UNIT - IV

11. MultiLayer perceptron

MultiLayer perceptron also called feed forward Network

- this is a N/w that consists of one or more hidden Layers
- so that it can perform complex tasks
- we can also perform tranformation (e.x XOR gate)
- (Extention) Existence of hidden layers we can perform complex tasks

ex- Hand written digits recognition

- This one called as feed forward N/w because I/p signal is forwarded from to next layer then next layer i.e only 1 direction the data is forwarded.

horizontal lines - 1st level hidden layer
V.L → 2nd level
Curve L → 3rd Levels
of an Image

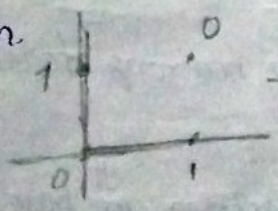
Hence it is called feed forward Neural N/w.

- Here it has one or more hidden layers.
- In the sklearn we have MLP classifier to create MultiLayer Perceptron.

Advantages (of MLP over Single Layer perceptron)

- The MLP can perform complex tasks.
- Perform transformation.

e.g: XOR gate



→ when it hyper plane it can only classifies

Single layer P fails this task even though the classification is binary

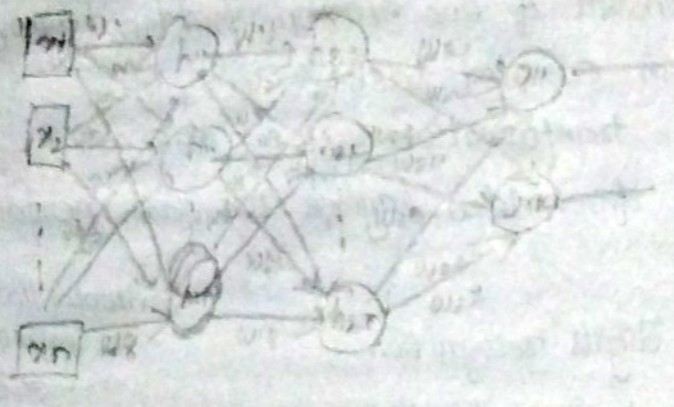
i.e 3D

Importance of Hidden Layers

- Having hidden layers let MLP
  ① Applies Non-linear transformations so that the data is linear separable
  ② can change dimensionality of data
  ③ Extract low level features
  ④ Compute gradient (used in error optimization)

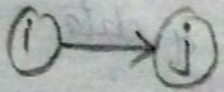
Structure of Multi Layer perceptron

Input layer | hidden layer 1 | hidden Layer 2 | .... | hidden layer N | olp Layer



- Every layer is well connected.
- The ilp layer passes data to hidden layer 1
- The data from hidden layer 1 is passed to higher hidden layers
- The hidden layers process the model to get the olp
- Each & every ilp is weighted
- No of features = No. of ilps
- If we have 1 olp neuron — only binary classification
  2 olp or more " — Multiclass "
- If we have the no. of olp neurons depends on the no. of features do you have.
- In sklearn the MLP classifier. It have 100 hidden layer neurons by defaults
- We need to compute the error at the olp layer

* Conventions



- w_{ij} - weight of the ilp from neuron i to j
- y_i - actual olp of neuron j
- d_j - target olp of neuron j (0) associated with neuron j

e_j - error value for neuron j
$\varphi_j(\ )$ - Activation fun^n for jth neuron.
v_j - induced local field value for neuron j.
for Multilayer perceptron

$$y_j = \varphi_j(v_j)$$
$$v_j = \sum_{i=1}^{n} (w_{ji} * x + b)$$
$$e_j = d_j - y_j$$
$$\text{cost fun}^n \; c(w_{ji}) = \tfrac{1}{2} \sum_{i=1}^{n} e_j^2$$
$$\text{Carg}(w_{ji}) = \frac{1}{N} \sum_{i=1}^{n} e_j^2$$

N = total no of samples
n = the i/p that neuron taken

16/9/22 **Implementing MLP**

Sklearn → neural_network → MLP classifier

→ we can use constructor of MLPclassifier.

1st parameter — (100,) → 100 neurons

1 hidden_layer_sizes = tuple
= (10, 5, 3) 70% (80) 90% used
↓ ↓ ↓
H_1 H_2 H_3
(Hidden layer with 10 neurons)

→ The 1st hidden layer – no. of neurons = 40% of inputs (or 90% of i/p features.

→ 2nd layer hidden ⇒ 2nd parameter

activation → identity
→ logistic
→ tanh
→ relu

3rd parameter – solver

→ we can specify optimization algorithm by using solver

→ lbfgs – Neurons
sgd – stochastic gradient descent
adam – variant of sgd.

4th parameter – max_iter

→ It specifies the max no of iterations
→ default is has 200 iterations
→ tol → tolerance, tol = $1e^{-4}$

5th parameter → learning rate 0.01.

(diabetes)

Program

```
from sklearn import datasets
d = datasets.load_iris()
d.data  # i/p features
```

o/p → array [[5.1, 3.5, 0.2],

d.target # target features olp :- array([0,0,0,0---])

d.feature_names

['Sepal length (cm)',
'Sepal width (cm)'
Petal length (cm)'
" width (cm)']

import pandas as pd # to create a dataframe

from sklearn.model_selection import train_test_split

from sklearn.neural_network import MLPClassifier

df ← pd.DataFrame(d.data, columns = ['sepal length', 'S.w', 'P.l', 'P.w'])

df.head()

df.info() # non-null values gives olp :- 150 entries with 4 columns no → nan values

df.columns # names of columns

x = df[['S.l', 'S.w', 'P.l', 'P.w']]

y = d.target # target labels

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

d.target_names # olp :- array(['setosa', '=', '='])

mc.fit(xtr, ytr)

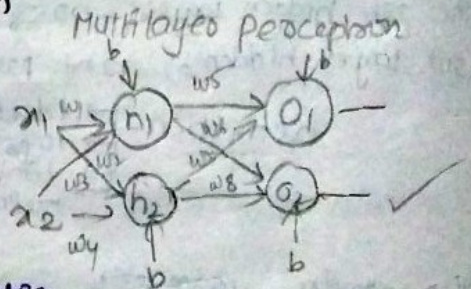
mc.predict(xte) olp :- array([2,1,0,2,---])

14/9/22 mc.score(xte, yte) # 0.9666 (linearly separable)

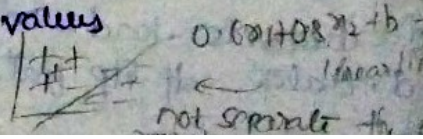
logistic - 0.2

tanh - 0.56

Back propagation Algorithm



- first, we need to compute the total error value which we call as E_{total}.
- the actual olp is produced at olp layer.
- we need to minimize the total error value by updating the free parameters (w, b).
- when we initialize the wt & bias with random values we may get the line that not separate the data. so we need to update until error is reduced
- we need to update all the weight values but all is not done possible



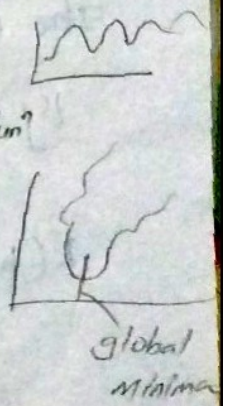
at same time.
- first we need to update the w/ps associated with o/p layer
- then the hidden layer w/p's will be updated.
- one of the reason for getting error is erroneous values
- if we finish the first one hidden layer then we need to move through another hidden layers.
- Back propagation alg make use of gradient descent Algorithm.
- It consists of two phase
  1) forward pass
  2) Backward pass.

forward Pass.

→ N/w fed with the w/p
→ neurons in (different) hidden layers process the w/p and forwards them to the next layers.

Backward pass

→ The total error value is computed
→ when there is a error
  - uses gradient descent to minimize the error values.
    ↳ It updates the wt, bias by computing the gradient of cost fun?
    $\frac{1}{2}\sum_{i=1}^{n} e_i^2$ → convex fun?
    - when we declare an eq? → we will get slope
    ↳ It will in reverse direction slope
    ↳ It will move to global minimum in reverse direction of slope.



- $w(n+1) = w(n) - \eta g(n)$
  ↳ Back propagation also make use of this simple rule.

But will make use of different procedures for w/p & different layers.

- Let us see how Back propagation Alg will produce the eqn

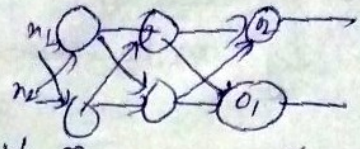
Computing gradient at

O/P Layer

$E_{total} = E_{O1} + E_{O2} + \dots + E_{On}$

Error (for one eg)

$E_{total} = E_{O1} + E_{O2}$



E_{O1} = cost fun^n at i/p one

$E_{O1} = \frac{1}{2}\sum_{i=1}^{m} e_i^2 = 1/2 \sum_{i=1}^{m} (d_{O1} - y_{O1})^2$ m = size of training set

$E_{O2} = \frac{1}{2}\sum_{i=1}^{m} e_i^2 = \frac{1}{2}\sum_{i=1}^{n} (d_{O2} - y_{O2})^2$

then

$E_{total} = \frac{1}{2}\sum_{i=1}^{n} (d_{O1} - y_{O1})^2 + \frac{1}{2}\sum_{i=1}^{n} (d_{O2} - y_{O2})^2$

20/9/22

⇒ when the o/p is not correct then we call that situation is called error situation.

⇒ O/P Layer neuron j:

$e_j = d_j - y_j$ — ①

$y_j = \varphi(v_j)$ v_j = induced local field value at jth neuron

= Sigmoid$(v_j) = \frac{1}{1+e^{-v_j}}$ — ②

The o/p y depends upon the activation fun^n we use.

$v_j = \sum_{i=1}^{m} w_{ji}\, x_i + b_0$ — ③ w_{ji} = ith weight for the jth i/p b_0 = bias

$E(w) = 1/2 \sum_{i=1}^{n} e_i^2$ — ④

$= 1/2 \sum_{i=1}^{n} (d_i - y_i)^2$

⇒ Update Rule

- as per gradient descent

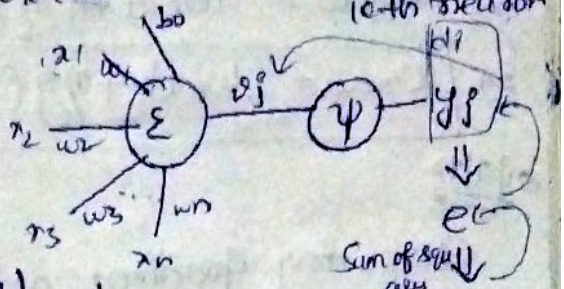
$w_{new} = w_{old} - \eta\, g(n)$ (or) $w(n+1) = w(n) - \eta\, g(n)$

g(n) is gradient of cost fun^n

→ Let us assume,
a neural N/w with k neurons in o/p layer

$$E_{total}(w) = \sum_{i=1}^{k} E_i(w) = E_1(w) + E_2(w) + \cdots + E_k(w) \quad \text{--- (5)}$$

$E_k(w)$ → error value at k-th neuron



→ Gradient of cost fun^n

$$g(n) = \frac{\partial E_{total}}{\partial w_{ji}}$$ (using derivative chain rule)

$$= \frac{\partial E}{\partial e} \times \frac{\partial e}{\partial y} \times \frac{\partial y}{\partial w_{ji}}$$

$$= \frac{\partial E_{total}}{\partial e_j} \times \frac{\partial e_j}{\partial y_i} \times \frac{\partial y_i}{\partial v_j} \times \frac{\partial v_j}{\partial w_{ji}}$$ → (8) → cost node

Etotal depend upon e
$e \rightarrow y_i \rightarrow v_j \rightarrow w_{ji}$
⇓
dependency b/w Etotal & weight

$$\frac{\partial E_{total}}{\partial e_j} = \frac{\partial}{\partial e_j}\left(E_1(w) + E_2(w) + E_3(w) + \cdots + E_k(w)\right)$$ [From (5)

$$= \frac{\partial}{\partial e_j}\left(\frac{1}{2}\sum_{j=1}^{m} e_1^2 + \frac{1}{2}\sum_{j=1}^{m} e_2^2 + \cdots + \frac{1}{2}\sum_{j=1}^{m} e_j^2 + \cdots + \frac{1}{2}\sum_{j=1}^{m} e_k^2\right)$$

$$= \frac{1}{2}\sum_{j=1}^{m} \frac{\partial}{\partial e_j}\left(e_1 + e_2 + e_3 + \cdots + 2e_j + \cdots e_k\right)$$

$$= \frac{1}{2} \times 2 \times e_j$$

$$\boxed{\frac{\partial E_{total}}{\partial e_j} = e_j}$$

$$\frac{\partial e_j}{\partial y_j} = \frac{\partial}{\partial y_i}(d_j - y_i) = -1 \quad \text{i.e.} \quad \boxed{\frac{\partial e_j}{\partial y_j} = -1}$$ [From (1)

$$\boxed{\frac{\partial y_j}{\partial v_j} = \frac{\partial}{\partial v_j}\left(\frac{e^{v_j}}{1+e^{v_j}}\right)}$$ [From (*)]

$$y_j = \frac{1}{1+e^{-v_j}} = \frac{1}{1+\frac{1}{e^{+v_j}}} = \frac{e^{v_j}}{1+e^{v_j}}$$

i.e. $\boxed{\frac{\partial y_j}{\partial v_j} = y_j(1-y_j)}$

= logistic sigmoid(x)(1 − sigmoid(x))

$= y_j(1-y_j)$

* taking derivation eq^n (2)

Consider

$$\frac{\partial v_j}{\partial w_{ji}} = \frac{\partial}{\partial w_{ji}}\left(w_0 \sum_{j=1}^{m} w_{ji} x_j + b_0\right) = \sum_{i=1}^{m} x_i$$

i.e. $\boxed{\frac{\partial v_j}{\partial w_{ji}} = \sum_{i=1}^{m} x_i}$

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

$$= e_j \times (-1) \times y_j(1-y_j) \times x_i$$

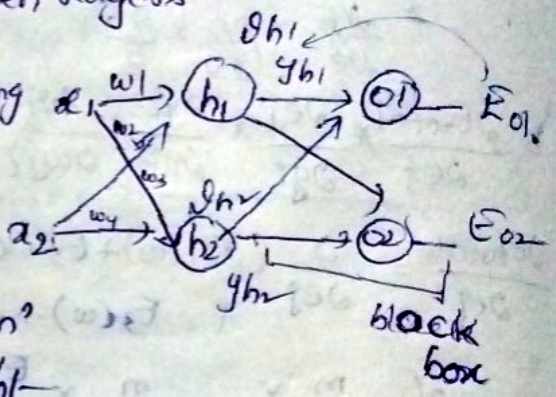
$$= -y_j(1-y_j) e_j x_i$$

$$\Rightarrow w_{new} = w_{old} + \eta\, g(n)$$

$$\Rightarrow \boxed{w_{new} = w_{old} + \eta y_j(1-y_j) e_j x_i} \quad -(6)$$

21/9/22

Computing gradient at hidden layers



- there is a possibility of getting error at the o/p layer.
  As per gradient descent Alg
  $w_{new} = w_{old} - \eta\, g(n)$
  $g(n)$ = Gradient of cost fun' with respect to weight

- E_{total} = error caused by hidden layer neuron i at o/p layer neuron j

  We look back by assuming entire o/p layer is black box.

- $g(n) = \frac{\partial E_{total}}{\partial w_{ji}}$ — (1)

  The o/p of hidden layer is cause for the error at o/p layer

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

$E_{total} \to y_j \to v_j \to w_{ji}$

↳ dependency

- $\frac{\partial E_{total}}{\partial y_i} = \frac{\partial E_{O1}}{\partial y_i} + \frac{\partial E_{O2}}{\partial y_i}$ — (3)

  ↓ hidden layer o/p

$$= \frac{\partial E_{O1}}{\partial y_{h_1}} + \frac{\partial E_{O2}}{\partial y_{h_1}}$$

$[y_{h_i} \to v_{oi} \to y_{oi} \to e_{oi} \to E_{oi}]$

Consider,

$$\frac{\partial E_{O1}}{\partial y_i} = \frac{\partial E_{O1}}{\partial e_{O1}} \times \frac{\partial e_{O1}}{\partial y_{oi}} \times \frac{\partial y_{oi}}{\partial v_{oi}} \times \frac{\partial v_{oi}}{\partial y_{hj}}$$

↳ (4)

[here we need to consider o/p layer neurons too]

↳ Since middle layers are all

$E_{oi} = \frac{1}{2}\sum_{i=1}^{?} e_{oi}^2$ ✓ $\Rightarrow \boxed{\frac{\partial E_{oi}}{\partial e_{oi}} = e_{oi}}$

$\frac{\partial E_{o1}}{\partial e_{o1}} = e_{o1}$, $\frac{\partial E_{o2}}{\partial e_{o2}} = e_{o2}$

$y_{oi} = \psi(v_i)$

$\Rightarrow \boxed{\frac{\partial y_{oi}}{\partial v_{oi}} = y_{oi}(1-y_{oi})}$ (for Sigmoid activation fun^n)

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

$\Rightarrow \boxed{\frac{\partial e_d}{\partial y_{oi}} = -1}$

$\Rightarrow \boxed{\frac{\partial v_{oi}}{\partial y_i} = w_{ji}}$

where $v_{oi} = \sum_{i=1}^{m} w_{ji} y_j + b_o$

From (4)

$$\frac{\partial E_{o1}}{\partial y_1} = \frac{\partial E_{o1}}{\partial e_{o1}} \times \frac{\partial e_{o1}}{\partial y_{o1}} \times \frac{\partial y_{o1}}{\partial v_{o1}} \times \frac{\partial v_{o1}}{\partial y_j}$$

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

$$= -y_{o1}(1-y_{o1})\, e_{o1}\, w_{ji}$$

$\boxed{\frac{\partial y_j}{\partial v_j} = y_j(1-y_j)}$ $\boxed{\frac{\partial v_j}{\partial w_{ji}} = x_i}$

From (3)

$$\frac{\partial E_{total}}{\partial w_{ji}} = -y_{o1}(1-y_{o1})\, e_{o1}\, w_{ji}\, y_j(1-y_j)\, x_i$$

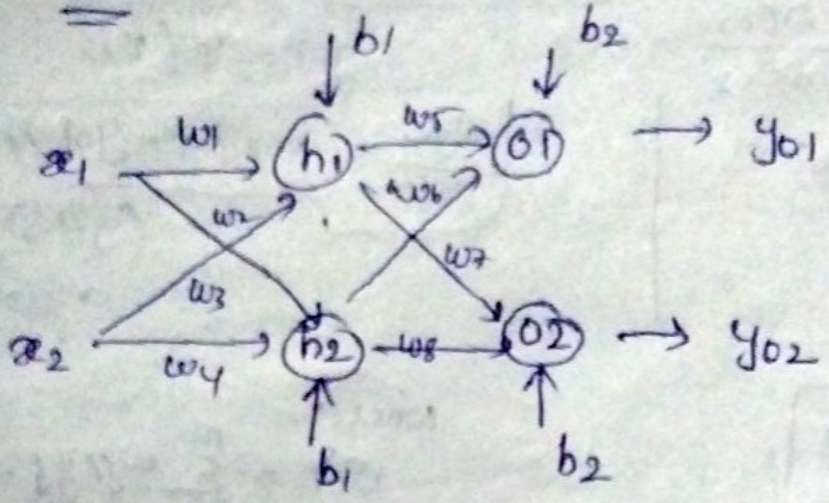
$$= -y_{o1}(1-y_{o1})\, y_j(1-y_j)\, e_o\, w_{ji}\, x_i$$

$$g(x) = -y_{o1}(1-y_{o1})\, y_j(1-y_j)\, e_o\, w_{ji} x_i$$

$$w_{new} = w_{old} + \eta\, y_{o1}(1-y_{o1})\, y_j(1-y_j)\, e_o\, w_{ji}\, x_i$$

↳ at hiddem layer.

Example



O1

$$v_{o1} = y_{h1} \times w_5 + y_{h2} \times w_6 + b_2$$

$$y_{o1} = \text{Sigmoid}(v_{o1}) = \frac{1}{1+e^{-v_{o1}}}$$

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

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

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

$$w_5(\text{new}) = w_5(\text{old}) - \eta\, g(n)$$

$$g(n) = -e_{o1} y_{o1} (1-y_{o1}) y_{h1}$$

$$w_5(\text{new}) = w_5(\text{old}) + \eta\, e_{o1} y_{o1} (1-y_{o1}) y_{h1}$$

$$w_6(\text{new}) = w_6(\text{old}) - \eta\, g(n)$$

$$g(n) = -e_{o1} y_{o1} (1-y_{o1}) y_{h2}$$

o2

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

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

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

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

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

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

$$w_7(new) = w_7(old) + \eta e_{o2} y_{o2} (1 - y_{o2}) y_{h1}$$

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

$$g(n) = -e_{o2} y_{o2} (1 - y_{o2}) y_{h2}$$

23/9/22

Example

Iteration 1:-

Forward Pass

at h1

$$v_{h1} = w_1 x_1 + w_2 x_2 + b_1$$

$$= 0.15 \times 0.05 + 0.20 \times 0.10 + 0.35$$

$$= 0.3825$$

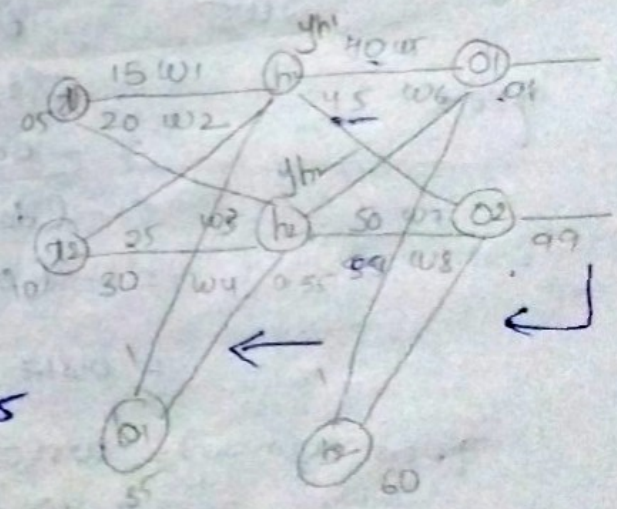
$$y_{h1} = \varphi(v_{h1}) = \frac{1}{1+e^{-v_{h1}}} = \frac{1}{1+e^{-0.3775}} = 0.5932$$

at h2

$$v_{h2} = w_3 x_1 + w_4 x_2 + b = 0.25 \times 0.05 + 0.30 \times 0.10 + 0.35$$

$$= 0.39$$

$$y_{h2} = \varphi(v_{h2}) = \frac{1}{1+e^{-0.39}} = 0.5962$$



at o_1

$v_{o1} = y_{h1} w_5 + y_{h2} w_7 + b_2$

$= 0.5941 \times 0.40 + 0.5962 \times 0.50 + 0.60$

$= 1.1359$

$y_{o1} = \psi(v_{o1}) = \frac{1}{1+e^{-v_{o1}}} = \frac{1}{1+e^{-1.1359}} = 0.7569$

at o_2

$v_{o2} = y_{h1} \times w_6 + y_{h2} \times w_8 + b_2 = 0.5941 \times 0.45 + 0.5962 \times 0.55 + 0.60$

$= 1.1954$

$y_{o2} = \psi(v_{o2}) = \frac{1}{1+e^{-v_{o2}}} = 0.767$

	d	y
o_1	0.01	0.7569
o_2	0.99	0.767

Backward pass

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

$g(m) = \frac{\partial E_{total}}{\partial w_5} = \frac{\partial E_{total}}{\partial e_{o1}} \times \frac{\partial e_{o1}}{\partial y_{o1}} \times \frac{\partial y_{o1}}{\partial v_{o1}} \times \frac{\partial v_{o1}}{\partial w_5}$

$= -e_d \times -1 \times y_{o1}(1-y_{o1}) \times y_{h1}$

$= -e_{o1} \times y_{o1} \times (1-y_{o1}) \times y_{h1}$

$= -(d_{o1} - y_{o1})(y_{o1}(1-y_{o1})) \times y_{h1}$

$= -0(0.01 - 0.7569)(1-0.7569) \times 0.5945$

$= 0.817$

$w_8(new) = w_5(old) - \eta g(m)$

$= 0.40 - 0.5 \times 0.817$

$= 0.3591$

23/9/22

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

$g(m) = \frac{\partial E_{total}}{\partial w_7} = \frac{\partial E_{total}}{\partial E_{o1}} \times \frac{\partial E_{o1}}{\partial o_1} \times \frac{\partial E_{o1}}{\partial y_{o1}} \times \frac{\partial y_{o1}}{\partial v_{o1}} \times \frac{\partial v_{o1}}{\partial w_7}$

$= e_{o1} \times -1 \times y_{o1} \times (1-y_{o1}) \times y_{h2}$

$= -(d_{o1} - y_{o1})\, y_{o1}(1-y_{o1})\, y_{h2}$

$= -(0.01 - 0.7569)\, 0.7569(1-0.75691)\, 0.5962$

$= 0.0819$

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

$= 0.50 - (0.5)(0.819) = 0.459$

$v_{o2} = y_{h1} w_6 + y_{h2} w_8 + b_2$

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

$g(n) = \frac{dE_{total}}{dw_6} = \frac{\partial E_{total}}{\partial e_{o2}} \times \frac{\partial e_{o2}}{\partial y_{o2}} \times \frac{\partial y_{o2}}{\partial v_{o2}} \times \frac{\partial v_{o2}}{\partial w_6}$

$= +e_{o2} \times -1 \times y_{o2}(1-y_{o2}) \times y_{h1}$

$= -e_{o2}(y_{o2})(1-y_{o2})\, y_{h1}$

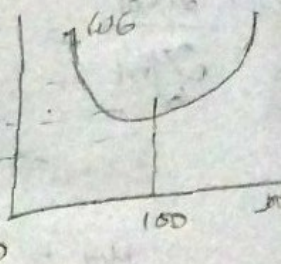
$= -(d_{o2} - y_{o2}) y_{o2}(1-y_{o2})\, y_{h1}$

$= -(0.99 - 0.767)(0.767)(1-0.767)\, 0.5945$

$= -0.023$

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

$= 0.45 - (0.5)(-0.023) = 0.461575$



w_6 is decrease Increment

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

$g(n) = \frac{\partial E_{total}}{\partial o_{o2}} \times \frac{\partial o_{o2}}{\partial e_{o2}} \times \frac{\partial e_{o2}}{\partial y_{o2}} \times \frac{\partial y_{o2}}{\partial v_{o2}} \times \frac{\partial v_{o2}}{\partial w_8}$

$= e_{o2} \times -1 \times y_{o2}(1-y_{o2}) \times y_{h1} = -(d_{o2} - y_{o2}) y_{o2}(1-y_{o2})\, y_{h1}$

$= -(0.99 - 0.769)(0.769)(1-0.769)\, 0.5962$

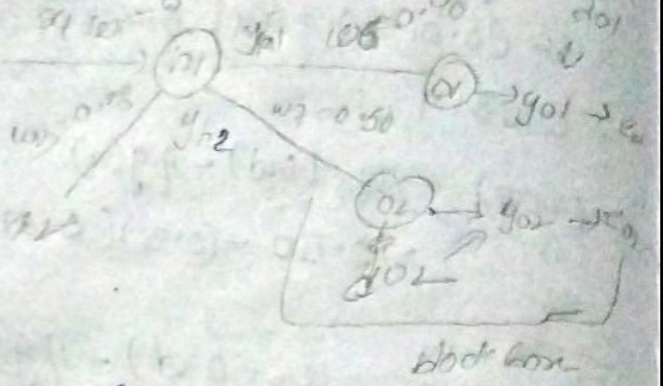
$= -0.0234$

$w_8(new) = w_8(old) - \eta g(n) = 0.55 - (0.5)(-0.0234)$

$= 0.5617$

At hidden Layer

At h_1



Updation of w_1

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

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

$$\frac{\partial E_{total}}{\partial w_1} = \frac{\partial E_{total}}{\partial h_1} \times \frac{\partial y_{h1}}{\partial v_{h1}} \times \frac{\partial v_{h1}}{\partial w_1}$$

$$\frac{\partial E_{total}}{\partial h_1} = \frac{\partial E_{01}}{\partial y_{h1}} + \frac{\partial E_{02}}{\partial y_{h1}} \quad \text{---(2)}$$

$$\frac{\partial E_{01}}{\partial h_1} = \frac{\partial E_{01}}{\partial e_{01}} \times \frac{\partial e_{01}}{\partial y_{01}} \times \frac{\partial y_{01}}{\partial v_{01}} \times \frac{\partial v_{01}}{\partial y_{h1}}$$

$$\frac{\partial E_{02}}{\partial h_1} = \frac{\partial E_{02}}{\partial e_{02}} \times \frac{\partial e_{02}}{\partial y_{02}} \times \frac{\partial y_{02}}{\partial v_{02}} \times \frac{\partial v_{02}}{\partial y_{h1}}$$

$$\Rightarrow \frac{\partial E_{01}}{\partial y_{h1}} = \frac{\partial E_{01}}{\partial e_{01}} \times \frac{\partial e_{01}}{\partial y_{01}} \times \frac{\partial y_{01}}{\partial v_{01}} \times \frac{\partial v_{01}}{\partial y_{h1}}$$

$$= e_{01} \times -1 \times y_0(1-y_{01}) \times w_5$$

$$= -e_{02}\, y_{01}(1-y_{01}) w_5$$

$$y_{01} = \frac{1}{1+e^{-v_{01}}}$$

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

$$v_{01} = y_{h1} \times w_5 + y_{h2} \times w_7 + b_2$$

$$\frac{\partial v_{01}}{\partial y_{h1}} = w_5$$

$$E_{01} = \frac{1}{2}\sum_{i=1}^{n} e_{01}^2$$

$$\frac{\partial E_{01}}{\partial e_{01}} = e_{01}$$

$$e_{01} = d_{01} - y_{01}$$

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

Similarly, $\frac{\partial E_{02}}{\partial y_{h1}} = \frac{\partial E_{02}}{\partial e_{02}} \times \frac{\partial e_{02}}{\partial y_{02}} \times \frac{\partial y_{02}}{\partial v_{02}} \times \frac{\partial v_{02}}{\partial y_{h1}}$

$$= e_{02} \times -1 \times y_{02}(1-y_{02}) \times w_6$$

$$= -e_{02}\, y_{02}(1-y_{02}) w_6$$

$\frac{\partial E_{01}}{\partial y_{h1}} = -e_{01} \times y_{01} \times (1-y_{01}) \times w_5$

$= -(d_{01} - y_{01}) \times y_{01} \times (1-y_{01}) \times w_5$

$= -(0.01 - 0.7569) \times 0.7569 (1-0.7569) \times 0.40$

$= 0.05497$

$\frac{\partial E_{02}}{\partial y_{h1}} = -e_{02} \times y_{02} \times (1-y_{02}) \times w_6$

$= -(d_{02} - y_{02}) \times y_{02} (1-y_{02}) w_6$

$= -(0.99 - 0.767) 0.767 (1-0.767) 0.45$

$= -0.01793$

$\frac{\partial E_{total}}{\partial h_1} = \frac{\partial E_{01}}{\partial y_1} + \frac{\partial E_{02}}{\partial y_{h1}} = 0.05497 - 0.01793$

$= 0.037$

$v_{h_1} = w_1 x_1 + w_3 x_2 + b$

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

$= 0.037 * y_{h1}(1-y_{h1}) \times x_1$

$= 0.037 \times (0.594) (1-0.574) (0.05)$

$g(x) = 4.465 \times 10^{-4} = 0.0004465$

$w_1(new) = w_1(old) - \eta g(x) \quad \eta = 0.5$

$= 0.15 - (0.5)(0.0004465)$

$= 0.1497$

* Updation of w_3

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

$g(x) = \frac{\partial E_{total}}{\partial w_3} = \frac{\partial E_{total}}{\partial y_{h1}} \times \frac{\partial y_{h1}}{\partial v_{h1}} \times \frac{\partial v_{h1}}{\partial w_3}$

$\frac{\partial E_{total}}{\partial h_1} = \frac{\partial E_{01}}{\partial y_{h1}} + \frac{\partial E_{02}}{\partial h_2}$ — ②

$\frac{\partial E_{01}}{\partial h_1} = \frac{\partial E_{01}}{\partial e_{01}} \times \frac{\partial e_{01}}{\partial y_{01}} \times \frac{\partial y_{01}}{\partial v_{01}} \times \frac{\partial v_{01}}{\partial y_{h1}} = e_{01} \times 1 \times y_{01}(1-y_{01}) \times w_5$

$\partial E = -(0.01 - 0.7569) \times 0.7569 (1-0.7569) \times 0.40$

$= 0.05497$

$$\frac{\partial E_{o2}}{\partial y_{h1}} = \frac{\partial E_{o2}}{\partial e_{o2}} \times \frac{\partial e_{o2}}{\partial y_{o2}} \times \frac{\partial y_{o2}}{\partial v_{o1}} \times \frac{\partial v_{o1}}{\partial y_{h1}}$$

$$= e_{o2} \times -1 \times y_o(1-y_{o2}) \times w_7$$

$$= -0.01793$$

$$\frac{\partial E_{total}}{\partial h_1} = \frac{\partial E_{o1}}{\partial y_{h1}} + \frac{\partial E_{o2}}{\partial y_{h2}} = 0.05497 - 0.01793$$

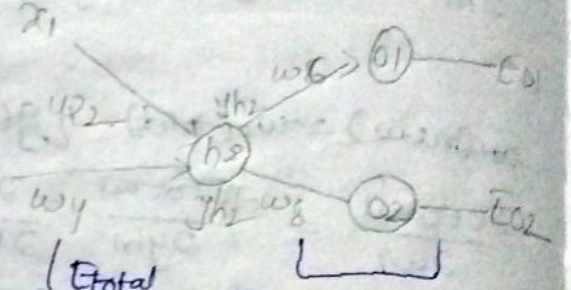
$$= 0.037$$

$$\frac{\partial E_{total}}{\partial w_3} = \frac{\partial E_{total}}{\partial y_{h1}} \times y_{h1} \times (1-y_{h1}) \times x_2 = 0.037 \times 0.594 \times (1-0.594) \times 0.10 = 0.00089$$

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

$$= 0.25 - 0.00089 = 0.249$$

At h_2



Updation of w_2

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

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

$$= \frac{\partial E_{total}}{\partial y_{h2}} \times \frac{\partial y_{h2}}{\partial v_{h2}} \times \frac{\partial v_{h2}}{\partial w_2} \quad \text{①}$$

E_{total} → y_{h2} → v_{h2} → w_2

E_{o1} → e_{o1} → y_{o1} → v_{o1} → y_{h2}

E_{o2} → e_{o2} → y_{o2} → v_{o2} → y_{h2}

$$\frac{\partial E_{total}}{\partial y_{h2}} = \frac{\partial E_{o1}}{\partial y_{h2}} + \frac{\partial E_{o2}}{\partial y_{h2}} \quad \text{②}$$

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

$$= e_{o1} \times -1 \times y_{o1}(1-y_{o1}) \times w_8$$

$$= -(d_{o1} - y_{o1}) \times y_{o1}(1-y_{o1}) \times w_8$$

$$= -(0.01 - 0.7569) \times 0.7569(1-0.7569) \times 0.50$$

$$= 0.068$$

$v_{o1} = y_{h1} \times w_5 + y_{h2} \times w_7 + b_2$

$$\frac{\partial E_{o2}}{\partial y_{h2}} = e_{o2} \times -1 \times y_{o2}(1-y_{o2}) \times w_8$$

$$= -(0.99 - 0.767)(1-0.767)(0.767) \times 0.55$$

$$= -0.02177$$

$v_{o2} = y_{h1} \times w_6 + y_{h2} \times w_8 + b_2$

$$\frac{\partial E_{total}}{\partial y_{h2}} = 0.068 - 0.02177$$

$$= 0.047$$

$\frac{\partial E_{total}}{\partial w_2} = 0.047 \times y_{h_2} \times (1 - y_{h_2}) \times x_1$

$= 0.047 \times 0.596 \times (1 - 0.596) \times 0.05 = 0.005$

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

$= 0.20 - 0.5 \times 0.85 = 0.199$

$= 0.199$

$w_4(new) = w_4(old) - \eta g(n)$

$= 0.30 - 0.5 \times 0.00011$

$= 0.299$

$g(n) = \frac{\partial E_{total}}{\partial w_4}$

$= \frac{\partial E_{total}}{\partial y_{h_2}} \times \frac{\partial y_{h_2}}{\partial v_{h_2}} \times \frac{\partial v_{h_2}}{\partial w_4}$

$= 0.047 \times y_{h_2} \times (1 - y_{h_2}) \times x_1$ ①

$= 0.04 \times (0.596)(1 - 0.596) \times 0.10$

$= 0.00011$

After Iteration 1

$w_1 = 0.149$
$w_2 = 0.199$
$w_3 = 0.249$
$w_4 = 0.299$
$w_5 = 0.359$
$w_6 = 0.5617$
$w_7 = 0.459$
$w_8 = 0.5617$

$x_1 = 0.05$
$x_2 = 0.10$

Iteration 2

Forward pass is done

⇒ Practical & Design Issues of Back propagation Algorithm

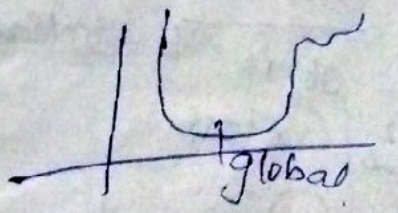
① Rate of Learning

- Specifies a step size taken by Learning algorithm.
- takes a value in blw $0 - 1$ $\quad 0 \le \eta \le 1$

η is very small

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

↓ learning rate



→ Slowly Converges to the global minimum
- It increase no. of iterations required to Converges
- algorithm can trapped into the local minimum point.

η = very large

→ algorithm can overshoot global minimum point.

→ Can oscillate

→ Avoids getting trap into local minima point.

Soln

momentum :-

- $\Delta w_{ji}(n) = \alpha \Delta w_{ji}(n-1) + \Delta w_{ji}(n)$
- Provides soln 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 towards global minima point.

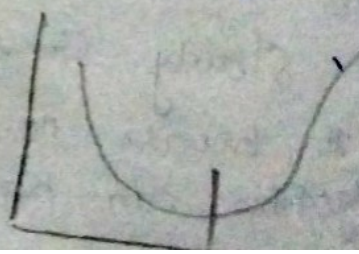
drawback

- It is theoritically difficult to determine stopping criteria.

10/22

* Design Issues in Backpropagation Algorithm

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



→ momentum

② Stopping Criteria

→ Backpropagation Alg
↳ Learning Alg

(1) when model produces error value that is acceptable. (95% of accuracy)

(2) when sum of the difference b/w y & d reaches Euclidian to minimum value

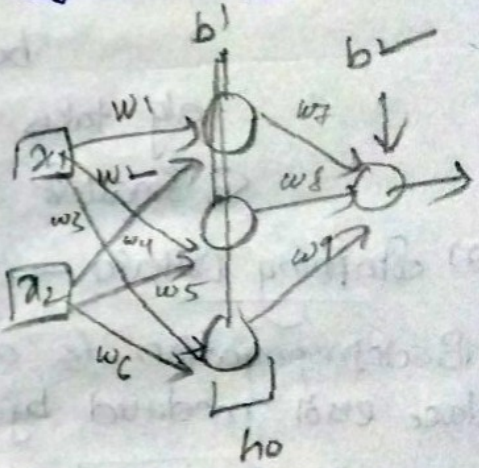
③ New sample
- not there in the training set.
↳ Generalization

Creates Powerset

* Implementation of Backpropagation Algorithm

1) Initialize free parameters

$$hw = \begin{bmatrix} w_1 \\ w_2 \\ w_3 \\ w_4 \\ w_5 \\ w_6 \end{bmatrix}_{6\times 1} \qquad ho = \begin{bmatrix} w_7 \\ w_8 \\ w_9 \end{bmatrix}_{3\times 1}$$



$$vh_1 = x_1 w_1 + x_2 w_4$$
$$vh_2 = x_1 w_2 + x_2 w_5$$
$$vh_3 = x_1 w_3 + x_2 w_6$$

$$x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}_{2\times 1}$$

$$hw = \begin{bmatrix} w_1 & w_4 \\ w_2 & w_5 \\ w_3 & w_6 \end{bmatrix}$$

random package
↳ randint
↳ to initialize values for w, b

Design Issues in Back propagation Algorithm

1) Learning rate (η) - due to this algorithm converges slowly or at fast rate. $\rightarrow w(new) = w(old) - \eta g(n)$

(i.e, update value of previous iteration is added to update formula)

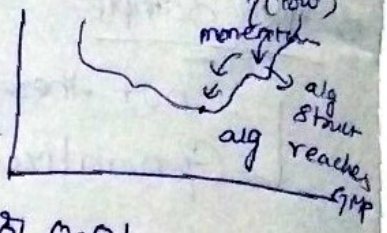
roto

alg stuck in local min point

momentum is γ applied (It accerates alg and alg move forward from local min point and to reach global min point.

then update formula changes

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



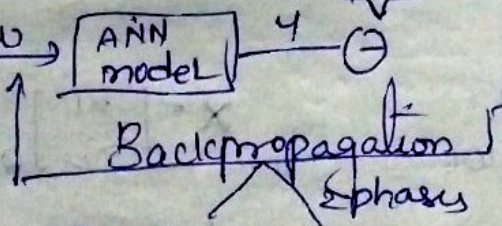
- α-momentum term
- lies blw 0 to 1 mostly

$\alpha = 0.001$ or 0.01

- α is multiplied with $\Delta w(n-1)$ because if $\Delta w(n-1)$ is too high then alg takes fewer & faster steps to reach GMP. So, Smallest value α is multiplied with $\Delta w(n-1)$

(2) Stopping Criteria —

Backpropagation is a learning / optimization alg used to reduce error produced by ANN



d-y=e → Simple perceptron → Gradient descent Alg

multiple layers

Goes to iteration process

goes to feed forward phase (read i/p produces o/p & identifies error then)

This process occurs iteratively.

If no stopping criteria exists, then Back propagation iterates infinitely.

when y=d - but leads to infinite loop
General stopping criteria used in i.e BP algo is stopped when error values reaches a particular threshold accuracy value (98%. - Stopping criteria = 0.5
correct value should predicted 95% accurate)
when euclidean ndim of d & y reaches to a min. value

3) new sample not there in training set
- Backpropagation goes through Generalization process (it tries to compare data with already trained previous data & make conclusions. (given sample sample)

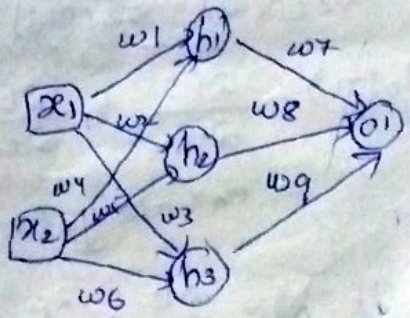
* powerset is created for the training data using which model is trained. powerset contains all features of dataset so that based on it model can make predictions for new data to powerset is used in Generalization.

12/10/22

Implementing Backpropagation

① Initialize parameters
1. weights
2. bias
3. Learning parameter

② Construct neural network
→ no. of i/ps
→ no. of hidden layers
- no. of neurons
→ no. of neurons in o/p layer



$v_{h1} = w_1 x_1 + w_4 x_2 + b$
$v_{h2} = w_2 x_2 + w_5 x_2 + b$
$v_{h3} = w_3 x_1 + w_6 x_2 + b$

$$w = \begin{bmatrix} w_1 & w_4 \\ w_2 & w_5 \\ w_3 & w_6 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} w_1 x_1 + w_4 x_2 \\ w_2 x_1 + w_5 x_2 \\ w_3 x_1 + w_6 x_2 \end{bmatrix}_{3 \times 1}$$

Size of i/p × neurons in hd

n × 1

$w_2 \begin{bmatrix} w_7 \\ w_8 \\ w_9 \end{bmatrix}_{3\times1}^{T} \Rightarrow [w_7\ w_8\ w_9]$ for olp layer

$\begin{bmatrix} y_{h1} \\ y_{h2} \\ y_{h3} \end{bmatrix}$

Size of hidden l × Size of olp layer

$\Rightarrow w_7 y_{h1} + w_8 y_{h2} + w_9 y_{h3}$

We use random module for creating w, b values.

If we want the more bias values

bias. $b_h = [\ ]_{n\times1}$ (n = no of hidden layer neurons).

$b_o = [\ ]_{1\times1}$

Step 3: Compute the error value

$e = d - y$

Since error occurs back propagation updates the weights

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

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

$g_o(n) = \frac{\partial E}{\partial w_2} = \frac{\partial E}{\partial e_o} \frac{\partial e_o}{\partial y_o} \frac{\partial y_o}{\partial v_o} \frac{\partial v_o}{\partial w_2}$

$= e_o \times -1 \times y_o(1-y_o)\, y_h$ (δ)

First Compute δ values

$g_h(n) = \frac{\partial E}{\partial w_1} = \frac{\partial E}{\partial y_h} \times \frac{\partial y_{h1}}{\partial v_{h1}} \times \frac{\partial v_{h1}}{\partial w_1}$ → contribution of olp by hidden layer nps

$= \frac{\partial E}{\partial e_o} \frac{\partial e_o}{\partial y_o} \frac{\partial y_o}{\partial v_o} \frac{\partial v_o}{\partial y_h} \times \frac{\partial y_h}{\partial v_h} \times \frac{\partial v_h}{\partial w_1}$

$\delta \times w_2$

Dataset $= \delta \times w_2 \times y_h(1-y_h) \times$

	hours studied	hours played	result
1	7	2	80
2	2	6	50
3	9	1	95

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
```

define activation fun^n

```
def Sigm(0):
    y = 1/(1+np.exp(-0))
    return y
def derivate(y):
    return (y*(1-y))
```

yp = predicted dp

```
def mseqerror(yp, y)
    E = ((yp-y)**2).sum()/2
    return E
```

define a dataset

```
x = np.array([[7,2],[2,6],[9,17]])
y = np.array([[80],[50],[96]])
#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) + b1
yh = sigm(vh)
vo = np.dot(yh, w2) + b2
yo = sigm(vo)
return yo

train(x, y)
# Back propagation phase
e1 = y - yo
delta_o = e1 * yo * (1 - yo)
e2 = delta_o @ w2.T
delta_h = e2 * yh * (1 - yh)
g0 = yh.T @ delta_o
gh = x.T @ delta_h

w1 = w1 - lr * gh
w2 = w2 - lr * go
b1 = b1 - lr * delta_h
b2 = b2 - lr * delta_o

# Backpropagation is steepest algorithm
for i in range(2000):
    train(x, y)
def forwardtest(x, y):
    global w1, w2, b1, b2
    # forward phase
    vh = np.dot(x, w1) + b1
    yh = sigm(vh)
    vo = np.dot(yh, w2) + b2
    yo = sigm(vo)
    return yo
```

forward test (x, y)

L_dim = ['none']
for i in len(l_dim)
w[w+str(i)] = np.
random.randn(l_dim[i+1])