11/10/2021

ML- week-6- observation

2019103555

Pranava Raman
BMS

Aim:-

To implement the Multiple Layer Percepta to solve

i) XOR problem

ii) iris-dataset

and implement single layer perceptron for iris-dataset.

The Multi-Layer Perceptron Algorithm

* Initialization

- all weights to small (+ve or -ve) randon values.

* Training

- repeat:

* foreach input vector:-

Forward Phase

- compute activation of each neuron j in the hidden layer(s) using

$$h_c = \sum_{i=0}^{L} x_i v_{ic}$$

$$a_c = g(h_c) = \frac{1}{1 + e^{\wedge}(-\beta h_c)}$$

* work through the network until you get the output layer neurons, which have activations

$$h_k = \sum_j a_j w_{jk} \qquad y_k = g(h_k) = \frac{1}{1+e^{\wedge}(-\beta h_k)}$$

· Backwards Phase

* compute the error of the output using:-

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

* compute the error in hidden layers using:-

$$\delta_h(c) = a_c(1 - a_c) \sum_{k=1}^{N} w_c \, \delta_o(k)$$

* update the output layer weights using:-

$$w_{ck} \leftarrow w_{ck} - \eta \, \delta_o(k) \, a_c^{hidden}$$

* update hidden layers using:-

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

* (if using sequential updting) randomize the order of input vectors so that you don't train in the same order

- until larning stops.

* Recall

- use the forward phase in the training section above.