***Neural Networks (Python)*** - **Implement a backpropagation algorithm from scratch (can use numpy, pandas, matplotlib) with the following data given in a CSV file:**

**Table

Description automatically generatedInput**:

**Expected Output**:

Table

Description automatically generated

**Notes –**

* Treat weights as matrices and hidden nodes as vectors
* Add 1 bias node to input and hidden layer
* Please use 4 input nodes, 1 hidden layer with 5 or 6 hidden nodes and output layer with 4 nodes ( i = 4, j = 5, k = 4)

1. **Set all weights to random numbers between 0 and 1 (random # generator seed should be constant) and use sigmoid activation.**

Text

Description automatically generated

1. **Back propagation error calculation using (For all weights jk (middle & output layer):**

Text

Description automatically generatedWhere the weighted sum is taken of the δ values of all units that receive output from unit j.

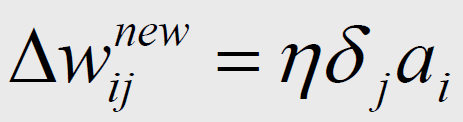
f '(S) = the derivative of the sigmoid function f

f '(S) = f (Sj)(1-f (Sj))

Sj = weighted sum of inputs to j

1. **Use the Delta rule to update weights: (with a learning coefficient = 0.5)**

Use ij for weights between input & middle layer. And jk for weights between middle & output layer

 A picture containing text, clock, antenna, gauge

Description automatically generated

1. **Next Training Pattern (3 in total):**

* Rinse, repeat until you have moved through the entire training set (ideally randomly\*).
* This is one iteration, (or epoch)
* Calculate (& plot) one RMS error value for each iteration.

\*Initially, move sequentially through all training patterns, and when it works, move to making it random

1. **Measure Training Performance using RMS and plot:**

A picture containing diagram

Description automatically generated

1. **Test on some data (can shuffle around same data or use some other dataset):**