# Retraining the Neural Network for Data Visualization

Viktor Medvedev, Gintautas Dzemyda
Institute of Mathematics and Informatics
Akademijos str. 4, LT-08663 Vilnius, LITHUANIA
{Viktor.m, Dzemyda}@ktl.mii.lt

**Abstract.** In this paper, we discuss the visualization of multidimensional data. A well-known procedure for mapping data from a high-dimensional space onto a lower-dimensional one is Sammon's mapping. The algorithm is oriented to minimize the projection error. We investigate an unsupervised backpropagation algorithm to train a multilayer feed-forward neural network (SAMANN) to perform the Sammon's nonlinear projection. Sammon mapping has a disadvantage. It lacks generalization, which means that new points cannot be added to the obtained map without recalculating it. The SAMANN network offers the generalization ability of projecting new data, which is not present in the original Sammon's projection algorithm. Retraining of the network when the new data points appear has been analyzed in this paper.

## 1 Introduction

Feature extraction is the process of mapping the original features into fewer features, which preserve the main information of the data structure. Feature extraction for exploratory data projection enables high-dimensional data visualization for better data structure understanding and for cluster analysis [4]. Furthermore, when the dimensionality of the projection space is two-dimensional the structure of the original dataset can be inspected visually and conclusions on clustering tendencies can be straightforwardly drawn.

The problem of data projection is defined as follows: given a set of high dimensional data points, project them to a low-dimensional space so that the result configuration would perform better than the original data in further processing such as clustering, classification, indexing and searching [3, 5]. Data projection has important applications in pattern analysis, data mining, and neural science. The visual inspection of the data can provide a deeper insight into the data, since clustering tendencies or a low intrinsic dimensionality in the data may become apparent from the projection. In general, this projection problem can be formulated

Please use the following format when citing this chapter:
Medvedev, Viktor, Dzemyda, Gintautas, 2006, in IFIP International Federation for Information Processing, Volume 204, Artificial Intelligence Applications and Innovations, eds. Maglogiannis, I., Karpouzis, K., Bramer, M., (Boston: Springer), pp. 27–34

as mapping a set of $n$ vectors from an $d$-dimensional space onto an $m$-dimensional space, with $m<d$.

A large number of approaches for data projection are available in pattern recognition literature [2, 3]. A well-known method to project data is Principal Component Analysis (PCA) which provides mean-square optimized linear projection of data. Another classic method is the Multi-Dimensional Scaling (MDS) that works with inter-point distances and gives a low-dimensional configuration that represents the given distances best. One of the popular MDS-type projection algorithms is Sammon's method [9]. It is a simple but useful nonlinear projection technique that attempts to create a two-dimensional configuration of points in which interpattern distances are preserved. Sammon's mapping is an iterative nonlinear procedure.

The problem of finding the right configuration in a low-dimensional space is an optimization problem: we are interested in obtaining such a configuration that some stress function yields minimum. In general, this optimization problem is difficult because of the very high dimensionality of the parameter space. The stress function is optimal when all the original distances $d_{ij}^{*}$ are equal to the distances of the projected points $d_{ij}$. However, this is not likely to happen exactly.

The finding a projected map usually starts from the initial configuration of points (e.g. randomly chosen), and then the stress is calculated. Next, the configuration is improved by shifting around all points in small steps to approximate better and better the original distances (thus decreasing the stress). This process is reiterated, until the map corresponding to a (local) minimum of the stress is found.

Mao and Jain [7] have suggested a neural network implementation of Sammon's mapping. A specific backpropagation-like learning rule has been developed to allow a normal feedforward artificial neural network to learn Sammon's mapping in an unsupervised way, called SAMANN. As an alternative to SAMANN's unsupervised learning rule, one could also train a standard feedforward artificial neural network, using supervised backpropagation on a previously calculated Sammon's mapping. Although it requires much more computation, as it involves two learning phases (one for Sammon's mapping, one for the neural network), it should perform at least as well as SAMANN [8].

In Mao and Jain's implementation the network is able to project new patterns after training – a property Sammon's mapping does not have. A drawback of using SAMANN is that the original dataset has to be scaled for the artificial neural network to be able to find a correct mapping, since the neural network can only map to points in the sigmoid's output interval, $(0,1)$. This scaling is dependent on the maximum distance in the original dataset. It is therefore possible that a new pattern, shown to the neural network, will be mapped incorrectly, when its distance to a pattern in the original dataset is larger than any of the original interpattern distances. Another drawback of using SAMANN is that it is rather difficult to train and it is extremely slow.

In this paper, we proposed two strategies for retraining the neural network that realizes multidimensional data visualization.

## 2 A Neural Network for Sammon's Projection

Sammon's nonlinear mapping is an iterative procedure to project high-dimensional data into low-dimensional configurations. Sammon used a steepest descent method (diagonal Newton method) for optimization.

Suppose that we have $n$ data points, $X_i = (x_{i1}, x_{i2} \ldots, x_{id})$, $i = 1, \ldots, n$, in a $d$-space and, respectively, we define $n$ points, $Y_i = (y_{i1}, y_{i2} \ldots, y_{im})$, $i = 1, \ldots, n$, in a $m$-space ($m<d$). The pending problem is to visualize these $d$-dimensional vectors $X_i, i = 1, ..., d$ onto the plane $R^2$. Let $d_{ij}^*$ denote the distance between $X_i$ and $X_j$ in the input space, and $d_{ij}$ denote the distance between the corresponding points $Y_i$ and $Y_j$ in the projected space. The Euclidean distance is frequently used. The projection error measure $E$ is as follows:

$$E = \frac{1}{\sum_{\substack{i,j=1 \\ i<j}}^{n} d_{ij}^*} \sum_{\substack{i,j=1 \\ i<j}}^{n} \frac{(d_{ij}^* - d_{ij})^2}{d_{ij}^*} \tag{1}$$

$E$ is commonly referred to as Sammon's stress. It is a measure of how well the interpattern distances are preserved when the patterns are projected from a higher-dimensional space to a lower-dimensional space. The stress equal to 0 indicates a lossless mapping. The steepest descent procedure may be used to search for a minimum of $E$. Sammon's stress is designed so that short distances contribute more to the value of $E$. In the process of minimizing $E$, therefore, the mapping gives a greeter priority to the preservation of short distances rather than the long ones. That is why the mapping is capable of unfolding high dimensional data manifolds. Though the algorithm also considers long distances, however, it may fail to unfold strongly twisted patterns.

Sammon's algorithm involves a large amount of computations. Since, $n(n-1)/2$ distances have to be computed for every step within an iteration, the algorithm soon becomes impractical for a large number of patterns. Sammon's algorithm does not provide an explicit function governing the relationship between patterns in the original space and in the configuration (projected) space. Therefore, it is impossible to decide where to place the new $d$-dimensional data in the final $m$-dimensional configuration created by Sammon's algorithm. Sammon's algorithm has no generalization capability. In order to project new data, one has to run the program again on pooled data (old data and new data) [4].

SAMANN network for two-dimensional projection is given in Figure 1. It is a feedforward neural network where the number of input units is set to be the feature space dimension $d$, and the number of output units is specified as the extracted feature space dimension $m$. They have derived a weight updating rule for the multilayer perceptron neural network that minimizes Sammon's stress, based on the gradient descent method. The general updating rule for all the hidden layers, $l = 1, \ldots, L-1$ and for the output layer ($l = L$) is:

$$\Delta\omega_{jk}^{(l)} = -\eta \frac{\partial E_{\mu\nu}}{\partial \omega_{jk}^{(l)}} = -\eta(\Delta_{jk}^{(l)}(\mu) y_j^{l-1}(\mu) - \Delta_{jk}^{(l)}(\nu) y_j^{l-1}(\nu)) \tag{2}$$

where $\omega_{jk}$ is the weight between the unit $j$ in the layer $l-1$ and the unit $k$ in the layer $l$, $\eta$ is the learning rate, $y_j^{(l)}$ is the output of the $j$th unit in the layer $l$, and $\mu$ and $\nu$ are two patterns. The $\Delta_{jk}^{(l)}$ are the errors accumulated in each layer and backpropagated to a preceding layer, similarly to the standard backpropagation. The sigmoid activation function whose range is $(0.0,1.0)$ is used for each unit. However, in the neural network implementation of Sammon's mapping the errors in the output layer are functions of the interpattern distances. In each learning step, the artificial neural network is shown by two points. The outputs of each neuron are stored for both points. The distance between the neural network output vectors can be calculated and an error measure can be defined in terms of this distance and the distance between the points in the input space. From this error measure a weight update rule can be derived. Since no output examples are necessary, this is an unsupervised algorithm.

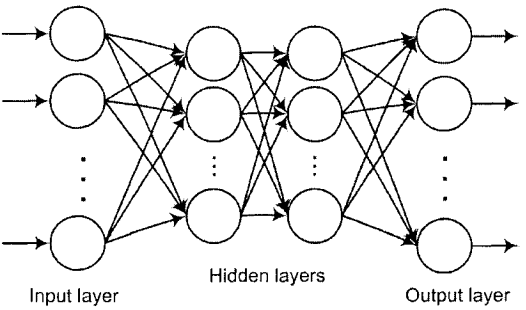


**Fig. 1.** SAMANN network for two-dimensional projection

The SAMANN Unsupervised Backpropagation Algorithm [7] is as follows:

1. Initialize the weights randomly in the SAMANN network.
2. Select a pair of patterns randomly, present them to the network one at a time, and evaluate the network in a feedforward fashion.
3. Update the weights in the backpropagation fashion starting from the output layer.
4. Repeat steps 2-3 a number of times.
5. Present all the patterns and evaluate the outputs of the network; compute Sammon's stress; if the value of Sammon's stress is below a prespecified threshold or the number of iterations (from steps 2-5) exceeds the prespecified maximum number, then stop; otherwise, go to step 2.

The rate, at which artificial neural networks learns, depends upon several controllable factors. Obviously, a slower rate means that a lot more time is spent in accomplishing the learning to produce an adequately trained system. At the faster learning rates, however, the network may not be able to make the fine discriminations possible with a system that learns more slowly. When the learning rate is very small, the weight adjustments tend to be very small. Thus, if $\eta$ is small when the algorithm is initialized, the network will probably take an unacceptably long time to converge.

## 3 Strategies for Retraining of the SAMANN Network

After training the SAMANN network, a set of weights of the neural network are fixed. A new vector shown to the network is mapped into the plane very fast and quite exactly without any additional calculations. However, while working with large data amounts there may appear a lot of new vectors, which entails retraining of the SAMANN network after some time. That is why two strategies for retraining the neural network that realizes multidimensional data visualization have been proposed and then analysis made. Retraining of the network has to be efficient and the training algorithm has to converge rapidly. It has been established that training of the SAMANN neural network requires much calculations, therefore we strive to obtain new weights and a precise data projection as soon as possible.

The strategies of the neural network retraining data are as follows:

1. The SAMANN network is trained by $N_1$ initial vectors, a set of weights $\omega_1$ is obtained, then the visualization error $E(N_1)$ is calculated and vector projections are localized on the plane. After the emergence of $N_2$ new vectors, the neural network is retrained with all the $N_1+N_2$ vectors, and after each iteration the visualization error $E(N_1+N_2)$ is calculated and the computing time is measured. The new set of SAMANN network weights $\omega_2$ is found.
2. The SAMANN network is trained by $N_1$ initial vectors, a set of weights $\omega_1$ is obtained, and the visualization error $E(N_1)$ is calculated. Since in order to renew the weights $\omega$, a pair of vectors $\mu$ and $\nu$ is simultaneously provided for the neural network, the neural network is retrained with $2*N_2$ vectors at each iteration: at each step of training one vector is taken from the primary dataset and the other from the new one. After each iteration the visualization error $E(N_1+N_2)$ is calculated and the computing time is measured. The new set of network weights $\omega_2$ is found.

Two datasets have been used in the experiments:

1. Iris Dataset (Fisher's iris dataset) [10]. A real dataset with 150 random samples of flowers from iris species setosa, versicolor, and virginica. From each species there are 50 observations of sepal length, sepal width, petal length, and petal width in cm. The iris flowers are described by 4 attributes.
2. 300 randomly generated vectors $X_i = (x_{i1},...,x_{in}) \in R^n$ (three spherical clusters with 100 vectors each, n=5):

$$x_{ij} \in [0,0.2],\ i=1,...,100;\ j=1,...,5,\ \sqrt{\sum_{j=1}^{n}(0.1-x_{ij})^2} \le 0.1$$

$$x_{ij} \in [0.4,0.6],\ i=101,...,200;\ j=1,...,5,\ \sqrt{\sum_{j=1}^{n}(0.5-x_{ij})^2} \le 0.1$$

$$x_{ij} \in [0.8,1],\ i=201,...,300;\ j=1,...,5,\ \sqrt{\sum_{j=1}^{n}(0.9-x_{ij})^2} \le 0.1$$

These two datasets were divided into two parts: the primary dataset and the set of new vectors. The first part is used for primary training of the SAMANN network, while the new part together with the primary dataset – for retraining the network.

In the analysis of strategies for the network retraining, a particular case of the SAMANN network was considered: a feedforward artificial neural network with one hidden layer and two outputs ($d$=2). In each case, the same number ($n_2$=20) of neurons of the hidden layer was taken and the set of initial weights was fixed in advance. To visualize the initial dataset, the following parameters were employed: the number of iterations $M$=10000, the training parameter $\eta = 10$; to visualize the set of new vectors: the training parameter was $\eta = 1$, and the number of iterations depended on the strategy chosen. One iteration in our research means showing all pairs of samples to the neural network once.

In the Iris dataset, 50 vectors were used for retraining. In the randomly generated set, 90 vectors were used for retraining: 30 vectors for the different clusters.

When calculating, the time of algorithm performance was measured. Figure 2 and Figure 3 demonstrate the results of calculation. Only the results of retraining the SAMANN network with the new vectors are indicated in the figures. The first strategy yield good results, however retraining of the network is slow. The best visualization results are obtained by taking points for network retraining from the primary dataset and the new dataset (second strategy). The second strategy enables us to attain good visualization results in a very short time as well as to get smaller visualization errors and to improve the accuracy of projection as compared to other strategies (Figure 3 illustrates this fact best in the experiment with the dataset of random numbers). The proposed second strategy makes it possible to reduce the duration of calculation a great deal in case there are considerably less new vectors than the initial ones.

Figures 4a and 4b illustrate mapping results of the iris dataset in two different cases: (1) the network has been trained by 150 vectors (Figure 4a); (2) the network has been trained by 100 vectors and retrained using the second strategy (Figure 4b). The interlocation of points is similar in the figures. This indicates a good quality of the retraining. Very high similarity of Figures 4a and 4b leads to the idea of possibility to minimize the training time consumption via dividing the training process into two subprocesses: (1) training of the network by a part of the data vectors; (2) retraining of the network by the remaining part of the dataset.

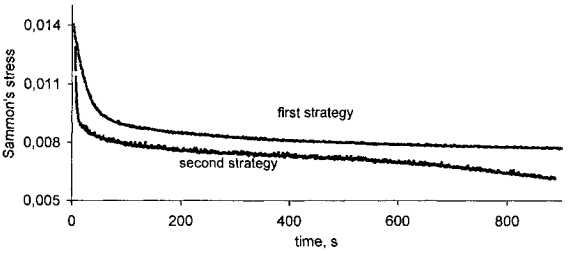


**Fig. 2.** Dependence of the projection error on the computing time for the Iris dataset

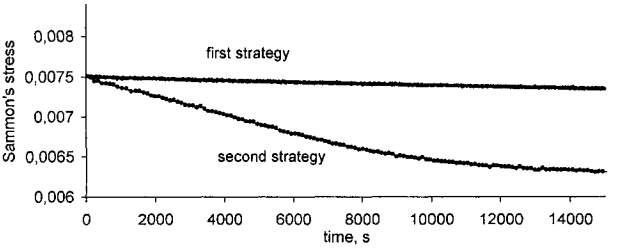


**Fig. 3.** Dependence of the projection error on the computing time for randomly generated vectors

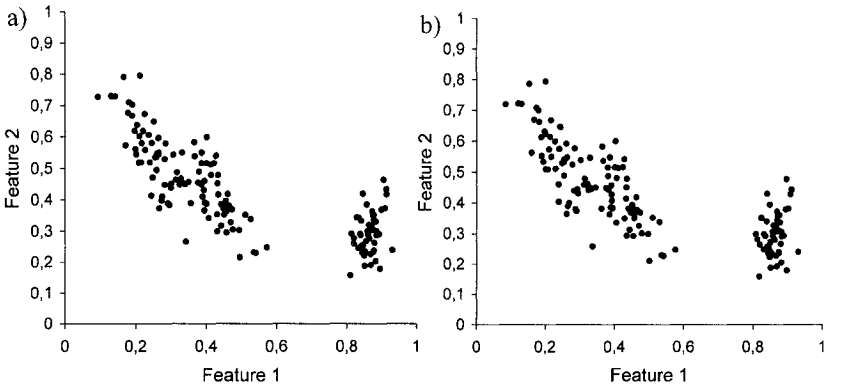


**Fig. 4.** Mapping results of the Iris dataset: a) training by 150 vectors; b) training by 100 vectors and retraining by 50 vectors

## 4 Conclusions

Mapping problem usually is formulated as an optimization one. The experiments were carried out both on artificial and real data. Retraining of the SAMANN network when the new data points appear and the ability of network generalization to visualize new data have been analyzed.

Two strategies for retraining the neural network that visualizes multidimensional data have been proposed and investigated. It is important that retraining of the neural network were efficient and the training algorithm were faster convergent, therefore effort was put to obtain a new set of weights in a shorter time. The experiments have shown that it is expedient to take one vector from the primary dataset and the other from the new one at every step of training. This strategy yields smaller visualization errors faster.

## References

1. Anderson, D. and McNeill, G.: Artificial neural networks technology. DACS State-of-the-Art Report ELIN: A011, Rome Laboratory, RL/C3C Griffiss AFB, NY 13441-5700, 20 Aug. 1992.
2. Fugunaga, K.: Introduction to Statistical Pattern Recognition, 2nd ed. New-York: Academic, 1990.
3. Jain, A.K. and Dubes, R.C.: Algorithms for Clustering Data. Prentice-Hall, 1988.
4. Jain, A.K. and Mao, J.: Artificial neural network for nonlinear projection of multivariate data, Neural Networks, 1992. IJCNN., International Joint Conference on Volume 3, 7-11 June 1992 Page(s):335 - 340 vol.3.
5. Jain, A.K., Duin, R., and Mao, J.: Statistical pattern recognition: A review. IEEE Trans. Pattern Analysis and Machine Intelligence, 22(1):4–37, Jan. 2000.
6. Lerner, B., Gutterman, H., Aladjem, M., Dinstein, I. and Romem, Y.: Feature extraction by neural network nonlinear mapping for pattern classification.
7. Mao, J. and Jain, A.K.: Artificial neural networks for feature extraction and multivariate data projection, IEEE Trans. Neural Networks, vol.6, pp. 296-317, 1995.
8. de Ridder, D., Duin, R.P.W.: Sammon's mapping using neural networks: A comparison. Pattern Recognition Letters 18 (1997), 1307-1316.
9. Sammon, J.J.: A nonlinear mapping for data structure analysis. IEEE Trans. Computer, C-18(5):401-409, May 1969.
10. Fisher, R. A.: The use of multiple measurements in taxonomic problem. Annual Eugenics, vol. 7, part II, pp 179-188, 1936.