- Neurons are massively interconnected by synapses
- Synapse is an interconnection between the axon of one neuron and a dendrite of another neuron
- Information propagation is achieved via electro-chemical signals

Artificial Neurons:



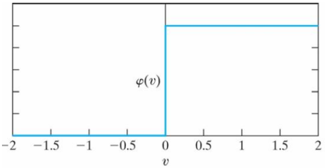
$$v_k = b_k + \sum_{i=1}^{m} w_{ki} x_i = \sum_{i=0}^{m} w_{ki} x_i, \quad w_{k0} = b_k, x_0 = 1$$

Activation Functions

Threshold function:

Neurons with this activation function can only be in two states: "on" and "off".

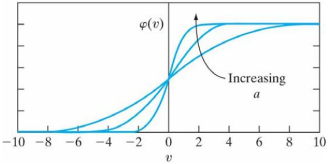
$$\varphi(u) = \begin{cases} 1, & u \geq u_{th}, \\ 0, & u < u_{th} \end{cases}$$



Logistic Sigmoid function:

The slope shows how fast a neuron moves from the "off" state to the "on" state:

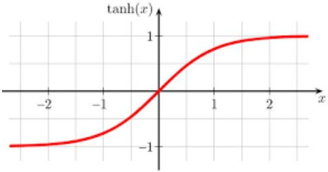
$$\varphi(u) = \frac{1}{1 + e^{-au}}$$



Activation Functions

Hyperbolic function:

$$\varphi(u) = \tanh(u)$$

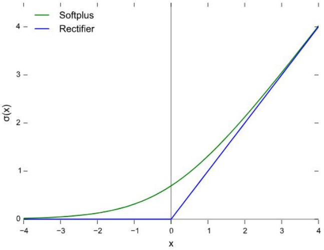


Rectifier Linear Unit (ReLU):

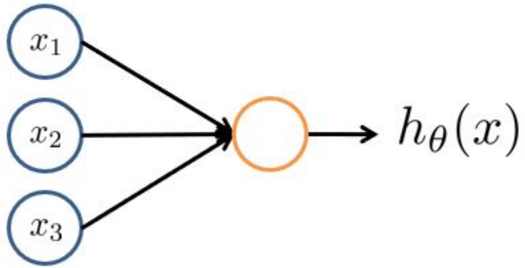
$$\varphi(u) = max(0, u) \triangleq u^+$$

Softplus or SmoothReLU:

$$\varphi(u) = ln(1 + e^u)$$



Neuron model: Logistic unit

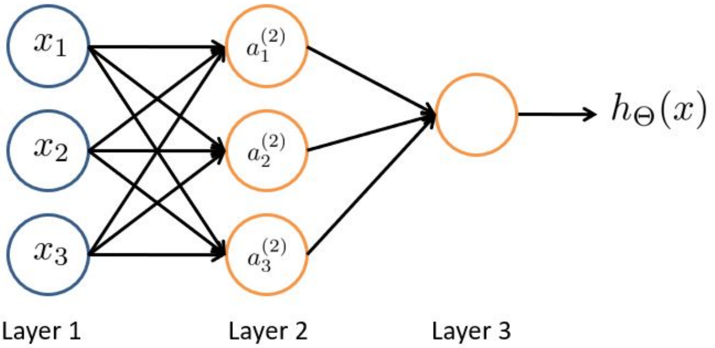


$$\text{Where } x = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix} \qquad \theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \\ \theta_3 \end{bmatrix}$$

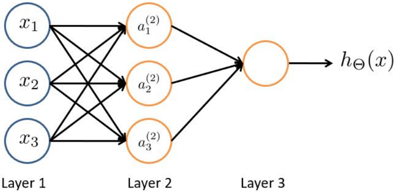
Sometimes we say that it is artificial neuron having Sigmoid (logistic) activation function.

Neural Network

Neural network is a group of neurons.



Neural Network

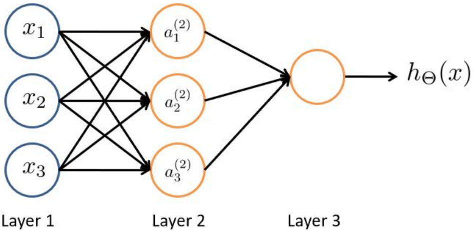


$a_i^{(j)}$ = "activation" of unit i in layer j
$\Theta^{(j)}$ = matrix of weights controlling function mapping from layer j to layer $j+1$

$$a_1^{(2)} = g(\Theta_{10}^{(1)}x_0 + \Theta_{11}^{(1)}x_1 + \Theta_{12}^{(1)}x_2 + \Theta_{13}^{(1)}x_3)$$
$$a_2^{(2)} = g(\Theta_{20}^{(1)}x_0 + \Theta_{21}^{(1)}x_1 + \Theta_{22}^{(1)}x_2 + \Theta_{23}^{(1)}x_3)$$
$$a_3^{(2)} = g(\Theta_{30}^{(1)}x_0 + \Theta_{31}^{(1)}x_1 + \Theta_{32}^{(1)}x_2 + \Theta_{33}^{(1)}x_3)$$
$$h_\Theta(x) = a_1^{(3)} = g(\Theta_{10}^{(2)}a_0^{(2)} + \Theta_{11}^{(2)}a_1^{(2)} + \Theta_{12}^{(2)}a_2^{(2)} + \Theta_{13}^{(2)}a_3^{(2)})$$

If network has s_j units in layer j, s_{j+1} units in layer $j+1$, then *Theta*$^{(j)}$ will be of dimension $s_{j+1} \times (s_j + 1)$.

Forward propagation: Vectorized implementation

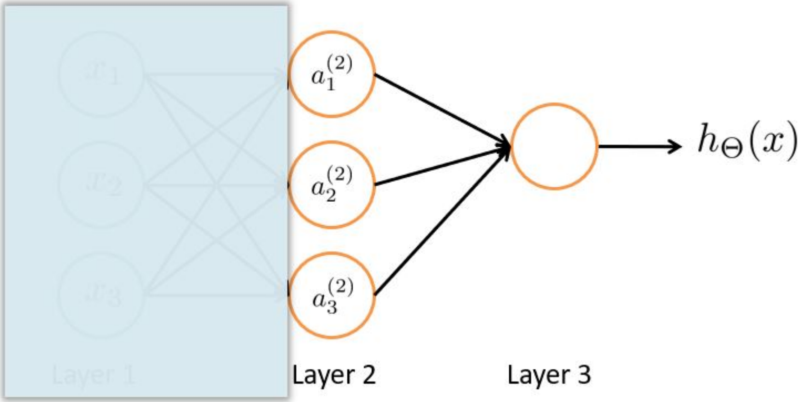


$$a_1^{(2)} = g(\Theta_{10}^{(1)} x_0 + \Theta_{11}^{(1)} x_1 + \Theta_{12}^{(1)} x_2 + \Theta_{13}^{(1)} x_3)$$
$$a_2^{(2)} = g(\Theta_{20}^{(1)} x_0 + \Theta_{21}^{(1)} x_1 + \Theta_{22}^{(1)} x_2 + \Theta_{23}^{(1)} x_3)$$
$$a_3^{(2)} = g(\Theta_{30}^{(1)} x_0 + \Theta_{31}^{(1)} x_1 + \Theta_{32}^{(1)} x_2 + \Theta_{33}^{(1)} x_3)$$
$$h_\Theta(x) = a_1^{(3)} = g(\Theta_{10}^{(2)} a_0^{(2)} + \Theta_{11}^{(2)} a_1^{(2)} + \Theta_{12}^{(2)} a_2^{(2)} + \Theta_{13}^{(2)} a_3^{(2)})$$

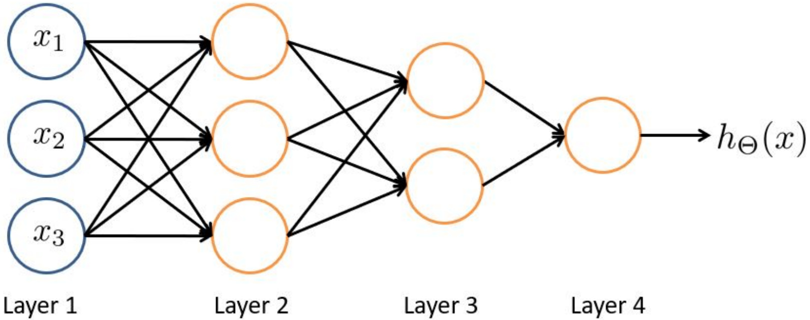
$$x = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix} \quad z^{(2)} = \begin{bmatrix} z_1^{(2)} \\ z_2^{(2)} \\ z_3^{(2)} \end{bmatrix}$$

$$z^{(2)} = \Theta^{(1)} a^{(1)}$$
$$a^{(2)} = g(z^{(2)})$$
$$\text{Add } a_0^{(2)} = 1$$
$$z^{(3)} = \Theta^{(2)} a^{(2)}$$
$$h_\Theta(x) = a^{(3)} = g(z^{(3)})$$

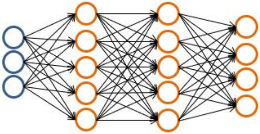
Neural Network learning its own features



Other network architectures



Multiple output units: One-vs-all.

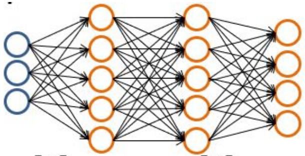


$$h_\Theta(x) \in \mathbb{R}^4$$

Want $h_\Theta(x) \approx \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}$, $h_\Theta(x) \approx \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}$, $h_\Theta(x) \approx \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}$, etc.

when pedestrian when car when motorcycle

Multiple output units: One-vs-all.



$$h_\Theta(x) \in \mathbb{R}^4$$

Want $h_\Theta(x) \approx \begin{bmatrix}1\\0\\0\\0\end{bmatrix}$, $h_\Theta(x) \approx \begin{bmatrix}0\\1\\0\\0\end{bmatrix}$, $h_\Theta(x) \approx \begin{bmatrix}0\\0\\1\\0\end{bmatrix}$, etc.

when pedestrian when car when motorcycle

Training set: $(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \ldots, (x^{(m)}, y^{(m)})$

$y^{(i)}$ one of $\begin{bmatrix}1\\0\\0\\0\end{bmatrix}$, $\begin{bmatrix}0\\1\\0\\0\end{bmatrix}$, $\begin{bmatrix}0\\0\\1\\0\end{bmatrix}$, $\begin{bmatrix}0\\0\\0\\1\end{bmatrix}$

pedestrian car motorcycle truck