Classification of the Republicans and the Democrats by Applying Pruning and Genetic Algorithm

Tianshu Wang

Research School of Computer Science, Australian National University

u6342392@anu.edu.au

Abstract. In this paper, neural network pruning and genetic algorithm have been implemented on classifying Congressional Voting Records Data Set, two methods both can respectively achieve competitive result among current techniques. When pruning, the distinctiveness has been employed to determine which patterns are unimportant and therefore removable, this method has proven to be not only easy to converge but also more efficient than a standard neural network. On the other hand, it has been founded that genetic algorithm in which inverse mean square error serves as fitness value requires relatively computationally expensive compared to pruning. Moreover, a hybrid method which combines genetic algorithm and pruning has been creatively applied and can yield a better result than that in pure implementation of genetic algorithm.

Keywords: neural network, pruning, genetic algorithm, classification

1 Introduction

It has been suggested that human manner can be regularized and predicted by applying scientific method, even though human manner is based on life experiences, and life experiences vary massively among people (Schlimmer, 1985). With neural network techniques and adequate patterns, plenty of features can be predicted and extracted - for instance, from voting records, machines are enabled to learn recognizing political standpoints of people, specifically the belonged politic parties. Furthermore, given adequate data describing different people from different politic parties, it is feasible to distinguish people from different parties by accessing limited information about them.

In the Congressional Voting Record Data Set, 16 voting decisions (agreed or disagreed) of 435 congressmen (267 democrats and 168 republicans totally) of the U.S have been recorded for every proposed project in 1984 (Schlimmer, 1985). This record dataset enables machines to detect features of the republican and the democrats and form knowledge which is applicable to several field like prediction of future policies and presidents in U.S However, the motivation of choosing the dataset does not only rely on the practical use of it, the dataset is also a fine benchmark which can rule performance of artificial neural network due to its proper size and balanced number of different classes.

Earlier research has proved that some units in neural networks are redundant and introduced pruning by distinctiveness, in which angles between different units have been calculated and according to the result, units which are determined as unnecessary ones would be removed by then (Gedeon et al., 1991).

Genetic algorithm is another breakthrough in the development of neural network. This algorithm was proposed by Fraser in 1957, and it has been inspired by Darwin's theory – mainly Natural Selection and Evolutionism (Darwin, 1859), these are simulated in the algorithm and after an amount of generation processes, optimal or nearly optimal solutions could be generated.

In this study, a single-layer feed-forward network has been trained to classify politic parties of U.S. congressmen by fed with data describing their voting decisions. Our produced results would be compared to that of a published paper written by Liu et al. in 1998, they used Probabilistic Feature Selection in which the same dataset – Congressional Voting Records Data Set has been used to evaluate the performance of their neural networks (Liu et al., 1998).

This paper has focused on implementation of pruning, genetic algorithm and the hybrid of them. By applying these techniques, it is founded that the pruning outperforms both basic neural networks, genetic algorithm and the hybrid method by producing higher accuracy and more concise network structures; for genetic algorithm, due to its high computational requirement when evolution processing, it might be inferior to pruning.

2 Methods

2.1 Preprocessing and Data Splitting

One of benefits of the dataset is that all raw data is already of characteristic type. Decision-making is represented by 'y'(yes) or 'n'(no) and names of politic parties are directly placed in the dataset. Hence, raw data is simple and concise, however, for the purpose of easier and quicker implementation processing, the republican class has been encoded with numeric 0, with the democrat class encoded with 1.

By splitting the whole dataset into training set (80%) and testing set (20%), the neural network can do supervised learning by learning from training set and the effect can be evaluated by testing how many percentage of instances in the testing set can be correctly classified - to do this, each row of data should be split into 2 parts – X and Y, which represent the known features and the unknown one respectively, by doing this we make sure that all tested data is not visible until they are used for testing the learning effect. Although the size of voting-record dataset is not large. K-fold validation is not desirable as it is time-consuming and if to achieve this we will need to run genetic algorithm for k times which is already computationally expensive to run even one time.

	Training Set	Testing Set
X	$348 * 16$	$87 * 16$
Y	$348 * 1$	$87 * 1$

Table 1. Size of datasets and their distribution(row*column)

2.2 Pruning by Distinctiveness

2.2.1 Hyperparameter Setting

Our neural network with 10 hidden neurons applying **sigmoid function** as activation function has been trained for **1000** epochs, and the learning rate has been set to **0.001**. Additionally, **Adam** serves as optimizer. All these hyperparameters were set after compared with other candidates. The comparison results are shown below. Note that each comparison has been obtained by running code over 10 times and they were extracted averages when all other hyperparameters were set to the best fitted ones, in this way inconsistence can be avoided.

	Adam	Resilient Propagation	SGD
Accuracy	95.54%	94.14%	60.70%
Running Time	0.39s	0.78s	0.39s

Table 2. Prediction accuracy and running time on testing set - using different optimizers

	500 epochs	1000 epochs	1500 epochs	2000 epochs
Accuracy	94.38%	97.20%	97.40%	95.45%
Running Time	0.20s	0.37s	0.54s	0.71s

Table 3. Prediction accuracy and running time on testing set - using different number of epochs

	0.1	0.01	0.001
Accuracy	94.84%	93.50%	97.73%
Running Time	0.20s	0.37s	

Table 4. Prediction accuracy and running time on testing set - using different learning rates

2.2.2 Stopping Condition

In our neural network, the stopping condition has been set to limit the number of training epochs to be less or equal to the maximum 1000 - that is one of popular fashions, the reason constraining the number of epochs rather than expected

accuracy is that the size of training set is 348 – it might cause overfitting and take a long time to converge if we stick on high accuracy on predicting accuracy on training set.

2.2.3 Distinctiveness

Basically, the method proposed by Gedeon et al. determines Distinctiveness of units by their output activation patterns of hidden neurons. Vectors were created for neurons when training whose size is supposed to fit the dimension of neurons, they represent significance or functionality of neurons. Therefore, vectors that are measured and recognized as no functionality or insignificant would be removed (Gedeon et al., 1991).

Specifically, the benchmark for neurons which could be left has been given in Gedeon et al. paper. Since all activations varies from 0 to 1, the normalized vector angle is set to 0.5, which is in the range of 0-180°. Nevertheless, angular separations of up to about 15° are considerably similar and one of them is supposed to be removed. The weight vector of the unit which is removed is added to the weight vector of the unit which remains. With low angular separations as above, the averaging effect is insignificant and the mapping from weights to pattern space remains adequate in that the error measure is no worse subsequently. (Gedeon et al., 1991).

Applying the pruning by distinctiveness technique, it is easily founded that in our neural network, the majority part of pruned neurons is from the 15° threshold, it means that the 165° threshold is set to be higher than expected based on our neural network.

	1st run removed neurons	2nd run Removed neurons	3rd run Removed neurons
<=15°	13	16	18
>=165°	0	0	0

Table 5. The number of pairs in which one of neurons is removable in several runs

2.3 Genetic Algorithm

2.3.1 Chromosome

In our genetic algorithm, weight from input neurons to hidden neurons and hidden neurons to output neurons in our neural net can be regard as genes, hence, n-dimensional arrays which represents the combination of genes form our chromosomes – that is the way chromosomes are encoded.

2.3.2 Fitness

Percentage of mean square error(MSE) of each individual in that of total has been served as fitness value, sorting population by descending order, then ranking population based on that order enables the neural network continually to look for networks whose MSE is relatively small and then select them to generate, mutate for reproducing better offspring. The rank-based choice is due to the straight-forward operations when implemented and without tradeoff with accuracy compared to other complex techniques, for example, Boltzmann Selection.

2.3.3 Population

Population consists of individuals. In our problem domain, an individual consists of weights of a neural network. With the population size set to be 100, our implementation generates 100 neural networks at every generation.

2.3.4 Selection and Crossover

One-point crossover based on roulette selection has been applied to simulate the natural reproduction in our implementation. Fitter individuals are more likely to be chosen, recalling the definition of natural selection.

$$\mathrm{P(xi)} = \frac{f(xi)}{\sum_{j=1}^{N} f(xj)} \quad (1)$$

P value represents the probability that each individual being selected to inherit into the next generation.

$$\mathrm{Q(xi)} = \sum_{j=1}^{i} P(xj) \quad (2)$$

Q function calculates the cumulative probability of being selected for individuals, Q value of each individual determines the possibility of being selected by roulette to reproduce offspring. In the process, a random value $r \in [0, 1]$ and individual $\boldsymbol{i}$ would be selected if it suffices the formula below,

$$Q(\boldsymbol{i}-1) < r < Q(\boldsymbol{i}) \tag{3}$$

Also, in each crossover, new generated chromosomes would be a mixture of parents' chromosomes by selecting a random point on every pair of chromosomes, achieving the goal of evolution and obtaining better offspring.

The crossover rate as a part of parameters has been adjusted and evaluated as below.

Crossover rate	0.65%	0.70%	**0.75%**	0.80%
Accuracy	95.41%	95.40%	**96.70%**	92.50%

Table 6. Prediction accuracy on testing set - using different crossover rates

2.3.5 Mutation

In this paper, mutation rate has been set to 0.05, as a hyperparameter, this was determined by comparing different value. Therefore, 5 in 100 individuals would mutate in each generation.

Mutation rate	0.1%	0.075%	**0.05%**	0.025%
Accuracy	89.22%	80.73%	**95.33%**	88.50%

Table 7. Prediction accuracy on testing set - using different mutation rates

2.3.6 Stopping Condition

Stopping condition of genetic algorithm should be the current generation reaching the maximum number of iterations. The choice of maximum iteration should maintain that no over-fitting happened but converge still be reached. Therefore, the following adjusting has been made on our neural network, and the optimal one among these have been chosen by then.

Maximum No. iteration	100	**200**	500	1000
Accuracy	87.35%	94.25%	93.10%	91.95%
Running Time	181.98s	357.75s	883.06s	1770.24s

Table 8. Prediction accuracy and running time on testing set - using different stopping criteria

2.4 Hybrid method combing genetic algorithm with pruning

In the hybrid method, the hyperparameter setting has followed the optimal setting in the implementation of genetic algorithm, pruning is still a post-training process with no re-train, helping to reduce unnecessary patterns, as well as maintaining a lower hardware requirement and relatively high prediction accuracy. Basically, pruning has been appended to the implementation of genetic algorithm in the hybrid method.

3 Results and Discussion

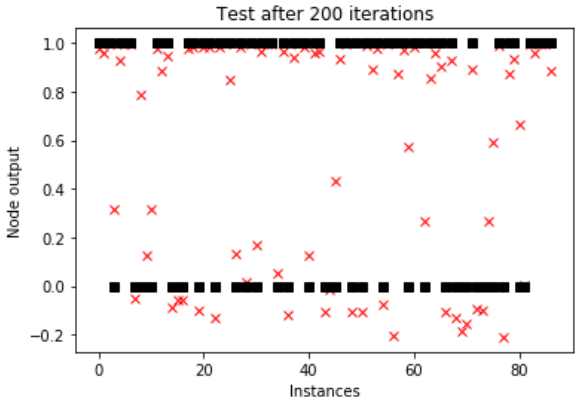
3.1 Comparison between neural network with and without pruning

	1st run	2nd run	3rd run	4th run	5th run	Average
Pruning	94.87%	97.00%	97.73%	94.19%	96.63%	96.09%
No pruning	96.25%	94.51%	94.68%	94.81%	96.88%	95.43%

Table 9. Prediction accuracy on testing set – pruning and without pruning

As observed, pruning with distinctiveness yields a minorly better result than basic implementation of neural network, with less patterns in the net, the results validates the correctness of mechanism raised by Gedeon et al. – that pruning is practically useful to achieve high accuracy while redundant units being removed.

3.2 Genetic algorithm



red crosses represent the predicted result while black points being the actual nodes.

Figure 1. Distribution of the actual nodes and the predicted nodes – genetic algorithm

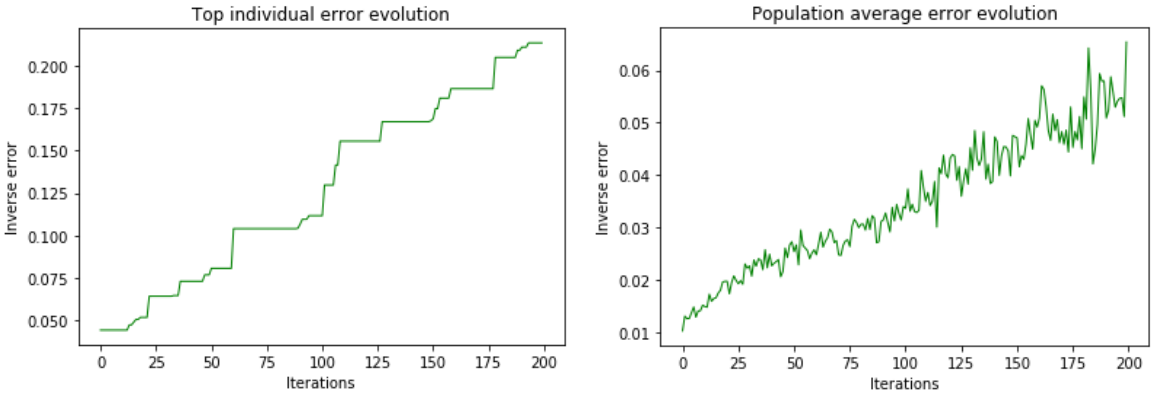


Figure 2. Top individual error and population average error evolution trend obtained in testing set by genetic algorithm

	1st run	2nd run	3rd run	4th run	5th run	Average
Genetic algorithm	93.10%	88.50%	95.40%	96.55%	95.40%	93.79%
Running time	356.59s	357.44s	356.83s	356.37s	358.92s	357.23s

Table 10. Comparison of prediction accuracy and running time on testing set

As the examined result shown, with the total number of iterations set to 200, a majority of target could be correctly predicted. However, due to the considerable time cost in the complex evolution process, simple neural network outperforms genetic algorithm with high efficiency and better result (accuracy: 96.09% versus 93.79% and running time: <1s versus 357.23s). This indicates that for this problem size, it is more suitable to applying pruning rather than genetic algorithm which might be more efficient when problem is larger.

3.3 Genetic algorithm with pruning by distinctiveness

red crosses represent the predicted result while black points being the actual nodes.

Figure 3. Distribution of the actual nodes and the predicted nodes in testing set - genetic algorithm with pruning

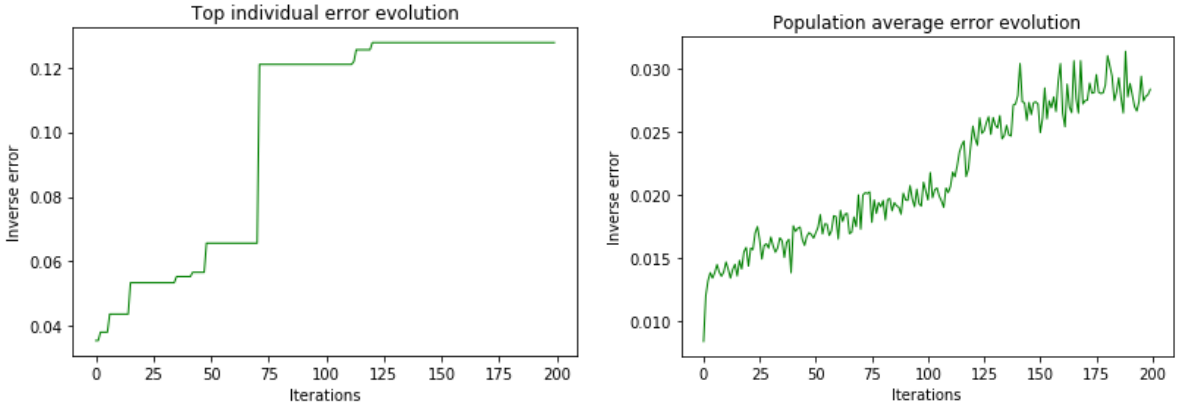


Figure 4. Top individual error and population average error evolution trend obtained in testing set by genetic algorithm with pruning

Comparing it with the result of pure genetic algorithm, it is obvious that the genetic algorithm with pruning which achieves a peak earlier than the previous method yields a slightly better result. And after averaging the accuracy of this method in 5 runs, we concluded that the general accuracy is 95.40% and it performs a bit better than pure genetic algorithm – this result is surprising however can be expected as neural network with pruning also outperforms plain neural network.

Now we compare the above experimental results and the results in the paper which implemented LVF method and was written by Liu et al. in 1998 (although Schlimmer has published a paper whose main topic is relevant about our chosen dataset, unfortunately we cannot get access to the whole paper hence cannot be compared(Schlimmer, 1987)). It is not difficult to find that the accuracy of the pruning network is higher than that in the paper. For both genetic algorithm which applying pruning and not applying it, the apparent drawback is the expensive running time, whereas, the accuracy is also higher than that in the compared paper. Unfortunately, the paper does not give a benchmark of running time and therefore cannot be compared based on computational efficiency (Liu et al., 1998).

	Neural network(NN)	NN with pruning	Genetic Algorithm(GA)	GA with pruning	LVF
Accuracy	95.43%	96.09%	93.79%	95.40%	94.7%
Running Time	0.38s	0.40s	357.23s	357.65s	unknowm

Table 11. Comparison of prediction accuracy and running time on testing set

4 Conclusion and Future Work

Overall, our artificial neural network applying genetic algorithm and pruning can obtain moderate result. Pruning can efficiently improve the prediction accuracy, whereas, due to the expensive time cost of evaluation in genetic algorithm, it is not practical and not recommended to solve this problem with genetic algorithm when hardware limitation is strict.

Even though we try hard to focus on details on this paper, some limitations remain. One thing is the size of dataset is 435, and the data has been collected 33 years ago, hence our study result is not able to reflect the actual situation or be put into practice. On the other hand, it might not be a good idea to simply compare the performances of pure neural network with pruning and that of genetic algorithm, as in the two methods, there are some significant difference, for example, activation functions and so on.

Moreover, to obtain better accuracy and higher efficiency based on current work, pruning by second order derivatives of the error function is supposed to further generalize our neural networks and lower hardware requirement. Additionally, Fuzzy representation could be expected to be applied in the chromosome encoding which enables neural network to simulate reality better in order to receive a better prediction accuracy.

References

1. Darwin, C., Duthie, J. F., & Hopkins, W. (1859). On the origin of species by means of natural selection: Or, The preservation of favoured races in the struggle for life. London: John Murray, Albemarle Street.

2. Gedeon, T. D., & Harris, D. (1991). Network reduction techniques. In Proceedings International Conference on Neural Networks Methodologies and Applications (Vol. 1, pp. 119-126).

3. Hassibi, B., & Stork, D. G. (1993). Second order derivatives for network pruning: Optimal brain surgeon. *In Advances in neural information processing systems* (pp. 164-171).

4. Liu, H., & Setiono, R. (1998). Incremental feature selection. Applied Intelligence, 9(3), 217-230

5. Schlimmer, J. (1985). *Congressional Voting Records Data Set*. Retrieved from UCI Machine Learning Repository: http://archive.ics.uci.edu/ml/datasets/Congressional+Voting+Records

6. Schlimmer, J. C. (1987). Concept acquisition through representational adjustment.