Multimedia Appendix 3: FFNN

The FFNN computation process with mathematical foundations is explained as follows:

Activation function

We used the hyperbolic tangent sigmoid, which is a logistic function and ranges from 0 to 1.

$$y_{v_i} = 2/\left(1 + exp(-2 * v_i)\right) - 1 \tag{3.1}$$

where y_i is the output of the ith node (neuron) and v_i is the weighted sum of the input connections.

Training

Learning occurs by changing connection weights after each piece of data is processed, based on the amount of error in the output compared to the expected result.

We can represent the degree of error in an output node j in the n th data point (training example) by :

$$e_j(n) = d_j(n) - y_j(n) \tag{3.2}$$

where $d_j(n)$ is the desired target value for nth data point at node j, and $y_j(n)$ is the value produced at node j when the n th data point is given as an input.

The node weights can then be adjusted based on corrections that minimize the error in the entire output for the n th data point, given by:

$$\varepsilon(n) = \frac{1}{2} \sum_{output\ node\ j} e_j^2(n) \tag{3.3}$$

To change the hidden layer weights, the output layer weights change according to the derivative of the activation function (backpropagation of the activation function).