Neural networks for estimating intrinsic dimension

A. Potapov* and M. K. Ali†
Department of Physics, The University of Lethbridge, 4401 University Dr. W. Lethbridge, Alberta, Canada T1K 3M4

(Received 4 December 2001; published 2 April 2002)

We consider the problem of feature extraction and determination of intrinsic dimensionality of observation data. One of the common approaches to this problem is to use autoassociative neural networks with a ‘‘bottleneck’’ projecting layer. We propose a different approach in which a neural network performs a topological mapping that creates a nonlinear lower-dimensional projection of the data. The mapping preserves relative distances of neighbors. This technique can be efficiently implemented with the help of radial basis function networks, and it is significantly faster than training an autoassotiative network. We show that the proposed technique can be used for estimating the dimension of minimal mathematical model from time series data.

DOI: 10.1103/PhysRevE.65.046212 PACS number(s): 05.45.−a, 84.35.+i, 87.18.Sn

I. INTRODUCTION

Analysis of observational or sensory data is very important in applications such as pattern recognition, diagnostics, prediction, and control. Usually, the complexity of the task to be accomplished depends on the properties of the data. For example, if one needs to obtain an approximation of an unknown function, the number of parameters to be estimated depends essentially on the dimensionality of the data. The greater the number of fitting parameters, the more complex becomes the task of optimizing their estimates. For nonlinear approximations with tools such as multilayer perceptrons, increase of the number of parameters typically adds to optimization problems because of rugged landscapes and spurious minima in the parameter space.

For this reason, many data processing tasks include preprocessing and feature extraction as a necessary step [1]. Sometimes this enables one to simplify the problem and to reduce the dimensionality of the input data by extracting the most essential components. For example, if we are processing vectors $\mathbf{z}_i \in R^m$, $i=1,\ldots,N$, and they in fact depend only on $d<m$ parameters $\mathbf{y}$, that is, $\mathbf{z}_i=\mathbf{z}(\mathbf{y}_i)$, $\mathbf{y}_i \in R^d$, then it is preferable to process the vectors $\mathbf{y}$ instead of $\mathbf{z}$. Note that there is no need for precise determination of the vectors $\mathbf{y}$ themselves, it is adequate to find the mapping $R^m \rightarrow R^d$:$\mathbf{u}=F(\mathbf{z})=F(\mathbf{z}(\mathbf{y}))$, such that $\mathbf{u}$ is equivalent to $\mathbf{y}$, that is, the mapping $\mathbf{y}\rightarrow\mathbf{u}$ is one-to-one, differentiable and invertible (y and z are diffeomorphic). The transformation $\mathbf{z}\rightarrow\mathbf{y}$ can be considered as a way to *encode* or *pack* data without loss of essential information.

If $\mathbf{z}$ depends on $\mathbf{y}$ linearly, that is, $\mathbf{z}=A\mathbf{y}$ where A is an $m\times d$ matrix, then the essential dimensions of $\mathbf{z}$ can be found with the help of the well-known principal component analysis (PCA). It is easy to show that the necessary directions are ones that correspond to d eigenvectors of the $m\times m$ matrix $N^{-1}\Sigma_i \mathbf{z}_i\mathbf{z}_i^T$ with the largest (nonzero) eigenvalues. Even if the relationship between y and z is more complex, PCA can still be very useful: if the surface $\mathbf{z}(\mathbf{y})$ can be projected one-to-one to a d-dimensional hyperplane, sometimes PCA can find such a plane. Also, PCA can reduce noise, and for this reason it is widely used in statistical applications.

For general nonlinear dependence of $\mathbf{z}$ on $\mathbf{y}$, PCA is usually of little help. In such cases, an idea of ‘‘bottleneck’’ autoassociative neural networks [1,2] can be helpful. Such a network uses an m-dimensional vector $\mathbf{z}$ both as input and output. However, the internal structure of the network includes one of the hidden layers with only d neurons. Thus the first half of the network works as a projection mapping $R^m \rightarrow R^d$, while the remaining part performs the restoration mapping $R^d \rightarrow R^m$. By building a number of such networks with different d and analyzing their errors in restoring $\mathbf{z}$ after projection, it is possible to find the intrinsic dimensionality of the data. It has been shown that if the neurons of such a network perform only a linear transformation, then this procedure coincides with PCA [1].

There are numerous problems in which it is very useful to have a knowledge of the intrinsic dimension. We present here two examples. The first one is related to building a control neural network that must analyze input sensory data and decide which control action must be taken. The learning efficiency of the network depends on the number of connections to be adjusted in the learning process. Preprocessing initial data with projecting subnetwork can substantially reduce the number of connections, which in turn simplifies learning and increases its efficiency. Moreover, if the number of essential parameters is small enough, then the structure of the projected d-dimensional sensory space may provide additional useful information, see Ref. [3] for experiments on controlling small robots.

Another example is related to nonlinear time series analysis. There is a class of techniques for system identification and prediction based upon the theory of dynamical systems and the procedure of delay reconstruction from a scalar or vector time series. The main hypothesis is that the observation data $x_i=x(t_i)$ are generated by a dynamical system $\dot{\mathbf{x}}=\mathbf{f}(\mathbf{x})$ or

$$\mathbf{x}(t+\tau)=\varphi^{\tau}(\mathbf{x}(t)), \tag{1}$$

and x_i are the values of an observable, $x_i=h(\mathbf{x}(t_i))$. One can use the Takens theorem [4]: if $\mathbf{x}$ belongs to a d-dimensional

*Email address: alexei.potapov@uleth.ca
†Email address: ali@uleth.ca

manifold M^d and φ and h have at least two continuous derivatives (the theorem has also a few additional assumptions that are typically satisfied), then for the mapping

$$\mathbf{x}(t)\to\mathbf{z}(t)\equiv\Lambda(\mathbf{x}) = \{x(t),x(t+\tau),\ldots,x[t+(m-1)\tau]\}\in R^m \quad (2)$$

the generic property is that it gives an embedding of M^d into R^m provided $m\geqslant 2d+1$. This means that typically in R^m there exists a d-dimensional surface $S^d=\Lambda(M^d)$ that is diffeomorphic to M^d, all vectors $\mathbf{z}\in S^d$ and there is one-to-one correspondence between x and z. Therefore, there exists an analog of Eq. (1) in z representation,

$$\mathbf{z}(t+\tau)=\Psi[\mathbf{z}(t)]=\Lambda[\varphi^{\tau}(\Lambda^{-1}(\mathbf{z}(t)))],\quad \mathbf{z}\in S^d\subset R^m. \quad (3)$$

The systems (1) and (3) can be regarded as a nondegenerate and invertible change of variable $\mathbf{x}\leftrightarrow\mathbf{z}$. Therefore, from the set of vectors $\mathbf{z}$ one can study the properties of attractor of Eq. (1) that are invariant under change of variables.

Based on this theorem a class of methods for system identification and prediction has been developed, see, e.g., reviews [6,5,7]. There are methods for estimating fractal dimensions d_F (which gives the lower bound for d), Lyapunov exponents, metric entropies, and time series prediction. Some methods do not use the dimension d explicitly (e.g., estimation of correlation exponent), while others essentially use it (e.g., matrix methods for estimating Lyapunov exponents and some prediction techniques).

In the context of dynamical systems, the parameter d has drawn attention of several authors. As a rule, *d is not equal to the dimension of the phase space* of Eq. (1): a system of 100 equations can have a limit cycle as its attractor with $d=1$ and not 100. The entity d has several names. For example, it may be called ‘‘local intrinsic dimension’’ [8] or ‘‘dynamical dimension’’ [5]. From mathematical point of view, it is related to the concept of *inertial manifold* developed in the mid 1980s [9]. Often an attractor of a dynamical system does not span all dimensions of the phase space, it may belong to a surface (manifold) of lower dimension. If such a manifold containing the attractor is invariant, differentiable and exponentially attracting all trajectories close to it, then it is called inertial manifold. We shall call an inertial manifold of the least possible dimension as a minimal inertial manifold (MIM). Its dimension d_I is the parameter that was discussed above, and the MIM can be considered as the manifold M^d used in the Takens theorem. The existence of MIM means that on attractor the dynamical system in fact depends only on d_I essential variables, other modes being ‘‘enslaved.’’ In fact, the theory of inertial manifolds gives precise formulation of the well-known Haken's slaving principle in synergetics [10]. So, in the context of dynamical systems, analysis of intrinsic dimension of observation data can give very important information about the underlying dynamical system.

A number of approaches for the determination of d_I (with different names for d_I) are available in the literature. For example, rational polynomial approximations are used in Ref. [11], local singular value decomposition is used in Refs. [12,8], and the technique of local false nearest neighbors is used in Refs. [13,5]. All of these approaches have worked for model systems, but they require rather large data sets, e.g., about 2×10^4 points for the Lorenz attractor. In this paper we present a technique that is reliable for small as well as large data sets, although it generally requires a greater amount of computation.

The purpose of this paper is to consider different neural network techniques for estimating the intrinsic dimension, which we shall denote d_I from numerical data. As our experiments show, the autoassociative multilayer perceptrons are not efficient enough from computational point of view when the dimension of the input data $m\sim10$ and the number of vectors to be processed $N\sim10^3$. Even on very fast machines, learning of a network takes too long. In contrast, it is more efficient to use radial basis function (RBF) networks and the concept of topological mapping. If we do not lose information under projection mapping, then topology, that is, the structure of neighborhood, before and after mapping should be the same. Note that there exists one-to-one mapping, for example, between one-dimensional unit segment and a two-dimensional unit square, but such mappings are not continuous and they do not preserve the neighborhood of points. Points that are neighbors in the square may not be neighbors on the line and vice versa. So we try to construct a continuous mapping $R^m\to R^d$ such that the distances between new d-dimensional $\mathbf{y}$ vectors are approximately the same as that for the original $\mathbf{z}$ vectors. In contrast to autoassotiative networks, topological RBF networks can learn several orders of magnitude faster.

A simple but common way to check the preservation of the neighborhood structure is to compare distances between points in the original space with distances between their images after a mapping: near neighbors must remain near neighbors, while remote neighbors should remain remote neighbors. This is the basis for a number of techniques in nonlinear time series analysis in which a reconstruction of lower dimension is often used instead of projection mapping. Examples include the false nearest neighbor method for estimating optimal embedding dimension and d_I [5,13], the technique for estimating optimal embedding dimension and time delay [14], and some other works. We also apply a technique based on comparison of distances, but we construct a mapping that gives an optimal value of a certain ‘‘topological’’ functional.

The structure of the paper is the following. In Sec. II, we consider various types of autoassociative networks and their shortcomings. In Sec. III, we describe the topological mapping, the algorithm for its fast numerical implementation, and in Sec. IV the numerical examples.

II. AUTOASSOCIATIVE NETWORKS AND THEIR VARIOUS IMPLEMENTATIONS

At present, neural networks are widely used for approximation of unknown dependencies. Like any other approximation technique, these networks use parameter fitting for a selected class of functions. There are two major types of approximation networks, namely, multilayer perceptrons and radial basis function networks [1].

Multilayer perceptrons consist of a number of layers of nonlinear elements, the neurons. The state of the neuron i in the layer j is denoted by $x_{i,j}$. Each neuron $x_{i,j}$ receives signals from neurons in the previous layer $x_{k,j-1}$ through the connections with weights $w_{i,k}$, and then performs a nonlinear transformation σ giving its present value. In other words, each layer performs the mapping

$$x_{ij} = \sigma\left(\sum_{k=1}^{n_{j-1}} w_{i,k} x_{k,j-1} + w_{i,0} \right),$$

where n_j is the number of neurons in the layer j, and $w_{i,0}$ is a bias for the given neuron. In order to simplify the procedure, sometimes a special unit neuron $x_{0,j} \equiv 1$ is added to each layer. This way the bias can be considered as a usual connection to this additional neuron. The function σ is usually chosen to be a sigmoid $\sigma(x) = 1/(1+e^{-x})$ or a function similar to it.

The input to the network is fed to the first layer $z_i = x_{i,0}$, and its output becomes the input for the next layer and so on. This way we obtain a feed-forward network architecture. There are also recurrent networks in which backward connections exist, but we shall not consider them in this paper. Let the output of the last layer be y_i. For the sake of brevity we shall denote the whole mapping performed by the network as $\mathbf{y} = G(\mathbf{z},\mathbf{w})$, where $\mathbf{w}$ is the set of weights to be adjusted during learning. Often such networks [i.e., the functions $G(\mathbf{z},\mathbf{w})$] are presented as a diagram, where neurons are shown as circles, and weights w_{ij} as lines connecting them.

Usually, learning of the network is performed by minimizing its errors in the set of the training examples, the known pairs $\{\mathbf{z}_k, \mathbf{y}_k\}$, $k = 1, \ldots, N$. It is necessary to find the minimum of the functional

$$E(\mathbf{w}) = \sum_{k=1}^{N} |\mathbf{y}_k - G(\mathbf{z}_k, \mathbf{w})|^2.$$

The minimization can be done by a number of standard procedures. For evaluation of ∇E there is a fast technique called error backpropagation. The details of numerical implementation can be found, e.g., in Ref. [1].

The autoassociative network, in fact, consists of two parts: the projecting part $\mathbf{y} = G_1(\mathbf{z},\mathbf{w}_1)$ and the restoring part $\mathbf{z} = G_2(\mathbf{y},\mathbf{w}_2)$. The error function has the form

$$E(\mathbf{w}) = \sum_{k=1}^{N} |\mathbf{z}_k - G_2(G_1(\mathbf{z}_k, \mathbf{w}_1), \mathbf{w}_2)|^2.$$

We implemented this technique and tested it with time series from the Lorenz attractor with various values of dimensions $m = \dim z$ and $d = \dim y$, $N = 500$. The technique worked, but we were not satisfied because of the following reasons. (i) The learning was rather slow. Tests with both three-layer G_1 and G_2, even for the conjugate gradient minimization learning, took about 50 000 steps or 20–30 min on a very fast machine (Alpha 667 MHz with speed a little bit higher than 1.9 GHz Pentium 4). Note that Kramer [2] used three-layer networks where only one layer used nonlinear transformation to increase the learning rate. However, to approximate dependencies of a general form, it may be insufficient. (ii) The results often demonstrated essential dependence on initial guess for $\mathbf{w}$. Hence, to make sure of global minimum (or close enough to it), one has to make many similar runs resulting in increase of computation time to several hours per network architecture. (iii) Processing of one time series takes a number of trials with different values of m, d, and probably N. This makes the whole procedure very time consuming.

Another common type of neural network is known as RBF network. Radial basis function networks provide approximations of a different kind. In the space of input N vectors one chooses $M < N$ centers $\mathbf{c}_i$ and searches for an approximation of the form

$$\mathbf{y} = \sum_{i=1}^{M} \mathbf{a}_i \psi(|\mathbf{z} - \mathbf{c}_i|) \equiv R(\mathbf{z},\mathbf{a}).$$

The parameters are selected by minimizing the error function

$$E(\mathbf{a}) = \sum_{k=1}^{N} \left\| \mathbf{y}_k - \sum_{i=1}^{M} \mathbf{a}_i \psi(|\mathbf{z} - \mathbf{c}_i|) \right\|^2.$$

Usually, the fitting is done only by adjusting $\mathbf{a}_i$ with fixed $\mathbf{c}_i$ and ψ. Then the problem reduces to the classical search for minimum of a quadratic function, and $\mathbf{a}_i$ satisfy a linear system of equations. This makes learning a very fast and rather easy task, but the results may essentially depend on the proper choice of $\mathbf{c}_i$ and ψ.

Building an autoassociative network with RBF is not so efficient, because the mapping $\mathbf{z} = R(R(\mathbf{z},\mathbf{a}),\mathbf{b})$ depends linearly on $\mathbf{b}$, but it has a strong nonlinear dependence on $\mathbf{a}$. As a result, the corresponding numerical algorithm is inefficient and calculations also take a very long time. So, as with multilayer perceptrons, this technique seems to be practical only for small data sets. Therefore, we need a faster technique producing similar results. We have developed it on the basis of topological considerations.

III. TOPOLOGICAL MAPPING AND ITS IMPLEMENTATION

As it follows from the above, the basic idea behind the use of autoassociative networks is that there exists a smooth mapping $R^m \to R^d$ such that there is no loss of information about the set of vectors $\mathbf{z}_i$. In particular, this means that topological properties should be preserved, that is, close neighbors should remain close, remote ones should remain remote, and two different points should not be projected into one. In other words, there should be a proportionality $\|\mathbf{y}_i - \mathbf{y}_j\| \sim \|\mathbf{z}_i - \mathbf{z}_j\|$. To use this idea, we need a way to adjust fitting parameters such that this proportionality is guaranteed.

There are at least two rather obvious ways to do so. Let us denote by K the number of pairs $(\mathbf{z}_i, \mathbf{z}_j)$ used for the estimates. Then, if we minimize the function

$$E=K^{-1}\sum_{\{i,j\}}\left(\frac{\|\mathbf{y}_i-\mathbf{y}_j\|^2}{\|\mathbf{z}_i-\mathbf{z}_j\|^2}-1\right)^2, \tag{4}$$

then we should receive a mapping that preserves the distances in an optimal way.

However, Eq. (4) may be very sensitive to situations when $\|\mathbf{y}_i-\mathbf{y}_j\|\gg\|\mathbf{z}_i-\mathbf{z}_j\|$, but not enough sensitive to that when $\|\mathbf{y}_i-\mathbf{y}_j\|\ll\|\mathbf{z}_i-\mathbf{z}_j\|$, and both cases are crucial for obtaining a good mapping $\mathbf{z}\rightarrow\mathbf{y}$. For this reason we used another form of topological functional,

$$E=K^{-1}\sum_{\{i,j\}}\left(\frac{\|\mathbf{y}_i-\mathbf{y}_j\|^2}{\|\mathbf{z}_i-\mathbf{z}_j\|^2}+\frac{\|\mathbf{z}_i-\mathbf{z}_j\|^2}{\|\mathbf{y}_i-\mathbf{y}_j\|^2}\right), \tag{5}$$

which has some features common with the one used in Ref. [14]. The theoretical minimum $E=2$ of Eq. (5) is achieved when $\|\mathbf{y}_i-\mathbf{y}_j\|=\|\mathbf{z}_i-\mathbf{z}_j\|$. So, numerical algorithm for obtaining a topological mapping is as follows: we choose a neural network architecture giving the mapping $\mathbf{y}=G(\mathbf{z},\mathbf{w})$ or $\mathbf{y}=R(\mathbf{z},\mathbf{a})$, and the training set of pairs $\{i,j\}$, and then adjust the weights of the network to minimize E. In the numerical results presented below we used the RBF approximation and minimized the following functional:

$$\begin{aligned}S(d)&=\min_{\mathbf{a}}E(\mathbf{a},d)\\&=\min_{\mathbf{a}}K^{-1}\sum_{\{i,j\}}\left(\frac{\|R(\mathbf{z}_i,\mathbf{a})-R(\mathbf{z}_j,\mathbf{a})\|^2}{\|\mathbf{z}_i-\mathbf{z}_j\|^2}\right.\\&\quad\left.+\frac{\|\mathbf{z}_i-\mathbf{z}_j\|^2}{\|R(\mathbf{z}_i,\mathbf{a})-R(\mathbf{z}_j,\mathbf{a})\|^2}\right).\end{aligned} \tag{6}$$

To find the intrinsic dimension d_I we must calculate $S(d)$ for a number of d values. If $d<d_I$, then dimension increase allows better unfolding of the data, and $S(d)$ must be decreasing: $S(d+1)<S(d)$. On the other hand, if $d\geqslant d_I$ the dimension increase should not influence $S(d)$. In other words, at $d=d_I$ the values of $S(d)$ should stabilize.

However, this approach also has its shortcomings. If we use all possible pairs $\{i,j\}$, then the training set enlarges dramatically: instead of N vectors we need to deal with $N^2/2$ pairs. For large data sets this can make this algorithm also impractical. Nonetheless, as in other approaches [14,13,5], if we confine ourselves only to *local* properties of the mapping, only $\sim N$ pairs of nearest neighbors should be adequate to extract the value of d. The cost of this acceleration is the possibility of poor global properties of the mapping: for some remote $\mathbf{z}$ points the corresponding $\mathbf{y}$ may become close or coincide. That is, there may be no good global inverse mapping $\mathbf{y}\rightarrow\mathbf{z}$. On the other hand, using all the pairs, most of which involve remote points, there is a risk of losing local properties.

We applied this technique to both multilayer perceptrons and RBF. Computations for the latter case were 10–100 times faster because (i) the dependence of $E(\mathbf{a},d)$ on fitting parameters is simpler, which means that fewer iterations are needed and (ii) ψ does not depend on $\mathbf{a}$, all values of $\psi(\|\mathbf{z}_i-\mathbf{c}_j\|)$ can be calculated only once, and then used at every iteration so that each iteration takes much less time.

As we have mentioned before, the performance of RBF networks depends on the choice of approximation centers and on the basis function ψ. Let us consider these points in more detail.

Usually, the number of centers M is chosen to be less than N to avoid overfitting. At the same time, for a good approximation, there should not be any point $\mathbf{z}_i$ located too far from all centers. To satisfy these requirements, we applied the following technique. In what follows, we shall denote by $\mathbf{c}(\mathbf{z}_i)$ the center $\mathbf{c}$, which is the closest one to the vector $\mathbf{z}_i$.

The first center is chosen to be the center of mass of all the points, $\mathbf{c}_1=\langle\mathbf{z}\rangle$. Then we apply the following iterative scheme assuming that n centers are chosen.

Step 1. For each $\mathbf{z}_i$ find its nearest center $\mathbf{c}(\mathbf{z}_i)$. This splits the set of $\mathbf{z}$ vectors into Dirichlet-Voronoi cells around each $\mathbf{c}_j$. We find vector $\mathbf{z}^*$, for which $\|\mathbf{z}_i-\mathbf{c}(\mathbf{z}_i)\|$ is the greatest, and take $\mathbf{c}_{n+1}=\mathbf{z}^*$. In other words, the biggest cell is split.

Step 2. Addition of a center changes the structure of the Voronoi cells. We find the new cells [again for each $\mathbf{z}_i$ find its $\mathbf{c}(\mathbf{z}_i)$], then move each $\mathbf{c}_j$ to the center of mass of its own cell.

Step 3. Calculate the mean distance $r=\langle\|\mathbf{z}_i-\mathbf{c}(\mathbf{z}_i)\|\rangle$. Compare it with the mean distance r_0 between points $\mathbf{z}$. If r/r_0 is not small enough, return to Step 1.

This algorithm provides steady decrease of r with the addition of every new center. The homogeneous distribution of $\mathbf{c}$ over the set of $\mathbf{z}$ is important since we study the distances between close neighbors of the points. This algorithm can be accelerated if at Step 1 we split $k_a\leqslant n$ biggest cells instead of only one.

Step 1(a). For each $\mathbf{z}_i$ find its $\mathbf{c}(\mathbf{z}_i)$.

Step 1(b). For each cell j find its $\mathbf{z}_j^*$, the vector belonging to this cell with the largest $\|\mathbf{z}-\mathbf{c}(\mathbf{z})\|$. Let us denote $d_j=\|\mathbf{z}_j^*-\mathbf{c}(\mathbf{z}_j^*)\|$, and $d^*=\max_j d_j$.

Step 1(c). Select k_a cells with the largest d_j values, and choose among them only those, for which $d_j>\frac{1}{2}d^*$. Set the corresponding $\mathbf{z}_j^*$ as the new centers.

For big k_a the increase in speed may be very significant, but the total number of centers for the same r/r_0 also becomes greater. We used the value $k_a=2$.

Usually, for the application of the RBF technique, it is not required that the centers $\mathbf{c}$ should coincide with one of the $\mathbf{z}$ vectors. On the other hand, since we approximate a d-dimensional surface in m-dimensional space, it may be desirable to restrict the approximation centers to this surface, and the easiest way to do it is to place the centers into some of the vectors $\mathbf{z}$. This restricted placement can be easily done if in Step 2 we move each center $\mathbf{c}_j$ not to the center of mass of its Voronoi cell, but to the vector that is closest to the center of mass.

Usually the restricted allocation of centers results in slightly greater (about 10%) number of centers for the same r/r_0. All test calculations below were performed for $k_a=2$ with both restricted and nonrestricted center allocation. The results in both cases coincide, so at present there are no numerical evidence in favor of the restricted allocation, though in some cases it may be better. The presented figures correspond to the restricted center allocation.

The choice of ψ is also crucial. One of the typical choices is $\psi(r)=\exp(-r^2/s^2)$. The performance of the method depends on the choice of s: for small s (effectively localized ψ) it is possible to obtain smaller values for the function S (5), but some effects of overfitting arise. The optimal choice of s proved to be close to the diameter of the set of $\mathbf{z}$.

The pairs $\{i,j\}$ used in Eq. (6) were selected to satisfy the following criteria: (i) the number of pairs should be close to N to expedite computations; (ii) for noisy data, the pairs should not be too close to one another; (iii) every or almost every point must participate in at least one pair. There may be many ways of pair selection, we found the following scheme satisfactory. According to the noise level, we set a minimal distance ϵ between points in pairs. In case of restricted center allocation for each $\mathbf{z}_i$, which is a center we find its ϵ-nearest neighbor (the nearest neighbor after discarding all neighbor closer than ϵ); if $\mathbf{z}_i$ is not a center we find its ϵ-nearest center. For nonrestricted allocation for each $\mathbf{z}_i$ we used the ϵ-nearest neighbor, which is not yet included in other pairs. Then the structure of the nearest neighbors is accounted for.

IV. NUMERICAL EXPERIMENTS

To test our approach, we used the following data sets: (1) two-dimensional surface in five-dimensional space, intrinsic dimension $d_I=2$; (2) time series for Lorenz attractor, $d_I=3$; (3) pseudorandom numbers for which there is no intrinsic dimension; (4) time series for 5 torus, $d_I=5$; and (5) time series for the Lorenz attractor with noise. For the sets (2)–(5), the scalar data were transformed to zero mean and unit variance before forming m-dimensional delay vectors from them. The topological mapping has been used in the form (6) with the techniques of selecting approximating centers and training pairs described above. The usual choice for the r/r_0 ratio was 1 or 2.

The test for the surface in two-dimensional space illustrates the approach. The five-dimensional data $\mathbf{z}$ were formed as follows. First two components z_1 and z_2 formed a regular 20×20 grid on a plane with identical steps in both directions, the third component $z_3=z_1^2+z_2^2$, $z_4=z_5=0$ [Fig. 1(a)]. For these $N=400$ points we made a set of centers $\mathbf{c}_j$, and performed minimization of the functional (6). For minimization we set small random initial values for all a, and then applied the method of conjugate gradients (see, e.g., Ref. [1] or any book in nonlinear optimization). Minimization takes a few thousands of steps, though usually after 1000–2000 steps the dependence of $S(d)$ is clearly visible. The obtained values were $S(1)\cong 4.8$ and for $d\geqslant 2$ $S(d)$ was almost exactly 2. Therefore, we can conclude that the intrinsic dimensionality of the data is $d_I=2$.

To show the influence of the choice of the radial basis function $\psi(r)=\exp(-r^2/s^2)$, we performed calculations for two values of s: the usual choice equal to the diameter of the data set $s_1=\max_{i,k}\|\mathbf{z}_i-\mathbf{z}_k\|$ and $s_2=0.1s_1$. The resulting two-

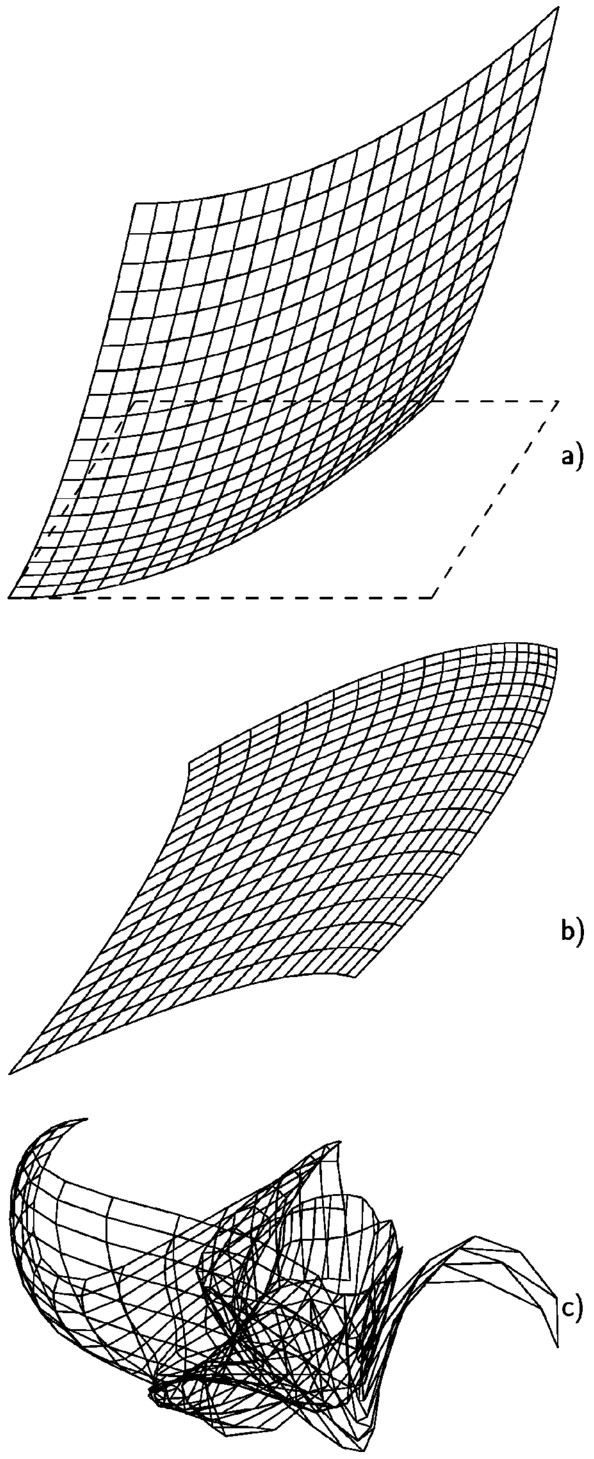

FIG. 1. Results of a topological mapping for two-dimensional surface in five-dimensional space. (a) The original surface. (b) Its two-dimensional projection $\mathbf{y}=R(\mathbf{z},\mathbf{a})$, the radial basis function parameter s equal to the diameter of the data set. (c) Similar 2D projection with s ten times less. For small s the effects of overfitting-type arise, though local properties of the original surface are captured correctly. In other examples below we used s equal to the diameter of the data set.

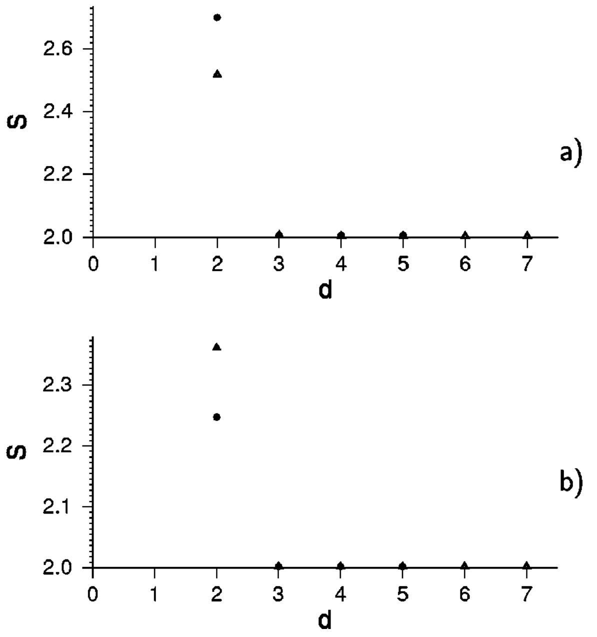

FIG. 2. Results for the topological mapping for delay reconstructions of the Lorenz attractor with $\tau = 0.03$, $m = 5$ (●), and $m = 7$ (△). The value of $S(1)$ is not shown, because it is too big. The length of time series is $N_T = 5000$ (a) and 600 (b). For $d \geqslant 3$ the increase of d does not influence $S(d)$, so we can conclude that the intrinsic dimension $d_I = 3$.

dimensional projections of the data $\mathbf{y}_i = R(\mathbf{z}_i, \mathbf{a})$ ($d = 2$) are shown in Figs. 1(b), 1(c), respectively. In both cases the local structure of data is captured properly, though for small s the effects similar to overfitting appear. Global properties of the mapping R are better in the case $s = s_1$. For this reason the choice $s = \max_{i,k} \|\mathbf{z}_i - \mathbf{z}_k\|$ has been used in all other examples below.

For the Lorenz attractor, we used scalar time series of the component x_1 with time step $\tau = 0.03$, the parameters of the system were standard—(10, 28, 8/3). We performed a number of calculations with different time series length N_T and embedding dimension m. To reduce the amount of computations we used only every third reconstructed vector, so $N = N_T/3$. Figure 2(a) shows the results for $N_T = 10^4$, $m = 5$, circles, and $m = 7$, triangles. In this figure and below $S(1) \gg S(2)$, and to show the details of the $S(d)$ dependence we do not show $S(1)$ in the figures. In both cases $m = 5$ and 7 S reaches its minimal value at $d = 3$ and further increase of d does not change the results. Therefore the technique gives the correct estimate of the dimension of the Lorenz system $d_I = 3$. Most interesting observation related with the Lorenz system is that its dimension can be estimated from a very short time series. Figure 2(b) shows the results for $N_T = 600$, and the correct intrinsic dimension still can be estimated.

For comparison, the results for random data are shown in Fig. 3. We processed the sets of $N = 1400$ five- and seven-dimensional random vectors with components independent and homogeneously distributed on [0,1]. It is clear that in both cases $m = 5$ and $m = 7$ $S_{\min}$ monotonously approaches its least value of 2.

Figure 4 shows the results for a 5 torus, a sum of five harmonic oscillations with incommensurate frequencies. We used it as an example of a higher-dimensional system. The results give a correct value $d_I = 5$.

In all these examples, the parameter ϵ was very close to 0, just to avoid pairs that almost coincide and can reduce the accuracy of numerical computations. To show an example of processing noisy data we made a signal with noise. We took a time series for the Lorenz system of the example above, transformed it to zero mean and unit variance and then added to it a noise term 0.2ξ, where ξ is uniformly distributed on $[-1,1]$. Figure 5 shows the results for different ϵ: for small values the results resemble random data, for greater values

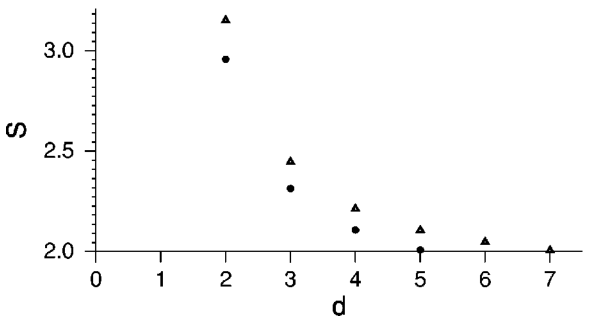

FIG. 3. Topological mapping for random data, $m = 5$ (●) and $m = 7$ (△). Note the steady decrease of $S_{\min}$ with d and the absence of stabilization.

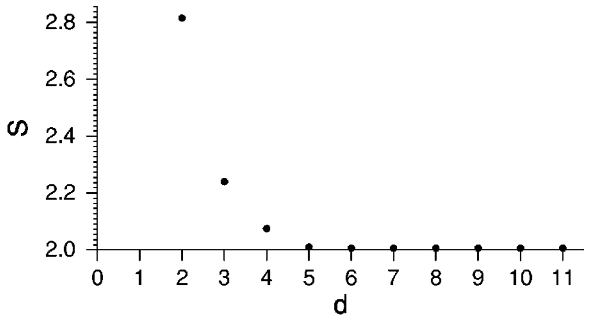

FIG. 4. Topological mapping for delay reconstructions of 5 torus with $\tau = 0.1$, $m = 11$. The estimated $d_I = 5$.

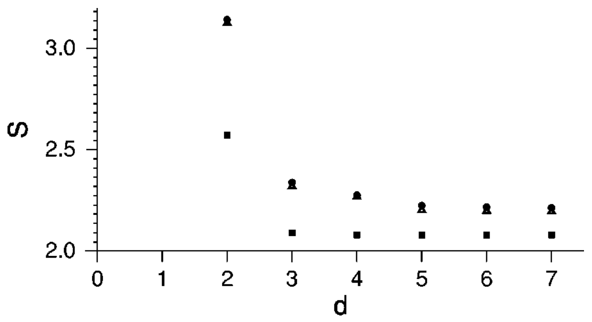

FIG. 5. Results for the Lorenz system with noise. To process the data it is necessary to increase the parameter ε: $\varepsilon = 0.001$ (●), $\varepsilon = 0.1$ (△), $\varepsilon = 0.3$ (■). For small ε noise increases the dimension estimate to 5, while in the last case, when ε is greater than the noise amplitude, the estimate of $d_I = 3$ is correct.

they give the correct estimate $d_I=3$.

It is important to note that the applicability of the described technique, like any other technique of dimension estimates [5], depends on the type and quality of data. The topological mapping extracts the properties of finite set of points, and to get a correct estimate, this set must correctly reflect the properties of the underlying object, e.g., the geometry of a dynamical system attractor. For low-dimensional dynamical system this is often satisfied. At the same time, there are situations when geometrical distortions can influence the arrangement of the data points. For high-dimensional systems the number of data points may be insufficient for detecting all dimensions and so on. This in turn may change the results of the algorithms. One of the well-known sources of distortions is the delay reconstruction (2) [15–17]. Other sources of distortions may arise in the course of measurements. Therefore, we cannot guarantee correct estimates of intrinsic dimensions of *all* kinds of data. It is necessary to make sure that there are no essential distortions in the data. Sometimes a combined application of several different algorithms may be recommended.

Comparing this algorithm with, e.g., the method of local false neighbors (LFN) [13], we can say, that it occupies a different ‘‘ecological niche.’’ The LFN algorithm requires rather big amount of data, about 10^4 data points and is very fast (for smaller amounts of data it becomes hard to detect false neighbors). Our technique works well for ten and more times smaller data sets. It requires essentially more computations, but the capabilities of modern Pentium III or Pentium 4 personal computers are enough for its application.

Compared to the use of perceptrons for approximation of the projection mapping, the application of RBF networks is better for two reasons. First, they often require less minimization steps and each step can be done faster (Sec. III). Second, the dependence of results on the initial guess for the weights is better. Quite often the minimization process stops at a local minimum. According to our experiments, for perceptrons such local minima often provide poor approximation, and usually it is necessary to use 5–10 different initial guesses and to choose the best result. For RBF approximations in Eq. (6) there are also numerous local minima, and the resulting mapping may also depend on initial guess for **a**, but almost all of these mappings are equivalent in terms of S values, and we almost never found essential differences. In one of the tests we used 50 different initial guesses for **a**, and after 30 000 steps of minimization in 26 cases we obtained $S=2.011$, in 21 cases $S=2.012$, and only three values were equal to 2.029, 2.047, and 2.085. Therefore, this technique does not require numerous repetitions. Another important point is that the type of dependence $S(d)$ can be observed typically after 1000–2000 minimization steps, sometimes earlier. Although the values of $S(d)$ are not yet established after these steps, the trend in d is usually stabilized and it is possible to draw conclusions about the dimensionality of data.

V. CONCLUSION

We have proposed an algorithm based on RBF networks, which, by analogy with Ref. [2], can be called ‘‘topological nonlinear PCA.’’ It extracts the intrinsic dimension of data. Intrinsic dimension can be important for diagnostic purposes or for designing a controlling neural network that should analyze observed data. This technique is versatile and fast compared to other neural approaches. On the other hand, it is based mainly upon *local* topological properties of data, while preservation of global topology is not guaranteed. For this reason, this technique may not be useful for data compression. Nonetheless, it gives an essential information for building an autoassociative network that can compress the data.

ACKNOWLEDGMENTS

This work has been supported through a grant to M.K.A. from the Defense Research Establishment Suffield under the Contract No. W7702-8-R745/001/EDM. The work was carried out using computing facilities provided by Multimedia Advanced Computational Infrastructure (MACI).

[1] C. M. Bishop, *Neural Networks for Pattern Recognition* (Clarendon Press, Oxford, 1995).
[2] M.A. Kramer, AIChE J. **37**, 233 (1991).
[3] S. Nolfi and J. Tani, Connection Science **11**, 129 (1999).
[4] F. Takens, Lect. Notes Math. **898**, 336 (1981).
[5] H.D.I. Abarbanel, R. Brown, J. Sidorowich, and L.S. Tsimring, Rev. Mod. Phys. **65**, 1331 (1993).
[6] J.P. Eckmann and D. Ruelle, Rev. Mod. Phys. **57**, 617 (1985).
[7] T. Schreiber, Phys. Rep. **64**, 1 (1999).
[8] T. Hediger, A. Passamante, and M.E. Farell, Phys. Rev. A **41**, 5325 (1990).
[9] R. Temam, *Infinite-Dimensional Dynamical Systems in Mechanics and Physics* (Springer, Berlin, 1988).
[10] H. Haken, *Synergetics. An Introduction* (Springer, Berlin, 1977).
[11] B.J. Bayly, I. Goldhirch, and S.A. Orszag, J. Sci. Comp. **2**, 111 (1987).
[12] D.S. Broomhead and R. Jones, Proc. R. Soc. London, Ser. A **423**, 103 (1989).
[13] H.D.I. Ababrbanel and M.B. Kennel, Phys. Rev. E **47**, 3057 (1993).
[14] W. Liebert, K. Pawelzik, and H.G. Schuster, Europhys. Lett. **14**, 521 (1991).
[15] M. Casdagli, S. Eubank, J.D. Farmer, and J. Gibson, Physica D **51**, 52 (1991).
[16] A.B. Potapov, Physica D **101**, 207 (1997).
[17] A.B. Potapov and J. Kurths, Physica D **120**, 369 (1998).