Experiment on the using evolutionary algorithm for neural network's input features selection and parameter optimization - using UNSW COMP1111 mark dataset

Zixuan Zheng

Research School Computer Science
The Australian National University
Canberra, ACT 0200, Australia
u6153124@anu.edu.au

Abstract. This paper aims to utilize the evolutionary algorithm to find the best input features combination and the parameters of a 3-layer neural network like the learning rate and the number of hidden neurons, which would obtain the most optimal test result of neural network model using the UNSW COMP1111 dataset. The following experiment would firstly explores the most optimal input features and parameters setting by running the evolutionary algorithm then explore how the allele frequency of different attributes changes in generations. Based on the analysis on allele frequency, the new setting of input features and parameters would be put forward. Finally, the paper would discuss the difference between such result part of input features selection and previous work of importance analysis, including magnitude analysis, distinctiveness analysis and sensitivity analysis would be used for analysis.

Keywords: feature selection · neural networks · evolutionary algorithm · parameters optimization · importance analysis

1 Introduction

In the previous paper, a simple feedforward neural network with the structure 14-10-1 is trained for analyse the importance of the input features. During the experiment of importance analysis, it is found that remove some features may not affect the final performance of the model. Thus, it is potentially that some of those input attributes is redundant for the training, which the importance analysis may not illustrates. Finding the overall optimal combination of the input attributes rather than simply sort the importance of the feature would be more helpful for not only train the data, but understand the practical meaning of the data. Besides, the parameters like the number of hidden neurons, learning rate or the number epochs is based on some manual trials, which sometimes relies on luck and may not be stable when training the model. To obtain a optimized combination for input selection and parameters settings, this paper would apply the evolutionary algorithm to the process of the training of the neural network. As for the training, this paper would still use the UNSW COMP1111 dataset. The original dataset consists of 153 input patterns and each pattern contains 16 attributes. The details can be viewed in the following table 1.

Table 1: Importance measured by 4 methods

Name	Type	Range	Validity	Explanation
Regno	Categorical		100%	
Crse/Prog	Categorical		100%	
S	Ordinary	$[0,4]$	100%	
ES	Categorical		100%	
Tutgroup	Categorical		95.4%	7 unknown
lab2	Numeric	$[0,3]$	84.3%	24 unknown
tutass	Numeric	$[0,3]$	94.8%	8 unknown
lab4	Numeric	$[0,3]$	85.0%	23 unknown
h1	Numeric	$[0,20]$	86.3%	21 unknown
h2	Numeric	$[0,20]$	64.7%	54 unknown
lab7	Numeric	$[0,3]$	64.7%	54 unknown
p1	Numeric	$[0,20]$	73.9%	40 unknown
f1	Numeric	$[0,20]$	61.4%	59 unknown
mid	Numeric	$[0,45]$	93.5%	10 unknown
lab10	Numeric	$[0,3]$	72.5%	42 unknown
final	Numeric	$[0,100]$	94.8%	8 unknown

2 Methods

2.1 Data pre-processing

The pre-process for input features would be operated in the following steps.

1. Remove Regno since it is ident.
2. Impute all the empty field with 0 for numeric attribute, and the mode for categorical attribute.
3. If the feature is numeric value, normalize it to $[0,1]$ by divided its up bound.
4. If the feature is Categorical or discrete, do the following steps,
   (a) If the feature is numeric value, normalize it to $[0,1]$ by divided its up bound.
   (b) If the feature is Categorical or discrete, do the following steps,
   (c) Normalized the value by dividing the set size of that attribute.

2.2 Optimization by evolutionary algorithm

Generally, the evolutionary algorithm would be used for optimizing the 3 layer feedforward neural network. This section would specifically introduce the design of chromosome, fitness methods, selection methods and reproduction methods for the algorithm.

First of all, the total length of the chromosome would be 33 bits. Specifically, the chromosome would use the first 14 bit as binary vector to present the selection of all those 14 input features. If one bit is set 0, then the corresponding input feature would not be selected and if it is set 1, it would be selected for training the model. The second part uses 4 bits using the gray code for encoding to represent the number of hidden neurons. The third part also use gray code for encoding for the next 6 bits, which represent the number of the learning rate. Please noted that the learning rate would be divided by 100 when actually applied to the neural network model. Choosing such length of gray code is based on the experience of previous paper, where the author found that when the learning rate is higher than 0.6, the accuracy would decrease rapidly and the total number of hidden neurons should not greater than the input neurons. Finally, the rest 9 bits represents the number of epochs for training. The reason to choose the epochs under 500 is to avoid over-fitting.

The fitness function would firstly apply those parameters stored in the chromosome to the neural network and then calculate the R^2 score of the model. Besides, this paper would use the strategy of rank-based selection. The basic thought of such selection method is that the entity with higher R^2 score would be more likely to be selected since the purpose is to choose the parameters lead to higher accuracy on predicting marks. Moreover, this paper use the proportion selection for parents and the two-point crossover for reproduction and use the random mutation. Please noted that the implementation of such process take advantages on the work of lab8, which should be acknowledged.

Then, as for the parameters of the evolutionary algorithm, the population is set to 50 and it would evolute for 100 generations. Due to the computation limit of the author's computer, it could take too much time for training and the accuracy cannot improve more when setting more populations or generations.

Table 2: Parameters setting for evolutionary algorithm, adapted from

Name	Value
Population size	50
Number of generations	100
Crossover rate	0.8
Mutation rate	0.02

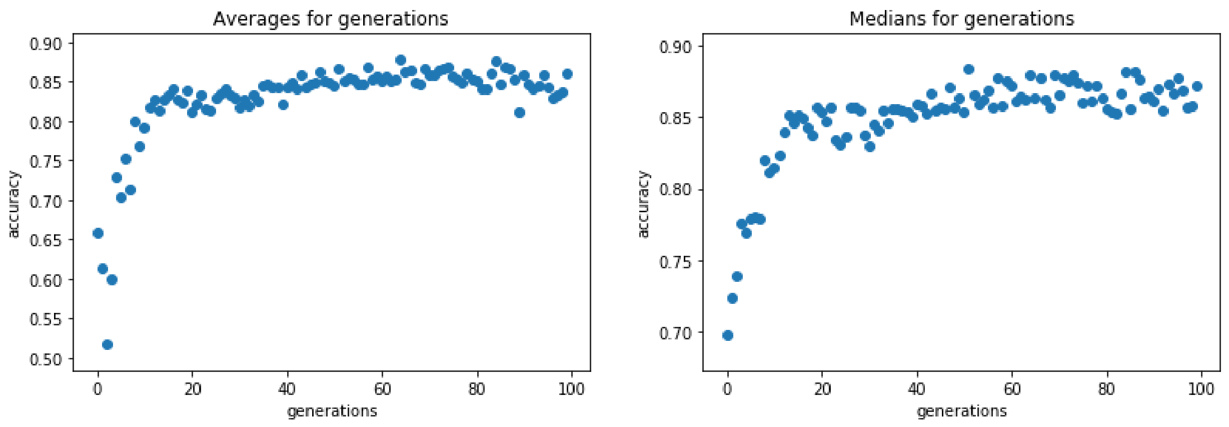
2.3 Training the neural network

As stated in the Introduction, the previous paper trained a 14-10-1 feedforward neural network and it is found that training results may fluctuate severely. To solve such problem, this paper uses evolutionary algorithm to optimize the parameters. Therefore, in this paper the number of neurons in both input and hidden layers would be trained by the evolutionary algorithm and only the output layer remains one neuron and the model is still evaluated by the R^2 score as the output is a sequence of number which is similar to evaluating the results of linear regression models. Besides, after the optimizations, the model would be retrained by the new input features and parameters settings to see the effect of evolutionary algorithm.

3 Result

3.1 Overall performance of evolutionary algorithm

Generally, the overall performance is increasing over the generations, where the median and average of the accuracy in figure 1 illustrate an obvious increasing trend. In the end, the general accuracy would be around 85% to 90% and the maximum accuracy could reach around 95%. This indicates that the evolutionary algorithm successfully maximize the accuracy of the populations.



(a) Mean of the accuracy (b) Median of the accuracy

Fig. 1: The accuracy trend over generations

3.2 The change of allele frequency

According to [5], the nature of the evolution is the natural selection on the species whose external phenotype fit the most on the environment, and the change of external phenotype reflects the change on the internal allele frequency over the generations in biology. Therefore, it is necessary and worthy to see the change of the distribution of the value of each gene in such bio-inspired algorithm.

Input feature selection As it shown in figure 2, since each input feature is randomly selected in the first generation, the allele frequency of each input feature is around 50%. After 100 generations of selection and reproduction, the frequency of input feature 1, 8, 10, 11, 13 improves significantly especially the last 4 features increases to nearly 100%. The feature 2 improves very slightly compared to them and the rest input features suffer a decrease to nearly 0.

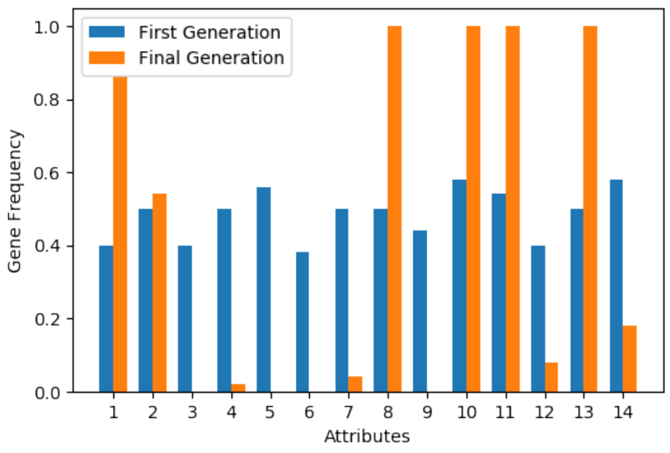
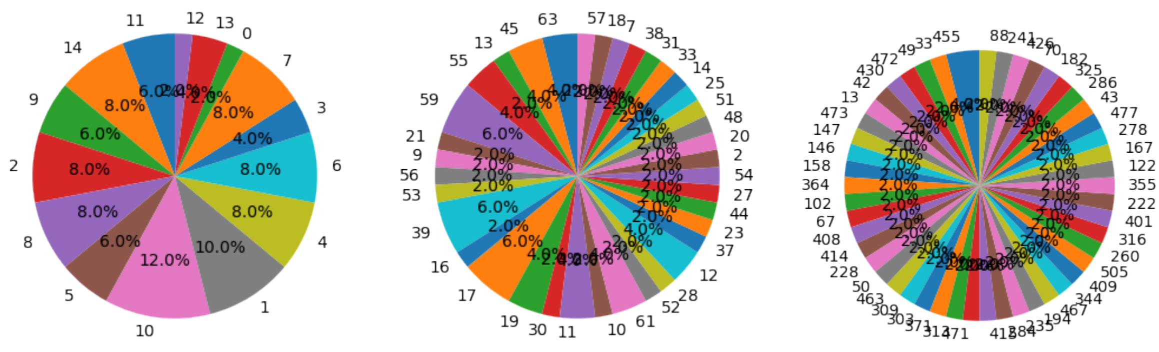


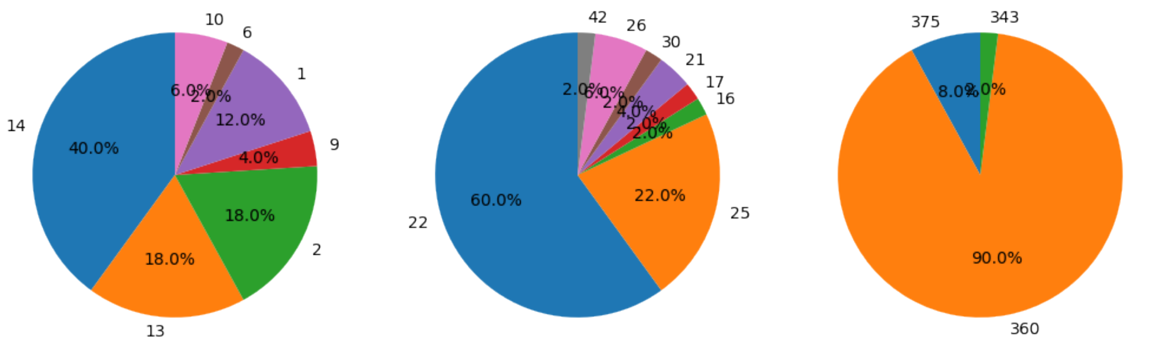
Fig. 2: The allele frequency of the attributes in the first and the final generation

Parameters of neural network According to Figure 3 and 4, the change of each parameter also indicates an obvious change on the allele frequency. At the very beginning, the distribution of each number is quite uniform due to the random assignment and in the end, the proportion of some number significantly increased especially for the training epochs, where the number 360 takes up 90% of the population in the final generations. As for the number of hidden neurons, 40% of the populations choose 14, and 60% of the populations take 0.22 as the learning rate.



(a) Hidden neurons' number (b) Learning rates (c) Training epochs

Fig. 3: The distribution the parameters of neural network among the populations in the first generations



(a) Hidden neurons' number (b) Learning rates (c) Training epochs

Fig. 4: The distribution the parameters of neural network among the populations in the final generations

New input features and parameters set Based on the previous analysis on the distribution of allele frequency, it is confidently to use the new set of input features and parameters to train the model. Generally, it would use those input features with more than 50% frequency and those parameters occupy the largest proportion of the population in the final generations. The result is shown in Table 3. Please noted that the rank in the input features does not matter.

Table 3: The final parameters and input feature decisions

Input Features					
1	2	8	10	11	13

Parameters		
Hidden neurons	Learning rates	Training epochs
14	22	360

3.3 Retraining the model

Based on the new settings on the input features and parameters, the model would be retrained and the result is in Table 4. Please noted that the negative values are caused by failed prediction, where all results are equal to 0. Such fail prediction may be caused by randomly initial weights.

Table 4: Accuracy of 10-fold for every 40 epochs and test set

Epoch	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Fold 6	Fold 7	Fold 8	Fold 9	Fold 10
40	-4.9002	0.7230	0.8151	-5.7615	0.7693	0.8089	-4.7513	0.8276	0.8067	0.7061
80	-4.9002	0.8196	0.8468	-5.7615	0.8277	0.8524	-4.7513	0.8664	0.8409	0.7903
120	-4.9002	0.8491	0.8656	-5.7615	0.8544	0.8712	-4.7513	0.8762	0.8585	0.8282
160	-4.9002	0.8643	0.8781	-5.7615	0.8683	0.8807	-4.7513	0.8817	0.8685	0.8499
200	-4.9002	0.8727	0.8863	-5.7615	0.8762	0.8873	-4.7513	0.8854	0.8748	0.8620
240	-4.9002	0.8774	0.8917	-5.7615	0.8812	0.8909	-4.7513	0.8882	0.8793	0.8693
280	-4.9002	0.8801	0.8952	-5.7615	0.8849	0.8935	-4.7513	0.8905	0.8821	0.8742
320	-4.9002	0.8819	0.8977	-5.7615	0.8878	0.8956	-4.7513	0.8922	0.8842	0.8776
360	-4.9002	0.8832	0.8992	-5.7615	0.8903	0.8975	-4.7513	0.8938	0.8858	0.8802
Test set	-9.2053	0.9257	0.8496	-2.3605	0.8323	0.9366	-16.2273	0.5017	0.7744	0.9465
Fold selected			6			Final Accuracy			0.9180	

4 Discussion

4.1 Evolutionary algorithm implementation

The overall accuracy performance in Figure 1 shows a quickly increase trend at the beginning, which proves that the evolutionary algorithm is efficient to optimize the model. The reason behind is that the initial random chromosome and the following mutation gives the optimization the opportunity to set those potential combination of ”good gene” in a relevantly large population and such ”good gene” would be emphasized by reproduction since the selection process would eliminate those ”bad gene” which unfit the optimization goal. The experiment proves that such process of emphasizing is powerful.

However, the author also notices that the evolutionary algorithm has a high requirement for computation resource. The whole process of 100-generation optimization spend around 90 minutes where the model is just a simple neural network which cannot be considered as very complicated ones. The reason may caused by the complexity of algorithm that the model would be trained for $n * m$ times, where n denotes the population size and m denotes the number of generations. If we take account of the complexity of training the model, such complexity would become very huge and potentially ”polynomial in the problem size” [6]. Therefore, although the evolutionary algorithm can theoretically obtain one good optimization which fit the target, using this method still need to be cautious since its potentially high time complexity.

4.2 Analysis on the new features

Input features In the previous paper, a feedforward neural network with the structure 14-10-1 is trained. Following the methods in [2, 1, 3, 4], an importance table 5 is generated and such data would be used for compare the result of evolutionary algorithm. Besides, the previous paper identifies that using the sensitivity calculated by arithmetic mean has the best performance while using the sensitivity calculated by standard deviation has the worst performance and there are no significant difference between using the magnitude and distinctiveness. Please noted that since using sensitivity by standard deviation has already proved worst in the previous paper, only the rest three methods would be discussed here.

Table 5: Importance measured by 4 methods

	Most													least
Magnitude	11	1	4	9	13	7	5	14	8	6	10	12	2	3
Sensitivity (Standard deviation)	7	4	6	12	8	10	13	5	3	2	11	1	9	14
Sensitivity (Mean)	11	9	14	1	5	4	7	12	13	6	3	2	8	10
Distinct	1	9	11	14	4	8	3	2	5	7	10	12	6	13

Based on the final input features selection in Table 3, there are some similarity among these two tables. The feature 1 and 11 are not only selected after the optimization of evolutionary algorithm, but also identified as the top 4 important features by the importance analysis methods. However, the feature 2, 8, 10 and 13 are identified not that important that their rank in the importance table is lower than the author's expectation. Therefore, it is confidently to identify the feature 1 and 11 are the most important features.

As for the aspect of data mining, it is also considerable to use the feature 1, 2, 8, 10, 11, 13 as the "minimum set" of the input features and such finding is also interesting. In this mark prediction context, the major and the stage of a student plays an important role on their final results. Besides, it is confident to say that the evaluation item of h1, lab7, p1 and midterm exam has much more influence on the result than other items. For practical purposes, both the students and the lecturers should think highly of such evaluation items.

Parameters of neural network Generally, the optimization result for training epochs has the best convergence at the final generation, where the number 360 takes 90% of the populations. Thus, for this mark prediction neural network model, such training epochs is highly convinced. The rest two parameters has shown a obvious distribution for each number but the modes do not occupy as large proportion as the training epochs. It may need more generations to train to obtain the larger proportion. Besides, it is interesting that the number of hidden neurons is equal to the original size of input features. Since the maximum size of the hidden neurons in the chromosome set to 14, some extra experiments like extending this size beyond 14 could be taken to explore whether there are some positive correlations between the accuracy and the number of hidden neurons in the future.

4.3 Retraining results

Generally, the new setting of the features obtain a very good result, where the final accuracy of the model increased from 85.35% in previous paper to 91.8%. However, since the training epochs increases 3 times than the previous one, over-fitting occurs in the 8th fold, where the accuracy of the test is significantly lower than the training. And since such optimization doesn't involve the initial random weight assigned by pytorch, there still 3 folds have all prediction equal to 0, which leads to negative R^2 score. Some further experiments can be designed to have such values optimized along with the other parameters.

5 Conclusion and future work

This paper extends and optimizes the previous paper with the evolutionary algorithm. After 100 generations of optimization, a new setting of input features and the parameters of the neural network is generated based on the allele frequency of the final generation. Finally, the neural network model is retraining with this new setting and the overall result is better than the previous paper, where all the input features are used and the parameters is manually selected.

There are still potential exploration related to the evolutionary algorithm in this context of mark prediction. Firstly, extend the range of the current parameters and see whether such values would hold with more competitors. Secondly, get more hyperparameter, like the initial weight of the training, involved in the evolution rather than randomly assigned a value every time. Although such actions may need more generations and time to train and optimize, it is still worth trying.

References

1. Gedeon, T. D., Harris, D.: Network Reduction Techniques. In: Proceedings International Conference on Neural Networks Methodologies and Applications, vol. 1, pp. 119–126. AMSE, San Diego (1991)
2. Gedeon, T. D.: DATA MINING OF INPUTS: ANALYSING MAGNITUDE AND FUNCTIONAL MEASURES. International Journal of Neural Systems **8**(02), 209–218 (1997)
3. Tchaban, T., Taylor, M. J., Griffin, J.P.: Establishing Impacts of the Inputs in a Feedforward Neural Network. Neural Comput & Applic **7**, 309–317 (1998)
4. Montaño, J. J., Palmer, A.: Numeric sensitivity analysis applied to feedforward neural networks. Neural Comput & Applic **12**, 119–125 (2003) https://doi.org/10.1007/s00521-003-0377-9
5. Gaughan R.: What is the Relationship Between Allele Frequency and Evolution? . Sciencing, https://sciencing.com/relationship-between-allele-frequency-evolution-18492.html. Last accessed 26 May 2020
6. He, J., Yao, X.: Drift analysis and average time complexity ofevolutionary algorithms. Artificial Intelligence **127**(2001), 57–85 (2001)