Support Vector Machines (SVM)

Support Vector Machines (SVMs) are a powerful and versatile class of supervised machine learning algorithms used for classification and regression tasks. At its core, SVM aims to find an optimal hyperplane that best separates data points into distinct classes. This hyperplane is chosen to maximize the margin between the two classes, meaning it should be positioned in such a way that it maximally separates the closest data points from each class. These closest data points are known as "support vectors." By focusing on support vectors, SVMs achieve better generalization to new, unseen data.

Introduction to Linear and Non linear data sets

Linear Data Sets

A linear dataset is one in which the relationship between the input features and the target variable can be approximated or modeled by a straight line or a hyperplane. Linear datasets are straightforward to work with because the underlying patterns are relatively simple and can be captured by linear models. Here's an example:

Example: Predicting House Prices

- Suppose you want to predict house prices based on features like the number of bedrooms, square footage, and age of the house. If the relationship between these features and the house price can be accurately represented by a linear equation like

  House Price = (Number of Bedrooms * w1) + (Square Footage * w2) + (Age of the House * w3) + b

- In this case, the dataset is considered linear. Linear regression is a common algorithm used for such problems.
- If you were to plot the data points on a graph, they would form a clear straight line or a hyperplane that approximates the relationship between the features and the target variable.

Non-Linear Data Sets

A non-linear dataset is one in which the relationship between the input features and the target variable cannot be effectively modeled by a straight line or a hyperplane. Non-linear datasets often exhibit complex, curved, or irregular patterns that require more sophisticated models to capture. Here's an example

Example: Predicting a Student's Exam Score

- Suppose you want to predict a student's exam score based on the number of hours they studied and the number of hours they slept the night before. The relationship between these features and the exam score might not be linear.
- It could be that studying more hours initially increases the score, but after a certain point, further studying becomes less effective. Also, the quality of sleep might interact with the effect of studying.
- These relationships can be highly non-linear and are better captured by non-linear models like decision trees, random forests, or support vector machines with non-linear kernels.
- In this case, plotting the data points would result in a curve or a complex shape that doesn't fit a straight line or a simple plane.

Understanding whether your data is linear or non-linear is crucial for selecting the appropriate machine learning model and algorithm. Linear regression, for example, is ideal for linear datasets, while non-linear datasets require more advanced techniques to capture the underlying patterns effectively.

Support Vector Machine (SVM)

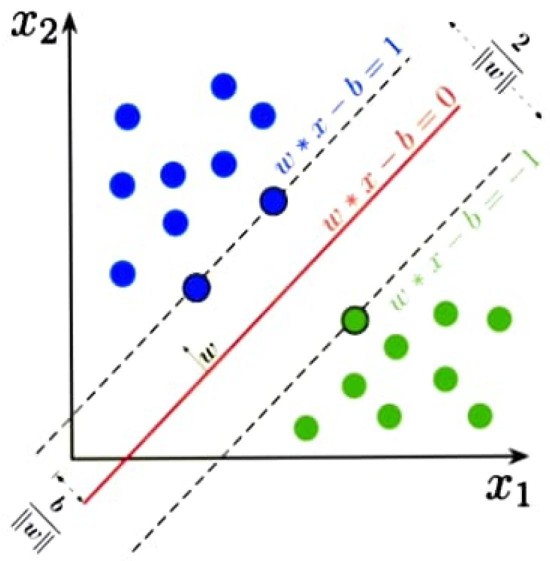
Support Vector Machine (SVM) is a powerful and versatile machine learning algorithm used for classification and regression tasks. Its primary goal is to find an optimal hyperplane that best separates different classes of data points in a high-dimensional space. SVM is particularly useful when dealing with both linearly and non-linearly separable datasets. Here's a brief introduction to SVM with relevant diagrams.

1. Intuition:

At its core, SVM aims to find a hyperplane that maximizes the margin between two classes of data points. This hyperplane is known as the decision boundary, and the margin represents the distance between the boundary and the nearest data points from each class. The idea is to ensure that the decision boundary has the greatest separation, making it robust to classify new, unseen data accurately.

2. Linear Separation

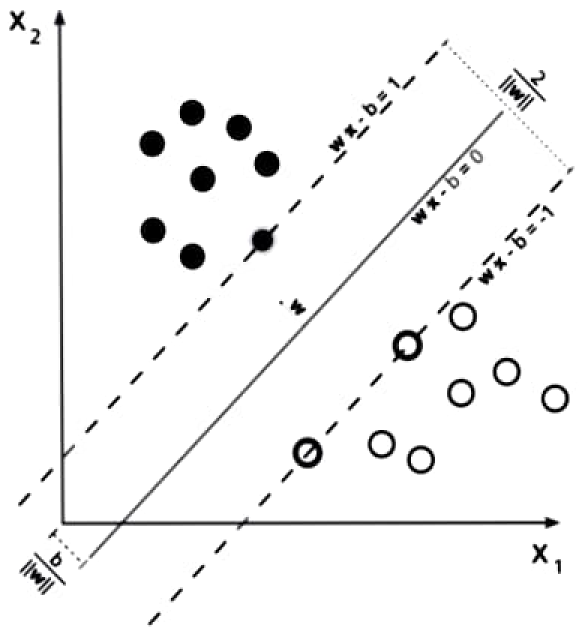
For linearly separable data, SVM finds a hyperplane that separates the two classes with the largest possible margin. In a two-dimensional feature space, this hyperplane is a straight line. Here's a simplified diagram to illustrate this concept:



The solid line is the decision boundary, and the dashed lines represent the margin.

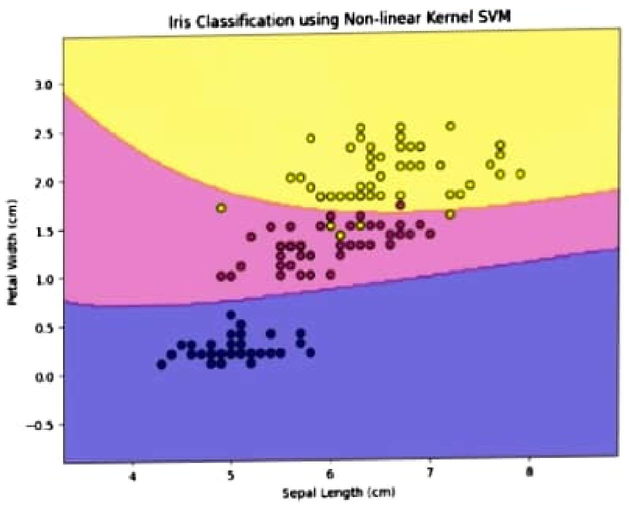
3. Support Vectors

Support vectors are the data points closest to the decision boundary. These points play a critical role in defining the margin and the overall performance of the SVM. The margin is determined by the distance between the support vectors and the decision boundary, as shown below:



4. Non-linear Separation

In many real-world scenarios, data may not be linearly separable. SVM can handle such cases by mapping the data into a higher-dimensional space where it becomes linearly separable. This transformation is done using a kernel function. Common kernel functions include the linear, polynomial, and radial basis function (RBF) kernels. The following diagram demonstrates this concept:



In this example, data is transformed into a 3D space where it becomes linearly separable.

In summary,

Support Vector Machine is a versatile machine learning algorithm that excels in both linear and non-linear classification tasks. It aims to find the optimal hyper plane that maximizes the margin between classes, and it uses support vectors to define this margin. By introducing kernel functions, SVM can handle non-linear separations, making it a valuable tool in various applications, from image classification to financial analysis.

SVMs advantages:

- They provide strong theoretical foundations, making them a preferred choice for many researchers and practitioners.
- They are effective with high-dimensional data, relatively resistant to overfitting, and can handle both binary and multiclass classification problems.
- Moreover, SVMs are also used in regression tasks, known as Support Vector Regression (SVR), where they aim to fit a hyperplane that approximates the target values as closely as possible.

Linear Discriminant functions for Binary classification using SVM:

Linear discriminant functions are essential in binary classification tasks, and Support Vector Machines (SVMs) utilize them to separate data into two distinct classes.

Let's explain this concept using simple equations and diagrams

In SVM, the linear discriminant function for binary classification can be represented as

$$f(x) = \text{sign}(w \cdot x + b)$$

Where

f(x) : The decision function that determines the class label of a data point x.

w: The weight vector that defines the orientation of the decision boundary.

x: The feature vector representing the input data point.

b: The bias or threshold value that shifts the decision boundary.

sign(.): The sign function, which assigns a class label based on the sign of the result.

Example

Let's illustrate this with a simple example. Suppose we have a 2D feature space for binary classification, where class A is represented by Blue points and class B is represented by Black points. The linear discriminant function is:

$$f(x) = \text{sign}(w_1 x_1 + w_2 x_2 + b)$$

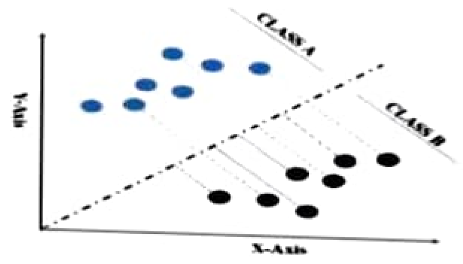
Where

w_1 and w_2 are the weights that define the orientation of the decision boundary.

x_1 and x_2 are the features of a data point.

b is the bias term.

Here's a diagram



In the diagram, the decision boundary is a straight line, represented by

$$w_1 x_1 + w_2 x_2 + b = 0$$

That separates the two classes. Any data point x falling on one side of the decision boundary is classified as Class A, and on the other side as Class B.

- SVM's objective is to find the optimal values of w and b that maximize the margin between the two classes while minimizing classification errors.
- The support vectors are the data points that are closest to the decision boundary, and they are used to define the margin in an SVM.
- The margin is the perpendicular distance from the decision boundary to the support vectors.
- SVM aims to maximize this margin while ensuring that data points are correctly classified.
- If the data points are not linearly separable, SVM can use kernel functions to map the data to a higher-dimensional space where linear separation is possible.

Large margin classifier for linearly separable data using SVM:

A key concept in Support Vector Machines (SVM) is the idea of a large margin classifier for linearly separable data. The goal is to find a hyperplane that maximizes the margin between two classes of data points. Here's an explanation with simple equations and diagrams:

- Consider a binary classification problem where you have two classes: Class A and Class B. The objective of SVM is to find a hyperplane that best separates these two classes while maximizing the margin. The equation for this hyperplane is

$$w \cdot x + b = 0$$

Where

W: The weight vector that defines the orientation of the hyperplane.

x: The feature vector representing an input data point.

b: The bias or threshold value that shifts the hyperplane.

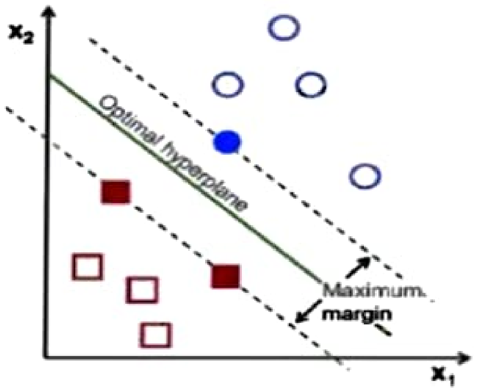
Maximizing the Margin:

- The margin is the distance between the hyperplane and the nearest data points from each class. To maximize this margin, we want to find w and b such that the distance from the hyperplane to the closest point in Class A and the closest point in Class B is maximized. Mathematically, this can be represented as:

$$\text{Margin} = \frac{2}{||w||}$$

Where ||w|| is the Euclidean norm (magnitude) of the weight vector w. The objective of SVM is to maximize this margin.

Here's diagrams illustrating this concept



In the diagram, the decision hyperplane (the straight line) separates Class A from Class B. The margin is the distance from the hyperplane to the closest data points from each class.

- SVM's objective is to find the optimal w and b that maximize this margin while ensuring that data points are correctly classified.
- In this ideal scenario of linearly separable data, the support vectors are the data points closest to the hyperplane, and they are used to define the margin.
- SVM finds these support vectors and optimizes the margin by solving a constrained optimization problem.

The large margin classifier provides a robust solution for linearly separable data, ensuring a wider separation between classes and making it less sensitive to noise in the data.

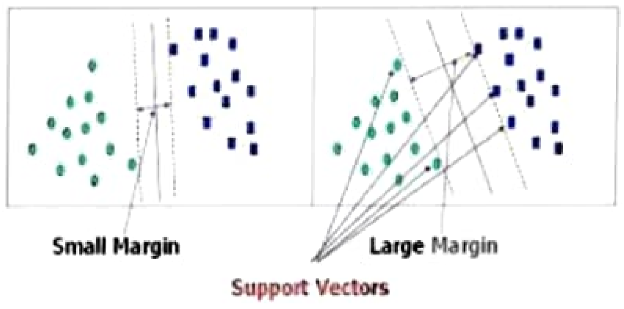
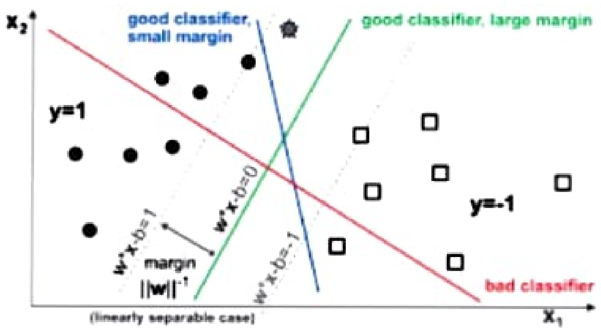


Fig: Representing small and large margin at SVM classifier model



x₂
good classifier, small margin
good classifier, large margin
y=1
y=-1
w*x-b=1
w*x-b=0
w*x-b=-1
margin
||w||⁻¹
bad classifier
(linearly separable case)
x₁

Fig: Representing good and bad SVM classifier models in small and large margin cases

A large margin classifier in SVM for linearly separable data aims to find an optimal hyperplane that maximizes the margin between two classes, ensuring a robust separation. Support vectors define this margin, and SVM finds the best hyperplane by minimizing classification errors while maximizing the margin, enhancing classification accuracy and robustness.

Linear soft margin classifier for over lapping classes

A linear soft margin classifier in SVM addresses overlapping classes by allowing for some classification errors. It introduces a penalty for misclassifications, striking a balance between maximizing the margin and tolerating some errors.

Linear Soft Margin Classifier:

- The linear soft margin classifier in SVM aims to find a hyperplane that best separates overlapping classes, even when perfect separation isn't possible. It introduces a "slack variable" (ξ) to account for classification errors. The objective function is modified as follows

$$\min_{w,b} \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n} \xi i$$

Subject to:

$$yi(w \cdot xi + b) \geq 1 - \xi i \quad ; \xi i \geq 0$$

Where

w: The weight vector.

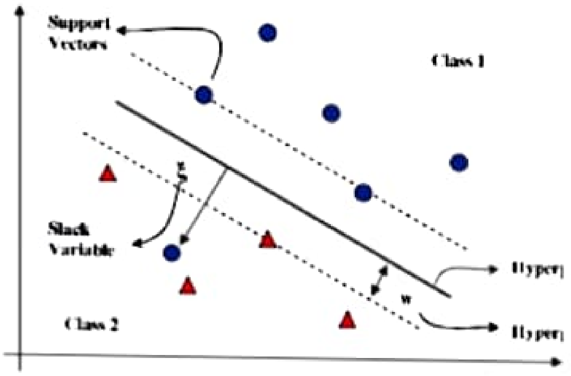
b: The bias or threshold.

C: A hyper parameter that controls the trade-off between maximizing the margin and minimizing the misclassification error.

ξ_i: Slack variable for the ith data point.

n: The number of data points.

yi: The class label of the ith data point.



A simple diagram illustrating the concept of a linear soft margin classifier:

Class 1 is represented by points labeled as -1.

Class 2 is represented by points labeled as +1.

- The decision hyperplane (a straight line) attempts to separate the classes, but due to overlapping, some data points may lie on the wrong side.
- The slack variables (ξ) allow for some misclassifications while trying maximizing the margin. The parameter C controls the balance between minimizing errors (small C) and maximizing the margin (large C).
- It helps SVM adapt to overlapping classes and create a margin that balances the trade-off between classification accuracy and margin size.

Kernel-induced feature spaces (Nonlinear classifier)

Kernel-induced feature spaces are a critical concept in Support Vector Machines (SVM) that allows SVM to handle non-linearly separable data by implicitly transforming it into a higher-dimensional space where it might become linearly separable.

- In SVM, the kernel function plays a central role. The kernel function, denoted as K(x, y), takes two input data points x and y and returns a measure of similarity between them.
- It implicitly maps the data into a higher-dimensional feature space where linear separation might be possible.

The equation for SVM's decision boundary in the feature space is:

$$f(x) = \text{sign}\left(\sum_{i=1}^{n} \alpha i\, y i\, K(x i, x) + b\right)$$

Where

$f(x)$: The decision function.

α_i: Lagrange multipliers determined during the SVM optimization.

y_i: Class labels of the data points.

$K(x_i, x)$: The kernel function that maps x_i and x into the feature space

b: The bias or threshold.

- Consider a simple 2D dataset where Class A (Green points) and Class B (blue points) are not linearly separable in the original feature space:

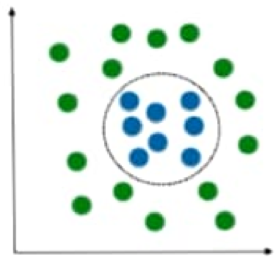


Fig: Non linear separable data using 2D Space

- In this diagram, it's evident that a straight-line decision boundary cannot separate the classes effectively in the original 2D space.
- Now, by using a kernel function, we implicitly map this data to a higher-dimensional feature space, often referred to as a "kernel-induced feature space." Let's say we use a radial basis function (RBF) kernel
- This RBF kernel implicitly maps the data to a higher-dimensional space where the classes might become linearly separable

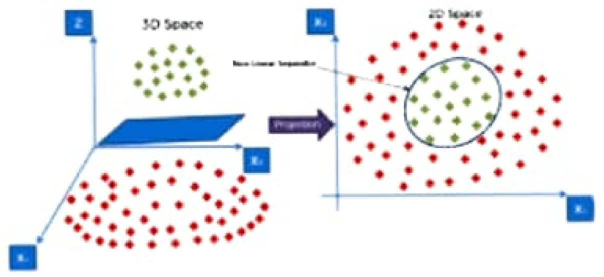


Fig: Non linear separable data using 3D Kernel Space and 2D Space

- In this new feature space, the data points might be linearly separable with the right choice of kernel and kernel parameters, enabling SVM to find an optimal decision boundary that maximizes the margin between classes.

The transformation into the kernel-induced feature space is implicit and doesn't require explicit calculation of the transformed feature vectors. It allows SVM to handle non-linearly separable data effectively.

Perceptron Algorithm:

The Perceptron algorithm is a simple supervised learning algorithm used for binary classification. It's designed to find a linear decision boundary that separates data points of two classes. Here's an explanation of the Perceptron algorithm with suitable diagrams:

The Perceptron algorithm works as follows:

1. Initialize the weights (w) and bias (b) to small random values or zeros.

2. For each data point (x) in the training dataset, compute the predicted class label ($\hat{y}$) using the following formula

$$\hat{y} = \text{sign}\,(w \cdot x + b)$$

Here, w represents the weight vector, x is the feature vector of the data point, and sign (.) is a function that returns +1 for values greater than or equal to zero and -1 for values less than zero.

3. Compare the predicted class label ($\hat{y}$) to the true class label (y) of the data point. If they don't match, update the weights and bias as follows:

$$w = w + \alpha \cdot (y - \hat{y}) \cdot x$$

$$b = b + \alpha \cdot (y - \hat{y})$$

Here, (α) is the learning rate, and $(y - \hat{y})$ is the classification error. These updates help the Perceptron adjust the decision boundary to classify the data points correctly.

4. Repeat the above steps for a fixed number of iterations or until the algorithm converges, meaning no more misclassifications occurs.

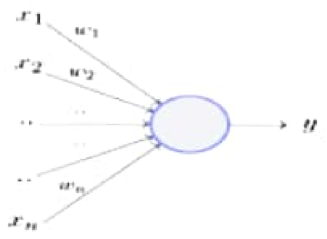


Fig: Perceptron

Let's illustrate the Perceptron algorithm with a simple 2D dataset and a linear decision boundary

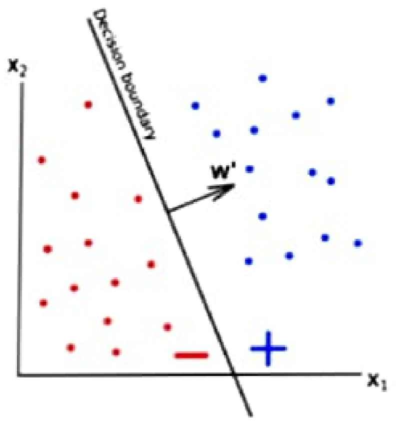


Fig: Linear decision boundary using Perceptron algorithm

- Suppose you have a 2D feature space with two classes, Class 1 (labeled as -1) and Class 2 (labeled as +1), and you want to separate them using the Perceptron algorithm.
- In the initial state, the decision boundary (a straight line) is randomly placed. The Perceptron algorithm starts making predictions and adjusting the decision boundary based on classification errors.
- As it iterates through the data points, it gradually shifts the decision boundary to correctly classify the points into their respective classes.
- The process continues until no misclassifications remain, or a maximum number of iterations are reached. The final decision boundary separates the two classes effectively
- The Perceptron algorithm finds a linear decision boundary that minimizes classification errors and correctly classifies the data points based on the training data. It's a basic algorithm suitable for linearly separable datasets and serves as the foundation for more complex neural networks.

The key difference between the Perceptron and SVM is that SVM aims to find the optimal hyperplane that maximizes the margin, whereas the Perceptron algorithm doesn't consider margin maximization. SVM is a more sophisticated and powerful classification algorithm, especially suitable for scenarios where data may not be perfectly separable and a margin is essential for generalization and reducing over fitting.

Regression by SVM (Super Vector Machine):

Support Vector Machines (SVM) are not just limited to classification tasks; they can also be used for regression. In regression tasks, the goal is to predict a continuous target variable rather than class labels. SVM for regression is known as Support Vector Regression (SVR).it is classified into two models.

1. Linear regression by SVM
2. Non-Linear Regression by SVM

Linear regression by SVM:

Linear regression using Support Vector Machines (SVM) is a variation of SVM designed for regression tasks. It aims to find a linear relationship between input features and a continuous target variable.

- In linear regression using SVM, the goal is to find a linear function that best approximates the relationship between input features and the target variable. This linear function is represented as:

$$f(x) = w \cdot x + b$$

  f(x): The predicted target variable.

  w: The weight vector.

  x: The feature vector representing the input data point.

  b : The bias or intercept term.

- The linear regression objective is to minimize the mean squared error (MSE) between the predictions and the true target values

$$\min_{w,b} \frac{1}{2}\|w\|^2 + C \sum_{i=1}^{n} (y_i - f(x_i))^2$$

  y_i: The true target variable of the ith data point.

  C: A regularization parameter controlling the trade-off between fitting the data and keeping the model simple.

- The target variable (y) is represented on the vertical axis, and the input features (x) are on the horizontal axis.
- The linear function f(x) = w.x + b is the best-fitting line that minimizes the mean squared error by adjusting the weight vector (w) and the bias term (b).

- This linear model can be used for regression tasks to predict continuous target variables based on input features.

Non-Linear Regression by SVM:

Non-linear regression by Support Vector Machines (SVM) uses the principles of SVM to model non-linear relationships between input features and a continuous target variable. The key idea is to use kernel functions to implicitly map the data into a higher-dimensional space, where a linear regression model can be applied effectively

- In non-linear regression using SVM, the goal is to find a non-linear function that best fits the relationship between input features and the target variable.
- Unlike linear regression, which assumes a linear relationship, non-linear regression allows for more complex, non-linear patterns.
- The non-linear regression objective is to minimize the mean squared error (MSE) between the predictions and the true target values:

$$\min_{w,b} \frac{1}{2}\|w\|^2 + C \sum_{i=1}^{n} (y_i - f(x_i))^2$$

y_i: The true target variable of the ith data point.

C: A regularization parameter controlling the trade-off between fitting the data and keeping the model simple.

To account for non-linearity, the non-linear function is represented as

$$f(x) = \sum_{i=1}^{n} \alpha_i K(x_i, x) + b$$

f(x): The predicted target variable.

α_i: Lagrange multipliers determined during the SVM optimization.

$K(x_i, x)$: The kernel function that implicitly maps x_i and x into a higher-dimensional feature space.

b : The bias or intercept term.

- The target variable (y) is represented on the vertical axis, and the input features (x) are on the horizontal axis.
- The non-linear function $f(x) = \sum_{i=1} \alpha_i K(x_i, x) + b$ captures non-linear relationships between input features and the target variable by implicitly mapping the data into a higher-dimensional feature space using the kernel function.
- The model can then make non-linear predictions based on the input features.