Support Vector Machines (SVM) Applied to MNIST Dataset

1. Introduction

Support Vector Machines (SVM) are a class of supervised learning algorithms designed for classification and regression tasks. Introduced by Vladimir Vapnik in the 1990s, SVM has become one of the most robust and widely used machine learning models due to its effectiveness in high-dimensional spaces and its ability to handle linear and nonlinear decision boundaries. SVM constructs a hyperplane or set of hyperplanes in a high-dimensional space to classify data points. The goal is to find the hyperplane that best separates data points of different classes while maximizing the margin, the distance between the hyperplane and the closest data points (called support vectors).
SVM is primarily designed for:

- **Binary Classification:** Separating data points into two classes by finding the optimal hyperplane.
- **Multiclass Classification:** Using strategies such as one-vs-one or one-vs-rest to handle more than two classes.
- **Regression Tasks:** Through a variation known as Support Vector Regression (SVR).

2. Mathematical formulation

Linear SVM

For a linearly separable dataset, SVM finds a hyperplane

$$\mathbf{w} \cdot \mathbf{x} + b = 0$$

that separates the two classes $y_i \in \{-1, 1\}$. The objective is to maximize the margin:

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

This is equivalent to solving the following optimization problem:

$$\min_{\mathbf{w},b} \frac{1}{2}\|\mathbf{w}\|^2$$

subject to the constraints:

$$y_i(\mathbf{w} \cdot \mathbf{x}_i + b) \geq 1, \quad \forall i.$$

Nonlinearly Separable Data

For data that are not linearly separable, SVM introduces slack variables ξ_i to allow misclassifications. The optimization problem becomes:

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

subject to:

$$y_i(\mathbf{w} \cdot \mathbf{x}_i + b) \geq 1 - \xi_i, \quad \xi_i \geq 0, \quad \forall i.$$

Here, C is a regularization parameter that controls the trade-off between maximizing the margin and minimizing the misclassification penalty.

Kernel Trick

The kernel trick allows SVM to efficiently operate in high-dimensional spaces without directly computing the transformation, making it highly effective for complex, non-linear decision boundaries. When the data is not linearly separable in its original feature space, Support Vector Machines (SVM) employ the *kernel trick* to transform the data into a higher-dimensional space where linear separation may be possible. This is achieved using a feature map $\phi(\mathbf{x})$ that maps the original features into a higher-dimensional space.
Instead of explicitly computing the feature map $\phi(\mathbf{x})$, SVM uses a kernel function $K(\mathbf{x}_i, \mathbf{x}_j)$ that computes the inner product in the transformed space:

$$K(\mathbf{x}_i, \mathbf{x}_j) = \phi(\mathbf{x}_i)^\top \phi(\mathbf{x}_j).$$

Using this kernel function, the dual formulation of SVM can be expressed entirely in terms of $K(\mathbf{x}_i, \mathbf{x}_j)$, avoiding the explicit computation of $\phi(\mathbf{x})$. This enables SVM to handle very high-dimensional or even infinite-dimensional feature spaces efficiently.
Commonly used kernel functions include:

- **Linear Kernel**:
$$K(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i^\top \mathbf{x}_j.$$
- **Polynomial Kernel**:
$$K(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}_i^\top \mathbf{x}_j + c)^d,$$
where $c \geq 0$ and d is the degree of the polynomial.
- **Radial Basis Function (RBF) Kernel**:
$$K(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2\sigma^2}\right),$$
where $\sigma > 0$ controls the width of the Gaussian kernel.
- **Sigmoid Kernel**:
$$K(\mathbf{x}_i, \mathbf{x}_j) = \tanh(\alpha \mathbf{x}_i^\top \mathbf{x}_j + c),$$
where $\alpha > 0$ and c are kernel parameters.

3. SVM and MNIST dataset

The MNIST dataset, comprising grayscale images of handwritten digits, is a standard benchmark for classification algorithms. Given its high-dimensional input space ($28x28$ pixels) and non-linear class boundaries, SVM is a compelling choice for this task due to its margin-maximizing property and kernel flexibility.
In this section we performe the implementation of SVM for the classification of the MNIST dataset. Utilizing the `scikit-learn` library, we preprocess data, train an SVM classifier, and evaluate its performance using key metrics such as accuracy and a confusion matrix. The implementation highlights the flexibility of SVM in handling high-dimensional data and non-linear class boundaries, making it suitable for tasks like handwritten digit recognition.

Why SVM is Suitable for MNIST Classification

- **Accuracy:** SVM provides state-of-the-art performance for many classification problems, including MNIST, especially with a well-chosen kernel.
- **High Dimensionality:** SVM performs well in high-dimensional spaces like MNIST's flattened images (784 dimensions).
- **Clear Class Boundaries:** MNIST has reasonably distinct class boundaries that can be effectively modeled using SVM with a non-linear kernel like RBF.
- **Limited Data:** SVM is effective even with relatively small datasets, which is advantageous for MNIST subsets.
- **Kernel Flexibility:** The RBF kernel captures the non-linear relationships in MNIST digit shapes, ensuring accurate class separation.

4. Methodology

Dataset Preparation

The MNIST dataset was accessed using `torchvision.datasets.MNIST`. Images were resized to 128×128 and normalized to the range $[-1, 1]$ to standardize input values. Each image was then flattened into a 1D feature vector ($128 \times 128 = 16,384$ dimensions).Next, the dataset was divided into training and testing subsets using PyTorch's `DataLoader`. The training set was used to optimize the SVM classifier, while the testing set was reserved for performance evaluation.

SVM Model Specification

An SVM classifier was instantiated using the `SVC` class from `scikit-learn` with the following parameters:

- **Kernel**: Radial Basis Function (RBF) for capturing non-linear relationships.
- **C**: Regularization parameter to balance margin maximization and misclassification penalties.
- **Gamma**: Kernel coefficient to control the influence of individual training samples.

The hyperparameters were set as follows:

```
from sklearn.svm import SVC
model = SVC(kernel='rbf', C=1.0, gamma=0.05)
```

Training

The SVM model was trained on the flattened MNIST training data. The `fit` method optimized the model's decision boundary based on the training samples:

$$f(\mathbf{x}) = \text{sign}\left(\sum_{i=1}^{n} \alpha_i y_i K(\mathbf{x}_i, \mathbf{x}) + b\right),$$

where α_i are Lagrange multipliers, $K(\cdot, \cdot)$ is the RBF kernel, and b is the bias term.

Predictions

The trained SVM model was utilized to predict the labels of unseen test data using the `predict` method from `scikit-learn`. Each test sample, represented as a flattened vector of pixel intensities, was classified into one of ten digit classes. The decision rule for predictions is expressed as:

$$\hat{y} = \text{sign}\left(\sum_{i=1}^{n} \alpha_i y_i K(\mathbf{x}_i, \mathbf{x}) + b\right),$$

where α_i are the learned Lagrange multipliers, y_i are the true labels, $K(\cdot, \cdot)$ is the kernel function (RBF in this case), and b is the bias term.

Performance Metrics

The performance of the model was evaluated using the following metrics:

Confusion Matrix: A confusion matrix was constructed to provide a detailed breakdown of the classifier's performance across all digit classes. Each entry (i, j) in the matrix represents the number of samples from class i that were predicted as class j. The diagonal entries indicate correct classifications, while off-diagonal entries correspond to misclassifications. The confusion matrix is defined as:

$$\text{Confusion Matrix} = \begin{bmatrix} \text{TP}_{00} & \text{FP}_{01} & \dots & \text{FP}_{0n} \\ \text{FN}_{10} & \text{TP}_{11} & \dots & \text{FP}_{1n} \\ \vdots & \vdots & \ddots & \vdots \\ \text{FN}_{n0} & \text{FN}_{n1} & \dots & \text{TP}_{nn} \end{bmatrix},$$

where TP, FP, and FN represent true positives, false positives, and false negatives, respectively.

Accuracy: Accuracy was computed to quantify the proportion of correctly classified samples across all test samples. It is given by:

$$\text{Accuracy} = \frac{\text{Number of Correct Predictions}}{\text{Total Number of Predictions}} = \frac{\sum_{i=1}^{n} \text{TP}_{ii}}{\sum_{i=1}^{n} \sum_{j=1}^{n} \text{CM}_{ij}},$$

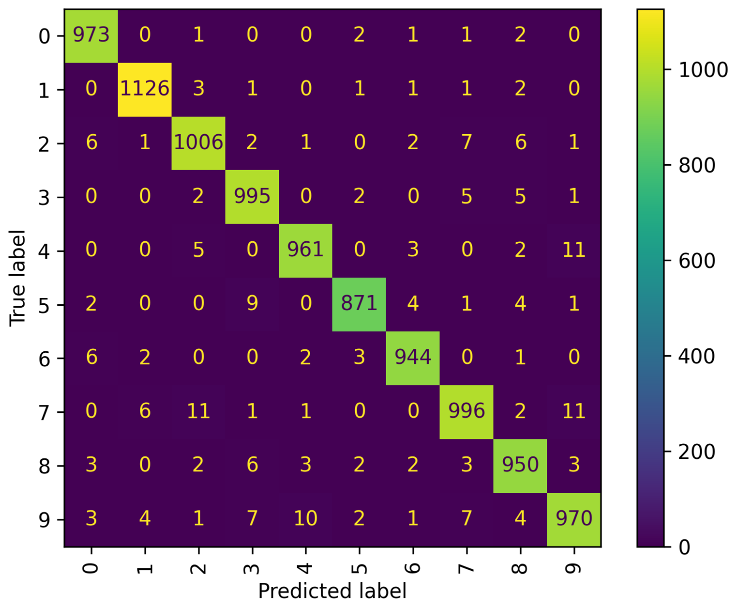
where TP_{ii} are the diagonal elements of the confusion matrix, and CM_{ij} are the matrix elements.
The confusion matrix was visualized using a heatmap to highlight the distribution of predictions. This visualization allowed for quick identification of misclassification patterns and helped in interpreting the performance of the model across digit classes.

5. Results

The SVM model achieved an accuracy of approximately **97.92%**, this high accuracy suggests that the model is proficient distinguishing between the various digits in the MNIST dataset. The confusion matrix in Figure (1) provides a detailed description of the SVM performance.
The diagonal elements of the confusion matrix, representing the number of correctly classified samples for each digit, reveal that the model performs with high accuracy across all classes. Specifically, the model achieved the following correct classifications:

- 973 instances of digit '0' were correctly identified as '0'
- 1126 instances of digit '1' were accurately classified.
- 1006 instances of digit '2' were correctly predicted, and so on.

Despite the overall high accuracy, a few misclassifications are observed. These errors are typically concentrated between visually similar digits, which is a common challenge in image classification tasks.



True label \ Predicted label	0	1	2	3	4	5	6	7	8	9
0	973	0	1	0	0	2	1	1	2	0
1	0	1126	3	1	0	1	1	1	2	0
2	6	1	1006	2	1	0	2	7	6	1
3	0	0	2	995	0	2	0	5	5	1
4	0	0	5	0	961	0	3	0	2	11
5	2	0	0	9	0	871	4	1	4	1
6	6	2	0	0	2	3	944	0	1	0
7	0	6	11	1	1	0	0	996	2	11
8	3	0	2	6	3	2	2	3	950	3
9	3	4	1	7	10	2	1	7	4	970

Figure 1: Confusion matrix for the MNIST test set using SVM

6. Conclusion

Support Vector Machines, implemented using `scikit-learn`, demonstrated strong performance on the MNIST classification task. By leveraging the RBF kernel and proper regularization, the model effectively classified high-dimensional, nonlinearly separable data. Despite its computational demands, SVM remains a powerful tool for tasks requiring high precision and robust generalization.

References

1. Cortes, C., & Vapnik, V. (1995). Support-vector networks. *Machine Learning*, 20(3), 273-297.

2. LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11), 2278-2324.

3. Pedregosa, F., Varoquaux, G., Gramfort, A., et al. (2011). Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, 12, 2011-2825.