Determining Input Contribution for Classification Neural Networks using Analysis Techniques and Genetic Algorithms

Solomon Uko Inyang

Research School of Computer Science,
Australian National University,
Acton ACT 2601 Australia,
u6701825@anu.edu.au

Commented [S1]:

Abstract. Determining which features are useful for producing accurate models has been a consistent problem for machine learning researchers. We construct a simple feed forward network to predict a student's final grade based on each of their assessment marks before the final exam. The first part of the paper uses weight matrix analysis techniques to determine the input contribution of each feature. The second part of this paper uses a genetic algorithm and a brute force approach to select the features with the highest contribution, retrain the model using these features, and compare the results to the previous model. We then compare network performance between each of these techniques. The results we achieve show that these techniques are an effective way of determining input significance for different features, although the results are less pronounced than in Gedeon's study.

Keywords: mark prediction, data encoding, classification, feature selection.

1 Introduction

Feature selection for neural networks remains a difficult but important task, despite a substantial amount of research being conducted in this area. In "Data Mining of Inputs: Analysing Magnitude and Functional Measures" by Tamas D. Gedeon [1], Gedeon examines the prospect of using weight matrix analysis techniques to calculate and rank the contribution of input features, and then eliminates inputs with a brute force method to determine the effect on performance.

In this paper we replicate this methodology for identifying and ranking input contributions and evaluate the effect the removal of less significant inputs has on network performance. We extend the work by incorporating a genetic algorithm that produces a set of the best input features and retrains the model after eliminating the inputs that are not included in this set. A genetic algorithm is chosen as this is an optimization problem with no optimal solution and thus this algorithm is an appropriate choice. We then compare the results of our brute force method to the baseline network and the genetic algorithm method to determine which is more effective in terms of network performance.

We create a simple feedforward neural network that aims to predict a student's final grade using their marks from assessments prior to the final exam. The motivation for this model is to create a system that gives students the opportunity to see what their final grade could be if they continued at their current work pace throughout the rest of the course. The main investigations of this task are to see if analysis techniques such as analysing magnitude can be utilised to determine the contribution of each input to an output grade, and then evaluate the effect on network performance when we remove the inputs with the smallest contribution. This shows which assessment items have high correlation to the final grade. For example, we could determine if a student that scored highly in the mid-semester exam could expect to receive a good mark in the final exam and receive a 'Distinction' grade. The elimination of inputs is done through two methods. The first is a brute force approach that ranks inputs based on their contribution value calculated from the analysis and eliminates the two least significant inputs. The second is a genetic algorithm that determines the most ideal inputs to use in the network from a pool of candidates through a process of mutation and reproduction.

2 Method

2.1 Preparing the Data

The assigned dataset paper was the "Comparison of Extracted Rules from Multiple Networks" [2] paper by Edwin Che Yiu Choi and Tamas Domonkos Gedeon. The accompanying dataset was a set of assessment marks for a hypothetical Computer Science course "COMP1111". The marks contributed to 40% of each students' total mark, with the remaining 60%, the final exam mark, being left out. The four grades were Distinction - a final mark greater than 75, Credit – a final mark between 65 and 74, Pass – a final mark between 50 and 64, and fail – a final mark less than 50.
In its original state, the data columns were not well defined, and the assessment marks were not in a readily usable form. Thus, the first task was to prepare the data for training and testing. The column names were specified by the assessment names found at the top of the data. Since the majority of assessment marks were numeric, we were required to transform this data from the object type to a numeric type. "N/A" values for each of the data columns were handled by assigning each missing value the average of its column to ensure our data is not skewed. For the columns that were categorical values, such as 'Tutgroup', we use integer encoding to transform their values to be numeric. Lastly, the student registration number column is dropped from the feature set as it does not prove to be useful in training. We endeavored to solve a classification problem, however the target column for this data was the final mark the students received, not their grade. Thus, the final mark column is split into 4 classes for classification, with an output of 0, 1, 2, or 3 corresponding to a 'Distinction', 'Credit', 'Pass', and 'Fail' grade, respectively.

2.2 Training the network

The network topology used was a three-layered neural network consisting of fourteen inputs, five hidden units and four output units. This topology was chosen as it was used in the dataset paper, and it corresponded with the 14 assessment marks (features) and the 4 output grades. In order to remain consistent with the technique paper, the network was trained using error-backpropagation and a basic sigmoid logistic function was used as the activation function. Because we are measuring the performance of a classification model, we use a Cross-Entropy loss function as it minimises the distance between probability distributions. After running the network a number of times with different epoch numbers and learning rates, we conclude that a learning rate of 0.01 and epoch number of 5000 provide the best results.

2.3 Analysis Techniques

We implement the following analysis technique for determining contribution, proposed in the paper "Feature Selection Using Neural Networks" by Linda Milne [3]:

$$\frac{\sum_{j=1}^{nhidden} \frac{w_{ji}}{\sum_{l=1}^{ninputs} w_{jl}} . w_{oj}}{\sum_{k=1}^{ninputs} \left(\sum_{j=1}^{nhidden} \frac{w_{jk}}{\sum_{l=1}^{ninputs} w_{jl}} . w_{oj} \right)}$$

With ninputs representing the number of inputs, nhidden representing the number of hidden units and noutputs representing the number of outputs respectively. A significant disadvantage with this method is that it can result in inconsistent values as the combination of positive and negative weights can cancel out contributions. Thus, we can use absolute values to account for this:

$$\frac{\sum_{j=1}^{nhidden} \frac{w_{ji}}{\sum_{l=1}^{ninputs} |w_{jl}|} . w_{oj}}{\sum_{k=1}^{ninputs} \left(\sum_{j=1}^{nhidden} \left| \frac{w_{jk}}{\sum_{l=1}^{ninputs} |w_{jl}|} . w_{oj} \right| \right)}$$

This technique is effective because it retains the importance of the magnitude of the input and its contribution to each output, whilst disregarding the sign of the contribution which is largely irrelevant.

2.4 Brute Force Analysis

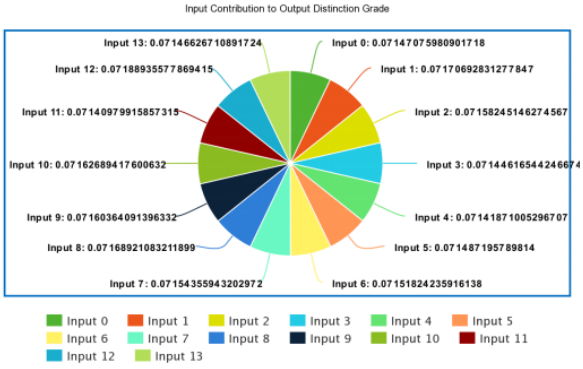
The brute force analysis technique presented in [1] is to eliminate inputs randomly and compare the results of the retrained network with the original. In our paper, we rank the input contributions we obtain and then eliminate the two inputs with the least significant contribution to test the effect on network performance.

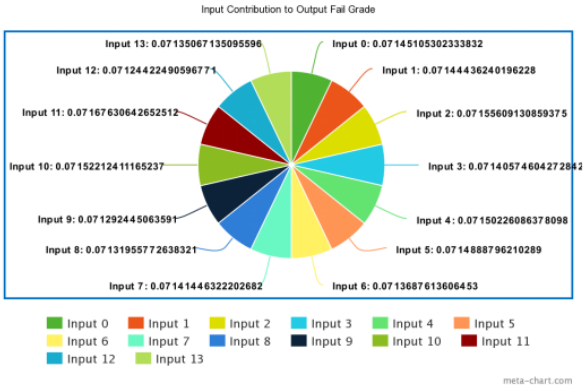
2.5 Feature Selection using a Genetic Algorithm

In addition to the brute force technique for input elimination, we present a method that utilises evolutionary algorithms. In this method, a genetic algorithm is trained to select the most desirable inputs within the feature set. We first assign random truth values for each input that determines the initial population and whether or not a specific will be included in this population. Then, for each generation, we select the four best parents in the population for mating and create the subsequent generation using crossover – reproduction of off-spring based on the genetic materials of the parents – and mutation – a small genetic tweak to the off-spring to diversify genetics. After iterating through each generation, we are left with an array of the most ideal features to use for the network based on our fitness function.
Following this process, we eliminate the columns not chosen by our genetic algorithm from the dataset and retrain the model with this reduced data. We also generate new contribution values for each input.

3 Results and Discussion

3.1 Input Contribution from Analysis





These results show that the input contribution for each feature was quite similar. There was no single feature that had a significant impact on final grade for either the Distinction grade or the Fail grade.

3.2 Performance of the neural network when inputs are eliminated

Table 1. Comparison of network performance with various numbers of features

Trial	Network Performance Before Feature Selection	Number of Features Before Feature Selection	Network Performance Before After Genetic Algorithm Feature	Number of Features After GA Feature Selection	Network Performance Before After Brute Force Feature Selection	Number of Features After BF Feature Selection

			Selection			
1	65.4%	14	61.8%	11	58.6%	12
2	57.6%	14	33.3%	5	77.4%	12
3	58.3%	14	55.2%	8	59.4%	12
4	61.5%	14	45.7%	5	64.3%	12
5	43.3%	14	55.2%	6	68.9%	12
6	50.0%	14	56.7%	7	53.1%	12
7	61.3%	14	62.1%	7	54.5%	12
8	56.2%	14	65.9%	8	59.375%	12
9	56.5%	14	45.2%	6	53.8%	12
10	61.3%	14	65.6%	7	55.2%	12
MEAN	**57.14%**		**54.67%**		**60.46%**	

To test the performance of the network with different feature selection methods, we run a number of trials and then determine a mean accuracy for that method. The mean performance of the network using all the original features was 57.14%. The network performed worse on average when using only features selected by the genetic algorithm, achieving only a 54.67% accuracy. The Brute Force method performed the best of the three achieving a mean accuracy of 60.46%.

Part of the genetic algorithms' poor performance can be attributed to the fact that it always had the smallest number of input features. Generally, having more features is beneficial to training neural networks, so the genetic algorithm using smaller sets of features compared to the original network and the brute force method likely contributed to it having the lowest score.

4 Conclusion and Future Work

In this paper, an analysis technique for determining the input contribution for each output class was evaluated. Two techniques were introduced for the elimination inputs based on contribution: A brute force approach that removed the two least significant inputs, and a genetic algorithm that determined the best feature set. The results show that the analysis technique presented in [1] can be effectively used to rank inputs according to their contributions to each output, although the differences in contribution are not as decidedly clear as they are in Gedeon's study. Further, we can conclude that various algorithms can be implemented to remove certain inputs based on their contributions, such as genetic algorithms or ranking-based elimination, with the latter being generally more successful.

Further work may include using sensitivity analysis as an additional contribution analysis technique, which utilises differentiation to determine the functional contribution of inputs to outputs. It would also be useful to consider how these analysis techniques fare when applied on multilayer networks larger than three-layers. Lastly, it may be beneficial to explore the effectiveness of other types of algorithms that can be employed to determine the most significant inputs, specifically Deep Learning or Fuzzy Logic approaches.

References

1. Gedeon, Tamás D. "Data mining of inputs: analysing magnitude and functional measures." International Journal of Neural Systems 8, no. 02 (1997): 209-218.
2. Choi, Edwin Che Yiu, and Tamás Domonkos Gedeon. "Comparison of extracted rules from multiple networks." In Proceedings of ICNN'95-International Conference on Neural Networks, vol. 4, pp. 1812-1815. IEEE, 1995.
3. Milne, Linda. "Feature selection using neural networks with contribution measures." In AICONFERENCE-, pp. 571-571. World Scientific Publishing, 1995.