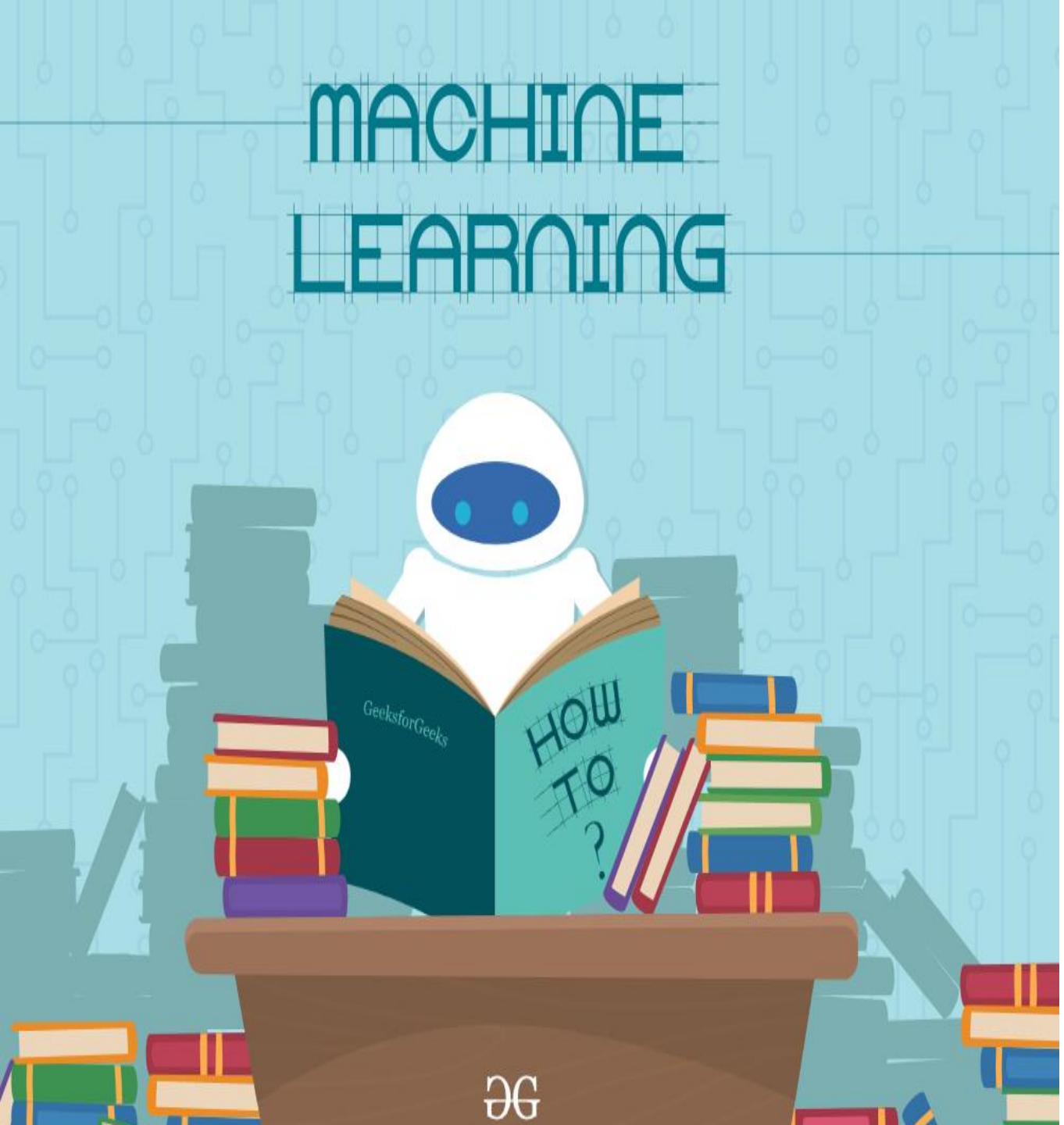


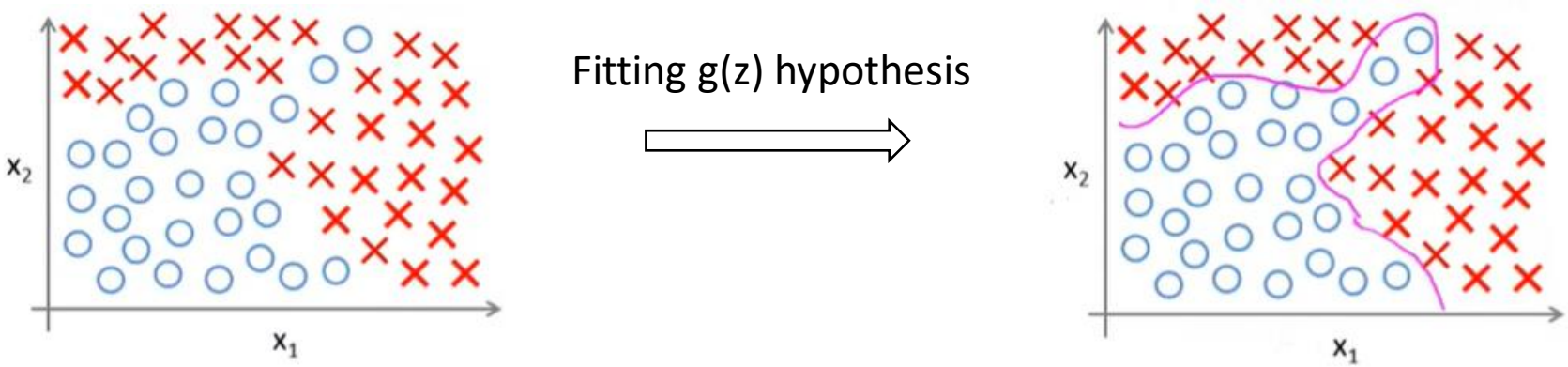
>TCC
THE CODERS' CLUB

NEURAL NETWORKS (REPRESENTATION)

WEEK 4

Non-linear Hypothesis

Consider a supervised learning classification problem as shown below.



There are two features – x_1 and x_2

Applying logistic regression to this problem with a lot of non-linear features (polynomial terms) gives a hypothesis i.e. sigmoid function which may look something like this for example.

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

This particular method works well only when we have two features. (x_1 and x_2)

So what if n>2?

Example:

x_1 = size
x_2 = # bedrooms
x_3 = # floors
x_4 = age
…
x_{100}

For a problem like this, if we were to include all the quadratic terms (second order terms), we may end up getting thousands of features.

Example:

x_1^2, x_1x_2, x_1x_3, x_1x_4, x_1x_{100}, x_2^2, x_2x_3, … and so on (≈ 5000 features)

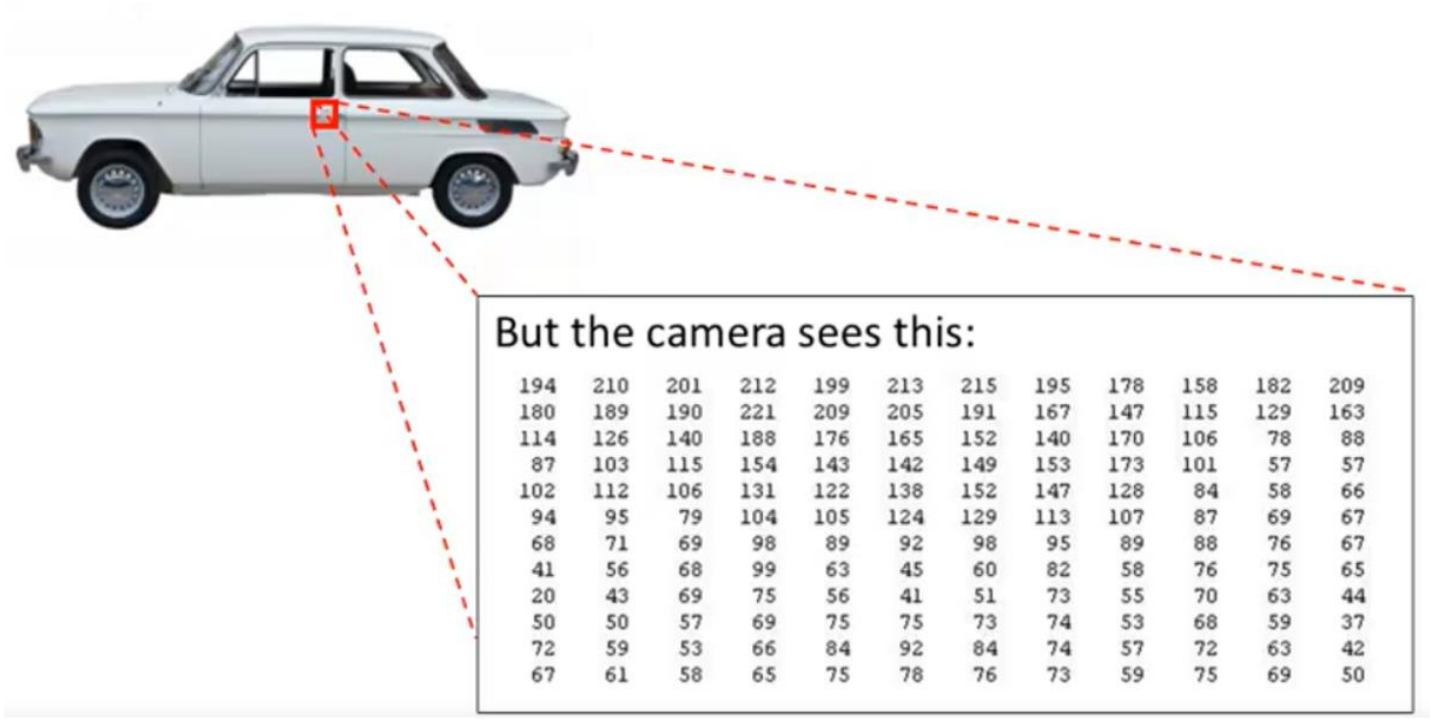
- No. of quadratic features is of the order of $O(n^2)$
- The no. of quadratic features is approximately equal to $n^2/2$

Thus, for n=100, no. of quadratic features = (100 x 100)/2 = 5000 is a lot which is computationally expensive and may end up with overfitting the training set.

For many Machine Learning problems, no. of features (n) will be very large.

Example: Computer Vision

Train a classifier to examine an image and tell us whether or not the image is a car.



⇐ Pixel Intensity Values

So, the Computer Vision problem here is to look at this matrix of pixel intensity values and tell us that these numbers represent the door handle of a car.

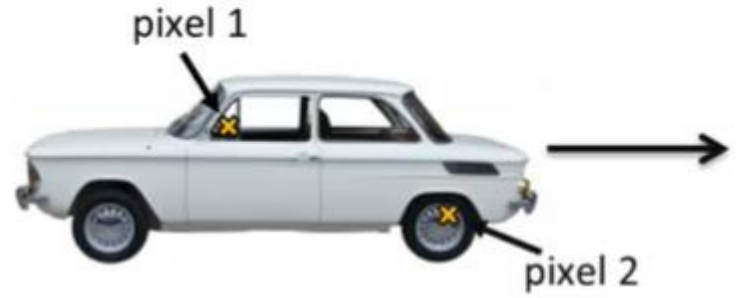
When we use Machine Learning to build a **car detector**, following are the steps –

- Prepare a labelled training set of cars
- Prepare a labelled training set of non-cars
- Training set is then given to a learning algorithm to train the classifier
- Test the classifier by giving a new image (test data)

Testing:

What is this?

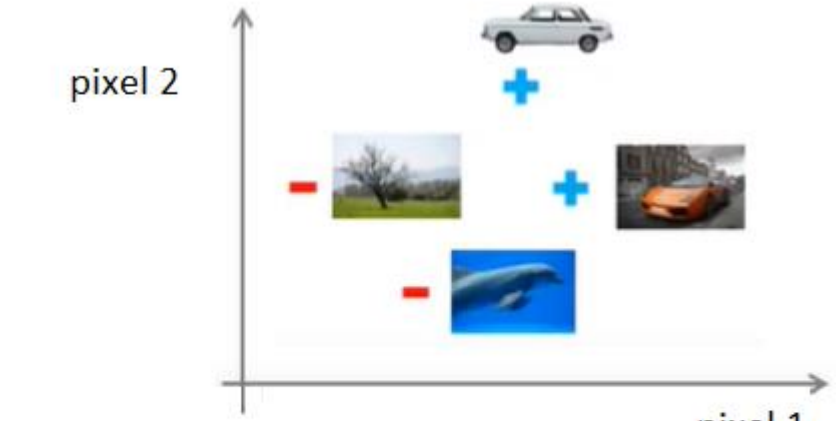
To understand why we need non-linear hypothesis, let's take a look at some of the images of cars and non-cars that we give as an input to our learning algorithm.



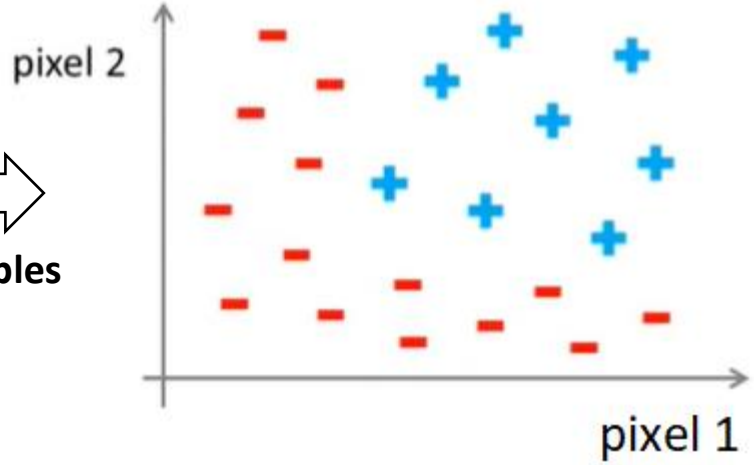
Learning Algorithm

Pick pixel1 and pixel2 locations and feed them in the learning algorithm.
The locations denote the intensity values.

- We take some more images of cars and non-cars and plot their pixel intensity values as shown.

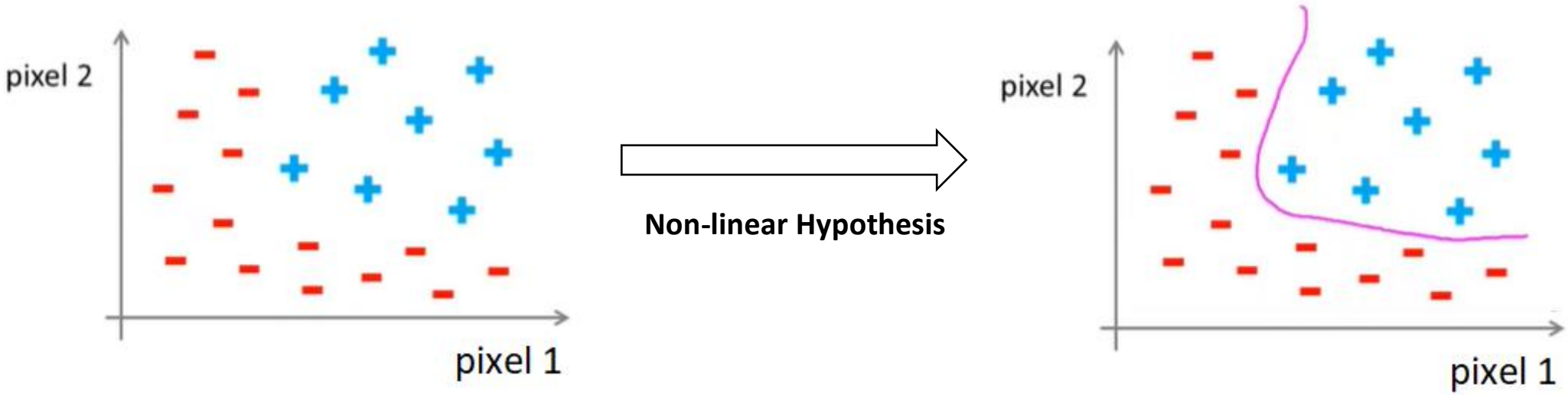


For more training examples



Cars

Non-cars



What is the dimension of the feature space?

Suppose our images are 50x50 pixels, then the dimension of the feature space will be, n=2500.

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

Note:
For grayscale images, n= 50 x 50 = 2500
For RGB images, n = 50 x 50 x 3 = 7500

No. of quadratic features = $x_i \cdot x_j \approx$ 3 million features

The value of pixel intensities range between 0-255.

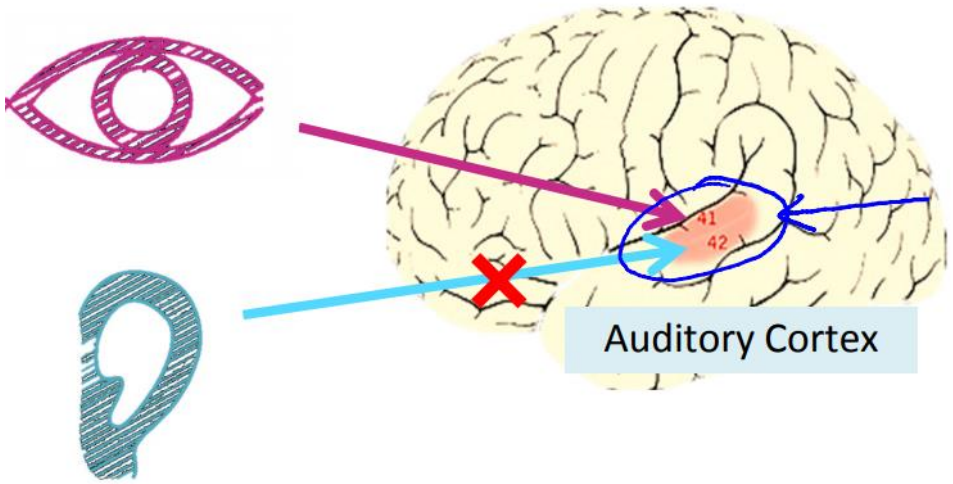
Suppose you are learning to recognize cars from 100×100 pixel images (grayscale, not RGB). Let the features be pixel intensity values. If you train logistic regression including all the quadratic terms ($x_i x_j$) as features, about how many features will you have?

- 5,000
- 100,000
- 50 million (5×10^7)
- 5 billion (5×10^9)

Neural Networks: Representation

- Neural Networks are algorithms that try to mimic the brain.
- They were widely used in 80s and early 90s but their popularity diminished in the late 90s.

Example 1:



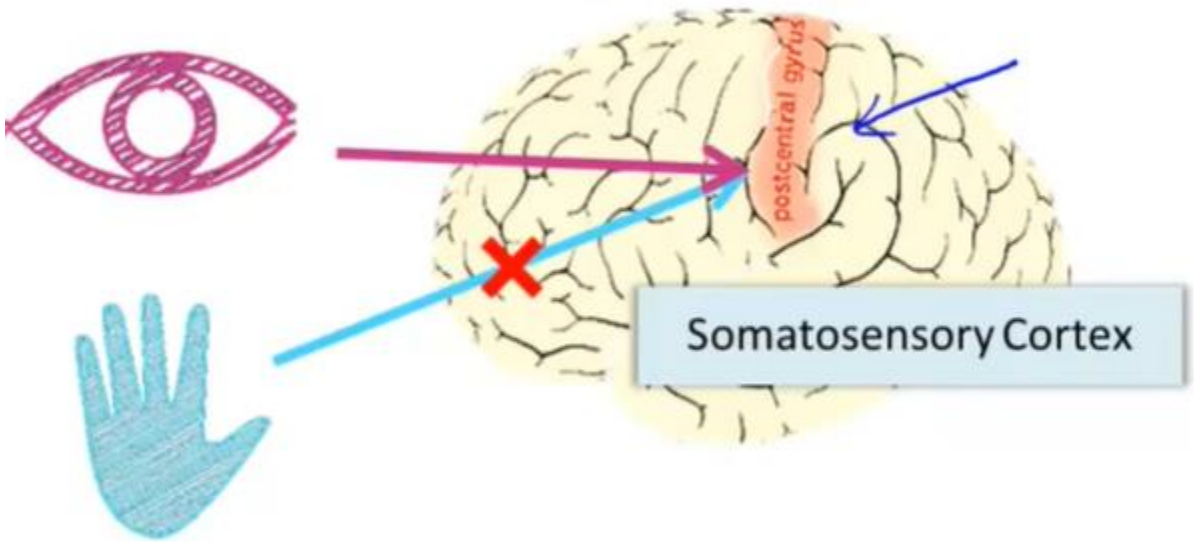
Sound signals from our ears are routed to the auditory cortex which helps us to hear.

Signals from our eyes are routed to the optic nerve which helps us to see.

If the signal from the eyes to the optic nerve is re-routed to the auditory cortex, the **auditory cortex learns to see!**

[Source: Roe et al., 1992]

Example 2:



If the signals from the eyes to optic nerve are re-routed to the somatosensory cortex (a brain tissue that senses touch), the **somatosensory cortex learns to see!**

[Source: Metin & Frost, 1989]

Sensor representations in the brain

Seeing with your tongue

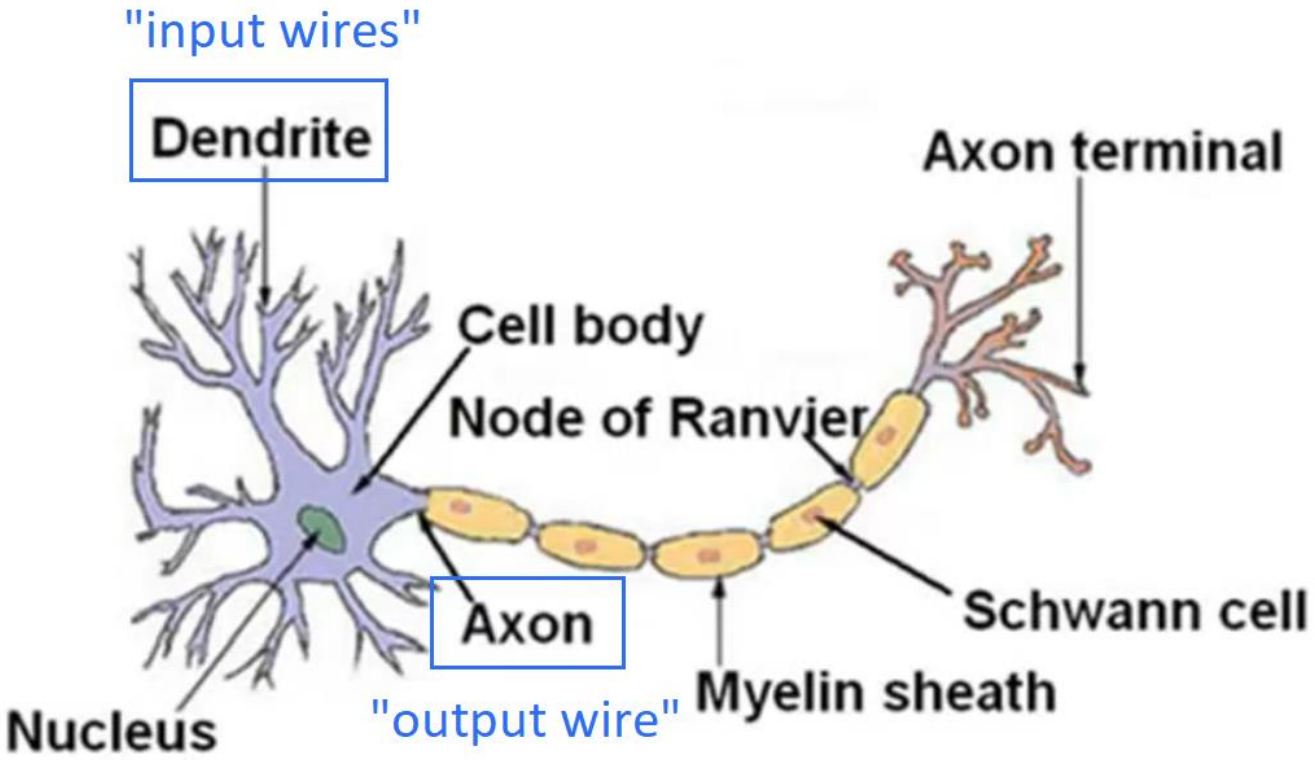
Human echolocation (sonar)

Haptic belt: Direction sense

Implanting a 3rd eye

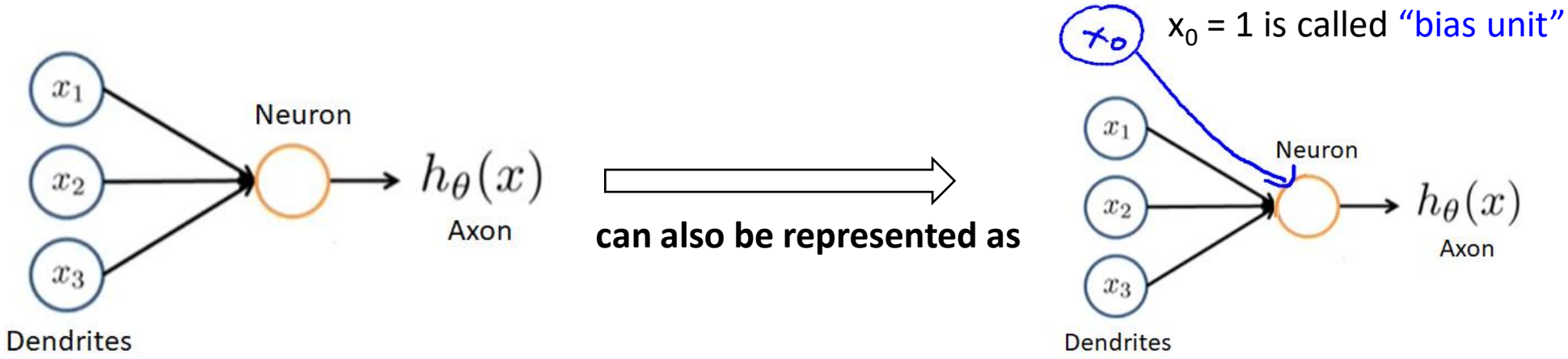
[Source: BrainPort; Welsh & Blasch, 1997; Nagel et al., 2005; Constantine-Paton & Law, 2009]

Neural Networks: Model Representation-I



Neurons are computational units that take **inputs (dendrites)** as electrical inputs (called “spikes”) that are channeled to **outputs (axons)**

Neuron Model: Logistic Unit



The neuron takes some inputs, does certain computations and gives the output

$h_\theta(x) = \frac{1}{1+e^{-\theta^T x}}$.

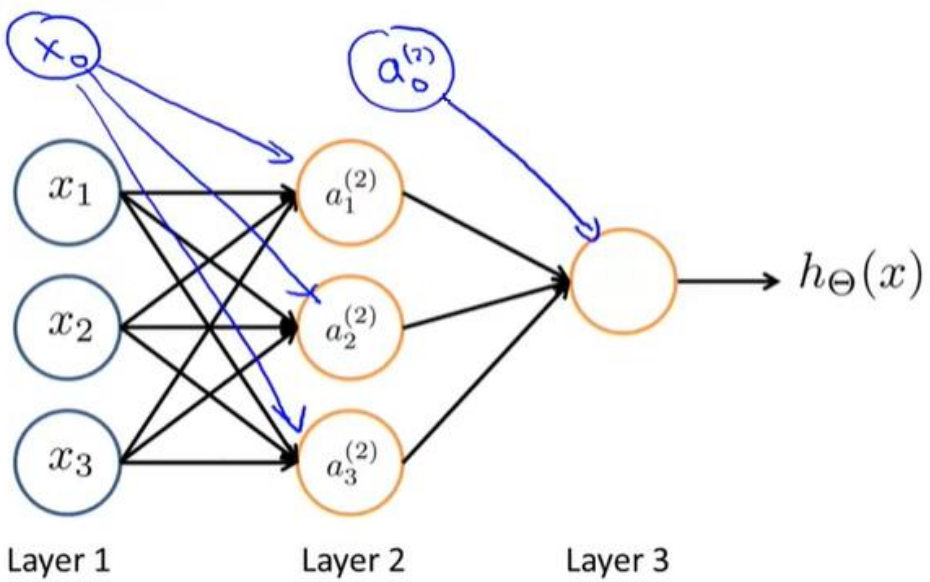
This function is called **"Activation Function"** which is the same as the logistic function.

Here, $x = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix}$ $\theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \\ \theta_3 \end{bmatrix}$

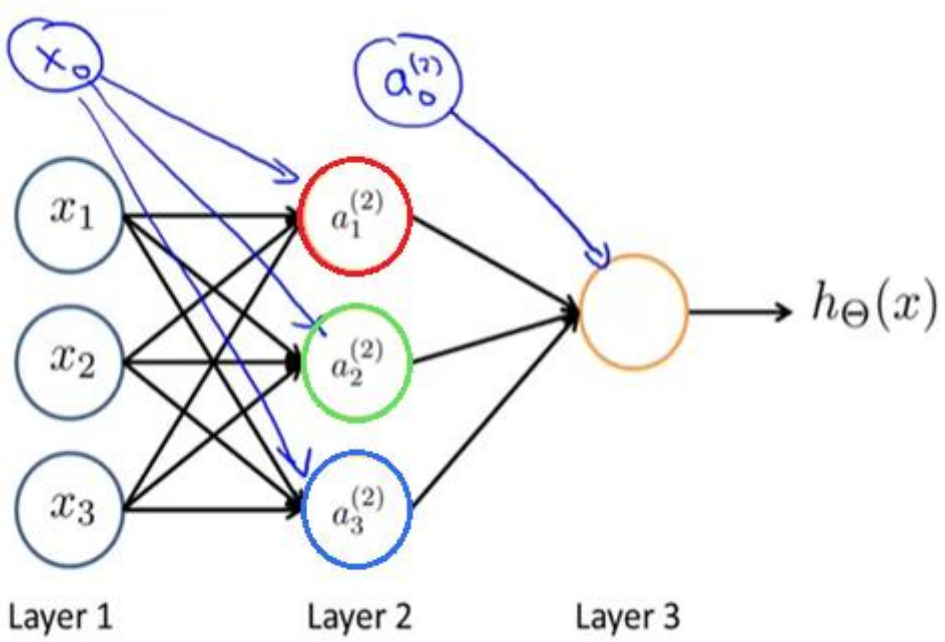
Note: θ (parameter vector) is also called **"weight vector"** sometimes and θ_0, θ_1, θ_2 and θ_3 are the weights of this vector.

Neural Network

Group of different neurons together is called a Neural Network.



- **Layer 1 :** input layer that consists of input units x_0, x_1, x_2, x_3
- **Layer 2:** hidden layer that consists of a_0, a_1, a_2, a_3 (artificial neurons) responsible for computations
- **Layer 3:** output layer that gives the output



$a_i^{(j)}$ = "activation" of unit *i* in layer *j*

$\Theta^{(j)}$ = matrix of weights controlling function mapping from layer *j* to layer *j+1*

$$a_1^{(2)} = g(\Theta_{10}^{(1)} x_0 + \Theta_{11}^{(1)} x_1 + \Theta_{12}^{(1)} x_2 + \Theta_{13}^{(1)} x_3)$$

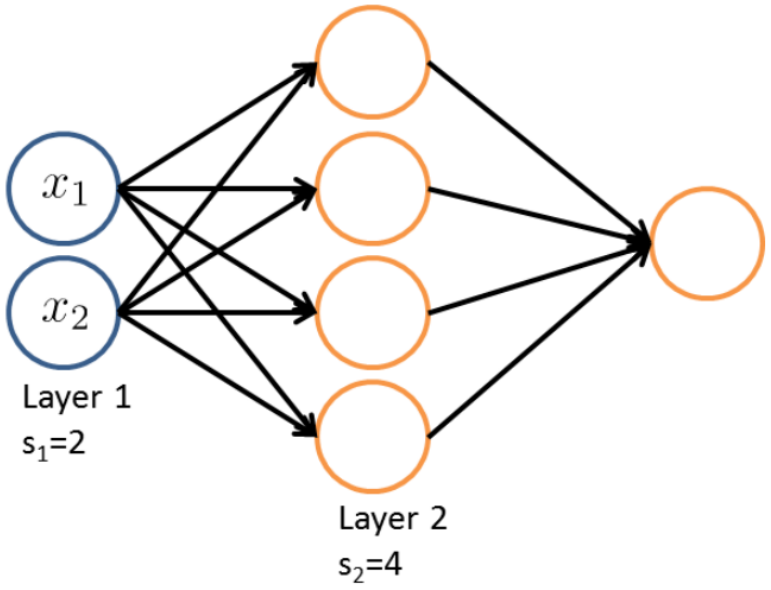
$$a_2^{(2)} = g(\Theta_{20}^{(1)} x_0 + \Theta_{21}^{(1)} x_1 + \Theta_{22}^{(1)} x_2 + \Theta_{23}^{(1)} x_3)$$

$$a_3^{(2)} = g(\Theta_{30}^{(1)} x_0 + \Theta_{31}^{(1)} x_1 + \Theta_{32}^{(1)} x_2 + \Theta_{33}^{(1)} x_3)$$

$$h_\theta(x) = a_1^{(3)} = g(\Theta_{10}^{(2)} a_0^{(2)} + \Theta_{11}^{(2)} a_1^{(2)} + \Theta_{12}^{(2)} a_2^{(2)} + \Theta_{13}^{(2)} a_3^{(2)})$$

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

Consider the following neural network:



What is the dimension of $\Theta^{(1)}$? (Hint: add a bias unit to the input and hidden layers)

- 2 x 4
- 4 x 2
- 3 x 4
- 4 x 3

Neural Networks: Model Representation-II

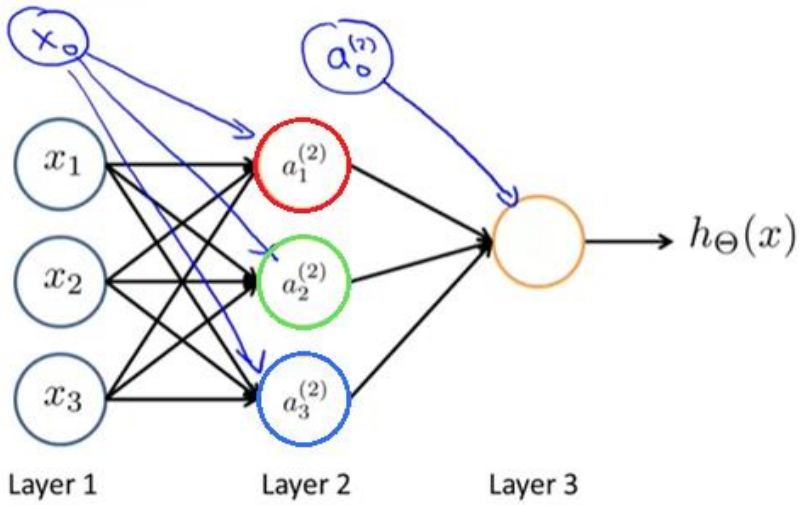
Consider an example of a neural network:

$a_1^{(2)} = g(\Theta_{10}^{(1)} x_0 + \Theta_{11}^{(1)} x_1 + \Theta_{12}^{(1)} x_2 + \Theta_{13}^{(1)} x_3)$

$a_2^{(2)} = g(\Theta_{20}^{(1)} x_0 + \Theta_{21}^{(1)} x_1 + \Theta_{22}^{(1)} x_2 + \Theta_{23}^{(1)} x_3)$

$a_3^{(2)} = g(\Theta_{30}^{(1)} x_0 + \Theta_{31}^{(1)} x_1 + \Theta_{32}^{(1)} x_2 + \Theta_{33}^{(1)} x_3)$

$h_\theta(x) = a_1^{(3)} = g(\Theta_{10}^{(2)} a_0^{(2)} + \Theta_{11}^{(2)} a_1^{(2)} + \Theta_{12}^{(2)} a_2^{(2)} + \Theta_{13}^{(2)} a_3^{(2)})$



A vectorized implementation of the above functions is represented by:

$a_1^{(2)} = g(z_1^{(2)})$

$a_2^{(2)} = g(z_2^{(2)})$

$a_3^{(2)} = g(z_3^{(2)})$

In other words, for layer j=2 and node k, the variable z will be:

$z_k^{(2)} = \Theta_{k,0}^{(1)} x_0 + \Theta_{k,1}^{(1)} x_1 + \ldots \Theta_{k,n}^{(1)} x_n$

The vector representation of x and z^j is:

$$x = \begin{bmatrix} x_0 \\ x_1 \\ \cdots \\ x_n \end{bmatrix} \quad z^{(j)} = \begin{bmatrix} z_1^{(j)} \\ z_2^{(j)} \\ \cdots \\ z_n^{(j)} \end{bmatrix}$$

Let $x = a^{(1)}$ i.e. **layer 1 artificial neurons** , we can rewrite the equation as: $z^{(j)} = \Theta^{(j-1)} a^{(j-1)}$

Example:

$z^{(2)} = \Theta^{(1)} a^{(1)}$

$a^{(2)} = g(z^{(2)})$ where $a^{(2)} = \begin{bmatrix} a_{(1)}^{(2)} \\ a_{(2)}^{(2)} \\ a_{(3)}^{(2)} \end{bmatrix}$ is a 3D vector.

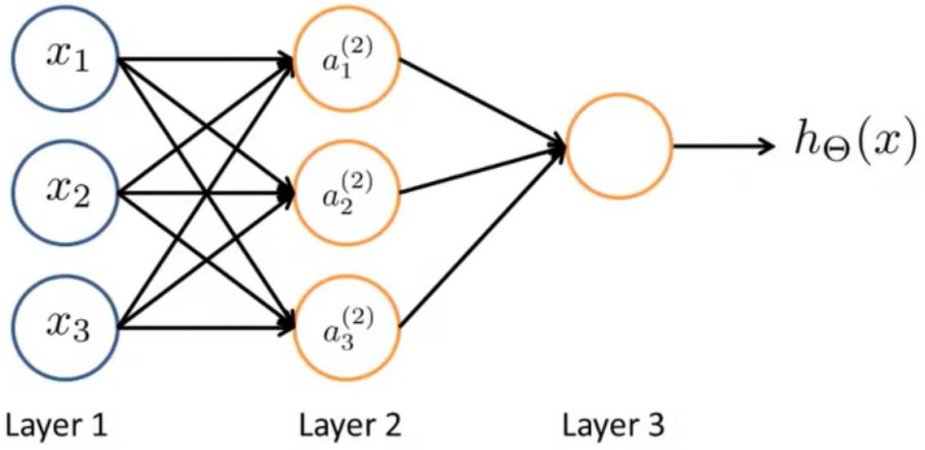
Now, add $a_0^{(2)} = 1$ (bias unit) $\Rightarrow a^{(2)} = \begin{bmatrix} a_{(0)}^{(2)} \\ a_{(1)}^{(2)} \\ a_{(2)}^{(2)} \\ a_{(3)}^{(2)} \end{bmatrix}$ is now a 4D vector.

Thus, $z^{(3)} = \Theta^{(2)} a^{(2)}$

$h_\Theta(x) = a^{(3)} = g(z^{(3)})$

This process of computing $h_\Theta(x)$ is also called **"Forward Propagation"**

Neural Network learning its own features:



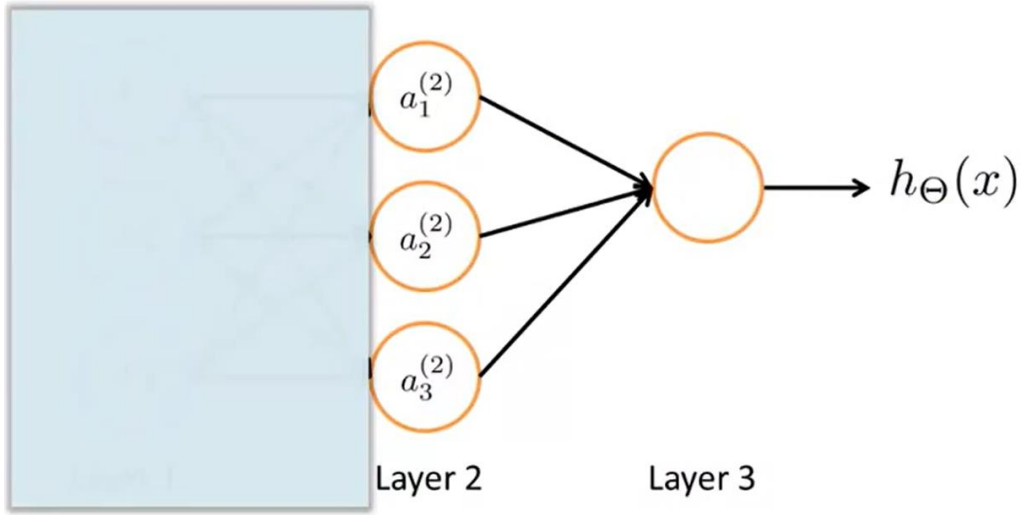
In the above neural network, we see that there are three layers.

Layer 1: input layer

Layer 2: hidden layer

Layer 3: output layer

If we cover up the Layer 1 of this neural network, we get



This looks a lot like logistic regression to make prediction of $h_\Theta(x)$ which is given by,

$h_\Theta(x) = g(\Theta_{10}^{(2)} a_0^{(2)} + \Theta_{11}^{(2)} a_1^{(2)} + \Theta_{12}^{(2)} a_2^{(2)} + \Theta_{13}^{(2)} a_3^{(2)})$

Hence, the output $h_\Theta(x)$ is calculated by using the new features a_1, a_2, a_3 from Layer 2 instead of using the original features x_1, x_2 and x_3 of Layer 1.

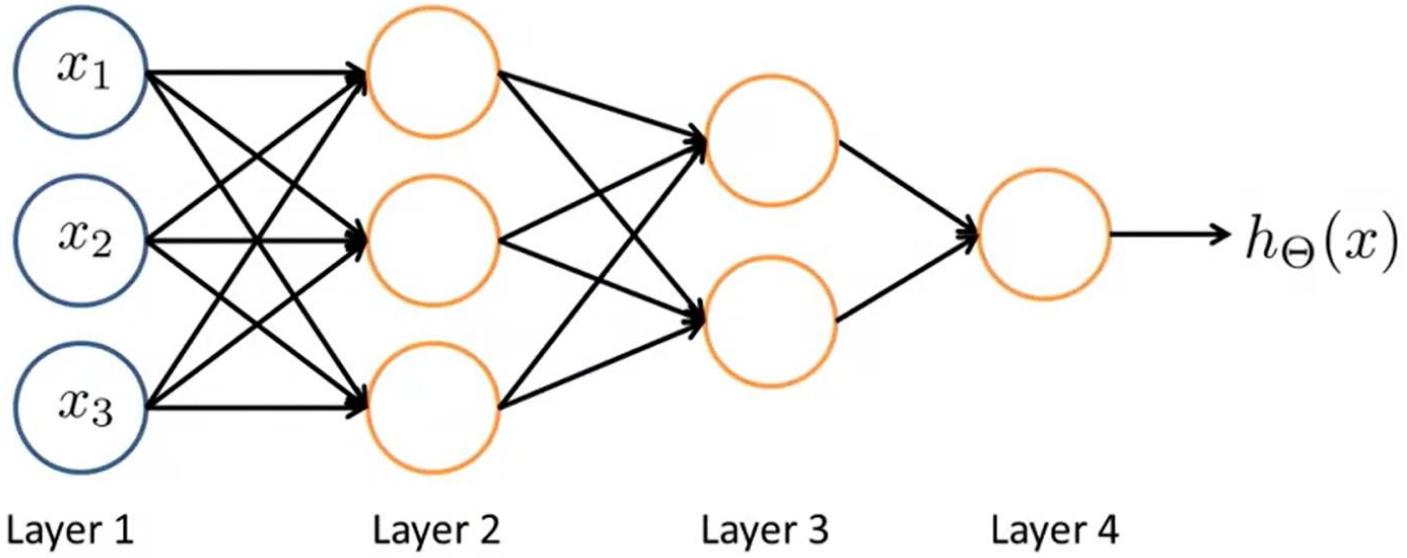
The features a_1, a_2, a_3 are themselves learned from the original features of Layer 1!

This procedure is called "Forward Propagation" in which one layer learns the features from its previous layer.

In simple words**, Layer 3 learns from Layer 2**, and **Layer 2 learns from Layer 1.**
This gives a much better hypothesis than using the original raw features.

Other neural network architectures:

There can be any number of layers in a neural network.



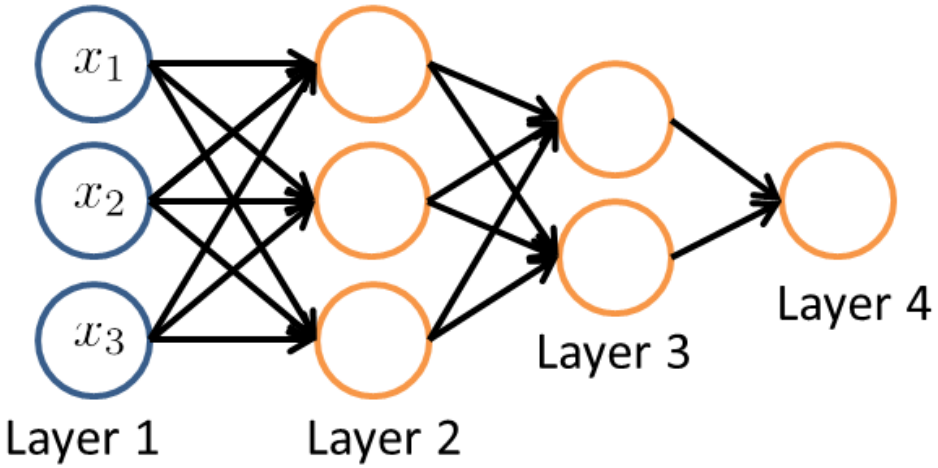
In the above example, there are 4 layers.

Layer 1: input layer

Layer 2, Layer 3: hidden layers

Layer 4: output layer

Consider the network:



Let $a^{(1)} = x \in R^{n+1}$ denote the input (with $a_0^{(1)} = 1$).

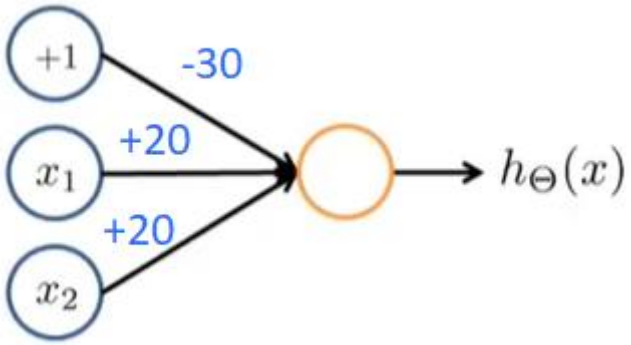
How would you compute $a^{(2)}$?

- $a^{(2)} = \Theta^{(1)} a^{(1)}$
- $z^{(2)} = \Theta^{(2)} a^{(1)}$; $a^{(2)} = g(z^{(2)})$
- $z^{(2)} = \Theta^{(1)} a^{(1)}$; $a^{(2)} = g(z^{(2)})$
- $z^{(2)} = \Theta^{(2)} g(a^{(1)})$; $a^{(2)} = g(z^{(2)})$

Neural Networks: Examples and Intuitions-I

Example: AND

Predict $y = x_1$ AND x_2 where $x_1, x_2 \in \{0, 1\}$ using Neural Networks.



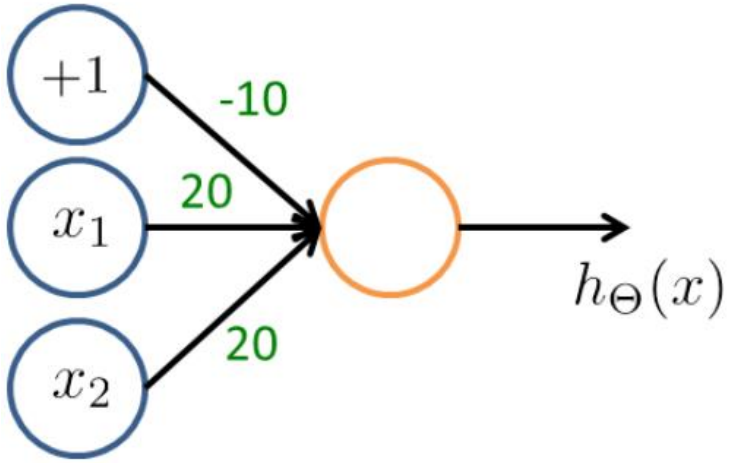
We assign some parameters (weights) to x_0, x_1 and x_2 as shown.

Thus, the hypothesis is given by, $h_\Theta(x) = g(-30 + 20x_1 + 20x_2)$

x_1	x_2	$h_\Theta(x)$
0	0	g(-30) ≈ 0
0	1	g(-10) ≈ 0
1	0	g(-10) ≈ 0
1	1	g(10) ≈ 1

The output of the truth table resembles logical AND

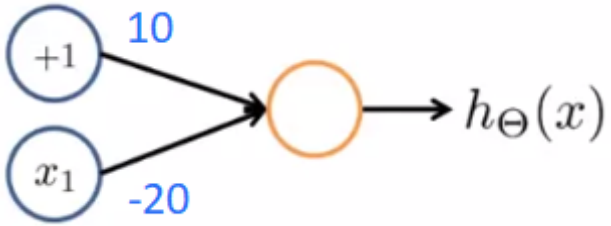
Suppose x1 and x2 are binary valued (0 or 1). What Boolean function does the network shown below approximately compute?



- x_1 AND x_2
- (NOT x_1) OR (NOT x_2)
- x_1 OR x_2
- (NOT x_1) AND (NOT x_2)

Negation (NOT):

Consider the following neural network.



Hypothesis, $h_\Theta(x) = g(10 - 20x_1)$

x_1	$h_\Theta(x)$
0	$g(10) \approx 1$
1	$g(-10) \approx 0$

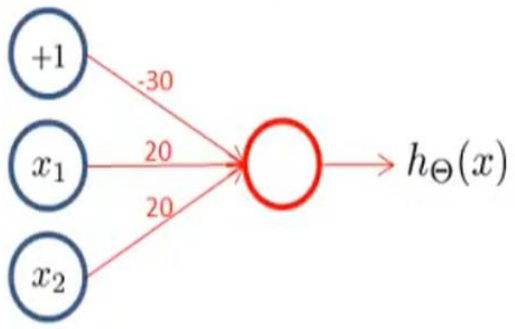
The output of the truth table resembles logical NOT

It can be deduced that,

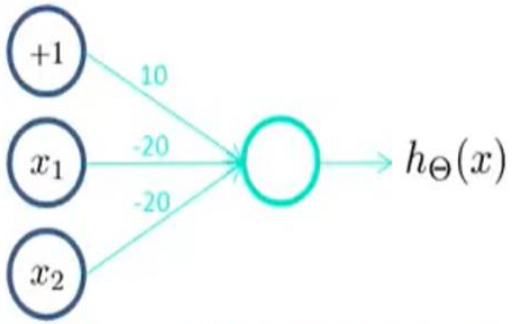
x_1 NOR x_2 = (NOT x_1) AND (NOT x_2)

The $\Theta^{(1)}$ matrices for AND, NOR and OR are:

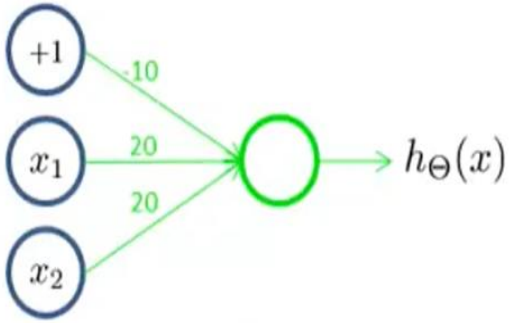
- **AND:** $\Theta^{(1)}$ = [-30 20 20]
- **NOR:** $\Theta^{(1)}$ = [10 -20 -20]
- **OR:** $\Theta^{(1)}$ = [-10 20 20]



x_1 AND x_2

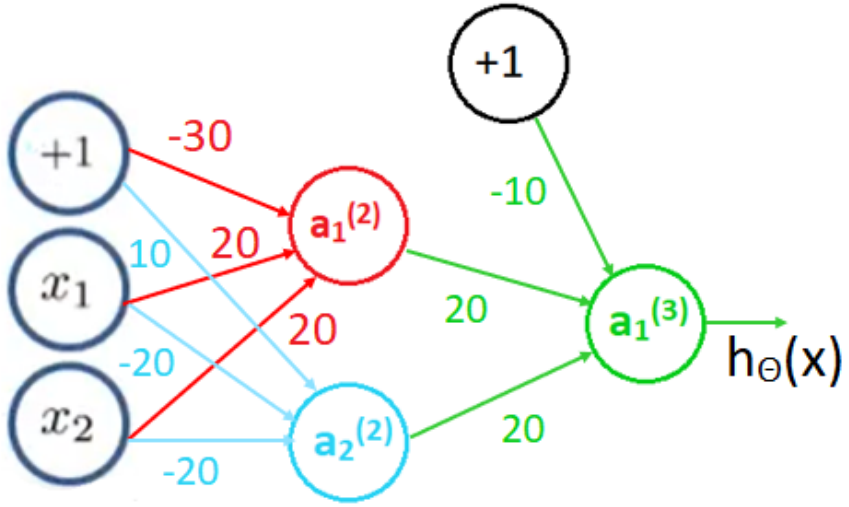


(NOT x_1) AND (NOT x_2)



x_1 OR x_2

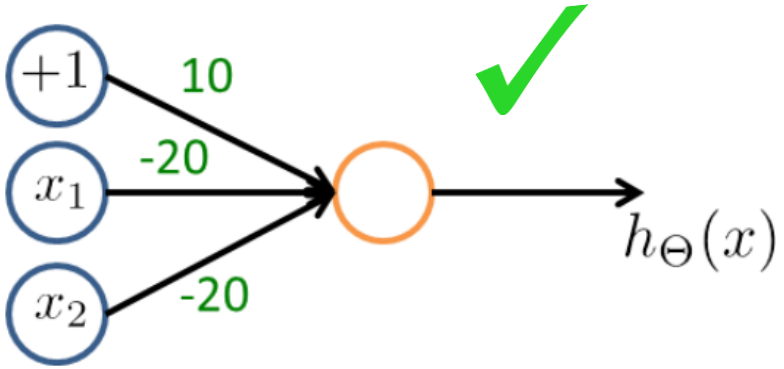
We can combine these to get the XNOR logical operator (which gives output 1 if x_1 and x_2 are both 0 or both 1).

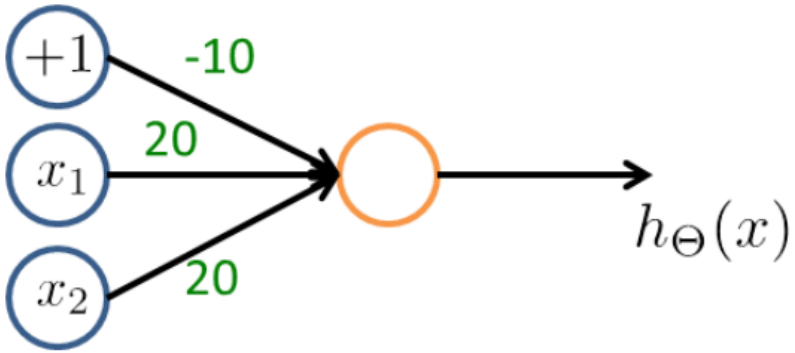


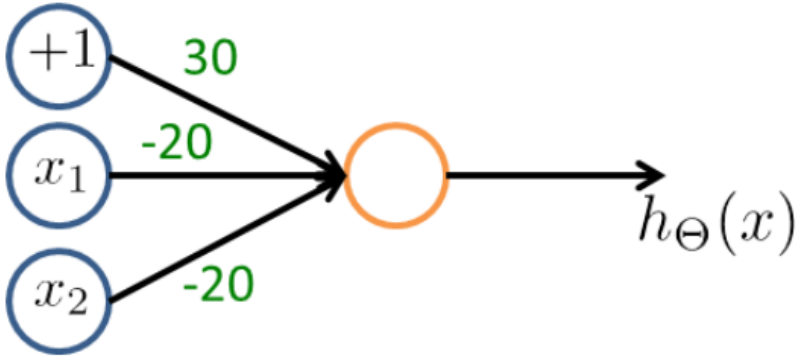
x_1	x_2	$a_1^{(2)}$	$a_2^{(2)}$	$h_\Theta(x)$
0	0	0	1	1
0	1	0	0	0
1	0	0	0	0
1	1	1	0	1

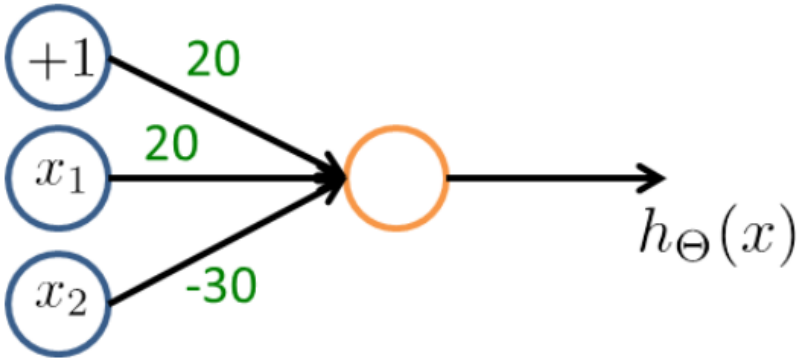
By adding **Layer 2** and **Layer 3,** we are able to compute these complex functions.

Suppose that x_1 and x_2 are binary valued (0 or 1). Which of the following networks (approximately) computes the Boolean function (NOT x_1) AND (NOT x_2)?



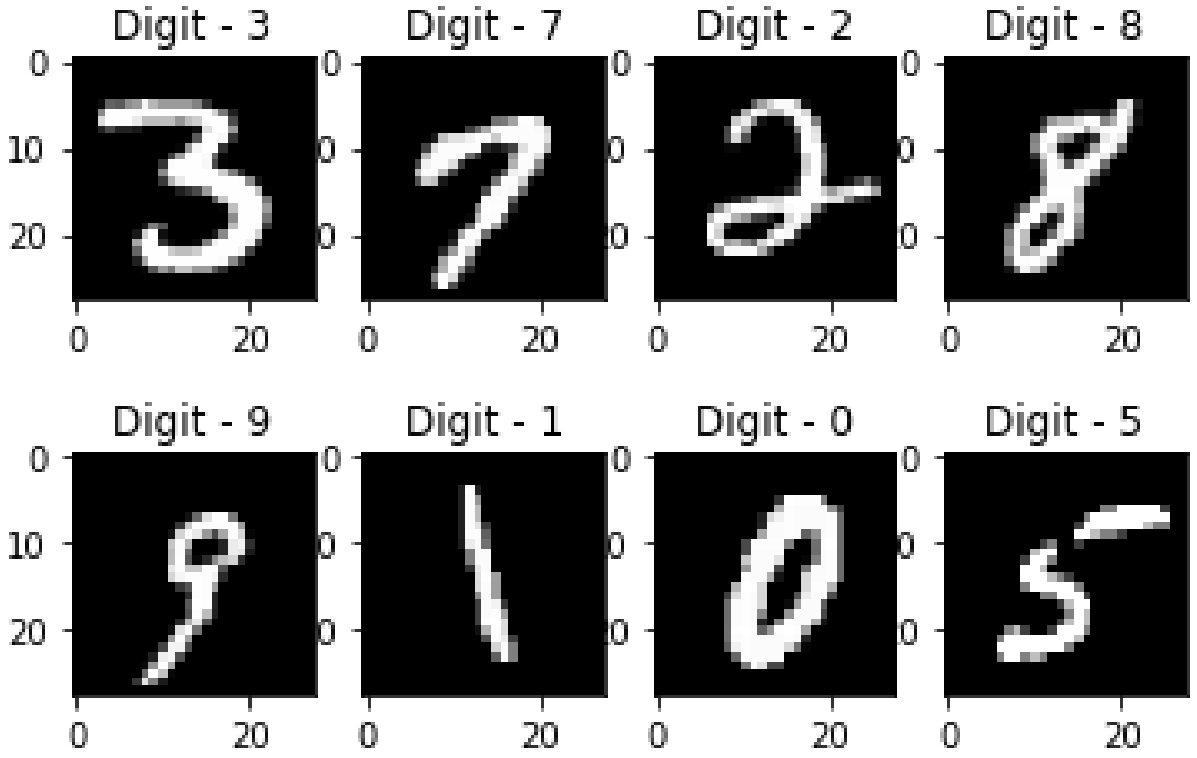






Handwritten Digit Classification:

Another classical example where Neural Networks are used for classifying or recognizing handwritten digits.



[Reference: https://www.youtube.com/watch?v=yxuRnBEczUU]

Neural Networks: Multi-class classification

- Till now, we saw neural networks that had a single output.
- To classify data into multiple classes, we let our hypothesis function return a vector of values.

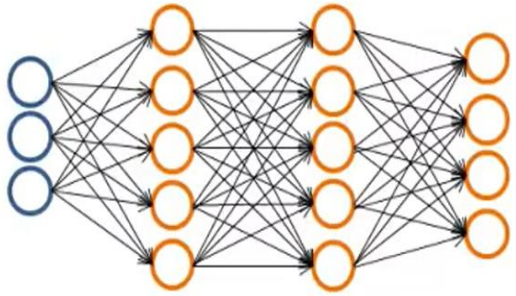
Multiple output units: One-vs-all.


Pedestrian


Car


Motorcycle


Truck



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

$h_\Theta(x) \approx \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}$ pedestrian

$h_\Theta(x) \approx \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}$ car

$h_\Theta(x) \approx \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}$ motorcycle

$h_\Theta(x) \approx \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$ truck

We can define our set of resulting classes as y:

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

- Each $y^{(i)}$ represents a different image corresponding to either a car, pedestrian, truck, or motorcycle.
- The inner layers, each provide us with some new information which leads to our final hypothesis function.
- The setup looks like:

$$\begin{bmatrix}x_0\\x_1\\x_2\\\cdots\\x_n\end{bmatrix} \rightarrow \begin{bmatrix}a_0^{(2)}\\a_1^{(2)}\\a_2^{(2)}\\\cdots\end{bmatrix} \rightarrow \begin{bmatrix}a_0^{(3)}\\a_1^{(3)}\\a_2^{(3)}\\\cdots\end{bmatrix} \rightarrow \cdots \rightarrow \begin{bmatrix}h_\Theta(x)_1\\h_\Theta(x)_2\\h_\Theta(x)_3\\h_\Theta(x)_4\end{bmatrix}$$

If $h_\Theta(x) = \begin{bmatrix}0\\0\\1\\0\end{bmatrix}$, the output represents a **“motorcycle”**

Suppose you have a multi-class classification problem with 10 classes. Your neural network has 3 layers, and the hidden layer (layer 2) has 5 units. Using the one-vs-all method, how many elements does $\Theta^{(2)}$ have?

- 50
- 55
- 60
- 66