Module - I Deep Learning

→ What is artificial intelligence?

- It is the ability of a computer to perform tasks commonly associated with intelligent beings.

→ What is machine learning?

- It is the study of algorithms that learn from data.
- P - Performance
  T - Task
  E - Experience
- Supervised Learning
  → Labelled dataset (categorical value)
  → Regression (continuous value), Classification - KNN, SVM, Decision Tree, Random Forest, Artificial Neural Networks.
- UnSupervised Learning:
  → Unlabelled dataset
  → Clustering, Association

- Regression: Trying to fit a line to a set of data points
  - The distance b/w the datapoints and the line is called error.
  - The best fit line will be the one with the less error.

* In machine learning, we are trying to find the equation of a line

$$y = mx + c$$

* In the case of regression, we find the best fit line to set of data points
* In classification, we are trying to find the line that partitions the data points.

Regression:

$$y = x + c$$

- y: dependant variable (what we're predicting)
- x: independant variable

$$b_1 = \frac{n\Sigma(x_i y_i) - \Sigma x_i \Sigma y_i}{n\Sigma x_i^2 - (\Sigma x_i)^2}$$

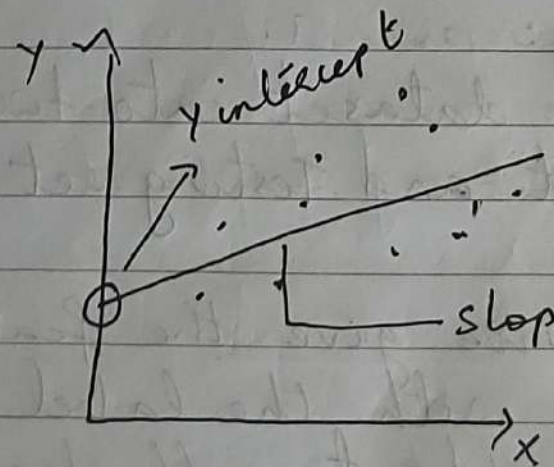
* If there are three variables (equation of a plane)

$$Y = b_0 + b_1 x_1 \cdot b_2 x_2$$

* If its only one variable, it is a threshold.
* If there are more than three variables we are trying to find the equation of a hyperplane.

$$y = mx + c$$

m → slope, c → Y intercept



If we keep changing the slope and Y intercept, we can find the best fit line, that has the lowest error.

Training

Linear Regression:

Residuals:
Difference b/w actual and predicted value.

* The best model is estimated by minimizing the sum of squares of these residuals

$$SS_{res} = \sum_{i}^{n} (y_i - f(x_i))^2$$

↓

cost function used to measure the error of the model.

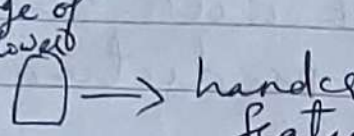
Model Training:
Partition the dataset into two:
Training set and Testing set.

* For training, we give the feature data along with the label to model and the output will be an equation $y = w_0 + w_1 x_1 + w_2 x_2 + \cdots w_n x_n$.
* After training, we will get a learned model.

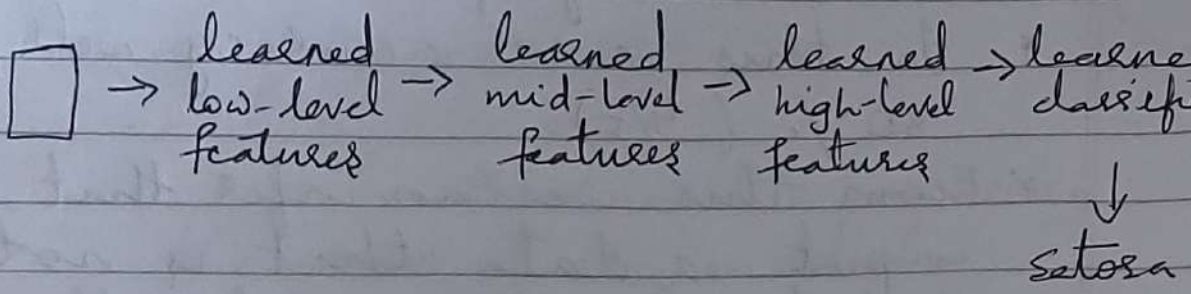
* In test data, we will mask the label and the model will predict the category. It may be correct or wrong.
* We can cross check with the label to verify if the prediction was correct or wrong.
* If 4 out of 5 predictions were right, we use evaluation metrics like accuracy, precision and recall.
* In this case accuracy will be 80% (4/5).
* From this we can infer that, if we input a data that is not in the label set, the model will predict correctly with 80% accuracy.
* In machine learning, we make the machines are learning the weights associated with the features.

* What is deep learning?

Traditional machine learning:



Deep "end-to-end" learning:



* Deep learning uses artificial neural networks to learn from lots of data without needing explicit programming

Artificial Neural Networks:

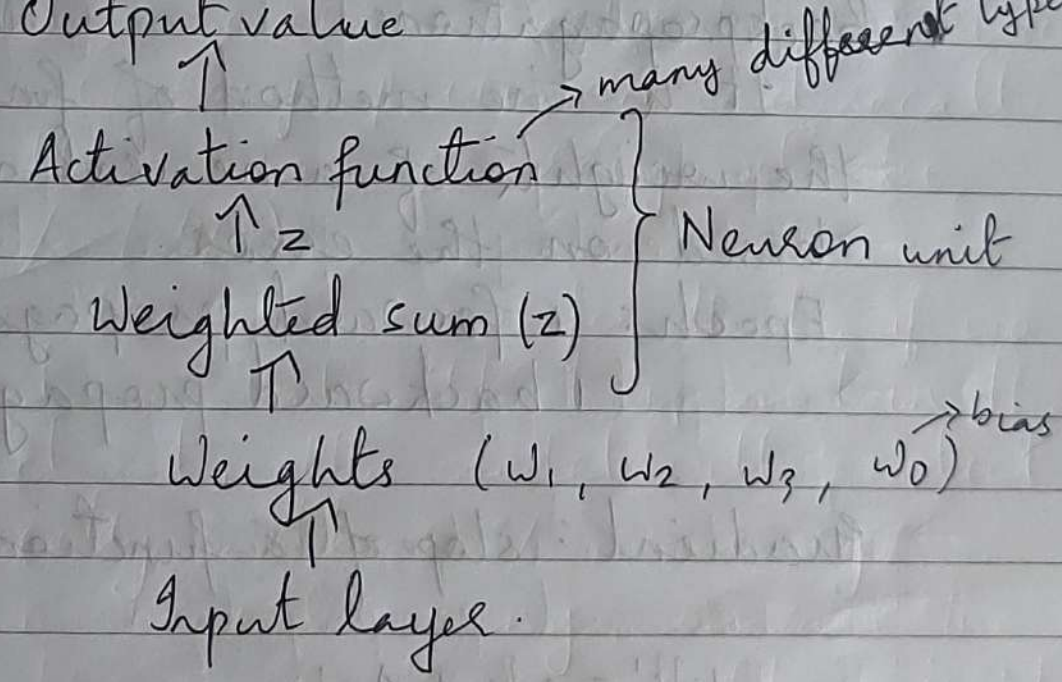
→ ANN are machine learning models that tries to mimics the complex functions of the human brain

→ ~~Simulate our brain~~

Why neural networks?

→ Can be used for wide variety of applications

w_0 → bias, it is the y intercept. To provide some control for the $ equation



if z → Regression function (continuous value)

$z >= 0 \rightarrow 1$

else $z \rightarrow -1$

} step function (Activation function)

↓

Transforms the weighted sum (z) to our desired output.

* For classification, we use sigmoid and softmax activation function

Sigmoid (0,-1):

$$s(z) = \frac{1}{1+e^{-z}}$$

* Backpropagation:
It is the method of fine Tuning the weights of a neural network based on the error.
Epoch: 1 forward propagation and 1 backward propagation.

Gradient: slope of a function.

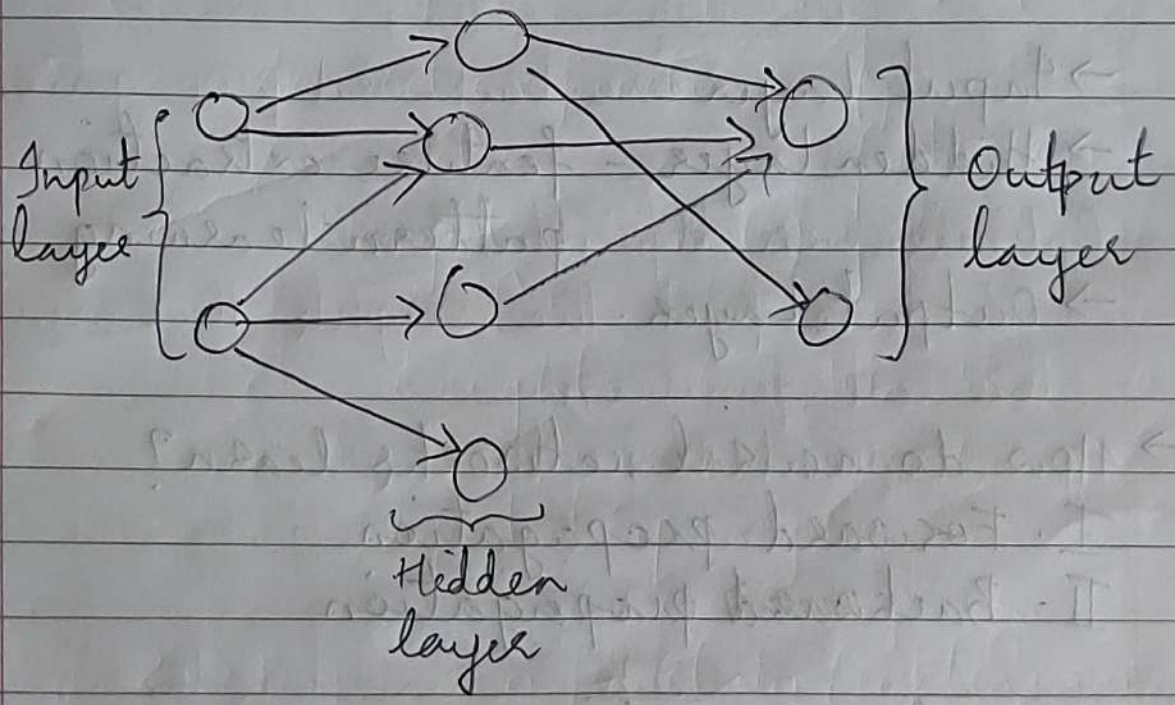
$$W_i' = W_i + \Delta W_i$$

* We find the values of w for which the error is minimal.
* We keep changing the weights iteratively till we get the minimum error.
* We use optimization algorithm to minimize the error.

* Example of optimization algorithm is gradient descent.
* Error / cost / loss

Perceptron → A single neuron structure sideways

Multi-layer Perceptron



* If there are more than two hidden layers then it is called <u>deep neural network</u>.

* Less than two - shallow neural network.

→ Why do we need multiple patterns?
- To get better results.

Hyperparameter: The user chooses:
↳ no. of hidden layers
↳ no. of neurons to include

Components / Architecture of Neural network-

→ Input Layer.
→ Hidden Layer - feature extraction, pattern learning
→ Output Layer.

→ How do neural networks learn?
I. Forward propagation
II. Backward propagation

Forward Propagation:
↳ Each layer accepts input and passes the output to the next layer in the forward direction.
↳ Pre activation: Calculating weighted sum

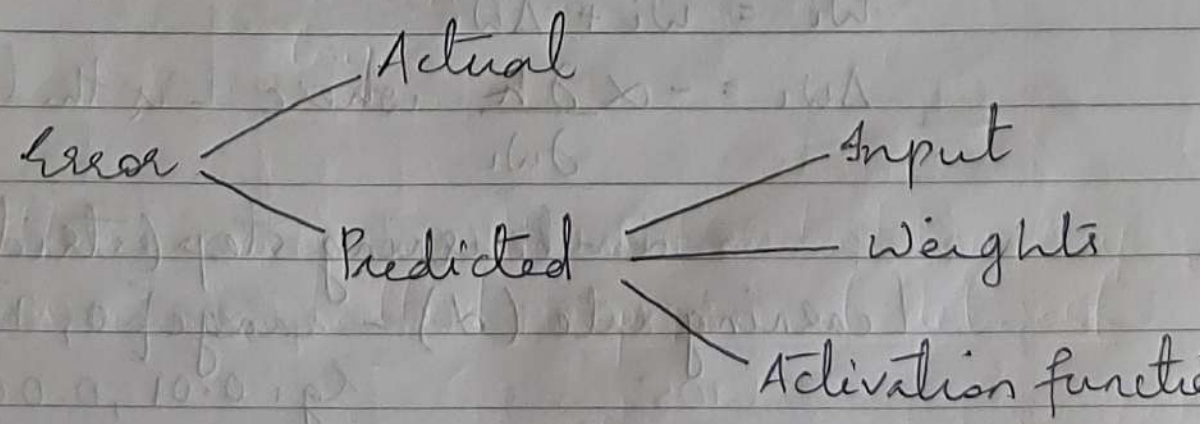
↳ Activation: Passing the sum into an activation function.

How does the computer know there is error in prediction?
↳ Use cost/loss/error function
Eg: Mean square error.

$$MSE = \frac{1}{n} \sum_{i=1}^{n} (Y_i - Y_i)^2$$

$$\frac{(\text{actual value} - \text{predicted value})^2}{n}$$

The error happens because of wrong weights learnt.



Now, we update the weights to fine tune the error.

Back propagation:
→ Trying to update weights to minimize the error.

One epoch - one forward propagation
& one backward propagation

Optimization Algorithm:
Used to find the weights that minimize the cost function.
Eg: Gradient Descent

$$W_i' = W_i + \Delta W_i$$

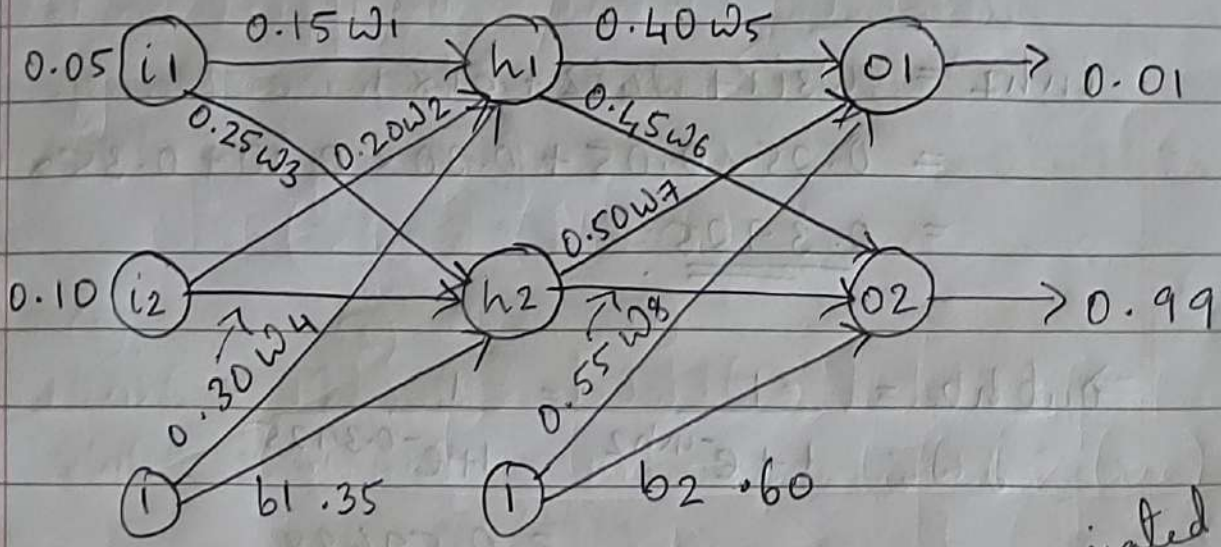
$$\Delta W_i = -\alpha \frac{\partial L}{\partial W_i}$$ where L is the loss function

α - learning step (stride)
Learning rate (α) - range [0, 1]
Eg: 0.01, 0.001, 0.0001

* Lesser the α, the more optimal the solution is.

Assess

How Backpropagation Works?



h1

$in_{h1} = w_1 i_1 + w_2 i_2 + b_1 \times 1$ → value associated with bias

(input to h1)

$= 0.15 \times 0.05 + 0.20 \times 0.10 + 0.35 \times 1$

$= 0.3775$

Assume activation function is sigmoid

$= \frac{1}{1+e^{-z}}$

$$outh_1 = \frac{1}{1+e^{-inh_1}} = \frac{1}{1+e^{-0.3775}}$$

$$= 0.59326.$$

h_2

$$inh_2 = w_3 i_1 + w_4 i_2 + b_1 \times 1$$

$$= 0.25 \times 0.05 + 0.30 \times 0.10 + 0.35 \times 1$$

$$= 0.3925.$$

$$outh_2 = \frac{1}{1+e^{-inh_2}} = \frac{1}{1+e^{-0.3925}}$$

$$= 0.59688$$

Now, we have finished the hidden layer activation.

O_1

$$inO_1 = w_5 outh_1 + w_6 outh_2 + b_2 \times 1$$

$$= 0.40 \times 0.59326 + 0.45 \times 0.59688 \times 0.60 \times 1$$

$$= 1.105$$

$$out_{O1} = \frac{1}{1+e^{-in_{O1}}} = \frac{1}{1+e^{-1.105}}$$

$$= 0.7511 \quad (\text{Expected is } 0.01)$$

O2.

$$in_{O2} = w_7 out_{h1} + w_8 out_{h2} + b_2 \times 1$$

$$= 0.50 \times 0.59326 + 0.55 \times 0.59688 + 0.60 \times 1$$

$$= 1.22484$$

$$out_{O2} = \frac{1}{1+e^{-1.22484}} = 0.7727 \quad (\text{Expected is } 0.99)$$

→ Everything till now is forward propagation.

Now compute the error using MSE

$$MSE = \frac{1}{n} \sum (y_i - y_i)^2$$

$$E_{total} = \frac{1}{n} [(target_{O1} - out_{O1})^2 + (target_{O2} - out_{O2})^2]$$

$$= \frac{1}{2} [(0.01 - 0.7511)^2 + (0.99 - 0.7727)^2]$$

$$= 0.2982 \rightarrow \text{Total Error.}$$

→ Now we use backpropagation to adjust the error.

→ First, we update weight w5, w6, w7, w8 since its closer.

$\frac{\partial E_{total}}{\partial w_5}$ → (What is the effect on total error, when we change w5) (Rate of change of error total w.r.t to w5)

$$\Rightarrow \frac{\partial E_{total}}{\partial w_5} = \underset{①}{\frac{\partial E_{total}}{\partial out_{O1}}} \times \underset{②}{\frac{\partial out_{O1}}{\partial in_{O1}}} \times \underset{③}{\frac{\partial in_{O1}}{\partial w_5}}$$

(Chain Rule)

$$① \frac{\partial E_{total}}{\partial out_{O1}} = \frac{\partial}{\partial out_{O1}} \left[\frac{1}{2} [target_{O1} - out_{O1}]^2 + [target_{O2} - out_{O2}]^2 \right]$$

since the second part doesn't contain out_{O1} we can ignore the rest.

$$= \frac{\partial}{\partial out_{O1}} \left[\frac{1}{2} [target_{O1} - out_{O1}]^2 \right]$$

$$= \frac{\partial}{\partial out_{O1}} \cdot \left[\frac{1}{2} (target_{O1}^2 - 2 \times target_{O1} \times out_{O1} + out_{O1}^2 \right]$$

net = in

$$= \frac{\partial}{\partial out_{01}} \left[\frac{1}{2} (out_{01}^2 - 2 \times target_{01} \times out_{01}) \right]$$

$$= \frac{\partial}{\partial out_{01}} = \left[\frac{1}{2} (2 out_{01} - 2 target_{01}) \right]$$

$$= \frac{\partial}{\partial out_{01}} = out_{01} + target_{01}$$

$$= 0.7511 - 0.01$$

$$= \underline{0.7411}.$$

② $$\frac{\partial out_{01}}{\partial in_{01}} = \frac{\partial}{\partial in_{01}} \left[\frac{1}{1+e^{-in_{01}}} \right] = out_{01}(1 - out_{01})$$

$$= 0.7511 (1 - 0.7511)$$

$$= \underline{0.1869}.$$

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

$\sigma'(x) : \sigma(x)(1 - \sigma(x))$ ← Derivative

③ $$\frac{\partial in_{01}}{\partial w_5} = \frac{\partial}{\partial w_5} \left[w_5 out_{h_1} + w_6 out_{h_2} + b_{2 \times 1} \right]$$

$$= out_{h_1} + 0 + 0$$

$$= out_{h_1}$$

$$= 0.5932.$$

$$\frac{\partial E_{total}}{\partial w_5} = 0.7411 \times 0.1869 \times 0.5932$$
$$= \underline{0.0821}.$$

Gradient Descent

$$w_5' = w_5 + \Delta w_5$$
$$\Delta w_5 = -\alpha \frac{\partial L}{\partial w_5}$$ [L → loss function here it is MSE function]

Assume that $\alpha = 0.5$

$$\Delta w_5 = -0.5 \times \frac{\partial E_{total}}{\partial w_5}$$
$$= -0.5 \times 0.0821$$
$$= -0.041$$

$$w_5' = w_5 + \Delta w_5$$
$$= 0.40 + -0.041$$
$$= \underline{0.359}.$$

↳ updated value of w_5

[w_6, w_7, w_8 can be calculated using this procedure]

w_1

$$\frac{\partial E_{total}}{\partial w_1} = \underset{①}{\frac{\partial E_{total}}{\partial out_{O1}}} \times \underset{②}{\frac{\partial out_{O1}}{\partial in_{O1}}} \times \underset{③}{\frac{\partial in_{O1}}{\partial out_{h1}}} \times \underset{④}{\frac{\partial out_{h1}}{\partial in_{h1}}} \times \underset{⑤}{\frac{\partial in_{h1}}{\partial w_1}}$$

① and ② we have already found out.

$$\frac{\partial E_{Total}}{\partial out_{O1}} = 0.7411 \qquad \frac{\partial out_{O1}}{\partial in_{O1}} = 0.1869$$

③ $$\frac{\partial in_{O1}}{\partial out_{h1}} = \frac{\partial}{\partial out_{h1}} [w_5 out_{h1} + w_6 out_{h2} + b_2 \times 1]$$

$$= \cancel{out_{h1} + 0 + 0}$$

$$= \cancel{out_{h1}} = w_5 + 0 + 0$$

$$= w_5$$

$$= \underline{0.40}.$$

④ $$\frac{\partial out_{h1}}{\partial in_{h1}} = \frac{\partial}{\partial in_{h1}} \left[\frac{1}{1+e^{-\cancel{in_{h1}}}} \right] = out_{h1}(1 - out_{h1})$$

$$= 0.59326(1 - 0.59326)$$

$$= \underline{0.24130}.$$

⑤ $\frac{\partial in_{h1}}{\partial w_1} = \frac{\partial}{\partial w_1}[w_1 i_1 + w_2 i_2 + b_1 \times 1]$

$= 1 i_1 + 0 + 0$

$= \underline{0.05}$

$\frac{\partial E_{total}}{\partial w_1} = 0.7411 \times 0.1869 \times 0.40 \times 0.24130 \times 0.05$

$= \underline{0.0006685}$

$w_1' = w_1 + \Delta w_1$

$\Rightarrow$

$\Delta w_1 = -\alpha \frac{\partial L}{\partial w_1}$

$= -\alpha \times \frac{\partial E_{total}}{\partial w_1}$

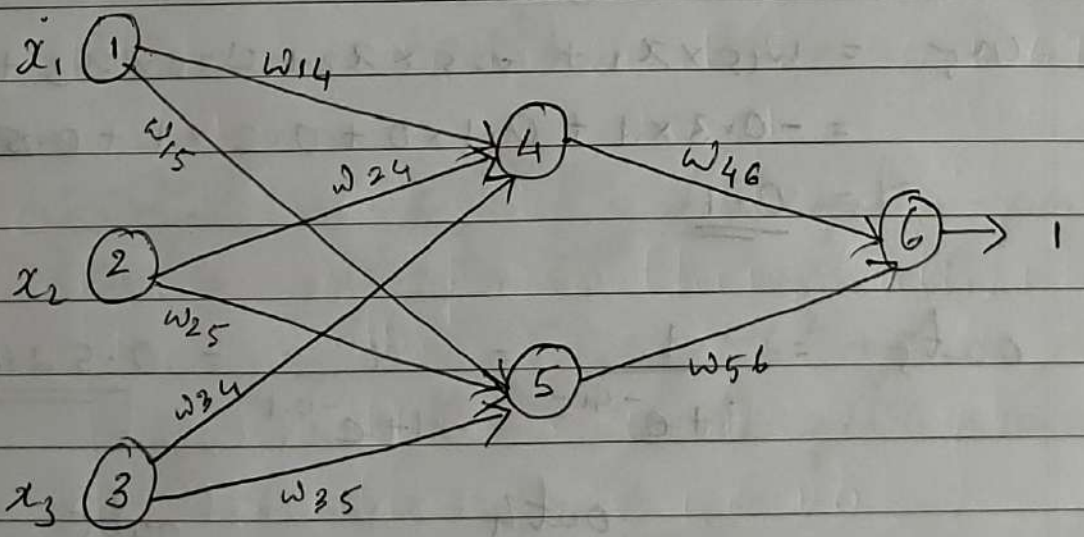
$= -0.5 \times 0.0006685$

$= -\underline{0.00033425}$

$w_1' = 0.15 + -0.00033425$

$= \underline{0.1496}$

Q2. For the given multilayer feed-forward neural network, let the learning rate be 0.9. The initial weight and bias values of the network are given in table along with the first training tuple; $X = \{1, 0, 1\}$ with a class label of 1. Find the updated weights and biases after 1 epoch.



x_1	x_2	x_3	w_{14}	w_{15}	w_{24}	w_{25}
1	0	1	0.2	-0.3	0.4	0.1

w_{34}	w_{35}	w_{46}	w_{56}	θ_4	θ_5	θ_6
-0.5	0.2	-0.3	-0.2	-0.4	0.2	0.1

$\theta_6, w_{46}, w_{56}, \theta_5, w_{14},$ 1.9045837
w_{35}

3.013752707

Forward propagation

$$in_4 = w_{14} \times x_1 + w_{24} \times x_2 + w_{34} \times x_3 + \theta_4$$
$$= 0.2 \times 1 + 0.4 \times 0 + -0.5 \times 1 + -0.4$$
$$= \underline{-0.7}.$$

$$out_4 = \frac{1}{1+e^{-in_4}} = \frac{1}{1+e^{-(-0.7)}} = \underline{0.3318}.$$

$$in_5 = w_{15} \times x_1 + w_{25} \times x_2 + w_{35} \times x_3 + \theta_5$$
$$= -0.3 \times 1 + 0.1 \times 0 + 0.2 \times 1 + 0.2$$
$$= \underline{0.1}.$$

$$out_5 = \frac{1}{1+e^{-in_5}} = \frac{1}{1+e^{-0.1}} = \underline{0.5249}.$$

$$in_6 = w_{46} \times out_4 + w_{56} \times out_5 + \theta_6$$
$$= -0.3 \times -0.7 + -0.2 \times 0.1 + 0.1$$
$$= \underline{-0.1045}.$$

$$out_6 = \frac{1}{1+e^{-in_6}} = \underline{0.4738}. \text{ (Expected 1)}$$

n → no. of output

$$MSE = \frac{1}{n}\left[(target6 - out6)^2\right]$$

$$= \frac{1}{2}\left[(1 - 0.4738)^2\right]$$

$$= (1 - 0.4738)^2 = \underline{0.2768}.$$

Backpropagation

$$\frac{\partial E_{total}}{\partial w_{46}} = \underset{①}{\frac{\partial E_{total}}{\partial out6}} \times \underset{②}{\frac{\partial out6}{\partial in6}} \times \underset{③}{\frac{\partial in6}{\partial w_{46}}}$$

$$① \frac{\partial E_{total}}{\partial out6} = \frac{\partial}{\partial out6}\left[\frac{1}{2}(target6 - out6)^2\right]$$

$$= \frac{\partial}{\partial out6}\left[target6^2 + out6^2 - 2 \times target6 \times out6\right]$$

$$= \frac{\partial}{\partial out6}\left[out6^2 - 2 \times target6 \times out6\right]$$

$$= \frac{\partial}{\partial out6}\left[target6^2 + out6^2 - 2\,target6 \cdot out6\right]$$

$$= 2\,out6 - 2\,target$$

$$= 2\left[0.474 - 1\right]$$

$$= \underline{-1.052}.$$

② $\frac{\partial out_6}{\partial in_6} = \frac{\partial}{\partial in_6}\left(\frac{1}{1+e^{-in_6}}\right)$

$= out_6 - (1 - out_6)$

$= 0.474 - (1 - 0.474)$

$= \underline{-0.052}$

③ $\frac{\partial in_6}{\partial w_{46}} = \frac{\partial}{\partial w_{46}}[out_4 \cdot w_{46} + out_5 \cdot w_{56} + \theta_6]$

$= out_4$

$= \underline{0.331}$

$-1.052 \times -0.052 \times 0.331 = 0.0181$

$w_{46}' = w_{46} + \Delta w_{46}$

$\Delta w_{46} = -\alpha \times \partial E_{total}$

$= -0.9 \times 0.0181$

$= -0.01629$

$w_{46}' = -0.3 + -0.01629$

$= \underline{-0.31629}$

Gradient Descent

→ Optimization algorithm

→ We are trying to find the equation of the best fit line

w_1 ↑ feature ↑

$y = mx_i + c$ ← w_0

this is what we're trying to Predict

→ We use error function to find the error.

$$MSE = \frac{1}{n} \sum_{i=1}^{n} (y_i - y_{ipred})^2$$

Replace y_{ipred} with $mx_i + c$

$$MSE = \frac{1}{n} \sum_{i=1}^{n} (y_i - (mx_i + c))^2$$

↓

We have to minimize this cost → gradient descent.

$m^2 x_i^2 = 2m x_i^2$ (derivative)

$2m^{2-1}$

$2mx_ic = 1x^{1-1}$

→ Predicted value $y_{ipred} = \overset{w_1}{m}x_i + \overset{w_0}{c}$.

$$MSE(\text{Cost fn } L) = \frac{1}{n}\sum_{i=1}^{n}(y_i - y_{ipred})^2.$$

$$= \frac{1}{n}\sum_{i=1}^{n}(y_i - (mx_i + c))^2$$

$$= \frac{1}{n}\sum_{i=1}^{n}(y_i^2 + (mx_i+c)^2 - 2y_i(mx_i+c))$$

$$L = \frac{1}{n}\sum_{i=1}^{n}(y_i^2 + m^2x_i^2 + c^2 + 2mx_ic - 2y_imx_i - 2y_ic)$$

$$\frac{\partial L}{\partial m} = \frac{1}{n}\sum_{i=1}^{n}(0 + 2mx_i^2 + 0 + 2x_ic - 2y_ix_i - 0)$$

$$\frac{\partial L}{\partial m} = \frac{1}{n}\sum_{i=1}^{n} 2x_i(\underbrace{mx_i + c}_{y_{ipred}} - y_i)$$

$$\frac{\partial L}{\partial m} = -\frac{2}{n}\sum_{i=1}^{n} x_i(y_i - y_{ipred})$$

$$m' = m + \Delta m$$

$$\Delta m = m - \alpha \frac{\partial L}{\partial m}$$

$$\frac{\partial L}{\partial c} = \frac{1}{n}\sum_{i=1}^{n}(0+0+2c+2mx_i-0-2y_i)$$

$$\frac{\partial L}{\partial c} = \frac{1}{n}\sum_{i=1}^{n}(2c+2mx_i-2y_i)$$

$$\frac{\partial L}{\partial c} = \frac{1}{n}\sum_{i=1}^{n}2(c+mx_i-y_i)$$

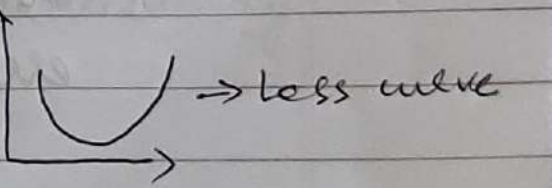
$$\frac{\partial L}{\partial c} = \frac{1}{n}\sum_{i=1}^{n}2((\underbrace{mx_i+c}_{y_{ipred}})-y_i)$$

$$\checkmark \quad \frac{\partial L}{\partial c} = \frac{-2}{n}\sum_{i=1}^{n}(y_i - y_{ipred})$$

$$\left.\begin{aligned}\checkmark \quad c' &= c + \Delta c \\ &= c - \alpha\frac{\partial L}{\partial c}\end{aligned}\right\}$$

~~Observations~~

* if we choose larger α or learning rate, gradient descent can overshoot.
* if we choose lower α, gradient descent will take small steps to reach local minima and will take a longer time to reach minimum.



* Binary – Sigmoid
* Multiclass – Softmax
* Regression – linear regression

} Activation function

Variants of gradient descent:

→ Stochastic – one forward propagation, computing the error and one backpropagation using the computed error to back-propagate and update the weights.

→ Mini-batch

→ Batch

→ If there are 4 samples, so 4 backpropagation will be there.

→ finding the average error to back-propagate. If there are two samples and combine the error and backpropagate. Samples will be divided into batches. No. of weight updation will be less.

→ Weight will be updated at the end of the iteration. Only one backpropagation will be there using the combined error.

Vanishing and Exploding gradient problem

Vanishing gradient:

→ This happens with deep neural network (more than 2 hidden layers).
→ When the updated weight is very minimal, it is negligible.
→ When the gradient is very minimal, weight updation is negligible.
→ If its a ~~non~~ complex problem, we need to use deep neural network.
→ It often happens when we use sigmoid and tanh activation functions.

Exploding gradient:
→ When the weight updation is too large, it is called exploding gradient
→ It happens with relu activation function.

Activations Function
→ Transform the weighted sum to required output.
→ Purpose: To put some non-linearity to the neural network.

Types of activation fn:

→ Binary step function

$$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$$

- Cannot use for multi-class classification problem

→ Linear Activation function

$$f(x) = x$$

- Weighted sum is the output.
- Used for regression function. (continuous values)
- Will not be using in hidden layers because the derivative will be 0.

→ Non-Linear Activation function:

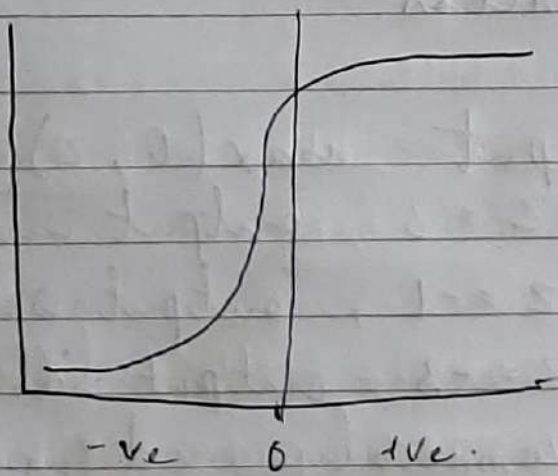
1. Sigmoid activation

$$= \phi(z) = \frac{1}{1+e^{-z}}$$

- The output will range from 0 to 1.
- Value of x Depends on the value of z.
- Used for binary classification problems.

x		Output.
3	Sigmoid	0.95
1.75	⟶	0.85
-2		0.12
0.5		0.62.

It is never a probability curve.

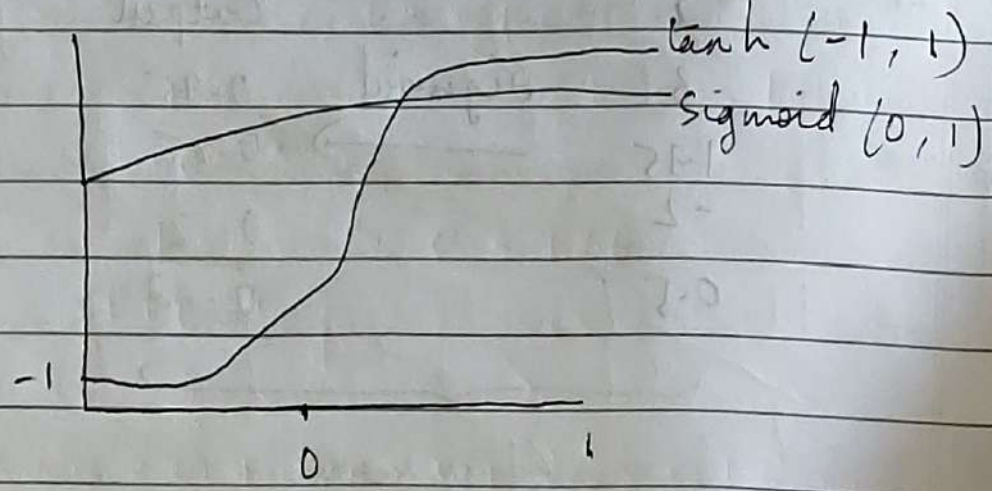


→ If value of z is -ve, the fn will push it toward 0 and if its +ve, the value is pushed toward 1.

2. Tanh function:

$$= \left(\frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}\right)$$

→ Output ranges from -1 to +1.
→ Used for binary classification.



3. Relu function:

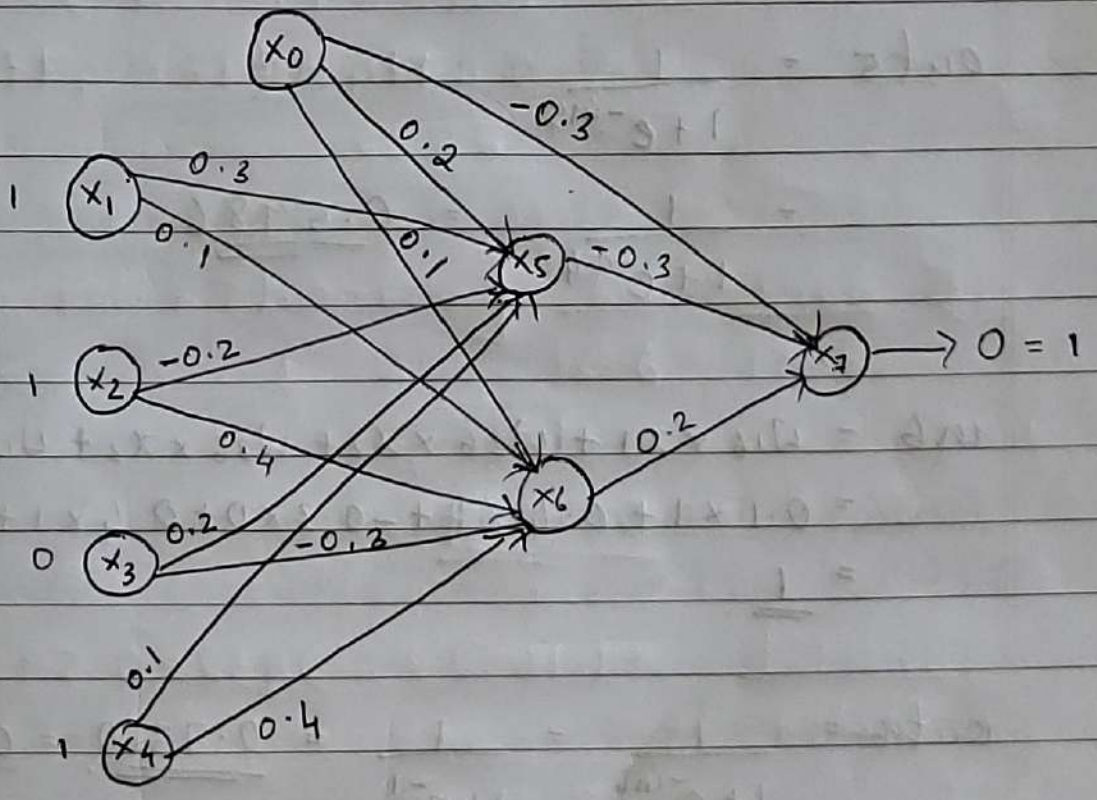
→ Output = max(0, z)
if z = 3, output will be 3.
z = -1, output will be 0.
z = -3, output will be 0.
→ Most commonly used hidden layer activation function.

Aswin Jayakumar

Vismaya Dinesh

Assignment

Q. For the ANN shown below, compute the updated weights of w_{57}, w_{67} and w_{07}. Assume learning rate = 0.8, activation function = sigmoid, output $O = 1$.



x_1	x_2	x_3	x_4
1	1	0	1

Forward Propagation:

$in5 = w15 \times x1 + w25 \times x2 + w35 \times x3 + w45 \times x4 + b05$

$= 0.3 \times 1 + -0.2 \times 1 + 0.2 \times 0 + 0.1 \times 1 + 0.2$

$= \underline{0.4}$.

$out5 = \frac{1}{1 + e^{-in5}}$.

$= \frac{1}{1 + e^{-0.4}} = \underline{0.5986}$.

$in6 = w16 \times x1 + w26 \times x2 + w36 \times x3 + w46 \times x4 + b06$

$= 0.1 \times 1 + 0.4 \times 1 + -0.3 \times 0 + 0.4 \times 1 + 0.1$

$= \underline{1}$.

$out6 = \frac{1}{1 + e^{-in6}} = \frac{1}{1 + e^{-1}} = \underline{0.7310}. = \underline{0.7311}$.

$in7 = w57 \times out5 + w67 \times out6 + b07$

$= -0.3 \times 0.5986 + 0.2 \times 0.7311 + -0.3$

$= -0.17961 + 0.14622 - 0.3$

$= \underline{-0.3334}$.

$$out_7 = \frac{1}{1+e^{-in_7}} = \frac{1}{1+e^{-0.3334}} = \underline{0.4174}. \text{ (Expected 1)}$$

$$MSE = \frac{1}{2n}(target_7 - out_7)^2$$

$$= \frac{1}{1}(1 - 0.4174)^2$$

$$= \underline{0.3394}. \rightarrow E_{total}$$

Backpropagation

w_{57}.

$$\frac{\partial E_{total}}{\partial w_{57}} = \underset{①}{\frac{\partial E_{total}}{\partial out_7}} \times \underset{②}{\frac{\partial out_7}{\partial in_7}} \times \underset{③}{\frac{\partial in_7}{\partial w_{57}}}$$

$$① \frac{\partial E_{total}}{\partial out_7} = \frac{\partial}{\partial out_7}\left[(target_7 - out_7)^2\right]$$

$$= \frac{\partial}{\partial out_7}\left[target_7^2 + out_7^2 - 2target_7 - 2(target_7 \times out_7)\right]$$

$$= (2out_7 - 2target_7)$$

$$= -2(target_7 - out_7)$$

$$= -2(0.2339)$$

$= -2(1 - 0.4174)$

$= -1.1652.$

② $\frac{\partial out7}{\partial in7} = \frac{\partial}{\partial in7}\left[\frac{1}{1+e^{-in7}}\right]$

$= out7 * (1 - out7)$

$= 0.4774 * (1 - 0.4774)$

$= 0.2432.$

③ $\frac{\partial in7}{\partial w57} = \frac{\partial}{\partial w57}\left[w57 \times out5 + w67 \times out6 + w07\right]$

$= out5$

$= 0.5987.$

$\frac{\partial Etotal}{\partial w57} = -1.1652 \times 0.2432 \times 0.5987$

$= -0.1697.$

$w57' = w57 + \Delta w57$

$\Delta w57 = -\alpha \times \frac{\partial Etotal}{\partial w57}$

$= -0.8 \times -0.1697$

$= 0.13576$

$$W_{57}' = -0.3 + 0.13576$$
$$= -0.16424$$

W_{67}.

$$\frac{\partial E_{total}}{\partial W_{67}} = \frac{\partial E_{total}}{\partial out_7} \times \frac{\partial out_7}{\partial in_7} \times \frac{\partial in_7}{\partial W_{67}}$$

$$\frac{\partial E_{total}}{\partial out_7} = -1.1652 \qquad \frac{\partial out_7}{\partial in_7} = 0.2432.$$

$$\frac{\partial in_7}{\partial W_{67}} = \frac{\partial}{\partial W_{67}} \left[W_{57} \times out_5 + W_{67} \times out_6 + W_{07} \right]$$
$$= out_6$$
$$= 0.7311.$$

$$\frac{\partial E_{total}}{\partial W_{67}} = -1.1652 \times 0.2432 \times 0.7311$$
$$= -0.2072.$$

$$W_{67}' = W_{67} + \Delta W_{67}$$
$$= 0.2 + (-0.8)(-0.2072)$$
$$= 0.2 + 0.16576$$
$$= 0.36576.$$

W_{07}

$$\frac{\partial E_{total}}{\partial W_{07}} = \frac{\partial E_{total}}{\partial out_7} \times \frac{\partial out_7}{\partial in_7} \times \frac{\partial in_7}{\partial W_{07}}$$

$$\frac{\partial in_7}{\partial W_{07}} = \frac{\partial}{\partial W_{07}} \left[W_{57} \times out_5 + W_{67} \times out_6 + W_{07} \right]$$

$$= \underline{1}.$$

$$\frac{\partial E_{total}}{\partial W_{07}} = -1.1652 \times 0.2432 \times 1$$

$$= \underline{-0.2834}.$$

$$W_{07}' = W_{07} + \Delta W_{07}$$
$$= -0.3 + (-0.8)(-0.2834)$$
$$= -0.3 + 0.22672$$
$$= \underline{-0.07328}.$$

	old value	updated value
W_{57}	-0.3	-0.16424
W_{67}	0.2	0.36576
W_{07}	-0.3	-0.07328

4. Leaky RELU:
- Attempt to solve the dying RELU problem.
- Helps increase the range of RELU.
- A small portion of negativeness is bypassed.
- leaky factor $a = 0.01$
- When a is not 0.01, it is called Randomized RELU.

Hidden layers activation fns:
- RELU (Rectified Linear Activation)
- Logistic (Sigmoid)
- Tanh (Hyperbolic tangent)

Output layer activation fns:
- Linear
- Sigmoid
- Softmax.

→ For multi-class classification problems, we use sigmoid softmax.
→ For multi-label classification problem, we use sigmoid.

Softmax fn $= \frac{e^{z_j}}{\sum^{K} e^{z_k}}$ for $j = 1 \ldots K$

→ It can The output can be also interpreted as a probability distribution.
→ Whichever class having the higher probability will be the output or predicted class.
→ Sum of the probabilities should be 1.

Loss / Cost Functions:

→ Choice of cost function depends on classification or regression

→ For regression, MSE (Mean Squared Error) is used.

→ Mean Absolute Error (MAE), also used for regression.
↳ This is used when there are outliers in the data.

$$MAE = \frac{1}{n} \sum_{i=1}^{n} |y_i - y_{ipred}|$$

↳ MAE treats all errors equally.

→ For binary classification problem, Binary cross Entropy (Log Loss) is used

↳ $\frac{1}{n}\sum_{i=1}^{n}\left[y_i \log(y_{ipred}) + (1-y_i)\log(1-y_{ipred})\right]$

→ For multi class classification, Categorical cross entropy. (one-hot encoding is used)

↳ $-\sum_{i=1}^{n} y_i \log(y_{ipred})$

→ Another loss function is Sparse Categorical cross Entropy, used for multi-class classification. But instead of one-hot encoding, the labels are represented as integers itself.

↳ Used when there is a huge number of classes.

* One-hot encoding is used to eradicate any differences in between the class labels.

1 - 1 0 0 3 - 0 0 1

2 - 0 1 0

→ Loss function
→ Optimizers
→ Evaluation metrics
} needed for defining a model.

→ Kullback - Leibler Divergence (KL Divergence).
↳ Used for unsupervised learning.
↳ finds the difference b/w two probabilities.

→ Hinge Loss.
↳ Used for SVM required tasks.
↳ where margin maximization is important.

Optimizers.
→ Used to adjust weights.
→ Eg: Gradient descent.
→ Stochastic Gradient descent (SGD)
↳ Used when data is plentiful and simple.
↳ Slow convergence but leads to good generalization. It should fit different or various data sets.

→ SGD with Momentum.
↳ Used when the optimization is slow or get stuck in local minima

* We are tryying to find global minima with lowest error.

→ RMSprop (Root Mean Square)
↳ Great for mini batch training and noise.
→ Adam (Adaptive Moment Estimation)
↳ General-purpose and widely used.

→ Adagrad
↳ Useful for sparse data
→ Adadelta - for preventing slow downs.
→ Nadam - when you need fast convergence.

Module - II.

Deep Learning:
→ Uses many ANNs to learn from data.
→ Why DL?
↳ Can learn complex data
↳ Large dataset training
↳ Data-driven learning.
↳ Automatically learns from data. Manual assistance is very low.

Convulational Neural Network (CNN).
→ Working with images and manually extracting features from them is very tedious.
→ That is why automatic extraction of features was introduced.

→ CNN model works in two steps:
↳ Feature extraction
↳ Prediction

→ Fully connected Layer / Dense: Every neuron is connected to every other neuron in the next layer.

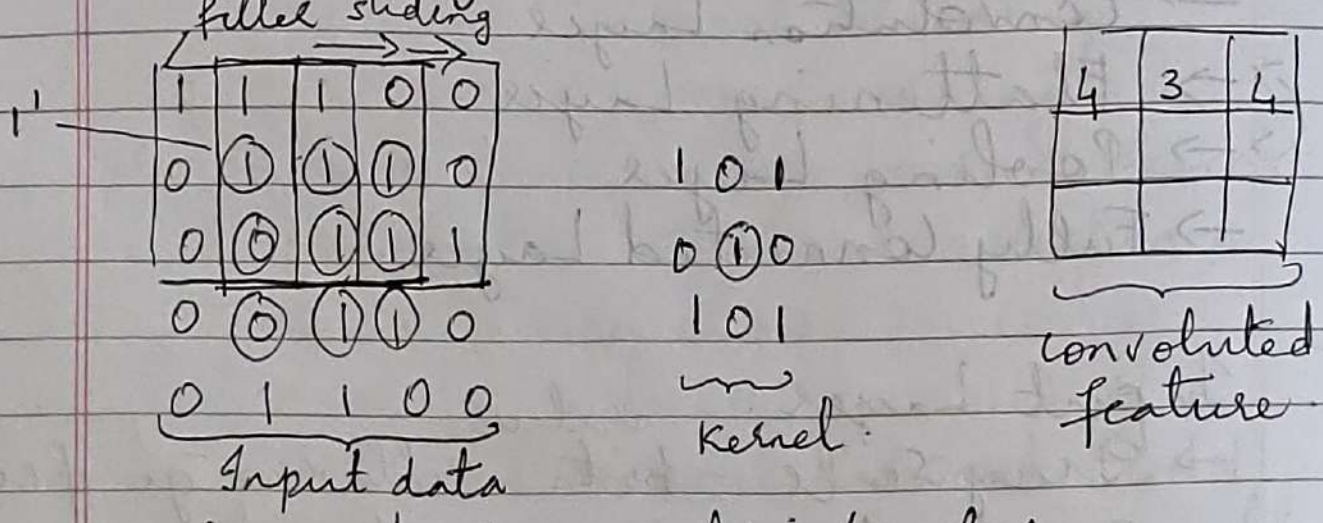
to make.

Building blocks of CNN:
→ Input Layer
→ Convolution Layer
→ Flattening Layer
→ Pooling Layer
→ Fully Connected Layer.

Input Layer: values
→ Gray scale - bits will range from 0 - 255. 8 bits are stored.
00000000 - 0
11111111 - 2^8 - 255.
→ Black and white binary pixel - 0/1
→ If its a color image, a full red image will have pixels indicating the shade of red from 0 to 255.

→ Convulational Layers:

- Applies filters over the image (kernel or masks) to extract features.
- Output of ~~this~~ This will be edges, corner or texture.
- Sliding a kernel over the input.



→ element-wise multiplication

$1_1' = 1\times1 + 1\times0 + 1\times1 + 0\times0 + 1\times1 + 1\times0 + 0\times1 + 0\times0 + 1\times1$

$= \underline{4}$

$1_2' = \underline{3}$

$1_3' = \underline{4}$

Why should we move this kernel?
→ Based on type values in the mask, we will different types features.
extract
→ Sobel vertical and sobel horizontal are edge detection masks or kernel.

Padding:
→ Entire image in the boundary will have zero. - Zero-padding.
→ This is done to make the response map and the actual image the same size.
→ convolved or convulated feature.

Stride:
→ How quickly window/filter/kernel slides
→ Stride 2 means, window moves by 2 pixels at a time

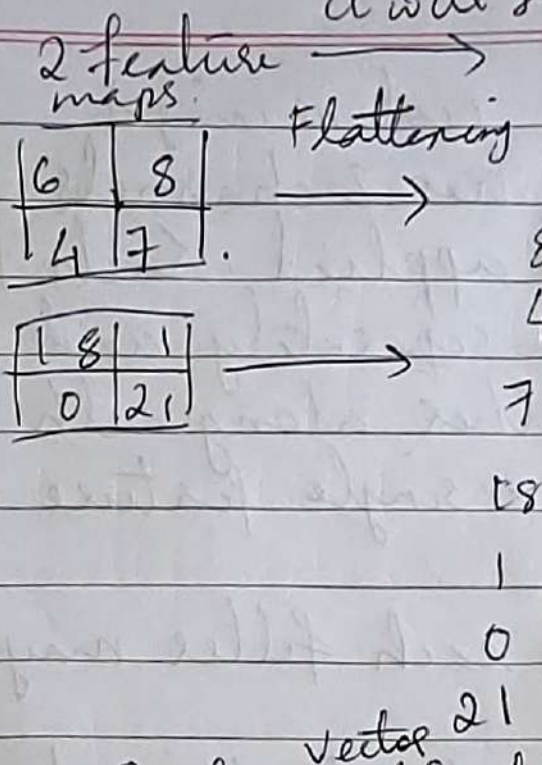
* Filter bank: A set of filters are applied on the image to extract features.

→ Activation function
↳ ReLU : Rectified linear unit.
It sets all negative values to zero.
$R(z) = \max(0, z)$
Introduces non-linearity.
and output is a feature map.

→ Pooling Layer:
↳ Trying to aggregate features
↳ Downsamples the feature maps to reduce dimensionality.
↳ Here also a mask is applied
↳ max pooling and average pooling.
↳ A blank mask is applied.

→ Flattening Layer:
↳ Now we have many feature maps after pooling.
↳ The 2D array is converted to a 1D array vector
↳ All the 2D feature maps are converted.

it will get appended.



vector

↳ The feature that we get will become the input to ANN.
↳ It will pass through the fully connected layer.

* Feature extraction is automated in CNN.
* For text input, it will be 1D array. So the mask size will also be one dimensional.

Convolution in colour image:

* Multi channel image will have multiple filter bank (no. of filters according to the no. of channels) and the outcome will be a single feature map.

(colour image)

* If an image has 3 channels, 3 kernels will be ~~2d~~ applied. Each kernel is calculated separately and is added together along with a bias to produce a single feature map.

* The values in each filter may not be same.
* Algorithm will decide the values in the filter. This is similar to the weights in ANN. The algorithm will learn on its own if there is any change.

What all parameters determine the convolution operation?
→ Size of the i/p image
→ Stride
→ padding.

To find the size of convolution image:

$$W_{out} = \frac{W - F + 2P}{S} + 1$$

P - no. of zero-padding
S - stride
F - filter size
W - size of the i/p image

So if there are n number of channels,
then $W_{out} \times n$ is used.

Pooling layer.

$$W_{out} = \frac{W - F}{S} + 1$$

W - response of the convolution
F - Pooling filter

Conv 1

Q: Input size = $28 \times 28 \times 1$.
Four hyperparameter
No. of kernels, $k = 16$
Size of each kernel = 5 (F)
Stride, $S = 1$
Paadding, $P = 2$.

$$W_2 = \frac{W - F + 2P}{S} + 1$$

$$= \frac{28 - 5 + 2 \times 2}{1} + 1$$

$$= 28$$

$$H_2 = 28.$$

Output will be $28 \times 28 \times 16$. } input to the pooling layer.
↓
because 16 filters are applied.

Pool 1

Pooling: $\frac{W-F}{S}+1 = \frac{28-2}{2}+1$

$= 14.$

$F = 2$ because its given that a 2×2 pooling filter.

So output of pooling is $14 \times 14 \times 16$.

This will the input to the next convolutional layer: Conv 2.

$K = 32$

size(k) = 5

$S = 1$

$P = 2.$

$$\frac{W-F+2P}{1}+1 = \frac{14-5+2(2)}{1}+1$$

$= 14.$

Output $= 14 \times 14 \times 32.$

↓

32 filters are applied

Pooling 2. F = 2, S = 2.

$$\frac{W - F}{S} + 1 = \frac{14 - 2}{2} + 1 = 7.$$

Output = 7 × 7 × 32. [32 7×7 images will be the input to the next layer, flattening layer].

→ The flattening layer will have 7×7×32 elements because it is a one-dimensional array.

→ This will be the input to the fully connected layer.

Computation of number of parameters learnt:

For a convolution layer:

→ Will depend on the filter size.

Parameters = (Fw × Fh × no. of channels + bias) * K.

Example, if its a colour image

Parameters = (3 × 3 × 3 + 1) × 5 = 140.

A 3 × 3 color image with 3 channels and 5 3×3 filters are applied.

Parameters → weights

For a pooling layer:
→ No parameters are learned because we are applying blank filters.

For a fully connected layer:

Parameters = (input units × output units) + output units

every neuron is associated with a bias.

128 input 64 output

Parameters = (128 × 64) + 64

= 8256.

Q. Compute the output shape and #parameters of each layer.

Layers:	Kernel size	No. of Kernels	Stride
Image (32×32×1)			
Convolution	5×5	6	1
Avg. pooling	2×2	6	2
Convolution	5×5	16	1
Avg. pooling	2×2	16	2
Convolution	5×5	120	1
FC (Neurons = 84)			
FC (Neurons = 10)			

* If padding is not given, it will be 0.

Conv 1

$$W_{out} = \frac{W - F + 2P}{S} + 1$$

$$= \frac{32 - 5}{1} + 1 = 28.$$

Output of conv1 = $28 \times 28 \times 6$.

Pool1

$$W_{out} = \frac{W - F}{S} + 1$$

$$= \frac{28 - 2}{2} + 1 = 14.$$

Output of pool1 = $14 \times 14 \times 6$.

Conv2

$$W_{out} = \frac{W - F + 2P}{S} + 1$$

$$= \frac{14 - 5}{1} + 1 = 10$$

Output of conv2 = $10 \times 10 \times 16$.

Pool2.

$$W_{out} = \frac{W - F}{S} + 1$$

$$= \frac{10 - 2}{2} + 1 = 5$$

Output of Pool2 = $5 \times 5 \times 16$

Conv3.

$$W_{out} = \frac{W - F + 2P}{S} + 1$$

$$= \frac{5 - 5}{1} + 1 = 1$$

Output = $1 \times 1 \times 120$.

Parameters learnt for Conv1.

$= (5 \times 5 \times 1 + 1) * 6$

$= 156$.

Parameter learnt for conv2:

6 → no. of channel of the previous layer

$= (5 \times 5 \times 6 + 1) \times 16$

$= 2416$.

Parameter learnt for conv3:

$= (5 \times 5 \times 16 + 1) \times 120$

$= 130840$. $= 48120$.

Parameters learnt for FC-1

$= (120 \times 84) + 84$

$= \underline{10164}$.

Parameters learnt for FC-2.

$= (84 \times 10) + 10$

$= \underline{850}$.

Total no. of parameters $= 156 + 2416 + 48120 + 10164 + 850$

$= \underline{61706}$.

* For exam, write the entire table with output feature map and no. of parameters.

* Padding $= (F-1)/2$.
  ↳ to find the padding size.
  F → filter size.

Q. Layers

Image (32×32×3)	Kernel size	No. of kernel	stride	Padding
Conv 1	11×11	96	4	Same
Max. Pooling	2×2	?	2	
Conv 2	5×5	256	1	Same
Max. Pooling	2×2	?	2	
Conv 3	3×3	384	1	Same
FC (Neurons = 84)				
FC (Neurons = 10)				

Conv 1.

$$= \frac{W - F + 2P + 1}{S}$$

$P = (F-1)/2$

$= (11-1)/2 = \underline{5}$

$$= \frac{32 - 11 + 2(5) + 1}{4}$$

$= 8.75 = \underline{8}.$

Output = $8 \times \underline{8} \times 96$.

Max Pooling 1.

$$= \frac{W - F}{S} + 1$$

$$= \frac{8 - 2}{2} + 2 = 4$$

Output = $4 \times 4 \times 96$.

Conv 2.

$$= \frac{W - F + 2P}{S} + 1$$

$P = (F-1)/2$
$= (5-1)/2 = 2$.

$$= \frac{4 - 5 + 2(2)}{1} + 1$$

$$= 4.$$

Output = $4 \times 4 \times 256$.

Max Pooling 2.

$$= \frac{W - F}{S} + 1 = \frac{4 - 2}{2} + 1 = 2$$

Output = $2 \times 2 \times 256$.

Conv 3.

$$\frac{W - F + 2P}{S} + 1$$

$$P = (F-1)/2$$
$$= (3-1)/2$$
$$= 1$$

$$= \frac{4 - 3 + 2(1)}{1} + 1$$

$$= 2.$$

Output $= 2 \times 2 \times 384$.

Parameters learnt in Conv1:
$= (11 \times 11 \times 3 + 1) * 96.$
$= 34944.$

Parameters learnt in conv2:
$= (5 \times 5 \times 96 + 1) * 256$
$= 614656.$

Parameters learnt in conv3:
$= (3 \times 3 \times 256 + 1) * 384$
$= 885120.$

Parameters learnt in FC-1

= ~~(384 × 84) + 84~~

= (2 × 2 × 384 * 84) + 84 (output of conv3).

= 129108.

Parameters learnt in FC-2

= (84 × 10) + 10

= 850.

Total parameters learnt

= 34944 + 614656 + 885120 + 129108 + 850

= 1664678.

Sparse connectivity in CNN:

→ In convolution layer, if usual we're using a 3×3 filter, then we are trying to learn 9 weights only.

→ This is called Sparse connectivity in CNN

→ In ANN, it is fully connected layer and depends on the input.