Improve Neural Networks with Hyperparameters Optimization and Pruning with Distinctiveness

Yuchen Gao

Research School of Computer Science, Australian National University
u6248355@anu.edu.au

Abstract. The objective is to explore two ways to improve the neural networks. First to determine whether evolutional algorithm can help to choose better hyperparameters. Second to explore the effect of pruning technique with distinctiveness. With the provided dataset, a classification problem was modelled. The article applied the Pytorch library to implement a multi-layer neural network. The result revealed that the data from observers' eye-gaze could highly classify their response correctness. The result of pruning technique with distinctiveness was worse with respect to time than expected. Then, with evolutional algorithm employed, the result remained stable as the original model result. Also, the article explored the effect of different number of population and generation in evolutionary algorithm and found that there were few changes in experiments.

Keywords: Multi-layer neural network, evolutional algorithm, prune hidden neurons with distinctiveness

1 Introduction

According to the Wiktionary [1], the mind-reading, also called thought-reading, is explained as the ability to sense what others are thinking. In fact, with the development of research [2] on micro-expression the analysis of human micro reactions can reflect mental activities, even recognize lies [3]. Additionally, the analysis of micro-expression is increasingly applied in our daily life. Based on the research [4], the researchers explored comparison the patterns of visualizations during observers answering the question. According to their result, there was few differences between two graph visualisations and observers' correctness. With significant potential values of eye-gaze, this article was expected to acquire more mental information about the observers and raised a problem that whether correctness of observers' response could be predicted just by data from their eye-gaze.

Today in the information age, with increasing number of businesses concerned with neural networks, the result of model is received more attention. There are increasing number of techniques appearing to improve the efficiency of neural networks to reduce the amount of information. On the one hand, a large amount of people pay attention to the optimization during the neural networks training process. According to results of researches [5][6], pruning the neural network with distinctiveness of neurons is a kind of technique which can reduce the number of unnecessary neurons to simplify the model hyperparameters. In addition, the technique can reduce the memory and save time for calculation.

On the other hand, others focus more on the optimization before neural networks training process. Neural networks can reach a better result by adjusting hyperparameter settings. According to the research [7], it discussed the importance of hyperparameters for neural networks. Bad hyperparameter settings would cause the model failure. The research applied the genetic algorithm to seek suitable hyperparameters and tuning. Also, the research [8] employed the genetic algorithm to find optimize the hyperparameters of the neural networks in actual business. The result showed the genetic algorithm could automatically find the best hyperparameter to improve the model.

This article first built a multi-layer neural network (MLP) in Python with Pytorch library to train the dataset. Then we employed the pruning technique with distinctiveness to explore its effect in terms of time and space of the neural network. Combined with the result of pruning, instead of using manual adjustment on hyperparameters, the evolutionary algorithm was employed to explore the better hyperparameter settings on the MLP model. The evolutionary algorithm was a simple genetic algorithm which simulated the Darwinian Evolution Theory – natural selection and survival of the fittest to explore the possible hyperparameter settings. We conducted comparative experiments to explore the effect of initial number of population and generation. With the result of evolutionary algorithm, corresponding hyperparameter settings were retested on neural network.

2 Method

2.1 Dataset and Problem of Classification

According to dataset [4], the data was collected from the during total six questions were asked to 24 observers for each visualization (radial and hierarchical). The statistical data is about observers' eye gaze fixations and saccades while they search for answers from two graph visualisations (radial and hierarchical) of public data. The data set contains the fundamental information (ID, Age, Gender, Education, Major, Language, Vision) of the observers and the detailed measure (Interface, Question, Correct Response, Response Time (s), No. of Fixations, Average Saccade Duration (ms), Average Fixation Duration (ms), Mean, Std, Max, Min, Range, First Derivative, Second Derivative) of eye pupil data.

The problem modelled was whether correctness of observers' response can be predicted just by data from their eye-gaze, which was different from the topics explored in the research [4]. The target was the column Correct Response whose type was categorial, as a result of which the problem was a classification problem. Then, the columns (Response Time (s), No. of Fixations, Average Saccade Duration (ms), Average Fixation Duration (ms)) was chosen as input features, which could represent the micro-reaction of the observer. According to the result [4], correctness rate was 80.6% and 81.3% correct for the radial and hierarchical visualisations respectively. It was hard to classify the correctness from different visualisations (radial or hierarchical). Therefore, we did not include the column (Interface) into input features.

According to the dataset quality, there was only one record with nan value in the input features and target. We removed the record with nan values. In terms of the neural network model, we normalized the input features into the interval [0, 1]. As for split data for training, test and validation data, we shuffled the normalized data and random chose 70% data as training data for training model, 20% as test data for evaluating the model and 10% for validation.

2.2 Model of Neural Network

Hyperparameters. We applied the Pytorch to define a simple multi-layer neural network which only contained three layers, including input layer, hidden layer and output layer. In terms of the hyperparameters, we set up the number of input neurons and output neurons according to input and target columns. Based on the rules [9], we defined the number of hidden neurons as 2/3×input neurons + output neurons and made sure it is an integer. And we set up the loss function as cross entropy, the optimizer as Adam function, and activation function as sigmoid. Then, we adjusted the remaining two hyperparameters (learning rate in range of [0.001, 0.05], epoch in range of [500, 5000]) because of random split of train, test, validation test. For each time the final results of the same setting of learning rate and epoch might be different. We conducted 100 times experiments to get the average results of validation data. Comparing the results of validation data, we decided to define the learning rate as 0.005 and epoch as 2000.

Evaluation. According to problem of classification, we created confusion matrix and calculated the accuracy to justify how well the neural networks classified the test data. The confusion matrix could reveal the details of the classification result – the number of data with different actual and predicted values. The accuracy could present the percent of data which are classified into right class.

2.3 Improve the Model of Neural Network with Pruning by Distinctiveness

Hyperparameters. We set up the number of hidden neurons as 2×input neurons according to the Rule of thumb method [9]. And only when the hidden neurons were over the number of input neurons and over the number of output neurons would we prune the unnecessary hidden neurons. Other hyperparameters were the same for the model as well.

Prune with Distinctiveness. Based on the description and rules of pruning in research [5], we found and established the vectors for each hidden neuron in model while training process. According to the formula of dot product in linear algebra, we calculated the angle of vectors with formula:

$$\text{angle}(a, b) = \arccos\left(a \cdot b \,/\, (\|a\| \cdot \|b\|)\right) \tag{1}$$

Because of the activation function was the sigmoid function, whose output was in range [0, 1]. With the formula (1), the angle of vectors was in range [0°, 90°]. Then we added a bias (-0.5) to each activation function output of each neuron vector, so that the output was in range [-0.5, 0.5] and the angle of vectors was in range [0°, 180°].

According to the angles of all the pairs of neurons, we got the smallest angle of the two vectors and the largest angle of the two vectors. Then we checked whether the smallest angle was lower than 15° and whether the largest angle was over 165° to decide the neurons needed to be removed. Instead of updating the weight of the hidden neurons to output, we updated the number of hidden neurons. Correspondingly, we rebuilt the neuron network with updated hyperparameters (the number of hidden neurons) and continued to use the pruned model training the data.

Evaluation. In order to evaluate the effect of the technique, we recorded the time the model beginning and finishing to get the execution time. Except the execution time, we compared confusion matrix and accuracy to check whether the result increased. Similar to the previous, 100 times experiments were conducted to get the average results for same hyperparameter settings.

2.4 Improve the Model of Neural Network with Evolutionary Algorithm

Hyperparameters. We set up the number of DNA as 8-bit binary number. The DNA contained three hyperparameters for the multi-layer neural networks – first 2 bits for number of hidden neurons, middle 3 bits for leaning rate and last 3 bits for number of epochs. Based on the Rule of thumb method [9], the number of hidden neurons is in range [3, 6]. The learning rate was in range [0.001, 0.1] and the number of epochs was in range [100, 3000]. As for other hyperparameters for neural network were the same. Then, we defined the cross rate as 0.8 and mutation rate as 0.03.

first 2 bit	hidden neurons	middle 3 bit	lr	last 3 bit	epochs
00	3	000	0.001	000	100
01	4	001	0.005	001	300
10	5	010	0.01	010	500
11	6	011	0.015	011	1000
		100	0.02	100	1500
		101	0.03	101	2000
		110	0.05	110	2500
		111	0.1	111	3000

Fig. 1. DNA was divided into 3 parts, first 2 bit, middle 3 bit and last 3 bit, correspondingly to the hyperparameters of neural network, number of hidden neurons, learning rate (lr in the figure) and epochs. This figure showed how the DNA could be translated to the hyperparameters.

We tried different population size and number of generations to acquire the final result of validation. All the initial population were produced randomly. Because DNA represents different combinations of hyperparameters for neural networks, the results of validation were recorded in terms of F1. The validation data was intended to evaluate the parameters for model. Also, we used F1 as fitness value for each in the population in the evolutionary model. F1 was a combination of precision and recall from confusion matrix to represent the fitness of the hyperparameter settings (number of hidden neurons, learning rate and number of epochs). The employment of F1 would has a good balance between each class in confusion matrix.

Evaluation. According to the results of evolutionary algorithm, we could acquire the most fitted DNA for each experiment. Translated from DNA, different combinations of hyperparameter settings were applied on the neural network. Comparing the confusion matrix and accuracy of the result of test data, we could explore the effect of evolutionary algorithm – whether the hyperparameter settings were better. Also, we compared the results of two effects of the improvements on the multi-layer neural networks.

3 Results and Discussion

3.1 Results

	Confusion Matrix		Accuracy (%)
Training Data	13.93	24.50	85.09
	5.65	158.19	
Test Data	1.71	3.89	82.57
	0.98	21.38	
Validation Data	3.28	7.69	83.04
	1.90	43.90	

Fig. 2. Three-layer neural network without pruning, with hyperparameters: number of input neurons = 4, number of output neurons = 2, number of hidden neurons = 6, learning rate = 0.005, number of epochs = 2000. The figure showed the confusion matrix and accuracy for each data including training, validation and test data. All the values in the figure were the average of 100 results of the same hyperparameters setting.

We tried all the combinations of different hyperparameter settings manually including the learning rate and number of epochs. For each experiment, we recorded 100 times and calculated the averages values of confusion matrix, as well as accuracy. And we chose the best result with hyperparameters which were shown in the Fig. 2. During the process, we found this hyperparameter settings were the extreme value. No matter how other hyperparameters changed, the accuracy of validation accuracy would decrease. Therefore, we defined this hyperparameters as our final choice for the multi-layer neural network. The result of the test data revealed that the information about the eye-gaze could significantly classify the correctness rate. And the prediction was about 83% correct.

	Exp 1				Exp 2			
	Confusion Matrix		Accuracy (%)	Time (s)	Confusion Matrix		Accuracy (%)	Time (s)
Training Data	14.11	24.36	85.11	1.86611	15.34	23.13	85.54	1.81564
	5.74	157.92			6.10	157.56		
Test Data	3.21	7.94	82.32		3.50	7.65	82.39	
	2.02	43.05			2.28	43.79		

Fig. 3. Exp1: model with pruning technique, Exp2: model without pruning technique. Both experiments were with the same hyperparameters: number of input neurons = 4, number of output neurons = 2, initial number of hidden neurons = 8, learning rate = 0.005, number of epochs = 2000. The figure showed the confusion matrix and accuracy for each data including training, validation and test data. All the values in the figure were the average of 100 results of the same hyperparameters setting. The implementation environment was: Intel(R) Core(TM) i5-6200U CPU @ 2.30GHz 2.4 GHz, RAM: 8.00 GB.

With the final results in Fig. 3 of comparative experiments, the values of confusion matrix of both model with pruning technique and model without pruning technique were similar, which revealed that the pruning technique has little effect on the final result of classification. In terms of the execution time, it was obvious that the time consumed by the model with pruning is a little more than the execution time of the model without pruning. The execution time presented that the implementation of pruning technique increases the time consumed by the model. Compared with the result of the research [5], the effect of the pruning technique with distinctiveness was much worse in terms of time. Additionally, the final results on test data were almost the same (82.32% and 82.39%). We could still predict the correctness rate from pupil information.

No.	Population	Generations	Most Fitted DNA	hidden neurons	lr	epochs
Exp 3	10	50	[0 0 1 1 0 0 0 1]	3	0.05	300
Exp 4	50	10	[0 0 1 0 0 1 0 0]	3	0.02	1500
Exp 5	30	100	[0 0 1 0 0 1 0 1]	3	0.02	2000
Exp 6	100	30	[0 1 0 1 1 1 0 1]	4	0.015	2000

Fig. 4. Exp 3 – Exp 6: model with evolutionary algorithm. lr: learning rate. All experiments were with the same hyperparameters: DNA SIZE = 8, CROSS RATE = 0.8, MUTATION RATE = 0.03. The figure showed the final best DNA for different population size and generations and corresponding hyperparameters of neural network. The implementation environment was the same, and each Exp cost about 1 hour.

	Exp 3			Exp 4		
	Confusion Matrix		Accuracy (%)	Confusion Matrix		Accuracy (%)
Training Data	13.02	25.32	84.43	16.79	22.17	85.63
	5.25	157.26		6.74	155.42	
Test Data	3.08	7.95	82.89	3.84	6.71	83.88
	1.87	44.52		2.50	44.01	
	Exp 5			Exp 6		
	Confusion Matrix		Accuracy (%)	Confusion Matrix		Accuracy (%)
Training Data	17.17	21.44	86.07	17.42	21.78	86.03
	6.57	156.37		6.47	157.41	
Test Data	4.06	6.91	83.20	3.70	6.93	83.18
	2.69	43.64		2.48	42.89	

Fig. 5. Exp 3 – Exp 6: corresponding to the experiments and hyperparameters of neural network in Fig. 4. The figure showed the confusion matrix and accuracy for each data including training and test data. All the values in the figure were the average of 100 results of the same hyperparameters setting.

According to the Fig. 4, we explored 4 different kinds of hyperparameters for evolutionary algorithm and acquired the final best DNA, which took about one hour for each experiment. And for each experiment, the initial population was produced by random and conducted only once. According to the results in Fig. 4, it revealed that the most fitted DNA was similar when then evolutionary algorithm was with different population size and generations. In total, it was more likely for the evolutionary algorithm to get the most fitted DNA in Exp 5. Then, we applied the corresponding hyperparameters to the multi-layer model. The average results of 100 times were recorded in Fig. 5. Comparing all the confusion matrix and accuracy of test data of Exp 3 – 6, the table presented that each model with different initial population size and generations could acquire over 80% correctness rate while predicting the target by inputs. And based on the Fig. 1, there was little increase or decrease after employing the evolutionary algorithm to choose better hyperparameters for neural network.

3.2 Discussion

We applied the multi-layer neural network with different hyperparameter settings to evaluate the final classification result. Because of the limited time and computer performance for exploring the neural networks, the result just kept stable.

Through the process of implementing two techniques to improve the multi-layer neural networks, we explored the effect of both ways. The dataset size was small and only contained 288 records for training neural network, which was not sufficient to show the effect. In terms of the implementation of pruning with distinctiveness, owe to the limitation on technique implementation, we reduced the number of hidden neurons and rebuilt the model to replace updated weight during the training process, which might cause the bad effect of pruning technique.

Additionally, the implementation of evolutionary algorithm was also flawed. While calculating the fitness value – F1 of the model, each individual in the population only calculated once. The fitness value was not stable and would vary as the random initial population. As a result, some unbalanced dataset or outlier would occur. The fitness value had low representativeness and persuasiveness. We also only explored several values of hyperparameters, which corresponding to limited number of DNA. Additionally, the time spent by evolutionary algorithm were too much to afford.

4 Conclusion and Future Work

Conclusively, we solved the problem of classification and got the result that the data from eye-gaze could efficiently classify the correctness of response. The effect of pruning technique with distinctiveness was worse with respect to execution time. The evolutionary algorithm cost much more time to get the potential better hyperparameters for neural network and improved little on the original model. The result of evolutionary algorithm was similar to the simple model without technique.

Further, we plan to replace multi-layer neural network with a more complex model, for example Convolutional Neural Network. With respect to the evolutionary algorithm, continuous values in hyperparameter settings, corresponding in more DNA in evolutionary algorithm, were better choice to improve the model. And we were expected to calculate the fitness value by repeating one experiment for 100 times. At the meanwhile, the time consumed need to reduce. As for pruning technique, we could update the weight more precisely. More general, large dataset were also needed to improve the difference of the result. We also could apply the evolutionary algorithm to optimize the hyperparameters of model with pruning technique.

References

1. mind-reading - Wiktionary, https://en.wiktionary.org/wiki/mind-reading
2. Duan, X., Dai, Q., Wang, X., Wang, Y., Hua, Z.: Recognizing spontaneous micro-expression from eye region. Neurocomputing, 217, 27-36. (2016)
3. Barathi, C.S.: Lie Detection based on Facial Micro Expression Body Language and Speech Analysis. International Journal of Engineering Research & Technology. (2016)
4. Hossain, M.Z., Gedeon, T., Caldwell, S., Copeland, L., Jones, R., Chow, C: Investigating differences in two visualisations from observer's fixations and saccades. In: Proceedings of the Australasian Computer Science Week Multiconference, p. 7. ACM (2018)
5. Gedeon, T.D., Harris, D.: Network reduction techniques. In: Proceedings International Conference on Neural Networks Methodologies and Applications, vol. 1, pp. 119-126. (1991)
6. Li, W., Plested, J.: Pruning Convolutional Neural Network with Distinctiveness Approach. International Conference on Neural Information Processing, pp. 448-455. Springer, Cham (2019)
7. Jakub, S., Jakub, J., Erik, G., Jan, R.: Genetic algorithm for automatic tuning of neural network hyperparameters. Autonomous Systems: Sensors, Vehicles, Security, and the Internet of Everything, International Society for Optics and Photonics, vol. 10643. (2018)
8. Di, F.C., Dumas, M., Federici, M., Ghidini, C., Maggi, F.M., Rizzi, W., Simonetto, L.: Genetic algorithms for hyperparameter optimization in predictive business process monitoring, Information Systems, 74, 67-83. (2018)
9. Panchal, F.S., Panchal, M.: Review on methods of selecting number of hidden nodes in artificial neural network. International Journal of Computer Science and Mobile Computing, 3(11): 455-464. (2014)