Visualization of hidden node activity in neural networks: I. Visualization methods.

Włodzisław Duch

School of Computer Engineering, Nanyang Technological University, Singapore, and Department of Informatics, Nicholaus Copernicus University, Grudziądzka 5, Toruń, Poland, http://www.phys.uni.torun.pl/~duch

Abstract. Quality of neural network mappings may be evaluated by visual inspection of hidden and output node activities for the training dataset. This paper discusses how to visualize such multidimensional data, introducing a new projection on a lattice of hypercube nodes. It also discusses what type of information one may expect from visualization of the activity of hidden and output layers. Detailed analysis of the activity of RBF hidden nodes using this type of visualization is presented in the companion paper.

1 Introduction

Feedforward networks provide non-linear mappings $M(\mathbf{X};\mathbf{W})$ that applied to complex feature space regions $\mathcal{X}$ convert them into localized (sometimes point-like) images of these regions, for $\mathbf{X} \in \mathcal{X}, M(\mathbf{X};\mathbf{W}) = \mathbf{Y} \in \mathcal{Y}$. Although this is a common knowledge the mapping properties in classification tasks are almost never used, with focus being on network decisions or estimation of probabilities of decisions. Performance of a typical neural network is evaluated using such measures as the mean square error (MSE), or estimation of the overall classification accuracy. These measures are global averages over the whole feature space, estimated on the whole training or validation set. Detailed properties of multidimensional mappings learned by the hidden layers, and mappings from inputs to outputs that they provide, are regarded as inscrutable, possibly containing singularities and various kinks. Neural mappings may exhibit surprising behavior in novel situations and there is no simple way to find out what potential problems they may hide.

The state of the network, expressed by the values of the weights and biases, may be visualized using Hinton diagrams [1] displayed by some neural network simulators. These diagrams contain interesting information about the network and allow for identification of important connections and hidden nodes. They are not helpful to see what type of internal representations have developed for a given data set, or how well the network performs on some data. A few attempts to visualize the training process are restricted to network trajectories in the weight space ([2], Kordos and Duch, in print). The usefulness of visualization of network outputs has been shown recently [3]. The focus of this paper is on visualization of images created by the hidden layer of feedforward networks with localized functions.

For classification problems with K categories images of the training vectors may be displayed as a scatterogram in K-dimensional space. For $K > 3$ this cannot be displayed directly in one figure, but a linear projection, parallel coordinate

representation or scatterograms for all pairs of outputs may be used. In the next section direct visualization methods are introduced, and a new projection on a lattice of hypercube nodes is described. This approach will be used to visualize patterns of activity of hidden neurons, displaying internal representations created during the learning process. In section three scatterograms of hidden node activities are advocated to reveal internal representations of networks. In the last section discussion and some remarks on the usefulness and further development of such visualization methods are given. Applications of these ideas to the visualization of the hidden layer activity, showing what type of information may be uncovered and how it can be used to improve the quality of neural solutions, are discussed in the companion paper. Because the use of color makes it easier to understand the figures the reader is advised to view the color PDF version of the figures available on-line [4].

2 How to visualize?

Neural classifiers, committees and several other classification systems, map inputs first to at least one internal space $\mathcal{X} \rightarrow \mathcal{H}$, and from there to the output space, $\mathcal{H} \rightarrow \mathcal{Y}$. In case of feedforward neural networks with more than one hidden layer one internal space per layer $\mathcal{H}_k$ is defined. The training data set $\mathcal{T} = \{\mathbf{X}^{(i)}\}, i = 1 \ldots n$ is defined in N-dimensional input space, $\mathbf{X}_i \in \mathcal{X}, i = 1..N$. In this paper only vector mappings $H(\mathbf{X})$ between the input space $\mathcal{X}$, and N_H-dimensional internal (or hidden) space $\mathcal{H}$ are considered (see [3] for the total network mappings).

For two hidden nodes scatterograms showing for all vectors $\mathbf{X} \in \mathcal{T}$ their images $\mathbf{H}(\mathbf{X}) = [H_1(\mathbf{X}), H_2(\mathbf{X})]$ give full details of the internal representation of the training data. For three dimensions scatterograms are still useful, especially with the ability to rotate the point of view that is easy to implement. The $\mathbf{H}(\mathbf{X})$ values lie inside a unit cube; projecting this cube on a plane perpendicular to the $(1,1,1)$ diagonal shows vectors near the $(0,0,0)$ vertex (no activation of outputs) as overlapping with those near the $(1,1,1)$ vertex (all outputs fully active). This information may be added by scaling the size of the markers, depending on the distance of the point to the $(1,1,1)$ vertex [3]. Linear projections on any 3 independent planes preserves full 3D information. In particular 3 pairs of scatterograms (H_1, H_2), (H_1, H_3) and (H_2, H_3) may be used.

For $N_H > 3$ dimensions multiple scatterograms to view various 2D projections may show all information, but $N_H(N_H - 1)/2$ pairs of scatterograms are needed. First two PCA components calculated for the hypercube vertices (2^{n-1} points if $(0,0,...0)$ point is removed) define an interesting projection, but for more than 3 dimensions it hides important relations among hypercube vertices. For example, in 4 dimensions all points on the diagonal $a[1,1,1,1]$ are mapped on the $(0,0)$ point. This diagonal provides an interesting first direction $\mathbf{W}^{(1)}$ because projections of hypercube vertices $\mathbf{P}^{(i)}$ on this direction give the number of "1" bits, $Y_1 = \mathbf{W}^{(1)} \cdot \mathbf{P}^{(i)}$, thus clearly separating the vertices with different number of non-zero bits (incidentally, this provides a general solution to the n-bit parity problem if $\cos(\omega Y_1)$ is taken as the output function, where ω is a single adaptive parameter). Projecting

the vertices on the space orthogonal to the main diagonal $(1-\mathbf{W}^{(1)}\mathbf{W}^{(1)T})\mathbf{P}^{(i)}$ and selecting the first PCA component for this data gives for some N_H an interesting second direction that maximizes variance.

For $N_H = 3$ this is not a good projections because pairs of vertices (1,0,0), (0,1,0), and (1,0,1), (0,1,1), overlap, but rotating the 3D plot is easy. For $N_H = 4$ the second direction is $\mathbf{W}^{(2)}$ =(–0.211, 0.789, -0.577, 0), and all vertices are clearly separated. Connecting vertices that differ on one bit leads to a lattice (Fig. 1), a non-planar graph representing the 4-dimensional hypercube. For $N_H = 5$ the hypercube has already 32 vertices. Using the same approach to generate the second direction results in several vertices that lie very close after the projection. Optimization of the average spacing leads to $\mathbf{W}^{(2)}$ = (–0.1, 0.2, –0.5, –0.33, 0.75), with all vertices clearly separated (Fig. 1). Lattice projection for 6 dimensions could also be used but with 64 vertices it is hardly legible. One solution could be to show only those parts of the lattice where data points are found, but it is clear that lattice projection is useful only for relatively small number of dimensions.

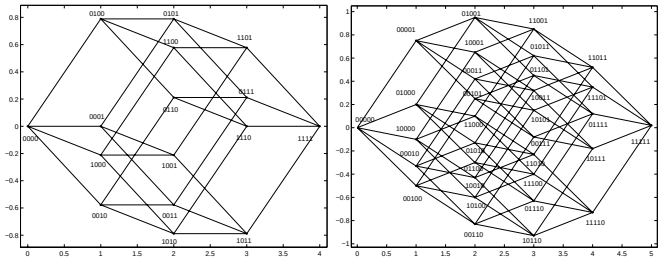

Fig. 1. Lattices corresponding to the vertices of 4 and 5-dimensional hypercubes, obtained by linear projection of their vertices.

Other interesting projection directions may be found, for example mapping the unit vectors of the hypercube axis into polygon vertices. For $N_H = 4$ projection directions $\mathbf{W}^{(1)} = 1/2(-1,1,1,-1)$; $\mathbf{W}^{(2)} = (-0.4,-0.6,1,1)$ give the mapping of hypercube from a different perspective, shown in Fig. 2. For $N_H = 5$ dimensions the first direction goes along the main diagonal $\mathbf{W}^{(1)} = (1,1,1,1,1)$, and $\mathbf{W}^{(2)} = 3/2(1.1,-1,2/3,-2/3,0.1)$, creating a view that shows all vertices at approximately the same distance from each other (Fig. 2).

Because many neural networks use small number of nodes specialized visualization methods are of considerable interest. If $N_H \leq 6$, scatterograms of the training data in hidden or the output space may be displayed using lattice or polygon projections. It is much easier to derive information from such images than from the analysis of numerical output. For more than 6 dimensions parallel coordinates, popular in bioinformatics, are frequently used, but in such visualization some loss of information is inevitable.

Other unsupervised visualization methods that may be used to create images of training data in the hidden and output space include ICA projections, multidi-

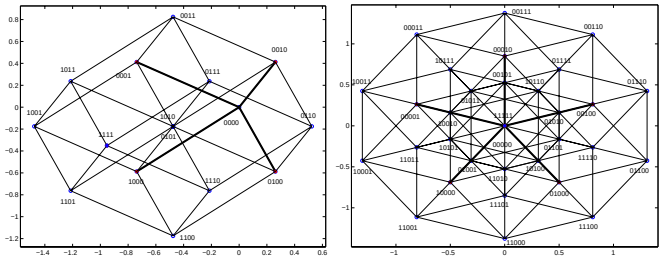

Fig. 2. Linear projection of hypercube vertices in 4 and 5-dimensions; the thick lines are the coordinate axes, ending in vertices that are on a square (left) and pentagon (right).

mensional scaling (see [5, 6] and the references therein), and barycentric mappings using Gaussian functions centered at the hypercube corners (Duch and Orlowski, unpublished; Piekniewski and Rybicki, in print).

3 What to visualize?

Visualization of network outputs has been used recently [3] to understand the dynamics of the learning process and expose its pathologies, show under- and overfitting effects, compare quality of different networks, display border vectors and outliers identifying input space regions where potential problems may arise, show differences in network solutions due to the regularization, early stopping and various optimization procedures, investigate stability of network classification under perturbation of original vectors, place new data samples in relation to the known data to estimate confidence in classification of a given sample. Here the main goal is to understand different internal representations created by the learning process.

In classification problems the training data is divided into labeled subsets $\mathcal{T}_k, k = 1 \ldots K$ corresponding to classes. Neural networks and other classification systems usually try to map each of the training subsets $\mathcal{T}_k$ into one of the K vertices that lie on the hypercube $[0, 0, .., 1, 0, ..0]$ coordinate axes. Hidden layer should map subsets $\mathcal{T}_k, k = 1 \ldots K$ of the training vectors from the feature (input) space creating images, or internal representations $\mathcal{H}_k$ of these subsets, in such a way that would make it easy to separate these images by the hyperplanes, based either on linear nodes or on perceptron nodes provided by the output layer. Without any loss of generality one may assume that all mappings are between hypercubes, with input data rescaled to N-dimensional hypercube, hidden data to N_H dimensional (for a single layer), and output data to the K-dimensional hypercube.

Networks with localized functions may map input $\mathbf{X} \in \mathcal{T}_k$ clusters into subsets $\mathcal{H}_k$ in the hidden space, allowing for reliable classification by inspection of the $H(\mathbf{X})$ images even in cases when linear separability cannot be achieved. Class-dependent distributions may of course overlap and then images $\mathcal{H}_k$ created by hidden layer will not make separable clusters. Visualization of these overlapping areas will identify the border vectors and the input space regions where only low confidence predictions are possible. Mappings provided by networks of different type

may differ in many respects. Neural networks may achieve the same classification in various ways; this is clearly seen visualizing the structure of internal representations of the training data $\mathcal{T}$.

For two-dimensional output space if the network outputs Y_i are independent (i.e. they are not forced to sum to 1) the desired answers may fall into $(1,0)$ and $(0,1)$ corners of a square in (Y_1, Y_2) coordinates. If the values of two outputs of the mapping are forced to sum to 1 to estimate posterior probabilities $p(C_i|\mathbf{X}; M) = \mathbf{Y}_i(\mathbf{X})$, images of all vectors will fall on a single line $Y_1 + Y_2 = 1$ connecting these two corners. For 2 or 3-dimensional feature spaces one may create a regular mesh of input points and look at their hidden and the output images to see the properties of the mapping. In higher number of dimensions this will be prohibitively expensive due to the exponentially growing number of points. High dimensional feature spaces are always almost empty, therefore such mapping should concentrate on the areas in the vicinity of the data. Scatterogram images of all training vectors, sometimes extended by adding noise to the training data, will show the character of mappings in the most important parts of the feature space, where the training vectors are concentrated.

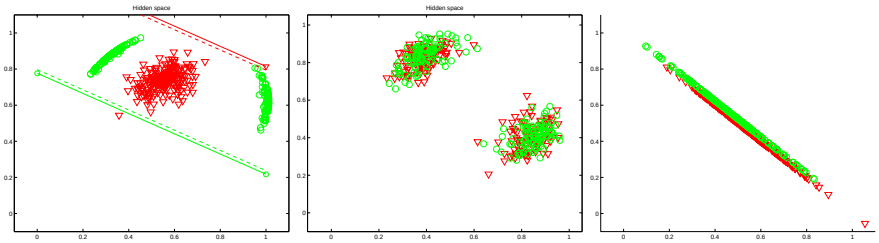

Fig. 3. Activity of two hidden Gaussian neurons for fuzzy XOR problem: good (left) and bad (middle) internal representations, give the same output image (right). Left plot shows lines based on the RBF output weights and biases.

In Fig. 3 a simple example of analysis based on scatterograms is shown for the noisy XOR problem (4 non-overlapping Gaussian clusters, 50 points each). RBF network [7] with two Gaussian nodes is not able to solve XOR problem, but in some runs perfect clusterization is observed in the hidden space. In other runs centers of the Gaussians are placed in a symmetrical way, leading to a complete overlap of the clusters from the two different classes in the internal space. In both cases RBF classification accuracy is close to the default 50% because the clusters are not linearly separable. The training algorithms aimed at reducing output errors is not able to improve internal representations – from the point of view of mean error it is as good as many other representations. The output weights calculated by the RBF program are shown as lines in the left subfigure. They are parallel, indicating that only biases are different and the output (Y_1, Y_2) values fall on the $Y_1 + Y_2 = 1$ line.

4 Discussion

Concentration on numbers makes evaluation of many aspects of neural network mappings rather difficult. In this paper lattice projection techniques have been introduced and the need for visualization of internal representations stressed. There is no reason why scatterogram images of the known data should not be displayed as a part of the neural network output. Such visualization may elucidate many aspects of neural network functions [3]. Visualization of internal representations may suggest different network architectures and training algorithms. Looking at Fig. 3 it is evident that paying more attention to the improvement of internal representations will increase accuracy of a network with the same architecture (2 hidden nodes) to 100%, instead of the default 50% that standard RBF training algorithm achieves [7].

Neural networks are used in various ways for data visualization. Self-Organized-Maps and other competitive learning algorithms, neural Principal and Independent Component Analysis algorithms, autoassociative feedforward networks and Neuroscale algorithms are all aimed at using neural algorithms to reduce dimensionality of the data or to display the data (for a summary of such visualization methods see [6]). All these methods may be used to visualize neural mappings, creating images of the hidden or output neuron activity for presentation of all training data, although in many cases linear projection techniques discussed here will be sufficient. More applications of visualization of the activity of hidden layers are presented in the companion paper.

References

1. Hinton, G.E., McClelland, J.L., Rumelhart, D.E. (1986): In: Parallel Distributed Processing, Vol. 1: Foundations, eds. D.E. Rumelhart, J.L. McClelland, MIT Press, Cambridge, pp. 77-109.
2. Gallagher, M., Downs, T. (2003): Visualization of Learning in Multi-layer Perceptron Networks using PCA. IEEE Transactions on Systems, Man and Cybernetics-Part B: Cybernetics, vol. 33, pp. 28-34.
3. Duch, W. (2003): Coloring black boxes: visualization of neural network decisions. Int. Joint Conference on Neural Networks, Portland, Oregon, 2003, Vol. 1, pp. 1735-1740.
4. PDF version of the color figures and the Matlab programs used in experiments are available at: http://www.phys.uni.torun.pl/kmk/publications.html
5. M. Jambu, *Exploratory and Multivariate Data Analysis*. Boston, MA: Academic Press, 1991.
6. A. Naud, *Neural and statistical methods for the visualization of multidimensional data*. PhD thesis, Dept of Informatics, Nicholaus Copernicus University, 2001; available at http://www.phys.uni.torun.pl/kmk/publications.html
7. C. Bishop, *Neural networks for pattern recognition*. Oxford: Clarendon Press, 1994.