Deep Learning

1. Subset of machine learning.
2. Neural networks with 3 or more layers.
3. Imitates how humans process data and make decisions.
4. Exponential growth in the last few years.

Applications of Deep Learning:

1. Natural Language Processing.
2. Speech and Image Recognition.
3. Self-driving cars.

Linear Regression:

One of the basic statistical concepts that is used in machine learning is linear regression.

Linear Regression is a linear model that explains the relationship between two or more variables.

We have a dependent variable y and an independent variable x. Y is dependent on X.

The model provided an equation to compute the value of y based on the value of x.

To compute this wee need two constants , a which is the slope and an intercept which is b.

`Y = aX+b`

Building a Linear Regression Model:

1. Find values for slope and intercept.
2. Use known values of x and y.

3. Multiple independent variables make it complex.

Logistic Regression:

A similar technique which is used in Deep Learning is Logistic Regression.

1. Logistic Regression is a binary model that defines the relationship between two variables.
2. The output Y is this case is either 0 or 1.

`Y = f(aX+b)d`

The formula is the same as Linear Regression, we just use an activation function called f to convert the continuous variable coming out of `aX+b` into a boolean value of 0 or 1.

An analogy for Deep Learning:

1. Deep Learning is a complex and iterative process that requires a series of trials to narrow down the parameters for the model.
2. Starts with random initialization and works towards the right values by trial and error.

In linear regression model, the model parameters are A and B.

3. The equation may not always provide a zero error but in most cases we are trying to minimize the error.

Perceptron:

The perceptron is the unit for learning in an artificial neural network.

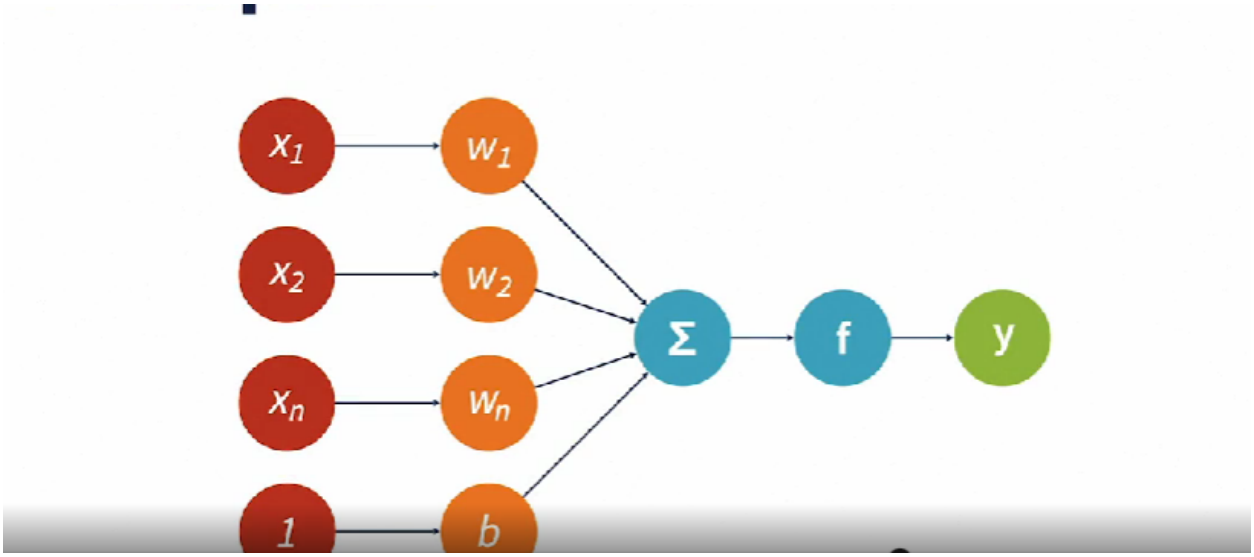
1. An algorithm for supervised learning for binary classification.
2. Resembles a cell in the human brain.
3. Multiple inputs are fed into the perceptron which in turn does the computations and returns the output as a boolean variable.
4. Represents a single cell neural network.
5. Based on Logistic Regression.

We replace the slope (a) with weight called as w and replace intercept (b) with bias called as b.

Weights and biases become the parameters for a neural network. We then apply an activation function f that outputs a boolean result based on the values.

This formula of perceptron is fundamental to deep learning.

```
Y = wX + b
```



1. We have multiple independent input variables, x1 to xn that are fed to the perceptron.
2. Each of them is multiplied by a corresponding weight. The number of weights is equal to the number of inputs.
3. We also feed in a 1 that is multiplied by the bias.
4. All the results are summed up.
5. An activation function is applied that delivers the value of Y which is either a 1 or a 0.

We build neural networks by connecting a number of perceptrons.

Artificial Neural Networks: (ANN)

1. An ANN is a network of perceptrons, modeled after the human brain.
2. Nodes (Perceptrons) are organized as layers.
3. A deep neural network has 3 or more layers. Each node has weights, biases and activation functions.
4. Each node is connected to all nodes in the next layer forming a dense network.
5. The nodes within a layer are not connected to each other.

In the input layer, there is one node for each independent variable.

In hidden layer, there are 3 layers in this example. Layer 1 has 4 nodes, layer 2 has 5 nodes and layer 3 has 3 nodes.

How ANN works?

1. Inputs(independent variables) are sent from the input layer.
2. Inputs are passed on to the nodes in the next hidden layer.
3. Each node is a perceptron consisting of a weight, bias and an activation function.

Training an ANN:

An ANN is created through a model training process.

1. A neural network model is represented by parameters and hyperparameters. This includes the values of weights, biases, nodes and layers etc.
2. Training a model means determining the best possible values for the parameters and hyperparameters that maximize accuracy.
3. Inputs, weights and biases might be n-dimensional arrays.

Training Process:

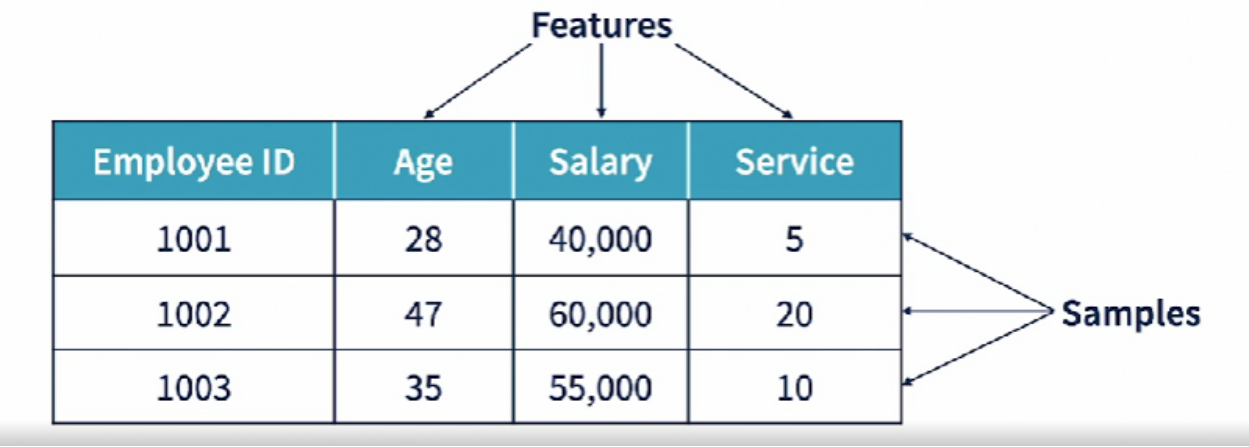
1. Use training data (known values of inputs and outputs).

2. Create network architecture with intuition.
3. Start with random values for weights and biases.
4. Minimize error in predicting known outputs from inputs.
5. Adjust weights and biases until error is minimized.
6. Improve model by adjusting layers, node counts and other hyperparameters.

The input layer:

Vectors:

1. The input to deep learning is a vector of numeric values.
2. A vector is a tuple of one or more values.
3. Vectors are used defined as a NumPy array.
4. It represents the feature variables or independent variables that are used for predictions as well as training.



Samples: Records in a database.

Features: Individual attributes or column headers.

For an image, each image is a sample and it's pixel representation becomes its features.

Input Preprocessing:

1. Features need to be converted to numeric representations.

a. Numeric input: Centering and scaling to normalize the values.

b. Categorical: Integer encoding, one-hot encoding.

c. Text: TF-IDF, embeddings.

d. Image: Pixels-RGB representation.

e. Speech: Time-series of numbers.

The hidden layers:

1. An ANN can have one or more hidden layers. The more the number of layers, the deeper the network is.
2. Each hidden layer can have one or more nodes.
3. A neural network architecture is defined my the number of layers and nodes.

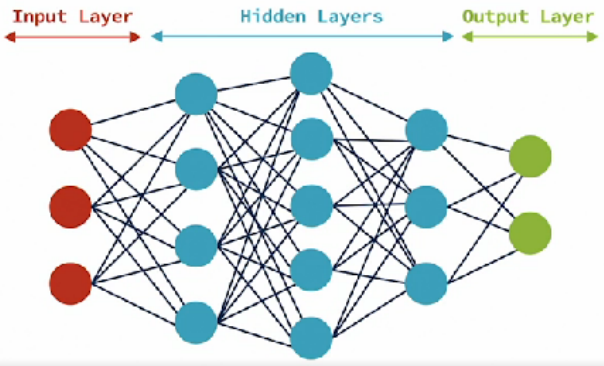
Inputs and Outputs:

1. The output of each node is the previous layer becomes the input for each node in the current layer.
2. Each node produces one output that is forwarded to the next layer.

The first hidden layer has four nodes producing 4 different outputs from the activation functions. The 4 outputs are passed to each of the fiver nodes in the second layer.

Weights and Biases:

1. Weights and biases represent the trainable parameters in an ANN.
2. Each input for each node has a weight associated with it.
3. A given node will have one bias value associated with it.



Layer	Inputs	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**

Activation functions:

a. An activation function takes in the matrix output of the node and determines which nodes propagate information to the next layer.

b. Activation functions acts as a filter to reduce noise and normalizes the output which can get fairly large due to matrix multiplication.

c. It Converts output to nonlinear.

1. Sigmoid Function: 0 to 1.
2. Tanh: -1 to +1. (Normalizes the output)
3. Rectified Linear Unit (RELU) : 0 if x<0: x otherwise
4. Softmax: Vector of probabilities with sum = 1, iit is used in classification problems. The class with highest probability with be considered for prediction.

The output layer:

1. There is one output layer in the neural network that produces the desired final prediction.
2. It has its own set of weights and biases.

Output Layer Size:

1 for regression problems.

n for n-class classification.

1 for binary classification.

Setup and initialization:

The input data is split into training, test and validation sets.

1. Training set: Used to fit the parameters like weights and biases.
2. Validation set: Once the model is created, the validation set is used to check for its accuracy and error rate. The result from this validation is then used to refine the model and recheck.
3. Test set: Used to measure the final model performance.

`80:10:10`

Weights Initialization:

1. Zero initialization: Initialize to zeros, not recommended.
2. Random Initialization: Initialize to random values from a standard normal distribution (mean = 0, SD =1)

Forward propagation:

Y is the actual value of the target in the training set. Y hat will be the value that is predicted through forward propagation.

Measuring accuracy and error:

For computing error, we use two functions.

1. Loss Function: measures the prediction error for a single sample.
2. Cost function: measures the error across a set of samples. Cost function provides an averaging effect over all the errors found on the training dataset.

Popular Cost Functions:

1. Mean Square Error (MSE) : Regression
2. Root Mean Square Error (RMSE): Regression
3. Binary Cross Entropy: Binary Classification
4. Categorical Cross Entropy: Multi-class Classification

Measuring Accuracy:

1. Send a set of samples through the ANN and predict outcomes.
2. Estimate the prediction error between the predicted outcome and expected outcome using a cost function.
3. Use back propagation to adjust weights based on the error value.

Back propagation:

1. Each node in a neural network contributes to the overall error in prediction (differing contributions)
2. A node’s contribution is driven by its weights and bias.
3. Weights and biases for each node needs to be adjusted in order to lower the error contribution.

How Back Propagation Works:

1. Back Propagation works in reverse of the forward propagation.
2. Start from the output layer.
3. Compute a delta value based on the error found.
4. Apply the delta to adjust the weights and biases in the layer. This results in new values of weights and values.
5. Derive a new error value.

Gradient descent:

Gradient descent is the process of repeating forward and backward propagations in order to reduce the error and move closer to the desired model.

1. Repeat the learning process which comprises of:

a. Forward propagation.

b. Estimate the error.

c. Backward propagation.

d. Adjust weights and biases.

We keep measuring the errors and computing deltas that would minimize the error contribution. There are hyperparameters to speed up and slow down the learning process.

Batches and epochs:

They control the number of passes through the neural network, during the learning process.

Batch: A batch is a set of samples sent through ANN in a single pass.

1. The training data set can be divided into one or more batches.
2. Training data is sent to the ANN one batch at a time.
3. Cost function and parameters are updated one batch at a time.
4. When a new batch comes in, it has a new set of weights and biases to it.

Batch Gradient Descent: batch size = training set size.

Mini-batch gradient descent: batch size <training set size

Typical batch sizes are 32, 64, 128 etc.

Epoch: The number of times the entire training set is sent through the ANN.

1. An epoch has one or more batches.
2. The same batch is sent repeatedly but will get to work with a different set of weights and biases.
3. When an epoch is complete, the training process is complete.
4. Epoch sizes can be higher to achieve better accuracy.

Training set size = 1000, batch size = 128, epoch = 50

a. Batches per epoch = ceil(1000/128) = 8
b. Total iterations or passes through ANN: 8*50 = 400

This means that the weight and biases are updated 400 times.

Batch sizes and epochs are considered as hyperparameters.

Validation and testing:

1. During learning, the predictions are obtained for the same data that is used to train the parameters (weights and biases).

2. After each epoch and corresponding parameter updates, the model can be used to predict for the validation data set.
3. We will measure accuracy and loss for the validation data set.

Testing:

1. The final step in model building is evaluation.
2. After all the fine tuning is completed and final model obtained, the test data set can be used to evaluate the model.
3. The test data set is used to evaluate the model.
4. Results obtained with the test data set is used to measure the performance of the model.

An ANN model:

Parameters:

1. Weights
2. Biases

Hyperparameters:

1. Number of layers, nodes in each layer, activation functions.
2. Cost functions, learning rate, optimizers.
3. Batch and epoch.

Prediction Process:

1. Preprocess and prepare inputs.
2. Pass inputs to the first layer.
3. Compute Y using weights and biases and activation.
4. Pass on to the next layer.

5. Predictions might also be post processed.