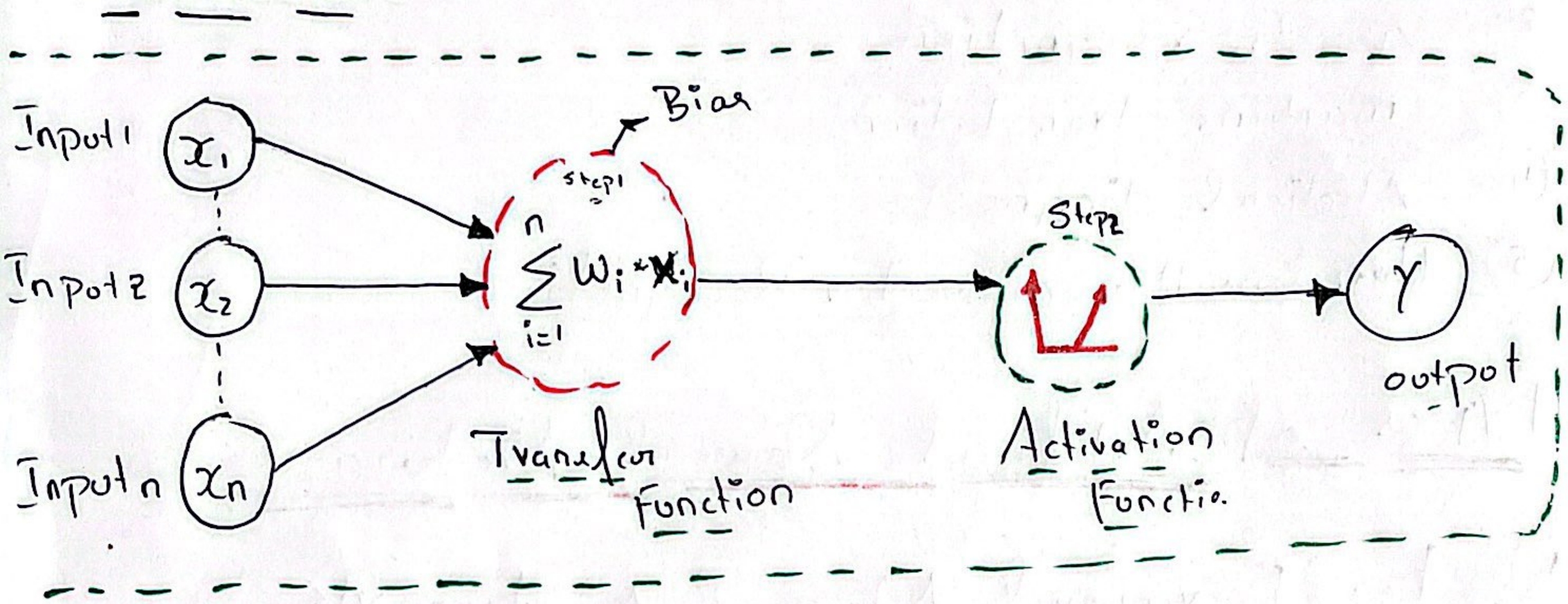
(ANN)

Artifical Neural Network.

(Classifier

Architecture



A computational network grounded on natural neural network that construct the structure of the human brain is known as Artifical Neural Network.

From Biological Aspect

Dendriten	Cell nuclei	Synapses	axons
↓	↓	↓	↓
Inputs	node	weights.	output

* ANN, dendrites from biological neural network. represent input, cell. and ANN is a non-linear Statistical model.

That demonstvate a complex relationship between input and output in order to uncover a new Pattrn.

Application

① image recognition
② Speech recognition.
③ machine translation
④ Medical diagnosis.
⑤ Handwritten character Recognition

Types of Artifical Neural Network.

① FeedForward Neural Network:

- ⊙ Information flow in FFNN is Single direction.
- ⊙ No feedback loops.
- ⊙ These neural Networks are commonly employed. In Supervised learning. for classification.
- ⊙ Image recognition.
- ⊙ we use them when data is not in consective order.
- ⊙ FF are comparable to CNN

② Feedback Neural Network.

- ⊙ Feedback loops are heart of ANN
- ⊙ used when data is Sequential or time-dependent.
- ⊙ FB are comparable to RNN.

Preprocessing Steps for ANN

for (historical text, numerical data)

Step 1: importing the data.
Step 2: understany the distribution of the data.
Step 3: Data cleaniny (No null value)
Step 4: Encodiny: (catayorical to numerical)
Step 5: Feature Scaliny (Normalization)
Step 6: Tvain - Test - split.

Preproccesiny for "image classification"

Step 1: Image Resizing

The imayes Need to be resized to a standard size to ensure that the model is not affected by the size of the image.

Step 2: Image Normalization

The pixel value of the image need to be Normalized. to ensure that they are in the same vanye and do not affect the moded.

Step 3: Image Augmentation.

To increase the size of the training dataset, image can be augmented by applying tranfamation Such as rotation, flipping and scaling.

Step 4: Image Filtering:

To remove any noise or unwanted features from the images, they can be filtered using techniques Such as median filtering & Gaussin filtering.

Step 5 Image Segmentation.

To isolate the objects of interest in the images they can be segmented using techniques Such as thresholding, edge detection, and morphological operation.

Step 6: Image Histogram Equalization:

To adjust the brightness and contrast of the image, they can be equalized using techniques. Such as histogram equalization.

Step 8: Image conversion:

The image needs to be ~~encoded into numerical~~ vectors converted to a format that is. compatible with the model. Such as grayscale or RGB.

Step 8: Image Encoding.

The labels for the images need to be encoded into numerical values to be used by the model.

Step 9: Image Preprocessing Pipeline:

The preprocessing steps need to be combined into a Pipeline to ensure that they are performed consistently and efficently.

Optimizers used by ANN algorithm.

Problems and How did they overcome it?

① Gradient Descent

Problems: It too Expensive to calculate the gradient if the size of the data is huge.
- can't determine how far to travel along the gradient for nonconvex function.

TO Overcome

② Stochastic Gradient Descent.

How did they overcame: instead of taking the whole dataset for each iteration we randomly select the batche of data. we take few samples from the dataset

Problem: as it is random path had noise.
- higher no of iteration to reach local minima
- Computation time increases.

TO overcome.

③ Mini Batch Gradient Descent

How did it overcome problem?

: Instead of taking single Sample a a time Batch of Samples were taken to reduce the time.

↓ Problems : as it has noise in path.
to overcom.

④ Stochastic Gradient Descent with Momentum.

SGD with Momentum overcome this problem by:

: Here momentum does help in faster convergence of the logs function.

we use Exponential weighted average.
Basic used in time Series.

Problem : Not completly Solve Noise problem But reduces efficiently. if the momentum is increased possiblity of reaching GM will be less.

↓ As of now lets keep this optimizers as Best Because, it reduces the noise, time, compolation cost.

SGD with Momentum was all about weights and Bias.

① Adagvad (Adaptive Gvadient Descent).

This optimizev is vegvding learning vate.

In Previous opt we know η (learning vate is constant).

But Adagvad η will be Adaptive.

Problem: It will degrew fast so it hard to find Globle minma. And becauer of Squared gvadienly denominatar. .L R Becomes Extoremely small. So model unable to acquire more knowledge..

TO overcome this

② Ada delta and RMS Prop.

~~Both~~ are RMS Prop ~~Adam~~ is to Bimprove. AdaGvad. It uses Esponential Moving average. which Similar to momentom. degvade to Speedup

Ada delte : focuse on learning vate component.

delta is difference blw current and new weight

Adadelta complety removes the need of learning vate parameter completely. By replacing it with D.

Adam optimizer. (Best in Market).

It is the combination of Momentom and. RMSProp.. Both adaptive learning rate and Smoothing is also done together.

It can handled Sparse gradient on noisy Problem

Activation Function's. Problem and How to Overcome it.

① Sigmoid: $\sigma(x) = \frac{1}{1+e^{-x}}$.

used for the output of Binary Probabilistic Function. Gives Smooth & slow converging in gradint and clear prediction with 1 to 0.

Problem:
= Prone to Vanishing Gradient problem.

derivative will not Exceed 0.25 at this Point it will be constant.

- Not a zero centric. (always give positive value).
- computationaly Expensive. (Exponential in nature)

Note: Should used this function for Output layers in Binary classification.

② Tanh: $x = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}$.

- It is 0 centric vale (-1 to 1).

Problem: ⊙ Prone to Vanishing Gradient function
⊙ compotation cost.

③ RELU (Rectified linear Unit).

$$\boxed{\max(0,x)}$$

It solves VGD By -ve value will be 0 and +ve value will be any positive integer.

Problem: -ve input will be Inactive Neuron. (=0)

④ Leakly Relu : $\boxed{f(x) = \max(0.01x, x).}$

-Ve value will be Small value.

It Solves dead neuron problem.

⑤ ELU (Exponential linear unit).

$$\boxed{f(x) = \begin{cases} x & \text{if } x > 0 \\ \alpha(e^{-x} - 1), & \text{otherwise.} \end{cases}}$$

- overcome dead relu problem; Smoother convergence for any negative axis value.
- ⊙ It Behave like Step function.

Proble: more computational intensive.

⑥ PRELU (Parametric RELU) : $\boxed{f(y_i) \begin{cases} y_i & \text{if } y_i > 0 \\ a_i y_i & \text{if } y_i \leq 0 \end{cases}}$

- It has learning Paramete function which fine-tuned. the activation based on learning rate. (no constant value).

⑦ **Soft Max** : $$S(x_j) = \frac{e^{x_j}}{\sum_{k=1}^{K} e^{x_k}}, \quad j = 1, 2, \ldots, K$$

used when output layer is Multical problem. more Probablity, one with highest probability, value. it will be considered.

⑧ **Swish (A self Gated) function**

$$y = x^* \text{ Sigmoid}(x).$$

used when there are more than 40 layers. like LSTM (Long Short Term Memory) Network.

Problem: computationally expensive.

⑨ **Max Out**: $$\max(w_1^T x + b_1, w_2^T x + b_2).$$

It is Learnable Activation function.

Parameter is, over comes are Generalised.

Relu & leaky relu. output will be.

Largest activation value.

⑩ **Softplus**: $$f(x) = \ln(1 + \exp x).$$

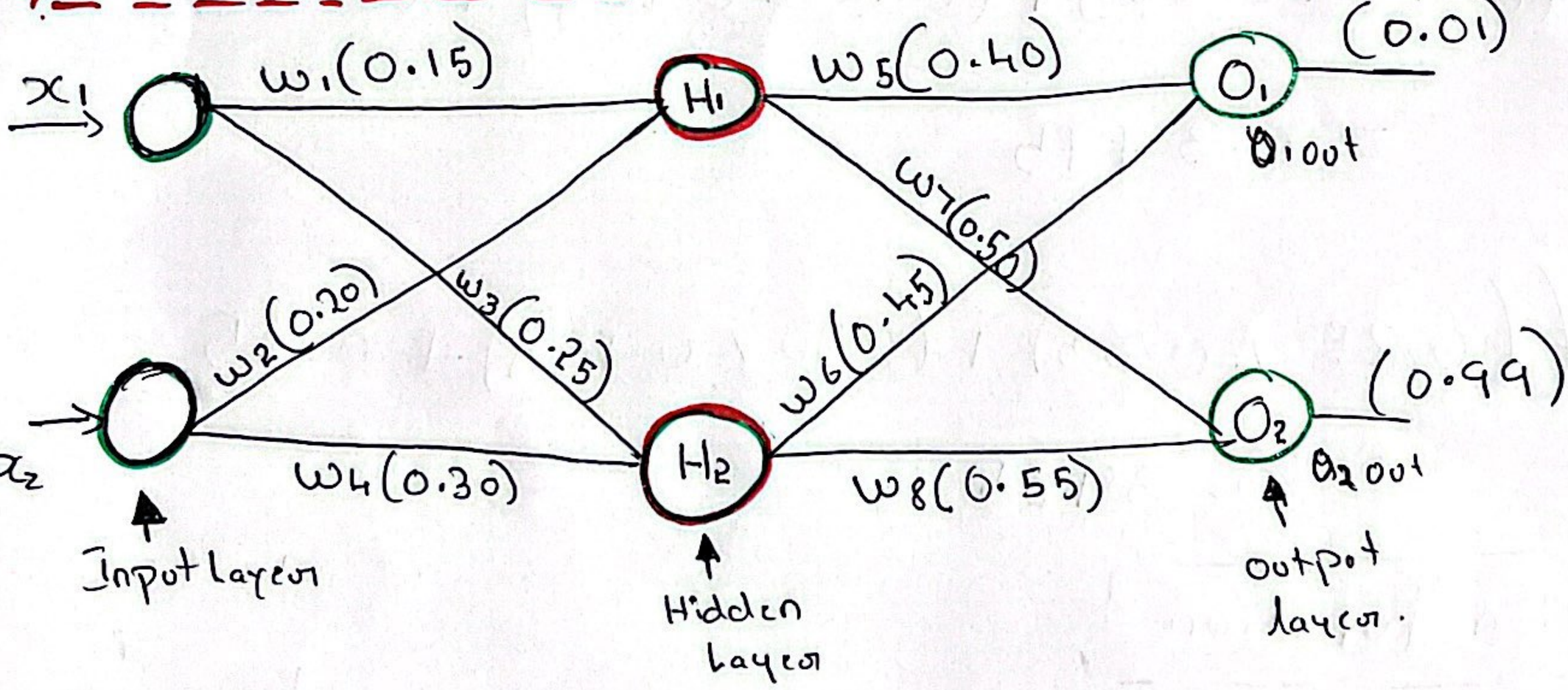
Similar to Relu & relative smooth. can handel VGD.

Problem: computationally expensive.

ANN Math Calculation.

Math with Algorithm Steps

Step 1: Define the architecture of the network including number of layers, the number of neurons in each layer, Activation function for each layer.



Input : $x_1 = 0.05$
$x_2 = 0.10$

Actual output

T_1	T_2
0.01	0.99

Weights ← taken randomenly → **Bias**

$w_1 = 0.15$
$w_2 = 0.20$
$w_3 = 0.25$
$w_4 = 0.30$
$w_5 = 0.40$
$w_6 = 0.45$
$w_7 = 0.50$
$w_8 = 0.55$
$b_1 = 0.35$
$b_2 = 0.60$

$\eta = 0.6$

Activation function: Sigmoid = $\frac{1}{1+e^{-y}}$

Step 2: Feed forword Propogate the input data through the network, calculating the Activation of each neuron.

Summation function = $\sum_{i=0}^{i} w_i x_i + b$

$H_1 = (w_1 x_1 + w_2 x_2) + b_1$ H_1 (In)

$H_2 = (w_3 x_1 + w_4 x_2) + b_1$ H_2 (In)

$H_1 = ((0.15 \times 0.05) + (0.10 \times 0.20)) + 0.35$

$= 0.3775$

$H_2 = ((0.25 \times 0.05) + (0.30 \times 0.10)) + 0.35$

$= 0.3925$

Activation function

$H_1 out = \frac{1}{1+e^{-H_1}}$ $H_2 out = \frac{1}{1+e^{-H_2}}$

$\sigma(H_1) = \frac{1}{1+e^{-0.3775}}$

$= 0.5932$

$\sigma(H_1) = \frac{1}{1+e^{-0.3925}}$

$= 0.59688$

Calculate O_1 (in and out)

$$O_1(in) = H_1(out) \times w_5 + H_2(out) \times w_6 + b_2$$

$$= 0.593 \times 0.4 + 0.596 \times 0.45 + 0.6$$

$$= 1.105$$

$$O_1(Out) = \frac{1}{1+e^{-O_1(in)}} = \boxed{0.7513}$$

Calculate O_2 (in and Out)

$$O_2(in) = H_1(out) \times w_7 + H_2(out) \times w_8 + b_2$$

$$= 0.5932 + 0.50 + 0.5968 + 0.55 + 0.6$$

$$O_2(out) = 1.22484$$

$$= \frac{1}{1+e^{-O_2(in)}} = \boxed{0.7729}$$

Step 3: Calculate the error: compare the Predicted output to the actual output and Calculate the error using a loss function such as Mean Squared Error.

Calculate logs (MSE)

$$E_{total} = \sum \frac{1}{2}(target - o/p)^2$$

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

$$= \frac{1}{2}(0.01 - 0.7513)^2 + \frac{1}{2}(0.99 - 0.7724)^2$$

$$= \frac{1}{2}(-0.7413)^2 + \frac{1}{2}(0.2171)^2$$

$$= 0.274 + 0.0235 = \boxed{0.29837} \text{ (approximately)}$$

Step:4

Now to minimize the loss we use.

Backward Propagation Method.

Calculating Back Propagation.

w_5, w_6, w_7 and w_8. (output → Hidden)

First lets adjust w_5. ——→ learning rate.

$$w_{5_{new}} = w_5 - \eta \frac{\partial E_{total}}{\partial w_5}$$

$$\frac{\partial E_{total}}{\partial w_5} = \frac{\partial E_{total}}{\partial out_{o1}} \times \frac{\partial out_{o1}}{\partial net_{o1}} \times \frac{\partial net_{o1}}{\partial w_5}$$

$$\frac{\partial total}{\partial out_{o1}} = Out_{o1} - target_{o1}$$

$$0.751365 - 0.01$$

$$= 0.7413565$$

$$\frac{\partial out_{o1}}{\partial net_{o1}} = Out_{o1}(1 - out_{o1})$$

$$= 0.751365(1 - 0.751365)$$

$$= 0.186815602$$

$$\frac{\partial net_{o1}}{\partial w_5} = Out_{h1} = 0.59326992$$

Now calculating together

$$\frac{\partial E_{total}}{\partial w_5} = 0.7413565 \times 0.186815602 \times 0.59326992$$

$$= \underline{0.08216704}$$

$$w_{5_{new}} = w_5 - \eta \frac{\partial E_{total}}{\partial w_5}$$

$$= 0.4 - 0.6 * 0.08216704$$

$$= \underline{0.35069477 6}$$

(ii) $w_{6_{new}} = w_6 - \eta \left[\frac{\partial E_{total}}{\partial w_6} \right]$

$$\frac{\partial E_{total}}{\partial w_6} = \frac{\partial E_{total}}{\partial out_{o_1}} * \frac{\partial out_{o_1}}{\partial net_{o_1}} * \frac{\partial net_{o_1}}{\partial w_6}$$

$$w_6 = 0.408666186$$

(iii) $w_{7_{new}} = w_7 - \eta \frac{\partial E_{total}}{\partial w_7}$

$$\frac{\partial E_{total}}{\partial w_7} = \frac{\partial E_{total}}{\partial out_{o_1}} * \frac{\partial out_{o_1}}{\partial net_{o_1}} * \frac{\partial net_{o_1}}{\partial w_7}$$

$$w_7 = 0.511301270$$

Step 5: Update weights of each connection in the network. Using optimization which is calculated in over steps. (Hidden → input layer)
(w_1, w_2, w_3, w_4).

First lets adjust w_1

$$w_{1_{new}} = w_1 - \eta \left[\frac{\partial E_{total}}{\partial w_1} \right]$$

$$\frac{\partial E_{total}}{\partial w_1} = \frac{\partial E_{total}}{\partial out_{H_1}} \times \frac{\partial out(H_1)}{\partial net_{H_1}} \times \frac{\partial net_{H_1}}{\partial w_1}$$

$$\frac{\partial E_{total}}{\partial out(H_1)} = \frac{\partial E_{O_1}}{\partial out_{H_1}} + \frac{\partial E_{O_2}}{\partial out_{H_1}}$$

$$\frac{\partial E_{O_1}}{\partial net_{O_1}} \times \frac{\partial net_{O_1}}{\partial out_{H_1}}$$

$$\frac{\partial E_{O_1}}{\partial out_{O_1}} \times \frac{\partial out_{O_1}}{\partial net_{O_1}}$$

$$\frac{\partial E_{O_2}}{\partial net_{O_2}} \times \frac{\partial net_{O_2}}{\partial out_{H_1}}$$

$$\frac{\partial E_{O_2}}{\partial out_{O_2}} * \left[\frac{\partial out_{O_2}}{\partial net_{O_2}}\right]$$

$$\frac{\partial E_{O_2}}{\partial out_{O_2}} = (out_{O_2} - target\, O_2)$$

$$= 0.772928465 - 0.99$$

$$= -0.217071535$$

$$\frac{\partial out_{O_2}}{\partial net_{O_2}} = out\, O_2 (1 - out\, O_2)$$

$$= 0.7729(1 - 0.7729)$$

$$= 0.175510052$$

$$\frac{\partial E_{O_1}}{\partial out_{O_1}} = (out_{O_1} - target_{O_1})$$

$$= (0.7513 - 0.01)$$

$$= \underline{0.7413}$$

$$\frac{\partial out_{O_1}}{\partial net_{O_1}} = out_{O_1}(1 - out_{O_1})$$

$$= (0.7513(1 - 0.7513))$$

$$= 0.18684831$$

$$\frac{\partial net_{O_1}}{\partial out_{H_1}} = \text{on } O_1 \text{ from } H_1 \Rightarrow w_5$$

$$= 0.4$$

$$\frac{\partial net_{O_2}}{\partial out_{H_1}} = \text{on } O_2 \text{ from } H_1 \Rightarrow w_7$$

$$= 0.50$$

$$0.138\,49856 \times 0.4 + (-0.0380982 \times 0.50)$$

$$= 0.055\,394\,427 + (-0.019\,049\,119)$$

$$\frac{\partial E_{total}}{\partial out_{(H_1)}} = 0.036\,35\,0306$$

$$\frac{\partial out_{h1}}{\partial net_{h1}} = out_{h1}(1 - out_{h1})$$

$$= 0.241300704$$

$$\frac{\partial net_{h1}}{\partial w_1} = \frac{\partial}{\partial w_1}(w_1 x_1 + w_2 x_2 + b_1)$$

$$= x_1 = 0.05$$

$$\frac{\partial E_{total}}{\partial w_1} = 0.036350306 \times 0.241300704 \times 0.05$$

$$= 0.00438568$$

$$w_{1,new} = w_1 - \eta \frac{\partial E_{total}}{\partial w_1}$$

$$= 0.15 - 0.6 \times 0.00438568$$

$$= 0.149736842$$

Step 6 : Repeat Step 2-5 : continuesly. feedforward, Error calculation Back Propagation. and weight update steps until the network convergens or until a pre-defined stopping criterion is met.

Bias update also works similarly as weight updation.

formula

$$b_{new} = b_{old} - \eta \frac{\partial L}{\partial b_{old}}$$

If we repeat the step 2-5 we will get a. w_2, w_3, b & values.

$w_2 = 0.19456143$

$w_3 = 0.24975114.$

$w_4 = 0.29450229$