Feature Engineering for Enhanced Accuracy in Image Classification

Zaima Sohail 2021-EE-53
Department of Electrical Engineering
University of Engineering and Technology, Lahore
zaimasohail20@gmail.com

Rohiya Shafiq 2021-EE-55
Department of Electrical Engineering
University of Engineering and Technology, Lahore
rohiyashafiq1020@gmail.com

1 Introduction:

Our project aims to enhance image classification accuracy through feature engineering techniques. We systematically explore various approaches to assess their impact on model performance. Feature engineering is crucial for selecting, transforming, and creating new features from raw data, improving the model's predictive ability. Leveraging the MNIST dataset, containing grayscale images of handwritten digits, provides an ideal platform for our study, serving as a benchmark for evaluating algorithm performance.

2 Background:

2.1 Linear Discriminant Analysis (LDA):

• LDA is utilized for dimensionality reduction and classification in machine learning. • It seeks a linear combination of features to maximize class separability. • Unlike PCA, LDA considers class labels to enhance discriminative power. • LDA finds applications in pattern recognition and face recognition due to its effectiveness.

2.2 Neural Networks:

• Inspired by the human brain, neural networks excel in learning complex patterns. • They consist of interconnected nodes organized into layers, facilitating iterative learning. • Neural networks adjust weights and biases to minimize prediction errors via backpropagation. • With deep learning advancements, neural networks achieve state-of-the-art performance in various domains.

2.3 Support Vector Machines (SVMs):

• SVMs are powerful classifiers widely used in classification tasks. • They identify the optimal hyperplane to maximize the margin between data points of different classes. • SVMs generalize well to unseen data, ensuring accurate classification predictions. • Support vectors, adjacent to the optimal hyperplane, play a crucial role in determining the maximal margin.

Preprint. Under review.

3 Data Preprocessing:

The code extracts the training data (training-data-m) and corresponding labels (training-labels-m) from the loaded dataset. It defines the size of the validation set (validation-m-size) to be 10,000. It separates the training data and labels into actual training data (Data-m-train) and labels (Labels-m-train) by excluding the validation set. It reshapes the training and validation data into a 2D array format suitable for feeding into machine learning models.

3.1 Purpose and Library Usage:

The purpose of this code is to prepare the MNIST dataset for training a machine learning model, particularly for LDA and an SVM classifier, by performing data loading and preprocessing steps. Library Usage: • Numpy is utilized for efficient handling and manipulation of numerical data arrays, which is essential for data preprocessing tasks such as reshaping and indexing. • Scikit-learn is a powerful library for machine learning in Python. Here, SVC is used for implementing Support Vector Classification, a popular classification algorithm, while accuracy-score is employed to evaluate the accuracy of the trained model. • Matplotlib is used for visualizing data, such as plotting graphs or displaying images. it's a commonly used library in machine learning projects for visualizing results or data distributions.

4 Feature Engineering:

4.1 Neural Network:

4.1.1 Importing Libraries:

numpy: For numerical operations and handling arrays. tensorflow: TensorFlow is an open-source machine learning framework. tensorflow.keras.layers and tensorflow.keras.models: These modules provide functions to define and create neural network models using the Keras API, which is integrated into TensorFlow.

4.1.2 Neural Network Architecture:

The neural network model is defined using the Keras Sequential API. It consists of a dense layer with 128 units and ReLU activation, followed by a dropout layer with a dropout rate of 0.2, and finally, an output layer with 10 units and softmax activation. This architecture is used for classification tasks, where the output layer produces probabilities for each class.

4.1.3 Feature Extraction:

A function is defined to create a model (feature-extractor) that includes only the part of the neural network up to the last hidden layer. This feature-extractor model is used to extract features from the data.

4.1.4 Model Compilation and Training:

The neural network model is compiled with the Adam optimizer, sparse categorical crossentropy loss function, and accuracy metric. It is then trained on the training data (Data-m-train and Labels-m-train) for 5 epochs, with validation data (Data-m-validation and Labels-m-validation) used for validation.

4.1.5 Feature Extraction and SVM Training:

Features are extracted from the trained neural network model using the feature-extractor model. An SVM classifier with a linear kernel is initialized. The SVM classifier is trained on the extracted features (features-train and Labels-m-train). Predictions are made on the validation set using the trained SVM classifier. The accuracy of the predictions is calculated using the accuracy-score function from scikit-learn.

4.1.6 Plotting Training History:

The training and validation accuracy values are plotted over epochs using Matplotlib.

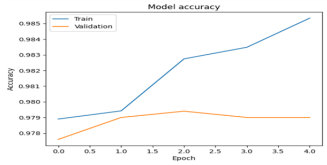

Figure 1: Epochs vs Accuracy

4.1.7 Purpose:

We engage in understanding neural networks and their libraries to attain high accuracy in tasks like feature extraction and pattern recognition. However, comprehending these concepts can be challenging despite investing significant time and effort due to the complexity of mathematical underpinnings and the intricate workings of deep learning frameworks. Achieving an accuracy of 98.08 percent may demonstrate proficiency in model implementation, but grasping the nuances of neural networks remains elusive, highlighting the difficulty in understanding their inner workings. This challenge persists despite the remarkable capabilities they offer, emphasizing the continuous learning, experimentation, and practice required for their effective utilization in real-world applications.

4.2 Linear Discriminant Analysis (LDA) with Linear kernal:

This code demonstrates the application of Linear Discriminant Analysis (LDA) for feature extraction on the MNIST dataset, followed by training a Support Vector Machine (SVM) classifier using the extracted LDA features. Here's a breakdown of each step and the purpose behind it:

4.2.1 Importing Libraries:

The necessary libraries are imported, including scikit-learn for LDA, SVM, and performance metrics calculation, matplotlib for visualization, and numpy for array manipulation.

4.2.2 Applying LDA for Feature Extraction:

LDA is instantiated as an object. LDA is applied to the training data (Data-m-train) along with their corresponding labels (Labels-m-train) to extract discriminative features. LDA is also applied to the validation data (Data-m-validation) to transform it into the same feature space as the training data.

4.2.3 Training SVM on LDA Features:

An SVM classifier with a linear kernel is initialized. The SVM classifier is trained on the LDA-transformed training data (Data-m-train-lda) and their labels (Labels-m-train).

4.2.4 Evaluating SVM on the Validation Set:

The trained SVM classifier is used to predict labels for the LDA-transformed validation data (Data-m-validation-lda). Accuracy and precision scores are calculated to evaluate the performance of the SVM classifier on the validation set.

4.2.5 Plotting the Results:

The accuracy and precision scores are visualized using a bar plot to provide a clear understanding of the SVM classification performance on the MNIST dataset with LDA features.

In the confusion matrix, rows represent actual labels (0-9) while columns represent predicted labels. Values indicate counts of data points classified into corresponding rows and columns. Ideally, high values align along the diagonal, indicating correct predictions. Off-diagonal values signify misclassifications. This confusion matrix displays mostly diagonal values, suggesting the SVM

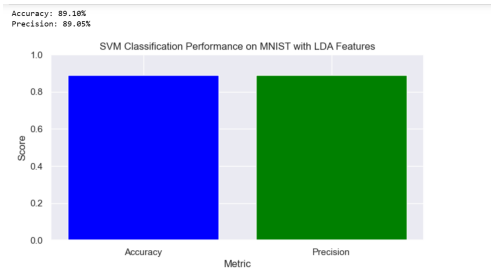

Figure 2: SVM Linear Classification with LDA Features

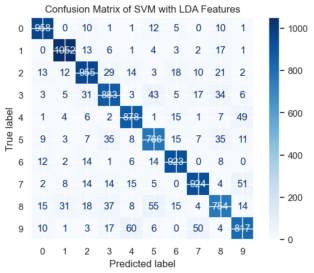

Figure 3: Confusion Matrix of SVM Linear with LDA Features

model's strong performance. Despite some errors, such as 13 images misclassified as 4 instead of 7, overall accuracy appears high.

4.2.6 Purpose:

LDA assumes that each class in the data is generated by a separate Gaussian distribution. It aims to find a projection that maximizes the separation between the means of these Gaussians while minimizing the variance within each class. The Linear Discriminant Analysis class internally performs calculations based on this assumption. It estimates the mean and covariance matrix for each class in the data. These estimated parameters relate to the properties of the underlying Gaussian distributions.

4.2.7 Key Observations:

• The code doesn't explicitly show the calculations involving Gaussians. These are handled internally by the LDA implementation. • While LDA assumes Gaussian distributions, it can still work well even if the data doesn't perfectly follow them. In essence, the libraries provide tools to implement the LDA algorithm and utilize its features for SVM classification. Even though the code doesn't directly manipulate Gaussians, the separation achieved by LDA is based on the underlying assumption of these class-specific Gaussian distributions. Overall, the purpose of this code is to demonstrate the utilization of LDA for feature extraction and its integration with SVM for classification on the MNIST dataset, showcasing the potential improvement in classification performance achieved through feature engineering techniques.

4.3 Linear Discriminant Analysis (LDA) with RBF kernal:

In this code, we apply several preprocessing and modeling steps, including scaling the data and utilizing the RBF kernel for the Support Vector Machine (SVM) classifier, in conjunction with Linear Discriminant Analysis (LDA) for feature extraction on the MNIST dataset.

4.3.1 Data Scaling:

Utilize StandardScaler to standardize features, enhancing model performance by ensuring similar feature scales.

4.3.2 LDA for Feature Extraction:

Train LDA on scaled data to extract discriminative features, aiming to reduce dimensionality while preserving class separability.

4.3.3 SVM Model with RBF Kernel:

Instantiate SVM model with RBF kernel for flexibility in capturing complex decision boundaries.

4.3.4 Training and Prediction:

Train SVM on LDA-transformed and scaled training data, then predict labels on LDA-transformed and scaled validation data.

4.3.5 Evaluation Metrics:

Compute accuracy and precision scores to assess SVM classifier performance, with precision providing insight into positive class predictions.

4.3.6 Visualization:

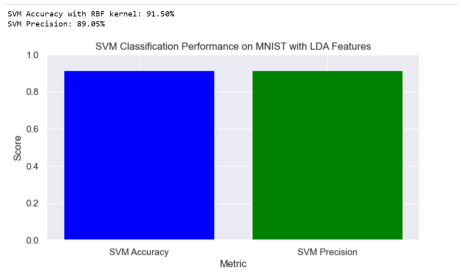

Figure 4: SVM RBF Classification with LDA Features

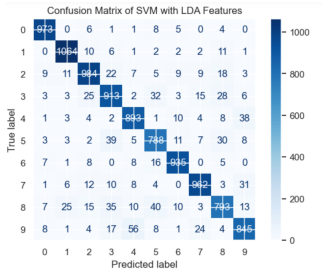

Figure 5: Confusion Matrix of SVM RBF with LDA Features

Visualize results through bar plots displaying accuracy and precision scores, alongside confusion matrix to represent SVM classifier performance in digit classification based on LDA features. In this confusion matrix, the diagonal represents the number of correct predictions made for each class in the image dataset. For instance, if the value at row 3 and column 3 is high, it means that the SVM model correctly classified many instances of the digit 3. When using the RBF kernel, particularly for digits 0, 1, 2, 6, 7, and 9, the high values along the diagonal indicate that the SVM model is performing well in correctly classifying these digits in the MNIST dataset. This suggests that the RBF kernel, with its ability to capture complex decision boundaries, enables the SVM model to effectively differentiate between these digits, resulting in higher accuracy compared to using a linear kernel.

4.3.7 Purpose:

Data scaling ensures feature comparability, enhancing SVM performance, especially with the RBF kernel. LDA extracts discriminative features, reducing dimensionality while maintaining class

separability for improved SVM classification. Utilizing the RBF kernel provides flexibility in capturing complex decision boundaries, potentially enhancing digit classification accuracy of about 91.50 percent . Accuracy and precision scores quantify SVM performance, while the confusion matrix visually interprets classification results, aiding model assessment.

5 Results and Analysis

The analysis highlights the challenges in understanding neural networks despite achieving high accuracy, emphasizing continuous learning. The purpose of using LDA with a linear kernel is to maximize separation between class means based on Gaussian assumptions. Although not explicitly shown, LDA's functionality relies on these assumptions. Similarly, LDA with the RBF kernel aims to enhance SVM performance by reducing dimensionality and maintaining class separability. The achieved accuracy of about 91.50 percent demonstrates the effectiveness of these techniques in digit classification on the MNIST dataset. Overall, the integration of LDA with SVM showcases potential improvements in classification performance through feature engineering.

6 Tackling Hurdles:

Initially, we attempted to extract features using a neural network. However, we encountered several challenges, primarily stemming from the shuffling process implemented at the outset. This shuffling procedure required more than 48 hours to execute, resulting in a meager accuracy rate of approximately 13 percent. After investing a week in analyzing the issue, we determined that the shuffling mechanism was the root cause of the problem. Upon removing the shuffling step, we observed a substantial improvement, achieving an accuracy of around 98 percent. Despite this success, we struggled to comprehend the intricate mathematical concepts and libraries associated with neural networks. Consequently, we opted to explore Linear Discriminant Analysis (LDA) as an alternative method for feature extraction from the MNIST dataset. We hypothesized that LDA, with its inherent ability to identify variances and means within the data, could offer a viable solution. Additionally, we incorporated the radial basis function (RBF) kernel to enhance the effectiveness of the Support Vector Machine (SVM) classifier.

7 Conclusion:

This study assessed feature engineering techniques' efficacy in enhancing MNIST dataset image classification accuracy. Challenges arose in comprehending neural network intricacies and addressing implementation issues. Linear Discriminant Analysis (LDA) was explored for feature extraction alongside Support Vector Machines (SVMs). Results showcased LDA's effectiveness, particularly with RBF kernels, in boosting SVM classification performance. Achieving approximately 91.50 percent accuracy underscores the potential advantages of this method. The project underscores feature engineering's significance in optimizing machine learning models while acknowledging the complexities of deep learning models like neural networks.

References

[1] Sharma, V. (2023, May 4). Exploring the MNIST dataset using Linear Discriminant Analysis (LDA). Level Up!. https://levelup.gitconnected.com/exploring-the-mnist-dataset-using-linear-discriminant-analysis-lda-7f0d025f4ce4

[2] Nikhil Sai, J. (n.d.). Digit classification using SVM on MNIST dataset [Code]. Kaggle. Retrieved from https://www.kaggle.com/code/jnikhilsai/digit-classification-using-svm-on-mnist-dataset

[3] Kothari, D. (n.d.). SVM with MNIST [Text file]. GitHub. Retrieved from https://dmkothari.github.io/Machine-Learning-Projects/SVM_with_MNIST.html

[4] TensorFlow Team. (2023, October 3). Training a neural network on MNIST with Keras. TensorFlow Datasets. Retrieved from https://www.tensorflow.org/datasets/keras_example