ESA Group 28 : Project Report

1. Introduction

This report delves into the exploration of the MNIST dataset. We examine the performance of four distinct classifiers trained on this dataset, highlighting their accuracy, efficiency, and overall effectiveness. To further enhance the models, we conduct hyperparameter optimization, showcasing its impact on improving predictive performance. Through this study, we aim to provide insights into the behavior of different classifiers and the role of fine-tuning hyperparameters in achieving optimal results.

2. Data Exploration & Preprocessing

The MNIST dataset comprises grayscale images of handwritten digits, where each image is represented as a 28x28 matrix of pixel values ranging from 0 to 255. This results in a total of 784 features per image, which capture the intensity of each pixel. The dataset is balanced, with class distributions ranging from 366 (smallest class) to 395 (largest class) examples per digit.

Figure 1: Example loaded data

As for pre-processing, we apply division by 255 (maximal channel value) and optionally apply scaler (where it's mentioned). Performance metric was accuracy among all conducted experiments. Dummy estimator pre-trained using most frequent strategy scored 0.077 on test set serving as a baseline model.

For the SVM classifier specifically, the dataset was transformed by flattening each 28x28 matrix into a 1D array of 784 values, creating a total of 3750 such arrays. This transformation was required to meet the SVM classifier's 2D array input format.

3. Regression with Default Hyperparameters & Optimization

KNN Classifier

The default hyperparameters that were used for the KNN classifier are:

- `k = 5` Number of neighbors
- `weights = 'uniform'` Weighting function
- `metric = 'minkowski'` Distance computation metric

Using the default hyperparameters, the following scores were obtained:

- **Accuracy**: 91.33%.
- **Auto-grading score**: 80/100.

The hyperparameters that were considered in the grid search are:

- `n_neighbors` with values `[1, 3, ..., 29]`. This resembles the `k` value. We used only odd values to prevent the risk of equal scores for multiple classes for single data points, and we stop at `k = 29` since the accuracy typically no longer increases once `k` reaches a certain value. As `k` increases, the decision boundaries typically become less complex.
- `weights` with values `['uniform', 'distance']`. These values represent the weighting function to be used in the classification, where `'uniform'` assumes uniform weights for each data point, while `'distance'` assumes weights based on the inverse of the distance between data points.
- `metric` with values `['minkowski', 'euclidean', 'manhattan', 'cosine']`, which determines how the distance values between data points are calculated.

Applying grid search yields the hyperparameters that give the highest accuracy: `n_neighbors = 3`, `weights = 'distance'`, and `metric = 'cosine'`. The accuracy using these hyperparameters is 94.53%.

SVM Classifier

The SVM classifier was trained using the default hyperparameters:

- `C = 1.0` Regularization parameter
- `gamma = 'scale'` Kernel co-efficient
- `kernel = 'rbf'` Radial Basis Function Kernel

The performance of the model was evaluated using accuracy on the test set and auto-grading via the scoring tool. The results were as follows:

- **Accuracy:** 95% on the test set.
- **Auto-grading score:** 100%.

The performance estimate of the model is fair because it was trained with default hyperparameters and evaluated on a separate 20% test subset of the dataset, ensuring it was tested on unseen data. Additionally, the model received a 100% score on the autograder, indicating strong generalization to new data.

The main hyperparameters chosen for optimization for SVM classifier are listed below:

- `C` (Regularization parameter): Low values of this parameter makes the decision boundary smoother, while higher values make it more complex: `[0.1, 0.5, 1.0, 2.0]`
- `gamma` (Kernel co-efficient): Determines influence of each datapoint. Low values allow for less complex decision boundaries. The `scale` value scales

the `gamma` value according to the training data. The `auto` value sets `gamma` based on the number of features: `[0.05, 0.1, 'scale', 'auto']`
- `kernel` (Kernel): `['linear', 'rbf']`

Once GridSearch is completed, the best hyperparameters are applied, and the model is evaluated on the test set to estimate its accuracy. The GridSearch resulted in a very small performance improvement of +0.01%. This minimal gain suggests that the model with the default hyperparameters already fits the data quite well.

Logistic Regression Classifier

Standard scaler with default configuration was used for pre-processing data. Similarly to SVM, several C values (inverse of regularization strength) were explored, class weight was conditionally adjusted to respect slighly uneven classes distribution. Grid search also tried training models with or without bias parameter, adjust different iterations count and inspect multiple approaches how parameters are penalized (either L1 or L2 penalty were used).

The best discovered parameters setting: - C: 0.1 `[out of 0.1, 0.5, 1.0, 2.0]` - class_weight: balanced `[out of None, 'balanced']` - fit_intercept: True `[out of True, False]` - max_iter: 150 `[out of 150, 1000]` - penalty: l2 `[out of 'l1', 'l2']`

- **Accuracy:** 90.4% on the test set.
- **Auto-grading score:** 30%.

Decision Tree

Standard scaler with default configuration was used for pre-processing data. The grid search was employed to uptrain model using different impurity criterions (gini or entropy) and again, taking classes distributions into account when setting weight for classes.

Optimal parameters set found:

- class_weight: None `[out of None, 'balanced']`
- criterion: entropy `[out of 'gini', 'entropy']`
- **Accuracy:** 76.4% on the test set.
- **Auto-grading score:** 45%.

4. Conclusion

Among the classifiers evaluated, the SVM and KNN classifiers demonstrated the strongest performance on the test set, with the SVM particularly excelling in generalization, as evidenced by its results on the Autograder. These results highlight the SVM's ability to effectively capture the underlying patterns in the dataset without overfitting.

In contrast, the logistic regression classifier showed signs of overfitting to the training data. Despite achieving a high accuracy on the test set, its performance on the Autograder was noticeably lower, indicating potential issues with its ability to generalize to unseen data.

The decision tree classifier similarly exhibited overfitting tendencies. However, unlike logistic regression, it underperformed on both the test set and the Autograder, reflecting its challenges in effectively capturing the complexities of the dataset.

These observations showcase the importance of balancing model complexity and generalization during classifier selection and highlight the advantages of hyperparameter optimization in achieving competitive performance.