ELL409 Assignment 3

Ashi Veerman 2021MT10241

November 10, 2024

Contents

List of Tables

List of Figures

1 Introduction

In this report, we explore three popular machine learning algorithms—Support Vector Machine (SVM), Random Forest, and AdaBoost.

1.1 Support Vector Machine (SVM)

Support Vector Machine (SVM) is a supervised learning algorithm commonly used for classification tasks. The main idea behind SVM is to find the hyperplane that best separates the classes in the feature space. The algorithm seeks to maximize the margin between the closest points from each class, known as support vectors. A wider margin often leads to better generalization. Why SVM is helpful:

- SVM works well in high-dimensional spaces, making it ideal for tasks where there are many features.
- It is robust to overfitting, especially in high-dimensional space.
- SVM can be effective when the classes are not linearly separable by using kernel functions (like the Radial Basis Function kernel) to map the data to a higher-dimensional space.

1.2 Random Forest

Random Forest is an ensemble learning method that builds a collection of decision trees, each trained on a different subset of the data. The predictions of all individual trees are then aggregated, typically by taking the majority vote (classification) or average (regression). Why Random Forest is helpful:

- It is less prone to overfitting compared to individual decision trees, as the randomness of the tree construction introduces variety.
- It performs well with both categorical and numerical data, and can handle missing values effectively.
- Random Forest is highly interpretable, as feature importance can be extracted from the model, showing which features influence the predictions the most.

1.3 AdaBoost

AdaBoost (Adaptive Boosting) is an ensemble method that combines multiple weak learners (typically decision trees) into a single strong learner. It iteratively adjusts the weights of misclassified instances, forcing the subsequent learner to focus on those hard-to-classify instances. Why AdaBoost is helpful:

- AdaBoost can significantly improve the performance of weak learners and is less prone to overfitting compared to other boosting methods.
- It works well with binary classification problems and can be easily adapted to multiclass problems.
- The method is robust to noisy data and is computationally efficient.

2 Assignment Overview

In the assignment, several steps were taken to build and fine-tune machine learning models for classification tasks. The primary tasks included data preprocessing, implementation of the algorithms and hyperparameter tuning for the three algorithms—SVM, Random Forest, and AdaBoost.

2.1 Data Preprocessing

In this assignment, the following preprocessing techniques were applied:

- **Resize**: The input images were resized to a consistent size (28,28) to ensure uniformity across the dataset. This step is particularly useful when dealing with image data to reduce computational complexity.
- **Normalization**: The pixel values were normalized to a range between 0 and 1, ensuring that all features contribute equally to the model's training.
- **Standardization**: The features were standardized by subtracting the mean and dividing by the standard deviation. This ensures that the features have a mean of 0 and a standard deviation of 1, which is essential for many machine learning algorithms, especially SVM.
- **Principal Component Analysis (PCA)**: PCA was applied to reduce the dimensionality of the data while retaining as much variance as possible. This helps in improving model performance by reducing noise and improving generalization.

2.2 Hyperparameter Tuning

Hyperparameter tuning was carried out to optimize the performance of the machine learning algorithms. This process involves searching for the best set of hyperparameters that yield the highest performance on the validation set. For each model, the following steps were performed:

- **Grid Search**: A grid search was employed to explore the different combinations of hyperparameters such as the number of trees (for Random Forest), kernel types (for SVM), and learning rates (for AdaBoost).
- **Best Hyperparameters Selection**: After performing the grid search, the hyperparameters that resulted in the best F1 scores were chosen for the final model.

3 SVM Hyperparameter Tuning and Performance

For the Support Vector Machine (SVM) model, a grid search was performed to identify the best combination of hyperparameters. The parameter grid used is as follows:

The hyperparameters tested were:

- **C**: Regularization parameter that controls the trade-off between maximizing the margin and minimizing the classification error. The values tested were {0.1, 1, 10}.

Parameter	Values	Description
C	0.1, 1, 10	Regularization parameter
Kernel	linear, rbf	Kernel type
Gamma	0.01, 0.1, 1.0	Kernel coefficient (used only with ‘rbf‘ kernel)

Table 1: SVM Hyperparameter Grid for Grid Search

- **Kernel**: Specifies the kernel type to be used in the algorithm. Both ‘linear‘ and ‘rbf‘ (radial basis function) kernels were tested.
- **Gamma**: Defines how far the influence of a single training example reaches, used specifically with the ‘rbf‘ kernel. The values tested were $\{0.01, 0.1, 1.0\}$.

This grid of parameters was used in a grid search approach to identify the best performing combination for the SVM model based on evaluation metrics.

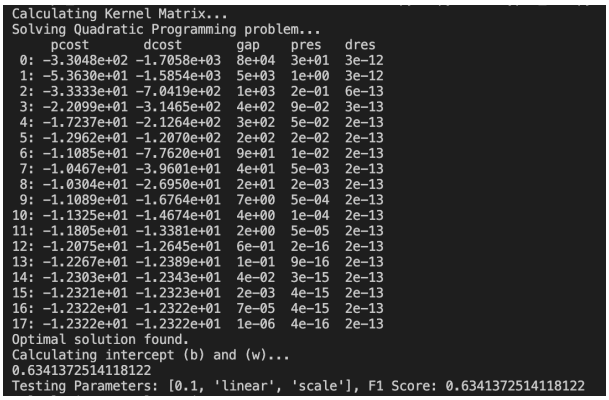

Figure 1: Gridsearch logs

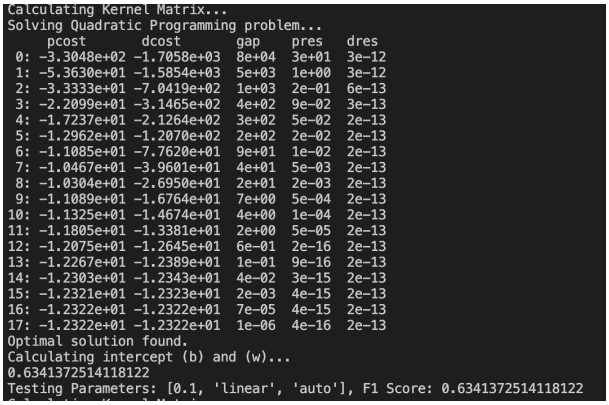

Figure 2: Gridsearch logs

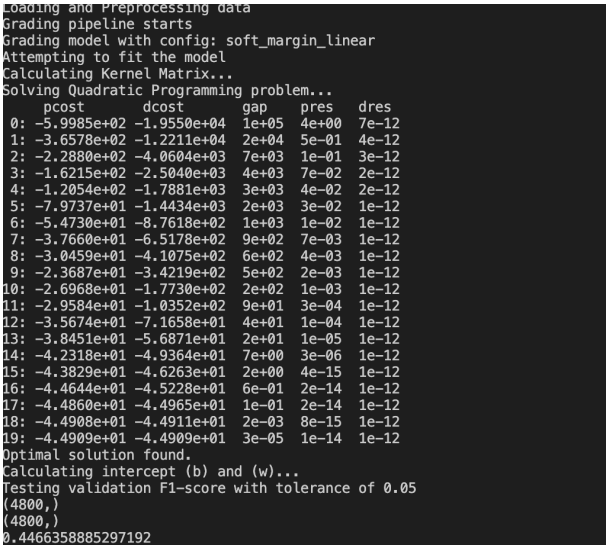

Figure 3: Gridsearch logs

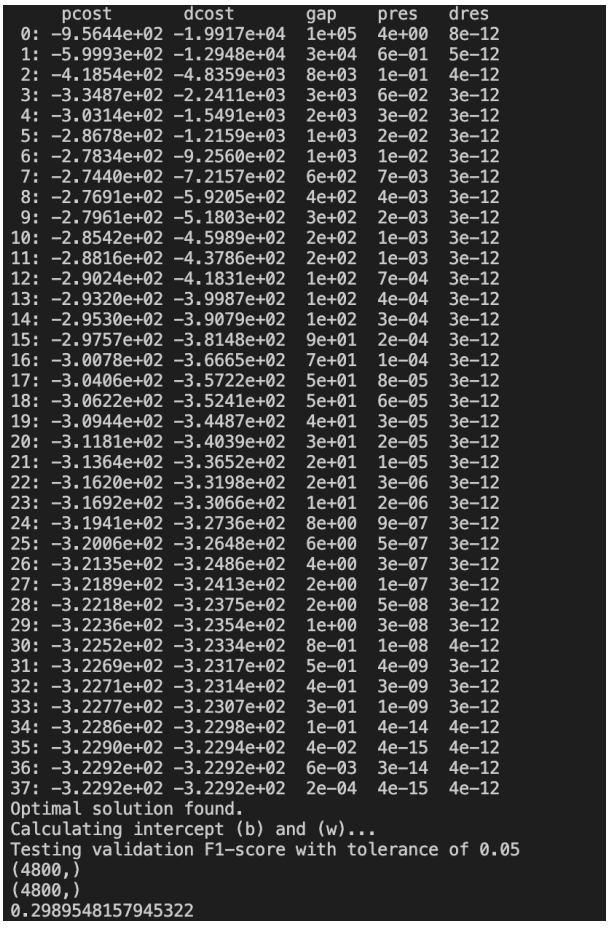

Figure 4: Gridsearch logs

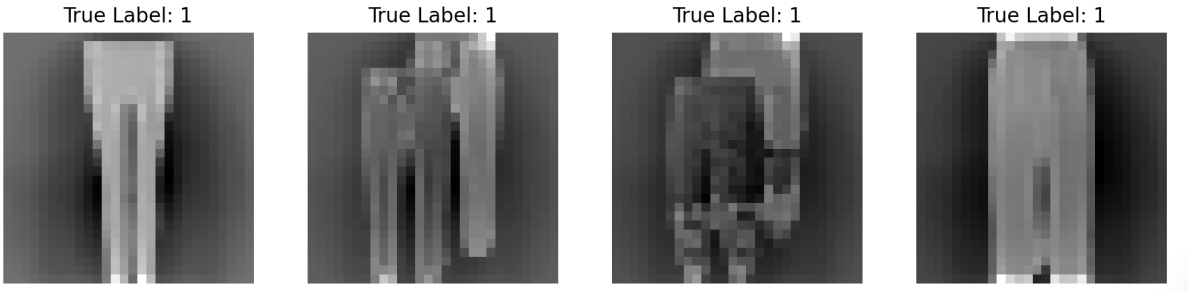

Figure 5: Training Images misclassified

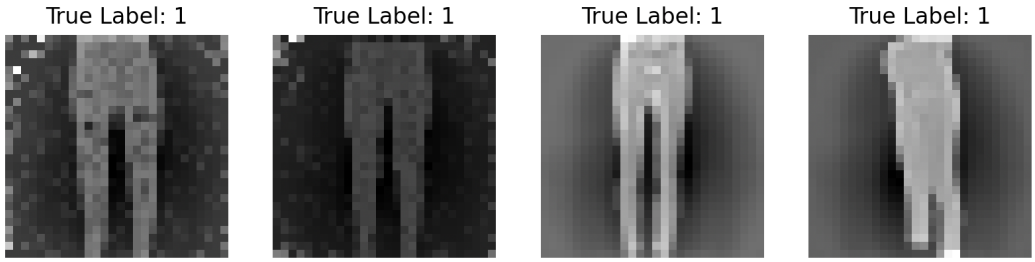

Figure 6: Validation Images misclassified

4 Random Forest Hyperparameter Tuning and Performance

The best hyperparameters for the Random Forest model were determined to be:

- **Number of Trees**: 100
- **Max Depth**: 3
- **Min Samples Split**: 200

The corresponding performance metrics are:

$$\text{Best F1 Score} = 0.87258$$

$$\text{Training Accuracy} = 0.93802$$

$$\text{Test Accuracy} = 0.90521$$

$$\text{Training F1 Score} = 0.61164$$

$$\text{Test F1 Score} = 0.87258$$

These results indicate that the model with the chosen hyperparameters performs well, with high test accuracy and F1 score, while maintaining a reasonable F1 score on the training set. The combination of **100 trees**, **max depth of 3**, and **min samples split of 200** provides the optimal balance for this Random Forest model on the given dataset.

Table 2: Random Forest Results Summary with All Parameters

Trees	**Max Depth**	**Min Samples Split**	**F1 Score**
10	2	None	0.58129
10	2	100	0.58123
10	2	200	0.58109
10	3	None	0.58122
10	3	100	0.58124
10	3	200	0.58125
10	4	None	0.58134
10	4	100	0.57929
10	4	200	0.58149
10	5	None	0.53841
10	5	100	0.52679
10	5	200	0.53192
10	7	None	0.53132
10	7	100	0.53075
10	7	200	0.52691
10	6	None	0.51077
10	6	100	0.50585
10	6	200	0.51720
10	8	None	0.50949
10	8	100	0.49236
10	8	200	0.49970
10	10	None	0.52829
10	10	100	0.52030
10	10	200	0.49476
20	2	None	0.58152
20	2	100	0.58150
20	2	200	0.58046
20	3	None	0.58206
20	3	100	0.58122
20	3	200	0.58147
20	4	None	0.58088
20	4	100	0.58097
20	4	200	0.87146
20	5	None	0.52758
20	5	100	0.52814
20	5	200	0.53506
20	7	None	0.52691

Continued on next page

Trees	Max Depth	Min Samples Split	F1 Score
20	7	100	0.52313
20	7	200	0.52440
20	6	None	0.51316
20	6	100	0.51671
20	6	200	0.51348
20	8	None	0.52101
20	8	100	0.51281
20	8	200	0.51098
20	10	None	0.52327
20	10	100	0.51059
20	10	200	0.52293
50	2	None	0.58167
50	2	100	0.58166
50	2	200	0.87160
50	3	None	0.87233
50	3	100	0.87258
50	3	200	0.58141
50	4	None	0.58009
50	4	100	0.87146
50	4	200	0.86975
50	5	None	0.53002
50	5	100	0.53347
50	5	200	0.52854
50	7	None	0.52760
50	7	100	0.52400
50	7	200	0.53196
50	6	None	0.51313
50	6	100	0.51312
50	6	200	0.51307
50	8	None	0.51405
50	8	100	0.50950
50	8	200	0.51657
50	10	None	0.52156
50	10	100	0.52101
50	10	200	0.52244
100	2	None	0.87251
100	2	100	0.58178
100	2	200	0.87251
100	3	None	0.58118
100	3	100	0.58183
100	3	200	0.87268
100	4	None	0.86991
100	4	100	0.87048
100	4	200	0.87070
100	5	None	0.53332

Continued on next page

Trees	Max Depth	Min Samples Split	F1 Score
100	5	100	0.79707
100	5	200	0.53108
100	7	None	0.52506
100	7	100	0.52605
100	7	200	0.52482
100	6	None	0.51522
100	6	100	0.51307
100	6	200	0.51183
100	8	None	0.51990
100	8	100	0.51961
100	8	200	0.51618
100	10	None	0.52305
100	10	100	0.52716
100	10	200	0.52307
150	2	None	0.87185
150	2	100	0.87258
150	2	200	0.87212
150	3	None	0.58178
150	3	100	0.58145
150	3	200	0.87233
150	4	None	0.58027
150	4	100	0.58088
150	4	200	0.87094
150	5	None	0.53088
150	5	100	0.53102
150	5	200	0.53107
150	7	None	0.52534
150	7	100	0.52843
150	7	200	0.52504
150	6	None	0.51153
150	6	100	0.51367
150	6	200	0.51269
150	8	None	0.51388
150	8	100	0.51210
150	8	200	0.51573
150	10	None	0.52497
150	10	100	0.52797
150	10	200	0.52629
Best Parameters			**0.87268**

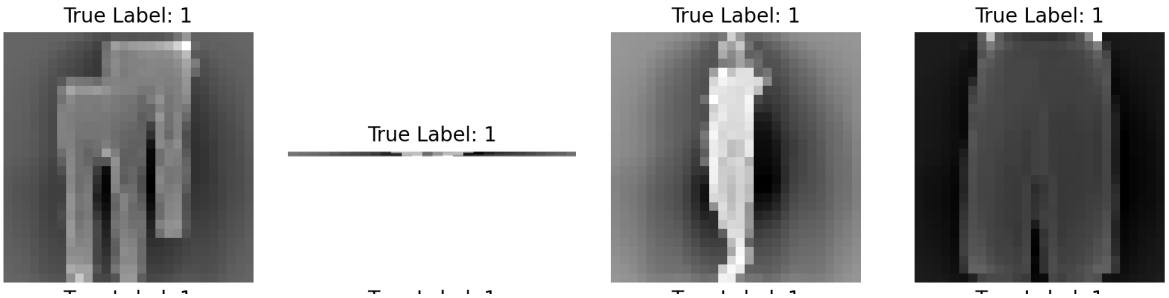

Figure 7: Training Images misclassified

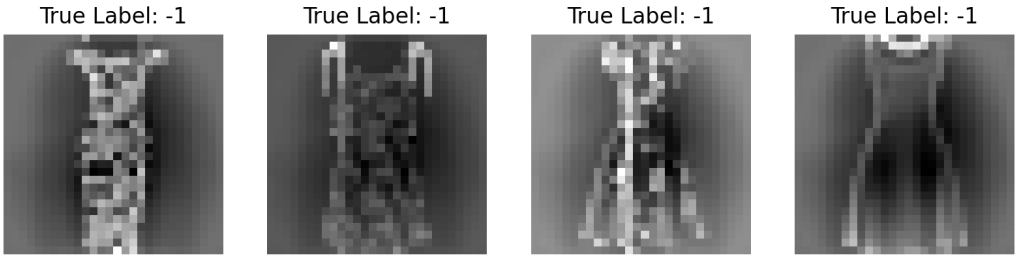

Figure 8: Validation Images misclassified

5 Adaboost Hyperparameter Tuning and Performance

The following table shows the hyperparameter tuning results for AdaBoost. The first parameter represents the number of trees, and the second parameter represents the maximum depth of the trees. The corresponding F1 scores for different combinations of these parameters are provided.

Number of Trees	**Maximum Depth**	**F1 Score**
10	1	0.4286
10	2	0.9114
10	5	0.8969
10	10	0.9177

Table 3: Hyperparameter Tuning Results for AdaBoost

Based on the hyperparameter tuning, the best parameters for AdaBoost are:

- **Best Parameters**: [10, 2]
- **Best F1 Score**: 0.9114

- **Training Accuracy**: 0.9485
- **Test Accuracy**: 0.9346
- **Training F1 Score**: 0.9314
- **Test F1 Score**: 0.9114

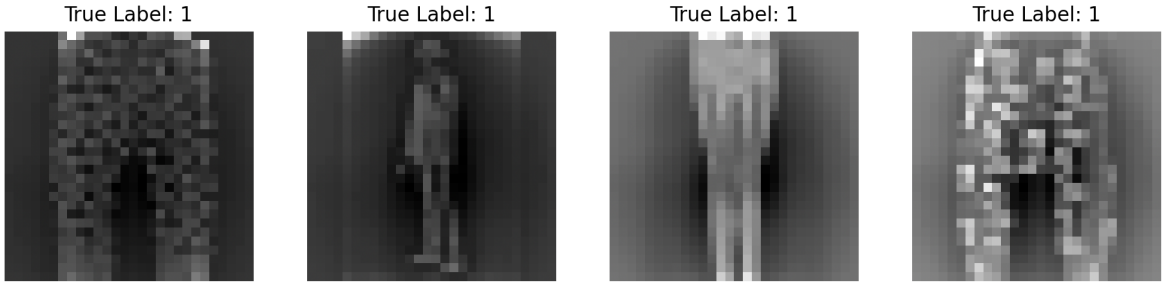

Figure 9: Training Images misclassified

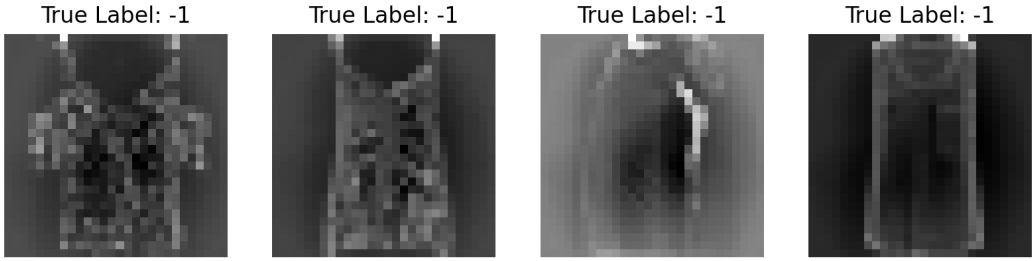

Figure 10: Validation Images misclassified