Visualization of hidden node activity in neural networks: II. Application to RBF networks.

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. Scatterograms of the images of training set vectors in the hidden space help to evaluate the quality of neural network mappings and understand internal representations created by the hidden layers. Visualization of these representations leads to interesting conclusions about optimal architectures and training of such networks. Depending on network parameters only some parts of the unit hypercube – called here admissible spaces – may be reached. The usefulness of visualization techniques is illustrated on parity problems solved with RBF networks.

1 Introduction

Understanding multidimensional mappings learned by neural networks is of primary importance. Visualization of network outputs has proved to be very useful in this respect [1]. In the part I of this paper [2] visualization methods based on lattice projections have been introduced and the motivation for this work outlined. This paper is focused on understanding internal representations created by the hidden layer of RBF networks [3]. Two different networks with the same mean square error, making identical number of classification errors, may create quite different internal mappings of the data, and this will have a strong influence on their generalization capabilities. Understanding what neural networks really do requires analysis of internal mappings, and visualization is the easiest way to achieve it.

Several examples using parity problems are presented to gain insight into internal representations that are created by RBF networks. All results were generated using the Netlab package [4], that trains RBF networks using the maximum likelihood approach [3]. Since 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 online [5]. All data are scaled to unit hypercubes. Input, hidden and output hypercubes thus refer to the normalized feature space, space of hidden neuron activities, and the space of output nodes activities.

In the next section the concept of admissible spaces is introduced, covering points in the hypercube that can be reached by network mappings. Section three shows some internal representations for parity problems. The last section contains short discussion.

2 Admissible spaces

Setting the dispersion of the Gaussian nodes, or other parameters determining localization of hidden node functions, is very important and frequently is not done automatically in the optimal way. A single glance at the image of the training data in the hidden space will reveal it. If hidden node functions are very smooth, the mapping is almost linear, all hidden activities are close to 1. The most significant changes of the activations are along the line connecting the cluster centers (assuming that the variance in this direction is the largest). In the limit of very smooth mapping the image of the training set will become a short “bend stick" object very near $(1, 1, ..1)$ corner. On the other extreme, for very localized functions most vectors are mapped to $(0, 0, ...0)$ point, and only those that are close to the function center are mapped along the coordinate lines. For noisy XOR data (a Gaussian distribution with 0.1 variance centered at each of the 4 points is used to generate 99 additional points) this is illustrated in Fig. 1.

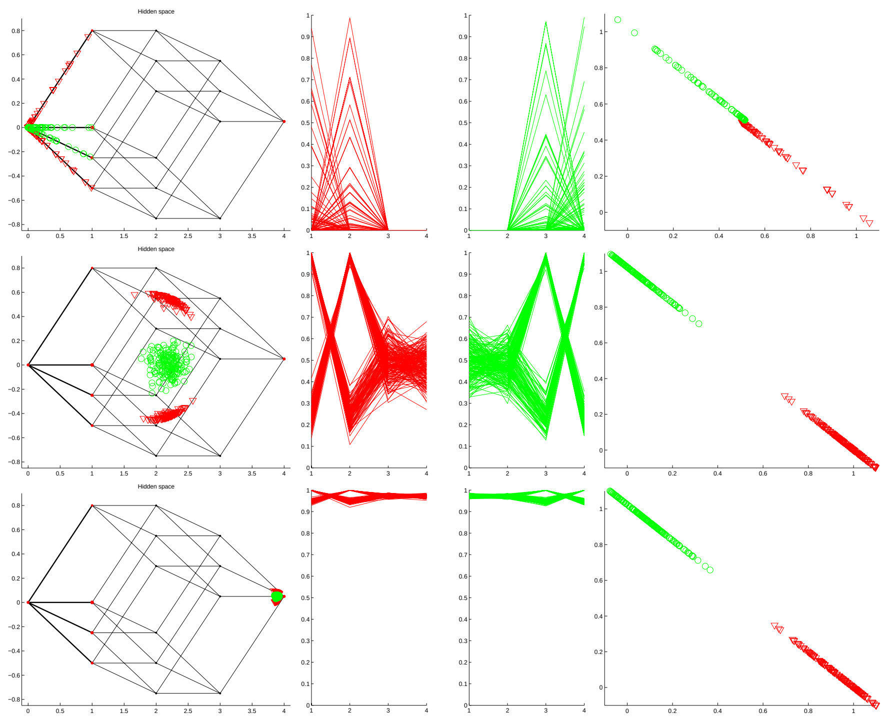

Fig. 1. RBF solution of the noisy XOR problem, using 4 Gaussians; for dispersion $\sigma = 1/7$ all activities are very small (top row), for $\sigma = 1$ optimum separation is achieved, and for $\sigma = 7$ the map is clustered at the (1,1,1,1) corner of the hypercube (bottom). Lattice projections in 4D are on the left, parallel coordinate representation for the two classes in the middle, and output representation in the right column.

Note that the activity of the two output nodes (Fig. 1, right column) is on the line (see [2] for explanation). For very small dispersions only those input vectors that are quite close to the centers of RBF nodes may excite one of these nodes in a stronger way, while other nodes show zero activity. As a result images of most vectors are very near to the hidden hypercube origin, and images of a few are along the coordinate axes. In the other extreme when dispersions are quite large input vectors always excite all 4 nodes strongly and all images are near the (1,1,1,1) corner. Surprisingly though, it is still possible to achieve quite good separation, as seen in the output plot; magnification of the area around the (1,1,1,1) corner reveals 4 well separated separated clusters. Additional experiments with more noisy data confirmed that degradation of accuracy even for very large dispersions is rather small, with moderate node dispersions (middle row), creating images closer to the center of the lattice (middle row in Fig. 1) being optimal. Parallel plots show in this case quite clear "signatures" of each cluster. For example, the plots for the first class may be roughly labeled as (1/3,1,1/2,1/2), and (1,1/3,1/2,1/2), and for the second class (1/2, 1/2, 1, 1/3) and (1/2, 1/2, 1/3, 1).

With localized network nodes taking all $\mathbf{X}$ from the input space hypercube $\mathcal{X}$ the mapping $H(\mathbf{X})$ does not reach all points in the hidden space hypercube. For all $\mathbf{X} \in \mathcal{X}$ vectors mapped by $H(\mathbf{X}) \in \mathcal{A}_H \subset \mathcal{X}$ comprise a subspace that will be called here H-admissible subspace. Functions localized in the input space will give small response to vectors that are far from their centers, mapping many such points on the $(0,..0)$ vertex. If hidden node functions have small overlap the area around $(1,..1..)$ vertex, corresponding to their highest simultaneous activation, is unreachable. If they do overlap the corners near vertices $(1,0,..0)$ to $(0,0,..1)$ will be unreachable, because strong activation of one node will lead to the activation of the second one. The volume of the admissible spaces is always < 1.

In low dimensional cases the RBF-admissible subspace of the hidden space may be visualized by generating the input points on a regular mesh and propagating such inputs through the trained network. Examples of $\mathcal{A}_H$ are shown in Fig. 2 for the Iris data and the XOR data. If functions are well localized the probability of finding an image of a randomly selected vector from high-dimensional input space will be the highest around the $(0..0)$ vertex. The shapes of the envelopes of $\mathcal{A}_H$ subspaces are difficult to determine even for the maps that are sums of Gaussian functions. These shapes depend very strongly on the Gaussian dispersions, as can be seen in Fig. 2. The volume of the subspace shrinks quickly with the decreasing dispersions of the Gaussian functions, breaking at some point into disconnected regions.

Although topological considerations are very interesting they cannot be pursued here. Networks with largest volume of admissible spaces have largest capacity for learning, therefore maximization of this volume could be used to optimize dispersions or other parameters of localized functions. Visualization method based on mesh propagation cannot be used for high-dimensional input data because the number of mesh points grows too quickly, therefore another method, based on adding noise to the input vector, is advocated to probe the mapping in the vicinity of the known data.

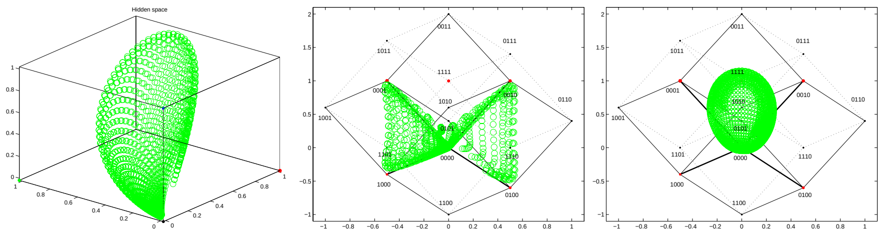

Fig. 2. RBF mapping with 3 hidden nodes trained on the Iris data (left) and with 4 nodes on the XOR data with σ= 0.25 and σ= 1.5.

3 Internal representations in RBF networks

RBF networks with localized response “cover" the input data regions where vectors from a single class dominate. The output layer provides a single reference hyperplane for each class. Non-linear perceptrons in MLPs define separating hyperplanes passing between clusters, but the role of linear outputs used in RBF networks is different. A strong output activation is achieved for vectors at a large distance from the hyperplane, therefore it is usually placed far from the data.

Parity problems have been studied in great details and are well suited for a discussion of internal representations. RBF solution to the XOR problem is rather easy if 4 hidden nodes are used. In the low noise situation (variance 0.1) clusters are well localized and separated, and in the high noise situation (variance 0.25) some overlap between 4 clusters occurs. Repeating the learning process with random initialization and observing the hidden space images shows that the network creates many different internal representations, corresponding to different ways of clusterization of the input data.

It is important to distinguish between representations that are permutationally equivalent and those that are really different. N_H hidden nodes may be assigned to the same number of clusters in $N_H!$ ways. To display only those plots that really differ the vector that leads to a maximum response of each hidden node is identified, and its class is determined. The nodes are then re-grouped according to the classes, and within each class according to their maximum response. Thus on the parallel plots usually the first dimension corresponds to the hidden node that responded in a strongest way to some vector from the first class (it may happen that no node responded strongly to the vectors from the first class). For the 4-node solution to the XOR problem 7 types of internal representations have been noticed, the most frequent solution corresponding to zero errors, and less common solutions with 8-75 errors.

If three nodes are used, activity of the nodes is mixed and simple correlation between the class labels and hidden node activity is lost (Fig. 3). Five different arrangements of clusters appear in different runs; they are easily recognized by the shapes of parallel plots. Two of them are rare, and the remaining three appear equally often. They are equivalent because the XOR problem has high symmetry. The training data images in the hidden space are not separable any more, and the classification error is around 40-50% (that is at default level). This is a clear indication that the complexity of the model is too low to achieve separability.

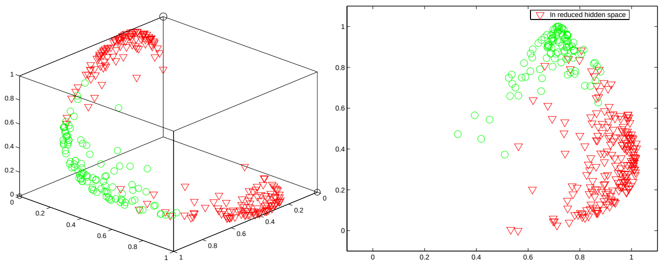

Fig. 3. Left: 3 hidden Gaussian nodes trained on the XOR data; right: reduced representation.

The number of dimensions may be reduced from the number of clusters in the data to the number of classes $K = 2$ by summing the outputs from nodes that belong to each class (Fig. 3, right). This reduced hidden space still shows overlapping areas and well separated regions, although detailed information about the number and structure of clusters is lost. Mapping a new vector of unknown class will show how typical it is, how close to decison border or how far from the training data. This internal mapping is still useful if the number of nodes is too small to achieve output separability, showing the cluster structure and allowing for correct classification by visual inspection both at the level of hidden space images and the reduced hidden space images (Fig. 3). Because the training is not successful also the position of localized functions may not be quite optimal. Algorithms that do not rely on the error only, but try to cover the data by modifying parameters of centralized functions, will still work well [7].

3-bit parity problem is quite difficult to solve for RBF network. Visualization of the hidden space may help to understand why it is difficult to converge to an optimal solution. For training 10 vectors were generated around each corner of a cube (Gaussian noise scaled by 0.1), labeled with the parity of the corresponding binary string, and for test 20 vectors were generated, with scaling factor 0.2. With 8 hidden nodes and optimal dispersion (about $\sigma = 0.5$) perfect solution may be easily found if each of the 8 clusters is mapped on one of the hypercube coordinate axis vertex in 8 dimensions. A good initialization procedure will easily find the corners of the cube and place a center of a Gaussian there, but many RBF programs (including Netalb [4]) rarely find the best initial configuration, and as a result give suboptimal solutions with accuracy in the range of 50-75%. Images in the hidden space are almost always well clustered but not separable, so a single hyperplane is usually not sufficient to find a solution. It may still be possible to obtain good results if more outputs are added and new output codes are defined. Visualization may help to design error-correcting output codes [6]. Reduced two-dimensional projections also show well-clustered images, and a number of methods ,such as the nearest neighbor rule or linear classifiers, will give perfect classification.

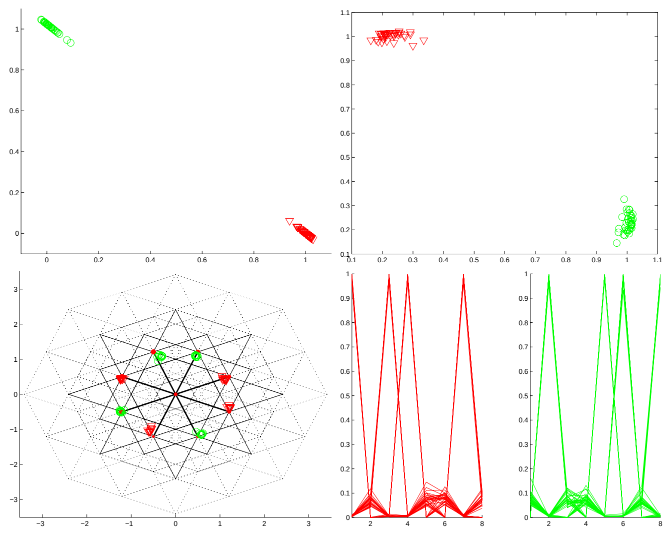

Fig. 4. 3 bit fuzzy parity problem solved with RBF network with 8 Gaussian nodes. Upper left: output, perfect separation, and right, reduced internal space with perfect clustering; lower left: clusters were formed at coordinate axes, with small activity or irrelevant units.

RBF based on maximum likelihood training does not find a good solution for the 3-bit parity problem with less than 8 hidden nodes, but even with 2 or 3 hidden nodes perfect clusterization is observed in the hidden space. Images (Fig. 5) show one cluster per group of input nodes with the same number of “on" bits. They are not separable, and the accuracy in the output space is at the 50% level. These solutions are found in less than 10% of the runs and correspond to 38 or 40 errors (out of 80), while the most common solutions has 39 errors and creates two mixed clusters in the hidden space. Of course the training algorithm is designed to reduce the errors at the output level and not to find compact and well separated clusters in the hidden space. As a result even if the number of hidden nodes is sufficient to find an optimal, large margin solution, many inferior solutions may be generated. Visual inspection allows for comparison of these solutions. Positions of Gaussian functions may be correlated with the signatures of clusters displayed using parallel coordinates – in case of parity data it is easy (clusters are in hypercube’s vertices), and in case of arbitrary data clusterization methods may be used first.

For 3-bit parity almost always two Gaussians are placed very close to each other, leaving one of the clusters poorly covered. Such solutions may give low number of training errors, but do not generalize well. In those rare cases when all Gaussians were placed near data clusters zero errors were obtained for training and the test sets, and the solution was most robust to perturbation of the training data by noise. If too many nodes are used instead of clear clusters of responses some nodes will give quite diffuse, random responses, easy to recognize in parallel plots.

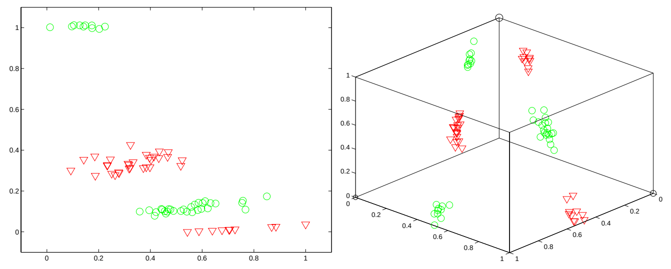

Fig. 5. 3 bit fuzzy parity problem solved with RBF network with 2 and 3 Gaussian nodes. Although errors are at the base rate level images in the internal space are well clustered.

4 Discussion and conclusions

It is obvious that RBF may solve classification problems either by providing sufficient number of localized basis functions – but then generalization of the network will be poor – or by analyzing the image of the training data in the hidden space with something more sophisticated than a single hyperplane per class. Images of the training data in the hidden space, and in the reduced hidden space, carry important information which may be used to evaluate new data vectors placing them in known clusters and informing the user how typical these cases are, how close to cases from other classes, or perhaps give “don’t know" answer. Most of this information is removed by the output layer that leaves just one information: how close to the decision hyperplane in the hidden space is the new case.

The need for modifications of the RBF algorithm that should improve convergence and retain more information is rather obvious, but there was no space to discuss it here. The main purpose of this article was to show visualization techniques that help to understand internal representations of networks. Representations in MLP networks are more complex and will be discussed in subsequent articles. As shown in [1] visualization of outputs is also very useful. There is no reason why neural networks users should restrict themselves to numerical outputs only.

References

1. Duch, W.: Coloring black boxes: visualization of neural network decisions. Int. Joint Conference on Neural Networks, Portland, Oregon, 2003, vol. I, pp. 1735-1740
2. Duch, W.: Visualization of hidden node activity in neural networks: I. Visualization methods. ICAISC 2004, Zakopane, Poland, pp. xx-yy
3. Bishop, C.: *Neural networks for pattern recognition*. Oxford: Clarendon Press, 1994
4. Nabnay, I., Bishop, C.: NETLAB software, Aston University, Birmingham, UK, 1997 http://www.ncrg.aston.ac.uk/netlab/
5. Color figures are available at: http://www.phys.uni.torun.pl/kmk/publications.html
6. Dietterich, T.G., Bakiri, G.: Solving Multiclass Learning Problems via Error-Correcting Output Codes. J. of Artificial Intelligence Research, **2** (1995) 263–286
7. Duch, W., Diercksen, G.H.F. : Feature Space Mapping as a universal adaptive system. Computer Physics Communication **87** (1995) 341–371