11-10-2021

MACHINE LEARNING

OBSERVATION - 06

SACHIN RAGHUL T

2019103573

The Multi Layer Perceptron Algorithm:

- **Initialisation**

Initialise all weights to small (+ve and -ve) random values.

- **Training**

Repeat:

Forward phase

- compute the activation of each neuron j in the hidden layers using,

$$h_j = \sum_{i=0}^{L} x_i v_{ij}$$

$$a_j = g(h_j) = \frac{1}{1 + \exp(-\beta h_j)}$$

- work through the network until you get to the output layer neuron, which have activations (although).

$$h_K = \sum_j a_j w_{jK}$$

$$y_K = g(h_K) = \frac{1}{1 + \exp(-\beta h_K)}$$

Backward phase:

- Compute the error at the output using,

$$\delta_o(k) = (y_k - t_k)\, y_k (1 - y_k)$$

- compute the error in the hidden layer.

$$\delta_h(s) = a_s(1 - a_s) \sum_{k=1}^{N} w_s \delta_o(k)$$

- Update the output layer weight using:

$$w_{sk} \leftarrow w_{sk} - \eta\, \delta_o(k)\, a_s \text{ hidden}.$$

- Update the hidden layer weight using:

$$v_i \leftarrow v_i - \eta\, \delta_h(k)\, x_i$$

*(If using sequential updating) randomise the order of the input vectors so that you don't train in exactly the same order each iteration

- until learning stops.