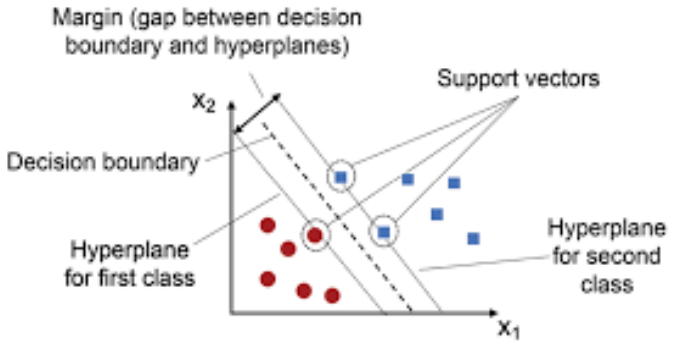
FAST NATIONAL UNIVERSITY CFD CAMPUS

Name:	**Mariam Fatima** **Muskan Ghani**
Roll No:	**22F-3168** **22F-3841**
Assignment:	**5**
Instructor:	**Dr. Hashim Yaseen**
Course:	**Machine Learning**

Support Vector Machine (SVM)

What is a Support Vector?

A support Vector in SVM is a data point that lies closest to the decision boundary (or hyperplane). These points are critical as they define the margin of the classifier. Removing a support vector can change the position of the hyperplane, making them essential for the SVM's construction.



Drive objective function of SVM for linearly separable data.

The objective of SVM is to find the hyperplane that maximizes the margin, which is the distance between the hyperplane and the closest data points (support vectors) from either class.

The equation of hyperplane is:

$$w^T x + b = 0,$$

Where **w** is the weight, **x** is the feature vector and **b** is the bias.

Constraints for correct classification are:

$$y_i(w^T x_i + b) \geq 1$$

Objective Function:

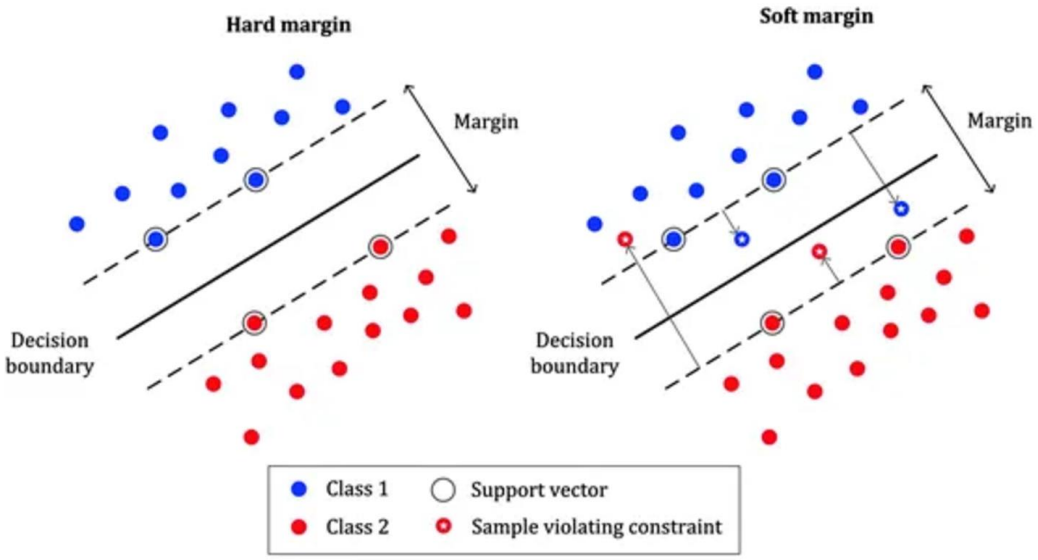
$$\text{Minimize: } \frac{1}{2}\|w\|^2,$$

$$\text{Subject to: } y_i(w^T x_i + b) \geq 1.$$

Hard Margin vs Soft Margin

Hard Margin SVM assumes that the data is perfectly separable. It tries to find the widest possible margin (distance between the classes) without allowing any misclassified points. This approach works only when there is no noise or overlap in data.

Soft Margin SVM allows some points to be misclassified or lie within the margin. This approach introduces flexibility, making it suitable for noisy or overlapping datasets by balancing margin maximization and classification error.



Problem:

Hard Margin - overfitting.

Soft Margin - underfitting.

Dataset Overview

The MNIST dataset contains **60,000** training images and **10,000** test images, each of size **28×28** pixels. The task is to classify each image into one of 10 categories (digits **0** to **9**).

Key Features:

- **Input size**: 784 features (flattened for SVM and ANN).
- **Output**: 10 classes (digits 0 to 9).

- **Preprocessing**:
    - SVM: Standardized features using **StandardScaler**.
    - ANN & CNN: Normalized pixel intensities to the range [0,1].

Support Vector Machine (SVM) Implementation

Two types of SVM classifiers were implemented:

1. **Linear SVM**: Uses a linear kernel to classify data by a hyperplane.
2. **Non-linear SVM (RBF Kernel)**: Employs a Radial Basis Function (Gaussian Kernel) to map data to a higher-dimensional space.

Implementation Steps:

1. Load MNIST data using **idx2numpy**.
2. Preprocess the data:
    - Flatten images to **28×28=784**.
    - Standardize features using **StandardScaler**.
3. Train SVM classifiers using SVC with the following settings:
    - Linear Kernel.
    - RBF Kernel (default parameters).

Results:

Training Accuracy:

```
Linear SVM Training Accuracy: 98.21%
```

```
RBF Kernel SVM Training Accuracy: 98.66%
```

Test Accuracy:

```
Linear SVM Test Accuracy: 92.93%
RBF Kernel SVM Test Accuracy: 96.60%
```

Analysis:

- The linear SVM achieved a high training accuracy of **98.21%** but performed relatively worse on the test set **(92.93%)** indicating overfitting or limited capacity for non-linear data patterns.
- The RBF Kernel SVM outperformed the linear SVM on the test set **(96.60%)** showcasing its ability to handle non-linear separations effectively.

Artificial Neural Network (ANN)

Architecture:

Input layer: 784 neurons (flattened image pixels).

- Hidden layers:
    - First hidden layer: 128 neurons, ReLU activation.
    - Second hidden layer: 64 neurons, ReLU activation.
- Output layer: 10 neurons, softmax activation (probabilities for each class).

Implementation Steps:

1. Normalize pixel values to the range [0,1]
2. Define the architecture using Sequential API in TensorFlow.
3. Train the model for 10 epochs with a batch size of 128.

Results:

```
ANN Test Accuracy: 97.62%
```

The **ANN** achieved a test accuracy of **97.62%,** demonstrating its capability to learn complex features.

Convolutional Neural Network (CNN)

Architecture:

- Input layer: **28×28×1** (preserves the image's spatial structure).
- Convolutional layers:
    - First layer: 32 filters, **3×3** kernel, ReLU activation, max pooling.
    - Second layer: 64 filters, **3×3** kernel, ReLU activation, max pooling.
- Dropout: 50%50\% to reduce overfitting.
- Fully connected layers:
    - Dense layer: 128 neurons, ReLU activation.
    - Output layer: 10 neurons, softmax activation.

Implementation Steps:

1. Normalize pixel values to the range [0,1].
2. Use Conv2D, MaxPooling2D, Flatten, and Dropout layers in the architecture.
3. Train the model for 10 epochs with a batch size of 128.

Results:

```
CNN Test Accuracy: 99.25%
```

The **CNN** achieved the highest test accuracy of **99.25%**, confirming its effectiveness in image recognition tasks.

Discussion

- **Linear SVM**: Simple and efficient for linearly separable data but struggles with non-linear patterns.
- **RBF Kernel SVM**: Effective for non-linear data, offering a significant improvement over the linear SVM on test accuracy.
- **ANN**: Performs well for general classification tasks but lacks the spatial-awareness capabilities of CNNs.
- **CNN**: Achieves high performance by leveraging convolutional and pooling layers to extract hierarchical features.

Conclusion

This highlights the strengths and limitations of different models:

- **SVMs** are suitable for simpler tasks and smaller datasets but may underperform on image data compared to deep learning models.
- **ANNs** offer robust performance but cannot leverage spatial structures.
- **CNNs** excel in image classification, achieving **99.25%** test accuracy on MNIST.

Q no 3. ANN

Architecture

- **Single Hidden Layer**: Boolean functions are simple and can be represented with **one** hidden layer, avoiding unnecessary complexity.
- **4 Neurons in Hidden Layer**: Enough to capture patterns without overfitting, balancing simplicity and functionality.
- **Sigmoid Activation**: Sigmoid is used because it outputs values between 0 and 1, which is perfect for binary classification. It's easy to train but can face issues in deeper networks, though that's not a concern here.
- **Biases**: Including biases allows the model to better fit data, even when inputs are zero, making it more flexible.

Different Activation Functions

1. **ReLU:**

   Prevents vanishing gradients.

```
Testing with relu in hidden and sigmoid in output:
Accuracy: 100.0%
```

2. **Tanh:**

   Zero-centered, good for balanced updates.

```
Testing with tanh in hidden and sigmoid in output:
Accuracy: 93.33333333333333%
```

3. **Sigmoid:**

   Ideal for binary problems.

```
Testing with sigmoid in hidden and sigmoid in output:
Accuracy: 93.33333333333333%
```

4. **Softmax:**

   Unnecessary here as our data is **Binary Classification Problem** but usable for comparison and **Multiple Classification Problems**.

1. **Results**: ReLU, and Tanh activation functions in the hidden layer and sigmoid output layer produces **100**% accuracy in this specific dataset, while using Sigmoid in both layers yields slightly lower accuracy (**93**.33%).