UNIT-IV
MultiLayer Perceptron

⇒ Hidden layer perform complex task and gradient computation (extract low level features)
⇒ It is also called ~~as feed-forward N/W~~ (MLP)
⇒ N/w has one or more hidden layers

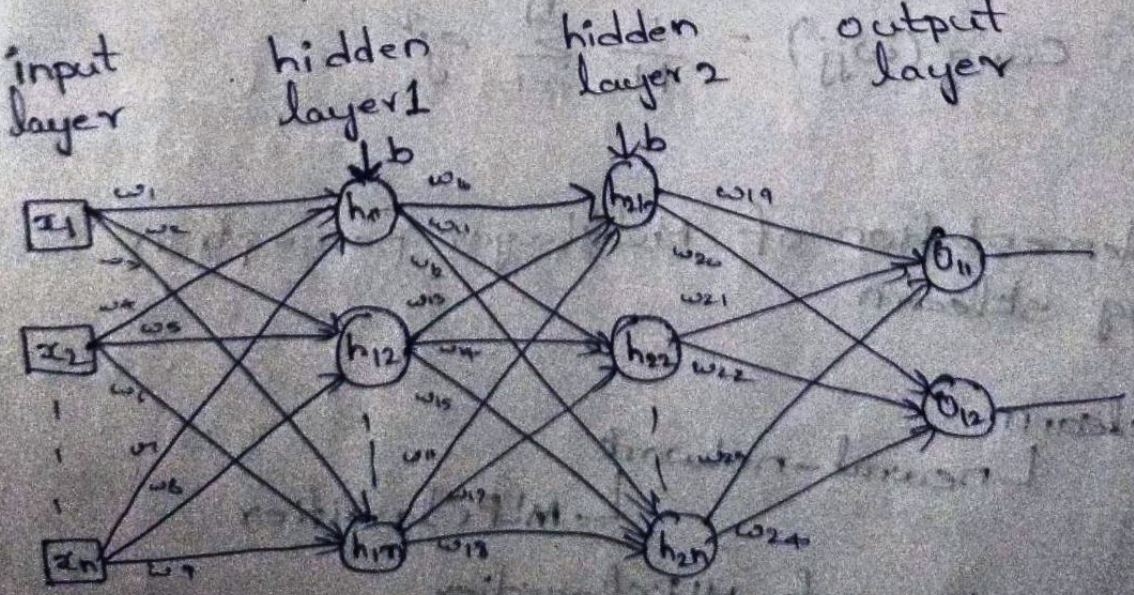
Advantages

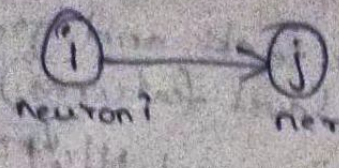
1) Complex tasks can be solved (eg: XOR)
2) Transformations are performed

Importance of Hidden Layers

Hidden Layers Let MLP Applies
1) Non-linear Transformations so that the data is linear separable
2) Can change dimentionality of data
3) Extract low level features
4) Compute gradient

Structure:





→ w_{ji} → weight of the i/p from neuron i to j

→ y_j → actual output of neuron j

→ d_j → target output associated with neu j.

→ e_j → error value for neuron j

→ $\varphi_j(.)$ → activation function for neuron j

→ v_j → induced local field value for neuron j

$$y_j = \varphi_j(v_j)$$

$$v_j = \sum_{i=1} (w_{ji} * x + b)$$

$$e_j = d_j - y_j$$

$$C(w_{ji}) = \frac{1}{2} \sum_{i=1}^{n} e_j^2$$

$$C_{avg}(w_{jj}) = \frac{1}{N} \sum_{i=1}^{n} e_j^2$$

Implementation of Multilayer perceptron using sklearn

sklearn
└ neural_network
└ MLPClassifier

→ Constructor of MLPClassifier

parameters:

i) 'hidden_layer_sizes = tupule
= (10,15,3)
70% or 90% in

- hidden layer 1 = no of neurons
  = 70-% of i/p features (or)
  90-%
- hidden layer 2 = no of neurons

3) Activation
   - identity
   - logistic
   - tanh
   - relu

3) Solver
   - optimization algorithm
     - lbfgs - Newtons
     - sgd - stochastic gradient descent
     - adam - variant of sgd

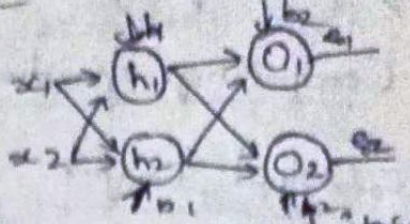
4) max_iter
   - the maximum no. of iterations
     - number (n)
     - tol (tolerence)
       stoping algorithm (1e-4)

Exp 5

```
>from sklearn import datasets
>d = datasets.load_iris()
> d.data   # to view the data values of features
> d.target  #  "      "      "    "  target
> d.feature_names
> import pandas as pd
> from sklearn.model_selection import
  train_test_split
> from sklearn.neural_network import
  MLPClassifier
> df = pd.DataFrame(d.data, columns =
        ["sepal length", "sepal width",
         "petal length", "petal width"])
> df.head()  #prints the dataframe
> df.info()  #non null values, count, dtype
> x = df[['sepal length', 'sepal width', 'petal length',
        'petal width']]
> y = d.target
> xtr, xte, ytr, yte = train_test_split(x, y,
        test_size = 0.2, random_state=0)
>mc = MLPClassifier(hidden_layer_sizes = (4,3),
        max_iter = 500,
        activation = "identity",
        solver = "sgd")
>mc  # prints the info contain it (same as above)
>d.target_names
>mc.fit(xtr, ytr)
>mc.predict(xte)
>mc.score(xte, yte)
```

Back Propagation Algorithm

To find error, we find error at every output neuron and get total error



=> To minimize the error we update free parameter (i.e, except i/p & o/p remaining which are randomly intialized)

=> If error are present classification will not be possible so we update to get linear separable line.

=> All weights should be updated

=> Reason for error
1) incorrect value of weights and bias
2)

=> The Backpropagation mainly consists two phases
1) forward pass
2) Backward pass

1) forward pass:
=> N/w fed with the i/p
=> neurouns in hidden layers will receive & process the i/p. and passes the o/p to next layer
=> N/w will produce actual o/p at the end

2) Backward pass:
=> It is also called error correction pass.
=> firstly, the total error value is computed
=> When there is an error, it uses gradient descent to minimize error value.
=> The gradient descent computes the gradient inorder to determine slope ie, to find local minimum error.

$$w(n+1) = w(n) - \eta g(n)$$

⇒ finding g(n) is easy incase of 1 i/p

Computing gradient at

Output Layer

$E_{total} = E_{o1} + E_{o2} + \dots E_{on}$

eg: $E_{total} = E_{o1} + E_{o2}$

$E_{o1} = \frac{1}{2} \sum_{i=1}^{n} e_i^2$ n = size of traing set

$E_{o2} = \frac{1}{2} \sum_{i=1}^{m} (d_{o1} - y_{o1})^2$

$E_{o2} = \frac{1}{2} \sum_{i=1}^{n} e_i^2$

$= \frac{1}{2} \sum_{i=1}^{m} (d_{o2} - y_{o2})^2$

20/9/22

Output layer neuron j

$e_j = d_j - y_j \rightarrow (1)$

$y_j = \psi(v_j)$

$y_j = \text{sigmoid}(v_j) = \frac{1}{1+e^{-v_j}} \rightarrow (2)$

$v_j = \sum_{i=0}^{m} x_i w_{ji} + b_0 \rightarrow (3)$

$E(w) = \frac{1}{2} \sum_{i=1}^{m} e_i^2 \rightarrow (4)$

$= \frac{1}{2} \sum_{i=1}^{m} (d_i - y_i)^2$

Update Rule:

Gradient descent

$w_{new} = w_{old} - \eta g(n)$

or

$w(n+1) = w(n) - \eta g(n)$

g(n) is gradient of the cost function c(w)

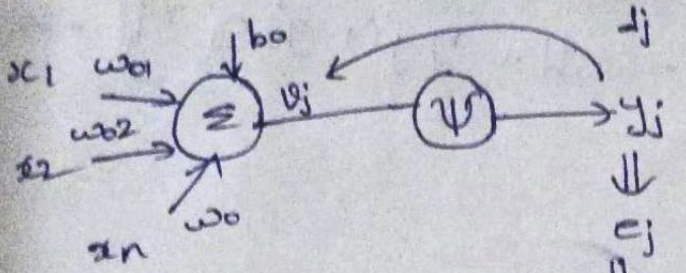
=> a neuron N/w with k neurons in output layer

$$E_{total}(w) = \sum_{i=1}^{k} E_i(w)$$

$$= E_1(w) + E_2(w) + \cdots + E_k(w) \rightarrow (5)$$

=> gradient of cost function :

$$g(n) = \frac{\partial E_{total}}{\partial w_{ji}}$$



$E_{total} \rightarrow e_j \rightarrow y_j \rightarrow v_j \xrightarrow{E_{total}} w_{ji}$ · ∃(-function jobs)

$$\frac{\partial E_{total}}{\partial w_{ji}} = \frac{\partial E_{total}}{\partial e_j} \times \frac{\partial e_j}{\partial y_j} \times \frac{\partial y_j}{\partial v_j} \times \frac{\partial v_j}{\partial w_{ji}}$$

(∵ By using chain rule)

$$\frac{\partial E_{total}}{\partial e_j} = \frac{1}{2} \times 2 \times e_j = e_j$$

$$\frac{\partial}{\partial e}\left(\frac{1}{2}\sum_{i=1}^{M} e_j^2\right) = \frac{1}{2} \times 2e_j$$

$$\frac{\partial e_j}{\partial y_j} = \frac{\partial}{\partial y_j}[d_j - y_i] = -1$$

$$\frac{\partial y_j}{\partial v_j} = \frac{\partial}{\partial v_j}\left[\frac{1}{1+e^{-v_j}}\right] = \frac{\partial}{\partial v_j}(\text{sigmoid}(v_j))$$

$$= \frac{\partial}{\partial v_j}[1+e^{-v_j}]^{-1} = \frac{\partial}{\partial v_j}(\text{sigmoid}(v_j) - \text{sigmoid}^2(v_j))$$

$$= -1[0 + e^{-v_j}] \times e^{-v_j}(-1) \Rightarrow y_j(1-y_j)$$

$$= -1 \times e^{-v_j}(-1)$$

$$= e^{-v_j}$$

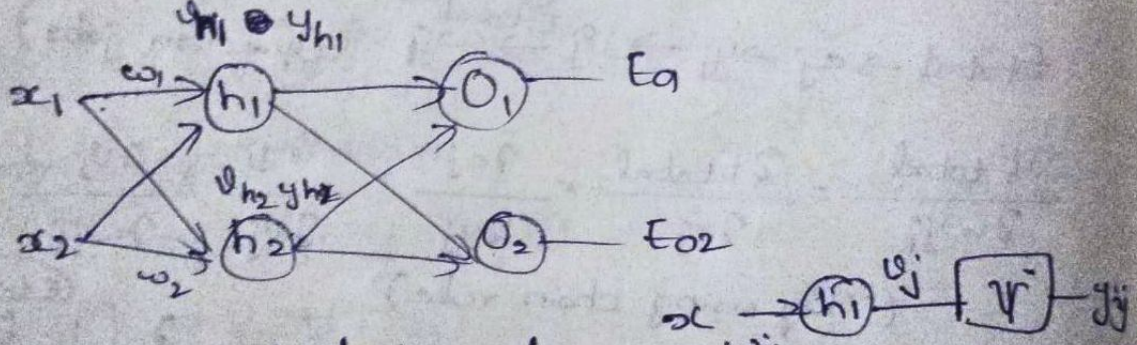
$$\frac{\partial v_j}{\partial w_{ij}} = \cdot \infty x_i$$

$$y_j = \sum_{i=1}^{m} w_{ji} x_i + b$$
$$= w_{ji} x + b_0$$
$$= 1 \times x_i + 0$$

$$\frac{\partial E_{total}}{\partial w_{ji}} = e_{ji} \times -1 \times y_j(1-y_j) \times x_i$$
$$= -e_{ji} \times y_i(1-y_i) \times x_i$$
$$= -y_i(1-y_i) e_j x_i$$

$$w_{new} = w_{old} - \eta g(n) = w_{old} + \eta y(1-y) e_j x_j \rightarrow *$$

Hidden Layer:



By using gradient descent:

$$w_{new} = w_{old} - \eta g(n)$$

$\Rightarrow g(n)$ → gradient of cost function w.r.t to weight

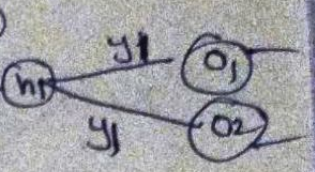
E_{total} → error caused by hidden layer neuron at output layer neuron)

$$g(n) = \frac{\partial E_{total}}{\partial w_{j1}} \rightarrow (1)$$

By applying chain rule

$$\frac{\partial E_{total}}{\partial w_{ji}} = \frac{\partial E_{total}}{\partial y_j} \times \frac{\partial y_j}{\partial v_j} \times \frac{\partial v_j}{\partial w_{ji}} \rightarrow (2)$$

$$\frac{\partial E_{total}}{\partial y_j} = \frac{\partial E_{o1}}{\partial y_j} + \frac{\partial E_{o2}}{\partial y_j} + \cdots \frac{\partial E_{oc}}{\partial y_j} \rightarrow (3)$$



$$\frac{\partial E_{total}}{\partial y_1} = \frac{\partial E_{o1}}{\partial y_{h1}} + \frac{\partial E_{o2}}{\partial y_{h1}}$$

$$\frac{\partial E_{oi}}{\partial y_j} = \frac{\partial E_{oi}}{\partial e_{oi}} * \frac{\partial e_{oi}}{\partial y_{oi}} * \frac{\partial y_{oi}}{\partial v_{oi}} * \frac{\partial v_{oi}}{\partial y_j} \rightarrow (4)$$

$$E_{oi} = \frac{1}{2} \sum_{i=1}^{M} e_{oi}^2$$

$$\frac{\partial o_i}{\partial e_{oi}} = e_{oi} \qquad \therefore e_{oi} = d_{oi} - y_{oi}$$

$$\frac{\partial e_{oi}}{\partial y_{oi}} = -1$$

$$\frac{\partial y_{oi}}{\partial v_{oi}} = y_{oi}(1 - y_{oi}) \qquad y_{oi} = V(v_{oi}) = y_{oi} \times (1 - y_{oi})$$

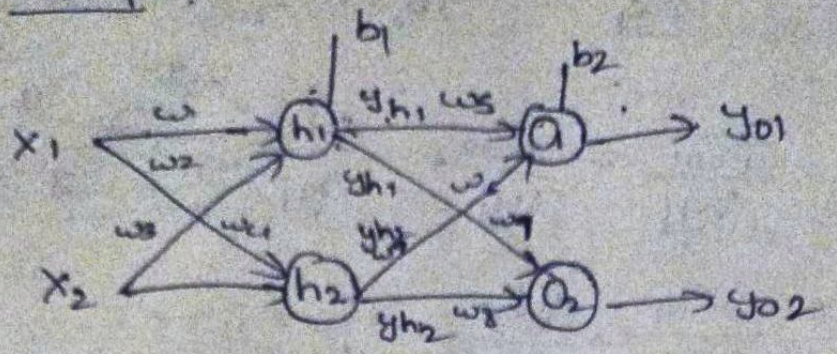
$$\frac{\partial v_{oi}}{\partial y_j} = w_{ji} \qquad v_i = \sum_{i=1}^{m} w_{ji} y_j + b_0 = w_{ji} + 0$$

$$\frac{\partial E_{oi}}{\partial y_j} = e_{oi} \times -1 \times y_{oi}(1 - y_{oi}) \times w_{ji}$$

$$= -y_{oi}(1 - y_{oi}) e_{oi} w_{ji}$$

$$w_{new} = w_{old} + \eta\, y_{oi}(1 - y_{oi}) e_{oi} w_{ji}$$

Example:



O_2

$$v_{o2} = y_{h1} + w_7 + y_{h2} \times w_8 + b_2$$

$$y_{o2} = \text{sigmoid}(v_{o2}) = \frac{1}{1+e^{-v_{o2}}}$$

$$e_{o2} = d_{o2} - y_{o2}$$

$$E_{o2} = \frac{1}{2} \sum_{i=1}^{n} e_{o2}^2(i)$$

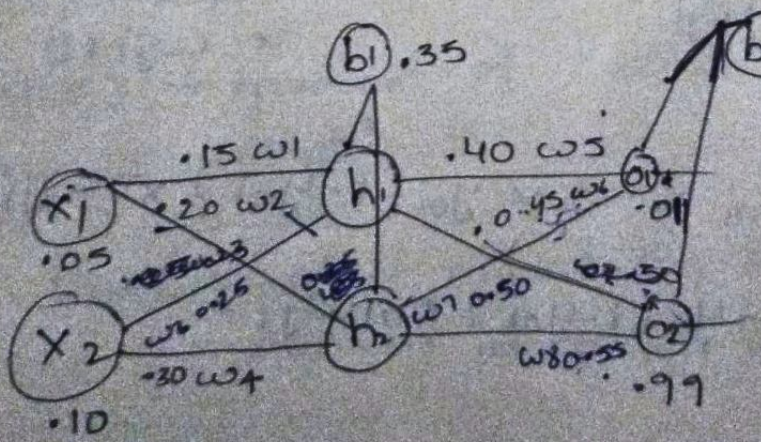
$$= \frac{1}{2} \sum_{i=1}^{n} (d_{o2}(i) - y_{o2}(i))^2$$

$$w_7(new) = w_7(old) - \eta g(n)$$

$$g(n) = -e_{o2} y_{o2} (1 - y_{o2}) y_{h1}$$

$$w_7(new) = w_7(old) + \eta e_{o2} y_{o2} (1 - y_{o2}) y_n$$

Eg



Iteration 1:

Forward pass:

At h_1:

$$v_{h_1} = (w_1 x_1 + w_3 x_2) + b_1$$
$$= (0.15 \times 0.05) + (0.25 \times 0.10) + 0.35$$
$$= 0.0075 + 0.02 + 0.35$$
$$= 0.3775$$

$$v_{h_2} = w_3 x_1 + w_4 x_2 + b$$

$$y_{h_1} = \psi(v_{h_1})$$

$$= \frac{1}{1+e^{-v_{h_1}}} = \frac{1}{1+e^{-0.3775}} = 0.5932$$

$$v_{h_2} = w_3 x_1 + w_4 x_2 + b_1$$
$$= 0.25 \times 0.05 + 0.30 \times 0.10 + 0.35$$
$$= 0.0125 + 0.03 + 0.35 = 0.505$$

$$y_{h_2} = \psi(v_{h_2})$$
$$= \frac{1}{1+e^{-v_{h_2}}} = \frac{1}{1+e^{-(0.505)}} = 0.6236$$

At h_1:

$$v_{h_1} = w_1 x_1 + w_3 x_2 + b_1$$
$$= 0.15 \times 0.05 + 0.25 \times 0.10 + 0.35$$
$$= 0.3825$$

$$y_{h_1} = \psi(v_{h_1}) = \frac{1}{1+e^{-v_{h_1}}} = \frac{1}{1+e^{-0.3825}} = 0.5945$$

$$v_{h_2} = w_2 x_1 + w_4 x_2 + b_1$$
$$= 0.20 \times 0.05 + 0.30 \times 0.10 + 0.35$$
$$= 0.39$$

$$y_{h_2} = \psi(v_{h_2}) = \frac{1}{1+e^{-v_{h_2}}} = \frac{1}{1+e^{-0.39}} = 0.5962$$

At O_1

$v_{o_1} = w_5 \times h_1 + w_7 \times h_2 + b_2$

$= 0.40 \times 0.5945 + 0.50 \times 0.5962 + 0.60$

$= 1.1359$

$y_{o_1} = \psi(v_{o_1})$

$= \frac{1}{1 + e^{-1.1359}} = 0.7569$

At O_2

$v_{o_2} = w_6 h_1 + w_8 h_2 + b_2$

$= 0.45 \times 0.5945 + 0.55 \times 0.5962 + 0.60$

$= 1.1954$

$y_{o_2} = \psi(v_{o_2})$

$= \frac{1}{1 + e^{-1.1954}} = 0.7677$

	d	y
O_1	0.01	0.7569
O_2	0.99	0.7677

error has occured so, we move to next pass i.e, Backward pass.

Backward pass:

By using gradient decent we are going to find new weights.

$w_5(new) = w_5(old) - \eta g(n)$

where,

$$g(n) = \frac{\partial E_{total}}{\partial w_5}$$

$$\frac{\partial E_{total}}{\partial w_5} = \frac{\partial E_{total}}{\partial e_{o1}} \times \frac{\partial e_{o1}}{\partial y_{o1}} \times \frac{\partial y_{o1}}{\partial v_{o1}} \times \frac{\partial v_{o1}}{\partial w_5}$$

$E_{total} = E_{o1} + E_{o2}$

$E_{o1} = \frac{1}{2}(d_{o1} - y_{o1})^2$

$= \frac{1}{2}(0.01 - 0.7569)^2$

$= 0.2789$

$E_{o2} = \frac{1}{2}(d_{o2} - y_{o2})^2$

$= \frac{1}{2}(0.99 - 0.7677)^2$

$= 0.0247$

$E_{total} = 0.2789 + 0.0247 = 0.3036$

$\frac{\partial E_{total}}{\partial e_{o1}} = e_{o1} = d_{o1} - y_{o1} = 0.01 - 0.7569 = -0.7469$ $e_{o1} = d_{o1} - y_{o1}$

$\frac{\partial e_{o1}}{\partial y_{o1}} = -1$

$\frac{\partial y_{o1}}{\partial v_{o1}} = y_{o1}(1 - y_{o1}) = 0.7569(1 - 0.7569) = 0.1840$

$\frac{\partial v_{o1}}{\partial w_5} = y_{h1} = 0.5945$

$\frac{\partial E_{total}}{\partial w_5} = (-0.7469) \times (-1) \times 0.1840 \times 0.5945$

$= 0.0817$

$$w_5(new) = w_5(old) - \eta g(n)$$
$$= 0.40 - (0.5)(0.0817)$$
$$= 0.35915$$

$$w_6(new) = w_6(old) - \eta g(n)$$

$$g(n) = \frac{\partial E_{total}}{\partial w_6}$$

$$\frac{\partial E_{total}}{\partial w_6} = \frac{\partial E_{total}}{\partial e_{O2}} \times \frac{\partial e_{O2}}{\partial y_{O2}} \times \frac{\partial y_{O2}}{\partial v_{O2}} \times \frac{\partial v_{O2}}{\partial w_6}$$

$$\frac{\partial E_{total}}{e_{O2}} = e_{O2} = (d_{O2} - y_{O2}) = (0.99 - 0.7677) = 0.2223$$

$$\frac{\partial e_{O2}}{\partial y_{O2}} = -1$$

$$\frac{\partial y_{O2}}{\partial v_{O2}} = y_{O2}(1 - y_{O2}) = (0.7677)(1 - 0.7677)$$
$$= 0.1783$$

$$\frac{\partial v_{O2}}{\partial w_6} = y_{h_2} = 0.5945$$

$$\frac{\partial E_{total}}{\partial w_6} = -0.2223 \times 0.1783 \times 0.5945$$
$$= -0.0236$$

$$w_6(new) = w_6(old) - \eta g(n)$$
$$= 0.45 - (0.5)(-0.0236)$$
$$= 0.4618$$

$w_7(new) = w_7(old) - \eta g(n)$

$g(n) = \frac{\partial E total}{\partial w_7}$

$\frac{\partial E total}{\partial w_7} = \frac{\partial E total}{\partial o_{01}} \times \frac{\partial o_{01}}{\partial y_{01}} \times \frac{\partial y_{01}}{\partial v_{01}} \times \frac{\partial v_{01}}{\partial w_7}$

$\frac{\partial E total}{\partial o_{01}} = e_{01} = (d_{01} - y_{01}) = -0.7469$

$\frac{\partial o_{01}}{\partial y_{01}} = -1$

$\frac{\partial y_{01}}{\partial v_{01}} = y_{01}(1 - y_{01}) = 0.1840$

$\frac{\partial v_{01}}{\partial w_7} = y_{h2} = 0.5962$

$\frac{\partial E total}{\partial w_7} = +0.7469 \times 0.1840 \times 0.5962$

$g(n) = 0.0819$

$w_7(new) = 0.50 - (0.5)(0.0819)$

$= 0.45905$

$w_8(new) = w_8(old) + \eta g(n)$

$g(n) = \frac{\partial E total}{\partial w_8}$

$\frac{\partial E total}{\partial w_8} = \frac{\partial E total}{\partial o_{02}} \times \frac{\partial o_{02}}{\partial y_{02}} \times \frac{\partial y_{02}}{\partial v_{02}} \times \frac{\partial v_{02}}{\partial w_8}$

$\frac{\partial E total}{\partial o_{02}} = e_{02} = (d_{02} - y_{02}) = 0.2225$

$\frac{\partial o_{02}}{\partial y_{02}} = -1$, $\frac{\partial y_{02}}{\partial v_{01}} = y_{02}(1 - y_{02}) = 0.1783$

$\frac{\partial v_{02}}{\partial w_8} = y_{h2} = 0.5962$

$$\frac{\partial E_{total}}{\partial w_8} = -0.2223 \times 0.1783 \times 0.5962$$

$$= -0.0236$$

$$w_8(new) = w_8(old) - \eta g(n)$$

$$= 0.55 - (0.5)(-0.0236)$$

$$= 0.5618$$

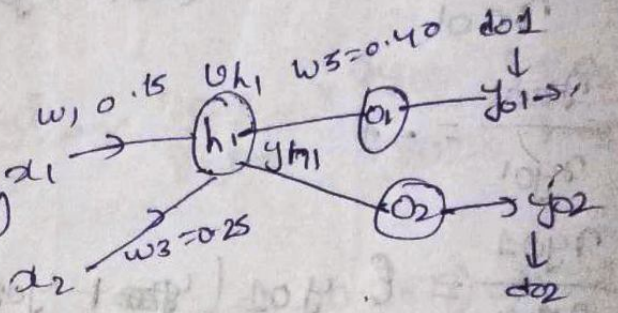
At hidden layer

h1

updation of w_1

$$w_1(new) = w_1(old) - \eta g(n)$$

$$g(n) = \frac{\partial E_{total}}{\partial w_1}$$



$$\frac{\partial E_{total}}{\partial w_1} = \frac{\partial E_{total}}{\partial y_{h1}} \times \frac{\partial y_{h1}}{\partial v_{h1}} \times \frac{\partial v_{h1}}{\partial w_1}$$

$$\frac{\partial E_{total}}{\partial y_{h1}} = \frac{\partial E_{o1}}{\partial y_{h1}} + \frac{\partial E_{o2}}{\partial y_{h1}}$$

$$\frac{\partial E_{o1}}{\partial y_{h1}} = \frac{\partial E_{o1}}{\partial e_{o1}} \times \frac{\partial e_{o1}}{\partial y_{o1}} \times \frac{\partial y_{o1}}{\partial v_{o1}} \times \frac{\partial v_{o1}}{\partial y_{h1}} \rightarrow (4)$$

$$\frac{\partial E_{o2}}{\partial y_{h1}} = \frac{\partial E_{o2}}{\partial e_{o2}} \times \frac{\partial e_{o2}}{\partial y_{o2}} \times \frac{\partial y_{o2}}{\partial v_{o2}} \times \frac{\partial v_{o2}}{y_{h1}} \rightarrow (5)$$

$w_1 \rightarrow v_{h1} \rightarrow y_{h1} \rightarrow E_{total} \rightarrow (1)$

$y_{h1} \rightarrow v_{o1} \rightarrow y_{o1} \rightarrow e_{o1} \rightarrow E_{o1} \rightarrow (2)$

$y_{h1} \rightarrow v_{o2} \rightarrow y_{o2} \rightarrow e_{o2} \rightarrow E_{o2} \rightarrow (3)$

* $E_{o1} = \frac{1}{2} \sum_{i=1}^{n} e_{o1}^2$

$\frac{\partial E_{o1}}{\partial o_{o1}} = e_{o1}$

$e_{o1} = d_{o1} - y_{o1}$

$\frac{\partial e_{o1}}{\partial y_{o1}} = -1$

From eq ④

$\frac{\partial E_{o1}}{\partial y_{h1}} = e_{o1} \times -1 \times y_{o1} \times (1 - y_{o1}) \times \omega_5$

$= -e_{o1} \times y_{o1} (1 - y_{o1}) \times \omega_5$

From eq ⑤

$\frac{\partial E_{o2}}{\partial y_{h1}} = e_{o2} \times -1 \times y_{o2} (1 - y_{o2}) \times \omega_6$

$= -e_{o2} \times y_{o2} (1 - y_{o2}) \times \omega_6$

~~$= -e \times (e_{o2} - y_{o2}) \times y_{o1} \times (1 -$~~

$\frac{\partial E_{o1}}{\partial y_{h1}} = -(0.01 - 0.7569) \times (0.7569) \times (1 - 0.7569) \times 0.46$

$= 0.054$

$\frac{\partial E_{o}}{\partial y_{h2}} = -(0.99 - 0.7676) \times (0.7676) \times (1 - 0.7676) \times 0.5$

$= -0.0178$

$\frac{\partial E_{total}}{\partial y_{h1}} = \frac{\partial E_{o1}}{\partial y_{h1}} + \frac{\partial E_{o2}}{\partial y_{h1}} = 0.0549 - 0.0178 = 0.0371$

$\frac{\partial E_{total}}{\partial \omega_1} = \frac{\partial E_{total}}{\partial y_{h1}} \times \frac{\partial y_{h1}}{\partial v_{h1}} \times \frac{\partial v_{h1}}{\partial \omega}$

$= 0.037 \times y_{h1} \times (1 - y_{h1}) \times \alpha_1$

$= 0.037 \times 0.594 \times (1 - 0.594) \times 0.05$

$= \times 0.000044465 \quad (0.000103)$

$g(n) = 0.000044465 + \quad (0.000103)$

$w_1(new) = w_1(old) - \eta g(n)$

$= 0.15 - 0.5 \times (0.000103)$

$= 0.1497$

At hidden Layer 1

$w_3(new) = w_3(old) - \eta g(n)$

$g(n) = \frac{\partial E total}{\partial w_3}$

$\frac{\partial E total}{\partial w_3} = \frac{\partial E total}{\partial y_{h1}} \times \frac{\partial y_{h1}}{\partial v_{h1}} \times \frac{\partial v_{h1}}{\partial w_3}$

$\frac{\partial E total}{\partial y_{h1}} = \frac{\partial E_{o1}}{\partial y_{h1}} + \frac{\partial E_{o2}}{\partial y_{h1}}$

$\frac{\partial E_{o1}}{\partial y_{h1}} = \frac{\partial E_{o1}}{\partial e_{o1}} \times \frac{\partial e_{o1}}{\partial y_{o1}} \times \frac{\partial y_{o1}}{\partial v_{o1}} \times \frac{\partial v_{o1}}{\partial y_{h1}}$

$= e_{o1} \times -1 \times y_o \times (1 - y_{o1}) \times w_5$

$$\frac{\partial E_{o2}}{\partial h_1} = \frac{\partial E_{o2}}{\partial e_{o2}} \times \frac{\partial e_{o2}}{\partial y_{o2}} \times \frac{\partial y_{o2}}{\partial v_{o1}} \times \frac{\partial v_{o1}}{\partial y_{h2}}$$

$$= e_{o2} \times -1 \times y_{o2} \times (1 - y_{o2}) \times w_n$$

$$\frac{\partial E_{total}}{\partial w_3} = \frac{\partial E_{total}}{\partial y_{h1}} \times y_{h1} \times (1 - y_{h1}) \times x_2$$

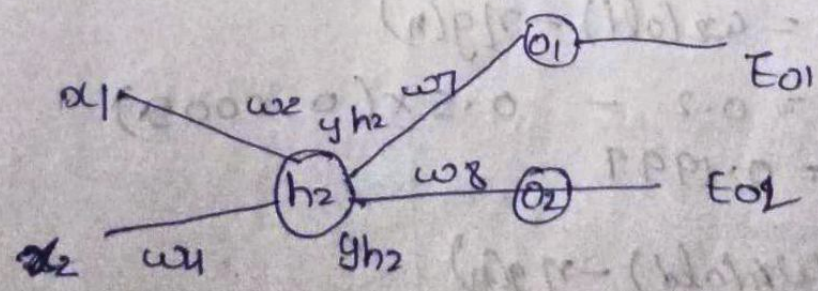
$$= 0.037 \times 0.594 \times (1 - 0.594) \times 0.101$$

$$= 0.00089$$

$$w_3(new) = w_3(old) - \eta g(n)$$

$$= 0.25 - 0.5 \times 0.00089$$

$$= 0.249$$



updation for w_2

$$w_2(new) = w_2(old) - \eta g(n)$$

$$g(n) = \frac{\partial E_{total}}{\partial w_2}$$

$$\frac{\partial E_{total}}{\partial \omega_2} = \frac{\partial E_{total}}{\partial y_{h2}} \times \frac{\partial y_{h2}}{\partial v_{h2}} \times \frac{\partial v_{h2}}{\partial \omega_2} \rightarrow ①$$

$$\frac{\partial E_{total}}{\partial y_{h2}} = \frac{\partial E_{o1}}{\partial y_{h2}} + \frac{\partial E_{o2}}{\partial y_{h2}} = 0.008 - 0.02177 = 0.047$$

$$\frac{\partial E_{o1}}{\partial y_{h2}} = \frac{\partial E_{o1}}{\partial v_{o1}} \times \frac{\partial e_{o1}}{\partial y_{o1}} \times \frac{\partial y_{o1}}{\partial v_{o1}} \times \frac{\partial v_{o1}}{\partial y_{h2}}$$

$$= e_{o1} \times -1 \times y_{o1}(1-y_{o1}) \times \omega_7$$

$$= -(d_{o1} - y_{o1}) \times y_{o1}(1-y_{o1}) \times \omega_7$$

$$= 0.068$$

$$\frac{\partial E_{o2}}{\partial y_{h2}} = \frac{\partial E_{o2}}{\partial e_{o2}} \times \frac{\partial e_{o2}}{\partial y_{o2}} \times \frac{\partial y_{o2}}{\partial v_{o2}} \times \frac{\partial v_{o2}}{\partial y_{h2}}$$

$$= e_{o2} \times -1 \times y_{o2}(1-y_{o2}) \times \omega_8$$

$$= -(d_{o2} - y_{o2}) \times y_{o2}(1-y_{o2}) \times \omega_8$$

$$= -0.02177$$

From eq①

$$\frac{\partial E_{total}}{\partial \omega_2} = 0.047 \times y_{h2} \times (1-y_{h2}) \times x_1$$

$$= 0.047 \times 0.596(1-0.596) \times 0.05$$

$$= 0.05$$

$$\omega_2(new) = \omega_2(old) - \eta g(n)$$

$$= 0.2 - 0.5 \times (0.0005)$$

$$= 0.1999$$

$$\omega_4(new) = \omega_4(old) - \eta g(n)$$

$$g(n) = \frac{\partial E_{total}}{\partial \omega_4}$$

$$\frac{\partial E_{total}}{\partial \omega_4} = \frac{\partial E_{total}}{\partial y_{h2}} \times \frac{\partial y_{h2}}{\partial v_{h2}} \times \frac{\partial v_{h2}}{\partial \omega_4} \rightarrow ①$$

$$\frac{\partial E_{total}}{\partial y_{h2}} = \frac{\partial E_{o1}}{\partial y_{h2}} + \frac{\partial E_{o2}}{\partial y_{h2}}$$

$$\frac{\partial E_{total}}{\partial \omega_4} = 0.047 \times y_{h_2} \times (1 - y_{h_2}) \times x_2$$

$$= 0.047 \times 0.596 \times (1 - 0.596) \times 0.10$$

$$= 0.00011$$

$$\omega_4(new) = \omega_4(old) - \eta g(h)$$

$$= 0.30 - 0.5 \times 0.00011$$

$$= 0.299$$

After iteration 1

$x_1 = 0.05$

$x_2 = 0.10$

$\omega_1 = 0.149$

$\omega_2 = 0.19$

$\omega_3 = 0.249$

$\omega_4 = 0.299$

$\omega_5 = 0.359$

$\omega_6 = 0.5617$

$\omega_7 = 0.459$

$\omega_8 = 0.5617$

Iteration 2

Forward Pass

$$\theta_{h_1} = x_1 \times \omega_1 + x_2 \times \omega_2 + b$$

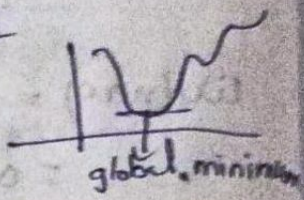
Practical & design issues of Back Propagation Algorithm:

① Rate of learning

⇒ Specifies a step size taken by learning algorithm

⇒ takes a value in between 0-1

$0 \leq \eta \leq 1$



[η is very small
↳ slowly converses the global minimum

$$\omega(new) = \omega(old) - \eta g(n)$$

↓ learning rate

⇒ It increase no. of iterations requried for converging

⇒ algorithm can trapped into local minimum point.

η very large!

⇒ Algorithm can over shoot global minimum point can oscillate

⇒ Avoids getting trap into local minima point solution.

Momentum:

⇒ $\Delta w_{ji}(n) = \alpha \Delta w_{ji}(n-1) + \Delta w_{ji}(n)$

⇒ Provides solution to problems associated with learning rate

⇒ Adds small amount of update from previous iteration controlled by momentum term (α)

Sequential or Batch mode:

epoch

↓

presentation of complete set of examples or samples.

=> Also called online mode

=> Update for weight vector computed sequentially by considering only one sample at a time

=> It requires less memory space as it slowly converges forward global minima point.

Drawback:

=> It is theoritically difficult to determine stopping criteria.

11/10/22

Design Issues in Back propagation

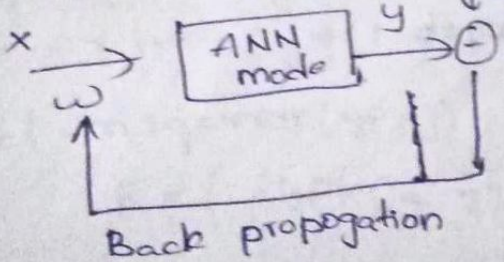
Algorithm

① Learning rate (η): (momentum)

=> Momentum does the update value of previous weight to present update weight.

$\Delta w(n) = \Delta w(n) + \alpha \Delta w(n-1)$

momentum term



② Stopping Criteria:

=> Euclidian Norm value of d & y are minimum

=> When model produces accepted level error that is accepted.

③ New sample not there in training set:

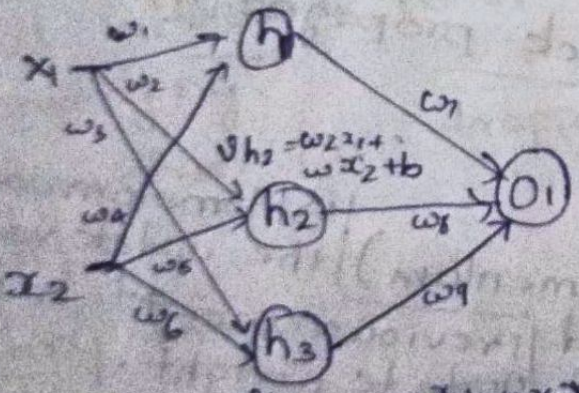
=> It goes through Generalization process. It tries to map with the already samples & it comes to conclusions & it is given as o/p.

model → Training

Implementation of Back propagation Alg

1) Initialize parameters
   1. Weights
   2. bias
   3. learning rate
2) Construct neural networks
   → no. of inputs
   → no. of hidden layers
   → no. of neurons
   → no of neurons in output layer

$$\vartheta_{h_1} = \omega_1 x_1 + \omega_4 x_2 + b$$



$$\vartheta_{h_3} = \omega_3 x_1 + \omega_6 x_2 + b$$

$$\begin{bmatrix} \omega_1 & \omega_4 \\ \omega_2 & \omega_5 \\ \omega_6 & \omega_6 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

Initializing Parameters

```
import numpy as np
inputsize = 2
hiddensize = 3
outputsize = 1
lr = 0.1
w1 = np.random.randn(inputsize, hiddensize) * 0.01
w2 = np.random.randn(hiddensize, outputsize) * 0.01
b1 = np.random.randn(hiddensize, 1) * 0.01
b2 = np.random.randn(outputsize, 1) * 0.01
```

defining Activation function:

```
def sigm(v):
    y = 1/(1+np.exp(-v))
    return y
def derivate(y):
    return (y*(1-y))
def msqerror(yp,y):
    E = ((yp-y)**2).sum()/2
    return E
```

define a dataset

```
x = np.array([[7,2],[2,6],[9,1]])
y = np.array([[80],[80],[95]])
# normalize
X = X/np.amax(X, axis=0)
Y = Y/100
```

```
# create a neural network and train
def train(x,y):
    global w1, w2, b1, b2, lr
# forward phase
vh = np.dot(x, w1)+bi  or (x @ w1) + b1
yh = sigm(vh)
v0 = np.dot(yh, w2)+b2 or (yh @ w2) + b2
y0 = sigm(v0)

#Backpropagation

    e1 = y0 - y
    delta_0 = e1 * y0 * (1-y0)
    e2 = delta_0 @ w2.T
    delta_h = e2 * yh * (1-yh)
    g0 = yh.T @ delta_0
    gh = X.T @ delta_h

w1 = w1 - η * gh
w2 = w2 - η * g0
b1 = b1 - η * delta_h
b2 = b2 - η * delta_0
for i in range(2000):
    train(x,y)
def forwardtest(x,y):
    global w1, w2, b1, b2
```