WEEK # 4

1. MOTIVATIONS.

1.1. Non-linear hypothesis:

Non-linear Classificatn:

$$g(\theta_0+\theta_1x_1+\theta_2x_2+\theta_3x_1x_2+\theta_4x_1^2x_2+\theta_5x_1^3x_2+\theta_6x_1x_2^2+\dots)$$

this method works well when we have only 2 features (x_1, x_2)

But for many ML problems would have a lot more features!

if we have $n=100$ features, you end up with ≈ 5000 features (quadratic features!)

$\approx \frac{n^2}{2}$. Because of many features, it may overfit the training set.

→ if you add qubic polynomials, there would be $\approx 170\,000$ features. $(O(n^3))$

ex/ if we have 50×50 pixel img, there are $n=2500$ (7500 if RGB) features.

& quadratic features $\approx$ 3 million features.

if you try to classify imgs only by looking the pxl values, probably you will overfit! cannot classify imgs.

1.2. Neurons and the Brain:

NNs: Algorithms that try to mimic the brain.

The "one learning algorithm" hypothesis. Audotory cortex learns to see. } neural-rewiring
Somatosensory cortex learns to see. } experiments.

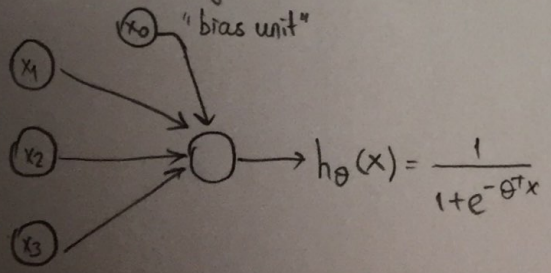
2. NEURAL NETWORKS

2.1. Model Representation -1:

Dendrite — "input wires"

Axon — "output wire"

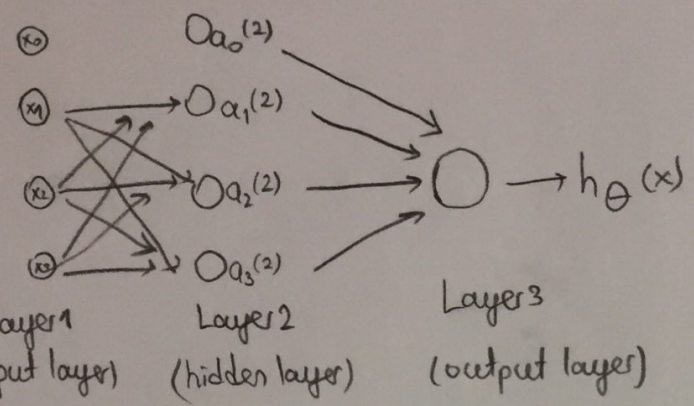
Neural model: Logistic unit



Sigmoid (logistic) activation fxn

Weights = θ's. = $\begin{bmatrix}\theta_0\\ \theta_1\\ \theta_2\\ \vdots\\ \theta_n\end{bmatrix}$ paramtrs.

NN:



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

$\theta^{(j)}$ = mtx of weights controlling fkn mapping from layer j to layer $j+1$

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

$$\vdots$$

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

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

2.2. Model Representatn -2:

Forward Propagatn: Vectorized implementatn.

$a_3^{(2)} = g(z_3^{(2)})$ $\quad x = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix}$ $\quad z^{(2)} = \begin{bmatrix} z_1^{(2)} \\ z_2^{(2)} \\ z_3^{(2)} \end{bmatrix}$ $\quad z^{(2)} = \theta^{(1)} \cdot x$

$a^{(2)} = g(z^{(2)})$ $\quad \hookrightarrow \mathbb{R}^3 \quad \hookrightarrow \mathbb{R}^3$

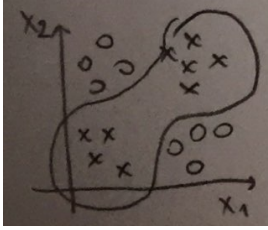
⇓

Instead of being contrained to feed the features (x_1, x_2, x_3) to logistic regression, it gets to learn its own features (a_1, a_2, a_3) to feed into logistic regression.

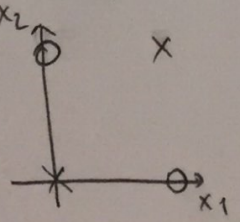
⇒ it can learn some pretty interesting and complex features & can end up with better hypothes

(3.) APPLICATIONS.

NN can be used to learn complex non-linear hypothesis.



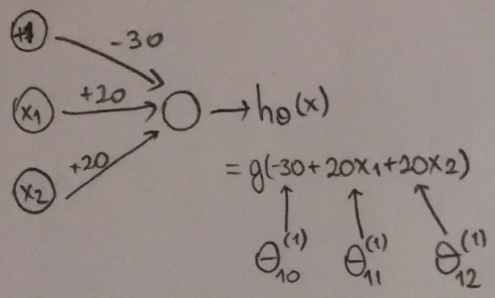
we'd like to learn a non-linear decision boundary

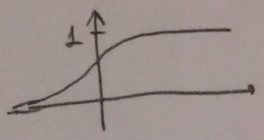


$y = x_1$ XOR x_2

x_1 XNOR x_2

AND $y = x_1$ AND x_2

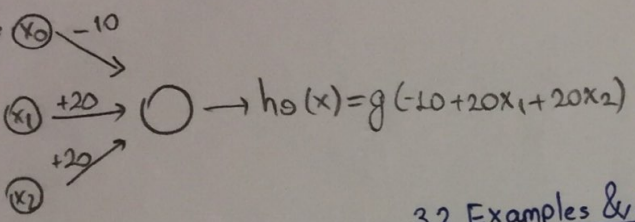




x_1	x_2	$h_\theta(x)$
0	0	$g(-30) \approx 0$
0	1	$g(-10) \approx 0$
1	0	$g(-10) \approx 0$
1	1	$g(10) \approx 1$

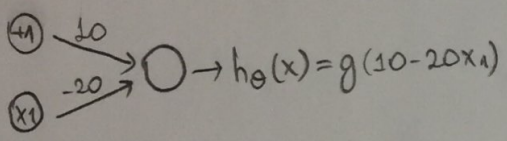
$h_\theta(x) \approx x_1$ AND x_2

OR:

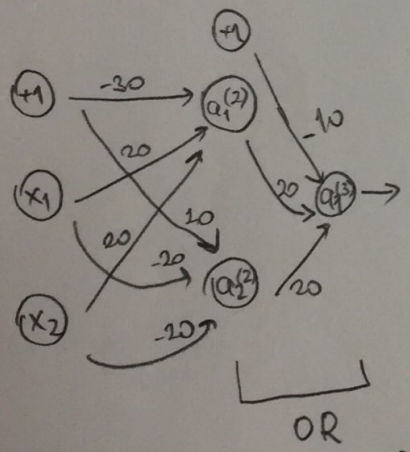


3.2. Examples & Intuitns-2:

NOT:



XNOR:



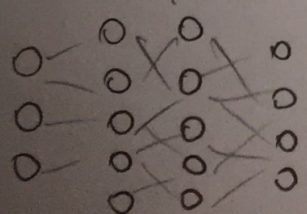
x_1	x_2	$a_1^{(2)}$	$a_2^{(2)}$	$h_\theta(x)$
0	0	0	1	1
0	1	0	0	0
1	0	0	0	0
1	1	1	0	1

3.3. Multiclass Classificatn:

multiple output units: one-vs-all (pedestrian, car, motorcycle, truck)

Want $h_\theta(x) = \begin{bmatrix}1\\0\\0\\0\end{bmatrix}$, $h_\theta(x) = \begin{bmatrix}0\\1\\0\\0\end{bmatrix}$, $h_\theta(x) = \begin{bmatrix}0\\0\\1\\0\end{bmatrix}$, $h_\theta(x) = \begin{bmatrix}0\\0\\0\\1\end{bmatrix}$

when pedestrian, when car, when motorcycle, when truck.



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