Input encoding of linguistic descriptions, indicators of hidden neuron functionality and the evolutionary algorithm for feature selection

Tianyao Jiang

Research School of Computer Science, The Australian National University
u6492108@anu.edu.au
May, 2020

Abstract. Classification problems in neural networks can become difficult when data are not numerical, or in other words some linguistic descriptions, due to the fact that textual data cannot be directly plugged into input neurons. Moreover, the techniques for pruning hidden neurons and selecting appropriate features are also essential to improving efficiency and performance of neural networks. The goal of this study is to build a neural network to classify unknown samples of petrographical descriptions to the correct one of four classes. In this study, a feedforward neural network is constructed to check the outcome of input encoding. Distinctiveness angle measure and evolutionary algorithm are applied to structurally simplify the hidden layer and select the appropriate features respectively. The resulting system with our input encoding techniques achieves an average prediction accuracy of 58.21% for the four-class classification problem. Furthermore, our research also shows that by applying the indicator of distinctiveness and genetic algorithm, the performance of our model can be improved further.

Keywords: Feedforward neural networks, Input encoding, Network reduction, Textual input, Feature selection, Evolutionary algorithm

1 Introduction

In practice, dataset may not only contain some numerical data which are easy to process, but also some linguistic terms which are unknow to computers or to be specific, neural networks. A suitable pattern representation is often important for good learning and generalisation in machine learning and also, to be specific, for neural networks [3]. This is why input encoding methods become essential to constructing a more accurate neuron network. In addition, some network reduction techniques are also vitally important to constructing a network with high efficiency and accuracy. Firstly, it is difficult to decide the number of hidden neurons we should include in our networks. Extra neurons will do nothing due to the fact that there is always a single neuron with similar functionality but reduce the speed of the whole neural networks by increasing its size [2]. The genetic algorithm (GA) for feature selection can also help us choose the most useful or in other words, informative features to train our model, so as to obtain better results.
The dataset used in this study is about the information of an oil well in a reservoir located in the North West Shelf, offshore Australia [1]. The reason why we choose this dataset is that all the features of this dataset are petrographical descriptions, and all of them require input encoding. A feedforward neural network (FNN) will be trained based on our encoding techniques, and then we will use the indicator of distinctiveness and evolutionary algorithm to prune this network in both the hidden and input layer respectively. This work targets at encoding textual features and employing network reduction methods to improve its performance further.
This work facilitates the following: improving the accuracy of neural networks for textual input. Since our input encoding techniques can help us convert textual input into numerical, this means that there will be more data we can use to train our models. Moreover, using a suitable method to encode can also help neural networks learn the samples in a dataset better. Improving the efficiency and accuracy of neural networks which are already trained. Our technique of determining the distinctiveness of hidden neurons can help us prune the hidden layer of a neural network, and GA can also reduce the number of input neurons.

2 Data descriptions

An oil well located in the North West Shelf, offshore Australia, provided an analysis report for this field study and there are 226 data samples obtained from it in total. The reservoir consists of sandstones, mudstones, and carbonate cemented facies [1]. In this original dataset, the target label is the porosity which ranges from 2% to 22% and there are 58 features belonging to 6 different characters in total. we use the same method as described in [1] to encode the label into four

classes: Very Poor <5%; Poor 5–10%; Fair 10–15%; and Good >15. Table 1 shows the detailed character–features relationships used in this paper.

Table. 1. Characters and attributes used for porosity classification (Descriptions are from [1])

Characters (no. of attributes)	Descriptions	Features
Grain size (12)	The general dimensions (average diameter or volume) of the particles in a sediment or rock, or of the grains of a particular mineral that made up a sediment or rock.	Very Fine, Very Fine to Fine, Fine, Fine to Medium, etc
Sorting (8)	The dynamic process by which sedimentary particles having some particular characteristic (e.g. similarity of size, shape, or specific gravity).	Well, Well-Moderate, Moderate-Well, Moderate, etc
Matrix (16)	The smaller or finer grained, continuous material enclosing, or filling the interstices between, the larger grains or particles of a sediment or sedimentary rock.	Argillaceous (Arg), Sideritic (Sid), Siliceous (Sil), Sid with Arg, Sid with Sil, etc
Roundness (8)	The degree of abrasion of a clastic particle as shown by the sharpness of its edges and corners as the ratio of the average radius of curvature of the maximum inscribed sphere.	Subangular (subang), Angular (Ang) to Subang, Subang to Subrounded (subrndd), Subrndd to Ang, Subang, Subrndd, etc.
Bioturbation (4)	The churning and stirring of a sediment by organisms.	Abundant bioturbation, Bioturbation, Decrease in bioturbation, Bioturbation in traces.
Lamina (10)	The thinnest or smallest recognisable unit layer of original deposition in a sediment or sedimentary rock	Irregular argular, Irregular Calcareous, Trace of Calcareous, Less Traces, Argillaceous, etc

3 Methodology

3.1 Input encoding methods

In this section, three different input encoding methods will be discussed.

3.1.1 Allocating values evenly distributed

As we can see from Table 1, the features of the characters such as ‘Grain size’, ‘Sorting’, ‘Bioturbation’ and ‘Lamina’ are some words which can be arranged in a sequence. For example, in ‘Bioturbation’, the feature ‘Abundant bioturbation’, ‘Bioturbation’, ‘Decrease in bioturbation’ and ‘Bioturbation in traces’ are clearly in a decrease sequence.

Thus, we can allocate values from 0 to 1 to encode these characters. To be specific, we can use 0.25, 0.5, 0.75 and 1 to encode these four features for 'Bioturbation'. There are two reasons for this encoding. The first one is that it can indicate the strongness of the features, and the other is that it can normalise them into the range of 0 to 1.

3.1.2 One-hot encoding

The character 'Matrix' consists of 16 features which have no connections between each other. We can choose one-hot encoding for this character. For 'Matrix', a vector of length 16 is required to encode it, and when one feature shows on, we will turn the 0 in the corresponding position to 1 and remain the others as 0. When this character is plugged into the neural network, 16 input neurons will be needed. The reason why we choose this encoding method is that every feature will have the same impact on the weight update of our neural network.

3.1.3 Circular encoding

The most complicated character is 'Roundness' as it is composed of some circular features. Since the features in this character are circular values, they cannot be simply assigned values which are evenly distributed. However, we can use other methods. For instance, an aspect of 0 is north, and so is 360. To handle this, we can use a vector representing the aspect such that each vector is a fixed distance from its neighbouring values [4]. For this character, two variables are required. To be specific, we can use (sin, cos) tuples to normalise the values from -1 to 1 to the range of 0 to 1. The most important property of this circular encoding is that the sum of the absolute values of the changes is the same for all adjacent points [1].

3.2 Feedforward neural network model design and performance measurements

A two-layer fully connected neural network is developed for this task. The input layer has 22 neurons which are corresponding to the number of features of our dataset after input encoding. The hidden layer is with specified hidden units, and the decision for this number will be explored in the following section. The output layer has 4 output neurons since the task is a four-class classification problem.

Given that the dataset is small, we choose to use 5-fold cross validation to obtain more robust results. As we can see from Fig. 1, the classes of this dataset are reasonably balanced ('Very Poor', 'Poor', 'Fair', 'Good' are labelled as 0, 1, 2, 3 respectively) so accuracy will be sufficient to evaluate the performance of our neural network [5]. Moreover, the median of the results of 5 folds will be used to represent the overall accuracy.

Fig. 1. Evaluation for balance of the linguistic petrographical dataset

3.3 Local optimal number of hidden neurons experiment

The number of hidden neurons is an important hyper-parameter to tune. The objective of this experiment is to find the local optimal number of hidden neurons that can achieve the highest test accuracy.

A range of number from 1 to 20 is tested using our FNN. The activation function in the hidden layer is Relu since it can reduce the overfitting which is a serious problem for our FNN model [13]. The loss function we choose for our study is cross entropy error function [6]. The Adam optimization algorithm is employed due to the fact that it is proved to be

effective for a broad range of problems and is computationally efficient [7]. The median accuracy of test set for each number of hidden neurons is calculated to reduce the randomness. Moreover, the median test accuracies with respect to the number of hidden neurons are plotted to observe the best local optimal number of hidden neurons.

3.4 Indicator for hidden neuron functionality

The technique we use to determine the distinctiveness of hidden neurons is to calculate the angles between the neuron output activation vectors. To be specific, for each hidden neuron, a vector of the same dimensionality will be constructed as the number of samples in the training set, each element of the vector corresponding to the output of activation of that particular neuron. This vector can show the functionality of this specific hidden neuron. Thus, by comparing the angles between two vectors of different hidden neurons, we can have an indicator for distinctiveness.

Our assumption is that the hidden layer of the neural network uses the sigmoid function as its activation function. Thus, we can have that the output of activation of one hidden neuron h(x):

$$h(x) = Sigmoid(x) \in [0, 1] \qquad (1)$$

We aim to model distinctiveness of hidden neurons as angles between 0 to 180 which means that h(x) should be normalised into [-0.5, 0.5]. Thus, we subtract 0.5 from every element of the activation vector. Then, cosine similarity as shown in equation (2) and (3) is employed to convert real numbers into angles. By using (2) and (3), we can obtain the angles between two arbitrary vectors a and b.

$$\cos(\mathrm{a}, \mathrm{b}) = \frac{a \cdot \mathrm{b}}{|a| \cdot |\mathrm{b}|} \qquad (2)$$

$$\mathrm{angle}(\mathrm{a}, \mathrm{b}) = \frac{\arccos(\cos(a,b))}{180 * pi} \qquad (3)$$

After we obtain the angles between two weight vectors of different hidden neurons, there are two different situations where different operations will be performed. Firstly, if the angle is greater than 165 degrees, these two neurons are considered as with opposite functionality. Thus, both of them will be removed. Another situation is that if the angle is less than 15 degrees, they will be considered with similar functionality. Therefore, one of them will be chosen to prune, and its weight vector will be added to another remained neuron [2].

3.5 Evolutionary algorithm design and implementation

Since after applying input encoding methods, the features will be changed comparing to the original dataset, we can employ the evolutionary algorithm to perform feature selection in order to improve the performance of our model further. The main procedure of the algorithm is shown in Fig. 2. To be specific, the binary representation is employed to indicate whether the feature is selected or not. Thus, each chromosome is of length 22 as there are 22 features in our dataset after applying input encoding. The average accuracy of the predictions on test set for 5 folds is used as criteria of assigning fitness score, which is corresponding to the previous experiments.
Different selection, crossover and mutation methods listed below are experimented due to the reason that they are key to the diversity of the population, which can have a significant impact on the result feature combination.

- Rank-Based Selection: Individuals will be ranked according to their fitness scores, and selection probability of every chromosome is allocated with respect to its rank [8, 9]. Furthermore, Individuals are selected as per their selection probability.
- Roulette Wheel Selection: In this method, all the chromosomes in the population are placed on the roulette wheel according to their fitness value [10, 11, 12]. Each individual is assigned a segment of roulette wheel. The size of each segment in the roulette wheel is proportional to the value of the fitness of the individual - the bigger the value is, the larger the segment is [8].
- Uniform Mutation: The mutation probability is fixed for each generation and each child has the same probability to mutate.
- Non-uniform Mutation: At the early generation, the mutation probability is relatively high, and it will decrease as the population evolves. Similarly, each child also has the same probability to mutate.

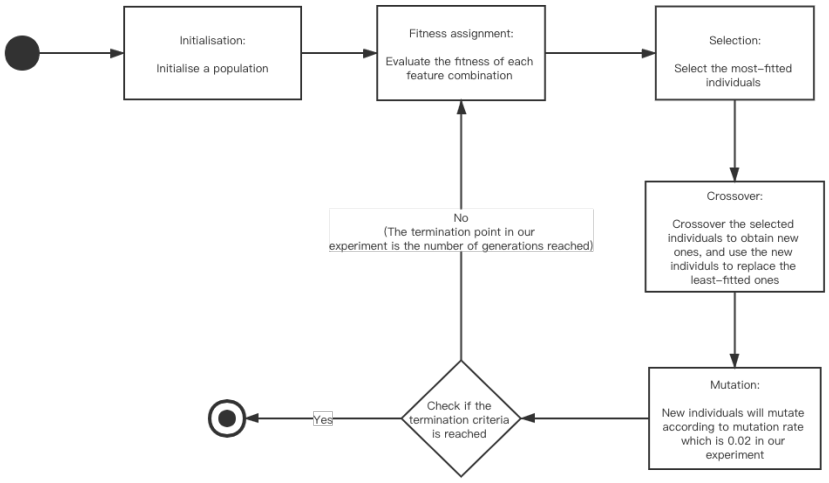

Fig. 2. The procedure of GA for feature selection

4 Results and discussion

4.1 Prediction performance of our feedforward neural network

As we can see from Fig. 3, most accuracies are above 50%, and when the number of hidden neurons is set to 13, the highest accuracy 62.5% can be obtained. Hence, the local optimal number of hidden neurons is 13, and by setting our hyper-parameter to this value, we can achieve that given an unknown petrographical data sample, there is a 62.5% probability that the FNN we constructed will correctly assign it to one of four classes.

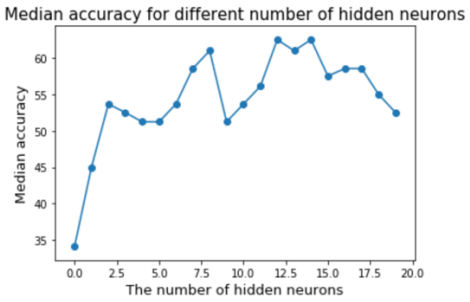

Fig. 3. Median accuracy for different number of hidden neurons

Moreover, the performance of the final model is evaluated, and it will be compared with the result described in [1]. The result of our final model is shown in Fig. 4. Hence, the average accuracy of the model is 58.21% and the highest accuracy achieved by it is 65.85%. [1] claims that the model can achieve 60.0% and 62.8% accuracy in 2-fold cross validation, so the highest accuracy of our model is higher than the result of the original paper while the average accuracy is lower that it. Since the dataset we use is slightly different from the one in [1], the result our model achieved is acceptable.

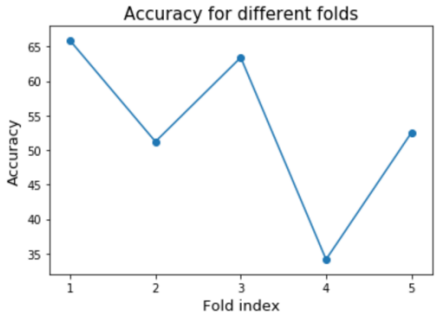

Fig. 4. Accuracy for different folds

4.2 Pruning networks by hidden neuron functionality

This section will be the discussion of applying the network reduction technique described in section 4.4 to our FNN. Firstly, as we can see from Fig. 5, when the training loss decreases significantly, the angles between these 6 neurons will also change by a large amount. Moreover, all the angles are between 60 to 120 degrees at the last epoch, which means that the functionality of these six hidden neurons are all relatively distinctive.

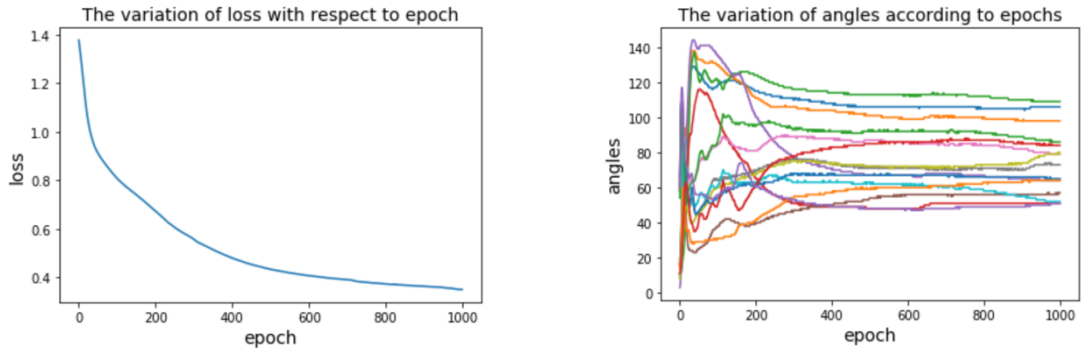

Fig. 5. Training loss and Angles between the last 6 hidden neurons when the number of hidden neurons is set to 13

However, our algorithm detects that the neuron 0 and 1 should be pruned, so the angles between neuron 0 to neuron 5 are plotted in Fig. 6. This figure shows that the angles between neuron 0 and 1 remains 0 degree for every epoch, which means that the functionality of these two neurons are same. The reason why the weights of them are not changed is that the activation function for the hidden layer we chose is Relu which will make some hidden neurons remain non-activated.

Since the angle between these two hidden neurons is less than 15 degrees, we should choose to prune one of them, and add the weight vector of it to another neuron [2]. After applying this network reduction technique, the better performance result is obtained. The new average accuracy is 60.30% which achieves an improvement of 2.09%, and the new highest accuracy is 76.19% which is also a significant improvement.

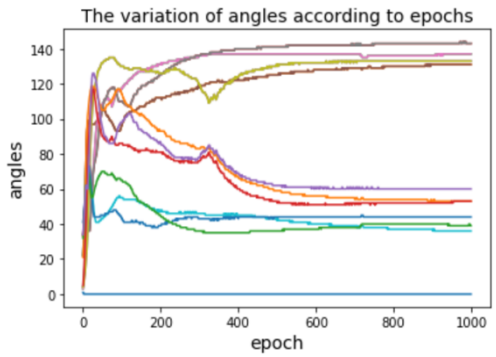

Fig. 6. Angles between the first 6 hidden neurons when the number of hidden neurons is set to 13

4.3 Evolutionary algorithm for feature selection

In order to evaluate the effect of feature selection achieved by the genetic algorithm, the model constructed in section 4.1 will be employed. There are 4 combinations of selection and mutation methods which are experimented in total. every combination was tested only once due to the reason that this genetic algorithm relies heavily on the computation capacity, and that each run of it will consume about 90 minutes.
As we can see from Table 2, employing roulette wheel selection and uniform mutation will help our FNN model achieve the highest average accuracy which is 60.74%. Comparing to our original FNN model, there is an 2.53% improvement in average accuracy. Moreover, the new FNN has only 7 input neurons due to the fact that there are only 7 features selected which are 'Grain size', 'Sorting', 'Matrix2', 'Matrix9', 'Matrix12', 'Roundness2', 'Laminae', so the efficiency of our model is also improved.
However, comparing to other combinations of which there are 12 or 13 features selected, the combination of Roulette Wheel Selection and Uniform mutation has only 7 features selected. This deserves our further research since there are more than half information dropped, which will have adverse impact on the performance of our FNN.

Table 2. Comparing the performance of different combinations of selection and mutation methods

	Rank-Based Selection, Uniform mutation	Roulette Wheel Selection, Uniform mutation	Rank-Based Selection, Non-uniform mutation	Roulette Wheel Selection, Non-uniform mutation
Average accuracy	58.76%	60.74%	56.83%	59.76%
# selected features	12	7	13	12

5 Conclusion and Future work

In our work, a two-layer neural network with 13 hidden units is developed. By applying our input encoding methods, the FNN achieves the average accuracy of 58.21% and the highest accuracy which is 65.85%. The result is similar to the one described in [1], and thus our methods and FNN model are proved to be functional. In the future research, we can employ the same technique which is the 2-fold cross validation used in [1] to perform further comparison. Another research proposal is that more methods of input encoding can be introduced to make the dataset more informative or easy to learn by a FNN.

Secondly, the distinctiveness network reduction technique is proved to be helpful for pruning some neurons with similar or opposite functionality. In our experiment, it improves the performance (average accuracy) of our model by 2.09%. However, since the neurons pruned in our study are non-activated due to the use of the Relu activation function, more experiments can be performed in this topic. Firstly, we can replace the Relu with Sigmoid activation function to follow the assumption of this technique as described in 3.4. Moreover, we can also increase the number of hidden neurons to check whether the number of neurons which are pruned will also prosper or not.

Furthermore, the use of GA for feature selection is also proved to be helpful for reducing the units in the input layer and improving the accuracy of our model. However, due to the computation capacity limit, only some basic selection and mutation methods are tested. Our future research for this area can be employing some more different combinations of these methods, such as Hall of Fame, Elitism and Tournament selection to improve the performance of our model further. Additionally, since some features we use are encoded using one-hot or circular encoding, there will be more than two columns which are occupied to represent the feature. Thus, we can explore that if there is one of the columns is dropped, whether the entire feature should be dropped or not.

When constructing the FNN, only the number of hidden units is tuned, while there are many other hyper-parameters, including learning rate, epoch and batch size. In order to improve the generalization ability of our model further, the tuning of other hyper-parameters should be investigated in the future work.

References

1. Gedeon T D, Tamhane D, Lin T, et al. Use of linguistic petrographical descriptions to characterise core porosity: contrasting approaches[J]. Journal of Petroleum Science and Engineering, 2001, 31(2-4): 193-199.
2. T. D. Gedeon, "Indicators of hidden neuron functionality: the weight matrix versus neuron behaviour," *Proceedings 1995 Second New Zealand International Two-Stream Conference on Artificial Neural Networks and Expert Systems*, Dunedin, New Zealand, 1995, pp. 26-29, doi: 10.1109/ANNES.1995.499431
3. Bustos R A, Gedeon T D. Decrypting neural network data: a GIS case study[C]//Artificial Neural Nets and Genetic Algorithms. Springer, Vienna, 1995: 231-234.
4. L K Miline, Gedeon T D, A.K. Skidmore. Classifying Dry Sclerophyll Forest from Augmented Satellite Data: Comparing Neural Network, Decision Tree & Maximum Likelihood.
5. Montana, D. J., & Davis, L. (1989, August). Training Feedforward Neural Networks Using Genetic Algorithms. In IJCAI (Vol. 89, pp. 762-767).
6. Bishop, C. M. (2006). Pattern Recognition and Machine Learning (Information Science and Statistics) (ch.5 Neural Network)
7. Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.
8. Kumar, Rakesh & Jyotishree,. (2012). Blending Roulette Wheel Selection & Rank Selection in Genetic Algorithms. International Journal of Machine Learning and Computing. 365-370. 10.7763/IJMLC.2012.V2.146.
9. J. E. Baker, “Adaptive selection methods for genetic algorithms”, Proceedings of an International Conference on Genetic Algorithms and their applications, pp 101-111, 1985.
10. D. E. Goldberg, Genetic algorithms in search, optimisation, and machine learning, Addison Wesley Longman, Inc., ISBN 0-201- 15767-5, 1989.
11. D. E. Golberg and K. Deb, “A comparative analysis of selection schemes used in genetic algorithms”, Foundations of Genetic Algorithms, San Mateo, CA, Morgan Kaufmann, pp 69-93, 1991.
12. K. A. De Jong, An Analysis of the behavior of a class of genetic adaptive systems (Doctoral dissertation, University of Michigan) Dissertation Abstracts International 36(10), 5140B University Microfilms No. 76/9381, 1975.
13. Xu B, Wang N, Chen T, et al. Empirical evaluation of rectified activations in convolutional network[J]. arXiv preprint arXiv:1505.00853, 2015.