Deep learning

1) Subset of machine learning
2) Neural networks with three (or) more layers.
3) Imitates how human Process data and make decisions.
4) Exponential growth in the last few years.
5) powered by advances in large scale data processing and inference.

Application of deep learning.

* NLP
* Speech recognition & synthesis
* Image ~~reg~~ recognition.
* Self driving cars.
* Customer experence, healthcare, and robotics.

Linear Regression:-

Linear Regression is the basic statistical concepts that is used in machine learning It forms a key foundation for deep learning.

linear regression is a linear model that Explains the ~~ex~~ relationship between two (or) more variable. we have a dependent variable y and an independent variable x.

The model provides an Equation to compute the value of y based on the value of x. To compute this. we need to have an

constant called a, which is slope & an intercept which is B.

The formula for computing y is

$$y = ax + b$$

This Provides a linear relationship between y and x

In reality the relationship may not be ~~not~~ be perfectly linear. There would be Error in prediction time.

* Linear regression is used in "regression problem to predict continous variable.
* It can be applied for multiple independent variable like $x_1, x_2 \ldots x_n$.

In which there will be an Equivalent slope of values $A_1, A_2 \ldots A_n$.

Building a linear Regression model.

* Find values for slope and intercept
* use known values of x & y (multiple sample)
* multiple independent variable make it complex.

A relative technique that is most used in deep learning is logistic Regression.

It is binary model.

It say Relationship between two or more variable.

the output y in this case is either zero or one.

$y = f(ax+b)$.

The formula is similar to linear regression except that we use an additional activation function called f to convert the continuous variable coming out of $ax+b$ into boolean value of zero (or) one.

There are multiple options & variation of function f. The above can be again extend to multiple independent variables $x_1, x_2 \ldots x_n$

$$y = f(a_1x_1 + a_2x_2 + \ldots a_nx_n + b)$$

Analogy for deep learning:-

* Deep learning is a complex and iterative process that require series of trails to narrow done the parameter for the model.

* It start's actually with random initialization of model parameters and works towards the right value of these parameter by trail and error.

In case of linear regression model parameter are A & B

The values of A & B of model that can determines the relationship between x and y.

Ex:- we will try to A, B for an example $10 = 3A + B$.

$10 = 3a + b$

Trail	a	b	3a+b	Error
1	1	1	4	6
2	4	3	15	-5
3	2	2	8	2
4	3	2	11	-1
5	2	4	10	0

we randomly initial the values A and B by trails By increasing trails we reduces the Error and got the A & B values.
we will only reduce Error to make things good.

Note:- The Equation may not provide a zero Error. But in most cases, we are tying to minimize the error to an acceptable value.

While the mathematical approach of using sample values and reducing the Equation work well with a small number of indenpent variable. But it becomes incredibly complex when the number of variables is really high.

What is a perceptron?
An algorithm for Supervised learning for binary classification.

The perceptron is the unit for learning in an artifical neural network.
An perceptron represent an algorithm for Supervised learning in ANN

- It Resembles a cell in the human brain.
- It represent a single cell in nebral network.
- It Build on logistic regression.

formula :-

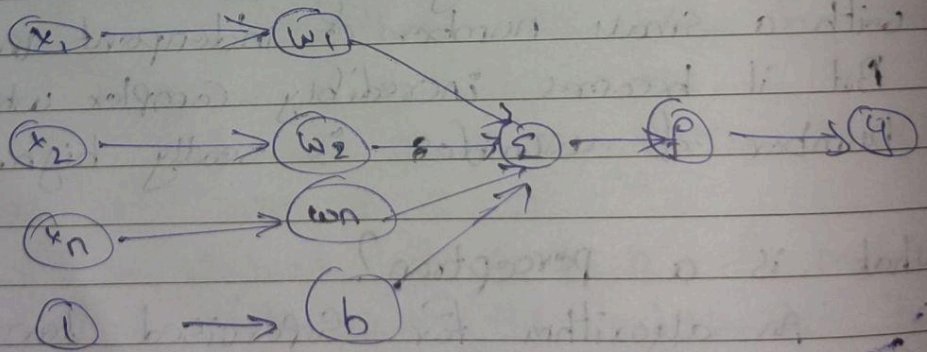
Logistic regression $\rightarrow y = f(a_1x_1 + a_2x_2 + \dots + a_nx_n + b)$

Perceptron Eq. $\rightarrow y = f(w_1x_1 + w_2x_2 + \dots + w_nx_n + b)$

we replace the slope a, with weight w, and ~~an~~ intercept b with bias called b.

w weight
b Bias
f Activation function

$f(x) = 1$ if value is > 0
$f(x) = 0$ if value is ≤ 0

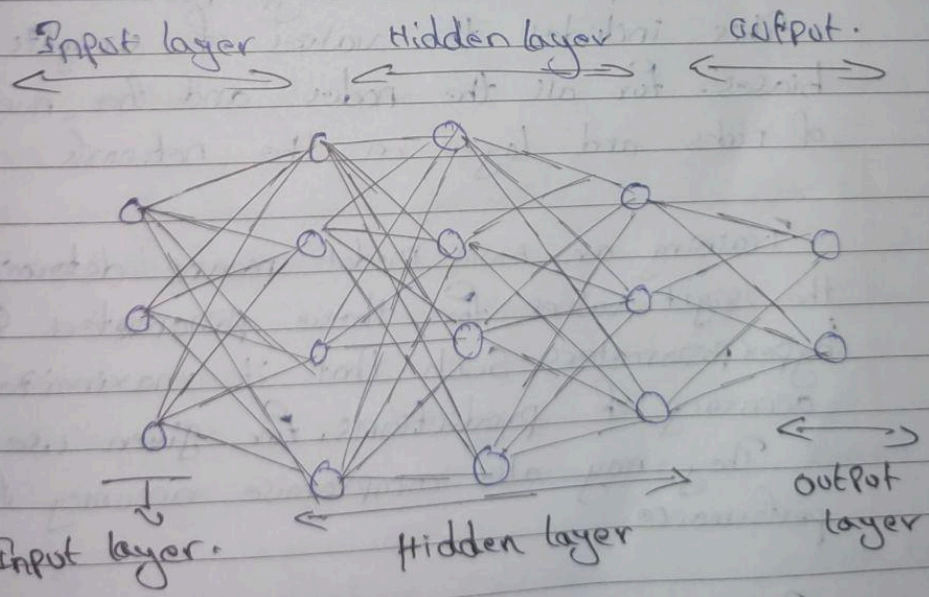


classmate
Date
Page

Artifical Neural Network (ANN)

A network of perceptron, modeled after the human brain.

* perceptron are called nodes in the neural network.
* using the network of perceptron we created ANN.
* nodes are organized ~~as~~ into multiple layers. in a neural network.
* A deep neural network usually has three or more layers.
* Each node has its own weights, biases and activation function ~~as~~.
* Each node is connected to next layer forming a dense network.
* The nodes within a layer are not connected with each other.



How ANN works?

* Input (independent variables) are sent from input layer.
* Data may be pre-processed before using them.
* Input passed on the nodes in the next hidden layer.
* Each node computes its output based on its weights, biases and activation function.
* Nodes output is then passed on as inputs to the next layer.

Training an ANN:

An ANN is created through model training process.

What does training mean?

A neural network model is represented by a set of parameter and hyperparameters. This includes the values of weights and biases for all the nodes and the number of nodes and layer in the network.

Training an ANN model means determining the right values for these parameters & hyperparameters. Such that it maximizes the accuracy of predictions for given use case. They may also compromise accuracy for performance.

Input, weight & biases might be n-dimensional array.

Training Process:-

* use training data
* create network architecture with intuition
* Start with random values for weight and biases.
* minimize Error in Predicting known output for from inputs.
* Based on Error we will adjust the weights and biases until Error is minimized.
* Improve model by adjusting layers, nodecounts and other hyper parameters.

Neural network architecture:-

* The input layer:-

Before knowning about input layer we try to know vector.

A vector is an ordered list of numeric values

The input to deep learning is usually a vector of numeric values.

A vector can be set to be a tuple of one or more values

Ex:- (1.0, 2.5, 3.9)

* vector are usually defined using a nompy array
* It represents feature variable or independent variable that are used for prediction as well as trainning.

* Input data set that are available from the real world for machine learning contain Samples and features
* A Sample is one instance of real word Example It is equivalent to records in a database.
* feature are individual attributes in the Sample

Samples and features:-

features → Age, Salary, Service

Employee ID	Age	Salary	Service	
1001	28	40,000	5	←
1002	47	60,000	20	← → Samples
1003	35	55,000	10	←

for text data Each document is Sample and it's numeric representation becomes its features.

for Image Each image is a sample and its pixel representation becomes it features

Input preprocessing
* Similar to regular machine learning input data need to be pre-processed and transformed to appropriate numeric representation before they are fed into a neural network.
They are popular preprocessing technique is used.

Input type	Preprocessing Needed.
Numeric	Centering & Scaling.
Categorical	Integer Econding, one-hot Encoding
Text	TF-IDF, Embeddings
Image	Pixel - RGB representation
Speech	

TF → Text Frequency
IDF → Inverse document frequency

There are many more advanced preprocessing techniques that are applied to data to prepare them for deep learning.

Example for how the input data is preprocessed.

Raw data

Age	Salary	services
28	40,000	5
47	60,000	20
35	55,000	10

→ Centered & scaled

x_1	x_2	x_3
-1.10	-1.37	-1.07
1.32	0.93	1.34
-0.21	0.39	-0.24

Transposed

x_1	-1.1	1.32	-0.21
x_2	-1.37	0.98	0.39
x_3	~~-0.97~~		
	-1.07	1.34	-0.27

~~for~~ ~~Feats~~

Here we have an How Employee data can be processed, Here we ~~to~~ represent Each attribute like Salary, age & service as individual feature like x_1, x_2 & x_3 we will center and scale the value to normalize them to standard values or standard ranges.

Optionally we will also transpose them Each sample as column

once input is ready it can be passed to deep learning networks for learning.

Hidden layers:-

Hidden layers in a neural network form the brain where knowledge is acquired to use.

An ANN can have one (or) more hidden layer the more, the number of layers, the deeper the network is each hidden layer, hidden layer can have one (or) more nodes, Typically node count is configured in the range of two power N

Example counts may be 8, 16, 32, 64, 128 etc

* A neural networks architecture is defined by the number of layer and nodes in that layer.

How input and output are connected?

* The output of each node is the previous layer becomes the input for each node in the current layer (fully connected)
* Similarly the output of each node in the current layer is passed to every node in next layer.

How to determine the right number of layers and nodes in a network?

Each node in neural networks learn about the relationship between the features variables and target variables.

This knowledge is persisted in its weights and biases.

when there are more nodes and layers it's usually results in better accuracy

Note:- This will always not true

more layer would also means more compute resource and time for both training and inference

The right architecture for a given problem is determined by experimentation.

Weights and Biases:-

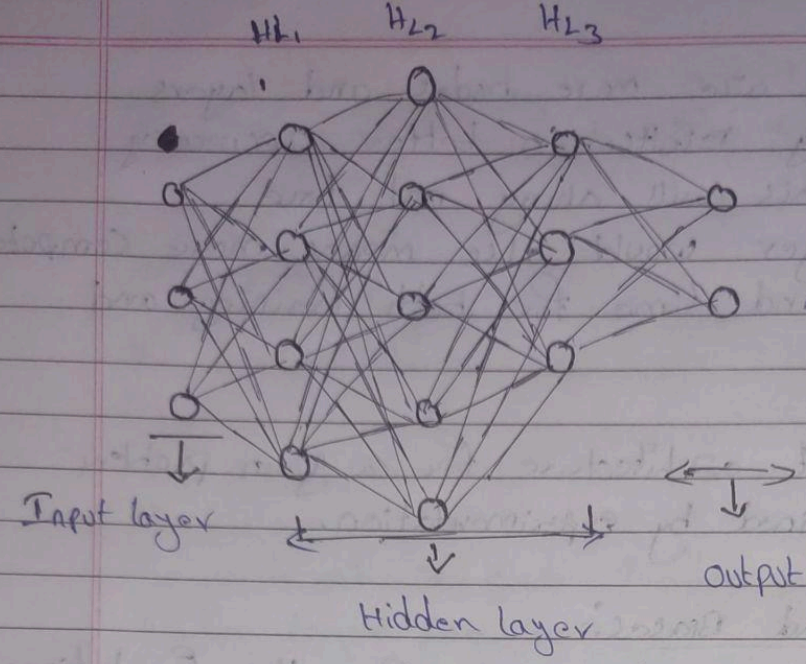
weights and Biases form the foundations for deep learning algorithms. How are this structure given neural network architecture.

weight and Biases are trainable parameters ~~for~~ in a neural networks.

During training process the values for these weights & biases are determined. such that they provide accurate predictions

* weights & Biases are nothing but a collection of numeric values
* Each input for each node will have an associated weight with it
* At a layer level these weight & biases are handled as arrays.

let's compute ~~As~~ The number of weights and bias values for the example network.



fig(a)

Layer	Input	nodes	weights	Biases
HL_1	3	4	12	4
HL_2	4	5	20	5
HL_3	5	3	15	3
output	3	2	6	2
Total			53	14

HL_1 → First Hidden layer
HL_2 → Second Hidden layer
HL_3 → Third Hidden layer

The above network has total of 53 weight and 14 bias values
The network does has 67 parameters

When training a neural network computation for hidden layer is done together and not for each node for optimization reasons
The weights and biases for each layer

The input is also taken as a matrix and output is produced as matrix. For given example neural networks the computing is for Hidden layer two.

The Hidden layer two has four inputs and five node. (see fig (a))
The four input recieve input as a matrices of one by four. The weights for all the nodes in the layer are maintain as a four by five matrix. Bias is one by five matrix covering the five node in the layer.

matrix multiplication is performed between the input and weight matrices to get an output of one by five. This is added to the bias matrics to produce an output of one by five.

The output matrix is then passed as input to the next layer

$$\begin{bmatrix} w_{11} & w_{21} & w_{31} & w_{41} \\ w_{12} & w_{22} & w_{32} & w_{42} \\ w_{13} & w_{23} & w_{33} & w_{43} \\ w_{14} & w_{24} & w_{34} & w_{44} \\ w_{15} & w_{25} & w_{35} & w_{45} \end{bmatrix} * \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{bmatrix} + \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \end{bmatrix}$$

$$w^T x + B = Y$$

Activation functions:-

An Activation function plays an important role in creating the output of a node in a neural network

An activation function takes in the matrix output of the node and determine if and how the node will propogate its information to the next layer.

* Activation acts as filters to reduces noise and also normalize the output which fairly large due to matrix multiplication. It converts the output to a nonlinear value
* They serve as a critical step in helping a specific pattern in data.

Here are the list of some popular Activation function

Activation function	output
Sigmiod	0 to 1
Tanh	-1 to +1
Rectified linear unit	0 if x<0; x otherwise
Softmax	vector of probabilities with sum=1

Choice depends on problem with and Experimentation.

Each of these function have specific advantages shortcoming and application

They can take in an output matrix & delivery another output matrix of the same ~~dem~~ dimension.

A Sigmoid function delivers an output in the range of zero to one, based on input values

Where it has output values as zero it means that it does not pass its learning to next layer

A Tanh function normalizes the output in the range of -1 to 1 similarly.

A ReLU produces a zero if output is negative, else, it reproduces the same input verbatism.

As Softmax activation function is used in case of classification problems. It produces a vector of probabilities for each of the possible classes in the outcome

The sum of probabilities will be equal to one

The class with high ~~possi~~ probability will be considered for prediction

The output layer:-

output layer is the final layer of neural network where desired prediction are obtained.

There is one output layer in neural network. It has its own set of weights and biases that are applied before the final output is derived.

* The activation function for the output layer may be different than the hidden layers

based on the problems

example:-

softmax activation is used to derive the final classes in a classification problem

* The output is a vector of values that may be need futher post-processing to convert them to business related values

For example:-

In classification problem, the output is a set of probabilities that needs to be mapped to corresponding business classes

→ How to determine the number of nodes in the output layers?

It depends on the problems

* In a binary classification problem, there is only 1 node that provides a probability of the positive outcome.

* In n class classification problem, there are n nodes, each producing the probability for a given class.

* For regression, there is only one node that produces the output

In this way, the number of ~~nodes~~ nodes may vary based on the type of problem being solved

Training a neural network:-

→ How does a deep learning model get trained?

Before we start training the neural network the input need to be prepared This includes applying a number of processing techniques to convert samples into numeric vectors. They are then transposed optionally to create the input vectors.
The target variable may undergo similar transformations.

To help with input data, the input data is usually split into training, test, and validation sets.

A training data set is used to run through the neural network and fit the parameter like weight and biases. Once model is created, the validation (check) data set is used to check for its accuracy & error rates

The results from this validation is then used to refine the model and recheck when a final model is obtained it is used to predict on test set to measure the final model's performance.
The usually split of input data is between the training, validation & test sets are

80 : 10 : 10

Model Architecture and Hyper Parameters

In order to create the initial model a set values need to be selected for various parameters and hyper parameters.

* This includes the number of layers and number of nodes in each layer. We also need to select the activation function for each layer.
* Then, there are hyperparameter like epoch batch size and Error function that need to be selected

→ How do we make the initial selection?

* It may be based on own intuition and, Experience
* It can also be based on reference in best practices and suitability of technique to the specific problem. Whatever values are selected they are then refined as the model is trained
* If the final results of the model are not acceptable, then we will go back adjust the parameters, and then retain the model.
* Finally, we also need to initialize the weight and biases for each node in the neural network
* we will start with some values and then the neural network will learn the right values for these based on the Error rates are obtained during the training process
* multiple techniques for initialization are available.

* In Zero Initialization, we will initialize all values to zero
* The preferred technique though, is random initialization, In random initialization we initialize the weight and biases to random values to obtained ~~for~~ from a standard normal distribution. whose mean is zero, and standard deviation is one.
* Once we are done will setup and installation we are ready to do some training

* Forward Propagation:-

once we are ready with input training data, we are ready to do a round of forward propagation

We known that we have the input data orgainzed as sample and features. The data has been pre-split into training, validation and test set.

For training set, for each sample, we have a target or the value to predict called y

y is the actual value of the target in the training set

$\hat{y}$ will be the value that will be predicted through forward propagation

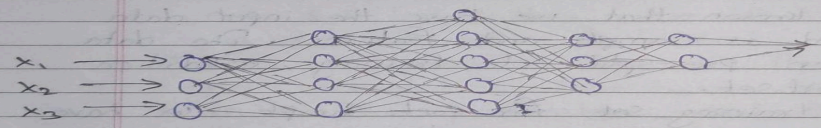
The Forward propagation step is exactly the same as doing as actual prediction with neural network

Inputs, Targets and predictions

	Sample 1	Sample 2	Sample 3	Sample 4
Feature - 1	x_{11}	x_{21}	x_{31}	x_{41}
Feature - 2	x_{12}	x_{22}	x_{32}	x_{42}
Feature - 3	x_{13}	x_{23}	x_{33}	x_{43}
Feature - 4	x_{14}	x_{24}	x_{34}	x_{44}
Target	y_1	y_2	y_3	y_4
Prediction	$\hat{y}_1$	$\hat{y}_2$	$\hat{y}_3$	$\hat{y}_4$

Forward Propagation : 1 sample (Single sample)

Inputs — Weight & Biases — Prediction: $\hat{y}$ — Actual: y



For each sample, the input are sent through the designated neural network. For each we compute the outputs based perceptron formula and pass them to next layer.

The final outcome $\hat{y}$ is then obtained at the end

Forward Propagation for all samples :-

As we keep sending the sample to the neural network we will collect the value of $\hat{y}$ for each sample

This process is repeated for all samples in d training dataset.
we will then compare the values of $\hat{y}$ and y and compute Error rate

measuring Accuracy and Error

Accuracy and Error are alternating terms that can be used to represent the gap b/w predicted values and the target values of the target variables

We have see that their might a changes in Error rate in forward propagation where we end up with a set of $\hat{y}$ values that need to be compared with actual value of y to compute error ~~(rate)~~

For computing Error we use two function
1. loss function
2. Cost function.

loss function:- A loss function measure the prediction Error for a single sample

Cost function:- A cost function measure the error across a set of sample.
The cost function provide an averaging effect over all the errors found on the training dataset
The terms loss function & cost function are used almost interchangeably and are used to measure the average error over a set of samples

There are number of popular cost function available, there are implemented in all deep learning ki libaries

Popular cost functions:-

Cost Function	Application
mean Squarie Error	Regression
Root mean Squarie Error	Regression
Binary cross entropy	Binary classification
Categorical cross entropy	multiclass classification

The mean Squarie Error (or) MSE measure errors in case Regression problem. It computes the difference between the predictor and the actual values and square them and sums them across all samples, and finally divides by the number of samples.

The Root mean Squarie Error (or) RMSE is more popular, as it provide error value in the same scale as the target variables This again used in regression problem

For binary classification we use Binary cross entropy cost function to compute the Error

A similare function called categorical cross entropy Exist for multi class classification problems.

So with cost function how do we measure accuracy?

We send a set of samples through ANN for forward propagation and predict the outcome
* we estimate the prediction error between the predicted outcome and expected outcome. using an cost function
* we then use backward propagation to adjust the weights and biases in the model based on the error obtained

Backward Propagation:-

once we have estimated the prediction error from forward propagation. we need to go back to back propagation to adjust weights and biases

→ what is the purpose of Backward Propagation?

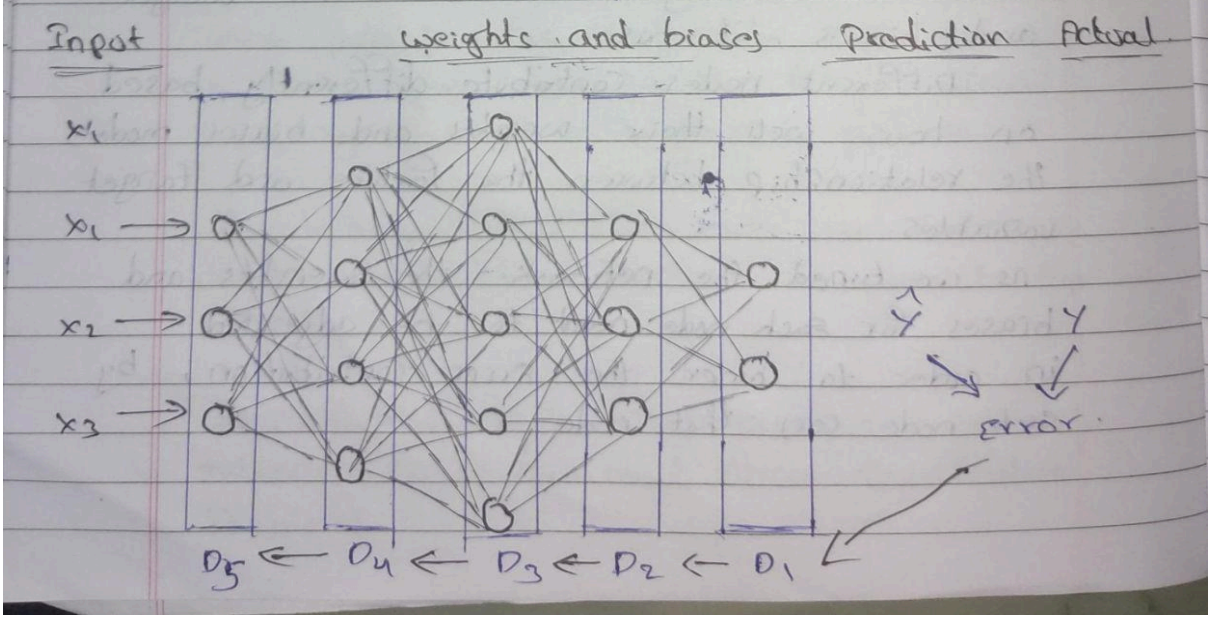
* we found an overall error based on the entire network during forward propagation
* each node in the neural network contributes to this overall error And nodes contribution is determined by the values for the weight and biases, it has
Different nodes contribute differently based on how well their weights and biases models the relationship between the feature and target variables.
As we tuned the network the weights and biases for each node need to be adjusted in order to lower the error contribution, by each node (or) that node.

How does Backward propagation work?

Backward propagation work in the reverse direction as the forward propagation

* we start from the output layer, we will compute a delta value for this layer based on the overall Error.
* This delta value is an adjustment that is applied to all weights and biases in the layer.
* This results in new values of weight and biases in the layer
* Then we derive a new delta value for the previous layer based on the new values in the current layer, it is then applied to the weights and biases in previous layer.
* The process of computing deltas, applying it to weight and biases and then back propagating continues until we reach the input layer.

Backward Propagation...



This diagram show the same back propagation process for network example.

The D_1(Delta$_1$) to D_4(Delta$_4$) are computed at each layer and applied to their weights and biases. They also propagate to the previous layer and influence their deltas

If we need to know the math topic of this and their working check on the Youtube (For better understanding purpose).

note:- Deep learning libaries take cares of these computation

At end of back propagation process, we will have updated set of weights and biases that would reduce the overall prediction error.

How do we continue to redue Error and Improve accuracy?

Using Gradient descent we would continuesly reduce Error and improve accuracy.

What is Gradient descent?

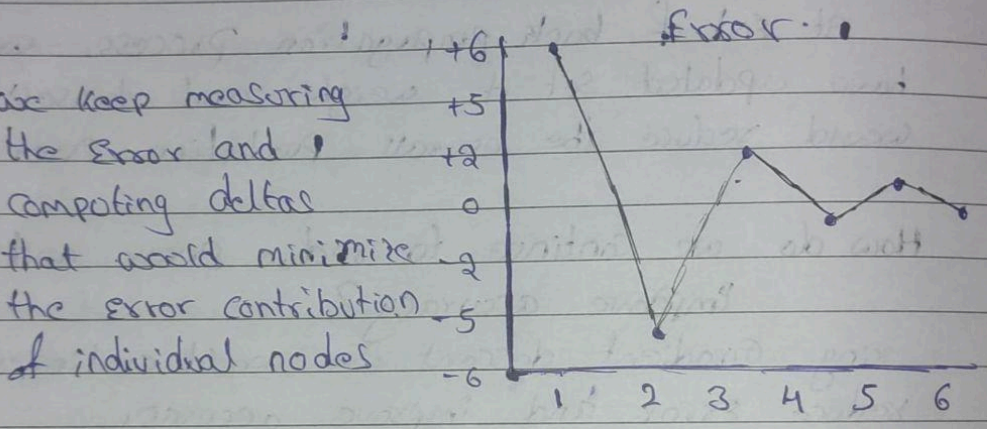
Gradient descent is the process of repeating forward and backward propagation in order to reduce Error and move closer to the desired model

one run of forward propagation results in predicting the outcomes based on weights and biases we computed the Error using a cost function. we then use back propagation to propagate ~~error~~ Error and adjust the weight and biases This is one pass of learning.

we have to ~~repeat~~ repeat this process again and again and pass them as weights and biases get refined, and Error gets reduced. This is called gradient descent.

In gradient descent we repeat the learning process of forward propagation, estimating error, backward propagation and adjusting weight & biases

As we do this the overall Error estimated by cost function will oscillate around and start moving closer to zero



We keep measuring the Error and computing deltas that would minimize the error contribution of individual nodes

There are situation where Error will stop reducing or producing they are additional hyperparameters to control that

There are also hyperparameter to speed up or slow down the learning process

Batches and Epoches:-

Batches and epoches help control the number of passes through the neural networks during the learning process

What is a batch?

A batch is a set of training samples, that are sent through the neural networks in one single pass

* The training data set is divided into one or more batches
* The neural network recieves one batch at a time, and does forward propagation
* Cost fondion are Executed and weights and biases updated after each batch. when the next batch come in, It will have a new set of weights and biases to work with
* There are two types of gradient descents
* When the batch size is equal to ~~train~~ training set size. it is called batch gradient descent
* Batch gradient descent : batch size = training set size.
* When the batch size is less than the training data set size. it's called mini-batch gradient descent.
* mini-gradient descent:
batchSize < training set size.
* Batch sizes are typically based on the two power & number like 32, 64, 128, etc
* A training data set is sent through the neural network multiple times during the learning process.
The total number of time the Entire training data set is sent through the neural network is called Epoch.
* An Epoch has one (or) more batches
* As more epochs happen, the same batch is sended repeatedly through neural network but will get to work with different set of weights & biases Each time.

* When all epochs are completed, the training process is complete.
* Higher epoch sizes can lead to better accuracy, while also delaying the training process.

Example of how epoch & batch compare.
Let say we have training set size 1000 we set the batch size to 128 & the Epoch count to 50

Training set size = 1000, batch size = 128
Epoch count = 50.

This means the number of batches in a given epoch is ceiling values of 1000 % 128.

Batches per Epoch = ceil (1000/128) = 8.

The last batch will have fewer sample than 128 the total interations (passes) through
ANN = 8 * 50 = 400.

total interation = total batches * total epochs

We have got 400 interation for this Example this mean weight & bias are updated 400 times

Batch sizes and epochs are consider as hyperparameters. They are tuned during the model learning process to improve model accuracy

Validation and testing

As we build models, we need to also validate & ~~the~~ test them against independant data sets to measure out of sample Error

During the input preparation process, we usually isolate validation and test data sets for this purpose

What is validation?

While performing learning & model improvement we are Comparing the predictions provided by the neural networks for the training samples against its actual values and measuring Errors. However it is an in sample Error and model has not graduate that it will perform the same against independent datasets

* So after epoch is completed and the weight and biases updated, we will also use the network to predict for the validation dataset
* we will measure accuracy and/or loss for the validation dataset & also investigate the same to make sure that it does not deviate significantly from the in sample Error observed
* The model can be fine tuned and learning process repeated based on results seen against the validation dataset

Evaluation:-

The final step in model building is Evaluation

* After all fine tuning is completed, and final model obtained the test data set can be used to evaluate the model.
* This is done only once at the end

* The evaluation results are then used to measure the performance of the model in terms of its final accuracy & error rates.

11 → What is an ANN model and what does it contain?

An ANN model is represented by a set of parameter namely weights, biases that are obtained during training

* When someone says that model has x parameters that they are mentaining the total count of weights and bias values in the model
* A model is also represented by a set of parameter. This includes the number of layer nodes in each layer, activation function in each node, cost functions, optimizer & the learning rate are also used It also use epoch and batch size, used to train the model.
* A model file typically contain the representation of ~~th~~ all these values
* model can be saved as to files & shared and loaded into other binaries if needed.
* once we have a model what does prediction process look like?

The prediction process looks like exactly the same as forward propagation step, except that the input data used is the feature attributes for prediction & the target value is not known

So, the steps involve:

* preprocess & prepare input
* passing them to layers
* computing the output in each node using

their final weights and biases and deriving the outcome ŷ in the final layer

The prediction may also need to be post processed for converting them to business representations.

Pratical aspects of building neural networks models.

How do we build a neural networks for use case?

An interesting fact about neural network is most neural network implementations are not designed & built from ~~stra~~ scratch.

* Designing a neural network with the right number of layers and nodes is a tedious, iterative and time consuming ~~pro~~ process.
* To begin with neural network there are ~~sev~~ several papers are published on it that have successfully implemented and proven
* we have open-source model for implement of neural networks models.

The model include all the trained parameters and hyperparameters packaged in standardized formats that are supported by ~~d~~ popular deep learning networks

Popular neural network architectures

* leNets → CNN
* Alex Net → CNN
* Res Net
* VGG → recurrent neural network
* Transformers (used for predicting sequence of texts)

How to pick the right open-source model for our use case?

First, when evaluating an open-source model understand their original purpose and use case.

This help us understand what this model is good at and whether this is applicable for our use case.

* It is also important to learn about the data that this model is trained on (Data could be public data or, may be focussed on specific use case)
* Explore the model's popularity & usage. (This can easily identified by number of downloads, forks and blocked related to this model).
* Download the model and build fine-~~tuner~~ tuning or inference pipeline around it.
* Test the model with specific data to your use case, to ensure that model perform ~~gerenate~~ in your specific scenario.

pratical workflow, using ~~three~~ Example.

(go to below link)

Check out below link for code which I have done praticaily

https://github.com/Pavankuamr14/deeplearning

Follow This link to checkout the practical knowledge using the code, I tried to simply the topic and explained the code in a simple way as my best