Sharif University of Technology
Electrical Engineering Department

Machine Learning and Vision Lab
Pre-Report 4

Amir Hossein Yari
99102507

November 11, 2023

No, the data is not linearly separable. It's not possible to draw a single straight line that perfectly separates Class1 from Class-1.

$x^1 = (1, 1, 1, 1) \quad x^2 = (1, -1, -1, 1) \quad x^3 = (1, 1, -1, -1) \quad x^4 = (1, -1, 1, -1)$

We know that $y^i(W^T\phi(x^i)) \geq 1$

$$\Rightarrow \begin{bmatrix} w_1 & w_2 & w_3 & w4 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} \geq 1 \Rightarrow w_1 + w_2 + w_3 + w_4 \geq 1$$

$$\Rightarrow \begin{bmatrix} w_1 & w_2 & w_3 & w4 \end{bmatrix} \begin{bmatrix} 1 \\ -1 \\ -1 \\ 1 \end{bmatrix} \geq 1 \Rightarrow w_1 - w_2 - w_3 + w_4 \geq 1$$

$$\Rightarrow -1 \times \begin{bmatrix} w_1 & w_2 & w_3 & w4 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ -1 \\ -1 \end{bmatrix} \geq 1 \Rightarrow -w_1 - w_2 + w_3 + w_4 \geq 1$$

$$\Rightarrow - \begin{bmatrix} w_1 & w_2 & w_3 & w4 \end{bmatrix} \begin{bmatrix} 1 \\ -1 \\ 1 \\ -1 \end{bmatrix} \geq 1 \Rightarrow -w_1 + w_2 - w_3 + w_4 \geq 1$$

By solve optimization problem we find $W = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$

$K(x, x') = \phi(x)\phi(x') = 1 + x_1x_1' + x_2x_2' + x_1x_2x_1'x_2'$

The kernel function $K(x, x')$ can capture more complex relationships between data points, including non-linear patterns that may not be evident in the original feature space. In this higher-dimensional space, a linear decision boundary can be found that separates the classes effectively.

To solve above optimization problem we can use python codes below to find w.

```python
from sklearn import svm
import numpy as np

# Define your data in the new feature space
X = np.array([[1, 1, 1, 1], [1, -1, -1, 1], [1, 1, -1, -1], [1, -1, 1, -1]])

# Define the corresponding class labels
y = np.array([1, 1, -1, -1])

# Create an SVM classifier with a linear kernel
clf = svm.SVC(kernel='linear')

# Train the SVM classifier
clf.fit(X, y)

# The weight vector w is stored in the coef_ attribute of the classifier
w = clf.coef_

print("Weight vector w:", w)
```

```python
import cvxpy as cp
import numpy as np

# Define your data in the new feature space
X = np.array([[1, 1, 1, 1], [1, -1, -1, 1], [1, 1, -1, -1], [1, -1, 1, -1]])

# Define the corresponding class labels
y = np.array([1, 1, -1, -1])

# Number of samples and features
n_samples, n_features = X.shape

# Define the weight vector w as a variable
w = cp.Variable(n_features)


# Formulate the SVM optimization problem
objective = cp.Minimize(0.5 * cp.norm(w, 2) )
constraints = [cp.multiply(y, X @ w) >= 1]

# Solve the optimization problem
prob = cp.Problem(objective, constraints)
prob.solve()

# Get the optimized weight vector w
w_optimized = w.value

# Get the optimized weight vector w
w_optimized = np.round(w.value, decimals=2)
```

```

    print("Optimized weight vector w:", w_optimized)
```