To the Graduate Council:

I am submitting herewith a thesis written by Michael Thomas Cozart entitled "Evaluation of the 'Neural Gas' Network Vector Quantization and Approximation Components." I have examined the final copy of this thesis for form and content and recommend that it be accepted in partial fulfillment of the requirements for the degree of Master of Science, with a major in Computer Science.

Bruce Whitehead, Major Professor

We have read this thesis
and recommend its acceptance:

Accepted for the Council:

Associate Vice Chancellor
and Dean of the Graduate School

EVALUATION OF THE 'NEURAL GAS' NETWORK VECTOR QUANTIZATION AND APPROXIMATION COMPONENTS

A Thesis

Presented for the

Master of Science

Degree

The University of Tennessee, Knoxville

Michael Thomas Cozart

May 1996

Dedication

This thesis is dedicated to my family and friends
who provided endless encouragement and understanding.

Acknowledgments

I thank my advisor and thesis committee chairman, Dr. Bruce Whitehead, for the guidance and assistance he has provided. I also thank the other members of my committee, Dr. Dinesh Mehta and Dr. Al Pujol. The merit of this work is due in no small part to their suggestions.

Abstract

One neural networking design technique for the prediction of nonlinear time-series implements two primary computational components, vector quantization and function approximation. A "Neural Gas" network which uses this design technique has reportedly produced superior prediction results of time-series. This research evaluates the effectiveness or influence of each component of the Neural Gas network by substituting other reputable methods of vector quantization and approximation for those of the Neural Gas network. The substitute quantization component is the Generalized Learning Vector Quantizer. The substitute approximation component is Radial Basis Functions. Results of this research indicate the approximation component of the Neural Gas network contributes most to the excellent prediction results. The prediction results from the substituted components versus the Neural Gas components are discussed.

TABLE OF CONTENTS

Chapter Page

Introduction 1

Neural Network Prediction Review 4

2.1 Neural Networking 4

2.2 Chaotic Time-Series 6

2.3 Quantization 7

2.4 Approximation 17

2.5 Prediction 27

Methods 28

3.1 Approach to Evaluation 28

3.2 Substitute Components 28

3.3 Time-Series Data 29

3.4 Performing the Evaluation 30

Results and Conclusions 34

4.1 Results 34

4.2 Conclusions 38

List Of References 41

Appendix 44

Vita 51

LIST OF FIGURES

Figure Page

2.1. 3-layer neural network.....5

2.2. Mackey-Glass time-series.....7

2.3. Locally optimal clustering from k-means or SHCM clustering.....11

2.4. Ideal clustering example.....11

2.5. 3-dimensional gaussian surface.....18

2.6. Summation of three gaussian curves.....25

3.1. Prediction process.....31

4.1. *NPE* from each network.....35

4.2. *NPE* versus two Neural Gas experimental parameters.....36

4.3. Neural Gas Network and Benchmark *NPE*.....36

4.4. Neural Gas and GLVQ cluster centers.....38

LIST OF TABLES

Table Page

3.1. Recommended Neural Gas parameter values and ranges used. 32

4.1. Best *NPE* from each network. 34

4.2. Best quantization errors from each network. 37

4.3. Execution time (seconds) for each component. 37

LIST OF EQUATIONS

Equation	Page
(2.1)	7
(2.2)	8
(2.3)	9
(2.4)	10
(2.5)	12
(2.6)	13
(2.7)	13
(2.8)	14
(2.9)	14
(2.10)	14
(2.11)	14
(2.12)	15
(2.13)	16
(2.14)	18
(2.15)	19
(2.16)	20
(2.17)	20
(2.18)	20
(2.19)	21
(2.20)	22
(2.21)	22
(2.22)	23
(2.23)	23
(2.24)	23

(2.25) 24
(2.26) 26
(2.27) 26
(2.28) 27
(3.1) 30

LIST OF SYMBOLS

a	Mackey-Glass time-series constant
b	Neural Gas hyper-plane intercept
c	Variable in the gaussian function
d	Index to dimensions, D
e	Exponential function
f	subscript on variables noting the final value of a range
g	GLVQ data vector weighted mismatch on all clusters
h	Neural Gas function controlling neighborhood learning rate
i	Index on winning cluster center, $\boldsymbol{u}$, and subscript on variables noting the initial value of a range
j	Index on losing cluster centers, $\boldsymbol{u}$
k	Index on input data vectors
m	Neural Gas index on cluster center ranking function
n	Index on input data vectors closest to a cluster center
p	Pass number
q	Number of data vectors closest to a cluster center
r	Index on all cluster centers
s	Neural Gas hyper-plane slope component
$\boldsymbol{s}$	Neural Gas hyper-plane slope vector
t	index on time steps in Mackey-Glass time-series
$\boldsymbol{u}$	Cluster center
$\boldsymbol{u}^{old}$	Cluster center prior to adjustment
$\boldsymbol{u}^{new}$	Cluster center following adjustment
u	1-dimensional component of $\boldsymbol{u}$
$\boldsymbol{v}$	Input data vector from training data or test data

v 1-dimensional component of $\boldsymbol{v}$

w Weight to apply to RBF output

x Mackey-Glass time-series function

y Index to $\boldsymbol{Z}^{des}$

z Mackey-Glass time-series constant

A Neural Gas matrix of data points relative to cluster center

B Membership function for vector quantization input data versus cluster center

D Number of dimensions of input data vectors

E Error, general

E^{NG} Neural Gas error function

$E^{RMS\ quant}$ Root mean squared quantization error

NPE Normalized Prediction Error

F GLVQ input data vector mismatch on all cluster centers

J Neural Gas learning rate adjustment function

K Neural Gas cluster center ranking function

L GLVQ mismatch error function

M Number of clusters centers in the neural network

N Number of input data vectors in the training data or test data

P Number of passes through the input data

R Domain of real values

S SVD diagonal matrix of singular values

T Superscript noting matrix transpose

U SVD column orthogonal matrix

V Probability density model of input data

X Neural Gas hyper-plane

Y RBF function value matrix for each input data vector on each cluster center

$\boldsymbol{Z}^{des}$ array of desired neural network output

Z^{out}	array of actual neural network output
Σ	Iterative summation
Ω	Set of all cluster centers
α	Vector quantization learning rate
β	Factor applied to RBF width
ε	Neural Gas experimental learning rate
ϕ	Gaussian function
η	Neighborhood of cluster centers
λ	Neural Gas Neighborhood learning rate adjustment
σ	Width of RBF
τ	Mackey-Glass time-series constant
∇	Gradient or directional derivative

LIST OF ABBREVIATIONS

GLVQ	Generalized Learning Vector Quantizer
NG	Neural Gas
NPE	Normalized Prediction Error
RBF	Radial Basis Function
SVD	Singular Value Decomposition

CHAPTER 1

Introduction

Vector quantization is the attempt to partition a set of vectors into subsets, referred to as clusters, each of which is represented by a reference vector at the center. This clustering produces an approximation to the probability density function of the set of vectors. Two innovative vector quantization techniques have been presented recently by Martinetz, Berkovich and Schulten [1] and Pal, Bezdek and Tsao [2]. The authors of [1] present a "Neural Gas" network to quantize the data being modeled, so named because of the gas-like behavior of the quantization cluster centers. The cluster centers for the Neural Gas clusters move around in the vector space similar to the movement of gas molecules in a closed container. Pal et al. introduce a Generalized Learning Vector Quantizer (GLVQ) which is a variation of the Learning Vector Quantizer (LVQ) pioneered by Kohonen [4]. The new GLVQ algorithm makes more efficient use of computational resources yet appears to be at least as robust as the LVQ. These vector quantization techniques yield very favorable quantization results. A vector quantizer in a neural network provides a means of partitioning or decoding input data for further processing by the network. In this research the quantization results are used for the classic problem of approximation and prediction of the chaotic Mackey-Glass time-series, Mackey and Glass [3]. Prediction of the Mackey-Glass time-series function is a relevant application because of the extensive use of this function as a benchmark for neural network approximation algorithms.

Approximation of functions may be accomplished using the clusters resulting from quantization of sample data from those functions. Each cluster from the quantization represents a piece of the function being approximated. The function's sample data within each cluster is mapped to a local function which approximates the

data in the cluster, now referred to as a local processing unit. Collectively the local functions in these local processing units represent an approximation of the entire modeled function or at least the data sampled from that function.

Martinetz et al. [1] present one method to approximate functions which uses the results of the Neural Gas clustering network. Their approximation technique performs a linear mapping of the clustered model data from the Mackey-Glass time-series to best-fit planes. The planes are the local functions of the local processing units. Martinetz et al. have reported prediction results for the time-series which surpass that of other published research. They attribute those results to their quantization technique but provide no evidence to support this claim.

This research examines the contribution to the prediction results in Neural Gas networks from the clustering and approximation techniques. The clustering and approximation techniques are the two "components" of the neural networks studied in this research. Prediction results from Neural Gas networks are compared with the results from alternative networks. The alternative networks have other quantization and approximation components substituted for the Neural Gas components. The results of the comparison indicate the strength of the contribution of each Neural Gas component to the prediction results. The alternative to the Neural Gas quantization component is the GLVQ from [2]. This quantization component was chosen for its excellent quantization results, robustness, and computational efficiency. The alternative approximation component is Radial Basis Functions. Moody and Darken [5] introduced Radial Basis Functions (RBFs) employing Gaussian surfaces for the local functions as a neural network approximation technique. RBFs are a tried and proven approximation technique and as such provide a suitable alternative to the linear mapping in the local processing units from [1]. Essentially, this study compares the performance of two quantization components used with two distinct approximation components. The performance is evaluated in the application of approximation and

prediction of the Mackey-Glass time-series function. Comparing and evaluating the four sets of prediction results obtained from the testing of both quantization components with both approximation components is the basic approach of the investigation.

Evidence resulting from the research indicates that the approximation component contributes much more than the clustering method to the time-series prediction results in [1]. The Neural Gas network's small prediction errors reported in [1] could not be reproduced; however, low prediction errors were obtained from the network. Further improvement of the Neural Gas results should be possible by substitution or improvement of the quantization and approximation components of the network.

Chapter 2 provides background information on the neural network architecture presented in this research. Quantization and approximation techniques are discussed. Chapter 3 presents the methods for evaluating the Neural Gas network. In Chapter 4 the results and conclusions of the research are presented.

CHAPTER 2

Neural Network Prediction Review

2.1 Neural Networking

Neural networks are so named because of their design which is similar to the makeup of biological neural pathways. In fact neural networks are not only designed similarly to biological information processing systems, they also perform the same basic task. This task is processing input sensory data, arranging this data into some sort of meaningful order, then passing on information about the meaning of the input sensory data. Neural networks can be thought of as mimicking the basic tasks of the biological brain [6]. An example of this information processing in the brain is accepting input from senses such as hearing and interpreting the signals as meaningful words.

The basic architecture of a neural network consists of layers of interconnected processing units. These processing units, referred to as neurodes or processing elements, are the functional counterpart of biological neurons. The number of layers and the degree of interconnection between neurodes varies among different neural network designs. The interconnections between the neurodes represent the flow of information. Frequently the inner layers of a network are referred to as hidden layers since they neither receive input from nor output information outside of the network. Figure 2.1 represents the neural network design presented in this research. The neurodes are capable of storing information which determines their output for any given input. By convention the information consists of a table of weights where typically there is one weight for each interconnection. Neural networks which can modify this stored information without error correction from outside the network are referred to as self-organizing or unsupervised.

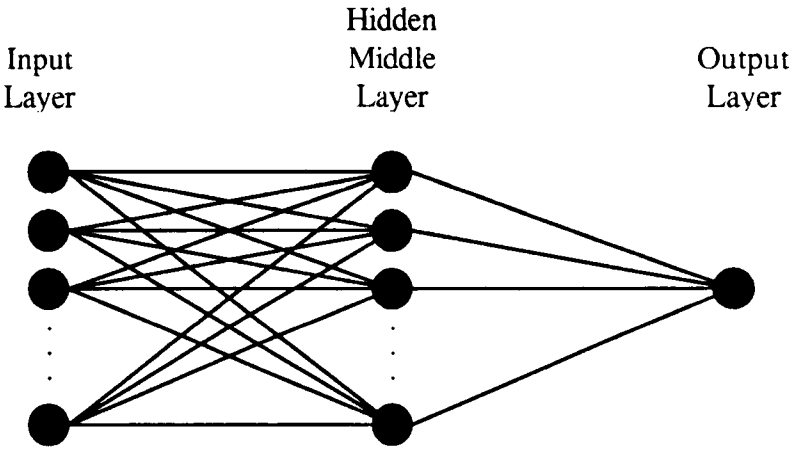


Figure 2.1. 3-layer neural network.

There are numerous and widely diversified practical applications of neural networks. These applications include robotic navigational and motor control [8], and speech [4] and image recognition [9]. In the speech and image recognition applications neural networks must adapt or organize themselves into processing systems capable of recognizing differing input patterns and classifying the input into categories. An example of this self-organization in the image recognition application is the classification of all electronically or mathematically encoded input patterns from scanned images of characters from some alphabet. The input patterns are classified into categories for each character of the alphabet. An extension of this classifying capability of neural networks would be a network which classifies data sampled from a mathematical function in order to provide an approximation of the function. This leads to another application of neural networks, approximation and prediction [10].

Approximation of a function by a neural network often is performed in two phases. These phases are termed the "training phase" and the "testing phase." In the training phase the neural network is provided a set of data samples from the function. This set of data is referred to as "training data." The network organizes itself into a system which provides output that approximates the function. Often this training phase

is a repetitive exposure of the network to the training data. The repetitions terminate when the output reaches some arbitrary level of approximation error or the error is no longer diminished with additional repetitions. In the testing phase the network is presented with a different set of data sampled from the same function. This set of data is referred to as "test data." An approximation of the function is then obtained using the organization established during the training phase. The approximation error is measured by comparing the approximation provided by the neural network to the original function itself. Obtaining the smallest approximation error possible is the goal of the process. In essence the neural network performs function interpolation. The training phase clusters or quantizes the input data then maps it to local functions. Each local function approximates the function only within its cluster. Collectively all the local functions represent an approximation to the entire modeled function. The test phase interpolates the value of the function for any input data point using the network's function approximation.

Neural networks may be applied to function prediction problems. A neural network performs function approximation [11] during the training phase, followed by function interpolation in the testing phase to arrive at a prediction. The approximation is obtained on one segment of a function during the training phase. Interpolation is then performed on a different segment of the function during the testing phase. One class of functions which are a natural choice for neural network prediction applications are chaotic time-series.

2.2 Chaotic Time-Series

One well-publicized chaotic time-series is derived from the Mackey-Glass Equation which was first presented as a model for human white blood cell count irregularities [3]. The Mackey-Glass equation is:

$$\frac{dx[t]}{dt} = -zx[t] + a\frac{x[t-\tau]}{1+x[t-\tau]^{10}} \tag{2.1}$$

The study of this and other related time-series equations in medical research has been aimed at predicting and developing treatment for medical dysfunctions. This differential equation typically generates the time-series training and test data in other related neural network research. It is used for benchmark purposes in this research with the constants $a = 0.2$, $z = 0.1$ and $\tau = 17.0$. Figure 2.2 shows a section of the time-series between $t = 0$ and $t = 1000$.

2.3 Quantization

Currently much work in prediction networks involves 3-layer architectures such as shown in Figure 2.1 and described in the works of [1, 5, 8,10]. The quantization

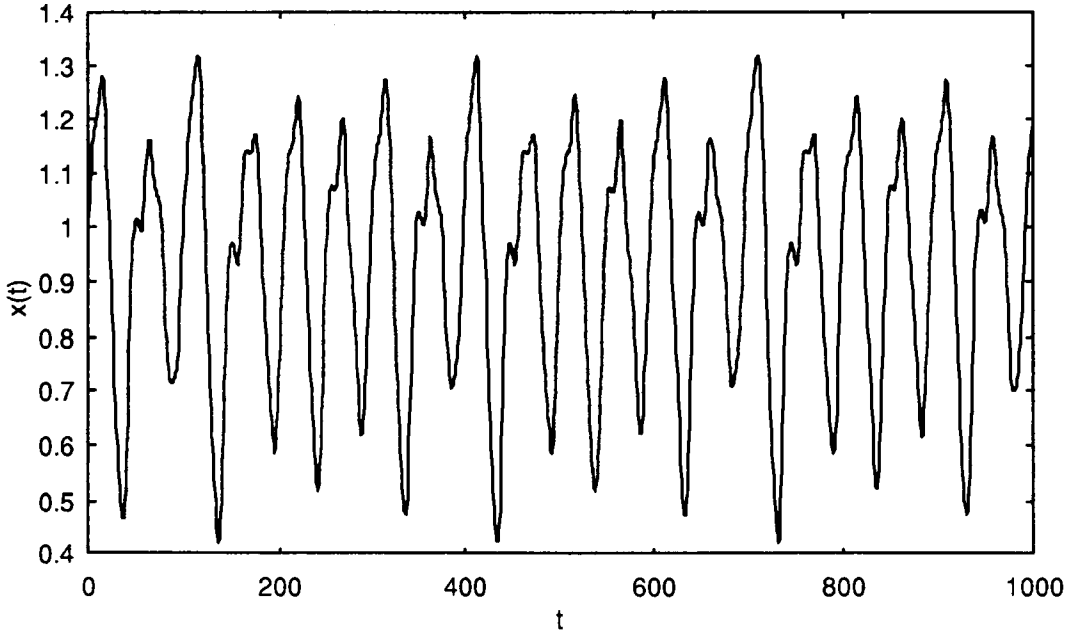


Figure 2.2. Mackey-Glass time-series.

performed by the neural network is accomplished in the hidden inner layer.

The vector quantization process represents the probability density model of the input data set with a smaller model which represents the features of the input data [1,4]. Consider the input data set V as a subspace of D-dimensional real vector space, i.e. $V \subseteq R^D$. The smaller model, $\Omega = (\boldsymbol{u}_1, \boldsymbol{u}_2, \ldots, \boldsymbol{u}_M)$, is composed of M cluster centers. $\boldsymbol{u}_i \subseteq R^D$, for $1 \le i \le M$ where D is the number of dimensions of the input data. Each neurode of the middle layer of the neural network has one cluster associated with it. Each data vector of the input data set $\boldsymbol{v} \in V$ is represented by the best matching cluster center. Typically the measure of a best match is the minimum euclidean distance between the input vector and the competing cluster centers. The euclidean distance between data vector $\boldsymbol{v}$ and cluster center $\boldsymbol{u}_i$ is represented by $\|\boldsymbol{v} - \boldsymbol{u}_i\| = \sqrt{(v_1 - u_{i1})^2 + (v_2 - u_{i2})^2 + \ldots + (v_D - u_{iD})^2}$. During the training process the neurodes are trained or "tuned" to represent subsets of the input data vectors. Cluster center tuning is accomplished by changing the values of cluster centers, u_{id}, for $1 \le i \le M$, $1 \le d \le D$. The average error $E^{RMS\ quant}$ for N input data vectors is obtained from Equation 2.2. B_{kr} represents a membership function that determines whether data vector k is best matched by cluster center r. This error is referred to as root mean squared quantization error.

$$E^{RMS\ quant} = \sqrt{\sum_{k=1,N} \sum_{r=1,M} B_{kr} [\boldsymbol{v}_k - \boldsymbol{u}_r]^2} \tag{2.2}$$

The quantization algorithm adjusts the cluster centers and attempts to find the minimum quantization error for all the data vectors in the training data set. The method by which the cluster centers are adjusted or "updated" during training varies among different vector quantization algorithms. A quantization process serves two purposes. The first purpose is to update the reference vectors representing cluster

centers of each neurode in the middle layer. This updating divides the input data into clusters. The second is to minimize quantization error. Historically, quantization is an iterative process, making multiple passes through the training data. Subsequent to clustering or self-organization is the approximation of the function which is discussed in the following section.

One of the simplest yet most widely employed methods of quantization is the *k*-means algorithm. The *k*-means algorithm updates the cluster centers in the following process. The cluster centers are initially distributed through the input data space. Frequently the distribution is random or the cluster centers are initialized with the first M training data vectors. The entire training data set is read into the neural network repetitively. These repetitions are referred to as "passes." Several passes through the training data are performed. For each input training data point $\boldsymbol{v}$ the nearest cluster center $\boldsymbol{u}$ to the data point is determined. Then the nearest cluster center is updated using Equation 2.3.

$$\boldsymbol{u}^{new} = \boldsymbol{u}^{old} + \alpha_p\left(\boldsymbol{v} - \boldsymbol{u}^{old}\right) \tag{2.3}$$

α_p is a scalar variable which decreases in magnitude for each pass p through the input data with $0 < \alpha_0 < 1$, $\alpha_p = \alpha_0(1 - p/P)$, $\boldsymbol{u}^{new}$ the new cluster center, $\boldsymbol{u}^{old}$ the previous cluster, and P the total number of passes to make through the data. The nearest cluster center's reference vector is rotated towards the data point and the amount of rotation is determined by α_p, the learning rate. Termination of the algorithm occurs when the desired quantization error is achieved, the quantization error remains unchanged after a pass, or the number of passes reaches P. The k-means algorithm represents one of the simplest forms of competitive unsupervised learning techniques in neural networks [4]. One may think of the cluster centers competing with one another to match the input data points.

Another clustering algorithm which is very similar to k-means is MacQueen's [2] sequential hard c-means (SHCM). SHCM differs from k-means by performing only one pass through the input data and by the cluster center update rule. Equation 2.4 describes the update rule for SHCM clustering. The update rule for SHCM uses $1/q_\mu$ in place of α_p where q_μ is the number of data points which have been found closest to cluster center $\boldsymbol{u}$. The update rule shown in Equation 2.4 is applied once for input data vector, $\boldsymbol{v}_k$, $1 < k < N$.

$$\boldsymbol{u}^{new} = \boldsymbol{u}^{old} + \frac{1}{q_u}\left(\boldsymbol{v}_k - \boldsymbol{u}^{old}\right)$$

(2.4)

Pal et al. [2] describes weaknesses common to k-means and SHCM clustering. In either algorithm it is possible to initialize cluster centers which never are a best match to any input data vector. These cluster centers are never updated, and remain unused. Another potential shortcoming is illustrated with Figure 2.3 where $\boldsymbol{u}_1$ is always the best match for $\boldsymbol{v}_1$ and only $\boldsymbol{v}_1$. Consequently only $\boldsymbol{w}_2$ is updated by the remaining input data resulting in a less than ideal solution. For the illustration in Figure 2.3 consider the training dataset $V = \{\boldsymbol{v}_1,\boldsymbol{v}_2,\boldsymbol{v}_3,\boldsymbol{v}_4,\boldsymbol{v}_5,\boldsymbol{v}_6\} \in R^2$ and the number of cluster centers, $\Omega = 2$. During the initialization process the reference vectors are initialized to be at or near $\boldsymbol{v}_1$ and $\boldsymbol{v}_2$. The quantization process terminates in a locally optimum solution as shown in Figure 2.3. Figure 2.4 depicts an ideal solution.

Algorithms which reduce the probability of locally optimal solutions are demonstrated by Kohonen [4]. These algorithms define a neighborhood of cluster centers, η_u, surrounding each cluster center, $\boldsymbol{u}$. A neighborhood quantization algorithm updates each cluster center within η_u whenever $\boldsymbol{u}$ is updated. Kohonen presents several methods to control the size of the neighborhood and update of neighboring cluster centers. The size or radius of η_u is an experimental parameter which can be adjusted

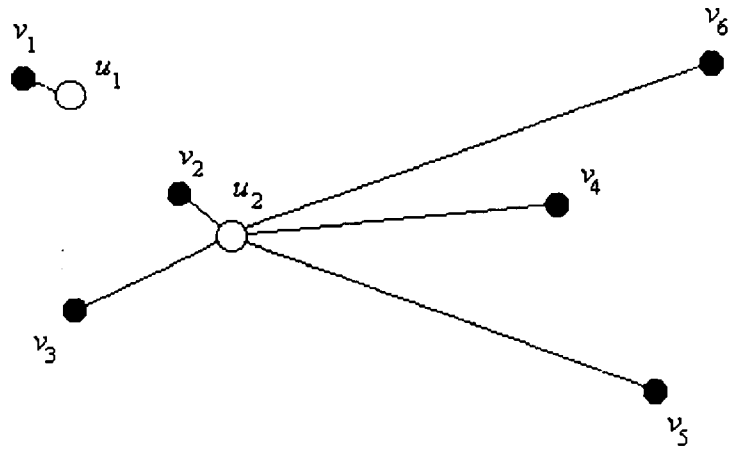


Figure 2.3. Locally optimal clustering from k-means or SHCM clustering.

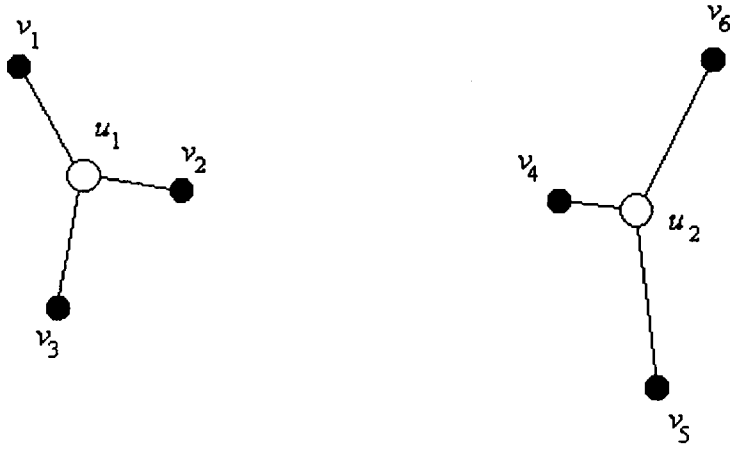


Figure 2.4. Ideal clustering example.

with each pass through the data. Typically it is best to have the radius of η_u large at the start and very small at the conclusion of clustering. Neighboring cluster centers are not necessarily updated with the full learning rate as is the best match. The members of η_u can be updated by some factor of the learning rate. Usually this factor is inversely proportional to the distance of the members from the center of the neighborhood or from the data point. Equation 2.5 represents a generalized form of a cluster center update rule for a neighborhood approach:

$$\boldsymbol{u}_r^{new} = \boldsymbol{u}_r^{old} + \alpha_p \lambda_{irp} \left(\boldsymbol{v} - \boldsymbol{u}_r^{old} \right) \tag{2.5}$$

where $\boldsymbol{u}_i$ is the best matching cluster center, $\boldsymbol{u}_r \in \eta(\boldsymbol{u}_i)$, and λ_{irp} is a learning rate adjustment function which decreases in value as $\|\boldsymbol{u}_i - \boldsymbol{u}_r\|$ increases and as p increases. This neighborhood approach requires the tuning of several experimental parameters:

1. Initial learning rate
2. Learning rate adjustment during training
3. Initial neighborhood size
4. Neighborhood size adjustment during training
5. Factor of learning rate to apply to neighboring cluster centers

Frequently these parameters are optimized by trial and error for each specific application of clustering. This parameter optimization process has proven to be a clumsy and time consuming process. Pal et al. [2] present a clustering algorithm with a neighborhood approach which eliminates the need to tune the neighborhood parameters.

Pal et al. [2] describe a generalized learning vector quantizer (GLVQ) which is a variation of Kohonen's neighborhood clustering algorithm. The GLVQ appears to eliminate the shortcomings of poor cluster center initialization. It also eliminates the need to tune neighborhood parameters, and yields excellent quantization results. The

GLVQ algorithm is based upon the formula for a mismatch error L_v of an input data point to all the cluster centers in the network. The mismatch error is weighted with respect to the best matching cluster center $\boldsymbol{u}_i$. Equation 2.6 shows the formula for the GLVQ mismatch error.

$$L_v = \sum_{r=1,M} g_{ir} \|v - u_r\|^2$$

$$g_{ir} = \begin{cases} 1 & , \text{if} \quad r = i \\ \dfrac{1}{\sum_{j=1,M} \|v - u_j\|^2} & , \text{otherwise} \end{cases}$$

(2.6)

To obtain formulas for the update of cluster centers the gradient of L_v is obtained by rewriting L_v:

$$\begin{aligned} L_v &= \sum_{r=1,M} g_{ir} \|v - u_r\|^2 \\ &= \|v - u_i\|^2 + \frac{\sum_{\substack{r=1,M \\ r \neq i}} \|v - u_r\|^2}{\sum_{j=1,M} \|v - u_j\|^2} \\ &= \|v - u_i\|^2 + \frac{\sum_{r=1,M} \|v - u_r\|^2}{\sum_{j=1,M} \|v - u_j\|^2} - \frac{\|v - u_i\|^2}{\sum_{j=1,M} \|v - u_j\|^2} \\ &= \|v - u_i\|^2 + 1 - \frac{\|v - u_i\|^2}{\sum_{j=1,M} \|v - u_j\|^2} \end{aligned}$$

(2.7)

followed by differentiation. Differentiating L_v with respect to $\boldsymbol{u}_i$ provides the directional derivative influenced by the best matching cluster center, $\boldsymbol{u}_i$:

$$\nabla L_v(v_i) = -2(v - u_i)\frac{F^2 - F + \|v - u_i\|^2}{F^2}$$

$$F = \sum_{r=1,M} \|v - u_r\|^2$$

(2.8)

Following the slope of $\nabla L_v(v_i)$ is the most direct route to a minimum of L_v by varying v_i. Therefore from Equation 2.8 an update rule for best matching cluster center is obtained:

$$u_i^{new} = u_i^{old} + \alpha_p\left(v - u_i^{old}\right)\frac{F^2 - F + \left\|v - u_i^{old}\right\|^2}{F^2}$$

(2.9)

Differentiating L_v with respect to u_j provides the directional derivative influenced by the other $M-1$ cluster centers:

$$\nabla L_v(v_j) = -2\left(v - u_j\right)\frac{\|v - u_i\|^2}{F^2}$$

(2.10)

From Equation 2.10 an update rule for the remaining cluster centers is obtained:

$$u_j^{new} = u_j^{old} + \alpha_p\left(v - u_j^{old}\right)\frac{\left\|v - u_i^{old}\right\|^2}{F^2}$$

(2.11)

In the update rules the constant, –2, has been replaced with the learning rate α_p. As is the case in the previously discussed quantization algorithms it is desirable to have an adjustable learning rate. A learning rate less than the maximum permits the network to retain part of the organization based on prior input data. The authors of [2] recommend the traditional scheme for α_p value adjustments. This scheme provides a

decrease in magnitude for each pass through the input data with $0 < \alpha_0 < 1$ and $\alpha_p = \alpha_0(1 - p/P)$ where P is the total number of passes through the training data.

The GLVQ has several desirable qualities. There is no need to define or adjust neighborhood parameters. The GLVQ cluster center update formulas provide that either all of the cluster centers are updated or none are updated. The update to non-best matching cluster centers is proportional to the mismatch error on the best matching cluster center. Fewer passes through the training data are required to achieve a comparably low quantization error as other neighborhood approaches. GLVQ performance is not affected by the choice of initial cluster centers.

Martinetz et al. [1] present another neural network clustering model which possesses many of the same desirable qualities as the GLVQ. In their Neural Gas algorithm the cluster center update rule is derived from the error function:

$$E^{NG}(\Omega,\lambda) = \frac{1}{2\sum_{r=1,M} h_\lambda(k_r)} \sum_{r=1,M} \sum_{k=1,N} h_\lambda\left[K_r(v_k,\Omega)\right] \left\| v_k - \mu_r \right\|^2$$

(2.12)

The Neural Gas error function is based upon a mismatch error of each cluster center with respect to the best matching cluster center as is the GLVQ algorithm. The GLVQ mismatch error is based entirely upon the euclidean distance between input data points and cluster centers. The Neural Gas mismatch error function uses a "ranking" of cluster centers. The function $K_r(v_k,\Omega)$ returns the ranking of cluster center u_r with respect to all the cluster centers ordered by proximity to data point v_k. $K \in \{0,1,...,M-1\}$ where $K=0$ for the best matching cluster center and $K=1$ for the second best. $K_r(v_k,\Omega)$ is the number of cluster centers which are closer to data point v_k than cluster center u_r. $h_\lambda(J)$ is the function $e^{-J/\lambda}$ where λ is an experimental neighborhood learning rate adjustment parameter.

The Neural Gas cluster center update rule is:

$$\boldsymbol{u}_r^{new} = \boldsymbol{u}_r^{old} + \varepsilon_p \left\{ h_{\lambda p}\left[K_r(\boldsymbol{v},\Omega)\right](\boldsymbol{v} - \boldsymbol{u}_r)\right\}$$

(2.13)

λ is an experimental neighborhood learning rate adjustment parameter which decreases in magnitude during training. ε_p is an experimental parameter which is the functional equivalent of the learning rate α_p in k-means clustering. ε_p decreases in magnitude with each training pass as $\varepsilon_p = \varepsilon_i(\varepsilon_f/\varepsilon_i)^{(p/P)}$ where ε_i and ε_f are experimental initial and final values for the ε parameter. The λ parameter is controlled similarly from $\lambda_p = \lambda_i(\lambda_f/\lambda_i)^{(p/P)}$ where λ_i and λ_f are experimental initial and final values. As $\lambda \rightarrow 0$ the value of $h_\lambda(J) \rightarrow 0$ and the update rule performs as the equivalent of the k-means update rule by adjusting only the best matching cluster center. The best matching cluster center is updated with the full learning rate since $h_\lambda[K_r(\boldsymbol{v},\Omega)] = e^{-0} = 1$ when $\boldsymbol{u}_r$ is the best matching cluster center regardless of the value of λ.

There are two considerations in the application of the Neural Gas algorithm. They are the number of experimental parameters and the need to sort the cluster centers by proximity to input data points to determine their ranking. Martinetz et al. indicate that the selection of the values for the four experimental parameters, $\{\varepsilon_i, \varepsilon_f, \lambda_i, \lambda_f\}$, is not critical to the network's performance. The k-means and GLVQ networks require tuning of only one experimental parameter, α_p. The sort required to rank the cluster centers for each application of the update rule in the Neural Gas network is computationally demanding. The update rule of other quantization techniques such as k-means and the GLVQ require no sort.

2.4 Approximation

The cluster centers resulting from vector quantization are an approximation to the sampled function's input data parameters. Each input data vector is represented by the closest cluster center. The goal of a function-approximating neural network is to approximate the function value of the input data, the target or desired value of the function, not the input data itself.

Approximation of functions in neural networks is based upon traditional mathematical curve fitting techniques such as least-squares. Presented with a set of discrete values of a function some technique is employed to produce a function which best fits the presented data. In other words the neural network's goal is to define a function which maps each input data point as closely as possible to its desired output value. Let $g:R^D \Rightarrow R$ be such a mapping with D dimensions of real input data. Given D-dimensional input vectors of v_1, v_2, ..., v_N and their 1-dimensional desired output values $Z_1^{des}, Z_2^{des}, \ldots, Z_N^{des}$, a mapping function, $g(\ )$, is found which approximates $Z_k^{des} = f(v_k)$. The best-fit function is an interpolating function.

The approach to approximation in a 3-layer feed-forward neural network is to break the function into pieces based on similarity of the input data during the clustering process. Then an interpolating function, $g_r(\ )$ where $1 \le r \le M$, is found for each piece or cluster. In this manner the entire function is approximated by combining the interpolating functions from each cluster. Some approximation techniques use only the cluster in which the input parameters fall to evaluate the approximate function value. Other techniques use a combination of each clusters' interpolating function's value. In the latter case the owning cluster's contribution usually weighs more heavily in the combination.

The approximated or interpolated values for the function are obtained from the best-fit mapping functions. The general form of the approximating functions is

predetermined and the best-fit coefficients of the mapping function are obtained through a mathematical technique such as least-squares. The approximating function's general form can be as simple as a plane in D dimensions, $g_r = s_1 v_1 + s_2 v_2 + \ldots + s_D v_D + b$, where $\boldsymbol{s}$ is the slope of the plane and b is the intercept. The neural network provides the optimum slope and intercept. A more complex approximating function could be the gaussian or normal curve which has a general form:

$$g_r = e^{\left[-\frac{\|\boldsymbol{v}-\boldsymbol{u}\|^2}{2(\sigma)^2} \right]}$$

(2.14)

where $\boldsymbol{u}$ is the center of the bell shaped curve, $\boldsymbol{v}$ is the input data, and σ controls the width of the bell or how rapidly the function drops to a near zero value. The values for $\boldsymbol{u}$ and σ are determined by the neural network. Figure 2.5 shows a 3-dimensional gaussian surface. The gaussian approximating function is examined thoroughly further into this section.

Neural networks can be implemented with one neurode for each input data vector rather than one neurode representing each cluster of input data. This technique is more computationally demanding even though a clustering process is eliminated and may result in poor "generalization" characteristics. Poor generalization occurs when the neural network correctly maps the training data to interpolating functions but is ill-

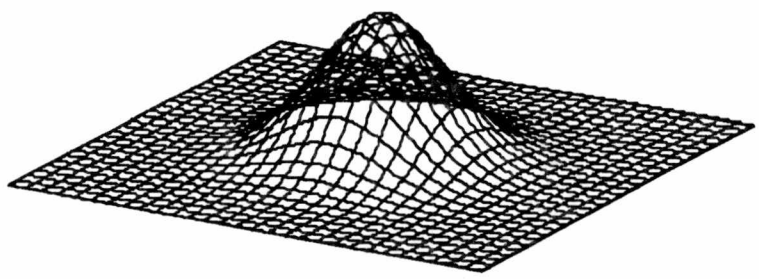


Figure 2.5. 3-dimensional gaussian surface.

suited to approximate other sets of data from the sampled function according to Bishop [16] and Platt [17]. Poor generalization is even more likely if the input data contains noise, random unpredictable errors. Usually the more neurodes implemented, the better the approximation if using noise-free data. Constraints on processing time and processor storage resources frequently make it impractical to implement one neurode for each input data vector.

Martinetz et al. [1] present a technique to model functions such as the Mackey-Glass time-series. Their technique uses a linear mapping of a multi-dimensional plane, referred to as a "hyper-plane," for the interpolating function of each cluster. This approach uses only the cluster in which the input data vectors fall to evaluate the approximate function value for each input data vector. A mathematical solution minimizes the squared error or distance between the actual and desired output for the function value of each data point in each cluster. This solution provides the best-fit hyper-planes through the clusters. A linear system is defined by each cluster of input data, the desired output value for the input data, and the best-fit hyper-plane. This linear system is represented in matrix form as $\boldsymbol{Z}^{des} = AX + E$. The input data points v_n, $1 \le n \le q$, are the q points closest to cluster center u_r . The data points are in D dimensions. The A matrix, Equation 2.15, consists of the input data points relative to their position in the cluster, and a column of 1s which are the coefficients of the intercept. Relative cluster position is obtained from $v_{nd} - u_{rd}$ where $1 \le d \le D$.

$$A = \begin{vmatrix} v_{11} - u_{r1} & v_{12} - u_{r2} & \cdots & v_{1D} - u_{rD} & 1 \\ v_{21} - u_{r1} & v_{22} - u_{r2} & \cdots & v_{2D} - u_{rD} & 1 \\ \vdots & \vdots & \vdots & \vdots & 1 \\ v_{q1} - u_{r1} & v_{q2} - u_{r2} & \cdots & v_{qD} - u_{rD} & 1 \end{vmatrix}$$

(2.15)

E is the error which is to be minimized as shown in Equation 2.16. Z^{des} is the desired or target output of the function which is being approximated. Z^{out} is the output of the mapping function.

$$E = \begin{vmatrix} Z_1^{des} - Z_1^{out} \\ Z_2^{des} - Z_2^{out} \\ \vdots \\ Z_q^{des} - Z_q^{out} \end{vmatrix}$$

(2.16)

Z^{des} is the desired output of each input data point:

$$Z^{des} = \begin{vmatrix} Z_1^{des} \\ Z_2^{des} \\ \vdots \\ Z_q^{des} \end{vmatrix}$$

(2.17)

The mathematical problem is to find the vector X for which matrix A maps X onto Z^{des} or approximately solve $Z^{des} = AX + E$ for X to minimize $E^2 = |Z^{des} - AX|^2$. There are several techniques to solve such a least-squares problem. The matrix product formula for a least-squares solution in Equation 2.18 [12,17] solves for X and minimizes E. s is the slope of the best-fit hyper-plane and b is the distance of the plane from the origin, the intercept.

$$X = \begin{bmatrix} s_1 \\ s_2 \\ \vdots \\ s_D \\ b \end{bmatrix} = \left(A^T A\right)^{-1} A^T Z^{des}$$

(2.18)

The solution can be obtained with Gauss-Jordan calculation of the A^TA matrix inverse [12,17]. The main problem with this process or formula is that the A^TA matrix for many linear systems may be singular, not invertible. In this case there may be an infinite number of solutions or there may be no solution. Singular Value Decomposition is an alternative method for solving these linear systems and is capable of solving virtually any linear system.

The process of Singular Value Decomposition (SVD) factors any given matrix into the product of a column-orthogonal matrix, a diagonal matrix of singular values, and the transpose of another orthogonal matrix [17,22]. Strang [17] provides proof that any matrix can be factored in this manner and demonstrates a relationship to eigenvalue-eigenvector factorization of a symmetric matrix. SVD is applied to the A matrix from the original linear system $\boldsymbol{Z}^{des} = A\boldsymbol{X} + E$:

$$A = U \cdot S \cdot V^T$$

$$S = \begin{vmatrix} S_1 & & & \\ & S_2 & & \\ & & \ddots & \\ & & & S_q \end{vmatrix}$$

(2.19)

Once the matrix has been factored the resulting matrices can be used in backsubstitution to solve for the values of $\boldsymbol{X}$.

The process of backsubstitution provides the best solution to $A\boldsymbol{X} = \boldsymbol{Z}^{des}$, solving for $\boldsymbol{X}$ while minimizing $(\boldsymbol{Z}^{des} - A\boldsymbol{X})$ or E, which also provides the minimum E^2, the least-squares solution goal. The ability of the combination of SVD and backsubstitution to solve for $\boldsymbol{X}$ and minimize E is due to the properties of the SVD matrices. Strang [17] and Vetterling et al. [22] provide proofs of these properties. The process of backsubstitution solves for $\boldsymbol{X}$ by substituting the inverse of the matrix factors in place of A^{-1} as shown in Equation 2.20.

$$
\begin{aligned}
A \cdot \boldsymbol{X} &= \boldsymbol{Z}^{des} \\
A^{-1} \cdot A \cdot \boldsymbol{X} &= A^{-1} \cdot \boldsymbol{Z}^{des} \\
\boldsymbol{X} &= A^{-1} \cdot \boldsymbol{Z}^{des} \\
A^{-1} &= V \cdot S \cdot U^{T} \\
\boldsymbol{X} &= V \cdot S \cdot U^{T} \cdot \boldsymbol{Z}^{des}
\end{aligned}
$$

(2.20)

The neural network determines the hyper-plane's slopes and intercepts, $\boldsymbol{X}$, for each cluster. Then the neural network approximation $\boldsymbol{Z}^{out}$ for any input vector $\boldsymbol{v}$ is obtained by first determining which cluster center $\boldsymbol{u}$ is the closest in euclidean distance to the input vector. The input vector's relative position within the cluster is used for matrix A. Relative postion is obtained from $(v_d - u_d)$ where $1 \le d \le D$. The values for $\boldsymbol{X}$ are the slope and intercept of the hyper-plane of the nearest cluster center. Calculation of $\boldsymbol{Z}^{out}$ is obtained from:

$$
\begin{aligned}
A &= \begin{vmatrix} v_1 - u_1 & v_2 - u_2 & \cdots & v_D - u_D & 1 \end{vmatrix} \\
\boldsymbol{X} &= \begin{bmatrix} s_1 \\ s_2 \\ \vdots \\ s_D \\ b \end{bmatrix} \\
\boldsymbol{Z}^{out} &= A\boldsymbol{X}
\end{aligned}
$$

(2.21)

Output from the entire network for any input data vector $\boldsymbol{v}_k$ is represented by $\boldsymbol{Z}^{out}(\boldsymbol{v}_k) = A_k \boldsymbol{X}_i$. A_k is the A matrix in Equation 2.21 for input data vector $\boldsymbol{v}_k$ and $\boldsymbol{X}_i$ is the best-fit hyper-plane of the nearest cluster to $\boldsymbol{v}_k$.

Radial Basis Functions (RBF) are another class of local functions which have become popular for 3-layer feed-forward neural networks. Their applicability for network architectures designed for function approximation has been researched

extensively [5,7,8,10,16,18,19]. RBF networks use a radially symmetric function in the inner layer neurodes for the mapping function of each cluster. The general form of the RBF function in each neurode is:

$$\phi(\|v - u\|)$$

(2.22)

where v is the input data vector, and u is the center of the function corresponding to the cluster center. Commonly the function ϕ is the normal error curve or gaussian function:

$$\phi(c) = e^{\left(-\frac{c^2}{\sigma^2}\right)}$$

(2.23)

A neurode with a gaussian RBF then uses for it's local function:

$$e^{\left(-\frac{\|v-u\|^2}{\sigma^2}\right)}$$

(2.24)

The multi-dimensional shape of this function can be described as spheroidal. The distance measure between the input data vectors v and the node center u is usually euclidean. Girosi [8] argues that the euclidean distance measure may be inappropriate for some RBFs and recommends other measures in cases where each dimension of the input data may not be equally relevant. σ is an experimental parameter governing the width or diameter of the spheroid.

The proper determination of the width of each neurode is important. The output of the network for each input data vector is a combination of the output of each neurode in the inner layer, not just the output of the neurode with the best matching center. The width should be great enough that each neurode's local function overlaps slightly with the neighboring neurodes but not so great that several of the neurodes

respond greatly to an input data vector. If the neurodes don't overlap slightly there may be gaps between the local functions which will lead to poor interpolation by the network. Moody and Darken [5] describe a "global first nearest neighbor" heuristic where all neurodes are assigned a uniform width based on the average distance from the center of each neurode to their respective nearest neighboring neurode's center. Platt [18] uses the distance to the nearest neighbor to assign unique widths to each node, the most commonly employed technique. Hartman and Keeler [10] recommend allowing the width of each node to vary in each dimension depending on differing significance of each dimension of the input data. Musavi et al. [20] propose a technique which calculates the width for each node based on the distance to each node which is closest in each dimension of the input data space.

The gaussian or normal error function is often described as a bell shaped curve. Figure 2.5 shows the gaussian bell shape. The value of the function is greatest at the center $\boldsymbol{u}$ and tapers off to near zero in any direction from the center. The most commonly chosen function for RBFs is the gaussian function [5,8,16] however, the use of a variety of alternatives has been explored. Wong [19] argues that gaussian RBFs are too smooth to detect high-frequency changes in a complex function and recommends a box-like or triangular function to model time-series functions. Hartman and Keeler [10] advocate gaussian bar functions which allow the adjustment of the width of the gaussian surface in any dimension of the input data. Quadratic or multiquadric functions are employed by Girosi [8].

Output from the inner layer of neurodes for any input data vector is a linear combination of the output from each neurode:

$$Z^{out}(\boldsymbol{v}) = \sum_{r=1,M} w_r \cdot e^{\left(- \frac{\|\boldsymbol{v}-\boldsymbol{u}_r\|^2}{(\sigma_r)^2} \right)}$$

(2.25)

This linear combination of output from each neurode differs from the Neural Gas method in which output is from only the best matching neurode. M is the total number of RBFs or clusters in the network, the number of neurodes in the hidden layer. v is the input data vector. The strength of the output from each node is controlled by weight w_r. Each RBF has a weight associated with it and it is these weights which must be trained by the approximation component of the RBF network. Figure 2.6 depicts a combination of three gaussian curves in two dimensions.

Training an RBF network adjusts the weights w between the hidden layer and the output layer. These weights control the contribution from each RBF to the output layer for the approximation of each input data vector. The first step in the training process is to construct a matrix Y which contains the RBF function value of each training data input vector for each neurode in the inner layer. The Y matrix contains one row for each input data vector and one column for each neurode. The Y matrix dimensions are N by M for N input vectors and M nodes. $Y_{k,r}$ is the RBF function value shown in Equation 2.24 of input data vector v_k on node u_r. Equation 2.26 shows the composition of the Y matrix.

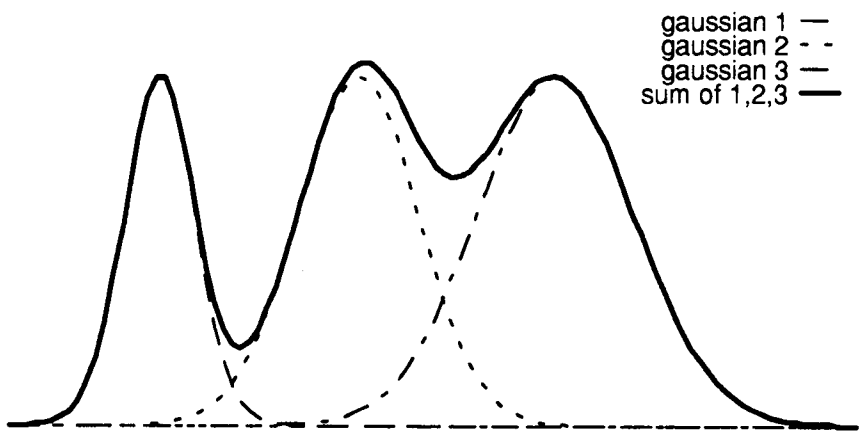


Figure 2.6. Summation of three gaussian curves.

$$Y = \begin{vmatrix} Y_{1,1} & Y_{1,2} & \cdots & Y_{1,r} & \cdots & Y_{1,M} \\ Y_{2,1} & Y_{2,2} & \cdots & Y_{2,r} & \cdots & Y_{2,M} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ Y_{k,1} & Y_{k,2} & \cdots & Y_{k,r} & \cdots & Y_{k,M} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ Y_{N,1} & Y_{N,2} & \cdots & Y_{N,r} & \cdots & Y_{N,M} \end{vmatrix}$$

(2.26)

Part of the attractiveness of RBF networks is that once the Y matrix has been constructed all that remains to complete the network training is solving a linear system of equations, $Yw=\mathbf{Z}^{des}$. The system is to be solved for w, the array of weights for each RBF. $\mathbf{Z}^{des}$ is the array of desired output values for each input data vector as shown in Equation 2.17.

SVD can be applied to solve this system in the same manner in which the solution to the neural gas system was determined. The Y matrix is factored into the decomposition matrices as was matrix A in Equation 2.19. Backsubstitution provides the best solution to the system, solving for w:

$$w = V \cdot S \cdot U^T \cdot \mathbf{Z}^{des}$$

(2.27)

Calculation of the w array completes the network training. Output from the trained network for function approximation for any single input data vector is obtained from Equation 2.25. In a more practical application of the network for approximation purposes, several input data vectors,$\{v_1,v_2,...,v_N\}$, could be evaluated by storing their RBF function values into a new Y matrix. The new Y matrix is constructed like the original Y matrix in Equation 2.26. Output from the network in this case would be an array, $\mathbf{Z}^{out} = Yw$, as shown in Equation 2.28.

$$\mathbf{Z}^{out} = \begin{vmatrix} \mathbf{Z}_1^{out} \\ \mathbf{Z}_2^{out} \\ \vdots \\ \mathbf{Z}_N^{out} \end{vmatrix}$$

(2.28)

2.5 Prediction

Neural networks may be applied to function-prediction problems. Consider the problem of predicting the Mackey-Glass time-series function. Training and testing phases accomplish the prediction. The network generates an approximation to a set of training data from the time-series in the training phase. The network then approximates the time-series function by interpolation for any input data which is sampled from the same range as the training data. The range of the input data for training and testing is explained in Section 3.3. The network performs interpolation on data sampled from a different segment of the time-series function to provide a prediction. Whether the network is predicting or merely approximating depends on the function segment from which the test data is sampled. The term "prediction" is applicable to approximating other function segments since the function is a time-series. Comparison of the approximation surfaces in the local processing units from the training phase with another set of time-series data, a test set, obtains the prediction in the testing phase.

CHAPTER 3

Methods

3.1 Approach to Evaluation

This research investigates the vector quantization and approximation components of the Neural Gas network. The primary goal of the research is to determine which component contributes more to the low prediction error reported by the authors. The approach of the research is to substitute other methods of quantization and approximation in place of the Neural Gas components and evaluate the resulting prediction errors. For example consider substituting only the quantization component of the Neural Gas model with some other algorithm. Suppose this results in a much greater prediction error than substituting only the approximation component. This result would lead to the conclusion that the quantization component does indeed contribute much more heavily to the low prediction errors reported in the Neural Gas research [1].

3.2 Substitute Components

The Neural Gas network consists of Neural Gas vector quantization and best-fit linear mapping approximation components. The alternative components chosen for substitution are selected because current research indicates they perform well in neural networks.

The substitute vector quantization component chosen for this research is the GLVQ [2]. The GLVQ is chosen for its computational efficiency and the neighborhood approach to cluster center updates. Pal et al. [2] demonstrate that the GLVQ clusters as quickly if not more quickly than Kohonen's LVQ [4]. The GLVQ is not prone to problems associated with poor cluster center initialization as are other

quantization techniques since the cluster centers are updated with a neighborhood approach.

Radial Basis Functions solved by Singular Value Decomposition is the substitute approximation component. RBFs are very popular in current research involving prediction networks and have been shown to yield low errors on time-series predictions [5,10,16,19]. Singular Value Decomposition combined with backsubstitution solves virtually any set of linear equations providing the optimum solution.

To evaluate the Neural Gas network with all combinations of substitute components requires the testing of four neural networks:

1. Neural Gas vector quantization and hyper-planes approximation
2. Neural Gas vector quantization and RBF approximation
3. GLVQ vector quantization and hyper-planes approximation
4. GLVQ vector quantization and RBF approximation

3.3 Time-Series Data

The differential equation from Equation 2.1 generates the time-series training data used in this research with the constants $a = 0.2$, $z = 0.1$ and $\tau = 17.0$. The datasets are constructed with four input values and one output value per sample in the following manner. Each data sample in the training dataset is obtained randomly from the portion of the series between $t = 500$ and $t = 4000$ with the input real values of $x[t]$, $x[t - 6]$, $x[t - 12]$, and $x[t - 18]$ and the output real value of $x[t + 85]$. The test dataset is constructed from sequential samplings of the portion of the series beyond t = 4000. In this research both datasets consist of 500 samples. Prior to input into the networks the data in each dataset was scaled by adjusting it to have a mean of 0 and a standard deviation of 0.1.

3.4 Performing the Evaluation

Prediction results from each network in this research are presented as normalized prediction error (*NPE*). *NPE* is the root mean squared error of a network's predictions Z^{out} divided by the standard deviation of the desired output Z^{des} from the network. Equation 3.1 shows the calculation of *NPE*.

$$NPE = \frac{\sqrt{\frac{1}{N}\sum_{k=1,N}\left(Z_k^{out} - Z_k^{des}\right)^2}}{\sqrt{\frac{1}{N}\sum_{k=1,N}\left(Z_k^{des} - \frac{1}{N}\sum_{y=1,N} Z_y^{des}\right)^2}} \tag{3.1}$$

Each evaluated network is composed of two components. The algorithms for the components used in this research are designed to be interchangeable since each evaluated network is constructed by substituting components. In other words either quantization component can be used with either approximation component. The appendix contains the algorithms of the four components in this research. The approximation component algorithms are in two parts, training and testing.

Prior to the application of backsubstitution to the SVD matrices it is necessary to define a threshold value for the elements in the matrix of singular values. The elements in the *S* matrix described in Equation 2.19 must be set to a value of 0.0 when they contain a value smaller than the threshold. This avoids an ill-conditioned matrix, one which is very sensitive to roundoff error [17]. In this research the threshold value is 0.000001, a typical choice according to Vetterling [22].

The complete prediction process for the networks is described in Figure 3.1. The process consists of two phases, training and testing. In the training phase quantization is performed on a training dataset and the approximation component produces mapping functions. During the testing phase the test data is assigned to the cluster centers obtained from the training phase. The mapping functions from the

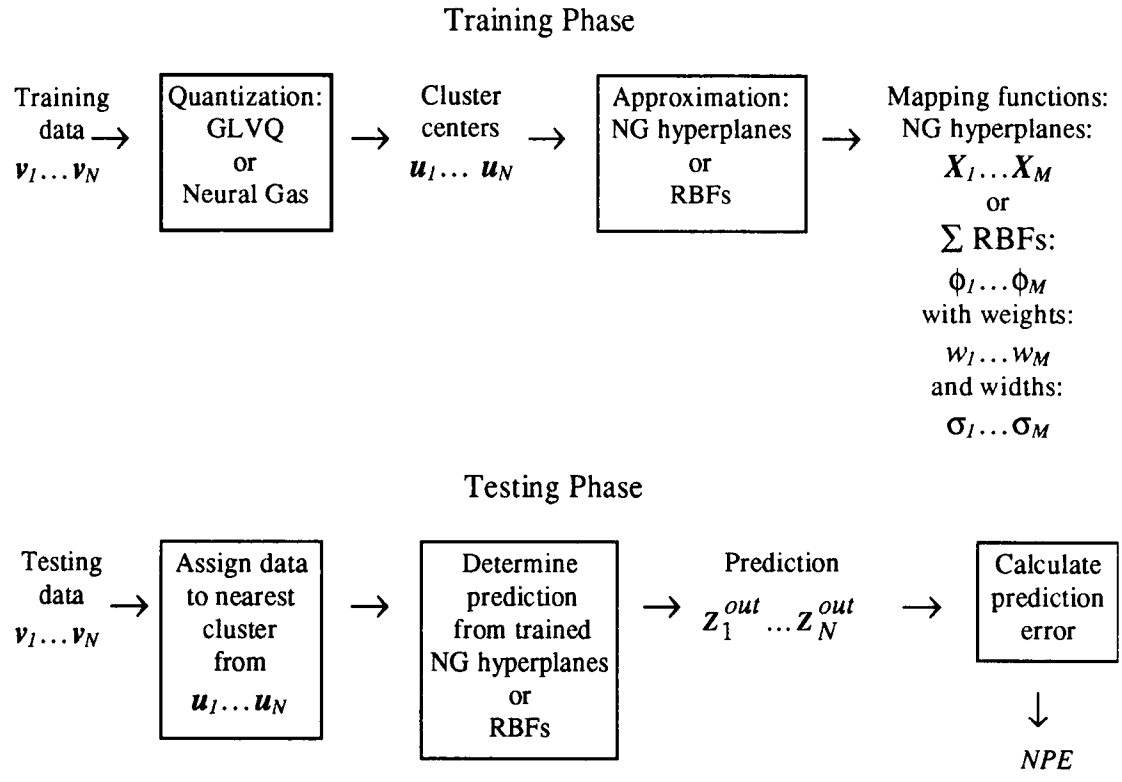


Figure 3.1. Prediction process.

training phase then produce the network predictions which are evaluated and reported in the form of *NPE*.

Each of the four networks is evaluated with different sizes. The size of the network is the number of cluster centers used in the inner layer. The sizes evaluated are 16, 32, 50, and 100. These sizes of networks are chosen so the results can be compared with related research. There are several reasons to evaluate different size networks. Related research usually involves different network sizes. A network's performance may vary with size. A network might be superior to others with one size but not with another size. The optimum number of cluster centers for a given network is experimental. Lower *NPE* is generally obtained with larger networks. There is the possibility of "overfitting" or poor "generalization" occurring with too many cluster centers. Poor generalization means that the network mapping functions fit the details

of the training data but do not represent the trends of the data [16]. Poor generalization usually results in poor interpolation.

There are several experimental parameters to consider in the evaluated networks. The values of these parameters are changed experimentally to obtain the smallest *NPE*. Finding the optimum values for these parameters is accomplished in this research by making multiple prediction runs for each network configuration while varying each parameter over a range of values. The high and low end of the ranges are determined by trial and error.

The Neural Gas network authors [1] claim "the particular choice for these parameter values is not very critical" to their clustering component. To validate this claim many runs of the Neural Gas clustering component are made while varying the experimental parameters over a range which includes the authors' suggested values. A range is also necessary since the optimum quantization parameters may differ for the two approximation components used with the Neural Gas clustering component. The parameters are changed between each successive run of the prediction network by a value equivalent to 1/7 of the range for each respective parameter. The stepsize for the 100 node network is reduced to 1/6 of the range because of long run times. The Neural Gas authors' recommended values and the ranges used in this research are listed in Table 3.1.

The GLVQ component uses one experimental parameter. This parameter is the learning rate α. The value for α varies over a range 0.1 ... 0.95 with an incremental

Table 3.1. Recommended Neural Gas parameter values and ranges used.

Parameter	Recommended value	Range used in this research
λ_i	M/3	2M ... 0.1
λ_f	0.0001	0.01 ... 0.000005
ε_i	0.99	2.7 ... 0.1
ε_f	0.001	0.3 ... 0.0001

step of 0.01. The reasoning behind varying α is the same as that for varying the Neural Gas parameters. In the 16 cluster-center network run the range for α was .0004 to .02 with a step size of 0.0002 because the GLVQ always failed for $\alpha > .02$. This GLVQ failure is discussed in Chapter 4.

The RBF approximation component contains one experimental parameter. An RBF width factor β varies the width of all the RBFs in a network. The width of each RBF in a network is a multiple β of the root mean squared quantization error of it's cluster center. The formula for RBF width σ calculation is described in step 3 of the RBF training phase approximation algorithm in the appendix. The value for β varies over the range 0.5 ... 7.0 with incremental steps of 0.05. The optimum β yields the lowest *NPE* from the network. The range of β is determined by trial and error.

The procedure for network evaluation in this research is presented in the appendix. This procedure outlines how both quantization components and both approximation components were evaluated in all possible combinations for each size network.

CHAPTER 4

Results and Conclusions

4.1 Results

All of the program code for this research was written in ANSI C on a UNIX platform. The runs were made on Silicon Graphics Indigo II and Power Challenge workstations. The Indigo II CPU was a 100 MHz MIPS R4000. The Power Challenge CPU was a 75 MHz MIPS R8000. Run times listed for any network or component were from the Indigo II.

Table 4.1 shows the best *NPE* for each network evaluated. Also shown are the parameters which were used to achieve those results. Underlined *NPE* indicates results where RBFs were superior to Neural Gas hyper-planes. Figure 4.1 is a plot of the *NPE* from Table 4.1. The legend in Figure 4.1 describes the networks plotted by listing the components used in the network.

Figure 4.2 is a plot of the *NPE* from the 16 cluster Neural Gas network versus two of the experimental parameters, ε_i and ε_f. Plots such as this were examined to

Table 4.1. Best *NPE* from each network.

		Quantization Technique										
		GLVQ 200 passes				Neural Gas 400 passes						
Approximation Technique	Clus-ters	$E^{RMS\ quant}$ Quantization Error	α	*NPE* Prediction Error	β	$E^{RMS\ quant}$ Quantization Error	ε_i	ε_f	λ_i	λ_f	*NPE* Prediction Error	β
RBFs	16	.0643	.001	.480	7.05	.0623	1.40	.150	.100	.010	.443	3.35
	32	.034	.56	.275	4.00	.0340	1.40	.0501	26.68	.00334	.263	4.40
	50	.0273	.17	.181	2.80	.0283	.100	.180	10.0	.004003	.164	2.90
	100	.0174	.92	.0800	3.00	.0179	.100	.06008	200.0	.0100	.081	3.25
Neural Gas hyper-planes	16	.0658	.0006	.303		.0556	2.70	.200	5.41	.010	.274	
	32	.034	.57	.193		.0357	2.70	.100	16.0	.000005	.198	
	50	.0266	.79	.171		.0301	1.66	.300	50.0	.000005	.171	
	100	.0174	.71	.151		.0175	1.14	.06008	120.0	.002004	.162	

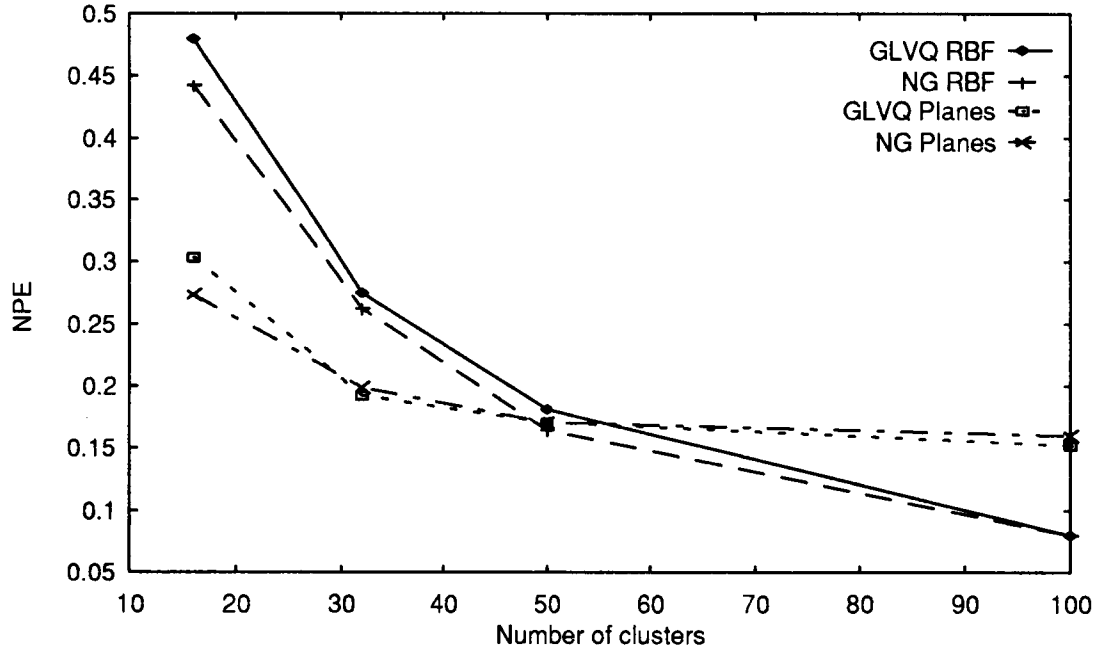


Figure 4.1. *NPE* from each network.

establish the ranges for the parameters in this research. The parameter ranges were selected to span the *NPE* surface where it is at a minimum. The circular contours transposed onto the ε_i-ε_f surface are useful for finding the minimal *NPE*. Plots such as this are only a guideline since the *NPE* surface shown is a best-fit surface to the actual *NPE* values obtained for the networks.

Figure 4.3 shows the *NPE* obtained from the Neural Gas networks tested in this research. Also shown in Figure 4.3 are the Neural Gas authors' reported results. Results from the research of Moody and Darken [5] are included for a benchmark. This research was not able to duplicate the low *NPE* of the Neural Gas research [1] but low prediction errors were obtained by comparison with the benchmark.

Table 4.2 lists the best root mean squared quantization error, $E^{RMS\ quant}$, for each network evaluated and the parameters used to obtain those errors.

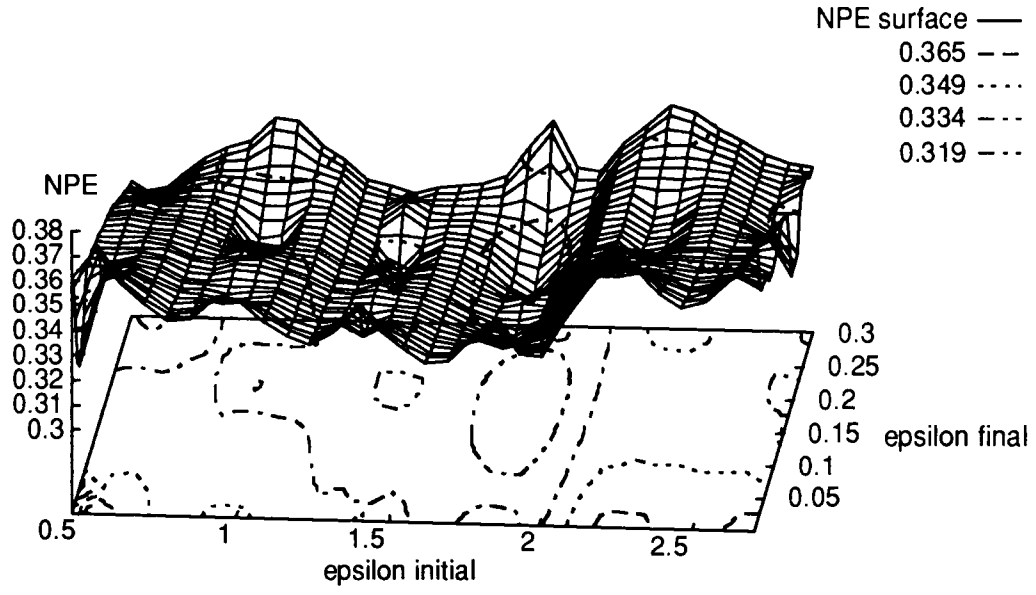


Figure 4.2. *NPE* versus two Neural Gas experimental parameters.

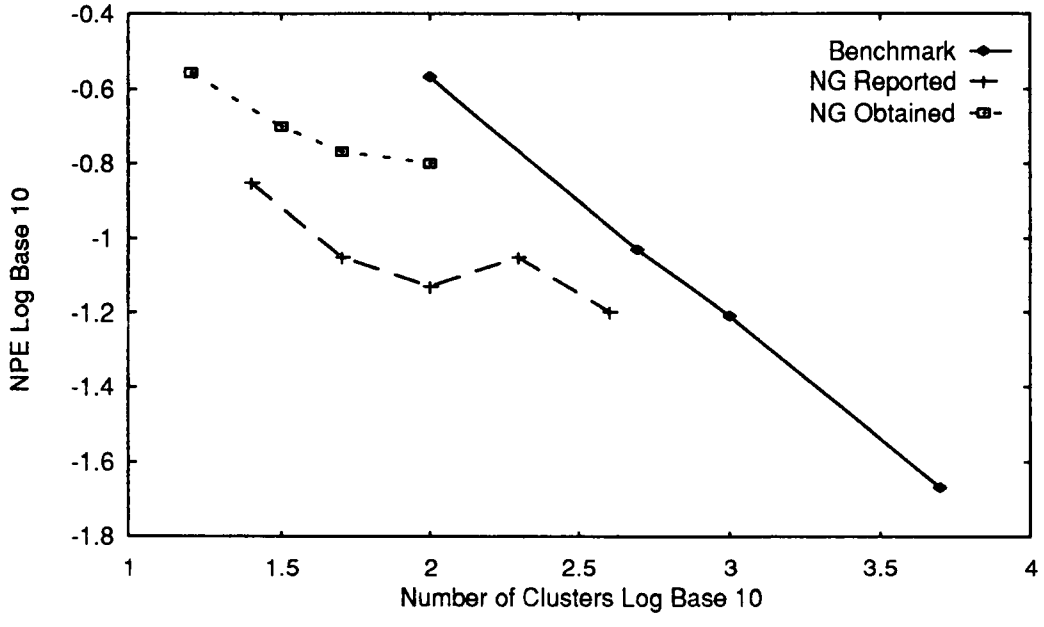


Figure 4.3. Neural Gas Network and Benchmark *NPE*.

Table 4.2. Best quantization errors from each network.

	GLVQ 200 passes		Neural-Gas 400 passes				
Nodes	$E^{RMS\ quant}$	α	$E^{RMS\ quant}$	ε_i	ε_f	λ_i	λ_f
16	.0602	.004	.0500	2.70	.0001	26.0	.000005
32	.0330	.41	.0331	2.70	.0001	10.73	.000005
50	.0259	.60	.0267	1.14	.06008	50.0	.006002
100	.0171	.95	.0170	.620	.06008	120.0	.000005

Figure 4.4 is a plot of cluster centers following quantization. These plots are from the 16 cluster GLVQ and Neural Gas networks which produced the smallest *NPE* with the hyper-planes approximation component. The algorithms cluster quite differently. Results from testing all 4 networks indicate that the clustering technique is not the primary factor controlling *NPE*, a set of clusters with a low $E^{RMS\ quant}$, not necessarily the best, will provide good prediction results. Visual inspection of the cluster centers shows both methods chose what appear to be valid positions for cluster centers. The cluster centers were spread throughout the entire set of data. Frequently, cluster centers were placed in sections where the density of data points is high.

Comparison of Tables 4.1 and 4.2 reveals that the best prediction errors were not obtained from the clusters which provided the minimum $E^{RMS\ quant}$. However, the clusters resulting in the best *NPE* did have an $E^{RMS\ quant}$ near the minimum.

Table 4.3 shows the execution times in seconds for each component tested in the research.

Table 4.3. Execution time (seconds) for each component.

	Component			
	Quantization		Approximation	
Clusters	GLVQ	NG	RBF	Planes
16	11.5	41.1	0.7	0.2
32	22.7	128.0	2.0	0.3
50	35.3	201.5	3.3	0.4
100	70.1	568.7	9.1	0.5

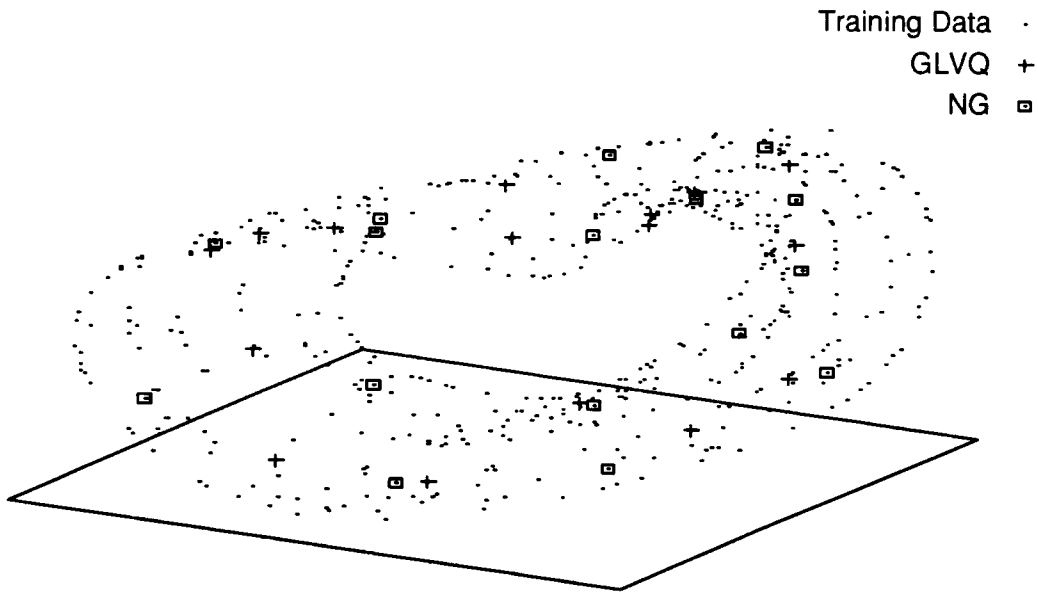


Figure 4.4. Neural Gas and GLVQ cluster centers.

4.2 Conclusions

There are two bands in the plot of *NPE* for the four evaluated networks in Figure 4.1. Each band contains the results from two of the evaluated networks. One band is the networks with the RBF approximation component. The other band is the networks with the hyper-plane approximation component. The *NPE* varies greatly between bands but very little within the bands. This leads the author to the following conclusion. The approximation component has a much greater role than the quantization component in the network's ability to predict the Mackey-Glass time-series. The prediction results were influenced most by the approximation component. The quantization component had little effect on the prediction error. There was a minor change in the *NPE* between networks using the same approximation component and different quantization components. In networks smaller than 50 clusters the hyper-plane approximation technique provided smaller prediction error. In the 100 cluster network RBFs were superior.

The results in Table 4.1 and Figure 4.1 show lower *NPE* in smaller networks using hyper-planes and lower *NPE* in larger networks using RBFs. It was expected that hyper-planes would be superior to RBFs in large networks. Instead hyper-plane performance was superior in the smaller networks. Other research [19] indicates that gaussian RBFs may be too smooth to accurately map rapidly changing functions such as the Mackey-Glass time-series. If smoothness is a large factor in mapping then the hyper-plane component should not outperform RBFs in smaller networks. Planes are smoother than gaussian surfaces. Bishop [16] indicates that too many clusters may lead to poor generalization. In the large networks evaluated in this research RBFs were superior to hyper-planes. Based on the results it appears that gaussian RBF networks are less prone to generalization problems than hyper-plane networks.

Hyper-planes were superior to RBFs in the small 16 cluster networks. Observe the twisted ribbon shape of the input data in Figure 4.4. Perhaps Neural Gas hyper-planes are well suited for mapping the nearly planar sections of the ribbon which are localized to the clusters of the hyper-planes. In the RBF networks the optimum values for the experimental node width parameter β were large, ranging from 3 to 7. There were few 0.0 or near 0 values in the *Y* matrices. This indicated all RBF clusters responded to each input data point. This is caused by the large values for β. Gaussian RBF network output is controlled by a linear combination of each clusters' gaussian function values. Apparently gaussian RBFs have difficulty mapping large nearly planar surfaces. The best nearly planar mappings were obtained by using a combination of many clusters' activations for each input data vector. This difficulty is overcome by increasing the number of clusters thereby reducing the sizes of the clusters and the sizes of the nearly planar segments of the ribbon. In fact the RBF networks outperformed hyper-planes in the 100 cluster networks.

Node width determination seems to be a key to improving RBF prediction results. During the early stages of this research a mistake in the code for the RBF node

width calculation resulted in nearly random node widths. Occasionally the prediction results from these incorrectly coded networks were superior to the results from the algorithms presented in this research.

Execution time for the Neural Gas quantization component is notably greater than that of the GLVQ as shown in Table 3.3. This is attributable to the Neural Gas algorithm's required sort of all cluster centers. This sort was discussed in Section 2.3.

Optimum values for the experimental parameters λ_i , λ_f , ε_i , and ε_f were often fairly distant from the Neural Gas authors' recommendations. Perhaps a gradient descent on the *NPE* to adjust the run parameters would be useful for tuning the Neural Gas experimental parameters.

The GLVQ quantization component failed to yield useful results in the 16 cluster network with a learning rate greater than 0.02. The GLVQ algorithm performed acceptably in the 16 cluster network until approximately 10 or 20 passes from the end of run, then the $E^{RMS\ quant}$ increased rapidly toward infinite. Reducing the number of passes did not correct the problem. One method to correct this shortcoming would be to rewrite the GLVQ algorithm to record the cluster centers after each pass. Then if the $E^{RMS\ quant}$ on a pass increases more than some small experimental amount stop quantizing and restore the recorded cluster centers.

Further research is recommended in the following areas. Alternative methods of RBF node width determination could be a key to further improvement of RBF network results. A range of Neural Gas experimental parameters was explored in this research. Still it is not clear that the optimum values for these parameters were found. A technique which adjusts those experimental values based on changes in $E^{RMS\ quant}$ or *NPE* over a series of quantization and approximation runs might lead to better choices for those parameters.

List Of References

[1] T. M. Martinetz, S. G. Berkovich, K. J. Schulten, "'Neural-Gas' Network for Vector Quantization and its Application to Time-Series Prediction," IEEE Trans. Neural Networks, vol. 4, pp. 558-569, 1993.

[2] N. R.Pal, J. C. Bezdek, E. C.-K. Tsao, "Generalized Clustering Networks and Kohonen's Self-Organizing Scheme," IEEE Trans. Neural Networks, vol. 4, pp. 449-557, 1993.

[3] M. C. Mackey, L. Glass, "Oscillation and Chaos in Physiological Control Systems," Science, vol. 197, pp. 287-289, 1977.

[4] T. Kohonen, "The Self Organizing Map," Proceedings of the IEEE, vol. 78, pp. 1464-1480, 1990.

[5] J. Moody, C. Darken, "Fast Learning in Networks of Locally-Tuned Processing Units," Neural Computation, vol. 1, pp. 281-294, 1989

[6] M. Caudill, C Butler, Naturally Intelligent Systems, MIT Press, 1990.

[7] W.C. Mead, R.D. Jones, Y.C. Lee, C.W. Barnes, G.W. Flake, L.A. Lee, M.K. O'Rourke, "Using CNLS-Net to Predict the Mackey-Glass Chaotic Time Series," International Joint Conference on Neural Networks, pp II-485-II-490, 1991.

[8] F. Girosi, "Some Extensions of Radial Basis Functions and Their Applications in Artificial Intelligence," Computers Mathematics Applications, vol. 24, no. 12, pp 61-80, 1992.

[9] T. D. Sanger, "A Tree-Structured Adaptive Network for Function Approximation in High-Dimensional Spaces," IEEE Transactions on Neural Networks, vol. 2, no. 2, pp 285-293, 1991.

[10] E. Hartman and J. D. Keeler, "Predicting the Future: Advantages of Semilocal Units," Neural Computation, vol. 3, pp 566-578, 1991.

[11] R. Deutsch, Estimation Theory, Prentice-Hall, 1965.

[12] P. Shields, Elementary Linear Algebra, Worth Publishers, 1980.

[13] S. D. Goggin, K.E. Gustafson, and K. M. Johnson,"An asymptotic singular value decomposition analysis of nonlinear multilayer neural networks," International Joint Conference on Neural Networks, pp I-785-I-789, 1991.

[14] P. A. Stark, Introduction to Numerical Methods, Macmillan Publishing Company, 1970.

[15] R. Beale and T. Jackson, Neural Computing: An Introduction, IOP Publishing Ltd, 1990.

[16] C. Bishop, "Improving the Generalization Properties of Radial Basis Function Neural Networks," Neural Computation, vol. 3, pp 579-588, 1991.

[17] G. Strang, Linear Algebra and Its Applications, Harcourt Brace Jovanovich, Inc., 1988.

[18] J. Platt, "A Resource-Allocating Network for Function Interpolation," Neural Computation, vol. 3, pp 213-225, 1991.

[19] Y. Wong, "How Gaussian Radial Basis Functions Work," International Joint Conference on Neural Networks, pp II-133-II-138, 1991.

[20] M. T. Musavi, W. Ahmed, K. H. Chan, K. B. Faris, and D. M. Hummels, "On the Training of Radial Basis Function Classifiers," Neural Networks, vol. 5, pp 595-603, 1992.

[21] J. E. Freund and R. E. Walpole, Mathematical Statistics, Prentice-Hall, Inc., 1980.

[22] William T. Vetterling, William H. Press, Brian P. Flannery, and Saul A. Teukolsky, Numerical Recipes, Cambridge University Press, 1992.

Appendix

Appendix

This appendix presents algorithms and procedures used in this research. The algorithms for the four components evaluated in this research are listed first. The approximation component algorithms are in two parts, training and testing. Following these algorithms is the procedure for network evaluation.

GLVQ quantization algorithm.

1. Given: input dataset $V = \{v_1, v_2, ..., v_N\} \subset R^D$.
   Determine the number of passes P and the number of clusters M.
2. Initialize the cluster centers: $\Omega = \{u_1 = v_1, u_2 = v_2, ..., u_M = v_M\}$.
3. For $p = 1,2,...,P$
   For $k = 1,2,...,N$
   Find $\|v_k - u_i\| = \min_{1 \le r \le M} \|v_k - u_r\|$
   /* u_i is the closest cluster center to v_k.. */
   For $j=1,2,...,M$
   If $u_j = u_i$ update u_j with Equation 2.9
   otherwise update u_j with Equation 2.11
   next j.
   next k.
   Calculate quantization error, $E^{RMS\ quant}$ with Equation 2.2.
   next p.
4. Output the cluster centers, $\Omega = \{u_1, u_2, ..., u_M\}$.

Neural Gas quantization algorithm.

1. Given: input dataset $V = \{v_1, v_2, ..., v_N\} \subset R^D$.
   Determine the number of passes P, the number of clusters M,
   ε_i, ε_f, λ_i, and λ_f.
2. Initialize the cluster centers: $\Omega = \{u_1, u_2, ..., u_M\}$randomly over the input data space.
3. For $p = 1,2,...,P$
   For $k = 1,2,...,N$
   Find $\|v_k - u_i\| = \min_{1 \le r \le M} \|v_k - u_r\|$
   /* u_i is the closest cluster center to v_k. */
   For $r=1,2,...,M$
   $RANK_r$ = number of cluster centers closer to v_k than u_r.
   next r.

$$\varepsilon_p = \varepsilon_i \left(\varepsilon_f / \varepsilon_i \right)^{p/P}$$

$$\lambda_p = \lambda_i \left(\lambda_f / \lambda_i \right)^{p/P}$$

   For $r=1,2,...,M$

$$h_{\lambda_p} = e^{-\left(RANK_r / \lambda_p \right)}$$

$$u_r = u_r + \varepsilon_p \cdot h_{\lambda_p} (v_k - u_r)$$

   next r.
   next k.
   Calculate quantization error, $E^{RMS\ quant}$. /* Equation 2.2. */
   next p.
4. Output the cluster centers, $\Omega = \{u_1, u_2, ..., u_M\}$.

Neural Gas hyper-planes training phase approximation algorithm.

1. Given: input dataset $V = \{v_1, v_2, ..., v_N\} \subset R^D$ and
   cluster centers, $\Omega = \{u_1, u_2, ..., u_M\}$.
2. For r=1,2,...,M
   $q = 0$
   For k = 1,2,...,N
   If v_k is closest to u_r
   $q = q + 1$
   /* Add a row to matrix A for $(v_k - u_r)$,
   as in Equation 2.15 */
   For d = 1,2,...,D
   $A_{q,d} = (v_{qd} - u_{rd})$
   $A_{q,d+1} = 1$
   next d
   /* The last element of each input data vector is the
   desired output Z^{des} */
   $Z^{des}_q = v_{k(D+1)}$
   next k.
   Apply SVD to matrix A: $A = U \cdot S \cdot V^T$, Equation 2.19.
   Apply backstitution to solve for X_r, the slope and intercept of the
   best-fit hyper-plane: $X = V \cdot S \cdot U^T \cdot Z^{des}$.
   next r.
3. Output the set of best–fit hyper-planes: $\{X_1, X_2, ..., X_M\}$.

Neural Gas hyper-planes testing phase approximation algorithm.

1. Given: input dataset $V = \{v_1, v_2, ..., v_N\} \subset R^D$ and
   cluster centers, $\Omega = \{u_1, u_2, ..., u_M\}$.
2. For k = 1,2,...,N
   Find $\|v_k - u_i\| = \min_{1 \le r \le M} \|v_k - u_r\|$
   /* u_i is the closest cluster center to v_k. */
   /* X_i is the hyper-plane for u_i */
   $A = \begin{vmatrix} v_{k1} - u_{i1} & v_{k2} - u_{i2} & \cdots & v_{kD} - u_{iD} & 1 \end{vmatrix}$
   $Z^{out}_k = A \cdot X_i$
   Next k.
3. Calculate NPE for Z^{out}.

RBF training phase approximation algorithm.

1. Given: input dataset $V = \{\boldsymbol{v}_1, \boldsymbol{v}_2, ..., \boldsymbol{v}_N\} \subset R^D$ and
   cluster centers, $\Omega = \{\boldsymbol{u}_1, \boldsymbol{u}_2, ..., \boldsymbol{u}_M\}$.
   Determine β, the experimental node width factor.
2. For r=1,2,...,M
   $q_r = 0$
   For k=1,2,...,N
   If $\boldsymbol{v}_k$ is closest to $\boldsymbol{u}_r$
   $q_r = q_r + 1$
   next k.
   next r.
3. For r=1,2,...,M

$$\sigma_r = \beta \sqrt{\frac{1}{q_r} \sum_{k=1,N} B_{kr} (\boldsymbol{v}_k - \boldsymbol{u}_r)^2}$$

   /* Recalling that B_{kr} is a function determining if $\boldsymbol{v}_k$ is closest to $\boldsymbol{u}_r$ */
   next r.
   For k=1,2,...,N
   For r=1,2,...,M

$$Y_{k,r} = e^{\left(- \frac{\|\boldsymbol{v}_k - \boldsymbol{u}_r\|^2}{(\sigma_r)^2} \right)}$$ /* from Equation 2.24 */

   next r.
   next k.
   Apply SVD to matrix Y. $Y = U \cdot S \cdot V^T$, Equation 2.19.
   Apply backstitution to solve for w, the array of weights for each
   RBF: $w = V \cdot S \cdot U^T \cdot \boldsymbol{Z}^{des}$, Equation 2.27.
4. Output the array of weights, $w = \{w_1, w_2, ..., w_M\}$.

RBF testing phase approximation algorithm.

1. Given: input dataset $V = \{v_1, v_2, ..., v_N\} \subset R^D$,
   cluster centers $\Omega = \{u_1, u_2, ..., u_M\}$ and $w = \{w_1, w_2, ..., w_M\}$.
   Determine β, an experimental node width factor.
2. For r=1,2,...,M
   $q_r = 0$
   For k=1,2,...,N
   If v_k is closest to u_r
   $q_r = q_r + 1$
   next k.
   next r.
3. For r=1,2,...,M

$$\sigma_r = \beta \sqrt{\frac{1}{q_r} \sum_{k=1,N} B_{kr} (v_k - u_r)^2}$$

   /* Recalling that B_{kr} is a function determining if v_k is closest to u_r */
   next r.
   For k=1,2,...,N
   For r=1,2,...,M

$$Y_{k,r} = e^{\left(- \frac{\|v_k - u_r\|^2}{(\sigma_r)^2} \right)}$$ /* from Equation 2.24 */

   next r.
   next k.
4. For k=1,2,...,N

$$Z_k^{out} = \sum_{r=1,M} w_r \cdot Y_{k,r}$$

   next k.
5. Calculate *NPE* for Z^{out}.

Procedure for network evaluation in this research.

For each size network (16, 32, 50, 100)
    For both quantization components
        For a range of quantization parameters
            Perform the quantization.
            Perform the Neural Gas hyper-planes prediction.
            For a range of the RBF parameter β
                Perform the RBF prediction.
            next β.
            Record the best *NPE* and parameters for both approximation techniques and the quantization technique.
        next set of quantization parameters.
    next quantization component.
next network size.

This procedure provides results from each quantization run. Quantization parameters, $E^{RMS\ quant}$, approximation parameters and the best *NPE* are recorded.

Vita

Michael Cozart was born in Nashville, Tennessee on March 7, 1957. He attended schools in the Metropolitan Nashville Public School System where he graduated from Hillsboro High School in 1975. He received the Bachelor of Science degree in Wildlife Biology from the The University of Tennessee at Martin in 1979. He received the Bachelor of Science degree in Computer Science from Middle Tennessee State University, Murfreesboro, Tennessee in 1984. In 1990 he entered the graduate program at The University of Tennessee Space Institute. He received the Master of Science degree from The University of Tennessee in 1996. He is presently employed as a systems programmer at Arnold Engineering Development Center, Arnold AF Base, Tennessee.