NANYANG TECHNOLOGICAL UNIVERSITY SINGAPORE

CZ3005
Artificial Intelligence

Neural Networks

Asst/P Mahardhika Pratama

Email: mpratama@ntu.edu.sg

Office: N4-02a-08

Outline

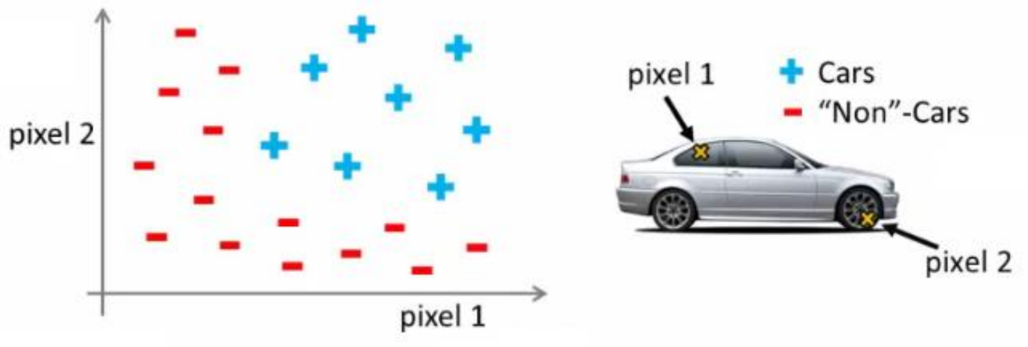
- Overview and Summary
- Model Representation
- Neural Network Example
- Multi-class Classification Problem

Overview and Summary

- Suppose we have a complex problem
  - can use high order logistic regression but only scalable for small input dimension
- Suppose we have a problem with 100 input variables
  - If we use second order polynomial, we will end up with 5000 terms
  - We can use a subset of them, but if you do not have enough

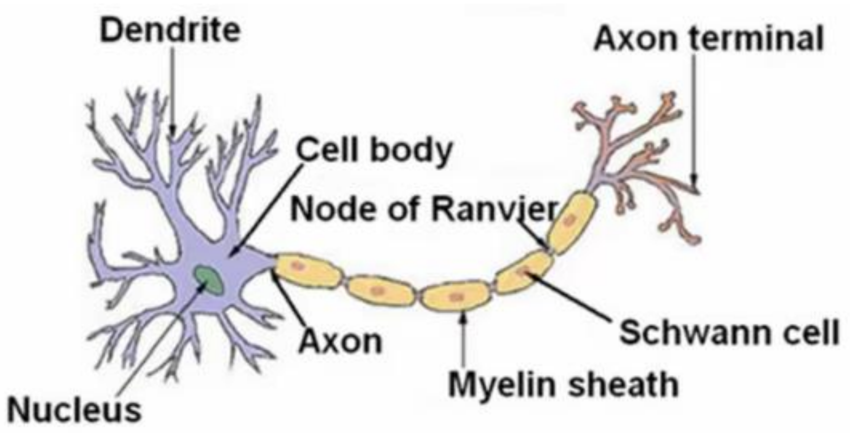
Computer Vision

- A matrix of pixel intensity values
- Build a car detector
  - ✓ Cars and Non Cars
  - ✓ Plot two pixels
  - ✓ Plot car and non car on the figure
- Need a nonlinear hypothesis
- Feature Space
  - ✓ If we use 50 by 50 pixels images – 2500 features
  - ✓ If we use RGB – 7500 features
- We need NN for a complex problem



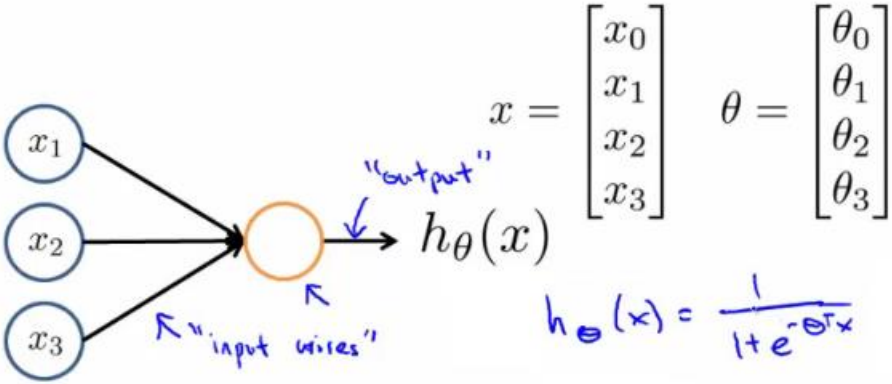
Model Representation

- NNs were developed as a way to simulate networks of neurons
- Three things to notice from the right
  - ✓ Cell Body
  - ✓ Number of Input Wires (Dendrites)
  - ✓ Output Wires (axon)
- Simple Level
  - ✓ Neurons gets one or more inputs through dendrites
  - ✓ Does processing
  - ✓ Sends output down to axon
- Neurons communicate through electric spikes
  - ✓ Pulse of electricity via axon to another neuron



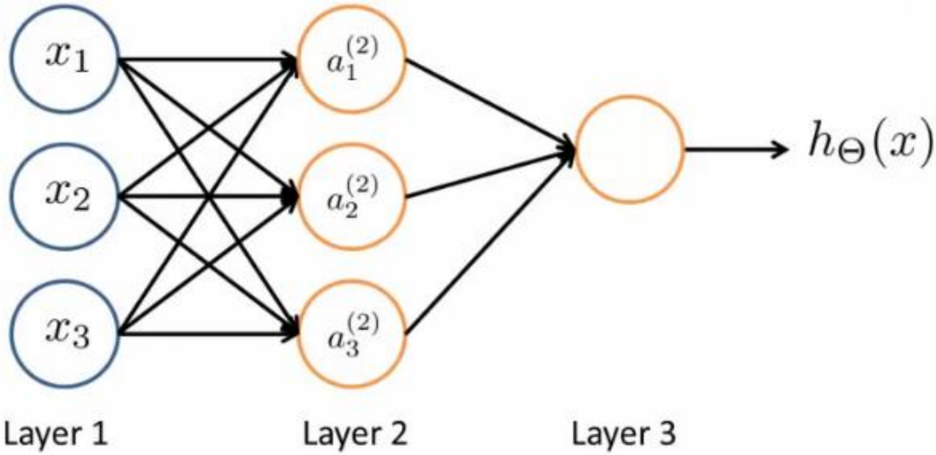
Model Representation

- In an artificial neural network, a neuron is a logistic unit
    - Feed input via input wires
    - Logistic unit does computation
    - Sends output down output wires
- That logistic computation is just like our previous logistic regression hypothesis calculation
- Often good to include an x_0 input - the **bias unit**
- This is an artificial neuron with a sigmoid (logistic) activation function
    - ϴ vector may also be called the **weights** of a model



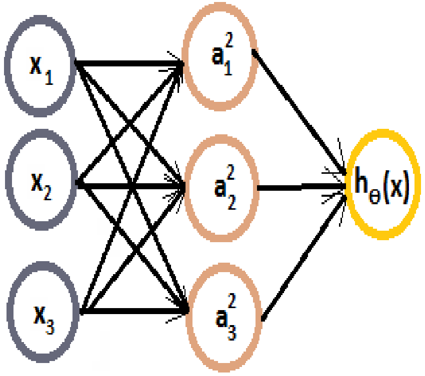
Model Representation

- Here, input is x_1, x_2 and x_3
  - Input activation on the first layer - i.e. (a_1^1, a_2^1 and a_3^1)
  - Three neurons in layer 2 (a_1^2, a_2^2 and a_3^2)
  - Final neuron produces the output
    - Which again we *could* call a_1^3
- First layer is the **input layer**
- Final layer is the **output layer** - produces value computed by a hypothesis
- Middle layer(s) are called the **hidden layers**
  - You don't observe the values processed in the hidden layer
  - Not a great name
  - Can have many hidden layers



Model Representation

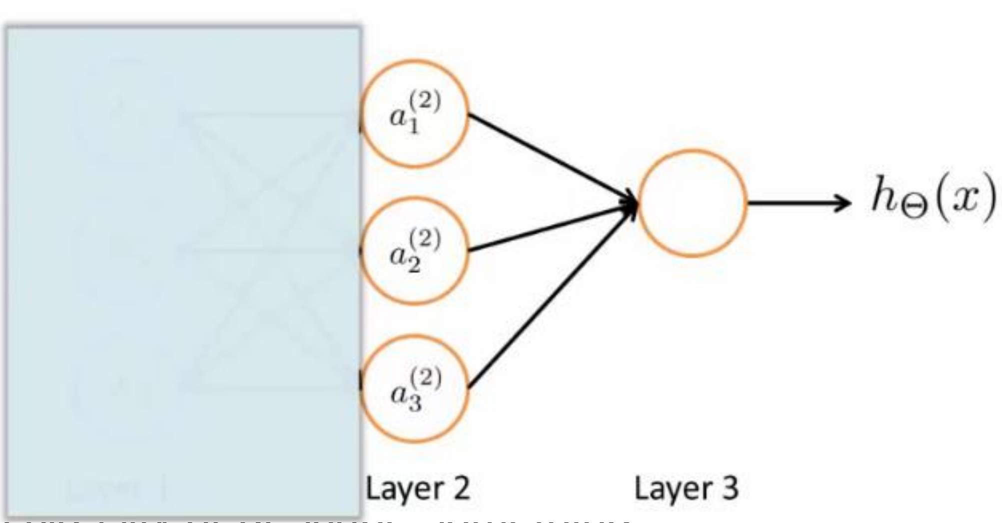
- $a_i^{(j)}$ - activation of unit *i* in layer *j*
  - So, a_1^2 - is the **activation** of the 1st unit in the second layer
  - By activation, we mean the value which is computed and output by that node
- $\Theta^{(j)}$ - matrix of parameters controlling the function mapping from layer *j* to layer *j* + 1
- Looking at the Θ matrix
  - Column length is the number of units in the following layer
  - Row length is the number of units in the current layer + 1 (because we have to map the bias unit)



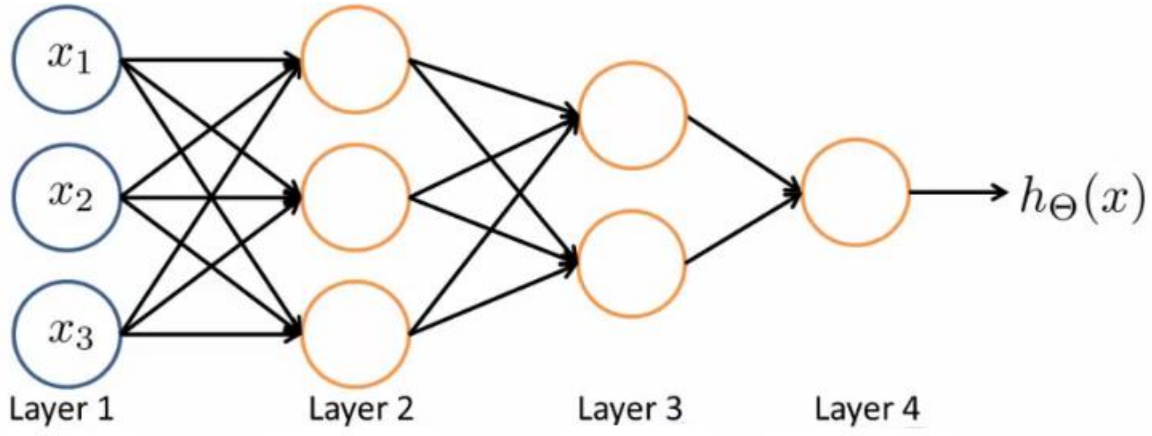
$$a_1^{(2)} = g(\Theta_{10}^{(1)}x_0 + \Theta_{11}^{(1)}x_1 + \Theta_{12}^{(1)}x_2 + \Theta_{13}^{(1)}x_3)$$
$$a_2^{(2)} = g(\Theta_{20}^{(1)}x_0 + \Theta_{21}^{(1)}x_1 + \Theta_{22}^{(1)}x_2 + \Theta_{23}^{(1)}x_3)$$
$$a_3^{(2)} = g(\Theta_{30}^{(1)}x_0 + \Theta_{31}^{(1)}x_1 + \Theta_{32}^{(1)}x_2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

Model Representation

- Layer 3 is a logistic regression node
  - The hypothesis output = g
- This is just logistic regression
  - the features are just value
- The features a_1^2, a_2^2, and a_3^2
  - a neural network can lear
- Flexibility to learn whatever fe ssion calculation
  - So, if we compare this to calculate your own excitin
    - to define the best way t
  - Here, we're letting the hidden layers do that, so we feed the hidden layers our input values,
    - and let them learn whatever gives the best final result to feed into the final output layer



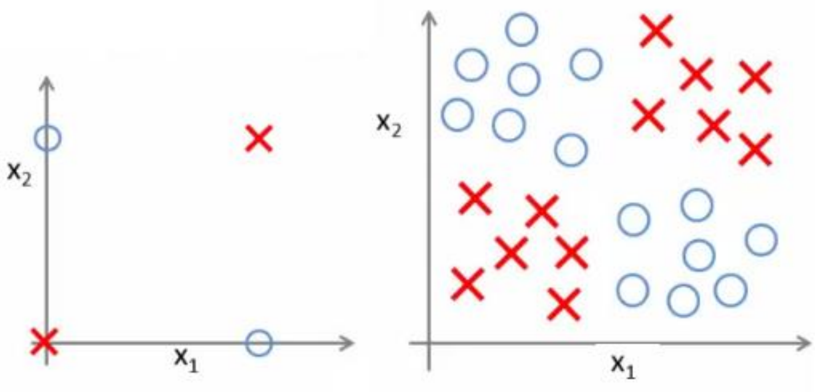
Model Representation



- As well as the networks already seen, other architectures (topology) are possible
- More/less nodes per layer
- More layers
- Once again, layer 2 has three hidden units, layer 3 has 2 hidden units by the time you get to the output layer you get very interesting non-linear hypothesis

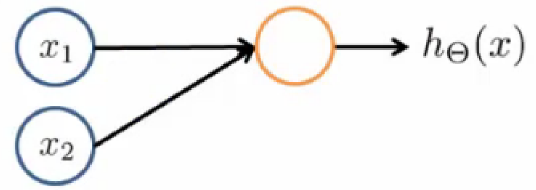
Neural Network Example

- Problem : XOR/XNOR
  - x_1, x_2 are binary
- $y = x_1$ XOR x_2
- x_1 XNOR x_2
- Where XNOR = NOT (x_1 XOR x_2)

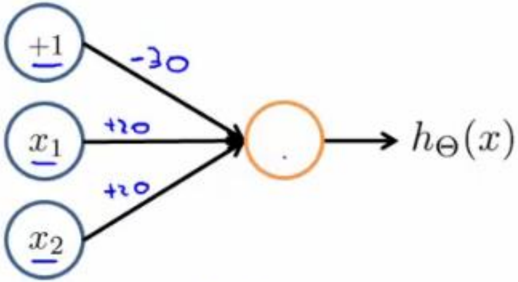


We want to learn a non-linear decision boundary to separate the positive and negative examples

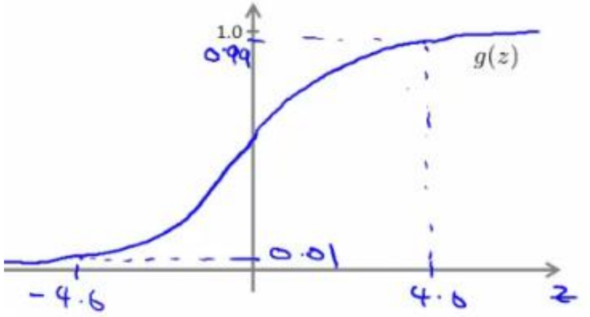
Neural Network Example: AND



Manually Assign the Weight



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

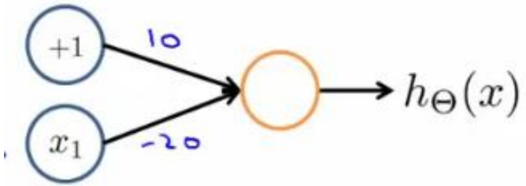


Sigmoid function (reminder)

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

$h_\Theta(x) \approx x_1$ AND x_2

Neural Network Example: NOT

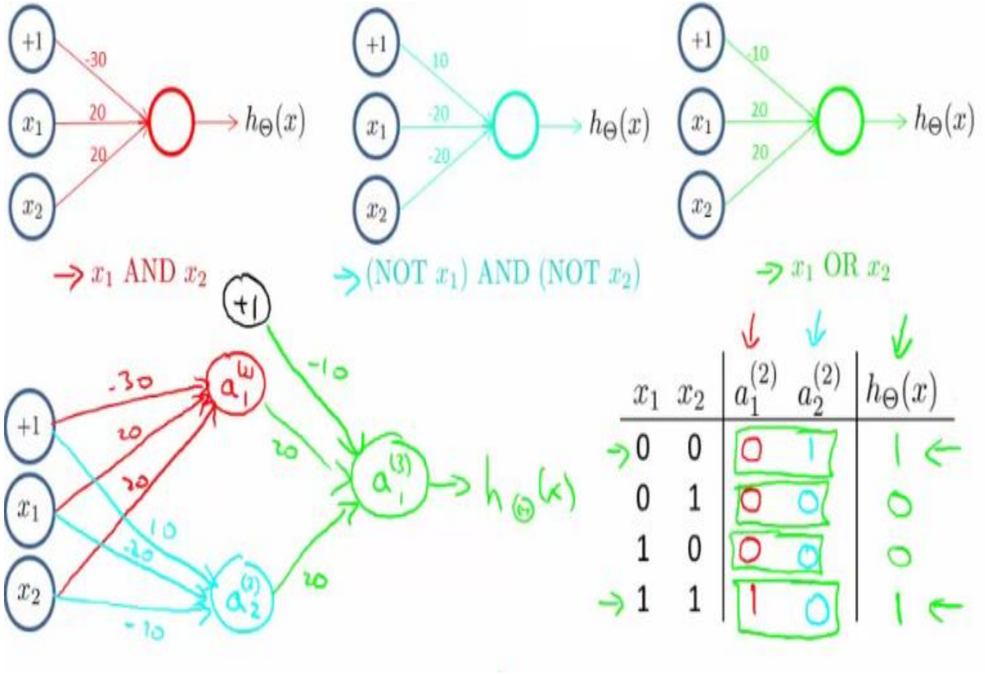


x_1	$h_\Theta(x)$
0	$g(10) \approx 1$
1	$g(-10) \approx 0$

Negation is achieved by putting a large negative weight in front of the variable you want to negate

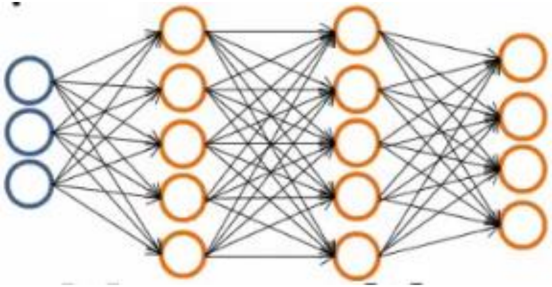
Neural Network Example: XNOR

- How can we implement XNOR function with NN?
  - XNOR = Not XOR
  - Positive output is produced if and only if
    - AND (both true)
    - Neither
- We can combine these into NN and make it work



Multiclass Problem

- One vs All
- Classify pedestrian, car, motorbike and truck
  - Build NN with 4 output units
    - 1 is 0/1 pedestrian
    - 2 is 0/1 car
    - 3 is 0/1 motorcycle
    - 4 is 0/1 truck
  - When an image of pedestrian [1,0,0,0]
- Training set is image of our four classification problems



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

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