Module 4: Critical Thinking

Improving the Accuracy of a Neural Network

Nolan Byrnes

CSC580 – Applying Machine Learning and Neural Networks - Capstone

Colorado State University – Global Campus

Dr. Isaac Gang

October 15th, 2023

Module 5: Critical Thinking

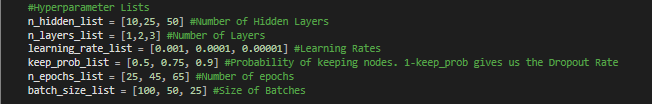
Last week, I developed a model that could predict if a chemical compound is toxic or not. With the accuracy rate of .58 in last week’s model, I needed to see if I could achieve a more accurate model by selecting the best hyperparameters. One way to optimize the hyperparameters is by using the grid search method. Navas explains that with the grid search, we create a “grid of possible discrete hyperparameter values then fit the model with every possible combination” and selecting the best model afterwards (Navas, 2022, para. 25). With the model architecture I used, there were six different hyperparameters to tune and wanted to try out three different values for each of the hyperparameters, which was a total of 729 different combinations of hyperparameters. Another machine learning algorithm that can be used for classification problems is the random forest classifier, instead of a trained model. This paper goes over how I performed hyperparameter tuning and the results, as well as how the optimized model performed against the random forest classifier algorithm.

Hyperparameter Tuning

For hyperparameter optimization, I had to consider what hyperparameter value options to choose, to hopefully come out with a better performing model. The hyperparameter choices that I applied can be viewed in figure 1 below.

Figure 1.

Hyperparameter Lists



Note. This figure displays the list of values used for each of the hyperparameters for the grid search method of hyperparameter optimization .

I thought 45 epochs could have been too many in last week’s project, so I wanted to see what would happen if I decreased the epochs, so I included 25, and 35. Last week’s batch size was 100, and I thought that maybe since the data was not a very large one, I could reduce the batch size and get better results, so I chose the additional options of 50 and 25 for the batch size. The default learning rate when using the Adam optimizer with Keras is at 0.001, so I placed that value as one of the options, since we were using the Adam optimizer (Keras, n.d.). Since I thought the model could have gone with less epochs last week, maybe by decreasing the learning rate, could increase the performance while keeping a higher epoch value. Overall, there were 729 different combinations of hyperparameters for the hyperparameter optimizer to evaluate.

Considering that there were 729 different combinations of hyperparameters, and for each set of hyperparameters, we were getting the average validation accuracy across three trials, the hyperparameter optimizer would be training the model a total of 2,187 times. I needed an efficient way to perform hyperparameter optimization, so I decided to use multiprocessing, since multiprocessing is “useful for CPU-bound processes, such as computationally heavy tasks” (Wong, 2023, para. 6).

To accomplish this, I loaded all the hyperparameter combinations to a queue, and then used 12 cores to consume all the queue items out of the hyperparameter queue, and train with the hyperparameters specified in each queue item. Instead of one model being trained at one time, I had 12, which significantly sped up the hyperparameter tuning process.

After the hyperparameter tuning, I evaluated the weighted classification accuracy of the validation data set. The top score had an average score of 0.74, or 74%. Considering the previous score from last week was 0.65, we saw a 9% increase in accuracy of the validation dataset. The best hyperparameter combinations for the top three combinations can be shown in figure 2 below.

Figure 2.

Best performing hyperparameter combinations

A screenshot of a computer

Description automatically generated

Note. This figure displays the top three combinations of hyperparameter values with the highest average accuracy rate of the validation dataset.

For the best combination of hyperparameters, I looked at the TensorBoard Time series graph to observe how the training of the model performed. Comparing this week’s graphs to last week, the loss time series did not have the same pattern of having clear segments of going up and down in its loss. Instead, the loss seemed to have fluctuated more frequently. By smoothing out the graph, I could see that in all three cases, the loss spiked initially, but sloped downward through the steps.

Since we had three trial runs for each of the hyperparameter combinations, the bold blue lines represent the three separate trials. You can view the time series graph for the model with the best hyperparameter combination in figure 3 below.

Figure 3.

TensorBoard Time Series Graph of Best Hyperparameter Combination

A graph showing a number of data

Description automatically generated with medium confidence

Note. This figure represents the loss throughout training for the three trials of training with the best performing hyperparameter combination.

I trained the model according to the recommendations of the hyperparameter tuning and saw that the model had a 96% accuracy rate on the training data, 71% on the validation data set, and 63% on the test data set. Last week’s test accuracy was at 58%, so we saw an increase of 5% on the test data set. You can view the accuracy of the trained model with optimized hyperparameters in figure 4 below.

Figure 4.

Classification Accuracy of trained model with optimized hyperparameters

A screen shot of a computer

Description automatically generated

Note. This figure displays the accuracy scores after training a model with the best combination of hyperparameters.

Random Forest Classifier

A random forest classifier is a “supervised learning algorithm” that “can be used for both classification and regression problems” (Donges, 2021, para. 3-4). I compared the results of the trained model with optimized hyperparameters to a random forest classifier and saw that the random forest classifier had a 69% accuracy rate on the validation dataset and a 67% accuracy rate on the test data. The results of the random forest classifier can be observed in figure 5 below. The random forest classifier having a higher accuracy rate on the test data set tells me that the random forest classifier would perform better in the real world than the optimized model that I came up with after hyperparameter optimization.

Figure 5.

Classification Accuracy of trained model with optimized hyperparameters

A black screen with white text

Description automatically generated

Note. This figure displays the accuracy scores after training a model with the best combination of hyperparameters.

Donges suggests that using the random forest classifier is great to use early in the model development process, just to see how it performs (Donges, 2021, para. 36). By doing so, you can see if using it provides acceptable results for your particular use case. In this use case, since we were trying to predict toxic compounds, we would want a higher accuracy rate on test data, so it was worth a shot to train a model and use hyperparameter optimization to see if we could get a better performing model.

Conclusion

Hyperparameter tuning by using the grid search method is an expensive process, due to having to retrain a model for every combination of parameters that we are wanting to try, with the end goal of having the best combination of hyperparameters to use to train the model. By including 3 different options, for the six hyperparameters in my machine learning model, there were 729 different parameter combinations, and the random forest classifier still had a better accuracy rate with the test data set. It is possible that by expanding the selections for each of the hyperparameters, we could have found a better combination, leading to a better performing model, but would take more time to complete the hyperparameter optimization.

If I had to choose either the best trained model after my hyperparameter optimization or the random forest classifier, I would have chosen the random forest classifier, since it had a better accuracy rate with the test data set. But since the use case is to predict toxic chemical compounds, a 70% accuracy rate would not be acceptable for the use case, considering that we would be predicting if a chemical compound is toxic or not, and the potential dangers of a false negative.

**REFERENCES**

Donges, W. (2021, July 22). Random forest: A complete guide for machine learning. Built In. <https://builtin.com/data-science/random-forest-algorithm>

Keras Team. (n.d.). Keras documentation: Adam. Keras: Deep Learning for humans. <https://keras.io/api/optimizers/adam/>

Navas, J. (2022, February 8). What is Hyperparameter tuning? Anyscale. <https://www.anyscale.com/blog/what-is-hyperparameter-tuning>

Wong, K. J. (2023, August 4). Multithreading vs. multiprocessing explained. Built In. <https://builtin.com/data-science/multithreading-multiprocessing>