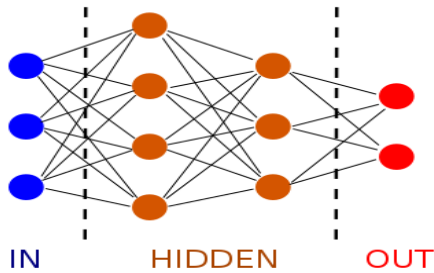


Neural Networks

Neural networks are limited imitations of how our own brains work.

Visual Representation



- Input and Output mandatory
- Hidden is optional but recommended
- 0..N Hidden Layer possible
- 0..N Neuron per each Hidden layer possible

Output Hypothesis calculation

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

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

$$a_2^{(2)} = g(\Theta_{20}^{(1)}x_0 + \Theta_{21}^{(1)}x_1 + \Theta_{22}^{(1)}x_2 + \Theta_{23}^{(1)}x_3)$$

$$a_3^{(2)} = g(\Theta_{30}^{(1)}x_0 + \Theta_{31}^{(1)}x_1 + \Theta_{32}^{(1)}x_2 + \Theta_{33}^{(1)}x_3)$$

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

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

Back propagation Algorithm

Given training set $\{(x^{(1)}, y^{(1)}) \dots (x^{(m)}, y^{(m)})\}$

- Set $\Delta_{i,j}^{(l)} := 0$ for all (L, i, j)

For training example $t=1$ to m :

- Set $a^{(i)} := x^{(t)}$

- Perform forward propagation to compute $a^{(l)}$ for $l = 2, 3, \dots, L$

- Using $y^{(t)}$, compute $\delta^{(L)} = a^{(L)} - y^{(t)}$

- Compute $\delta^{(L-1)}, \delta^{(L-2)}, \dots, \delta^{(2)}$ using $\delta^{(l)} = ((\Theta^{(l)})^T \delta^{(l+1)}) \cdot a^{(l)} \cdot (1 - a^{(l)})$

- $\Delta_{i,j}^{(l)} := \Delta_{i,j}^{(l)} + a_j^{(l)} \delta_i^{(l+1)}$

- $D_{i,j}^{(l)} := \frac{1}{m} (\Delta_{i,j}^{(l)} + \Delta_{i,j}^{(l+1)})$ if $j \neq 0$

- $D_{i,j}^{(l)} := \frac{1}{m} \Delta_{i,j}^{(l)}$ if $j = 0$

- The capital-delta matrix is used as an "accumulator" to add up our values as we go along and eventually compute our partial derivative.