ESANN'1997 proceedings - European Symposium on Artificial Neural Networks
Bruges (Belgium), 16-17-18 April 1997, D-Facto public., ISBN 2-9600049-7-3, pp. 97-102

Two Neural Network Methods for Multidimensional Scaling

Michiel C. van Wezel, Joost N. Kok and Walter A. Kosters

Leiden University, Dept. of Mathematics and Computer Science
P.O. Box 9512, 2300 RA Leiden, The Netherlands
Email: {michiel, joost, kosters}@wi.leidenuniv.nl

Abstract. Multidimensional scaling (MDS) embeds points in a Euclidean space given only dissimilarity data. Only very recently MDS has gotten some attention from neural network researchers. We propose two neural network methods for MDS and evaluate them using both artificially generated and real data. Training uses two inputs at a time.

1. Introduction

Multidimensional scaling (MDS) is a well-known statistical technique that has been applied successfully to a variety of problems in the past. Generally stated, the classical MDS techniques attempt to find a coordinate representation for various objects between which only dissimilarities are given. Dissimilarities between objects are monotonically related to Euclidean distances between the objects.

Various types of MDS procedures and objective functions have been presented in the past. Recently MDS also got some attention from neural network researchers [2, 3]. However, in [3] the focus is on MDS as a dimensionality reduction technique, and the proposed method still assumes coordinate representations for all the objects. In [2] a mean field approach to the MDS problem is presented, which does not work as intuitively as the methods we will present. Moreover, excellent results are already obtained with our simpler methods.

In this paper, two new neural network methods for MDS are proposed that only assume dissimilarities between objects, analogous to the original MDS formulation (see e.g. [6] or a textbook on MDS, e.g. [1, 5]). The first neural network is a feedforward neural network trained with a gradient based learning rule. The second neural network is an unsupervised competitive neural network. Both networks are tested on both artificial and real data.

The remainder of this paper is structured as follows: in Section 2 the two neural network methods for MDS are presented: first the feedforward neural network and next the unsupervised competitive neural network. In Section 3, experiments on artificially generated data are described. In Section 4, experiments on data from a real world problem are discussed. Finally, directions for further research and conclusions are given in Section 5.

ESANN'1997 proceedings - European Symposium on Artificial Neural Networks
Bruges (Belgium), 16-17-18 April 1997, D-Facto public., ISBN 2-9600049-7-3, pp. 97-102

2. Description of the Neural Networks

As stated above, two neural network methods are described here. The first one is a multilayer perceptron trained with a backpropagation-like learning rule. The second one is an unsupervised competitive neural network.

2.1. Description of the Feedforward Neural Network

The feedforward neural network performs a gradient descent on the cost function STRESS $= E = \sum_{\xi,\zeta} E_{\xi\zeta}$, where $E_{\xi\zeta} = (\sigma_{\xi\zeta} - d_{\xi\zeta})^2$, and the summation runs over all pairs of objects ξ, ζ. Here, $\sigma_{\xi\zeta}$ represents the known dissimilarity between object ξ and object ζ, and $d_{\xi\zeta}$ represents the distance between object ξ and object ζ in the k-dimensional coordinate space. A normalised version of STRESS is NSTRESS $=$ STRESS$/n^2$, with n equal to the number of objects to be scaled. In traditional MDS procedures, the dissimilarities $\sigma_{\xi\zeta}$ are required to be either proportional or monotonically related to distances. In the first case, one speaks of "metric" MDS, in the latter case one speaks of "non-metric" MDS.

To implement this gradient descent in a neural network, we propose an architecture as in Figure 1. Here, every object we wish to find a coordinate representation for, is represented by one input neuron. If a certain neuron i in the input layer is activated, we wish to see the coordinate representation of the i-th object at the output layer. The network is a feed-forward network with linear output units.

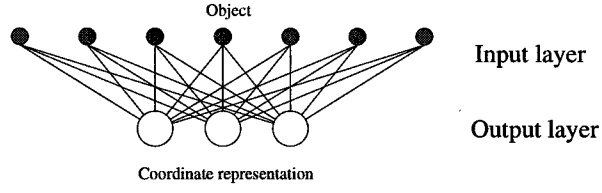

Figure 1: Architecture of a feedforward neural network for MDS.

The network is trained with a gradient descent learning rule in the following way. Two different objects ξ and ζ are selected at random. First, the i-th object ξ is offered to the network, meaning that input node i is activated, and its corresponding output is calculated. Then, the j-th object ζ is offered to the network, and its corresponding output is calculated. Now, the distance between the coordinate representations of these two objects can be computed, whereas their dissimilarity was already known. Next the weights are updated according to the learning rule given below, in which the values of the output units are combined through the distance vector $d_{\xi\zeta}$.

Before we give the learning rule, we introduce some notation. By w_{ij} we denote the weight of the connection from input node i to output node j ($1 \leq$

ESANN'1997 proceedings - European Symposium on Artificial Neural Networks
Bruges (Belgium), 16-17-18 April 1997, D-Facto public., ISBN 2-9600049-7-3, pp. 97-102

$i \leq n$, the number of input nodes, and $1 \leq j \leq k$, the number of output nodes). Let $Y_{i|\xi}$ denote the value of input node i when object ξ is presented to the network ($1 \leq i \leq n$). (Note that this is somewhat more general than the input behaviour prescribed in the previous paragraph.) Furthermore, let $y_{j|\xi}$ denote the value of output unit j when object ξ is presented to the network ($1 \leq j \leq k$). For $1 \leq j \leq k$ we have:

$$y_{j|\xi} = \sum_{i=1}^{n} w_{ij} Y_{i|\xi}.$$

Finally, let η be the learning rate. Now the weights can be updated according to the following learning rule:

$$\Delta w_{ij} = -\eta \left(-2 \frac{(\sigma_{\xi\zeta} - d_{\xi\zeta})}{d_{\xi\zeta}} (y_{j|\xi} - y_{j|\zeta})(Y_{i|\xi} - Y_{i|\zeta}) \right).$$

The derivation of this rule is as follows. First note that

$$d_{\xi\zeta} = \sqrt{\sum_{j=1}^{k} (y_{j|\xi} - y_{j|\zeta})^2}$$

is the Euclidean distance, and:

$$\begin{aligned} \frac{\partial E_{\xi\zeta}}{\partial w_{ij}} &= \frac{\partial E_{\xi\zeta}}{\partial d_{\xi\zeta}} \frac{\partial d_{\xi\zeta}}{\partial w_{ij}} = -2(\sigma_{\xi\zeta} - d_{\xi\zeta}) \frac{\partial d_{\xi\zeta}}{\partial w_{ij}} \\ &= -2(\sigma_{\xi\zeta} - d_{\xi\zeta})(y_{j|\xi} - y_{j|\zeta})(Y_{i|\xi} - Y_{i|\zeta})/d_{\xi\zeta}. \end{aligned}$$

If we let $Y_{i|\xi}$ be 1 if ξ is the i-th object, and 0 otherwise, we get the desired network.

After updating the weights a number of times, the value of STRESS is computed. Training is stopped if a maximum number of iterations is exceeded or STRESS drops below a certain threshold. This training procedure, where two patterns are selected each time the weights are updated, is similar to the training procedure proposed for the SAMANN network in [4].

2.2. Description of the Unsupervised Neural Network

The unsupervised neural network works simpler and more intuitive than the feedforward neural network. In the unsupervised neural network, every object is represented by a neuron. Every neuron has a weight vector with the same dimensionality as the coordinate representation the object should get. Weight vector j represents the position of object j in the coordinate space. An example of an unsupervised competitive MDS neural network is depicted in Figure 2.

ESANN'1997 proceedings - European Symposium on Artificial Neural Networks
Bruges (Belgium), 16-17-18 April 1997, D-Facto public., ISBN 2-9600049-7-3, pp. 97-102

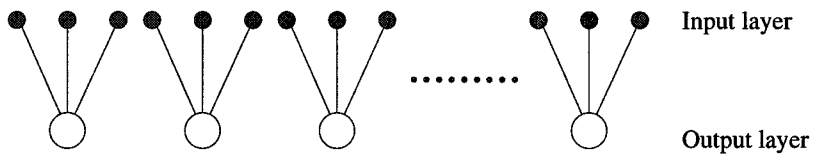

Figure 2: Architecture of an unsupervised competitive neural network for MDS.

The training of the neural network now proceeds as follows. Two different patterns ξ and ζ are selected at random. Next, the distance between the weight vectors of the neurons corresponding to ξ and ζ is calculated (this is $d_{\xi\zeta}$). This distance is compared to the known dissimilarity $\sigma_{\xi\zeta}$, and the weight vectors are pushed away from each other if $d_{\xi\zeta} < \sigma_{\xi\zeta}$, and are pulled towards each other if $d_{\xi\zeta} > \sigma_{\xi\zeta}$. Note that this training scheme is very much like the training scheme of an unsupervised competitive network, where the weight vectors of the units are pulled towards the inputs vectors. Therefore the name "unsupervised neural network" seems appropriate.

3. Experiments on Artificial Data

We generated artificial data to test our methods. These data were uniformly distributed in the $[0:1]$-hypercube. The idea was that each point represented the coordinates of an "object".

In each iteration of the neural networks, two artificially generated data points (say ξ and ζ) were picked randomly. The distance $d_{\xi\zeta}$ was calculated, and Gaussian distributed noise was added to this distance yielding the dissimilarity. In real life, this noise factor could for example be caused by a different perception of two stimuli by different respondents. One person may e.g. find two brands of cola to taste very similar (say 2 on a scale from 1 to 10 expressing similarity), whereas another person may judge them to taste somewhat less similar (say 3 on the same scale from 1 to 10). Given the two indices ξ and ζ, and their dissimilarity, a training iteration could be performed as described above.

The aim of the training process in the case of this artificial data, is to find the original coordinates as the coordinate representations of each object. Of course, the solution quality is invariant under rotations and reflections, as is the value of (N)STRESS.

We tested the feedforward neural network on artificial data for different dimensionalities and dataset sizes. The value of NSTRESS yielded by the network was 0.00032 ± 0.00006 while the number of objects was 50, 100, 200, 300, 400, 500, and the number of dimensions 2, 4, 6, 8, 10. For 10 and 25 objects the variance was only slightly larger.

ESANN'1997 proceedings - European Symposium on Artificial Neural Networks
Bruges (Belgium), 16-17-18 April 1997, D-Facto public., ISBN 2-9600049-7-3, pp. 97-102

For the unsupervised competitive neural network results were even better. The final value of NSTRESS was 0.00016 ± 0.00005, for all experiments.

Figure 3 shows an artificial 2-dimensional problem (100 samples of the function $0.5 * \sin(10x) + x^2 - 0.5 * x - 0.5$ in the interval $[-1 : 1]$), and the solutions yielded by the two networks. The form of the original problem is clearly recovered by both methods.

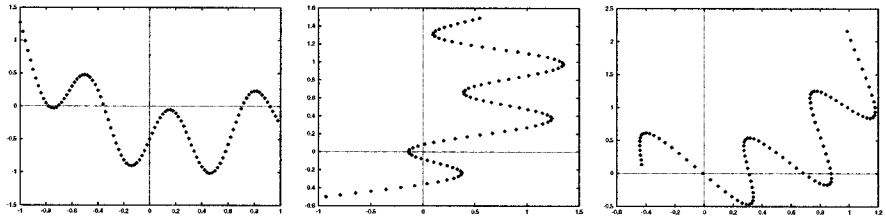

Figure 3: Artificial problem and solutions found. Left: original data, middle: solution feedforward neural network, right: solution unsupervised competitive neural network.

4. Experiments on Real Data

After testing the methods on artificially generated data, we went on to test the methods on two real datasets.

The first dataset is Fisher's famous Iris dataset. This dataset contains 150 samples describing four features of iris plants. In total, three classes of iris plants are present in the database. The left side of Figure 4 shows the 2-dimensional representation of this dataset provided by the feedforward neural network. The value of NSTRESS for this solution was 0.006899. The three classes are clearly visible in this 2-dimensional representation.

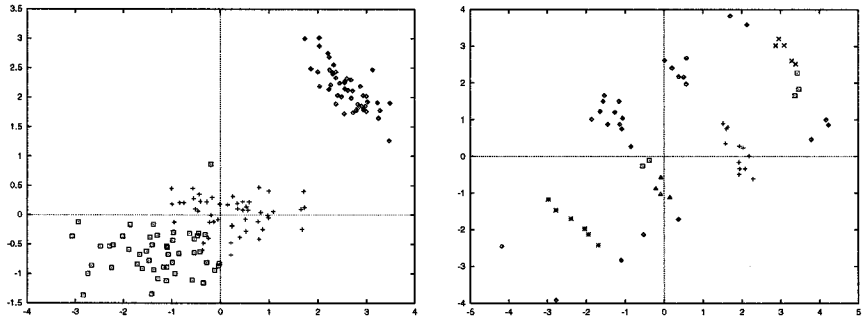

Figure 4: 2-dimensional representations of the 4-dimensional iris dataset (left), and the 16-dimensional zoo dataset (right).

ESANN'1997 proceedings - European Symposium on Artificial Neural Networks
Bruges (Belgium), 16-17-18 April 1997, D-Facto public., ISBN 2-9600049-7-3, pp. 97-102

The second dataset we used was the so-called "zoo" dataset, electronically available from: `ftp://ftp.ics.uci.edu/pub/machine-learning-databases/zoo/zoo.data`. It is a 16-dimensional set containing 101 instances describing animal features. The animals are divided into seven classes. Figure 4 shows the 2-dimensional representation of this dataset generated by the feedforward neural network. The value of NSTRESS for this solution was 0.172777. The heterogeneity in the data and its natural structure are clearly visible in the 2-dimensional representation, although our method uses no class information.

5. Conclusions and Future Research

We have presented two simple and fast neural network methods for multidimensional scaling. Both methods perform well. Our networks are particularly useful for cases where only dissimilarity data are available (as is often the case in psychology and marketing). Furthermore, our neural networks can be used for dimensionality reduction, exploratory data analysis, and data visualisation.

Possible directions for further research include the development of a new error measure, which takes less time to evaluate than the current $O(n^2)$. Another enhancement could be the addition of a weight factor, such that small distances make a relatively bigger contribution to the total error than the big ones. This would produce a non-linear mapping, possibly showing more structure.

References

[1] Davidson, Mark L., *Multidimensional Scaling,* Wiley Series in Probability and Mathematical Statistics, Wiley, New York, 1983.

[2] Hofmann, Thomas and Joachim M. Buhmann, *Multidimensional Scaling and Data Clustering,* in Advances in Neural Information Processing Systems 7, Morgan Kaufmann Publishers, 1995.

[3] Lowe, David and Michael E. Tipping, *Feed-forward Neural Networks and Topographic Mappings for Exploratory Data Analysis,* Neural Computing and Applications 4, 83–95, 1996.

[4] Mao, Jianchang and Anil K. Jain, *Artificial Neural Networks for Feature Extraction and Multiviariate Data Projection,* IEEE Transactions on Neural Networks 6, 296–317, 1995.

[5] Schiffman, Susan S., M. Lance Reynolds and Forrest W. Young, *Introduction to Multidimensional Scaling — Theory, Methods and Applications,* Academic Press, New York, 1981.

[6] Wish, Myron and J. Douglas Carroll, *Multidimensional Scaling and its Applications,* in *Krishnaiah, P. R. and L.N. Kanal (eds.)*: Handbook of Statistics, Vol. 2: Classification, Pattern Recognition and Reduction of Dimensionality, 317–345, North Holland, Amsterdam, 1982.