Dustin McAfee

Project 4 Report

Fall 2018

1 Objective

There is a dataset of email attributes, 4601 observations each with 55 continuous real attributes, 2 continuous integer attributes, and one binary categorical attribute. The objective is to predict the binary categorical attribute, 'Spam', by training an online artificial neural network with a softmax output layer. There are many hyper-parameters to tune: learning rate, number of neurons per hidden layer, and number of hidden layers. Average performance metrics such as confusion matrices and F1 scores are computed and shown for different K-Fold cross-validations of these hyper-parameters. The optimal hyper-parameters are chosen and run on the testing dataset and the performance metrics are shown. PCA is also performed on the entire dataset to reduce the dimensions in an attempt to increase the neural network performance on the testing dataset.

2 Data Preprocessing

There are no missing values from the data, and 58 total dimensions (57 not including the binary categorical attribute). The last column of the dataset represents the categorical attribute, and the rest of the columns are z-normalized. The dataset is split into separate testing and training datasets, in which the testing dataset represents about 13% (600) of all observations (See files "TestingData.txt" and "TrainingData.txt").

3 Methods

K-fold cross validation is implemented and performed with 5 groups (folds) of training sets (See function "n_fold_split" in "net.py"). Each K-fold cross-validation training set has about 920 observations, which correspond to about 3680 observations for validation of each K-fold. The artificial neural network is tuned for the learning rate, number of neurons per hidden layer, and number of hidden layers. For each one of these hyper-parameters being cross-validated, the others are held constant. The maximum number of epochs is always held constant at 1000 (since each K-fold of the training data takes several minutes with this many epochs, this number seemed like a practical maximum), and the cutoff tolerance for the sum of squared errors for the output of each epoch is 0.001. The error is printed to screen for each 10 epochs, and a keyboard interrupt continues from the back propagation loop. While cross-validating, the learning rate is held at 0.1, the number of neurons per hidden layer is held at 3, and the number of hidden layers is held at 2. The first hyper-parameter tuned is the learning rate, which is tested from 0.01, 0.1, 0.2, 0.3, and 0.5. Then, the number of neurons per hidden layer, which is tested in the range of 1 to 6. Last, is the number of hidden layers, which is tested in the range of 1 to 5 (2 to 6 total layers, including the output layer). A confusion matrix is shown from the average performance metrics of the K-fold cross-validation datasets for each hyper-parameter. A final confusion matrix is produced for the testing dataset

with the chosen optimal hyper-parameters.

3.1 Learning Rate

For a learning rate of 0.01, the average of the 5 folds yields 80.17% accuracy, with 57.02% sensitivity, 88.58% precision, 95.19% specificity, and a F1 score of 69.20%. The average sum of squared errors for the output of each K-fold is 369.79 after each fold reaches 1000 epochs. The confusion matrix for this is shown below, in Figure 1.

		Prediction outcome		
		n	**p**	**Total**
Actual value	**n′**	530.6	26.8	557.4
	p′	155.6	207.0	362.6
	Total	686.2	233.8	

Figure 1: Confusion Matrix for Learning Rate = 0.01

This has a very high false negative rate, and a rather too high false positive rate. During this run, the sum of squared errors monotonically decreases. Because of this, it is likely, that if the neural network is trained for a longer time (more epochs), then there would be a better fit for this data. Unfortunately, more epochs are not practical given the complexity of the algorithm (See file "net.py").

For a learning rate of 0.1, the average of the 5 folds yields 78.63% accuracy, with 50.25% sensitivity, 91.74% precision, 97.01% specificity, and a F1 score of 64.71%. The average sum of squared errors for the output of each K-fold is 382.72 after each fold reaches 1000 epochs. The confusion matrix for this is shown below, in Figure 2.

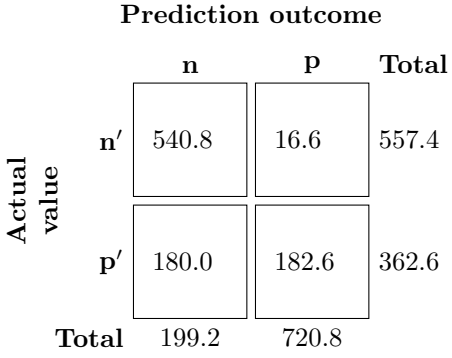

Figure 2: Confusion Matrix for Learning Rate = 0.1

This run has less false positives, which means there are less important emails that are misclassified as spam. It has slightly less accuracy, less sensitivity, but higher precision and specificity. Like the last architecture, the sum of squared errors monotonically decreased during the entire execution on this dataset.

For a learning rate of 0.2, the average of the 5 folds yields 79.02% accuracy, with 51.61% sensitivity, 91.29% precision, 96.80% specificity, and a F1 score of 65.85%. The average sum of squared errors for the output of each K-fold is 406.40 after each fold reaches 1000 epochs. The confusion matrix for this is shown below, in Figure 3.

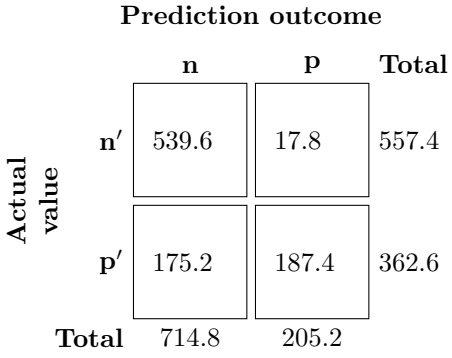

Figure 3: Confusion Matrix for Learning Rate = 0.2

For a learning rate of 0.3, the average of the 5 folds yields 78.28% accuracy, with 50.16% sensitivity, 90.44% precision, 96.51% specificity, and a F1 score of 64.35%. The average sum of squared errors for the output of each K-fold is 437.23 after each fold reaches 1000 epochs. The confusion matrix for this is shown below, in Figure 4.

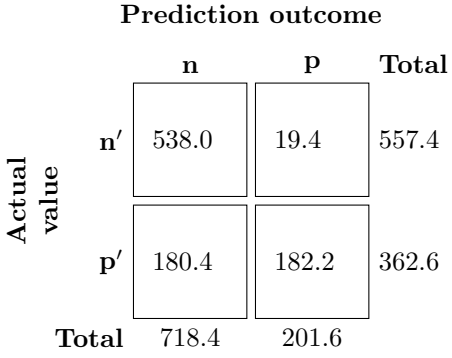

Figure 4: Confusion Matrix for Learning Rate = 0.3

For a learning rate of 0.5, the average of the 5 folds yields 70.48% accuracy, with 27.71% sensitivity, 90.55% precision, 98.09% specificity, and a F1 score of 42.43%. The average sum of squared errors for the output of each K-fold is 509.11 after each fold reaches 1000 epochs. The confusion matrix for this is shown below, in Figure 5.

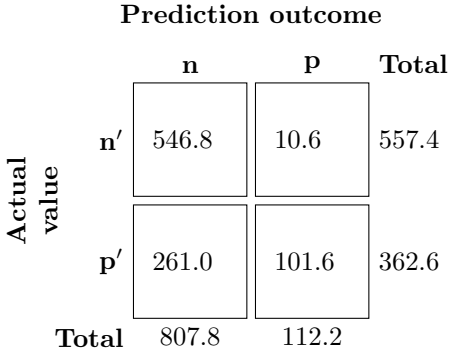

Figure 5: Confusion Matrix for Learning Rate = 0.5

The optimal learning rate chosen is 0.1, since, other than with a learning rate of 0.5, it has the lowest false positive rate with better performance metrics than most of the others (though it is very close to the learning rate of 0.01 with a slightly higher accuracy and precision). It is important to note, that each execution of the neural network with variable learning rates monotonically decreased and did not necessarily converge to a specific sum of squared errors value (See files "l_rate_01," "l_rate_1," "l_rate_2," "l_rate_3," "l_rate_5," and "l_rate_8").

3.2 Number of Neurons per Hidden Layer

In this subsection, the hyper-parameter tuned is the number of neurons per hidden layer. Recall that for cross-validation, the other hyper-parameters are held constant, and the number of hidden layers is held at 2. For 1 neuron per hidden layer, the average of the 5 folds yields 80.39% accuracy, with 55.55% sensitivity, 91.27% precision, 96.52% specificity, and a F1 score of 69.01%. The average sum of squared errors for the output of each K-fold is 392.74 after each fold reaches 1000 epochs. The confusion matrix for this is shown below, in Figure 6.

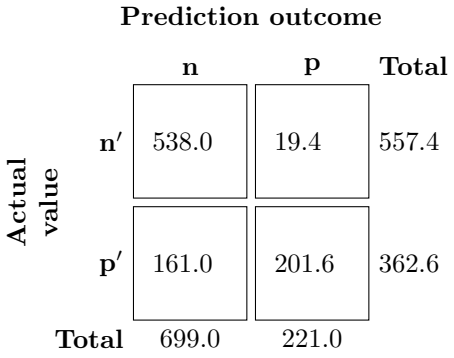

Figure 6: Confusion Matrix for Number of Neurons per Hidden Layer = 1

For 2 neurons per hidden layer, the average of the 5 folds yields 79.65% accuracy, with 53.39% sensitivity, 91.35% precision, 96.65% specificity, and a F1 score of 67.10%. The average sum of squared errors for the output of each K-fold is 385.92 after each fold reaches 1000 epochs. The confusion matrix for this is shown below, in Figure 7.

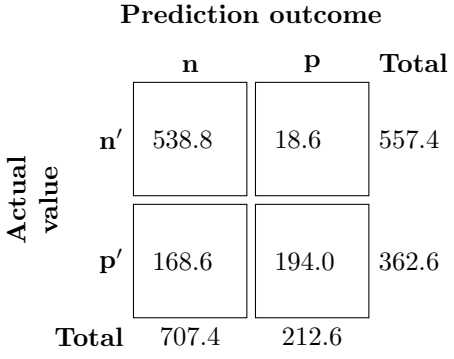

Figure 7: Confusion Matrix for Number of Neurons per Hidden Layer = 2

For 3 neurons per hidden layer, refer back to Figure 2 in Section 3.1. For 4 neurons per hidden layer, the average of the 5 folds yields 78.87% accuracy, with 51.03% sensitivity, 91.57% precision,

96.90% specificity, and a F1 score of 65.33%. The average sum of squared errors for the output of each K-fold is 381.74 after each fold reaches 1000 epochs. The confusion matrix for this is shown below, in Figure 8.

		Prediction outcome		
		n	**p**	**Total**
Actual value	**n′**	540.2	17.2	557.4
	p′	177.2	185.4	362.6
	Total	717.4	202.6	

Figure 8: Confusion Matrix for Number of Neurons per Hidden Layer = 4

For 5 neurons per hidden layer, the average of the 5 folds yields 79.15% accuracy, with 51.88% sensitivity, 91.46% precision, 96.83% specificity, and a F1 score of 66.06%. The average sum of squared errors for the output of each K-fold is 382.70 after each fold reaches 1000 epochs. The confusion matrix for this is shown below, in Figure 9.

		Prediction outcome		
		n	**p**	**Total**
Actual value	**n′**	539.8	17.6	557.4
	p′	174.2	188.4	362.6
	Total	714	206	

Figure 9: Confusion Matrix for Number of Neurons per Hidden Layer = 5

For 6 neurons per hidden layer, the average of the 5 folds yields 79.17% accuracy, with 52.20% sensitivity, 91.07% precision, 96.66% specificity, and a F1 score of 66.20%. The average sum of squared errors for the output of each K-fold is 382.93 after each fold reaches 1000 epochs. The confusion matrix for this is shown below, in Figure 10.

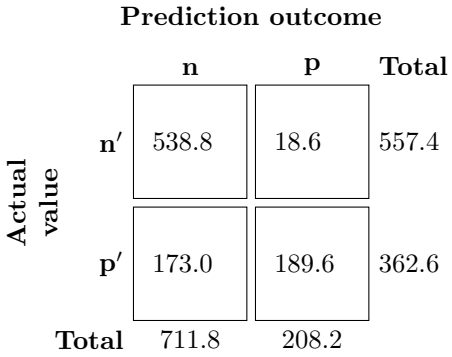

Figure 10: Confusion Matrix for Number of Neurons per Hidden Layer = 6

The metrics in this section do not vary much; there is not much significance in the difference in performance. So, the optimal parameter value chosen here is 3 neurons per hidden layer, since part of the goal here is to have a simple predictive model, and with 3 neurons per hidden layer, the performance metrics yields the lowest false positive rate.

3.3 Number of Hidden Layers

In this subsection, the final hyper-parameter is tuned: The number of hidden layers. Recall, the number of neurons per hidden layer is held at 3. For 1 hidden layer, the average of the 5 folds yields 75.91% accuracy, with 44.01% sensitivity, 89.08% precision, 96.41% specificity, and a F1 score of 58.11%. The average sum of squared errors for the output of each K-fold is 462.72 after each fold reaches 1000 epochs. The confusion matrix for this is shown below, in Figure 11.

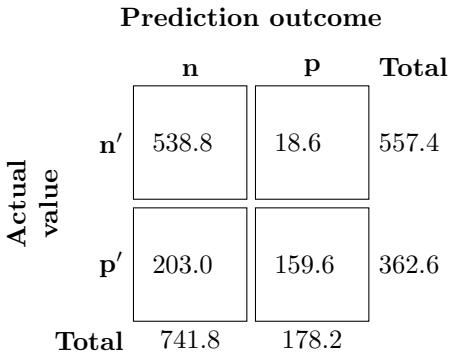

Figure 11: Confusion Matrix for Number of Hidden Layers = 1

For 2 hidden layers, refer back to Figure 2 in Section 3.1. For 3 hidden layers, the average of the 5 folds yields 80.28% accuracy, with 56.51% sensitivity, 89.64% precision, 95.68% specificity, and a

F1 score of 69.22%. The average sum of squared errors for the output of each K-fold is 402.96 after each fold reaches 1000 epochs. The confusion matrix for this is shown below, in Figure 12.

		Prediction outcome		
		n	**p**	**Total**
Actual value	**n′**	533.4	24.0	557.4
	p′	157.4	205.2	362.6
	Total	960.8	229.2	

Figure 12: Confusion Matrix for Number of Hidden Layers = 3

For 4 or more hidden layers, the sum of squared errors for the output of each epoch quickly converges to a high number (around 930 to 1000), and the accuracy drops to 50% or below. The confusion matrices and exact metrics for 4 and 5 hidden layers are not shown here, as it suffices to say that they do not perform well. To this end, the optimal value for the number of hidden layers is chosen to be 3. The performance metrics between, 2 and 3 hidden layers are very similar, but 3 is chosen for having higher accuracy and sensitivity, and a lower false positive rate.

4 Testing Dataset and Principal Component Analysis

The optimal hyper-parameters are chosen from the previous section and run on the testing dataset. These values are 0.1 learning rate, 2 hidden layers, and 3 neurons per hidden layer (See Figure 2 in Section 3.1 for metrics on the training dataset). The network is trained on the training dataset and the testing dataset is propagated forward through the neural network to make predictions. This run yields 86.67% accuracy, with 94.98% sensitivity, 76.95% precision, 81.16% specificity, and a F1 score of 85.02%. The sum of squared errors for the output is 401.02 after 1000 epochs. The confusion matrix for this is shown below, in Figure 13.

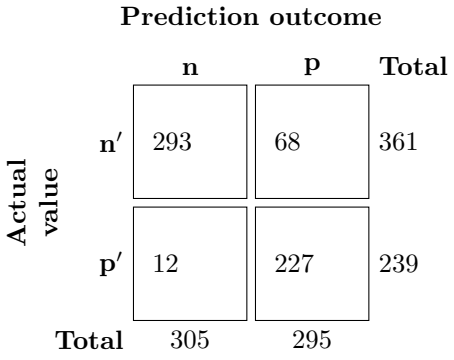

Figure 13: Confusion Matrix for Testing Dataset

This seems to have a higher false positive rate than foreseen, but the overall metrics of this is much higher than that of the training dataset.

Principal component analysis is performed to reduce the data to a few dimensions in an attempt to increase performance. The first three principal components cover about 99.99% of the variance of the dataset, and so, the testing dataset is projected to the first three principal components. The optimal hyper-parameters are chosen to run the neural network on this projected dataset (Learning rate of 0.1, 2 hidden layers, and 3 neurons per hidden layer). The accuracy of this run is 66.50%, with 89.12% sensitivity, 54.90% precision, 51.52% specificity, and a F1 score of 67.94%. The confusion matrix for this is shown, below, in Figure 14.

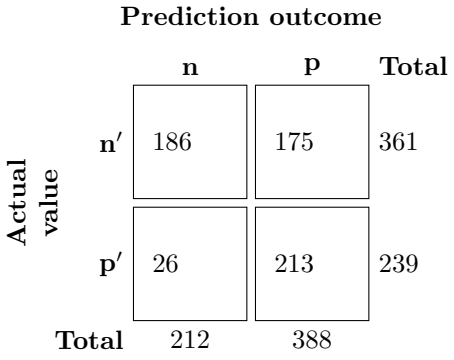

Figure 14: Confusion Matrix for Testing Dataset Projected to first 3 PCs

This is a major decrease in performance. In an attempt to increase the performance on this projected dataset, the learning rate is dropped to 0.02, the neurons per layer is set at 2, and the number of hidden layers is set to 1. This will simplify the model for the now three dimensional dataset. The confusion matrix for this is shown below, in Figure 15. The accuracy of this run is

69.50%, with 67.36% sensitivity, 60.53% precision, 70.91% specificity, and a F1 score of 63.76%.

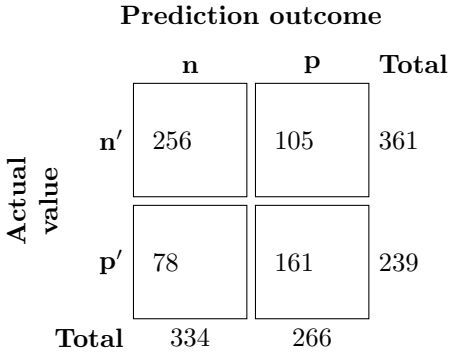

Figure 15: Confusion Matrix for Testing Dataset Projected to first 3 PCs

Overall, projection onto the first three principal components did not help the performance of the neural network. Of course, only a very small number of hyper-parameters were tested, and Principal Component Analysis could very well increase overall performance with a specific architecture of the neural network.

5 Results

The performance metrics did not vary much during cross-validation of the hyper-parameters, which makes it difficult to choose optimal values. When choosing the optimal hyper-parameters to reduce false positives, there is an unexpectedly high false positive rate in the testing dataset. To this end, the optimal hyper-parameters chosen are 0.1 learning rate, 2 hidden layers, and 3 neurons per hidden layer (coincidentally, this is the first confusion matrix, Figure 2). The first three principals components cover about 99.99% of the dataset, and the dataset is projected to its first 3 principal components in an attempt to increase performance. Unfortunately, this decreased performance in the testing dataset.

6 Conclusion

An artificial neural network is implemented and its hyper-parameters are cross-validated using K-fold cross validation. The performance of each cross-validation did not vary much, and each had a significantly high false negative rate. The neural network is trained on the training dataset with chosen optimal hyper-parameters, and the testing dataset is forwarded through the network, which leads to significantly higher performance metrics than any of the K-fold executions. Performance metrics would likely vary more or even increase if the program was allowed to run further than 1000 epochs. Unfortunately, the complexity of the algorithm makes further testing impractical in the time period of this writing.