ELSEVIER

Pattern Recognition Letters 18 (1997) 1307–1316

Pattern Recognition Letters

Sammon's mapping using neural networks: A comparison

Dick de Ridder *, Robert P.W. Duin

Pattern Recognition Group, Faculty of Applied Physics, Delft University of Technology, Lorentzweg 1, 2628 CJ Delft, The Netherlands

Abstract

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 inter-pattern distances. A major disadvantage of the original algorithm lies in the fact that it is not easy to map hitherto unseen points. To overcome this problem, several methods have been proposed. In this paper, we aim to compare some approaches to implement this mapping on a neural network. © 1997 Elsevier Science B.V.

Keywords: Projection methods; Sammon's mapping; Feed-forward neural networks; SAMANN

1. Introduction

Sammon's mapping (Sammon Jr, 1969) is a useful tool in pattern recognition practice. It is an algorithm for finding a mapping of a dataset of dimensionality d onto a non-linear subspace of m dimensions (where $m < d$), preserving as well as possible the inter-pattern distances. The algorithm is often used to visualize high-dimensional data in two or three dimensions but can be used to map a dataset to any low-dimensional space, i.e. the output is not restricted to be two dimensional. The distance-preserving aspect can be of importance when one wants to use classifiers sensitive to these distances, such as the nearest-neighbour classifiers.

A disadvantage of the original Sammon mapping algorithm is that it, unlike for example Principal Component Analysis (PCA), does not yield a mathematical or algorithmic mapping procedure for previously unseen data points. That is, when a new point has to be mapped, the whole mapping procedure has to be repeated. A proposal by Mao and Jain (1995) to solve this problem is SAMANN, a feed-forward neural network with a specialized, unsupervised learning rule to learn Sammon's mapping. In this paper we test this technique and compare it with two other generalization techniques for Sammon's mapping: triangulation and a normal feed-forward ANN trained with back-propagation. We also investigate the influence of pre-training SAMANN with a PCA mapping and the effect of the choice of output transfer function. The goal is to investigate mapping qualities only; no classifiers or other mechanisms are used to quantify performance. We will show that SAMANN performs well but no better than an ordinary ANN.

* Corresponding author.

PII S0167-8655(97)00093-7

2. Sammon's mapping and neural networks

2.1. Sammon's mapping

Sammon's mapping is a technique for transforming a dataset from a high-dimensional space (say, d-dimensional) to a space with lower dimensionality m. Many criteria can be optimized in this process, such as class separability in classification tasks (in Fisher's linear discriminant analysis) or the amount of variance retained in the mapped set (in PCA). Sammon's mapping tries to preserve inter-pattern distances. This is achieved by minimizing an error criterion which penalizes differences in distances between points in the original space and the mapped space. If we denote the distance between two points $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$, $i \neq j$, in the original space by d_{ij} and the distance between $\boldsymbol{x}'_i$ and $\boldsymbol{x}'_j$ in the mapped space by d'_{ij}, then Sammon's stress measure E is defined as

$$E = \frac{1}{\sum_{i=1}^{n-1}\sum_{j=i+1}^{n} d_{ij}} \sum_{i=1}^{n-1}\sum_{j=i+1}^{n} \frac{\left(d_{ij} - d'_{ij}\right)^2}{d_{ij}}. \qquad (1)$$

Most often, the distance measure $d_{ij} = D(\boldsymbol{x}_i, \boldsymbol{x}_j)$ used is the Euclidean distance, $\|\boldsymbol{x}_i - \boldsymbol{x}_j\|$. Note that all pattern distances are weighed equally due to the division by the original pattern distance d_{ij}. Also, E is insensitive to scaling, since only relative distance differences are used.

This error measure can be minimized using any minimization technique. In (Sammon Jr, 1969) a technique is proposed which is known as pseudo-Newton minimization (Becker and Le Cun, 1989), but which is referred to as steepest descent in other publications, e.g. in (Kohonen, 1995),

$$x'_{i_k}(t+1) = x'_{i_k}(t) - \alpha \frac{\partial E(t)/\partial x'_{i_k}(t)}{|\partial^2 E(t)/\partial x'_{i_k}(t)^2|}, \qquad (2)$$

where x'_{i_k} is the kth coordinate of the position of point $\boldsymbol{x}'_i$ in the mapped space.

A problem with the update rule given in Eq. (2) is the division by the second derivative. This causes problems at inflection points (areas of small curvature) where the second derivative is very small (de Ridder, 1996). Although in (Kohonen, 1995) an optimal setting of 0.3–0.4 for α is given, there is no reason to expect this range to be optimal for all problems. However, one can also use other minimization techniques, which do not possess this problem, such as normal gradient descent,

$$x'_{i_k}(t+1) = x'_{i_k}(t) - \alpha \frac{\partial E(t)}{\partial x'_{i_k}(t)}. \qquad (3)$$

2.2. Triangulation

As explained in the Introduction, once a dataset is mapped, Sammon's algorithm does not give a way in which to map new points without recalculating the entire map. One very intuitive way of solving this is to use the dataset neighbourhood relations in the original set. This method is known as triangulation (Lee et al., 1977; Biswas et al., 1981). The two nearest neighbours in the original space are found. Given the distances to these neighbours, one can find a mapping in which these distances are preserved exactly. If there is more than one point fulfilling the requirements, a third nearest neighbour is used to decide which of these points is chosen (see Fig. 1).

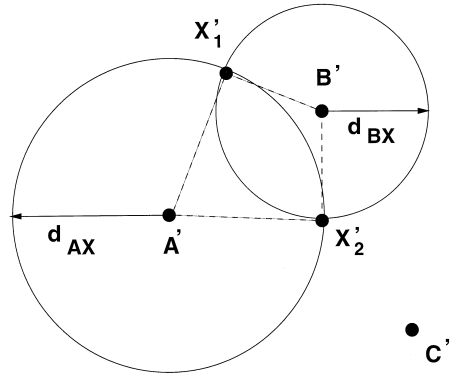

Fig. 1. Triangulation: to find the mapping X' of a point X, the distances to points A and B in the original space, d_{AX} and d_{BX}, will have to be preserved. This gives a desired value for the distances between candidate points X' and the mappings of A and B, A' and B'. The requirements can result in no candidate points (if the circles do not overlap), one candidate point (if the circles touch) or two candidate points. In the first case a simple linear interpolation weighed with d_{AB} and d_{BD} is used; in the latter case a third point C decides which point is chosen, as indicated in this figure.

2.3. SAMANN

Another approach to overcome the generalization problem of the original Sammon's mapping is the use of an artificial neural network (ANN) to interpolate and extrapolate the mapping. In (Mao and Jain, 1995), a specific back-propagation-like learning rule is developed to allow a normal feed-forward ANN to learn Sammon's mapping in an unsupervised way, called SAMANN. In each learning step, the ANN is shown two points. The outputs of each neuron are stored for both points. The distance between the ANN 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.

A drawback of using SAMANN is that the original dataset will have to be scaled for the ANN to be able to find a correct mapping, since the ANN can only map to points in the sigmoid's output interval, $\langle 0,1 \rangle$. This scaling is dependent on the maximum distance in the original dataset. It is therefore possible that a new pattern shown to the ANN will be mapped incorrectly, when its distance to a pattern in the original dataset is larger than any of the original inter-pattern distances. This makes testing the mapping qualities of SAMANN using a test set difficult: learning set and test set would have to be scaled using the same factor, based on the inter-pattern distances of the combined set. Clearly, this is non-typical of pattern recognition practice. Another approach to overcome the problem could be to use linear output units, to allow SAMANN to perform any $\mathbb{R}^d \rightarrow \mathbb{R}^m$ mapping.

2.4. Standard feed-forward ANN

As an alternative to SAMANN's unsupervised learning rule, one could also train a standard feed-forward ANN using supervised back-propagation on a previously calculated Sammon's mapping. Although this has a higher computational demand since it involves two learning phases (one for Sammon's mapping, one for the ANN) it should perform at least as well as SAMANN.

3. Experiments

3.1. A comparison

Although SAMANN has been applied to a large number of datasets and compared to several other mapping techniques (Mao and Jain, 1995), there has been no comparison between the SAMANN approach and the original Sammon mapping. An important question is how SAMANN performs compared to Sammon's mapping in conjunction with triangulation, or compared to a standard feed-forward ANN.

To judge the different approaches, in our experiments we used the following.

- Triangulation on Sammon's mapping;
- SAMANN with sigmoid output units, trained on a scaled dataset;
- SAMANN with linear output units, trained on the original dataset;
- an identical ANN with sigmoid output units, trained using back-propagation on a scaled Sammon's mapping;
- an identical ANN with linear output units, trained using back-propagation on Sammon's mapping.

All techniques were tested using distinct learning and tests sets. Furthermore, for ANNs an independent validation set was used to stop training. All ANNs had one hidden layer of 20 units and an output layer of two units, i.e. all mappings were onto two dimensions.

Since here we are just interested in the capabilities of these methods to perform and extend Sammon's mapping, the only error measure we use is Sammon's stress as given in Eq. (1). Often other measures are used, such as the performance of some classifier trained on the mapped data (see, for example, (Lerner et al., 1996; Mao and Jain, 1995; Kraaijveld et al., 1995)); however, these measure quantities do not necessarily say anything about the quality of the methods in terms of mapping performance.

3.2. Datasets

We used eight datasets in our experiments; they are listed in Table 1. Note that, although the set sizes themselves are rather small, they contain $\frac{1}{2}n(n-1)$ inter-pattern distances for n points.

Most of the sets are artificial. Two sets, *Hyper-*

Table 1
An overview of the datasets used in the experiments. Note that, for the *Hypercube* and *Sphere* datasets, the test set is on purpose chosen from another part of the distribution than the learning set. For the *Hypercube* set, the training set is used as a validation set

Name	Description	Source	Dimension	Learning set size	Validation set size	Test set size
Hypercube	Vertices (learning set) and half-way edge points (test set) of a hypercube	Artificial	4	32	-	16
Sphere	Two caps ($\|z\| > 0.5$: learning set) and points from the remaining area ($\|z\| \leqslant 0.5$: test set) of a unit sphere	Artificial	3	100	100	100
Curves	Two elongated clusters as described in (Kraaijveld et al., 1995); intrinsically two-dimensional (see Fig. 2)	Artificial	3	100	100	100
Gauss	Two normally distributed clusters: $\Sigma_1 = \Sigma_2 = I$, $\mu_1 = \mu_2 = (1,1,\ldots,1)^T$	Artificial	3	100	100	100
Random	Uniformly distributed data in a unit hypercube	Artificial	10	100	100	100
Iris	Measurements on 3 classes of flowers (Fisher, 1936)	Real	4	100	25	25
Chromosomes	Chromosome banding profiles from 24 classes, a subset of the routine dataset described in (Lundsteen et al., 1986)	Real	30	120	120	120
Digits	Handwritten digits (10 classes) in a 16×16 grid, range $[-1,1]$ from the NIST database (Wilson and Garris, 1990; de Ridder, 1996)	Real	256	100	100	100

cube and *Sphere*, serve a special purpose: to investigate the interpolation qualities of the mapping mechanisms. In both these cases, the test set is taken from a different area of the distribution than the learning set. In the *Hypercube* set, test points lie on the edges of the hypercube, always on a straight line between vertices. We would expect a linear interpolation routine to perform quite well here. In the *Sphere* set, in which the test set is chosen from the equator of a sphere and the learning set from the two remaining caps, the interpolation should be non-linear. This is illustrated in Fig. 2(a).

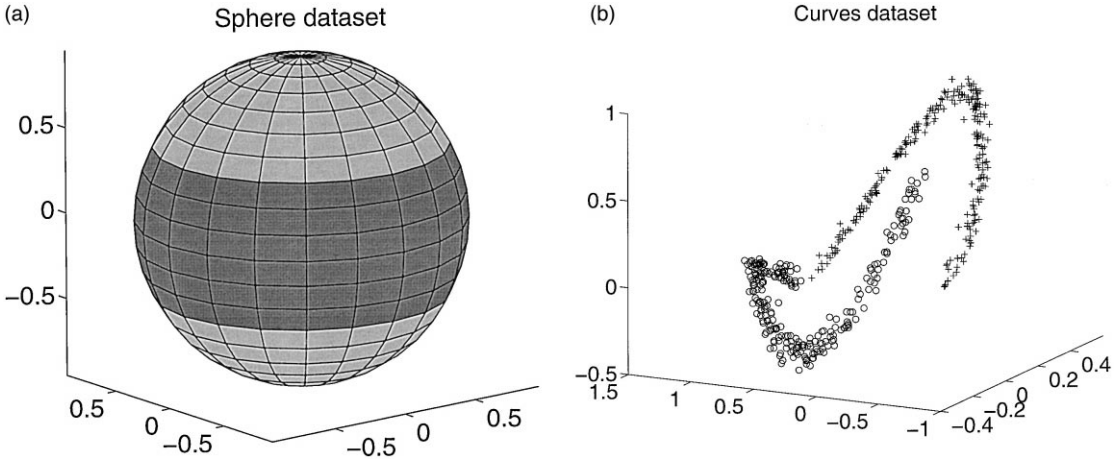

Fig. 2. The *Sphere* (a) and *Curves* (b) datasets. For the *Sphere* dataset, the learning set is taken from a uniform distribution on the caps (light grey), the test set from the equatorial band (dark grey).

3.3. Training

Both the ANNs and the original Sammon's mapping were initialized with random values from a uniform distribution with range $[-0.01,0.01]$. We trained all ANNs using the conjugate gradient descent method. The learning rate was set to 0.5 for the ANNs having sigmoid outputs and to 0.05 for ANNs with linear output units. No momentum was used. All ANNs were trained for 1 000 000 cycles or until the stress on the validation set began to rise, whichever came first. Results are given in Table 2.

The traditional Sammon's mapping was trained with normal gradient descent (Eq. (3)), with a learning rate α of 0.5, and with the pseudo-Newton method (Eq. (2)) with $\alpha = 0.05$. In several cases, the latter method did not converge even with this modest setting of the learning rate. Therefore, for training the normal ANNs, Sammon's mapping calculated with the gradient descent method was used.

We also tried pretraining SAMANNs with Principal Component Analysis (PCA), as suggested in (Mao and Jain, 1995). To this end, the SAMANN was first trained for 20 000 cycles (using the same learning parameters) by normal supervised backpropagation with, for each sample, its PCA mapping onto two dimensions as targets.

Another proposed initialization consists of setting the weights between the input layer and the first hidden layer of SAMANN to the h eigenvectors corresponding to the h largest eigenvalues of the PCA (where h is the number of hidden units) as proposed in (Lerner et al., 1996). This initialization will only work in cases where the number of hidden units is less than or equal to the number of input units, imposing a serious limitation on the networks flexibility. We have not used this initialization mechanism.

4. Discussion

4.1. Convergence

In our experience, the SAMANNs were rather hard to train and converged slowly when compared to normal ANNs. This can be understood from the fact that one cycle means showing one pair of samples. Therefore, for a 100 sample dataset, it takes on average 5000 (i.e. $\frac{1}{2}n(n-1)$) cycles for SAMANN

Table 2
Residual stress of various mapping methods on a number of datasets. E_L denotes the residual stress on the learning set, E_T on the test set. 'Sammon' means the original Sammon's mapping; 'sigmoid' and 'linear' indicate the transfer function in the output layer of the ANN used

	Dataset							
	Hypercube		Sphere		Curves		Gauss	
Method	E_L	E_T	E_L	E_T	E_L	E_T	E_L	E_T
Sammon, triangulation	0.0952	0.1674	0.0254	0.2635	0.0001	0.0056	0.0607	0.1833
SAMANN, sigmoid	0.1005	0.1123	0.0596	0.0489	0.0050	0.0053	0.1581	0.1785
SAMANN, linear	0.2679	0.2288	0.4485	0.6403	0.4229	0.4190	0.8645	0.8697
SAMANN/PCA, sigmoid	0.1023	0.1131	0.0364	0.1054	0.0015	0.0015	0.0850	0.1004
SAMANN/PCA, linear	0.2598	0.2234	0.3426	0.4409	0.1652	0.1620	0.7766	0.7958
Sammon, ANN, sigmoid	0.0952	0.1181	0.0268	0.1224	0.0002	0.0002	0.0641	0.1224
Sammon, ANN, linear	0.0952	0.1081	0.0256	0.1234	0.0002	0.0002	0.0611	0.1342
	Random		Iris		Chromosomes		Digits	
Method	E_L	E_T	E_L	E_T	E_L	E_T	E_L	E_T
Sammon, triangulation	0.1176	0.2286	0.0078	0.0340	0.0680	0.1695	0.1307	0.2602
SAMANN, sigmoid	0.1424	0.1740	0.0359	0.0373	0.1144	0.1096	0.1451	0.1635
SAMANN, linear	0.8154	0.8264	0.9790	0.9820	0.8852	0.8854	0.9699	0.9646
SAMANN/PCA, sigmoid	0.1361	0.1940	0.0517	0.0429	0.2864	0.2915	0.1379	0.1437
SAMANN/PCA, linear	0.4953	0.6881	0.5718	0.6025	0.5069	0.5035	0.9075	0.9111
Sammon, ANN, sigmoid	0.1308	0.1817	0.0103	0.0184	0.0703	0.0820	0.1379	0.1673
Sammon, ANN, linear	0.1305	0.2006	0.0141	0.0303	0.0860	0.0970	0.1884	0.1915

to see the entire set of inter-pattern distances once. SAMANNs with linear outputs did not train well at all. Although they showed convergence, it was extremely slow; the results shown are all reached after 1 000 000 cycles.

There is no reason to expect SAMANN to train

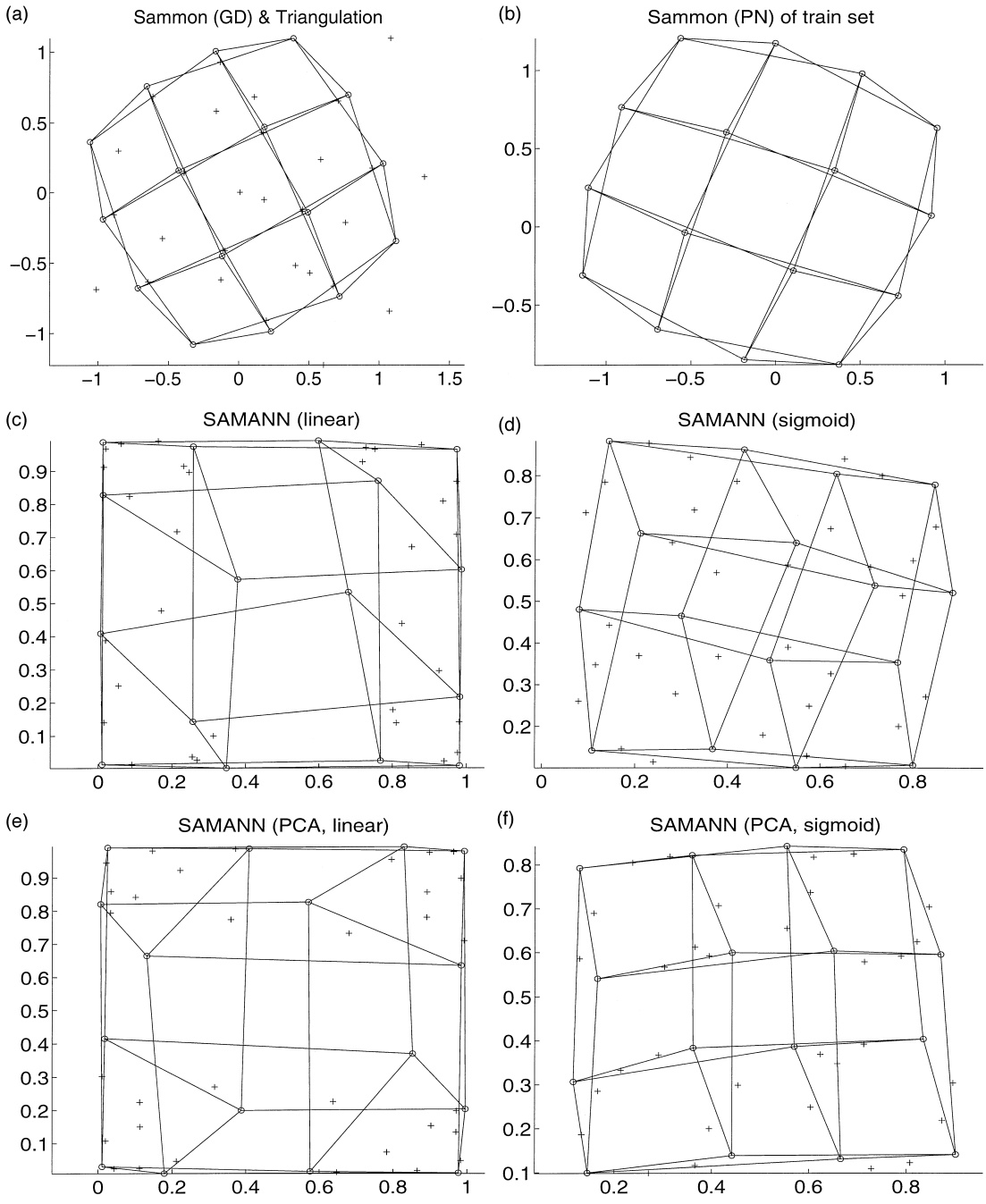

Fig. 3. Maps of the *Hypercube* dataset, generated by different methods. The training set is indicated by '*o*', the test set by '∗'.

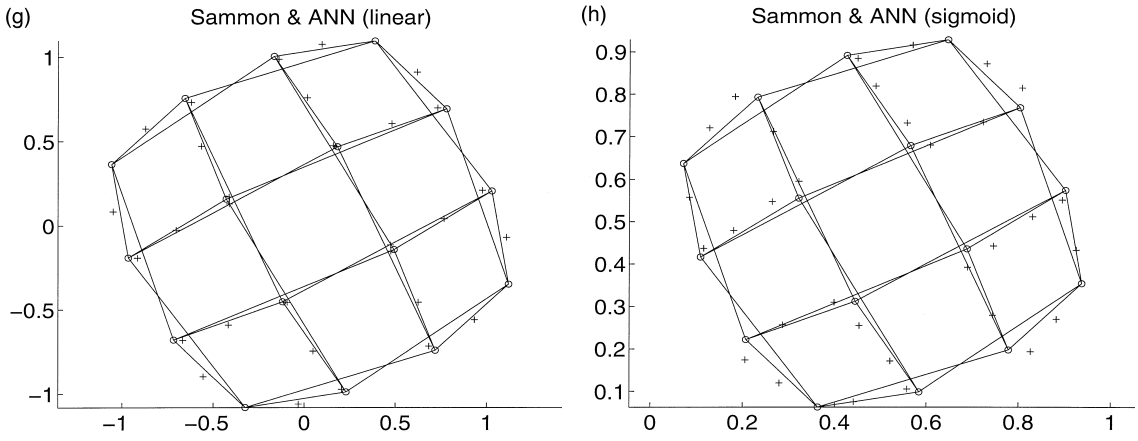

Fig. 3 (continued).

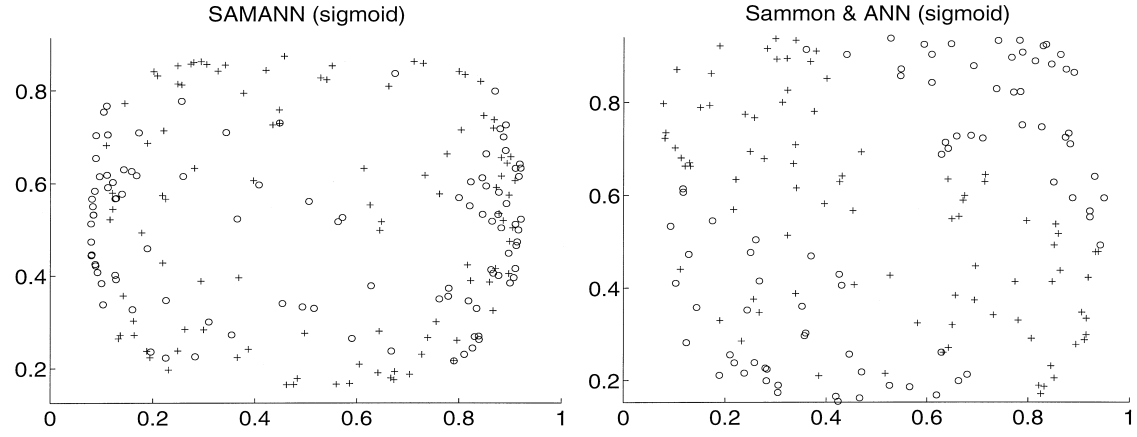

Fig. 4. Maps of the *Sphere* dataset, generated by different methods. The training set is indicated by '*o*', the test set by '∗'.

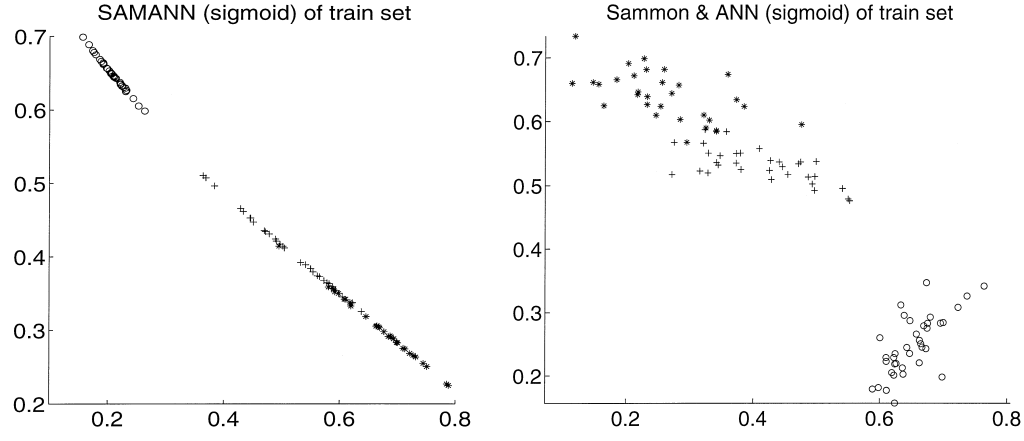

Fig. 5. Maps of the *Iris* dataset, generated by different methods. SAMANN seems to be stuck in a local minimum. Classes: '*o*' is *Iris setosa*, '+' is *Iris versicolor* and '∗' is *Iris virginica*.

any faster than traditional Sammon's mapping, as in both cases the same error criterion is minimized, the only difference being the number of parameters in the model. If SAMANN shows a speed improvement, it is only because of the fact that the method uses fewer than the $m \times n$ parameters Sammon's mapping uses for mapping n points to m-dimensions, which diminishes the mapping power.

Another way to speed up Sammon's mapping may be to train a normal feed-forward ANN on a small but representative subset of the data to be mapped, and mapping the rest of the data using the trained ANN (the so-called *frame method*). This allows the network to be more complicated than SAMANN, since the time complexity depends quadratically on the number of points in the set to be mapped, but only linearly on the number of weights in the ANN.

The ANNs trained on Sammon's mapping data (calculated using normal gradient descent) trained quite well, both with sigmoid and linear output units.

4.2. Initialization

Pretraining on a PCA mapping did not improve performance in all cases; although it speeded up the training process on most datasets, we found that pretraining can get SAMANN stuck in bad local minima. This happened for the case of the *Iris* and *Chromosomes* data and, to a lesser extent, for the *Hypercube* set.

4.3. Mapping differences

Figs. 3–5 illustrate the fact that the different methods can give quite different results. This is due to the different underlying minimization mechanisms. For example, the pseudo-Newton method may take large steps through the error space when the second derivative is small, reaching an entirely different region and therefore resulting in a different solution. The SAMANN method updates ''per sample'' and will therefore most likely take a different path than Sammon's mapping.

4.4. Mapping performance and generalization

In general, the linear output SAMANNs do not perform very well since training was stopped, before the error converged, after 1 000 000 cycles. All other ANNs converged quite well.

Triangulation is the next worst method. It only performs well in situations where the mapping of neighbouring points does say something about the local mapping. This is for example the case in the *Hypercube* dataset, where all test points lie on a line between two learning points. In other cases, such as the sparse *Digit* dataset, performance is poor.

SAMANNs with sigmoid output units show somewhat better generalization properties than the ANNs trained on Sammon's mapping, since the relative difference between the error on the learning and the error on the test set is smaller. However, in nearly all cases (except *Random*, *Digits* and especially *Sphere*) a normal ANN trained on Sammon's mapping outperforms SAMANN. The reason for the difference on *Sphere* can be seen in Fig. 4: SAMANN has (suboptimally) mapped the training set onto the xy plane while Sammon's mapping has mapped the dataset onto the xz plane (for the distribution of the original data, see Fig. 2). The first situation makes it easier to map the test data than the latter.

5. Conclusions

Although SAMANN is a step forward compared to traditional Sammon's mapping, there is a problem: the need to scale the dataset. This can degrade performance on hitherto unknown samples, since they will have to be scaled too with a possibly suboptimal factor. If we use linear output units to avoid this problem, training does not converge well. Initialization based on the PCA mapping does not solve the latter problem.

When a standard feed-forward ANN is trained on Sammon's mapping using back-propagation, convergence is fast and performance seems to be somewhat better than that of SAMANN. Also, ANNs with linear transfer functions train well, circumventing the scaling problem.

In terms of computing requirements, it is our experience that SAMANN needs a large number of training cycles before converging. Although Sammon's mapping is computationally very demanding, training SAMANN is too. This is to be expected,

since both methods will have to be shown a large number of inter-pattern distances.

In conclusion, it would seem that SAMANNs unsupervised learning rule is not necessarily the best choice for training an ANN on Sammon's mapping. The only advantage over original Sammon's mapping seems to be the easy way to restrict the number of parameters used (by choosing the network layout), thereby reducing the time needed for the training process.

Discussion

Mao: I have a comment on the convergence problem. I found that the reason why the convergence of the Sammon mapping is very slow is because you have a nonlinear function. I found that for most data sets Sammon's mapping configurations are similar to the principal component analysis maps. That means that linear components are very important. But the standard network is a highly nonlinear function. So it is very inefficient to approximate linear components using a nonlinear architecture. I have some idea to overcome this problem by introducing connections between the input and output layer.

De Ridder: So you can implement linear functions directly?

Mao: That would make the network faster. But I haven't done the experiment.

Raghavan: I first have a question and then later a comment. I wanted to ask you about the stress function. Why is that particular function being used, and are there other, similar functions that could be used?

De Ridder: It is the most natural stress measure that you can think of, at least if you want to retain all distances as well as possible. You could take out the normalization factor. Normalization is just to ensure that the measure is between one and zero. There are other measures; I believe Andrew Webb has written a paper last year (Proc. 13-th ICPR, D-635, 1996) in which he used all kinds of measures, for example with exponential weighing, so that small distances have larger weights than larger distances. They seem to be useful, but each of them in specific applications.

Raghavan: My comment is just to relate this work to the work that my co-author Choubey presented yesterday. The work that we did was in the context of image retrieval, based on work done a number of years ago by Lev Goldfarb of the University of New Brunswick, Canada. Of course the particular method is not related to the use of neural networks, but it still has the idea of mapping from original space to a new space and in that context, for retrieval purposes we used this evaluation measure called R-norm, short for normalized recall. Other people who are planning work in this direction might be interested in looking at that measure. I also think that Goldfarb's approach to get this kind of mapping is very powerful.

Acknowledgements

This research is partly supported by the Foundation for Computer Science in the Netherlands (SION) and the Dutch Organization for Scientific Research (NWO).

References

Becker, S., Le Cun, Y., 1989. Improving the convergence of back-propagation learning with second order methods. In: Touretzky, D., Hinton, G., Sejnowski, T. (Eds.), Proc. of the 1988 Connectionists Models Summer School, Carnegie-Mellon University. Morgan Kauffmann, Los Altos, CA.

Biswas, G., Jain, A.K., Dubes, R.C., 1981. Evaluation of projection algorithms. IEEE Trans. Pattern Anal. Machine Intell. 3 (6), 701–708.

Fisher, R.A., 1936. The use of multiple measurements in taxonomic problems. Ann. Eugen. 7, 178–188.

Kohonen, T., 1995. Self-organizing maps. Springer Series in Information Sciences. Springer, Berlin.

Kraaijveld, M.A., Mao, J., Jain, A.K., 1995. A nonlinear projection method based on Kohonen's topology preserving maps. IEEE Trans. Neural Networks 6 (3), 548–559.

Lee, R.C.T., Slagle, J.R., Blum, H., 1977. A triangulation method for the sequential mapping of points from n-space to two-space. IEEE Transactions on Computers C-27, 288–292.

Lerner, B., Guterman, H., Aladjem, M., Dinstein, I., Romem, Y., 1996. Feature extraction by neural network nonlinear mapping for pattern classification. In: Proc. ICPR '96, vol. D. IEEE, pp. 320–324.

Lundsteen, C., Gerdes, T., Maahr, J., 1986. Automatic classification of chromosomes as part of a routine system for clinical analysis. Cytometry 7, 1–7.

Mao, J., Jain, A.K., 1995. Artificial neural networks for feature extraction and multivariate data projection. IEEE Trans. Neural Networks 6 (2), 296–317.

Ridder, D. de, 1996. Shared weights neural networks in image analysis. Master's Thesis. Delft University of Technology.

Sammon Jr, J.W., 1969. A nonlinear mapping for data structure analysis. IEEE Trans. Comput. 18, 401–409.

Wilson, C.L., Garris, M.D., 1990. Handprinted character database 2. National Institute of Standards and Technology; Advanced Systems division.