Introduction to Deep Learning

Classification vs Regression

Classification:

Classification algorithms are used when the Target Column is Discrete in nature. The discrete variables can be two or more.

- Binary Classification: There are only 02 variables in target column.
- Multi Classification: There are more than 02 variables in target column.

Regression

Regression algorithms are used when the Target Column is Continuous in nature. It can be integers or decimal point numbers.

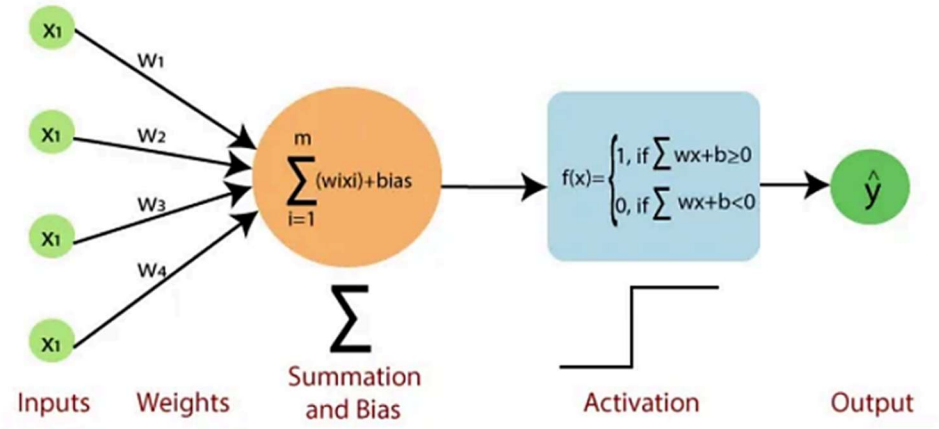
Note: Nature of problem is decided based on the nature of Target Column.

Basic Perceptron

A perceptron is a neural network unit (an artificial neuron) that does certain computations to detect features or business intelligence in the input data.

A basic perceptron is a type of artificial neuron used in machine learning for binary classification tasks.

It is the simplest form of a neural network and serves as the building block for more complex neural networks.



Single-layer Perceptron can learn only linearly separable patterns.

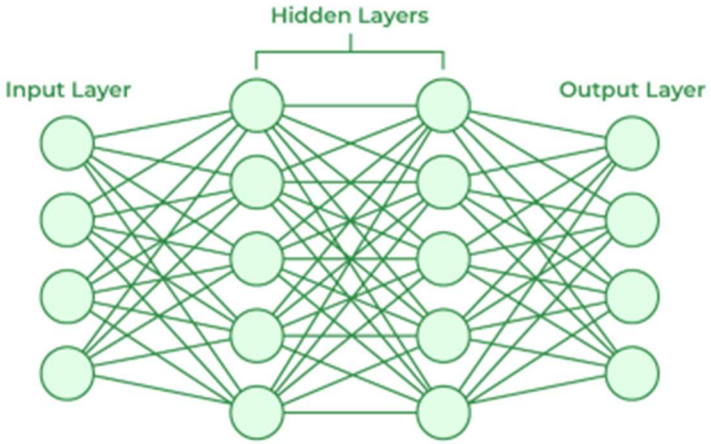
For classification we use Activation function as a threshold to predict class. And for Regression, we do not need the Activation function (Thresholding) as we can use a linear function to predict continuous value.

Input is typically a **feature** vector x multiplied by **weights** w and added to a **bias** b: y = w * x + b

$$y = \varphi\left(\sum_{i=1}^{n} w_i x_i + b\right) = \varphi(w^T x + b)$$

Normally in Neural Networks there are;

- Input Layer
- Multiple Hidden Layers
- Output Layer



Perceptron vs Neuron (Key Differences)

Scope:

- **Perceptron**: A specific model used for binary classification tasks.
- **Neuron**: A general term for the basic unit of artificial neural networks, used in a wide range of tasks including classification, regression, and more.

Activation Functions:

- **Perceptron**: Uses a step function.
- **Neuron**: Can use various activation functions like sigmoid, tanh, ReLU, etc.

Complexity:

- **Perceptron**: Limited to solving linearly separable problems.
- **Neuron**: Can be part of complex, multi-layer networks that solve non-linear problems.

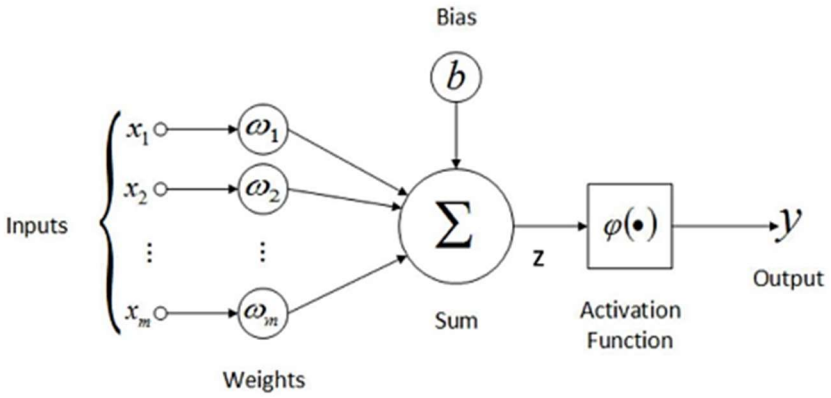
Learning Process of Deep Learning Model

Learning happens in two ways, Forward propagation and backward propagation

Feedforward

Feedforward refers to the process by which inputs are passed through the neural network to producean output. Here's how it works:

1. **Input Layer:** The input data is fed into the input layer of the network.
2. **Hidden Layers:** The data is passed through one or more hidden layers. Each neuron in these layers performs a weighted sum of its inputs and passes the result through an activation function (such as ReLU, sigmoid, or tanh) to produce an output.
3. **Output Layer:** The outputs from the last hidden layer are passed to the output layer to produce the final predictions of the network.



Mathematically, for a single neuron in a hidden layer, this process can be described as:

$$z = \sum_{i=1}^{n} w_i x_i + b$$
$$y = \varphi(z)$$

where:

- x_i are the inputs
- w_i are the weights
- b is the bias
- z is the weighted sum
- φ is the activation function
- y is the activation result (output of the neuron)

Note: Initially the Weights assigned are Random Numbers which are changed over time due to multiple Backpropagation Steps.

Activation Function:

Backpropagation

Backpropagation is the process used to train the neural network by adjusting the weights and biases based on the error of the network's predictions. It consists of two main steps: forward pass and backward pass.

1. **Forward Pass:** Perform a feedforward operation and calculate the output of the network.

2. **Calculate Loss:** Compute the loss (error) between the network's prediction and the actual target values. Common loss functions include mean squared error for regression tasks and crossentropy loss for classification tasks.

3. **Backward Pass:**

- ***Calculate Gradients:*** Compute the gradient of the loss function with respect to each weight and bias in the network. This is done using the chain rule of calculus. The gradient indicates the direction and rate at which the weights should be adjusted to minimize the loss.
- ***Update Weights and Biases:*** Adjust the weights and biases in the direction that reduces the loss. This is typically done using an optimization algorithm like stochastic gradient descent (SGD), Adam, or RMSprop.

The backpropagation process can be summarized with the following steps for a single weight update:

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

where:

- w_{new} is the updated weight
- w_{old} is the current weight
- η is the learning rate
- $\frac{\partial L}{\partial w}$ is the gradient of the loss function L with respect to the weight w

Summary

- **Feedforward**: Propagates inputs through the network to obtain outputs.
- **Backpropagation**: Uses the outputs to calculate the error and propagates it backward through the network to update the weights and biases, thus training the model.

These processes are iterated over multiple epochs (complete passes through the training data) until the model's performance converges to an acceptable level.

Epoch vs Batch Size

The concepts of "epoch" and "batch size" are fundamental in training neural networks.

Epoch

An epoch refers to one complete pass through the entire training dataset.

During one epoch, every sample in the training set is used exactly once to update the model's parameters (weights and biases).

Batch Size

Batch size refers to the number of training samples used in one iteration to update the model's parameters.

Instead of updating the parameters after each individual training sample (which can be very slow),or after processing the entire dataset (which can be memory-intensive), the dataset is divided into smaller groups called batches.

After each batch is processed, the model's parameters are updated.

Relationship

If you have a dataset of 1000 samples and a batch size of 100, it means you will have 10 batches per epoch.

The model's parameters will be updated 10 times during one epoch.

Example

Epoch: If you have a training dataset with 1000 samples, one epoch means that all 1000 samples have been processed by the model.

Batch Size: If your batch size is 100, then the 1000 samples will be divided into 10 batches, each containing 100 samples.

Libraries Used in Deep Learning

- Tensorflow (Google)
- PyTorch (Meta)
- Keras

Scikit-Learn Perceptron

Perceptron

```
class sklearn.linear_model.Perceptron(*, penalty=None, alpha=0.0001, l1_ratio=0.15,
fit_intercept=True, max_iter=1000, tol=0.001, shuffle=True, verbose=0, eta0=1.0,
n_jobs=None, random_state=0, early_stopping=False, validation_fraction=0.1,
n_iter_no_change=5, class_weight=None, warm_start=False)
```

[source]

Linear perceptron classifier.

alpha

Hyper Parameter that ensures that the model is not over-fitted. (It does so by L1 and L2

Regularization)

A small alpha makes learning slow but steady, while a large alpha speeds up learning but risks overshooting the optimal solution. Properly tuning alpha is crucial for effective model training.

---

early_stopping

This concept is used to stop the Model Training in case there is no change in Model Accuracy even with multiple train cycles.

---

random_state

random_state = 0

This **disables** the randomization/shuffling of Data in Train-Test Split step of Model Training i.e.

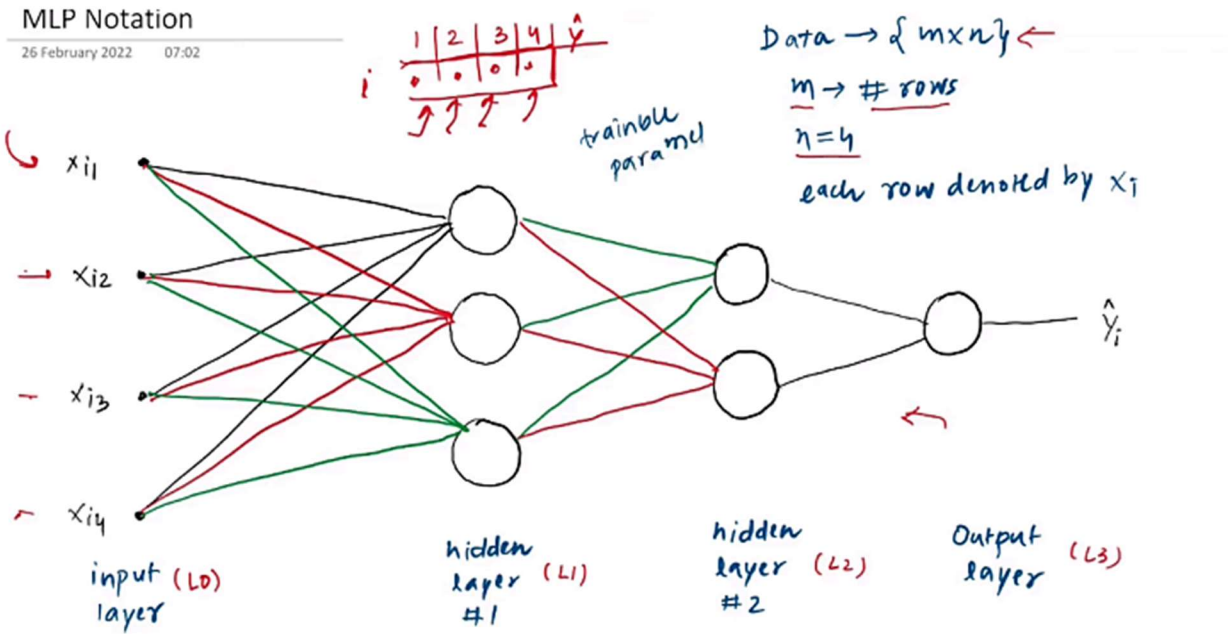
(Locking the Training Data). This enables more accurate results of Model.

Steps of Machine Learning

- Pre-process your Data (Cleaning) i.e. Improving quality of Data
- Features Extraction
- Split your Data in Input and Output Columns. (Dependent and Independent Variables)
- Split your Data in Train and Test Sets
- Scaling (Conversion of Text to Numbers etc.)
- Creating a Model
- Validate Results of your Model (By testing it with Unseen Data)

Note: In Perceptron (Deep Learning), we are passing the results of Linear Regression from an Activation Function. (Which is outside the scope of Machine Learning)

Deep Learning Model Architecture



- X_in are the Features of Input of Data
- $Circle$ are the Neurons of Neural of Network
- Output of One layer is the input of Next Layer
- O_in are the outputs, where i is the Layer# and n is the Neuron#
- Last Layer acts as the **Activation Function**