Homework Assignment #3: Support Vector Machine (SVM)

Vikrant Bhati

Introduction

For the given assignment we wanted to classify the data, for this, I used a support vector machine (SVM), which is a supervised machine learning algorithm that classifies the data. The support vector machines can be used for classification problems as well as regression problems over linear and non-linear data types. However, for most of the part, I will talk about linearly separable binary classification unless stated.

A support vector machine (SVM) algorithm classifies the data by finding an optimal line or hyperplane that maximizes the distance between the classes. SVMs don't use any hyperplane that divides the datasets but the one that has the maximum distance between itself and the two data points. This place is called the maximum-margin hyperplane.

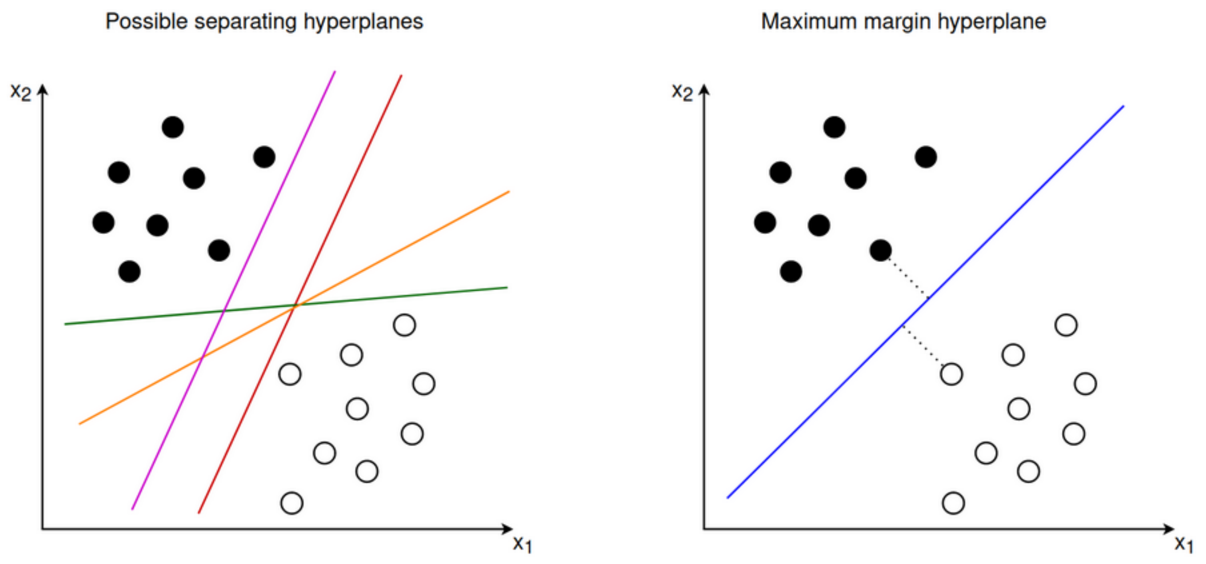


Figure 1: Different possible plane to separate data

As discussed earlier the SVM is applied to linear or nearly linear data. When we are using SVM for linear data, the points that are used to define the separating hyperplanes are called support vectors and this type of SVM is called the Primal approach - Hard-margin SVM. However, most of the real-world data is not linearly separable, and therefore the hyperplane separates the data well while ignoring some noisy examples and outliers and is called soft-margin SVM.

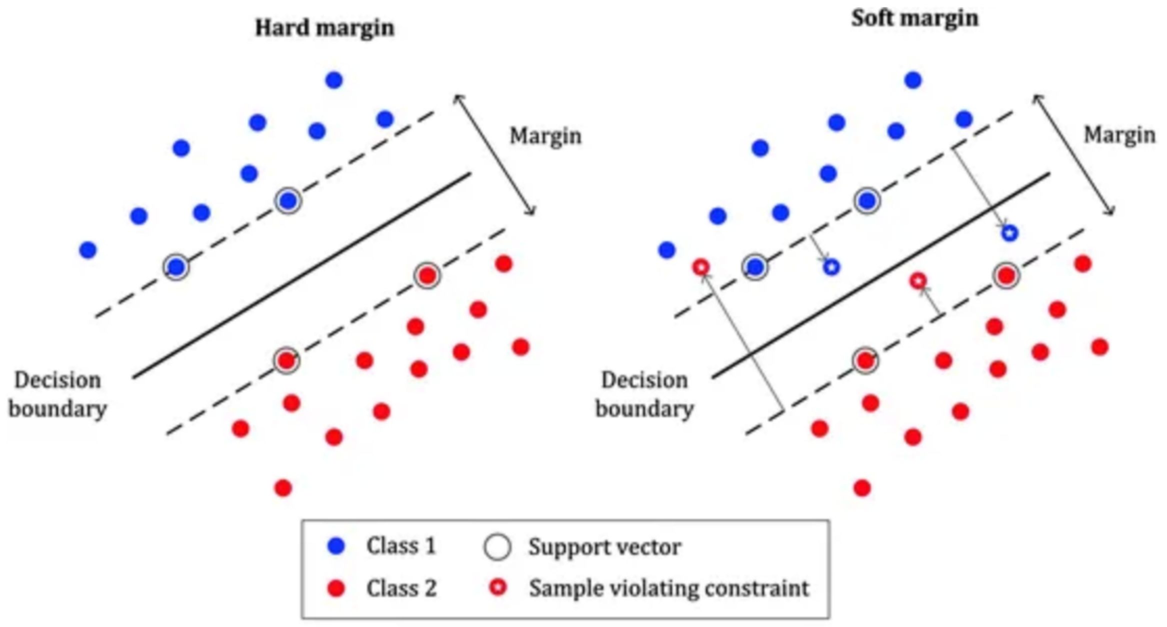


Figure 2: Hard Margin vs Soft Margin

Problem Statement

For most of assignment 3, we were tasked to classify the output by training the model based on input data using stochastic gradient descent (SGD) algorithm and SVMs. For calculating stochastic gradient descent, we will need to define the loss function. In stochastic gradient descent, we try to minimize the loss by calculating the gradient. This gradient is defined as ∂L/∂w∂b (where L is the loss function, w is the weights for each feature and b is the bias of the model). For the loss function, I used the following equation:

$$\sum_{1}^{N} max\left(0, 1 - yi(wxi + b)\right) + (\alpha|w|)\text{^}2 \qquad (1)$$

However, the scikit-learn library already has a class *SGDClassifier* that takes inputs as loss functions like "*hinge*" loss and penalties like *L1, and L2* for regularization. We can directly use this class on datasets like the *MNIST* dataset of *sklearn.datasets* package as we did in Question 2. Alternatively, we can also create our class to apply SVM and update the weights with every iteration for instance we used gradient calculation in Question 1 and implemented it in Question 3. Further, we can also use SGDClassifier to classify the data on multiclass datasets like we did in Question 4. And finally, in Question 5, we have categories for the *CIFAR 100* dataset which has an image data set of 100 categories using SVMs.

Approach(es)

The assignment was intended to o deepen our understanding of Support Vector Machines (SVM) by utilizing inbuilt functions like SGDClassifier, which allows specifying loss functions and regularization terms as input parameters. Additionally, I implemented an SVM-based gradient descent class using a given loss function.

The overall workflow followed below process:

Preprocessing of data → Splitting the data in train and test sets → Define model → Training model on train data → Evaluate model based on test using evaluation metrices.

For Question 2, a simple binary classification task was performed using SGDClassifier on the MNIST dataset. Since MNIST consists of 10 different digit classes, I selected two visually similar digits for classification, such as (1 and 9) or (3 and 8). The selected digits were relabeled, with one class assigned +1 and the other -1, transforming it into a binary classification problem. The model was then trained using this relabeled dataset, and its performance was evaluated using key metrics, including accuracy score and confusion matrix.

Then for Question 3, I created my class MySGDClassifierSVM that applies Stochastic Gradient Descent (SGD) for optimizing the hinge loss function of SVM. The training process is controlled by epochs (epoc), where the model iteratively updates weights and bias using a learning rate and regularization parameter by calculating gradient using the loss function from equation 1. Data preprocessing was done by converting y values into -1 and 1, ensuring compatibility with the hinge loss function used in SVMs. The model was trained using the *fit()* method and predictions were made using *predict()*, following standard *sklearn* conventions. After training, the coefficients were compared with those from *SGDClassifier* to evaluate performance for the MNIST dataset.

For Question 4, I extended the understanding of multi-class classification using the MNIST dataset. Unlike Question 1 of the binary classification task, this time I applied SGDClassifier to all digits from 0 to 9 to analyze its performance on a multi-class problem.

To further assess the robustness of SVM in the presence of noise, I experimented with two configurations:

- **Without Noise** – The model was trained on the original MNIST dataset.
- **With Added Noise** – Gaussian noise with a mean of 0 and a variance of 0.2 was introduced to the input data.

For **Question 5**, I extended our understanding of multi-class classification to a high-dimensional image dataset, where each image had dimensions 32×32×3 (RGB channels). Given the increased complexity of the dataset, I implemented different dimension reduction techniques to handle the high-dimensional feature space efficiently.

- Flattening the Data:
  - Initially, each image was flattened into a (rows × 32×32×3) matrix, where each row represented an image with 32×32×3 = 3072 features.
  - SGDClassifier was then applied to classify the images into 100 categories.
- Dimensionality Reduction Using ResNet:
  - Due to the high dimensionality, SVM struggled with direct classification. To mitigate this, I leveraged a ResNet model to extract lower-dimensional feature representations.
  - The extracted feature vectors were then used as inputs to the SGDClassifier for classification.
- Further Dimensionality Reduction with PCA:
  - To further optimize SVM performance, Principal Component Analysis (PCA) was applied to reduce dimensionality while preserving essential variance in the data.
  - The reduced feature set was then classified using SGDClassifier, improving computational efficiency and potentially enhancing classification accuracy.

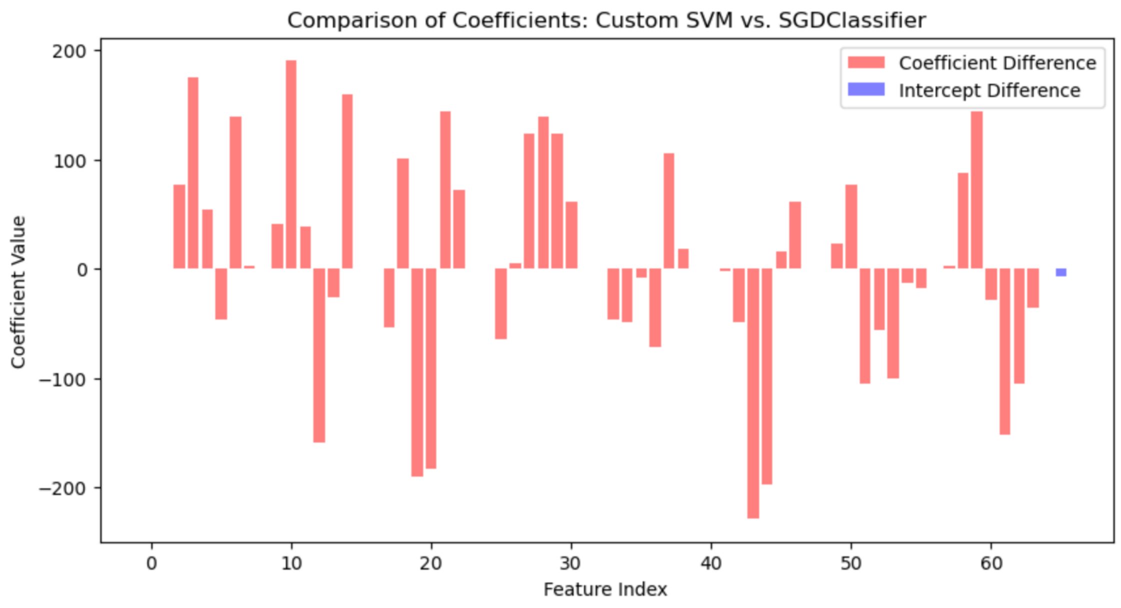
Through these steps, I tried to improve the performance of SVM for high-dimensional multi-class classification and explored the impact of feature extraction (ResNet) and dimensionality reduction (PCA) on its performance.

Experimental Results

When I used the SGDClassifier with MNIST for Binary Classification with loss as *hinge* and penalty as *l2.* For binary classification, I used two similar-looking digits i.e. 1 and 9, which resulted in an Accuracy Score of 0.9863013698630136. And confusion matrix for 1 and 9 is as below:

Confusion matrix is: [[34 0] [1 38]]

For Part 3, I trained my custom SGD-based SVM classifier (MySGDClassifierSVM) on the MNIST dataset, using hinge loss (Equation 1) for gradient calculation. The model's coefficients and intercept were then compared with those from SGDClassifier in sklearn. The results show that the coefficients were highly comparable, with 53 out of 64 coefficients exactly matching. This demonstrates that my custom implementation effectively mimics SGDClassifier, confirming its correctness. Below is the comparission between MySGDClassifierSVM vs SGDClassifier coefficients:



```
Number of differing coefficients: 53 out of: 64
```

Figure 3: Compression of MySGDClassifierSVM coefficient and SGDClassifier

In addition to comparing the coefficients, I also evaluated the model performance using accuracy score, F1 score, and confusion matrix. The results showed that my custom SGD-based SVM classifier (MySGDClassifierSVM) and SGDClassifier from sklearn produced exactly matching performance metrics on the MNIST dataset.

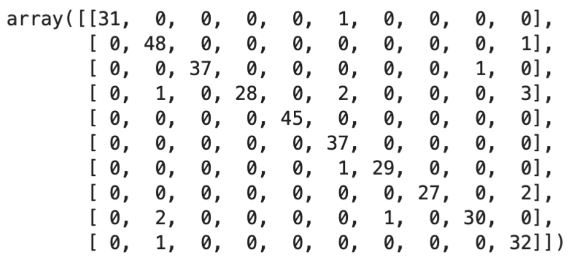
	MySGDClassifierSVM	**SGDClassifier**
Accuracy Score	0.9714285714285714	0.9714285714285714
F1 Score	0.9855072463768116	0.9855072463768116

Table 1: Comparison of Evaluation Metrics MySGDClassifierSVM and SGDClassifier

Then for part 4, I used SVM to identify the 10-digit MNIST dataset with and without Gaussian noise. And calculated the precision score, f1 score and confusion matrix:

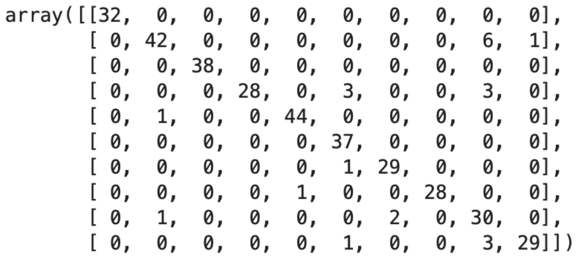
	With Gaussian Noise	**Without Gaussian Noise**
Accuracy Score	0. 9441775279678505	0.9592943630232958
F1 Score	0.9372853899962426	0. 9556115565571763

Table 2: Comparison of Evaluation Metrics of SGDClassifier with and without Gaussian Noise



```
array([[31,  0,  0,  0,  0,  1,  0,  0,  0,  0],
       [ 0, 48,  0,  0,  0,  0,  0,  0,  0,  1],
       [ 0,  0, 37,  0,  0,  0,  0,  0,  1,  0],
       [ 0,  1,  0, 28,  0,  2,  0,  0,  0,  3],
       [ 0,  0,  0,  0, 45,  0,  0,  0,  0,  0],
       [ 0,  0,  0,  0,  0, 37,  0,  0,  0,  0],
       [ 0,  0,  0,  0,  0,  1, 29,  0,  0,  0],
       [ 0,  0,  0,  0,  0,  0,  0, 27,  0,  2],
       [ 0,  2,  0,  0,  0,  0,  1,  0, 30,  0],
       [ 0,  1,  0,  0,  0,  0,  0,  0,  0, 32]])
```

Figure 3a: Confusion Matrix without Gaussian Noise



```
array([[32,  0,  0,  0,  0,  0,  0,  0,  0,  0],
       [ 0, 42,  0,  0,  0,  0,  0,  0,  6,  1],
       [ 0,  0, 38,  0,  0,  0,  0,  0,  0,  0],
       [ 0,  0,  0, 28,  0,  3,  0,  0,  3,  0],
       [ 0,  1,  0,  0, 44,  0,  0,  0,  0,  0],
       [ 0,  0,  0,  0,  0, 37,  0,  0,  0,  0],
       [ 0,  0,  0,  0,  0,  1, 29,  0,  0,  0],
       [ 0,  0,  0,  0,  1,  0,  0, 28,  0,  0],
       [ 0,  1,  0,  0,  0,  0,  2,  0, 30,  0],
       [ 0,  0,  0,  0,  0,  1,  0,  0,  3, 29]])
```

Figure 3b: Confusion Matrix with Gaussian Noise

And finally for Question 5, the results with when the Flattened input data, after applying ResNet and finally after applying PCA for dimension reduction was:

	F1 Score	**Precision Score**	**Execution Time(s)**
Flattened Data without Dimension Reduction	0.04792438973975628	0.24639886778631143	291.68
ResNet Dimension Reduction	0.3523949033954511	0.3729626411629848	83.78
PCA with ResNet Dimension Reduction	0.3905984109571352	0.39105588837758526	21.36

Discussion

Upon analyzing the results for Question 2, it became evident that SVM is highly robust for binary classification tasks, particularly when the data is linearly separable or nearly linearly separable. The model demonstrated high accuracy and precision, effectively distinguishing between the two classes (1 and 9) with minimal misclassification with only one false positive error as evident in the confusion matrix.

For Question 3, It was evident that MySGDClassifierSVM was comparable to SGDClassifier, even when tested on a large dataset like MNIST. The coefficients were highly similar, demonstrating that the custom implementation effectively learned the decision boundary. This was further confirmed by the exact match in evaluation metrics, including accuracy, F1-score, and the confusion matrix, proving that both models made identical predictions.

The slight differences in coefficients can be attributed to variations in:

- **Hyperparameter configurations** of SGDClassifier, such as learning rate scheduling and regularization.
- **Implementation differences in hinge loss**, as SGDClassifier may use a slightly different version compared to the manually defined loss function in **Equation 1**.

When analyzing the impact of Gaussian noise on multi-class classification in Question 4, I observed that SVM sustained strong performance with no significant degradation with the added noise of a standard deviation of 0.2. This demonstrates that SVM is robust to noise and can effectively handle multi-class classification tasks. These results suggest that SVM can generalize well even in real-world applications where I would see high noise in training data, making it a reliable choice for classification problems where data may contain some level of randomness or noise.

Upon analyzing the results for Question 5, it became evident that SVM struggles with raw image data, where features are not explicitly defined as in structured datasets. In images, high-level features correspond to patterns such as edges, while low-level features relate to the spatial arrangement of these high-level features. Due to the lack of explicit feature representation, the performance of SVM on raw pixel data was very poor. However, the performance improved with we use feature extraction techniques like ResNet and PCA.

Raw Feature Representation:

- When using the original feature dimensions (10,000 samples × 3,072 features), SVM exhibited poor performance.

Feature Extraction & Dimensionality Reduction:

- Extracting important features using ResNet reduced the feature dimensions to 50,000 × 2,048, significantly improving SVM performance. However, the overall accuracy remained below 40%, indicating that SVM still struggled with complex image classification.

Further Dimensionality Reduction Using PCA:

- Applying PCA further reduced the dimensionality to 50,000 × 529.
- While this did not lead to a substantial increase in accuracy, it improved computational efficiency, reducing training time by a factor of 10.

Conclusion

Our experiments show that SVM performs exceptionally well for binary classification when data is linearly or nearly linearly separable and remains robust in multi-class classification, even with added noise. However, it struggles with high-dimensional image data. While SGDClassifier delivered strong results on structured datasets like MNIST, its performance declined when applied directly to raw image pixels, revealing its limitations in capturing spatial dependencies. Using ResNet for feature extraction significantly improved classification accuracy and applying PCA further enhanced computational efficiency. These results indicate that SVM is a powerful classifier and robust to noise which can be applied to nonlinear data and multi-classification problems.

References:

- https://alpopkes.com/posts/machine_learning/support_vector_machines/#2-primal-approach---hard-margin-svm

- https://ankitnitjsr13.medium.com/math-behind-svm-support-vector-machine-864e58977fdb
- https://www.ibm.com/think/topics/support-vector-machine
- M.A. Hearst , S.T. Dumais ; E. Osuna ; J. Platt ; B. Scholkopf, "Support vector machines," IEEE (Links to an external site.)Intelligent Systems and their Applications (Links to an external site.) (Volume: 13 , Issue: 4 (Links to an external site.) , July-Aug. 1998)