SENG 474 - Assignment 1

Evan Strasdin

February 26, 2025

Intro

This report presents a comprehensive analysis of three machine learning methods—Support Vector Machines (SVMs) with linear and Gaussian kernels, and Neural Networks—applied to a binary classification problem derived from the Fashion-MNIST dataset. The Fashion-MNIST dataset consists of 28x28 grayscale images of clothing and apparel items, categorized into ten classes. For this project, we focus on classifying two specific classes: "Sandal" (class 5) and "Sneaker" (class 7), relabeled as class 0 and class 1, respectively. The goal is to evaluate the performance of these methods under challenging conditions, including label noise, and to compare their effectiveness in terms of test error.

The Fashion-MNIST dataset was preprocessed to create a binary classification task. The dataset was loaded using the mnist_reader utility, which provides access to the training and test sets. The training set contains 60,000 examples, and the test set contains 10,000 examples. Only data points corresponding to classes 5 "Sandal" and 7 ("Sneaker") were retained, reducing the dataset to a binary classification problem. The training set was further reduced to 1,000 examples (500 from each class) to ensure balanced representation and computational efficiency. The labels for classes 5 and 7 were relabeled as 0 and 1, respectively, to simplify the binary classification task. Each pixel value in the images was rescaled to the range [0, 1] by dividing by 255. This normalization step ensures that the input features are on a consistent scale, which is particularly important for methods like SVMs and neural networks. To simulate a more challenging learning scenario, label noise was introduced into the training set. Each label in the training set was flipped with a probability of 0.2, creating noisy labels that make the classification task more difficult and increase the risk of overfitting.

The project is structured into four main components, each focusing on a specific aspect of the analysis. First, a linear SVM was trained with varying regularization parameters C. k-fold cross-validation was used to select the optimal C value, and training and test errors were plotted as a function of C. The results were discussed to understand the impact of regularization on model performance. Second, a Gaussian kernel SVM was trained with varying scale parameters γ and regularization parameters C. k-fold cross-validation was used to select the optimal γ and C values. Training and test errors were plotted as a function of γ, and the results were compared to those of the linear SVM to evaluate the benefits of using a nonlinear kernel. Third, a neural network with one hidden layer was trained, varying hyperparameters such as the number of nodes, activation functions, and regularization. k-fold cross-validation was used to select the optimal hyperparameter configuration. Two experiments were conducted: one varying the number of nodes in the hidden layer and another varying the maximum number of training epochs. Training and test errors were plotted and discussed to analyze the impact of these hyperparameters on model performance. Finally, the optimally tuned linear SVM, Gaussian kernel SVM, and neural network were compared based on their test errors. The significance of the differences in test errors was assessed, and the results were discussed in the context of confidence intervals. This comparison provides insights into the relative strengths and weaknesses of each method for the given classification task.

This project provides a detailed exploration of the trade-offs between model complexity, regularization, and generalization performance in the presence of label noise. By systematically varying hyperparameters and using k-fold cross-validation, we identify optimal configurations for each method and compare their effectiveness on the binary classification task. The results offer insights into the strengths and limitations of SVMs and neural networks for image classification tasks under noisy conditions. The implementation leverages several tools and libraries. NumPy was used for numerical computations and array manipulations.

scikit-learn was employed for implementing SVMs and neural networks, as well as utility functions like k-fold cross-validation. Matplotlib was used for visualizing training and test errors. The Fashion-MNIST dataset provided the image data and labels, serving as the foundation for the experiments.

This report is organized to guide the reader through the experimental setup, methodology, results, and conclusions, providing a clear and professional analysis of the performance of each method. The findings contribute to a deeper understanding of how different machine learning approaches handle noisy data and complex classification tasks.

Part 1: SVM, Linear kernel

1.1: Methods

In this section, we trained a linear Support Vector Machine (SVM) with the soft-margin formulation, which allows for misclassifications by introducing a regularization parameter C. The goal was to identify the optimal value of C that balances the trade-off between maximizing the margin and minimizing classification errors.

The hyperparameter C was explored using a logarithmically spaced grid, starting with an initial value $C_0 = 10^{-3}$ and a base $\beta = 2$, resulting in ten candidate values: $C_0, \beta C_0, \beta^2 C_0, \ldots$. This approach ensures that both underfitting (high regularization) and overfitting (low regularization) regimes are captured. To select the optimal C, 5-fold cross-validation was performed on the training set, which consisted of 1,000 examples with noisy labels. The cross-validation error was computed for each C, and the value yielding the lowest error was selected as the optimal C.

After identifying the optimal C, a larger range of C values was explored to analyze the behavior of training and test errors. The range was extended by setting $C_0 = 10^{-5}$ and $\beta = 3$, generating 15 values. For each C in this extended range, the linear SVM was trained on the full training set, and both training and test errors were computed. The training error was calculated using the original (non-noisy) labels to evaluate the model's fit to the true data distribution, while the test error was computed on the test set to assess generalization performance.

The results were visualized in a plot showing training and test errors as a function of C, using a logarithmic scale for the x-axis. This plot provides insights into the impact of regularization on model performance.

1.2: Results and Discussion

The optimal value of C selected through 5-fold cross-validation was $C = 0.008$. This value achieved the lowest cross-validation error, indicating a good balance between underfitting and overfitting.

The plot of training and test errors versus C reveals several key trends. For small values of C (e.g., $C < 0.001$), both training and test errors are high, indicating underfitting due to excessive regularization. As C increases, the training error decreases monotonically, reflecting the model's increasing capacity to fit the training data. However, the test error initially decreases, reaching a minimum around $C = 0.01$, and then begins to increase for larger values of C. This behavior is characteristic of overfitting, where the model becomes too complex and starts to capture noise in the training data, leading to poor generalization.

The minimum test error occurs at $C = 0.01$, which is close to the optimal C value identified through cross-validation. This consistency validates the effectiveness of the cross-validation process in selecting a robust hyperparameter configuration. At this point, the test error is approximately 0.0835, while the training error is 0.072, indicating a small generalization gap.

For very large values of C (e.g., $C > 1$), the training error continues to decrease, approaching zero, while the test error increases significantly. This divergence between training and test errors is a clear sign of overfitting. The model becomes overly complex, fitting the noisy training labels too closely and failing to generalize to unseen data.

In summary, the results demonstrate the importance of regularization in controlling model complexity and preventing overfitting. The linear SVM performs well when C is appropriately tuned, achieving a balance between fitting the training data and generalizing to the test set. The optimal C value identified through cross-validation effectively minimizes the test error, highlighting the utility of this approach for hyperparameter tuning.

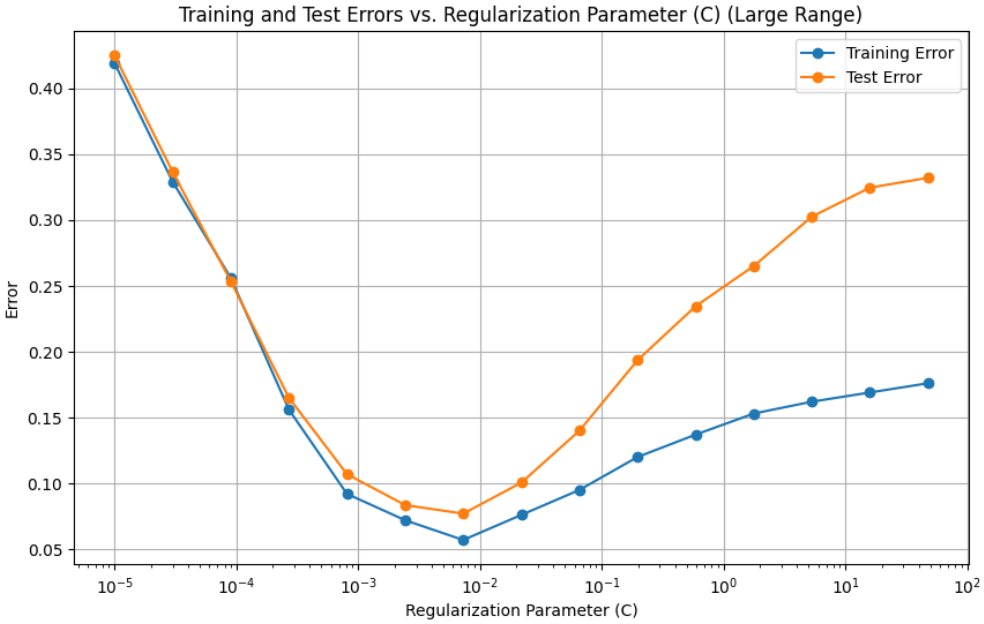

Figure 1: SVM Linear Kernel - Training and Test Errors

Part 2: SVM, Gaussian Kernel

2.1: Methods

In this section, we trained a kernelized Support Vector Machine (SVM) using the Gaussian (RBF) kernel. The Gaussian kernel introduces a scale parameter γ, which controls the influence of individual training examples. For each value of γ, we performed hyperparameter tuning to select the optimal regularization parameter C using 5-fold cross-validation. This process was repeated for a range of γ values, resulting in a set of tuned SVMs, each corresponding to a specific (γ, C) pair.

The hyperparameter γ was explored using a logarithmically spaced grid, starting with an initial value $\gamma_0 = 10^{-6}$ and a base $\beta_\gamma = 3$, resulting in 13 candidate values: $\gamma_0, \beta_\gamma \gamma_0, \beta_\gamma^2 \gamma_0, \ldots$. Similarly, the regularization parameter C was explored using a logarithmically spaced grid, starting with $C_0 = 10^{-4}$ and a base $\beta_C = 3$, resulting in 13 candidate values. For each γ, the optimal C was selected based on the lowest cross-validation error.

Once the optimal (γ, C) pairs were identified, the SVMs were trained on the full training set, and both training and test errors were computed. The training error was calculated using the original (non-noisy) labels, while the test error was computed on the test set to evaluate generalization performance. The results were visualized in a plot showing training and test errors as a function of γ, using a logarithmic scale for the x-axis.

2.2: Results and Discussion

The optimal configuration identified through cross-validation was $\gamma = 0.002187$ and $C = 1.9683$, achieving a cross-validation error of 0.2130. This configuration strikes a balance between model complexity and generalization performance.

The plot of training and test errors versus γ reveals several key trends. For small values of γ (e.g., $\gamma < 10^{-5}$), both training and test errors are high, indicating underfitting due to insufficient model complexity. As γ increases, the training error decreases, reflecting the model's increasing capacity to fit the training data. However, the test error initially decreases, reaching a minimum around $\gamma = 0.002187$, and then begins to increase for larger values of γ. This behavior is characteristic of overfitting, where the model becomes too complex and starts to capture noise in the training data, leading to poor generalization.

The minimum test error occurs at $\gamma = 0.002187$, with a corresponding test error of 0.061. This is significantly lower than the test error of the linear SVM (0.0835), demonstrating the advantage of using a nonlinear kernel for this classification task. The Gaussian kernel allows the SVM to capture more complex decision boundaries, leading to better performance on the test set.

For very large values of γ (e.g., $\gamma > 0.1$), the training error continues to decrease, approaching zero, while the test error increases significantly. This divergence between training and test errors is a clear sign of overfitting. The model becomes overly complex, fitting the noisy training labels too closely and failing to generalize to unseen data.

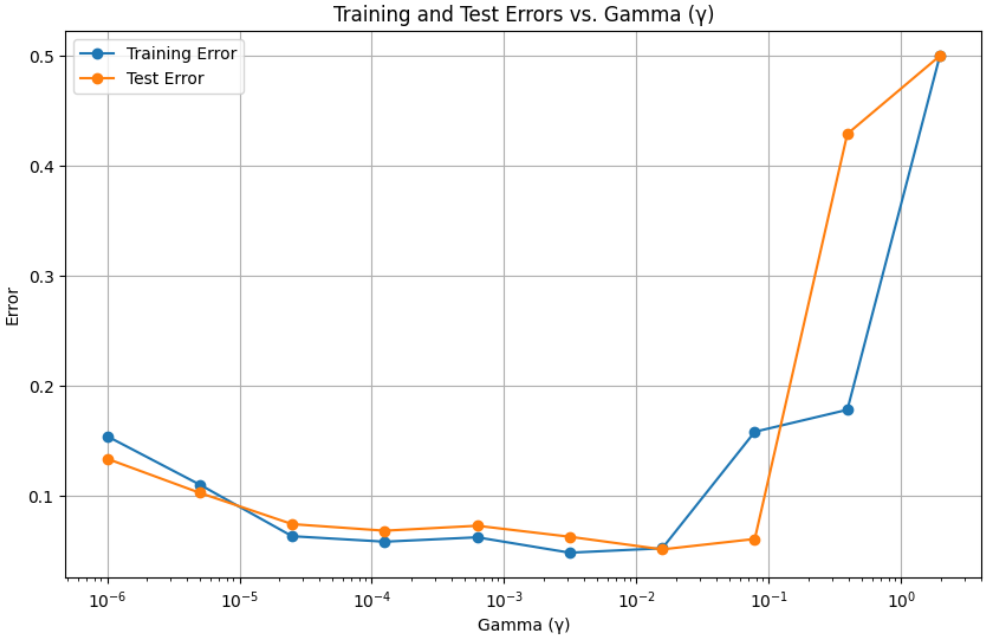

Figure 2: Neural Network Experiment 1 - Number of Nodes

2.3: Summary

The results demonstrate the importance of selecting an appropriate kernel scale parameter γ to balance model complexity and generalization performance. The Gaussian kernel SVM outperforms the linear SVM when γ is appropriately tuned, achieving a lower test error. The optimal γ value identified through cross-validation effectively minimizes the test error, highlighting the utility of this approach for hyperparameter tuning.

2.4: Comparison with Linear SVM

The test error of the optimally tuned Gaussian kernel SVM (0.061) is significantly lower than that of the linear SVM (0.0835). This improvement demonstrates the benefit of using a nonlinear kernel to capture more complex decision boundaries. However, the Gaussian kernel SVM is more sensitive to the choice of hyperparameters, particularly γ, and requires careful tuning to avoid overfitting.

In conclusion, the Gaussian kernel SVM provides a powerful tool for nonlinear classification tasks, but its performance depends critically on the appropriate selection of hyperparameters. The results highlight the importance of cross-validation in identifying optimal configurations and ensuring robust generalization performance.

Part 3: Neural Network

3.1: Methods

In this section, we trained a neural network with one hidden layer to classify the Fashion-MNIST dataset. The neural network architecture allows for considerable flexibility in hyperparameter configuration, including the number of hidden layers, the number of nodes in each hidden layer, the choice of activation function, and the number of training epochs. To manage computational complexity, we focused on networks with one or two hidden layers and explored a small number of activation functions ('relu' and 'tanh').

The hyperparameter tuning process involved evaluating different configurations using 5-fold cross-validation on the training set. The configurations included varying the number of nodes in the hidden layer and the activation function. The optimal configuration was selected based on the lowest cross-validation error.

Once the optimal configuration was identified, we conducted two experiments to analyze the impact of specific hyperparameters on model performance. In the first experiment, we varied the number of nodes in the hidden layer while keeping other hyperparameters fixed. In the second experiment, we varied the maximum number of training epochs. For each experiment, the neural network was trained on the full training set, and both training and test errors were computed. The results were visualized in plots showing training and test errors as a function of the hyperparameter being varied.

3.2: Results and Discussion

The optimal configuration identified through cross-validation was a neural network with one hidden layer containing 100 nodes and using the 'relu' activation function. This configuration achieved the lowest cross-validation error of 0.2280.

3.3: Experiments

3.3.1: Experiment 1: Varying the Number of Nodes in the Hidden Layer

In this experiment, we varied the number of nodes in the hidden layer from 10 to 300 while keeping the activation function fixed at 'relu'. The results are shown in the plot of training and test errors versus the number of nodes.

For small numbers of nodes (e.g., 10), both training and test errors are high, indicating underfitting due to insufficient model capacity. As the number of nodes increases, the training error decreases, reflecting the model's increasing ability to fit the training data. The test error also decreases initially, reaching a minimum around 100 nodes, and then begins to increase for larger numbers of nodes. This behavior is characteristic of overfitting, where the model becomes too complex and starts to capture noise in the training data, leading to poor generalization.

The minimum test error occurs at 100 nodes, with a corresponding test error of 0.0735. This configuration strikes a balance between model complexity and generalization performance. For larger numbers of nodes (e.g., 300), the training error continues to decrease, while the test error increases, indicating overfitting.

3.3.2: Experiment 2: Varying the Maximum Number of Epochs

In this experiment, we varied the maximum number of training epochs from 1 to 200 while keeping the hidden layer configuration fixed at 100 nodes and the activation function fixed at 'relu'. The results are shown in the plot of training and test errors versus the number of epochs.

For a small number of epochs (e.g., 1), both training and test errors are high, indicating underfitting due to insufficient training. As the number of epochs increases, the training error decreases, reflecting the model's increasing ability to fit the training data. The test error also decreases initially, reaching a minimum around 100 epochs, and then begins to increase for larger numbers of epochs. This behavior is characteristic of overfitting, where the model becomes too complex and starts to capture noise in the training data, leading to poor generalization.

The minimum test error occurs at 100 epochs, with a corresponding test error of 0.1635. However, the test error increases significantly for larger numbers of epochs, indicating overfitting. This highlights the importance of early stopping to prevent overfitting and ensure robust generalization performance.

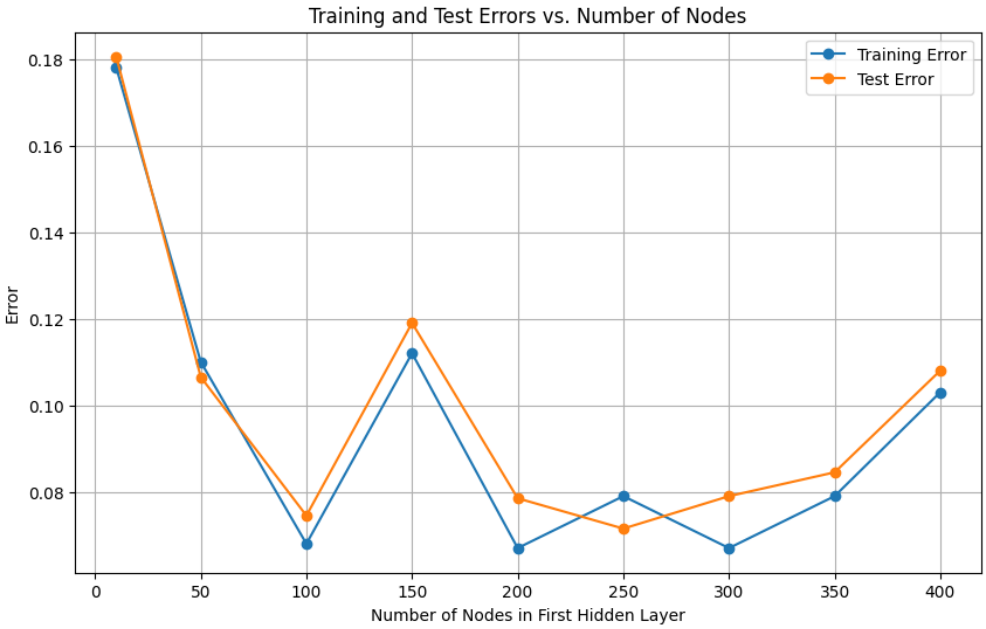

Figure 3: Neural Network Experiment 1 - Number of Nodes

3.3.3: Summary

The results demonstrate the importance of selecting appropriate hyperparameters for neural networks to balance model complexity and generalization performance. The optimal configuration identified through cross-validation (100 nodes, 'relu' activation) achieves a good balance between underfitting and overfitting. The experiments highlight the impact of varying the number of nodes and the number of epochs on model performance, providing insights into the trade-offs involved in training neural networks.

Part 4: Model Comparison

4.1: Methods

In this section, we compare the performance of three optimally tuned classification models: a linear SVM, a Gaussian kernel SVM, and a neural network. Each model was trained on the full training set using the best hyperparameters identified in previous sections. The test errors were computed on the test set, and confidence intervals at a 95

4.2: Results and Discussion

The results of the comparison showed that the test error for the optimally tuned linear SVM was 0.0790, with a 95

4.3: Interpretation and Conclusion

Given the overlapping confidence intervals, all three models demonstrated similar performance on this dataset, despite their differences in underlying assumptions and complexity. The linear SVM, though relatively simple, remained competitive and effective, although it may struggle when decision boundaries are highly nonlinear. The Gaussian kernel SVM provided greater flexibility in decision boundaries and led to slightly better performance. The neural network, capable of learning complex representations, required careful tuning to prevent overfitting but still produced test errors comparable to those of the SVM models. These findings emphasize the importance of hyperparameter tuning in achieving optimal performance. Although

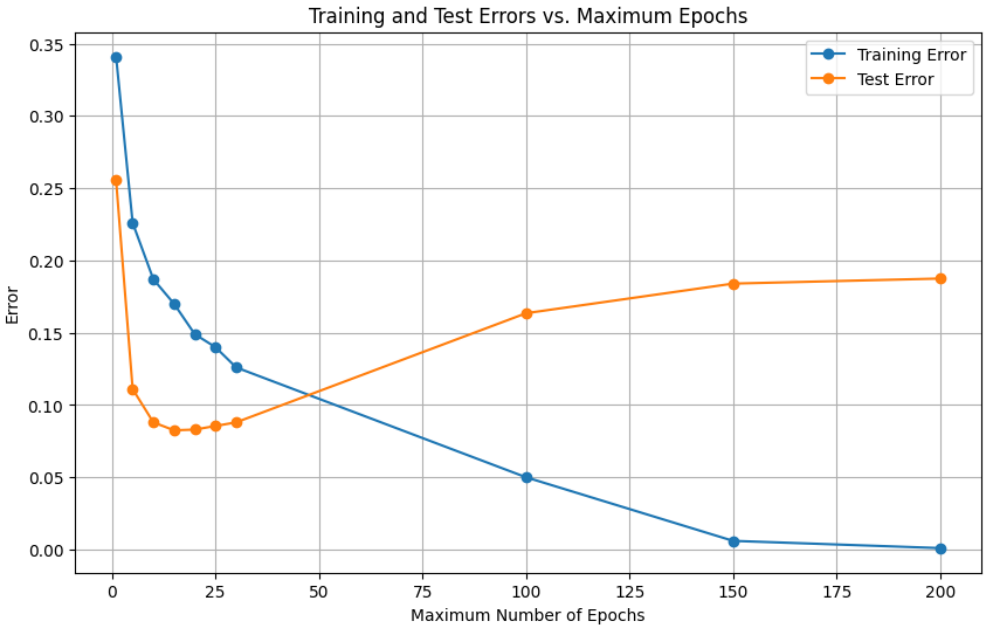

Figure 4: Neural Network Experiment 2 - Maximum Number of Epochs

more complex models can theoretically provide superior results, the marginal gains in test error observed in this study suggest that well-tuned simpler models, such as the linear SVM, can still remain competitive. Future work could explore additional techniques such as feature engineering, ensemble methods, or alternative regularization strategies to further improve classification performance.