ML NOTES

1. Introduction to Machine Learning

What is Machine Learning?

Machine Learning is the field of study that gives computers the capability to learn without being explicitly programmed. ML is one of the most exciting technologies that one would have ever come across. As it is evident from the name, it gives the computer that makes it more similar to humans: The ability to learn. Machine learning is actively being used today, perhaps in many more places than one would expect.

Features of Machine learning

- Machine learning is data driven technology. Large amount of data generated by organizations on daily bases. So, by notable relationships in data, organizations makes better decisions.
- Machine can learn itself from past data and automatically improve.
- From the given dataset it detects various patterns on data.
- For the big organizations branding is important and it will become more easy to target relatable customer base.
- It is similar to data mining because it is also deals with the huge amount of data.

1. Supervised Machine Learning

Supervised learning is defined as when a model gets trained on a **"Labelled Dataset"**. Labelled datasets have both input and output parameters. In **Supervised Learning** algorithms learn to map points between inputs and correct outputs. It has both training and validation datasets labelled.

Example: Consider a scenario where you have to build an image classifier to differentiate between cats and dogs. If you feed the datasets of dogs and cats labelled images to the algorithm, the machine will learn to classify between a dog or a cat from these labeled images. When we input new dog or cat images that it has never seen before, it will use the learned algorithms and predict whether it is a dog or a cat. This is how **supervised learning** works, and this is particularly an image classification.

There are two main categories of supervised learning that are mentioned below:

- Classification
- Regression

Classification

Classification deals with predicting **categorical** target variables, which represent discrete classes or labels. For instance, classifying emails as spam or not spam, or predicting whether a patient has a high risk of heart disease. Classification algorithms learn to map the input features to one of the predefined classes.

Here are some classification algorithms:

- **Logistic Regression**

- **Support Vector Machine**
- **Random Forest**
- **Decision Tree**
- **K-Nearest Neighbors (KNN)**
- **Naive Bayes**

Regression

Regression, on the other hand, deals with predicting **continuous** target variables, which represent numerical values. For example, predicting the price of a house based on its size, location, and amenities, or forecasting the sales of a product. Regression algorithms learn to map the input features to a continuous numerical value.

Here are some regression algorithms:

- **Linear Regression**
- **Polynomial Regression**
- **Ridge Regression**
- **Lasso Regression**
- **Decision tree**
- **Random Forest**

2. Unsupervised Machine Learning

Unsupervised Learning Unsupervised learning is a type of machine learning technique in which an algorithm discovers patterns and relationships using unlabeled data. Unlike supervised learning, unsupervised learning doesn’t involve providing the algorithm with labeled target outputs. The primary goal of Unsupervised learning is often to discover hidden patterns, similarities, or clusters within the data, which can then be used for various purposes, such as data exploration, visualization, dimensionality reduction, and more.

Example: Consider that you have a dataset that contains information about the purchases you made from the shop. Through clustering, the algorithm can group the same purchasing behavior among you and other customers, which reveals potential customers without predefined labels. This type of information can help businesses get target customers as well as identify outliers.

There are two main categories of unsupervised learning that are mentioned below:

- Clustering
- Association

Clustering

Clustering is the process of grouping data points into clusters based on their similarity. This technique is useful for identifying patterns and relationships in data without the need for labeled examples.

Here are some clustering algorithms:

- **K-Means Clustering algorithm**
- **Mean-shift algorithm**

Association

Association rule learning is a technique for discovering relationships between items in a dataset. It identifies rules that indicate the presence of one item implies the presence of another item with a specific probability.

Here are some association rule learning algorithms:

- **Apriori Algorithm**
- **Eclat**
- **FP-growth Algorithm**

3. Semi-Supervised Learning

Semi-Supervised learning is a machine learning algorithm that works between the supervised and unsupervised learning so it uses both **labelled and unlabelled** data. It's particularly useful when obtaining labeled data is costly, time-consuming, or resource-intensive. This approach is useful when the dataset is expensive and time-consuming. Semi-supervised learning is chosen when labeled data requires skills and relevant resources in order to train or learn from it.

We use these techniques when we are dealing with data that is a little bit labeled and the rest large portion of it is unlabeled. We can use the unsupervised techniques to predict labels and then feed these labels to supervised techniques. This technique is mostly applicable in the case of image data sets where usually all images are not labeled.

Example: Consider that we are building a language translation model, having labeled translations for every sentence pair can be resources intensive. It allows the models to learn from labeled and unlabeled sentence pairs, making them more accurate. This technique has led to significant improvements in the quality of machine translation services.

Types of Semi-Supervised Learning Methods

There are a number of different semi-supervised learning methods each with its own characteristics. Some of the most common ones include:

- **Graph-based semi-supervised learning:** This approach uses a graph to represent the relationships between the data points. The graph is then used to propagate labels from the labeled data points to the unlabeled data points.
- **Label propagation:** This approach iteratively propagates labels from the labeled data points to the unlabeled data points, based on the similarities between the data points.
- **Co-training:** This approach trains two different machine learning models on different subsets of the unlabeled data. The two models are then used to label each other's predictions.
- **Self-training:** This approach trains a machine learning model on the labeled data and then uses the model to predict labels for the unlabeled data. The model is then retrained on the labeled data and the predicted labels for the unlabeled data.

- **Generative adversarial networks (GANs):** GANs are a type of deep learning algorithm that can be used to generate synthetic data. GANs can be used to generate unlabeled data for semi-supervised learning by training two neural networks, a generator and a discriminator.

4. Reinforcement Machine Learning

Reinforcement machine learning algorithm is a learning method that interacts with the environment by producing actions and discovering errors. **Trial, error, and delay** are the most relevant characteristics of reinforcement learning. In this technique, the model keeps on increasing its performance using Reward Feedback to learn the behavior or pattern. These algorithms are specific to a particular problem e.g. Google Self Driving car, AlphaGo where a bot competes with humans and even itself to get better and better performers in Go Game. Each time we feed in data, they learn and add the data to their knowledge which is training data. So, the more it learns the better it gets trained and hence experienced.

Here are some of most common reinforcement learning algorithms:

- **Q-learning:** Q-learning is a model-free RL algorithm that learns a Q-function, which maps states to actions. The Q-function estimates the expected reward of taking a particular action in a given state.
- **SARSA (State-Action-Reward-State-Action):** SARSA is another model-free RL algorithm that learns a Q-function. However, unlike Q-learning, SARSA updates the Q-function for the action that was actually taken, rather than the optimal action.
- **Deep Q-learning:** Deep Q-learning is a combination of Q-learning and deep learning. Deep Q-learning uses a neural network to represent the Q-function, which allows it to learn complex relationships between states and actions.

Example: Consider that you are training an AI agent to play a game like chess. The agent explores different moves and receives positive or negative feedback based on the outcome. Reinforcement Learning also finds applications in which they learn to perform tasks by interacting with their surroundings.

Types of Reinforcement Machine Learning

There are two main types of reinforcement learning:

Positive reinforcement

- Rewards the agent for taking a desired action.
- Encourages the agent to repeat the behavior.
- Examples: Giving a treat to a dog for sitting, providing a point in a game for a correct answer.

Negative reinforcement

- Removes an undesirable stimulus to encourage a desired behavior.
- Discourages the agent from repeating the behavior.
- Examples: Turning off a loud buzzer when a lever is pressed, avoiding a penalty by completing a task.

PROCESS OF MACHINE LEARNING

1. Gathering Data:

Data Gathering is the first step of the machine learning life cycle. The goal of this step is to identify and obtain all data-related problems.

In this step, we need to identify the different data sources, as data can be collected from various sources such as **files**, **database**, **internet**, or **mobile devices**. It is one of the most important steps of the life cycle. The quantity and quality of the collected data will determine the efficiency of the output. The more will be the data, the more accurate will be the prediction.

This step includes the below tasks:

- **Identify various data sources**
- **Collect data**
- **Integrate the data obtained from different sources**

2. Data preparation

After collecting the data, we need to prepare it for further steps. Data preparation is a step where we put our data into a suitable place and prepare it to use in our machine learning training.

In this step, first, we put all data together, and then randomize the ordering of data.

This step can be further divided into two processes:

- **Data exploration:**
  It is used to understand the nature of data that we have to work with. We need to understand the characteristics, format, and quality of data.
  A better understanding of data leads to an effective outcome. In this, we find Correlations, general trends, and outliers.
- **Data pre-processing:**
  Now the next step is preprocessing of data for its analysis.

3. Data Wrangling

Data wrangling is the process of cleaning and converting raw data into a useable format. It is the process of cleaning the data, selecting the variable to use, and transforming the data in a proper format to make it more suitable for analysis in the next step. It is one of the most important steps of the complete process. Cleaning of data is required to address the quality issues.

It is not necessary that data we have collected is always of our use as some of the data may not be useful. In real-world applications, collected data may have various issues, including:

- **Missing Values**
- **Duplicate data**
- **Invalid data**
- **Noise**

So, we use various filtering techniques to clean the data.

It is mandatory to detect and remove the above issues because it can negatively affect the quality of the outcome.

4. Data Analysis

Now the cleaned and prepared data is passed on to the analysis step. This step involves:

- **Selection of analytical techniques**
- **Building models**
- **Review the result**

The aim of this step is to build a machine learning model to analyze the data using various analytical techniques and review the outcome. It starts with the determination of the type of the problems, where we select the machine learning techniques such as **Classification**, **Regression**, **Cluster analysis**, **Association**, etc. then build the model using prepared data, and evaluate the model.

Hence, in this step, we take the data and use machine learning algorithms to build the model.

5. Train Model

Now the next step is to train the model, in this step we train our model to improve its performance for better outcome of the problem.

We use datasets to train the model using various machine learning algorithms. Training a model is required so that it can understand the various patterns, rules, and, features. 7. Deployment

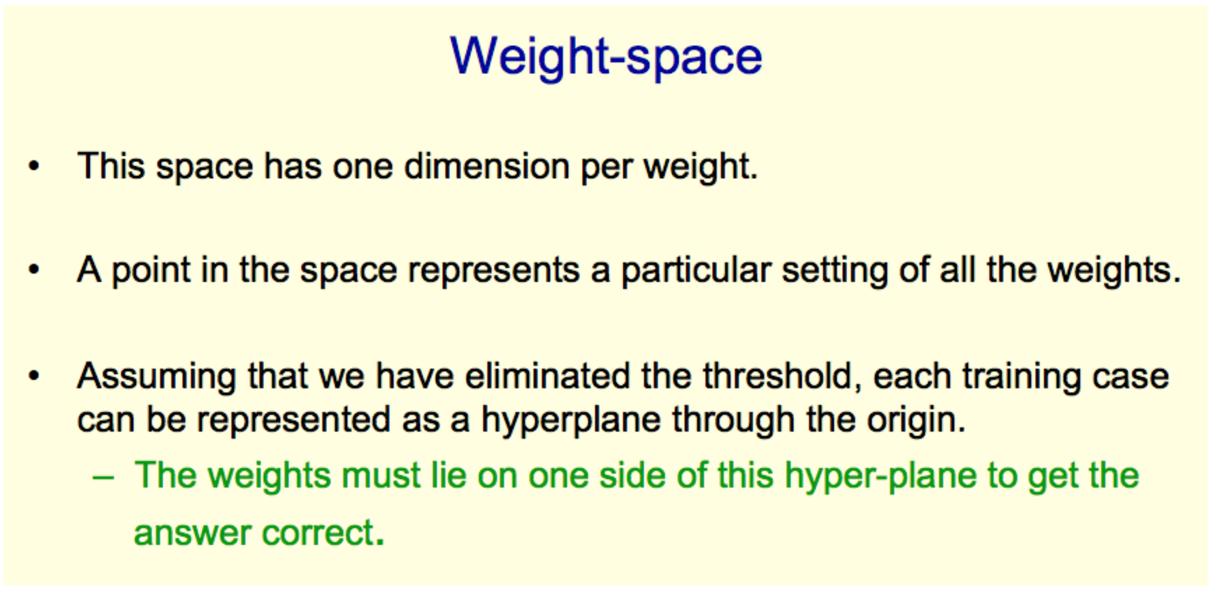
The last step of machine learning life cycle is deployment, where we deploy the model in the real-world system.

If the above-prepared model is producing an accurate result as per our requirement with acceptable speed, then we deploy the model in the real system. But before deploying the project, we will check whether it is improving its performance using available data or not. The deployment phase is similar to making the final report for a project.

6. Test Model

Once our machine learning model has been trained on a given dataset, then we test the model. In this step, we check for the accuracy of our model by providing a test dataset to it.

Testing the model determines the percentage accuracy of the model as per the requirement of project or problem.



Weight-space

- This space has one dimension per weight.
- A point in the space represents a particular setting of all the weights.
- Assuming that we have eliminated the threshold, each training case can be represented as a hyperplane through the origin.
  - The weights must lie on one side of this hyper-plane to get the answer correct.

What is the Curse of Dimensionality?

- The Curse of Dimensionality refers to the phenomenon where the efficiency and effectiveness of algorithms deteriorate as the dimensionality of the data increases exponentially.
- In high-dimensional spaces, data points become sparse, making it challenging to discern meaningful patterns or relationships due to the vast amount of data required to adequately sample the space.
- The Curse of Dimensionality significantly impacts machine learning algorithms in various ways. It leads to increased computational complexity, longer training times, and higher resource requirements. Moreover, it escalates the risk of overfitting and spurious correlations, hindering the algorithms' ability to generalize well to unseen data.

How to Overcome the Curse of Dimensionality?

To overcome the curse of dimensionality, you can consider the following strategies:

Dimensionality Reduction Techniques:

- Feature Selection: Identify and select the most relevant features from the original dataset while discarding irrelevant or redundant ones. This reduces the dimensionality of the data, simplifying the model and improving its efficiency.
- Feature Extraction: Transform the original high-dimensional data into a lower-dimensional space by creating new features that capture the essential information. Techniques such as Principal Component Analysis (PCA) and t-distributed Stochastic Neighbor Embedding (t-SNE) are commonly used for feature extraction.

Data Preprocessing:

- Normalization: Scale the features to a similar range to prevent certain features from dominating others, especially in distance-based algorithms.
- Handling Missing Values: Address missing data appropriately through imputation or deletion to ensure robustness in the model training process.

TESTING ML ALGORITHMS

Machine learning testing is the process of evaluating and validating machine learning models to ensure they are accurate, correct, and robust. It differs from traditional software testing because it includes additional layers to account for the complexity of machine learning models.

Here are some types of machine learning testing:

- Unit and integration testing

These are standard software testing methods that are also used for machine learning models.

- Data testing

Since machine learning models are made up of code and data, additional tests are needed at various levels to ensure reliability.

- Acceptance testing

This type of testing often involves user studies to evaluate the system's usability and effectiveness.

- Regression testing

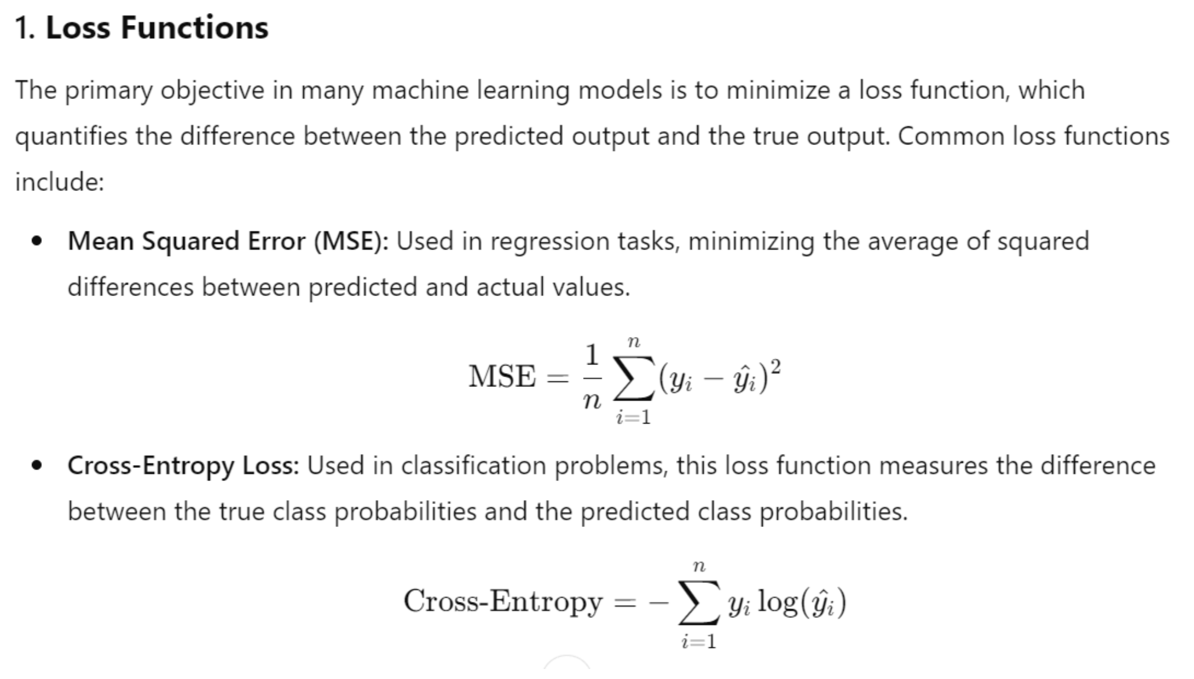
This type of analysis can help improve predictions from data.

- AI and ML for test result analysis

AI and ML algorithms can analyze test results to identify patterns and anomalies that may indicate defects

MINIMISING RISK ALGORITHM

In machine learning, risk minimization typically refers to reducing the potential error or loss in model predictions. This can involve optimizing a loss function, regularization techniques to avoid overfitting, and making the model robust to uncertainties or adversarial attacks.



1. **Loss Functions**

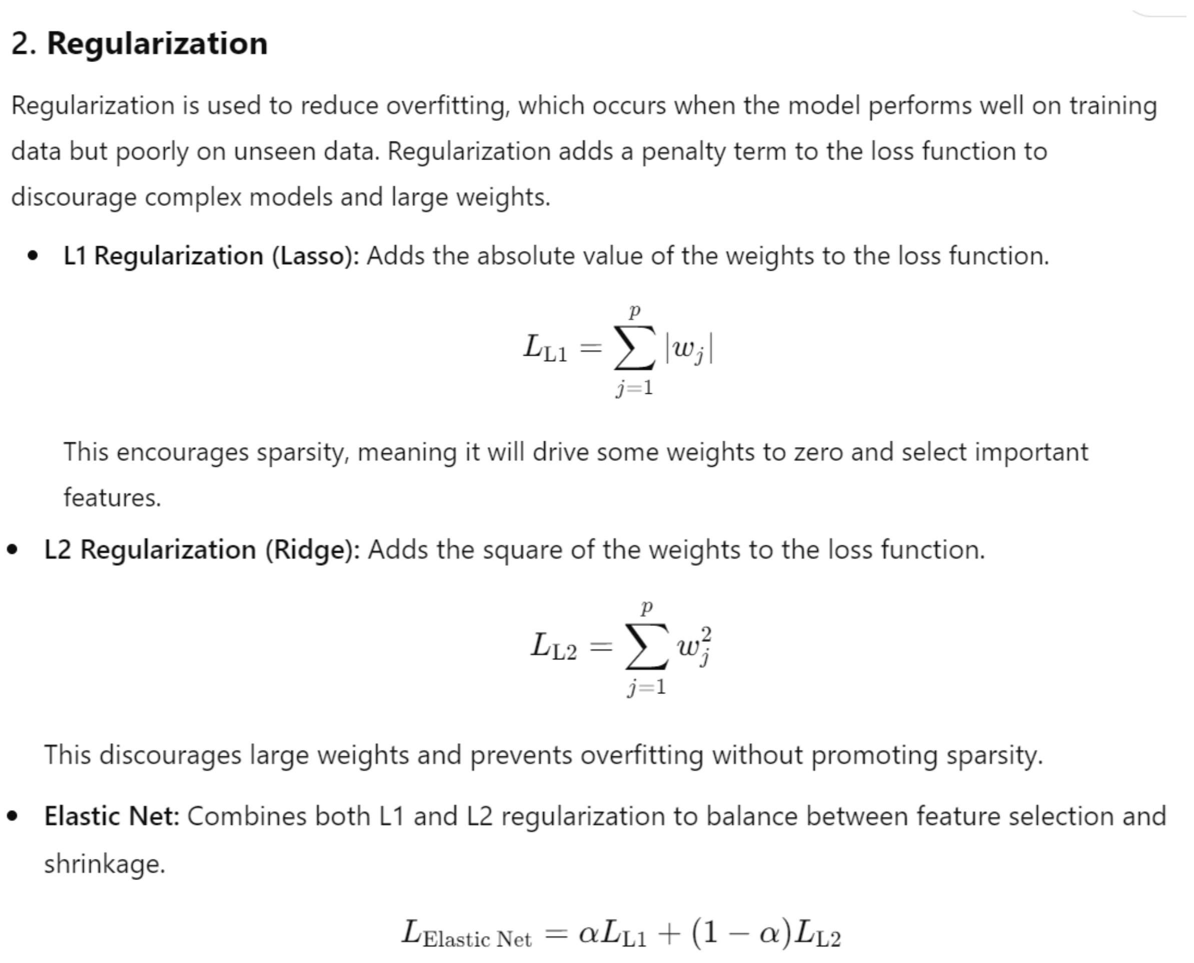
The primary objective in many machine learning models is to minimize a loss function, which quantifies the difference between the predicted output and the true output. Common loss functions include:

- **Mean Squared Error (MSE):** Used in regression tasks, minimizing the average of squared differences between predicted and actual values.

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

- **Cross-Entropy Loss:** Used in classification problems, this loss function measures the difference between the true class probabilities and the predicted class probabilities.

$$\text{Cross-Entropy} = -\sum_{i=1}^{n} y_i \log(\hat{y}_i)$$



2. **Regularization**

Regularization is used to reduce overfitting, which occurs when the model performs well on training data but poorly on unseen data. Regularization adds a penalty term to the loss function to discourage complex models and large weights.

- **L1 Regularization (Lasso):** Adds the absolute value of the weights to the loss function.

$$L_{\text{L1}} = \sum_{j=1}^{p} |w_j|$$

  This encourages sparsity, meaning it will drive some weights to zero and select important features.

- **L2 Regularization (Ridge):** Adds the square of the weights to the loss function.

$$L_{\text{L2}} = \sum_{j=1}^{p} w_j^2$$

  This discourages large weights and prevents overfitting without promoting sparsity.

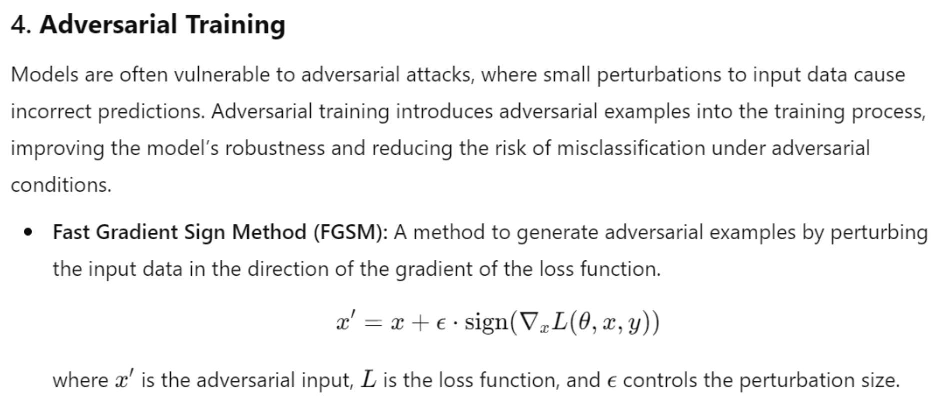
- **Elastic Net:** Combines both L1 and L2 regularization to balance between feature selection and shrinkage.

$$L_{\text{Elastic Net}} = \alpha L_{\text{L1}} + (1-\alpha) L_{\text{L2}}$$

3. Bayesian Risk Minimization

In Bayesian learning, risk minimization involves considering the posterior distribution of parameters to make predictions. By incorporating prior knowledge and model uncertainty, Bayesian methods provide a more robust approach to minimizing risk.

- **Bayesian Neural Networks (BNN):** Instead of point estimates, BNNs treat the weights of a neural network as distributions, allowing them to quantify uncertainty in predictions.
- **Monte Carlo Dropout:** This is an approximation of Bayesian inference in deep learning. By applying dropout at test time, it provides uncertainty estimates, making the model less prone to overconfident and risky predictions.



4. **Adversarial Training**

Models are often vulnerable to adversarial attacks, where small perturbations to input data cause incorrect predictions. Adversarial training introduces adversarial examples into the training process, improving the model's robustness and reducing the risk of misclassification under adversarial conditions.

- **Fast Gradient Sign Method (FGSM):** A method to generate adversarial examples by perturbing the input data in the direction of the gradient of the loss function.

$$x' = x + \epsilon \cdot \text{sign}(\nabla_x L(\theta, x, y))$$

  where x' is the adversarial input, L is the loss function, and ϵ controls the perturbation size.

NAÏVE BAYES CLASSIFIER

The **Naive Bayes classifier** is a simple yet powerful machine learning algorithm based on **Bayes' Theorem** with the assumption of independence among predictors. It's particularly useful for classification tasks, especially in text classification, spam detection, sentiment analysis, and medical diagnosis.

1. **Bayes' Theorem**: It provides a way to calculate the posterior probability $P(y|x)$, where y is the target class and x is the feature vector.

$$P(y|x) = \frac{P(x|y) \cdot P(y)}{P(x)}$$

- $P(y|x)$: Posterior probability of class y given the feature vector x.
- $P(x|y)$: Likelihood of feature vector x given the class y.
- $P(y)$: Prior probability of class y.
- $P(x)$: Probability of feature vector x (normalization constant).

2. **Naive Assumption**: The Naive Bayes algorithm assumes that the features are independent of each other, i.e., the presence of one feature does not influence the presence of another. This makes the calculation of $P(x|y)$ much simpler:

$$P(x|y) = P(x_1|y) \cdot P(x_2|y) \cdot \ldots \cdot P(x_n|y)$$

where $x_1, x_2, \ldots, x_n$ are the features in the feature vector x.

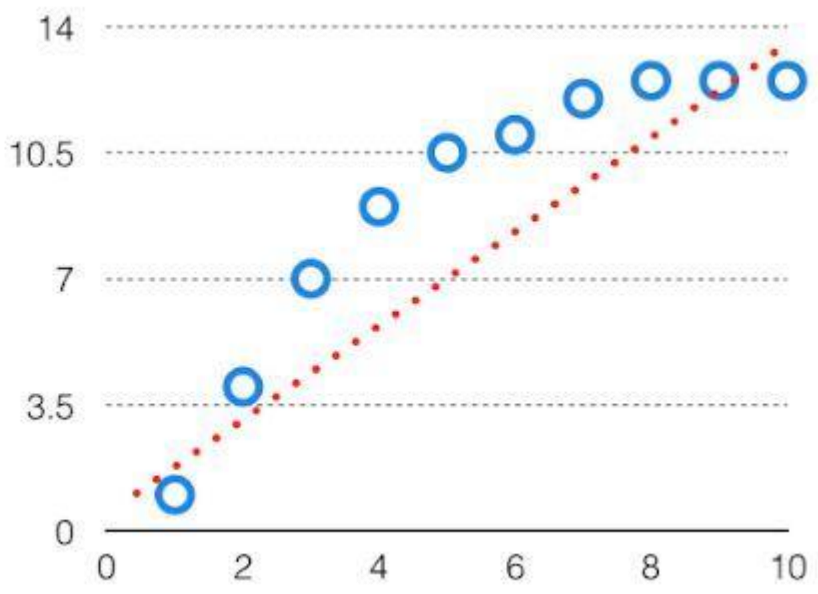
3. **Classification Rule**: For a given feature vector x, the Naive Bayes classifier predicts the class y that maximizes the posterior probability $P(y|x)$:

$$\hat{y} = \arg\max_y P(y|x) = \arg\max_y P(y) \cdot P(x|y)$$

Since $P(x)$ is constant for all classes, it can be ignored in the maximization process.

What is Bias?

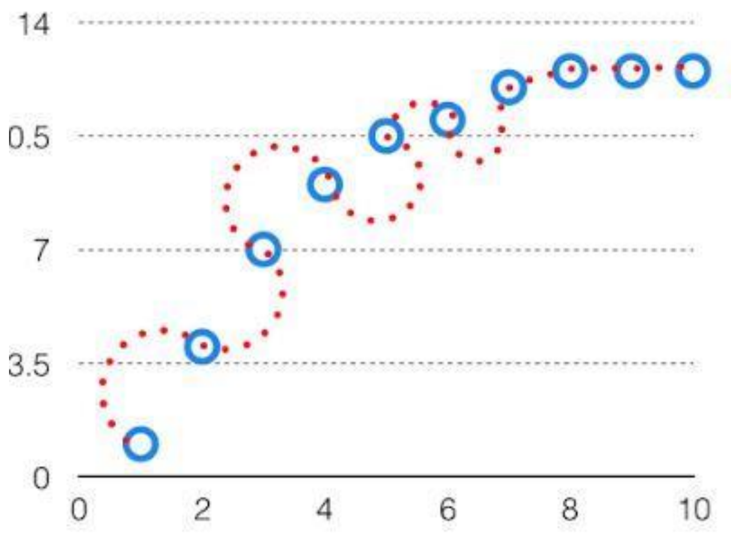
The bias is known as the difference between the prediction of the values by the Machine Learning model and the correct value. Being high in biasing gives a large error in training as well as testing data. It recommended that an algorithm should always be low-biased to avoid the problem of underfitting. By high bias, the data predicted is in a straight line format, thus not fitting accurately in the data in the data set. Such fitting is known as the **Underfitting of Data**. This happens when the hypothesis is too simple or linear in nature. Refer to the graph given below for an example of such a situation.



High Bias in the Model

What is Variance?

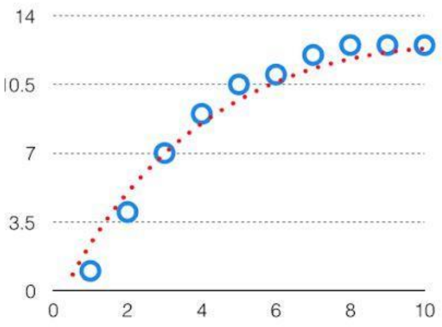
The variability of model prediction for a given data point which tells us the spread of our data is called the variance of the model. The model with high variance has a very complex fit to the training data and thus is not able to fit accurately on the data which it hasn't seen before. As a result, such models perform very well on training data but have high error rates on test data. When a model is high on variance, it is then said to as **Overfitting of Data**. Overfitting is fitting the training set accurately via complex curve and high order hypothesis but is not the solution as the error with unseen data is high. While training a data model variance should be kept low. The high variance data looks as follows.



High Variance in the Model

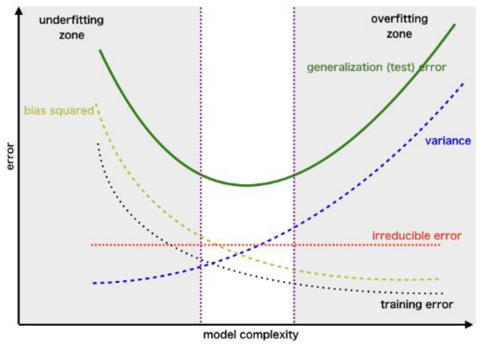
Bias Variance Tradeoff

If the algorithm is too simple (hypothesis with linear equation) then it may be on high bias and low variance condition and thus is error-prone. If algorithms fit too complex (hypothesis with high degree equation) then it may be on high variance and low bias. In the latter condition, the new entries will not perform well. Well, there is something between both of these conditions, known as a Trade-off or Bias Variance Trade-off. This tradeoff in complexity is why there is a tradeoff between bias and variance. An algorithm can't be more complex and less complex at the same time. For the graph, the perfect tradeoff will be like this.



We try to optimize the value of the total error for the model by using the Bias-Variance Tradeoff.

The best fit will be given by the hypothesis on the tradeoff point. The error to complexity graph to show trade-off is given as –



Region for the Least Value of Total Error

This is referred to as the best point chosen for the training of the algorithm which gives low error in training as well as testing data.

MODULE 2: Linear Model for Classification

2.1 Linear Basis Function Model

The **Linear Basis Function Model** is a way of extending simple linear regression to model more complex, non-linear relationships between input features and the target variable. Instead of using the raw features directly, this approach uses **basis functions** to transform the input data into a new space, where linear regression can still be applied effectively.

1. **Linear Model in Original Space**: A linear regression model has the form:

$$y(x) = w_0 + w_1 x_1 + w_2 x_2 + \cdots + w_d x_d$$

where $x_1, x_2, \dots, x_d$ are the input features, and $w_0, w_1, \dots, w_d$ are the weights.

However, this model assumes a linear relationship between the features x_i and the target y, which might not always hold.

2. **Basis Functions**: The **basis function** approach allows us to apply linear models in higher-dimensional spaces by transforming the inputs into a new set of features. This enables the model to capture non-linear relationships.

Suppose $\phi_1(x), \phi_2(x), \dots, \phi_M(x)$ are basis functions applied to the input x, the linear model becomes:

$$y(x) = w_0 + w_1 \phi_1(x) + w_2 \phi_2(x) + \cdots + w_M \phi_M(x)$$

where $\phi_i(x)$ is a transformed version of the original input x using a basis function.

3. **Linear in Parameters**: Even though the basis functions may be non-linear, the model is **linear in the parameters** $w_1, w_2, \dots, w_M$. Therefore, techniques such as **least squares regression** can still be used to estimate these parameters.

4. **Choice of Basis Functions**: Different types of basis functions can be used depending on the nature of the data and the complexity of the relationship you're trying to model:

   - **Polynomial Basis Functions**:

$$\phi_j(x) = x^j \quad \text{for } j = 0, 1, 2, \dots, M$$

These allow the model to fit polynomial relationships of different degrees.

- **Gaussian Basis Functions**:

$$\phi_j(x) = \exp\left(-\frac{(x-\mu_j)^2}{2s^2}\right)$$

where μ_j and s are parameters of the Gaussian. These are useful for fitting localized data patterns.

- **Sigmoidal Basis Functions**:

$$\phi_j(x) = \frac{1}{1 + \exp(-(x-\mu_j))}$$

These are useful for classification problems where the response variable is binary.

- **Fourier Basis Functions**: Often used for periodic data.

$$\phi_j(x) = \sin(jx), \cos(jx)$$

5. **Matrix Representation**: In the linear basis function model, the transformation of inputs can be represented as a matrix Φ, where each row corresponds to a transformed feature vector for a given data point.

   - Input matrix X (raw features):

$$X = \begin{bmatrix} x_1^{(1)} & x_2^{(1)} & \dots & x_d^{(1)} \\ x_1^{(2)} & x_2^{(2)} & \dots & x_d^{(2)} \\ \vdots & \vdots & & \vdots \\ x_1^{(N)} & x_2^{(N)} & \dots & x_d^{(N)} \end{bmatrix}$$

- Transformed matrix Φ (basis functions applied):

$$\Phi = \begin{bmatrix} \phi_1(x^{(1)}) & \phi_2(x^{(1)}) & \dots & \phi_M(x^{(1)}) \\ \phi_1(x^{(2)}) & \phi_2(x^{(2)}) & \dots & \phi_M(x^{(2)}) \\ \vdots & \vdots & & \vdots \\ \phi_1(x^{(N)}) & \phi_2(x^{(N)}) & \dots & \phi_M(x^{(N)}) \end{bmatrix}$$

The parameters w are estimated by solving:

$$w = (\Phi^T \Phi)^{-1} \Phi^T y$$

where y is the target vector and w are the model parameters.

2.2 Bayesian Regression

Bayesian regression is a type of linear regression that uses Bayesian statistics to estimate the unknown parameters of a model. It uses Bayes' theorem to estimate the likelihood of a set of parameters given observed data. The goal of Bayesian regression is to find the best estimate of the parameters of a linear model that describes the relationship between the independent and the dependent variables.

The main difference between traditional linear regression and Bayesian regression is the underlying assumption regarding the data-generating process. Traditional linear regression assumes that data follows a Gaussian or normal distribution, while Bayesian regression has stronger assumptions about the nature of the data and puts a prior probability distribution on the parameters. Bayesian regression also enables more flexibility as it allows for additional parameters or prior distributions, and can be used to construct an arbitrarily complex model that explicitly expresses prior beliefs about the data. Additionally, Bayesian regression provides more accurate predictive measures from fewer data points and is able to construct estimates for uncertainty around the estimates. On the other hand, traditional linear regressions are easier to implement and generally faster with simpler models and can provide good results when the assumptions about the data are valid.

Bayesian Regression can be very useful when we have insufficient data in the dataset or the data is poorly distributed. The output of a Bayesian Regression model is obtained from a probability distribution, as compared to regular regression techniques where the output is just obtained from a single value of each attribute.

1. **Prior Distribution**:
   - Represents your beliefs about the model parameters w before observing the data.
   - Often a normal distribution: $P(w) = \mathcal{N}(w_0, \sigma^2)$.
2. **Likelihood**:
   - Based on the observed data $D = (X, y)$, it measures how likely the data is given the parameters w.
   - Usually modeled as a Gaussian: $P(y|X, w) = \mathcal{N}(Xw, \sigma^2)$.
3. **Posterior Distribution**:
   - Using **Bayes' Theorem**, the posterior distribution of the parameters is computed by combining the prior with the likelihood:

   $$P(w|X, y) = \frac{P(y|X, w)P(w)}{P(y|X)}$$

   This gives the updated belief about w after seeing the data.
4. **Prediction**:
   - Instead of a single prediction, Bayesian regression provides a distribution of possible outcomes. The predictive distribution is:

   $$P(y_{\text{new}}|X_{\text{new}}, X, y) = \int P(y_{\text{new}}|X_{\text{new}}, w)P(w|X, y)\, dw$$

2.3 Discriminant Functions

A **discriminant function** is a function used in classification algorithms to assign input data points to one of several predefined classes. The function computes a score for each class, and the class with the highest score is chosen as the predicted label for the input data.

1. **Discriminant Function**:
   - A mathematical function $g(x)$ that takes an input x and returns a score for each possible class.
   - For classifying into K classes, the discriminant function is defined as:

   $$g_k(x) \quad \text{for} \quad k = 1, 2, \ldots, K$$

   - The predicted class is the one with the largest discriminant score:

   $$\hat{y} = \arg\max_k g_k(x)$$

2. **Types of Discriminant Functions**:
   - **Linear Discriminant Function**: When the discriminant function is linear in x, it has the form:

     $$g_k(x) = w_k^T x + b_k$$

     This is used in **Linear Discriminant Analysis (LDA)**.
   - **Quadratic Discriminant Function**: If the discriminant function includes quadratic terms of x, it is used in **Quadratic Discriminant Analysis (QDA)**:

     $$g_k(x) = x^T A_k x + w_k^T x + b_k$$

     where A_k is a matrix that models quadratic interactions.

2.4, 2.5 Probabilistic Generative Models and Probabilistic Discriminative Models

Probabilistic models are an essential component of machine learning, which aims to learn patterns from data and make predictions on new, unseen data. They are statistical models that capture the inherent uncertainty in data and incorporate it into their predictions. Probabilistic models are used in various applications such as image and speech recognition, natural language processing, and recommendation systems. In recent years, significant progress has been made in developing probabilistic models that can handle large datasets efficiently.

Categories Of Probabilistic Models

These models can be classified into the following categories:

- Generative models
- Discriminative models.

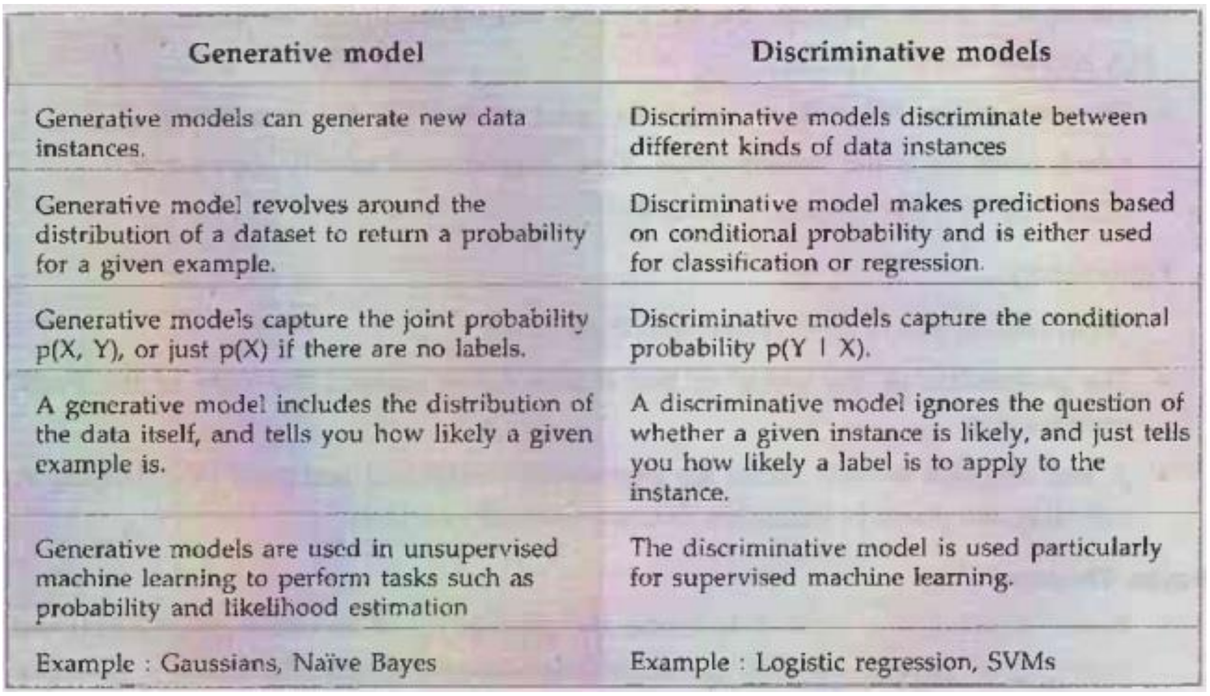
Generative models:

Generative models aim to model the joint distribution of the input and output variables. These models generate new data based on the probability distribution of the original dataset. Generative models are powerful because they can generate new data that resembles the training data. They can be used for tasks such as image and speech synthesis, language translation, and text generation.

Discriminative models

The discriminative model aims to model the conditional distribution of the output variable given the input variable. They learn a decision boundary that separates the different classes of the output variable. Discriminative models are useful when the focus is on making accurate predictions rather than generating new data. They can be used for tasks such as image recognition, speech recognition, and sentiment analysis.

Difference between Generative and Discriminative Models



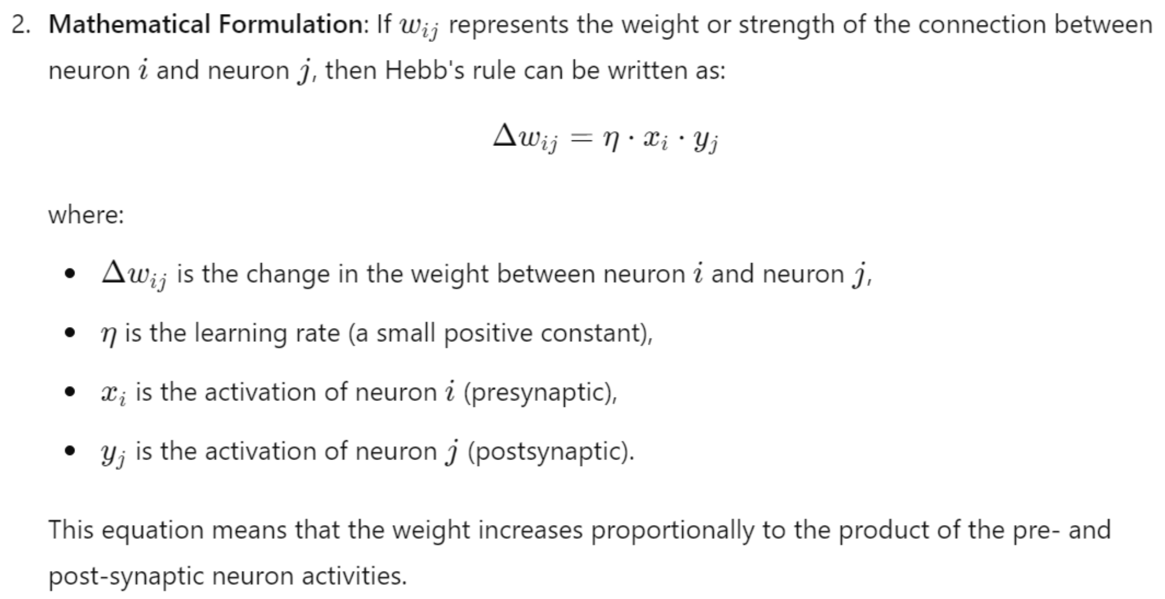
Generative model	Discriminative models
Generative models can generate new data instances.	Discriminative models discriminate between different kinds of data instances
Generative model revolves around the distribution of a dataset to return a probability for a given example.	Discriminative model makes predictions based on conditional probability and is either used for classification or regression.
Generative models capture the joint probability p(X, Y), or just p(X) if there are no labels.	Discriminative models capture the conditional probability p(Y | X).
A generative model includes the distribution of the data itself, and tells you how likely a given example is.	A discriminative model ignores the question of whether a given instance is likely, and just tells you how likely a label is to apply to the instance.
Generative models are used in unsupervised machine learning to perform tasks such as probability and likelihood estimation	The discriminative model is used particularly for supervised machine learning.
Example : Gaussians, Naïve Bayes	Example : Logistic regression, SVMs

MODULE 3: Neurons, Neural Networks and Linear Discriminants

HEBB'S RULE

Hebb's Rule is a foundational learning rule in neural networks and cognitive neuroscience, introduced by psychologist Donald Hebb in 1949. It describes how synaptic connections between neurons strengthen or weaken over time based on their activity. In essence, it states that **"neurons that fire together, wire together."**

1. **Synaptic Strengthening**:
    - Hebb's rule suggests that if two neurons (a presynaptic neuron and a postsynaptic neuron) are repeatedly activated at the same time, the synaptic connection between them strengthens.
    - In other words, when neuron A consistently helps neuron B to fire, the connection between them becomes stronger.



2. **Mathematical Formulation**: If w_{ij} represents the weight or strength of the connection between neuron i and neuron j, then Hebb's rule can be written as:

$$\Delta w_{ij} = \eta \cdot x_i \cdot y_j$$

where:

- Δw_{ij} is the change in the weight between neuron i and neuron j,
- η is the learning rate (a small positive constant),
- x_i is the activation of neuron i (presynaptic),
- y_j is the activation of neuron j (postsynaptic).

This equation means that the weight increases proportionally to the product of the pre- and post-synaptic neuron activities.

1. **Biological Interpretation**:
    - Hebb's rule reflects a basic biological principle of how learning and memory work in the brain. When neurons repeatedly activate together, their connection is strengthened, leading to more efficient communication between them in the future.
    - This mechanism is thought to underlie long-term potentiation (LTP), a process observed in the brain that strengthens synapses during learning.
2. **Simple Explanation**:
    - If neuron A frequently activates neuron B, the synapse connecting them becomes stronger.
    - Conversely, if neuron A does not activate neuron B, the synapse may weaken over time.
3. **Limitations**:

- Hebb's rule alone does not account for cases where synaptic connections need to weaken (known as **synaptic pruning** or **long-term depression**).
- It also does not include mechanisms for normalizing the total input strength, which could lead to unconstrained weight growth.

Applications:

- **Neural Networks**: Hebb's rule has inspired modern learning algorithms like **Hebbian Learning** in artificial neural networks.
- **Memory Formation**: In cognitive neuroscience, Hebb's rule is central to theories of how memory is formed in the brain through synaptic plasticity.

Mc Culloch PITTS NEURON MODEL

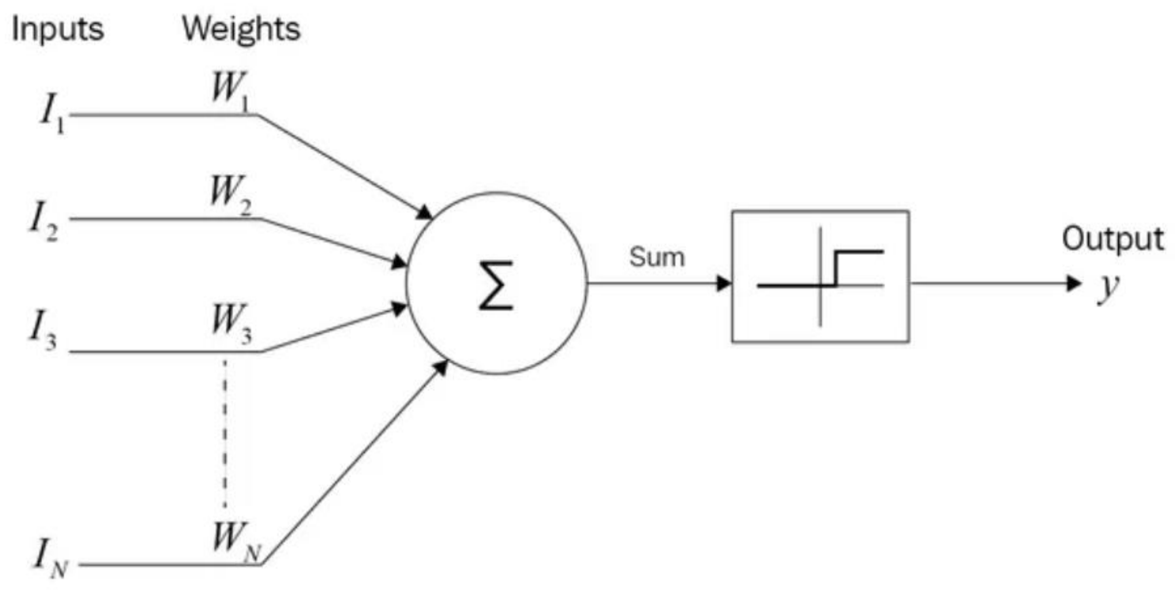
What are Biological Neurons?

Biological neurons are the fundamental units of the brain. They consist of:

- **Dendrite:** Receives signals from other neurons.
- **Soma:** Processes the information.
- **Axon:** Transmits the output to other neurons.
- **Synapse:** Connection points to other neurons.

A neuron functions like a tiny biological computer, taking input signals, processing them, and passing on the output.

What is McCulloch-Pitts Neuron?



The McCulloch-Pitts Neuron is the first computational model of a neuron. It can be divided into two parts:

1. **Aggregation:** The neuron aggregates multiple boolean inputs (0 or 1).

2. **Threshold Decision:** Based on the aggregated value, the neuron makes a decision using a threshold function.

Example Scenario

Imagine wanting to predict whether to watch a football game. The inputs (boolean values) could be:

- X1: Is Premier League on? (1 if yes, 0 if no)
- X2: Is it a friendly game? (1 if yes, 0 if no)
- X3: Are you not at home? (1 if yes, 0 if no)
- X4: Is Manchester United playing? (1 if yes, 0 if no)

Each input can be excitatory or inhibitory. For instance, X3 is inhibitory because you can't watch the game at home.

Thresholding Logic

The neuron fires (outputs 1) if the aggregated sum of inputs meets or exceeds a threshold value (θ). For example, if you always watch the game when at least two conditions are met, θ would be 2.

Note: It's a foundational model. It uses binary inputs (0 or 1) and lacks learning mechanisms, which later models introduced.

Boolean Functions Using the McCulloch-Pitts Neuron

The McCulloch-Pitts Neuron can represent various boolean functions:

- **AND Function:** Fires when all inputs are ON ((x1 + x2 + x3 >= 3)).
- **OR Function:** Fires when any input is ON ((x1 + x2 + x3 >= 1)).
- **Inhibitory Input Function:** Fires only when specific conditions are met (e.g., (x1) AND NOT (x2)).
- **NOR Function:** Fires when all inputs are OFF.
- **NOT Function:** Inverts the input.

Geometric Interpretation

The McCulloch-Pitts Neuron can be visualized geometrically by plotting inputs in a multi-dimensional space and drawing a decision boundary:

- **OR Function:** In 2D, the decision boundary is a line ((x1 + x2 = 1)).
- **AND Function:** The decision boundary is a line ((x1 + x2 = 2)).
- **Generalization:** The decision boundary becomes a plane in higher dimensions for more inputs.

Limitations of McCulloch-Pitts Neuron

Despite its pioneering role, the McCulloch-Pitts Neuron has limitations:

- Inability to handle non-boolean inputs.

- The requirement to manually set thresholds.
- All inputs are treated equally; no weighting mechanism.
- Cannot handle functions that are not linearly separable like XOR.

THE PERCEPTRON

Perceptron is one of the simplest Artificial neural network architectures. It was introduced by Frank Rosenblatt in 1957s. It is the simplest type of feedforward neural network, consisting of a single layer of input nodes that are fully connected to a layer of output nodes. It can learn the linearly separable patterns. it uses slightly different types of artificial neurons known as threshold logic units (TLU). it was first introduced by McCulloch and Walter Pitts in the 1940s.

Types of Perceptron

- **Single-Layer Perceptron:** This type of perceptron is limited to learning linearly separable patterns. effective for tasks where the data can be divided into distinct categories through a straight line.
- **Multilayer Perceptron**: Multilayer perceptrons possess enhanced processing capabilities as they consist of two or more layers, adept at handling more complex patterns and relationships within the data.

Basic Components of Perceptron

A perceptron, the basic unit of a neural network, comprises essential components that collaborate in information processing.

- **Input Features:** The perceptron takes multiple input features, each input feature represents a characteristic or attribute of the input data.
- **Weights**: Each input feature is associated with a weight, determining the significance of each input feature in influencing the perceptron's output. During training, these weights are adjusted to learn the optimal values.
- **Summation Function**: The perceptron calculates the weighted sum of its inputs using the summation function. The summation function combines the inputs with their respective weights to produce a weighted sum.
- **Activation Function**: The weighted sum is then passed through an activation function. Perceptron uses Heaviside step function functions. which take the summed values as input and compare with the threshold and provide the output as 0 or 1.
- **Output**: The final output of the perceptron, is determined by the activation function's result. For example, in binary classification problems, the output might represent a predicted class (0 or 1).
- **Bias**: A bias term is often included in the perceptron model. The bias allows the model to make adjustments that are independent of the input. It is an additional parameter that is learned during training.
- **Learning Algorithm (Weight Update Rule)**: During training, the perceptron learns by adjusting its weights and bias based on a learning algorithm. A common approach is the

perceptron learning algorithm, which updates weights based on the difference between the predicted output and the true output.

These components work together to enable a perceptron to learn and make predictions. While a single perceptron can perform binary classification, more complex tasks require the use of multiple perceptrons organized into layers, forming a neural network.

Working of a Perceptron:

1. **Weighted Sum**:
   - The perceptron calculates the weighted sum of its inputs:

$$z = w_1x_1 + w_2x_2 + \ldots + w_nx_n + b$$

2. **Activation**:
   - The perceptron applies an activation function to the weighted sum z. The output y is determined by:

$$y = \begin{cases} 1 & \text{if } z \geq 0 \\ 0 & \text{if } z < 0 \end{cases}$$

   This means that if the weighted sum is above a certain threshold (usually 0), the perceptron outputs 1 (indicating the positive class); otherwise, it outputs 0 (indicating the negative class).

3. **Learning Process**:
   - The perceptron adjusts its weights and bias based on the errors made during classification. This is done using a simple learning rule:

$$w_i \leftarrow w_i + \eta(y_{\text{true}} - y_{\text{pred}})x_i$$

   where:
   - η is the learning rate,
   - y_{true} is the actual class label,
   - y_{pred} is the predicted class label.

Training:

- The perceptron is trained using a dataset. During each iteration (epoch), it processes each input, computes the output, and updates the weights based on the error until the model converges (i.e., minimizes the classification error on the training data).

Linear Separability

Linear Separability refers to the data points in binary classification problems which can be separated using linear decision boundary. if the data points can be separated using a line, linear function, or flat hyperplane are considered linearly separable.

- Linear separability is an important concept in neural networks. If the separate points in n-dimensional space follows
  $w_1x_1 + w_2x_2 + ... + w_nx_n + b = 0$
  then it is said linearly separable
- For two-dimensional inputs, if there exists a line (whose equation is
  $w_1x_1 + w_2x_2 + b = 0$
) that separates all samples of one class from the other class, then an appropriate perception can be derived from the equation of the separating line. such classification problems are called "Linear separable" i.e, separating by a linear combination of i/p.

What is Linear Regression?

Linear regression is a type of supervised machine learning algorithm that computes the linear relationship between the dependent variable and one or more independent features by fitting a linear equation to observed data.

When there is only one independent feature, it is known as Simple Linear Regression, and when there are more than one feature, it is known as Multiple Linear Regression.

Similarly, when there is only one dependent variable, it is considered Univariate Linear Regression, while when there are more than one dependent variables, it is known as Multivariate Regression.

Why Linear Regression is Important?

The interpretability of linear regression is a notable strength. The model's equation provides clear coefficients that elucidate the impact of each independent variable on the dependent variable, facilitating a deeper understanding of the underlying dynamics. Its simplicity is a virtue, as linear regression is transparent, easy to implement, and serves as a foundational concept for more complex algorithms.

Linear regression is not merely a predictive tool; it forms the basis for various advanced models. Techniques like regularization and support vector machines draw inspiration from linear regression, expanding its utility. Additionally, linear regression is a cornerstone in assumption testing, enabling researchers to validate key assumptions about the data.

Types of Linear Regression

There are two main types of linear regression:

Simple Linear Regression

This is the simplest form of linear regression, and it involves only one independent variable and one dependent variable. The equation for simple linear regression is:
y=β0+β1X*y*=*β*0+*β*1*X*
where:

- Y is the dependent variable
- X is the independent variable
- β0 is the intercept
- β1 is the slope

Multiple Linear Regression

This involves more than one independent variable and one dependent variable. The equation for multiple linear regression is:
$y=\beta0+\beta1X1+\beta2X2+\dots\dots\dots\beta nXn$*$y=\beta0+\beta1X1+\beta2X2+\dots\dots\dots\beta nXn$*
where:

- Y is the dependent variable
- X1, X2, …, Xn are the independent variables
- β0 is the intercept
- β1, β2, …, βn are the slopes

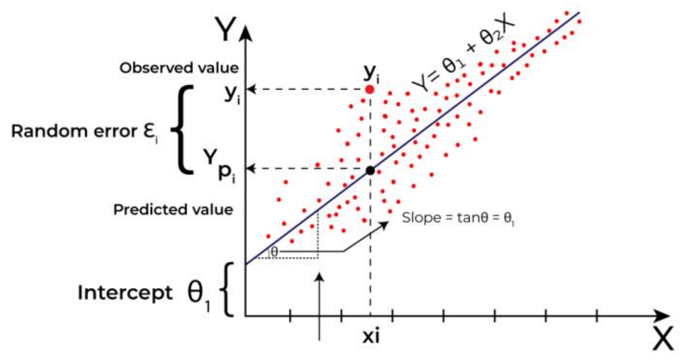
The goal of the algorithm is to find the best Fit Line equation that can predict the values based on the independent variables.

In regression set of records are present with X and Y values and these values are used to learn a function so if you want to predict Y from an unknown X this learned function can be used. In regression we have to find the value of Y, So, a function is required that predicts continuous Y in the case of regression given X as independent features.

What is the best Fit Line?

Our primary objective while using linear regression is to locate the best-fit line, which implies that the error between the predicted and actual values should be kept to a minimum. There will be the least error in the best-fit line.

The best Fit Line equation provides a straight line that represents the relationship between the dependent and independent variables. The slope of the line indicates how much the dependent variable changes for a unit change in the independent variable(s).



Linear Regression

Here Y is called a dependent or target variable and X is called an independent variable also known as the predictor of Y. There are many types of functions or modules that can be used for regression. A linear function is the simplest type of function. Here, X may be a single feature or multiple features representing the problem.

Linear regression performs the task to predict a dependent variable value (y) based on a given independent variable (x)). Hence, the name is Linear Regression. In the figure above, X (input) is the work experience and Y (output) is the salary of a person. The regression line is the best-fit line for our model.

We utilize the cost function to compute the best values in order to get the best fit line since different values for weights or the coefficient of lines result in different regression lines.