Data Mining – Assignment 2

(SENG 474)

Connor Ebert – V00930043

October 25, 2023

Introduction

During this assignment, the exploration of hyperparameter configuration for Linear SVMs, Gaussian SVMs, and Neural Networks was attempted. The performance of these algorithms was evaluated on the Fashion-MNIST dataset. The scikit-learn library was used for all machine learning algorithms, and the matplotlib library was used for all graphical generation. During data preprocessing, noise was added to the training data at a value of 20%, meaning that 20% of the training labels were flipped. Use of a logarithmic scale for testing different hyperparameter ranges was frequently used during testing, the formula for a regularization parameter C specifically was given a C_0 and β the C value was $C_0\beta^i$ for values of $i \in \{1, 2, 3, \dots k\}$. Any mention of a beta value in this text is referencing this formula.

1.0 Linear SVM

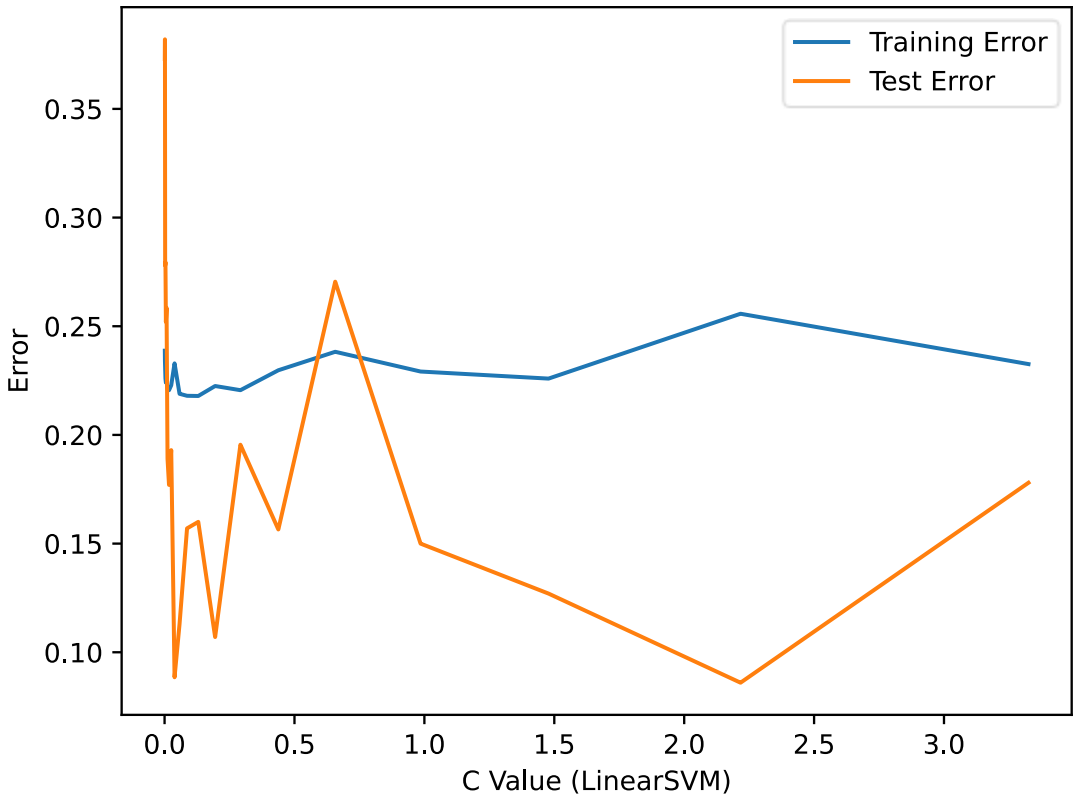
The first part was tuning the hyperparameter configuration for a linear SVM. The only hyperparameter that was considered was C, the regularization parameter. During the experiment the demonstration of both underfitting and overfitting was attempted in order to get the full range of C value potential.

1.1 Initial C Value

Initially, C was varied in a range between 0.01 and 50 with 10 intervals in between on a logarithmic scale with a β value of 1.5. However, it was soon discovered that the larger values of C had a negligible difference between them, so ultimately it was decided to start at a smaller C value of 0.001 and scale only to just over 3.

1.2 Fine Tuned C Value

The final C range that was decided on started at 0.001 and moved up on a logarithmic scale for 20 potential values with a β value of 1.5. This range was chosen in order to show both over and underfitting while still maintaining a reasonable scale.



As can be seen above, the value of the training error stays fairly constant, however, in comparison the test error fluctuates quite a bit. The values of underfitting can be seen in the C range between 0.001 0.566. However, there is a significant drop at 0.00225. After the C value reaches 2.2168 (the overall optimal value for C), we see that the test error begins to trend upwards, this trend is maintained for greater values of C and it is at this point that overfitting occurs. However, a noticed difference between the SVMs and the decision trees analyzed in the previous assignment, is that overfitting does not cause a radical drop in training error.

2.0 Gaussian SVM

For the Gaussian SVM, two hyperparameters were considered, the regularization value C and the bandwidth value γ. To get the best overall value, C was varied in a range starting at 1 and moving up logarithmically with a β value of 1.5 for 10 iterations. Then, for each C value gamma was varied in a range from 0.01 and moving up logarithmically with a β value of 1.5 for 10 iterations as well. In this way, each combination of C and γ were compared using k-fold cross validation with a k value of 5 (this value was chosen to speed the tuning process). From there, the optimal values of γ were found for each C value, and all that was left was to compare those optimal pairings. Again, using k-fold cross-validation, the optimized pairings were compared to find the best tuned Gaussian SVM.

2.1 Best C, γ Pair

The best (C, γ) pairings found during the initial step are as follows:

(1.0, 0.0225),

(1.5, 0.0225),

(2.25, 0.015),

(3.375, 0.01),

(5.0625, 0.01),

(7.59375, 0.01),

(11.390625, 0.01),

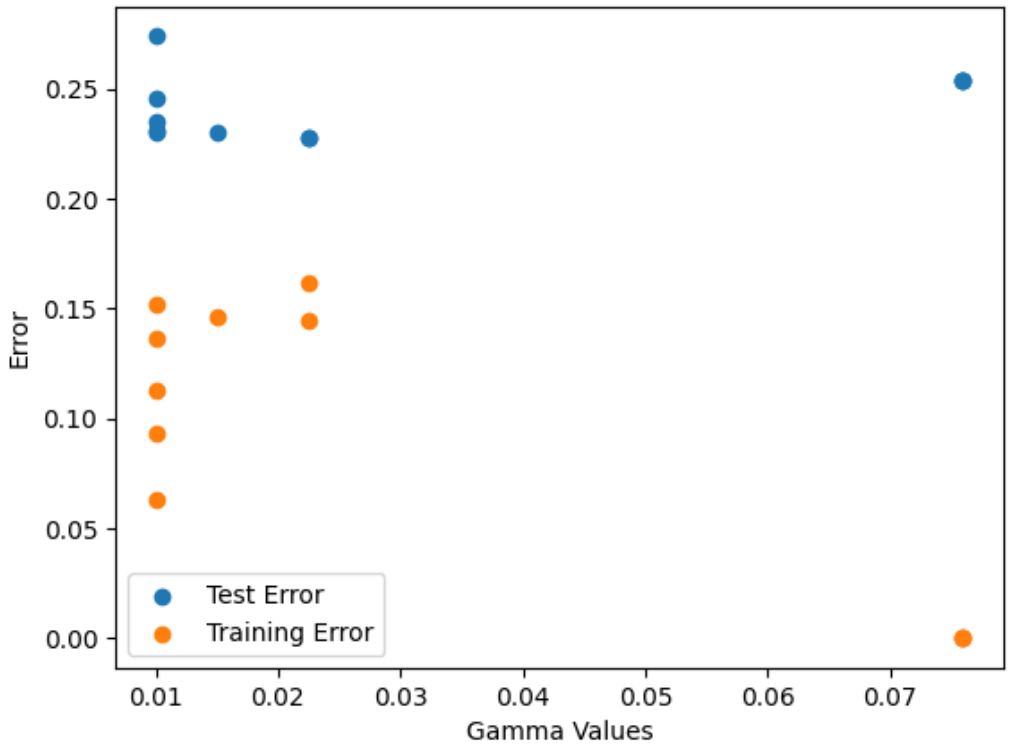
(17.0859375, 0.01),

(25.62890625, 0.0759375),

(38.443359375, 0.0759375),

(57.6650390625, 0.0759375).

It was noted that some gamma values were optimal for multiple values of C. The results of the k-fold cross validation are below.



Looking at the distribution of the scatter plot, we see that underfitting occurred with all the lowest values of gamma (0.01). As gamma increased test error decreased until reaching a minimum value at the C, γ pairing of (1.5, 0.0225). After that we see definitive examples of overfitting all three of the gammas with the value 0.0759375 achieving the lowest test error and the highest training error (on the plot those three points have the same error value, so they are overlapping at those points).

2.2 Comparing to Linear SVM

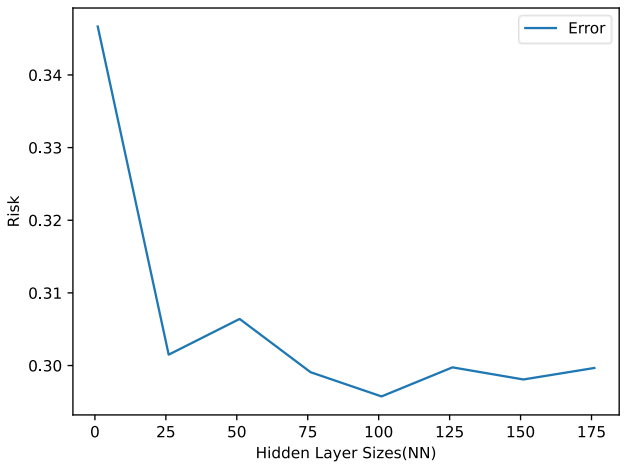
One difference between the Linear and Gaussian SVMs is the training error. Gaussian SVM had a training error that peaked at 1.63 and had lowest values of close to 0. On the other hand the Linear SVM training error stayed in between 0.20 and 0.25. In terms of test error, the Linear SVM showed a much greater range of values than the Gaussian SVM. Gaussian stayed between 0.223 and 0.264, while Linear SVM ranged between much less than 0.1 and greater than 0.35.

3.0 Neural Networks

The next step was to tune a neural network. The one chosen was the MLPClassifier from scikit-learn. To tune the classifier, first a reasonable set of hyperparameters had to be constructed. To do this each combination of three hyperparameters (leaving the rest as the scikit-learn defaults), hidden layer size, number of hidden layers, and choice for non-linearity (or activation) was tested using k-fold cross-validation. The configuration that produced the minimum risk would then be our hyperparameter configuration of choice. After testing was completed, it was revealed that the defaults from scikit-learn were the best apart from activation, which was changed from 'relu' to 'identity'. The values of hidden layer size (100) and hidden layer number (1) proved to be the most effective so were left as default. After this, the effects of varying a single hyperparameter in isolation were tested. To do this, the same three hyperparameters that were tuned previously were varied in isolation to see the effects on test error.

3.1 Hidden Layer Size

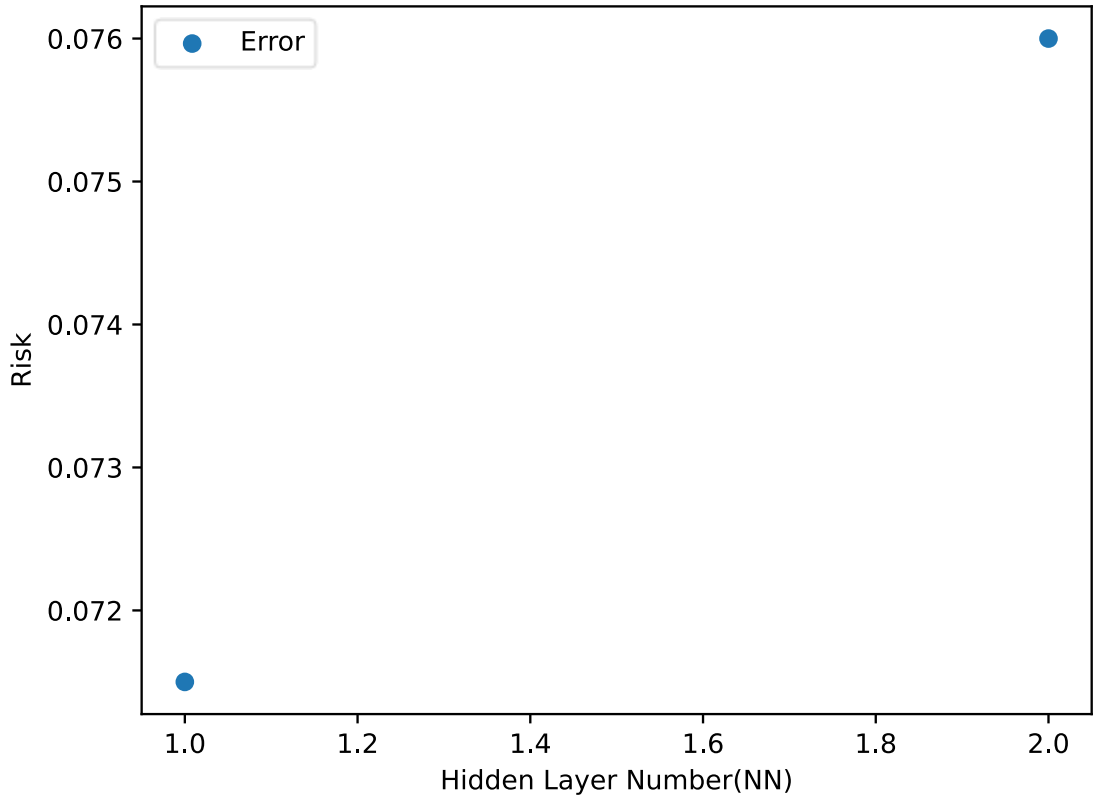
Hidden layer size was varied in a range from 1 to 201 by increments of 25. The range was chosen as it was the minimum that captured the effects of both over and underfitting while still maintaining a reasonable scale for visibility purposes.



Looking at the graph, we see that the error is minimized at 101 (the value closest to the default value of 100). We see that before 100 error trends down as layer size increases, indicating underfitting for those lower values. Error then trends up again after 100 as overfitting begins to occur.

3.2 Number of Hidden Layers

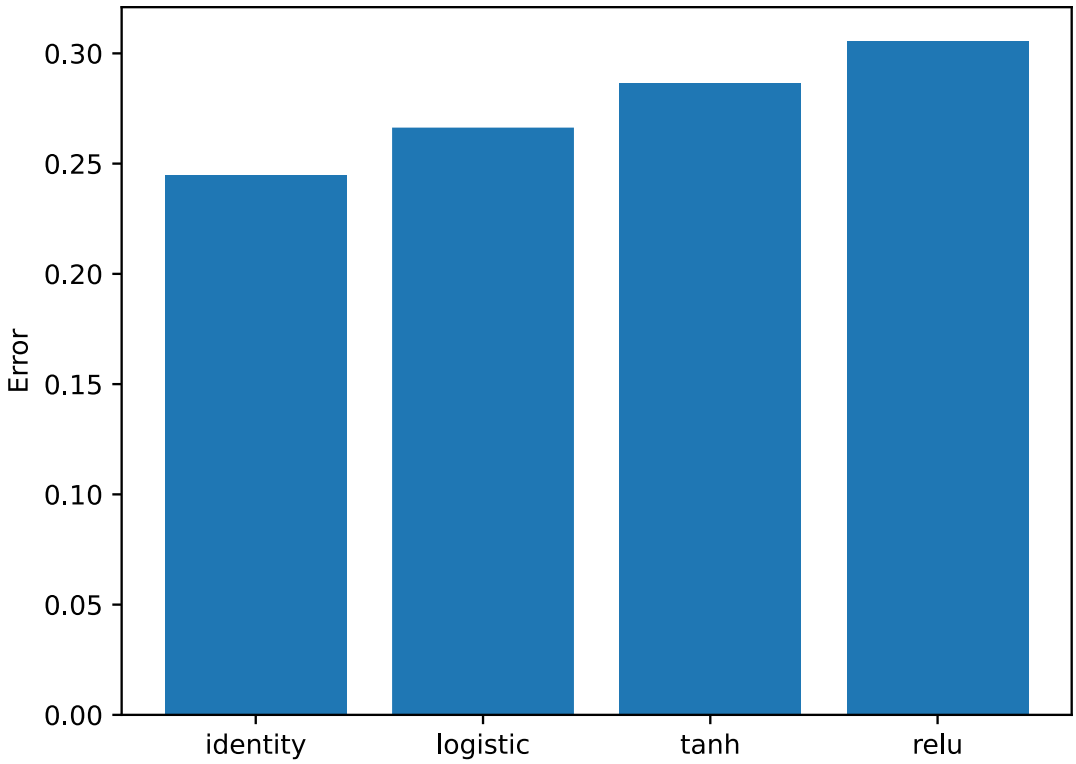
Only 2 values were considered for number of hidden layers, one and two.



Looking at the graph we see that the second layer causes an uptick in error, however the change is not that drastic. It was noted that the additional layer of complexity coming with a second hidden layer was more than in required for the dataset.

3.3 Non-Linearity

Four choices for non-linearity were available to test using the scikit-learn library: identity (f(x) = x), logistic ($f(x) = 1/(1 + e^{-x})$), tanh ($f(x) = \tanh(x)$), and relu ($f(x) = \max(0, x)$).



This was the only parameter that had worse performance on the default (which was relu) than on a tuned alternative (identity). Additionally, relu was the worst overall performer. Identity performed the best, indicating that adding non-linearity to the algorithm was unnecessary in this case as identity is the equivalent of having no activation function at all.

4.0 Comparing the Methods

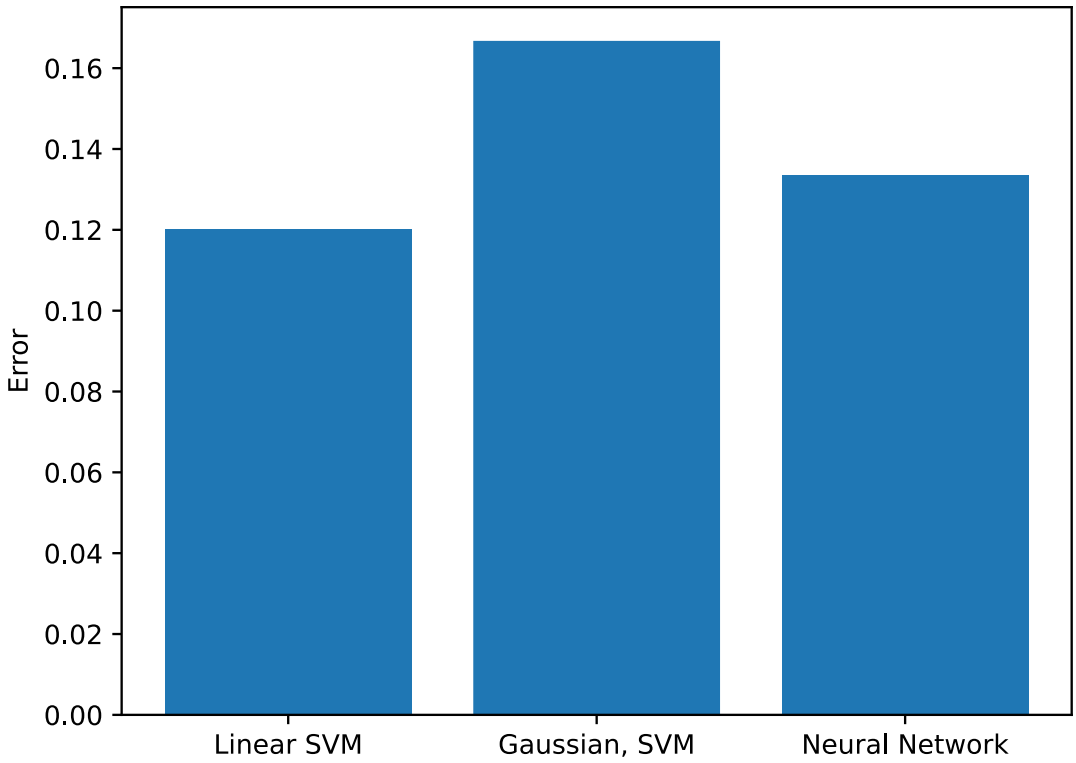
The optimal hyperparameter configuration for each algorithm are as follows:

Linear SVM: C value = 2.2168378200531005

Gaussian SVM: C value = 1.5, γ value = 0.0225

Neural Network: Activation = 'identity', Hidden Layer Number = 1, Hidden Layer Size = 100

Each of these algorithms was trained on the entire training set, then compared on the test set by directly comparing their test errors.



Looking at the graph, we see that Linear SVM has the lowest error, followed by Neural Network, and finally Gaussian SVM. There is potential that the Neural Network or Gaussian SVM are overly complex for the classification problem presented, and as such the Linear SVM is outperforming them. However, the errors are all very close, within around 0.04 of each other, and given a test set of 2000 the confidence intervals around them would likely overlap.