Contents

Neural Networks
Non linear hypothesis

June 2, 2016

Part I

Week 4

0.1 Introduction

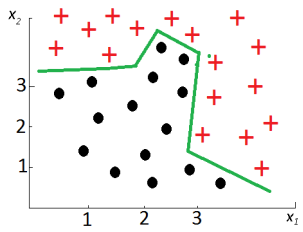


Figure 1: Non-linear classification

In this example with 2 features, one could use logistic regression with polynomial terms:

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

However, for larger number of features, the number of quadratic features will grow asymptotically. For example in a problem with 100 features $(x_1, ...x_{100})$, if we include all the 2nd order term: $x_1^2, x_1 x_2, x_2^2, ..$, we will end up with 5000 features (typically the number of quadratic features grow as $n^2/2$).

0.2 Model representation

0.2.1 Single Neuron model : logistic unit

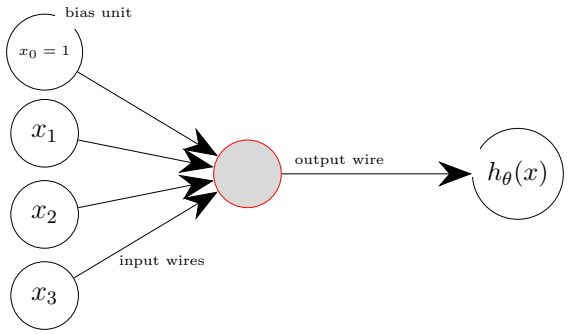
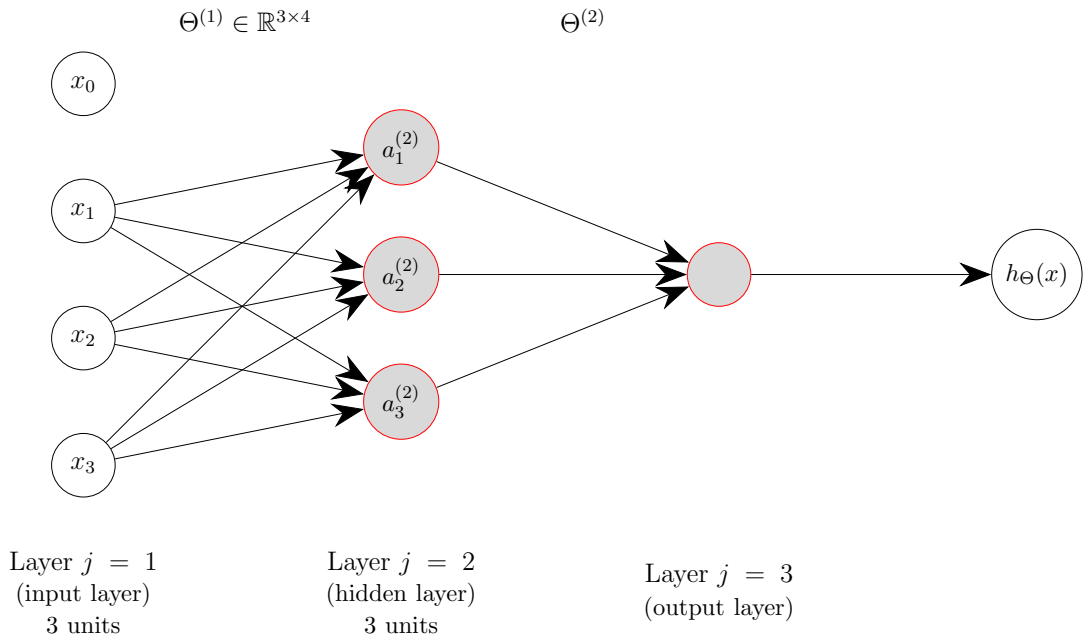


Figure 2: Single neuron with a Sigmoid (logistic) activation function, where $h_\theta(x) = \frac{1}{1+e^{(-\theta^T x)}}$

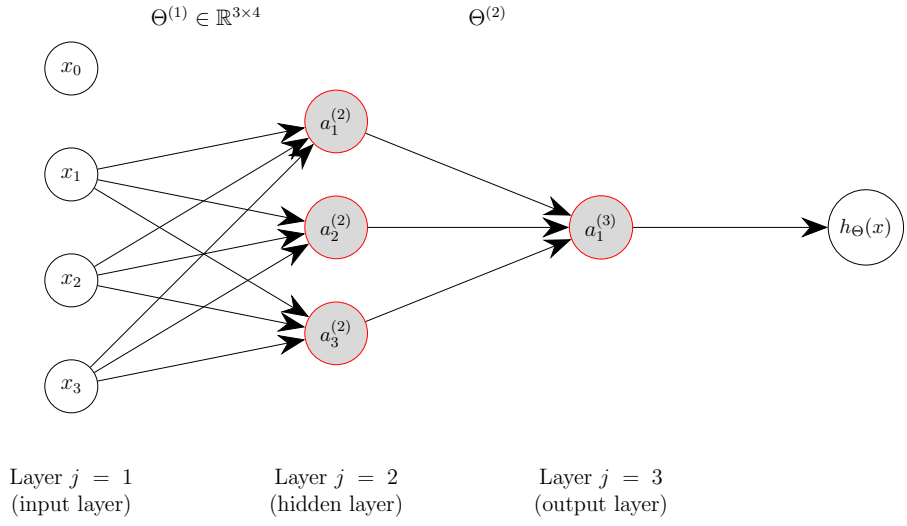
In neural network applications, the parameters $\theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \\ \theta_3 \end{bmatrix}$ are often called "**weights**". Note that the purpose of the activation function is to introduce non-linearity into the network. Non-linear means that the output cannot be reproduce from a linear combination of the inputs. Without a non-linear activation function in the network, a NN, no matter how many layers it had would behave just like a single perceptron. A common activation function used is *tanh*

0.2.2 Neuronal model (Diagram representation)



where $a_i^{(j)}$ is the activation of unit i in layer j. For example $a_1^{(2)}$ is the activation (value) of **unit 1** in **layer 2**.
$\Theta^{(j)}$ is the matrix of "weights" controlling the function mapping from layer j to layer $(j+1)$.

0.2.3 Mathematical representation



The activation value of the 3 hidden units of Layer 2 are:

$$a_1^{(2)} = g\left(\Theta_{1,0}^{(1)}x_0 + \Theta_{1,1}^{(1)}x_1 + \Theta_{1,2}^{(1)}x_2 + \Theta_{1,3}^{(1)}x_3\right)$$
$$a_2^{(2)} = g\left(\Theta_{2,0}^{(1)}x_0 + \Theta_{2,1}^{(1)}x_1 + \Theta_{2,2}^{(1)}x_2 + \Theta_{2,3}^{(1)}x_3\right)$$
$$a_3^{(2)} = g\left(\Theta_{3,0}^{(1)}x_0 + \Theta_{3,1}^{(1)}x_1 + \Theta_{3,2}^{(1)}x_2 + \Theta_{3,3}^{(1)}x_3\right)$$

The activation value of the Layer3 is:

$$
\begin{aligned}
h_\Theta(x) &= a_1^{(3)} \\
&= g\left(\Theta_{1,0}^{(2)} a_0^{(2)} + \Theta_{1,1}^{(2)} a_1^{(2)} + \Theta_{1,2}^{(2)} a_2^{(2)} + \Theta_{1,3}^{(2)} a_3^{(2)}\right)
\end{aligned}
$$

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

0.2.4 Vectorized implementation

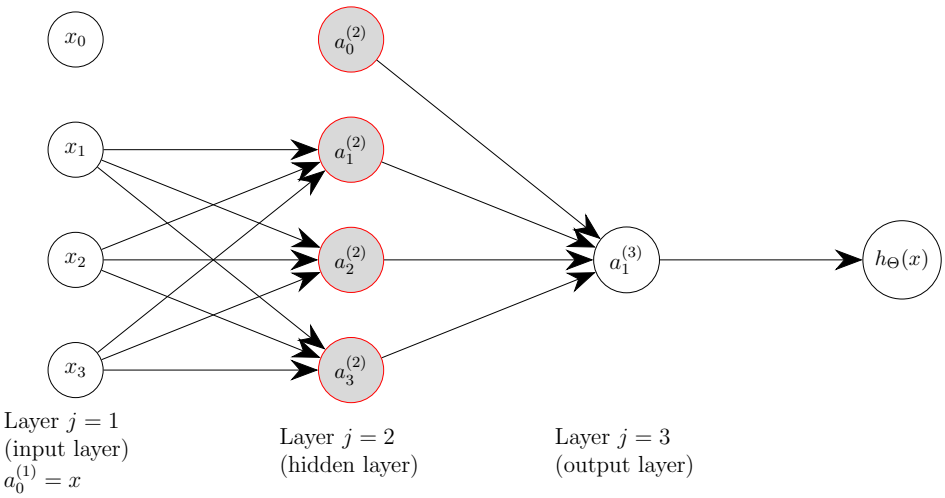
$$
\begin{aligned}
a_1^{(2)} &= g\underbrace{\left(\Theta_{1,0}^{(1)} x_0 + \Theta_{1,1}^{(1)} x_1 + \Theta_{1,2}^{(1)} x_2 + \Theta_{1,3}^{(1)} x_3\right)}_{z_1^{(2)}} = \Theta^{(1)} x = g(z_1^{(2)}) \\
a_2^{(2)} &= g\underbrace{\left(\Theta_{2,0}^{(1)} x_0 + \Theta_{2,1}^{(1)} x_1 + \Theta_{2,2}^{(1)} x_2 + \Theta_{2,3}^{(1)} x_3\right)}_{z_2^{(2)}} = \Theta^{(2)} x = g(z_2^{(2)}) \\
a_3^{(2)} &= g\underbrace{\left(\Theta_{3,0}^{(1)} x_0 + \Theta_{3,1}^{(1)} x_1 + \Theta_{3,2}^{(1)} x_2 + \Theta_{3,3}^{(1)} x_3\right.}_{z_3^{(2)}} = \Theta^{(3)} x = g(z_3^{(2)})
\end{aligned}
$$

where $z^{(2)}$ is for layer #2.
Remember that the activation applies the sigmoid function element-wise, i.e to each elements of $z^{(2)}$.
To make the notation more consistent, we define:

- $a_0^{(2)} = 1$ the bias unit
- $a^{(1)} = x$ (i.e activation of first layer)
- $h_\Theta(x) = a^{(3)} = g(z^{(3)})$ where $z^{(3)} = \Theta^{(2)} a^{(2)}$

$$
\begin{aligned}
h_\Theta(x) &= g(\underbrace{\Theta_{1,0}^{(2)} + \Theta_{1,1}^{(2)} a_1^{(2)} + \Theta_{1,2}^{(2)} a_2^{(2)} + \Theta_{1,3}^{(2)} a_3^{(2)}}_{z^{(3)}}) \\
&= g(z^{(3)}) \\
&= g(\Theta^{(2)} a^{(2)})
\end{aligned}
$$

$\Theta^{(1)} \in \mathbb{R}^{3\times 4}$ $\Theta^{(2)}$



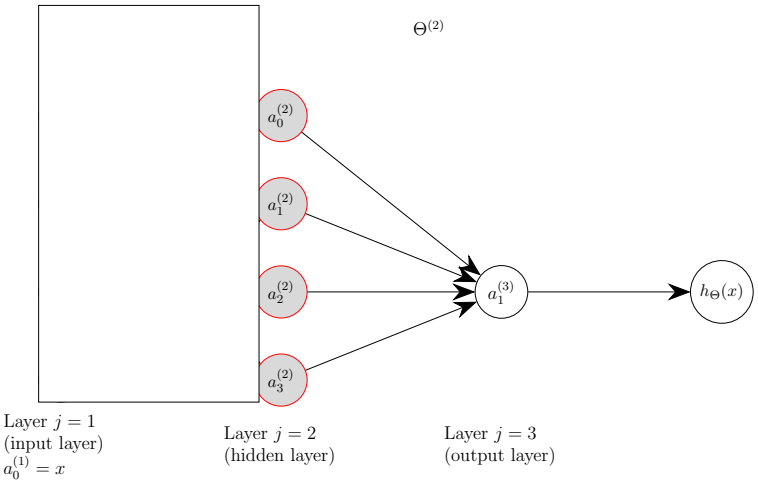
This method of computing $h_{(\Theta}(x)$ is called **forward propagation** : we start by calculating the activation of the input $\Rightarrow$ hidden $\Rightarrow$ output layers.

0.2.5 what else?

If the Layer1 is covered up, the diagram is similar to a logistic regression, with $h_\Theta(x)$:

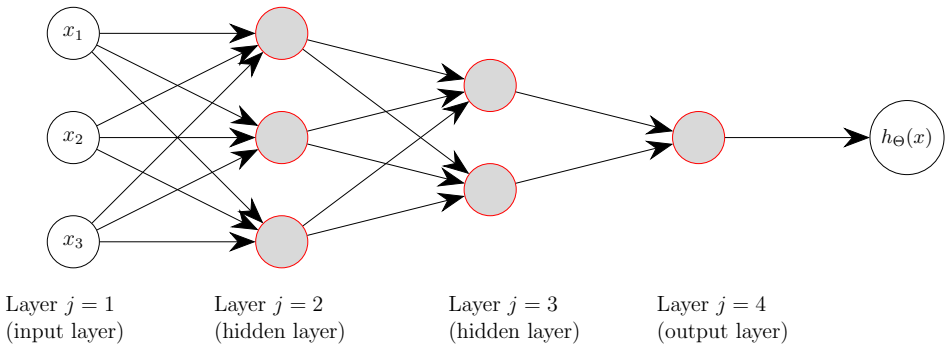
$$h_\Theta(x) = g(\Theta_0 a_0 + \Theta_1 a_1 + \Theta_2 a_2 + \Theta_3 a_3)$$

where the features used are now a_1, a_2, a_3 (rather than x_1, x_2, x_3), but computed by the hidden layer.



In the 3 layer neural network, the function mapping from layer 1 to layer 2 is determined by the set of parameters Θ_1, i.e the neural network learned its own features a_1, a_2, a_3 to feed into logistic regression.

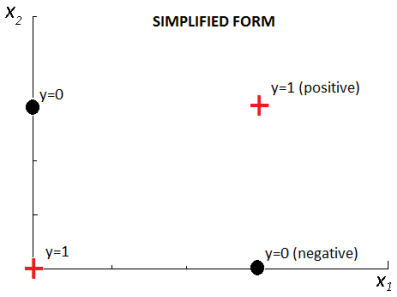
There can be other **network architectures**:

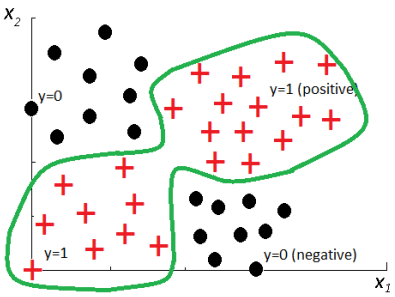


0.3 Non-Linear classification examples

0.3.1 Exemple I

Let's consider the following problem on the left (simplified version)



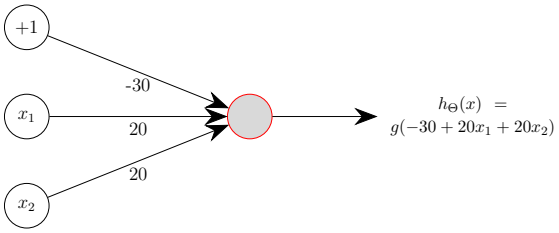


where x_1 and x_2 are binary $\in \{0, 1\}$.

- XOR = exclusive OR (output TRUE ($y = 1$) only when inputs differ $x_1 \neq x_2$).
- XNOR = NOT(x_1 OR x_2) (complement of XOR) (output TRUE ($y = 1$) if ($x_1 = x_2$) and FALSE ($y = 0$) if ($x_1 \neq x_2$)

0.3.2 Logical AND function

x_1 and $x_2 \in \{0, 1\}$ **AND** statement means: $y = 1$ if x_1 and $x_2 = 1$.

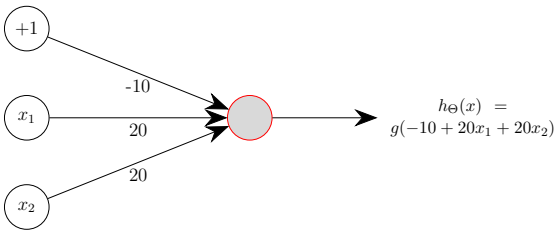


x_1	x_2	$h_\Theta(x)$	
0	0	g(-30) $\approx$ 0	0
0	1	g(-30) $\approx$ 0	0
1	0	g(-30) $\approx$ 0	0
1	1	g(-30) $\approx$ 1	1

This represent the logical AND function, where $h_\Theta(x) = 1$ if $x_1 = 1$ **AND** $x_2 = 1$.

0.3.3 Logical OR function

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

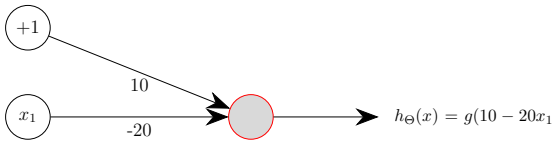


x_1	x_2	$h_\Theta(x)$	
0	0	g(-10) $\approx$ 0	0
0	1	g(10) $\approx$ 1	1
1	0	g(10) $\approx$ 1	1
1	1	g(30) $\approx$ 1	1

This represent the logical OR function, where $h_\Theta(x) = 1$ if $x_1 = 1$ **OR** $x_2 = 1$.

0.3.4 Function Negation NOT x_1

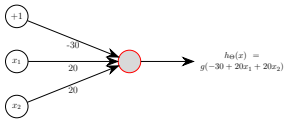
x_1 and $x_2 \in \{0, 1\}$



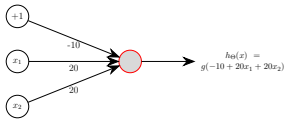
x_1	$h_\Theta(x)$	
0	g(10) $\approx$ 1	1
1	g(-10) $\approx$ 1	0

This represent the logical Negate x_1: $y = 1$ if **NOT**(x_1)=TRUE.

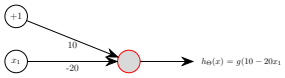
0.3.5 Computing x_1 XNOR x_2



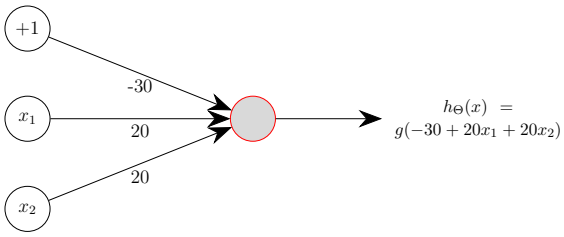
x_1 AND x_2



(NOT x_1) AND (NOT x_2)



x_1 NOT 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

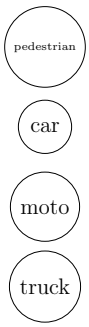
In summary:

- $h_\Theta(x) = 1$ if ($x_1 = 0$ AND $x_2 = 0$) OR ($x_1 = 1$ AND $x_2 = 1$)
- $h_\Theta(x) = 0$ otherwise

This is modeled well the problem in the figure

0.3.6 ExampleII: Multiclass classification

In this example, we define 4 classes of objects (Pedestrian, Car, Moto and Truck), and want to classify a new object within those classes. This is a multiple output unit (One-vs-all). We will build a neuro-network with 4 output units (i.e $h_\Theta(x) \in \mathbb{R}^4$):



So, we want the ouput $h_\Theta(x)$ to take the form:

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

pedestrians car moto truck

The training set would be of the form: $(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), ..., (x^{(m)}, y^{(m)})$, where $y^{(i)}$ is either

$$\begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} \text{ or } \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} \text{ or } \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} \text{ or } \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

Appendices

.1 One-vs-All assignment

.2 Answers to Quiz

- Suppose you have a multi-class classification problem with 3 classes trained with a 3 layer network. Let $a_1^{(3)} = (h_\Theta(x))_1$ be the activation of the 1st output unit, and similarly $a_2^{(3)} = (h_\Theta(x))_2$, $a_3^{(3)} = (h_\Theta(x))_3$. Then for any input x, it must be the case that $a_1^{(3)} + a_2^{(3)} + a_3^{(3)} = 1$. $\Rightarrow$ FALSE