TC3020 | Machine Learning (ML)
08 Artificial Neural Networks

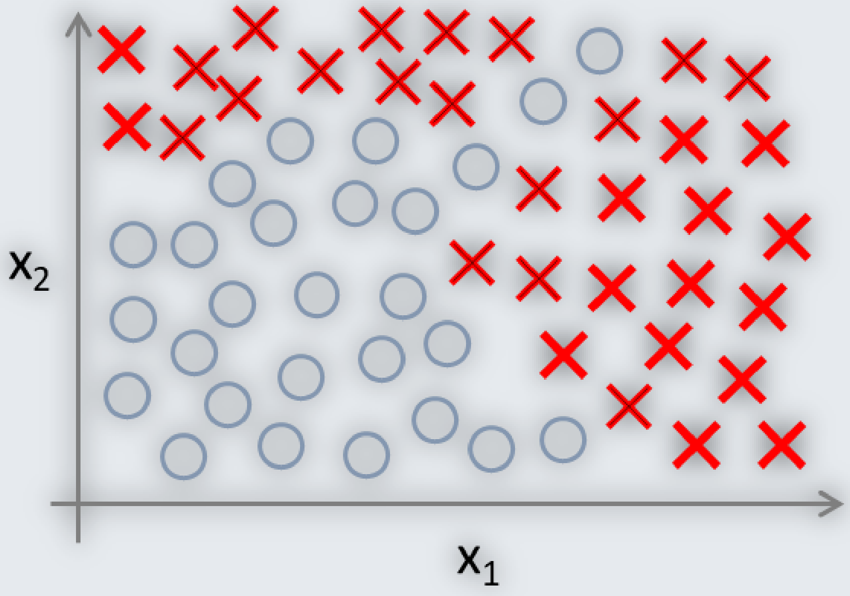
Profr. Rafael Pérez Torres

r.perez@tec.mx

EiC | ITESM | Campus MTY

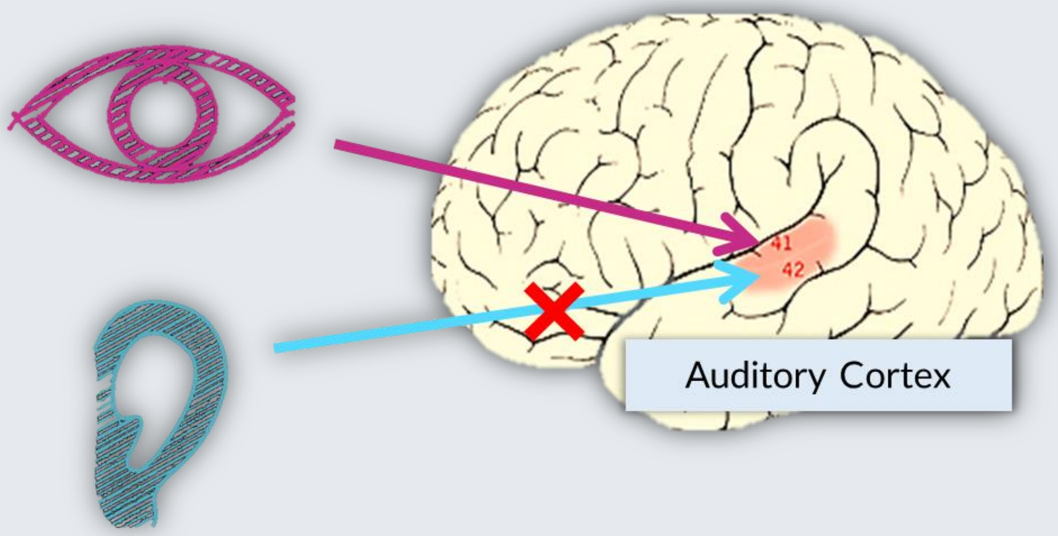
Non-linear hypothesis

- When dealing with non-linear decision boundaries it is possible to perform feature engineering to achieve separation of classes.
  - However, this is a risky operation, we might end up with way too many features.
  - Doing that operation by hand is cumbersome.
- **There should be a way to better select our features.**



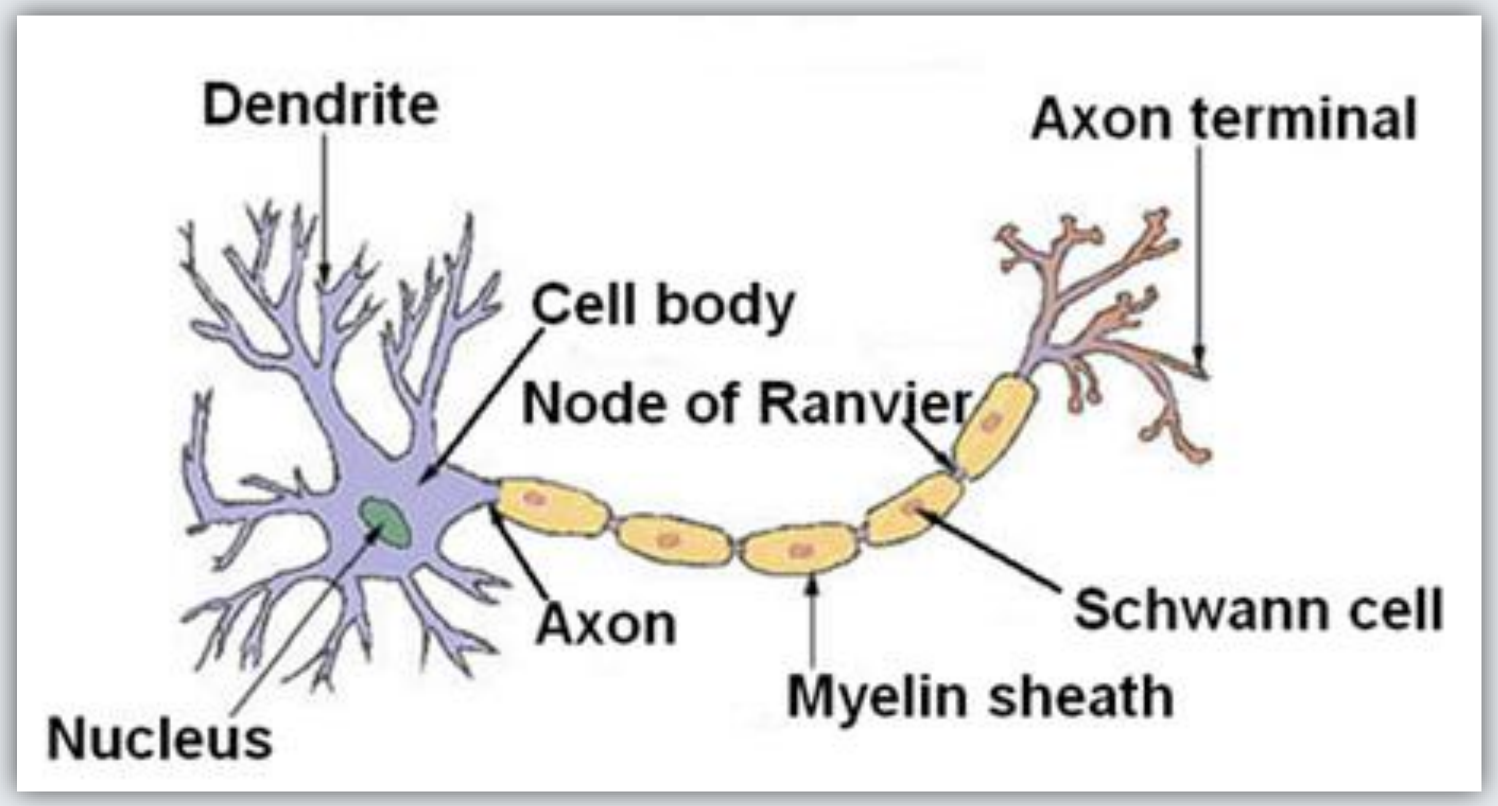
Artificial Neural Networks (ANN)

- Origins: algorithms that try to mimic the human brain
- It is said that mammal brains follow **a single recipe to learn/process different types of stimuli**
- If that is the case, it could be possible to design an algorithm to learn or process any type of information



[Roe et al., 1992]

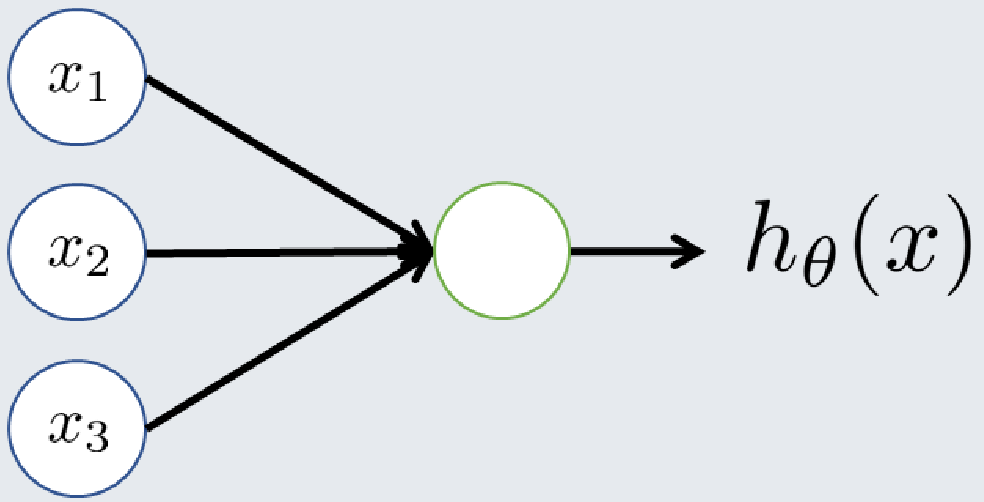
Model representation



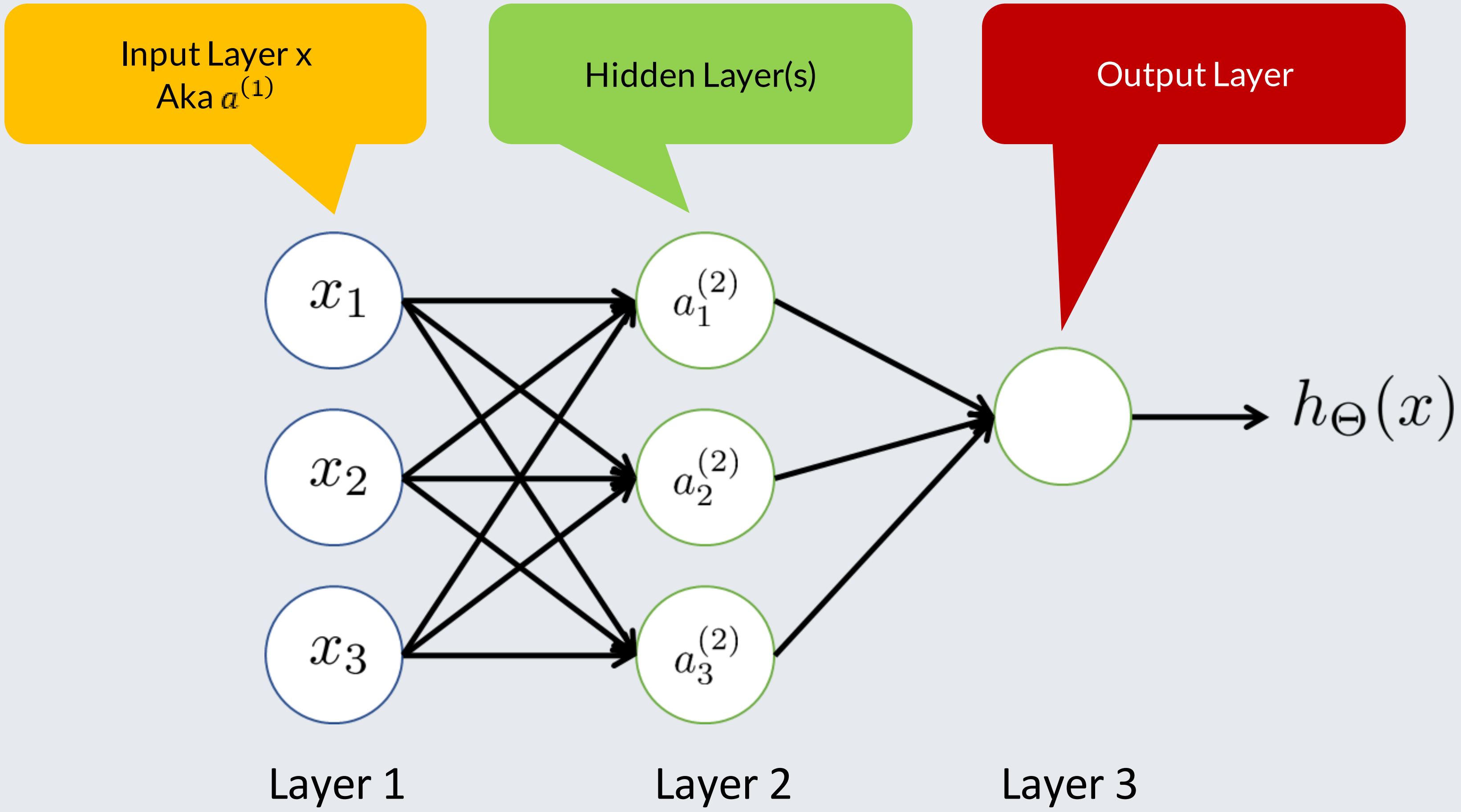
Model representation

- The perceptron
- Parameters are called **weights**
- We still add a **bias unit**
- We need an activation function: sigmoid $\frac{1}{1+e^{-\theta^T x}}$
  - 0 output: inhibited neuron
  - 1 output: excited neuron

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



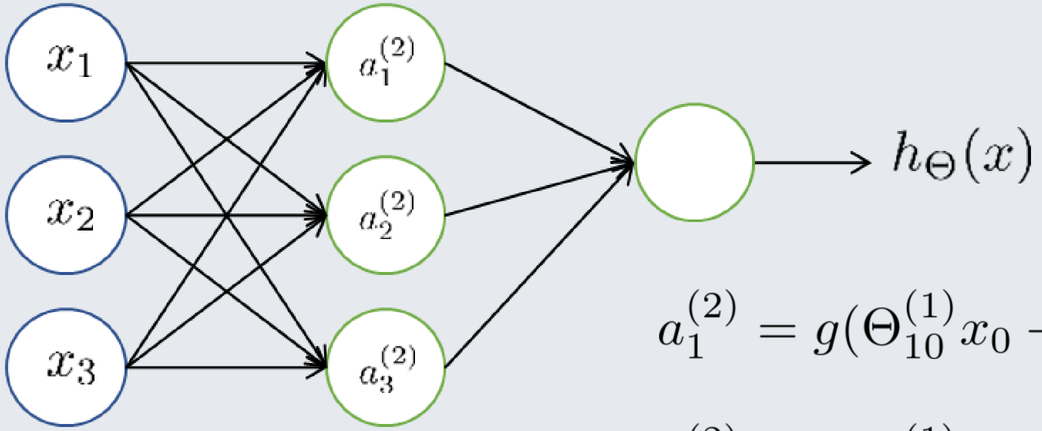
ANN architecture



ANN's activations

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

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



Θ_{23} : arrives to neuron 2, coming from neuron 3

$\Theta^{(1)} \in \mathbb{R}^{3\times 4}$

It arrives to 3 neurons coming out from 4 neurons

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

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

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

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

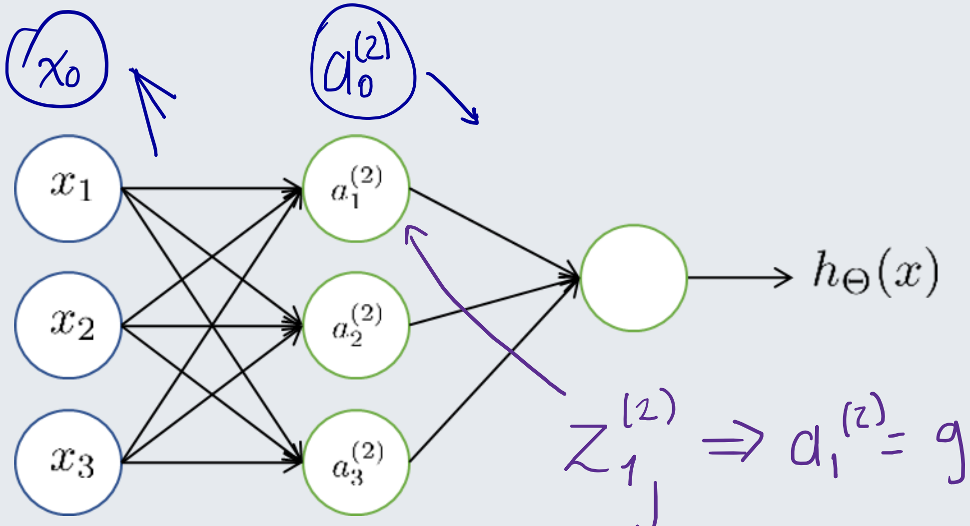
If network has s_j units in layer j, s_{j+1} units in layer $j+1$, then $\Theta^{(j)}$ will be of dimension $s_{j+1} \times (s_j + 1)$

E.g., for $\Theta^{(2)}$. Layer 2 has 3 units, layer 3 has 1 unit. $\Theta^{(2)}$ is of size 1 x 4

ANN's operation

- Working with a neural network involves two steps:
  - **Feed forward**: to predict $h_\theta(x)$
  - **Backpropagation**: to adjust the weights in Θ:
    - This is the actual core of the training for an ANN.
    - When a good value of Θ has been found, the model could be exported easily (just the matrix of weights Θ plus the calculation of $h_\theta(x)$ is enough to predict the class for new examples).
- As we are dealing with matrixes, using vector operations can speed up and clarify the feed forward and backpropagation processes.

ANN vectorized implementation



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

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

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

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

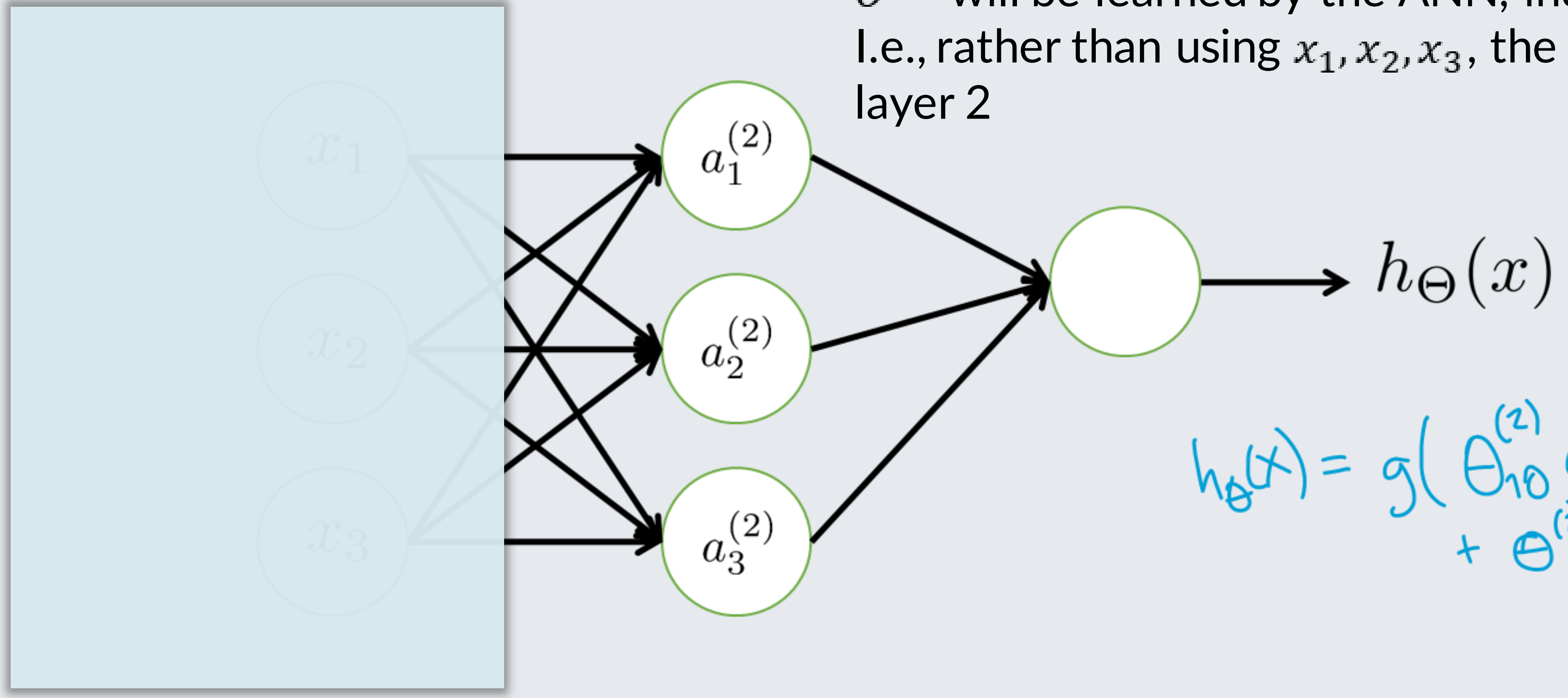
$$z^{(2)} = \Theta^{(1)} x$$
$$a^{(2)} = g(z^{(2)})$$

Add $a_0^{(2)} = 1$

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

ANN learning its own features

$\Theta^{(2)}$ will be learned by the ANN, independently of input x.
I.e., rather than using x_1, x_2, x_3, the ANN is using the activations in layer 2



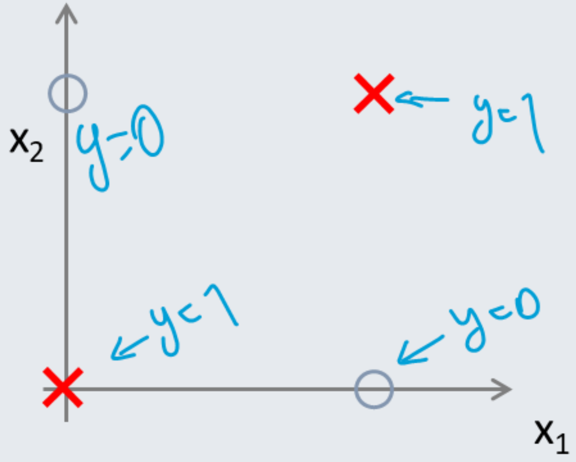
$$h_\Theta(x) = g\left(\Theta_{10}^{(2)} a_0^{(2)} + \Theta_{11}^{(2)} a_1^{(2)} + \Theta_{12}^{(2)} a_2^{(2)} + \Theta_{13}^{(2)} a_3^{(2)}\right)$$

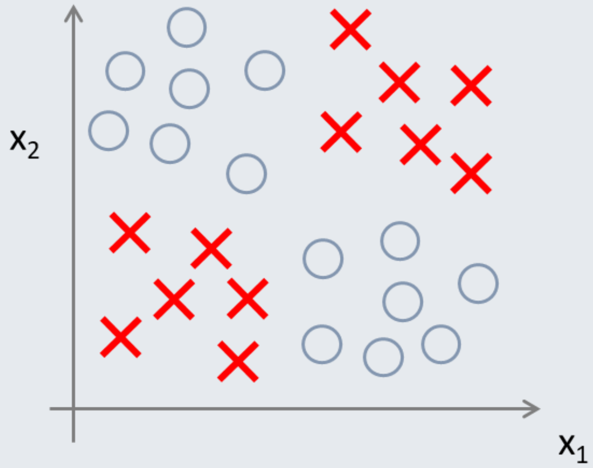
- **This is a lot like Logistic Regression!**
- Take each perceptron (neuron) individually, we find Logistic Regressions.
- With lots of them, it is clear how ANNs produce non-linear boundaries.
- Somehow the features become something more complex during the process.

Non-linear classification example

- "XNOR" function: negating the XOR

x1	x2	Xor	Xnor
0	0	0	1
0	1	1	0
1	0	1	0
1	1	0	1

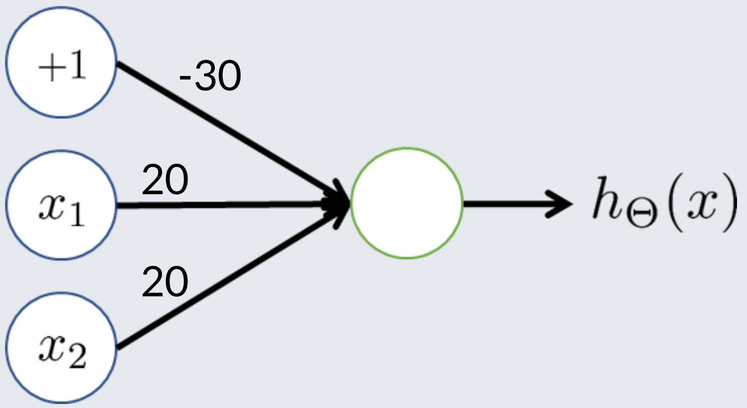




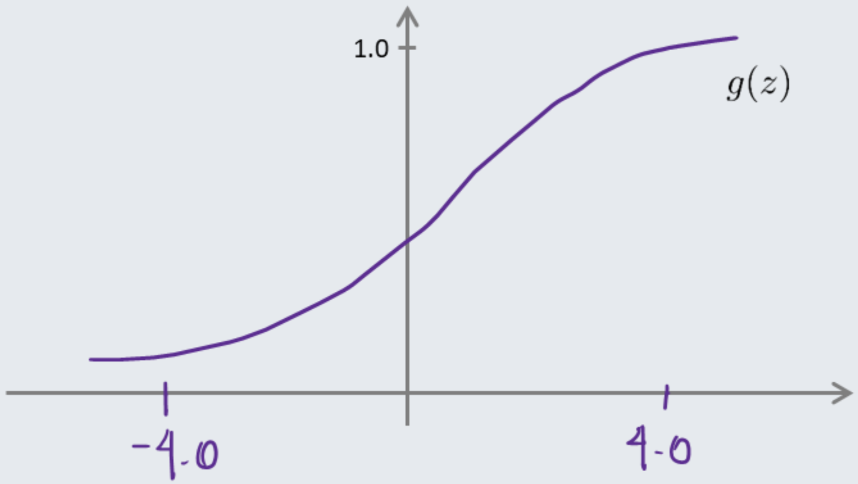
Example: Simple AND gate

$x_1, x_2 \in \{0, 1\}$

$y = x_1 \text{ AND } x_2$

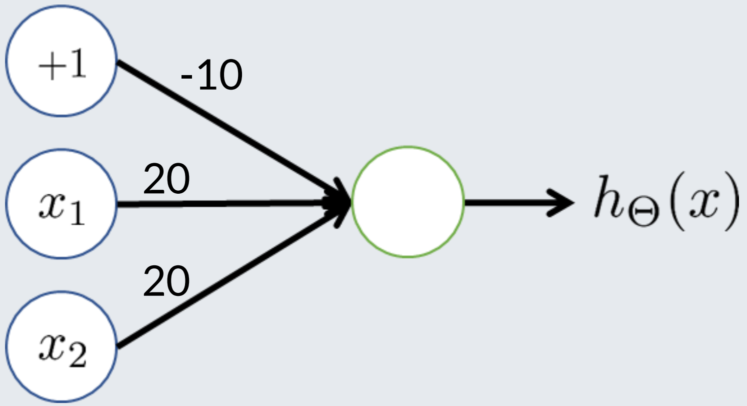


$$h_\theta(x) = g(-30 + 20x_1 + 20x_2)$$



x_1	x_2	$h_\Theta(x)$
0	0	
0	1	
1	0	
1	1	

Example: Simple OR gate



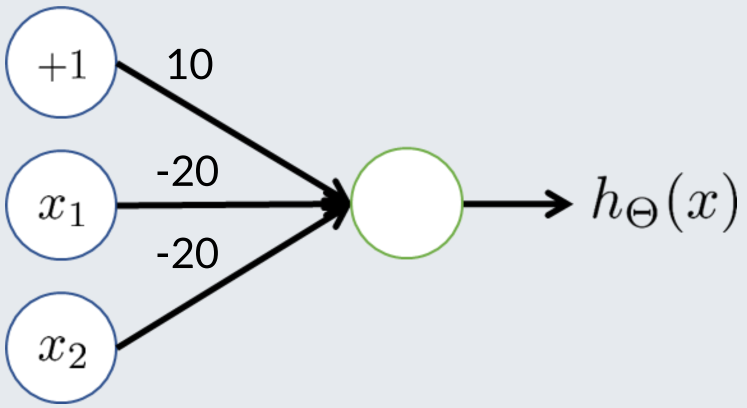
x_1	x_2	$h_\Theta(x)$
0	0	
0	1	
1	0	
1	1	

$$h_\theta(x) = g(-10 + 20x_1 + 20x_2)$$

Example: Simple NOR gate

$$(\text{NOT } x_1) \text{ AND } (\text{NOT } x_2)$$

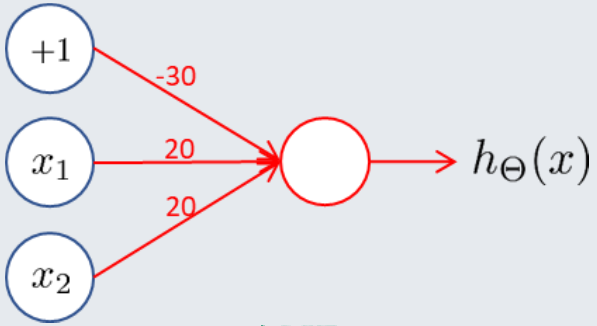
1 iff both are = 0



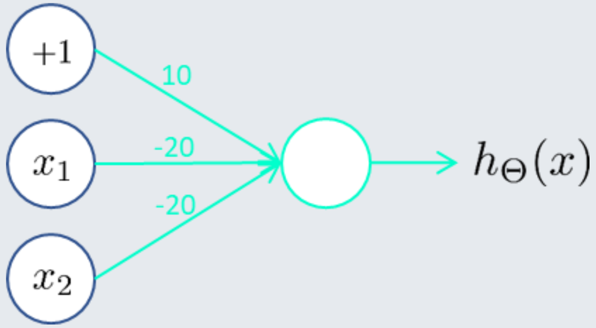
x_1	x_2	$h_\Theta(x)$
0	0	
0	1	
1	0	
1	1	

$$h_\theta(x) = g(10 - 20x_1 - 20x_2)$$

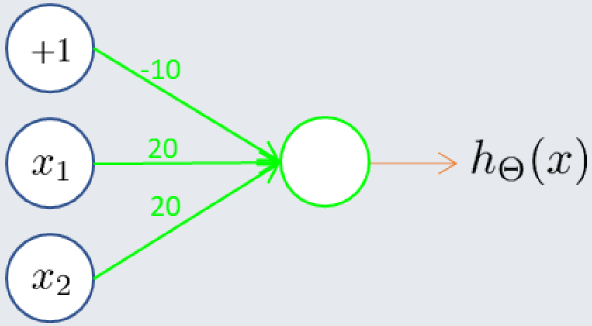
Example: Putting it all together for XNOR



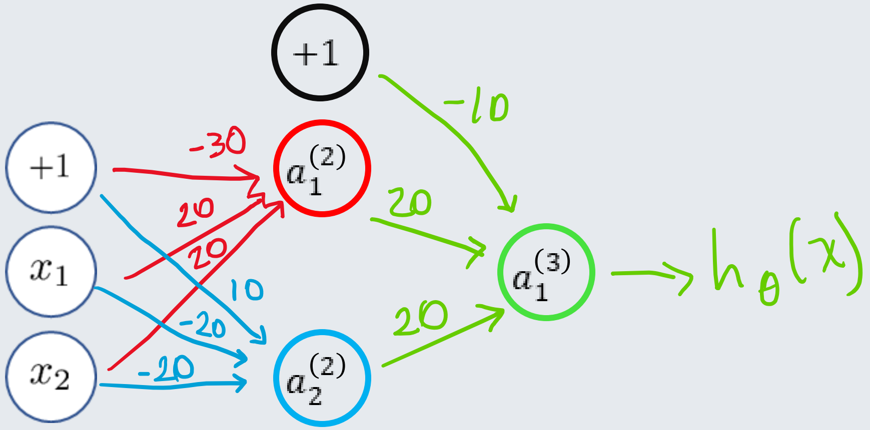
x_1 AND x_2



(NOT x_1) AND (NOT x_2)



x_1 OR x_2



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

x1	x2	Xor	Xnor
0	0	0	1
0	1	1	0
1	0	1	0
1	1	0	1

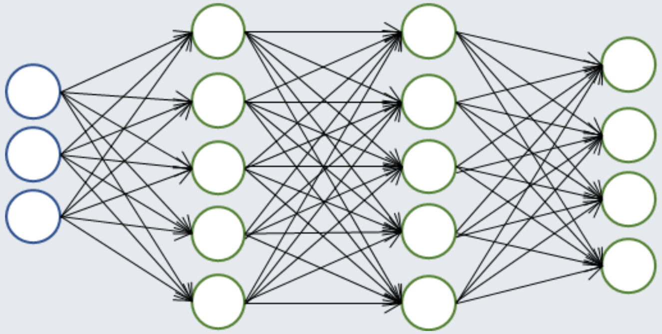
Multiclass classification

Pedestrian

Car

Motorcycle

Truck



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

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

when pedestrian

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

when car

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

when motorcycle

etc

Summary

- In this session we introduced Artificial Neural Networks.
  - Its inspiration in nature.
  - Its architecture.
  - The basic process to predict a class.
  - How it can create non-linear separations.
  - How it could be linked to Logistic Regression.
  - How it helps to follow a one-vs-all approach.

* Portions of this material are based on the ML course available at `https://www.coursera.org/learn/machine-learning`