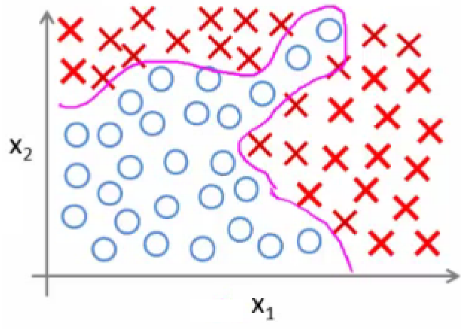
Week 8

July 16, 2021

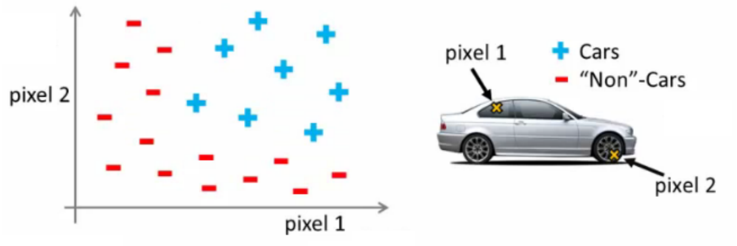
Why do we need neural networks?

- We can have complex data sets with many (1000's) important features.
- Using logistic regression becomes expensive really fast.
- The only way to get around this is to use a subset of features. This, however, may result in less accuracy.



Problems where n is large - computer vision

- Computer vision sees a matrix of pixel intensity values.
- To build a car detector: Build a training set of cars and not cars. Then test against a car.
- Plot two pixels (two pixel locations) and car or not car on the graph.



Feature space

Problems where n is large - computer vision

- If we used 50 x 50 pixels $--> 2500$ pixels, so n = 2500.
- If RGB then 7500.
- If 100 x 100 RB then $-->$ 50 000 000 features.
- Way too big.

Neurons and the brain

- The desire to construct computers that mimicked brain activities drove the creation of neural networks (NNs).
- Used a lot in the 80s. Popularity diminished in 90s.
- Large-scale neural networks have just recently become computationally feasible due to the computational cost of NNs.

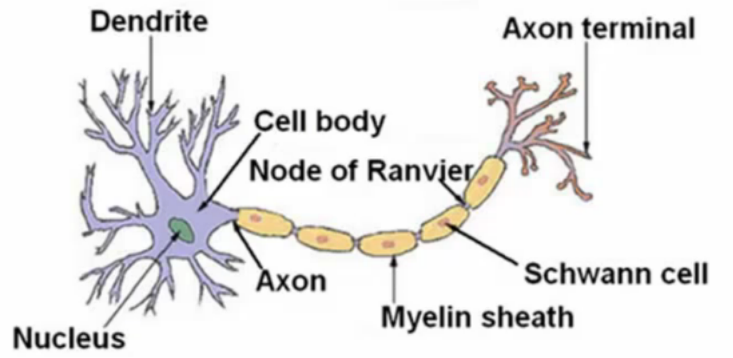
Brain

- Hypothesis is that the brain has a single learning algorithm.
- If the optic nerve is rerouted to the auditory cortex, the auditory cortex is learns to see.
- If you rewrite the optic nerve to the somatosensory cortex then it learns to see.

Model representation I

Neuron:

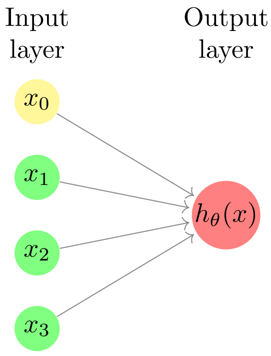
- Cell body
- Number of input wires (dendrites)
- Output wire (axon)



- Neurone gets one or more inputs through dendrites.
- Does processing.
- The output is sent along the axon.
- Neurons communicate through electric spikes.

Artificial neural network - representation of a neurone

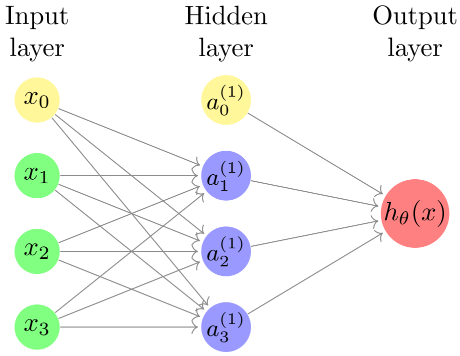
- Feed input via input wires.
- Logistic unit does computation.
- Sends output down output wires.



$$x = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix}$$

$$\theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \\ \theta_3 \end{bmatrix}$$

- The diagram above represents a single neurone.
- x_0 is called the bias unit.
- θ vector is called the weights of a model.



- 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.
- Every input/activation goes to every node in following layer.

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

Model representation II

In this section, we'll look at how to do the computation efficiently using a vectorized approach. We'll also look at why NNs are useful and how we can use them to learn complicated nonlinear things.

Let's define a few more terms:

$$z_1^{(2)} = \Theta_{10}^{(1)}x_0 + \Theta_{11}^{(1)}x_1 + \Theta_{12}^{(1)}x_2 + \Theta_{13}^{(1)}x_3$$

We can now write:

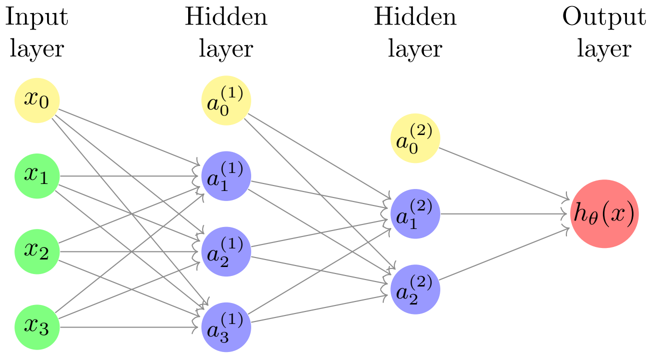
$$a_1^{(2)} = g(z_1^{(2)})$$

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

- $z^{(2)}$ is a $3x1$ vector.
- $a^{(2)}$ ia also a $3x1$ vector.
- Middle layer(s) are called the hidden layers.
- $g()$ applies the sigmoid (logistic) function element wise to each member of the $z^{(2)}$ vector.
- Obviously the "activation" for the input layer is just the input!
- $a^{(1)}$ is the vector of inputs.
- $a^{(2)}$ is the vector of values calculated by the $g(z^{(2)})$ function.
- We send our input values to the hidden layers and let them learn which values produce the best final result to feed into the final output layer.
- This process is also called forward propagation.

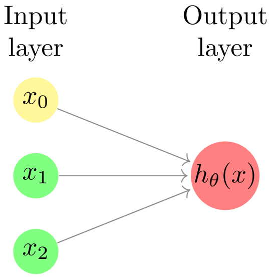
Other architectural designs are also possible:

- More/less nodes per layer.
- More layers.



Layer 2 has three hidden units (plus bias), layer 3 has two hidden units (plus bias), and by the time you get to the output layer, you have a really intriguing non-linear hypothesis.

AND function

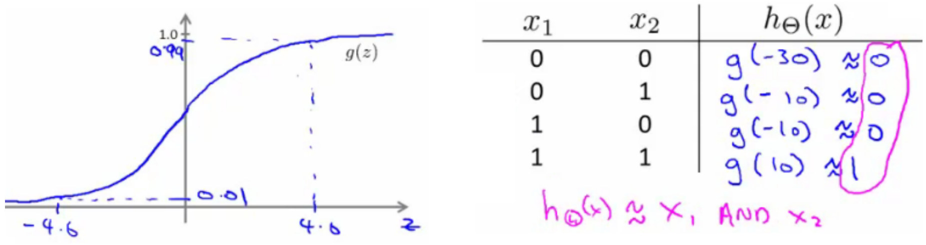


Let $x_0 = 1$ and theta vector be:

$$\Theta_1^{(1)} = \begin{bmatrix} -30 \\ 20 \\ 20 \end{bmatrix}$$

Then hypothesis is:

$$h_\Theta(x) = g(-30 \cdot 1 + 20 \cdot x_1 + 20 \cdot x_2)$$

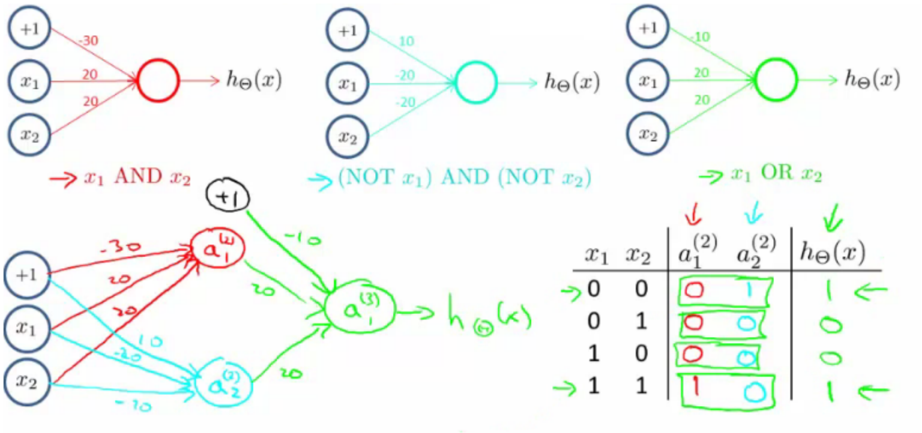


x_1	x_2	$h_\Theta(x)$
0	0	$g(-30) \approx 0$
0	1	$g(-10) \approx 0$
1	0	$g(-10) \approx 0$
1	1	$g(10) \approx 1$

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

Sigmoid function (reminder)

XNOR function



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