Exercise for Chapter #4 (ANN)

Objective: The objective of this section is to exercise the NN presentation and to be familiar with the NN coding (in MATLAB, attached separately).

In the NN lecture note, we used the following figure to present the Neural Network (See Figure: Chapter 4, Page 23, Figure H).

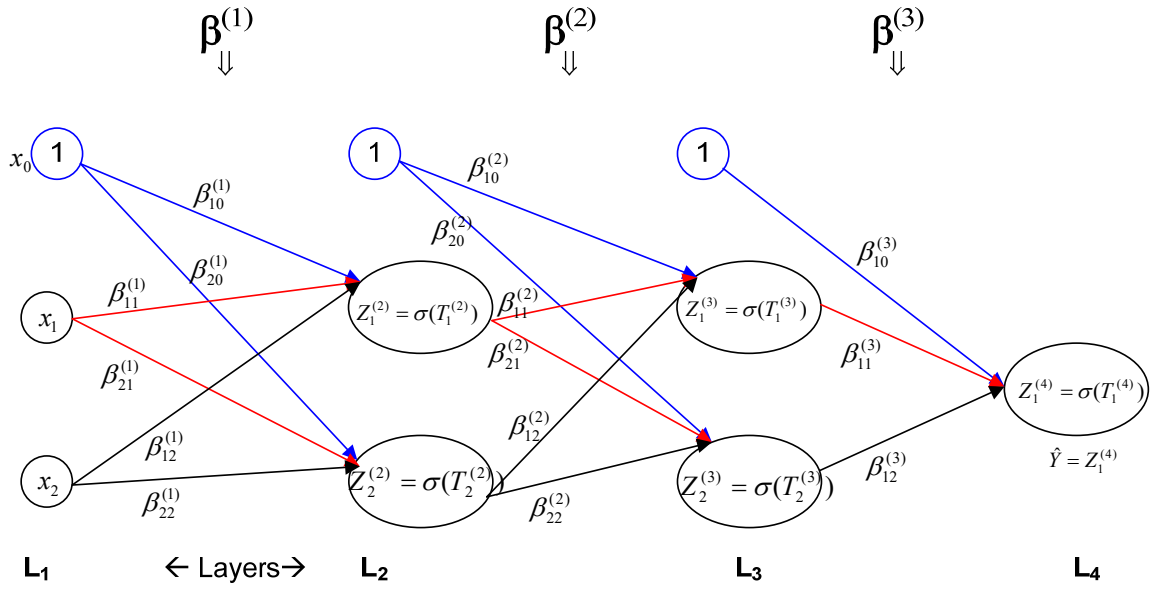

Figure A: A multilayer neural network demonstrating the notations.

For our coding purpose, we will be using the following figure, where the changes are trivial – we will consider the bias unit to be at the bottom rather than at the top.

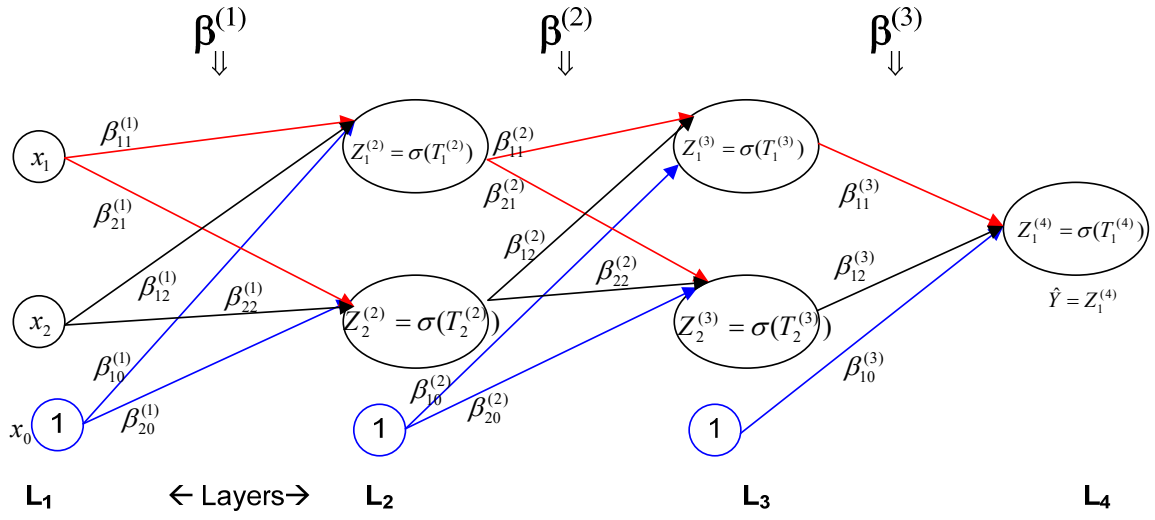

Figure B: A multilayer neural network demonstrating the notations.

From Figure (B), we can write:

Input vector, $X = \begin{bmatrix} X_1 \\ X_2 \\ X_0 \end{bmatrix}$, and Beta or Weight vector$^{(1)}$, $\beta^{(1)} = \begin{bmatrix} \beta_{11}^{(1)} & \beta_{21}^{(1)} \\ \beta_{12}^{(1)} & \beta_{22}^{(1)} \\ \beta_{10}^{(1)} & \beta_{20}^{(1)} \end{bmatrix}$

In general for β, if number of units in a layer is U_l, we can say $\beta^{(l)} \in \Re^{(U_l+1)\times U_{(l+1)}}$

The Equation for term ($T^{(2)}$) can be considered:

$$T_1^{(2)} = X_1\beta_{11}^{(1)} + X_2\beta_{12}^{(1)} + X_0\beta_{10}^{(1)}$$

$$T_2^{(2)} = X_1\beta_{21}^{(1)} + X_2\beta_{22}^{(1)} + X_0\beta_{20}^{(1)}$$

Also the terms ($T^{(2)}$) can be presented as the inner product of $\beta^{(1)}$ and X:

$$T^{(2)} = \begin{bmatrix} T_1^{(2)} \\ T_2^{(2)} \end{bmatrix} = \beta^{(1)^T} X = \begin{bmatrix} \beta_{11}^{(1)} & \beta_{12}^{(1)} & \beta_{10}^{(1)} \\ \beta_{21}^{(1)} & \beta_{22}^{(1)} & \beta_{20}^{(1)} \end{bmatrix} \cdot \begin{bmatrix} X_1 \\ X_2 \\ X_0 \end{bmatrix} = \begin{bmatrix} X_1\beta_{11}^{(1)} + X_2\beta_{12}^{(1)} + X_0\beta_{10}^{(1)} \\ X_1\beta_{21}^{(1)} + X_2\beta_{22}^{(1)} + X_0\beta_{20}^{(1)} \end{bmatrix}$$

Therefore, in short-form we can write:

$$T^{(2)} = \beta^{(1)^T} X \tag{1}$$

In general we can assume Z is the output of each of the layer including the input layer, therefore (1) can also be written as

$$T^{(2)} = \beta^{(1)^T} Z^{(1)} \tag{2}$$

$$Z^{(2)} = f_{sig}(T^{(2)}) = f_{sig} \cdot \left(\beta^{(1)^T} Z^{(1)}\right), \tag{3}$$

sigmoid operation in (3) is element wise operation.

Equation (3) will produce the following:

$$Z^{(2)} = \begin{bmatrix} Z_1^2 \\ Z_2^2 \end{bmatrix} \tag{4}$$

However, since bias term is added in each layer (and it is not produced as a result of the produce in Equation (2) or (3) for example, therefore we will need to add the bias term in Equation (4) :

$$Z^{(2)} = \begin{bmatrix} Z_1^{(2)} \\ Z_2^{(2)} \\ Z_0^{(2)} \end{bmatrix} \tag{5}$$

where, $Z_0^{(2)} = 1$.

Similarly for the other layers (of Figure B), we can write:

$$T^{(3)} = \beta^{(2)^T} Z^{(2)} \tag{6}$$

$Z^{(3)} = f_{sig} \cdot \left(T^{(3)}\right)$ and then add $Z_0^{(3)} = 1$

And forming:

$$Z^{(3)} = \begin{bmatrix} Z_1^{(3)} \\ Z_2^{(3)} \\ Z_0^{(3)} \end{bmatrix} \tag{7}$$

$$T^{(4)} = \beta^{(3)^T} Z^{(3)} \tag{8}$$

$$\hat{Y} = Z^{(4)} = f_{sig}(T^{(4)}) \tag{9}$$

So far these equations from (1) to (9), form the vectorized implementation of the feed-forward propagation of the network presented in Figure B.

To recap, we summarize the feed-forward propagation as:

$$\begin{aligned}
&Z^{(1)} = X \\
&T^{(2)} = \beta^{(1)^T} Z^{(1)} \\
&Z^{(2)} = f_{sig}.(T^{(2)}) \quad \text{and add } Z_0^{(2)} \\
&T^{(3)} = \beta^{(2)^T} Z^{(2)} \\
&Z^{(3)} = f_{sig} \cdot \left(T^{(3)}\right) \text{ and add } Z_0^{(3)} \\
&T^{(4)} = \beta^{(3)^T} Z^{(3)} \\
&Z^{(4)} = \hat{Y} = f_{sig}(T^{(4)})
\end{aligned} \tag{10}$$

We will review the backpropagation algorithm described in lecture #4, page 14, especially in the context of implementation.

Algorithm 1: Error Back-propagation

BEGIN

1. For a data point (x_i, y_i), apply an input vector x_i to the network and forward propagate through the network following Equation Set #(**10**).

2. For each of the output node / unit compute the error term δ^L:

$$\delta^{(L)} = (Z_k^{(L)} - Y_k).*Z_k^L \cdot*(1 - Z_k^L)$$

Here, L = the last layer or the output layer and
Note that "$.*Z_k^L \cdot*(1-Z_k^L)$" is done element wise.

3. For each of the hidden layer node/unit compute the error term $\delta^{(L-1)}$:

$$\delta^{(L-1)} = Z^{(L-1)}.*(1 - Z^{(L-1)}).*\sum_{k=1}^{K}[\delta^{(L)^T}\beta^{(L-1)}]$$

And Compute: $\delta^{(L-2)}, \delta^{(L-3)}, ..., \delta^{(2)}$, except $\delta^{(1)}$.

4. Update the weights (β) of the network for i=*1*, *2*, …, (*L*-1)

$$\beta^{(i)}(t+1) = \beta^{(i)}(t) - \alpha.*[Z^{(i)}\delta^{(i+1)^T}]$$

$$\beta_0^{(i)}(t+1) = \beta_0^{(i)}(t) - \alpha.*[\delta^{(i+1)^T}] \qquad \text{[For bias terms]}$$

END

Next we discuss MATLAB code, implementing binary classification for our regular toy problem "Cancer prediction" ∈ {Benign, Malignant}. Please check the exercise folder for data and code (ANN_matlab.txt).