E. Neural Network: Representation

I. Motivations

II. Neural Network

1. Model representation I

Represent a hypothesis function using neural networks: At a very simple level, neurons are basically computational units that take inputs (dendrites (đuôi gai)) as electrical inputs (called "spikes" (gai)) that are channeled to outputs (axons (sợi trục)). In this model, dendrites are like the input features $x_1...x_n$, and the output is the result of our hypothesis function. In this model our x_0 input node is sometimes called the "bias unit." It is always equal to 1.

In neural networks, use the same logistic function as in classification $\frac{1}{1+e^{-\theta^T x}}$, sometimes call it a sigmoid (logistic) activation function. In this situation, our "theta" parameters are sometimes called "weights".

Visually, a simplistic representation looks like:

$$\begin{bmatrix} x_0 \\ x_1 \\ x_2 \end{bmatrix} \rightarrow [\quad] \rightarrow h_\theta(x)$$

Our input nodes (layer 1), also known as the "input layer", go into another node (layer 2), which finally outputs the hypothesis function, known as the "output layer".

We can have intermediate layers of nodes between the input and output layers called the "hidden layers."

In this example, we label these intermediate or "hidden" layer nodes $a_0^2...a_n^2$ and call them "activation units."

$$a_i^{(j)} = \text{"activation" of unit } i \text{ in layer } j$$
$$\Theta^{(j)} = \text{matrix of weights controlling function mapping from layer } j \text{ to layer } j+1$$

If we had one hidden layer, it would look like:

$$\begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix} \rightarrow \begin{bmatrix} a_1^{(2)} \\ a_2^{(2)} \\ a_3^{(2)} \end{bmatrix} \rightarrow h_\theta(x)$$

The values for each of the "activation" nodes is obtained as follows:

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

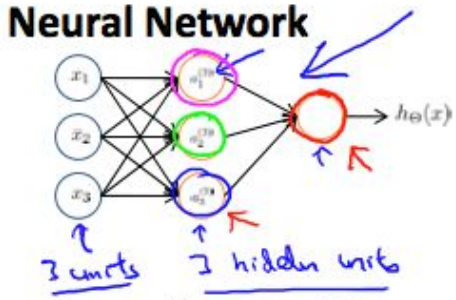
Compute activation nodes by using a 3×4 matrix of parameters. Apply each row of the parameters to our inputs to obtain the value for one activation node. Hypothesis output is the logistic function applied to the sum of the values of our activation nodes, which have been multiplied by yet another parameter matrix $\Theta^{(2)}$ containing the weights for our second layer of nodes.

Each layer gets its own matrix of weights, $\Theta^{(j)}$.

The dimensions of these matrices of weights is determined as follows:

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

The +1 comes from the addition in $\Theta^{(j)}$ of the "bias nodes," x_0 and $\Theta_0^{(j)}$. In other words the output nodes will not include the bias nodes while the inputs will. The summarizes model representation:



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

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

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

$h_\Theta(x)$

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

$h_\Theta(x)^{(2)}$

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

$s_{j+1} \times (s_j + 1)$

2. Model representation II

An example of a neural network:

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

Define a new variable $z_k^{(j)}$ that encompasses the parameters inside g function. In previous example if replaced by the variable z for all the parameters, we would get:

$$
\begin{aligned}
a_1^{(2)} &= g(z_1^{(2)}) \\
a_2^{(2)} &= g(z_2^{(2)}) \\
a_3^{(2)} &= g(z_3^{(2)})
\end{aligned}
$$

In other words, for layer j=2 and node k, the variable z will be:

$$
z_k^{(2)} = \Theta_{k,0}^{(1)}x_0 + \Theta_{k,1}^{(1)}x_1 + \cdots + \Theta_{k,n}^{(1)}x_n
$$

The vector representation of x and z^j is:

$$
x = \begin{bmatrix} x_0 \\ x_1 \\ \cdots \\ x_n \end{bmatrix} \quad z^{(j)} = \begin{bmatrix} z_1^{(j)} \\ z_2^{(j)} \\ \cdots \\ z_n^{(j)} \end{bmatrix}
$$

Setting $x = a^{(1)}$, we can rewrite the equation is:

$$
z^{(j)} = \Theta^{(j-1)}a^{(j-1)}
$$

Multiplying matrix $\Theta^{(j-1)}$ with dimensions $s_j(n+1)$ (where s_j is the number of activation nodes) by vector $a^{(j-1)}$ with height (n+1). Output is vector $z^{(j)}$ with height s_j. Get a vector of our activation nodes for layer j as follows:

$$
a^{(j)} = g(z^{(j)})
$$

Where our function g can be applied element-wise to vector $z^{(j)}$.
Then add a bias unit (equal to 1) to layer j after we have computed $a^{(j)}$. This will be element $a_0^{(j)}$ and will be equal to 1. To compute final hypothesis, let's first compute another z vector:

$$
z^{(j+1)} = \Theta^{(j)}a^{(j)}
$$

We get this final z vector by multiplying the next theta matrix after $\Theta^{(j-1)}$ with the values of all the activation nodes we just got. This last theta matrix $\Theta^{(j)}$ will have

only **one row** which is multiplied by one column $a^{(j)}$ so that our result is a single number. We then get our final result with:

$$h_\Theta(x) = a^{(j+1)} = g(z^{(j+1)})$$

Notice that in this last step, between layer j and layer j+1, exactly the same thing as logistic regression. Adding all these intermediate layers in neural networks allows to more elegantly produce interesting and more complex non-linear hypotheses.

III. Application

1. Examples and intuitions 1

A simple example of applying neural networks is by predicting x_1 AND x_2, which is the logical 'and' operator and is only true if both of them are 1. The graph of functions:

$$\begin{bmatrix} x_0 \\ x_1 \\ x_2 \end{bmatrix} \rightarrow \left[g(z^{(2)}) \right] \rightarrow h_\Theta(x)$$

x_0 is a bias variable and is always 1. The first theta matrix as:

$$\Theta^{(1)} = [-30 \quad 20 \quad 20]$$

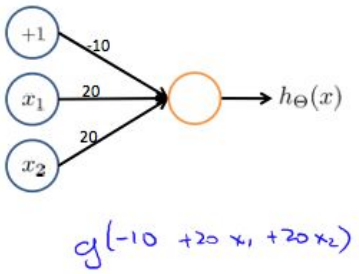
The output of the hypothesis to only be positive if both x_1 and x_2 are 1. In other words:

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

$$\begin{aligned} &x_1 = 0 \text{ and } x_2 = 0 \text{ then } g(-30) \approx 0 \\ &x_1 = 0 \text{ and } x_2 = 1 \text{ then } g(-10) \approx 0 \\ &x_1 = 1 \text{ and } x_2 = 0 \text{ then } g(-10) \approx 0 \\ &x_1 = 1 \text{ and } x_2 = 1 \text{ then } g(10) \approx 1 \end{aligned}$$

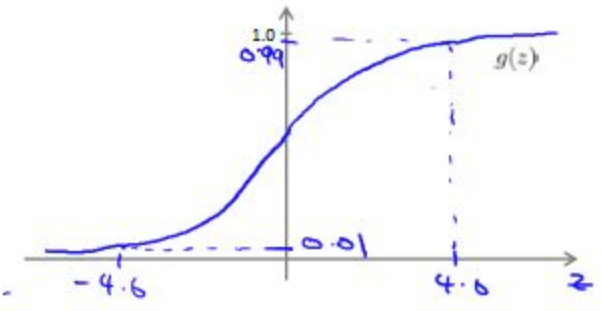
Other logical gate:

Example: OR function



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

Where g(z) is the following:



2. Examples and intuitions 2

The $\Theta^{(1)}$ matrices of AND, NOR, OR are:

$$
\begin{aligned}
&AND: \\
&\quad \Theta^{(1)} = [-30 \quad 20 \quad 20] \\
&NOR: \\
&\quad \Theta^{(1)} = [10 \quad -20 \quad -20] \\
&OR: \\
&\quad \Theta^{(1)} = [-10 \quad 20 \quad 20]
\end{aligned}
$$

Can combine these to get the XNOR logical operator (which gives 1 if x_1 and x_2 are both 0 or both 1).

$$
\begin{bmatrix} x_0 \\ x_1 \\ x_2 \end{bmatrix} \rightarrow \begin{bmatrix} a_1^{(2)} \\ a_2^{(2)} \end{bmatrix} \rightarrow \left[a^{(3)} \right] \rightarrow h_\Theta(x)
$$

For the transition between the first and second layer, use a $\Theta^{(1)}$ matrix that combines the values for AND and NOR:

$$
\Theta^{(1)} = \begin{bmatrix} -30 & 20 & 20 \\ 10 & -20 & -20 \end{bmatrix}
$$

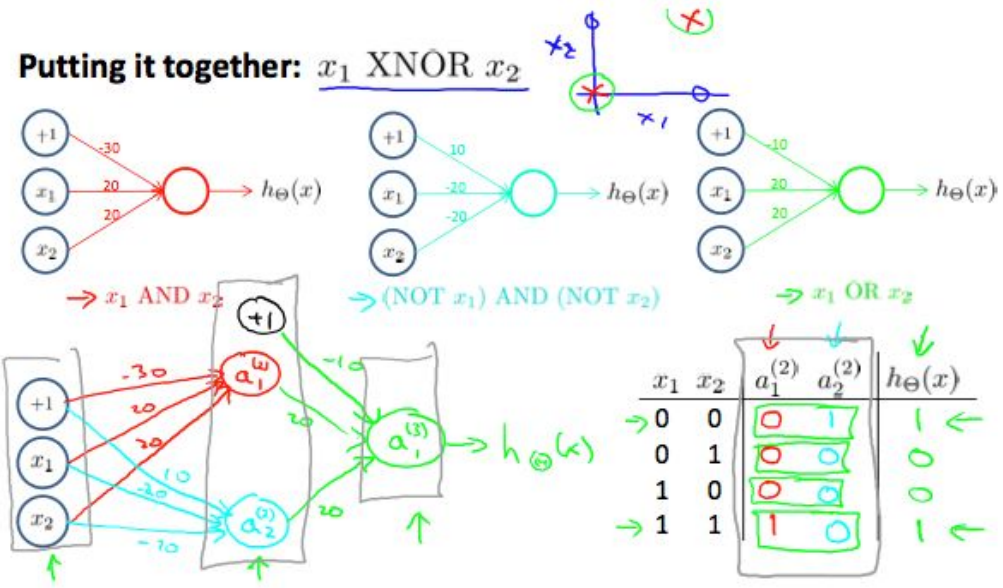
For the transition between the second and third layer, we'll use a $\Theta^{(2)}$ matrix that uses the value for OR:

$$
\Theta^{(2)} = [-10 \quad 20 \quad 20]
$$

Let's write out the values for all nodes:

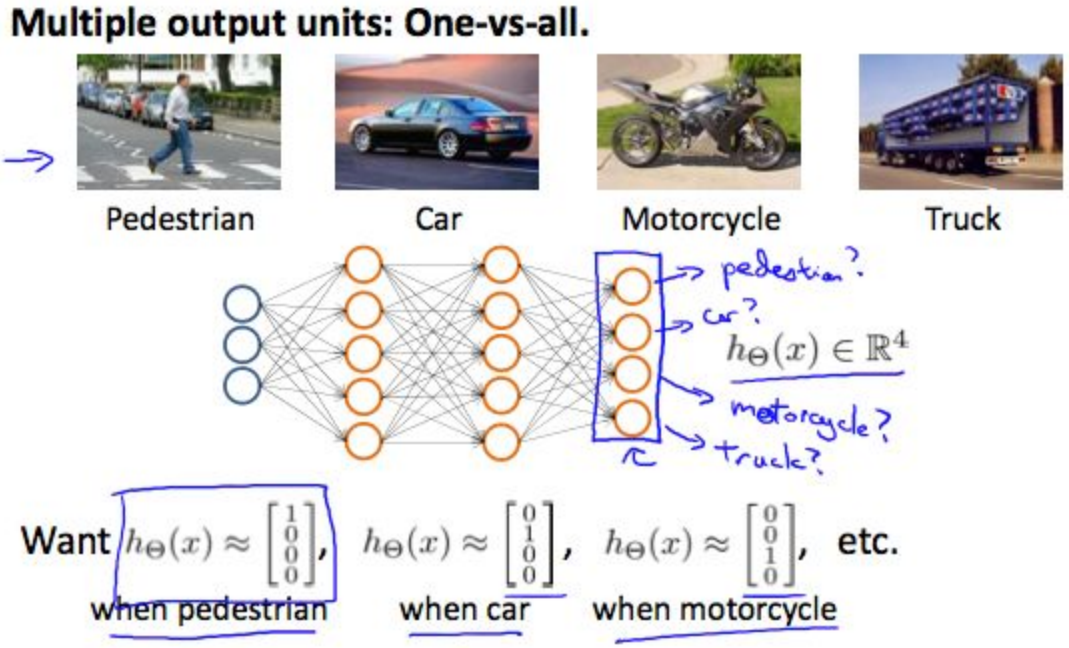
$$
\begin{aligned}
a^{(2)} &= g(\Theta^{(1)} \cdot x) \\
a^{(3)} &= g(\Theta^{(2)} \cdot a^{(2)}) \\
h_\Theta(x) &= a^{(3)}
\end{aligned}
$$

Output: the XNOR operator using a hidden layer with two nodes. The following summarizes:



3. Multiclass classification

To classify data into multiple classes, the hypothesis function returns a vector of values. Example: classify the data into one of four categories:



Define the set of resulting classes as y:

$$y^{(i)} = \begin{bmatrix}1\\0\\0\\0\end{bmatrix}, \begin{bmatrix}0\\1\\0\\0\end{bmatrix}, \begin{bmatrix}0\\0\\1\\0\end{bmatrix}, \begin{bmatrix}0\\0\\0\\1\end{bmatrix},$$

Each $y^{(i)}$ represents a different image corresponding to either a car, pedestrian, truck, or motorcycle. The inner layers provide some new information which leads to final hypothesis function. The setup looks like:

$$\begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ \cdots \\ x_n \end{bmatrix} \rightarrow \begin{bmatrix} a_0^{(2)} \\ a_1^{(2)} \\ a_2^{(2)} \\ \cdots \end{bmatrix} \rightarrow \begin{bmatrix} a_0^{(3)} \\ a_1^{(3)} \\ a_2^{(3)} \\ \cdots \end{bmatrix} \rightarrow \cdots \rightarrow \begin{bmatrix} h_\Theta(x)_1 \\ h_\Theta(x)_2 \\ h_\Theta(x)_3 \\ h_\Theta(x)_4 \end{bmatrix}$$

The resulting hypothesis for one set of inputs may look like:

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

In which case, resulting class is the third one down, or $h_\Theta(x)_3$, which represents the motorcycle.