Using Evolutionary Algorithm to improve Neuron Reduction Techniques Based on Distinctiveness

Rong Xin

Research School of Computer Science, The Australian National University, Canberra, Australia
u6256440@anu.edu.au

Abstract. Network pruning techniques have drawn increasing attention in recent years due to the growing popularity of neural networks in machine learning, as it could reduce the enormous computational and time resources required to train the network. Network pruning techniques detect hidden units that are functionally trivial and remove them from the network to have a compact network that could achieve similar performance as the original network. Pruning by distinctiveness [5] is a network reduction technique that relies on the distinctiveness property of hidden units, and these undesirable neurons will be eliminated. However, some of the implementation details of such a technique remained unclear in the original work, the order of removing these trivial neurons was also unclear for the data compression task. Thus, this paper discusses some findings on how to sort the order of deleting hidden units. Genetic Algorithm is a very good way to search the order. This paper will apply a genetic algorithm to find the optimal deleting order of hidden units. The paper also analyzes whether similar nodes or complementary nodes should be pruned firstly.

Keywords: Pruning techniques · Neural Network · Evolutionary Algorithm · Network Reduction · Data Compression · Genetic Algorithm.

1 Introduction

Nowadays, back-propagation is commonly used to train feed-forward neural networks, as this technique ensures that the optimal weights could be learned and the loss is minimized, given a specific network architecture and a set of hyper-parameters. Nevertheless, a satisfactory performance cannot often be reached as the chosen architecture and hyper-parameters are not suitable for one specific task. An alternative approach to overcome such a problem is **genetic algorithm**, a metaheuristic algorithm that is inspired by neuroevolution [7]. The genetic algorithm not only allows to search for the optimal hyper-parameters, but it could also even help to learn other factors, such as activation function, architectures, and even the algorithm itself [10].

Genetic Algorithm is inspired by the biological evolution process, especially evolution by natural selection, where the fittest individuals are more likely to survive and reproduce [7]. This algorithm mimics the natural selection process in a given population of solutions and searches for the optimal solution for a given problem; populations will also evolve through several different selections, reproduction, and mutation operations. When applying the genetic algorithm to optimize certain factors of a neural network, a set of neural networks with some randomly selected values of factors is the initial population. The likelihood of an individual to survive is then measured by its fitness; the fitter the solution is, the more likely the solution is selected to survive and reproduce to generate a new population. The main components of a genetic algorithm are a population of certain numbers, a fitness function that evaluates the survival strength of each individual, the chromosome representation that encodes the solution, a selection operator, a crossover operator and a mutation operator [3]. The selection operator is used to select new populations; the crossover operator is for reproducing a new population, and the mutation operator could mutate some individuals in the new population to introduce some randomness to this algorithm.

Network pruning is another emerging field that has been developed over the years. It is a task that involves systematically removing parameters or hidden units from existing neural networks, and the goal of such pruning is to have a reduced network that could achieve similar performance as the initial network. As genetic algorithm can learn algorithms for the network, it could be applied to the network reduction technique by distinctiveness [4, 5, 11], to optimize on some factors that could affect the loss of the pruned network.

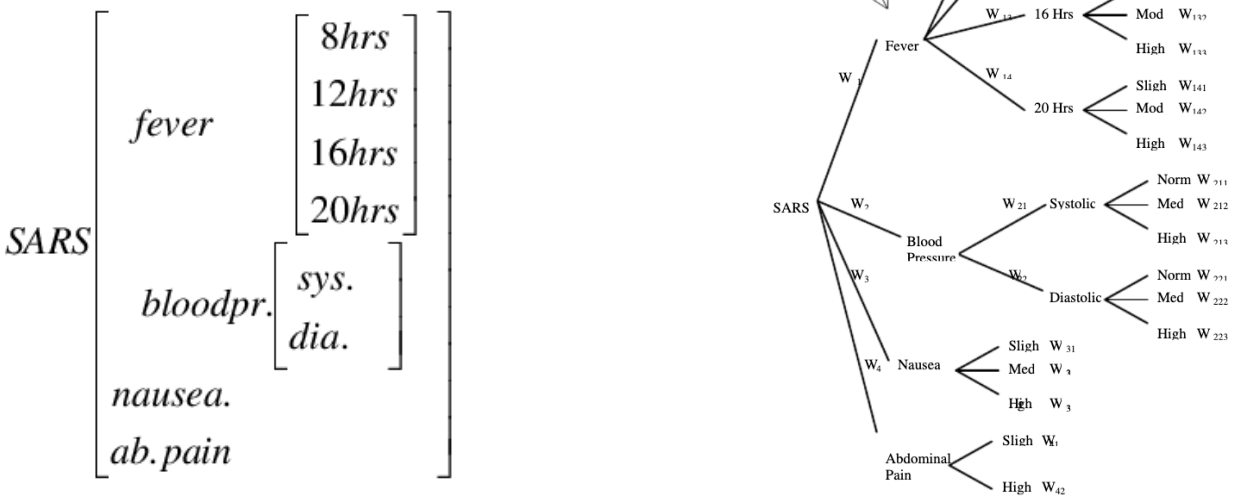
In the previous work [11], some implementation details of the pruning by distinctiveness technique have been discussed. In the paper, to reduce the computation needed for updating weights and calculating the angels between output activation vectors, the author proposed to first eliminate complementary pairs of neurons, and then prune neurons similar to other existing neurons [11]. However, it is not investigated that whether the proposed pruning order would affect the performance of the pruned network, and if there exists such an impact, whether the proposed pruning order could guarantee the loss of pruned network is minimized. Similarly, this work also suggested pruning the least distinctive neurons in the first round of pruning, which is a decision based on human intuition. Nevertheless, there is no experimental evidence that could support this decision. Therefore, in this paper, a genetic algorithm is applied to the network reduction by distinctiveness technique to investigate how the pruning order of unnecessary neurons may affect the performance of the pruned network for a data compression task; we will also discuss some

observations we made from the experiments that may indicate some the pruning order preference for this task by analyzing 100 optimum pruning orders found by genetic algorithm. This work performs a data compression task using a data set with medical data [8], which is the same data set used in the previous work [11]. Since this work is aimed to improve on the previous work and to have the results from the previous work as the baseline model, the same data set is thus chosen.

2 Data Set

As mentioned previously, the fuzzy medical data set [8] used in the previous work [11] is used. This data set contains fuzzy signatures of some patients medical data. The data set contains 4000 samples, 23 features, including a class label indicating whether the patient is normal, having SARS, high blood pressure or pneumonia. These features describe some medical observations or symptoms for the patients, and the data were recorded using fuzzy signature. In order to compare the results from this work to the previous work, identical data pre-processing techniques were used; only 18 features were used to perform the data compression task, as the features indicating nausea and abnormal pain were removed.

This data set used fuzzy signature to mathematically model the ambiguity appeared in medical data; by learning this data, the network could model the decision making process based on uncertainty, which would be more similar to the human reasoning process. Fuzzy signatures are vectors of fuzzy values, where each element of this fuzzy signature can be another vector of fuzzy values [9]. An example of fuzzy signature for SARS patients is illustrated in Figure 1. This example shows the ability of fuzzy signature to model degrees and ambiguity in real life.



(a) Fuzzy signature in vector structure

(b) Fuzzy signature in tree structure

Fig. 1: SARS fuzzy signature in tree and vector structure [9]

3 Method

In this work, we focused on optimizing the pruning order for feed-forward neural networks with one hidden layer. This section first briefly summarized the pruning by the distinctiveness method, and then discussed how the genetic algorithm was applied to learn the pruning order for this network reduction technique.

3.1 Network Reduction by Distinctiveness

The pruning technique was based on the property of distinctiveness [5]; this property measures the functionality of hidden units based on the angle between output activation vectors of neurons in the input pattern space. The original work stated that neurons that have small weights are considered to be unnecessary as they contributed trivially to the output. Moreover, neurons that share similar functionalities with other hidden units and a group of neurons that jointly has a constant or no effect on the network are also considered undesirable, and thus should be pruned. Similar hidden units could be detected by searching for the pairs of neurons that have an angel less than 15°between their output activation vectors, whereas a pair of complementary neurons often have an angel greater than 165°between the output activation vectors. One of the hidden units within a pair of similar neurons would be removed and the weights were transferred to the remaining neuron. On the other hand, if a group of neurons has a joint constant force on the network, all neurons within this group would be eliminated and the bias was updated accordingly. 0.001 was used as the threshold for detecting neurons with small weights, which is the same as the previous work.

In our last work [11], some implementation details of this pruning technique were carefully discussed; we proposed a general pruning process based on the consideration for computational time and resources. The previous work suggested to firstly prune units with small weights, then remove complementary groups of hidden units, and lastly remove hidden units that share similar functionalities with other units. The main idea behind this pruning process was to reduce the number of times required to update the weight matrix and the angels between output activation vectors. However, this process did not guarantee the performance of the pruned network. Since hidden units were pruned in sequential orders, the pruning order would have an impact on the pruned neurons. For instance, given a neuron n_i is similar to n_j, and n_i would remain if n_j was already pruned beforehand; yet n_i may be added to the pruned list of neurons if n_j was not removed. Theoretically, pruning either n_i and n_j would make no difference to the network, and the pruning order would have no impact on the performance of the pruned network; however, it is usually not the case in reality. Since hidden units that were considered to be similar or complementary in functionalities are not identical or opposite, there will be some errors introduced when transferring or updating the weights. The original work also suggested to not retrain the network under most of the conditions due to the consideration for computation, these minor errors introduced when pruning is inevitable. Thus, the pruning order will, in real-life application, have an impact on the performance of the pruned network.

The implementation method proposed in the previous work makes a trade-off between accuracy and computational time. For each round of pruning, small sets of nodes are eliminated in the order stated above and the weights are updated; several rounds of pruning and update will be made until there are no neurons are considered undesirable.

3.2 Genetic Algorithm

Genetic algorithm is a bio-inspired meta-heuristic algorithm that is heavily inspired by the natural selection process. A general genetic algorithm includes an initial population of a given size, the chromosome representation of solutions, a fitness function used to evaluate how well a solution is, a selection operator, a crossover operator, and a mutation operator. A genetic algorithm starts with initializing the population of solutions that are already encoded as chromosomes and then is followed by evaluating each individual using the fitness function. Theoretically, the higher the fitness score of a solution, the greater chance this solution will survive in the next round of evolution. In the context of applying genetic algorithms to neural networks, the fitness function is often related to the loss of the network. After the evaluation for all individuals, reproduction was performed using the crossover operation to create new offspring solutions. Solutions are selected to reproduce based on their fitness score; the fitter the solution is, the more likely it is selected to crossover. Parent solutions were selected using the selection operation. Various crossover operations could be performed, and the ones used in this work will be discussed later. Offspring solutions are then mutated using the mutation operator to introduce some randomness to the algorithm. Lastly, the replacement was performed to generate a new population of the same size as the initial population. The steps from evaluation to replacement are looped until a termination condition is met, such as having a certain number of generations or reaching a certain fitness score.

In this work, the genetic algorithm was applied to optimize the pruning order for a network that was trained to compress data. Generally, in the context of neuroevolution for neural networks, each solution is usually a neural network. However, since in our case, the problem to be optimized is the pruning order instead of the network itself, each individual is thus a sequence of hidden unit pairs to be pruned, and the initial population is 100 different sequences of pruning pairs.

Initial Population For a trained neural network, the angels between each pair of output activation vector for the hidden units were calculated, and all pairs of hidden neurons that were considered to be either similar or complementary were the possible pruning pairs. The removal of each complementary pair or one unit from a similar pair is considered as a possible action, and one of the permutations of all these possible actions is one possible solution. For instance, there are two possible pruning pairs of neurons (n_1, n_2) and (n_3, n_4), and the pruning order of either pruning hidden units from (n_3, n_4) first or (n_1, n_2) first would yield two different possible solutions. Thus,

the initial population for this pruning order problem is 100 random possible permutations from all possible ordering sequences of the possible actions.

Chromosome Representation However, instead of using a sequence of neuron pairs as the solution, an order encoding was used to the solution. Instead of having a sequence of possible pruning pairs, each pair of hidden units was assigned with an order. For the previous example of (n_1, n_2) and (n_3, n_4), instead of having $[(n_1, n_2), (n_3, n_4)]$ as one possible solution, (n_1, n_2) and (n_3, n_4) were encoded as 1 and 2 respectively. Therefore, the encoded solution in this example is $[1, 2]$. The actual population is the encoded permutations of possible pruning pairs.

Fitness Function Having the encoded population, a fitness function was required to evaluate each individual. To evaluate how effective one specific order of pruning is, an un-pruned network trained on the medical data set on data compression task was then pruned in the order given by each individual, and the test loss of the pruned network could then be used to indicate how well the pruning order is. In principle, the higher the fitness score, the better an individual is. However, for the implementation of this work, it is declared that the lower the fitness score, the better an individual is, as the fitness function is set to be equaled to the test loss of the pruned network. Note while pruning the network, the result of pruning would not be identical although all individuals share the same set of possible pruning pairs. This was because that when one hidden unit or a pair of neurons were pruned, all pairs of neurons containing these units in the latter of the pruning sequence will be ignored to avoid repeated removal of neurons. The result of pruning thus differs from each other.

Selection Operation After all solutions are evaluated, some solutions are required to be selected for reproduction. Tournament selection [6] was chosen to be the selection method. This method is often used to select one individual from a population by having a few rounds of the tournament between randomly chosen individuals. The effectiveness of this selection method is heavily dependent on the tournament size. When tournament size equals 1, this selection is equivalent to randomly selecting a solution for reproduction, whereas when the tournament size is large, the best individual from the population is often returned as the weaker individuals is having less chance to win. Thus, the selection of tournament size is often essential, and in this work, 3 is used, as it ensures that some of the fittest solutions can be found while still preserving some degrees of diversity as there is the chance for all individuals to be selected to participate in the tournament. An individual is thus selected using this tournament selection to become a parent for the crossover operation.

Crossover Operation Given all parents being selected using tournament selection, crossover operations are needed for reproducing offspring. Since each individual solution is represented using order encoding; thus, the selected reproduction operator has to be suitable for this type of encoding to ensure there will be no repeated elements or loss of elements in the resulted offspring solution. One common crossover operation for order-encoded solutions is order crossover, introduced by Davis [2]. This crossover operation selects a sub-string from one of the parent individual and copies the sub-string to the corresponding position of the offspring solution. Elements that do not appear in the sub-string are copied to the offspring in the sequence that is identical to their sequences in another parent, to preserve the relative order of the second parent. This operation preserves orders from both parents, which is desired for this task, and the resulting offspring still contain all possible pruning pairs without any duplication or missing elements. This makes this order crossover operation suitable for the task. The crossover probability is 0.8.

Mutation Operation For the mutation operation, a shuffle mutation operation was chosen because it remained all the elements within the order sequence after mutation. This means that after mutation operation is performed, the mutated individual still contained all encoded pruning pairs without duplication. It also introduced some randomness to the algorithm yet without completely changing the individual. The mutation probability was 0.2.

Replacement Lastly, after having a set of new solutions, replacement is required to maintain the same population size N. Some common replacement includes generational replacement that discards all previous solutions and includes N offspring, and truncation replacement by selecting N best from both offspring solution and the original population. The replacement scheme used in this work is the elitist replacement, which always preserves the best 3 individuals from the original population, and selects $N - 3$ offspring solutions to jointly form the new population, which is proven to improve the performance of the genetic algorithm [1].

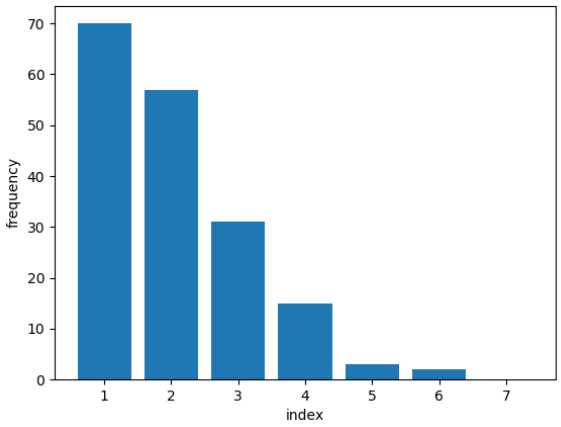
4 Results and Discussion

In this section, some empirical experiments have been conducted to analyze the optimal pruning order for the network reduction by the distinctiveness method. This section observed the experimental results from 100 times of applying genetic algorithm to find the best pruning order, and intended to address two questions: (1) Is pruning complementary hidden units first more preferred in the optimal result? (2) Is pruning hidden units that have smaller angels between its similar pair or units that have greater angels between its complementary pair more preferred in the optimal result?

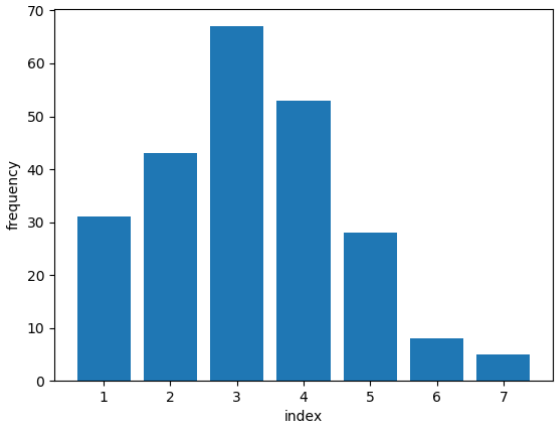
This network pruning technique was previously implemented in the field of image compression [4], and the methods were proved to be quite effective. We want to reproduce this experiment on some other forms of data that require data compression, such as medical data. Thus, the selected data set containing medical data from normal person, SARS, high blood pressure and pneumonia patients [8] was used, and the pruning method was applied to the trained network to get a more compact representation of the fuzzy medical data.

All experiments are conducted using a one-hidden-layer neural network with 15 hidden units, which was trained with 500 epochs, using a mini-batch with a size of 10, and the learning rate was set to 0.01. This network could learn features from the medical data and compress the essential information to a lower-dimensional space.

After 100 runs of genetic algorithm on the network pruning approach, these optimal sequences were recorded. The average loss of pruned networks on data compression task after following the optimal pruning sequence was 0.156, which improved 13.33% from the benchmark result, 0.18 [11]. The average number of neurons being pruned was around 4 whereas the benchmark model pruned 2 units. This suggested that there was still room for improvement for our benchmark pruned network that was found using our previously proposed algorithm.



(a) Similar Pairs of Hidden Units



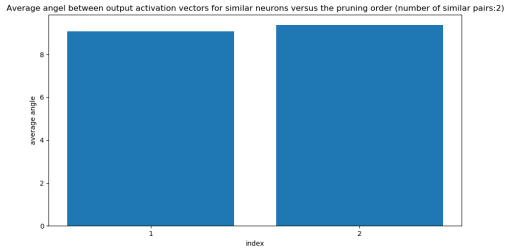
(b) complementary Pairs of Hidden Units

Fig. 2: Frequency Histogram of Both Similar Pairs and Complementary Pairs appeared in the optimal pruning order

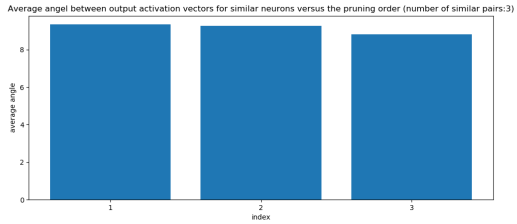
By observing the result in Figure 2, one could observe clearly that among 100 times of experiments, nearly 70% of the experiments began with pruning hidden units from a pair of similar hidden units. There is an obvious trend shown in Figure 2a indicating that hidden units that have similar functionalities with other units were pruned earlier in the pruning order. On the other hand, hidden units from complementary pairs are often pruned in the later of the solution. We have also calculated the average index of both similar and complementary pairs in the pruning sequence to observe the trend; it has shown that the average index for similar pairs was 2.045 and 3.204 for complementary pairs. However, by closely examining the experiment results, it has been found that there existed some solutions that only contained similar pairs or complementary pairs. By excluding these solutions from the analysis for the average index, it has shown that the average index for similar pairs was 2.022 and complementary pairs were 3.467. Moreover, the average index of the first appeared similar pair in the optimal eliminating sequence was 1.268, whereas the average index for the complementary pair was 2.564. These findings have again proved our observation that similar pairs are more preferred to be pruned earlier in the pruning sequence for optimal solutions. This observation was contradicted what was proposed in the previous work, which started to prune complementary pairs before similar pairs because it would reduce computational time spent on updating weights and output activation vectors.

However, this observation may indicate that modifying the algorithm proposed in our previous work [11] by altering the steps in the algorithm could further boost the performance of the pruned network. Instead of pruning complementary pairs before similar pairs, these two steps could be switched in their orders. The loss of the pruned network following the updated pruning process was 0.164, which improved 8.89% from the benchmark model. This has proved that the pruning order indeed would have an impact on the performance of the pruned network, and moreover, pruning neurons from similar pairs before complementary pairs would positively impact the performance of the pruned network. One possible reason for this could be that since two neurons were pruned for each pair of complementary hidden units, more errors were likely to be introduced in the pruning process. On the other hand, if similar neurons were pruned first, fewer neurons were pruned in each iteration of pruning, and thus fewer errors would be introduced in each iteration before the weight update. This potentially minimized the loss for the pruned network as less information or learned knowledge was lost in the pruning process.

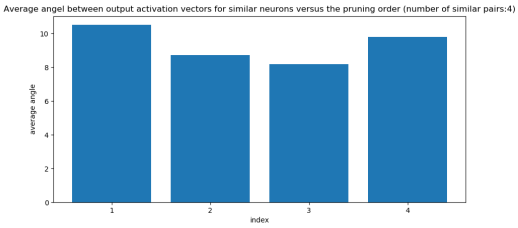
Apart from the pruning order between similar pairs and complementary pairs, another problem that is worth investigating is the ordering within similar or complementary pairs. Is pruning the most similar neurons earlier in the sequence improve the performance of the pruned network? Theoretically, we hypothesized that the smaller the index, the smaller the average angle between the output activation vectors of two similar hidden units. Figure 3 illustrated the average angle between pruned similar pairs versus the index in the pruning sequence. From Figure 3, such a hypothesis was not proved, as it seemed that there was no clear relationship between how similar the hidden units were compared to other neurons and the pruning order. There were some trends shown in Figure 3d, however, this was because that the sample size for N=5 was 1. Thus, the conclusion could be drawn from this one



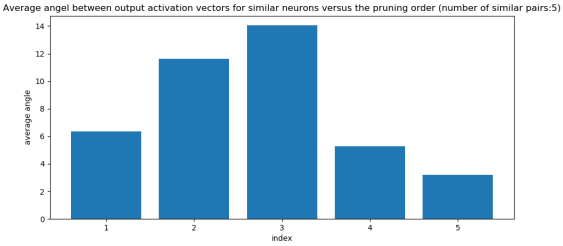
(a) N=2



(b) N=3

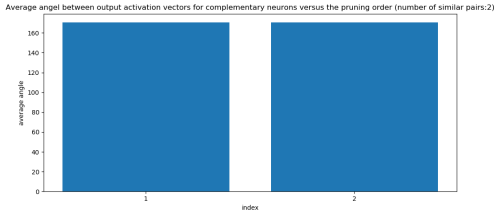


(c) N=4

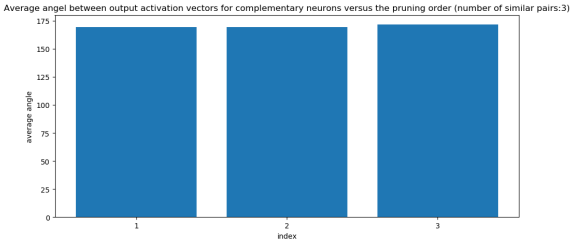


(d) N=5

Fig. 3: Average angel between output activation vectors for similar neurons versus the pruning order (with various number of similar pairs N)



(a) N=2



(b) N=3

Fig. 4: Average angel between output activation vectors for complementary neurons versus the pruning order (with various number of similar pairs N)

sample. A similar trend has also been observed for complementary pairs, as shown in Figure 4. Nevertheless, a conclusion that there exists no relationship or correlation between the angle of output activation vectors for similar or complementary pairs and the pruning order could not yet be drawn, as the sample size for this experiment was rather small. There were also not enough neurons being pruned; ideally, a more extensive experiment with a larger sample size and more pruning pairs should be conducted to make a conclusion. Therefore, a data set with a greater amount of features, and a network with greater hidden units would be required to conduct such an experiment.

5 Conclusion

To conclude, this work implemented the network reduction by distinctiveness technique [5] using the process proposed in [11], and applied generic algorithm to this pruning algorithm in order to investigate how the pruning order of neurons would affect the performance of the pruned network using a data compression task on a fuzzy signature medical data set. The previously proposed pruning process suggested pruning complementary pairs of hidden units before neurons that are functionally similar to other hidden units. However, after examining the results from 100 experiments of apply genetic algorithm to optimize the pruning order, it was observed that pruning units from similar pairs of neurons before complementary pairs were more preferred in the optimal pruning orders. Based on this observation, the proposed algorithm in our previous work was modified, and the result has proved the effectiveness of this modification as it reduced the loss of the pruned network. Another experiment was also conducted to explore the potential relationship between the angle of output activation vectors for similar or complementary pairs and the pruning order. However, due to the limitation caused by the data set and the task, not enough pruning pairs could be found for this one-hidden-layer network; thus, the experimental results were considered not strong enough to draw inference from.

Future works can be done on performing data compression using this approach with a data set with more features. This would allow the choices of the initial number of hidden units for the network to be sparser, and one could potentially observe more on the relationship between angels within pruning pairs and optimal pruning orders. Similarly, a greater amount of experiments could also be conducted to have more sample sizes to analyze.

References

1. Anu: Improved performance of replacement strategies in ga. International Journal of Advanced Research in Computer Science and Software Engineering **3**(9), 344–346 (2013)
2. Davis, L.: Applying adaptive algorithms to epistatic domains. In: Proceedings of the 9th International Joint Conference on Artificial Intelligence - Volume 1. p. 162–164. IJCAI'85, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA (1985)
3. Engelbrecht, A.P.: Computational intelligence. CRC Press, 2 edn. (2008)
4. Gedeon, T.: Indicators of hidden neuron functionality: the weight matrix versus neuron behaviour. Proceedings of 1995 Second New Zealand International Two-Stream Conference on Artificial Neural Networks and Expert Systems (1995). https://doi.org/10.1109/annes.1995.499431
5. Gedeon, T., Harris, D.: Network reduction techniques. Tech. rep., Department of Computer Science,Brunel University (1991)
6. Goldberg, D.E., Deb, K.: A comparative analysis of selection schemes used in genetic algorithms. Foundations of Genetic Algorithms, vol. 1, pp. 69–93. Elsevier (1991). https://doi.org/https://doi.org/10.1016/B978-0-08-050684-5.50008-2, https://www.sciencedirect.com/science/article/pii/B9780080506845500082
7. Katoch, S., Chauhan, S.S., Kumar, V.: A review on genetic algorithm: past, present, and future. Multimedia Tools and Applications **80**(5), 8091–8126 (2020). https://doi.org/10.1007/s11042-020-10139-6
8. Mendis, B.S., Gedeon, T.D., Koczy, L.T.: Investigation of aggregation in fuzzy signatures. In: Proceedings of 3rd International Conference on Computational Intelligence, Robotics and Autonomous Systems, Singapore (2005)
9. Mendis, B., Mendis, U., Gedeon, T., Koczy, L.: Flexibility and robustness of hierarchical fuzzy signature structures with perturbed input data (2006)
10. Stanley, K.O., Clune, J., Lehman, J., Miikkulainen, R.: Designing neural networks through neuroevolution. Nature Machine Intelligence **1**(1), 24–35 (2019). https://doi.org/10.1038/s42256-018-0006-z
11. Xin, R.: An empirical study of neuron reduction techniques based on distinctiveness on classification and data compression task (2021)