Handwritten Digit Recognition using Support Vector Machines

1. Introduction:

Handwritten digit recognition is a classic problem in the field of machine learning and computer vision. The goal of this project is to develop a system that can accurately classify handwritten digits (0-9) based on images of handwritten digits.

In this project, we aim to develop a system for handwritten digit recognition using Support Vector Machine (SVM), a powerful supervised learning algorithm known for its effectiveness in classification tasks. SVM works by finding the optimal hyperplane that best separates the data into different classes while maximizing the margin between the classes.

2.Dataset:

The MNIST dataset is a widely used benchmark dataset in the field of machine learning and computer vision. It consists of a large collection of 28x28 pixel grayscale images of handwritten digits (0-9), along with their corresponding labels indicating the digit they represent.

The dataset consists of two files:

1. mnist_train.csv
2. mnist_test.csv

The mnist_train.csv file contains the 60,000 training examples and labels. The mnist_test.csv contains 10,000 test examples and labels. Each row consists of 785 values: the first value is the label (a number from 0 to 9) and the remaining 784 values are the pixel values (a number from 0 to 255).

Link: https://www.kaggle.com/datasets/oddrationale/mnist-in-csv

3. Data Preprocessing:

Before scaling the dataset , the dataset is viewed and many insights regarding the dataset were obtained.

i) The dataset is described in such a way that each features mean and standard deviation is calculated and it also shown us the min and max element in each feature.
ii) No of null values(missing values) in each column is calculated and viewed.
iii) No of samples respective to each label is viewed in both training and test dataset.
iv) The frequency of each pixel value is shown in a plot.
v) How many samples of each label are present in a particular pixel for some pixels are shown
vi) Some digits are visualized.

Since the average of different features varied from 140 to 0 the dataset is scaled using standard scalar.

4. Model Training:

Intially 3 different kinds of SVM models are initialized and trained and compared to see which model gave better accuracy on the test dataset. The accuracy, confusion matrix, precision, recall, and F1-score were computed to evaluate the performance of each model on the testing dataset.

The three models are:

i) **Linear SVM (Support Vector Machine):** Linear SVM is a variant of the Support Vector Machine algorithm that aims to find the optimal hyperplane separating different classes in the feature space. It utilizes a linear kernel function to compute the dot product between feature vectors, resulting in a linear decision boundary. Linear SVM is particularly suitable for datasets with linearly separable classes or when the number of features exceeds the number of samples. Due to its computational efficiency and interpretability, Linear SVM is widely used for binary classification tasks where the relationship between features and class labels is primarily linear. The decision boundary produced by Linear SVM is straightforward to interpret

and visualize, making it beneficial for understanding the underlying patterns in the data.

ii) **Polynomial SVM (Support Vector Machine):** Polynomial SVM extends the capabilities of the SVM algorithm by employing a polynomial kernel function to map the input space into a higher-dimensional feature space. Unlike Linear SVM, Polynomial SVM can capture non-linear relationships between features and class labels, resulting in a more flexible decision boundary. By introducing polynomial terms, Polynomial SVM can model complex patterns in the data, making it suitable for datasets with non-linear decision boundaries. However, the complexity of Polynomial SVM increases with the degree of the polynomial kernel, leading to higher computational costs and potential overfitting. Nevertheless, Polynomial SVM remains a valuable tool for classification tasks requiring the discrimination of non-linearly separable classes.

iii) **RBF SVM (Radial Basis Function SVM):** RBF SVM is another variant of the Support Vector Machine algorithm that utilizes a radial basis function (RBF) kernel to map the input space into an infinite-dimensional feature space. This allows RBF SVM to capture complex and non-linear relationships between features and class labels, resulting in highly flexible decision boundaries. The Gaussian nature of the RBF kernel enables RBF SVM to adapt to intricate patterns in the data, making it suitable for a wide range of classification tasks, including those with overlapping classes and non-linear decision boundaries. However, the interpretability of RBF SVM may be limited due to the complexity of its decision boundary, especially in high-dimensional feature spaces or with large datasets. Nonetheless, RBF SVM remains a powerful and versatile algorithm for classification problems requiring robust generalization capabilities.

5. Hyperparameter Tuning:

Since the best model turned out to be RBF SVM the hyper parameter tuning is done on RBF SVM. A grid search technique was used to find the optimal hyperparameters for the RBF SVM model. The hyperparameters tuned were the regularization parameter C and the kernel coefficient gamma. K-fold Cross-validation was used to assess the performance of different hyperparameter combinations.

The best hyperparameters turned out to be C=10 and Gamma=0.001.

6. Final Model building and Evaluation:

With the best hyperparameters once again a new RBF SVM model is built and trained and tested with the test dataset. The accuracy, confusion matrix, and class-wise precision, recall, and F1-score were computed to assess the model's performance.

7. Interface Development:

In addition to training and evaluating the SVM models, an interactive interface was developed to provide users with the ability to draw handwritten digits and have them classified by the trained model in real-time. The interface was built using the Tkinter library in Python and allowed users to draw digits using a mouse or touchpad. Once a digit was drawn, it was processed, and the predicted label was displayed on the interface.

Interface Features:

- **Canvas:** The main window of the interface contains a canvas where users can draw handwritten digits using their mouse cursor.
- **Drawing Functionality:** Users can draw digits by clicking and dragging their mouse on the canvas. The drawing functionality is activated upon mouse click and deactivated upon mouse release.
- **Prediction Display:** Upon releasing the mouse, the interface processes the drawn digit and provides a real-time prediction of the digit using the trained SVM model. The predicted digit label is displayed on the canvas.

8. Results:

The accuracy of the three models are as below:

Linear SVM : 92.8%

Non-Linear Polynomial SVM : 96.11%

Non-Linear RBF SVM : 96.56%

The final RBF SVM model achieved an accuracy of approximately 97.24% on the testing dataset. The confusion matrix and class-wise metrics provided insights into the model's performance for each digit class.

9.Conclusion:

In conclusion, the SVM model trained in this project demonstrates the effectiveness of using SVM for handwritten digit recognition tasks. The model achieves high accuracy and can accurately classify handwritten digits from the MNIST dataset. Further improvements could be made by exploring other feature extraction techniques or using deep learning models such as convolutional neural networks (CNNs). Overall, the project contributes to the field of image processing and pattern recognition by providing a reliable solution for handwritten digit recognition.