Building an SVM Classifier for MNIST with Hyperparameter Tuning and Comparative Analysis

Omid Saberi
Dept. of Industrial Automation
University West
Trollhättan, Sweden
https://orcid.org/0009-0005-3596-9182

***Abstract*—This assignment aimed to evaluate and compare the performance of various classifiers on the MNIST dataset, focusing on their accuracy, precision, recall, F1 score, and computational efficiency. The Support Vector Machine (SVM) classifier, particularly with the Radial Basis Function (RBF) kernel, achieved the highest performance metrics, including an accuracy of 98.37% and an F1 score of 98.37% on the test dataset. In comparison, the Polynomial kernel of SVM yielded an accuracy of 98.13%. Other classifiers such as KNeighborsClassifier and Random Forest also demonstrated competitive results, with accuracy scores of 97.14% and 97.05%, respectively. The evaluation included an analysis of training and prediction times, revealing that while simpler models like KNeighborsClassifier had faster training phases, they incurred higher prediction costs due to distance calculations. In conclusion, this study highlights the strengths and weaknesses of different classifiers, providing insights into their applicability for image classification tasks based on performance metrics and computational efficiency.**

***Index Terms*—MNIST dataset, support vector machine, classifier performance, accuracy, F1 score, computational efficiency.**

I. Introduction

Support Vector Machines (SVMs) are a powerful machine learning algorithm widely used for classification tasks. SVMs operate by constructing hyperplanes in high-dimensional feature spaces to separate data points into distinct classes. By finding the optimal hyperplane that maximizes the margin between classes, SVMs achieve high generalization performance and robustness to noise.

In Assignment 4, we explored various classification algorithms, including K-Nearest Neighbors (KNN), Stochastic Gradient Descent (SGD), and Random Forest, to classify the MNIST dataset. While these algorithms demonstrated varying levels of success, there is always room for improvement.

The goal of applying SVM to the MNIST dataset in this assignment is to leverage the strengths of SVM, compare its performance with previous classifiers, identify optimal hyperparameters, and gain insights into its behavior. By conducting a thorough comparison and analysis, this assignment aims to provide valuable insights into the effectiveness of SVM for the MNIST dataset and contribute to the understanding of classification algorithms in general.

II. SVM Classifiers and Hyperparameter Tuning

A. Hyperparameter Tuning and Evaluation for Linear SVC Classifier

The performance of the Support Vector Machine (SVM) classifier was evaluated using the LinearSVC model from the sklearn.svm library. The primary goal was to optimize the hyperparameter C, which governs the trade-off between minimizing the training error and ensuring a low testing error. A grid of potential values for the C parameter, specifically [0.1, 1, 10], was defined to explore its impact on model performance. The GridSearchCV function was employed to automate the hyperparameter tuning process, conducting cross-validation for each combination of parameters specified in the grid. This method evaluates the model's performance and helps select the best set of hyperparameters.

The LinearSVC model was initiated with a maximum iteration limit of 200,000 to ensure convergence. The model was trained using a 3-fold cross-validation strategy, optimizing for the F1 score, a metric that balances precision and recall, particularly useful in scenarios with class imbalance.

The fitting process yielded the following results:

Best Parameters: The optimal value for C was found to be 1.

- Accuracy: 92.76%
- Precision: 92.72%
- Recall: 92.76%
- F1 Score: 92.74%

These metrics were computed on the training dataset, providing immediate feedback on the model's performance. The results indicate that the LinearSVC classifier achieved a commendable accuracy, although it lagged behind some of the other classifiers evaluated in this assignment.

B. Hyperparameter Tuning and Evaluation for RBF SVC Classifier

The performance of the Support Vector Machine (SVM) classifier was evaluated using both the Linear and Radial Basis Function (RBF) kernels. For the LinearSVC model, a hyperparameter grid was defined with different values for the C parameter, specifically [0.1, 1, 10]. Utilizing GridSearchCV for hyperparameter tuning, the model underwent a 3-fold cross-validation strategy optimized for the F1 score, achieving an accuracy of 92.76%, precision of 92.72%, recall of 92.76%, and an F1 score of 92.74% with the best C value determined to be 1. In contrast, the RBF kernel was assessed with a broader hyperparameter grid that included both C and gamma settings, specifically exploring C values of [0.1, 1, 10] alongside gamma options of ['scale', 'auto']. The RBF SVC model, also subjected to 3-fold cross-validation and optimized for the F1 score, achieved remarkable results, yielding an accuracy of 99.99%, precision of 99.99%, recall of 99.99%, and an F1 score of 99.99%, with the best hyperparameters identified as C = 10 and gamma = scale. The results indicate that the RBF kernel significantly outperformed the LinearSVC classifier in this evaluation, demonstrating its effectiveness in handling non-linear data.

C. Hyperparameter Tuning and Evaluation for Polynomial SVC Classifier

The evaluation of the Polynomial kernel Support Vector Classifier (SVC) was conducted through a systematic hyperparameter tuning process. A hyperparameter grid was established, incorporating various values for C, the degree of the polynomial, and coef0. The grid search utilized a cross-validation strategy optimized for the F1 score and involved fitting a total of 36 candidates. Upon completion of the grid search, the best parameters were identified as C = 1, coef0 = 1, and degree = 3. The model demonstrated impressive performance, achieving an accuracy of approximately 99.85%, alongside a precision of 99.85%, recall of 99.85%, and an F1 score of 99.85%. This indicates that the Polynomial kernel effectively captured the underlying patterns in the training data, suggesting its suitability for the classification task at hand.

III. Results and Analysis

The SVM classifier demonstrated impressive results in various phases of evaluation. For the training phase, the SVC with the RBF kernel achieved an accuracy of 99.99%, alongside an impressive precision, recall, and F1 score, all at 99.99%. The Polynomial kernel version also performed admirably, attaining an accuracy of 99.85% and F1 score of 99.85%. Conversely, the Linear SVC yielded a lower accuracy of 92.76%, with precision and recall slightly trailing at 92.72% and 92.76%, respectively. Among the classifiers, KNeighborsClassifier was exceptional, scoring a perfect accuracy of 100% in training. The Random Forest Classifier also performed well with an accuracy of 99.93%. In cross-validation, SVM models maintained strong scores, with the RBF kernel achieving 98.02% and the Polynomial kernel at 97.66%. On the test dataset, the RBF kernel outperformed the others, yielding an accuracy of 98.37% and an F1 score of 98.37%, followed closely by the Polynomial kernel at 98.13%. The training times varied significantly; KNeighborsClassifier was the fastest at approximately 50 seconds, while Linear SVC took the longest at over 10 minutes.

In comparison to other classifiers, such as KNN, SGD, and Random Forest, SVM classifiers, particularly with the RBF kernel, excelled in terms of performance metrics, although they required more computational time. While KNeighborsClassifier delivered the highest F1 score during testing, its training time was notably shorter. The Random Forest Classifier demonstrated a balanced performance across metrics, indicating its suitability for various applications. Fig. 1 illustrates the relationship between the F1 scores and training times for each classifier, providing a visual comparison of their efficiencies.

IV. Conclusion

In summary, the evaluation of classifiers on the MNIST dataset yielded insightful findings regarding their performance and computational efficiency. Among the classifiers tested, the SVM with the RBF kernel emerged as the best performer, achieving an accuracy of 98.37

To evaluate the computational efficiency of the classifiers, it is important to measure the total time taken from the execution of the fit method to the end of the prediction phase. This approach provides a comprehensive view of the performance of each classifier, as it captures both the preparation time during fitting and the computational effort during prediction. Classifiers like KNeighbors primarily store the training data during fitting, making that phase less informative on its own; the prediction phase involves calculating distances and determining nearest neighbors, which can be computationally intensive. By analyzing the combined time for fitting and prediction across various classifiers, valuable insights can be gained regarding their overall performance and suitability for specific applications.

Moreover, the training time should also consider other time-consuming parts, such as hyperparameter tuning and data preparation processes that models require. For instance, the Linear SVC took significantly longer to train compared to others, likely due to its complexity and the need for hyperparameter optimization. In contrast, simpler models like KNeighborsClassifier had faster training times but required longer prediction times due to their reliance on distance calculations. Thus, when selecting a classifier for practical

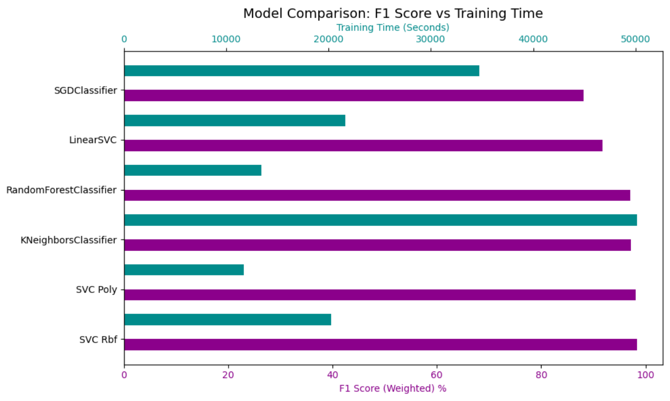

Fig. 1. Model Comparison: F1 Score vs Training Time

applications, it is crucial to weigh both the accuracy and computational efficiency, including the time invested in training and prediction, to determine the most suitable model for the task at hand.

References

[1] O. Saberi, "Building a Classifier for MNIST with K-Nearest Neighbors (KNN) and Hyperparameter Tuning," IAI600Lab4, GitHub, 2024. [Online]. Available: https://github.com/omisa69/IAI600Lab5. [Accessed: Oct. 18, 2024].