"Neural-Gas" Network for Vector Quantization and its Application to Time-Series Prediction

Thomas M. Martinetz, *Member, IEEE,* Stanislav G. Berkovich, and Klaus J. Schulten

***Abstract*—As a data compression technique, vector quantization requires the minimization of a cost function—the distortion error—which, in general, has many local minima. In this paper, a neural network algorithm based on a "soft-max" adaptation rule is presented that exhibits good performance in reaching the optimum, or at least coming close. The soft-max rule employed is an extension of the standard K-means clustering procedure and takes into account a "neighborhood ranking" of the reference (weight) vectors. It is shown that the dynamics of the reference (weight) vectors during the input-driven adaptation procedure 1) is determined by the gradient of an energy function whose shape can be modulated through a neighborhood determining parameter, and 2) resembles the dynamics of Brownian particles moving in a potential determined by the data point density. The network is employed to represent the attractor of the Mackey–Glass equation and to predict the Mackey–Glass time series, with additional local linear mappings for generating output values. The results obtained for the time-series prediction compare very favorably with the results achieved by back-propagation and radial basis function networks.**

I. Introduction

ARTIFICIAL as well as biological information processing systems that are involved in storing or transfering large amounts of data often require the application of coding techniques for data compression. In a wide range of applications, including speech and image processing, the data compression procedure is based on "vector quantization" techniques (for a review see [1]).

Vector quantization techniques encode a data manifold, e.g., a submanifold $V \subseteq \mathcal{R}^D$, utilizing only a finite set $\boldsymbol{w} = (\boldsymbol{w}_1, \cdots, \boldsymbol{w}_N)$ of reference or "codebook" vectors (also called cluster centers) $\boldsymbol{w}_i \in \mathcal{R}^D, i = 1, \cdots, N$. A data vector $\boldsymbol{v} \in V$ is described by the best-matching or "winning" reference vector $\boldsymbol{w}_{i(\boldsymbol{v})}$ of $\boldsymbol{w}$ for which the distortion error $d(\boldsymbol{v}, \boldsymbol{w}_{i(\boldsymbol{v})})$, e.g., the squared error $||\boldsymbol{v}-\boldsymbol{w}_{i(\boldsymbol{v})}||^2$, is minimal. This procedure divides the manifold V into a number of subregions

$$V_i = \{\boldsymbol{v} \in V | ||\boldsymbol{v} - \boldsymbol{w}_i|| \leq ||\boldsymbol{v} - \boldsymbol{w}_j|| \forall j\} \tag{1}$$

called Voronoi polygons or Voronoi polyhedra, out of which each data vector $\boldsymbol{v}$ is described by the corresponding reference vector $\boldsymbol{w}_i$. If the probability distribution of data vectors over the manifold V is described by $P(\boldsymbol{v})$, then the average distortion or reconstruction error is determined by

$$E = \int d^D v P(\boldsymbol{v})(\boldsymbol{v} - \boldsymbol{w}_{i(\boldsymbol{v})})^2 \tag{2}$$

and has to be minimized through an optimal choice of reference vectors $\boldsymbol{w}_i$.

The straightforward approach to minimizing (2) would be a gradient descent on E which leads to Lloyd and MacQueen's well-known K-means clustering algorithm [2], [3]. In its online version, which is applied if the data point distribution $P(\boldsymbol{v})$ is not given *a priori*, but instead a stochastic sequence of incoming sample data points $\boldsymbol{v}(t = 1), \boldsymbol{v}(t = 2), \boldsymbol{v}(t = 3), \cdots$ which is governed by $P(\boldsymbol{v})$ drives the adaptation procedure, the adjustment steps for the reference vectors or cluster centers $\boldsymbol{w}_i$ is determined by

$$\Delta \boldsymbol{w}_i = \epsilon \cdot \delta_{ii(\boldsymbol{v}(t))} \cdot (\boldsymbol{v}(t) - \boldsymbol{w}_i) \qquad i = 1, \cdots, N \tag{3}$$

with ϵ as the step size and δ_{ij} as the Kronecker delta. However, adaptation step (3) as a stochastic gradient descent on E can, in general, only provide suboptimal choices of reference vectors $\boldsymbol{w}_i$ for nontrivial data distributions $P(\boldsymbol{v})$ and nontrivial numbers of reference vectors, due to the fact that the error surface E has many local minima.

To avoid confinement to local minima during the adaptation procedure, a common approach is to introduce a "soft-max" adaptation rule that not only adjusts the "winning" reference vector $i(\boldsymbol{v})$ as in (3), but affects all cluster centers depending on their proximity to $\boldsymbol{v}$. One interesting approach of this kind, to which we will refer as "maximum-entropy" clustering [4], employs the adaptation step

$$\Delta \boldsymbol{w}_i = \epsilon \cdot \frac{e^{-\beta(\boldsymbol{v}-\boldsymbol{w}_i)^2}}{\sum_{j=1}^{N} e^{-\beta(\boldsymbol{v}-\boldsymbol{w}_j)^2}} \cdot (\boldsymbol{v} - \boldsymbol{w}_i) \qquad i = 1, \cdots, N \tag{4}$$

which corresponds to a stochastic gradient descent on the cost function

$$E_{mec} = -\frac{1}{\beta} \int d^D v \, P(v) \ln \sum_{i=1}^{N} e^{-\beta(\boldsymbol{v}-\boldsymbol{w}_i)^2} \tag{5}$$

instead of (2). As we can see in (4), with a data vector $\boldsymbol{v}$ not only the "winning" reference vector $\boldsymbol{w}_{i(\boldsymbol{v})}$ but each $\boldsymbol{w}_i$ is updated, with a step size that decreases with the distance $||\boldsymbol{v} - \boldsymbol{w}_i||$ between $\boldsymbol{w}_i$ and the currently presented $\boldsymbol{v}$. For

Manuscript received November 11, 1991; revised September 13, 1992. This work was supported by the National Science Foundation under Grant DIR-90-15561 and by a Fellowship of the Volkswagen Foundation to T. M. Martinetz.

The authors are with the Beckman Institute and the Department of Physics, University of Illinois at Urbana-Champaign, Urbana, IL 61801.

IEEE Log Number 9204650.

$\beta \to \infty$ the cost function E_{mec} becomes equivalent to E. Rose *et al.* [4] have shown what the form of (5) suggests, namely that adaptation step (4) is, in fact, a deterministic annealing procedure with β as the inverse temperature and E_{mec} as the free energy. By starting at a high temperature which, during the adaptation process, is carefully lowered to zero, local minima in E are avoided. However, to obtain good results, the decrement of the temperature has to be very slow. In theory it must hold $\beta \propto \ln t$ with t as the number of iteration steps (4) performed [5]. Adaptation step (4) is also used in the context of "maximum likelihood" approaches where, as reported in [6], the goal is to model the data distribution $P(\boldsymbol{v})$ through overlapping radial basis functions (RBF's) (adaptation step (4) corresponds to Gaussians as RBF's).

Kohonen's topology-conserving feature map algorithm [7]–[9] is another well-known model that has been applied to the task of vector quantization ([10]–[18]) and that incorporates a soft-max adaptation rule. In Kohonen's model every reference vector $\boldsymbol{w}_i$ is assigned to a site i of a lattice A. Each time a data vector $\boldsymbol{v}$ is presented, not only the "winning" reference vector $\boldsymbol{w}_{i(\boldsymbol{v})}$ is adjusted according to (3), but also the reference vectors $\boldsymbol{w}_i$ that are assigned to lattice sites i adjacent to $i(\boldsymbol{v})$ are updated, with a step size that decreases with the lattice distance between i and $i(\boldsymbol{v})$. The corresponding adaptation step is of the form

$$\Delta \boldsymbol{w}_i = \epsilon \cdot h_\sigma(i, i(\boldsymbol{v})) \cdot (\boldsymbol{v} - \boldsymbol{w}_i) \qquad i = 1, \cdots, N \quad (6)$$

with $h_\sigma(i,j)$ as a unimodal function that decreases monotonically for increasing $||i-j||$ with a characteristic decay constant σ. For $\sigma = 0, h_\sigma(i,j) = \delta_{ij}$ and (6) becomes equivalent to the K-means adaptation rule (3). Kohonen's model is particularly interesting since, through (6), a mapping between the data space V and the chosen lattice A emerges that maintains the topology of the input space V as completely as possible [9], [19]. Kohonen's topology-conserving maps have been employed successfully in a wide range of applications, from the modeling of biological feature maps found in the cortex [20] to technical applications like speech recognition [10], [11], [15], image processing [13], [14], and the control of robot arms [12], [16]–[18], [21].

Adaptation step (6) provides reasonable distributions of the reference vectors with only a small number of iteration steps, which is essential especially in technical applications like the control of robot arms [12], [16]–[18], [21]. However, to obtain good results with Kohonen's algorithm as a vector quantizer, the topology of the lattice A has to match the topology of the space V which is to be represented. In addition, there exists no cost function that yields Kohonen's adaptation rule as its gradient [22], [23]. In fact, as long as $\sigma > 0$, one cannot specify a cost function that is minimized by (6).

II. The "Neural-Gas" Algorithm

In this paper, we present a neural network model which, applied to the task of vector quantization, 1) converges quickly to low distortion errors, 2) reaches a distortion error E lower than that resulting from K-means clustering, maximum-entropy clustering (for practically feasible numbers of iteration steps) and from Kohonen's feature map, and 3) at the same time obeys a gradient descent on an energy surface (like the maximum-entropy clustering, in contrast to Kohonen's feature map algorithm). For reasons we will give later, we call this network model the "neural-gas" network. Similar to the maximum-entropy clustering and Kohonen's feature map the neural-gas network also uses a "soft-max" adaptation rule. However, instead of the distance $||\boldsymbol{v} - \boldsymbol{w}_i||$ or of the arrangement of the $\boldsymbol{w}_i$'s within an external lattice, it utilizes a "neighborhood-ranking" of the reference vectors $\boldsymbol{w}_i$ for the given data vector $\boldsymbol{v}$.

Each time data vector $\boldsymbol{v}$ is presented, we determine the "neighborhood-ranking" $(\boldsymbol{w}_{i_0}, \boldsymbol{w}_{i_1}, \cdots, \boldsymbol{w}_{i_{N-1}})$ of the reference vectors with $\boldsymbol{w}_{i_0}$ being closest to $\boldsymbol{v}$, $\boldsymbol{w}_{i_1}$ being second closest to $\boldsymbol{v}$, and $\boldsymbol{w}_{i_k}, k = 0, \cdots, N-1$ being the reference vector for which there are k vectors $\boldsymbol{w}_j$ with $||\boldsymbol{v} - \boldsymbol{w}_j|| < ||\boldsymbol{v} - \boldsymbol{w}_{i_k}||$. If we denote the number k associated with each vector $\boldsymbol{w}_i$ by $k_i(\boldsymbol{v}, \boldsymbol{w})$, which depends on $\boldsymbol{v}$ and the whole set $\boldsymbol{w} = (\boldsymbol{w}_1, \cdots, \boldsymbol{w}_N)$ of reference vectors, then the adaptation step we employ for adjusting the $\boldsymbol{w}_i$'s is given by

$$\Delta \boldsymbol{w}_i = \epsilon \cdot h_\lambda(k_i(\boldsymbol{v}, \boldsymbol{w})) \cdot (\boldsymbol{v} - \boldsymbol{w}_i) \qquad i = 1, \cdots, N. \quad (7)$$

The step size $\epsilon \in [0,1]$ describes the overall extent of the modification, and $h_\lambda(k_i(\boldsymbol{v}, \boldsymbol{w}))$ is unity for $k_i = 0$ and decays to zero for increasing k_i with a characteristic decay constant λ. In the simulations we describe as follows, we chose $h_\lambda(k_i(\boldsymbol{v}, \boldsymbol{w})) = e^{-k_i(\boldsymbol{v}, \boldsymbol{w})/\lambda}$. For $\lambda \to 0$, (7) becomes equivalent to the K-means adaptation rule (3), whereas for $\lambda \neq 0$ not only the "winner" $\boldsymbol{w}_{i_0}$ but the second closest reference vector $\boldsymbol{w}_{i_1}$, third closest reference vector $\boldsymbol{w}_{i_2}$, etc., is also updated.

As we show in Appendix I, the dynamics of the $\boldsymbol{w}_i$'s obeys a stochastic gradient descent on the cost function

$$E_{ng}(\boldsymbol{w}, \lambda) = \frac{1}{2C(\lambda)} \sum_{i=1}^{N} \int d^D v \, P(\boldsymbol{v}) h_\lambda(k_i(\boldsymbol{v}, \boldsymbol{w}))(\boldsymbol{v} - \boldsymbol{w}_i)^2 \quad (8)$$

with

$$C(\lambda) = \sum_{i=1}^{N} h_\lambda(k_i) = \sum_{k=0}^{N-1} h_\lambda(k)$$

as a normalization factor that only depends on λ. E_{ng} is related to the framework of fuzzy clustering [24], [25]. In contrast to hard clustering where each data point $\boldsymbol{v}$ is deterministically assigned to its closest reference vector $\boldsymbol{w}_{i(\boldsymbol{v})}$, fuzzy clustering associates $\boldsymbol{v}$ to a reference vector $\boldsymbol{w}_i$ with a certain degree $p_i(\boldsymbol{v})$, the so-called "fuzzy membership" of $\boldsymbol{v}$ to cluster i. In the case of hard clustering holds $p_{i_0}(\boldsymbol{v}) = 1$ and $p_i(\boldsymbol{v}) = 0$ for $i \neq i_0$. If we choose a "fuzzy" assignment of data point $\boldsymbol{v}$ to reference vector $\boldsymbol{w}_i$, which depends on whether $\boldsymbol{w}_i$ is the nearest, next-nearest, next-next-nearest, etc., neighbor of $\boldsymbol{v}$, i.e., if we choose $p_i(\boldsymbol{v}) = h_\lambda(k_i(\boldsymbol{v}, \boldsymbol{w}))/C(\lambda)$, then the average distortion error we obtain, and which has to be minimized, is given by E_{ng}, and the corresponding gradient descent is given by adaptation rule (7).

Through the decay constant λ we can modulate the shape of the cost function E_{ng}. For $\lambda \to \infty$ the cost function E_{ng} becomes parabolic, whereas for $\lambda \to 0$ it becomes equivalent to the cost function E in (2), i.e., the cost function we ultimately want to minimize, but which has many local minima. Therefore, to obtain good results concerning the set of reference vectors, we start the adaptation process determined by (7) with a large decay constant λ and decrease λ with each adaptation step. By gradually decreasing the parameter λ we expect the local minima of E to emerge slowly, thereby preventing the set $\boldsymbol{w}$ of reference vectors from getting trapped in suboptimal states.

III. The Network's Performance on a Model Problem

To test the performance of the neural-gas algorithm in minimizing E and to compare it with the three other approaches we described (K-means clustering, maximum-entropy clustering, and Kohonen's topology-conserving map), we choose a data distribution $P(\boldsymbol{v})$ for which 1) the global minimum of E is known for large numbers of reference vectors and 2) which reflects, at least schematically, essential features of data distributions that are typical in applications. Data distributions that occur in applications often consist of, eventually separated, clusters of data points. Therefore, also for our test we choose a model data distribution that is clustered. To be able to determine the global minimum, in our model data distribution the clusters are of square shape within a two-dimensional input space. Since we choose $N = 4 \times$*number of clusters* and separate the clusters far enough from each other, the optimal set of $\boldsymbol{w}_i$'s is given when each of the square clusters is represented by four reference vectors, and when the four reference vectors within each cluster are arranged in the known optimal configuration for a single square.

In Fig. 1 we see the neural-gas adapting into a representation of our model data distribution with 15 clusters and $N = 60$ reference vectors. With each adaptation step, a data point within one of the squares is stochastically chosen with equal probability over each square. Subsequently, adjustments of the $\boldsymbol{w}_i$'s according to (7) are performed. We show the initial state, the state after 5000, 15 000, and finally after 80 000 adaptation steps. In the simulation run depicted in Fig. 1 the neural-gas algorithm was able to find the optimal representation of the data distribution.

However, depending on the initial choice of the $\boldsymbol{w}_i$'s (chosen randomly) and depending on the speed with which the parameter λ is decreased, i.e., depending on the total number of adaptation steps $t_{\max}$ employed, it might happen that the reference vectors converge to a configuration that is only close but not exactly at the optimum. Therefore, to demonstrate the average performance of the neural-gas algorithm we show in Fig. 2 the mean distortion error for different total numbers of adaptation steps $t_{\max}$. For each of the different total numbers of adaptation steps we averaged over 50 simulation runs, for each of which not only the initialization of the $\boldsymbol{w}_i$'s were chosen randomly but also the 15 clusters of our model data distribution were placed randomly. Since we know the minimal distortion error E_0 that can be optimally achieved

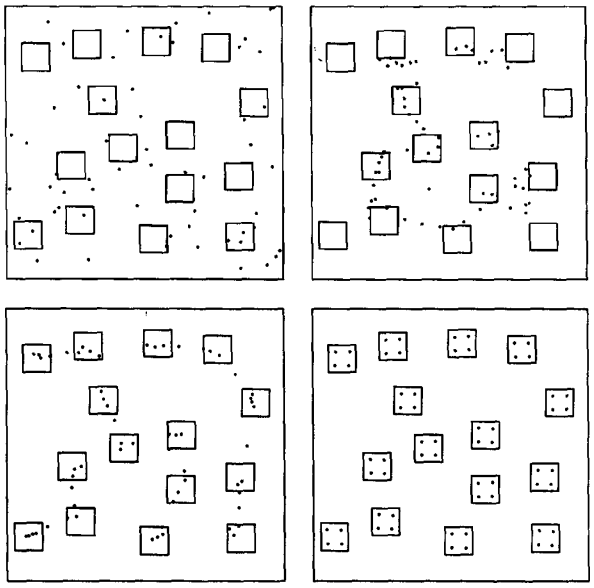

Fig. 1. The neural-gas network representing a data distribution in $\mathcal{R}^2$ that consists of 15 separated clusters of square shape. On each cluster the density of data points is homogeneous. The 60 reference vectors $\boldsymbol{w}_i$ are depicted as points. The initial values for the $\boldsymbol{w}_i$'s are chosen at random, which is shown in the top left picture. We also show the state after 5000 (top right), 15 000 (bottom left), and after 80 000 adaptation steps (bottom right). At the end of the adaptation procedure the set of reference vectors has converged to the optimal configuration, i.e., each cluster is represented by four reference vectors.

for our model data distribution and the number of reference vectors we employ, we choose $\alpha = (E(t_{\max}) - E_0)/E_0$ as a performance measure with $E(t_{\max})$ as the final distortion error reached. $\alpha = 0$ corresponds to a simulation run which reached the global minimum, whereas, e.g., $\alpha = 1$ corresponds to a very large distortion error, i.e., a distortion error which is twice as large as the optimum. As we can see in Fig. 2, for $t_{\max} = 100\,000$ the average performance of the neural-gas network is $\alpha = 0.09$, which means that the average distortion error E for $t_{\max} = 100\,000$ is 9% larger than what can be optimally achieved.

For comparison, we also show in Fig. 2 the result achieved by the K-means clustering, the maximum-entropy clustering, and Kohonen's feature map algorithm. Up to $t_{\max} = 8000$, only the distortion error of the K-means clustering is slightly smaller than the distortion error of the neural-gas algorithm. For $t_{\max} > 8000$, all three procedures perform worse than the neural-gas algorithm. For a total number of 100 000 adaptation steps the distortion error of the maximum-entropy clustering is more than twice as large as the distortion error achieved by the neural-gas algorithm. Theoretically, for the maximum-entropy approach the performance measure α should converge to zero for $t_{\max} \to \infty$. However, as mentioned already in the introduction, the convergence might be extremely slow. Indeed, all four clustering procedures, including the maximum-entropy approach and the neural-gas algorithm, do not improve their final distortion error significantly further within the range

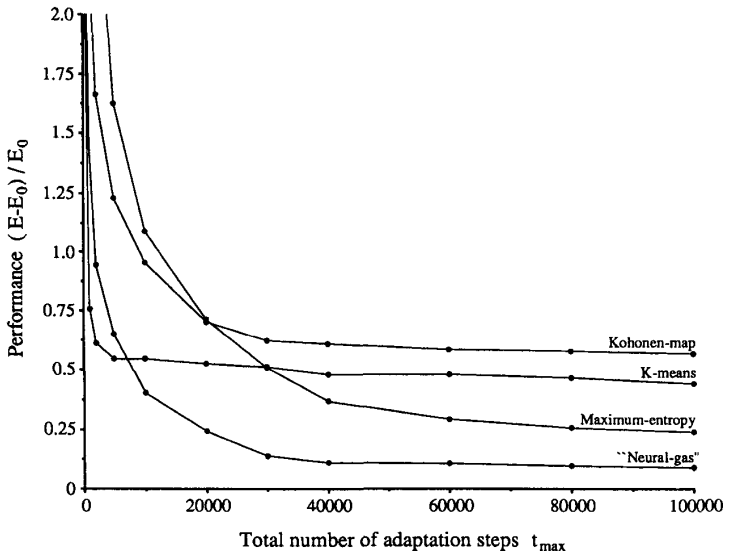

Fig. 2. The performance of the neural-gas algorithm in minimizing the distortion error E for the model distribution of data points which is described in the text and an example of which is shown in Fig. 1. Depicted is the result for different total numbers of adaptation steps $t_{\max}$. The performance measure is $(E - E_0)/E_0$ with E_0 as the minimal distortion error that can be achieved for the type of data distribution and the number of reference vectors we used for the test. For comparison we also show the result obtained with the standard K-means clustering, maximum-entropy clustering, and Kohonen's feature map algorithm. Up to $t_{\max} = 8000$ only the distortion error of the K-means clustering is slightly smaller than the distortion error of the neural-gas algorithm. For $t_{\max} > 8000$, the three other approaches perform worse than the neural-gas model. For a total number of 100 000 adaptation steps the distortion error of the neural-gas algorithm is smaller by more than a factor of two than the distortion error achieved by the maximum-entropy procedure.

$100\,000 < t_{\max} < 500\,000$. $t_{\max} = 500\,000$ is the limit up to which the tests were made.

Fig. 2 demonstrates that the convergence of the neural-gas algorithm is faster than the convergence of the three other approaches. This is important for practical applications where adaptation steps are "expensive," e.g., in robot control where each adaptation step corresponds to a trial movement of the robot arm. Applications that require the learning of input–output relations, vector quantization networks establish a representation of the input space that can then be used for generating output values, either through discrete output values [26], local linear mappings [12], or through radial basis functions [27]. Kohonen's topology-conserving map as a vector quantizer, together with local linear mappings for generating output values, has been studied for a number of robot control tasks [12], [16]–[18]. However, for the reason of faster convergence, we took the neural-gas algorithm for an implementation of the learning algorithms [16]–[18] on an industrial robot arm [28]. Compared to the versions that are based on Kohonen's feature map and require about 6000 adaptation steps (trial movements of the robot arm) to reach the minimal positioning error [18], only 3000 steps are sufficient when the neural-gas network is employed [28].

For the simulations of the neural-gas network as presented in Fig. 2 we chose $h_\lambda(k) = \exp(-k/\lambda)$ with λ decreasing exponentially with the number of adaptation steps t as $\lambda(t) = \lambda_i(\lambda_f/\lambda_i)^{t/t_{\max}}$ with $\lambda_i = 10, \lambda_f = 0.01$, and $t_{\max} \in [0, 100\,000]$. Compared to other choices for the neighborhood function $h_\lambda(k)$, e.g., Gaussians, $h_\lambda(k) = \exp(-k/\lambda)$ provided the best result. The step size ϵ has the same time dependence as λ, i.e., $\epsilon(t) = \epsilon_i(\epsilon_f/\epsilon_i)^{t/t_{\max}}$ with $\epsilon_i = 0.5$ and $\epsilon_f = 0.005$. The similarity of the neural-gas network and the Kohonen algorithm motivated the time dependence $x(t) = x_i(x_f/x_i)^{t/t_{\max}}$ for ϵ and λ. This time dependence has provided good results in applications of the Kohonen network [16]–[18]. The particular choice for $\lambda_i, \lambda_f, \epsilon_i$, and ϵ_f is not very critical and was optimized by "trial and error." The only simulation parameter of the the adaptive K-means clustering is the step size ϵ, which was chosen in our simulations identical to that of the neural-gas algorithm. In contrast to the three other vector quantization procedures, the final result of the K-means clustering depends very much on the quality of the initial distribution of the reference vectors $\boldsymbol{w}_i$. Therefore, to avoid a comparison in favor of the neural-gas network, we initialized the K-means algorithm more prestructured by exploiting a priori knowledge about the data distribution. Rather than initializing the $\boldsymbol{w}_i$'s totally at random, they were randomly assigned to vectors lying within the 15 clusters. This choice prevents that some of the codebook vectors remain unused. For the Monte Carlo simulations of the maximum-entropy clustering the step size ϵ was also chosen as for the neural-gas algorithm. The inverse temperature β had the time dependence $\beta(t) = \beta_i(\beta_f/\beta_i)^{t/t_{\max}}$ with $\beta_i = 1$ and $\beta_f = 10\,000$. This scheduling of β provided the best results for the range of total numbers of adaptation steps $t_{\max}$ that was investigated. Also for Kohonen's feature map algorithm the step size ϵ was chosen as for the neural-gas algorithm and the other two clustering procedures. The function $h_\sigma(i, j)$ that determines the neighborhood relation between site i and site j of the lattice A of Kohonen's feature map algorithm was chosen to be a Gaussian of the form $h_\sigma(i, j) = \exp(-||i - j||^2/2\sigma^2)$ [9], [16]–[18]. The decay constant σ, like λ, decreased with

the adaptation steps t according to $\sigma(t) = \sigma_i(\sigma_f/\sigma_i)^{t/t_{\max}}$ with $\sigma_i = 2$ and $\sigma_f = 0.01$. The values for σ_i and σ_f were optimized.

IV. "Gas-like" Dynamics of the Reference Vectors

In this section we clarify the name neural-gas and give a quantitative expression for the density distribution of the reference vectors.

We define the density $\varrho(\boldsymbol{u})$ of reference vectors at location $\boldsymbol{u}$ of $V \subseteq \mathcal{R}^D$ through $\varrho(\boldsymbol{u}) = F_{i(\boldsymbol{u})}^{-1}$ with $i(\boldsymbol{u})$ being the reference vector closest to $\boldsymbol{u}$ and $F_{i(\boldsymbol{u})}$ being the volume of Voronoi polygon $V_{i(\boldsymbol{u})}$. According to the definition of V_i, which was given in (1), $\boldsymbol{u} \in V_{i(\boldsymbol{u})}$ is valid. Hence, $\varrho(\boldsymbol{u})$ is a step function that is constant on each Voronoi polygon V_i. In the following, we study the case where the Voronoi polygons change their size F_i slowly from one Voronoi polygon to the next. Then we can regard $\varrho(\boldsymbol{u})$ as being continuous, which allows to derive an expression for $\varrho(\boldsymbol{u})$'s dependence on the density distribution of data points.

For a given $\boldsymbol{v}$, the density distribution $\varrho(\boldsymbol{u})$ determines the numbers $k_i(\boldsymbol{v}, \boldsymbol{w}), i = 1, \cdots, N$, which are necessary for an adjustment of the reference vectors $\boldsymbol{w}_i$. $k_i(\boldsymbol{v}, \boldsymbol{w})$ is the number of reference vectors within a sphere centered at $\boldsymbol{v}$ with radius $||\boldsymbol{v} - \boldsymbol{w}_i||$, i.e.,

$$k_i(\boldsymbol{v}, \boldsymbol{w}) = \int_{||\boldsymbol{u}|| < ||\boldsymbol{v} - \boldsymbol{w}_i||} \varrho(\boldsymbol{v} + \boldsymbol{u})\, d^D u. \tag{9}$$

In the following, we look at the average change $\langle \Delta \boldsymbol{w}_i \rangle$ of a reference vector with an adaptation step (7), given through

$$\langle \Delta \boldsymbol{w}_i \rangle = \epsilon \int d^D v\, P(\boldsymbol{v}) h_\lambda(k_i(\boldsymbol{v}, \boldsymbol{w}))(\boldsymbol{v} - \boldsymbol{w}_i). \tag{10}$$

In case of a small decay constant λ, i.e., a λ for which the range of $h_\lambda(k_i(\boldsymbol{v}, \boldsymbol{w}))$ is small compared to the curvature of $P(\boldsymbol{u})$ and $\varrho(\boldsymbol{u})$, we may expand the integrand of (10) around $\boldsymbol{w}_i$ since only the data points $\boldsymbol{v}$ for which $||\boldsymbol{v} - \boldsymbol{w}_i||$ is a small contribute to the integral. If, as in the simulations described previously, λ decreases against zero with the number of adaptation steps, the range of $h_\lambda(k_i(\boldsymbol{v}, \boldsymbol{w}))$ will always become that small at some point during the adaptation process.

The expansion of the integrand together with (9) yields to the leading order in λ

$$\langle \Delta \boldsymbol{w}_i \rangle \propto \frac{1}{\varrho^{1+2/D}} \left(\partial_{\boldsymbol{u}} P(\boldsymbol{u}) - \frac{2+D}{D} \frac{P}{\varrho} \partial_{\boldsymbol{u}} \varrho(\boldsymbol{u}) \right). \tag{11}$$

$\partial_{\boldsymbol{u}}$ denotes the gradient with respect to the coordinates of the data space. Equation (11) states that the average change of a reference vector $\boldsymbol{w}_i$ at location $\boldsymbol{u}$ is determined by two terms, one which is proportional to $\partial_{\boldsymbol{u}} P(\boldsymbol{u})$ at $\boldsymbol{u}$, and one which is proportional to $\partial_{\boldsymbol{u}} \varrho(\boldsymbol{u})$ at $\boldsymbol{u}$. The derivation of (11) is provided in Appendix II.

Equation (11) suggests the name neural-*gas* for the algorithm introduced. The average change of the reference vectors corresponds to an overdamped motion of particles in a potential $V(\boldsymbol{u})$ that is given by the negative data point density, i.e., $V(\boldsymbol{u}) = -P(\boldsymbol{u})$. Superimposed on the gradient of this potential is a "force" proportional to $-\partial_{\boldsymbol{u}} \varrho$, which points toward the direction of the space where the particle density $\varrho(\boldsymbol{u})$ is low. This "force" is the result of a repulsive coupling between the particles (reference vectors). In its form it resembles an entropic force and tends to homogeneously distribute the particles (reference vectors) over the input space, like in case of a diffusing gas.

The stationary solution of (11), i.e., the solution of

$$\partial_{\boldsymbol{u}} P - \frac{2+D}{D} \frac{P}{\varrho} \partial_{\boldsymbol{u}} \varrho = 0$$

is given by

$$\varrho(\boldsymbol{u}) \propto P(\boldsymbol{u})^\gamma \tag{12}$$

with

$$\gamma = \frac{D}{D+2}. \tag{13}$$

This relation describes the asymptotic density distribution of the reference vectors $\boldsymbol{w}_i$ and states that the density $\varrho(\boldsymbol{u})$ of reference vectors at location $\boldsymbol{u}$ is nonlinearly proportional to the density of data points $P(\boldsymbol{u})$. An asymptotic density distribution of the reference vectors that is proportional to $P(\boldsymbol{u})^\gamma$ with $\gamma = D/(D+2)$ is optimal for the task of minimizing the average quadratic distortion error (2) [29].

We tested (13) for a one-dimensional data distribution, i.e., for $D = 1$. For this purpose we chose a data density distribution of the form $P(u) = 2u, u \in [0, 1]$ and $N = 50$ reference vectors. The initial values for $w_i \in \mathcal{R}$ were drawn randomly from the interval [0, 1]. For the parameters ϵ and λ we chose the small but finite values $\epsilon = 0.01$ and $\lambda = 2$, which were kept constant during a subsequent performance of 5 000 000 adaptation steps (7). A double-logarithmic evaluation of the final result, i.e., of the 50 data pairs $x_i = w_i, y_i = \varrho(w_i) = 2/(w_{i+1} - w_{i-1}), i = 1, \cdots, 50$, yielded $\gamma = 0.323$, which compares well to the theoretical value $\gamma = 0.333$ given by (13) for $D = 1$.

V. On the Complexity of the Neural-Gas Network

The computationally expensive part of an adaptation step of the neural-gas algorithm is the determination of the "neighborhood-ranking," i.e., of the $k_i, i = 1, \cdots, N$. In a parallel implementation of the neural-gas network, each reference vector $\boldsymbol{w}_i$ can be assigned to a computational unit i. To determine its k_i, each unit i has to compare the distance $||\boldsymbol{v} - \boldsymbol{w}_i||$ of its reference vector to the input $\boldsymbol{v}$ with the distance $||\boldsymbol{v} - \boldsymbol{w}_j||$ of all the other units $j, j = 1, \cdots, N$. If each unit performs this comparison in a parallelized way, each unit i needs $\mathcal{O}(\log N)$ time steps to determine its "neighborhood-rank" k_i. In a subsequent time step, each computational unit adjusts its $\boldsymbol{w}_i$ according to equation (7). Hence, in a parallel implementation the computation time required for an adaptation step of the neural-gas algorithm increases like $\log N$ with the number N of reference vectors.

A scaling like $\log N$ is an interesting result since the computation time for an adaptation step of a "winner-take-all" network like the K-means clustering algorithm, which requires much less computation because only the "winning" unit has to be determined, also scales like $\log N$ in a parallel implementation. In a serial implementation, of course, the computation

time required for an adaptation step of the neural-gas algorithm increases faster with N than the corresponding computation time for a step of the K-means clustering. Determining the $k_i, i = 1, \cdots, N$ in a serial implementation corresponds to sorting the distances $||\boldsymbol{v} - \boldsymbol{w}_i||, i = 1, \cdots, N$, which scales like $N \log N$. Searching for the smallest distance $||\boldsymbol{v} - \boldsymbol{w}_{i_0}||$ to perform a step of the K-means clustering scales only linearly with the number of reference vectors.

VI. Application to Time-Series Prediction

A very interesting learning problem is the prediction of a deterministic, but chaotic, time-series, that we want to take as an application example of the neural-gas network. The particular time series we choose is the time-series generated by the Mackey–Glass equation [30]. The prediction task requires to learn an input–output mapping $y = f(\boldsymbol{v})$ of a current state $\boldsymbol{v}$ of the time-series (a vector of D consecutive time-series values) into a prediction of a future time-series value y. If one chooses D large enough, i.e., $D = 4$ in the case of the Mackey–Glass equation, the D-dimensional state vectors $\boldsymbol{v}$ all lie within a limited part of the D-dimensional space and form the attractor V of the Mackey–Glass equation. In order to approximate the input–output relation $y = f(\boldsymbol{v})$ we partition the attractor's domain into N smaller subregions $V_i, i = 1, \cdots, N$ and complete the mapping by choosing local linear mappings to generate the output values y on each subregion V_i. To achieve optimal results by this approach, the partitioning of V into subregions has to be optimized by choosing V_i 's the overall size of which is as small as possible.

A way of partitioning V is to employ a vector quantization procedure and take the resulting Voronoi polygons as subregions V_i. To break up the domain region of an input–output relation by employing a vector quantization procedure and to approximate the input–output relation by local linear mappings was suggested in [12]. Based on Kohonen's feature map algorithm, this approach has been applied successfully to various robot control tasks [12], [16]–[18] and has also been applied to the task of predicting time series [31]. However, as we have shown in Section III, for intricately structured input manifolds the Kohonen algorithm leads to suboptimal partitionings and, therefore, provides only suboptimal approximations of the input-output relation $y = f(\boldsymbol{v})$. The attractor of the Mackey–Glass equation forms such a manifold. Its topology and fractal dimension of 2.1 for the parameters chosen makes it impossible to specify a corresponding lattice structure. For this reason, we employ the neural-gas network for partitioning the input space V, which allows to achieve good or even optimal subregions V_i, also in the case of topologically intricately structured input spaces.

A hybrid approximation procedure that also uses a vector quantization technique for preprocessing the input domain to obtain a convenient coding for generating output values has been suggested by Moody and Darken [27]. In their approach, preprocessing the input signals, for which they used the K-means clustering, serves the task of distributing the centers $\boldsymbol{w}_i$ of a set of radial basis functions, i.e., Gaussian's, over the input domain. The approximation of the input–output relation is then achieved through superpositions of the Gaussians. Moody and Darken demonstrated the performance of their approach also for the problem of predicting the Mackey–Glass time-series. A comparison of their result with the performance we achieve with the neural-gas network combined with local linear mappings is given below.

A. Adaptive Local Linear Mappings

The task is to adaptively approximate the function $y = f(\boldsymbol{v})$ with $\boldsymbol{v} \in V \subseteq \mathcal{R}^D$ and $y \in \mathcal{R}$. V denotes the function's domain region. Our network consists of N computational units, each containing a reference or weight vector $\boldsymbol{w}_i$ (for the neural-gas algorithm) together with a constant y_i and a D-dimensional vector $\boldsymbol{a}_i$. The neural-gas procedure assigns each unit i to a subregion V_i as defined in (1), and the coefficiants y_i and $\boldsymbol{a}_i$ define a linear mapping

$$\tilde{y} = y_i + \boldsymbol{a}_i \cdot (\boldsymbol{v} - \boldsymbol{w}_i) \tag{14}$$

from $\mathcal{R}^D$ to $\mathcal{R}$ over each of the Voronoi polyhedra V_i. Hence, the function $y = f(\boldsymbol{v})$ is approximated by $\tilde{y} = \tilde{f}(\boldsymbol{v})$ with

$$\tilde{f}(\boldsymbol{v}) = y_{i(\boldsymbol{v})} + \boldsymbol{a}_{i(\boldsymbol{v})} \cdot (\boldsymbol{v} - \boldsymbol{w}_{i(\boldsymbol{v})}). \tag{15}$$

$i(\boldsymbol{v})$ denotes the computational unit i with its $\boldsymbol{w}_i$ closest to $\boldsymbol{v}$.

To learn the input–output mapping we perform a series of training steps by presenting input–output pairs $(\boldsymbol{v}, y = f(\boldsymbol{v}))$. The reference vectors $\boldsymbol{w}_i$ are adjusted according to adaptation step (7) of the neural-gas algorithm. To obtain adaptation rules for the output coefficients y_i and $\boldsymbol{a}_i$, for each i we require the mean squared error $\int_{V_i} d^D v P(\boldsymbol{v})(f(\boldsymbol{v}) - \tilde{f}(\boldsymbol{v}))^2$ between the actual and the required output, averaged over subregion V_i, to be minimal. A gradient descent with respect to y_i and $\boldsymbol{a}_i$ yields

$$\begin{aligned}\Delta y_i &= \epsilon' \int_{V_i} d^D v\, P(\boldsymbol{v})(y - y_i - \boldsymbol{a}_i \cdot (\boldsymbol{v} - \boldsymbol{w}_i)) \\ &= \epsilon' \int_{V} d^D v\, P(\boldsymbol{v}) \delta_{ii(\boldsymbol{v})}(y - y_i - \boldsymbol{a}_i \cdot (\boldsymbol{v} - \boldsymbol{w}_i))\end{aligned} \tag{16}$$

and

$$\begin{aligned}\Delta \boldsymbol{a}_i &= \epsilon' \int_{V_i} d^D v\, P(\boldsymbol{v})(y - y_i - \boldsymbol{a}_i \cdot (\boldsymbol{v} - \boldsymbol{w}_i)) \cdot (\boldsymbol{v} - \boldsymbol{w}_i) \\ &= \epsilon' \int_{V} d^D v\, P(\boldsymbol{v}) \delta_{ii(\boldsymbol{v})}(y - y_i - \boldsymbol{a}_i \cdot (\boldsymbol{v} - \boldsymbol{w}_i)) \\ &\quad \cdot (\boldsymbol{v} - \boldsymbol{w}_i).\end{aligned} \tag{17}$$

For $\lambda \to 0$ in adaptation step (7) the neural-gas algorithm provides an equilibrium distribution of the $\boldsymbol{w}_i$'s for which $\int_{V_i} d^D v P(\boldsymbol{v})(\boldsymbol{v} - \boldsymbol{w}_i) = 0$ for each i, i.e., $\boldsymbol{w}_i$ denotes the center of gravity of the subregion V_i. Hence, $\int_{V_i} d^D v P(\boldsymbol{v}) \boldsymbol{a}_i (\boldsymbol{v} - \boldsymbol{w}_i)$ in (16) vanishes and the adaptation step for the y_i's takes on a form similar to the adaptation step of the $\boldsymbol{w}_i$'s, except that only the output of the "winner" unit is adjusted with a training step. To obtain a significantly faster convergence of the output weights y_i and $\boldsymbol{a}_i$, we replace $\delta_{ii(\boldsymbol{v})}$ in (16) and (17) by $h_{\lambda'}(k_i(\boldsymbol{v}, \boldsymbol{w}))$, which has the same form as $h_\lambda(k_i(\boldsymbol{v}, \boldsymbol{w}))$ in adaptation step (7), except that the decay constant λ' might be of a different value than λ. By this replacement we achieve that the y_i and $\boldsymbol{a}_i$ of *each* unit is updated in a training step,

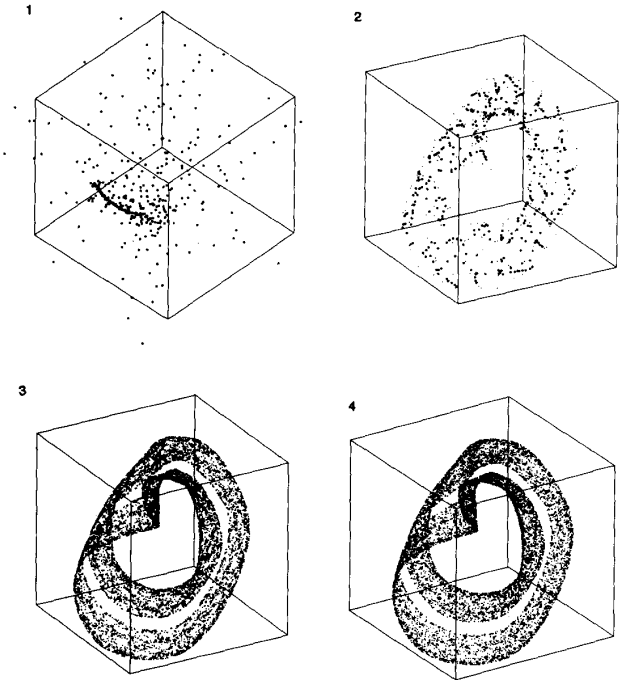

Fig. 3. The temporal evolution of the neural-gas while adapting to a representation of the Mackey–Glass attractor. Presented is a three-dimensional projection of the distribution of the reference vectors (big dots) at four different stages of the adaptation procedure, starting with a very early state at the top left part of this figure. The final state is shown at the bottom right. The small dots depict already presented inputs $\boldsymbol{v}$, forming the attractor which has to be partitioned.

with a step size that decreases with the unit's "neighborhood-rank" to the current input $\boldsymbol{v}$. In the beginning of the training procedure, λ' is large and the range of input signals that affect the weights of a unit is large. As the number of training steps increases, λ' decreases to zero and the fine tuning of the output weights to the local shape of $f(\boldsymbol{v})$ takes place. Hence, in their on-line formulation, the adaptation steps we use for adjusting y_i and $\boldsymbol{a}_i$ are given by

$$\begin{aligned}\Delta y_i &= \epsilon' \cdot h_{\lambda'}(k_i(\boldsymbol{v},\boldsymbol{w})) \cdot (y - y_i)\\ \Delta \boldsymbol{a}_i &= \epsilon' \cdot h_{\lambda'}(k_i(\boldsymbol{v},\boldsymbol{w})) \cdot (y - y_i - \boldsymbol{a}_i \cdot (\boldsymbol{v} - \boldsymbol{w}_i)) \cdot (\boldsymbol{v} - \boldsymbol{w}_i).\end{aligned} \tag{18}$$

B. Prediction of the Mackey–Glass Time Series

The time series we want to predict with our network algorithm is generated by the Mackey–Glass equation

$$\dot{x}(t) = \beta x(t) + \frac{\alpha x(t-\tau)}{1 + x(t-\tau)^{10}}$$

with parameters $\alpha = 0.2, \beta = -0.1$, and $\tau = 17$ [30]. $x(t)$ is quasi-periodic and chaotic with a fractal attractor dimension 2.1 for the parameter values we chose. The characteristic time constant of $x(t)$ is $t_{\text{char}} = 50$, which makes it particularly difficult to forecast $x(t + \Delta t)$ with $\Delta t > 50$.

Input $\boldsymbol{v}$ of our network algorithm consists of four past values of $x(t)$, i.e.,

$$\boldsymbol{v} = (x(t), x(t-6), x(t-12), x(t-18)).$$

Embedding a set of time-series values in a state vector is common to several approaches, including those of Moody and Darken [27], Lapedes and Farber [32], and Sugihara and May [33]. The time span we want to forecast into the future is $\Delta t = 90$. For that purpose we iteratively predict $x(t+6), x(t+12)$, etc., until we have reached $x(t+90)$ after 15 of such iterations. Because of this iterative forecasting, the output y which corresponds to $\boldsymbol{v}$ and which is used for training the network is the true value of $x(t+6)$.

We studied several different training procedures. First, we trained several networks of different sizes using 100 000 to 200 000 training steps and 100 000 training pairs $\boldsymbol{v} = (x(t), x(t-6), x(t-12), x(t-18)), y = x(t+6)$. One could deem this training as "on-line" because of the abundant supply of data. Fig. 3 presents the temporal evolution of the neural-gas adapting to a representation of the Mackey–Glass attractor. We show a three-dimensional projection of the four-dimensional input space. The initialization of the reference vectors, presented in the top left part of Fig. 3, is random. After 500 training steps, the reference vectors have "contracted" coarsely to the relevant part of the input space (top right). With further training steps (20 000, bottom left), the network assumes the general shape of the attractor, and at the end of the adaptation procedure (100 000 training steps, bottom right), the reference vectors are distributed homogeneously over the Mackey–Glass attractor. The small dots depict already-presented inputs $\boldsymbol{v}$, whose distribution is given by the shape of the attractor.

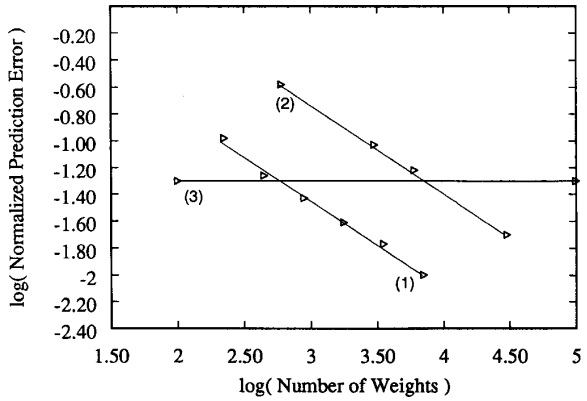

Fig. 4. The normalized prediction error versus the size of the network. The neural-gas algorithm combined with local linear mappings (1) compares well against Moody and Darken's K-means RBF method (2) and against back-propagation (3). To obtain the same prediction error, the K-means RBF method requires about 10 times more weights than the neural-gas algorithm with local linear mappings.

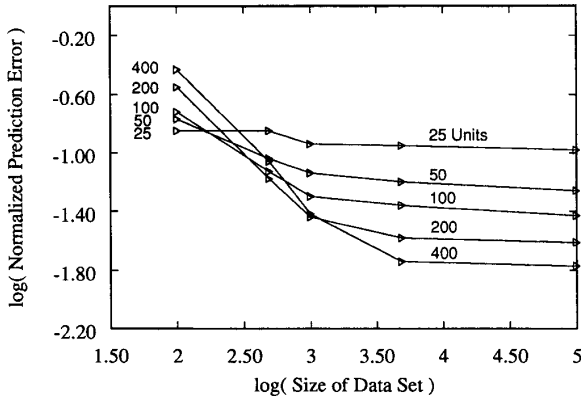

Fig. 5. The normalized prediction error versus training set size for different network sizes. Due to the effect of overfitting, small networks achieve a better performance than large networks if the training set of data points is small. With increasingly large amounts of data, the prediction error for the different network sizes approaches its lower bound.

Fig. 4 shows the normalized prediction error as a function of network size. The size of a network is the number of its weights, with nine weights per computational unit (four for each $\boldsymbol{w}_i$ and $\boldsymbol{a}_i$, plus one for y_i). The prediction error is determined by the rms value of the absolute prediction error for $\Delta t = 90$, divided by the standard deviation of $x(t)$. As we can see in Fig. 4, compared to the results Moody and Darken obtained with K-means clustering plus radial basis functions [27], the neural-gas network combined with local linear mappings requires about 10 times fewer weights to achieve the same prediction error. The horizontal line in Fig. 4 shows the prediction error obtained by Lapedes and Farber with the back-propagation algorithm [32]. Lapedes and Farber tested only one network size. On a set of only 500 data points they achieved a normalized prediction error of about 0.05. However, their learning time was on the order of an hour on a Cray X-MP. For comparison, we trained a network using 1000 data points and obtained the same prediction error of 0.05, however, training took only 90 s on a Silicon Graphics IRIS, which achieves 4MFlops for LINPACK benchmarks. To achieve comparable results, Moody and Darken employed about 13 000 data points, which required 1800 s at 90 KFLops. Hence, compared to our learning procedure, Moody and Darken's approach requires a much larger data set but is twice as fast. However, because of possible variations in operating systems and other conditions, both speeds can be considered comparable.

Fig. 5 shows the results of a study of our algorithm's performance in an "off-line" or scarce data environment. We trained networks of various sizes through 200 epochs (or 200 000 steps, whichever is smaller) on different sizes of training sets. Due to the effect of overfitting, small networks achieve a better performance than large networks if the training set of data points is small. With increasingly large amounts of data the prediction error for the different network sizes saturates and approaches its lower bound.

As in the simulations described in Section III, the parameters $\epsilon, \lambda, \epsilon'$, and λ' had the time dependence $x = x_i(x_f/x_i)^{t/t_{\max}}$ with t as the current and $t_{\max}$ as the total number of training steps. The initial and final values for the simulation parameters were $\epsilon_i = 0.99, \epsilon_f = 0.001; \lambda_i = N/3, \lambda_f = 0.0001; \epsilon'_1 = 0.5, \epsilon'_f = 0.05; \lambda'_i = N/6$, and $\lambda'_f = 0.05$. As in the simulations of Section III, the particular choice for these parameter values is not very critical and was optimized by trial and error. Both $h_\lambda(k)$ and $h_{\lambda'}(k)$ decreased exponentially with k.

VII. Discussion

In this paper we presented a novel approach to the task of minimizing the distortion error of vector quantization coding. The goal was to present an approach that does not require any prior knowledge about the set of data points and, at the same time, converges quickly to optimal or at least near optimal distortion errors. The adaptation rule we employed is a soft-max version of the K-means clustering algorithm and resembles to a certain extent both the adaptation rule of the maximum-entropy clustering and Kohonen's feature map algorithm. However, compared to the maximum-entropy clustering, it is a distance ranking instead of the absolute distance of the reference vectors to the current data vector that determines the adaptation step. Compared to Kohonen's feature map algorithm, it is not the neighborhood ranking of the reference vectors within an external lattice but the neighborhood-ranking within the input space that is taken into account.

We compared the performance of the neural-gas approach with K -means clustering, maximum-entropy clustering, and Kohonen's feature map algorithm on a model data distribution which consisted of a number of separated data clusters. On the model data distribution we found that 1) the neural-gas algorithm converges faster and 2) reaches smaller distortion errors than the three other clustering procedures. The price for the faster convergence to smaller distortion errors, however,

is a higher computational effort. In a serial implementation the computation time of the neural-gas algorithm scales like $N \log N$ with the number N of reference vectors, whereas the three other clustering procedures all scale only linearly with N. Nonetheless, in a highly parallel implementation the computation time required for the neural-gas algorithm becomes the same as for the three other approaches, namely $\mathcal{O}(\log N)$.

We showed that, in contrast to Kohonen's feature map algorithm, the neural-gas algorithm minimizes a global cost function. The shape of the cost function depends on the neighborhood parameter λ, which determines the range of the global adaptation of the reference vectors. The form of the cost function relates the neural-gas algorithm to fuzzy clustering, with an assignment degree of a data point to a reference vector that depends on the reference vector's neighborhood rank to this data point. Through an analysis of the average change of the reference vectors for small but finite λ, we could demonstrate that the dynamics of the neural-gas network resembles the dynamics of a set of particles diffusing in a potential. The potential is given by the negative density distribution of the data points, which leads to a higher density of reference vectors in those regions where the data point density is high. A quantitative relation between the density of reference vectors and the density of data points could be derived.

To demonstrate the performance of the neural-gas algorithm, we chose the problem of predicting the chaotic time-series generated by the Mackey–Glass equation. The "neural-gas" network had to form an efficient representation of the underlying attractor, which has a fractal dimension of 2.1. The representation (as a discretization of the relevant parts of the input space) was utilized to learn the required output, i.e., a forecast of the time-series, by using local linear mappings. A comparison with the performance of K-means clustering combined with radial basis functions showed that the neural-gas network requires an order of magnitude fewer weights to achieve the same prediction error. Also, the generalization capabilities of the neural-gas algorithm combined with local linear mappings compared favorably with the generalization capabilities of the RBF-approach. To achieve identical accuracy, the RBF-approach requires a training data set that is larger by an order of magnitude than the training data set which is sufficient for a neural-gas network with local linear mappings.

APPENDIX I

We show that the dynamics of the neural-gas network, described by adaptation step (7), corresponds to a stochastic gradient descent on a potential function. For this purpose we prove the following:

Theorem: For a set of reference vectors $\boldsymbol{w} = (\boldsymbol{w}_1, \cdots, \boldsymbol{w}_N)$, $\boldsymbol{w}_i \in \mathcal{R}^D$ and a density distribution $P(\boldsymbol{v})$ of data points $\boldsymbol{v} \in \mathcal{R}^D$ over the input space $V \subseteq \mathcal{R}^D$, the relation

$$\int_V d^D v\, P(\boldsymbol{v}) h_\lambda(k_i(\boldsymbol{v}, \boldsymbol{w}))(\boldsymbol{v} - \boldsymbol{w}_i) = -\frac{\partial E}{\partial \boldsymbol{w}_i} \tag{19}$$

with

$$E = \frac{1}{2} \sum_{j=1}^{N} \int_V d^D v\, P(\boldsymbol{v}) h_\lambda(k_j(\boldsymbol{v}, \boldsymbol{w}))(\boldsymbol{v} - \boldsymbol{w}_j)^2 \tag{20}$$

is valid. $k_j(\boldsymbol{v}, \boldsymbol{w})$ denotes the number of reference vectors $\boldsymbol{w}_l$ with $||\boldsymbol{v} - \boldsymbol{w}_l|| < ||\boldsymbol{v} - \boldsymbol{w}_j||$.

Proof: For notational convenience we define $\boldsymbol{d}_i(\boldsymbol{v}) = \boldsymbol{v} - \boldsymbol{w}_i$. This yields

$$-\frac{\partial E}{\partial \boldsymbol{w}_i} = R_i + \int_V d^D v\, P(\boldsymbol{v}) h_\lambda(k_i(\boldsymbol{v}, \boldsymbol{w}))(\boldsymbol{v} - \boldsymbol{w}_i) \tag{21}$$

with

$$R_i = -\frac{1}{2} \sum_{j=1}^{N} \int_V d^D v P(\boldsymbol{v}) h'_\lambda(k_j(\boldsymbol{v}, \boldsymbol{w}))\, \boldsymbol{d}_j^2 \frac{\partial k_j(\boldsymbol{v}, \boldsymbol{w})}{\partial \boldsymbol{w}_i}. \tag{22}$$

$h'_\lambda(\cdot)$ denotes the derivative of $h_\lambda(\cdot)$. We have to show that R_i vanishes for each $i = 1, \cdots, N$.

For $k_j(\boldsymbol{v}, \boldsymbol{w})$ the relation

$$k_j(\boldsymbol{v}, \boldsymbol{w}) = \sum_{l=1}^{N} \theta(\boldsymbol{d}_j^2 - \boldsymbol{d}_l^2) \tag{23}$$

is valid, with $\theta(\cdot)$ as the Heavyside step function

$$\theta(x) = \begin{cases} 1, & \text{for } x > 0 \\ 0, & \text{for } x \leq 0. \end{cases}$$

The derivative of the Heavyside step function $\theta(x)$ is the delta distribution $\delta(x)$ with

$$\delta(x) = 0 \quad \text{for } x \neq 0$$

and

$$\int \delta(x)\, dx = 1.$$

This yields

$$\begin{aligned} R_i = &\int_V d^D v P(\boldsymbol{v}) h'_\lambda(k_i(\boldsymbol{v}, \boldsymbol{w}))\, \boldsymbol{d}_i^2\, \boldsymbol{d}_i \sum_{l=1}^{N} \delta(\boldsymbol{d}_i^2 - \boldsymbol{d}_l^2) \\ &- \sum_{j=1}^{N} \int_V d^D v P(\boldsymbol{v}) h'_\lambda(k_j(\boldsymbol{v}, \boldsymbol{w}))\, \boldsymbol{d}_j^2\, \boldsymbol{d}_i \delta(\boldsymbol{d}_j^2 - \boldsymbol{d}_i^2). \end{aligned} \tag{24}$$

Each of the N integrands in the second term of (24) is nonvanishing only for those $\boldsymbol{v}$'s for which $\boldsymbol{d}_j^2 = \boldsymbol{d}_i^2$ is valid, respectively. For these $\boldsymbol{v}$'s we can write

$$\begin{aligned} k_j(\boldsymbol{v}, \boldsymbol{w}) &= \sum_{l=1}^{N} \theta(\boldsymbol{d}_j^2 - \boldsymbol{d}_l^2) \\ &= \sum_{l=1}^{N} \theta(\boldsymbol{d}_i^2 - \boldsymbol{d}_l^2) \\ &= k_i(\boldsymbol{v}, \boldsymbol{w}), \end{aligned} \tag{25}$$

and, hence, we obtain

$$R_i = \int_V d^D v P(v) h'_\lambda(k_i(\boldsymbol{v},\boldsymbol{w}))\, \boldsymbol{d}_i^2\, \boldsymbol{d}_i \sum_{l=1}^{N} \delta(\boldsymbol{d}_i^2 - \boldsymbol{d}_l^2) - \int^V d^D v P(\boldsymbol{v}) h'_\lambda(k_i(\boldsymbol{v},\boldsymbol{w}))\, \boldsymbol{d}_i^2\, \boldsymbol{d}_i \sum_{j=1}^{N} \delta(\boldsymbol{d}_j^2 - \boldsymbol{d}_i^2). \tag{26}$$

Since $\delta(x) = \delta(-x)$ is valid, R_i vanishes for each $i = 1, \cdots, N$.

APPENDIX II

In the following we provide a derivation of (6). The average change $\langle \Delta \boldsymbol{w}_i \rangle$ of a reference vector with adaptation step (7) is given by

$$\langle \Delta \boldsymbol{w}_i \rangle = \epsilon \int_V d^D v\, P(\boldsymbol{v}) h_\lambda(k_i(\boldsymbol{v},\boldsymbol{w}))(\boldsymbol{v} - \boldsymbol{w}_i) \tag{27}$$

with $h_\lambda(k_i(\boldsymbol{v},\boldsymbol{w}))$ being a factor that determines the size of the adaptation step and that depends on the number k_i of reference vectors $\boldsymbol{w}_j$ being closer to $\boldsymbol{v}$ than $\boldsymbol{w}_i$. We assume $h_\lambda(k_i)$ to be unity for $k_i = 0$ and to decrease to zero for increasing k_i with a characteristic decay constant λ. We can express $k_i(\boldsymbol{v},\boldsymbol{w})$ by

$$k_i(\boldsymbol{v},\boldsymbol{w}) = \int_{||\boldsymbol{u}|| < ||\boldsymbol{v} - \boldsymbol{w}_i||} \varrho(\boldsymbol{v} + \boldsymbol{u})\, d^D u \tag{28}$$

with $\varrho(\boldsymbol{u})$ as the density distribution of the reference vectors in the input space $V = \mathcal{R}^D$.

For a given set $\boldsymbol{w} = (\boldsymbol{w}_1, \cdots, \boldsymbol{w}_N)$ of reference vectors, $k_i(\boldsymbol{v},\boldsymbol{w})$ depends only on $\boldsymbol{r} = \boldsymbol{v} - \boldsymbol{w}_i$ and, therefore, we introduce

$$\boldsymbol{x}(\boldsymbol{r}) = \hat{\boldsymbol{r}} \cdot k_i(\boldsymbol{r})^{1/D}. \tag{29}$$

In the following we assume the limit of a continuous distribution of reference vectors $\boldsymbol{w}_i$, i.e., we assume $\varrho(\boldsymbol{u})$ to be analytical and nonzero over the input space V. Then, since $\varrho(\boldsymbol{u}) > 0$, for a fixed direction $\hat{\boldsymbol{r}}, \boldsymbol{x}(\boldsymbol{r})$ increases strictly monotonically with $||\boldsymbol{r}||$ and, therefore, the inverse of $\boldsymbol{x}(\boldsymbol{r})$, denoted by $\boldsymbol{r}(\boldsymbol{x})$, exists. This yields for (27)

$$\langle \Delta \boldsymbol{w}_i \rangle = \epsilon \int_V P(\boldsymbol{w}_i + \boldsymbol{r}(\boldsymbol{x})) h_\lambda(x^D) \boldsymbol{r}(\boldsymbol{x}) J(\boldsymbol{x})\, d^D x \tag{30}$$

with $J(\boldsymbol{x}) = \det(\partial r_\mu / \partial x_\nu), \mu, \nu = 1, \cdots, D$ and $x = ||\boldsymbol{x}||$.

We assume $h_\lambda(k_i(\boldsymbol{v} - \boldsymbol{w}_i))$ to decrease rapidly to zero with increasing $||\boldsymbol{v} - \boldsymbol{w}_i||$, i.e., we assume the range of $h_\lambda(k_i(\boldsymbol{r}))$ within V to be small enough to be able to neglect any higher derivatives of $P(\boldsymbol{u})$ and $\varrho(\boldsymbol{u})$ within this range. Then we may replace $P(\boldsymbol{w}_i + \boldsymbol{r}(\boldsymbol{x}))$ and $J(\boldsymbol{x})$ in (30) by the first terms of their Taylor expansion around $\boldsymbol{x} = 0$, yielding

$$\langle \Delta \boldsymbol{w}_i \rangle = \epsilon \int h_\lambda(x^D) \left(P(\boldsymbol{w}_i) + x_\mu \frac{\partial P}{\partial x_\mu} + \cdots \right) \cdot \left(J(0) + x_\mu \frac{\partial J}{\partial x_\mu} + \cdots \right) \boldsymbol{r}(\boldsymbol{x})\, d^D x. \tag{31}$$

The Taylor expansion of $\boldsymbol{r}(\boldsymbol{x})$ can be obtained by determining the inverse of $\boldsymbol{x}(\boldsymbol{r})$ in (29) for small $||\boldsymbol{x}||$. For small $||\boldsymbol{r}||$ (it holds $||\boldsymbol{u}|| \leq ||\boldsymbol{r}||$ in (28)), the density $\varrho(\boldsymbol{v} + \boldsymbol{u})$ in (28) is given by

$$\varrho(\boldsymbol{v} + \boldsymbol{u}) = \varrho(\boldsymbol{w}_i + \boldsymbol{r} + \boldsymbol{u}) = \varrho(\boldsymbol{w}_i) + (\boldsymbol{r} + \boldsymbol{u}) \partial_{\boldsymbol{r}} \varrho(\boldsymbol{w}_i) + \mathcal{O}(r^2) \tag{32}$$

with $\partial_{\boldsymbol{r}} = \partial / \partial \boldsymbol{r}$. Together with (29), this yields

$$\boldsymbol{x}(\boldsymbol{r}) = \boldsymbol{r}(\tau_D \varrho(\boldsymbol{w}_i))^{1/D} \left(1 + \frac{\boldsymbol{r} \cdot \partial_{\boldsymbol{r}} \varrho(\boldsymbol{w}_i)}{\varrho(\boldsymbol{w}_i) D} + \mathcal{O}(r^2) \right) \tag{33}$$

with

$$\tau_D = \frac{\pi^{D/2}}{\Gamma\left(\frac{D}{2} + 1\right)} \tag{34}$$

as the volume of a sphere with radius one in dimension D. As the inverse of (33) we obtain for small $||\boldsymbol{x}||$

$$\boldsymbol{r}(\boldsymbol{x}) = \boldsymbol{x}(\tau_D \varrho)^{-1/D} \left(1 - (\tau_D \varrho)^{-1/D} \frac{\boldsymbol{x} \cdot \partial_{\boldsymbol{r}} \varrho}{\varrho D} + \mathcal{O}(x^2) \right) \tag{35}$$

which yields the first terms of the Taylor expansion of $\boldsymbol{r}(\boldsymbol{x})$ around $\boldsymbol{x} = 0$.

Because of (35) it holds

$$\frac{\partial r_\mu(\boldsymbol{x})}{\partial x_\nu} = \delta_{\mu\nu} (\tau_D \varrho)^{-1/D} \cdot \left(1 - (\tau_D \varrho)^{-1/D} \frac{\boldsymbol{x} \cdot \partial_{\boldsymbol{r}} \varrho}{\varrho D} - (\tau_D \varrho)^{-1/D} \frac{x_\mu}{\varrho D} \frac{\partial \varrho}{\partial r_\mu} \right) - (1 - \delta_{\mu\nu}) (\tau_D \varrho)^{-2/D} \frac{x_\mu}{\varrho D} \frac{\partial \varrho}{\partial r_\nu} + \mathcal{O}_{\mu\nu}(x^2) \tag{36}$$

and, therefore,

$$\frac{\partial P}{\partial \boldsymbol{x}} = \frac{\partial P}{\partial \boldsymbol{r}} \frac{\partial \boldsymbol{r}}{\partial \boldsymbol{x}} = (\tau_D \varrho)^{-1/D} \partial_{\boldsymbol{r}} P \tag{37}$$

is valid at $\boldsymbol{x} = 0$.

The off-diagonal terms in (36) contribute only in quadratic order to $J(\boldsymbol{x}) = \det(\partial r_\mu / \partial x_\nu)$. Considering the diagonal terms up to linear order in $\boldsymbol{x}$ yields

$$J(\boldsymbol{x}) = (\tau_D \varrho)^{-1} \left(1 - (\tau_D \varrho)^{-1/D} \left(1 + \frac{1}{D} \right) \frac{\boldsymbol{x} \cdot \partial_{\boldsymbol{r}} \varrho}{\varrho} \right) + \mathcal{O}(x^2) \tag{38}$$

and, therefore,

$$\frac{\partial J}{\partial \boldsymbol{x}} = -(\tau_D \varrho)^{-(1+1/D)} \left(1 + \frac{1}{D} \right) \frac{1}{\varrho} \partial_{\boldsymbol{r}} \varrho \tag{39}$$

is valid at $\boldsymbol{x} = 0$.

Taking into account (35), (37)–(39) we obtain, for (31),

$$\langle \Delta \boldsymbol{w}_i \rangle = \epsilon (\tau_D \varrho)^{-1/D} \int \boldsymbol{x} h_\lambda(x^D) (P(\boldsymbol{w}_i) + (\tau_D \varrho)^{-1/D} \boldsymbol{x} \cdot \partial_{\boldsymbol{r}} P + \cdots) \cdot \left((\tau_D \varrho)^{-1} - \left(1 + \frac{1}{D} \right) (\tau_D \varrho)^{-[1+(1/D)]} \frac{\boldsymbol{x} \cdot \partial_{\boldsymbol{r}} \varrho}{\varrho} + \cdots \right) \cdot \left(1 - (\tau_D \varrho)^{-1/D} \frac{\boldsymbol{x} \cdot \partial_{\boldsymbol{r}} \varrho}{\varrho D} + \cdots \right) d^D x. \tag{40}$$