Assignment

Q1. What is the mathematical formula for a linear SVM?

Ans:In a linear Support Vector Machine (SVM), the goal is to find a hyperplane that separates the input data into different classes. The mathematical formula for a linear SVM can be expressed as follows:

Given a set of training data points

$(\blacklozenge 1,\blacklozenge 1),(\blacklozenge 2,\blacklozenge 2),...,(\blacklozenge\blacklozenge,\blacklozenge\blacklozenge)$

$(x$

1

$,y$

1

$),(x$

2

$,y$

2

$),...,(x$

n

$,y$

n

), where

$$x_i$$

is the input vector and

$$y_i$$

is the corresponding class label ($$y_i \in \{-1,1\}$$), the linear SVM aims to find a hyperplane defined by the equation:

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

Here:

- w
- **w** is the weight vector (coefficients) perpendicular to the hyperplane.

- x
- **x** is the input feature vector.
-
- b is the bias term.

The decision function is then used to classify a new data point:

$$\text{Prediction}(\text{})=\text{sign}(\text{}(\text{}))$$

$$\text{Prediction}(x)=\text{sign}(f(x))$$

The objective of the SVM is to find the

w

w and

b that maximize the margin between the two classes while minimizing the classification error.

This can be formulated as a constrained optimization problem:

$$\text{Minimize}\frac{1}{2}||\text{w}||^2\text{Subject to}\text{}(\text{w}\cdot\text{x}\text{}+\text{})\geq 1\text{for }\text{}=1,2,...,\text{}$$

Minimize

Subject to

2

1

$||\mathbf{w}||$

2

y

i

$(\mathbf{w} \cdot \mathbf{x}$

i

$+b) \geq 1 \text{for } i=1,2,...,n$

Here,

$||w||$

$||\mathbf{w}||$ represents the Euclidean norm of the weight vector. The inequality constraint ensures that each data point is on the correct side of the decision boundary with a margin of at least 1. The solution to this optimization problem provides the parameters (

w

w and

$\hat{}$

b) of the hyperplane that best separates the data.

Q2. What is the objective function of a linear SVM?
Ans:The objective function of a linear Support Vector Machine (SVM) is formulated as a convex optimization problem. The goal is to find the parameters
w

w (weight vector) and

$\hat{}$

b (bias term) that define the hyperplane, maximizing the margin between different classes while minimizing the classification error. The objective function for a linear SVM is given by:

minw,$\frac{1}{2}$||w||2

min

2

1

$||\mathbf{w}||$

2

Subject to the following constraints for each data point

(x$_i$,y_i)

($\mathbf{x}_i$,y_i):

$y_i(\mathbf{w}\cdot\mathbf{x}_i+b)\geq 1$

Here:

- $12||w||2$
  - 2
  - 1
  -
- $||\mathbf{w}||$
- 2
- is the regularization term that penalizes the magnitude of the weight vector
- w
- **w**. The factor
- 12
  - 2
  - 1
  -
- is included for mathematical convenience, as it simplifies the subsequent calculations.
- $||w||$
- $||\mathbf{w}||$ is the Euclidean norm of the weight vector.
- ��
- y
- i
-
- is the class label for the
- ﷺ
- i-th data point (
- ﷺ$\in\{-1,1\}$
- y
- i
-
- $\in\{-1,1\}).$
- xﷺ
- $\mathbf{x}$
- i
-
- is the feature vector for the
- ﷺ
- i-th data point.

- The inequality constraint
- ��(w·x�+�)≥1
- y
- i
-
- $(\mathbf{w} \cdot \mathbf{x}$
- i
-
- $+b) \geq 1$ ensures that each data point lies on the correct side of the decision boundary with a margin of at least 1.

The objective function seeks to minimize the norm of the weight vector, which corresponds to maximizing the margin between the classes. The constraints ensure that all data points are correctly classified with a margin of at least 1. The solution to this optimization problem provides the optimal parameters (

w

w and

�

b) for the linear SVM. The optimization problem is convex, allowing for efficient and reliable

solutions.

Q3. What is the kernel trick in SVM?
Ans:The kernel trick is a technique used in Support Vector Machines (SVMs) to handle non-linear decision boundaries by implicitly mapping input data into a higher-dimensional feature space. In the original SVM formulation, a linear hyperplane is used to separate data points of different classes. However, when the data is not linearly separable in the original feature space, the kernel trick allows SVMs to operate in a higher-dimensional space without explicitly computing the transformation.

The kernel trick is based on the idea of using a kernel function

�(x�,x�)

$K(\mathbf{x}$

i

,**x**

j

) that calculates the dot product of the transformed feature vectors

Φ(x�)·Φ(x�)

Φ(**x**

i

) · Φ(**x**

j

) without explicitly computing the transformation

Φ(x�)

Φ(**x**

i

) for each data point. This dot product is used in place of the linear dot product in the SVM optimization problem.

The general form of the SVM optimization problem with the kernel trick is:

Minimize12∑�=1�∑�=1�(x�,x�)−∑�=1Subject to∑�=1=0and0≤≤for =1,2,...,

Minimize

Subject to

$$\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}$$

α_i

α_j

y_i

y_j

$K(\mathbf{x}_i$

,**x**

j

)−

i=1

$\sum$

n

α

i

i=1

$\sum$

n

α

i

y

i

=0and0≤α

i

≤*C*for *i*=1,2,...,*n*

Here:

- ��
- α
- i
-
- are the Lagrange multipliers.
- ￼
- C is a regularization parameter.
- ￼￼
- y
- i
-
- is the class label for the
- ￼
- i-th data point.
- ￼(x￼,x￼)
- $K(\mathbf{x}$
- i
-
- ,$\mathbf{x}$
- j
-
-) is the kernel function.

Commonly used kernel functions include:

Linear Kernel:
￼(x￼,x￼)=x￼·x￼
$K(\mathbf{x}$
i

,$\mathbf{x}$
j

)=$\mathbf{x}$
i

· $\mathbf{x}$
j

Polynomial Kernel:

$$K(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}_i \cdot \mathbf{x}_j + c)^d$$

Radial Basis Function (RBF) or Gaussian Kernel:

$$K(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\frac{||\mathbf{x}_i - \mathbf{x}_j||^2}{2\sigma^2}\right)$$

)

Sigmoid Kernel:

$$\text{K}(\text{x}_i,\text{x}_j)=\tanh(\alpha\text{x}_i\cdot\text{x}_j+c)$$

The kernel trick allows SVMs to capture complex, non-linear decision boundaries in a more computationally efficient way by avoiding the explicit computation of the higher-dimensional feature vectors.

Q4. What is the role of support vectors in SVM Explain with example

Ans:In Support Vector Machines (SVMs), support vectors play a crucial role in defining the decision boundary. Support vectors are the data points that lie closest to the decision boundary (hyperplane) and have a non-zero contribution to the determination of the hyperplane. These are the most critical data points as they essentially "support" the decision boundary.

The role of support vectors can be explained in terms of the SVM optimization problem, where the goal is to find a hyperplane that maximally separates different classes while maintaining a margin. The decision boundary is determined by the support vectors, and these points have non-zero Lagrange multipliers (

$$\alpha_i$$

) in the SVM optimization problem.

Here's a step-by-step explanation with an example:

Hyperplane Definition:
In a linear SVM, the decision boundary is defined by the equation:

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

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

The support vectors are the data points

$$\mathbf{x}_i$$

for which

$$y_i(w\cdot x_i+b)=1$$

$$(\mathbf{w}\cdot\mathbf{x}_i+b)=1$$

, and they lie on the margin.

Margin Maximization:

The SVM optimization problem aims to maximize the margin between the classes, subject to the constraint that all data points are correctly classified. The support vectors are the data points that define the margin, and their presence ensures that the margin is maximized.

Contribution to Decision Boundary:

Support vectors are the data points that have a non-zero value for the Lagrange multiplier (

$$\alpha_i$$

α
i

) in the optimization problem. These non-zero
α
i

contribute to the determination of the hyperplane. The decision boundary is essentially a weighted combination of these support vectors.

Example:

Consider a binary classification problem with two classes, +1 and -1. The decision boundary is determined by a hyperplane defined by
$w \cdot x+b=0$
$\mathbf{w} \cdot \mathbf{x}+b=0$. The support vectors are the data points closest to this hyperplane. The importance of these support vectors is demonstrated in the fact that changing or removing other data points (non-support vectors) would not alter the position of the hyperplane.
In the example above, the filled circles and squares represent support vectors, and the decision boundary is determined by these points. Changing the position of non-support vectors would not affect the decision boundary, but altering the position of support vectors would have a significant impact.

In summary, support vectors are critical for the definition of the decision boundary in SVM, and they play a central role in maximizing the margin between classes while ensuring correct classification.

Q5. Illustrate with examples and graphs of Hyperplane, Marginal plane, Soft margin and Hard margin in
SVM?
Ans:illustrate the concepts of Hyperplane, Marginal Plane, Soft Margin, and Hard Margin in Support Vector Machines (SVM).

1. Hyperplane:

The hyperplane is the decision boundary that separates different classes in an SVM. In a 2D space, it's a line; in a 3D space, it's a plane; and in higher dimensions, it's a hyperplane.

Example:

Consider a simple 2D dataset with two classes (blue and orange). The hyperplane is the line that separates these classes.

2. Marginal Plane:

The marginal plane is parallel to the hyperplane and defined by the support vectors. It plays a role in determining the margin (distance between the hyperplane and the nearest data point of any class).

Example:

Using the same 2D dataset, the marginal plane is defined by the support vectors (filled circles). The margin is the distance between the hyperplane and this marginal plane.

3. Soft Margin:

In the Soft Margin SVM, the algorithm allows some data points to be on the wrong side of the margin or even the wrong side of the hyperplane. The objective is to find a balance between maximizing the margin and minimizing the classification error.

Example:

Consider a dataset with outliers. A soft margin SVM allows for a certain degree of misclassification or points within the margin.

4. Hard Margin:

In contrast, a Hard Margin SVM requires that all data points are correctly classified, and there is no allowance for misclassification. It is more sensitive to outliers and works well when the data is perfectly separable.

Example:

For a perfectly separable dataset, a hard margin SVM enforces a strict margin with no misclassified points.

In these examples, the blue and orange points represent two different classes, and the filled circles indicate support vectors. The lines/plane represent the hyperplane, and the dotted lines/planes illustrate the margin or the allowed space between classes.

Remember that the choice between soft and hard margin depends on the nature of the data. Soft margin SVMs are more robust to outliers and work well when the data is not perfectly separable, while hard margin SVMs are suitable for well-behaved, separable datasets.

Q6. SVM Implementation through Iris dataset.

Bonus task: Implement a linear SVM classifier from scratch using Python and compare its performance with the scikit-learn implementation.
~ Load the iris dataset from the scikit-learn library and split it into a training set and a testing setl
~ Train a linear SVM classifier on the training set and predict the labels for the testing setl
~ Compute the accuracy of the model on the testing setl
~ Plot the decision boundaries of the trained model using two of the featuresl
~ Try different values of the regularisation parameter C and see how it affects the performance of
the model.
implement a linear SVM classifier on the Iris dataset using both a custom implementation and scikit-learn. We'll also compare the performance by varying the regularization parameter

C.

python

Copy code

```
import       as
import                  as
from         import
from                        import
from                      import
from               import
```

```
from             import


class LinearSVM
 def __init__                    0.01         1000    1



 def fit


             0

 for       in range
 for   in range
                                                          1
 if
                                      2
 else
                                      2



 def predict
 return


                 2
                   0    2   1


                                                              0.2
             42


                        1
```

```
                        'linear'    1
```

```
print f"Accuracy (Custom SVM): {accuracy_custom}"
print f"Accuracy (Scikit-learn SVM): {accuracy_sklearn}"


def plot_decision_boundary
     .02
                    0  min     1       0  max     1
                    1  min     1       1  max     1
```

```
                                                 0.3
                0       1                                    'k'

            'Feature 1'
            'Feature 2'
```

```
                                                      'Decision Boundary
(Custom SVM)'
                                                       'Decision Boundary
(Scikit-learn SVM)'
```

In this example, we load the Iris dataset, split it into training and testing sets, standardize the features, and then train a linear SVM using both a custom implementation and scikit-learn. We also compare the accuracies and visualize the decision boundaries for both models. Finally, you can experiment with different values of the regularization parameter

C to observe how it affects the performance of the model.