A note on the low-dimensional display of multivariate data using neural networks

Gilbert Reibnegger, Gabriele Werner-Felmayer, and Helmut Wachter

Institute for Medical Chemistry and Biochemistry, University of Innsbruck, Innsbruck, Austria

A novel neural network technique has been proposed (Livingstone et al. J. Mol. Graphics 1991, 9, 115–118) which is useful for a low-dimensional display of multivariate data sets. The method makes use of the activity values of the hidden neurons in a trained three-layer feed-forward network to produce the low-dimensional display. It was claimed that in contrast to conventional techniques, such as principal components analysis or nonlinear mapping, this technique could be used also to reconstruct, from a given point in the low-dimensional display, the corresponding multivariate input vector via the completely known weight matrices of a suitably trained network. We show here that this claim is unjustified in this general form. When previously unknown, grossly different input vectors are presented to the trained network, they can occupy, for example, exactly the same point in the low-dimensional display which is occupied also by a given training vector, if certain linear relationships between the vector components are fulfilled. Thus, an infinite set of different linearly dependent input vectors is projected onto one single point in the low-dimensional display. Reconstruction of a multivariate vector, starting from this point in the low-dimensional display, is able to lead back to only one multivariate vector (in the example given, to the original training vector).

Keywords: unsupervised learning, pattern recognition, artificial intelligence, quantitative structure–activity relationships

INTRODUCTION

Basic concepts of connectionist computing schemes, often referred to as *neural networks*, date back to the 1940s.[1] After an initial burst of enthusiasm associating the invention of the "perceptron,"[2] early hopes were disappointed, and only few pioneers continued studying such models.[3] The demonstration that the principles of theoretical physics of many-particle systems are applicable to connectionist data-processing schemes,[4] gave new respectability to the field, and the invention of efficient learning schemes such as the error back-propagation method[5] led to a remarkable renaissance of interest in the field.

Recently, a novel application of neural network architecture for the purpose of low-dimensional display of multidimensional data matrices was proposed.[6] The method made use of a so-called reversible nonlinear dimensionality reduction (ReNDeR) procedure. Here, a multilayered error back-propagation network with a central layer consisting of only two "hidden neurons" is used. The input layer consists of as many neurons as there are variables per case, and so does the output layer. The important feature is that the output values which are used in the training phase of the network, are chosen to be equal to the input vectors. Thus, no external classification is introduced. Obviously, the network is thus forced to learn to perform the following task: a given input vector has to be passed through the "bottleneck" of two hidden neurons, and the resulting output has to be a vector equal to the input vector. The idea underlying this procedure is that, after training, the response values of the two hidden neurons upon an input vector can be used as x- and y-coordinates for a two-dimensional display. Using several different data sets from the problem domain of quantitative structure–activity relationships (QSAR), the method was shown to yield very satisfactory results.

The technique was claimed to possess a particular advantage: unlike other techniques used to reduce the dimensionality of multidimensional input data, such as, e.g., principal components analysis or nonlinear display, the neural network technique was thought to possess the capability to reproduce, from two-dimensional data points, the original multivariate data vectors again because all the connection weights between neurons are known.

We show here that this claim is not justified in this general form. In fact, if a trained network is used to interpret new input vectors, a new input vector might be projected, for example, onto a certain point of the two-dimensional plane already occupied by an input vector from the original training set. As a consequence of the reduction of information when passing from the multidimensional to two-dimensional space, however, the new input vector might be grossly different from the original training vector, provided only that certain linear relationships hold between the vector

Address reprint requests to Dr. Gilbert Reibnegger at the Institute for Medical Chemistry and Biochemistry, University of Innsbruck, Fritz Pregl Strasse 3, A-6020 Innsbruck, Austria.
Received 6 July 1992; accepted 21 July 1992

components. Reconstruction of a multivariate vector from this two-dimensional point, then, will invariably yield the original training vector; the actually used input vector would not be reconstructed.

METHODS

In order to make our point clear, we are using three different, very simple examples which allow one to understand the principle. For all examples, standard error back-propagation[5] was used for the learning process. The first and second example both employ the same *2-1-2* architecture, shown in Figure 1: an input layer consisting of two input neurons, a hidden layer consisting of only one neuron (thus making the computations even simpler), and an output layer consisting, again, of two neurons. The "bias" neurons which always "fire" with an output activity of 1.0, are just an efficient way to introduce the necessary threshold values for the sigmoid transformations involved.[5] In the first example (Figure 1, left panel), the network was trained on only a single training vector, namely, the vector [0.5,0.5]. In the second example (Figure 1, right panel), two training vectors were used: [0.1,0.9] and [0.9,0.1]. For the third example a neural network having a *5-2-5* architecture was used. As input vectors, we chose atomic charges of 5 atoms of the pyrimidine moiety of a small series of pterin and guanine derivatives, computed by an *ab initio* Hartree Fock technique and published previously.[7] The charges computed for the atoms N_1, C_{8A}, C_{4A}, C_4, and O were used for the computations. The two basic compounds, pterin and guanine, are shown in Figure 2 with their conventional atomic numbering systems. For convenience, the data actually used are shown in Table 1. The data were transformed prior to their presentation to the network: for each atom, the charges computed were linearly projected onto the interval [0.1,0.9].

For training of all networks, the NEUROCOMPILER software (Neuro Informatik GmBH, Berlin, Germany) was used. Constant learning parameters were used throughout. Learning was terminated when the average error function dropped below 0.0001 for examples 1 and 2, and below 0.001 for example 3.

To compare the results obtained by this neural network procedure with those obtained by other techniques, the data of example 3 were also analyzed using conventional cluster analysis of cases (program BMDP2M of the BMDP Statistical Software, Cork, Ireland), and Kohonen's method of self-organizing feature maps.[8] For the latter, a network was employed consisting of five input neurons projecting onto a 12 × 12 layer of neurons (Kohonen layer). The data were transformed in the same way as described above before presentation to the network. To visualize the self-organization of the Kohonen layer, after a training phase of 90 steps, each input vector is presented to the network. The input

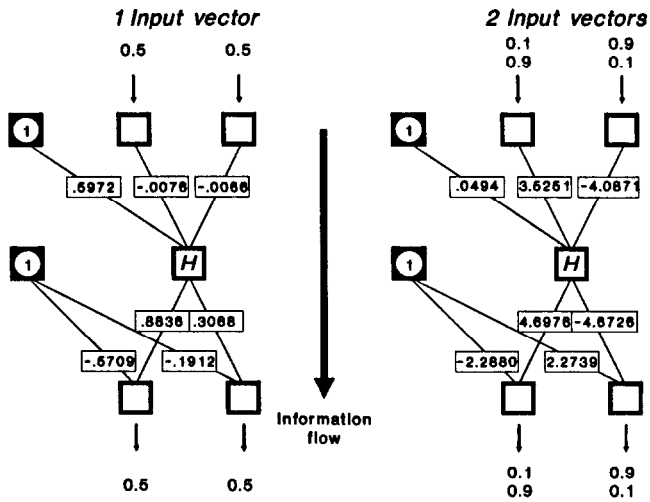

Figure 1. Neural networks with 2-1-2 *architecture after training. The extra neurons designated* 1 *are the so-called bias neurons, providing the threshold values for computing the sigmoid activation values. The central neurons, designated* H, *are the hidden neurons. The weight factors are given in detail. Left panel: Network trained on one single input vector. Right panel: Network trained on two input vectors.*

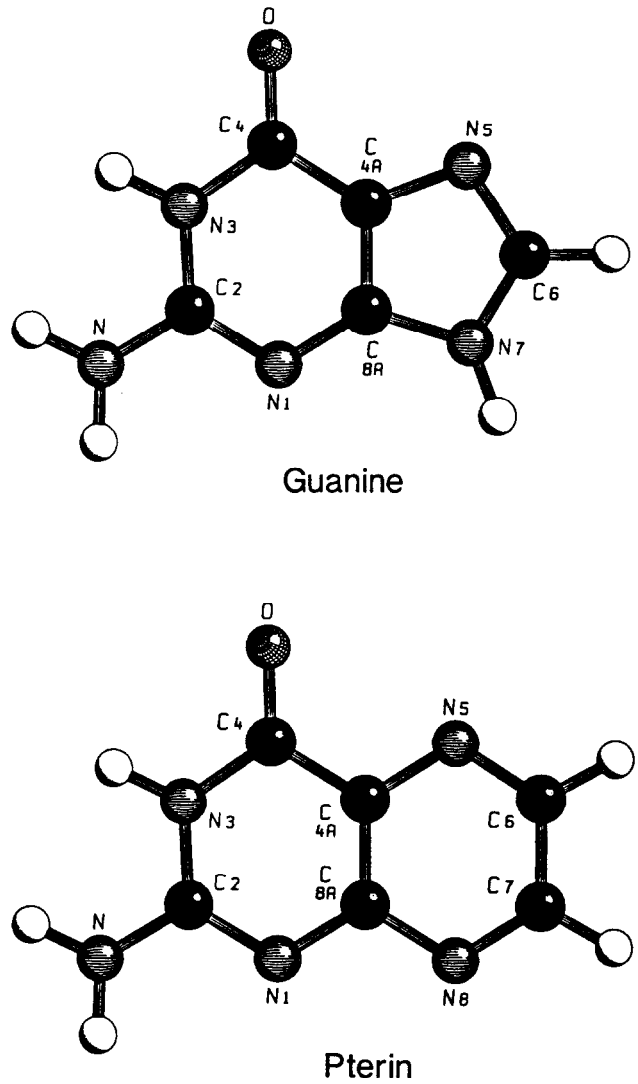

Figure 2. Chemical structure and conventional numbering scheme of guanine and pterin.

Table 1. Partial charges of five atoms common to nine guanine and pterin derivatives[7]

Compound	N_1	C_{8A}	C_{4A}	C_4	O
Guanine	−0.689	0.594	−0.544	0.753	−0.195
5-Deazaguanine	−0.695	0.713	−0.908	0.833	−0.217
5-Methyl-5-deazaguanine	−0.696	0.711	−0.822	0.836	−0.226
Pterin	−0.666	0.534	−0.418	0.721	−0.164
6-Methylpterin	−0.668	0.534	−0.413	0.720	−0.168
6-Hydroxymethylpterin	−0.669	0.535	−0.410	0.720	−0.164
Pterin-6-aldehyde	−0.664	0.547	−0.412	0.721	−0.158
6-Aminopterin	−0.668	0.483	−0.364	0.715	−0.174
7,8-Dihydropterin	−0.725	0.743	−0.524	0.728	−0.195

vector which evokes maximal activity is recorded for each neuron, and these data are used for graphical presentation, together with the maximal activity values obtained for each neuron.

RESULTS

Example 1: *2-1-2* network trained on only one input vector

Figure 1, left panel, shows the complete network after training with the input vector [0.5,0.5]. Obviously, the hidden neuron *H* computes its net input according to

$$0.5972386 - 0.0075636x_1 - 0.0065870x_2 = 0.5901633$$

if x_1 and x_2 are both set equal to 0.5 (note that four decimal digits are given in the figure for simplicity). However, each combination of x_1 and x_2 which obeys the equation

$$x_1 = 0.9354408 - 0.8708816x_2$$

yields the same net input for the hidden neuron. Hence, all points lying on the straight line shown in Figure 3, left panel, yield the same net input to the hidden neuron. And naturally, the output vector resulting from all these different input vectors is the vector [0.5,0.5]. (It might be noted that due to the small numerical values of the weight factors between input and hidden layer in comparison to the bias weight, this network is quite insensitive overall against different input vectors.)

Example 2: *2-1-2* network trained on two different input vectors

Figure 1, right panel, presents an analogous network after training with two input vectors, [0.1,0.9] and [0.9,0.1]. In the first case, $x_1 = 0.1$ and $x_2 = 0.9$, the hidden neuron *H* computes its net input according to

$$0.0493767 + 3.5251458x_1 - 4.0871601x_2 = -3.2765528$$

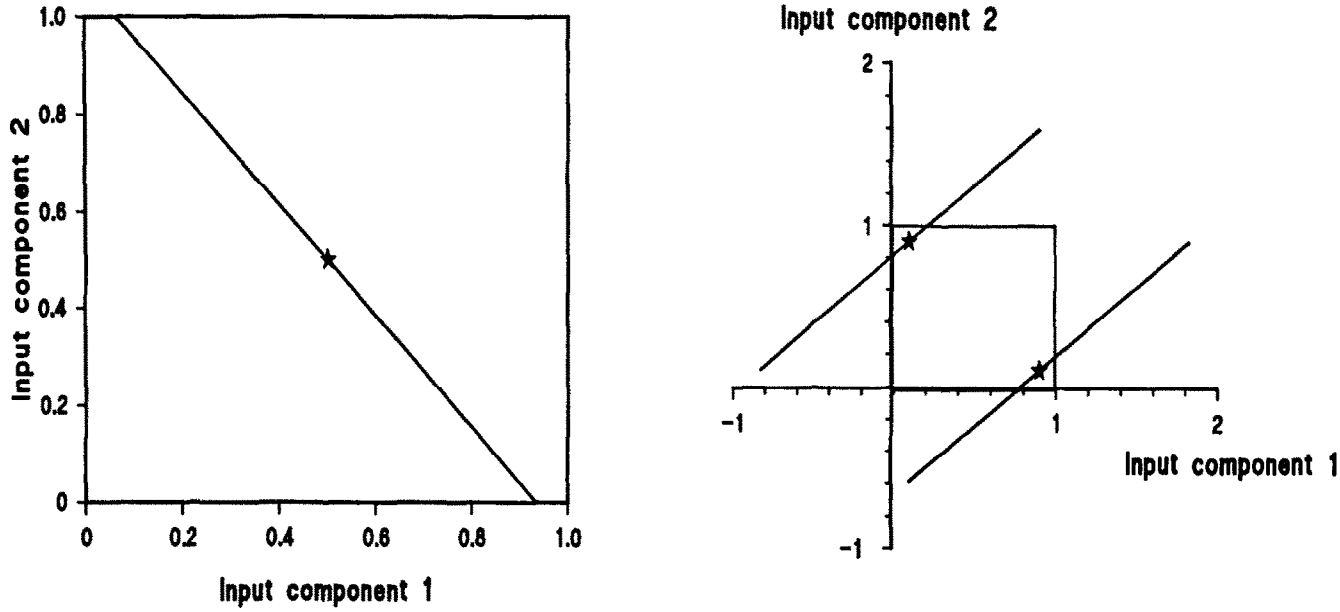

Figure 3. Sets of different input vectors (given by the straight lines) which all lead to the same activation value of the hidden neuron in the 2-1-2 *networks shown in Figure 1. The equations for the lines are derived in the main text of the paper. The star symbolizes the original input vectors used for training of the respective network. Left panel: Network trained on one single input vector. Right panel: Network trained on two input vectors.*

This leads to the equation

$$x_1 = 1.1594301x_2 - 0.9434871$$

giving all possible combinations of x_2 and x_2 which lead to the same net input of the hidden neuron as the first training vector. Analogously, if $x_1 = 0.9$ and $x_2 = 0.1$, one obtains

$$0.0493767 + 3.5251458x_1 - 4.0871601x_2 = 2.8132919$$

and, hence, the linear equation

$$x_1 = 1.1594301x_2 + 0.7840570$$

gives all input vectors leading to the same net input of the hidden neuron as the second training vector. These two linear relationships between x_1 and x_2 are represented by the two parallel lines in Figure 3, right panel.

Example 3: *5-2-5* network trained on molecular data

A *5-2-5* network as shown in Figure 4 was trained on the atomic charges of the nine molecules given in Table 1, after appropriate transformation. Again, the output vectors were chosen to be equal to the input vectors. When the response of the two hidden neurons in the trained network is recorded after presentation of each of the nine input vectors, the two-dimensional display shown in Figure 5 is obtained. Obviously, four clusters are obtained: *cluster 1*, guanine; *cluster 2*, 5-deazaguanine and 5-methyl-5-deazaguanine; *cluster 3*, all aromatic pterins; *cluster 4*, 7,8-dihydropterin. Thus, from a chemical viewpoint, the network has indeed found a very reasonable cluster structure.

Using the same data set, Figure 6 shows the result of a classical cluster analysis of cases. Figure 7 shows the structure obtained by Kohonen's technique; for each of the 12 × 12 neurons of the Kohonen layer, the input vector inducing maximum response was recorded, and a symbol for the corresponding molecule was written on the physical position of the respective neuron, if the maximum response exceeded the value 0.99. Obviously, all methods have detected the same cluster structure in the data set.

Due to the more extended network, the weight values are

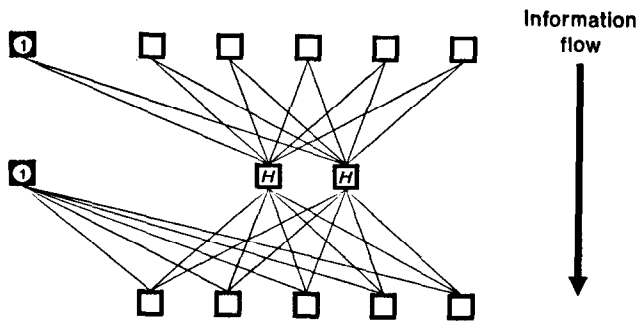

Figure 4. Neural network with 5-2-5 architecture used for the atomic charge data given in Table 1.

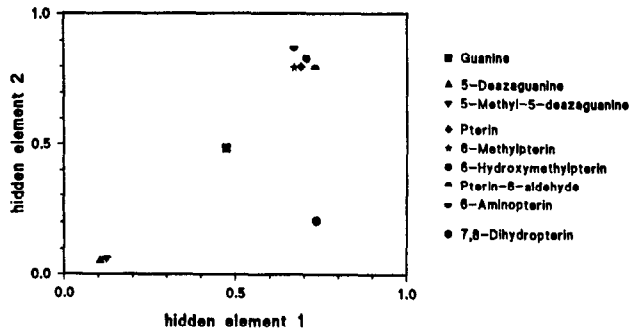

Figure 5. Activation attained by the two hidden neurons in an error back-propagation network with 5-2-5 architecture (see Figure 4), trained on the atomic charges given in Table 1.

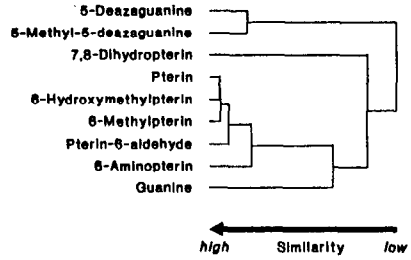

Figure 6. Result of a cluster analysis of cases, using the atomic charges given in Table 1.

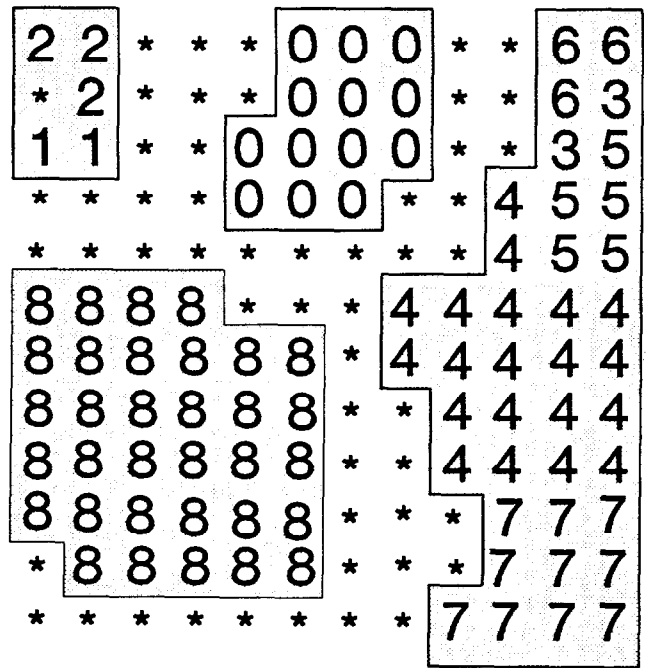

Figure 7. Result of training of a 12 × 12 Kohonen network on the atomic charges given in Table 1. For each neuron, the input vector leading to maximum stimulation was recorded. If the activation value attained exceeded 0.99, a numerical symbol was drawn on the physical position of the neuron in the network: 0, guanine; 1, 5-deazaguanine; 2, 5-methyl-5-deazaguanine; 3, pterin; 4, 6-methylpterin; 5, 6-hydroxymethylpterin; 6, pterin-6-aldehyde; 7, 6-aminopterin; and 8, 7,8-dihydropterin.

not reported in detail. Clearly, the same problem exists for this somewhat more complicated case, as was discussed above: by means of simple linear relationships one can easily generate artificial input vectors which would correspond to dramatically changed atomic charges and, hence, grossly different chemical properties of the underlying hypothetical molecules (not shown in detail).

DISCUSSION

By no means does our work criticize the proposal of using neural networks as a means for obtaining low-dimensional displays for multivariate data sets.[6] In contrast, the results shown in Figure 5 clearly underscore the usefulness of this procedure.

Our work shows, however, that the claimed ability of such systems to reproduce the multidimensional vector from a point in the low-dimensional display is limited. In fact, the reduction in dimensionality leads to a set of insufficiently determined linear equations. The result is, as we have shown here, that an infinite set of different input vectors can lead to the same pattern of hidden neuron activation, and thence to the same output vector. The condition is that the components of the input vectors obey certain simple linear relationships.

The consequences are straightforward: if a low-dimensional representation of an input vector is considered which had been a member of the training vectors, then the multidimensional vector can be fully reconstructed via the known weight factors of the network. Suppose, however, that an input vector is presented to the trained network in the application phase which was previously unknown (i.e., not used in training). Suppose further that the components of this input vector happen to obey one of the many (in case of multidimensional input vectors) possible linear relationships with one of the training vectors: clearly, the new input vector then will produce the same pattern of hidden neuron activation as the training vector, and on reconstruction invariably the training vector is obtained, but not the original input vector actually used.

The example of the atomic charges used in our exposition of the problem shows that particularly in the field of QSAR one has to use the described neural network technique with caution. The method is certainly well-suited for exploratory analyses of large data sets. Application of the trained network to new and unknown cases, however, should be made only in connection with other multivariate display techniques in order to eliminate the dangers which are caused by the potentially massive reduction in dimensionality of the problem. Otherwise grossly different input vectors might be projected onto the same region of the low-dimensional display. Using, e.g., Kohonen networks in addition to the low-dimensional display would easily resolve the problem: the linearly dependent, however grossly different, new input vector would be extremely unlikely to stimulate just the same Kohonen neuron as the original training vector.

REFERENCES

1 McCulloch, W.S. and Pitts, W.H. A logical calculus of the ideas imminent in neural nets. *Bull. Math. Biophys.* 1943, **5,** 115–133

2 Rosenblatt, F. *Principles of Neurodynamics.* Spartan, New York, 1943

3 Cooper, L.N. A possible organization of animal memory and learning. In: *Proceedings of the Nobel Symposium on Collective Properties of Physical Systems* (B. Lundquist and S. Lundquist, Eds.) Academic Press, New York, 1973, pp. 252–264

4 Hopfield, J.J. Neural networks and physical systems with emergent collective computational abilities. *Proc. Natl. Acad. Sci. USA* 1982, **79,** 2554–2558

5 Rumelhard, D.E., Hinton, G.E., and Williams, R.J. Learning internal representations by error propagation. In: *Parallel Distributed Processing: Exploration in the Microstructure of Cognition* (D.E. Rumelhard and J.L. McClelland, Eds.) MIT Press, Cambridge, 1986, pp. 318–364

6 Livingstone, D.J., Hesketh, G., and Clayworth, D. Novel method for the display of multivariate data using neural networks. *J. Mol. Graphics* 1991, **9,** 115–118

7 Reibnegger, G., Fuchs, D., Hausen, A., and Wachter, H. Quantum chemical aspects of pterins in comparison with guanine and deazaguanine derivatives. In: *Biochemical and Clinical Aspects of Pteridines.* (H.C. Curtius, W. Pfleiderer, and H. Wachter, Eds.) Walter de Gruyter, New York, 1983, pp. 35–52

8 Kohonen, T. *Self-Organization and Associative Memory.* Berlin, Springer, 1984