Jessica Miron

BME 598: Applied Programming

Homework 12

Executive Summary

Tuberculosis, or TB, is a deadly disease that poses a great public health challenge especially when it comes to diagnosis. Building a classifier that is able to differentiate between TB and non-TB patients (both healthy and differential diagnoses) from whole blood would enable greater treatment due to improved diagnosis. From the limited dataset of whole-blood genome-wide transcriptional profiles of patients with TB, active sarcoid, non-active sarcoidosis, and no disease, a beginning classifier was created but overfitting was a problem. The analysis seeks to fix the overfitting of the model and obtain both a loss and validation loss less than 0.01. The steps to do this are shown below.

1. Download the dataset and Python packages
2. Select out only the disease states being evaluated
3. Select only the values of the genes
4. Select the top 1000 genes with the greatest variance
5. Scale the full dataset based on the combined training and validation sets
6. Map the labels to 0 for TB and 1 for every other disease state
7. Create the new binary model
8. Tune the model hyperparameters to reduce training and validation loss
9. Plot training versus validation loss and accuracy
10. Write the biological interpretation

Biological Interpretation

The beginning classifier is a multiclass classifier which overfits the data. The model classifies into four categories using two dense layers and two dropout layers. The first dense layer has 600 neurons, the first dropout layer has a rate of 0.3, the second dense layer has 200 neurons, and the second dropout layer has a rate of 0.2. After training the model with 100 epochs, an accuracy of 0.9701, loss of 0.1969, validation accuracy of 0.8333, and a validation loss of 0.4453 were obtained. The loss and accuracy per epoch graphs are shown in figure 1. The classifier is over fitting the model due to the loss and accuracy continuing to improve while the validation loss and validation accuracy stagnant or get worse. On the loss versus epoch graph, the training loss continues to decrease across epochs reaching below 0.2 while the validation loss plateaus at 20 epochs and remains around 0.5. Overfitting is when the model learns too much noise because the model is too complex for the amount of data present [1]. The initial model shows this with the gap between the training and validation losses.

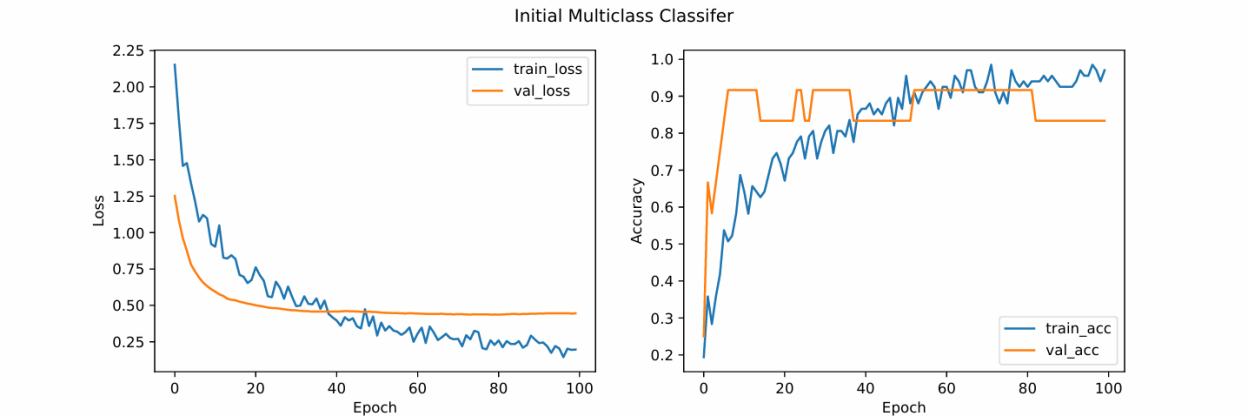

Figure 1: Loss vs epoch and accuracy vs epoch for the initial multiclass classifier. Training metrics are shown in blue. Validation metrics are shown in orange. The graphs show the model is overfitting the data due to the validation loss and accuracy stagnating and getting worse while the training loss and accuracy continue to improve.

To correct the overfitting, two strategies are used. First, changing from a multiclass to a binary classifier. Due to the limited amount of data, the multiclass classifier is attempting to create too many categories from too few datapoints causing it to just learn the individual datapoints instead of patterns across points. By changing to a binary classifier, there will be more data in the non-TB category giving a better chance of correctly classifying. This decreases the model complexity giving the model less chances of learning the noise of the training data [2]. Second, increasing the number of epochs to give the model more time to train to the data. The larger amount of epochs pushes both the training loss and the validation loss lower to get a better result. Learning rate optimization was also performed to ensure the correct learning rate was being used. Figure 2 shows the learning rate optimization with rates of 0.01, 0.001, and 0.0001. A learning rate that is too slow takes a long time to reach the solution and more steps are needed [3]. A learning rate that is too fast will never reach the solution because the drastic updates cause the solution to missed [3]. The correct learning reaches the solution in an acceptable amount of time [3]. A learning rate of 0.01 is too fast as the validation loss reaches its minimum early and stops improving. A learning rate of 0.0001 is too show as it takes too many steps and would take too long to reach the solution. A learning rate of 0.001 was chosen as the correct learning rate as the validation loss has the same curve as the validation loss and continues to improve as the training loss improves.

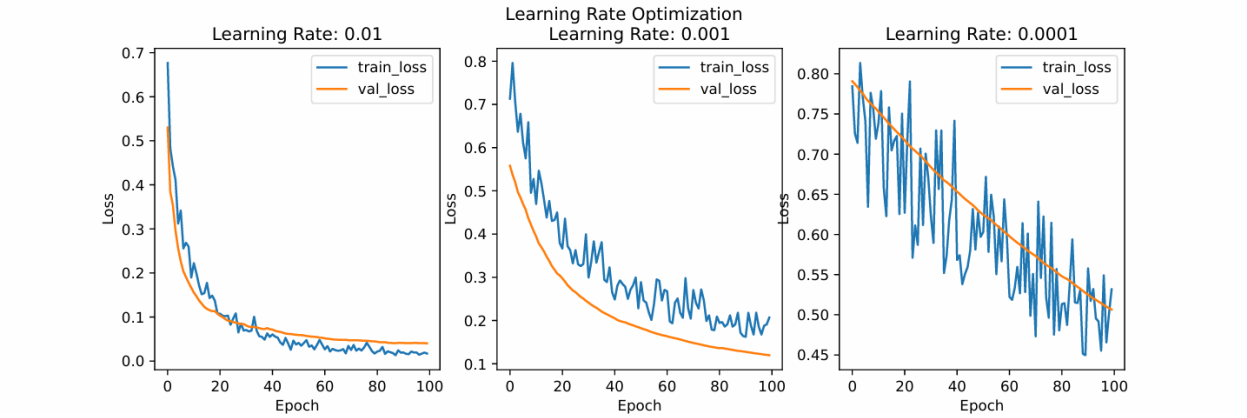

Figure 2: Learning rate optimization. The same model was run three times with three different learning rates of 0.01, 0.001, and 0.0001 for 100 epochs. Larger rates are faster. Learning rate of 0.001 was chosen as the correct learning rate due to following the curve of training loss and continuing to improve.

To implement these changes, a few changes were made. First, the labels were changed from one-hot encoded to 0 for non-TB and 1 for TB. Next, the last dense layer was changed from 4 to 1 neurons and from sigmoid to softmax activation. In the compiler the loss function was changed from categorical_crossentropy to binary_crossentropy. The learning rate was set to 0.001 as found by the learning rate optimization. The last change was running 3500 epochs which was chosen from running the model multiple times while tuning and finding the number of epochs where the loss and validation loss stopped improving.

The final classifier performed much better than the initial classifier and overfitting was greatly reduced. Figure 3 shows the same loss and accuracy graphs from the initial classifier but with the improved model. For loss, both metrics were greatly improved as now the training loss begins at around 0.8 and the validation loss begins at around 0.4 while the previous model was at around 2.2 and 1.25 respectively. The ending losses are also much improved with an end training loss of 0.0022 and an end validation loss of 0.0122. The biggest signal that the overfitting was fixed is that the validation and training losses are much closer and both continued to improve as training went on. For the improved model the losses were 0.01 while the initial model was 0.2428 away. Looking at the accuracy graph, the training accuracy gets to 1 at around 1000 epochs with a few reductions that are corrected further on. Validation accuracy is at 1 throughout the whole training process. These metrics show that the new model fixed the overfitting and improved the training and validation loss and accuracy.

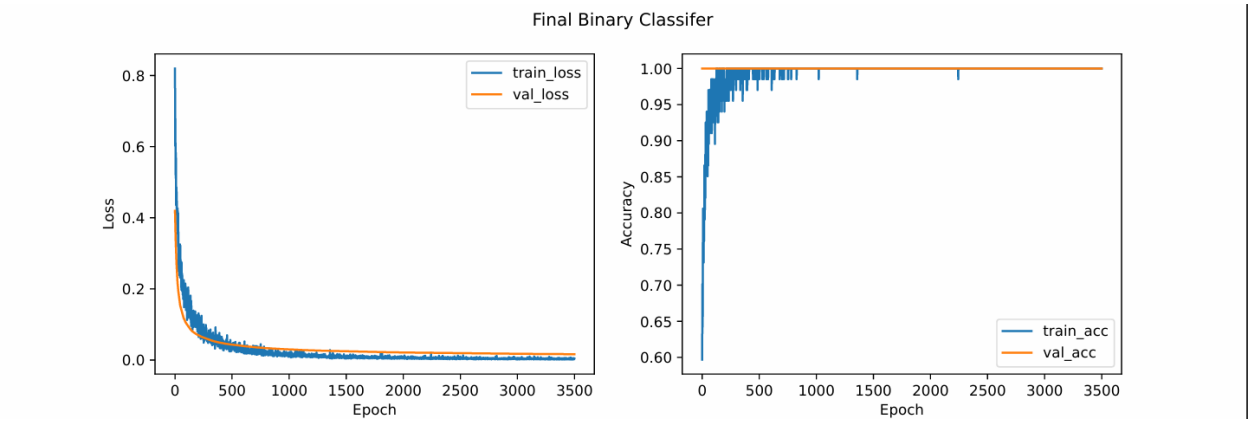

Figure 3: Loss vs epoch and accuracy vs epoch for the final binary classifier. Training metrics are shown in blue. Validation metrics are shown in orange. The graphs show the model fixed the overfitting and greatly improved the training and validation loss and accuracy due to the validation metrics remaining close to the accuracy metrics.

If more data could have been generated, the approach to fix the overfitting would have been different. Data from all disease states to make equal sized groups would be the most helpful. Equally the amount of data in each group would get rid of the class imbalance and enable the making of a multiclass classifier. By equaling the amount of data in each group and increasing the amount of data in each group, a multiclass classifier would have enough information to be

more complex and not overfit. The study should be structured the same so the format and content match the current data. If a different set up is used there could be differences within the training data that could influence the model unequally across groups and create false information.

References

[1] R. Holbrook, “Overfitting and Underfitting,” Kaggle, https://www.kaggle.com/code/ryanholbrook/overfitting-and-underfitting.

[2] “What is overfitting? - overfitting in machine learning explained - AWS,” https://aws.amazon.com/what-is/overfitting/.

[3] A. Bilogur, “Tuning your learning rate,” Kaggle, https://www.kaggle.com/code/residentmario/tuning-your-learning-rate.