FML MODULE 5

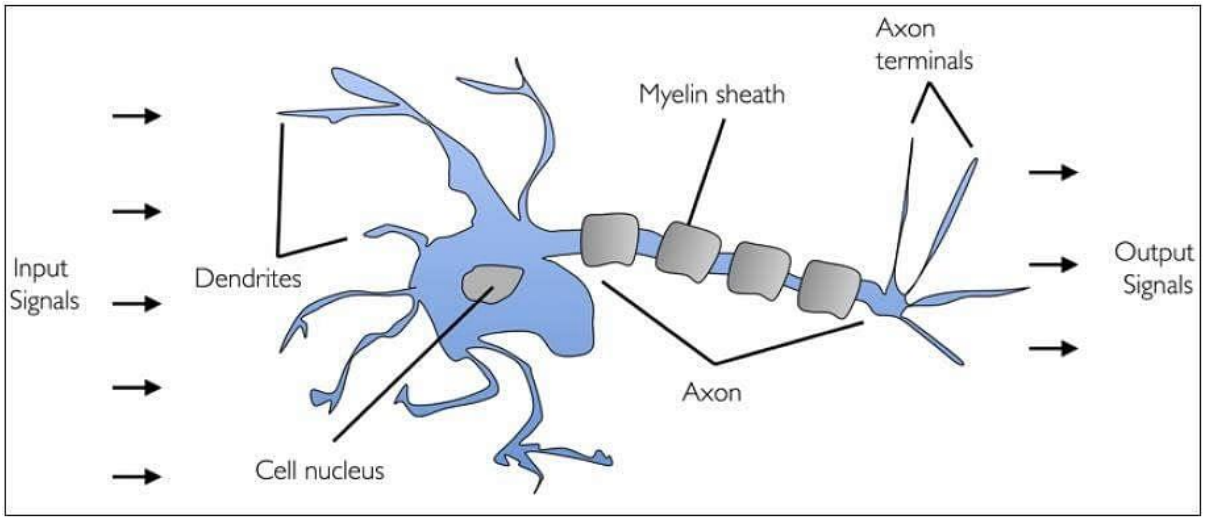
1)DEFINE NEURAL NETWORK? HOW IT RESEMBLES HUMAN BRAIN?

A neural network is a series of algorithms that endeavors to recognize underlying relationships in a set of data through a process that mimics the way the human brain operates. In this sense, neural networks refer to systems of neurons, either organic or artificial in nature.

Neural networks can adapt to changing input; so the network generates the best possible result without needing to redesign the output criteria. The concept of neural networks, which has its roots in artificial intelligence, is swiftly gaining popularity in the development of trading systems.

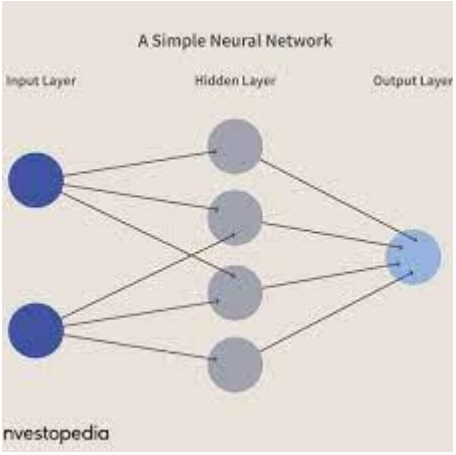
Biological Neuron

A human brain has billions of neurons. Neurons are interconnected nerve cells in the human brain that are involved in processing and transmitting chemical and electrical signals. Dendrites are branches that receive information from other neurons.



Cell nucleus or Soma processes the information received from dendrites. Axon is a cable that is used by neurons to send information. Synapse is the connection between an axon and other neuron dendrites.

An artificial neuron is a mathematical function based on a model of biological neurons, where each neuron takes inputs, weighs them separately, sums them up and passes this sum through a nonlinear function to produce output.



Simple neural network architecture

A basic neural network has interconnected artificial neurons in three layers:

Input Layer

Information from the outside world enters the artificial neural network from the input layer. Input nodes process the data, analyze or categorize it, and pass it on to the next layer.

Hidden Layer

Hidden layers take their input from the input layer or other hidden layers. Artificial neural networks can have a large number of hidden layers. Each hidden layer analyzes the output from the previous layer, processes it further, and passes it on to the next layer.

Output Layer

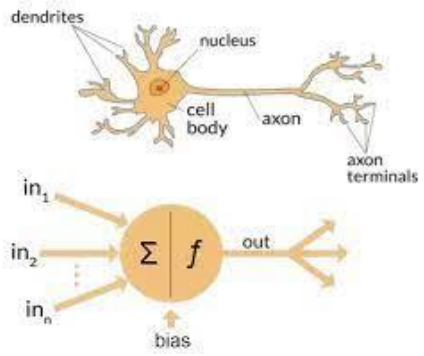
The output layer gives the final result of all the data processing by the artificial neural network. It can have single or multiple nodes. For

instance, if we have a binary (yes/no) classification problem, the output layer will have one output node, which will give the result as 1 or 0. However, if we have a multi-class classification problem, the output layer might consist of more than one output node.

Deep neural network architecture

Deep neural networks, or deep learning networks, have several hidden layers with millions of artificial neurons linked together. A number, called weight, represents the connections between one node and another. The weight is a positive number if one node excites another, or negative if one node suppresses the other. Nodes with higher weight values have more influence on the other nodes.

Theoretically, deep neural networks can map any input type to any output type. However, they also need much more training as compared to other machine learning methods. They need millions of examples of training data rather than perhaps the hundreds or thousands that a simpler network might need.



2)EXPLAIN THE CONCEPT OF BAYESIAN VIEW OF LEARNING IN NEURAL NETWORKS?

Bayesian statistics allow us to draw conclusions based on both evidence (data) and our prior knowledge about the world. This is often contrasted with frequentist statistics which only consider evidence. The prior knowledge captures our belief on which model generated the data, or what the weights of that model are. We can

represent this belief using a *prior distribution* p(w) over the model's weights.

As we collect more data we update the prior distribution and turn in into a *posterior distribution* using Bayes' law, in a process called *Bayesian updating*:

$$p(w|X, Y) = \frac{p(Y|X,w)p(w)}{p(Y|X)}$$

This equation introduces another key player in Bayesian learning — the *likelihood*, defined as p(y|x,w). This term represents how likely the data is, given the model's weights *w*.

Neural networks from a Bayesian perspective

A neural network's goal is to estimate the likelihood p(y|x,w). This is true even when you're not explicitly doing that, e.g. when you minimize MSE.

To find the best model weights we can use *Maximum Likelihood Estimation* (MLE):

$$\begin{aligned} w^{MLE} &= \text{argmax}_w \log P(D|w) \\ &= \text{argmax}_w \sum_i \log P(y_i|x_i, w) \end{aligned}$$

Alternatively, we can use our prior knowledge, represented as a prior distribution over the weights, and maximize the posterior distribution. This approach is called *Maximum Aposteriori Estimation* (MAP)**:**

$$\begin{aligned} w^{MAP} &= \text{argmax}_w \log P(w|D) \\ &= \text{argmax}_w \log P(D|w) + \log P(w) \end{aligned}$$

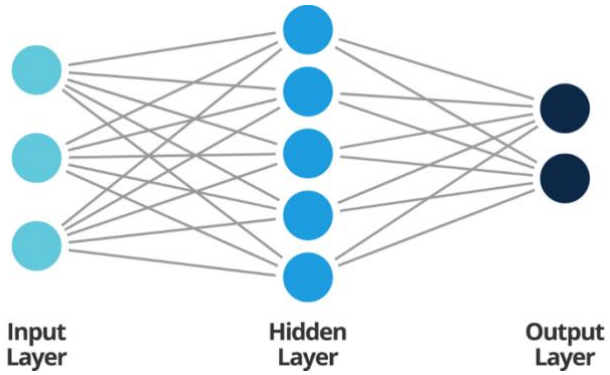
The term *logP(w)*, which represents our prior, acts as a regularization term.

3)HOW NEURAL NETWORKS SUPPORTS PARALLEL PROCESSING?

- Training and evaluation of each node in large networks can take an incredibly long time
- However, neural networks are "embarrassingly parallel"
  - Computations for each node are generally independent of all other node

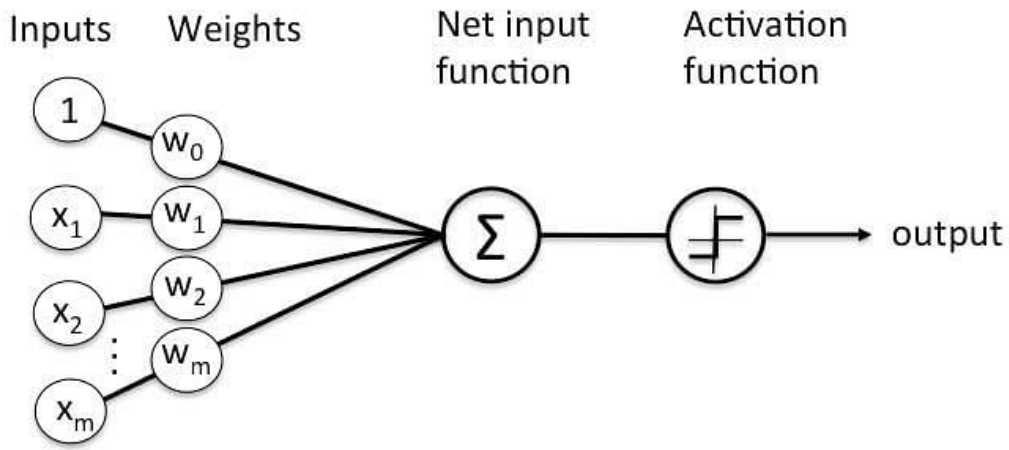
Typical structure of an ANN:

- For each training session
  - For each training example in the session
    - For each layer (forward and backward)
      - For each neuron in the layer
        - For all weights of the neuron
          - For all bits of the weight value



4)WHAT DO YOU MEAN BY PERCEPTRON ?EXPLAIN ITS ROLE IN NEURAL NETWORK?

Perceptron was introduced by Frank Rosenblatt in 1957. He proposed a Perceptron learning rule based on the original MCP neuron. A Perceptron is an algorithm for supervised learning of binary classifiers. This algorithm enables neurons to learn and processes elements in the training set one at a time.



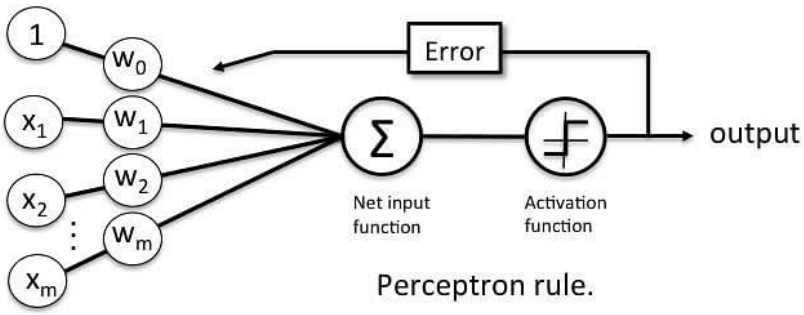
There are two types of Perceptrons: Single layer and Multilayer.

- Single layer - Single layer perceptrons can learn only linearly separable patterns
- Multilayer - Multilayer perceptrons or feedforward neural networks with two or more layers have the greater processing power

The Perceptron algorithm learns the weights for the input signals in order to draw a linear decision boundary.

This enables you to distinguish between the two linearly separable classes +1 and -1.

Perceptron Learning Rule states that the algorithm would automatically learn the optimal weight coefficients. The input features are then multiplied with these weights to determine if a neuron fires or not.



The Perceptron receives multiple input signals, and if the sum of the input signals exceeds a certain threshold, it either outputs a signal or does not return an output. In the context of supervised learning and classification, this can then be used to predict the class of a sample.

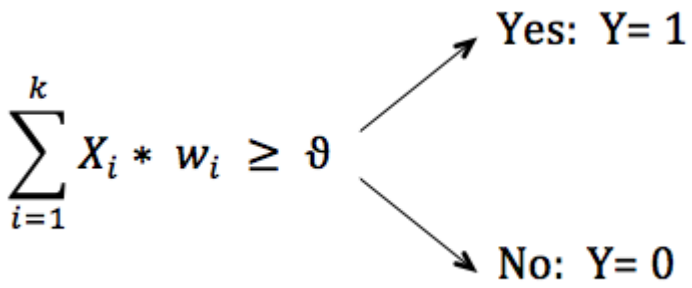
5)HOW DO TRIAN PERCEPTRON TO IMPLEMENT STOCHASTIC GRANDIENT DESCENT?

A perceptron unit is a linear function which acts as described above and which has an output of 1 or 0. Therefore the result of a perception unit is a line that separates two different planes where on one side the output is zero and on the other it is equal to 1.

In neural network there are two cases which we are going to consider:

- Thresholded output: perceptron rule

- Un-thresholded values: gradient descent/delta rule



Functions like “AND”, “OR” and “NOT” are expressible as perceptron units.

In regards to training a perceptron when given examples it’s a matter of finding the weights that map the inputs to the outputs.

Perceptron rule:

Where we have: *x* as input; *y* as target; *y^* as output; *n* as learning rate; *w* as weight.

The perceptron tries to find the delta-weight needed to adjust the weight in order to find the correct output when compared to a threshold.

(theta is treated as a bias (-theta corresponding weight) so we can ultimately compare to zero)

$$\textit{Repeat } x, y: \quad \{$$

$$w_i = w_i + \Delta w_i$$

$$\Delta w_i = n\left(y - \hat{y}\right) * x_i$$

$$\hat{y} = \left(\sum_{i=1}^{k} X_i * w_i \geq 0\right)$$

$$\}$$

That way it finds the separation line that differentiates the training examples.

When the data sets are linearly separable, then the perceptron will always find the line that separates them.

Gradient descent:

The gradient descent is instead used when there is non-linear separability among the data set and the output is not thresholded, but continuous.

In this case we have an "activation function" ("*a*"); we have an error which is defined as the sum of the squared values of the difference between the target and the activation. We then try to minimize this error by applying the derivative of it over the weights.

$$a = \sum_{i=1}^{k} X_i * w_i$$

$$Error = \frac{1}{2} \sum_{x,y \in D} (y - a)^2$$

6)EXPLAIN MULTILAYER PERCEPTRON ?

A **multilayer perceptron** (**MLP**) is a fully connected class of feedforward artificial neural network (ANN). The term MLP is used ambiguously, sometimes loosely to mean *any* feedforward ANN, sometimes strictly to refer to networks composed of multiple layers of perceptrons (with threshold activation); see § Terminology. Multilayer perceptrons are sometimes colloquially referred to as "vanilla" neural networks, especially when they have a single hidden layer.[1]

An MLP consists of at least three layers of nodes: an input layer, a hidden layer and an output layer. Except for the input nodes, each node is a neuron that uses a nonlinear activation function. MLP utilizes a supervised learning technique called backpropagation for training.[2][3] Its multiple layers and non-linear activation distinguish MLP from a linear perceptron. It can distinguish data that is not linearly separable.[4]

Input Layer

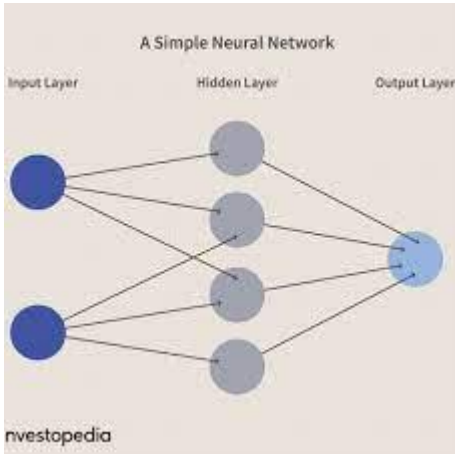
Information from the outside world enters the artificial neural network from the input layer. Input nodes process the data, analyze or categorize it, and pass it on to the next layer.

Hidden Layer

Hidden layers take their input from the input layer or other hidden layers. Artificial neural networks can have a large number of hidden layers. Each hidden layer analyzes the output from the previous layer, processes it further, and passes it on to the next layer.

Output Layer

The output layer gives the final result of all the data processing by the artificial neural network. It can have single or multiple nodes. For instance, if we have a binary (yes/no) classification problem, the output layer will have one output node, which will give the result as 1 or 0. However, if we have a multi-class classification problem, the output layer might consist of more than one output node.



7)EXPLAIN BACKPOPAGATION ALGORITHM?

Backpropagation is an algorithm that back propagates the errors from output nodes to the input nodes. Therefore, it is simply referred to as backward propagation of errors. It uses in the vast applications of neural networks in data mining like Character recognition, Signature verification, etc.

Backpropagation is a widely used algorithm for training feedforward neural networks. It computes the gradient of the loss function with respect to the network weights and is very efficient, rather than naively directly computing the gradient with respect to each individual weight. This efficiency makes it possible to use gradient methods to train multi-layer networks and update weights to minimize loss; variants such as gradient descent or stochastic gradient descent are often used.

The backpropagation algorithm works by computing the gradient of the loss function with respect to each weight via the chain rule, computing the gradient layer by layer, and iterating backward from the last layer to avoid redundant computation of intermediate terms in the chain rule.

Working of Backpropagation:

Neural networks use supervised learning to generate output vectors from input vectors that the network operates on. It Compares generated output to the desired output and generates an error report if the result does not match the generated output vector. Then it adjusts the weights according to the bug report to get your desired output.

Backpropagation Algorithm:

Step 1: Inputs X, arrive through the preconnected path.

Step 2: The input is modeled using true weights W. Weights are usually chosen randomly.

Step 3: Calculate the output of each neuron from the input layer to the hidden layer to the output layer.

Step 4: Calculate the error in the outputs

```
Backpropagation Error= Actual Output – Desired Output
```

Step 5: From the output layer, go back to the hidden layer to adjust the weights to reduce the error.

Step 6: Repeat the process until the desired output is achieved.

8)EXPLIAN TRAINING PROCEDURES IN DETAIL?

1. Pick a neural network architecture. This implies that you shall be pondering primarily upon the connectivity patterns of the neural network including some of the following aspects:
    - **Number of input nodes**: The way to identify number of input nodes is identify the number of features.
    - **Number of hidden layers**: The default is to use the single or one hidden layer. This is the most common practice.
    - **Number of nodes in each of the hidden layers**: In case of using multiple hidden layers, the best practice is to use same number of nodes in each hidden layer. In general practice, the number of hidden units is taken as comparable number to that of number of input nodes. That means one could take either the same number of hidden nodes as input nodes or maybe twice or thrice the number of input nodes.
    - **Number of output nodes**: The way to identify number of output nodes is to identify the number of output classes you want the neural network to process.
2. Random Initialization of Weights: The weights are randomly intialized to value in between 0 and 1, or rather, very close to zero.
3. Implementation of forward propagation algorithm to calculate hypothesis function for a set on input vector for any of the hidden layer.

4. Implementation of cost function for optimizing parameter values. One may recall that cost function would help determine how well the neural network fits the training data.
5. Implementation of back propagation algorithm to compute the error vector related with each of the nodes.
6. Use gradient checking method to compare the gradient calculated using partial derivatives of cost function using back propagation and using numerical estimate of cost function gradient. The gradient checking method is used to validate if the implementation of backpropagation method is correct.
7. Use gradient descent or advanced optimization technique with back propagation to try and minimize the cost function as a function of parameters or weights.

9)EXPLAIN KNN ALGORITHM IN DETAIL?

- K-Nearest Neighbour is one of the simplest Machine Learning algorithms based on Supervised Learning technique.
- K-NN algorithm assumes the similarity between the new case/data and available cases and put the new case into the category that is most similar to the available categories.
- K-NN algorithm stores all the available data and classifies a new data point based on the similarity. This means when new data appears then it can be easily classified into a well suite category by using K- NN algorithm.
- K-NN algorithm can be used for Regression as well as for Classification but mostly it is used for the Classification problems.
- K-NN is a **non-parametric algorithm**, which means it does not make any assumption on underlying data.
- It is also called a **lazy learner algorithm** because it does not learn from the training set immediately instead it stores the dataset and at the time of classification, it performs an action on the dataset.
- KNN algorithm at the training phase just stores the dataset and when it gets new data, then it classifies that data into a category that is much similar to the new data.
- **Example:** Suppose, we have an image of a creature that looks similar to cat and dog, but we want to know either it is a cat or dog. So for this identification, we can use the KNN algorithm, as it works on a similarity measure. Our KNN model will find the similar features of the new data set to the cats and dogs images and based on the most similar features it will put it in either cat or dog category.

he K-NN working can be explained on the basis of the below algorithm:

- **Step-1:** Select the number K of the neighbors
- **Step-2:** Calculate the Euclidean distance of **K number of neighbors**
- **Step-3:** Take the K nearest neighbors as per the calculated Euclidean distance.
- **Step-4:** Among these k neighbors, count the number of the data points in each category.
- **Step-5:** Assign the new data points to that category for which the number of the neighbor is maximum.
- **Step-6:** Our model is ready.

10)EXPLAIN THE IMPORTANCE OF THE STRUCTURAL ADAPTATION IN TUNING THE NETWORK SIZE?

Adaptation **helps neural networks to generalize and adapt the data easily**. Adaptive neural network framework is a collection of techniques in which structure of the network is adapted during the training according to a given problem

Tune the hyperparameters

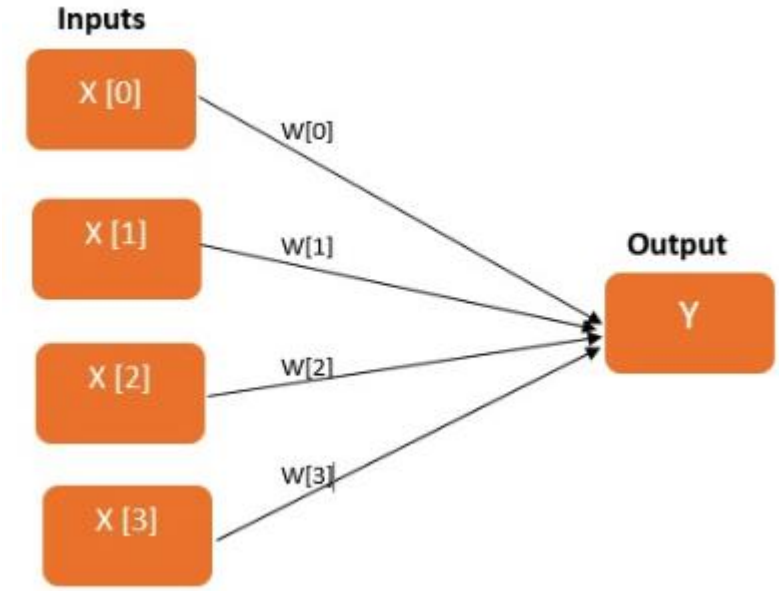
The first hyperparameter to tune is the number of neurons in each hidden layer. In this case, the number of neurons in every layer is set to be the same. It also can be made different. The number of neurons should be adjusted to the solution complexity. The task with a more complex level to predict needs more neurons. The number of neurons range is set to be from 10 to 100.

Tune the Layers

Layers in Neural Network also determine the result of the prediction model. A smaller number of layers is enough for a simpler problem, but a larger number of layers is needed to build a model for a more complicated problem. The number of layers can be tuned using the “for loop” iteration. This demonstration tune the number of layers two times. Each time, the number of layers is tuned between 1 to 3.

11)EXPLAIN THE USE OF BAYEAIAN APPROACH I N TRAINING THE NEURAL NETWORKS?

Bayesian Neural Networks (BNNs) refers to extending standard networks with posterior inference in order to control over-fitting. From a broader perspective, the Bayesian approach uses the statistical methodology so that everything has a probability distribution attached to it, including model parameters (weights and biases in neural networks). In programming languages, variables that can take a specific value will turn the same result every-time you access that specific variable.



Instead of taking into account a single answer to one question, Bayesian methods allow you to consider an entire distribution of answers. With this approach, you can naturally address issues such as:

- regularization (overfitting or not)
- model selection/comparison,without the need for a separate cross-validation data set

Bayesian neural nets are useful for solving problems in domains where data is scarce, as a way to prevent overfitting. Example applications are molecular biology and medical diagnosis (areas where data often come from costly and difficult experimental work).

Bayesian nets are universally usefu

They can obtain better results for a vast number of tasks however they are extremely difficult to scale to large problems.

BNNs allow you to automatically calculate an error associated with your predictions when dealing with data of unknown targets.

allow you to estimate uncertainty in predictions, which is a great feature for fields like medicine

12)EXPLIAN THE IMPORTANCE OF SAMMON MAPPING IN REDUCING THE DIMENSIONS OF NEURAL

NETWORK?

Sammon's mapping is a technique for transforming a dataset from a high-dimensional space (say, *d*-dimensional) to a space with lower dimensionality *m*. Many criteria can be optimized in this process, such as class separability in classification tasks (in Fisher's linear discriminant analysis) or the amount of variance retained in the mapped set (in PCA). Sammon's mapping tries to preserve inter-pattern distances. This is achieved by minimizing an error criterion which penalizes differences . In distances between points in the original space and the mapped space.

13)EXPLAIN TIME DELAY NEURAL NETWORKS?

Time delay neural network (**TDNN**)[1] is a multilayer artificial neural network architecture whose purpose is to 1) classify patterns with shift-invariance, and 2) model context at each layer of the network.

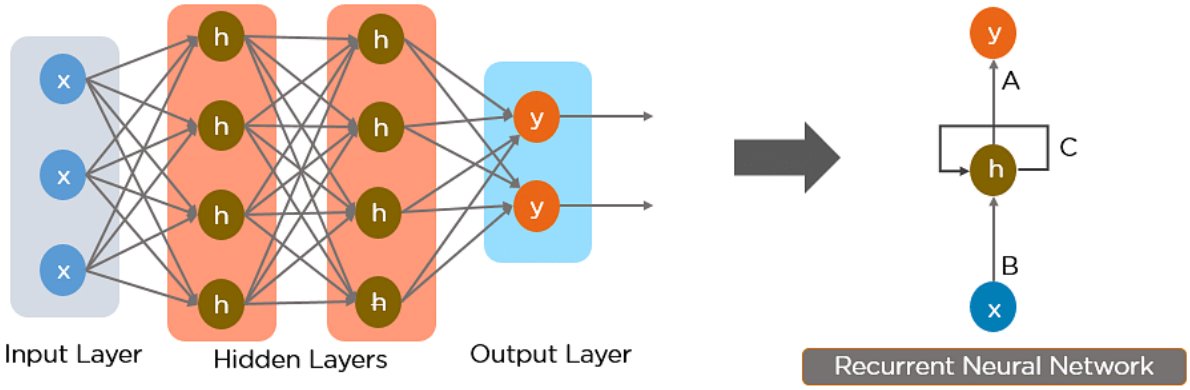
Shift-invariant classification means that the classifier does not require explicit segmentation prior to classification. For the classification of a temporal pattern (such as speech), the TDNN thus avoids having to determine the beginning and end points of sounds before classifying them.

For contextual modelling in a TDNN, each neural unit at each layer receives input not only from activations/features at the layer below, but from a pattern of unit output and its context. For time signals each unit receives as input the activation patterns over time from units below. Applied to two-dimensional classification (images, time-frequency patterns), the TDNN can be trained with shift-invariance in the coordinate space and avoids precise segmentation in the coordinate space.

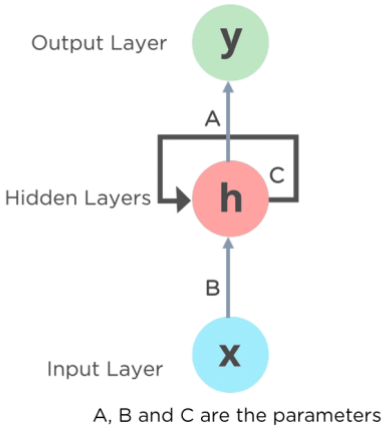
14)EXPLAIN THE IMPORTANCE OF RECURRENT METHODS IN DETAIL?

RNN works on the principle of saving the output of a particular layer and feeding this back to the input in order to predict the output of the layer.

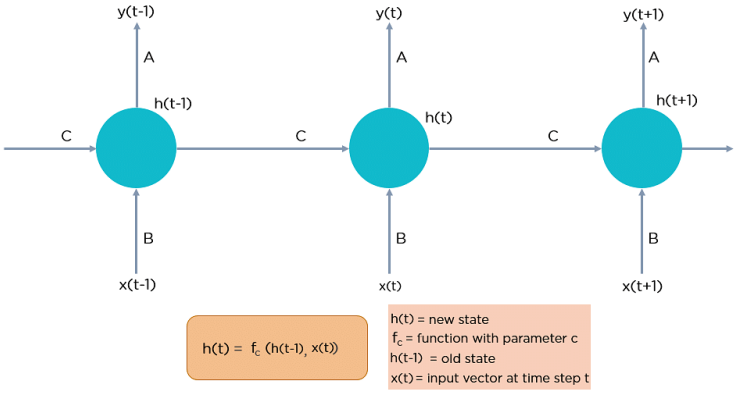
Below is how you can convert a Feed-Forward Neural Network into a Recurrent Neural Network:



The nodes in different layers of the neural network are compressed to form a single layer of recurrent neural networks. A, B, and C are the parameters of the network.



Here, “x” is the input layer, “h” is the hidden layer, and “y” is the output layer. A, B, and C are the network parameters used to improve the output of the model. At any given time t, the current input is a combination of input at x(t) and x(t-1). The output at any given time is fetched back to the network to improve on the output.



15)EXPLAIN KERNEL MACHINES AND EXPLAIN ITS IMPORTANCE IN DETAIL?

In machine learning, **kernel machines** are a class of algorithms for pattern analysis, whose best known member is the support-vector machine (SVM). The general task of pattern analysis is to find and study general types of relations (for example clusters, rankings, principal components, correlations, classifications) in datasets. For many algorithms that solve these tasks, the data in raw representation have to be explicitly transformed into feature vector representations via a user-

specified *feature map*: in contrast, kernel methods require only a user-specified *kernel*, i.e., a similarity function over pairs of data points in raw representation.

Kernel methods owe their name to the use of kernel functions, which enable them to operate in a high-dimensional, *implicit* feature space without ever computing the coordinates of the data in that space, but rather by simply computing the inner products between the images of all pairs of data in the feature space. This operation is often computationally cheaper than the explicit computation of the coordinates. This approach is called the "**kernel trick**".[1] Kernel functions have been introduced for sequence data, graphs, text, images, as well as vectors.

Algorithms capable of operating with kernels include the kernel perceptron, support-vector machines (SVM), Gaussian processes, principal components analysis (PCA), canonical correlation analysis, ridge regression, spectral clustering, linear adaptive filters and many others.

16)What is the significance of optimal separating hyper plane in SVM? AND WHAT IS HPERPLANE?

A hyperplane is **a decision boundary that differentiates the two classes in SVM**. A data point falling on either side of the hyperplane can be attributed to different classes. The dimension of the hyperplane depends on the number of input features in the dataset.

The optimal separating hyperplane is one of the core ideas behind the support vector machines. In particular, **it gives rise to the so-called support vectors which are the data points lying on the margin boundary of the hyperplane**.

17)EXPLIAN ABOUT SOFT MARGIN HYPERPLANE?

Soft Margin Classifier

In practice, real data is messy and cannot be separated perfectly with a hyperplane.

The constraint of maximizing the margin of the line that separates the classes must be relaxed. This is often called the soft margin classifier. This change allows some points in the training data to violate the separating line.

An additional set of coefficients are introduced that give the margin wiggle room in each dimension. These coefficients are sometimes called slack variables. This increases the complexity of the model as there are more parameters for the model to fit to the data to provide this complexity.

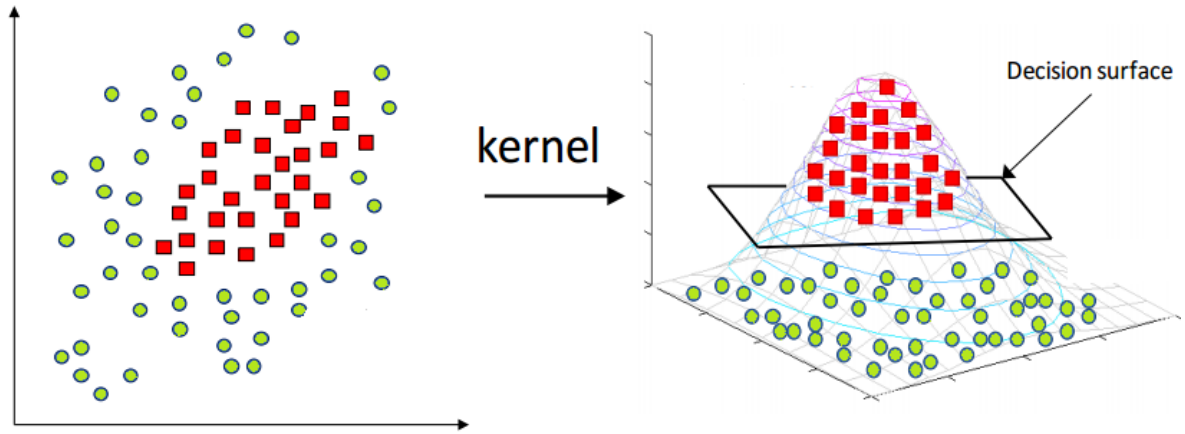
A tuning parameter is introduced called simply C that defines the magnitude of the wiggle allowed across all dimensions. The C parameters defines the amount of violation of the margin allowed. A C=0 is no violation and we are back to the inflexible Maximal-Margin Classifier described above. The larger the value of C the more violations of the hyperplane are permitted.

During the learning of the hyperplane from data, all training instances that lie within the distance of the margin will affect the placement of the hyperplane and are referred to as support vectors. And as C affects the number of instances that are allowed to fall within the margin, C influences the number of support vectors used by the model.

- The smaller the value of C, the more sensitive the algorithm is to the training data (higher variance and lower bias).
- The larger the value of C, the less sensitive the algorithm is to the training data (lower variance and higher bias).

18)EXPLAIN ABOUT KERNEL TRICK FOR CINVERSION OF NON LINEAR DATA TO LINEAR DATA?

Kernel Trick is widely used in Support Vector Machines (SVM) model to bridge linearity and non-linearity. It converts non-linear lower dimension space to a higher dimension space thereby we can get a linear classification. So, we are projecting the data with some extra features so that it can convert to a higher dimension space.



Let's suppose if we want to classify red squares and green dots its impossible to differentiate because it's in the non-linear form. In real-world also data is scattered and it's impossible to separate this data. So, we can use a Decision Surface where it can classify both green dots as well as red squares. Kernel Trick uses only the original feature space because when the dimension space increases it becomes more and more complex to classify.

Steps involved in SVM:-

i) Collects the Data and plot it accordingly

ii) Apply the Kernel Trick

iii) Learns Linear Line that classifies the data

iv) Projects back the data

To Conclude, Kernel Trick converts non-linear classification to linear classification by shifting the Lower Dimension Space to Higher Dimensional Space. There's lots of Math involved in Kernel Trick to learn the Linear Line that classifies the Data. ***Mercer's theorem*** helps to convert non-linear data points into Higher dimensions Space.

Mercer's Theorem determines which functions can be used as a kernel function. In mathematics, specifically functional analysis, Mercer's theorem states that a symmetric, and positive-definite matrix can be represented as a sum of a convergent sequence of product functions.

SVM algorithms use a set of mathematical functions that are defined as the kernel. The function of kernel is to take data as input and transform it into the required form. Different SVM algorithms use different types of kernel functions. These functions can be different types. For example ***linear, nonlinear, polynomial, radial basis function (RBF), and sigmoid.***

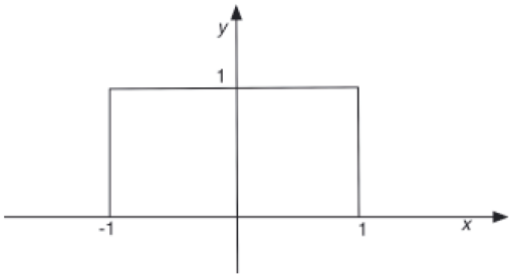
Introduce Kernel functions for sequence data, graphs, text, images, as well as vectors. The most used type of kernel function is **RBF.** Because it has localized and finite response along the entire x-axis.

The kernel functions return the inner product between two points in a suitable feature space. Thus by defining a notion of similarity, with little computational cost even in very high-dimensional spaces.

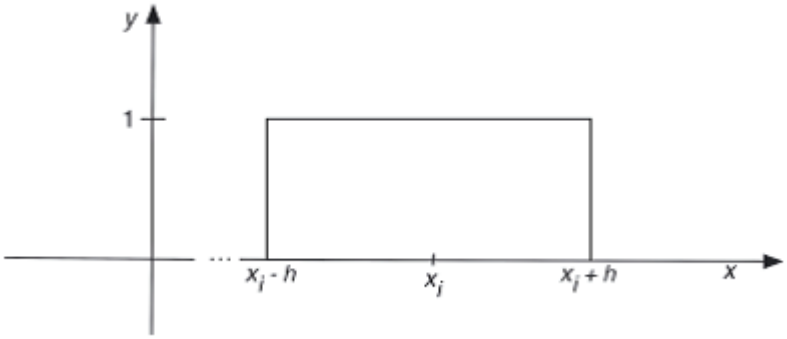
Define kernel or a window function as follows:

$$K\left(\overline{x}\right) = \begin{matrix} 1 & \text{if} \left\|\overline{x}\right\| \leq 1 \\ 0 & \text{otherwise} \end{matrix}$$

This value of this function is 1 inside the closed ball of radius 1 centered at the origin, and 0 otherwise . As shown in the figure below:



For a fixed xi, the function is K(z-xi)/h) = 1 inside the closed ball of radius h centered at xi, and 0 otherwise as shown in the figure below:



19)EXPLAIN VARIOUS GENERAL PURPOSE KERNEL FUNCTIONS?

Kernel functions

Kernel Function is a method used to take data as input and transform it into the required form of processing data. “Kernel” is used due to a set of mathematical functions used in Support Vector Machine providing the window to manipulate the data. So, Kernel Function generally transforms the training set of data so that a non-linear decision surface is able to transform to a linear equation in a higher number of dimension spaces. Basically, It returns the inner product between two points in a standard feature dimension.

A kernel is a function used in SVM for helping to solve problems. They provide shortcuts to avoid complex calculations. The amazing thing about kernel is that we can go to higher dimensions and perform smooth calculations with the help of it.

Polynomial Kernel Function

The polynomial kernel is a general representation of kernels with a degree of more than one. It's useful for image processing.

There are two types of this:

- **Homogenous Polynomial Kernel Function**

$\mathbf{K(x_i,x_j) = (x_i.x_j)^d}$, where '.' is the dot product of both the numbers and d is the degree of the polynomial.

- **Inhomogeneous Polynomial Kernel Function**

$\mathbf{K(x_i,x_j) = (x_i.x_j + c)^d}$ where c is a constant.

2. Gaussian RBF Kernel Function

RBF is the radial basis function. This is used when there is no prior knowledge about the data.

It's represented as:

$\mathbf{K(x_i,x_j) = exp(-\gamma||x_i - x_j||)^2}$

3. Sigmoid Kernel Function

This is mainly used in neural networks. We can show this as:

$\mathbf{K(x_i,x_j) = tanh(\alpha x^a y + c)}$

4. Hyperbolic Tangent Kernel Function

This is also used in neural networks. The equation is:

$\mathbf{K(x_i,x_j) = tanh(kx_i.x_j + c)}$

5. Linear Kernel Function

This kernel is one-dimensional and is the most basic form of kernel in SVM. The equation is:

$\mathbf{K(x_i,x_j) = x_i.x_j + c}$

6. Graph Kernel Function

This kernel is used to compute the inner on graphs. They measure the similarity between pairs of graphs. They contribute in areas like **bioinformatics**, **chemoinformatics**, etc.

7. String Kernel Function

This kernel operates on the basis of strings. It is mainly used in areas like **text classification**. They are very useful in text mining, genome analysis, etc.

8. Tree Kernel Function

This kernel is more associated with the tree structure. The kernel helps to split the data into tree format and helps the SVM to distinguish between them. This is helpful in language classification and it is used in areas like **NLP**.

20)DESCRIBE IN DETAIL CONSTRUCTING A NEW KERNEL BY COMBINING NEW KERNELS?

Constructing kernels from kernels

A function is a valid **kernel function** if it is a real-valued positive definite function, whose definition is recalled below.

> *A real-valued function K on $\mathcal{X}^2$ is called a positive definite function if it is symmetric and*
>
> $$\forall n \in \mathbb{N}^*,\ \forall \{\boldsymbol{x}_i\}_{i=1}^n \in \mathcal{X}^n,\ \forall \{a_i\}_{i=1}^n \in \mathbb{R}^n,\quad \sum_{i=1}^n \sum_{j=1}^n a_i a_j K(\boldsymbol{x}_i, \boldsymbol{x}_j) \geq 0.$$

This condition can be written more compactly in terms of the positive semi-definiteness of the kernel matrix (or Gram matrix), which is defined for a data set $\{\boldsymbol{x}_i\}_{i=1}^n \in \mathcal{X}^n$ as

$$\boldsymbol{K} = \begin{bmatrix} K(\boldsymbol{x}_1, \boldsymbol{x}_1) & \dots & K(\boldsymbol{x}_1, \boldsymbol{x}_n) \\ \vdots & & \\ K(\boldsymbol{x}_n, \boldsymbol{x}_1) & \dots & K(\boldsymbol{x}_n, \boldsymbol{x}_n) \end{bmatrix}.$$

With these notations a function K is a valid kernel if, for any set of points $\{\boldsymbol{x}_i\}_{i=1}^n \in \mathcal{X}^n$, it gives rise to a kernel matrix $\boldsymbol{K}$ that is positive semi-definite, i.e., if

$$\forall \boldsymbol{a} \in \mathbb{R}^n,\quad \boldsymbol{a}^T \boldsymbol{K} \boldsymbol{a} \geq 0.$$

In all the following, K_1 and K_2 are assumed to be valid kernel functions and $\boldsymbol{K}_1$ and $\boldsymbol{K}_2$ their respective Gram matrices.

Scalar multiplication

The validity of a kernel is conserved after multiplication by a positive scalar, i.e., for any $\alpha \geq 0$, the function

$$K(\boldsymbol{x}, \boldsymbol{x}') = \alpha K_1(\boldsymbol{x}, \boldsymbol{x}')$$

is a valid kernel. Show the proof

Adding a positive constant

For any positive constant $\alpha \geq 0$, the function

$$K(\boldsymbol{x}, \boldsymbol{x}') = K_1(\boldsymbol{x}, \boldsymbol{x}') + \alpha$$

is a valid kernel function. Show the proof

Linear combination **(includes the sum of two kernels as a special case)**

A linear combination of kernel functions involving only positive weights, i.e.,

$$K(\boldsymbol{x}, \boldsymbol{x}') = \sum_{j=1}^m \alpha_j K_j(\boldsymbol{x}, \boldsymbol{x}'), \qquad \text{with } \alpha_j \geq 0,$$

is a valid kernel function. Show the proof

Product

The product of two kernel functions, i.e.,

$$K(\boldsymbol{x}, \boldsymbol{x}') = K_1(\boldsymbol{x}, \boldsymbol{x}')K_2(\boldsymbol{x}, \boldsymbol{x}'),$$

is a valid kernel function. Show the proof

Polynomial functions of a kernel output

Given a polynomial $P : \mathbb{R} \to \mathbb{R}$ with *positive coefficients*, the function

$$K(\boldsymbol{x}, \boldsymbol{x}') = P\left(K_1(\boldsymbol{x}, \boldsymbol{x}')\right)$$

is a valid kernel. Show the proof

Exponential function of a kernel output

The function

$$K(\boldsymbol{x}, \boldsymbol{x}') = \exp\left(K_1(\boldsymbol{x}, \boldsymbol{x}')\right)$$

is a valid kernel. Show the proof