academicJournals

Vol. 9(14), pp. 645-651, 30 July, 2014
DOI: 10.5897/SRE2014.5895
Article Number: F21242846447
ISSN 1992-2248 © 2014
Copyright©2014
Author(s) retain the copyright of this article
http://www.academicjournals.org/SRE

Scientific Research and Essays

Full Length Research Paper

Visualization analysis of feed forward neural network input contribution

Jamal Alsakran[1]*, Ali Rodan[1], Nouh Alhindawi[2] and Hossam Faris[1]

[1]The University of Jordan, Amman, Jordan.
[2]Jadara University, Irbid 21110, Jordan.

Received March 31, 2014; Accepted 1 July, 2014

The complexity of domain problem can slow or even hinder the learning process of neural networks. It is rather difficult to overcome such an obstacle because neural networks, as cited today in the literature, lack the interpretability of their internal structures. In this paper, we present a visualization approach capable of enhancing the understanding of neural networks. Our approach visualizes input and weight contributions, sensitivity analysis, and provides guidance in pruning less influential features and consequently reducing the complexity of domain problem while maintaining acceptable error rates. We conduct experiments on various datasets to show the effectiveness of our approach.

Key words: Neural network, visualization, input contribution, sensitivity analysis

INTRODUCTION

The human brain has the ability to perform multi-tasking. These tasks include controlling the human body temperature, blood pressure, heart rate, breathing, and other tasks that enable human beings to see, hear, smell and so on. The brain can perform at a rate that is far less than the rate at which the conventional computer can perform the same tasks (Haykin, 1999). Very little is known about how the brain actually works but there are computer models that try to simulate the same task that the brain carries out. These computer models are called *Artificial Neural Networks* (ANN), and the method by which the neural network is trained is called a *Learning Algorithm*, which has the duty of training the network and modifying weights in order to obtain a desired response.

Achieving a near optimal ANN for a specific problem requires a prior knowledge of the domain problem and an intelligent choice of network parameters such as weights, size of hidden layers, learning rate, etc. On the other hand, the lack of interpretability of the internal characteristics of a trained network hinders the construction of near optimal ANN (Sjöberg et al., 1995). Moreover, when the complexity of domain problem increases, reaching desired results becomes more difficult and unmanageable.

Reducing the complexity of domain problem by removing less influential features can ease the construction of desired ANN. However, as we do not have a good understanding of how neural networks work it is rather difficult to know which features available are the most useful in describing the key properties of the

*Corresponding author. E-mail: j.alsakran@ju.edu.jo
Author(s) agree that this article remain permanently open access under the terms of the Creative Commons Attribution License 4.0 International License

input vector class. Besides, manual pruning of domain problem is considered a tedious and error-prone task (Abraham, 2004).

Visualization can help understand complex systems (Viste and Skartveit, 2004). In this effort, we present a visualization approach to help understand input contributions and internal structures of multiclass feed forward neural networks. Our approach provides the necessary guidance, through means of visualization, to reduce the complexity of the problem, by pruning less important features, while achieving good results with acceptable errors. We attempt to visualize the knowledge learned by neural network such as input and weight contributions, sensitivity analysis. To the best of our knowledge, our visualization approach is the first to visualize input significance of multiclass output classification problems.

RELATED WORK

There has been much research devoted to studying sensitivity analysis and feature extraction of neural networks. Readers are referred to a comprehensive review and comparison of methods to study the contribution of input variables (Gevrey et al., 2003). Garson (1991) and Goh (1995) partition the connection weights to determine the relative importance of inputs. Milne (1995) proposes a modified version of Garson's method to determine input variable importance. Her calculation takes into account that weights can be positive or negative, and gives a better measure of contribution of inputs to outputs. Numerical sensitivity analysis is proposed in Montao and Palmer (2003) to interpret the effect of input variables on output without making an assumption about the nature of the data. More recently, Paliwal and Kumar (2011) propose a method based on the interquartile range of the distribution of the network weights obtained from training the network. They have shown that their method performs better than the connection weight approach proposed by Olden and Jackson (2002).

Visualization of the structure and behavior of neural networks has witnessed some research attention. Fischer and Zell (2000) develop an interactive visualization tool to help gain insights into neural network architectures and how the learning progresses and knowledge are stored in a network. Steeler et al. (2001) describe an interactive tool used to examine the weights and topology of neural networks. An evolutionary adaptation process is used in their tool to allow weights to be adjusted during training. They also present a compact matrix representation allowing many neural networks to be compared and organized in a tree structure. In the work presented by Tzeng and Ma (2005), the significance of input units and connection weights is mapped to color-coded units and edges of varying size, respectively. Their work merely visualizes one output class classification tasks while in our work we deal with multiclass classification tasks. Moreover, we visualize input contribution and sensitivity analysis in pursuit of better understanding of data features and reducing the complexity of the domain problem.

ARTIFICIAL NEURAL NETWORKS

Feedforward Neural Network (FFNN) model is called a multilayer perceptron network that consists of input, hidden, and output layers. Let a FFNN with K input units, N hidden units, and L output units, where $s = (s^1, s^2,\ldots,s^K)^T$, $x = (x^1, x^2,\ldots,x^N)^T$, and $y = (y^1, y^2,\ldots,y^L)^T$, are the inputs of the input nodes, the outputs of the hidden nodes, and outputs of the output nodes, respectively. A three-layer FFNN is shown in Figure 1. In FFNN, all the network weights are assigned random values initially, and the goal of the training is to find the set of network weights that causes the output of the network to match the teacher values as closely as possible.

Input contribution

As proposed by Garson (Garson, 1991), input contribution measures the influence of input towards output class. Input that has large contribution carries essential features of the data. We adopt a modified version of Garson's which is proposed by (Milne, 1995) because it takes into account that weights could be negative or positive. Contribution of input i to output l is computed as follows:

$$\frac{\sum_{n=1}^{N}\frac{|w_{in}|}{\sum_{k=1}^{K}|w_{kn}|}|w_{nl}|}{\sum_{k=1}^{K}\left(\sum_{n=1}^{N}\left|\frac{|w_{in}|}{\sum_{k=1}^{K}|w_{kn}|}|w_{nl}|\right|\right)} \quad (1)$$

where, in general, w_{xy} represents the weight between unit x in layer n and unit y in layer $n + 1$.

Input that has a contribution close to zero can be excluded from training. A large positive contribution of input towards an output favors that output class while a large negative value tends to favor other output classes.

Weight contribution

The knowledge learned by the neural network is internally stored in the network weights. Thus, it is critical to measure the positive or negative contribution of those weights. For hidden-output weights, we measure the contribution by the value of weight they hold when the training process is done as follows:

$$C_{nl} = w_{nl} \quad (2)$$

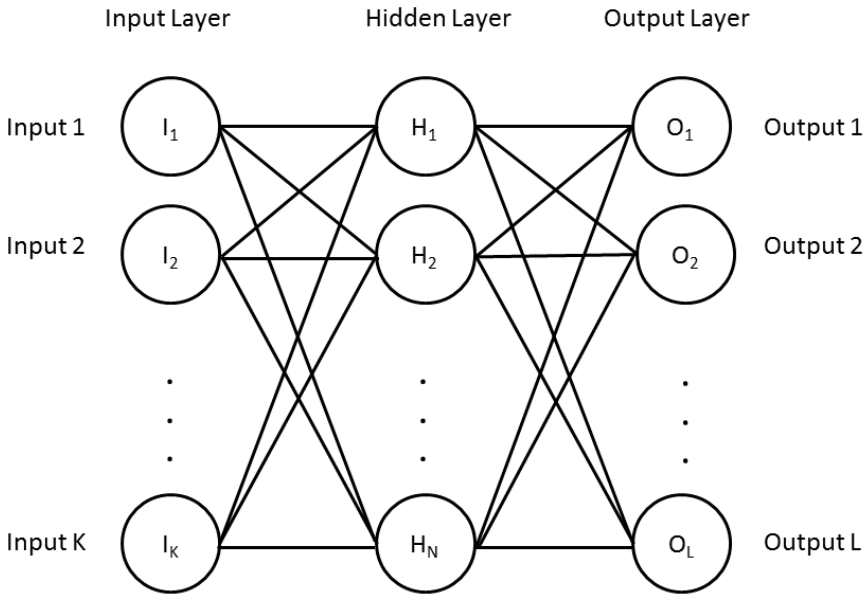

Figure 1. An example of the topology of a three-layer feedforward neural network.

In FFNN, the error propagates back between the current output and target output in order to modify the weights. Therefore, in measuring input-hidden weight contributions we propagate the layers influenced by multiplying the weights between input-hidden by all hidden-output weights connected to them as follows:

$$C_{kn} = w_{kn} \sum_{i=1}^{L} w_{ni} \quad (3)$$

Sensitivity analysis

Sensitivity analysis measures the effect that changes in input I_k have on output O_l. The effect indicates the significance of input on class output. Sensitivity analysis is calculated by taking the partial derivatives of input I_k with respect to output O_l (Montao and Palmer, 2003; Engelbrecht and Cloete, 1998) as follows:

$$\frac{\partial o_l}{\partial i_k} = o_l(1 - o_l) \sum_{n=1}^{N} w_{nl} y_n (1 - y_n) w_{kn} \quad (4)$$

Where y_n is a hidden unit activation, w_{nl} is the weight between hidden n and output l, and w_{kn} is the weight between input unit k and hidden unit n.

The sensitivity depends on information learned by the neural network, which is stored in w_{nl} and w_{kn}, and also depends upon the activation of the neurons in the hidden and output layers. In Equation 4, different input patterns p (input vectors) can provide different values for the change of effect. Thus, the sensitivity S is measured by taking the maximum effect to output due to a change in input as follows:

$$S_{lk} = \max_{k=1,\ldots,K} \sqrt{\frac{\sum_{p=1}^{P} \left(\frac{\partial o_l}{\partial i_k}\right)^2}{P}} \quad (5)$$

where P is the number of input patterns (or input vectors in the data).

Similarly, sensitivity is calculated for hidden units, input-hidden weights, and hidden-output weights as shown in Equations 6, 7 and 8, respectively.

$$\frac{\partial o_l}{\partial y_n} = o_l(1 - o_l) w_{nl} \quad (6)$$

$$\frac{\partial o_l}{\partial w_{nl}} = o_l(1 - o_l) y_n \quad (7)$$

$$\frac{\partial o_l}{\partial w_{kn}} = o_l(1 - o_l) w_{nl} y_n (1 - y_n) i_k \quad (8)$$

NEURAL NETWORK VISUALIZATION

In designing our visualization approach, we opt to preserve the familiar structure of neural network so that understanding our visualization would come natural to anybody who has been exposed to neural network structure. We choose to visualize input, hidden, and output units as round shapes, and weights as edges connecting those round shapes.

Input contribution visualization

Figure 2 shows an example of input contribution

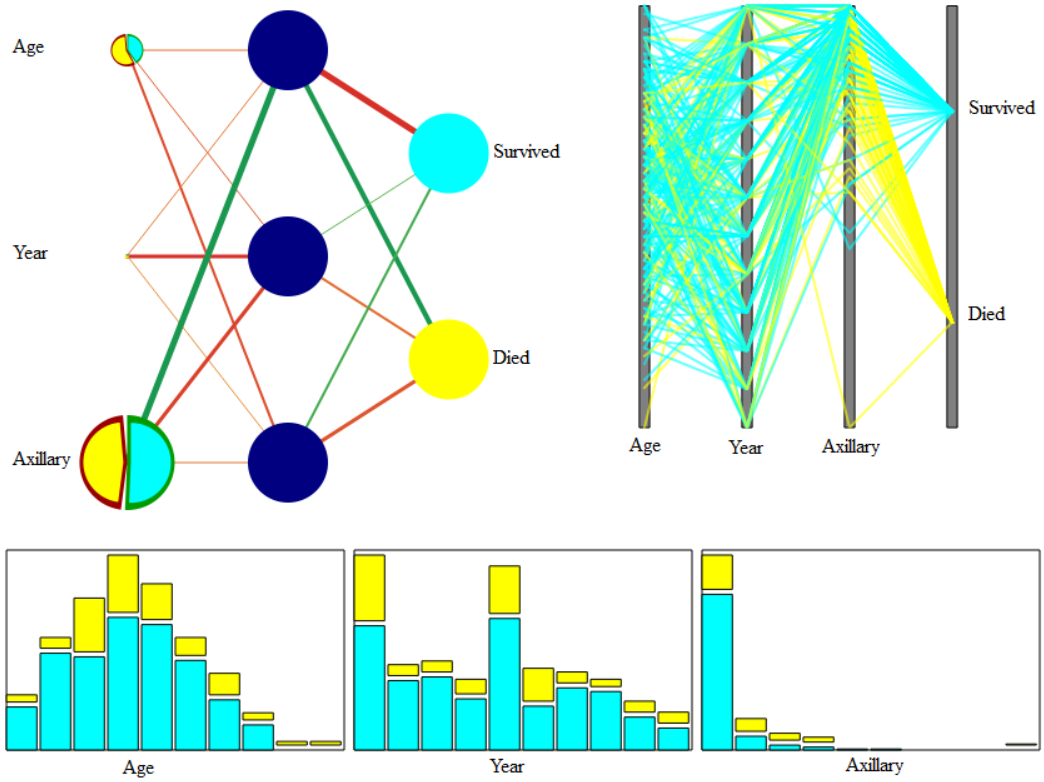

Figure 2. Visualization of input contribution. (Top left) neural network visualization of input and weight contributions, (top right) parallel coordinates plot shows the distribution and relation between inputs and outputs, (bottom) data histogram plot shows data distribution over output classes.

visualization on the *Haberman's Survival* dataset (Bache and Lichman, 2013) which consists of three input variables (age of patient at time of operation, patient's year of operation, and number of positive axillary nodes detected) and two output classes (the patient survived five years or longer and the patient died within 5 years). As shown in Figure 2, the total contribution of a unit towards the class output is represented by the size of that input unit. Within a unit, sectors represent the effect of that input unit with respect to each output class. The sector size signifies the magnitude of the contribution while color indicates to which particular output. As described earlier, input contribution can be positive or negative. Therefore, sectors are surrounded by bands of red color for positive contribution and green color for negative contribution.

Figure 2 clearly shows that *Axillary* has a strong effect on the class output, while *Age* and *Year* have moderate and low effect, respectively. To investigate our results we plot the parallel coordinates Figure 2 (top right) and data histogram Figure 2 (bottom) to study the structure and distribution of the dataset. The plots show that for the less significant inputs (*Age* and *Year*) both classes (*Survived* and *Died*) occur over most of the data range while for *Axillary* the classes do not span the entire range of data; for instance, only *Died* occurs in the upper range of data which makes *Axillary* a more decisive input variable. Besides the overall contribution of inputs, Figure 2 suggests that *Axillary* negatively contributes to output class *Survived* and positively contributes to *Died*.

Weight visualization

The knowledge gained by the neural network is stored, in its entirety, in the network weights. In our visualization, we represent weights as varying-width color coded edges. The width indicates the magnitude of contribution and the color (red for positive and green for negative) indicates the sign of contribution. Figure 2 shows the visualization of connection weights. Most significant weights, for instance the weight between *Axillary* and hidden 1 and weights between hidden 1 and both output classes (*Survived* and *Died*) are readily conceivable which supports our previously mentioned findings.

Sensitivity visualization

Our approach for visualizing sensitivity is similar to our visualization of input contribution previously discussed, where unit size indicates the significance of input and

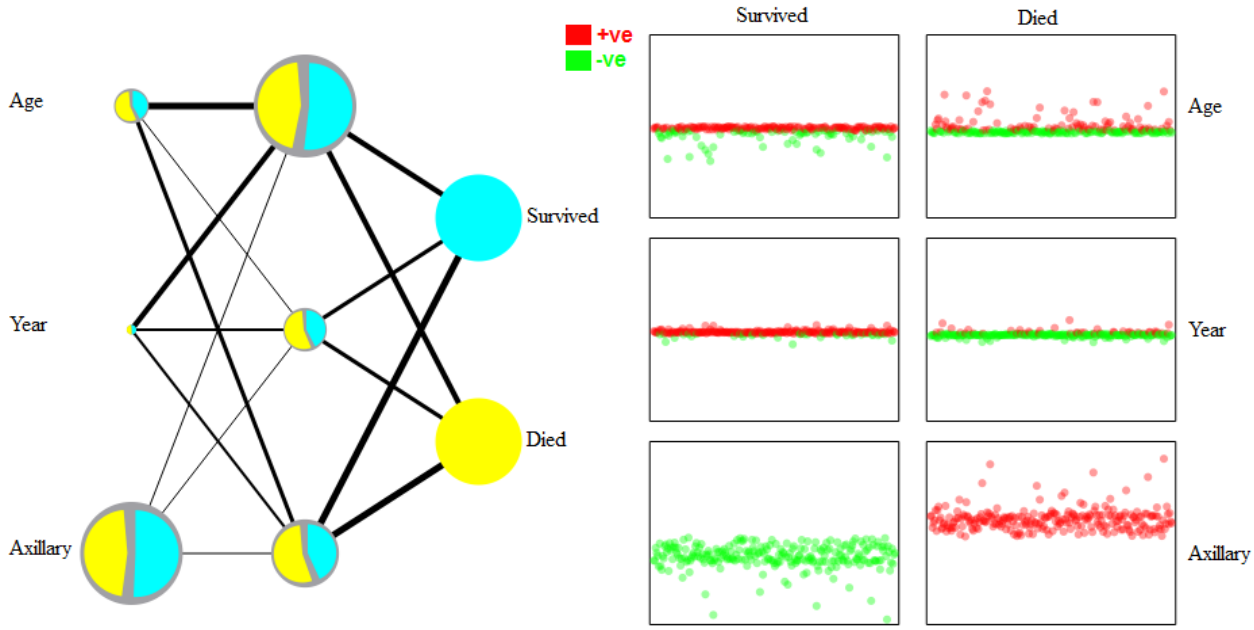

Figure 3. Visualization of sensitivity analysis. (Left) neural network visualization of sensitivity analysis, (right) plot of changes of output for every pattern in the dataset.

hidden units, and edge width indicates the network weights. Unit sectors determine the significance with respect to each output class. As described in Equation 5, sensitivity is defined by the maximum change of input with respect to output class; therefore it does not carry a positive or negative sign. Figure 3 (left) shows an example of our sensitivity visualization. Sectors are slightly spread out and laid over a gray background color to enhance the visual comprehension. In Figure 3 (right), we plot the sensitivity of output classes (*Survived* and *Died*) due to changes in input variables (*Age*, *Year*, and *Axillary*) for every pattern in the dataset. This plot further justifies our findings in Figure 3 (left). It is clearly obvious that *Axillary* is the most influential variable and as suggested by Figure 3 (right) it positively changes with respect to *Died* class and negatively with respect to *Survived* class.

INPUT PRUNING AND ERROR ANALYSIS

When receiving training on a neural network, one should use the smallest system that will fit the data (Reed, 1993). Unfortunately, this is rarely the case because datasets usually contain a lot of noise data that does not contribute much and even slows the learning. In input pruning, we leave out less important features and evaluate the error; if the network performance deteriorates beyond an acceptable rate we plug the features back.

Figure 4 shows an example of our visualization on the *Car Evaluation* dataset (Bache and Lichman, 2013). It contains 6 input variables (*buying*, *maintenance*, *doors*, *persons*, *lug boot* and *safety*) and 4 output classes (*unacceptable*, *acceptable*, *good*, *very good*) and 1728 instances. Figure 4(a) and (b) show visualizations of input contributions and sensitivity analysis, respectively. The figures show that *Safety* contributes the most to the class output and that both *Safety* and *Persons* are the most influential input variables. The figures suggest that the remaining input variables are less important and can be excluded to reduce the complexity of domain.

A common way to carry out input pruning while maintaining acceptable results consists of comparing errors made by the network from the original patterns with the errors made with excluding the input of interest. In Figure 5, we plot error rates resultant from pruning less important inputs. Figure 5 (row 1, left) shows the error rate when including all 6 inputs. Pruning the less important inputs *doors*, *lug boot*, and *maintenance* improves the error rates as shown in Figure 5 (row 1, right), (row 2, left), and (row 2, right), respectively. As expected, the error increases when more important inputs are pruned as shown when inputs *buying* Figure 5 (row 3, left) and input *persons* Figure 5 (row 3, right) are left out. The figure also shows the change of error for each output class due to pruned inputs. The results shown in the figure supports our findings.

Conclusion

We seek to promote the understanding of neural network internal structures by presenting a visualization approach

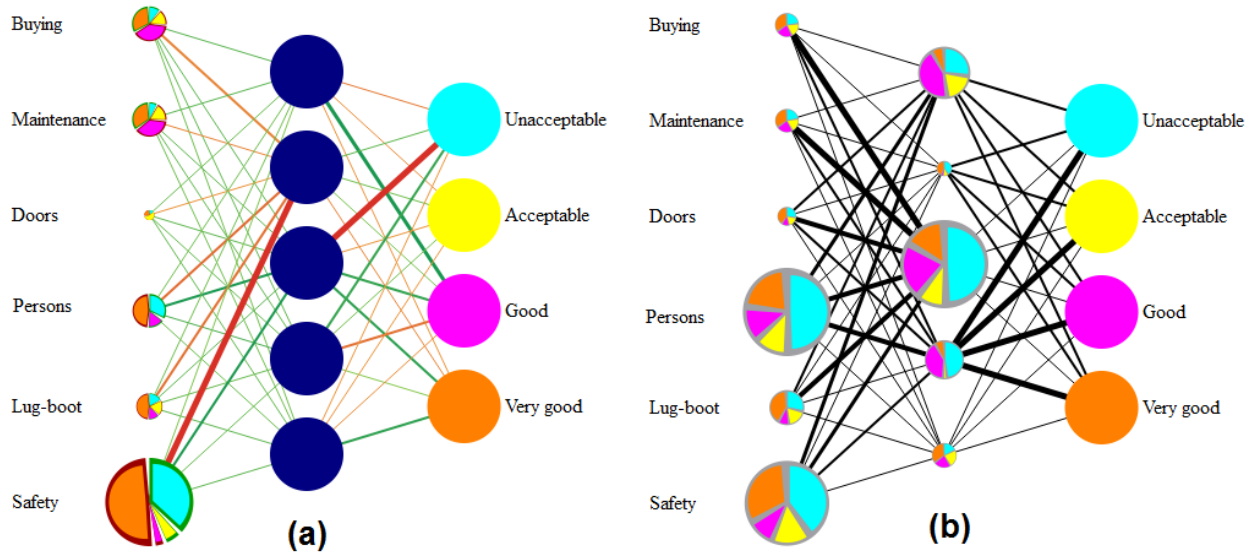

Figure 4. Visualization of car evaluating dataset. (a) input contribution visualization (b) sensitivity analysis visualization.

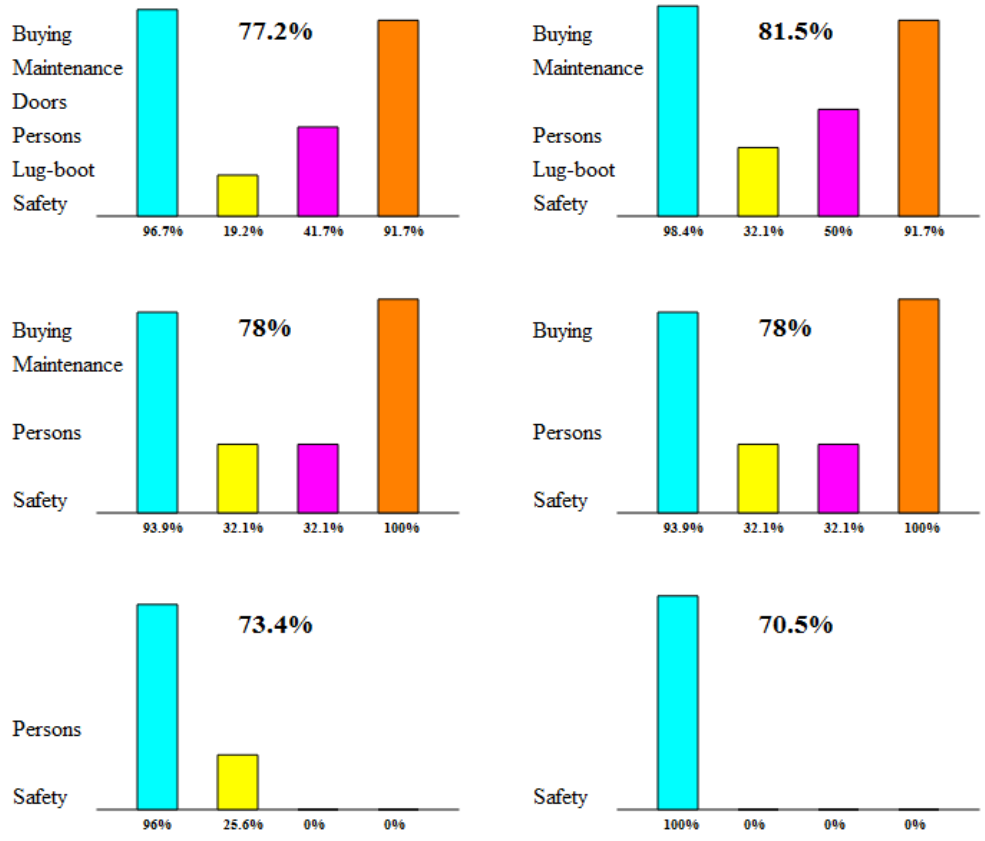

Figure 5. Visualization of error rates after pruning less significant input variables.

that is capable of enhancing the perception of input contributions and reducing the complexity of domain problems. Our approach guides the pruning of less important features found via visual representations of input contributions and sensitivity analysis. We show that our approach can maintain a high rate of performance of

a neural network while excluding noise data and less influential features.

Conflict of Interests

The author(s) have not declared any conflict of interests.

REFERENCES

Abraham A (2004). Meta learning evolutionary artificial neural networks. Neurocomput. 56:1-38. http://dx.doi.org/10.1016/S0925-2312(03)00369-2

Bache K, Lichman M (2013). UCI Machine Learning Repository [http://archive.ics.uci.edu/ml/]. Irvine, CA: University of California, School of Information and Computer Science.

Engelbrecht A, Cloete I (1998). Feature extraction from feedforward neural networks using sensitivity analysis.In Proceedings of the International Conference on Systems, Signals, Control, Computers, pp. 221-225.

Fischer I, Zell A (2000).Visualization of neural networks using java applets. In Proceedings of the 11th Annual Conference of the EAEEIE. pp. 71–76.

Garson GD (1991). Interpreting neural-network connection weights. AI Expert 6(4):46-51.

Gevrey M, Dimopoulos I, Lek S (2003). Review and comparison of methodsto study the contribution of variables in artificial neural network models. Ecol. Model. 160(3):249-264. http://dx.doi.org/10.1016/S0304-3800(02)00257-0

Goh ATC (1995). Back-propagation neural networks for modeling complex systems.AI Eng. 9(3):143-151.

Haykin S (1999). Neural Networks: A Comprehensive Foundation. Princeton Hall, 2nd Edition.

Milne L (1995). Feature selection using neural networks with contribution measures. AI-Conference pp. 571-571.

Montao JJ, Palmer A (2003). Numeric sensitivity analysis applied to feedforward neural networks. Neural Comput. Appl. 12(2):119-125. http://dx.doi.org/10.1007/s00521-003-0377-9

Olden JD, Jackson DA (2002). Illuminating the black box: A randomization approach for understanding variable contributions in artificial neural networks. Ecol. Model. 154(1):135-150. http://dx.doi.org/10.1016/S0304-3800(02)00064-9

Paliwal M, Kumar UA (2011). Assessing the contribution of variables in feed forward neural network. Appl. Soft Comput. 11(4):3690-3696. http://dx.doi.org/10.1016/j.asoc.2011.01.040

Reed R (1993). Pruning algorithms - A survey. IEEE Trans. Neural Netw. 4(5):740-747. http://dx.doi.org/10.1109/72.248452

Sjöberg J, Zhang Q, Ljung L, Benveniste A, Delyon B, Glorennec PY, Hjalmarsson H, Juditsky A (1995). Nonlinear black-box modeling in system identification: A unified overview. Automatica 31(12):1691-1724. http://dx.doi.org/10.1016/0005-1098(95)00120-8

Steeler MJ, Ward MO, Alvarez SA (2001). Nvis: An interactive visualization tool for neural networks. In Proceedings of SPIE Symposium on Visual DataExploration and Analysis. pp. 234–241.

Tzeng FY, Ma KL (2005). Opening the black box - Data driven visualization of neural network. IEEE Visualization. p. 49.

Viste M, Skartveit HL (2004). Visualization of complex systems - The two shower mode. Psychnol. J. 2(2):229-241.