Using Evolutionary Algorithms and Distinctiveness Pruning to Analyse Effects on Brainwave Patterns from Music

Josh Murkett
u6948400@anu.edu.au

Research School of Computer Science, Australian National University

Abstract. This paper looks at using distinctiveness pruning together with evolutionary algorithm hyperparameter tuning to predict what genre of music a person is listening to based on their brainwave patterns. Distinctiveness pruning is a technique where all of the hidden neurons are compared against each other based on their activations for each of the input patterns. Neurons are then pruned if they are not distinctive enough from other neurons based on the distinctiveness threshold. The evolutionary algorithm was used to tune the number of epochs, learning rate, and distinctiveness threshold hyperparameters. The overall performance of the model using the distinctiveness pruning and evolutionary algorithm hyperparameter tuning was found to be very poor as it was only able to achieve an average testing accuracy of 33.91% which is only marginally better than a random guess. This poor performance was likely due to a combination of poorly chosen parameters for the evolutionary algorithm as well as the dataset used being highly limited.

Keywords: Music, Brainwave, Neural Network, Distinctiveness, Pruning, Evolutionary Algorithms, Hyperparameter Tuning

1 Introduction

The task of this paper is to determine the effectiveness of using evolutionary algorithms together with distinctiveness pruning to improve the accuracy of a neural network. This will be done by comparing the performance of a standard neural network with the performance of a neural network that uses evolutionary algorithms and distinctiveness pruning. This task is worth solving because if this approach is found to be effective, then it could become a useful tool to help improve the performance of a neural network.

The dataset for this paper comes from *Brain Melody Informatics: Analysing Effects of Music on Brainwave Patterns* [4]. This paper looks at the effect of different genres of music on brainwave patterns. The authors were able to reach an accuracy of 97.6% for predicting the genre of the music a subject is listening to based on the subject's brainwave patterns. The authors were able to accomplish through using a neural network with genetic algorithm feature selection.

The hidden neuron activation based pruning technique comes from *Indicators of Hidden Neuron Functionality: the Weight Matrix versus Neuron Behaviour* [2]. This paper presents a novel criterion as to which neurons in hidden layers should be pruned. The criterion is that neurons which are not sufficiently distinct from each other should be pruned. The idea behind this being that if two neurons are very not distinct, then this suggests that one of the neurons is redundant and so should be able to be removed without affecting performance too much. The paper measures the distinctiveness between two neurons as the angle between the neuron output activation vector over the pattern set for each of the two neurons.

Evolutionary algorithms are used in this paper for tuning and optimisation of hyperparameters. Determining the optimal values for hyperparameters is a very complex problem. Hyperparameters can be manually tuned through trial and error but this is very time-consuming. Using evolutionary algorithms to tune hyperparameters allows for the process to be automated efficiently [5].

2 Data

The dataset contains 576 unique data points each with 25 feature columns. The features represent various statistical features taken from the F7 channel such as mean, sum, and average amplitude change [4]. The data points are labelled according to what class of music genre they belong to. The 3 classes are classical, instrumental, and pop [4]. The data is evenly split into the classes with 192 data points for each class.

Figure 1 shows a heat map of correlations contained in the data. All of the feature columns are floats and so no re-encoding was required. Preprocessing of the data was performed by normalising each of the feature columns.

3 Method

The data was split into training, validation, and testing sets. The testing set consists of 20% of the total data, the training set consists of 80% of the remaining data and the validation sets consists of the other 20%. The evolutionary

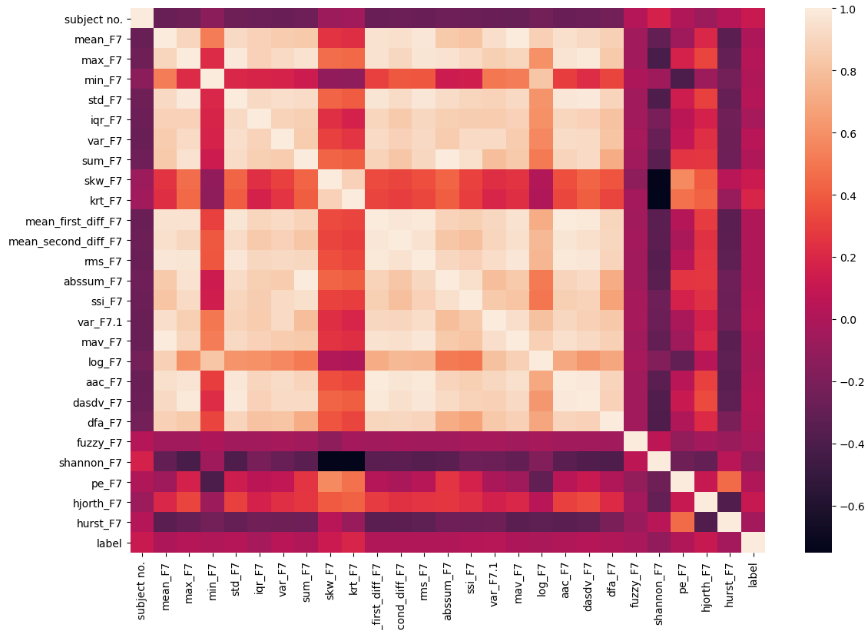

Fig. 1. Heat map showing correlations within the data

algorithm used the validation set to determine the fitness value of each individual. The overall performance of the neural network with the tuned hyperparameters was then determined using the testing set.

The evolutionary algorithm was run with a population of 15 and for 20 generations. These relatively small values were chosen because higher values resulted in the algorithm taking far too long to complete. A crossover probability of 0.8 and a mutation probability of 0.2 were used in the algorithm. Mating was done using two-point crossover and selection was done using tournament selection of size 3. The genes of each chromosome corresponded to the number of epochs, the learning rate, and the distinctiveness threshold. The initial value of each gene is randomly picked from its allowed range which is shown in table 1. Mutation was done by randomly picking one gene from the chromosome and randomly setting its value according to its allowed range.

Table 1. Table showing what values each gene is allowed to take. For example, Learning rate can be 0.5 because $0.001 \leq 0.5 \leq 1$ but number of epochs cannot be 600 because $600 > 500$.

Gene	Range start	Range end
Number of epochs	100	500
Learning rate	0.001	1
Distinctiveness threshold	1	45

The fitness function works by creating, training, and pruning a neural network according to the genes in each individual's chromosome and then returning the testing loss to be minimised. Each neural network contained 25 neurons in the input layer for each of the 25 input features. The hidden layers contained 50 neurons. Relatively large hidden layers were used so that there is lots of opportunities for neurons to be pruned. The output layers contained 3 neurons for each of the 3 possible music genre labels. Cross-entropy was used as the loss function and Adam was used as the optimiser.

The distinctiveness pruning was accomplished by creating a matrix where each column corresponded to a single neuron in the hidden layer and each row corresponded to a single input pattern from the training set. Each cell of the matrix corresponded to the output activation of each hidden neuron for each of the input patterns. The resulting columns of the matrix represent the activations of a hidden neuron over the input pattern training set. The angle between every combination of column is calculated. If the angle between two columns is less than the distinctiveness threshold in degrees, then one of the corresponding neurons is pruned and its weights are added to the remaining neuron. This is done because the two neurons are considered to not be distinct enough and so the function of both of the neurons is delegated to now only one of the neurons. Similarly, if the angle between two columns is greater than 180 minus the distinctiveness threshold, then both of the corresponding neurons are pruned. This is done because the two neurons are considered to be very distinct from each other and so the two neurons are working against each other resulting in no overall net difference. When all neurons that meet the two criteria are pruned, the network was then evaluated again on its accuracy at predicting the training and test sets.

When the evolutionary algorithm was completed, the fittest hyperparameters were used to train 30 neural networks which were then evaluated using the test set that was not used during the running of the algorithm.

4 Results

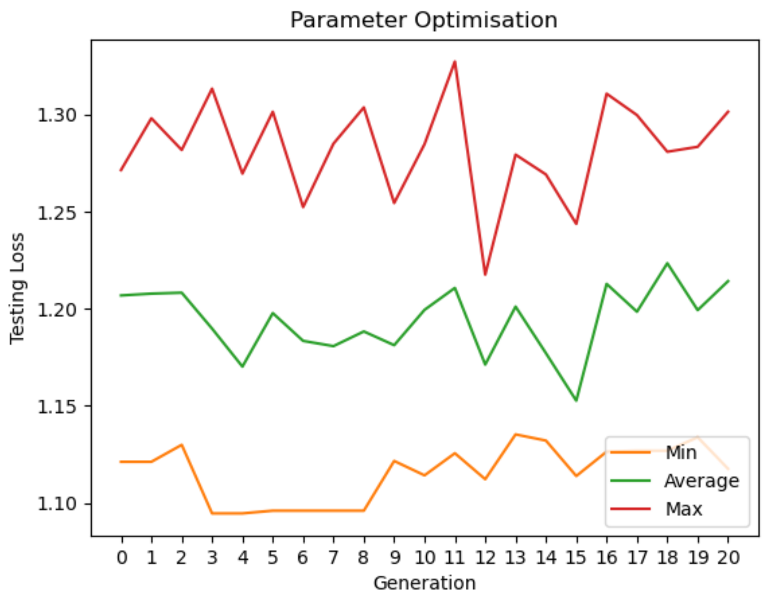

Fig. 2. Graph showing how the testing loss changed through the generations.

Table 2. Table showing a comparison between a basic neural network and the neural network that has been discussed in this paper.

	Basic neural network	Custom neural network
Average non-pruned training accuracy	81.27%	34.29%
Average non-pruned testing accuracy	38.79%	33.89%
Average number of neurons pruned	20.3	47.53
Average pruned training accuracy	46.72%	33.39%
Average pruned testing accuracy	36.58%	33.91%

5 Discussion

Figure 2 shows how the testing loss changed through the generations. We can see from this graph that the testing loss which was being used as the fitness function did not steadily decrease over time as we would hope it would.

Table 2 shows how the approach we used in this paper compares to a more basic approach. The basic approach consists of no hyperparameter tuning through an evolutionary algorithm. The number of epochs, learning rate, and distinctiveness threshold hyperparameters are instead manually chosen to be 350, 0.01 and 15°respectively. These values were chosen because they are fairly typical default values for these hyperparameters. For the custom neural network, the hyperparameters were taken from the fittest individual across all generations that was found in the evolutionary algorithm. The fittest individual was in the third generation and had genes corresponding to a number of epochs of 311, learning rate of 0.145865, and a distinctiveness threshold of 43. The results show that the basic approach is able to achieve a higher accuracy for both training and test sets for both the non-pruned and pruned versions of the neural networks. The approach we used in this paper was only able to achieve an average pruned testing accuracy of 33.91%. Randomly labelling the data, we would expect to achieve an average testing accuracy of 33.33% for 3 classes. Therefore, the network is only marginally better than a random guess.

This poor performance is likely a combination of several factors.

Firstly, a population size of only 15 and a number of generations of only 20 were used. These are relatively small values for these parameters and so poor performance is fairly likely. The small population size means that the number of combinations of genes in the population is very limited. The small number of generations means that there might be enough time for an optimal solution to be found through mutation and crossover. These two factors could be responsible for the poor performance of the evolutionary algorithm.

Secondly, the fittest individual had a distinctiveness threshold of 43. This is a very high value to be used for the distinctiveness threshold because it ends up meaning that a very high number of neurons are being pruned. Table 2 shows that on average 47.54 neurons are pruned from each neural network compared to 20.3 when the distinctiveness threshold is 15°. Pruning a very high number of neurons means that the remaining model is very simple and likely no longer has enough power to successfully predict music genre based on brainwave patterns.

Thirdly, the data being used only contains statistics collected from the F7 channel which is highly limited compared to the data that was used by the original dataset paper [4]. The dataset from the original paper contains statistics from several channels other than the F7 channel which allowed the paper to report a testing accuracy of 97.5%.

6 Future Work

Future work should be done on trying the evolutionary algorithm with a greater number of individuals and generations. This will hopefully mean that the hyperparameters of fittest individual found will result in much better testing accuracy.

Future work should also be done on using a much more complete dataset with enough features to be able to reliably predict the classes of the data.

Future work could also be done on using other approaches such as Bayesian optimisation [3] or grid search [1] to find optimal hyperparameters.

7 Conclusion

In this paper, the performance of a neural network which uses an evolutionary algorithm to tune its hyperparameters and uses distinctiveness pruning to prune its hidden neurons was compared against a basic neural network. It was found that the basic approach was able to achieve a much better accuracy than what the custom approach used in this paper was able to obtain. This is likely because the population size and number of generations was set too low which resulted in the fittest individual being closer to a random guess than an optimised set of genes. The fittest individual had a distinctiveness threshold of 43 which is a very high value for this parameter and resulted in the pruned neural networks having too few neurons to work properly.

The dataset used was taken from *Brain Melody Informatics: Analysing Effects of Music on Brainwave Patterns* [4]. This data was highly limited in its features which meant that the achieved network testing accuracies were very poor. The distinctiveness pruning technique was taken from *Indicators of Hidden Neuron Functionality: the Weight Matrix versus Neuron Behaviour* [2].

Until further work has been done by using larger parameters for the evolutionary algorithm and using a more complete dataset, it is unclear whether the approach used in this paper can ever be used to improve the performance of a neural network.

References

1. Bardenet, R., Brendel, M., Kégl, B., Sebag, M.: Collaborative hyperparameter tuning. In: International conference on machine learning. pp. 199–207. PMLR (2013)
2. Gedeon, T.D.: Indicators of hidden neuron functionality: the weight matrix versus neuron behaviour. In: Proceedings 1995 Second New Zealand International Two-Stream Conference on Artificial Neural Networks and Expert Systems. pp. 26–29. IEEE (1995)
3. Joy, T.T., Rana, S., Gupta, S., Venkatesh, S.: Hyperparameter tuning for big data using bayesian optimisation. In: 2016 23rd International Conference on Pattern Recognition (ICPR). pp. 2574–2579. IEEE (2016)
4. Rahman, J.S., Gedeon, T., Caldwell, S., Jones, R.: Brain melody informatics: Analysing effects of music on brainwave patterns. In: 2020 International Joint Conference on Neural Networks (IJCNN). pp. 1–8. IEEE (2020)
5. Safarik, J., Jalowiczor, J., Gresak, E., Rozhon, J.: Genetic algorithm for automatic tuning of neural network hyperparameters. In: Autonomous Systems: Sensors, Vehicles, Security, and the Internet of Everything. vol. 10643, p. 106430Q. International Society for Optics and Photonics (2018)