UNIT 2

SUBJECT: MACHINE LEARNING TECHNIQUES

What is Regression

Regression is a technique in machine learning and statistics used to find relationships between variables and make predictions. It helps us understand how the value of one thing (called the dependent variable) changes when other things (called independent variables) change.

For example- If you want to predict the price of a house based on its size, regression can help create a model that shows how the house price changes as the size (in square feet) increases.

This way, you can estimate the price of a house just by knowing its size.

In simple terms, regression is like drawing a line (or curve) through data points to see how one thing affects another and make predictions about future outcomes.

Types of Regression

1. Linear Regression

Linear Regression is a simple statistical method used to predict a continuous outcome (like price or temperature) based on the relationship between two variables, where one is the input (independent variable) and the other is the output (dependent variable). It models this relationship using a straight line to make predictions.

The formula for linear regression is:

Y=mX+b

Where:

- **Y** is the predicted output (dependent variable).
- **X** is the input (independent variable).
- **m** is the slope of the line (how much Y changes with X).
- **b** is the intercept (the value of Y when X is 0).

Key Points:

- Linear regression assumes a **linear relationship** between the input and output.
- It tries to minimize the difference between the actual data points and the line (using a method called least squares).

Example:

If you want to predict a house's price based on its size, linear regression can help you find a straight line that best fits the data points representing different house sizes and their corresponding prices.

2. Logistic Regression

Logistic Regression is a statistical method used in machine learning to predict a categorical outcome (like yes/no, true/false) based on one or more input variables. Unlike linear regression, it predicts probabilities and uses a logistic function to model a binary outcome (0 or 1).

In simple terms, it's used when the result is something like "Will it happen or not?" instead of predicting a continuous value.

Example:

If you want to predict whether an email is spam or not based on features like the presence of certain words, the sender's address, or the email length, logistic regression can help you estimate the probability that an email is spam.

The formula for Logistic Regression is:

$$P(Y=1)=\frac{1}{1+e^{-(b_0+b_1X)}}$$

Where:

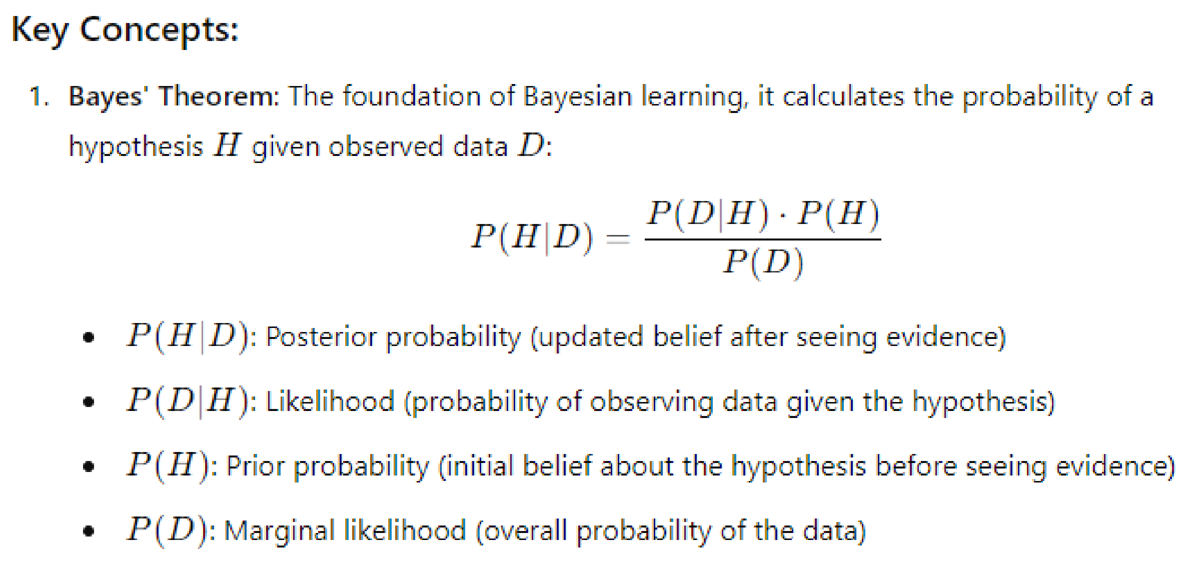
- $P(Y=1)$ is the predicted probability that the dependent variable Y equals 1 (e.g., event occurs).
- X is the input (independent variable).
- b_0 is the intercept (the value of the log-odds when X is 0).
- b_1 is the coefficient (the change in the log-odds of Y for a one-unit change in X).
- e is the base of the natural logarithm (approximately equal to 2.71828).

Explanation:

- The logistic function transforms the linear combination of inputs b_0+b_1X into a probability between 0 and 1, making it suitable for binary classification tasks.

What is Bayesian Learning

Bayesian Learning is a statistical approach to machine learning that applies Bayes' theorem to update the probability of a hypothesis as new evidence is acquired. It focuses on incorporating prior knowledge and evidence to improve predictions and decision-making.



Key Concepts:

1. **Bayes' Theorem:** The foundation of Bayesian learning, it calculates the probability of a hypothesis H given observed data D:

$$P(H|D) = \frac{P(D|H) \cdot P(H)}{P(D)}$$

- $P(H|D)$: Posterior probability (updated belief after seeing evidence)
- $P(D|H)$: Likelihood (probability of observing data given the hypothesis)
- $P(H)$: Prior probability (initial belief about the hypothesis before seeing evidence)
- $P(D)$: Marginal likelihood (overall probability of the data)

1. **Prior Knowledge**: Bayesian learning allows the incorporation of prior beliefs or information about a problem, which can influence the learning process.
2. **Updating Beliefs**: As new data becomes available, Bayesian learning updates the prior beliefs to form a new posterior belief, allowing for continuous learning and adaptation.
3. **Probabilistic Models**: Bayesian learning often employs probabilistic models, such as Bayesian networks or Gaussian processes, to represent uncertainties and make predictions.

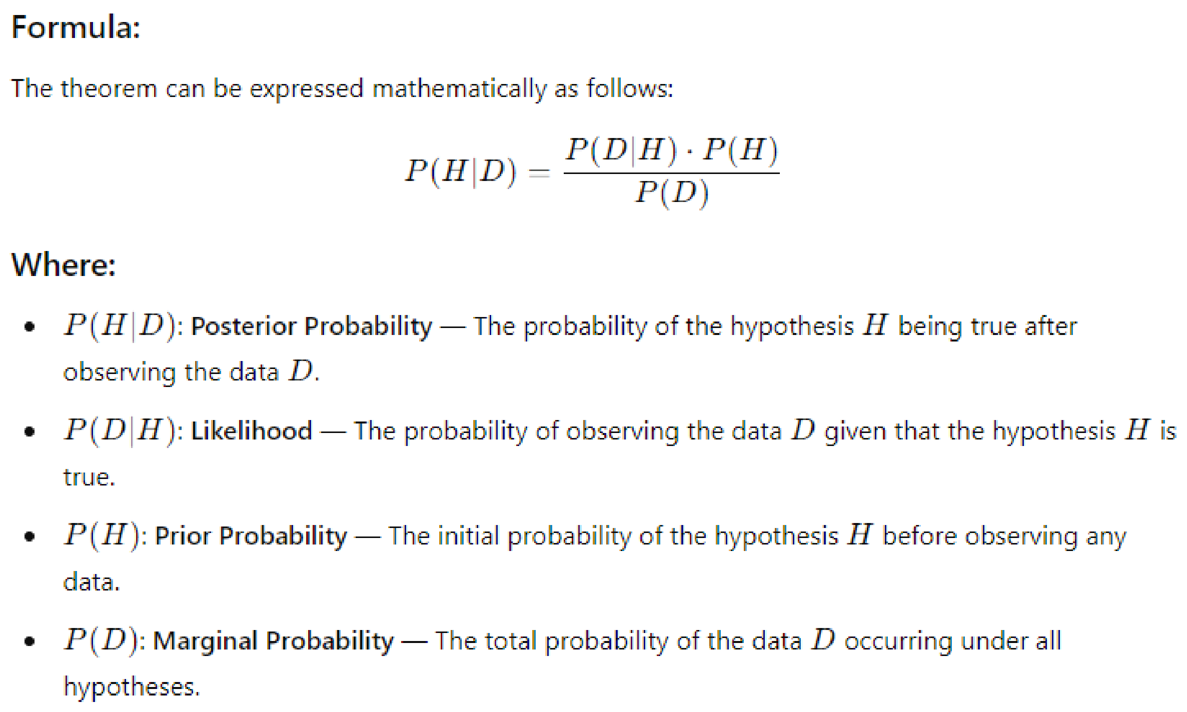
Applications:

- **Medical Diagnosis**: Updating the probability of diseases based on symptoms and test results.
- **Spam Detection**: Classifying emails as spam or not spam by updating beliefs based on previous data.
- **Recommender Systems**: Personalizing recommendations by incorporating user preferences and behavior.

In summary, **Bayesian Learning** is a powerful framework for modeling uncertainty and making informed predictions by combining prior knowledge with observed data.

What is Bayes Theorem

Bayes' Theorem is a fundamental concept in probability theory that describes how to update the probability of a hypothesis based on new evidence. It provides a way to calculate the conditional probability of an event based on prior knowledge and observed data.



Formula:

The theorem can be expressed mathematically as follows:

$$P(H|D) = \frac{P(D|H) \cdot P(H)}{P(D)}$$

Where:

- $P(H|D)$: **Posterior Probability** — The probability of the hypothesis H being true after observing the data D.
- $P(D|H)$: **Likelihood** — The probability of observing the data D given that the hypothesis H is true.
- $P(H)$: **Prior Probability** — The initial probability of the hypothesis H before observing any data.
- $P(D)$: **Marginal Probability** — The total probability of the data D occurring under all hypotheses.

Explanation:

Prior Probability reflects your belief about the hypothesis before considering the evidence.

Likelihood measures how well the hypothesis explains the observed data.

Posterior Probability is the updated belief after taking the evidence into account.

Example:

Suppose you want to determine the probability that someone has a disease (hypothesis H) after testing positive for it (evidence D).

You would use Bayes' theorem to update your belief about the disease's probability based on the test's accuracy and the general prevalence of the disease in the population.

Importance:

Bayes' theorem is widely used in various fields, including statistics, machine learning, finance, and medicine, as it provides a coherent method for reasoning about uncertainty and making decisions based on incomplete information.

<u>Question Example</u>

Question:

You are a doctor, and you know that there is a disease (F) known as "tuberculosis." You have observed that:

- 5% of people have tuberculosis.
$$P(F) = 0.05$$
- If someone has tuberculosis, 90% of them have a persistent cough.
$$P(C|F) = 0.9$$
- However, if someone does not have tuberculosis, only 10% of them have a persistent cough.
$$P(C|\neg F) = 0.1$$

If someone has a persistent cough (C), what is the probability that they have tuberculosis (P(F|C))?

Answer Steps

1. **Prior Probability:**

$$P(F) = 0.05$$

-

$$P(\neg F) = 1 - P(F) = 1 - 0.05 = 0.95$$

2. **Likelihood:**

$$P(C|F) = 0.9$$

- $$P(C|\neg F) = 0.1$$

3. **Total Probability $P(C)$:**

$$P(C) = P(C|F) \cdot P(F) + P(C|\neg F) \cdot P(\neg F)$$

- $$P(C) = (0.9 \cdot 0.05) + (0.1 \cdot 0.95)$$

- $$P(C) = 0.045 + 0.095 = 0.14$$

4. Posterior Probability using Bayes' Theorem:

$$P(F|C) = \frac{P(C|F) \cdot P(F)}{P(C)}$$

$$P(F|C) = \frac{0.9 \cdot 0.05}{0.14} = \frac{0.045}{0.14} \approx 0.3214$$

Final Answer:

So, if someone has a persistent cough, the probability that they have tuberculosis is approximately **32.14%** (0.3214).

What is Concept Learning

Concept Learning is a fundamental aspect of machine learning, where a machine identifies and understands general rules or concepts from specific examples. This process allows the machine to classify new instances based on learned patterns. Here's a breakdown of the key elements and processes involved in concept learning:

In simple terms, concept learning is about teaching a machine to recognize a category based on a set of examples. For instance, if we want the machine to learn the concept of "fruit," we provide it with examples of fruits (like apples and bananas) and non-fruits (like carrots and potatoes).

Components of Concept Learning:

1. Instances: These are the individual examples being classified. For example, instances can be specific fruits or animals.

2. Target Concept: This is the actual concept we want the machine to learn, such as "fruit" or "bird."

3. Hypothesis Space: The set of all possible rules or hypotheses that the machine can consider based on the given examples.

4. Hypothesis: The final rule or concept that the machine learns to classify the examples correctly.

5. Positive and Negative Examples:

Positive Examples: Instances that belong to the target concept (e.g., fruits like apple, banana).

Negative Examples: Instances that do not belong to the target concept (e.g., vegetables like carrot, cucumber).

Example of Concept Learning:

Consider teaching a machine the concept of a “bird.”

Positive examples: Sparrow, Eagle, Parrot (they are birds).

Negative examples: Bat, Dog, Airplane (they are not birds).

The machine learns that birds usually have features such as wings and feathers, and many can fly. Its goal is to develop a hypothesis (a rule) that can help it predict if a new animal (like a Penguin) is a bird.

Process of Concept Learning:

1. Representation of the Hypothesis Space: The machine begins by defining all potential rules or hypotheses based on features of the instances (e.g., wings, size).

2. Generalization and Specialization:

Generalization: The machine looks for common features in positive examples.

Specialization: The machine refines its hypothesis to ensure it excludes negative examples.

This process is iterative and continues until the machine finds the best hypothesis.

3. Evaluating the Hypothesis: Once a hypothesis is formed, the machine evaluates it based on its effectiveness in classifying the training examples and its ability to generalize to new instances.

Key Challenges in Concept Learning:

Overfitting: This occurs when the hypothesis is too specific, making it perform well on training data but poorly on new, unseen examples.

Underfitting: This happens when the hypothesis is too broad and fails to capture the underlying structure of the data, leading to inaccurate classifications.

Importance of Concept Learning in Machine Learning:

Concept learning is crucial because it forms the basis for many classification tasks in machine learning. It enables machines to:

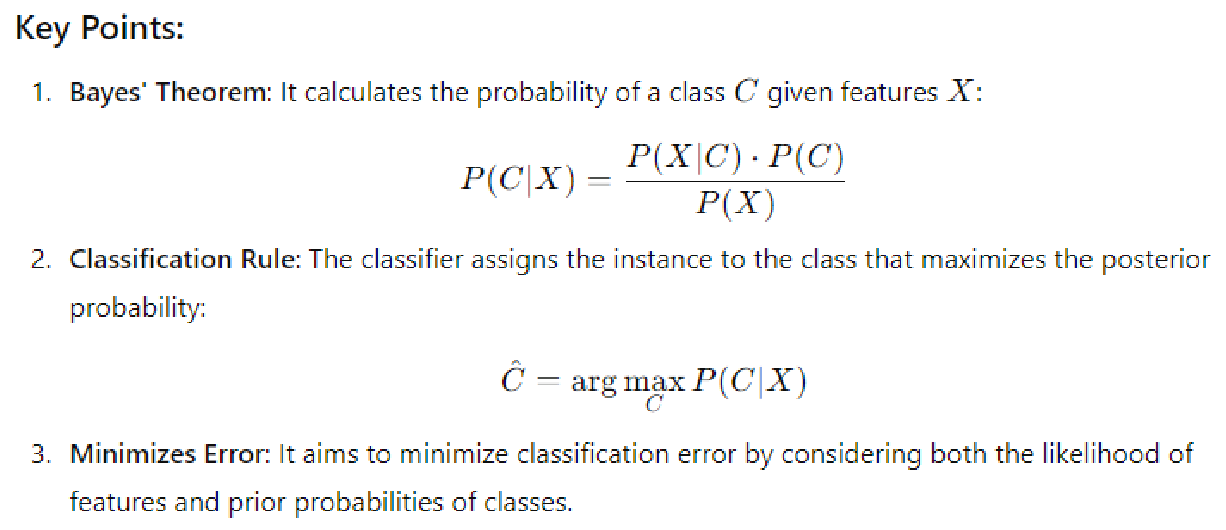
Recognize patterns in data.

Make decisions based on learned concepts.

Generalize knowledge to new data, which is essential for real-world applications such as spam detection, image recognition, and medical diagnostics.

What is Bayes Optimal Classifier

The Bayes Optimal Classifier is a statistical model used in machine learning for classification tasks. It utilizes Bayes' theorem to predict the class of an instance by calculating the posterior probabilities of each class based on observed features.



Key Points:

1. **Bayes' Theorem**: It calculates the probability of a class C given features X:

$$P(C|X) = \frac{P(X|C) \cdot P(C)}{P(X)}$$

2. **Classification Rule**: The classifier assigns the instance to the class that maximizes the posterior probability:

$$\hat{C} = \arg\max_{C} P(C|X)$$

3. **Minimizes Error**: It aims to minimize classification error by considering both the likelihood of features and prior probabilities of classes.

Example:

In spam email classification, it would compute the probability of an email being spam or not spam based on the presence of certain keywords, choosing the class with the highest probability.

Limitations:

- **Computationally Intensive**: Requires significant computation, especially with many features or classes.
- **Independence Assumption**: Assumes features are independent given the class, which may not be true in real scenarios.

Importance:

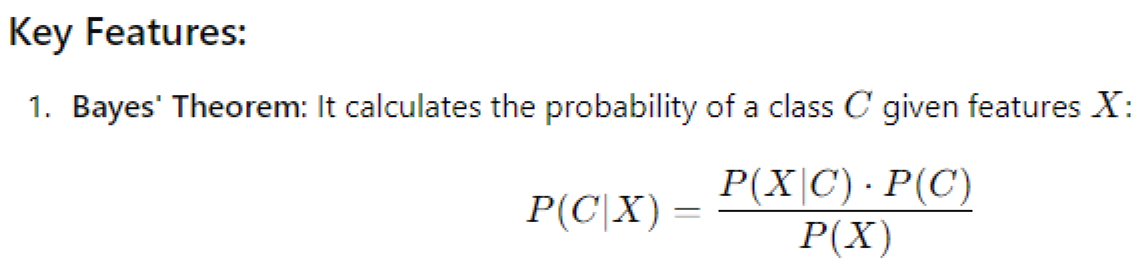
It serves as a benchmark for evaluating other classification algorithms, like Naive Bayes, which simplifies some of the assumptions of the Bayes Optimal Classifier.

What is Naïve Bayes Classifier

The Naïve Bayes Classifier is a simple yet powerful classification algorithm based on Bayes' theorem. It is called "naïve" because it assumes that the features (or attributes) used for classification are independent of each other, which is often not the case in real-world data.

Key Features:

1. Bayes' Theorem: It calculates the probability of a class C given features X:



Key Features:

1. **Bayes' Theorem**: It calculates the probability of a class C given features X:

$$P(C|X) = \frac{P(X|C) \cdot P(C)}{P(X)}$$

2. Independence Assumption: The classifier assumes that the presence of a feature in a class is independent of the presence of any other feature. This simplifies calculations and leads to:

$$P(X|C) = P(x_1|C) \cdot P(x_2|C) \cdots P(x_n|C)$$

where $X = (x_1, x_2, ..., x_n)$ are the features.

3. Classification Rule: It predicts the class that maximizes the posterior probability:

$$\hat{C} = \arg\max_C P(C|X) = \arg\max_C P(X|C) \cdot P(C)$$

Types of Naïve Bayes Classifiers:

1. Gaussian Naïve Bayes: Assumes that features follow a Gaussian (normal) distribution.

2. Multinomial Naïve Bayes: Suitable for discrete data, often used in text classification.

3. Bernoulli Naïve Bayes: Assumes binary features (e.g., presence/absence of a feature).

Example:

In email spam detection, the Naïve Bayes classifier would evaluate the presence of certain words in an email to classify it as "spam" or "not spam." Each word is treated as an independent feature, and the classifier calculates the probabilities accordingly.

Advantages:

Simplicity: Easy to understand and implement.

Efficiency: Works well with large datasets and is computationally efficient.

Performance: Often performs surprisingly well, even with the independence assumption.

Limitations:

Independence Assumption: The assumption of feature independence is rarely true, which can lead to inaccuracies.

Zero Probability Problem: If a feature is not present in the training data for a particular class, it may lead to a probability of zero. This can be mitigated using techniques like Laplace smoothing.

Importance:

The Naïve Bayes classifier is widely used in various applications, including text classification (spam detection, sentiment analysis), document categorization, and recommendation systems, due to its efficiency and effectiveness in handling large datasets.

What is Bayesian Belief Networks (BBNs)

Bayesian Belief Networks (BBNs), also known as Bayesian Networks, are graphical models that represent the probabilistic relationships among a set of variables using a directed acyclic graph (DAG).

Key Features:

1. Graph Structure:

Nodes represent random variables.

Directed edges indicate dependencies (causal relationships) between variables.

2. Conditional Probability Tables (CPT):

Each node has a CPT that defines the probability of the variable given its parent nodes.

3. Joint Probability Distribution:

The joint distribution of all variables can be expressed as the product of the conditional probabilities.

Example:

In a BBN involving Rain, Traffic Jam, and Accident:

Rain affects Traffic Jam, and Traffic Jam affects Accident.

Applications:

Used in medical diagnosis, decision-making systems, and predictive analytics.

Advantages:

Handles uncertainty well and shows causal relationships clearly.

What is EM Algorithm (Expectation-Maximization Algorithm)

The Expectation-Maximization (EM) Algorithm is a statistical method used for estimating the parameters of probabilistic models, particularly when dealing with incomplete or missing data. It is commonly applied in various fields like machine learning, computer vision, and natural language processing.

Key Concepts:

1. Latent Variables: These are variables that are not directly observed but are inferred from the observed data. The EM algorithm is particularly useful in scenarios involving latent variables.

2. Steps of the Algorithm:

Initialization: Start with initial guesses for the parameters.

E-Step (Expectation Step): Calculate the expected value of the log-likelihood function based on the current parameter estimates. This step involves estimating the distribution of the latent variables given the observed data.

M-Step (Maximization Step): Update the parameter estimates to maximize the expected log-likelihood calculated in the E-step.

3. Convergence: The algorithm iterates between the E-step and M-step until the parameter estimates converge, meaning they stabilize and do not change significantly.

Applications:

Clustering: Used in Gaussian Mixture Models to find clusters in data.

Image Processing: Helps in segmenting images with incomplete information.

Natural Language Processing: Useful for training models with missing data points.

Support Vector Machine (SVM)

Support Vector Machine (SVM) is a supervised learning algorithm used for both classification and regression tasks, though it is mostly used for classification problems. The basic concept of SVM is that it creates a hyperplane to classify data points that separate two classes.

Key Concepts of SVM:

1. **Hyperplane**: The main aim of SVM is to find a hyperplane that can efficiently separate the classes. In a two-dimensional space, a hyperplane is a line, while in higher-dimensional spaces, it can be a plane or a higher-dimensional surface.
2. **Support Vectors**: The data points that are closest to the hyperplane are called "Support Vectors." These points are critical to SVM as they determine the position of the hyperplane and help differentiate between the classes.
3. **Margin**: The distance between the hyperplane and the nearest support vectors on both sides is called the margin. SVM aims to find a hyperplane with the maximum margin, as a larger margin results in more robust classification.
4. **Linear and Non-Linear SVM**:

- **Linear SVM**: When the data can be linearly separated (by a straight line), linear SVM is used.
- **Non-Linear SVM**: Sometimes the data is not linearly separable, and SVM uses the “kernel trick” to map the data into a higher-dimensional space where linear separation is possible.

5. **Kernel Trick**: When data is not linearly separable, SVM applies kernel functions to project the data into higher dimensions, where it can be separated easily. Popular kernel functions include **Polynomial kernel**, **Radial Basis Function (RBF)**, and **Gaussian kernel**.

Types of SVM Kernels:

1. Linear Kernel:

Definition: The linear kernel is used when the data can be separated by a straight line (or hyperplane in higher dimensions). It is the simplest form of kernel.

Usage: It is typically used when the data is linearly separable, meaning the two classes can be separated by a straight boundary.

Example: Classifying spam and non-spam emails when the features (like words) can be separated by a linear boundary.

2. Polynomial Kernel:

Definition: The polynomial kernel is useful for datasets that are not linearly separable but can be separated using polynomial decision boundaries. It adds more complexity by creating curved boundaries.

Usage: It is used when the relationship between the data points is non-linear but can be captured by polynomial functions.

Example: When the data points form a circular or curved pattern, a polynomial kernel can be used to classify them.

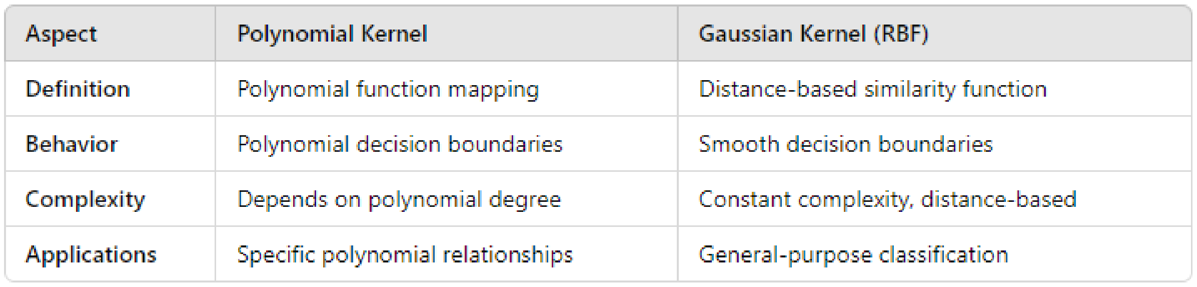
3. Gaussian Kernel (RBF - Radial Basis Function):

Definition: The Gaussian kernel (or RBF) is one of the most commonly used kernels for SVMs. It maps data into an infinite-dimensional space, allowing complex, non-linear boundaries to separate the classes.

Usage: It is used when the data is not linearly separable, and we need a flexible decision boundary to handle complex patterns.

Example: It is ideal for tasks like face recognition or handwriting classification, where the data cannot be separated by a simple linear boundary.

Polynomial Kernel vs Gaussian Kernel (RBF)



Aspect	Polynomial Kernel	Gaussian Kernel (RBF)
Definition	Polynomial function mapping	Distance-based similarity function
Behavior	Polynomial decision boundaries	Smooth decision boundaries
Complexity	Depends on polynomial degree	Constant complexity, distance-based
Applications	Specific polynomial relationships	General-purpose classification

Hyperplane – Decision Surface

The Hyperplane is a crucial concept in SVM (Support Vector Machine). Its function is to classify data points into different categories. The main goal of SVM is to create a hyperplane that serves as the optimal boundary between data points belonging to different categories.

Hyperplane: It is a line or plane that separates data points of different classes (categories).

If the data is in a 2D space, the hyperplane is a line.

If the data is in a 3D space, the hyperplane becomes a plane.

In higher dimensions, the hyperplane becomes a higher-dimensional surface.

Example:

Imagine you have red and blue points. The hyperplane creates a straight line or plane that separates these red and blue points.

Decision Surface:

The hyperplane is also called the decision surface because it helps SVM decide which category the points belong to.

Best Hyperplane: SVM's task is to find the hyperplane that creates the maximum margin between the classes, meaning it tries to keep the points of different classes as far away from the hyperplane as possible for more accurate classification.

Some properties OF SVM:

1. Maximum Margin Classifier:

SVM's objective is to find the maximum margin between classes to separate them as effectively as possible. The farther the hyperplane is from the class points, the better the classification.

2. Works Well with High-Dimensional Data:

SVM performs well even with high-dimensional data. If you have many features, SVM is efficient in understanding the relationship between them.

3. Effective with Non-Linear Data:

- SVM can efficiently classify non-linear data by using kernels (like Polynomial or Gaussian). By using kernels, SVM maps data to higher dimensions where the non-linear data can become linear.

4. Robustness to Overfitting:

If the data is well-separated, SVM reduces the chances of overfitting. Overfitting occurs when a model fits the training data so well that it doesn't perform well on testing data.

5. Support Vectors:

In SVM, only a few important points (called support vectors) are used to define the hyperplane. These points lie very close to the hyperplane and help form the decision boundary.

<u>Issues in SVM (Challenges with SVM)</u>

1. Computational Complexity:

If you have a very large dataset, the training process of SVM can be slow, as finding the best hyperplane requires significant computation.

2. Choice of Kernel:

For non-linear data, selecting the right kernel function is crucial. If the wrong kernel is chosen, SVM's performance can degrade. It's often difficult to determine which kernel (Polynomial, Gaussian, or others) is the best fit for the data.

3. Overfitting with Noisy Data:

If the data is very noisy (with many outliers), the hyperplane generated by SVM may not be accurate. SVM is sensitive to noisy data, which can lead to incorrect decision boundaries.

4. Interpretability:

The results of SVM can be harder to interpret. When working in high dimensions, it's not always easy to understand how or why the hyperplane is being formed.

5. Not Suitable for Large Datasets:

SVM works best with small to medium-sized datasets. For very large datasets, it becomes computationally expensive, and other algorithms like Decision Trees or Random Forests may perform better.