UNIT III

Probabilistic Modeling: Classification by Density Estimation, Statistical Estimation, Naïve Bayes Models, Prediction [Reference 1] Neural Networks: Bio-inspired Multi-Layer Networks, The Back-propagation Algorithm, Initialization and Convergence of Neural Networks, Beyond two layers, Breadth vs Depth, Basis Functions

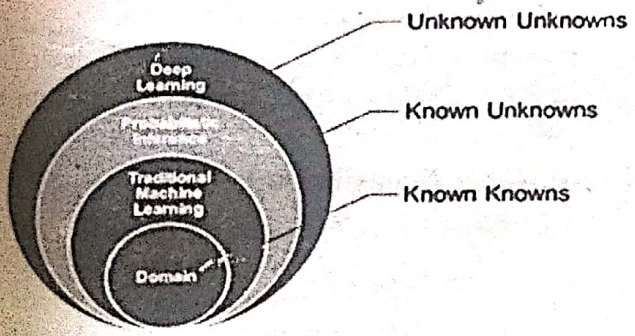
3.1 PROBABILISTIC MODELING

Q1. What are Probabilistic Models in Machine Learning?

Ans :

Introduction

Probabilistic Models in Machine Learning is the use of the codes of statistics to data examination. It was one of the initial methods of machine learning. It's quite extensively used to this day. Individual of the best-known algorithms in this group is the Naive Bayes algorithm.



Probabilistic modelling delivers a framework for accepting what learning is. The probabilistic framework defines how to signify and deploy reservation about models. Predictions have a dominant role in scientific data analysis. Their role is also so important in machine learning, automation, cognitive computing and artificial intelligence.

DescriptionProbabilistic models are presented as a prevailing idiom to define the world. Those were described by using random variables for example building blocks believed together by probabilistic relationships.There are probabilistic models along with non-probabilistic models in machine learning. The information about basic concepts of probability for example random variables and probability distributions would be helpful in order to have a well understanding of probabilistic models.Portrayal inference from noisy or ambiguous data is an imperative part of intelligent systems. In probability theory particularly.

Bayes' theorem helps as a principled framework of combining prior knowledge and empirical evidence.Importance of probabilistic ML models:One of the key benefits of probabilistic models is that they give an idea about the uncertainty linked with predictions. We may get an idea of how confident a machine learning model is on its prediction. For example, if the probabilistic classifier allocates a probability of 0.9 for the 'Dog' class in its place of 0.6, it means the classifier is extra confident that the animal in the image is a dog. These concepts connected to uncertainty and confidence are very valuable when it originates to critical machine learning uses for example disease diagnosis and autonomous driving. Moreover, probabilistic consequences would be worthwhile for many methods linked to Machine Learning for instance

Active Learning.Bayesian Inference:At the center of Bayesian inference is the Bayes' rule sometimes called Bayes' theorem. It is used to define the probability of a hypothesis with former knowledge. It is contingent on conditional probability.The formula for Bayes' theorem is known as;

$$P(\text{hypothesis} \% \text{data}) = P(\text{data} \% \text{hypothesis}) = P(\text{hypothesis})/P(\text{data})$$

Bayes rule states that how to do inference about hypotheses from data.Learning and prediction may be understood as forms of inference.The typical Bayesian inference with Bayes' rule is needing for a mechanism to straight regulate

the target posterior distribution. For example, the inference process is a one-way procedure that plans the earlier distribution to the posterior by detecting empirical data. In supervised learning and reinforcement learning, our final goal is to put on the posterior to learning tasks. That is applied with some measurement on the performance for instance prediction error or expected reward.

An upright posterior distribution should have a small prediction error or a great expected reward. Furthermore, by way, the large scale knowledge bases are built and crowdsourcing platforms are broadly accepted to gather human data, it is needed to include the outside information into statistical modelling and inference when building an intelligent system.

Naive Bayes algorithm

Naive Bayes algorithm is a supervised learning algorithm. It is created on the Bayes theorem and used for resolving sorting problems. It is chiefly used in text classification that comprises a high-dimensional training dataset. The naïve Bayes algorithm is one of the simple and best operational Classification algorithms that support construction the of fast machine learning models which may create rapid predictions.

The naive Bayes algorithm is a probabilistic classifier. It means that it predicts on the basis of the probability of an object. More or less prevalent instances of Naïve Bayes Algorithm are;

Spam filtration

Sentimental analysis

Classifying articles

A narrowly correlated model is the logistic regression. That is sometimes well thought-out to be the "hello world" of modern machine learning. Don't be deceived by its name as log reg is a classification algorithm somewhat a regression algorithm. Considerably like Naive Bayes, up till now, it's quite useful to this day as log reg predates computing for a long time, Thanks to its modest and multipurpose nature. It's frequently the first thing a data scientist would attempt on a dataset to become a feel for the classification task at hand.

Types of Naive Bayes Model:

There are the following three types of Naive Bayes Model:

1. **Gaussian**

   The Gaussian model takes responsibility that features monitor a normal distribution. This means that if analysts take nonstop values rather than separate, then the model takes up that these values are tested from the Gaussian distribution.

2. **Multinomial**

   It is used when the data is multinomial circulated. It is mainly used for document classification problems. It means a specific document goes to that category for example Sports, education, and Politics etc. The classifier uses the rate of words for the predictors.

3. **Bernoulli**

   The Bernoulli classifier do work alike to the Multinomial classifier. Then the predictor variables are the self-governing Booleans variables. For example, if a specific word is present or not in a document. This model is as well well-known for document classification tasks.

Uses of Naive Bayes Model

- The Naive Bayes Classifier used;
- For Credit Scoring.
- In medical data classification.
- It may be used in real-time predictions as Naïve Bayes Classifier is a keen learner.
- In-Text classification for example Spam filtering and Sentiment analysis.
- Pros and Cons of Bayes Classifier

Pros

Naive Bayes is one of the easy and fast machine learning algorithms to foresee a class of datasets.It may be used for Binary also as Multi-class Classifications.It does well in Multi-class predictions for example likened to the other Algorithms.It is the greatest widespread selection for text classification problems.

Cons

Naive Bayes accepts that all sorts are autonomous or disparate. Therefore it cannot learn the association between features. Objective Functions We may gaze at its Objective Function permissible to recognize whether a specific model is probabilistic or not. We wish to enhance a model to excel at an exact task in machine learning. The target of having an objective function is to deliver a value based on the model's outputs. Therefore, optimization may be done by moreover exploiting or curtailing the actual value. Usually, the objective is to reduce prediction error in Machine Learning. Therefore, we describe what is called a loss function for example the objective function and attempts to reduce the loss function in the training phase of a machine learning model.

3.2 Classification by Density Estimation

Q2. Explain about classification by density estimation.

Ans :

Probability density is the relationship between observations and their probability.

Some outcomes of a random variable will have low probability density and other outcomes will have a high probability density.

The overall shape of the probability density is referred to as a probability distribution, and the calculation of probabilities for specific outcomes of a random variable is performed by a probability density function, or PDF for short.

It is useful to know the probability density function for a sample of data in order to know whether a given observation is unlikely, or so unlike', as to be considered an outlier or anomaly and whether it should be removed. It is also helpful in order to choose appropriate learning methods that require input data to have a specific probability distribution.

It is unlikely that the probability density function for a random sample of data is known. As such, the probability density must be approximated using a process known as probability density estimation.

Histogram plots provide a fast and reliable way to visualize the probability density of a data sample.

Parametric probability density estimation involves selecting a common distribution and estimating the parameters for the density function from a data sample. Nonparametric probability density estimation involves using a technique to fit a model to the arbitrary distribution of the data, like kernel density estimation.

Kick-start your project with my new book Probability for Machine Learning, including step-by-step tutorials and the Python source code files for all examples.Let's get started.

Probability Density

A random variable x has a probability distribution p(x).

The relationship between the outcomes of a random variable and its probability is referred to as the probability density, or simply the "density."

If a random variable is continuous, then the probability can be calculated via probability density function, or PDF for short. The shape of the probability density function across the domain for a random variable is referred to as the probability distribution and common probability distributions have names, such as uniform, normal, exponential, and so on.

Given a random variable, we are interested in the density of its probabilities.

For example, given a random sample of a variable, we might want to know things like the shape of the probability distribution, the most likely value, the spread of values, and other properties.

Knowing the probability distribution for a random variable can help to calculate moments of the distribution, like the mean and variance, but can also be useful for other more general considerations, like determining whether an observation is unlikely or very unlikely and might be an outlier or anomaly.

The problem is, we may not know the probability distribution for a random variable. We rarely do know the distribution because we don't have access to all possible outcomes for a random

variable. In fact, all we have access to is a sample of observations. As such, we must select a probability distribution.

This problem is referred to as probability density estimation, or simply "density estimation," as we are using the observations in a random sample to estimate the general density of probabilities beyond just the sample of data we have available.There are a few steps in the process of density estimation for a random variable.

The first step is to review the density of observations in the random sample with a simple histogram. From the histogram, we might be able to identify a common and well-understood probability distribution that can be used, such as a normal distribution. If not, we may have to fit a model to estimate the distribution.

In the following sections, we will take a closer look at each one of these steps in turn.

We will focus on univariate data, e.g. one random variable, in this post for simplicity. Although the steps are applicable for multivariate data, they can become more challenging as the number of variables increases.

Summarize Density with a Histogram

The first step in density estimation is to create a histogram of the observations in the random sample.

A histogram is a plot that involves first grouping the observations into bins and counting the number of events that fall into each bin. The counts, or frequencies of observations, in each bin are then plotted as a bar graph with the bins on the x-axis and the frequency on the y-axis.

The choice of the number of bins is important as it controls the coarseness of the distribution (number of bars) and, in turn, how well the density of the observations is plotted. It is a good idea to experiment with different bin sizes for a given data sample to get multiple perspectives or views on the same data.

For example, observations between 1 and 100 could be split into 3 bins (1-33, 34-66, 67-100), which might be too coarse, or 10 bins (1-10, 11-20, ... 91-100), which might better capture the density.

A histogram can be created using the Matplotlib library and the hist() function. The data is provided as the first argument, and the number of bins is specified via the "bins" argument either as an integer (e.g. 10) or as a sequence of the boundaries of each bin (e.g. [1, 34, 67, 100]).The snippet below creates a histogram with 10 bins for a data sample.

```
1...
2# plot a histogram of the sample
3.pyplot.hist(sample,bins=10)
4.pyplot.show()
```

We can create a random sample drawn from a normal distribution and pretend we don't know the distribution, then create a histogram of the data. The normal() NumPy function will achieve this and we will generate 1,000 samples with a mean of 0 and a standard deviation of 1, e.g. a standard Gaussian.

The complete example is listed below.

```
1.  # example of plotting a histogram of a random sample
2.  from matplotlib import pyplot
3.  from numpy.random import normal
4.  # generate a sample
5.  sample=normal(size=1000)
6.  # plot a histogram of the sample
7.  pyplot.hist(sample,bins=10)
8.  pyplot.show()
```

Running the example draws a sample of random observations and creates the histogram with 10 bins. We can clearly see the shape of the normal distribution.

Note that your results will differ given the random nature of the data sample. Try running the example a few times.

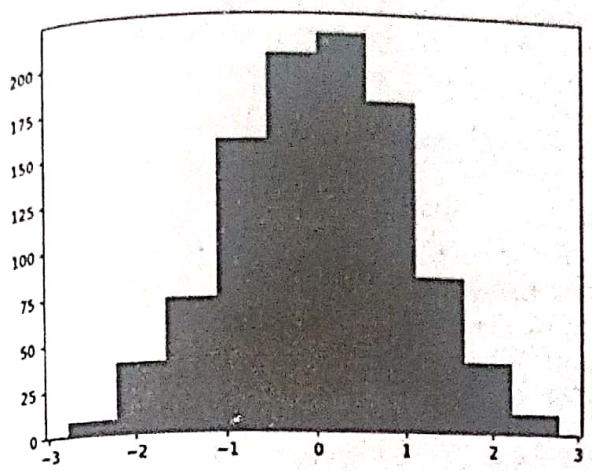


Fig.: Histogram plot with 10 Bins of Random Sample

Running the example with bins set to 3 makes the normal distribution less obvious.

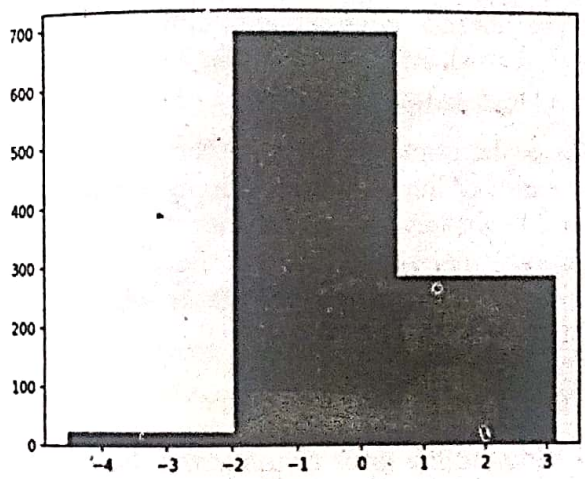


Fig.: Histogram plot with 3 Bins of Random Sample

Reviewing a histogram of a data sample with a range of different numbers of bins will help to identify whether the density looks like a common probability distribution or not.

In most cases, you will see a unimodal distribution, such as the familiar bell shape of the normal, the flat shape of the uniform, or the descending or ascending shape of an exponential or Pareto distribution.You might also see complex distributions, such as multiple peaks that don't disappear with different numbers of bins, referred to as a bimodal distribution, or multiple peaks, referred to as a multimodal distribution. You might also see a large spike in density for a given value or small range of values indicating outliers, often occurring on the tail of a distribution far away from the rest of the density.

3.3 Statistical Estimation

Q3. Write and explain Statistical Estimation.

Ans :

Statistics, as we know, is the study of gathering data, summarizing & visualizing the data, identifying patterns, differences, limitations and inconsistencies and extrapolating information regarding the population from a sample.

This process of extrapolating information regarding the population from a sample is called Estimation. As it's impossible to collect information from every single member of the population we instead gather information from a sample and work our way to estimate the information absout the population. And in this process, we use something called an "Estimator" to generate the estimation.

When a value is calculated for the entire population it's called a parameter and the corresponding term for a subset of the population also known as the sample is called a statistic.

Estimator

'u' (read mu) is referred to as the Mean of the population or the true average of the population. But in most cases, we don't know this value, so instead, we try to determine a statistic called xbar which is the Mean of the sample. Xbar then becomes our estimator.

It may seem reasonable to use xbar to estimate u. But let's suppose I take on this mission to determine the average height of all the people in my city. It's really not possible for me to go to every single person in the city and ask them their heights. So instead I decide to choose a smaller sample i.e. people in my building to estimate the average height of people. Owing to my absent-mindedness, I end up making an incorrect entry of putting the decimal at the wrong place for one of the values (8.0 becomes 80.0).

My sample looks like this in inches: [4,5.3,5.2,5.5,5.8,6.0,6.1,80.0]. The mean 'xbar' of this sample is 14.73 inches. Are people really that tall on average? 'Xbar' in this case doesn't seem like the best choice for estimating 'u'.

So instead I decide to use another statistic called Median. The Median is nothing but the middle value. And the median for this sample is 5.65' which seems much more reasonable and close to the Is my estimator biased or unbiased?Variance is a commonly used estimator which is used to determine the spread in the data usually given by the following formula:

$$S^2 = \frac{1}{n}\sum(x_i - \bar{x})^2$$

But this formula of variance tends to underestimate the value of variance when we move to a larger sample (or population). In other words, it tends to be biased. Bias is nothing but the difference between the expected value and actual value or 'xbar-u'. When this bias equals 0 we say that the estimator is unbiased. The above formula yields a non-zero bias.

So, instead, we use the following formula to calculate the sample variance where we replace 1/n by 1/(n-1)

$$S^2_{n-1} = \frac{1}{n-1}\sum(x_i - \bar{x})^2$$

This formula yields a bias that is equal to zero. It also doesn't underestimate population variance. Intuitively, since the denominator is a smaller number now, the value of variance is larger, and hence for a larger sample (or the population), we naturally expect a larger variance.

The sample mean is always an unbiased estimator of the population mean. That is because the expected value of mean equals the actual value or true mean of the population. Some samples might have a larger mean than the population mean and some might have a sample mean lower than the population mean. However, when this process is repeated over many iterations and the average of the estimates is calculated over these iterations, the mean of these sampling experiments will eventually equal the population mean.

Determination of best estimator

That really depends on whether we are trying to minimize the error or maximize the chance of getting the right answer.If we are trying to minimize the error we use something called Mean Squared Error or Root Mean Squared Error. In a real experiment, the process of calculating the estimator is iterated many times over numerous different samples. In the absence of an outlier, the sample mean, 'xbar', minimizes the Mean Squared Error (MSE) :

$$MSE = \frac{1}{m}\sum(\bar{x} - \mu)^2$$

where 'm' is the number of iterations. RMSE is nothing but the Square root of MSE.But is minimizing MSE/RMSE always the best choice? Imagine the case of a dice roll. The mean of a dice roll is (1+2+3+4+5+6)/6 = 3.5. We can never roll a 3.5 on a dice. Again, imagine we roll a 6 sided dice 3 times and we are asked to estimate the sum of the dice rolls. If we use the MSE approach and try to determine the value that minimizes the MSE, we would conclude that the expected value of the sum would be 3*3.5 = 10.5.

But the sum of dice rolls will never be a decimal number. In this case, we should choose an estimator which has the highest chance of getting the answer right also called Maximum Likelihood Estimator. For the dice roll case if we say the sum of 3 dice rolls would be 3 or 10 or 12, our chances of getting the answer right increase.

Distribution of the estimators

Going back to my inquisitiveness for finding the average height of the people in my city. I picked up a sample of people in my building (where evidently I committed a mistake which I eventually corrected) and calculated a statistic. The mean of the sample. The sample size was 8 there. I wasn't completely satisfied with my analysis, so I decided to iterate this process over more samples of the same size. I decided to extend this sample collection to 4 of my friends' buildings. Each sample then yielded a statistic (Incorrigible me is still sticking with the sample mean despite its flaws and inability to cope up with outliers).

Following are the samples I have:

Sample1:[4,5.3,5.2,5.5,5.8,6.0,6.1,8.0] Mean1: 5.74 Std dev: 1.13

Sample2:[4.8,6.1,6.3,5.0,5.5,5.9,5.8,5.6] Mean2: 5.62 Std dev: 0.51

Sample3:[6.1,6.2,6.3,6.0,5.11,5.10,6.0,6.1]Mean3: 5.86 Std dev: 0.48

Sample4:[5.1,5.2,5.4,5.9,5.5,5.10,5.9,5.5] Mean4: 5.45 Std dev: 0.32

Sample5:[2.5,4.11,5.7,5.3,5.8,5.9,5.2,5.1] Mean5: 4.95 Std dev: 1.14

Now that I have some basic idea of mean heights, I decided to generate simulation over some 100 samples and calculate the means of each sample. I am using a normal distribution with a mean of 5.5, a standard deviation of 0.5 and a sample size of 8 to generate the sample means.

```
import numpy as np
import random
means = [5.74,5.62,5.86,5.45,4.95]
for i in range(100):
    x = np.random.normal(5.5, 0.5, 8)
xbar = np.mean(x)
means.append(xbar)
```

This very distribution of the statistics i.e. the sample means is called the sampling distribution. Visualizing the distribution of the sample means:

```
import seaborn as sns
sns.displot(means,color='green')
```

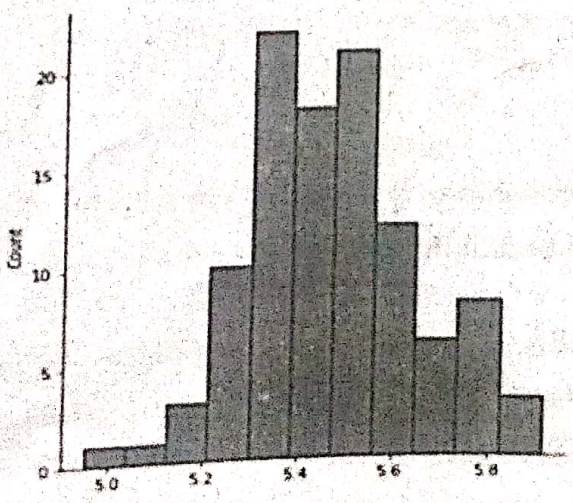


Fig.: Distribution of Sample Means Imagine,

Imagine, If I based my analysis solely on Sample3:[6.1,6.2,6.3,6.0,5.11,5.10,6.0,6.1]. From the looks of the sample, it seems like I chose the tallest of the people. This variation in the estimate due to random selection or noise, where a sample may not perfectly reflect the true population is called sampling error.

3.4 Naive Bayes Classifier Algorithm

Q4. Explain about Naive Bayes Classifier Algorithm.

Ans :

- Naive Bayes algorithm is a supervised learning algorithm, which is based on Bayes theorem and used for solving classification problems.
- It is mainly used in text classification that includes a high-dimensional training dataset.
- Naive Bayes Classifier is one of the simple and most effective Classification algorithms which helps in building the fast machine learning models that can make quick predictions.
- It is a probabilistic classifier, which means it predicts on the basis of the probability of an object.
- Some popular examples of Naive Bayes Algorithm are spam filtration, Sentimental analysis, and classifying articles.

Why is it called Naïve Bayes?

The Naïve Bayes algorithm is comprised of two words Naïve and Bayes, Which can be described as:

- **Naive:** It is called Naïve because it assumes that the occurrence of a certain feature is independent of the occurrence of other features. Such as if the fruit is identified on the bases of color, shape, and taste, then red, spherical, and sweet fruit is recognized as an apple. Hence each feature individually contributes to identify that it is an apple without depending on each other.
- **Bayes:** It is called Bayes because it depends on the principle of Bayes' Theorem.

Bayes' Theorem:

- Bayes' theorem is also known as Bayes' Rule or Bayes' law, which is used to determine the probability of a hypothesis with prior knowledge. It depends on the conditional probability.
- The formula for Bayes' theorem is given as:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

Where,

P(A|B) is Posterior probability: Probability of hypothesis A on the observed event B.

P(B|A) is Likelihood probability: Probability of the evidence given that the probability of a hypothesis is true.

P(A) is Prior Probability: Probability of hypothesis before observing the evidence.

P(B) is Marginal Probability: Probability of Evidence.

Working of Naive Bayes' Classifier:

Working of Naïve Bayes' Classifier can be understood with the help of the below example:

Suppose we have a dataset of weather conditions and corresponding target variable "Play". So using this dataset we need to decide that whether we should play or not on a particular day according to the weather conditions. So to solve this problem, we need to follow the below steps:

1. Convert the given dataset into frequency tables.
2. Generate Likelihood table by finding the probabilities of given features.
3. Now, use Bayes theorem to calculate the posterior probability.

Advantages

1. Naive Bayes is one of the fast and easy ML algorithms to predict a class of datasets.
2. It can be used for Binary as well as Multi-class Classifications.
3. It performs well in Multi-class predictions as compared to the other Algorithms.
4. It is the most popular choice for text classification problems.

Disadvantages

1. Naive Bayes assumes that all features are independent or unrelated, so it cannot learn the relationship between features.

Applications of Naive Bayes Classifier

- It is used for Credit Scoring.
- It is used in medical data classification.
- It can be used in real-time predictions because Naive Bayes Classifier is an eager learner.
- It is used in Text classification such as Spam filtering and Sentiment analysis.

Types

There are three types of Naive Bayes Model, which are given below:

- **Gaussian:** The Gaussian model assumes that features follow a normal distribution. This means if predictors take continuous values instead of discrete, then the model assumes that these values are sampled from the Gaussian distribution.
- **Multinomial:** The Multinomial Naive Bayes classifier is used when the data is multinomial distributed. It is primarily used for document classification problems, it means a particular document belongs to which category such as Sports, Politics, education,etcThe classifier uses the frequency of words for the predictors.
- **Bernoulli:** The Bernoulli classifier works similar to the Multinomial classifier, but the predictor variables are the independent Booleans variables. Such as if a particular word is present or not in a document. This model is also famous for document classification tasks.

Python Implementation of the Naïve Bayes algorithm

Now we will implement a Naive Bayes Algorithm using Python. So for this, we will use the "user_data" dataset, which we have used in our other classification model. Therefore we can easily compare the Naive Bayes model with the other models.

Steps to implement:

- Data Pre-processing step
- Fitting Naive Bayes to the Training set
- Predicting the test result
- Test accuracy of the rest(Creation of Confusion matrix)
- Visualizing the test set result.

Prediction of Naive Bayes Model

Naive Bayes classifier calculates the probability of an event in the following steps:

Step 1:

Calculate the prior probability for given class labels.

Step 2:

Find Likelihood probability with each attribute for each class.

Step 3:

Put these value in Bayes Formula and calculate posterior probability.

3.5 Neural Networks

Q5. What is neural networking in machine learning?

Ans :

A neural network is a method in artificial intelligence that teaches computers to process data in a way that is inspired by the human brain. It is a type of machine learning process, called deep learning, which uses interconnected nodes or neurons in a layered structure that resembles the human brain.

Artificial Neural Networks and Its components

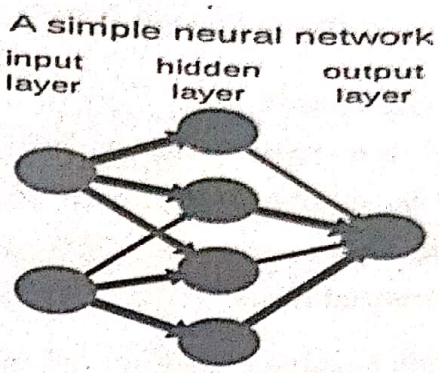
Neural Networks is a computational learning system that uses a network of functions to understand and translate a data input of one form into a desired output, usually in another form. The concept of the artificial neural network was inspired by human biology and the way neurons of the human brain function together to understand inputs from human senses.

In simple words, Neural Networks are a set of algorithms that tries to recognize the patterns, relationships, and information from the data through the process which is inspired by and works like the human brain/biology.

Components / Architecture of Neural Network

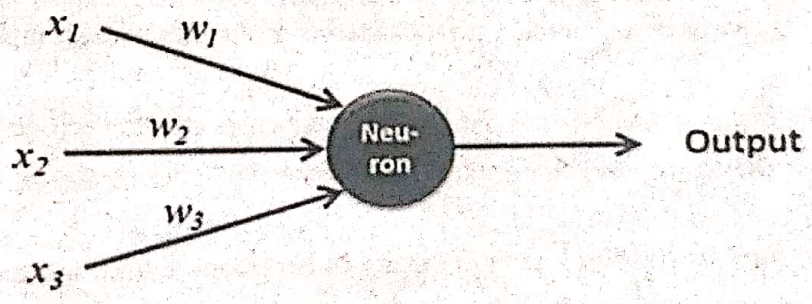
A simple neural network consists of three components :

- Input layer
- Hidden layer
- Output layer

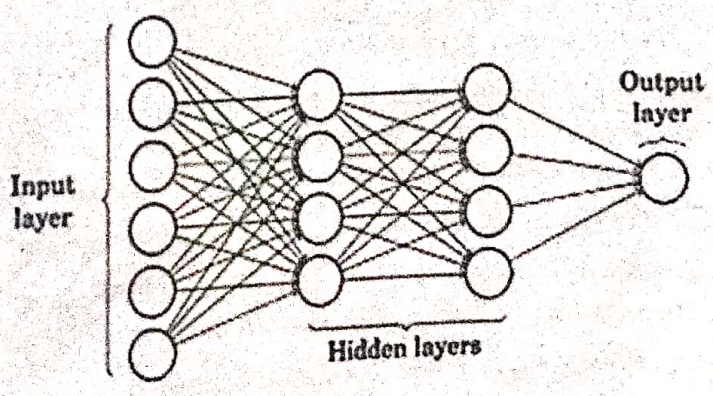


- **Input Layer:** Also known as Input nodes are the inputs/information from the outside world is provided to the model to learn and derive conclusions from. Input nodes pass the information to the next layer i.e Hidden layer.
- **Hidden Layer:** Hidden layer is the set of neurons where all the computations are performed on the input data. There can be any number of hidden layers in a neural network. The simplest network consists of a single hidden layer.
- **Output layer:** The output layer is the output/conclusions of the model derived from all the computations performed. There can be single or multiple nodes in the output layer. If we have a binary classification problem the output node is 1 but in the case of multi-class classification, the output nodes can be more than 1.

Perceptron and Multi-Layer PerceptronPerceptron is a simple form of Neural Network and consists of a single layer where all the mathematical computations are performed.



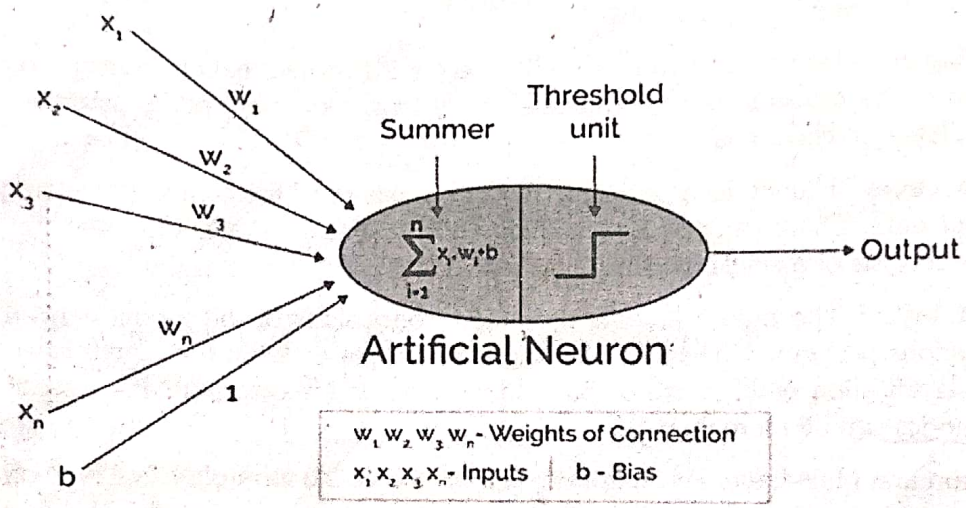
Whereas, Multilayer Perceptron also known as Artificial Neural Networks consists of more than one perception which is grouped together to form a multiple layer neural network.



In the above image, The Artificial Neural Network consists of four layers interconnected with each other:

- An input layer, with 6 input nodes·
- Hidden Layer 1, with 4 hidden nodes/4 perceptrons·
- Hidden layer 2, with 4 hidden nodes·
- Output layer with 1 output node.

Step by Step Working of the Artificial Neural



Network

- In the first step, Input units are passed i.e data is passed with some weights attached to it to the hidden layer. We can have any number of hidden layers. In the above image inputs $x_1, x_2, x_3, \ldots x_n$ is passed.
- Each hidden layer consists of neurons. All the inputs are connected to each neuron.
- After passing on the inputs, all the computation is performed in the hidden layer(Blue oval in the picture)

Computation performed in hidden layers are done in two steps which are as follows:

First of all, all the inputs are multiplied by their weights. Weight is the gradient or coefficient of each variable. It shows the strength of the particular input. After assigning the weights, a bias variable is added. Bias is a constant that helps the model to fit in the best way possible.

$$Z_1 = W_1 * In_1 + W_2 * In_2 + W_3 * In_3 + W_4 * In_4 + W_5 * In_5 + b$$

W_1, W_2, W_3, W_4, W_5 are the weights assigned to the inputs $In_1, In_2, In_3, In_4, In_5$ and b is the bias.

Then in the second step, the activation function is applied to the linear equation Z1. The activation function is a nonlinear transformation that is applied to the input before sending it to the next layer of neurons. The importance of the activation function is to inculcate nonlinearity in the model.

There are several activation functions that will be listed in the next section.

- The whole process described in point 3 is performed in each hidden layer. After passing through every hidden layer, we move to the last layer i.e our output layer which gives us the final output.

The process explained above is known as forwarding Propagation.

- After getting the predictions from the output layer, the error is calculated i.e the difference between the actual and the predicted output.

If the error is large, then the steps are taken to minimize the error and for the same purpose, Back Propagation is performed.

3.5.1 Bio-inspired Multi-Layer Networks

Q6. Explain in detail about Bio-inspired Multi-Layer Networks

Ans :

Convolutional Neural Networks: The Biologically-Inspired Model:

Can you recognize the people in the picture above? If you're a fan of science fiction, you'll instantly recognize that they are Harry, Ron, and Hermione from J.K. Rowling's worldwide phenomenon Harry Potter book series. The picture is a scene from Part 1 of Harry Potter and the Deathly Hallows — in which Harry, Ron, and Hermione are interrogating the thief Mundungus Fletcher, captured moments earlier by the elves Dobby and Kreacher. Well, I know this because I've watched the movie and read the book so many times! But what would it take for a computer to understand this image as you or I do? Let's think explicitly of all the pieces of knowledge that have to fall in place for it to make sense:

- You recognize it is an image of a bunch of people and understand that they are in a room.
- You recognize that there are stacks of newspapers and a long wooden table with chairs, so the location is most likely a kitchen or a common living room.
- You recognize Harry Potter from the few pixels that make up the glasses in his face. It helps that he has black hair as well.
- Similarly, you recognize Ron Weasley because of his red hair and Hermione Granger because of her long hair.
- You recognize the other man who is bald and wears old-fashioned clothing, which suggests that he is much older.
- You recognize the other 2 creatures (Dobby and Kreacher) who are not human, even if you're not familiar with them. You've used their heights, facial structure, body measurement, in addition to your knowledge of normal people's looks to figure it out.
- Harry, Ron, and Hermione are interrogating the bald man. You derive this because you know their body posture gearing towards him, you sense their eyesight appearing doubtful, and you also see

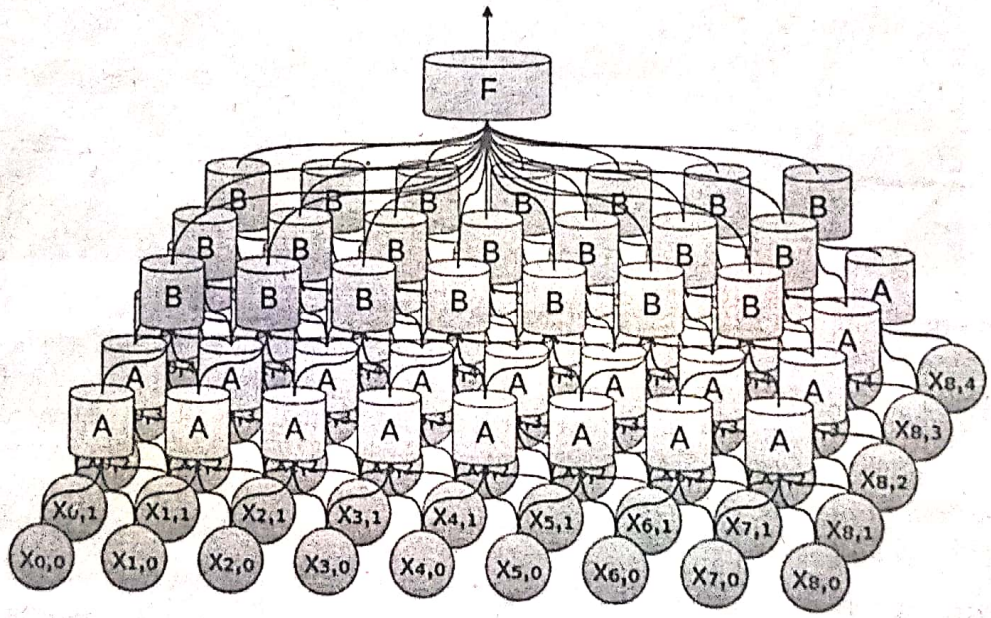
Hermione holding a wand in her hand (wand is the magic weapon in Harry Potter wizarding world).

- You understand that the bald man are feeling scared. You understand that he is hiding something, given that his hands covering his chest. You start to reason about implications of the events that are about to unfold seconds after this scene, and become curious about what secrets will be revealed.
- The 2 creatures are looking towards Harry and Ron. Looks like, they are trying to say something. In other words, you are reasoning about state of mind of those creatures. Whoa, you can be a mind-reader.

The Convolution Process

According to Chris Olah, Research Scientist at Google Brain:"

At its most basic, convolutional neural networks can be thought of as a kind of neural network that uses many identical copies of the same neuron. This allows the network to have lots of neurons and express computationally large models while keeping the number of actual parameters — the values describing how neurons behave — that need to be learned fairly small."



Note the term being used there: identical copies of the same neuron. This is loosely based on the process of how the human brains work. By using the same brain memory spot, humans can spot and recognize patterns without having to re-learn the concept. For example, we recognize the identity of the digits above no matter angle we look at. The feed-forward neural network can't do this. But Convolutional Neural Net can because it understands translation invariance — where it recognizes an object as an object, even when its appearance varies in some way.

In very simple explanation, the convolution process works like this:

- First CNN uses sliding window search to break an image into overlapping image tiles.
- Then CNN feeds each image tile into a small neural network, using the same weights for each tile.
- Then CNN saves the results from each tile into a new output array.
- After that, CNN down-samples the output array to reduce its size.
- Last but not least, after reducing a big image down into a small array, CNN predicts whether the image is a match or not.

3.6 The Back-Propagation Algorithm

Q7. What is Back Propagation and How it works?

Ans:

Back Propagation is the process of updating and finding the optimal values of weights or coefficients which helps the model to minimize the error i.e difference between the actual and predicted values.

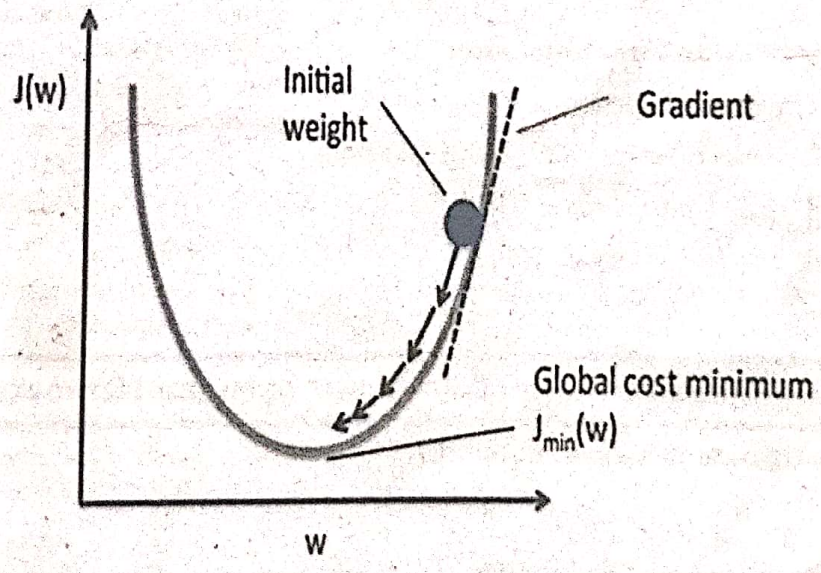
But here are the question is: How the weights are updated and new weights are calculated?

The weights are updated with the help of optimizers.

Optimizers are the methods/ mathematical formulations to change the attributes of neural networks i.e weights to minimizethe error.

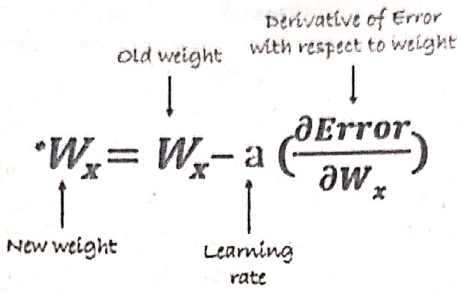
Back Propagation with Gradient Descent

Gradient Descent is one of the optimizers which helps in calculating the new weights. Let's understand step by step how Gradient Descent optimizes the cost function.In the image below, the curve is our cost function curve and our aim is the minimize the error such that J_{min} i.e global minima is achieved.



Steps to achieve the global minima:

1. First, the weights are initialized randomly i.e random value of the weight and intercepts are assigned to the model while forward propagation and the errors are calculated after all the computation. (As discussed above)
2. Then the gradient is calculated i.e derivative of error w.r.t current weights
3. Then new weights are calculated using the below formula, where a is the learning rate which is the parameter also known as step size to control the speed or steps of the backpropagation. It gives additional control on how fast we want to move on the curve to reach global minima.



4. This process of calculating the new weights, then errors from the new weights, and then updation of weights continues till we reach global minima and loss is minimized.

A point to note here is that the learning rate i.e a in our weight updation equation should be chosen wisely. Learning rate is the amount of change or step size taken towards reaching global minima. It should not be very small as it will take time to converge as well as it should not be very large that it doesn't reach global minima at all. Therefore, the learning rate is the hyper parameter that we have to choose based on the model.

Brief about Activation Functions:

Activation functions are attached to each neuron and are mathematical equations that determine whether a neuron should be activated or not based on whether the neuron's input is relevant for the model's prediction or not. The purpose of the activation function is to introduce the nonlinearity in the data.

Various Types of Activation Functions are :

- Sigmoid Activation Function·
- TanH / Hyperbolic Tangent Activation Function·
- Rectified Linear Unit Function (ReLU)·
- Leaky ReLU·
- Softmax

3.7 Initialization and Convergence of Neural Networks

Q8. What is convergence in neural network?

Ans:

In the context of conventional artificial neural networks convergence describes a progression towards a network state where the network has learned to properly respond to a set of training patterns within some margin of error.Weight initialization is an important design choice when developing deep learning neural network model.

Historically, weight initialization involved using small random numbers, although over the last decade, more specific heuristics have been developed that use information, such as the type of activation function that is being used and the number of inputs to the node.

These more tailored heuristics can result in more effective training of neural network models using the stochastic gradient descent optimization algorithm.

In this tutorial, you will discover how to implement weight initialization techniques for deep learning neural networks.

Convergence in neural network?

In the context of conventional artificial neural networks convergence describes a progression towards a network state where the network has learned to properly respond to a set of training patterns within some margin of error.

Convergence

In Netlab, there are two different senses, or connotations, of the word convergence, which can be used to describe two related types of convergence.

- **Adaptive Convergence is just "convergence."** This is the conventional form of the word, as it is used within most existing artificial neural network literature. It describes a set of weights during supervised training, as they begin to find (converge on) the values needed to produce the correct (trained) response.
- **Reactive Convergence is the special sense of the word convergence, which is used in Netlab™.** It describes convergence of propagating signals within networks that employ signal feedback (i.e., "reactive feedback"). It has nothing to do with changes in weight-values or training.

Simply, adaptive convergence describes convergence of weight values, while reactive convergence describes convergence of signal values.

There is a physical, neurobiological connotation of the word as well, which will also be described in this entry (see the section below titled "Convergence In Biology").

Adaptive Convergence(or just: Convergence)

When used without a qualifier in the field of neural networks, convergence is generally understood to mean the conventional usage. In the context of conventional artificial neural networks convergence describes a progression towards a network state where the network has learned to properly respond to a set of training patterns within some margin of error. A convergence error of 10 percent for example, means a network has converged on a training set when it produces output responses that are within ten percent of the desired output values for all the input patterns in the trained repertoire.

There is a form of adaptive convergence, which is peculiar to Netlab™ and it's multitemporal synapses. This usage will be described further, in its own section, below (titled "Mixed Mode Convergence Scenarios").

Reactive Convergence

In Netlab™ the term convergence may also be used to describe a process that is entirely based on the propagation of signals (stimuli) through the network. Unlike the conventional ANN usage, this connotation has nothing to do with training, or changing weight values. When used in this fashion, it will usually be qualified as reactive convergence, though may not always be so qualified.

In conventional feed-forward-only networks, the term "reactive convergence" has no meaning, since, without feedback, there are no "convergence" dynamics that need to be described in such a fashion.

In neural networks that employ reactive feedback, the network may oscillate, or be unstable for multiple iterations before it settles on a given set of responses to a given set of inputs. In this case, the network will be said to converge when (or where) the oscillations (or "ringing") settle, and the network is producing some usable output. The output is not required to be static (steady) to be converged, only to be providing usable, correct, responses to a given present-moment encounter. Such responses may, in fact, be cyclical in nature, over time.

Note also, that the network doesn't necessarily always converge if it has not fully learned how to respond to a given situation. In this case, the oscillations may very well serve as a form of trial and error for whatever learning processes might be used, but the un-converged characteristic of the stimuli, when described in reactive terms, is purely a description of the reactive, propagating, signals, and not of any changes in the connection-weights.

This reactive connotation of the word "convergence" is an exact match for how it is used in electronics design (see conversation below on SPICE®).

Mixed-Mode Convergence Scenarios

In that last example, a network that hadn't reactively converged had signal values that were oscillating randomly between extremes. Notice that this is similar to how the weight-values might be said to be behaving in a network that has not achieved adaptive (conventional) convergence.In this sense, the "un-converged" label being applied to the reactive signal propagation, looks almost identical to how the un-converged nature of the weight values is expressed.

In Netlab™ networks, it is often the case that both of these two forms of convergence are happening simultaneously and in tandem with each other. The erratically changing signal values are helping the short-term weights within multitemporal synapses "hunt" for values that mimic responses already started by the long term weights.

3.8 Beyond Two Layers

Q9. Explain about Beyond Two Layers?

Ans:

The definition of neural networks and the back-propagation algorithm can be generalized beyond two layers to any arbitrary directed

Acyclic graph. In practice, it is most common to use a layered network like that shown unless one has a very strong reason (aka inductive bias) to do something different. However, the view as a directed graph sheds a different sort of insight on the backpropagation algorithm.

Suppose that your network structure is stored in some directed a cyclic graphWe index nodes in this graphas u, v. The activation before applying non-linearity at a node is auand after non-linearity is hu. The graph has a single sink, which isthe output node y with activation ay (no non-linearity is performed.

Algorithm ForwardPropagation(x)

1. for all input nodes u do
2. hu ← corresponding feature of x
3. end for
4. for all nodes v in the network whose parent's are computed do
5. av ← "u"par(v) w(u;v) hu
6. hv ← tanh(av)
7. end for
8. return ay

Algorithm BackPropagation(x, y)

1. run ForwardPropagation(x) to compute activations
2. ey ← y " ay // compute overall network error
3. for all nodes v in the network whose error ev is computed do
4. for all u " par(v) do
5. gu,v ← "evhu // compute gradient of this edge
6. eu ← eu + evwu,v(1 " tanh2 (au)) // compute the "error" of the parent node
7. end for
8. end for
9. return all gradients g_e

3.9 Breadth versus Depth in Neural Networks

Q10. Explain about Breadth versus Depth in Neural Networks?

Ans:

At this point, you've seen how to train two-layer networks and how to train arbitrary networks.

You've also seen a theorem that says that two-layer networks are universal function approximators. This begs the question: if two-layer networks are so great, why do we care about deeper networks? To understand the answer, we can borrow some ideas from CS theory, namely the idea of circuit complexity. The goal is to show that there are functions for which it might be a "good idea" to use a deep network. In other words, there are functions that will require a huge number of hidden

units if you force the network to be shallow, but can be done in a small number of units if you allow it to be deep.

The example that we'll use is the parity function which, ironically enough, is just a generalization of the XOR problem. The function is defined over binary inputs as:

parity(x) =Σd xd mod 2/= (1 if the number of 1s in x is odd 0 if the number of 1s in x is even nnet:paritydeep: deep function for computing parity It is easy to define a circuit of depth O(log2 D) with O(D)-many gates for computing the parity function. Each gate is an XOR, arranged in a complete binary tree. (If you want to disallow XOR as a gate, you can fix this by allowing the depth to be doubled and replacing each XOR with an AND, OR and NOT combination, like you did at the beginning of this chapter.) This shows that if you are allowed to be deep, you can construct a circuit with that computes parity using a number of hidden units that is linear in the dimensionality. So can you do the same with shallow circuits? The answer is no. It's a famous result of circuit complexity that parity requires exponentially many gates to compute in constant depth. The formal theorem is below:

Theorem: (Parity Function Complexity). Any circuit of depth $K < \log_2 D$ that computes the parity function of D input bits must contain O_e^D gates

This is a very famous result because it shows that constant-depth circuits are less powerful that deep circuits. Although a neural network isn't exactly the same as a circuit, the is generally believed that the same result holds for neural networks. At the very least, this gives a strong indication that depth might be an important consideration in neural networks.

3.10 Basis Functions

Q11. Explain in detail about basis functions.

Ans :

$$h(x, w) = w^T x. \qquad \ldots(1)$$

We say that a model is linear if it's linear in the parameters not in the input variables. However, (1) is linear in both the parameters and the input variables, which limits it from adapting to nonlinear relationships. We can augment the model by replacing the input variables with nonlinear basis functions of the input variables

$$h(x, w) = w_0\phi_0(x) + \ldots + w_{M-1}\phi_{M-1}(x)$$

$$= \sum_{m=0}^{M-1} w_m \phi_m(x)$$

$$= w^T\phi(x),$$

where

$$\phi(x) = \begin{pmatrix} \phi_0(x) \\ \vdots \\ \phi_{M-1}(x) \end{pmatrix}$$

By using nonlinear basis functions it is possible for h to adapt to nonlinear relationships of x, which we will see shortly — we call these models linear basis function models. We already looked at one example of basis functions, where we augmented the simple linear regression model with basis functions of powers of x, i.e.,

$$\phi_i(x) = x^i$$

Another common basis function is the Gaussian

$$\phi_i(x) = \exp\left(-\gamma_i \| \mu_i - x \|_2^2\right)$$

Following the same derivation, we find the maximum likelihood solutions for w and α to be

$$w_{MLE} = \left(\Phi^T\Phi\right)^{-} \Phi^T t \text{ and}$$

$$\alpha_{MLE} = \frac{1}{N}\sum_{i=2}^{N}\left(t_n - w_{MLE}^T\phi\right)(x_i))^2$$

where,

$$\Phi = \begin{pmatrix} \phi_0(x_1) & \phi_1(x_1) & \cdots & \phi_{M-1}(x_1) \\ \phi_0(x_2) & \phi_1(x_2) & \cdots & \phi_{M-1}(x_2) \\ \vdots & \vdots & \ddots & \vdots \\ \phi_0(x_N) & \phi_1(x_N) & \cdots & \phi_{M-1}(x_N) \end{pmatrix}$$

The image below shows a linear basis function model with M-1 Gaussian basis functions. We can see that increasing the number of basis functions makes a better model, until we start overfitting.

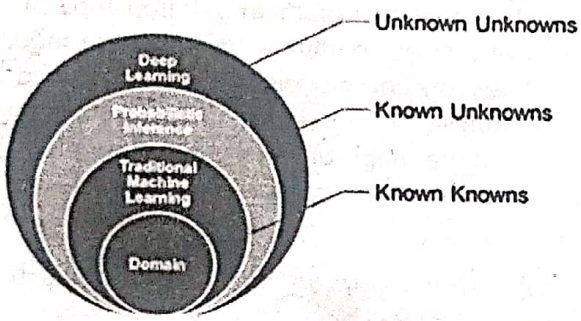
Short Question and Answers

1. What are Probabilistic Models in Machine Learning?

Ans :

Introduction

Probabilistic Models in Machine Learning is the use of the codes of statistics to data examination. It was one of the initial methods of machine learning. It's quite extensively used to this day. Individual of the best-known algorithms in this group is the Naive Bayes algorithm.



Probabilistic modelling delivers a framework for accepting what learning is. The probabilistic framework defines how to signify and deploy reservation about models. Predictions have a dominant role in scientific data analysis. Their role is also so important in machine learning, automation, cognitive computing and artificial intelligence.

DescriptionProbabilistic models are presented as a prevailing idiom to define the world. Those were described by using random variables for example building blocks believed together by probabilistic relationships.There are probabilistic models along with non-probabilistic models in machine learning. The information about basic concepts of probability for example random variables and probability distributions would be helpful in order to have a well understanding of probabilistic models.Portrayal inference from noisy or ambiguous data is an imperative part of intelligent systems. In probability theory particularly.

Bayes' theorem helps as a principled framework of combining prior knowledge and empirical evidence.Importance of probabilistic ML models:One of the key benefits of probabilistic models is that they give an idea about the uncertainty linked with predictions. We may get an idea of how confident a machine learning model is on its prediction. For example, if the probabilistic classifier allocates a probability of 0.9 for the 'Dog' class in its place of 0.6, it means the classifier is extra confident that the animal in the image is a dog. These concepts connected to uncertainty and confidence are very valuable when it originates to critical machine learning uses for example disease diagnosis and autonomous driving. Moreover, probabilistic consequences would be worthwhile for many methods linked to Machine Learning for instance

2. Explain about Naive Bayes algorithm & its types.

Ans:

Naive Bayes algorithm is a supervised learning algorithm. It is created on the Bayes theorem and used for resolving sorting problems. It is chiefly used in text classification that comprises a high-dimensional training dataset. The naïve Bayes algorithm is one of the simple and best operational Classification algorithms that support construction the of fast machine learning models which may create rapid predictions.

The naive Bayes algorithm is a probabilistic classifier. It means that it predicts on the basis of the probability of an object.

The types are : (i) Gaussian
(ii) Multinomial
(iii) Bernouli

3. Write the pros and cons on Navie Bayes Model.

Ans:

Uses of Naive Bayes Model

- The Naive Bayes Classifier used;
- For Credit Scoring.
- In medical data classification.

- It may be used in real-time predictions as Naïve Bayes Classifier is a keen learner.
- In-Text classification for example Spam filtering and Sentiment analysis.
- Pros and Cons of Bayes Classifier

Pros

Naive Bayes is one of the easy and fast machine learning algorithms to foresee a class of datasets.It may be used for Binary also as Multi-class Classifications.It does well in Multi-class predictions for example likened to the other Algorithms.It is the greatest widespread selection for text classification problems.

Cons

Naive Bayes accepts that all sorts are autonomous or disparate. Therefore it cannot learn the association between features. Objective Functions We may gaze at its Objective Function permissible to recognize whether a specific model is probabilistic or not. We wish to enhance a model to excel at an exact task in machine learning. The target of having an objective function is to deliver a value based on the model's outputs. Therefore, optimization may be done by moreover exploiting or curtailing the actual value. Usually, the objective is to reduce prediction error in Machine Learning. Therefore, we describe what is called a loss function for example the objective function and attempts to reduce the loss function in the training phase of a machine learning model.

4. Explain about Naive Bayes Classifier Algorithm.

Ans :

- Naive Bayes algorithm is a supervised learning algorithm, which is based on Bayes theorem and used for solving classification problems.
- It is mainly used in text classification that includes a high-dimensional training dataset.
- Naive Bayes Classifier is one of the simple and most effective Classification algorithms which helps in building the fast machine learning models that can make quick predictions.
- It is a probabilistic classifier, which means it predicts on the basis of the probability of an object.
- Some popular examples of Naive Bayes Algorithm are spam filtration, Sentimental analysis, and classifying articles.

Why is it called Naïve Bayes?

The Naïve Bayes algorithm is comprised of two words Naïve and Bayes, Which can be described as:

- **Naive**

  It is called Naïve because it assumes that the occurrence of a certain feature is independent of the occurrence of other features. Such as if the fruit is identified on the bases of color, shape, and taste, then red, spherical, and sweet fruit is recognized as an apple. Hence each feature individually contributes to identify that it is an apple without depending on each other.

- **Bayes**

  It is called Bayes because it depends on the principle of Bayes' Theorem.

Bayes' Theorem:

- Bayes' theorem is also known as Bayes' Rule or Bayes' law, which is used to determine the probability of a hypothesis with prior knowledge. It depends on the conditional probability.
- The formula for Bayes' theorem is given as:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

Where,

P(A|B) is Posterior probability

Probability of hypothesis A on the observed event B.

P(B|A) is Likelihood probability

Probability of the evidence given that the probability of a hypothesis is true.

P(A) is Prior Probability

Probability of hypothesis before observing the evidence.

P(B) is Marginal Probability

Probability of Evidence.

5. What is neural networking in machine learning?

Ans :

A neural network is a method in artificial intelligence that teaches computers to process data in a way that is inspired by the human brain. It is a type of machine learning process, called deep learning, which uses interconnected nodes or neurons in a layered structure that resembles the human brain.

Artificial Neural Networks and Its components

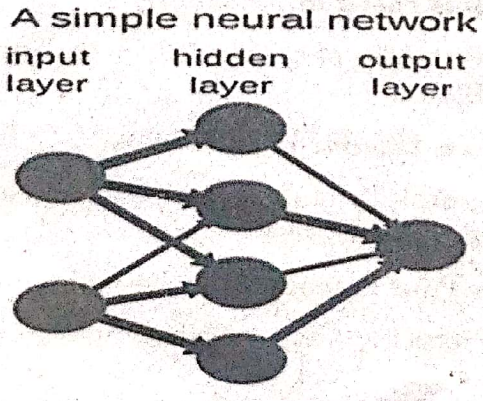
Neural Networks is a computational learning system that uses a network of functions to understand and translate a data input of one form into a desired output, usually in another form. The concept of the artificial neural network was inspired by human biology and the way neurons of the human brain function together to understand inputs from human senses.

In simple words, Neural Networks are a set of algorithms that tries to recognize the patterns, relationships, and information from the data through the process which is inspired by and works like the human brain/biology.

Components / Architecture of Neural Network

A simple neural network consists of three components :

- Input layer
- Hidden layer
- Output layer



- **Input Layer**

  Also known as Input nodes are the inputs/ information from the outside world is provided to the model to learn and derive conclusions from. Input nodes pass the information to the next layer i.e Hidden layer.

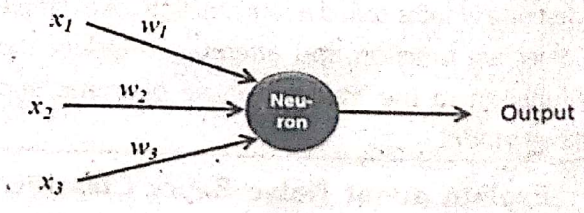
- **Hidden Layer**

  Hidden layer is the set of neurons where all the computations are performed on the input data. There can be any number of hidden layers in a neural network. The simplest network consists of a single hidden layer.

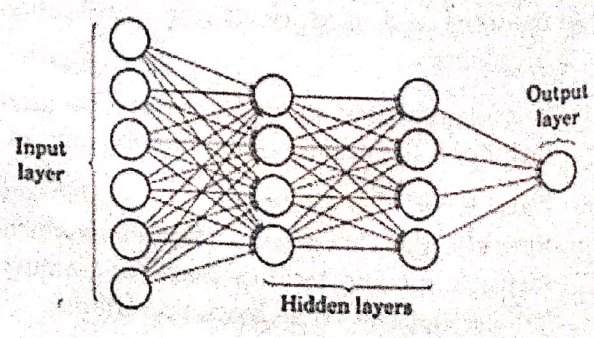
- **Output layer**

  The output layer is the output/conclusions of the model derived from all the computations performed. There can be single or multiple nodes in the output layer. If we have a binary classification problem the output node is 1 but in the case of multi-class classification, the output nodes can be more than 1.

Perceptron and Multi-Layer Perceptron Perceptron is a simple form of Neural Network and consists of a single layer where all the mathematical computations are performed.



Whereas, Multilayer Perceptron also known as Artificial Neural Networks consists of more than one perception which is grouped together to form a multiple layer neural network.



6. What is Back Propagation and How it works?

Ans:

Back Propagation is the process of updating and finding the optimal values of weights or coefficients which helps the model to minimize the error i.e difference between the actual and predicted values.

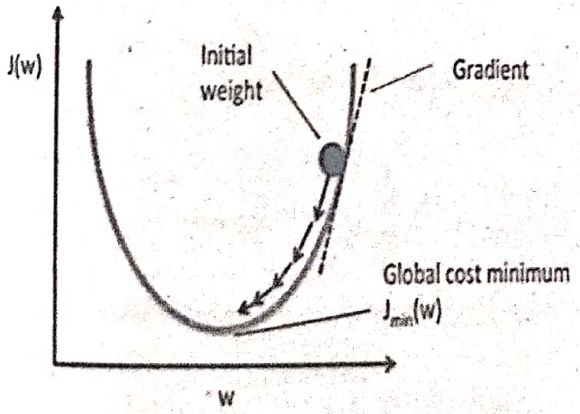
But here are the question is: How the weights are updated and new weights are calculated?

The weights are updated with the help of optimizers.

Optimizers are the methods/ mathematical formulations to change the attributes of neural networks i.e weights to minimizethe error.

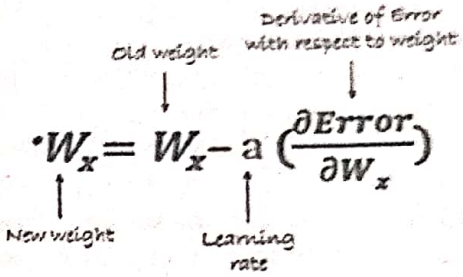
Back Propagation with Gradient Descent

Gradient Descent is one of the optimizers which helps in calculating the new weights. Let's understand step by step how Gradient Descent optimizes the cost function.In the image below, the curve is our cost function curve and our aim is the minimize the error such that J_{min} i.e global minima is achieved.



Steps to achieve the global minima:

1. First, the weights are initialized randomly i.e random value of the weight, and intercepts are assigned to the model while forward propagation and the errors are calculated after all the computation. (As discussed above)
2. Then the gradient is calculated i.e derivative of error w.r.t current weights
3. Then new weights are calculated using the below formula, where a is the learning rate which is the parameter also known as step size to control the speed or steps of the backpropagation. It gives additional control on how fast we want to move on the curve to reach global minima.



4. This process of calculating the new weights, then errors from the new weights, and then updation of weights continues till we reach global minima and loss is minimized.

A point to note here is that the learning rate i.e a in our weight updation equation should be chosen wisely. Learning rate is the amount of change or step size taken towards reaching global minima. It should not be very small as it will take time to converge as well as it should not be very large that it doesn't reach global minima at all. Therefore, the learning rate is the hyper parameter that we have to choose based on the model.

7. Explain about Breadth versus Depth in Neural Networks?

Ans:

At this point, you've seen how to train two-layer networks and how to train arbitrary networks.

You've also seen a theorem that says that two-layer networks are universal function approximators. This begs the question: if two-layer networks are so great, why do we care about deeper networks? To understand the answer, we can borrow some ideas from CS theory, namely the idea of circuit complexity. The goal is to show that there are functions for which it might be a "good idea" to use a deep network. In other words, there are functions that will require a huge number of hidden units if you force the network to be shallow, but can be done in a small number of units if you allow it to be deep.

The example that we'll use is the parity function which, ironically enough, is just a generalization of the XOR problem. The function is defined over binary inputs as:

parity(x) =Σd xd mod 2 = (1 if the number of 1s in x is odd 0 if the number of 1s in x is even nnet:paritydeep: deep function for computing parity It is easy to define a circuit of depth O(log2 D) with O(D)-many gates for computing the parity function. Each gate is an XOR, arranged in a complete binary tree, (If you want to disallow XOR as a gate, you can fix this by allowing the depth to be doubled and replacing each XOR with an AND, OR and NOT combination, like you did at the beginning of this chapter.) This shows that if you are allowed to be deep, you can construct a circuit with that computes parity using a number of hidden units that is linear in the dimensionality. So can you do the same with shallow circuits? The answer is no. It's a famous result of circuit complexity that parity requires exponentially many gates to compute in constant depth. The formal theorem is below:

Theorem: (Parity Function Complexity). Any circuit of depth $K < \log_2 D$ that computes the parity function of D input bits must contain O_e^D gates

This is a very famous result because it shows that constant-depth circuits are less powerful that deep circuits. Although a neural network isn't exactly the same as a circuit, the is generally believed that the same result holds for neural networks. At the very least, this gives a strong indication that depth might be an important consideration in neural networks.

8. Explain basis function.

Ans:

$$h(x, w) = w^T x. \qquad ....(1)$$

We say that a model is linear if it's linear in the parameters not in the input variables. However, (1) is linear in both the parameters and the input variables, which limits it from adapting to nonlinear relationships. We can augment the model by replacing the input variables with nonlinear basis functions of the input variables

$$h(x, w) = w_0\phi_0(x)+.....+w_{M-1}\phi_{M-1}(x)$$

$$= \sum_{m=0}^{M-1} w_m\phi_m(x)$$

$$= w^T\phi(x),$$

where

$$\phi(x) = \begin{pmatrix} \phi_0(x) \\ \vdots \\ \phi_{M-1}(x) \end{pmatrix}$$

By using nonlinear basis functions it is possible for h to adapt to nonlinear relationships of x, which we will see shortly — we call these models linear basis function models. We already looked at one example of basis functions, where we augmented the simple linear regression model with basis functions of powers of x, i.e.,

$$\phi_i(x) = x^i$$

Another common basis function is the Gaussian

$$\phi_i(x) = \exp\left(-\gamma_i \| \mu_i - x \|_2^2\right)$$

Following the same derivation, we find the maximum likelihood solutions for w and α to be

$$w_{MLE} = \left(\Phi^T\Phi\right)^- \Phi^T t \text{ and}$$

$$\alpha_{MLE} = \frac{1}{N}\sum_{i=2}^{N}\left(t_n - w_{MLE}^T\phi\right)(x_i))^2$$

where,

$$\Phi = \begin{pmatrix} \phi_0(x_1) & \phi_1(x_1) & \cdots & \phi_{M-1}(x_1) \\ \phi_0(x_2) & \phi_1(x_2) & \cdots & \phi_{M-1}(x_2) \\ \vdots & \vdots & \ddots & \vdots \\ \phi_0(x_N) & \phi_1(x_N) & \cdots & \phi_{M-1}(x_N) \end{pmatrix}$$

The image below shows a linear basis function model with M-1 Gaussian basis functions. We can see that increasing the number of basis functions makes a better model, until we start overfitting.

One Mark Answers

1. Define Weight initialization

Ans :

Weight initialization is an important consideration in the design of a neural network model. The nodes in neural networks are composed of parameters referred to as weights used to calculate a weighted sum of the inputs.

2. What is convergence in neural network?

Ans :

In the context of conventional artificial neural networks convergence describes a progression towards a network state where the network has learned to properly respond to a set of training patterns within some margin of error.

3. What is Back Propagation and How it works?

Ans :

Back Propagation is the process of updating and finding the optimal values of weights or coefficients which helps the model to minimize the error i.e. difference between the actual and predicted values.

4. Define Basis functions.

Ans :

This is a generalization of linear regression that essentially replaces each input with a function of the input. (A linear basis function model that uses the identity function is just linear regression.)