Journal of Global Optimization (2006) 35: 607–623
DOI 10.1007/s10898-005-5368-1

Optimization of the Local Search in the Training for SAMANN Neural Network

VIKTOR MEDVEDEV and GINTAUTAS DZEMYDA
Institute of Mathematics and Informatics, Akademijos str. 4, LT-08663 Vilnius, Lithuania (e-mail: Viktor.m@ktl.mii.lt)

(Received 17 November 2005; accepted in revised form 21 November 2005)

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. This algorithm preserves as well as possible all interpattern distances. 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. To save computation time without losing the mapping quality, we need to select optimal values of control parameters. In our research the emphasis is put on the optimization of the learning rate. The experiments are carried out both on artificial and real data. Two cases have been analyzed: (1) training of the SAMANN network with full data set, (2) retraining of the network when the new data points appear.

Key words: Learning rate, SAMANN neural network, Sammon's mapping, Visualization

1. Introduction

Feature extraction is the process of mapping the original features (measurements) 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 [5]. Furthermore, when the dimensionality of the projection space is two-dimensional (2D) 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 [4, 6]. 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 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 [4]. 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 [13]. It is a simple but useful nonlinear projection technique that attempts to create a 2D configuration of points in which interpattern distances are preserved. Sammon's mapping is an iterative nonlinear procedure.

Mapping problem usually is formulated as an optimization one.

A mapping f transforms a pattern X of a d-dimensional space to a pattern Y of an m-dimensional projected space, $m < d$, that is, $Y = f(X)$, such that a criterion J is optimized. The mapping $f(X)$ is determined from among all the transformations $g(X)$, as the one that satisfied, $J\{f(X)\} = \max_g J\{g(X)\}$. The mappings differ by the functional forms of $g(Y)$ and by the criteria they have to optimize.

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 the 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. Therefore, the found distances will be distorted representations of the relations within the data. The larger the stress, the greater the distortion.

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 [8] 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 [11].

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 look for optimal values of the control parameters of SAMANN network training seeking to speed up the learning process. Two cases are analyzed: (1) training of the SAMANN network with full data set, (2) retraining of the SAMANN network when the new data points appear.

2. Local Search in Sammon's Nonlinear Mapping

Sammon's nonlinear mapping is an iterative procedure to project high-dimensional data into low-dimensional configurations. Suppose that we have n data points, $X_i = (x_{i1}, x_{i2} \dots, x_{id})$, $i = 1, \dots, n$, in a d-space and, respectively, we define n points, $Y_i = (y_{i1}, y_{i2} \dots, y_{im})$, $i = 1, \dots, n$, in a m-space $(m < d)$. The pending problem is to visualize these d-dimensional vectors X_i, $i = 1, \dots, d$ 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:

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

The coordinates y_{ik}, $i = 1, \dots, n, k = 1, 2$ (i.e. $m = 2$) of the 2D vectors $Y_i \in R^2$ are optimized by the iteration formula:

$$y_{ik}(p'+1) = y_{ik}(p') - \eta \frac{\frac{\partial E(p')}{\partial y_{ik}(p')}}{\left|\frac{\partial^2 E(p')}{\partial y^2_{ik}(p')}\right|} \qquad \frac{\partial E}{\partial y_{ik}} = -\frac{2}{c} \sum_{\substack{j=1 \\ i \neq j}}^{n} \left(\frac{d^*_{ij} - d_{ij}}{d^*_{ij} d_{ij}}\right)(y_{ik} - y_{jk})$$

$$\frac{\partial^2 E}{\partial y_{ik}^2}=-\frac{2}{c}\sum_{\substack{j=1\\ i\neq j}}^{n}\frac{1}{d_{ij}^* d_{ij}}\left[\left(d_{ij}^*-d_{ij}\right)-\frac{\left(y_{ik}-y_{jk}\right)^2}{d_{ij}}\left(1+\frac{d_{ij}^*-d_{ij}}{d_{ij}}\right)\right],\quad c=\sum_{\substack{i,j=1\\ i<j}}^{n} d_{ij}^*.$$

Here p' denotes the iteration order number, η is called by the learning rate. Usually such search gives the local minimum of E. It depends on the starting configuration of $Y_i, i=1,n$. The convergence rate and manner depends on η. One of the problems is to optimize η.

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 loss less 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.

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) [5].

3. A Neural Network for Sammon's Projection

SAMANN network for 2D 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:

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

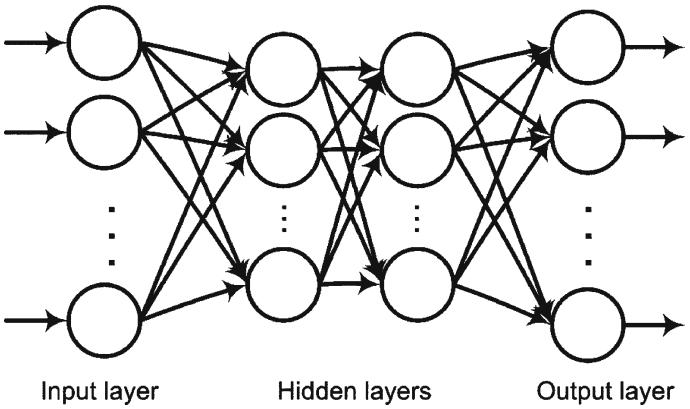

Figure 1. SAMANN network for 2D projection.

where ω_{jk} is the weight between the unit j in the layer $l-1$ and the unit k in the layer l, η is the learning rate, $y_j^{(l)}$ is the output of the jth unit in the layer l, and μ and ν 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.

The SAMANN unsupervised backpropagation algorithm [8] 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.

One iteration in our research means showing all pairs of samples to the neural network once.

4. Control Parameters of the SAMANN Algorithm

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 η is small when the algorithm is initialized, the network will probably take an unacceptably long time to converge.

As usual, several factors besides time have to be considered when discussing the training task. Network complexity, size, paradigm selection, architecture, type of the learning rule or rules employed, and a desired accuracy must everything be considered. These factors play a significant role in determining how long it will take to train a network. Changing any one of these factors may either extend the training time to an unreasonable length or even result in an unacceptable accuracy.

Learning rate η usually is positive and usually ranging between zero and one. If the learning rate is greater than one, it is easy for the learning algorithm to overshoot in correcting the weights, and the network will oscillate. Small values of the learning rate will not correct the current error fast, but if small steps are taken in correcting errors, there is a good chance of arriving at the best minimum convergence.

The backpropagation algorithm is made more powerful by adding a momentum term as follows:

$$\Delta\omega_{\text{current}} = -\eta \frac{\partial E}{\partial \omega_{\text{current}}} + \alpha \Delta\omega_{\text{previous}}$$

where $\Delta\omega_{\text{previous}}$ represents the previous weight adjustment, and $0 \leqslant \alpha \leqslant 1$. Thus, the new component $\alpha \Delta\omega_{\text{previous}}$ represents a fraction of the previous weight adjustment for a given weight. A momentum component will help to damp the oscillations around the optimality by encouraging the adjustments to stay in the same direction.

5. Optimization for Control Parameters

This paper deals with the projection of datasets onto 2D's using the SAMANN network. It has been noticed that the training depends on different parameters. In the experiments, Mao and Jain [8] have employed the next parameters when training the SAMANN network: a two-layer (one

hidden layer) network with 20 hidden units; the SAMANN network is trained using the standard backpropagation algorithm with the learning rate of 0.7 and the momentum value of 0.3 for 2,00,000 iterations; afterwards, the unsupervised backpropagation algorithm is used to train the SAMANN network for 60,000 iterations, the learning rate and momentum in the unsupervised backpropagation algorithm being 0.02 and 0.01, respectively, for the artificial datasets, and 0.05 and 0.02, respectively, for the real datasets. The experiments, done in this paper, show in what way the SAMANN network training depends on the learning rate and the momentum term.

In this paper we apply the gradient steepest descent procedure (diagonal Newton method) to search for a minimum of the stress. The possibilities to apply more sophisticated methods for this purpose are investigated in Pohlheim [14].

Pohlheim [14] uses two different methods for optimization. First, a standard optimization method included in MATLAB was employed (BFGS Quasi-Newton method with a mixed quadratic and cubic line search procedure – [15]). Later, a more robust search method was employed, the RPROP algorithm [16]. The RPROP algorithm uses only the changes in the sign of the gradient for step size control. It is widely used in the field of neural networks. Both optimization algorithms (BFGS Quasi-Newton and RPROP) produced good results. However, there are no advantages signed compared with the classic descent procedure. In all the cases, using gradient-based optimization the projection with a very small error can be searched for.

It is possible to formulate an optimization problem to search for optimal set of parameter values. The Bayesian approach [9, 10] or other regular methods of global optimization maybe used. The objective function for the minimization here is the mapping error obtained after the application of the network training procedure. The training of the SAMANN network is very time-consuming operation. It restricts essentially the application of the methods above. Therefore, we use simpler strategy for optimization of the control parameters.

The following datasets were used in the experiments:

1. Iris dataset (Fisher's iris dataset) [2]. 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. Salinity dataset [12]. That is a set of measurements of water salinity (i.e., its salt concentration) and river discharge is taken in North Carolina's Pamlico Sound. 28 subjects, 4 variables (Lagged Salinity, Trend, Discharge, Salinity).

3. HBK dataset (Artificial dataset generated by Hawkins et al. [3]). It consists of 75 four-dimensional patterns. The HBK dataset is an artificially constructed dataset with 14 outliers.
4. Artificial datasets generated randomly. The six random datasets are comprised of 10, 20, 50, 100, 150 and 200 four-dimensional patterns.

In all the experiments a two-layer (one hidden layer) network with 20 hidden units is used.

First of all, three experiments with Iris dataset have been carried out that show in what way the network training rate depends on the momentum term:

a. The SAMANN network is trained using the standard backpropagation algorithm with a learning rate of 0.7 and without momentum term for 2,60,000 iterations.
b. The SAMANN network is trained using the standard backpropagation algorithm with a learning rate of 0.7 and momentum value of 0.3 for 2,00,000 iterations. Then the unsupervised backpropagation algorithm is used to train the SAMANN network for 60,000 iterations. The learning rate and momentum in unsupervised algorithm are 0.05 and 0.02, respectively.

In Figure 2 the dependence of the projection error on the number of iterations is presented. Note that, when experimenting with the momentum term with parameters described in case b, better results have been obtained, that is, the momentum term makes it possible to speed up the network training and the better results thereby are obtained in a shorter time interval. By using the parameters proposed by Mao and Jain [8], we have succeeded in projecting the Iris dataset onto the plane (Figure 5c), the projection error (Sammon's stress) being $E = 0.00518$. Meanwhile, Mao and Jain [8] have obtained a similar error $E = 0.007$ in the projection of the Iris dataset using the SAMANN network with the same network parameters.

When projecting data, it is of great importance to achieve good results in a short time interval. In the consideration of the SAMANN network, it has been observed that the projection error depends on different parameters. The latest investigations have revealed that in order to achieve good results, one needs to correctly select the learning rate η. It has been started so far that projection yields the best results if the η value is taken from the interval $(0, 1)$. Mao and Jain [8] used $\eta = 0,7$ in their study. In that case, however, the network training is very slow. One of the possible reasons is that, in the case of SAMANN network, the interval $(0, 1)$ is not the best one. Thus, it is reasonable to look for the optimal value of the learning parameter that may not necessarily be within the interval $(0, 1)$.

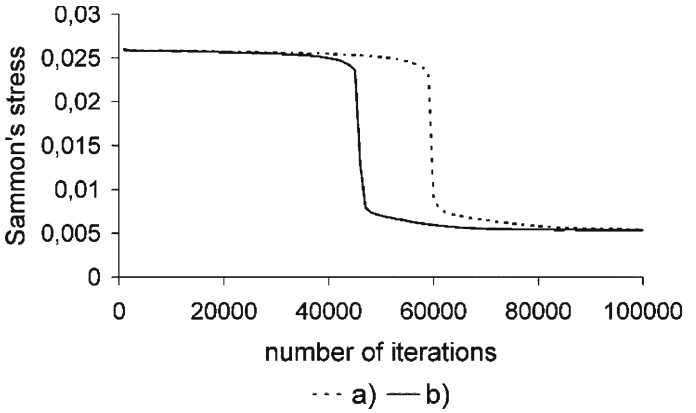

Figure 2. Dependence of the projection error on the number of iterations.

The experiments have been done with real datasets, using 28, 75 and 150 four-dimensional patterns (Salinity dataset, HBK dataset, Iris dataset). At first the dependence of the data projection accuracy on the learning rate η has been defined for $\eta \in (0, 1)$. The results obtained are illustrated in Figure 3. This figure demonstrates that with an increase in the learning rate value, a better projection error is obtained. That is why the experiments have been done with higher values of the learning rate beyond the limits of the intervals $(0, 1)$. The results are presented in Figure 4. It has been noticed that the best results are at $\eta \in [1, 100)$. We can conclude from Figures 3 and 4 that the optimal value of the learning rate for the datasets considered is within the interval $[10, 30]$. In the case of the Salinity dataset, the optimal value of the learning rate is $\eta = 10$, for HBK dataset $\eta = 10$, and for Iris dataset $\eta = 30$. At these values of the learning rate we obtain the best projection results, that is, the data are projected more rapidly and more exactly. For the fixed number of iterations, good projection results are obtained in a shorter time interval than that taking the learning rate values from the interval$(0, 1)$.

While experimenting the computing time and the errors obtained in each iteration have been defined as well. In Figures 5–7, the dependence of the projection error on the number of iterations is presented at different values of the learning rate for three real datasets.

Figure 5c specifies Figure 5b at the beginning of iterations (first 1000 iterations). In each figure only several cases are described. Figures 5–7 indicate that the higher the value of the learning rate, the more rapidly one succeeds in getting good results (i.e., sufficiently low projection error). However, with an increase in the value of the learning rate, the error variations also increase, which can cause certain network training problems. Figure 8(a–c) illustrates 2D projection maps of the Salinity, HBK and Iris datasets

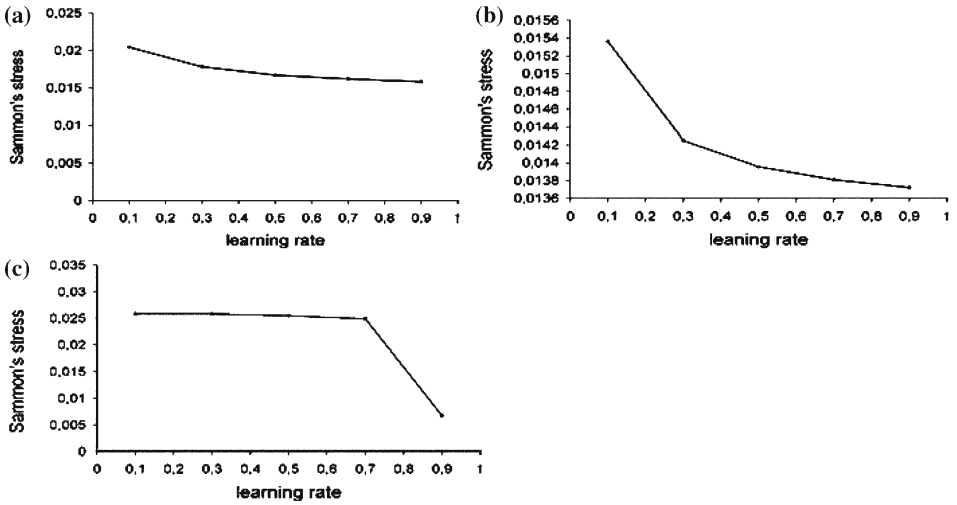

Figure 3. (a – Salinity dataset, b – HBK dataset, c – Iris dataset) The dependence of the data projection accuracy on the learning rate $\eta, \eta \in (0, 1)$.

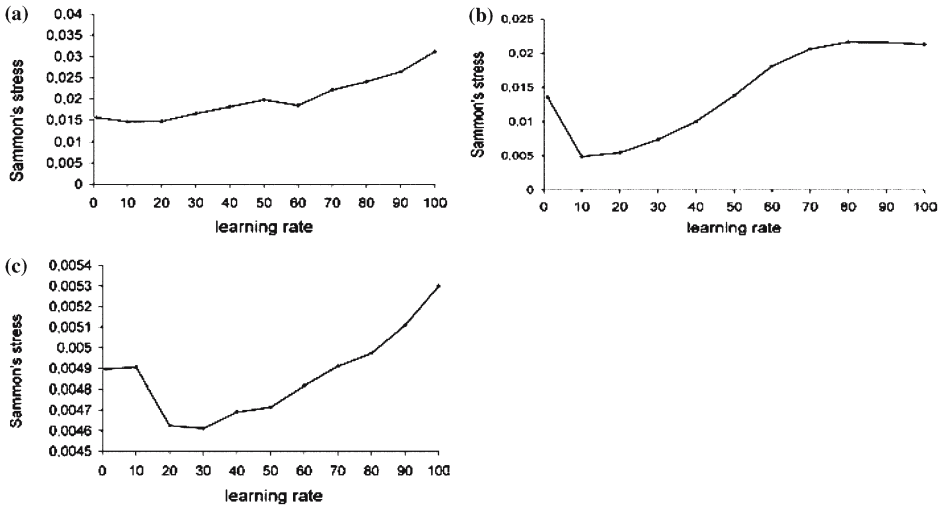

Figure 4. (a – Salinity dataset, b – HBK dataset, c – Iris dataset) The dependence of the data projection accuracy on the learning rate $\eta, \eta \in [1, 100)$.

using the SAMANN network at the optimal value η of the learning rate, defined before.

The investigations have also been performed with datasets generated randomly. The target was to find out what the optimal learning rate value

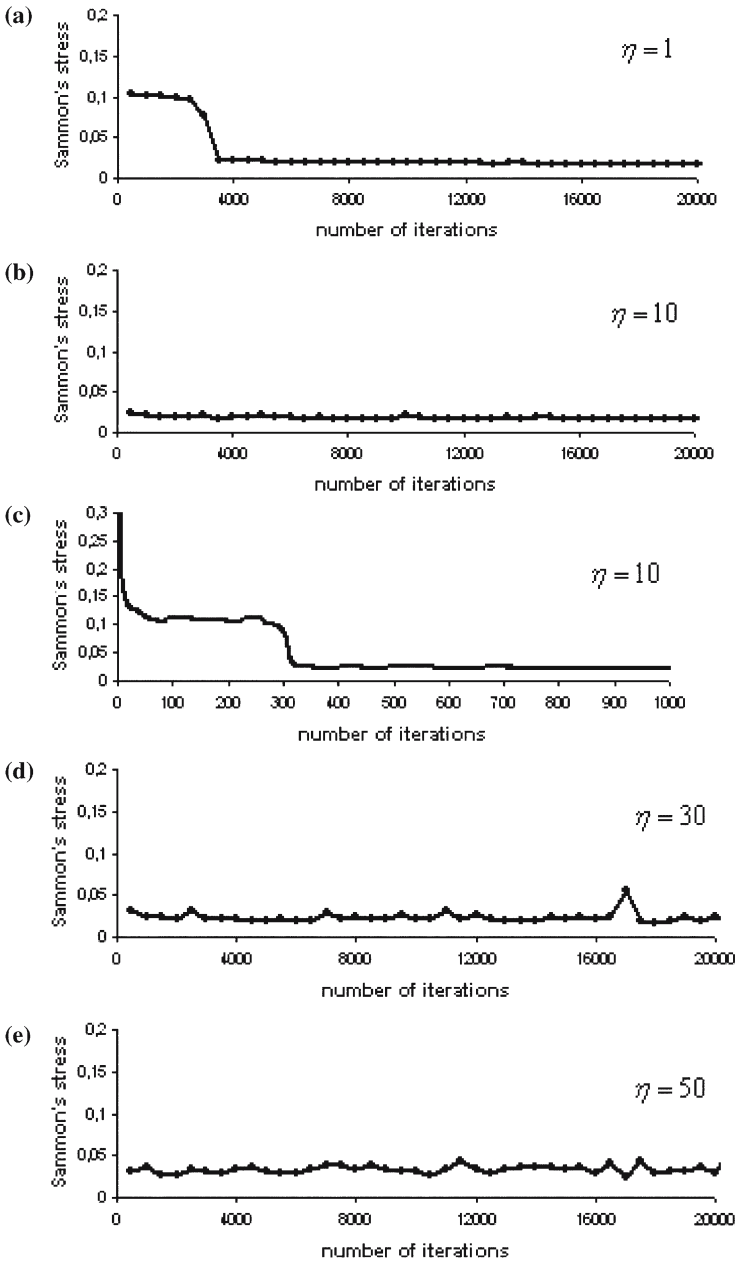

Figure 5. The dependence of the projection error on the number of iterations for the Salinity dataset.

should be and how it changes with an increase in the number of training vectors. The datasets considered are comprised of 10, 20, 50, 100, 150 and 200 vectors.

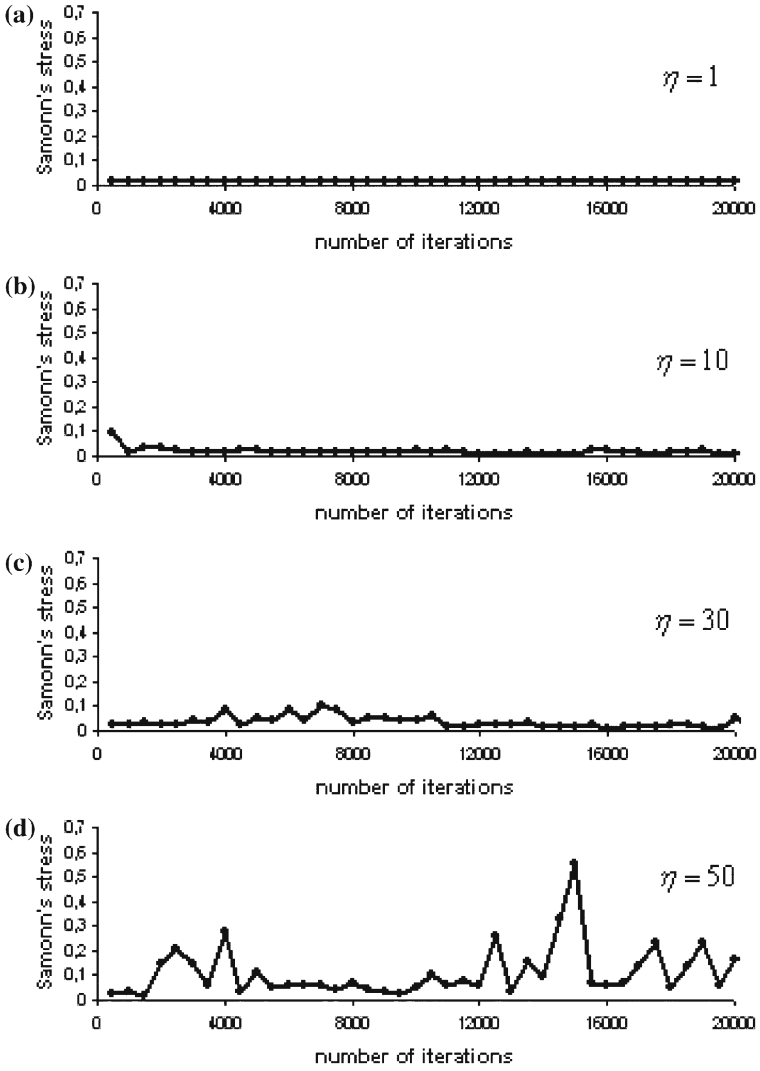

Figure 6. The dependence of the projection error on the number of iterations for the HBK dataset.

The SAMANN network was initialized at random values ranging $(-0.01, 0.01)$. The network was trained for 7000 iterations. The momentum term was set to be 0.05, and the learning rate for each experiment was in within the interval $[1, 50]$. The results are given in Figure 9. Different values of the optimal learning rate have been got for different datasets, dependent on the number of vectors of the training dataset. The higher the dataset dimension, the higher the value of the learning rate. The optimal learning rate value η for the datasets considered in within the interval $[5, 35]$.

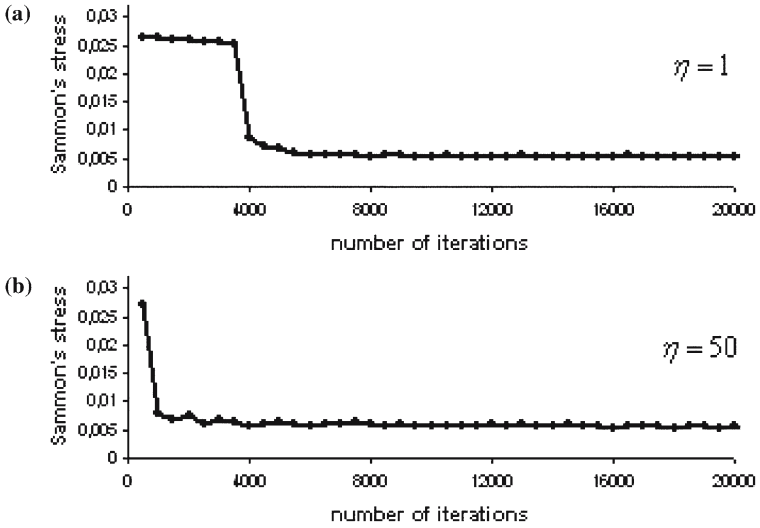

Figure 7. The dependence of the projection error on the number of iterations for Iris dataset.

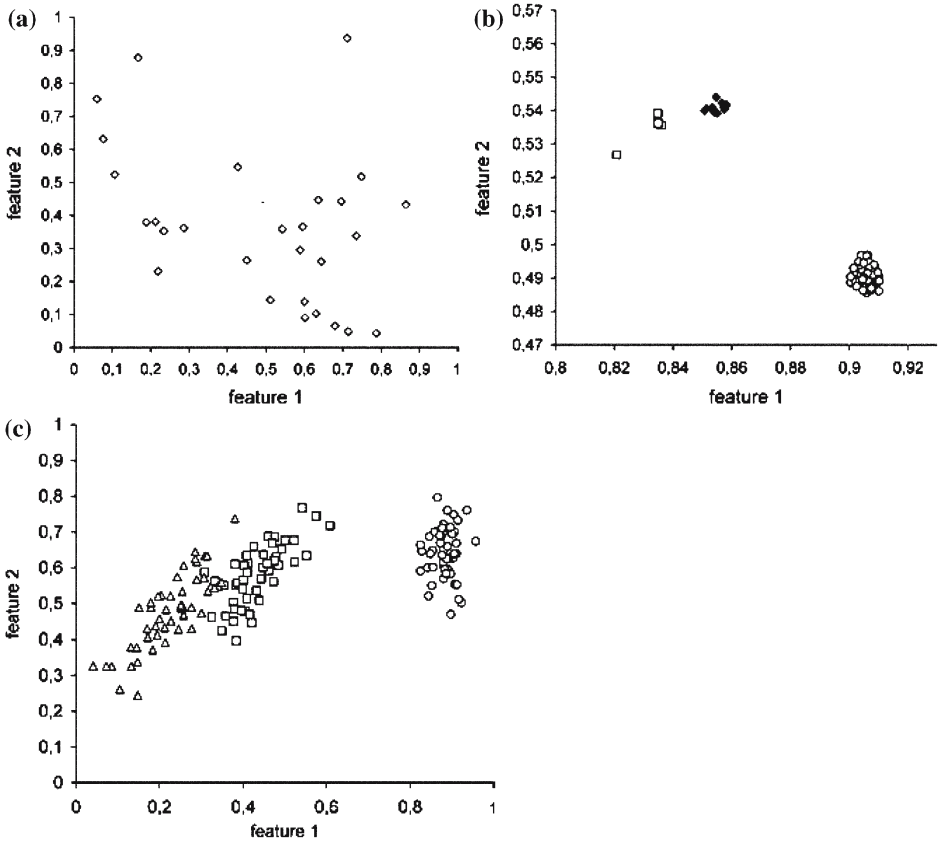

Figure 8. (a – Salinity dataset, b – HBK dataset, c – Iris dataset) 2D projection maps of real datasets using the SAMANN network.

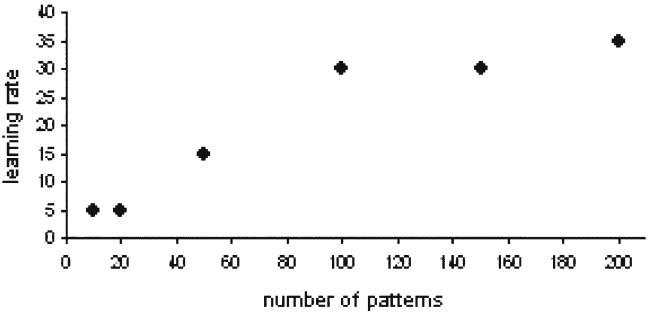

Figure 9. The dependence of the optimal learning rate on the number of patterns in the dataset.

6. 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 ω_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 ω_2 is found.
2. The SAMANN network is trained by N_1 initial vectors, a set of weights ω_1 is obtained, and the visualization error $E(N_1)$ is calculated. Since in order to renew the weights ω, a pair of vectors μ and ν 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 ω_2 is found.

Two datasets have been used in the experiments:

1. Iris dataset [2];
2. 300 randomly generated vectors $X_i = (x_{i1}, \ldots, x_{in}) \in R^n$ (three spherical clusters with 100 vectors each, $n = 5$):

$$x_{ij} \in [0, 0.2], \quad i = 1, \ldots, 100; \quad j = 1, \ldots, 5, \quad \text{sqrt}\left[\sum_{j=1}^{n} (0.1 - x_{ij})^2\right] \leqslant 0.1$$

$$x_{ij} \in [0.4, 0.6], \quad i = 101, \ldots, 200; \quad j = 1, \ldots, 5, \quad \text{sqrt}\left[\sum_{j=1}^{n} (0.5 - x_{ij})^2\right] \leqslant 0.1$$

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

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 $M = 10,000$, 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.

When calculating, the time of algorithm performance was measured. Figures 10 and 11 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 11 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.

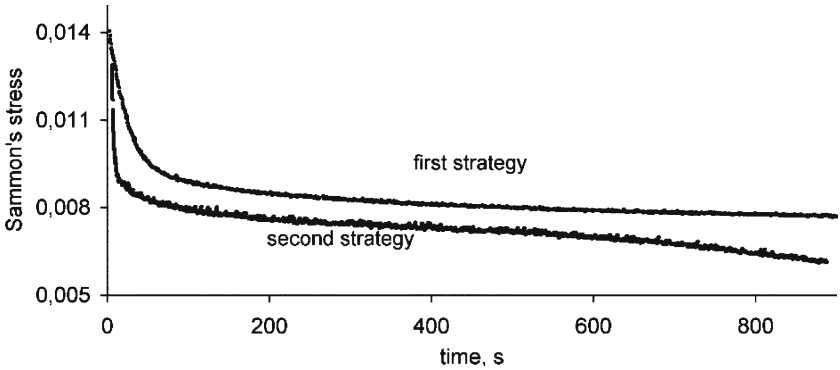

Figure 10. Dependence of the projection error on the computing time for the Iris dataset.

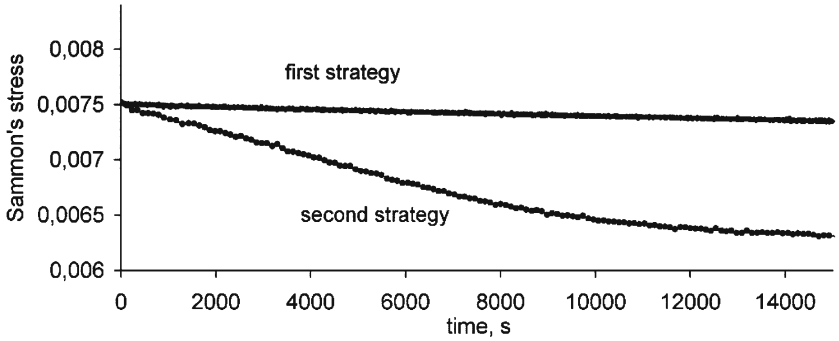

Figure 11. Dependence of the projection error on the computing time for randomly generated vectors.

7. Conclusions

Mapping problem usually is formulated as an optimization one. Seeking better values of the optimization criterion, it is necessary to optimize control parameters that influence the convergence of the local search. The optimal values of the control parameters of SAMANN network training have been discovered seeking to speed up the learning process. The experiments were carried out both on artificial and real data. Two cases were analyzed: training of the SAMANN network with full data set, and retraining of the SAMANN network when the new data points appear.

The experimental investigation shows that the optimal value of the learning rate is in the interval [5,30]. By selecting such values of the learning rate, the significant economy of the computing time is possible (up to 5–7 times) for the fixed number of iterations. Smaller values of the learning rate within the interval (0,1) guarantee a more stable convergence to the minimum of the mapping error. Some fluctuations in the result are observed when the rate is set to be larger. However, these fluctuations are rather small when the learning rate is in the interval [5,30].

The ability of network generalization to visualize new data has 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. (1992), Artificial neural networks technology. DACS State-of-the-Art Report ELIN: A011, Rome Laboratory, RL/C3C Griffiss AFB, NY 13441-5700, 20 Aug.
2. Fisher, R.A. (1936), The use of multiple measurements in taxonomic problem. *Ann. Eugenics* 7, 179–188.
3. Hawkins, D.M., Bradu, D. and Kass, G.V. (1984), Location of several outliers in multiple regression data using elemental sets. *Technometrics* 26, 197–208.
4. Jain, A.K. and Dubes, R.C. (1988), *Algorithms for Clustering Data*. Prentice-Hall.
5. Jain, A.K. and Mao, J. (1992), Artificial neural network for nonlinear projection of multivariate data, Neural Networks. In: *IJCNN, International Joint Conference on Volume 3*, 7–11 June 1992, vol. 3, pp. 335–340.
6. Jain, A.K., Duin, R. and Mao, J. (2000), Statistical pattern recognition: A review. *IEEE Trans. Pattern Analysis and Machine Intelligence* 22(1), 4–37.
7. Lerner, B., Gutterman, H., Aladjem, M., Dinstein, I. and Romem, Y. Feature extraction by neural network nonlinear mapping for pattern classification.
8. Mao, J. and Jain, A.K. (1995), Artificial neural networks for feature extraction and multivariate data projection. *IEEE Trans. Neural Networks*, 6, 296–317.
9. Mockus J. (1989), *Bayesian Approach to Global Optimization*. Kluwer, Dordrecht, Netherlands.
10. Mockus, J., Eddy, W., Mockus, A., Mockus, L. and Reklaitis, G. (1996), *Bayesian Heuristic Approach to Discrete and Global Optimization*. Kluwer, Dordrecht, Netherlands.
11. de Ridder, D., Duin, R.P.W. (1997), Sammon's mapping using neural networks: A comparison. *Pattern Recognition Letters* 18, 1307–1316.
12. Ruppert D. and Carroll, R.J. (1980), Trimmed least squares estimation in the linear model. *Journal of the American Statistical Association* 75, 828–838.
13. Sammon, J.J. (1969), A nonlinear mapping for data structure analysis. *IEEE Trans. Computer, C* 18(5), 401–409.
14. Pohlheim, H. (2005) Multidimensional scaling for evolutionary algorithms – Visualization of the path through search space and solution space using Sammon mapping. Submitted to *Artificial Life Special Issue – The State of the Art in Visualizing Complex Adaptive Systems*.
15. Mathworks, (1994–1999), *The Matlab – User Guide*. The Mathworks, Inc., Natick, MA.
16. Riedmiller, M., Braun, H. (1993), A direct adaptive method for faster backpropagation learning: The RPROP algorithm. In Ruspini, H. (ed.), *Proceedings of the IEEE Int. Conf. on Neural Networks (ICNN)*, pp. 586–591.