Unit 2: Deep Neural Networks

Introduction to Neural Networks

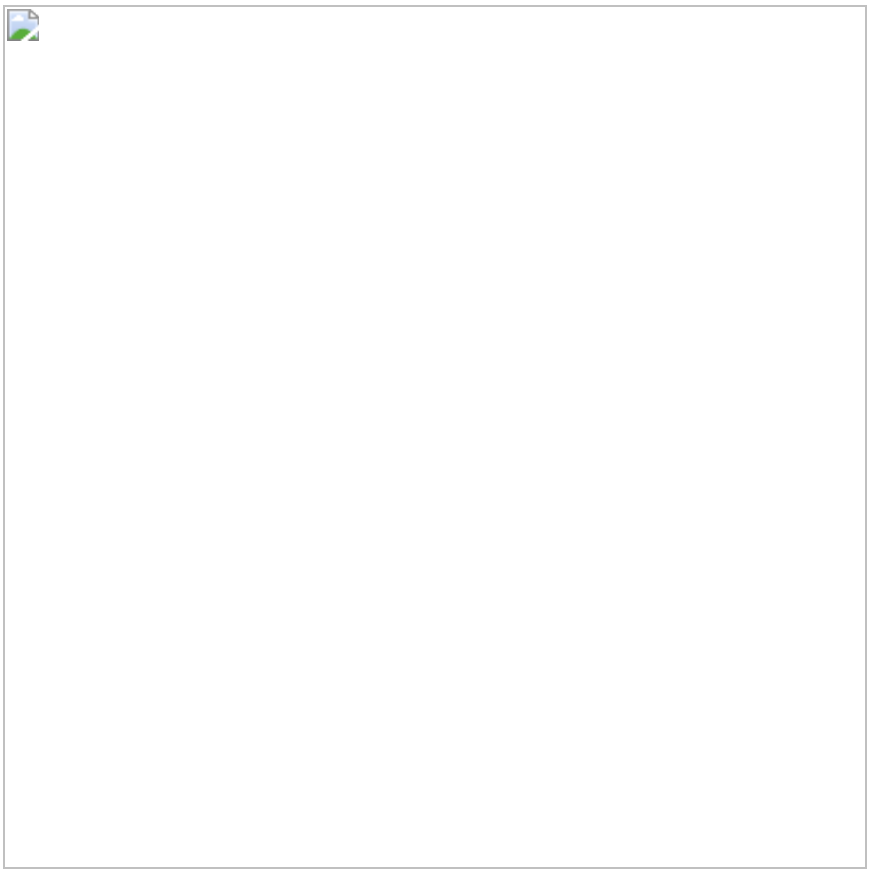
Neural Network → Structure and operation of the human brain served the inspiration for a particular form of ML model called the neural network

Its made of layers of neurons, which are interconnected processing units. Input data is processed by each neuron, and the result is passed on the next layer till it reaches the output layer.

Neural networks → Image classification, translation, playing games.

Biological Neuron

1. Nervous system → Receiving sensory information from the environment, processing it, and sending signals to the appropriate organs and muscles.
2. They play a critial role in sensory perception, various cognitive processes including learning and memory.



3. Basic structure of Neuron :
    a. Cell body (SOMA) : Contains the nucleus, houses genetic material of the cell.
    b. Dendrites : Branch like structures extending from the cell body and recieve signals from other neurons.
    c. Axon: Long fibre that extends from the cell body and carries signals away from the cell to other neurons.
    d. Synapse : Junction between axon of one neuron and dendrites of another neuron. When they communicate they release chemicals called neurotransmitters.
4. Process

> Neuron recieves signal from another neuron through dendrites → Electrical signal is processed in the cell body → Signal travels down the axon → Electrical signal triggers

> release of neurotransmitters at the end of axon → Binds to the receptors on the dendrites or cell body of the next neuron in the chain.

This forms the basis of information processing in the nervous system.

Artificial Neural Network

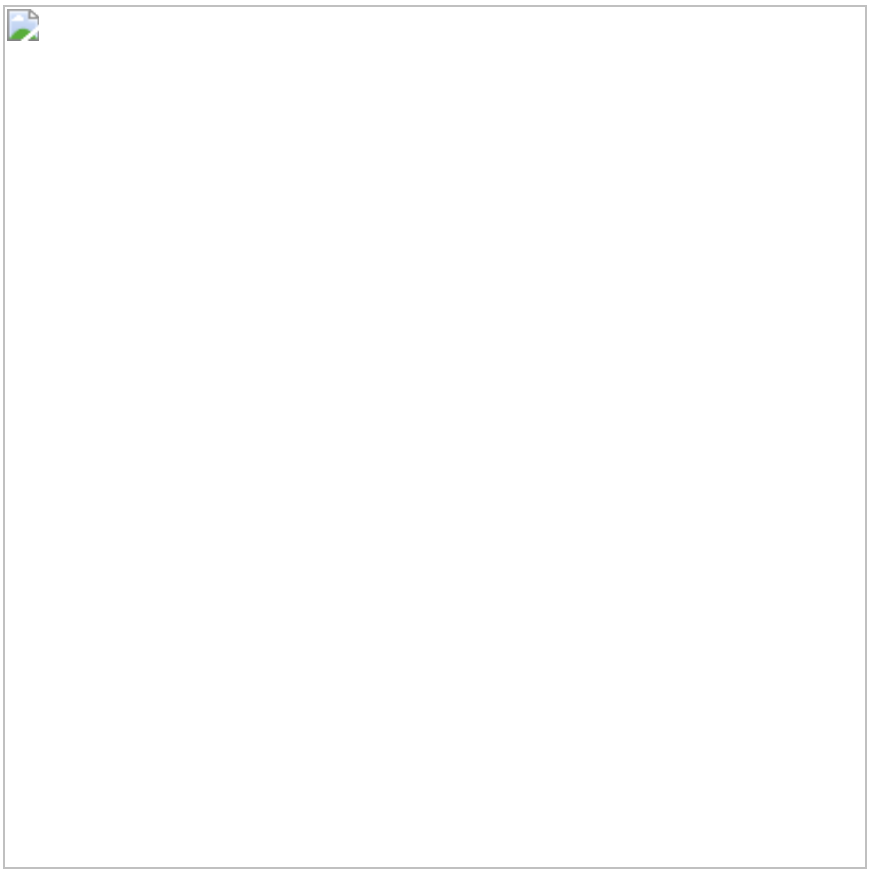
1. Type of ML model that is inspired by the structure and the function of biological neural networks.
2. Basic building block of the neural network is the → Artificial neuron → Recieves input from other neurons and external sources → Computes a weighted sum→ applies non linear activation function to the result.→ This result is passed to other neurons → Till the final output
3. NNs can be trained using
    a. Supervised learning
    b. Unsupervised learning
    c. Reinforcement learning (Network learns to make decisions based on feedback from its enviornment)
4. Comparison between ANN and Biological neural network (BNN) :
    a. Structure
        i. ANN → Layered structure, each layer with multiple neurons that perform tasks
        ii. BNN → More complex, highly interconnected structure allowing parallel processing.
    b. Processing
        i. ANN → Mathematical computations on input data, use digital signals
        ii. BNN → Communicate via electrical and chemical signals, use analog signals.

c. Learning
    i. ANN → Learn from data through backpropogation, where errors are fed back to the network to adjust the weights.
    ii. BNN → Experience dependent plasticity, where the strength of the connections is modified based on experience and feedback from the enviornment.
d. Robustness
    i. ANN → Less robust and adaptable
    ii. BNN → Highly robust and adaptable
e. Energy efficient
    i. ANN → Less energy efficient than BNN
    ii. BNN → Highly energy efficient than ANN

Perceptron

Simple type of ANN introduced.

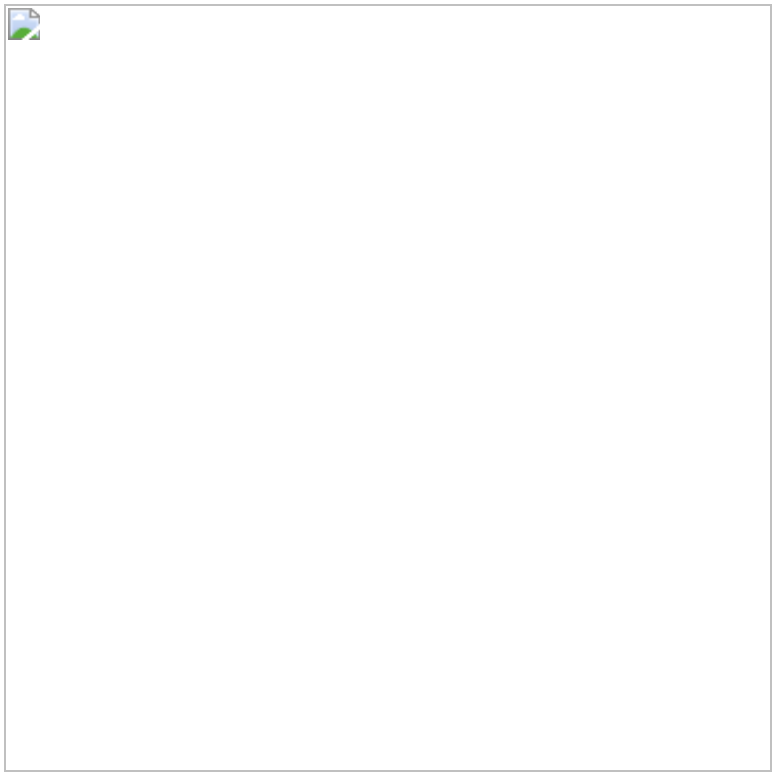
It was one of the first algos to be developed for binary classification, where goal is to classify data into 1 or 2 classes.



1. Perceptron is used to classify data into 1 or 2 categories.
2. Takes input from several sources → applies weight to each input → weighted sum is calculated → compares the sum to the threshold.
    a. if sum > threshold → Output is 1, else 0
3. Training → Adjusting the weights and threshold values to improve accuracy.
4. Perceptron consists of a single layer of neurons each are connected to inputs and produces an output.
5. Training continues till the perceptron reaches a level of accuracy that is acceptable.
6. Perceptrons can handle and classify linearly seprable data but not non linear separable data. For that more complex NNs are required.
7. This laid the foundation for Multi-layered perceptrons.

Multilayer feedforward network

1. Consists of numerous interconnected layers of artificial neurons.
2. They are called “feed forward” networks because the information flows through one direction only (input to output) , without looping back.
3. Input layer → Provides the input data and output layer generates the final prediction,
4. Layers between the input and the output layers are called hidden layers.



5. MLPs are trained using supervised algorithsm such as backpropogation, where the network is presented with labeled examples and it adjusts its weight based on the prediction error.
6. The training process continues until the network converges to a set of weights that produce accurate predictions on the training data.
7. Applications: Speech recognition, Image classification, NLP.
8. These are building blocks for CNNs and RNNs.

9. eg. For image classification
    a. Input layer → Recieves pixel values of an image → Hidden layers process the data through operations and activation functions → output layer gives the predicted value for the image.
        - Model adjusts the weights and biases based on the error between the predicted output and the actual label.

Simple Case study of Neural network

1. Data collection
2. Data preparation
3. Model training
4. Model evaluation
5. Model improvement
6. Deployment

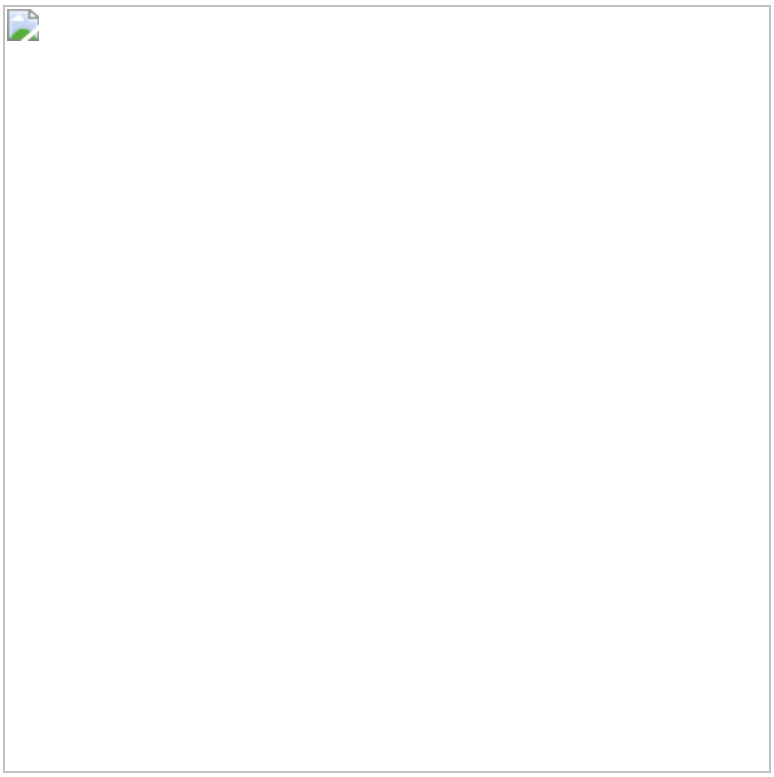
Training Neural Networks

1. Involves adjusting the values of weights and biases so the network can predict the output for a given input.
2. Usually performed using supervised learning, where the network is presented with labelled examples and then the weights and biases are updated.
3. Most common algorithm → Backpropogation
    a. Involves the gradient of loss function with respect to the networks weights and biases to update the weights and biases.
4. Training process begins with → Initialising weights and biases with random values → network is presented with training data → weights and biases are updated using gradient descent → processes is repeated multiple times using different batches > until prediction error reaches a satisfiying level.

5. Its important to monitor the process to avoid overfitting. To prevent overfitting, various techniques like dropout, early stopping and regularisation should be used.
6. Once training is complete, network can be used to make predictions on unseen examples.

Backpropogation

1. Backward propogation and forward propogation are used for training the neural network.
2. Backpropogation → Updating the weights and biases of NN during training. This adjustment is done to make accurate predictions for training. It uses the gradient of the loss function to calculate the updates.
3. Step by step explaination
    a. Initialise the network weights → typically -0.5 to 0.5
    b. Forward propogation → Input data is fed and propogated forward, and each neuron performs a weighted sum of inputs and the this produces an output.
    c. Calculate the error → Output of the network is compared to the true value of the data and the error is calculated using a loss function.
    d. Backward propogation → Error is propogated backwards from the output layer to the hidden layers.
    e. Weight update → Weights are updated using the derivative of the error wrt the weights, multiplied by learning rate. The learning rate determines the step size and is usually set to a small value to avoid overshooting.
    f. REPEAT.

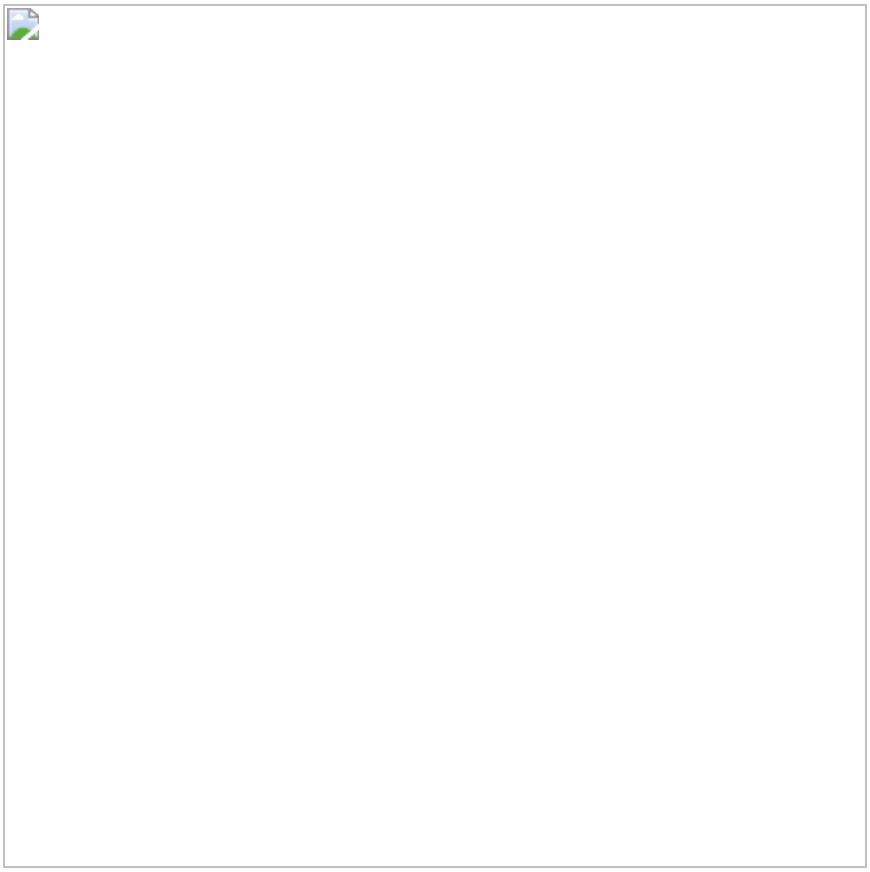


Forward propogation

1. Also known as feed forward neural network is used for classification and regression.
2. Input is passed through the network layer by layer and then the activation of each neuron is calculated using the weighted sum of inputs.Lastly, the result is passed into a activation function, producing the final outputs.
3. Steps:
    a. Initialise the network weights : -0.5 - 0.5
    b. Input data → Fed into the network
    c. Forward propogation → Input data is propogated through the layers with each neuron performing weighted sum and applying activation function to the output result. This produces an output for each neuron in the next layer.

d. Output layer → Final layer. Classification tasks → Output can be a probability of input belonging to a class.

   Regression tasks → Continous value.



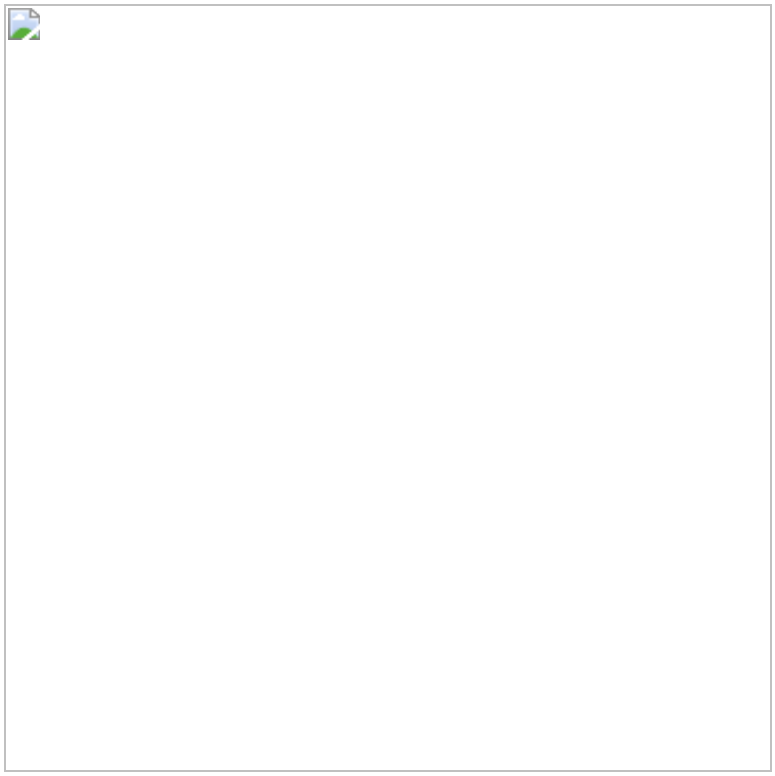
Activation Functions

Activation functions are used for introducing non linearity into the output of each neuron.

They are applied to the weighted sum of inputs to produce neurons activation which is used in the next layer.

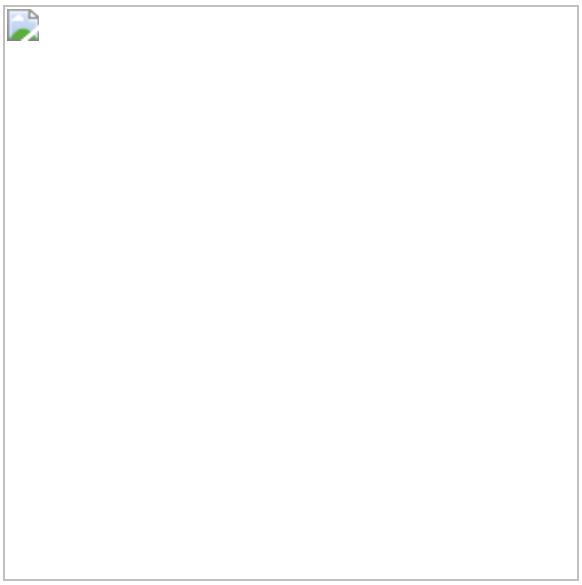
Types of activation functions :

1. Sigmoid → Maps input to a value between 0 and 1.
    a. Used for binary classification.
    b. Has a smooth S shaped curve that allows it to model relationships between input and output
    c. It is also differentiable, making it possible to use gradient descent to update weights and biases.

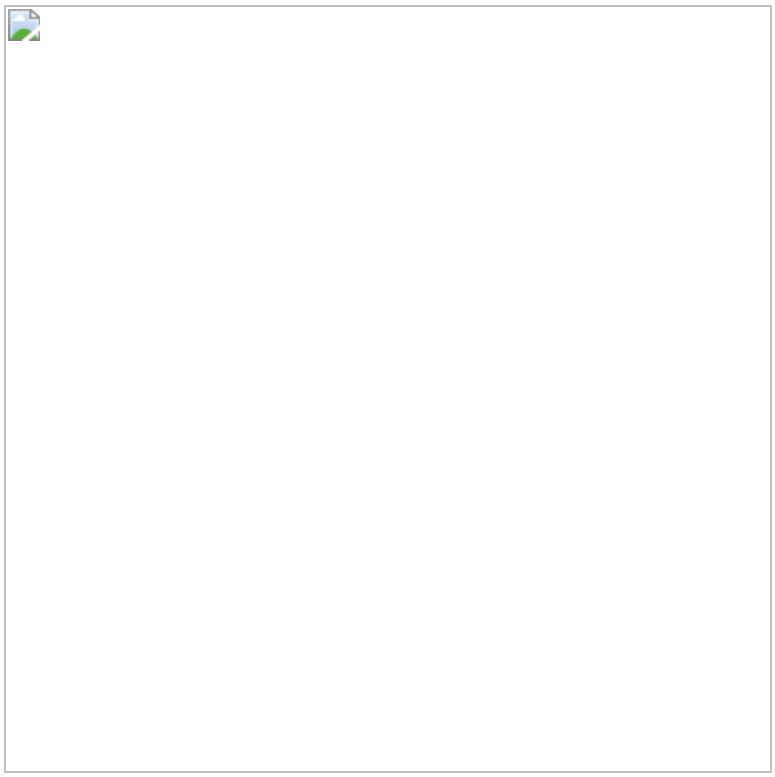


x = Weighted sum of inputs

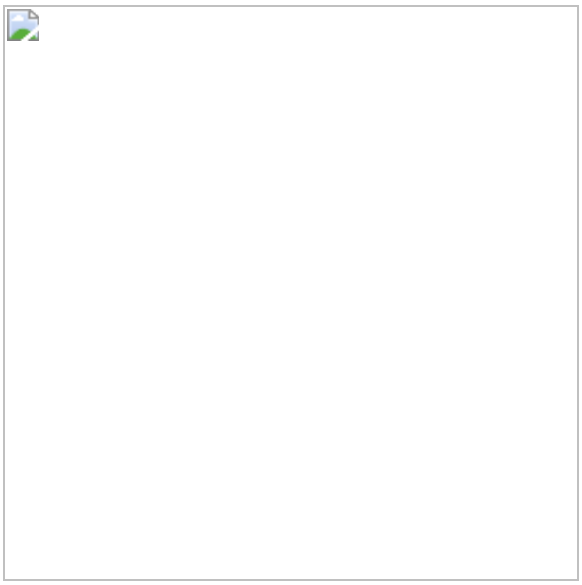
e = mathematical constant approx equal to 2.718



2. Rectified Linear Unit (ReLU)
    a. Maps input between 0 to 0, and any input above 0 to same value.
    b. Its computationally efficient and found to work well.
    c. Useful in computer vision
    d. Returns input if its positive, and 0 if negative.
    e. Its simple, fast and it can alleviate the vanishing gradient problem.
    f. Vanishing gradient → Issue that occurs in DNNs when gradient becomes very small, making it hard for the network to learn.
    g. RELU tackles this prb by introducting a non linear activation function.
    h. However it is susceptible to the dying RELU problem, that neurons with. negative weights never activate, making it hard for the network to learn, this can be mitigated by using leaky RELU or parametric RELU.



3. Hyperbolic tangent (tanh) :
    a. Maps input from -1 to +1
    b. Useful for data with a similar range.
    c. Models complex relationships due to S shaped curve.
    d. Equation: f(x) = tanh(x)
    e. It is also differentiable, making it possible to use gradient descent to update weights and biases.
    f. tanh is used in the hidden layers to introduce non linearity. Its output is centered around 0, making it easier for the network to learn.



4. Softmax function :
    a. Used in output layer of multi-classification problems.
    b. Maps the activations of output neurons to probability distribution of possible classes.
    c. Takes a vector of real numbers and returns a corrresponding vector of non negative values summing upto 1, thus a probability distribution over a set of classes.

d. It is also differentiable, making it possible to use gradient descent to update weights and biases.
e. Its computationally expensive to compute, especially for large number of classes.

5. Linear Functions
    a. Maps weighted sum directly to the output of the neuron
    b. f(x) = x
    c. Used in regression problems, where goal is to predict continuous values.
    d. In order to capture non linear relationship, RELU or Sigmoid is used.

6. Hard tanh function
    a. Clips output range to [-1,1]

- Its non-linear, can be used to introduce non linearity into the output of the neuron.
- Clipping can help prevent the output from exploding.

Choice of activation function depends on the problem.

Sigmoid → Binary classification

Softmax → Multiclass classification

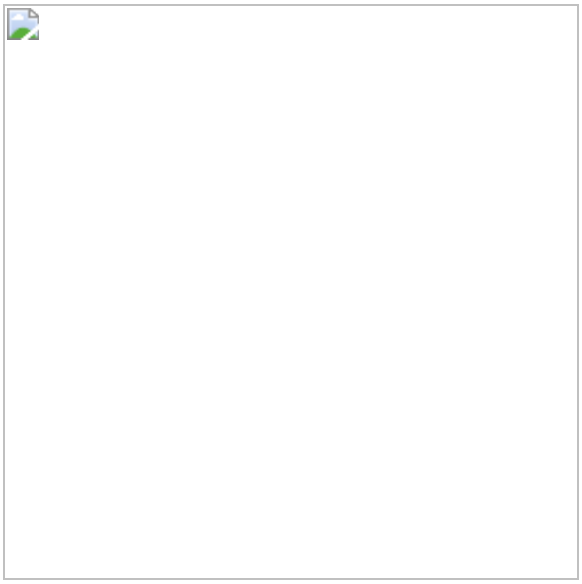
Loss Functions

1. Used to measure the discrepancy between predicted output and true output.
2. Goal → Minimise the value of loss function while training
3. Types of loss functions:
    a. Loss functions for regression
    b. Loss functions for classification
    c. Loss functions for reconstruction

Loss function for regression

1. Mean Squared error (MSE) : Average of squared differences between predicted values and true values.

2. Mean absolute error (MAE) : Average of absolute differences between predicted and true values.



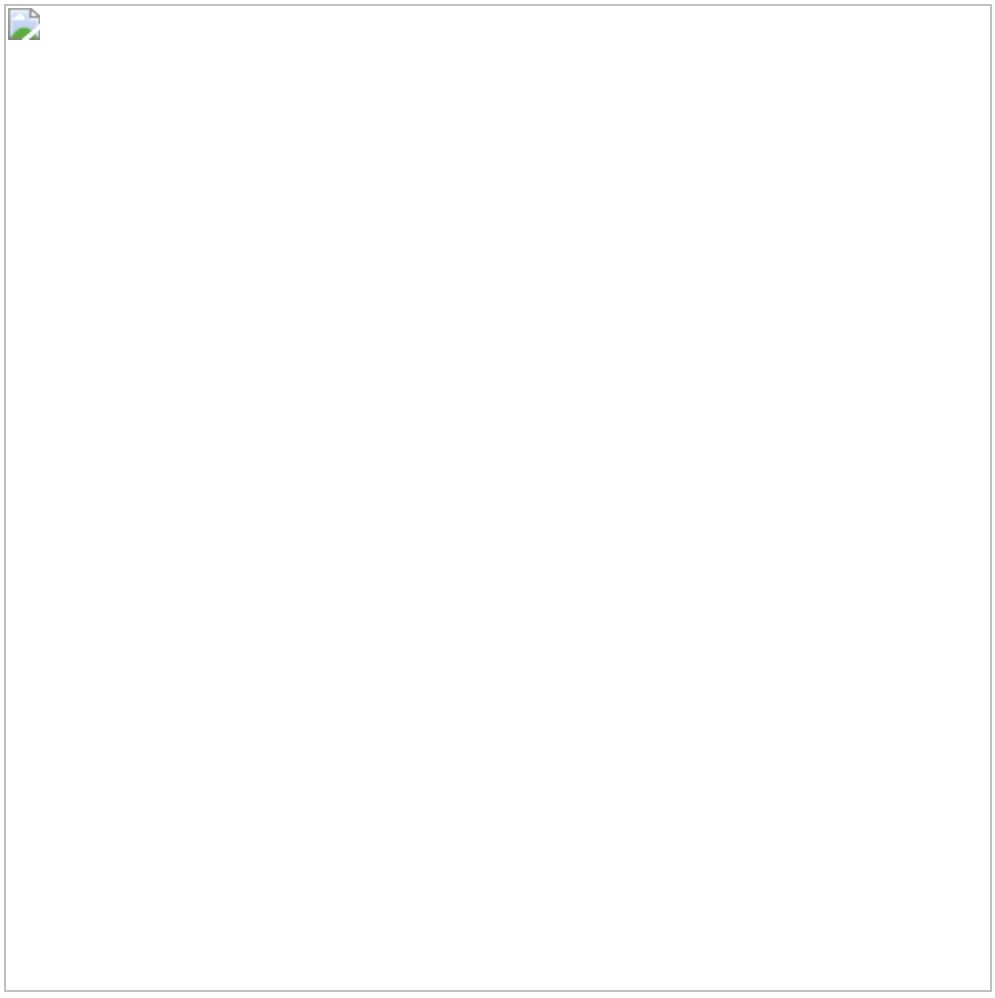
3. Huber loss : Combination of mean squared error and mean absolute error. More robust to outliers than MSE.

Loss functions for Classification

1. Binary Cross entropy loss
    a. Used for binary classification problems, goal is to predict one of 2 classes

2. Categorical Cross entropy loss
    a. Used for Multiclass classification, where goal is to predict one of classes

3. Hinge loss:
    a. Used for Support Vector machine and can be used for both binary and multiclass classification problems.



Loss function for reconstruction

- Mean squared error
- Mean absolute error
- Binary cross entropy loss

Refer from above

Hyperparameters

- Parameters in a machine learning model that are set prior to training and control the overall behavior of the model.
- Common hyper parameters :
    1. Number of layers: The number of layers in a neural network can have a large impact on its performance. deeper network with more layers can learn more complex representations of the data, but can also be more difficult to train due to the risk of overfitting.
    2. Number of units per layer: The number of units (neurons) in each layer can also impact the performance of the network. A larger number of units can allow the network to learn more complex representations of the data, but can also lead to overfitting.
    3. Learning rate: The learning rate controls the step size used to update the model parameters during training. A smaller learning rate will cause the model to converge more slowly, but will also reduce the risk of overshooting the optimal solution. A larger learning rate will cause the model to converge more quickly, but can cause the model to oscillate or miss the optimal solution.
    4. Mini-batch size: The mini-batch size is the number of training examples used in each iteration of stochastic gradient descent (SGD). Larger mini-batch size can speed up training, but can also increase the risk of overfitting. Asmaler mini-batch size can slow down training, but can also help prevent overfiting.
    5. Regularization strength: Regularization is a technique used to reduce overfiting ni amodel. Common forms of regularizationni neural networks include 1 regularization, 21 regularization, and dropout.

Learning Rate

- Determine the step size at which optimizer makes updates to model parameters. Controls the speed at which model learns and makes changes to parameters.
- High learning rate → model converge quickly but may result in overshooting the optimal soln.
- Low learning rate may converge a better soln but may take a long time.
- Good learning rate should make good progress in reasonable amount of time.
- Some cases adaptive learning rates are used to adjust learning rate during training.

Regularisation of Hyperparameters

- Prevent over fitting, where a model fits the training data too well, but performs poorly on new, unseen data.
- Regularization works by adding a penalty term to the loss function that the model optimizes during training. The penalty term discourages the model from fitting the training data too closely, encouraging it to have a more general solution.
    - L1 regularization adds a penalty term ot the loss function that si proportional to the absolute value of the model parameters. Encourages model to have sparse parameters.
    - 12 regularization adds a penalty term to the loss function that is proportional to the square of the magnitude of the model parameters. Encourages model to have dense solution.
    - Dropout is a regularization technique that randomly sets some of the activations in the model to zero during each forward pass.
- Amount of regularization applied is controlled by an hyper parameter called regilarization coefficient.

Momentum

- Its a hyperparameter in many optimisation algorithms used in ML.
- It determines the extent to which the optimiser takes the past gradients into account when making updates the model parameters.
- Gradient descent → Optimiser updates parameters in the direction of negative gradient of the loss function with respect to parameters.
- Momentum → Controls the amount of past gradient that is carried over to the current update.
- When momentum is set high → Optimiser takes into account a larger portion of past gradient which leads to convergence faster and better performance.
- Low momentum allows the optimizer to adapt to changes in gradient, but it results in slow convergence.
- In short, momentum is a hyperparameter used in optimisation algorithms.

It determines the extent to which optimiser

takes the past gradients into account when making updates to the parameters. The optimal value depends on the specifics of the model being optimised.

Sparcity in Hyperparameters

- Sparsity is a property of a model ni which many of the model parameters have a value close to zero. sparsity can be used as a form of regularization to prevent overfitting and improve the interpretability of a model.
- Sparcity is encouraged by
  - L1 regularization (definition above)
  - Sparse constraint methods enforce a hard constraint on the number of non-zero parameters in the model.
- Amount of sparcity is controlled by a hyperparameter which determine strength of sparcity.

Deep Feed Forward Neural Network

- Type of ANN composed of multiple layers of interconnected artificial neurons. Info flows in a straightforward manner from input to output without looping back.
- Nodes in each layer are connected to nodes in next layer. The connections between nodes have weights that can be adjusted in training. Weights represent the strength of connection between 2 nodes and determine the flow of info.
- The intermediate layers are hidden layers. They perform variety of tasks including classification, regression and generation. It understands complex representations of input data and make highly accurate predictions.
- EX OR Deep Feed Forward Network:
    - XOR is simple binary classification problem in which goal is to classify a given input into one of two classes based on whether the input satisfies the XOR relationship or not. Input is in one class if exactly one of its two input values is 1 and the other class otherwise.
    - DFNN can be used to solve XOR problem by learning non linear decision boundary that separates 2 classes.
    - Input layer [binary values] → hidden layer [perform computations to extract features] → output layer [make prediction]
    - Weights adjusted in training process. Adjusted in such a wat to minimize difference between predicted and actual values.

A **hidden unit** in a deep feedforward network is a node or neuron in the hidden layers of the network. It receives inputs from the previous layer, processes the information using an activation function, and passes the result to the next layer as output. The hidden units work together to perform complex non-linear transformations on the inputs, allowing the network to learn intricate relationships between inputs and outputs. The number of hidden units in a network can have a significant impact on its ability to learn and generalize to new data.

Cost Function

- Measure of the difference between the predicted output and the true output. Decide optimal weights for the network. Cost is calculated and is used to update the weights.
- Its a measure of how well the model's predictions match the actual values. Its used during the training phase to adjust the model weights to min the error between predicted and actual values.
- Different algo have diff functions. The choice of cost function depends on the specific problem and the desired output of the network. Some Cost functions :
    1. Mean Squared Error (MSE) : This measures the average squared difference between the predicted and actual output.
    2. Binary Cross-Entropy: This is used in binary classification problems, where the goal is to predict one of two possible outcomes.
    3. Categorical Cross-Entropy : This is used in multi-class classification problems, where the goal is to predict one of multiple possible outcomes.

Error Backpropogation

- Backpropagation is the process of calculating the gradients of the cost function with respect to the weights. The gradients are then used to update weights during training. Goal is to minimize the cost function by finding optimal weight.
- Gradients are calculated using backpropagation algo, which involves forward propagating the input through network to calculate predicted output and then backpropagating the error between predicted output and true output to calculate the gradient. Uses gradient of cost function with respect to weights to update the weights to reduce the cost.

Gradient Based Learning

- Method for optimizing the weights in a DFNN by minimizing the cost function. The cost function measures the difference between the predicted output and the true output, and the goal of the training process is to find the optimal weights that minimize this cost.
- Uses gradient of cost function with respect to weights to update the weights to reduce the cost. Gradients are calculated using backpropagation algo, which involves forward propagating the input through network to calculate predicted output and then backpropagating the error between predicted output and true output to calculate the gradient.
- It is simple efficient can handle high dimensional problems. Optimization is performed using techniques like gradient descent, stochastic gradient descent, Adam. Signmoid is used for activation.
- Vanishing and Exploding Gradients:
    - vanishing gradient problem occurs when the gradients become very small during backpropagation, making difficult for the optimization algorithm to update the weights effectively. This results in slow convergence or poor performance. Encountered in DL because there is multiplication involved making the gradient smaller everytime.
    - exploding gradient problem occurs when the gradients become very large during backpropagation, causing optimization algorithm to overshoot and produce large, unstable updates to the weights. This results the unstable convergence or divergence. Encountered when activation functions have high values, leading to larger gradients.
    - There are many techniques employed to solve this like weight initialization, activation function with bounded outputs (ReLu), normalization techniques (batch normalization).

Sentiment Analysis

- The goal of sentiment analysis is to categorize the text as positive, negative, or neutral in terms of the sentiment expressed.
- Approaches to sentiment analysis :

    - rule-based methods, which use a set of predefined rules to categorize the text
    - machine learning based methods, which train models on large annotated datasets to make predictions about the sentiment of new texts. involve preprocessing the text to remove stop words, stemming, and lemmatization, and then converting the text into numerical features that can be input into a model.
- Depends on the subjectivity of language and the nuances of sentiment expression. The accuracy can be increased with combination of multiple models and transfer learning.
- DFNN (Deep Forward NN) can be used for Sentiment Analysis. The input is representation of text like word embedding/bag of words. The output is a predicted label.
- To train a DFNN a labelled dataset of texts and their corresponding sentiments is used. Network is trained to minimize the difference between its predictions and the true sentiments. Weights are updated using optimization algorithm. It can be used to predict sentiment of new texts. They can capture complex relationships between input features and sentiment labels. Theyre expensive to train and achieve large labelled datasets to achieve good performance.

Pytorch

- Open Source ML library that provides high level interface for building and training DL models. Easy to use and fast.
- Provides a range of pre built Neural Network Architectures like feedforward network, convolutional neural network, recurrent neural network and transformers.
- Provides tools for data loading and preprocessing, evaluation and prediction.
- Install the library. To train a model, you need to define a loss function, an optimizer and a training loop.

    - Loss Function : measures the difference between the model's predictions and the true labels,
    - Optimizer : updates the model weights to minimize the loss.
    - Training Loop : iterates over the training data, updating the model weights and evaluating the performance of the model on a validation set.
- It also provides tools for deploying models to production.

Jupyter

- Open-source web-based platform for interactive computing that is commonly used for data science and machine learning tasks like DL. You can create and share documents that contain code, equations, etc.
- Supports variety of programming languages. You can run code directly in browser. You will need to install Jupyter Platform and libraries like PyTorch/TensorFlow for using it or DL.
- Provides a wide range of tools for visualizing and exploring the data, including Matplotlib and seaborn for data visualization, pandas for data manipulation. These are for preprocessing.
- It gives interactive elements which makes it easy to experiment with different hyperparameters and model configs. Its ideal for building and experimenting with deep learning models.

Colab

- Google colab is free cloud based platform for ml and dl research. It allows to run jupyter notebooks in cloud.
- It provides access to GPUs and TPUs making it easy to run large and computationally intensive DL models.
- You can connect to google drive to access, save your data and collaborate with others realtime. You can run code using libraries like PyTorch and

Tensorflow.

- You can go to website and create a nb. You will need a google account. It is a powerful tool used by many people for DL research.