SENG 474 Data Mining - Assignment 2

Andrew Braun - V00955955

February 26, 2025

1 Component 1: Experiment And Analysis

1.1 Introduction

This assignment centers on the implementation, experimentation, and analysis of 2 machine learning models: Support Vector Machines (SVMs) and Neural Networks (NNs) with the primary objective of evaluating their performance on a binary classification task derived from the Fashion-MNIST dataset. The Fashion-MNIST dataset comprises 28x28 grayscale images of clothing items divided into ten classes, but for this assignment, the task is simplified to binary classification by focusing on two specific classes: "Sandal" (class 5) and "Sneaker" (class 7). The dataset is preprocessed by normalizing pixel values to the range [0, 1]. To increase the complexity of the task, label noise is introduced by flipping labels in the training set with a fixed probability *p* (chosen between 0.1 and 0.3), making the classification problem more challenging and prone to overfitting. The assignment is structured into four main components, each addressing specific aspects of model training, hyperparameter tuning, and performance evaluation.

The first model, Support Vector Machines (SVMs), involves implementing the soft-margin version, which allows for the use of linear and Gaussian (RBF) kernels. The regularization parameter *C* is a key hyperparameter, with higher values leading to less regularization, while the Gaussian kernel introduces an additional scale parameter γ to control the influence of individual training examples. The second model, Neural Networks (NNs), focuses on implementing a feedforward neural network with at least one hidden layer. Key hyperparameters include the number of hidden layers, nodes per layer, activation functions, regularization strength, and optimization algorithms such as stochastic gradient descent or Adam. K-fold cross-validation is used for hyperparameter tuning and model evaluation, with the implementation reused from Assignment 1 to ensure robust validation and avoid overfitting to the test set.

The assignment is divided into four main experiments. The first experiment involves training a linear SVM, exploring a range of regularization parameters *C* using k-fold cross-validation to identify the optimal value, and analyzing the trade-off between underfitting and overfitting by plotting training and test errors as functions of *C*. The second experiment extends to kernelized SVMs with a Gaussian kernel, where the scale parameter γ is varied, and the optimal regularization parameter $C\gamma$ is determined for each value. The performance of these tuned SVMs is evaluated on the test set and compared to the linear SVM. The third experiment focuses on neural networks, exploring various hyperparameter configurations such as network architecture (e.g., number of layers and nodes) and activation functions. k-fold cross-validation is used to select the optimal configuration, and the impact of varying individual hyperparameters on training and test errors is analyzed. The final experiment compares the optimally tuned versions of the linear SVM, Gaussian kernel SVM, and neural network based on their test errors, with the significance of differences assessed using confidence intervals.

The primary goal of this assignment is to gain hands-on experience with implementing and tuning machine learning models, understanding the impact of hyperparameters, and evaluating model performance on a real-world dataset. By introducing label noise and exploring different preprocessing techniques, the assignment emphasizes the challenges of overfitting and the importance of robust validation strategies. This report documents the experimental setup, results, and analysis for each component, providing insights into the strengths and limitations of the models under study.

1.1 Analysis

1.1.1 Part 1: Linear SVM

Our first experiment involved training a linear SVM. The core idea behind SVMs is to find the optimal hyperplane that separates the two classes by maximizing the margin between them. The margin is defined as the distance between the hyperplane and the nearest support vector (data point). In this experiment, we used the soft-margin version of SVM, which is designed to handle data that is not necessarily linearly separable. Since we used a linear kernel, the only hyperparameter we tuned was the regularization parameter *C*. A small *C* value emphasizes maximizing the margin, potentially leading to underfitting, while a larger *C* value focuses on correctly classifying the training data, which may result in a smaller margin and overfitting. For this experiment, we employed k-fold cross-validation and tested a logarithmically spaced array of *C* values ranging from 10^{-3} to 10^{3}.

Before discussing the results, it is important to note two key points. First, to reduce computational time, we used a subset of the training data for cross-validation, specifically 1,000 examples of class 0 (Sandal) and 1,000 examples of class 1 (Sneaker), resulting in a total subset of 2,000 examples. Second, the introduction of label noise (with probability *p*=0.2) led to higher validation errors. However, this noise was intentionally added to encourage the model to generalize better and become more robust, as it prevents the model from overfitting to the training data. With these considerations in mind, the results aligned with our expectations. The optimal *C* value obtained from cross-validation was 0.464158. In Part 4 of the analysis, we will train a model on the full training set using this optimal *C* value to evaluate its performance on the test set.

The next experiment with the linear SVM focused on observing the evolution of training and test errors as *C* varied. For this experiment, we expanded the range of *C* values from 10^{-4} to 10^{4}

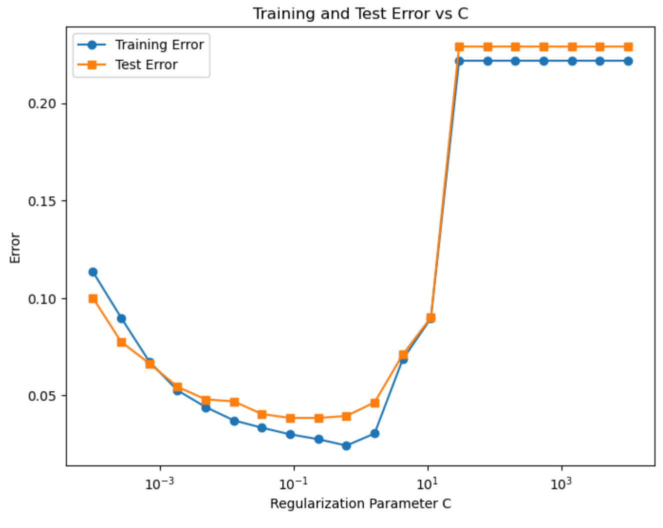

Figure 1: Test and training error vs C values

and trained each model on the full training set. The results are plotted in Figure 1. The lowest test error achieved was 0.0385, corresponding to a *C* value of 0.0886. Notably, this *C* value differs from the optimal value obtained during cross-validation, as the latter was trained on a subset of the data, while this experiment used the full training set. An interesting observation from the plot is the sudden increase in test error as *C* becomes very large. This behavior occurs because excessively large *C* values cause the model to focus too much on fitting the noisy training data, leading to overfitting and poor generalization. In hindsight, exploring such large *C* values was unnecessary, but it provided valuable insight into the model's sensitivity to the regularization parameter.

In this experiment, we explored the behavior of a linear SVM on a binary classification task derived from the Fashion-MNIST dataset. By varying the regularization parameter *C*, we observed the trade-off between margin maximization and classification accuracy. The introduction of label noise added complexity to the task, but it also encouraged the model to learn more robust patterns, ultimately improving generalization. The optimal *C* value obtained through cross-validation (0.464158) provided a balance between underfitting and overfitting, while the exploration of a wider range of *C* values revealed the dangers of excessive regularization (or lack thereof). The sudden spike in test error at large *C* values highlighted the importance of regularization in preventing overfitting, especially in the presence of noisy data. These findings underscore the critical role of hyperparameter tuning in building effective machine learning models and provide a foundation for comparing the performance of linear SVMs with other methods, such as kernelized SVMs and neural networks, in subsequent experiments.

1.1.2 Part 2: Gaussian SVM

In this section, we move beyond linear classifiers and explore the use of kernelized Support Vector Machines (SVMs) with a Gaussian kernel. Unlike linear SVMs, which are limited to finding linear decision boundaries, kernelized SVMs can capture complex, nonlinear relationships in the data by mapping the input features into a higher-dimensional space. The Gaussian kernel, also known as the Radial Basis Function (RBF) kernel, introduces a scale parameter *γ* (sometimes referred to as the "bandwidth" parameter), which controls the influence of individual training examples on the decision boundary. A small *γ* value results in a smoother decision boundary, while a larger *γ* value allows the model to fit more intricate patterns, potentially leading to overfitting.

The goal of the first experiment is to tune both the scale parameter *γ* and the regularization parameter *C* to achieve optimal performance. For each value of *γ* in a logarithmically spaced range from 10^{-5} to 10^{4}, we use k-fold cross-validation on the training subset to determine the optimal *Cγ*. This results in a set of tuned SVMs, each corresponding to a specific (*γ*,*Cγ*) pair. We then perform another round of cross-validation to identify the optimal *γ* value. We ended with an optimal (*γ*,*Cγ*) pair of (10^{-5}, 10^{4}). In part 4 we will train a model on this optimal pairing.

The next experiment involved varying *γ* and observing the evolution of the training and test errors. For each *γ* value in the range, we determined the optimal *Cγ* and plotted the resulting errors, as shown in Figure 2. The best test error achieved was 0.0180, corresponding to a (γ, $C\gamma$) pair of (0.0100, 10.0). This represents a substantial improvement over the linear SVM, demonstrating the power of the Gaussian kernel in capturing nonlinear patterns in the data.

In this section, we explored the use of kernelized SVMs with a Gaussian kernel, which enables the model to capture complex, nonlinear relationships in the data. By systematically tuning the scale parameter γ and the regularization parameter C, we identified an optimal (γ,$C\gamma$) pair of $(10^{-5}, 10^{4})$. Further experimentation with varying γ values revealed that the Gaussian kernel significantly outperforms the linear SVM, achieving a best test error of 0.0180 with the pair (0.0100,10.0). These results highlight the importance of kernelized SVMs in handling nonlinear classification tasks and the critical role of hyperparameter tuning in achieving optimal performance. In the next section, we will compare the performance of the tuned Gaussian kernel SVM with that of the linear SVM and neural networks, providing a comprehensive evaluation of the methods explored in this assignment.

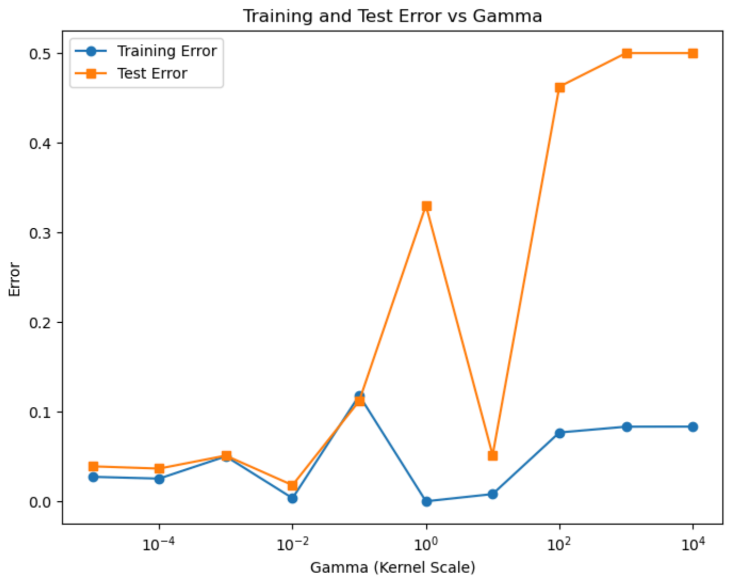

Figure 2: Test and training error vs gamma

1.1.3 Part 3: Neural Network

In this section, we explore the training of neural networks, a powerful and flexible class of models capable of capturing complex, nonlinear relationships in data. Unlike linear models or kernelized SVMs, neural networks offer significant freedom in their architecture and training configuration. Key design choices include the number of hidden layers, the number of nodes in each layer, the choice of activation functions (nonlinearities), the amount of regularization, and the optimization algorithm. Additionally, hyperparameters such as the learning rate schedule, the number of training epochs, and the use of early stopping can significantly impact the model's performance.

To manage the complexity of hyperparameter tuning, we focus on a computationally feasible set of configurations. Specifically, we explore neural networks with one or two hidden layers, varying the number of nodes in each layer and experimenting with different activation functions (e.g., ReLU, sigmoid). Regularization is kept simple, with a small number of regularization parameter values, while the primary emphasis is on structural hyperparameters. Using k-fold cross-validation on the training set, we identify the optimal hyperparameter configuration for the neural network. This approach ensures that the model is tuned without overfitting to the test set. It turns out that the optimal set of hyperparameters was {'hidden_layer_sizes': (50, 50), 'activation': 'relu', 'alpha': 0.001, 'solver': 'adam'}. In part 4 we will train a model on these optimal settings.

The next experiment involved varying the hidden layer architecture, testing configurations with one or two hidden layers. The best-performing architecture was, as we can see in Figure 3, (300, 200), achieving a remarkably low test error of 0.0155. The results revealed a strong correlation between training and test errors, suggesting that the label noise in the training data encouraged better generalization. However, the error curve was not smooth, with noticeable jumps at certain configurations, indicating that increasing model complexity does not always lead to better performance. While the error generally decreased with complexity, the optimal configuration was not the most complex, demonstrating that simpler models can sometimes outperform more complex ones.

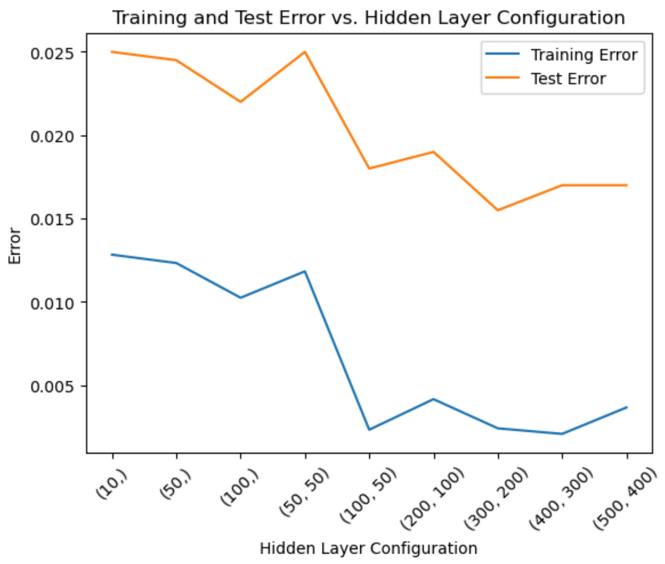

Figure 3: Training and test error vs hidden layer configuration

The next experiment focused on varying the number of training epochs, testing a range of 10 to 300. The test error flattened out after 200 epochs, indicating that the model had likely converged by this point. Figure 4 shows that the optimal number of epochs was 100, achieving a test error of 0.0150. This highlights the importance of early stopping and careful tuning of the training duration to avoid unnecessary computation and potential overfitting. These experiments underscore the flexibility of neural networks and the critical role of hyperparameter tuning in achieving optimal performance.

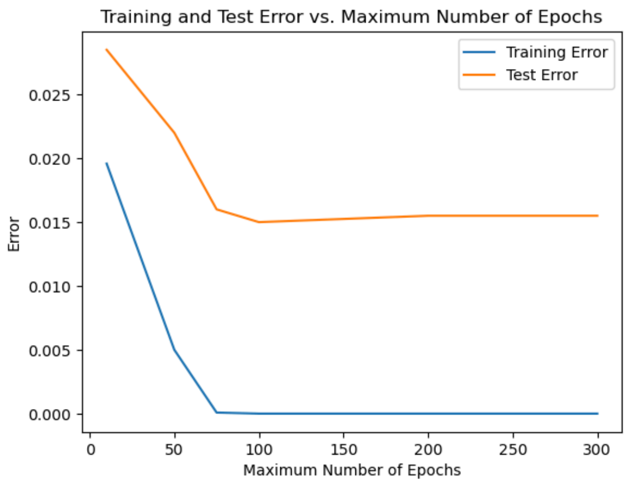

Figure 4: Test and training error vs number of epochs

In this section, we conducted a series of experiments to explore the training and tuning of neural networks for the binary classification task. By systematically varying key hyperparameters—such as the hidden layer architecture and the number of training epochs—we identified optimal configurations that balance model complexity and generalization performance. The best-performing hidden layer architecture, (300, 200), achieved a remarkably low test error of 0.0155, while the optimal number of training epochs was found to be 100, yielding a test error of 0.0150. These results highlight the importance of careful hyperparameter tuning and the flexibility of neural networks in capturing complex patterns in data.

1.1.4 Part 4: Comparison of Optimal Hyperparameters

In this section, we compare the performance of the three optimally tuned models: the linear SVM, the Gaussian kernel SVM, and the neural network. Each model was trained on the entire training set using its respective optimal hyperparameters, and their performance was evaluated on the test set. The linear SVM achieved a test error of 0.0385 with an optimal *C* value of 0.464158, while the Gaussian kernel SVM achieved a slightly higher test error of 0.0390 with an optimal (γ,C) pair of (10^{-5}, 10^{4}). The neural network outperformed both SVM models, achieving a test error of 0.0255 with the best hyperparameter configuration of {'hidden_layer_sizes': (50, 50), 'activation': 'relu', 'alpha': 0.001, 'solver': 'adam'}. These results reveal interesting insights into the performance of each model on the binary classification task.

The linear SVM and Gaussian kernel SVM performed similarly, with test errors of 0.0385 and 0.0390, respectively. This suggests that, for this specific task, the additional complexity of the Gaussian kernel did not provide a significant performance improvement over the linear SVM. This could be due to the nature of the data or the presence of label noise, which may limit the benefits of nonlinear decision boundaries. On the other hand, the neural network demonstrated its ability to capture complex patterns in the data, achieving a significantly lower test error of 0.0255. This highlights the flexibility and power of neural networks, even in the presence of label noise.

To assess whether the differences in test errors are statistically significant, we can consider the confidence intervals for each model's error rate. Given the test set size of 2,000 examples, the standard error for a test error *p* can be approximated as $\sqrt{\frac{p(1-p)}{n}}$, where n=2000. For the neural network, the standard error is approximately 0.0035, resulting in a 95% confidence interval of 0.0186 to 0.0324. For the linear SVM, the standard error is approximately 0.0043, with a confidence interval of 0.0301 to 0.0469, and for the Gaussian kernel SVM, the standard error is also approximately 0.0043, with a confidence interval of 0.0306 to 0.0474. The confidence intervals for the neural network and the SVMs do not overlap, indicating that the neural network's performance is significantly better than both SVM models. However, the test errors of the linear SVM and Gaussian kernel SVM are very close, and their confidence intervals overlap, suggesting that their performance differences are not statistically significant.

In conclusion, the comparison of the three tuned models highlights the strengths and limitations of each approach. The neural network achieved the lowest test error, demonstrating its ability to capture complex patterns in the data and outperform both SVM models. The linear SVM and Gaussian kernel SVM performed similarly, with no significant improvement observed from using the Gaussian kernel. The confidence intervals confirm that the neural network's performance is significantly better than both SVM models, while the differences between the linear and Gaussian kernel SVMs are not statistically significant. These results underscore the importance of model selection and hyperparameter tuning in achieving optimal performance. While neural networks offer greater flexibility and power, they also require careful tuning and computational resources. In contrast, SVMs provide a simpler and more interpretable alternative, particularly when the data does not require highly nonlinear decision boundaries.

1.2 Conclusion

This assignment provided a comprehensive exploration of three machine learning models—linear SVM, Gaussian kernel SVM, and neural networks—on a binary classification task derived from the Fashion-MNIST dataset. By systematically tuning hyperparameters and evaluating performance, we gained valuable insights into the strengths and limitations of each approach. The linear SVM achieved a test error of 0.0385, while the Gaussian kernel SVM performed similarly with a test error of 0.0390, suggesting that the additional complexity of the Gaussian kernel did not yield significant improvements for this task. In contrast, the neural network outperformed both SVM models, achieving a significantly lower test error of 0.0255, demonstrating its ability to capture complex patterns in the data. Confidence intervals confirmed that the neural network's performance was statistically better than both SVMs, while the differences between the linear and Gaussian kernel SVMs were not significant. These results highlight the importance of model selection and hyperparameter tuning, with neural networks offering greater flexibility and power at the cost of increased complexity, while SVMs provide a simpler and more interpretable alternative for tasks where nonlinear decision boundaries are less critical. Overall, the assignment underscored the trade-offs between model complexity, generalization, and computational efficiency in machine learning.