Opening the Black Box — Data Driven Visualization of Neural Networks

Fan-Yin Tzeng* Kwan-Liu Ma*

Department of Computer Science
University of California at Davis

ABSTRACT

Arti cial neural networks are computer software or hardware models inspired by the structure and behavior of neurons in the human nervous system. As a powerful learning tool, increasingly neural networks have been adopted by many large-scale information processing applications but there is no a set of well de ned criteria for choosing a neural network. The user mostly treats a neural network as a black box and cannot explain how learning from input data was done nor how performance can be consistently ensured. We have experimented with several information visualization designs aiming to open the black box to possibly uncover underlying dependencies between the input data and the output data of a neural network. In this paper, we present our designs and show that the visualizations not only help us design more ef cient neural networks, but also assist us in the process of using neural networks for problem solving such as performing a classi cation task.

Keywords: Arti cial Neural Network, Information Visualization, Visualization Application, Classi cation, Machine Learning.

1 INTRODUCTION

In a human brain, a massively parallel information processing system is formed by about ten billion nerve cells (neurons) and their synapses. Arti cial neural networks (ANN) [30] are a class of techniques that mimic the processes found in biological neural networks and well established in the machine learning community for predicting and learning from a given set of data. There are over 50 different types of ANN in use today. ANNs and other learning tools such as Support Vector Machines [3] have gained increased use in a variety of application areas, which should help dispel the misconceptions of Arti cial Intelligence (and ANN).

ANNs are based on the combination of neurons, connections and transfer functions with various learning algorithms and layout methods for the neurons and their connections. After learning, an ANN represents a high-dimensional non-linear function.

A common problem in using ANN is that they act essentially as a black box [25] that performs the assigned tasks for the user. The information stored in a neural network is a set of numerical weights and connections that provides no direct clues as to how the task is performed or what the relationship is between inputs and outputs. This limits the usage and acceptance of ANN since in many applications in science and engineering it is demanded to use techniques based on analytical functions that can be understood and validated. Further complicating the use of neural networks is the tedious process of parameter selection. Even when performing very similar tasks, the proper choice of network parameters can vary widely. These parameters, which include neural network structure, error bound, learning rate, training algorithm, hidden layer size, and the data vector used, are often chosen in a trial-and-error process.

We believe visualization, which proves to help illustrate and understand the behaviors of complex systems, can also help us understand ANNs and design better ANNs. Previous attempts in using visualization to gain understanding into ANNs, as discussed in Section 3, mainly studied the weights and connections of a neural network and analyzed neural networks in isolation; the data used by the neural network were mostly not looked at.

We therefore take a data-driven approach to the problem of visualizing ANN since gaining insights into a neural network requires the study of not only the network but also how it responds to the input data that it was designed to process. The methods we present enable the interactive exploration of both the input data and the neural network so as to gain more complete picture of how the neural network performs its task. The visualizations can also assist in the selection of network structure and other parameters for an assigned task, with the objectives to achieve better results and minimize the cost in terms of both space and time. Equally important is that the user can apply our visualization methods to study how neural networks use data, and gain further understanding into the potentially complex data relationships. In our work, we have applied visualization techniques to feed-forward neural networks trained with the back-propagation training algorithm [23], which is one of the most popular neural networks used for classi cation. Our designs and ndings have helped us develop better intelligent visualization systems, and should also help others gain both understanding and con dence in using ANNs.

2 ARTIFICIAL NEURAL NETWORKS

Figure 1 shows the structure of a three-layer arti cial neural network. Each node in a layer is connected to all nodes (neurons) in the adjacent layer. Each connection between neurons has a weight, with the weights modulating the value across the connection. If the nodes in the input layer are represented by I_1, I_2, I_3, ... ,I_m, the nodes in the hidden layer are H_1, H_2, H_3, ... , H_n, and W_{ij} is the weight on the connection between I_i and H_j, the value of a node in the hidden layer can be shown as

$$H_j = TF(\sum_{i=1}^{n} W_{ij} \times I_i).$$

Likewise, an output node O_k of the neural network can be shown as

$$O_k = TF(\sum_{i=1}^{m} W_{jk} \times H_j).$$

In order for a neural network to model non-linear relationships between inputs and outputs, a non-linear transformation is required. $TF(x)$ is the non-linear transfer function shown in the right side of the nodes in Figure 1. In our work we use the standard sigmoid function which can be expressed as $f(x) = 1/\left(1+e^{-x}\right)$, and is the most commonly used transfer function for classi cation tasks in neural networks. When calculating the value of an output node, the

*IDAV & Department of Computer Science, University of California, One Shields Avenue, Davis 95616, {tzengf,ma}@cs.ucdavis.edu

IEEE Visualization 2005
October 23-28, Minneapolis, MN, USA
0-7803-9462-3/05/$20.00 ©2005 IEEE.

same transfer function is applied after summing up the results from the previous layer. This transfer function can convert the neural network from a linear to a non-linear system.

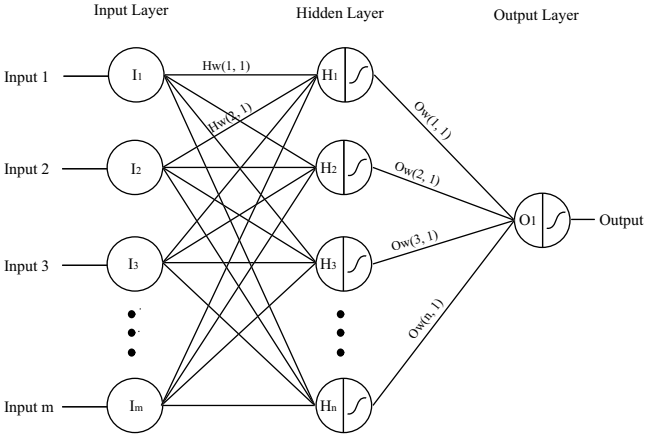

Figure 1: A three-layered artificial neural network

A training process is required to activate the neural network. To train a network, a set of training inputs and desired outputs are required. At the beginning, the weights are set at random, and are iteratively modified to obtain a network which minimizes the error at the output for the training data. Once training has occurred, the network can be applied to data that was not part of the training set.

After training, that is, when the error between neural network outputs and the desired outputs is lower than a threshold, the neural network can be used to process data similar to the training samples by taking the new set of data as input, and calculating the output value O_k using the same formula above.

3 Related Works

To better understand the underlying behavior of a neural network, there has been some research devoted in visualizing the neural networks. Craven and Shavlik [5] surveyed a number of visualization techniques for understanding the learning and decision-making processes of neural networks, including Hinton diagrams, bond diagrams, hyperplane diagrams, response-function plots, and trajectory diagrams. A Hinton diagram uses a data matrix to represent nodes with showing the topological information of the neural network. A Bond diagram shows the neural network topology and applies triangles with different sizes to show the weights. These techniques are used to illustrate the idea of neural networks but are not practical due to the difficulty of showing a large network clearly. Streeter et al. [26] described an interactive visualization tool for feed-forward neural networks. Tree/graph based visualization is used in their work. They display network topology and connection weights as well as the evolutionary adaptation process when the user is allowed to interactively adjust training parameters during adaptation. The weights are used directly without taking the weight in the next levels into account when a large weight does not necessarily indicate the importance of an input. A small weight in the next layer can cancel the influence of the previous weight. They also demonstrated that a larger network can be handled, but when a large number of weights are used, the visualization can become too complex and difficult to understand. In our work, we not only visualize the weights along with the selected data, but convert the weight information and the statistics of the selected data into color and size representations for the input nodes. More recently, Duch [8, 9] introduced a new projection on a lattice of hypercube nodes to visualize the hidden and output node activities in a high dimensional space. This method can be applied to any type of neural networks. However, only the nodes are shown when the connections might also provide valuable information.

In addition to visualization of the neural network, rule extraction and numerical methods are also applied to study the contribution of variables in a neural network [14]. Rule extraction was first mentioned by Gallant [11] to describe the neural networks with a more understandable representation. Andrews et al. [2] and Tickle et al. [28] survey the rule extraction methods and divide them into categories. Garson [12] and Goh [15] multiply the weights between layers to obtain the relative importance of the input variables. Since the absolute values of the weights are used, the result does not provide the direction of the relationship. Olden and Jackson [21] introduce a randomization approach to statistically analyze the input importance based on Garson's method [12]. Dimopoulos et al. [6] compute the partial derivatives of the neural network's output according to the inputs. The results can be positive or negative. If the partial derivative is negative, it indicates that the output of the neural network increases when the studied input variable decreases. A SSD (Sum of Square Derivatives) value can also be calculated which indicates the importance of each input variable. Scardi and Harding [24] modify only one of the input variables at a time and the corresponding output is used to determine the influence of each input variable. The stepwise methods [20, 27] add or reject one input variable at a time and the MSE (mean square error) of the output is used to identify the most important variables. Both the Scardi and Harding's and the stepwise methods are computational expensive since the network needs to be re-calculated a lot of times to obtain results corresponding to different input conditions.

We modify Garson's method for defining node importance in our data-driven neural network visualization, and provide a more convenient way for the user to interpret the results. In addition to node importance, the uncertainty and errors are also visualized and discussed to help both the designer and the user of a neural network.

4 ANN Visualization

Our work focuses on the application of visualizing neural networks. In order to study this application, we apply our techniques in using neural networks to solve two specific problems, volume classification and spam classification.

Neural networks have been used for higher-dimensional classification in biomedical imaging [13, 16, 22] and volume visualization [29] to identify and show features more precisely. In our work, we use the volume classification framework described in our previous research [29]. The neural network is first trained by a small set of input data including the corresponding class IDs provided by the user. The input vector of a training data includes the voxel's scalar value, gradient magnitude, its six neighbors' scalar values, and its position. For example, the user provides sample data from regions of the volume they would like to visualize, and the network can learn to classify the entire volume.

The second application of our work is spam classification. With the wide usage of email, spam has become a problem that limits the effectiveness of email as a communication media. In early approaches, classification rules were defined by hand, but were costly and impractical since the spammer also learns and adapts their messages. Therefore, machine learning techniques are becoming more and more popular for learning and performing text classification [1, 4, 7, 31].

Neural network is employed in our spam classification work. The neural network is trained by a set of pre-classified spam and non-spam email where words or phrases in an email form the input vectors. The trained network can recognize the pattern of spam and filter new incoming messages. The user can then select a para-

graph, an email, or a set of email as the input data when visualizing the network.

The information of a neural network is stored in its weights. However, the weights are dif cult to interpret and represent only through mathematical formula or numerical analysis methods. We show the neural network analysis through visualization since understanding of the data-driven neural network requires interactivity to explore different sets of selected data, and visualization allows the user to perceive and process large amounts of information rapidly and make effective comparisons. The visualization methods include dual-space interactive weight visualization, which allows the user to probe into the data domain and visualize the corresponding network, errors, and uncertainty visualization to help both the designer and the user of a neural network.

4.1 Visualizing Weights – with single data

In our system, the user is provided with a probe to select data of interest in the data domain and visualize the neural network as data are passing through the network. That is, the user is able to look at local properties of the data selected hrough the dual-space interaction.

To highlight the information in a neural network, we color the input and output nodes based on a selected voxel's value, where low value maps to blue, middle value maps to yellow, and high value maps to pink. The data is then multiplied by the weights and added together at the nodes as described in Section 2. When visualizing the weights in a neural network, we focus on the input-hidden layer so that the results can be mapped to the input data domain, which the user is more familiar with. A more important connection has higher weights on it, and its connected nodes have a higher impact to the output result, which is also valuable information about the data. We set the connection's width based on how important this connection is. However, a large weight between the input-hidden layer might connect to a small weight between the hidden-output layer and the effect will be canceled. That is, the resulting visualization could be misleading if the weights were used directly. Therefore, we propagate all the layer's in uence by multiplying each weight between the input-hidden layer and the weight between the hidden-output layer which connect to the same hidden node.

Figure 2 shows the result of visualizing a neural network which performs volume classi cation to classify the brain material from an MRI head data set. The neural network is rst trained by examples of brain and non-brain materials provided by the user. A voxel is then selected to perform the classi cation. When a voxel at the lower left of the slice is selected, the color of the output node shows that this data is not in the classi ed material. In addition, the connections indicate that the position and data value are the main factors of the selected voxel's classi ed result. In the right image, another neural network is shown where the user selected a voxel within the brain, and almost all inputs except the gradient magnitude are used for classifying this voxel to the brain material.

Although the trained network is the same, different behaviors occur when different inputs are selected, and can help the user to better understand the data set.

4.2 Visualizing Weights – with a set of data

In addition to selecting a single piece of data, our system also allows the user to select a region of data or even an entire data set, and visualize the neural network with the selected data.

The connections of a neural network are shown with width representing the weight strength. Input node size is assigned based on the node importance, and input nodes are colored based on their statistical information. A hidden node's size shows its contribution

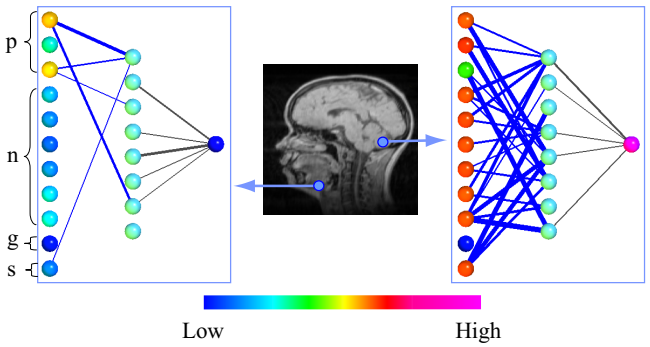

Figure 2: Dual-domain interaction between the data and neural network. In the input layer, s is the scalar value, g is the gradient magnitude, n is the neighboring information, and p is the x, y, and z position of the voxel. When a different voxel is probed, the visualization of the data-driven neural network would change. The left image shows the importance of position to assign the selected voxel on the left to a class, and the right image indicates that classifying the brain relies on all dimensions except the gradient magnitude.

to the nal result. The user can then remove nodes that are not necessary using the computed node importance as a reference.

To estimate the importance of each input variable, we adapt Garson's method to our approach which considers the selected region of data. In Garson's method [12], the contribution of input node i to the output o through a hidden node j is computed by multiplying the input-hidden weight strength and the hidden-output weight strength.

$$c_{ijo} = w_{ij} \times w_{jo}$$

The relative contribution from each input node k to a hidden node j can be represented as

$$r_{ijo} = \frac{|C_{ijo}|}{\sum_{k=1}^{m} |C_{kjo}|},$$

and the total contribution from an input node i is

$$S_i = \sum_{j=1}^{n} r_{ijo}.$$

Finally, we can calculate the relative importance of an input node as

$$RI_i = (S_i / \sum_{k=1}^{m} S_k).$$

The relative contribution is used to show the width of the connections between input and hidden layers.

For two input variables which have the same in uence on the results, the weights connected to them can be very different since the classi cation process includes the multiplication of input variables and weights. For example, if an input variable is small, the connections through this input node need to be larger to bring the input variable to the same level of importance as other nodes which have higher values. However, when only interpreting the network, this property is ignored.

Instead of using the network's weights directly, for selected data we divide the weights between the input-hidden layer by the mean, which is the average of all the data. This can make the node importance evaluation more accurate and speci c to the data used. The use of mean is based on the assumption that the selected data have similar properties so that the input values are close to their mean. To

compensate the estimation of using mean to represent a set of data, we also show the standard deviation (std) on the nodes to indicate data spread. The standard deviation can be represented as

$$S = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(X_i - M)^2},$$

where n is the number of all the data, X_i is the value of the ith data, and M is the mean.

The colors of input nodes are assigned based on the similarity of the mean and standard deviation using the table in Figure 3. The color's red (R) component increases when the mean increases, green (G) increases when the standard deviation decreases, and blue (B) remains constant. From an input node's color, the user can obtain information about what the selected data's distribution is.

In addition to the input nodes, hidden nodes can also provide valuable information. For hidden node j, a value H_j is calculated by passing the mean value of the selected data to the neural network.

$$H_j = Sigmoid(\sum_{i=0}^{n} W_{ij} \times Mean_i),$$

where n is the number of input nodes and W_{ij} is the weight on the connection between input node i and hidden node j. The sum of all hidden nodes is equal to the output of the neural network after applying a sigmoid function. The percentage of H_j to the sum of all hidden nodes can thus be used to represent the relative contribution of H_j to the output, and used to assign the size of hidden node j.

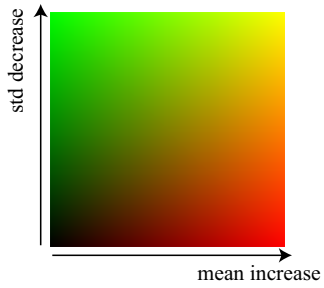

Figure 3: A table for assigning an input node's color based on its mean and standard deviation. A green node indicates the inputs have low mean and high standard deviation, and a red node represents high mean and low standard deviation distribution.

In Figure 4, two neural networks are shown. The lower left image is a neural network used to classify the head material and the rest of the MRI head using five properties as inputs, and all the training data assigned to the head class are used as the selected data for visualization with the neural network. A thicker connection from the input layer shows the corresponding input node is more important to the final result, and there is a threshold that hides the connections that are less important. As shown in Figure 4, scalar value is the most important feature to classify the data when the gradient and position have only minor impacts on the result. This matches the fact that the head can be easily separated with a traditional 1-D transfer function, which maps data value to opacity directly.

The right image in Figure 4 shows the result of classifying the boundaries, which are the regions with high gradient magnitude. In this example, the gradient magnitude and neighbors play a more important role in the classification, and the scalar value and position do not contribute significantly to the final result. Based on the size of hidden nodes, we remove four less important hidden nodes (the 1st, 10th, 13th, and 14th from the bottom) from the neural network after it is trained. The cost of classification is reduced by 15% when the result is only about 0.5% different from the original result.

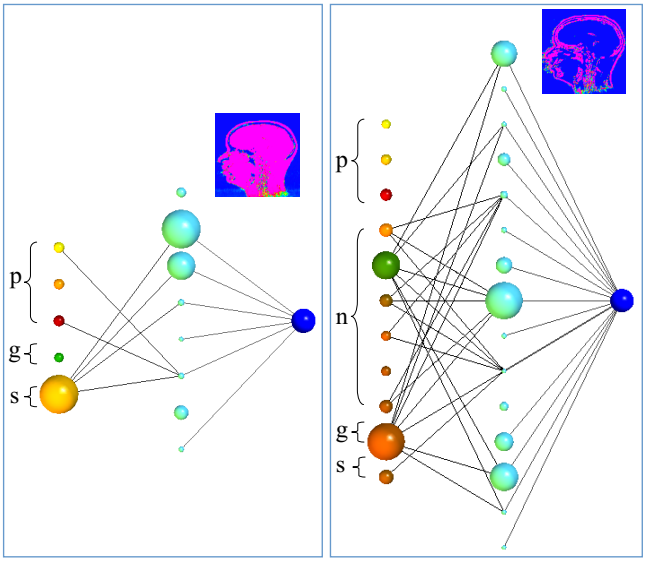

Figure 4: The left image shows a neural network which is trained for classifying the entire head from the data set. The scalar value is the main criterion considered in this classification. The right image is the result of classifying the boundaries. In this case, neighbors and gradient magnitude are shown to be more important. The classification result is shown at the upper right of each network.

The most direct way to measure the performance of a neural network is to look at the error between the training results and the desired outputs over time. This error shows how well the neural network learned to perform classification, and also provides information about convergence. To validate our method and results obtained so far, we calculate the mean square error for different parameter combination, and compare the results with our visualization of neural networks.

Figure 5 presents an example of visualizing both the weights and the errors together using 20 hidden nodes and 11 input nodes. In the top image, scalar and gradient information are shown to be unimportant because the weights are smaller than the threshold and the connections are not shown. This can be validated by visualizing the errors in the bottom image. When only using scalar and gradient as inputs, the mean square errors are high and converge at the end. This indicates that even if more training time is given, the neural network cannot improve further. With additional neighboring information, the neural network can learn better but still with relatively high errors compared to the case of using scalar, gradient, neighbors, and position. The orange and blue curves show the results of removing scalar and gradient from the input dimensions. These two input combinations obtain results similar to when using all input dimensions.

With this system, we discovered that when the hidden layer size is small, scalar and gradient information are more important than the neighboring information because the network is not able to learn the complex relationship and direct criteria for classification such as scalar value and gradient are more helpful. When a larger network is used, it is able to learn indirect relationships such as texture, gradient, and local data range from the neighbors. This makes the scalar and gradient information, which can be derived from the neighbors, less important.

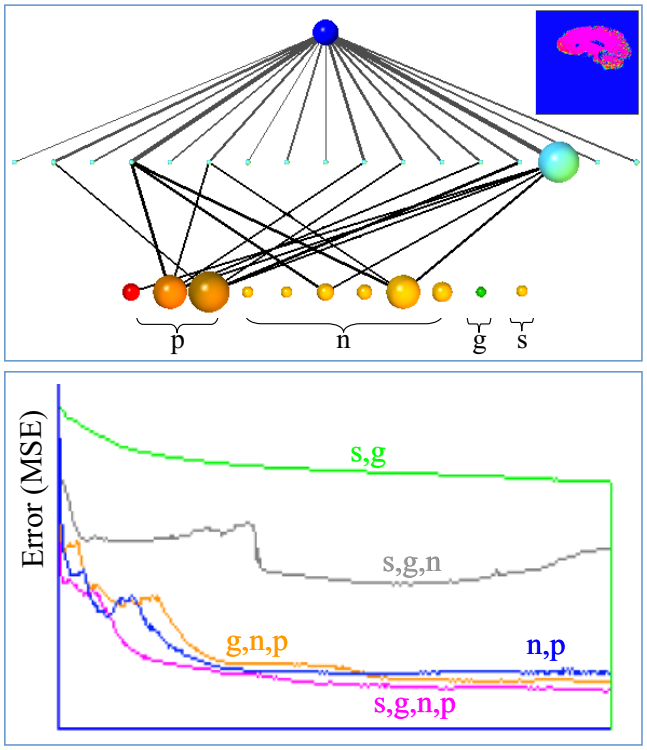

Figure 5: The neural network for classifying the brain material is shown on the right. Scalar and gradient magnitude are unimportant compared to the neighboring information and position, and this can be verified by visualizing errors shown at the bottom.

4.3 Spam Classifier

To demonstrate our methods with a larger network, we choose the application of spam classi cation using a data set that contains 400 email in the training set with 117 spam and 283 non-spam messages, and 200 email messages in the testing set where 61 of them are spam email.

Neural networks are powerful because of their ability to process high-dimensional data and to learn the non-linear relationships between inputs. Therefore, in most neural network applications such as text classi cation, the inputs are high-dimensional. Dimension reduction techniques are common and often required in the eld of neural networks since large networks lead to slow performance. The selection of important input variables and the removal of unimportant ones can help to improve performance when maintaining the classi cation or clustering ability [19].

Figure 6 shows a spam-classi er neural network using 82 input nodes and 100 hidden nodes. The input, hidden, and output layers are shown from bottom to top. The network is trained to assign the email into two classes, spam and not spam. The data feeding through the network is a subset of the spam in the training set.

From the visualization of input nodes, we can identify important terms for classifying spam based on the nodes' visual properties. In Figure 6, the nodes for free and need are large, which indicate that they are two important words to distinguish spam and not spam in the data set. Http occurs in almost all the spam, which is indicated by the red color of the node since red represents high mean and low standard deviation. High mean and low standard deviation are obtained when the elements in the data have similarly high values, and a high value is assigned when the word is in spam. However, http is also a commonly used term in normal email, so the node size is not as big as free and need . In Figure 6, the node represents information is very small since information is used in both spam and not spam with similar frequency, and not a useful criterion for classifying spam. Problem is a medium size node shown in green in the network. Green is assigned to nodes with low mean, that is, the data does not exist in most spam.

When more nodes and connections are used in a neural network, the visualization becomes more cluttered and dif cult to study. Figure 7 shows the result of ordering the input nodes onto a panel according to the statistics information where the colors and sizes of each node in Figure 6 and Figure 7 are the same. From left to right are the input nodes with increasing frequency of appearing in spam. This simpli es the visualization and provides a more organized view of the input nodes.

4.4 Visualizing Uncertainty

During classi cation, the neural network outputs a value representing the uncertainty of the classi cation. High and low values indicate the input belongs to the two user-speci ed classes with low uncertainty, and middle values indicate high uncertainty where the data is dif cult for the neural network to classify. Parallel coordinate [18] is a method to represent multi-dimensional data and is a well-known technique for information visualization [10, 17]. In our work, parallel coordinates are used to show the inputs and outputs when training or classifying using a neural network.

Figure 8 shows the result of using parallel coordinates when the task is to classify volume data into brain and all the other materials. The input vector includes each voxel's scalar value, gradient magnitude, six neighbors' scalar values, and the position. The last dimension of the parallel coordinate system is the output (or desired output for training), and the other dimensions are used for the input vectors. Figure 8 shows the training samples with the desired outputs on the left, and the classi cation results on the right. In the left image, the desired outputs are binary, and the material of interest is mapped to red when the others are mapped to light blue. In the right image, outputs are the results of applying the trained neural network to the whole data set. Dark blue lines are used to highlight the data that are dif cult for the system to classify.

When looking at the dark blue data, more understanding of the classi cation can be obtained. For example, the training data (left image) only contains data with low values for position z, and in the right image, the dark blue data always have high values for position z. That is, the training set includes insuf cient number of examples with high z value for the neural network to learn this case. The user can then provide more training samples that consider high z values. This provides immediate understanding of the types of data that are well-classi ed and not well-classi ed.

An additional application designed speci cally for volume data is to visualize the uncertainty of the classi cation result in the original volume data space. The left image in Figure 9 is a slice of the volume data with colors showing the classi cation result. When a voxel is mapped to pink or blue, it shows that this voxel is well classi ed to one of the classes. Colors in the middle of the colorbar, for example, yellow and green, are used when the data are not well-classi ed by the neural network. This gives the user a better understanding of the data in the spatial domain. However, the green-to-yellow colors not only represent data that are dif cult to classify, but also the boundaries between the brain and other materials due to the interpolation during rendering. Therefore, when rendering the uncertainty information of the entire volume, a thin green-to-yellow layer will cover the whole brain. This can cause misunderstanding of the classi cation results as shown in the middle image of Figure 9.

To remove the uncertainty caused by interpolation, we modify the coloring method as shown in Figure 10. In the left is a curved surface representing the boundary of the classi cation. Two adjacent voxels in the direction of the surface normal are assigned to

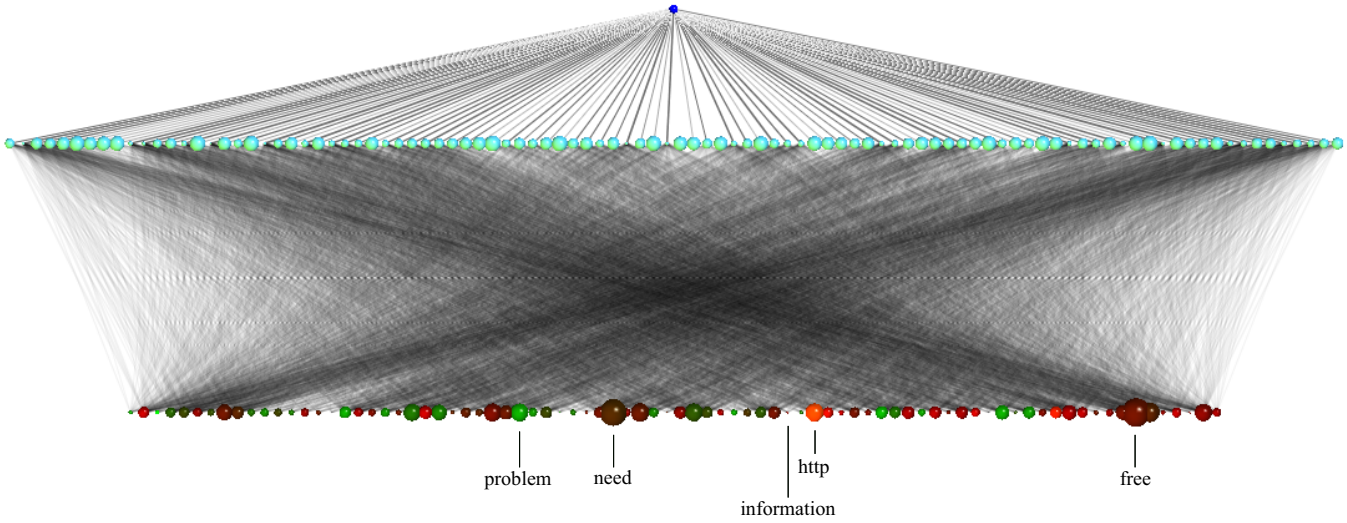

Figure 6: A spam classifier with 82 input nodes and 100 hidden nodes. From bottom to top: Input layer, hidden layer, and output layer. The network is trained to classify email into spam and not spam where each input is a term in the email, and the data feeding through the network is a subset of the spam in the training set.

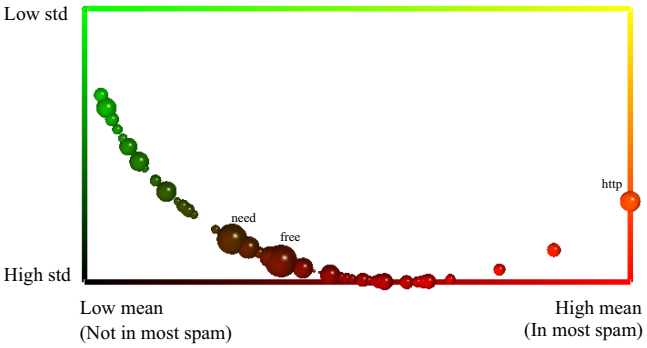

Figure 7: Result of ordering the input nodes onto a panel according to the statistics information where the colors and sizes of each node are the same as those in Figure 6. This provides a simple view of the input nodes.

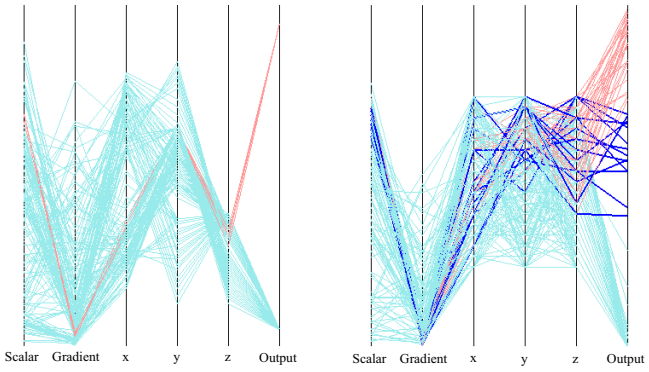

Figure 8: Parallel coordinates showing the classification uncertainty. The red and light blue lines represent two different classes, and the dark blue lines represent the data that is not well-classified. The left image shows the training data where data with high z values is missing, and the right image is the result of classification where the blue data also have high values for position z. This suggests that the training set is not wide enough and more samples with high z values are needed.

pink and blue based on the user de ned color map. During rendering, the opacity of a voxel is obtained by looking up a transfer function with the interpolated data value, and case 1 shows the segment between the two voxels with the interpolated color. After being multiplied by the interpolated opacity, the color on the classi cation boundary becomes yellow, which is not desired. Case 2 is the desired color assignment. There is a binary color assignment between the two voxels. Our method is shown in Case 3. For a voxel p_2, the opacity is assigned based on the interpolated result, and the voxel p_1, which is one voxel away from p_2 in the opposite direction of the surface normal, is used to look up the color assigned to p_2. The result is shown in the right image of Figure 9. The colors are assigned by the neighboring voxel along the opposite direction of the normal so that only the classi cation uncertainty is shown. This image can help the user to identify regions that are not well classi- ed in the 3D volume and guide the user to provide more training samples or add classi cation criteria to the current neural network.

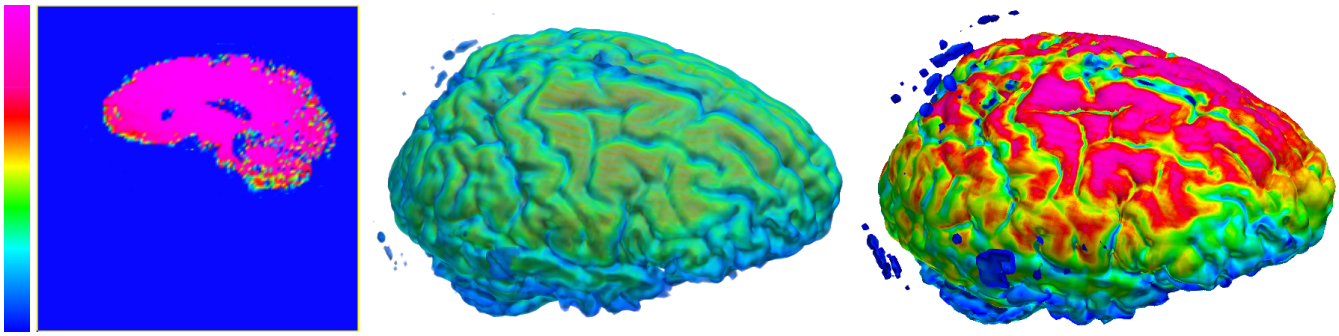

Figure 9: Classification uncertainty shown in the volume data domain. The colorbar is used for color mapping to different uncertainty values. The left image is a slice classified by a trained neural network. The middle image the result of rendering the uncertainty directly as a volume, where a thin layer of green material is introduced by the interpolation during rendering. The right image shows the result of color assignment using our method as presented in Figure 10.

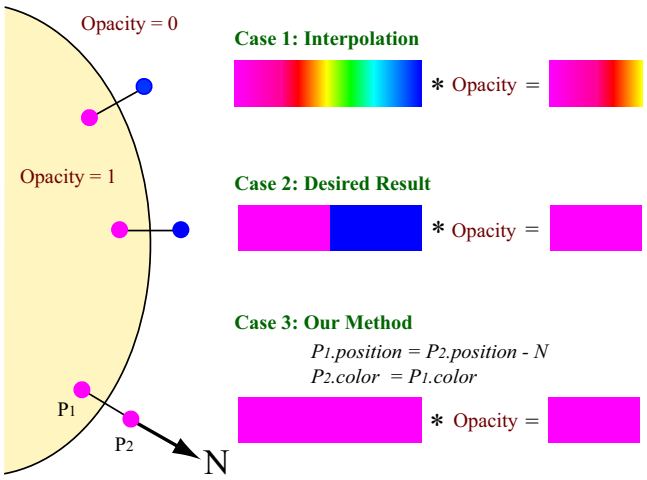

Figure 10: Three different methods for assigning a interpolated color for classification uncertainty. Case 1: Assign color based on the interpolate value by looking up the color map. This will cause a thin layer of wrong color because of the interpolation. Case 2: The desired color assignment. The color changes at the boundary of two materials where no blending region exists. Case 3: Assign a voxel's color based on the value of the neighboring voxel along the opposite normal direction. After applying the interpolated opacity, the same color as in Case 2 can be obtained.

5 Conclusion

Machine learning is gaining widespread use in a variety of application areas. Methods such as arti cial neural networks prove to be powerful in performing certain tasks and would become even more widely employed if they can be better understood by the users. Generally, users need to know how a decision is made and how cost/performance can be better managed. We have shown that properly designed visualizations can give us a sense of the behaviors of the network, how input data are used in the decision, and the level of uncertainty. In particular, we show that it is advantageous to couple visualization of network with visualization of the data. While the visualization cannot explain the learning, it effectively provides pointers to the user for re ning their problem solving strategies using machine learning. Future work includes studying different neural networks and other machine learning methods.

Acknowledgments

This work has been sponsored in part by the U.S. National Science Foundation under contracts ACI 9983641 (PECASE), ACI 0222991, and ANI 0220147 (ITR), ACI 0325934 (ITR), and the U.S. Department of Energy under Lawrence Livermore National Laboratory Agreement No. B537770, No. 548210 and No. 550194. The authors would like to thank members of the UCD visualization and graphics group for the valuable discussion and providing the test data sets.

References

[1] Kjersti Aas and Line Eikvil. Text categorization: A survey. Technical Report 941, Norwegian Computing Center, 1999.

[2] Robert Andrews, Joachim Diederich, and Alan B. Tickle. A survey and critique of techniques for extracting rules from trained artificial neural networks. *Knowledge Based Systems*, 8(6):373–389, 1995.

[3] Bernhard E. Boser, Isabelle M. Guyon, and Vladimir N. Vapnik. A comparison of neural network and fuzzy clustering techniques in segmenting magnetic resonance images of the brain. In *Proceedings of the fifth annual workshop on Computational learning theory*, pages 144–152, 1992.

[4] William W. Cohen and Yoram Singer. Context-sensitive learning methods for text categorization. *ACM Transactions on Information Systems (TOIS)*, 17(2):141–173, 1999.

[5] Mark Craven and Jude Shavlik. Visualizing learning and computation in artificial neural networks. *International Journal on Artificial Intelligence Tools*, 1(3):399–425, 1992.

[6] Ioannis Dimopoulos, J. Chronopoulos, Aikaterini Chronopoulou-Sereli, and Sovan Lek. Neural network models to study relationships

between lead concentration in grasses and permanent urban descriptors in athens city (greece). *Ecological Modelling*, 120(2–3):157–165, 1999.

[7] Harris Drucker, Donghui Wu, and Vladimir N. Vapnik. Support vector machines for spam categorization. *IEEE Transaction on Neural Networks*, 10(5):1048–1054, September 1999.

[8] Wlodzislaw Duch. Visualization of hidden node activity in neural networks: I. visualization methods. In *Proceedings of the International Conference on Artificial Intelligence and Soft Computing*, pages 38–43, 2004.

[9] Wlodzislaw Duch. Visualization of hidden node activity in neural networks: Ii. application to rbf networks. In *Proceedings of the International Conference on Artificial Intelligence and Soft Computing*, pages 44–49, 2004.

[10] Ying-Huey Fua, Matthew O. Ward, and Elke A. Rundensteiner. Hierarchical parallel coordinates for exploration of large datasets. In *IEEE Visualization 1999 Proceedings*, pages 43–50, 1999.

[11] Stephan I. Gallant. Connectionist expert systems. *Communications of the ACM*, 31(2):152–169, 1988.

[12] G. David Garson. Interpreting neural-network connection weights. *AI. Expert*, 6(4):47–51, 1991.

[13] Erol Gelenbe, Yutao Feng, K. Ranga, and R. Krishnan. Neural networks for volumetric MR imaging of the brain. pages 194–202, August 1996.

[14] Muriel Gevrey, Ioannis Dimopoulos, and Sovan Lek. Review and comparison of methods to study the contribution of variables in artificial neural network models. *Ecological Modelling*, 160:249–264, 2003.

[15] A.T.C. Goh. Back-propagation neural networks for modeling complex systems. *Artificial Intelligence in Engineering*, 9(3):143–151, 1995.

[16] Lawrence O. Hall, Amine M. Bensaid, Laurence P. Clarke, Robert P. Velthuizen, Martin S. Silbiger, and James C. Bezdek. A comparison of neural network and fuzzy clustering techniques in segmenting magnetic resonance images of the brain. In *IEEE Transactions on Neural Networks*, volume 3, pages 672–682, September 1992.

[17] Helwig Hauser, Florian Ledermann, and Helmut Doleisch. Angular brushing of extended parallel coordinates. In *In Proceedings of IEEE Symposium on Information Visualization 2002 (InfoVis 2002)*, pages 127–130, 2002.

[18] Alfred Inselberg. The plane with parallel coordinates. *The Visual Computer*, 1(2):69–92, 1985.

[19] Savio L. Y. Lam and Dik Lun Lee. Feature reduction for neural network based text categorization. In *Proceedings of the Sixth International Conference on Database Systems for Advanced Applications (DASFAA)*, pages 195–202, 1999.

[20] Holger R. Maier, Graeme C. Dandy, and Michael D. Burch. Use of artificial neural networks for modelling cyanobacteria anabaena spp. in the river murray, south australia. *Ecological Modelling*, 105:257–272, 1998.

[21] Julian D. Olden and Donald A. Jackson. Illuminating the"black box" a randomization approach for understanding variable contributions in artificial neural networks. *Ecological Modelling*, 154(1–2):135–150, 2002.

[22] Leonid I. Perlovsky. *Neural Networks and Intellect: Using Model-Based Concepts*. Oxford University Press, 2000.

[23] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning representations by backpropagation error. *Nature*, 323:533–536, 1986.

[24] Michele Scardi and Lawrence W. Harding. Developing an empirical model of phytoplankton primary production: a neural network case study. *Ecological Modelling*, 120:220–233, 1999.

[25] Jonas Sjberg, Qinghua Zhang, Lennart Ljung, Albert Benveniste, Bernard Delyon, Pierre-Yves Glorennec, Hkan Hjalmarsson, and Anatoli Juditsky. Nonlinear black-box modeling in system identification: a unified overview. *Automatica (Journal of IFAC)*, 31(12):1691–1724, 1995.

[26] Matthew J. Streeter, Matthew O. Ward, and Sergio A. Alvarez. Nvis: An interactive visualization tool for neural networks. In *Proceedings of SPIE Symposium on Visual Data Exploration and Analysis VII*, pages 234–241, 2001.

[27] A. H. Sung. Ranking importance of input parameters of neural networks. *Expert Systems with Applications*, 15:405–411, 1998.

[28] Alan B. Tickle, Robert Andrews, Mostefa Golea, and Joachim Diederich. The truth will come to light: Directions and challenges in extracting the knowledge embedded within trained artificial neural networks. *IEEE Transactions on Neural Networks*, 9(6):1057–1068, 1998.

[29] Fan-Yin Tzeng, Eric B. Lum, and Kwan-Liu Ma. An intelligent system approach to higher-dimensional classification of volume data. *IEEE Transactions on Visualization and Computer Graphics*, 11(3):273–284, 2005.

[30] Paul Werbos. *Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences*. PhD thesis, Department of Applied Mathematics, Harvard University, 1974.

[31] Erik Wiener, Jan O. Pedersen, and Andreas S. Weigend. A neural network approach to topic spotting. In *Proceedings of SDAIR '95, 4th Annual Symposium on Document Analysis and Information Retrieval*, pages 317–332, 1995.