Performance Modeling of a Class of Queuing Systems with Self-Similar Characteristics.

by

Homayoun Yousefi'zadeh

A Dissertation Presented to the
FACULTY OF THE GRADUATE SCHOOL
UNIVERSITY OF SOUTHERN CALIFORNIA
In Partial Fulfillment of the
Requirements for the Degree
DOCTOR OF PHILOSOPHY
(Electrical Engineering)

May 1997

Dedication

To my parents, *Bahar* **and** *Naser*
with my deepest love

Acknowledgments

I would like to take this opportunity to thank all of the great people who have helped me accomplish this important task. My special thanks go to John Silvester and Edmond Jonckheere, my committee chair people for their guidance and advice while pursuing toward Ph.D. degree at the University of Southern California. I would also like to thank Deborah Estrin and Scott Shenker my committee members for their time and efforts in completing this work.

By dedicating this dissertation to my parents, I have tried to express my deepest respect to two most important people in my life without whom I was never able to stand at the spot where I am standing today. I also want to send my love and cherish to my brother Hooman and my sister Hedieh who have always been a great support for me in different stages of my life.

Next, I wish to send my greetings and salutes to my wonderful colleagues at Procom Technology management team and software engineering department for their understanding, help, encouragement, and protection. It is certainly my pleasure to appreciate their efforts in providing a very dynamic, challenging and peaceful work environment during the last two years.

Finally, I am so grateful to have so many wonderful friends here in United States and back home in Iran. I wish to express my appreciation to those who have always been there for me in the most difficult times and supported me with their great attitudes. I specially want to thank Masoud Zargari, Ramin Hojati, Kidra Shadroo, Bahram Afghani, Nader Toghrol, Iman Adibi, Cyrus Shahabi, Majid Baba'zadeh and Reza Sadri for their enthusiastic pure friendship. Thank you all and I always keep your great memories.

Contents

List Of Tables

List Of Figures

Abstract

This dissertation report investigates the application of deterministic chaotic maps and perceptron neural networks to model the source- and aggregated-level bursty traffic patterns in packet based computer communication networks. It is motivated in part by recent measurement studies which indicate the presence of significant statistical features in packet traffic belong to the fractal nature of the processes rather than their stochastic nature and by widespread presence of self-similar patterns in modern high-speed network traffic. It starts by illustrating the required background about the tools being used, i.e., chaotic systems theory and perceptron neural networks, and continues by describing the statistical features of the measured traffic.

The report then outlines potential performance synthesis approaches based on chaotic maps and neural networks which can be used to assess the traffic significance of fractal properties and indicates that simple nonlinear models can capture a number of fractal properties observed in self-similar patterns.

Finally, the report shows the modeling power of the above mentioned tools by introducing new efficient chaotic- and neural-based techniques to artificially generate self-similar packet traffic, to reduce packet loss, and to estimate average packet latency in typical multiple source environments. The techniques may be applied to dynamic buffer management as well as latency estimation algorithms in computer communication networks.

The most important conclusion of the report is that chaotic maps and neural networks may provide practical tools to accurately model transient and steady state behavior of packet traffic generators despite encountering significant analytical and numerical difficulties.

Key Words: Chaos, Chaotic Maps, Chaotic Modeling, Fractals, Entropy, Perceptron Feedforward Neural Networks, Back Propagation Algorithm, Neural Network Modeling, Packet Network, ON-OFF Source Model, Bursty Traffic, Self-Similarity, Multiple Source Model, Packet Loss, Packet Latency, Dynamic Buffer Management, Scheduling.

Chapter 1

Chaotic Systems: An Analysis

1.1 Introduction

Chaos is a nonlinear phenomenon that manifests itself in many fields of science. Despite being singled out as an important research area only recently, chaos has been around for a long time. It has almost graced the innocent eyes of many scientists in the past, to be dismissed as physical noise. Now, it is known that chaos can readily occur everywhere nonlinearity is present. In addition, Picci has recently shown [35] that it is possible for chaos to occur in a linear system with infinite dimension. Studying chaotic phenomena in discrete time dynamical systems has been very attractive, because these maps occur naturally in many physical, social, and economical systems. They also arise frequently in the Poincare analysis of systems modeled by ordinary differential equations.

The study of chaotic systems' theory is done as follows. Section 1.2 contains the definition of chaos and the relevant properties. In section 1.3, the qualitative behavior of discrete time transformations is reviewed briefly. Section 1.4 deals with the dynamical systems classifications and ergodic theory. Section 1.5 introduces some useful chaos measurement tools including fractal dimensions, Lyapunov exponents, and the concept of entropy based on the Kolmogrov-Sinai theorem.

1.2 Dynamical Systems and Chaotic Behavior

1.2.1 Dynamical Systems and Their Properties

Dynamical systems are generally divided into two categories as autonomous and non-autonomous systems.

Autonomous dynamical systems are described with the following state equation

$$\boldsymbol{x}_{n+1} = \boldsymbol{f}(\boldsymbol{x}_n) \tag{1.1}$$

where x_n is the d-dimensional state vector of the system at time n. The function $f : R^n \rightarrow R^n$ is called vector field. This is important to note that the vector field $f(.)$ is independent of time and hence the initial condition may be considered at $n = 0$. The solution of relation 1.1 is called a state trajectory. The dynamical system is linear if $f(x_n)$ is linear.

Non-autonomous dynamical systems are described with the following state equation where the vector field is a function of time and hence the solution of the system depends on the initial condition.

$$\boldsymbol{x}_{n+1} = \boldsymbol{f}(\boldsymbol{x}_n, n) \tag{1.2}$$

Again, the system is linear if $f(x, n)$ is linear. If $N > 0$ can be found such that $\forall n$ and x, $f(x_n, n) = f(x_n, n + N)$ then the system is called time periodic. The smallest N with the above mentioned property is called minimal period. A non-autonomous system of degree d can always be transformed to an autonomous system of degree $d + 1$ as follows.

$$\begin{aligned} \boldsymbol{x}_{n+1} &= \boldsymbol{f}(\boldsymbol{x}_n, \tfrac{\theta_n N}{2\pi}) \\ \theta_{n+1} &= \frac{2\pi}{N}\theta_0 = \frac{2\pi n_0}{N} \end{aligned} \tag{1.3}$$

Note that the above discrete time equations are obtained by generalizing the continuous time equations. If the solution of dynamical system ϕ_n is diffeomorphic[1], then the followings hold

(1) Trajectories of autonomous systems are specified uniquely from their initial conditions, i.e.,

$$[\phi(x_n) = \phi(y_n)] \equiv [x_n = y_n] \tag{1.4}$$

(2) Although the trajectories of non-autonomous systems are specified uniquely from their initial state, i.e.,

$$[\phi(x_n, n_0) = \phi(y_n, n_0)] \equiv [x_n = y_n] \tag{1.5}$$

[1] A function is called diffeomorphic, if it is invertible and its derivative and its inverse function derivative exist.

it is possible to have $[\phi(x_n, n_0) = \phi(y_n, n_1)]$ for $x_n \neq y_n$ and $n_0 \neq n_1$ or in other words trajectories of non-autonomous systems may intersect.

1.2.2 Steady State Behaviors and Limit Sets

First let me introduce some useful definitions.

Steady state is the asymptotic behavior of the system which is required to be bounded. The difference between the system solution and the steady state behavior is called transient behavior of the system.
A point y is a limit point of x if for every neighborhood u of y, $\phi(x_n)$, enters u alternately for $n \to \infty$.
The set of all limit points of x is called the limit set of x, $L(x)$.
If there exists an open neighborhood of limit sets such that $\forall x \in u \quad L(x) = L$ then the set is called an attracting limit set.
Basin of attraction, $B(L)$ of an attracting limit set is defined as the union of all of the neighborhoods of u with the above mentioned characteristic. Each trajectory starting from $B(L)$ will asymptotically tend to L.
The phrase strange attractor is used to describe the trajectories of chaotic systems. While there exists only one limit set for a linear system and the steady state behavior is independent of initial conditions, this is not the case for the nonlinear systems. In fact, every nonlinear system may have a number of limit sets and in specific attracting limit sets with different basin of attractions. The initial conditions then specify the basin of attraction that would attract the system trajectory.

There are four different steady state behaviors for a nonlinear system.

1.2.2.1 Equilibrium Points

For an autonomous system, an equilibrium point is the fixed solution of relation 1.1 in the form of $\phi(x_{eq}) = x_{eq}$. As an example, the fixed point of the logistic map may be mentioned. The logistic map is identified by the equation

$$x_{n+1} = 4Wx_n(1 - x_n) = f(x_n, W) \tag{1.6}$$

and has a set of fixed points $\{0, 3/4\}$. [2] This is important to note that the limit set for an equilibrium point is the equilibrium point itself.

1.2.2.2 Periodic Solutions

$\phi(x^*_n, n_0)$ is called a periodic solution if for all the times n and the minimal period $N\prime > 0$ the relation $\phi(x^*_n, n_0) = \phi(x^*_{n+N\prime}, n_0)$ holds. For a non-autonomous system $N\prime$ is generally a multiple of system's period.

Definition 1.1 [32]: A periodic solution is called isolated if there exists a neighborhood of it such that there is no other periodic solution in that neighborhood.

For an autonomous system, a periodic solution is called a limit cycle which is a self-sustained oscillation and is not observed in the case of a linear system. Discretized Van der Pol's equation has a limit cycle in the form of

$$x_{n+2} + (x_n^2 - 1)x_{n+1} + x_n = 0 \tag{1.7}$$

which is obtained form its two-dimensional state space equation.

$$\begin{aligned} x_{n+1} &= y_n \\ y_{n+1} &= (1 - x_n^2)y_n - x_n \end{aligned} \tag{1.8}$$

1.2.2.3 Quasi Periodic Solutions

A quasi periodic solution can be expressed as a summation of a number of periodic functions as

$$x_n = \sum_j (h_n)_j \tag{1.9}$$

where h_i has a minimal period of N_i and frequency of $f_i = 1/N_i$. In addition, the set of base frequencies $\{f_1, ..., f_p\}$ can be found with the following characteristics:
(1) The elements of the set are linearly independent with the interpretation that it is not possible to find the set of nonzero integers $\{c_1, ..., c_p\}$ such that $c_1 f_1 + ... + c_p f_p = 0$ holds.
(2) Every arbitrarily chosen frequency can be expressed as a linear combination of the base frequencies with nonzero integer multiples, i.e., $\exists\{c_1, ..., c_p\}$ such that $f_i = c_1 f_1 + ... + c_p f_p$.

[2] These are the fixed points of the first iteration of logistic map. There also exist some other fixed points for higher iterations. For more details see [37].

Note that although the number of the elements of the base set is fixed, the set is not specified uniquely. A quasi periodic solution is called p-periodic if its base frequency set has p elements. One of the simplest examples for this kind of behavior is the amplitude modulation in the form of

$$x_n = m_n sin(2\pi f_c n) \tag{1.10}$$

where m_n is the periodic message signal with period f_m. If f_c and f_m are incommensurate [3], then x_n is 2-periodic.

1.2.2.4 Chaotic Solutions

Unfortunately, there is no universally accepted mathematical definition for chaos. Practically, chaotic behavior may be considered as a steady state behavior of a system which is different from all of the above mentioned behaviors while the system still remains bounded. A chaotic spectrum is, in fact, very close to the noise spectrum and has a continuous nature in the frequency domain. The two most important characteristics of a chaotic system are as follow.

(1) The limit set of a chaotic system belongs to the cantor and fractal sets.

(2) A chaotic system is extremely sensitive to the variations of initial conditions, i.e., two very close initial conditions can lead to have two uncorrelated steady state solutions. Since in practice it is not possible to specify the initial condition with an infinite precision, it will not be possible to predict the steady state behavior of a chaotic system! That is why chaotic systems have been described in the past as deterministic systems which exhibit random behaviors.

Based on the above mentioned characteristics, it is possible to introduce a mathematical definition for chaos.

First remind that the composition of two functions is denoted by $f \circ g(x) = f(g(x))$ and the n-fold composition of a function with itself is denoted by

$$f^n(x) = \underbrace{f \circ f \circ ... \circ f}_{\text{n-times}}(x)$$

[3]Two real numbers are called incommensurate if they do NOT have a rational ratio.

Now, again, the fixed point of a function f is described by $f(x^*) = x^*$ and the periodic point of period n is identified by $f^n(x^*) = x^*$.

Definition 1.2 [45]: A map $f : J \to J$ is called topologically transitive, if for any pair of open sets $U, V \subset J$ there exists $k > 0$ such that $f^k(U) \cap V \neq \emptyset$.

Hence, topologically transitivity implies that the map has points which eventually move under iteration from one arbitrarily small neighborhood to any other. The map is therefore irreducible, i.e., J cannot be decomposed into subsets that remain disjoint under iterations of the map. This property is termed mixing by Peitgen et al and has a close relationship with the concept of ergodicity that more would be said about, later in this report.

Definition 1.3: $f : J \to J$ has sensitive dependence on initial conditions (SIC) if there exists $\delta > 0$ such that, for any $x \in J$ and any neighborhood N of x, there exists $y \in N$ and $n \geq 0$ such that $|f^n(x) - f^n(y)| > \delta$.

Again, SIC implies that two trajectories starting with almost identical initial conditions will gradually escape from each other or in other words the uncertainties in initial conditions are magnified at an exponential rate when SIC presents, making the steady state behavior of dynamical system unpredictable. Remind that SIC only requires the existence of one point in every neighborhood of x to separate from x under iteration, and not every $y \in N$ need exhibit this behavior.
The following definition of chaos is due to Devaney [46]:

Definition 1.4: Let V be a set. A map $f : V \to V$ is called chaotic on V if
(1) f has sensitive independence on initial conditions.
(2) f is topologically transitive.
(3) Periodic points are densed in V.

There are also other definitions that call systems satisfying condition (1) weakly chaotic and those satisfying definition 1.4 strongly chaotic. Although condition (1) is popularly taken to be the defining property of chaos -since it is associated with the unpredictability of the chaotic systems- Banks et al [44] have recently shown that condition (1) can be obtained from conditions (2) and (3) and is therefore redundant.
As an interesting example, a class of maps may be introduced as follows:

Example 1.1: The class of maps $E = \{f\}, f : [0,1] \rightarrow [0,1]$ satisfying the following conditions

(1) There is a partition $0 = a_0 < a_1 < ... < a_r = 1$ of $[0,1]$ such that for each integer $i = 1, .., r$ the restriction of f to the interval $(a_{i-1}, a_i), f_i$ is a C^2 function;

(2) f_i is onto, i.e., $f(a_{i-1}, a_i) = (0,1)$, $i = 0, ..., r$;

(3) There exists $\lambda > 1$ such that $f'(x) \geq \lambda$ for $x \neq a_i, i = 0, ..., r$.

is strongly chaotic. See [5] for the proof.

1.3 Qualitative Behavior of Discrete Time Chaotic Maps

Consider a real closed interval $I = [a, b]$ and a nonlinear function $f(.)$ which transforms any point x of I into a point x' in the same interval, i.e., $f : I \rightarrow I$. Choose an arbitrary initial condition x_0 in I and iterate it via the algorithm

$$x_{n+1} = f(x_n) \qquad n = 0, 1, 2, ... \tag{1.11}$$

As it was mentioned earlier, relation 1.11 is called a discrete map on the interval I. This scalar discrete map can be generalized into a d-dimensional interval map via the vector equation

$$\boldsymbol{x}_{n+1} = \boldsymbol{f}(\boldsymbol{x}_n) \tag{1.12}$$

where $\boldsymbol{x}_{n+1}$ and $\boldsymbol{x}_n$ are d-dimensional vectors and $f(.)$ is a nonlinear vector function on $\boldsymbol{x}_n$. Each component x_{n_i} of $\boldsymbol{x}_n$ takes values in a closed interval $I_i = [a_i, b_i]$. In spite of their apparent simplicity, the dynamics of many simple nonlinear, even polynomial, discrete maps are extremely complicated. They may have stable fixed points, periodic orbits of different periodicity, and chaotic regimes. Moreover, bifurcation among these behaviors can be observed in a parameter-dependent family of maps over various non-empty parameter ranges.

As a benchmark example, the behavior of one-dimensional discrete-time logistic map is studied. Remind that relation 1.6 gives the equation of the map. Here the interval I is identified as $I = [0,1]$ and W is a parameter which varies from 0 to 1. The logistic map is an example of a one-parameter family of discrete maps which transform the interval I into itself. Although logistic map is a very peculiar one, its qualitative behavior holds for a broad class of one-dimensional maps. Specifically, for those maps where the function $f(.)$ in 1.11 is a differentiable unimodal function, the same qualitative behavior as for the logistic map is observed. The rate of the decay $\frac{w_{k+1}-w_k}{w_k-w_{k-1}}$ of the interval width between period doubling for logistic map has been found to be the same for any differentiable unimodal function. This property is, sometimes, referred to as metrical universality. Figure 1.1 indicates the fixed points, and periodic points of logistic map.

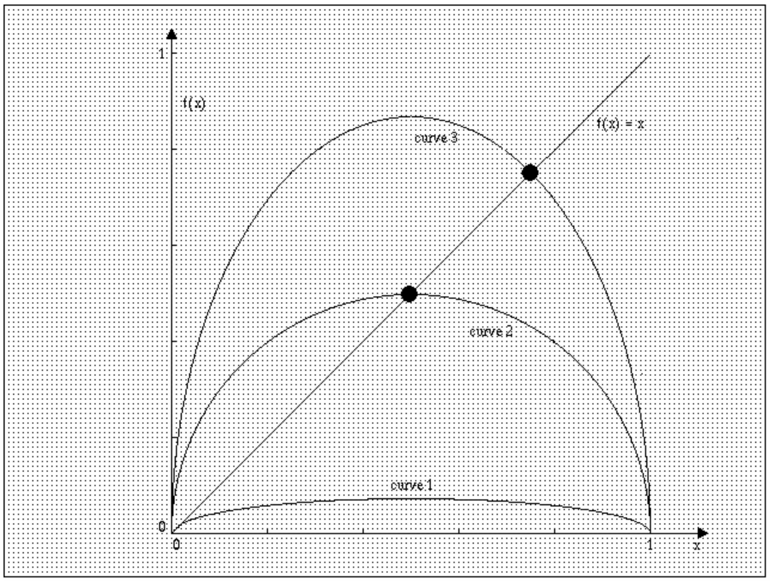

Figure 1.1: Fixed points of logistic map for $W < 0.25$ (curve 1) $0.25 < W < 0.75$ (curve 2) and $W > 0.75$ (curve 3).

Higher-dimensional discrete maps can be either conservative (area-preserving) or dissipative (area-contracting), invertible or non-invertible. While some results on one-dimensional mappings can be generalized to higher dimensions, the latter are usually much richer in their dynamical behavior than the former. For instance, consider the famous two-dimensional Henon map described by

$$\begin{aligned} x_{n+1} &= 1 - Ax_n^2 + y_n \\ y_{n+1} &= Bx_n \end{aligned} \tag{1.13}$$

This map which reduces to a one-dimensional quadratic map for $B = 0$ reveals many new phenomena quite typical of multi-dimensional discrete maps. First, the asymptotic behavior of the system depends on the initial point, i.e., different initial points could give rise to different periodic or aperiodic orbits. This behavior can not occur in one-dimensional unimodal maps, where at most one stable period-1 orbit can exist. Second, for certain parameters and initial points, the system converges to an attractor with a self-similar internal structure. Third, it has been proved that there exist intersections between stable and unstable manifolds of the Henon map. These intersections which are called homoclinic points, give rise to extremely complicated dynamics, including chaos.

1.4 Abstract Dynamical Systems, Ergodicity and Histogram Analysis

In this section, an analytical study of dynamical systems is given. This study is worth doing since it plays a crucial role on the development of analytical tools for measuring chaos.

1.4.1 Abstract Dynamical Systems

The concept of abstract dynamical systems (ADS) was first introduced by Kolmogrov. An ADS consists of four main elements shown by the notation (X, Σ, μ, T) where X is the sample space, Σ is an algebra of events, e.g., σ-algebra of subsets of X which is a subset that remains stable under the traditional changes, μ is the invariant measure, and finally T is the dynamical shift. The first three components, in deed, deal with the probabilistic aspect of the system and are also called probability space, and the fourth one deals with the dynamical properties of the system and is defined as $T : X \to X$.
As it was mentioned intuitively, μ has the measure preserving property, i.e., for all measurable subsets A of the sample space X, $\mu(A) = \mu(T^{-1}A)$ for $T : x_n \mapsto x_{n+1}$ [4]. Note that in some books, the measure preserving property is described as $\mu(A) = \mu(TA)$ although, for non-invertible maps the above statements are not the same.

Logistic map with the following elements is a suitable example for an ADS. The sample space X is defined as $X = [0, 1]$, Σ is defined as measurable subsets of $[0, 1]$, the invariant measure μ is $\mu(dx) = \frac{dx}{\pi\sqrt{x(1-x)}}$ and the dynamical shift $T : [0, 1] \to [0, 1]$ is $T : x \mapsto 4x(1 - x)$. Now, it remains to show that the invariant measure has the measure preserving property.

[4] T is defined as $T^{-1}A = \{w \in X : Tw \in A\}$ and $T^{-1}A$ is the inverse image of A.

Proof: Suppose x and y are the roots of the equation $z = 4S(1-S)$ or in other words x and y are the inverse images of z. It yields that

$$\begin{aligned} z &= 4x(1-x) \\ z &= 4y(1-y) \end{aligned}$$

hence it needs to be shown that $T^{-1}[z, z+dz] = [x, x+dx] \cup [y, y+dy]$ where $T : x_n \mapsto x_{n+1}$ is the dynamical shift defined as $T(x_n) = 4x_n(1-x_n)$. Consequently, one needs to check whether the relation $|\mu(dx)| + |\mu(dy)| = |\mu(dz)|$ holds.

$$\mu(dx) = \frac{dx}{\pi\sqrt{x(1-x)}} = \frac{1}{4\pi} \frac{1}{\sqrt{\frac{1-\sqrt{1-z}}{2}(1-\frac{1-\sqrt{1-z}}{2})}} \frac{dz}{\sqrt{1-z}}$$

After a little bit of mathematical manipulation, one can simply obtain the following relation:

$$\mu(dx) = \frac{1}{2} \frac{dz}{\pi\sqrt{z(1-z)}}$$

Similarly the expression for the $\mu(dy)$ is given by

$$\mu(dy) = \frac{-1}{2} \frac{dz}{\pi\sqrt{z(1-z)}}$$

This completes the proof as $|\mu(dx)| + |\mu(dy)| = |\mu(dz)|$. Q.E.D

Also, note that $\int_0^1 \mu(dx) = \int_0^1 \frac{dx}{\pi\sqrt{x(1-x)}} = 1$. Hence logistic map is an ADS by the sense of Kolmogrov.

It is worth mentioning that the ADS is a generalization of the probability space which is the foundation of stochastic theory. A stochastic process may be viewed as an ADS if one can find the dynamical shift such that the measure preserving property holds.

1.4.2 Ergodic Theory, and Histogram Analysis

Definition 1.5 [35]: Assume that (X, Σ, μ, T) is an ADS with a measure preserving property, i.e., $\mu(A) = \mu(T^{-1}A)$. Poincare recurrence scheme simply says that for any arbitrary initial condition x_0 in any measurable subset E of the sample space X with $\mu(E) > 0$, the iterative map resulting from affecting dynamical shift on that initial condition, returns to the subset E infinitely many often. The proof is based on the contradiction and is given in [35].

Now, based on the Poincare recurrence scheme, the histogram analysis may be introduced as an elegant method for obtaining a unique measure under the system flow for an ergodic system. The concept of ergodicity is defined later in this section. The analysis approximates the invariant measure of a subset E as the following limit.

$$\lim_{n\to\infty} \frac{\text{No. of times} \quad x_k \quad \text{returns to} \quad E}{n} \cong \mu(E) \tag{1.14}$$

Mathematically, the above limit is expected to be the time average based on the concept of the indicator of a set [5] as shown in figure 1.2. As the result of applying histogram

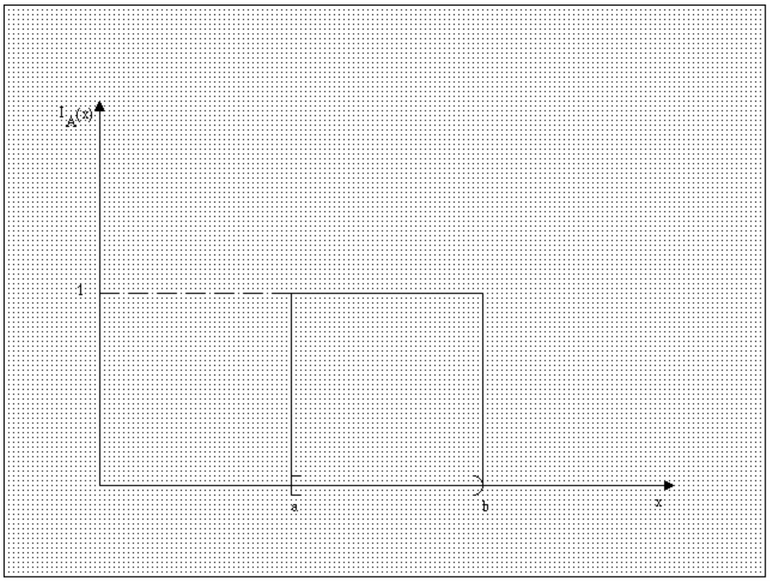

Figure 1.2: The indicator function of a set.

analysis,

$$\lim_{n\to\infty} \frac{\sum_{k=1}^{n} I_E(x_k)}{n} = \int I_E(x)\mu(\,dx) \tag{1.15}$$

Now the idea can be used to estimate any function over a given interval since each function f may be approximated by $f(x) = \alpha I_A + \beta I_B + ...$ and hence

$$\lim_{n\to\infty} \frac{\sum_{k=1}^{n} f(x_k)}{n} = \int f(x)\mu(\,dx) \tag{1.16}$$

[5]The indicator of a set is one over the set and zero otherwise.

or with $x_k = T^k(x_0)$

$$\lim_{n\to\infty} \frac{\sum_{k=0}^{n-1} f(T^k x_0)}{n} = \lim_{n\to\infty} \frac{1}{n} \sum_{k=0}^{n-1} f(T^k x_0) = \overline{f} \tag{1.17}$$

Conceptually, if the limit exists and does not depend on the initial condition, the system is said to be ergodic. The concept of mixing was introduced earlier in this chapter. It is, now, related to the concept of ergodicity directly. Suppose that the ADS (X, Σ, μ, T) is a decomposition, i.e., there exist subsets E and F of the sample space X such that $X = E \cup F, E \cap F = \emptyset$ and also $\mu(E) > 0$, $\mu(F) > 0$, $T^{-1}E = E$, $T^{-1}F = F$. In other words if one starts iterating from either of those regions (s)he never goes to the other region. Now based on the result of histogram analysis, each region has the measure zero and this is a contradiction. Since the histogram analysis is true only in the case of ergodic ADS's the obvious result is that the system is not ergodic. Therefore the ADS is ergodic, if and only if it is not a decomposition. This means ergodicity is the lack of mixing.

1.5 Measuring Chaos: Analytical Tools

1.5.1 Fractal Dimensions

In this section, the concept of dimension is introduced. Based on the various definitions of dimension, it is very convenient to specify chaotic systems. A system may simply be considered chaotic if the attractor corresponding to it has a non-integer dimension. Dimension has been defined from many different points of view in the literature. One of the simplest definitions based on the Brouwer's Domain invariance theorem says an attractor may be assumed to be d-dimensional if it looks like an open subset of R^d in a neighborhood of all of the points belonging to it.

Brouwer's Domain invariance theorem mentions that the dimension is preserved under a homeomorphic mapping, i.e., the map is continuous and continuously invertible. As a direct result, there does not exist a homeomorphic mapping between Euclidean spaces with different dimensions, e.g., $R^2 \to R^3$. For example, a limit cycle is one-dimensional since it looks like an interval. Strange attractors usually have non-integer dimensions because they do not look like any specific Euclidean space.

Poincare in his definition of dimension says a space has dimension d if and only if its

"walls" have dimension $d-1$. Obviously, this is a recursive definition of dimension cause the definition of the so-called walls should be defined.
In the following, five different definitions of dimension are discussed. In some cases illustrative examples are also given.

1.5.1.1 Capacity Dimension

It is almost the simplest defined dimension. The main goal here is to cover an attractor with simple volume elements of diameter ϵ such as sphere and/or cube. Consider an attractor A, and suppose $N(\epsilon)$ is the required number of volume elements for covering A. As the diameter ϵ becomes smaller the covered volume by the volume elements gets closer to the real volume of A. Let A be D-dimensional, then the required number of volume elements is inversely proportional to the ϵ or in the other words $N(\epsilon) = k\epsilon^{-D}$. The capacity dimension is, then, defined by solving the last equation for D when the parameter ϵ tends to zero.

$$D_{cap} = \lim_{\epsilon \to 0} \frac{\ln(N(\epsilon))}{\ln(1/\epsilon)} \tag{1.18}$$

There are two important points that need to be considered. First, if the limit does not exist, the capacity dimension is undefined. Second, if there are a number of different ways to cover the attractor, then the capacity dimension is the minimum number among all different situations.
The most popular example for an attractor with a fractal dimension is the middle third cantor set. Consider the interval $[0,1]$ and divide it to three equal parts, then drop the middle part and repeat the action for each of two remaining parts separately. The middle third cantor set results from doing the same procedure indefinitely. Figure 1.3 illustrates the idea. For the middle third cantor set the capacity dimension is obtained by assuming $\epsilon = 1/3^n$ in which case $N(\epsilon) = 2^n$. Hence,

$$D_{cap} = \lim_{n \to \infty} \frac{\ln(2^n)}{ln(3^n)} = 0.6309...$$

1.5.1.2 Information Dimension

The capacity dimension does not include any information about the behavior of the dynamical system. The information dimension is a probabilistic dimension based on the number of the observed frequencies over a state trajectory. Again the idea is to cover the

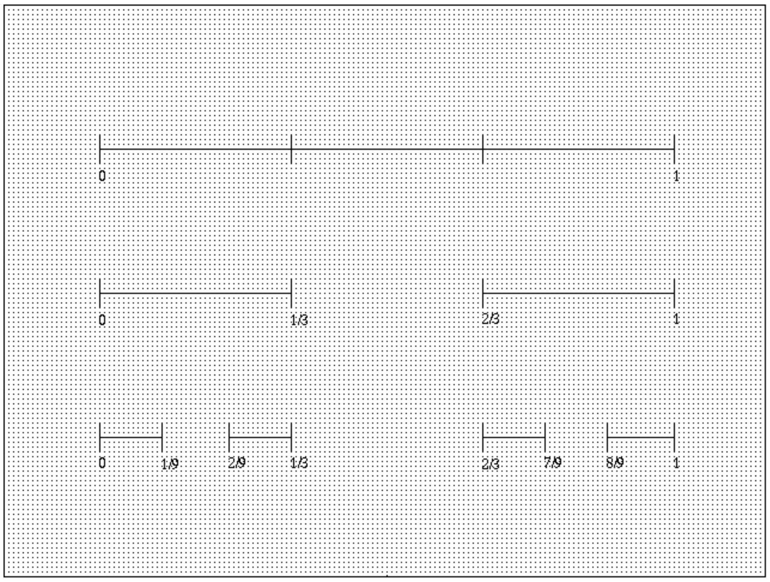

Figure 1.3: Middle third cantor set.

attractor with the volume elements of diameter ϵ but the information dimension is defined based on the concept of entropy which will be discussed in the next section.

$$D_i = \lim_{\epsilon \to 0} \frac{\ln(S(\epsilon))}{\ln(1/\epsilon)} \tag{1.19}$$

where $S(\epsilon)$ is the entropy defined as

$$S(\epsilon) = -\sum_{i=1}^{N(\epsilon)} P_i \ln(P_i) \tag{1.20}$$

and where P_i is the relative frequency with which the state trajectory visits the i-th volume element.

1.5.1.3 Correlation Dimension

The correlation dimension is another probabilistic dimension based on the number of observed frequencies. It is defined as

$$D_c = \lim_{\epsilon \to 0} \frac{\ln(\sum_{i=1}^{N(\epsilon)} P_i^2)}{\ln(\epsilon)} \tag{1.21}$$

where

$$P_i = \lim_{N \to \infty} \frac{n_i}{N} \tag{1.22}$$

In the above expression n_i is the number of points inside the i-th volume element, and N is the total number of required volume elements.

1.5.1.4 Lebesgue and Hausdorff Measure of Dimension

In this case suppose the fractal set or strange attractor can be embedded in a d-dimensional Euclidean space. The main idea, again, is to cover the attractor with a number of volume elements u_i. For each volume element which can be chosen arbitrarily, the diameter is defined as

$$diam(u) = \sup_{x,y \in u} ||x - y|| \tag{1.23}$$

Now, suppose a covering of fractal set E is taken by sets u_i such that $E \subseteq \cup_i u_i$ with $diam(u_i) \leq \epsilon$, then the S-dimensional Hausdorff measure of set E is given by

$$H^S(E) = \lim_{\epsilon \to 0} \sum_i (diam(u_i))^S \tag{1.24}$$

where $\epsilon = \inf_i diam(u_i)$. Only if the volume elements are supposed to be hypercubes, the Lebesgue measure of E is defined as

$$L(E) = \inf_i \sum_i vol(u_i) \tag{1.25}$$

As a comparison, it makes sense to apply the results on the middle third cantor set. For the cantor set, the diameter is, in fact, the length specified by $(2/3)^j$ for the j-th iteration and the only way to have a finite value for $H^S(E)$ is that $3^{-S} = 2^{-1}$ in which case the dimension is $S = \log_3 2$ and the measure itself is one.

It can be seen that $0 < S < 1$ which is a consistent result. The Lebesgue measure of the middle third cantor set is zero because $\inf_{j \to \infty} (2/3)^j = 0$ and this, in general, is a valid result, i.e., the Lebesgue measure of strange attractors is always zero.

1.5.2 Lyapunov Exponent, and Kolmogrov-Sinai Entropy

In this section, two analytical tools for measuring the degree of chaoticness are introduced. The first tool, Lyapunov exponent, measures how sensitive the system is to the variations of the initial conditions while the second one, Kolmogrov-Sinai entropy, measures the rate of information change in an ADS. Although, the measured quantities are some how different, both imply the same intuitive result and that is the higher the measured value, the higher the degree of chaoticness of the system.

1.5.2.1 Lyapunov exponents

Consider the one-dimensional iterative equation $x_{n+1} = f(x_n)$ which is the state equation of a discrete-time system, with x_0 as the initial condition. Then, the Lyapunov exponent of such a system is defined as

$$\lambda(x_0) = \lim_{n\to\infty} \frac{1}{n} \log \frac{df^n(x_0)}{dx_0} \tag{1.26}$$

and considering $\frac{df^n(x_0)}{dx_0} = f'(x_{n-1})...f'(x_0)$ relation 1.26 can also be written in the form of

$$\lambda(x_0) = \lim_{n\to\infty} \frac{1}{n} \sum_{i=0}^{n-1} \log |f'(x_i)| \tag{1.27}$$

The following example explains the idea more clearly. For the continuous-time system $\dot{x} = \lambda x$, the discretized version of the dynamical system is expressed as $x_{n+1} = ax_n$ where $a = e^{\lambda T}$ and T is the sampling period. The Lyapunov exponent, therefore, is obtained by observing $f(x) = ax$ and $f'(x) = a$. Consequently, $\lambda(x_0) = \lambda$.
One of the nice features of the definition of Lyapunov exponents is that they are invariant under the change of coordinates, i.e., if $y = h(x)$ and $h(.)$ is a homeomorphic [6] map, the Lyapunov exponents $\lambda(x_0)$ and $\lambda(y_0)$ are the same because

$$\begin{aligned}
\lambda(y_0) &= \lim_{n\to\infty} \frac{1}{n} \log \frac{dy_n}{dy_0} \\
&= \lim_{n\to\infty} \frac{1}{n} \frac{d}{dy_0} \overbrace{h(f^n(\underbrace{h^{-1}(y_0)}_{x_0}))}^{y_n} \\
&= \lim_{n\to\infty} \frac{1}{n} h'(x_n) \frac{dx_n}{dx_0} \frac{dx_0}{dy_0} \\
&= \underbrace{\lim_{n\to\infty} \frac{1}{n} \log \frac{h'(x_n)}{h'(x_0)}}_{zero} + \lim_{n\to\infty} \frac{1}{n} \log \frac{dx_n}{dx_0} \\
&= \lambda(x_0)
\end{aligned} \tag{1.28}$$

[6] A function is called homeomorphic, if it is invertible, it is a continuous and its inverse function iscontinuous

The definition of Lyapunov exponents may simply be generalized for multi-dimensional dynamical systems. Suppose $\boldsymbol{x_0}$ is a d-dimensional vector. Let $\{\lambda_1, \lambda_2, ...\}$ be the set of eigen values of the Jacobian matrix [7], then the Lyapunov exponent is defined as

$$\Lambda(\boldsymbol{x_0}) = \lim_{n\to\infty} \frac{1}{n} \sum_{|\lambda_i|>1} \lambda_i \tag{1.29}$$

1.5.2.2 Kolmogrov-Sinai Entropy

The second analytical tool for measuring the degree of chaoticness of a system is the Kolmogrov-Sinai entropy. It basically measures the rate of the information change in an abstract dynamical system.

Suppose (X, Σ, μ, T) is an ADS with the measure preserving property $\mu(A) = \mu(T^{-1}A)$. Now suppose the set $\{A_1, ..., A_m\}$ is a measurable partitioning of the sample space X, i.e., $X = A_1 \cup ... \cup A_m$ where for $i \neq j$, $A_i \cap A_j = \emptyset$ and $\mu(A_i \cap A_j) = 0$, then the entropy of the partitioning is defined as

$$h(A) \stackrel{\text{def}}{=} -\sum_i \mu(A_i) \log \mu(A_i) \tag{1.30}$$

It is claimed that "h(course partitioning) $\leq$ h(fine partitioning)" and the reason intuitively is that with a finer partition a higher amount of information is accessed. It is also claimed that if the set $A_1 \cup ... \cup A_m$ is a measurable partitioning of the sample space then so is the set $T^{-1}A_1 \cup ... \cup T^{-1}A_m$. The proof has two steps. First note that,

$$\begin{aligned} X &= T^{-1}X = T^{-1}(A_1 \cup ... \cup A_m) \\ &= \{w : Tw \in (A_1 \cup ... \cup A_m)\} \\ &= \{w : (Tw \in A_1) or ... or (Tw \in A_m)\} \\ &= \{w : (w \in T^{-1}A_1) or ... or (w \in T^{-1}A_m)\} \\ &= T^{-1}A_1 \cup ... \cup T^{-1}A_M \end{aligned} \tag{1.31}$$

second, with the same line of reasoning one can show

$$T^{-1}A_i \cap T^{-1}A_j = \emptyset, \qquad i \neq j \tag{1.32}$$

[7]The $m \times m$ partial derivative matrix with respect to each component.

The set $A \vee T^{-1}A$ is called a common refinement of the sample space X and the sample space may be expressed as $X = \cup_{i,j}(A_i \cap T^{-1}A_j)$. The common refinement partitioning obviously includes a higher degree of information and if this procedure is kept doing, finer and finer common refinements result which contain higher and higher amount of information. The concept of entropy is defined based on the above mentioned observation as

$$KS = \sup \lim_{n \to \infty} h(A \vee T^{-1}A \vee ... \vee T^{-n+1}A) \tag{1.33}$$

where 'sup' is taken over all finite measurable partitionings. Sinai showed that there exists a finite partitioning with the property that its usage drops the need of having the 'sup' operator. For the piecewise linear operators, there is a straight forward method for choosing such a partition. The partition simply consists of segments of sample space corresponding to each linear piece of the map.
This is shown in the following example. For the tent map defined as

$$x_{n+1} = \begin{cases} 2x_n & : \quad (0 \leq x_n < 1/2) \\ 2(1 - x_n) & : \quad (1/2 \leq x_n \leq 1) \end{cases} \tag{1.34}$$

the entropy is calculated with the choose of a generating partition in the form of $[0, 1/2] \cup [1/2, 1]$. The common refinement partitions are obtained as

$$X = A_1 \cup A_2 = [0, 1/2] \cup [1/2, 1] \tag{1.35}$$

$$\begin{aligned} X &= A \vee T^{-1}A \\ &= (A_1 \cap T^{-1}A_1) \cup (A_1 \cap T^{-1}A_2) \cup (A_2 \cap T^{-1}A_1) \cup (A_2 \cap T^{-1}A_2) \\ &= [0, 1/4] \cup [1/4, 1/2] \cup [1/2, 3/4] \cup [3/4, 1] \end{aligned} \tag{1.36}$$

and the entropy of the each partitioning is calculated as

$$h(A) = -\sum_{1,2} 1/2 \log_2 1/2 = 1 \tag{1.37}$$

$$h(A) = -\sum_{1,2,3,4} 1/4 \log_2 1/4 = 1 \tag{1.38}$$

Similarly, the entropy is computed as

$$KS = \lim_{n \to \infty} h(A \vee T^{-1}A \vee ... \vee T^{-n+1}A) = 1 \tag{1.39}$$

Similar to the Lyapunov exponents, entropy is fixed under the change of coordinates. Only the interpretation is a little bit different in this case. The idea is best explained by assuming that the invertible function h(.) maps the ADS $(X_1, \Sigma_1, \mu_1, T_1)$ to $(X_2, \Sigma_2, \mu_2, T_2)$. Then $KS(X_1, \Sigma_1, \mu_1, T_1) = KS(X_1, \Sigma_2, \mu_2, T_2)$. The proof is fairly complicated and is skipped in this report. Please see [35] for the proof.

The entropy has a continuous spectrum from zero to infinity, when one starts from the regular deterministic systems and goes toward stochastic systems. For the regular deterministic systems the entropy is zero, for the chaotic systems it is a positive finite number, and for the stochastic systems it is infinity. The latter is true because the entropy is fixed under any dynamical shift. Now, since every stochastic process can be obtained from the output of a dynamical system with the white noise as the input, and since the entropy of white noise is infinity so is the entropy of the stochastic process.

For an ADS it is possible to obtain an upper bound for the value of entropy. Suppose (X, Σ, μ, T) is an ADS, then the following inequality holds.

$$KS(X, \Sigma, \mu, T) \leq N \log \lambda \tag{1.40}$$

where N is the dimension of the sample space, i.e., $N = dim(X)$ and λ is the amplification factor of boundary measure defined as

$$\lambda = \sup_{A} \frac{S(T^{-1}\partial(A))}{S(\partial(A))} \tag{1.41}$$

The operator ∂ is the boundary of a geometric object specified as ∂A = closure(A) - interior(A) and the operator S indicates the length of the boundary ∂A. For more detailed information please see [35].

At the end of this section, it is worth mentioning that although in both cases the map is said to be chaotic if the measure - either Lyapunov exponent or the entropy - is positive and finite, it has been shown [35] that in some cases such as the automorphism of the torus, the relation "$entropy = \log(Lyapunov \quad exponent)$" holds and hence, it is safer to say the system is chaotic if its Lyapunov exponent is greater than one.

Chapter 2

Neural Networks: Theory and Applications

2.1 Introduction

In this chapter, the fundamental precepts of the neural networks are briefly reviewed. Neural networks are a class of nonlinear systems which are able to learn and to perform tasks done by other systems. They are suitable for speech and signal processing, pattern recognition, systems modeling, and servo mechanism control. They acquire the required information based on the examples supplied to them. The various kinds of neural networks, generally, have energy functions. The learning procedure of neural networks is, indeed, nothing more than decreasing these energy functions until reaching local minimum levels. Neural networks are robust in the sense that if there is a relatively small error in the system, the network will continue its desired action. This characteristic of the neural nets makes them quite suitable for the traffic modeling task. In this chapter different kinds of neural networks along with their learning algorithms are studied. This includes Hopfield, Perceptron, Kohonen and Temporal Differences networks. The emphasis is on perceptron neural networks as the main modeling tool, their learning algorithm known as back propagation, and the ways of improving the performance.

2.2 General Description

The basic cellular unit of the nervous system is called neuron. It receives and combines different signals from other neurons via dendrites. The output signals are sent to the other neurons via axons. Synapses connect axons to dendrites. If the received dendrite signals are strong enough, they induce the neuron to create an output axon signal. Although this kind of data transformation has a chemical nature, it also has electrical effects which can be measured.

The human brain has approximately ten billion interconnected neurons. Data transfer through the synaptic connections has an electro-chemical nature and the amount of transmitted signal is a function of the chemical substance released by the axon and the chemical substance received by the dendrite. This chemical substance is called neurotransmitter and its electro- chemical interference forms the fundamental mechanism of the memory in the brain.

In an artificial neural network, the unit similar to the neuron is called processing element (PE). A PE has a large number of input paths and combines them by a simple weighted addition. Usually, the combined input is further processed by a transfer function. This transfer function may have the form of a threshold function which passes the information only when the combined input signal has reached a certain level, or it may be a continuous function from the combined input signal.

Normally, the output of the transfer function connects directly to the output path of the PE. The output path of the PE may be connected to the input paths of other PEs by a number of weighting functions. Each of the individual input signals to a PE is adjusted by these weighting functions before combining with other input signals in the transfer function. Figure 2.1 shows (a) a real neuron and (b) a PE. There are two phases in the

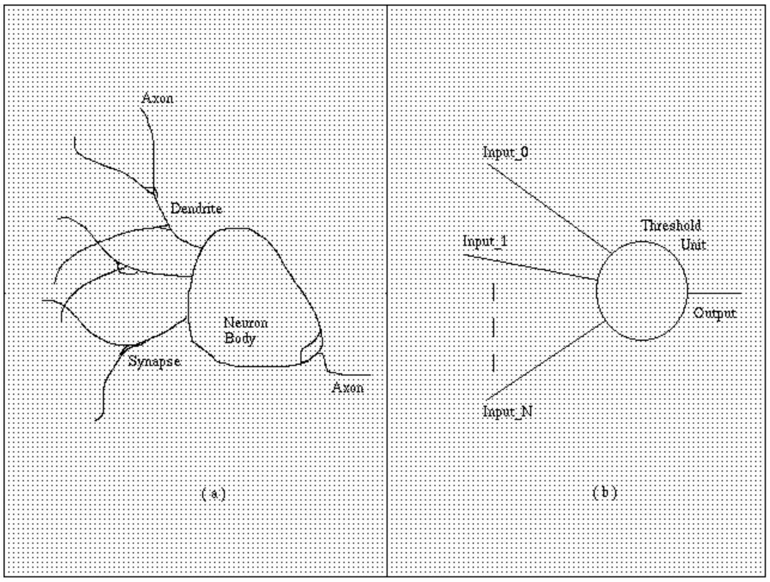

Figure 2.1: (a) Neuron body (b) Processing element.

operation of a neural network. The first phase is called learning and the second phase is called recalling. It is possible to have one combined learning and recalling phase in a network, although in most of the networks, they are separate. The learning procedure is, indeed, nothing more than the adjustment of weighting functions (synaptic strength) such that the network can respond suitably to the input stimulation. If the desired output(s) is(are) different from the input(s), then the trained network is called Hetero Associative (HA), otherwise it is called Auto Associative (AA). If in the learning process, the network does not have access to any desired output, this procedure is called Unsupervised Learning (UL) [1]. If the network does have access to desired outputs, the learning procedure is called Supervised Learning (SL). There is also a combination of the above two mentioned learnings which is called Reinforced Learning (RL). In the latter case, the supervisor just lets the network know whether its output is suitable for a specified input or not.

The number of samples required for training depends on the system which the network attempts to model. There are cases in which the network may need a very large number of samples to be trained.

A network enters the recalling phase when it is able to create a suitable response after the appearance of a specific stimulation. This phase may be a part of the learning phase as this is the case when the network output must be compared to a desired output in order to create an error signal.

A network may operate as a synchronous network, i.e, all of the PEs send their outputs at the same time or as an asynchronous network, i.e, every PE may send its output at a time independent of the other PEs.

There are two separate operations for the layers. The first one is called normalization. In this case the output of the layer is adjusted at a fixed level. In biological systems, this is done by connecting each PE in a layer to other PEs at the same layer. As a result every PE of a layer has a criterion about the total output of the layer and can adjust its output corresponding to that. Hence, the level of the operation of every layer remains fixed approximately. The second kind is called competitive operation. In the latter case, the operation of a layer is a consequence of interactions of PEs operations in that layer.

[1] UL is also called Self-Organized learning.

In spite of the first case, now only few PEs are able to create the layer output. As an example, a case may be mentioned in which only the PE with the highest level of activity will specify the output of the layer.

2.3 Differences among Neural Networks, Expert Systems, and Computer Algorithms

In this section the differences among neural networks, expert systems, and computer algorithms are investigated. As opposed to expert systems which obtain the required information from the explicit rules, neural networks obtain the required information from the samples they have accessed to. The learning procedure includes the adjustment of weighting functions such that the network is able to respond suitably to an input stimulation. In unsupervised learning, only a group of stimuli are given to the network and the network organizes itself such that each of the PEs of each hidden layer can respond suitably in the presence of a group of stimuli. In supervised learning, corresponding to each stimulus, the network gets a reading on the desired output in order to be able to create the suitable output. This kind of learning, generally, is related to one of the following methods:

- Hebbian Learning: The corresponding weighting function will increase, if both input and output levels are in a high level.
- Delta Learning: The weighting functions are adjusted such that the error between the desired output of the PE and its present output decreases.
- Competitive Learning: The PE with the strongest output to the input stimulus, has the chance to be closer to the desired output.

The second difference is the quality of information storage in neural networks. The weighting functions of a neural network perform its memory function. All available input/output pairs propagate in the network in order to increase the knowledge of the units and this is the adjustment procedure of the weighting functions. This mechanism allows the network to respond suitably to a deficient or noisy input.

The third difference is the robustness of the network to small errors. In computer algorithms, the existence of a small error leads to the function of the algorithm being disabled while this is not the case in neural networks, i.e, the existence of a smaller error or malfunctioning of a PE results in a small disturbance of the operation of the system. In addition, there exist networks which operate on the base of correcting the error. In these networks, occurrence of the error does not have any special effect on the final output.

The ability of the human mind to convert letters, words, sentences, and pages of a book to meaningful concepts is called pattern recognition and requires the simultaneous adaptation of a huge amount of information and their processing. Neural network have the pattern recognition property while expert systems and computer algorithms suffer from the lack of this property.

2.4 Hopfield Networks

The Hopfield model is a typical recursive model in which all of the nodes are connected to one another. Figure 2.2 shows a simple Hopfield network with four PEs. Using the

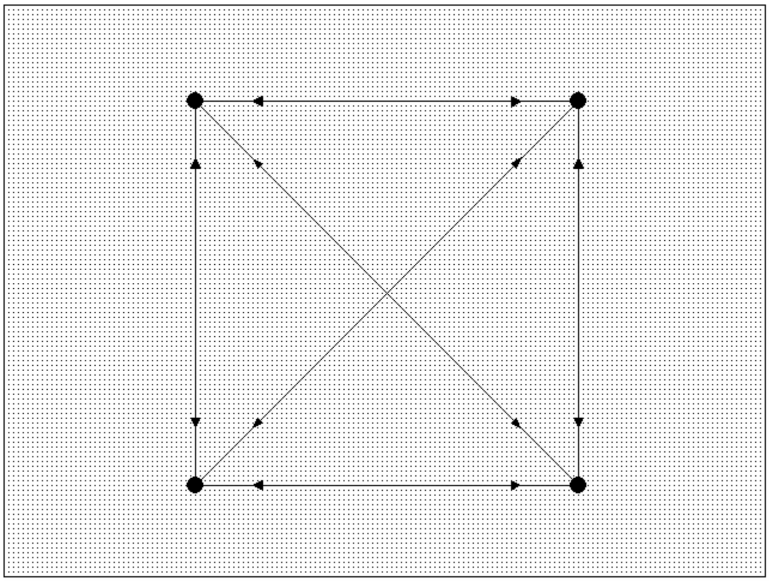

Figure 2.2: A fully recursive Hopefield network.

following notation

- x_i : The present output state of the i-th PE.
- $x_{j,prv}$: The previous output state of the j-th PE.
- w_{ij} : Weighting function of the connection between the i-th PE and the j-th PE.

the activity of each PE is determined by the overall signal received from all of the other PEs in the network according to the rule whether or not this signal exceeds a given threshold, i.e.,

$$x_i = \begin{cases} 1 & : \ (\sum_{i \neq j} w_{ij} x_{j,prv} > 1/2) \\ -1 & : \ (\sum_{i \neq j} w_{ij} x_{j,prv} < 1/2) \end{cases} \tag{2.1}$$

The rule corresponds to have an all-or-nothing threshold device, so the PE is always fully on or off, with the corresponding values set to 1 and -1 respectively. Historically, this is a model of restoring an image pattern from a corrupted noisy pattern. By suitably adjusting the weighting functions the above transition rule can be used to recall only image patterns from a predetermined set. This is done, for a pattern, by setting each weighting function either to 1, if the pixels x_i at either end of the link are both on, or -1 otherwise. The weighting functions of the each link for all patterns are added together to give an integer number which is the value of that link.

Hopfield's main contribution was to show that the action of the above mentioned algorithm leads the network to reach to a local minimum energy function by sliding the image patterns down valleys, known as basins of attractions [2], in a landscape where the bottom of the valleys lie the predetermined set. Give an incomplete or corrupted image, as it slides down the slope in the landscape, its wrong pixel values flip over to recover the complete image which lies at the bottom of the basin of attraction. This, in fact, is following the principle of least action, which drives the physical systems in equilibrium with their environment to states of least energy.

2.5 Perceptron Networks

The perceptron network is arguably the most popular neural network architecture,and certainly the trigger of the current widespread explosion of the activity in the field. A typical percpetron network is arranged in layers, but now only nodes in adjacent layers are connected. In addition, the layers are differentiated. A network usually contains several layers with complete or random connections between consecutive layers. The input layer takes the data from the outside environment, the output layer gives the data to the outside environment, and the other layers known as hidden layers process the data inside the network. The function of the network is to reproduce certain target output patterns at the

[2] This is the same concept discussed in chapter one.

last layer of nodes. This action which can be interpreted as a filtering action is achieved by adjusting the weighting functions of each interconnecting link according to a rule which compares the activity patterns at the output nodes with the desired target patterns and propagate the difference back through the network leading to have a small adjustment to each link's weighting function. Figure 2.3 shows a simple neural network with one hidden layer. A simple perceptron network does not have any feedback connection between two

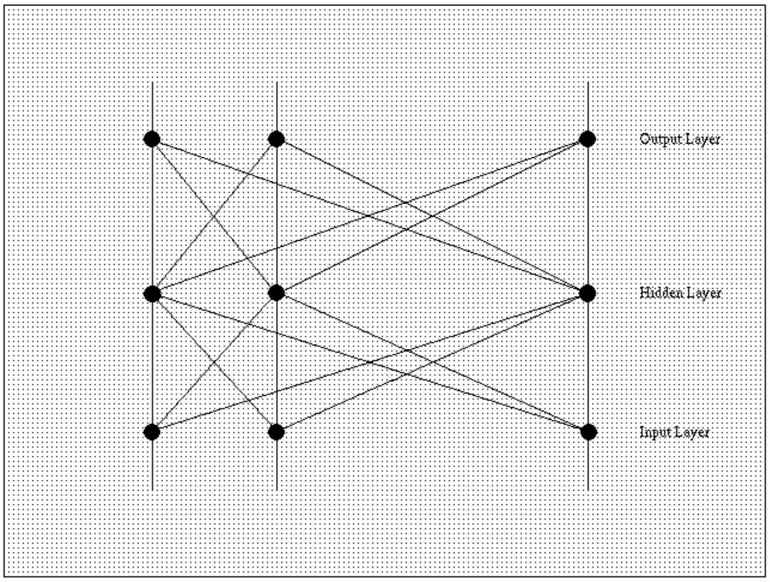

Figure 2.3: A simple neural network with one hidden layer.

different layers or a layer with itself. Such kind of network is called a feedforward network. In this situation, the input data from the input layer appears in the output layer via the interface of hidden layers. Feedforward networks are generally considered because of their nonlinear properties. The data flow in feedback networks is more complicated.

2.5.1 Back Propagation Algorithm

Generally, a neural network must learn how to classify input patterns. It has been experimentally observed that, as the number of layers of a network increases, it can classify more and more complicated patterns. A significant problem is how a network can determine the error between its output and the desired output. The network then overcomes the mismatch between the desired and the actual outputs by adjusting the weightings of interconnections. This is called credit assignment.

The back propagation algorithm (BPA) which performs simple gradient descent solves the adjustment problem by using all of the PEs and adjusting their total interconnections. This is done by propagating the output layer error to the preceding layer via the existing connections. In fact, this operation is repeated until reaching the input layer. In other words, output error moves from each layer to the preceding layer -just opposite the direction of the movement of the original information- until reaching the input layer.

The classical form of a back propagation network consists of one input layer, one output layer, and one or two hidden layers. There is no limit on the number of hidden layers theoretically, but it has been observed that it is possible to solve very complicated problems with four hidden layers. In a back propagation network, each layer is fully connected to the next layer. In the learning phase information may come back through the network in order to update the weighting functions. The network may also be hetero associative or auto associative. The following notation is used for the purpose of explaining the back propagation algorithm.

- $x_j[s]$: The present output state of the j-th PE from the layer s.
- $w_{ji}[s]$: Weighting function of the connection between i-th PE from layer $(s-1)$ and the j-th PE from layer s.
- $I_j[s]$: The combined input of the j-th PE of layer s.

The compound input of each PE is calculated from the relation

$$I_j[s] = \sum_i (w_{ji}[s].x_i[s-1]) \tag{2.2}$$

Hence a PE in a back propagation network transfers its output as

$$x_j[s] = f\{\sum_i (w_{ji}[s].x_i[s-1])\} = f\{I_j[s]\} \tag{2.3}$$

where f may be every continuous function. The most popular functions are sigmoid, hyperbolic tangent, and sine function. This is because of their interesting properties as nonlinear functions in vanishing the output error of the network. The sigmoid function is defined as

$$f(z) = \frac{1}{1+e^{-z}} \tag{2.4}$$

Suppose that the network has an absolute error function which is differentiable with respect to all of the weighting functions. Then the critical parameter which is back propagated to the network is:

$$e_j[s] = -\frac{\partial E}{\partial I_j[s]} \tag{2.5}$$

where E is the absolute error function. It will be shown that this value may be obtained as a criterion of the relative error of the j-th PE in layer s.

By using the chain rule twice, one may introduce the relationship between the relative error of a specified PE in layer s and the local errors in layer $(s+1)$.

$$e_j[s] = f\prime\{I_j[s]\}\,.\,\sum_k\{e_k[s+1].w_{kj}[s+1]\} \tag{2.6}$$

Note that in relation 2.6 there must be a layer above the layer s, and hence it is not possible to use this relation for the output layer. If the function f is the sigmoid function defined in 2.4 then it is possible to express its derivative in terms of itself.

$$f\prime\{I_j[s]\} = f(I_j[s]).\{1 - f(I_j[s])\} \tag{2.7}$$

By using relation 2.7, one may rewrite relation 2.6 as follows:

$$e_j[s] = x_j[s].(1 - x_j[s])\,.\,\sum_k\{e_k[s+1].w_{kj}[s+1]\} \tag{2.8}$$

Note that this relation has been obtained for sigmoid function. The back propagation network is, then, described by the following statement.

First information propagates from the input layer to the output layer. Then the error between the desired output and the network output propagates from the output layer to the input layer in the return path. This procedure materializes itself according to the general relation 2.6 or in the case of sigmoid function as the relation 2.8.

The objective of the learning procedure is to minimize the absolute error function. In this section, the procedure is studied based on the concept of local error. Assume that the weighting functions of the network are given in the form of $w_{ji}[s]$. In order to decrease the absolute error function, one may change the weighting functions in the opposite direction of the gradient vector.

$$\Delta w_{ji}[s] = -lc\frac{\partial E}{\partial w_{ji}[s]} \tag{2.9}$$

where lc denotes the learning coefficient. Hence it is reasonable to say that each of the weighting functions vary according to the magnitude and opposite direction of the gradient vector on the error surface.

Partial derivatives of relation 2.9 may directly be computed based on the relative error relation. Based on the relation 2.5 and chain rule one can easily obtain:

$$\frac{\partial E}{\partial w_{ji}[s]} = \frac{\partial E}{\partial I_j[s]} \cdot \frac{\partial I_j[s]}{\partial w_{ji}[s]} = -\, e_j[s].x_i[s-1] \tag{2.10}$$

Combining relations 2.8 and 2.10 will lead to

$$\Delta w_{ji}[s] = lc.e_j[s].x_i[s-1] \tag{2.11}$$

In order to specify the above mentioned relations in the back propagation algorithm, it is necessary to define the absolute error function precisely. Assume that o is the present output of the network to the input i, and d is the corresponding desired output. Now the absolute error function may be defined as

$$E = \frac{1}{2}\sum_k (d_k - o_k)^2 \tag{2.12}$$

The index k denotes the various elements of d and o. Here, the raw error function is $(d_k - o_k)$ and from relation 2.12 it is obvious that the scaled relative error in every PE of output layer is obtained by

$$\begin{aligned} e_k(o) &= -\frac{\partial E}{\partial I_k}(o) = -(\frac{\partial E}{\partial d_k}).(\frac{\partial d_k}{\partial I_k}) \\ &= (d_k - o_k).f\prime(I_k) \end{aligned} \tag{2.13}$$

In neural networks, the scaled local error which propagates through the network is stored in the error field of each PE. As relation 2.12 shows, the absolute error function is defined on the pair (i, d). In this case with every new pair (i, d) the BPA adjusts the weighting functions to decrease the absolute error function.

Now the standard BPA may be introduced as follows:

- Propagate the input i in the forward direction through the network until reaching to the output o. During propagating this information through the network all of the combined inputs I_j and output states x_j for each PE are set.

- For each PE in the output layer calculate the scaled local error from the relation 2.13 then obtain the variations of weighting functions from relation 2.11.

- For each PE in layer s which is located below the output layer and above the input layer obtain the scaled relative error and the variation in weighting functions from relations 2.8 and 2.11 respectively.

- Update all of the weighting functions by adding the variations to the old values.

2.5.2 Performance Issues

In order to increase the performance of the back propagation algorithm, the standard algorithm has to be changed slightly according to the following concepts.

2.5.2.1 Inserting Momentum Term

Remember that the gradient method updates the weighting functions as linear functions of partial derivatives according to the relations 2.10 and 2.11. The problem is that, this method fails for the large values of learning coefficient lc. On the contrary, having a small learning coefficient leads to decrease the speed of learning procedure. In order to overcome this problem, the concept of momentum term is used. Inserting this term changes the equation 2.11 as follows:

$$\underbrace{\Delta w_{ji}[s]}_{(k+1)-\text{th step}} = lc.e_j[s].x_i[s-1] + M(\underbrace{\Delta w_{ji}[s]}_{k-\text{th step}}) \tag{2.14}$$

where M stands for the momentum. This relation is, in deed, a low pass filter which omits the oscillatory behavior and hence a reasonable learning speed might be obtained while having a small learning coefficient.

2.5.2.2 Derivatives Correction

Fahlman has introduced several methods to increase the convergence speed of the standard BPA. These methods work based on the derivatives correction and nonlinear error functions. One of the simple techniques that he has used, is to add a small positive offset to the sigmoid function's derivative before scaling the error. To explain the performance of the method, it may be said that when the weighting functions of PEs increase, the combined input increases and as the result the operating levels go to the saturation and hence the derivative and the scaled relative error vanish. In this case adding a small offset overcomes this problem.

2.5.2.3 Fast Back Propagation

In this method relation 2.11 is replaced by the relation

$$\Delta w_{ji}[s] = lc.e_j[s].(x_i[s-1] + e_i[s-1]) \tag{2.15}$$

or in simple words the error of layer $(s-1)$ will be added to the operating level before updating the corresponding weighting functions. It is also possible to insert a coefficient in this new relation.

$$\Delta w_{ji}[s] = lc.e_j[s].(x_i[s-1] + k.e_i[s-1]) \tag{2.16}$$

The value of the parameter k provides more flexibility in the implementation and getting better results. Note that for $k = 0$ the standard BPA and for $k = 1$ relation 2.15 are obtained.

This section is closed by mentioning several possible forms for absolute error functions and transfer functions. First note that the absolute error function defined in the relation 2.12 is proportional to the square of Euclidean distance between the desired output and the network output while it is possible to define other absolute error functions. For example, the choice of

$$\begin{aligned} E &= \frac{1}{3}\sum_k |d_k - o_k|^3 \\ E &= \frac{1}{4}\sum_k |d_k - o_k|^4 \end{aligned} \tag{2.17}$$

result the quadratic and cubic relative error functions as follow,

$$\begin{aligned} e_k(o) &= (d_k - o_k)^2 . f\prime(I_k) \\ e_k(o) &= (d_k - o_k)^3 . f\prime(I_k) \end{aligned} \tag{2.18}$$

It is also possible to use any other smooth function like sine and hyperbolic tangent instead of sigmoid as a transfer function. The sigmoid and hyperbolic tangent functions are very similar, i.e, sigmoid is the flat approximation of the step function on the interval $[0,1]$ and hyperbolic tangent is the same on the interval $[-1,1]$. In addition, the derivative of the hyperbolic tangent may be related to itself similar to the sigmoid function.

$$f\prime(z) = [1 + f(z)].[1 - f(z)] \tag{2.19}$$

while $f(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$. In this case relation 2.6 will be in the form

$$e_j[s] = (1 + x_j[s]).(1 - x_j[s]). \sum_k (e_k[s+1] w_{kj}[s+1]) \tag{2.20}$$

which is the same result as before.

2.6 Kohonen Networks

Unlike the previously mentioned structures, Kohonen network adjusts the links interconnections neither by using a predetermined set of states to recall, nor by attempting to reproduce correct classifications externally supplied by a tutor. The Kohonen network simply inspects the data for regularities, and organizes itself in such a way as to form an ordered description of data. The main difference in the layout of a self organizing networks is the presence of lateral connections as shown in figure 2.4. These connections mimic similar connectivities found in cerebral cortex. The network is one of the best analytically understood neural architectures. The large incoming arrow in the figure represents a pyramid of links bringing information from the input data into each node in the network, while the outgoing arrows denote the excitation value of each node in response to data.

The operation of the network is explained by considering the organization of a small network which is presented with three directions in space represented in figure 2.5 by arrows. This is the input data. Initially, the links between each node and the input data have random values which represents arbitrary directions. The following steps are repeated

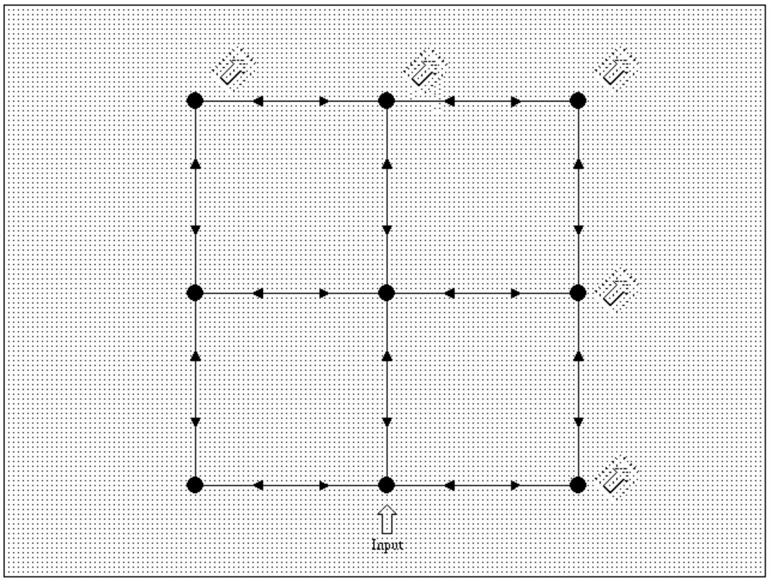

Figure 2.4: A Kohonen self-organizing network.

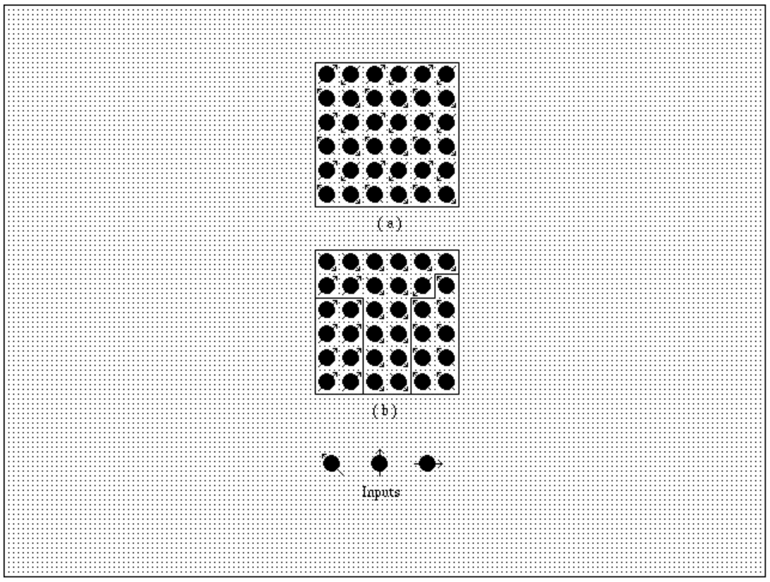

Figure 2.5: Operation of a Kohonen network: (a) random initial state, (b) organized weight distribution of a trained network with boundaries separating the directions nearest to each of the three prototype states.

until the links between PEs and the input window where the input data are presented one at a time, stop changing.

- Present one input sample consisting of the first arrow.
- Select the node whose direction is closest to this arrow.
- Adjust the link weights for this node and its nearest neighbors by a small amount towards the direction of the input sample.
- Take the next input sample which is represented by another arrow and return to the first step.

As training proceeds, the size of the neighborhood is reduced. The trained network would have each node set to the directions shown at the bottom of figure 2.5. The network displays a topological organization of changes between the directions of neighboring nodes as one travels across the network.

The competition between different nodes which selects the neighborhood where the links change for each input sample is thought to be realized in biological networks by inhibitory connections laterally between adjacent nodes in the network. The Kohonen network is claimed to be a very powerful network in quantizing geometrically complex surfaces in any number of spatial dimensions.

2.7 Temporal Differences Networks

This network can be used efficiently in modeling time dependent systems. The main difference between this network and perceptron network is that it is able to learn goal oriented tasks where the eventual outcome of each task is not known until several steps later. As a very good example for the application of this method, the problem of controlling an inverted pendulum located on a moving entity may be mentioned. A very simple network is used to model this problem consisting of two PEs as shown in figure 2.6. The first PE is the engine labeled as the controller in figure 2.6 which decides the direction to apply to the moving object, and the second is the supervisory PE which attempts to predict the likelihood that the current sequence of controller action will gradually lead to failure by dropping the pole. This is called reward/penalty reinforcement training method as it works based on the following facts:

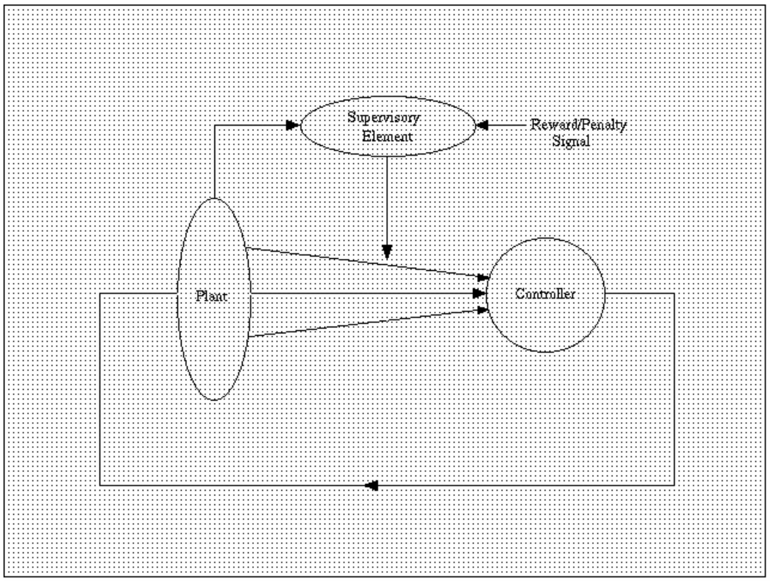

Figure 2.6: Temporal difference network used to model the moving inverted pendulum problem.

- The controller uses the penalty or failure signals to eventually avoid choosing pathways in the input space which lead to have wrong control actions.
- The supervisory PE updates its forecasts of the likelihood of the failure by comparing each prediction of the eventual outcome of the control process with the previous one.
- The difference obtained from the above comparison is used to update the weighting functions between the supervisory PE and the plant states.
- The term reinforced learning stands to describe that the weighting functions increase if the succeeding forecasts agree and reduce otherwise.

Despite having difficulties in proving the convergence of the learning mechanism, the temporal differences learning method has been shown to extend the back propagation method in the case of time varying systems.

For the case of moving inverted pendulum, the two PE system learns to balance the pole by moving the object rapidly in the opposite direction at a right rate. The network is first trained to balance the pole without reference to where the moving object is, and then a penalty signal for the position of the moving object is introduced.

2.8 Applications

In this section a number of applications for neural networks are introduced. Pattern recognition, nonlinear systems control, and system modeling are among the most important applications of neural networks. In this section each of these application are briefly studied.

For the case of system modeling, I introduce the research I have conducted prior to accomplishing the current dissertation which led to extract the idea of using perceptron neural networks as a tool of bursty traffic modeling.

2.8.1 Pattern Recognition

Pattern recognition and classification are among the most popular applications of neural networks. Pattern classifiers accept an input feature vector, and produce an output class code. The feature vector corresponds to a point in feature space and the class code is the label assigned to the region in feature space in which the point lies. Figure 2.7 shows a simple perceptron network with one hidden layer used to solve exclusive-OR classification problem. It is important to note that conventional template matching classifiers are not

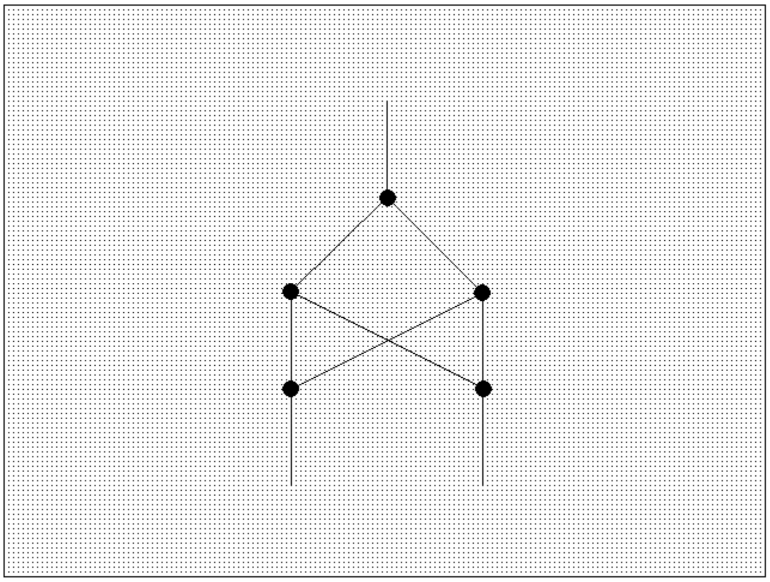

Figure 2.7: A simple perceptron network used for solving exclusive-OR classification problem.

able to solve this problem.

2.8.2 Nonlinear Systems Adaptive Control

Nonlinear adaptive control is the second application of neural networks which is discussed in this section. Nonlinear control problems usually arise in areas such as hydraulics, robotics, vehicle control, and traffic control. Neural networks can be used to substitute and to extend the standard linear adaptive control schemes. Figure 2.8 shows a typical nonlinear control system based on a feedforward network that can be used to correct the system output.

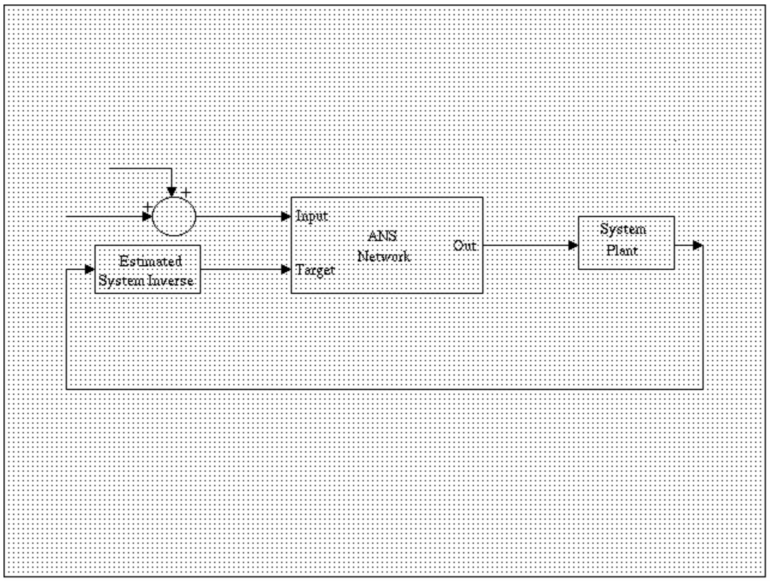

Figure 2.8: A non-linear adaptive control system using a feedforward perceptron neural network controller.

2.8.3 System Modeling

System modeling is one of the most important applications of neural networks without any doubt. This is usually done by attempting to predict the dynamical behavior of a complicated system after learning the dynamics of the system. The network usually obtains the information required for the learning procedure from a number of available samples. This is an alternate way of solving the describing differential or difference equations of the systems.

In the following section, I provide a detailed discussion about the research I have conducted to use perceptron neural networks for modeling one- and two-dimensional chaotic time series.

2.8.3.1 Neural Network Modeling of Discrete-Time Nonlinear Arrays

One-Dimensional Discrete Time Maps The first group consists of a broad class of one-dimensional discrete time maps whose next state is a third order polynomial of the present state. The members of this group are described by the relation:

$$x_{n+1} = a + bx_n + cx_n^2 + dx_n^3 \tag{2.21}$$

It is obvious that by choosing the parameters a, b, c, and d a large number of famous one-dimensional discrete maps can be generated, e.g, with $a = 0$, $b = 4$, $c = -4$, and $d = 0$ the logistic map with the equation

$$x_{n+1} = 4.x_n.(1 - x_n) \tag{2.22}$$

is obtained or with $a = V$, $b = 0$, $c = -\frac{1}{2}$, and $d = 0$ the relation 2.21 will lead to have the parabolic map equation which is in the form:

$$x_{n+1} = V - \frac{1}{2}x_n^2 \tag{2.23}$$

Now the modeling procedure is explained. For the task of modeling a fixed structure perceptron network with two hidden layers is chosen. This network has four PEs in its input layer, ten PEs in each of its two hidden layers, and one PE in its output layer. The nonlinear transfer function of each PE is the sigmoid function. As it has been seen previously, the interesting property of this function is that its derivative can be described in terms of itself as:

$$f\prime(z) = f(z).[1 - f(z)] \tag{2.24}$$

where $f(z) = [1 + e^{-z}]^{-1}$, this is a very useful characteristic that can be used in the learning procedure of the network and may reduce the cost of computation significantly. Figure 2.9 shows the structure of the network. As it can be seen in the figure, this network is a fully connected feed forward perceptron network. In this network, the learning phase is attached to the recalling phase. The learning phase is, in deed, the adjusting procedure of weighting functions in response to an applied input in order to have a suitable output in the output layer. The learning procedure is supervised as the network has access to a number of samples in its learning phase. In addition, because the desired output is different from all of the inputs in each step, the trained network is hetero associative. Four input samples are four consecutive samples of the discrete time mapping, and the output

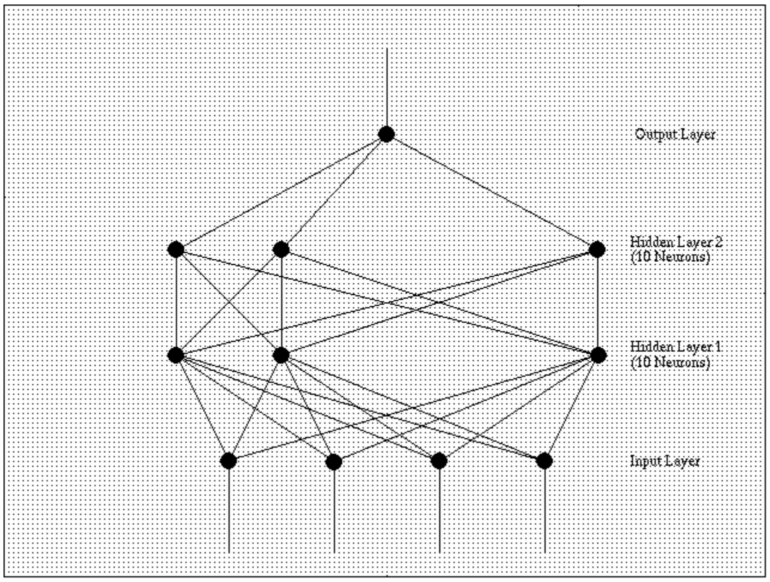

Figure 2.9: Fixed structure neural network used for modeling One-dimensional discrete time arrays.

which the network must be able to predict is the fifth sample of the mapping. The required number of samples for training of the network based on the steady state behavior of the mapping differs from thirty thousands for simpler mappings to one hundred thousands for the more complicated mappings, i.e, mappings with chaotic behavior.

In each iteration, a sample set consisting of five consecutive samples $(x_1, x_2, x_3, x_4, x_5)$ is provided for the network among which the first four are used as the input array and the fifth one, x_5, as the output. In the next iteration, the first sample is discarded and the second sample through the fifth sample (x_2, x_3, x_4, x_5) are considered as the input array and the sixth sample x_6 is considered as the output sample and so on and so forth. The network operation is based on the comparison of its output and the desired output. Based on this comparison, an error signal is generated and by back propagating this signal through the network, the weighting functions are adjusted. In the recalling procedure, the error is approximately zero, i.e, a bound 10^{-4} is used with which the error must not exceed from that bound. The network is a feedforward network with no feedback connection. As the learning procedure is based on the correction of the error, the network is able to have a suitable output when the input is deficient or noisy.

The absolute error function is built on the basis of the second momentum output error, i.e.,

$$E = \frac{1}{2}\left(y_d - y\right)^2 \tag{2.25}$$

By using relation 2.13 and substituting $f\prime(I_k)$ by $y(1-y)$ from the sigmoid function, the relative error e_o is:

$$\begin{aligned} e_o &= -\frac{\partial E}{\partial I_o} = -\frac{\partial E}{\partial y}\frac{\partial y}{\partial I_o} = (y_d - y).f\prime(I_o) \\ &= (y_d - y).y.(1-y) \end{aligned} \tag{2.26}$$

where y is the network output, y_d is the desired output, I_o is the compound input of the output PE, and f is the sigmoid function. It is interesting to know that multiplying the error by the derivative of the transfer function, scales it. In the learning procedure the gradient method is considered. For implementing the back propagation algorithm, the algorithm discussed in the previous section is used and for having a better performance, the improved version of relation 2.11 as in relation 2.16 is used. Note again that the objective of the learning phase is to minimize the absolute error function.

Two-Dimensional Discrete Time Maps The second group consists of a broad class of two-dimensional discrete time polynomials:

$$\begin{aligned} x_{n+1} &= f_{11}(x_n) + f_{12}(y_n) \\ y_{n+1} &= f_{21}(x_n) + f_{22}(y_n) \end{aligned} \tag{2.27}$$

where the f_{ij} functions are second order polynomials of the corresponding variables, i.e, $f_{ij}(x) = a_{ij} + b_{ij}x + c_{ij}x^2$. Again, this is a broad class of two-dimensional maps including those which exhibit chaotic behavior. As an example, choosing parameters a_{ij}, b_{ij}, and c_{ij} as follow,

$$\begin{aligned} a_{11} &= \frac{1}{2}, \quad b_{11} = 0, \quad c_{11} = -A \\ a_{12} &= \frac{1}{2}, \quad b_{12} = 1, \quad c_{12} = 0 \\ a_{21} &= 0, \quad b_{21} = B, \quad c_{21} = 0 \\ a_{22} &= 0, \quad b_{22} = 0, \quad c_{22} = 0 \end{aligned}$$

leads to have Henon map with the equation

$$\begin{aligned} x_{n+1} &= 1 + y_n - A\,x_n^2 \\ y_{n+1} &= B\,x_n \end{aligned}$$

Generally, with the appropriate choose of parameters, it is possible to have various kinds of steady state behaviors. Again, a fixed structure neural network same as before is used except that the present neural network has two PEs in its output layer corresponding to the x_{n+1} and y_{n+1} outputs. Four inputs are two consecutive samples of the signals x_n and y_n. Figure 2.10 shows the structure of the network. The mechanism is exactly the same

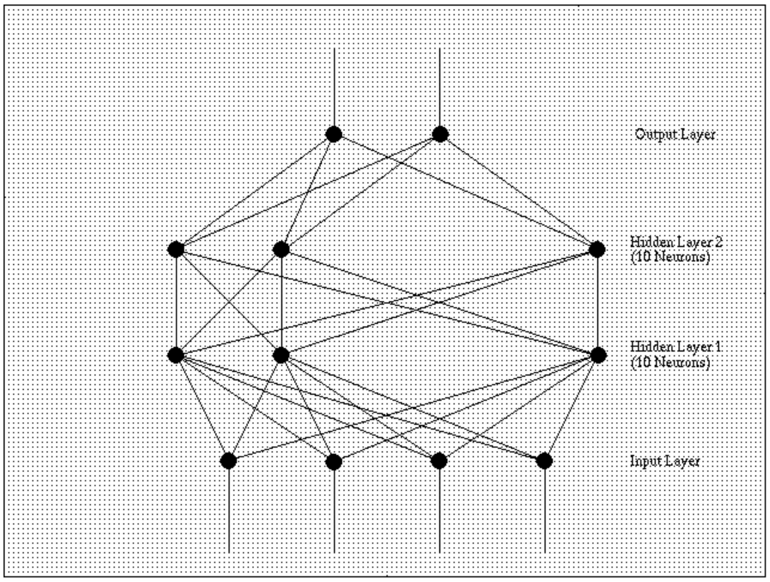

Figure 2.10: Fixed structure neural network used for modeling two-dimensional discrete time arrays.

as before. The only difference here is the absolute error function which is defined as

$$E = \frac{1}{2}\left[(y_d - y)^2 + (x_d - x)^2\right] \tag{2.28}$$

where x_d, y_d are desire outputs and x, y are network outputs and the relative output errors are:

$$\begin{aligned} e_{ox} &= (x_d - x).x.(1 - x) \\ e_{oy} &= (y_d - y).y.(1 - y) \end{aligned} \tag{2.29}$$

All of the other steps are the same as the steps implemented in the case of first group. An interesting point is observed here and it can be explained by the following statement. The learning algorithm converges with a higher speed, i.e, the number of required examples for network to be trained are less in the latter case despite having a richer dynamic compared to the former case. That is because of the correlation between x and y samples.

Experimental Results In this section, first the simulation results of the modeling task are discussed, then some of the practical problems in the implementation of the algorithm are briefly mentioned [3].

First, the performance of the modeling tool is evaluated in case of logistic map. As it is observed in figure 2.11, for the initial condition $x_0 = 0.75$ which yields to a stable fixed point, the network is able to follow the behavior of the map very rapidly. Figure 2.12

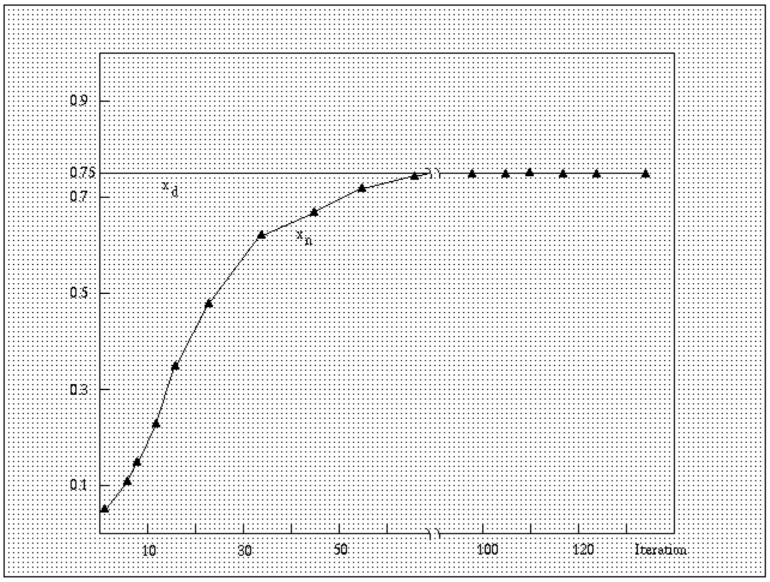

Figure 2.11: Logistic map modeling result for initial condition = 0.75

shows the result for the initial condition $x_0 = 0.1$ as an example of modeling chaotic solutions. The fixed structure neural network follows the chaotic map with an absolute error less than 0.001 discontinously but it takes almost 94'000 iterations for it to be able to predict the next 40 samples of the map within the specified error bound. After that, the

[3] For all of the graphs appearing in this section, the initial weighting functions were chosen to be 0.1 between every single pair of PEs connected together. Only for the graph appearing in figure 2.14 a combination of values 0.1, 0.3, and 0.7 was chosen in order to make a significant initial difference.

neural network loses the track because of the reason explained earlier, i.e., the net becomes chaotic itself and since it is not possible to have a zero value for the error, the error is built up as time proceeds and finally exceeds the acceptable error bound. At this stage the learning process needs to be repeated in order to force the network to track the chaotic array within the acceptable error limits. Qualitatively, the same pattern is repeated in

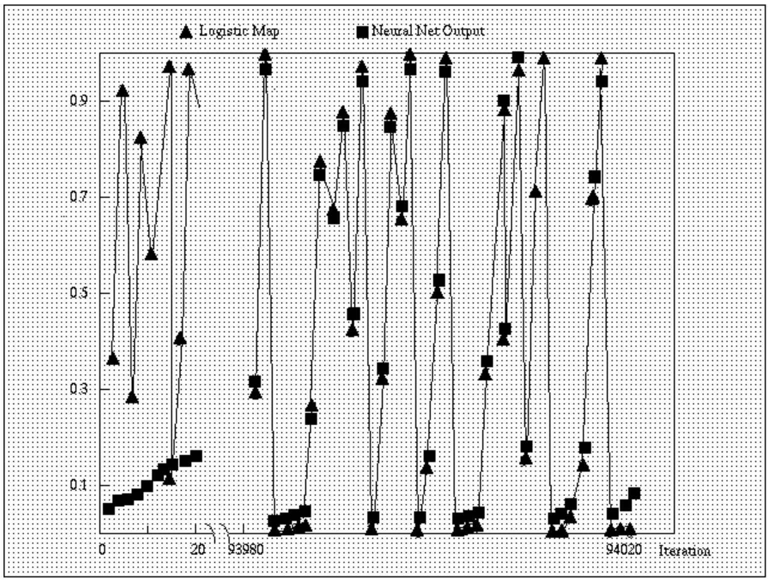

Figure 2.12: Logistic map modeling result for initial condition = 0.1

case of $x_0 = 0.3$ and a number of other initial conditions. Finally, it is very important to note that the choice of the initial conditions both in case of chaotic map and the weighting functions of the neural network greatly affects, the speed of convergence specially in case of chaotic solutions.

Figure 2.13 shows the result of the modeling task for the Henon map with the parameters $(A, B) = (4, 1)$ and the initial condition $(x_0, y_0) = (0.5, 0.5)$. It shows that the network is able to reach to the fixed point very rapidly. Figure 2.14 shows the result of the modeling task for the Henon map with the parameters $(A, B) = (0.8910, 0.0017)$ and the initial condition $(x_0, y_0) = (0.67, 0.21)$. This is an example of modeling periodic solutions and again, the figure shows that the network is able to learn the periodic behavior of the map very rapidly.

It is important to remind that although the dynamical behavior of the map is very much more complicated compare to logistic map, it is relatively difficult in practice to choose

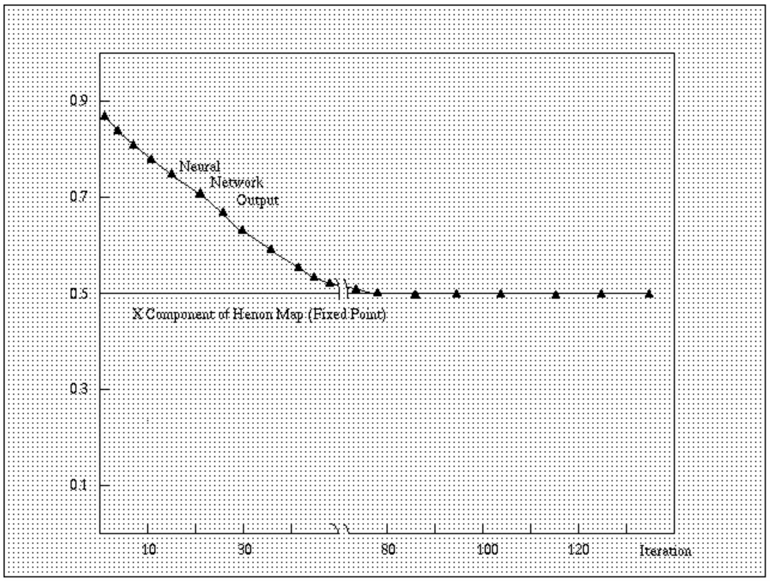

Figure 2.13: Henon map x state component modeling result for initial condition $(x_0, y_0) = (0.5, 0.5)$ and parameters $(A, B) = (4, 1)$.

the parameters such that the map exhibits chaotic behavior. This is mainly because of the round off errors that shift the map out of the narrow chaotic regions. Now, its time to mention a number of problems encountered in the implementation task.

- Time Consuming Learning Algorithm

  The learning algorithm is time consuming, in deed, several hundred thousand samples are required in order to train the network. In addition, there is a specific problem in the case of chaotic systems, which I have focused on. To explain the problem, it can be simply said that although it is possible to reach to a very small network error at some steps during the learning phase, if the network error is studied for further samples, it is observed that this error begins to grow as time proceeds. The reason is summarized in the chaoticness of the system, i.e, since the nonlinear network wants to model a chaotic array, it becomes chaotic itself. In this situation, the small error may be considered as a small difference between two close initial conditions for the desired output and the network output. As a characteristic of a chaotic system, the error will begin to grow very soon and this is nothing but high sensitivity to the variations of initial conditions. As a matter of fact, this can be interpreted as a good sign for a network which has tried to model a chaotic map and it has become chaotic itself. One way of relieving the effect of having an error component which grows

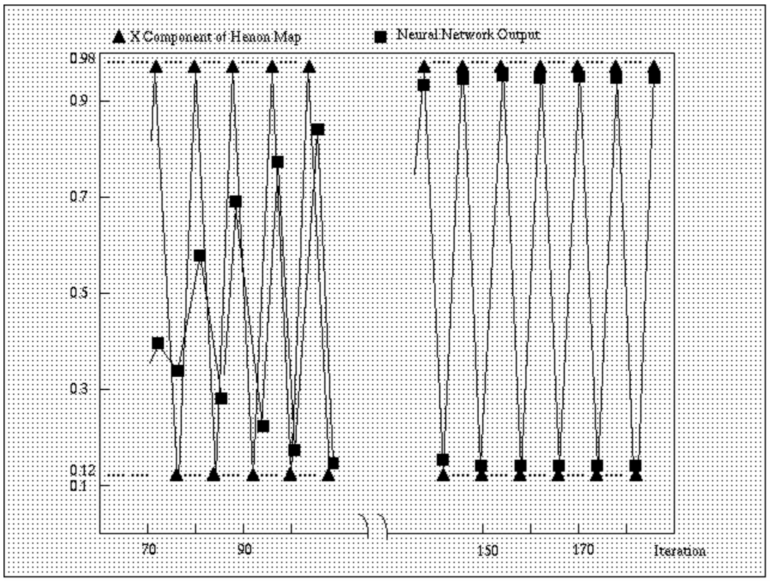

Figure 2.14: Henon map x state component modeling result for initial condition $(x_0, y_0) = (0.67, 0.21)$ and parameters $(A, B) = (0.8910, 0.0017)$.

with time is to repeat the learning phase followed by the recalling phase periodically, otherwise the results are not reliable any more. In practice after the first learning phase with a couple of hundred examples, the neural network would be able to predict less than one hundred samples and then the learning phase has to be repeated with the same number of examples as in the previous recalling phase and so on to have reliable results.

- Large Size of Output Vectors

In order to increase the performance of numerical methods, it is better not to store all of the samples of the desired and the network outputs in a vector pair. Hence it is more reasonable to keep only one out of each of several consecutive samples. In this case, caution must be taken in order to avoid having a hidden problem, if the omitted samples are placed in a region which the signal has very fast variations, the signal information may be lost. Specially, the occurrence of this situation is more probable in the case of chaotic signals, because the chaotic behavior can be interpreted as a kind of random behavior itself. In these cases all or most of the samples must be kept in order to predict the time series properly. In practice, I have not had access to any special rule dealing with the location of these regions. Hence, that has left the try and error method as the only tool which has been observed to be extremely time consuming.

- Effects of Learning Coefficient and Momentum Term

  It was mentioned that the variation of weighting functions are derived by the relation

$$\Delta w_{ji[s]} = lc.e_j[s].x_i[s-1]$$

  The network learning speed which is, in fact, the speed of weighting functions adjustment procedure is a function of the value of learning coefficient lc. Although bigger values of the learning coefficient may lead to have faster learning procedure, it is not possible to have very big learning coefficients. This is the direct effect of applying the gradient method, i.e, weighting functions are changed as a linear function of the error surface of the partial derivatives in the gradient method . The approximation is valid as long as the error surface is relatively linear while this assumption is not valid for large values of lc. On the contrary, having small values for lc leads to have a very low learning speed. In order to overcome this problem I have used the improved version of the above relation as in 2.16. The suitable value of lc is different from case to case but for not having the effects of nonlinear error surface the lc must be less than one in most of the cases. Moreover the second momentum of Δw_{ji} acts more suitably in most of the cases.

- Low Convergence Speed Near the Optimum Point

  The main problem of the gradient method is due to its low convergence speed near the optimum point, i.e, although for a big initial difference moving toward the direction of the gradient vector decreases the difference rapidly and forces to have a value close to the optimum point, the method loses its performance around the optimum point and the speed of convergence reduces significantly. For omitting this deficiency, a multiple of layer $(s-1)$ error can be added to the weighting functions adjustment algorithm as in relation 2.14. The final result has been in the form of

$$\underbrace{\Delta w_{ji}[s]}_{(k+1)-\text{th step}} \quad = \quad lc.e_j[s].\{x_i[s-1] + k.e_i[s-1]\}$$

$$+ \quad M(\underbrace{\Delta w_{ji}[s]}_{k-\text{th step}}) \tag{2.30}$$

where M stands for the momentum. The k coefficient is, in deed, the term that the weighting functions adjustment algorithm has shown the most sensitivity to its variation. The suitable value of this parameter is again different from case to case but in the most cases a value on the interval $[0, 1]$ has led to a suitable result.

- Effects of Initial Condition

All of the convergence results are affected strongly by the initial conditions. It can be said that the initial values of the parameters play a crucial role in the convergence of the algorithm. It is even possible to have a divergent algorithm, if the initial values of the parameters are chosen unsuitably. As a practical result, it is better to set the initial values of the parameters on $w_{ji}(0) = 0.01 \quad \forall i, j$.

- Floating Point Overflow Problems

As the last important factor, the overflow problem is discussed. In every broad numerical computation, one must always consider the probability of having very large or very small numerical values. Storing such kinds of values may lead to produce invalid results because of round off operation of the computer. In this special case, the huge variation of compound inputs of PEs can create many problems in their storing procedure. Based on the asymptotic behavior of the transfer function which is sigmoid in this case, this problem may be avoided. Observe that the sigmoid function reaches its asymptotic value with a very good approximation when the absolute value of its argument is less than or equal 11, i.e,

$$\frac{1}{1+e^{-z}} \cong \begin{cases} 0 & : \quad z \leq -11 \\ 1 & : \quad z \geq 11 \end{cases} \tag{2.31}$$

In other words, there exist some limits for the compound input and for all of the values beyond these limits the transfer function almost acts the same. Hence, an upper limit of $z = 11$ and a lower limit of $z = -11$ are considered for the compound input to avoid having the overflow problem in the numerical algorithm.

Chapter 3

Analysis of Self-Similar Packet Traffic

3.1 Introduction

Teletraffic analysis as an application of queuing theory to the performance analysis of computer communication networks, has definitely been one of the most important applications of mathematical modeling. In fact, the rapid rate of development of telecommunication theory has been one of the most effective inductions for the development of queuing theory. Recently, the widespread deployment of packet switching, specifically, services from ISDN through ATM and beyond, has generated a set of challenging problems in queuing theory. The problem of bursty traffic packet arrival modeling is considered one of the most important problems in this category without any doubt. Given that performance models are only reliable when their underlying assumptions are satisfied, the problem of obtaining an accurate model of packet traffic is particularly important in performance evaluation of all packet based networks. Although numerous models of packet arrival processes have been proposed during the last few years [1], there is still a lack of complete understanding of the features in packet traffic that impact performance. This is partly due to uncertainties in the traffic characteristics of the emerging networks and services, and to the difficulties in characterizing the traffic arrival models and resource usage patterns in the emerging networks.

Analysis of traffic data from networks and services such as ISDN traffic, Ethernet LAN's, Common Channel Signaling Network (CCNS) and Variable Bit Rate (VBR) video have all convincingly demonstrated the presence of features such as self-similarity, long range dependence, slowly decaying variances, heavy-tailed distributions and fractal dimensions.

[1] See Ramaswami et al [6], Hellstern et al [10], Sriram et al [13], Heffes et al [14], Bolotin.[16] for more details.

These features, in deed, are more characteristics of fractal processes than those of conventional stochastic processes. Conventional traffic processes from regular telephone traffic or the Poisson and Poisson-based models seem to be Markovian in nature characterized by exponential decays. The type of packet traffic with the above mentioned characteristics is interpreted to be bursty in nature. To be more specific, Leland and Wilson from Bellcore research center have presented a preliminary statistical analysis of Ethernet traffic, on the presence of "burstiness" across a wide range of time scales [2]: traffic spikes ride on the longer term ripples, that in turn ride on longer term swells, so on and so forth. This is also explained in terms of self-similarity, i.e., self-similar phenomena show structural similarities across all or at least a very wide range of time scales. The degree of self-similarity measured via the Hurst parameter typically depends on the utilization level of the transmission medium and can be used to measure burstiness of the traffic.

As another important difference between the aggregated bursty traffic and the so called Poisson-like conventional models, it could be mentioned that the aggregated traffic is expected to become less bursty or smoother as the number of traffic sources increases based on the conventional models, but it has very little to do with the reality. In fact, it has been observed that the burstiness of LAN traffic intensifies as the number of traffic sources increases as opposed to commonly held views.

Conventional characterizations suppose that packet traffic consists of alternating active and silent periods with well defined statistics. On the contrary, measurement studies have noted that there is no actual burst length, and bursts occur over many time scales. At every step, examination of the data shows that the bursts resolve into bursts over smaller time scales. This burst within burst structure captures the fractal properties observed in actual traffic data.

In this chapter the following concepts are studied: First, the concept of self-similarity is described and a statistical analysis is given. Then, the application of fractals in engineering of packet traffic is investigated. The most important goal of this chapter is to provide a solid background for the modeling task of ON-OFF traffic sources, and the aggreagted traffic patterns with self similar nature generated by them.

3.2 Fractal Properties of Packet Traffic: A Statistical Analysis

The main objective of the current section is to establish a solid foundation for a statistically well-defined property of time series called self-similarity. Intuitively, self-similar phenomena display structural similarities across so many time scales. The degree of self-similarity is usually specified by measuring a single parameter called Hurst parameter.
The following sub-section discusses about the mathematical and statistical properties of the self-similar processes.

3.2.1 Second-Order Self-Similarity

Let

$$X = (X_t : t = 0, 1, 2, ...) \tag{3.1}$$

be a covariance stationary stochastic process with mean μ, variance σ^2, and autocorrelation function $r(k), \quad k \geq 0$. In particular suppose X has an autocorrelation function of the form

$$r(k) \sim a_1 k^{-\beta}, \quad \text{as} \qquad k \to \infty \tag{3.2}$$

where $0 < \beta < 1$ and constants $a_1, a_2, ...$ denote finite positive integers. For each $m = 1, 2, 3, ...$ let

$$X^{(m)} = (X_k^{(m)} : k = 1, 2, 3, ...) \tag{3.3}$$

denote the new covariance stationary time series with corresponding autocorrelation function $r^{(m)}$ obtained by averaging the original series X over non-overlapping blocks of size m, i.e., for each $m = 1, 2, 3, ...,$ $X^{(m)}$ is given by

$$X_k^{(m)} = 1/m(X_{km-m+1} + ... + X_{km}), \qquad k \geq 1 \tag{3.4}$$

The process X is called exactly second-order self-similar with self-similarity parameter $H = 1 - \beta/2$ if the corresponding $X^{(m)}$ has the same correlation structure as X, i.e., $r^{(m)}(k) = r(k)$ for all $m = 1, 2, 3, ...,$ and $k = 1, 2, 3, ...$ X is called asymptotically second-order self-similar with self-similarity parameter $H = 1 - \beta/2$ if $r^{(m)}(k)$ agrees asymptotically with $r(k)$ given by 3.1, for large m and k. In other words, X is exactly or asymptotically second-order self-similar if the aggregated processes $X^{(m)}$ are the same as X or become indistinguishable from X with respect to their correlation functions. Fractal Gaussian Noise (FGN) is a good example of an exactly self-similar process with self-similarity parameter $H, \quad 1/2 < H < 1$. Fractional Arima processes with the parameters

(p, d, q) such that $0 < d < 1/2$ are examples of asymptotically second-order self-similar processes with self-similarity parameter $d + 1/2$.

Mathematically, self-similarity manifests itself in a number of equivalent ways as follows. (1) The variance of sample mean decreases more slowly than the reciprocal of the sample size. This is called slowly decaying variance property which means

$$var(X^{(m)}) \sim a_2 m^{(-\beta)}, \qquad m \to \infty, \quad 0 < \beta < 1 \tag{3.5}$$

(2) The autocorrelations decay hyperbolically rather than exponentially fast, implying a non-summable autocorrelation function $\sum_k r(k) = \infty$. This is called long range dependence property which means $r(k)$ satisfies relation 3.1.
(3) The spectral density $f(.)$ obeys a power-law near the origin. This is the concept of $1/f$ noise with the meaning

$$f(\lambda) = k\lambda^{-\gamma} \tag{3.6}$$

as $\lambda \to \infty$ with $0 < \gamma < 1$ and $\gamma = 1 - \beta$.

It seems that the most striking feature of self-similar processes is that their aggregated process $X^{(m)}$ possess a non-degenerate correlation function as $m \to \infty$. This is in stark contrast to typical packet traffic models currently considered in literature, all of which have the property that their aggregated processes $X^{(m)}$ tend to second order pure noise, i.e., $r^{(m)} \to 0$ as $m \to \infty$. As an equivalent method of description, they may be characterized by the following properties.
(1) The sample mean variance decreases like the reciprocal of the sample mean.
(2) The autocorrelation function decrease exponentially fast, implying a summable autocorrelation function. This, in fact, is equivalent to the short-range dependence property.
(3) The spectral density is bounded at the origin.

The concept of self-similar processes provides a very elegant explanation of an empirical law commonly referred to as the Hurst effect. In order to describe the Hurst effect, it should be mentioned that for a given set of observations $(X_k : k = 1, 2, ..., n)$ with sample mean $\overline{X}(n)$ and sample variance $S^2(n)$, the rescaled adjusted range or the R/S statistic is given by

$$\frac{R(n)}{S(n)} = \frac{1}{S(n)}[max(0, W_1, W_2, ..., W_n) - min(0, W_1, W_2, ..., W_n)] \tag{3.7}$$

with

$$W_k = (X_1 + ... + X_k) - k\overline{X}(n), \qquad k \geq 1 \tag{3.8}$$

While many naturally occurring time series appear to be well represented by the relation $E[R(n)/S(n)] \sim k_1 n^H$, as $n \to \infty$, with Hurst parameter H typically about 0.73, observations X_k from a short-range dependent model are known to satisfy $E[R(n)/S(n)] \sim k_2 n^{0.5}$, as $n \to \infty$. This discrepancy is usually referred to as the Hurst effect.

3.2.2 Degree of Self-Similarity: Estimation Methods

In this part, methods of estimating self-similarity degree are introduced based on the properties of covariance stationary second-order self-similar processes, namely slowly decaying variances, long-range dependence, and a spectral density obeying a power-law. Hence the problem may be approached in three ways:
(1) Time-domain analysis based on the R/S statistic.
(2) Analysis of variances of the aggregated processes.
(3) Periodogram-based analysis in the frequency domain.

The objective of the first method is to estimate the Hurst parameter H via the Hurst effect. Briefly, the approach consists of plotting the $\log(R(n)/S(n))$ versus $\log(n)$ in the logarithmic scale which results in a plot called "rescaled adjusted range plot" or the "pox diagram of R/S". For a well-defined parameter H, a typical rescaled adjust range plot starts with a transient zone showing the nature of short-range dependence and continues with a steady state part which is a straight line with a certain slope. There are also some fluctuations around that line. In fact, if such asymptotic behavior appears, then graphical R/S analysis may be used to estimate the self-similarity degree. An estimate $\hat{H}$ of self-similarity parameter H is given by the line's asymptotic slope which can take any value between 1/2 and 1. The most useful feature of the R/S analysis is its relative robustness against changes of marginal distribution.

In the second method, the variances of the aggregated second-order self-similar processes

$X^{(m)}, m \geq 1$, decrease linearly in log-log plots against m with slopes arbitrarily flatter than m. This behavior is, in fact, seen for the large values of m as the representative of time. The so-called variance-time plots are obtained by plotting the $\log(var(X^{(m)}))$ against $\log(m)$ and by fitting a simple least squares line through the resulting points in the plane. Values of the estimate $\hat{\beta}$ of the asymptotic slope between -1 and 0 suggest self-similarity with a degree of $\hat{H} = 1 - \hat{\beta}/2$.

In contrast to the previous two methods, the third method takes advantage of the presence of limit law for a more refined data analysis like the existence of confidence levels for H. This is simply done by using maximum likelihood types estimates (MLE) based on the periodogram-based analysis in the frequency domain. As an example, Whittle's approximate MLE may be mentioned to be used for the approximately Gaussian processes. A combination of MLE-type approach and the above one of the mentioned aggregation methods lead to an operational procedure for obtaining confidence intervals for the self-similarity parameter H. Plots of the point estimates $\hat{H}^{(m)}$ of $H^{(m)}$ versus m with their specified confidence level will typically vary a lot for small aggregation levels but will stabilize after a while and fluctuate around a constant value, the final estimate of self-similarity parameter H. [2]

3.2.3 Mathematical Explanation of Self-Similarity: A Simple Aggregation Argument

Mathematically, self-similarity in measurements from aggregated traffic of Ethernet, ISDN, CCSN, and VBR traffic can be explained by a simple aggregation argument: aggregating many elementary renewal reward processes representing individual user traffic produces self-similarity in limit as the number of users increases. First, let me define the concept of infinite variance syndrome. A random variable is said to exhibit an infinite variance syndrome or is called heavy-tailed if

$$P[U \geq u] \sim u^{-\alpha} L(u) \tag{3.9}$$

where $L(u)$ is a slowly varying function at infinity and $0 < \alpha < 2$. The crucial property that distinguishes the renewal reward process source model from the commonly assumed source model is that the inter-renewal arrivals, i.e., the lengths of the active/inactive periods are

[2]For a complete discussion, see reference [2].

heavy tailed or in terms of Mandelbort terminology, exhibit the infinite variance syndrome. A number of evidences support the existence of infinite variance syndrome in packet traffic measurements. Hellstern et al [10] have observed that the extreme variability of ISDN data cannot be adequately captured using traditional packet traffic models but instead to be best described by the concept of heavy-tailed distributions. Duffy et al [15] have observed the same evidence in the CCSN traffic studies. They have noticed that the call holding time distribution for calls originating during high traffic periods is heavy-tailed with an estimated value of about 2.0, and for calls originating during light traffic periods, the estimated value drops down to about 1.0. As it was studied in chapter one, fractal dimensions may be used to characterize the fractal-like nature of the traffic measurements. This idea was first proposed by Erramilli et al [11]. Intuitively, a dimension is an indication of the extent to which a set, e.g., arrival times, fills the space in which it is embedded. As an example, the so-called correlation dimension associated with a measure known as correlation integral -defined in chapter 1- is an appropriate tool to characterize the behavior of self-similar sets.

3.2.4 An Evidence of Self-Similarity: Ethernet and VBR Video Traffic Measurements

In [4], Leland et al from Bellcore research center have used high quality, high time-resolution LAN traffic data collected between August 1989 and February 1992 on several Ethernet LAN's and have observed the absence of natural length of a burst. This behavior is very different from conventional telephone traffic and from currently considered formal models of packet traffic. The same feature has also been observed in recent studies of VBR video traffic by J.Beran et al in [7]. With the available data sets, Leland et al investigated the persistent feature of Ethernet traffic across the network and across the time irrespective of the medium utilization level. The following figure explains the internal nature of Ethernet traffic using the graphical tools explained in the previous section. It depicts the pox plot of R/S and the variance time-curve corresponding to the time series that describes the number of bytes sent over every 10 milliseconds of the measurement period. The pox plot shows an asymptotic slope estimated to be about 0.79. The normalized variance-time curve shows an asymptotic slope estimated to be about -0.40, resulting in a practically identical estimate of Hurst parameter H of about 0.80. In general, the degree of self-similarity depends on the utilization level of the medium. For the Ethernet it increases as the utilization increases. The result of MLE-based estimation method combined with the aggregation method is shown in figure 3.1. Under the same circumstances, Leland

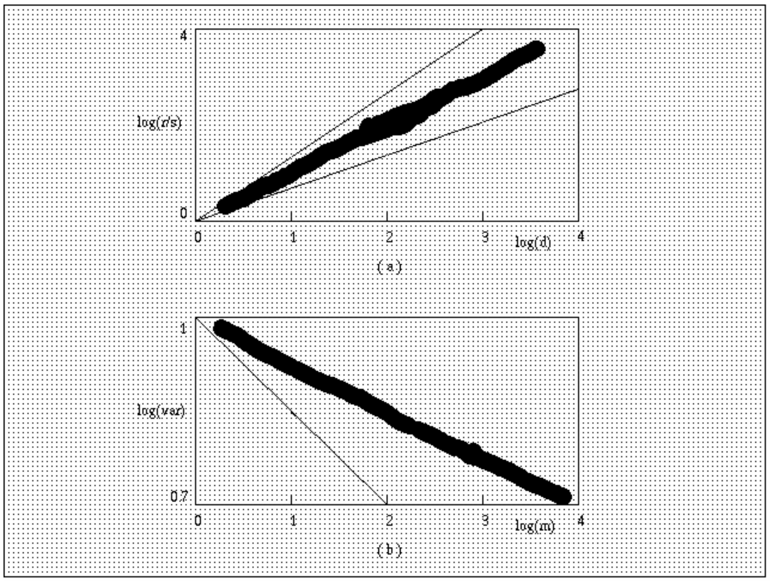

Figure 3.1: Graphical methods (a) typical pox plot of R/S (b) typical variance-time plot, for checking self-similarity of the time series representing the number of bytes per time unit during the medium activity period.

et al considered the aggregated time series $X^{(m)}$ with $m = 100, 200, 300, ..., 1900, 2000$ representing the packet counts per $1, 2, 3, ..., 19, 20$ seconds, respectively. The Hurst parameter estimates $\hat{H}^{(m)}$ of $H^{(m)}$ and the corresponding 95% confidence interval bands seem to be stable and fluctuate slightly around 0.9 throughout the considered aggregation levels.

The combined MLE/aggregation method has also been applied to other available data sets in order to indicate the accuracy of the graphical estimators compared to statistically more rigorous Whittle's estimator.

3.3 Engineering of Realistic Packet Traffic Processes: Applications of Fractals

In the present section, the application of fractals is investigated. Fractal characterization is, in fact, applicable to many aspects of teletraffic systems such as arrival, service time, buffering, quality of service, and queuing. Although, theoretically classical Markovian models can always be used to describe any finite set of traffic measurements, the resulting systems are very complex and highly parameterized in case of fractal processes. Hence, its

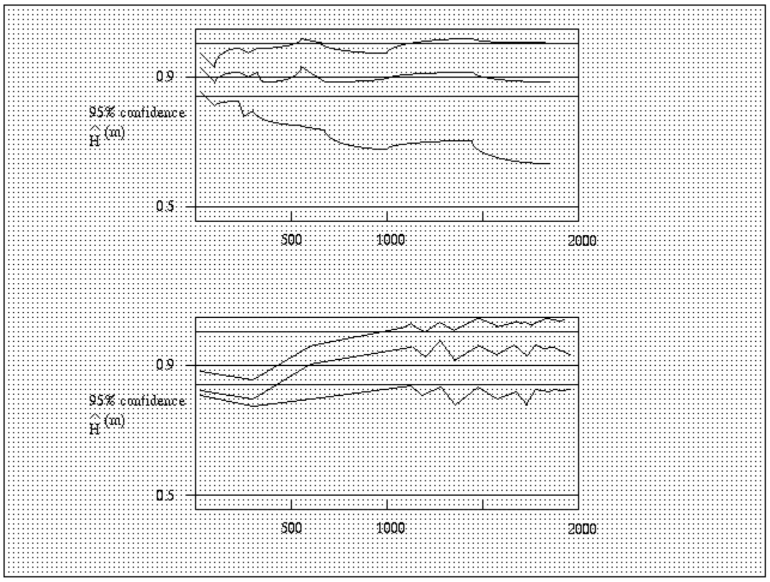

Figure 3.2: Periodogram-based MLE estimate $\hat{H}^{(m)}$ of H and 95% confidence intervals, as a function of aggregation level m. The estimate of H based on the variance-time plot and the R/S-based estimate of H have also been given for the purpose of comparison.

better to use simpler and more effective models. In this section, the major findings from the most recent real network environment measurements are summarized.

3.3.1 Packet Traffic Measurements

Packet traffic measurements collected from different network sources have all shown the existence of fractal nature. These include high time-resolution traffic measurements on Ethernet LAN's reported by Leland et al [4], variable-bit-rate (VBR) video traffic measurements reported by Beran et al [7], present integrated services digital network (ISDN) traffic measurements reported by Hellstern et al [10], and measurements from Common Channel Signaling Network (CCSN) reported by Duffy et al [15].

Packet measurement methods monitor the traffic levels by measuring either packet counts or switch element occupancies. Measurements are typically made over intervals of the order of 15 to 30 minutes. Capacity exhaust of the monitored resource is then indicated if busy period traffic levels consistently exceed a specified rate or a threshold, which is typically in the range of 60%-80% of the maximum throughput of the network element. The threshold is usually determined on the basis of prior testing. Although the existing packet networks are often lightly loaded according to such measurements, i.e., loaded below recommended utilization thresholds, performance degradation due to the network congestion or packet call blocking is observed regularly. The fact that traffic related problems occur at such low occupancy levels may be explained by the fractal nature of packet traffic. Intuitively, the problem arises because packet traffic bursts occur over time scales much smaller than the measurements intervals. As a result, traffic peaks are averaged out to much lower values. All of the measured data sets have, in fact, shown the same qualitative behavior. Erramilli et al investigated the performance-engineering consequences of this effect by using a stream of ISDN packet traffic as the input to an FCFS queuing server with deterministic service times and unlimited buffer space to show that the service time starts to increase very earlier than reaching the utilization threshold [3]. Some of the practical impacts of the packet traffic measurements are investigated in the following subsections.

3.3.2 Heavy-tailed Service Densities

Heavy-tailed densities as a characteristic of fractal processes, are suitable for modeling a number of applications such as call holding times [16], and individual call records [17]. In general, they are expected to be seen in switched data services as well as packet based services when there are resources that need to be held for a duration of a call or a session. As an example, constant bit rate (CBR) services in ATM networks may be mentioned. From the practical point of view, there are numerous difficulties in accurately engineering these services even when the well known insensitivity of the Erlang-B results are used to characterize the service time. The major problem here is the very slow rate of convergence that allows the considerable deviations from the theory over time scales of engineering interest. For a more detailed discussion see reference [17].

Assuming there is a convergence, the rate of convergence problem may be resolved by extending the length of period, however, for long interval observations, the assumptions about the stationarity of arrival processes do not hold and hence the Erlang-B results are not applicable. Intuitively, it looks like that the service rate over smaller time intervals can be much greater than the long term, rate conditioned on a departure, and hence fractal scaling of the service processes should be applicable here.

[3] For a more detailed discussion please see reference [3].

NO. of PKTs	NO. of Losses	Loss Prob	Loss Rate	Fractal Dim
250000	4749	0.019	0.34	0.54
1000000	7315	0.0073	0.27	0.52

Table 3.1: Packet statistics for a typical self-similar pattern.

3.3.3 Packet Losses

Packet loss processes are very well known to be highly bursty although usually characterized by their long term rates. The limitations of using long term rates in order to describe bursty processes and the problem of serial correlation in losses have been identified to be due to the periodicities in the arrival process. The work was done by Ramaswami et al [6], Erramilli et al [11], and Mandelbort [18]. Briefly, any packet loss rate measurement is likely to be arbitrary over a wide range of time scales, and the long term rate is probably too low to be meaningful. On the contrary, with the cases of transmission errors and packet arrivals, fractal characterizations are applicable in describing packet loss processes. In order to illustrate the above mentioned point, Erramilli et al [3] have analyzed the loss processes in simulations driven by Ethernet traffic traces. The following table shows the results of correlation analysis for two data sets. The burstiness of the loss process is indicated by the fractional correlation dimension. The table also shows that when the packet losses occur, they occur at much higher rates than the long term rates, and hence there will be a considerably more impact on the applications than that indicated by the long term rate. In addition, other fractal parameters such as Hurst parameter are also applicable to the loss process.

3.3.4 Fractal Queuing

The presence of fractal properties in actual arrival, service time, and QoS processes may serve as a motivation for the development of the fractal queuing to analyze the performance implications of the processes with long range dependence. One possibility, is that fractal properties impact performance indirectly by biasing the long term traffic measurements, then they can be accounted for transforming inputs to conventional queuing models. The direct analysis of models that use fractal characterizations as the input is another possibility although the lack of a Markovian structure, makes such models extremely difficult to analyze. There are, however, three promising approaches, the first one based on a self-similar stochastic model, specifically, fractal Brownian motion [18], the other one based on dynamical system approach using chaos theory, and the last one based on the neural networks theory. While the first approach is covered briefly, the last two are the main focus of this dissertation and will be discussed in detail.

Chapter 4

Stochastic and Chaotic Modeling of Self-Similar Packet Traffic

4.1 Introduction

Based on what was mentioned in the chapters 1 and 3, it is reasonable to claim that chaos can be used as a strong tool to capture the complexity in ways that can be analyzed and applied in practice. As it was discussed previously , the beauty of chaos is the existence of a low order dynamical system which is able to exhibit complex, seemingly random behavior. Because the trajectories of the chaotic systems are mostly fractal in nature, they may be used as convenient generators of fractal structures. Hence using the concept of chaos, one should be able to capture the complexity of actual packet traffic by low order systems requiring a very few parameters.

The most important goal of this chapter is to demonstrate the modeling power of chaotic maps to parsimoniously describe complex features of the observed packet traffic. This is done by relating the queuing analysis to the study of deterministic nonlinear dynamical systems. The performance approach is considered to be an alternative to traditional Markovian approaches and the performance significance of the approach is shown by analyzing queuing behavior of the maps that are used to capture the fractal properties observed in actual traffic.

So far, it has been demonstrated that there exists a lot of data sources which produce statistically self-similar data. It has also been shown that it is possible to clearly distinguish between actual data traffic and data generated from currently used traffic models and none of the formal models for packet traffic considered in literature are able to capture the self-similar nature of the real traffic. The following lists a number of implications of the self-similar nature of packet traffic for engineering, operations, and performance evaluation of high-speed networks.

(1) Source models for individual users are expected to show extreme variability.

(2) Aggregating self-similar packet streams typically does not produce a smooth Poisson-like process.

(3) The Hurst parameter and fractal dimensions such as correlation dimension provide a more satisfactory measure of burstiness for self-similar traffic than currently used measures such as the index of dispersion of counts.

(4) The nature of congestion produced by self-similar network traffic models differs from that predicted by standard models. More specifically, proposed congestion control schemes for high-speed networks depend on how well they perform under self-similar traffic scenarios.

In this chapter, two new approaches are discussed which deal with the fractal properties of the aggregated traffic. Both approaches, self-similar stochastic modeling and deterministic non-linear chaotic map models, are motivated by the desire for a relatively simple description of the complex packet traffic generation process. Moreover, both modeling approaches introduce a single parameter that describes the fractal nature of traffic. This, in deed, is the Hurst parameter for stochastic models and the fractal dimension for chaotic models. While conventional models are not able to capture the notion of burstiness, these models seem to be able to do so intuitively.
In this chapter the following concepts are studied: First, stochastic-modeling of self-similar traffic is briefly described. Then, chaotic-modeling of self-similar traffic is discussed in detail.

4.2 Stochastic-Modeling

Since in practice one always deals with finite data sets, it is not possible to decide exactly whether the asymptotic relations of section 3.2 hold or not. For the processes that are not self-similar, the correlation will eventually decrease exponentially, the spectral density function looks to be continuous at the origin, the variances of the aggregated processes will decrease as m^{-1}, and the R/S statistic will increase as $n^{0.5}$. For finite sample sizes, distinguishing between these asymptotics and the ones corresponding to self-similar processes is, in general, problematic. This is due to the need for having a very large number of sample data required for estimating the large number of parameters in the short range dependence case. In contrast, the long-range component of the process can be modeled by only one parameter in a self-similar process. There are two formal mathematical models that yield elegant representation of the self-similarity phenomenon but not provide any physical explanation. These are namely fractional Gaussian noise and the class of fractional autoregressive integrated moving average (ARIMA) processes. Fractional Gaussian noise $X = (X_k : k \geq 0)$ with parameter $H \in (0, 1)$ is a stationary Gaussian process with mean μ, variance σ^2, and autocorrelation function $r(k) = 1/2(|k+1|^{2H} - |k|^{2H} + |k-1|^{2H}), k > 0$. In [23], it has been shown that this process is exactly second-order self-similar with self-similarity parameter H, as long as $1/2 < H < 1$. The estimation methods for the three unknown parameters μ, σ^2, and H have already been addressed in section 3.2. Leland et al in [4] say that Joyeux and Hausking have shown that Fractional ARIMA processes with the parameters (p, d, q) are asymptotically second-order self-similar with the self-similarity parameter $d + 1/2$, as long as $0 < d < 1/2$. Fractional ARIMA processes are much more flexible than fractional Gaussian noise with regard to the simultaneous modeling of the short-term and long-term behavior of a time series. This is mainly because the latter, having only three parameters μ, σ^2, and H, has a very rigid correlation structure and is not capable of capturing the wide range of low-lag correlation structures encountered in practice.

Finally, a construction of self-similar processes based on aggregating many simple renewal processes exhibiting inter-renewal times with infinite variances is mentioned in [22]. Although the work was originally cast in an economical frame work involving commodity prices, it was found to be appealing in the context of high-speed packet traffic. In the simplest form, this construction requires a sequence of i.i.d integer valued random variables $U_0, U_1, U_2, ...$ - as an interpretation of inter-renewal times- with heavy tails as it was mentioned earlier, i.e., with the property

$$P[U \geq u] \sim u^{-\alpha} h(u), \quad \text{as} \qquad u \to \infty \tag{4.1}$$

where H is slowly varying at infinity and $0 < \alpha < 2$. Furthermore, let $W_0, W_1, W_2, ..$ be an independently identically distributed (i.i.d) sequence of rewards with mean zero and finite variance, independent of U's. Next let

$$S_k = S_0 + \sum_{j=1}^{k} U_j, k \geq 0 \tag{4.2}$$

denote the delayed renewal sequence derived from $(u_j)_{j\geq 0}$ where S_0 is chosen such that the sequence $S_k \quad k \geq 0$ is stationary. The renewal reward process $W = (W(t) : t = 0, 1, 2, ..)$ is then defined by

$$W(t) = \sum_{k=0}^{t} W_k I_{(S_{k-1}, S_K)}(t) \tag{4.3}$$

with $I_A(.)$ denoting the indicator function of set A. By aggregating M i.i.d copies $W^{(1)}, W^{(2)}, ..., W^{(M)}$ of W, the model of interest is obtained, namely the process W^* given by

$$W^*(T, M) = \sum_{t=1}^{T} \sum_{m=1}^{M} W^{(m)}(t) \tag{4.4}$$

with $W^*(0, M) = 0$. It has been shown in [24] and [25] that for T and M both large with $T \ll M$, W^* behaves like fractional Brownian motion, i.e., properly normalized $W^*(T, M)$ converges to the integrated version of fractional Gaussian noise. In other words, it converges to a zero-mean Gaussian process

$$B_H = (B_H(s) : s \geq 0, \quad 1/2 < H < 1) \tag{4.5}$$

with the correlation function

$$R(s, t) = 1/2(s^{2H} + T^{2H} - \mid s - t \mid^{2H}) \tag{4.6}$$

4.3 Chaotic Modeling

In this section the application of deterministic chaotic maps to model traffic sources in packet based networks is investigated. Also a potential performance analysis based on chaotic maps that can be used to assess the traffic significance of fractal properties is introduced. It is shown that the low order nonlinear maps can capture a number of the fractal properties observed in actual data. As a result while there are considerable analytical difficulties, chaotic maps may allow accurate and concise models of packet traffic with a potential for transient and steady state analysis.

As it was mentioned previously, recent measurements have revealed that actual packet traffic has features that can be more efficiently described in terms of fractal processes. Although fundamentally chaos and fractals are independent and unrelated concepts, fractal geometry is invoked to describe the irregular trajectories of chaotic systems in the state space as it was discussed in chapter one. Hence, chaotic systems are often used as convenient generators of fractal processes. The most important goal of this section is to use the above mentioned concept to find precise and tractable models of complex traffic processes in packet based networks and to demonstrate the potential for analyzing queuing processes using nonlinear dynamical systems.

Specifically, the simplest class of chaotic systems, known as one-dimensional chaotic maps, in which the evolution of a state variable x over discrete time n is described by a deterministic nonlinear map $x_{n+1} = f(x_n)$, is analyzed. Packet traffic sources may be modeled by establishing a relationship between the state variable x_n and the source activity. One of the elegant approaches is to consider the source is active and generating traffic at a peak rate if the state variable exceeds a threshold, and is idle otherwise. This is specially important when one tries to predict the behavior of ON-OFF source models. The modeling challenge is then to find a mapping f such that the generated actual traffic has the prescribed statistical behavior. It is also mentioned that the usage of two-dimensional deterministic transformations has some potential for the transient and steady state analysis of queuing systems driven by fractal arrival processes.

There are a number of motivations for using deterministic chaotic maps as models of packet traffic:

- Chaotic maps can match the range of packet traffic behavior. Depending on the application, packet traffic can be constant or periodic, or random with correlations ranging over many time scales. As an example, the well known logistic map can capture the full range of dynamic behavior by adjusting a single parameter.

- Parsimonious modeling of wide range of ON-OFF source models. A useful abstraction in packet traffic modeling is the notion of two state ON-OFF behavior. In this model, the source is considered to transmit at a peak rate in the ON state and is considered to be silent in the OFF state with a geometrically distributed times in the two states. Hellstern et al [10] have indicated the need to generalize this abstraction to include a wide range of ON-OFF behavior.

- Deterministic nature of packet models. There is a degree of determinism in packet models, traffic sources as well as service times. Recent measurements indicate that over short time scales, there are complex, deterministic correlations in packet traffic. This is shown in figure 4.1, which shows a phase plot of successive interarrival times (t_{i+1} vs. t_i), for Ethernet traffic data. The deterministic structure (clusters, vertical and horizontal lines, and diagonals) is obviously striking. Erramilli et al [1] have obtained qualitatively similar plots for every other Ethernet and ISDN analyzed data sets which shows that these structures are protocol or application independent. Structures can be explained mostly by streams of deterministically spaced packets, and their superpositions. Deterministic chaotic models may allow a more compact description of these structures.

- Fractal properties in packet traffic. As described earlier, chaotic maps are suitable generators of fractal properties. Therefore, it is important to assess the impacts of these properties on packet network design and engineering. As an example, the intermittency map to be discussed, has two parameters and is able to capture many of these features.

- Need for transient analysis. Packet networks are capable of having interesting time behavior that can substantially impact performance and hence it is important to analyze both their transient and steady state dynamical behavior. For a given chaotic map as a reasonable model of traffic there exists the possibility of modeling the dynamic behavior of the packet network.

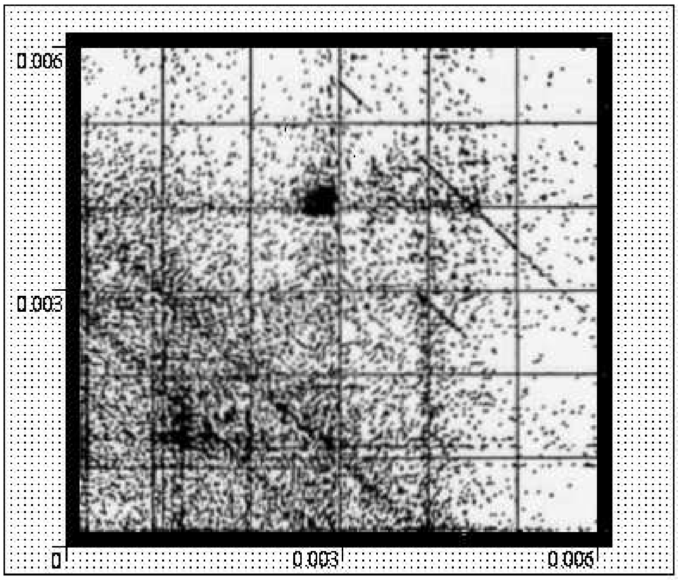

Figure 4.1: Phase plot of successive interarrival times t_{i+1} versus t_i, showing deterministic structure.

- Experience in other disciplines. Chaotic maps have been proposed as alternatives to stochastic models in a number of other disciplines. They have been used to generate random numbers [28], as well as to model ion channel activity in nerve cells [9].

4.3.0.1 Chaotic maps as models of packet traffic sources

In this section the approach to model a wide range of ON-OFF source models encountered in packet traffic analysis is introduced. Consider a one-dimensional map in which the state variable x_n evolves over discrete time $n = 0, 1, 2, ...$ according to the following nonlinear map:

$$x_{n+1} = \begin{cases} f_1(x_n) & : \quad (0 \leq x_n < d) \\ f_2(x_n) & : \quad (d \leq x_n \leq 1) \end{cases} \tag{4.7}$$

For the map to be chaotic, $f_1(.)$ and $f_2(.)$ should satisfy the SIC. Note that the formulation is completely deterministic and a given initial condition fully defines a trajectory in the phase plane. This is analogous to a realization of a stochastic process. A packet generation process now can be modeled by assuming that the source is in a passive or active state at time n depending on whether x_n is below or above a threshold. This is shown in figure 4.2. Every iteration of the map in active state is taken to generate a packet or a batch of packets. Therefore the packet arrival process is described by the output of the state space model y_n as

$$y_n = \begin{cases} 0 & : \quad (0 \leq x_n < d) \\ 1 & : \quad (d \leq x_n \leq 1) \end{cases} \tag{4.8}$$

It is also possible to define other source interpretations, e.g., a different threshold for the indicator variable y_n from the threshold used for $f(.)$ may be defined. Alternatively, it is possible to define a dynamical system that directly generates interarrival times. Furthermore, it is also possible to introduce additional source states by considering maps with more than two segments. Finally, the usage of continuous time processes is applicable. In

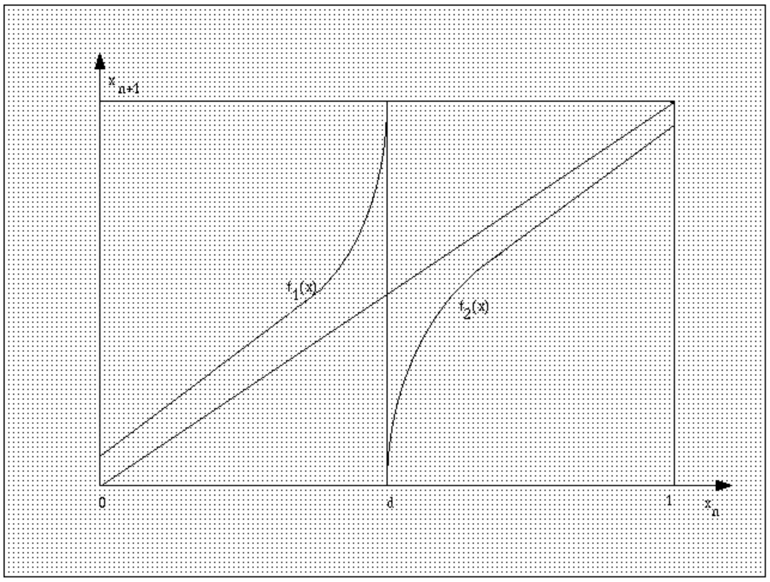

Figure 4.2: A simple nonlinear map as a packet generator. When $x_n \geq d$ the source is active, and is generating packets at a peak rate.

this section, only the discrete sources described by equations 4.7 and 4.8 are considered. In practice, the y_n is observed while the x_n is hidden and the challenge is to find suitable $f_1(.)$ and $f_2(.)$ such that y_n matches the properties of the actual traffic relevant to the queuing performance.

As the map evolves along a trajectory, a reasonable question to ask is with what frequency the trajectory visits a given neighborhood, $(x, x + dx)$ over an observing interval of n iterations. Such questions are addressed by investigating the effect of the map on an arbitrary density of points. The effect of one iteration of the map is to move a point x_n to $f(x_n)$, i.e., a density $\delta(x - x_n)$ is transformed to $\delta(x - f(x_n))$. Using the well-known random Nikodym theorem, the effect of one iteration of the map on an arbitrary density of points $\rho_n(x)$ is then given by:

$$\rho_{n+1}(x) = \int_0^1 \delta(x - f(z))\rho_n(z)\,dz \tag{4.9}$$

If $\rho_n(x)$ is independent of time n, it is called invariant density of the map $f(x)$ as it was mentioned in chapter one and it describes the density of the iterates of x_n in the interval $(0, 1)$ as $n \to \infty$. The invariant density is the solution of the Frobenius-Perron equation described as:

$$\rho(x) = \int_0^1 \delta(x - f(z))\rho(z)\,dz \tag{4.10}$$

or in operator notation, $\rho = P\rho$, where P is referred to as the Frobenius-Perron operator. Note that here an implicit assumption that the invariant density is independent from the initial condition x_0 is made. Given that an initial condition corresponds to a distinct trajectory, this is equivalent to assume that the concept of ergodicity holds. This is a property of underlying map and does not hold in general, i.e., different initial conditions may lead to distinct attractors, or steady state solutions.
From the definition of P, it is easy to see that the following properties hold:

(p1) $P(af + bg) = aP(f) + bP(g)$
(p2) $P(f) >= 0$ if $f >= 0$
(p3) $\int_X P(f(x))\mu(\,dx) = \int_X f(x)\mu(\,dx)$
(p4) If S_n is the n-fold composition of S and P_n is the Frobenius-Perron operator corresponding to S_n, then $P_n = P^n$, where P is the Frobenius-Perron operator corresponding to S.

The first step in analyzing the properties of the time series y_n is to find the invariant density $\rho(x)$ of the associated map. In the following, the known results from the numerical approaches to solve relation 4.10 are given.

Theorem 4.1 [27]: Let (X, Σ, μ) be a measure space, $S : X \to X$ a nonsingular transformation and P the Frobenius-Perron operator associated with S. Consider a nonnegative $f \in L^1$. Then a measure μ_f given by

$$\mu_f(B) = \int_B f(x)\mu(dx) \tag{4.11}$$

is invariant if and only if f is a fixed point of P.

In general, P may have several fixed points, e.g., the Bernoulli shift to be discussed later in this chapter has two fixed points. A density function is also assumed to be asymptotically stable.

Definition 4.1 [5]. Let (X, Σ, μ) be a measure space, $P : L^1 \to L^1$ be a Frobenius-Perron operator. Then $\{P^n\}$ is said to be asymptotically stable if there exists a unique $f^* \in D$ such that $Pf^* = f^*$ and

$$\lim_{n\to\infty} \|P^n f - f^*\| = 0 \qquad \forall f \in D \tag{4.12}$$

Thus for any mapping with asymptotically stable associated P, the invariant density can be calculated by repeatedly operating P on an arbitrary initial density. The following theorem assures the asymptotic stability of a class of the maps defined earlier.

Theorem 4.2 [27]: If $S : [0,1] \to [0,1]$ satisfies the following conditions

(c1) There is a partition $0 = a_0 < a_1 < ... < a_r = 1$ of $[0,1]$ such that for each integer $i = 1, ..., r$ the restriction of the S to the interval (a_{i-1}, a_i) is a C^2 function;
(c2) $S(a_{i-1}, a_i) = (0,1)$, that is, S is onto;
(c3) There is a $\lambda > 1$ such that $S'(x) \geq \lambda$ for $x \neq a_i, i = 0, ..., r$;
(c4) There is a real finite constant c such that

$$-S''(x)/[S'(x)]^2 \leq c$$

for

$$x \neq a_i, \qquad i = 0, ..., r$$

and P is the Frobenius-Perron operator associated with S, then $\{P^n\}$ is asymptotically stable.

The dynamical system representing the source can be coupled to another one describing the evolution of a queue to provide a frame work for performance analysis to be discussed next.

4.3.0.2 Performance analysis

In this section an approach to the analysis of queuing systems driven by arrival processes generated by chaotic maps is discussed. The approach is illustrated by considering a server with a deterministic service time equal to the iteration interval of the underlying chaotic maps. This is the representative of cell processing in ATM networks. The performance of the server can be analyzed in the framework of a discrete queuing system whose state at time n is given by the queue length l_n. If every iteration of the map in the active state produces a batch of k packets, the server can be completely described by the following coupled system of nonlinear equations:

$$l_{n+1} = \max(l_n - 1, 0) + k y_n(x_n) \tag{4.13}$$

$$x_{n+1} = f(x_n) \tag{4.14}$$

where the max(.,.) function accounts for the boundary conditions at $l_n = 0$. Note that relation 4.13 and relation 4.14 are fully deterministic, but the formulation nevertheless captures randomness in the arrival process. The deterministic formulation allows an interesting geometric interpretation of a queuing system. The state of the source and the queue is represented by the point (x_n, l_n) and the evolution of the queue from any initial condition which is fully governed by relation 4.13 and relation 4.14 can be represented by the motion in the plane with $x_n \in (0, 1)$ and integer l_n. Figure 4.3 illustrates the idea. Specifically, this queuing system can be viewed as a two-dimensional deterministic transformation $\Phi(x, l)$ in which

(1) Segments $(x_n \geq d, l_n)$ are expanded according to the function $f_2(x)$ and mapped onto the segment $(x_n \in (0, 1), l_n + 1)$ except at the boundary $l_n = 0$ which maps onto the segment $(x_n \in (0, 1), 2)$.

(2) Segments $(x_n < d, l_n)$ are stretched according to $f_1(x)$ and mapped onto the segment $(x_n \in (0, 1), l_n - 1)$ except at the boundary $l_n = 0$, which maps onto the segment

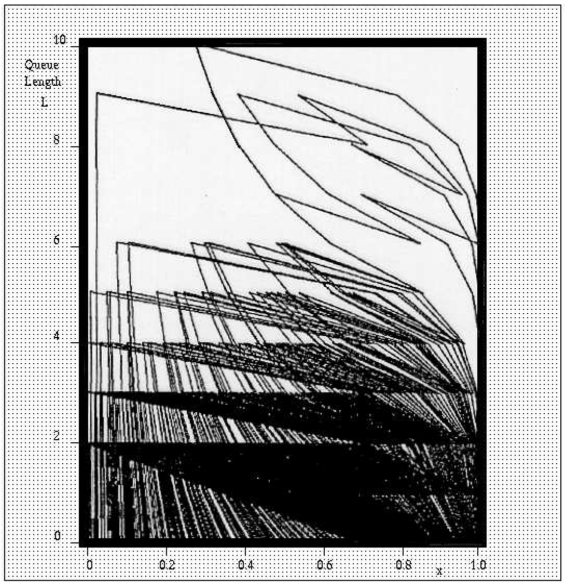

Figure 4.3: Queuing as motion in the (x, l) plane. 10'000 iterations of relation 4.13 and relation 4.14 using the intermittency map and a batch size of $k = 2$.

$(x_n \in (0, 1), 0)$.

When the source is active, the system moves up and to the left, where as in the passive state, it moves down and to the right. The case of $k = 2$ is therefore analogous to a generalized birth and death process and is of special interest.

Theorem 4.3 [5]: Given the queuing system described by relation 4.13 and relation 4.14 with $k = 2$, the probability mass function of the queue lengths $P(l)$ is given by

$$\begin{aligned} P(l+1) &= \frac{A_l}{\overline{A_{l+1}}} P(l), \qquad (l > 2) \\ P(1) &= \frac{A_0}{\overline{A_1}} P(0) \\ P(2) &= \frac{1}{\overline{A_2}} P(1) \end{aligned} \tag{4.15}$$

where

$$A_l = \int_d^1 \rho(x|\, l)\, dx, \qquad \overline{A_l} = 1 - A_l \tag{4.16}$$

and $\rho(x|l)$ is the density of iterates of the 2-D map over the line l. See [5] for a formal proof.

This approach can be generalized to more general batch sizes, and service time distributions, e.g., by representing the state of the system by the virtual waiting time process modeled by another continuous 1-D map. The queuing system is then viewed as a continuous 2-D transformation over the plane.

4.4 Chaotic Modeling Research Challenges

In this section, the current research issues are discussed. This basically includes different chaotic maps that are suitable for the task of modeling and also the tools for evaluating the performance characteristic of the models from the theory of nonlinear systems and chaos. The most important tool is the histogram analysis discussed in chapter one which is a suitable tool to investigate whether the chaotic model characteristics match the real characteristics of the traffic or not.

4.4.1 Piecewise linear maps as traffic source models

In this case, the chaotic map is considered to consist of a number of linear branches. From chapter one, it is obvious that the invariant density of such a map satisfying conditions of example (1.1) is uniform, i.e., $\rho(x) = 1$.

Proof: It is sufficient to show the $\rho(x) = 1$ is the fixed point of the Frobenius-Perron operator. For the class of maps in example (1.1), relation 4.10 becomes

$$\rho(x) = \sum_i \int_{a_i}^{a_{i+1}} \delta[x - f_i(z)]\rho(z)\,dz = \sum_i \frac{\rho(f_i^{-1}(x))}{|f_i'(f_i^{-1}(X))|} \tag{4.17}$$

from the definition of the map $f_i'(f_i^{-1}(x)) = c_i = (a_{i+1} - a_i)^{-1}$, and substituting $\rho(x) = 1$ yields $\sum_i (a_{i+1} - a_i) = 1 = \rho(x)$.

It is also possible to show that an arbitrary but normalized initial density converges to this uniform density under repeated operations of relation 4.17. Consider the evolution of a normalized initial density $\rho_0(x)$ after m iterations of the map

$$\rho_m(x) = \int_0^1 \delta[x - f^m(z)]\rho_0(z)\,dz \tag{4.18}$$

The effect of m iterations of the map is to segment the interval $(0, 1)$ into r^m segments $(0, d_1), (d_1, d_2), ..., (d_{r^m-1}, 1)$. Then relation 4.18 can be written as

$$\rho_m(x) = \sum_{j=0}^{(r^m-1)} \int_{d_j}^{d_{j+1}} \delta[x - f^m(z)]\rho_0(z)\, dz \tag{4.19}$$

substituting for $u = f^m(z)$, and noting that $\frac{du}{dz} = \frac{1}{d_{j+1}-d_j}$ is in the interval (d_j, d_{j+1}) the following relation can be obtained

$$\rho_m(x) = \sum_{j=0}^{(r^m-1)} (d_{j+1} - d_j)\rho_0(u) \tag{4.20}$$

As the number of iterations tends to infinity the sum can be replaced by the integral as,

$$\rho_m(x) \to \int_0^1 \rho_0(u)\, du = 1 \tag{4.21}$$

Hence $\rho(x) = 1$ is the only relevant steady state solution.

Example 4.1. The Bernoulli shift is an example consisting of two linear segments to model the active and passive states of the source

$$x_{n+1} = \begin{cases} \frac{x_n}{(1-\lambda)} & : \quad (0 \leq x_n < (1 - \lambda)) \\ \frac{x_n-(1-\lambda)}{\lambda} & : \quad ((d = 1 - \lambda) \leq x_n \leq 1) \end{cases} \tag{4.22}$$

Figure 4.4 shows the Bernoulli shift. The associated indicator variable y_n representing the packet generation process is as before equal to 1 when x_n exceeds the threshold d, and is 0 otherwise. It is shown that the y_n sequence form a Bernoulli process. Note that the absolute values of the slope $\alpha_i \quad (i = 1, 2)$ of the two linear segments are greater than one and hence the map is expansive. The fact that the invariant density is uniform, is used to derive the following results for the output.

(1) $E[y_n] = \int_{(1-\lambda)}^1 \rho(x)\, dx = \lambda$
(2) Likewise, $E[y_n^2] = \lambda$, and $Var[y_n] = \lambda(1 - \lambda)$
(3) The autocovariance of the time series y_n is calculated as

$$E[y_n y_{n+m}] = P(y_{n+m} = 1, y_n = 1) = \int_{1-\lambda}^1 \rho(x) y_{n+m}(f^m(x))\, dx \tag{4.23}$$

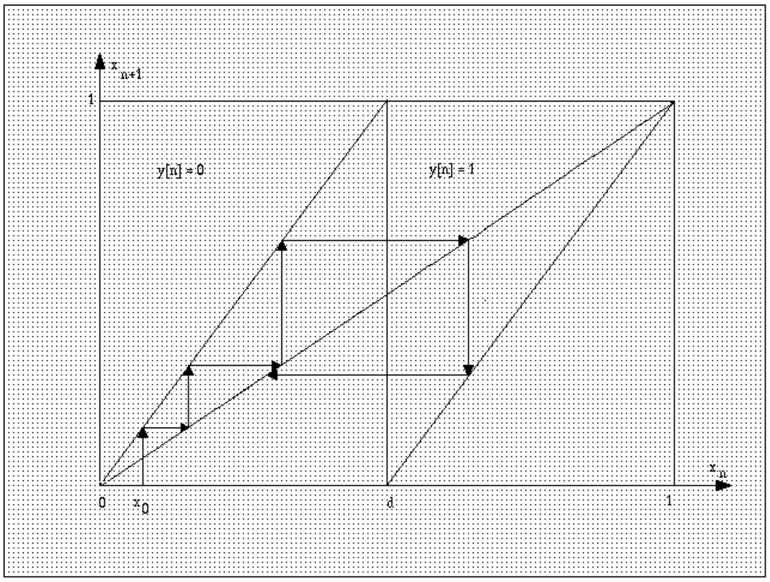

Figure 4.4: Evolution of the state variable from an initial condition in the Bernoulli shift.

In the above relation, $f^m(x)$ is a piecewise linear with 2^m segments defined over intervals (a_j, a_{j+1}) for $(j = 0, 1, 2, ..., 2^m - 1)$ where $a_0 = 0$, and $a_{2^m} = 1$. Each interval is mapped onto $[0, 1]$ and by substituting $z = f^m(x)$ in the above integral the following result is obtained.

$$P(y_{n+m} = 1, y_n = 1) = \sum_{j=2^{m-1}}^{2^m-1} \int_{a_j}^{a_{j+1}} \rho(x)\, dx = \lambda \sum_{j=2^{m-1}}^{2^m-1} (a_{j+1} - a_j) = \lambda^2 \tag{4.24}$$

One can show with the similar line of reasoning,

$$P(y_{n+m} = 0, y_n = 0) = (1 - \lambda)^2$$

$$P(y_{n+m} = 0, y_n = 1) = \lambda(1 - \lambda) \tag{4.25}$$

Thus the output of the map is a Bernoulli shift process with arrival rate λ, even though the time series of the state variable x_n is correlated and has a mean value of 0.5.

This example is further used to investigate the performance analysis of the modeling approach. Fortunately, the same balance equations that one would obtain from a conventional Birth Death process formulation of a discrete queue with batch Bernoulli arrivals of batch size of 2 and service time of 1, are generated for the arrivals of the Bernoulli shift of relation 4.22 and $\rho(x|l) = 1$ as

$$
\begin{aligned}
P(l+1) &= \frac{\lambda}{1-\lambda} P(l), (l > 2) \\
P(1) &= \frac{\lambda}{1-\lambda} P(0) \\
P(2) &= \frac{1}{1-\lambda} P(1)
\end{aligned}
\tag{4.26}
$$

This is important to note that while the equations are the same, the underlying physical frame works are very different, i.e., in the present frame work, the queue length is the marginal invariant density describing the evolution of a 2-D deterministic dynamical system. The solution of the above equation has already been shown to be

$$P(l) = \frac{\lambda^{l-1}}{(1-\lambda)^l}(1-2\lambda) \tag{4.27}$$

and can be verified by direct substitution.

Now let us move on a little further. While the Bernoulli arrivals are used to model the simplest traffic models, they are obviously not able to capture the clustering, or burst issues observed in the typical packet networks. More sophisticated traffic models may be simply obtained by removing the restriction that each linear segment is mapped onto the whole interval $[0, 1]$. The following defines a class of piecewise linear expanding maps known as Markov maps:

(1) There is a partition $0 = a_0 < a_1 < ... < a_r = 1$ of $[0, 1]$ such that for each integer $i = 1, ..., r$ the restriction of f to the interval (a_{i-1}, a_i) is linear.
(2) The partition endpoints are mapped onto the partition endpoints, i.e., $f(a_i) = a_j$.
(3) There is a $\lambda > 1$ such that $|f'(x)| \geq \lambda$ for $x \neq a_i$, $i = 0, ..., r-1$.

Hunt et al [26] have shown that the Frobenius-Perron operator for Markov maps has a matrix representation which can be solved to calculate the invariant density. This motivates a general technique to calculate the invariant density of an arbitrary nonlinear map approximated by a Markov map. This is achieved by discretizing $\rho(x)$ and solving the resulting matrix equations. The solution converges asymptotically to the exact invariant density, i.e., when the number of discretization steps tends to infinity.

The sojourn times of the map in each state are defined as the number of iterations from the instant the map is injected into the segment till it leaves the segment. Because the sojourn times are of special interest in this kind of application, the following theorem is given to show the linearity in the vicinity of a fixed point leads to geometrically distributed sojourn times.

Theorem 4.4 [5]: Let $f : [0,1] \to [0,1]$ be a Markov map with piecewise linear segments defined over partitions (a_{i-1}, a_i) and $x_{-1} \notin (a_{i-1}, a_i), x_0 = f(x_{-1} \in (a_{i-1}, a_i))$. Further let K be such that for

$$k < K, f^k(x_0) \in (a_{i-1}, a_i), f^K(x_0) \notin (a_{i-1}, a_i)$$

If the followings hold

(1) The re-injection density $\gamma_i(x_0)$ into segment i is constant over the interval, where $\gamma_i(x_0)$ defined by

$$\gamma_i(x_0) = \sum_{(} j \neq i) \frac{\rho(f_j^{-1}(x))}{|f_j'(f_j^{-1}(x))|} \tag{4.28}$$

(2) There is a fixed point of the map f within the interval $[a_{i-1}, a_i]$, such that $f(\hat{x}) = \hat{x}$.

Then the CDF is given by $P(K > k) = \alpha_i^{-k}$, $(k = 0, 1, 2, ..)$.

Example 4.2: The Liebovitch map has been studied by Liebovitch et al [9] in a modeling study of ion channel activity that yields independent silence periods. The map is defined as

$$\begin{aligned} x_{n+1} = \alpha_1 x_n &\quad : \quad (0 < x_n \leq d_1) \\ x_{n+1} = \frac{d_2 - x_n}{d_2 - d - 1} &\quad : \quad (d_1 < x_n \leq d_2) \\ x_{n+1} = 1 - \alpha_2(1 - x_n) &\quad : \quad (d - 2 < x_n < 1) \end{aligned} \tag{4.29}$$

where d_i $(i = 1, 2)$ are the subintervals endpoints and α_i $(i = 1, 2)$ denotes the slopes of the linear maps corresponding to the active and passive states, respectively. Figure 4.5 shows the Liebovitch map. The map is, in fact, able to generate an interrupted Bernoulli

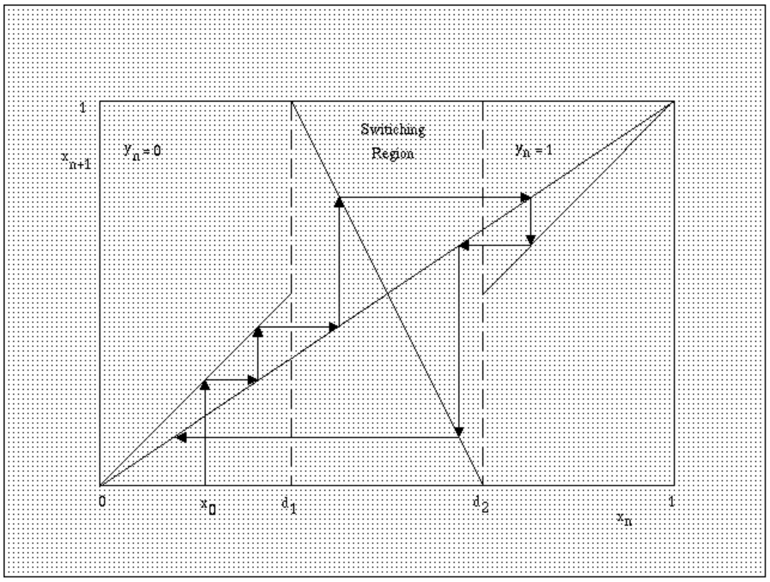

Figure 4.5: The Liebovitch map: an interrupted Bernoulli process generated by adding an additional piecewise linear segment.

process and consists of three piecewise linear segments on non-overlapping subintervals of the interval $(0, 1)$. The three segments may be interpreted as various states of the source, e.g., the first and the third segments can be interpreted as the passive and active phases, with the second segment showing the switching region between the states. The state variable evolves according to the map's equation and produces a variety of dwell times in both active and passive states as well as in the switching region. The two endpoints, 0 and 1, are absorbing states. Sustained switching between the two states occurs only when both α_i are greater than one. The values of α_i and d_i determine dwell times in each state as well as in the switching region. Constraining the four parameters of the map, it is possible to achieve the desirable behavior. As an example, the map is desired to spend minimum time in the switching region. In deed, imposing appropriate constraint as follows creates a situation that both subintervals $(0, d_1)$ and $(d_2, 1)$ are reachable from each other after exactly two iterations passing the switching region.

$$\begin{aligned} \alpha_1 &= \frac{d_2}{d-1}[1-(d_2-d_1)] \\ \alpha_2 &= \frac{1}{d_2-1}[(d_2-1)-d_1(d_2-d_1)] \end{aligned} \tag{4.30}$$

Applying relation 4.10, the invariant density equations of the Liebovitch map can be written as

$$\begin{aligned}\rho(x) &= \frac{1}{\alpha_1}\rho\,(\frac{x}{\alpha_1}I(0 < x \leq \alpha_1 d - 1) + (d_2 - d_1)\rho[d_2 - (d - 2 - d - 1)x] \\ &+ \frac{1}{\alpha_2}\rho\,(\frac{x + \alpha_2 - 1}{\alpha_2})I((1 - \alpha_2(1 - d_2)) \leq x < 1)\end{aligned} \tag{4.31}$$

where the $I(L_i)$ is the indicator function of the interval L_i as it was mentioned in chapter one, i.e.,

$$I(L_i) = \begin{cases} 1 & : \quad x \in L_i \\ 0 & : \quad x \notin L_i \end{cases} \tag{4.32}$$

Applying the required conditions in relation 4.31 and solving the resulting matrix equation yields the following expressions for the invariant density.

$$\begin{aligned}\rho_1 = \beta &\quad : \quad 0 \leq x < d_1 \\ \rho_2 = \frac{\beta}{\alpha_2} &\quad : \quad d_1 \leq x < \alpha_1 d_1 \\ \rho_3 = 0 &\quad : \quad \alpha_1 d_1 \leq x < 1 - \alpha_2(1 - d_2) \\ \rho_4 = \beta\frac{\alpha_1 - 1}{\alpha_1(d_2 - d_1)} &\quad : \quad 1 - \alpha_2(1 - d_2) \leq x < d_2 \\ \rho_5 = \beta\frac{\alpha_2(\alpha_1 - 1}{\alpha_1(d_2 - d_1)} &\quad : \quad d_2 \leq x < 1\end{aligned} \tag{4.33}$$

where β may be found from the normalization condition $\int_0^1 \rho(x)\,dx = 1$. Again, the solution can be verified by substituting in relation 4.31. It is worth paying attention to the point that there is a transient interval in the switching region for which once the map leaves that interval it will not revisit it and therefore the invariant density is zero in it. Figure 4.6 shows a graphical sketch of the invariant density function. From the invariant density equations one can simply derive the source statistics as

$$\begin{aligned}E[y] &= E[y^2] = \rho_5(1 - d_2) \\ Var[y] &= \rho_5(1 - d_2)[1 - \rho_5(1 - d_2)]\end{aligned} \tag{4.34}$$

Based on theorem 4.4, it may be said that the sojourn times in the active state are geometrically distributed, and it is also possible to make the idle periods geometrically distributed by not advancing the time index in the switching region. The Liebovitch map is hence, analogous to widely used stochastic models such as interrupted Bernoulli process or interrupted Poisson model and may be used to explain a way of constructing piecewise

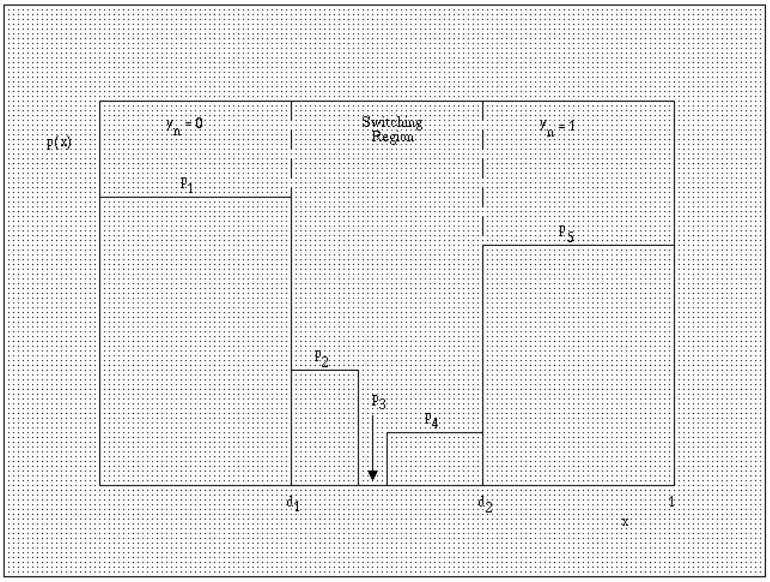

Figure 4.6: Piecewise uniform invariant density of the Liebovitch map.

linear maps in which the sojourn times in each partition are geometrically distributed. It is also important to mention that the usage of chaotic maps does not offer any advantage in this regard and the real power of chaotic maps lies in using the nonlinear segments in order to capture fractal properties.

4.4.2 Piecewise nonlinear maps as traffic source models

Using nonlinear segments in a neighborhood of a fixed point leads to have a more general sojourn time behavior. More specifically, the following theorem indicates a way of realizing sojourn times that are distributed as power laws.

Theorem 4.5 [5]: Let $f : [0,1] \to [0,1]$ be an expanding map defined over partitions (a_{i-1}, a_i) and $x_{-1} \notin (a_{i-1}, a_i), x_0 = f(x_{-1}) \in (a_{i-1}, a_i)$. Further let K be such that for $k < K, f^k(x_0) \in (a_{i-1}, a_i), f^K(x_0) \notin (a_{i-1}, a_i)$.

(1) If $f_i(x)$ can be translated to the canonical form where $x \in [0, d]$ and $c = (1 - d^{m-1})/d^{m-1}$, then,

$$g(x) = \frac{x}{(1 - cx^{m-1})^{\frac{1}{m-1}}}$$

(2) If the re-injection density $\gamma(x_0)$ is slowly varying over the interval, then the CDF for K is given by

$$P(K > k) \sim \gamma(0) k^{-\frac{1}{m-1}}, k \to \infty$$

Choosing a rapidly evolving map in the pre-images of a neighborhood leads to have a slowly varying re-injection density into that map. If the re-injection point x_0 is not close to the fixed point, the resulting sojourn time is relatively short. However x_0 can be close enough to the fixed point with a finite probability such that the state variable evolves very slowly as

$$x_{n+1} \approx x_n + \frac{c}{m-1} x_n^m \tag{4.35}$$

In this case, it takes many iterations for the map to exit this segment. This is an example of a phenomenon called intermittency in which system behavior is dominated by infrequent variations of large amplitude. Figure 4.7 shows this phenomenon.

In specific, choosing m in the range $(3/2, 2)$ generates heavy-tailed sojourn times with $1 < \alpha < 2$. In the following examples several maps that exploit this behavior are introduced to generate the wide range of time scales observed in packet traffic measurements.

Example 4.3: Fixed point single intermittency map with index m = 2. Consider the map:

$$x_{n+1} = \begin{cases} \frac{x_n}{(1-cx_n)} & : \quad (0 \leq x_n < d) \\ \frac{x_n - d}{1-d} & : \quad (d \leq x_n \leq 1) \end{cases} \tag{4.36}$$

where $c = \frac{1-d}{d}$.

From theorem 4.5 the sojourn times in the inactive state $(x_n < d)$ are described by a power law. It can be verified by substitution that the α/x is the fixed point of the Frobenius-Perron operator. Note that because condition (c3) of theorem 4.2 is violated at point $x = 0$, it is not possible to normalize the equation and interpret it as a probability density. Physically, the map spends so many iterations in a neighborhood of the fixed point zero that in effect makes the invariant measure unnoramlizable. In practice the length of intervals limit the observation time of interest but it is possible to use a perturbation of intermittency maps in order to bound the maximum length of sojourn times. This is illustrated using the single intermittency map.

Example 4.4: Single intermittency map. This map is an extension of the Bernoulli shift in which the piecewise linear map representing the passive interval is replaced by a nonlinear segment. The map is then defined as

$$x_{n+1} = \begin{cases} \epsilon + x_n + cx_n^m & : \quad (0 \leq x_n < d) \\ \frac{x_n - d}{1-d} & : \quad (d \leq x_n \leq 1) \end{cases} \tag{4.37}$$

where $c = \frac{1-\epsilon-d}{d^m}$.

The active period $d \leq x_n \leq 1$, as before, is presented by a piecewise linear map. The main

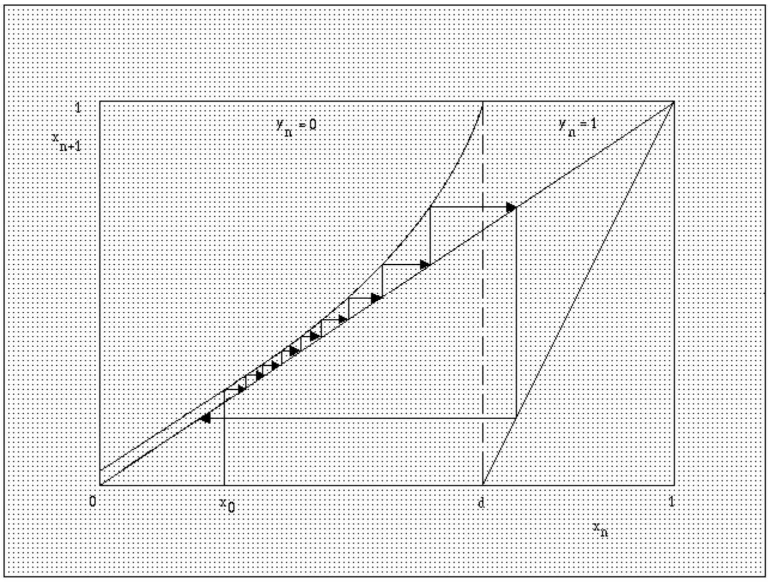

Figure 4.7: Single intermittency map with the property that it may spend many iterations in the narrow region, generating idle periods spanning many time scales.

objective of the modeling procedure is the steady state behavior of the idle periods which are related to many of the fractal properties observed in actual data. In the following, the properties of a source model resulting from a quadratic intermittency map ($m = 2$) are compared with those of an actual traffic source. First, note that substituting for $f_1(.)$ and $f_2(.)$ in relation 4.10 yields the following equation for the invariant density:

$$x_{n+1} = \begin{cases} (1-d)\rho[(1-d)x + d] & : \quad (0 \leq x < \epsilon) \\ \frac{\rho\frac{\sqrt{1+4cx-4c\epsilon}-1}{2c}}{\sqrt{1+4cx-4c\epsilon}} + (1-d)\rho[(1-d)x + d] & : \quad (\epsilon \leq x \leq 1) \end{cases} \tag{4.38}$$

As an illustration of the difficulties encountered associated with nonlinear systems analysis, it should be noted that an exact analytical solution does not appear feasible because of the discontinuities in the given invariant density. However, since the map satisfies the expansion condition, $\inf_x |(f^k)'| > 2$ for some integer $k \geq 1$ [26], the invariant density can be numerically calculated by assuming a uniform arbitrary initial density and iterating over corresponding to relation 4.38. The speed of convergence for this procedure is high and figure 4.8 shows a plot of estimated density for the parameters $d = 0.7$ and $\epsilon = 0.0001$. As it can be seen, the invariant density is characterized by a sharp peak near the origin corresponding to the many iterations spent by the map in the vicinity of the fixed point zero. In the absence of an exact analytical expression for $\rho(x)$, it is not possible to derive exact expressions for the source statistics. Instead a qualitative evaluation of the traffic source characteristics are given in order to be related to the fractal properties discussed earlier. As discussed earlier, actual data sets indicate heavy-tailed interarrival densities.

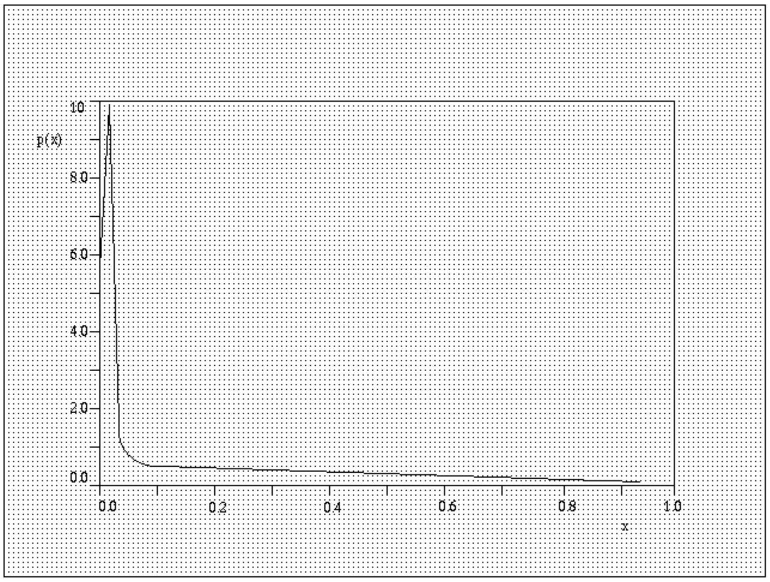

Figure 4.8: Invariant density of the intermittency map.

Given the map starts an idle period at a point x_0, the duration of the idle period τ is approximately given by ([29])

$$\tau = \frac{1}{\sqrt{c\epsilon}}[\arctan \frac{d\sqrt{c}}{\sqrt{\epsilon}} - \arctan \frac{x_0\sqrt{c}}{\sqrt{\epsilon}}] \tag{4.39}$$

Therefore the maximum interarrival time k that can be generated with a given set of parameters is given by

$$\max k = \frac{\pi}{2\sqrt{c\epsilon}} \tag{4.40}$$

and then from theorem 4.10 the sojourn times in the OFF state are distributed as:

$$P(k) \propto k^{-2}, \qquad E(k) \propto \epsilon^{-1/2} \tag{4.41}$$

where the expected duration is proportional to $1/\sqrt{\epsilon}$. Thus the passive period distribution has a tail that falls of as a power law and the expected value can be made arbitrarily large by reducing ϵ. The ϵ is, in deed, the "throw-away" parameter in the sense that it is determined by cut-off parameters such as the maximum idle period which are difficult to quantify in practice. It turns out that the regular operational measurements that are typically available in practice such as long term average rates or utilizations sensitively depend on ϵ and hence ill posed. As it can be seen in figure 4.8, the invariant density falls off quickly to an almost constant value after a peak near the origin. The re-injection probability into the active state is then approximately uniform and indicates that the active periods are geometrically distributed noting the fact that the map is linear in this region.

Now, let us discuss the issue of power spectra. As it was discussed earlier, the power spectra of the real traffic show a sharp divergence near the origin, which is referred to as $1/f$ noise. Figure 4.14 shows the plot of power spectra of traffic processes generated by the studied maps. The intermittency map shows a sharp divergence near the origin and the effect of nonzero ϵ is to limit the divergence range from below with a cutoff frequency. In contrast, the power spectra generated by piecewise linear maps are flat near the origin which is the characteristic of short range dependent models. Note that a traffic stream consisting of extended bursts, e.g., the long file transfers, can also show $1/f$ noise because it is a result of extended periods of either activity or inactivity. Given the almost geometric distribution of the active periods in the intermittency map, the extended idle periods are the cause of $1/f$ noise illustrated in figure 4.9.

The following result [29] supported by numerical simulations relates the index of the intermittency map to the exponent in $1/f$ noise.

Result 1 [29]. The process y_n generated by the single intermittency map of relation 4.38 exhibits $1/f$ noise with $H = (3m-4)(2m-2)$. Again choosing m in the range $(3/2, 2)$ generates processes which match $1/f$ noise behavior observed in actual traffic with H in the range $(1/2, 1)$.

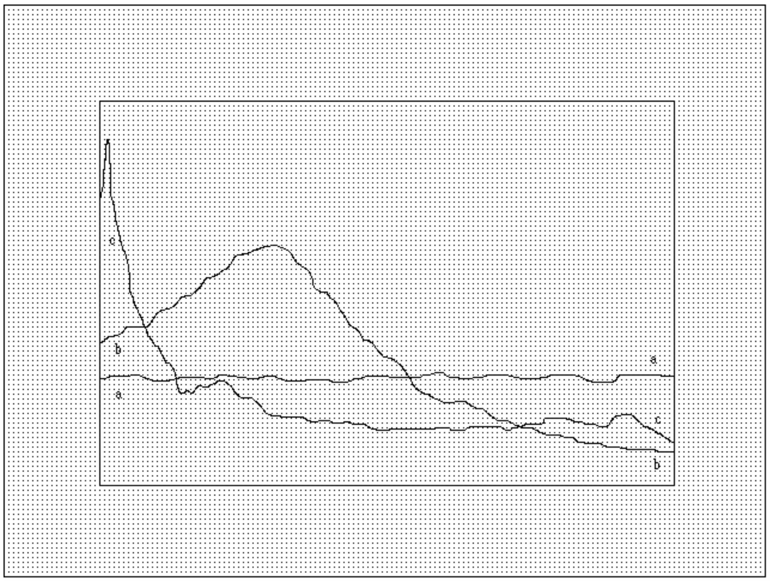

Figure 4.9: Low frequency power spectra of arrivals y_n for the maps: (a) Bernoulli shift, (b) Liebovitch map, (c) Single Intermittency map.

Finally a brief discussion about the queuing behavior is given. The following result was made by Erramilli et al [5] based on some heuristic analysis and numerical evidence.

Result 2 [5]. For the queuing system of relation 4.13 and relation 4.14, driven by the single intermittency map the 2-D invariant density is given by

$$\rho(x,l) = A_0 \Psi(x) P(l), \qquad l > 2 \tag{4.42}$$

where A_0 is the normalization constant. Hence, the queue distribution is given by a geometric decay in the form of

$$P(l) \propto \gamma^l, \qquad l > 2 \tag{4.43}$$

This result is analogous to the one obtained from GI/M/. theory. The parameter γ is referred to as generalized occupancy and is the solution to the following functional equation.

$$\gamma = [\int_d^1 \Psi(x)\, dx]/[1 - \int_d^1 \Psi(x)\, dx] \tag{4.44}$$

Note that the form of the queue length distribution of relation 4.43 is a consequence of using a linear function for the active segment and using nonlinear segments for the ON state can lead to a very different behavior.

Example 4.5: Double intermittency map. This map is obtained by replacing the active region in the single intermittency map with a nonlinear function satisfying relation 4.35 near the fixed point $x = 1$. Hence the describing equations of the map are as follow,

$$x_{n+1} = \begin{cases} \epsilon_1 + x_n + c_1 x_n^m & : \quad (0 \le x_n < d) \\ -\epsilon_2 + x_n - c_2(1 - x_n)^m & : \quad (d \le x_n \le 1) \end{cases} \tag{4.45}$$

$$\begin{aligned} c_1 &= [1 - \epsilon_1 - d]/d^m \\ c_2 &= -[\epsilon_2 - d]/[(1-d)^m] \end{aligned} \tag{4.46}$$

Figure 4.10 shows the double intermittency map. In this map a heavy-tailed ON-OFF behavior as $P(K > k) \sim \gamma(\hat{x}) k^{-\frac{1}{m-1}}, k \to \infty$ is observable and provides a connection between the current approach and the fractional Brownian motion model proposed in [25] to model self-similar traffic. It has been shown that aggregating a large number of heavy-tailed ON-OFF sources in limit leads to the fractional Brownian motion with the index $H = (3m - 4)(2m - 2)$, $3/2 < m < 2$. Thus the double intermittency map may be used as an efficient generator of fractional Brownian motion, an application currently under investigation [25]. This is important to note that the heavy-tailed queuing behavior

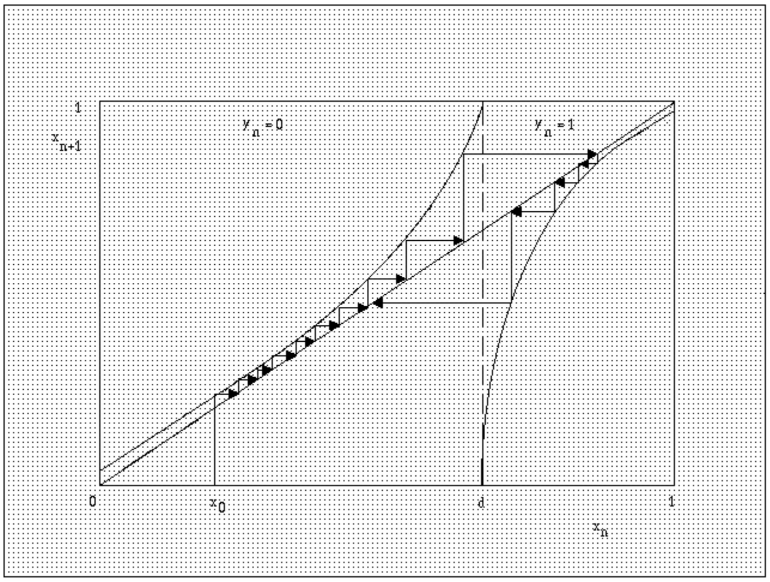

Figure 4.10: Double intermittency map matches ON-OFF behavior observed in measurements.

characteristic of self-similar traffic sources has obvious engineering significance including buffer allocation.

Chapter 5

Neural Network Modeling of Self-Similar Packet Traffic

5.1 introduction

In this section, a new approach is introduced which is able to deal with the fractal properties of the aggregated traffic. This approach provides a very elegant solution for self-similar traffic modeling and has the advantage of simplicity compare to the previously investigated approaches namely stochastic and deterministic nonlinear chaotic map models. It is motivated by the desire of having a relatively simple model of the complex packet traffic generation process. As oppose to the above mentioned modeling approaches, it does not introduce a parameter that describes the fractal nature of traffic and hence need not cope with the complexity of estimating Hurst parameters or fractal dimensions. The approach simply takes advantage of using a fixed structure nonlinear system which is able to predict either the number of packets generated by a traffic source or the number of arrived packets in a buffer after being trained by accessing to a number of samples of the generated or arrival pattern. [1]

The number of samples required for the training procedure in general depends on the complexity of the source and network dynamics. The usage of neural networks seems to provide a very simple approach for the task of modeling as it works based on indirect learning of the source or network dynamics. This is accomplished by using the information available in a number of samples.

The following two sections introduce a novel approach for modeling bursty traffic relying on neural networks as the modeling tool. The approach includes modeling of individual ON-OFF sources as well as aggregated traffic patterns at the gateway level.

[1] Please refer to chapter 2 for a complete description.

5.2 Modeling Individual Source Traffic Patterns

A fixed structure perceptron neural network is used for the task of modeling in this section. The network consists of an input layer with up to eight neurons, three or four hidden layers with twenty neurons in each layer, and an output layer with one neuron. The inputs of the network are eight consecutive samples of the traffic pattern and the output of the network is the ninth sample which is supposed to be predicted. Based on the richness of the dynamic, it might be possible to reduce either the number of the neurons in the input layer or the number of hidden layers but as the standard structure, the above mentioned structure is used unless otherwise is stated. Figure 5.1 shows the structure of the model.

In the following, three different approaches based on the type of the input samples

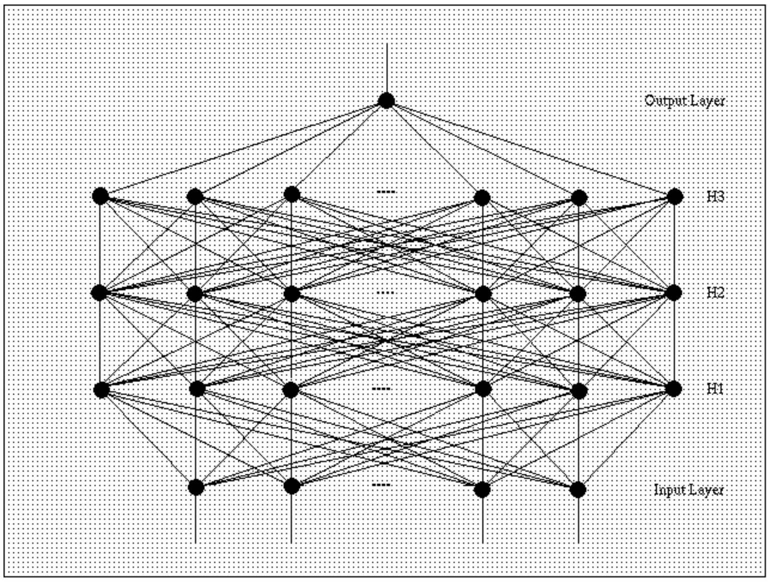

Figure 5.1: Fixed structure neural network with 8 neurons in the input layer and 10 neurons in each of the hidden layers used for the traffic modeling task.

used for training of the neural network are introduced.

- The first method makes a direct use of the available traffic samples. In order not to deal with very large numbers, a sample with value one is inserted to the neural network when the source is active and a sample with value zero is inserted to the neural network when the source is passive. This, in deed, is the normalized version of the peak packet generation rate divided by the peak rate. The method suffers from a major drawback, though. Since the samples provided for the network are discrete values equal to either zero or one, the network learning speed is very low. In fact, having a continuously distributed sample spectrum over the interval $[0, 1]$ leads to have a quite faster learning procedure.

- The second method can be used in case of generating artificial traffic patterns by chaotic maps. The approach simply accomplishes the task of modeling by inserting a level of indirection, i.e., the neural network concentrates on predicting consecutive samples of the chaotic map instead. By modeling the chaotic map, and generating its samples , it would be very easy to generate the same artificial traffic pattern using the same threshold value as far as the neural network is able to follow the corresponding chaotic map. This, in deed, is a combination of the approaches introduced in chapter 2 as neural network modeling of chaotic maps, and chapter 4 as chaotic modeling of bursty traffic.

- The third method provides a sophisticated and an elegant learning approach for well-behaved sources. I define a well-behaved source as a source that does not generate more than a specified number of packets in a time frame, i.e., there is an upper limit on the number of packets generated by the source. The most significant point about this approach is that it uses the real traffic samples where the samples have been arranged to create a continuous range of numbers distributed in $[0, 1]$ interval. Suppose that the source generates no more than a specified number of traffic packets, say P in a period of time T. Then, considering an origin for the time, the cumulative distribution function of the traffic pattern for the period T is defined as the number of packets generated since the beginning of the time divided by the maximum number of packets, i.e., p/P. Obviously, this is a monolithic increasing function starting at zero and ending at one. [2] The samples of this functions can be used to provide the desired sample set. At the end of the period, the desired output is compared with the network output and if the value of error has not entered the acceptable bound the training procedure is repeated. Figure 5.2 shows a typical cumulative distribution

[2] In some cases, the source may generate a number of packets less than the maximum number. That leads to have a monolithic function ending at a value less than one.

function.

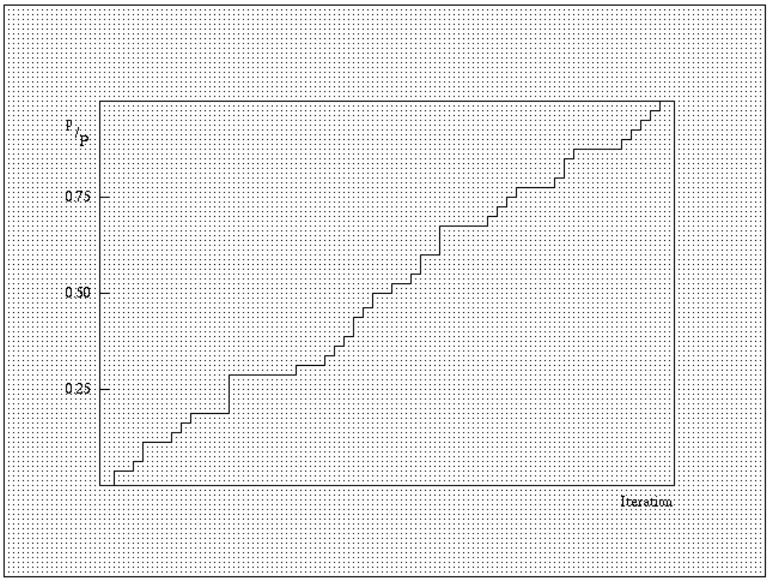

Figure 5.2: A typical cumulative distribution function used for the third method training procedure.

Relying on the above three mentioned training algorithms, the fixed structure neural network is used to model a number of artificially generated traffic patterns by chaotic maps, namely single and double intermittency maps as discussed previously. The use of artificial traffic patterns provides the possibility of being able to compare the results obtained from all three approaches.

Figures 5.3, 5.4, 5.5, and 5.6 show the modeling results in the case of single and double intermittency maps for initial conditions $x_0 = 0.1$ and $x_0 = 0.3$. Comparing all three approaches together, it seems that the second approach provides the best results in terms of tracking. Comparing the first and third approaches, it is easy to observe the third approach provides more reliable results as it is able to follow the traffic in a longer period of time. In general, and as it can be seen from the figures, the familiar ON-OFF follow-up learning pattern is observed, i.e., the neural net learns to follow the traffic pattern after approximately 700'000 and 1'000'000 million iterations in case of single and double intermittency maps respectively and is able to stay within the acceptable error bound for the next 60 and 40 samples. The network then goes out of sync and needs to be trained again in order to be able to follow the pattern prpoerly.

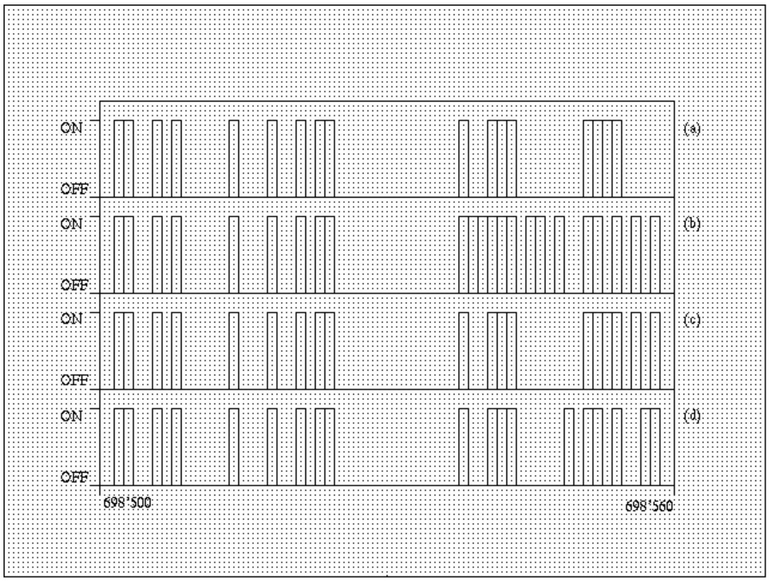

Figure 5.3: Results of modeling the ON-OFF traffic pattern generated by single intermittency map for the initial condition $x_0 = 0.1$. The horizontal axis displays the discrete time while the vertical axis displays the normalized version of packet generation at a peak rate. Results are shown for (a) the artificial traffic pattern, (b) the first modeling method, (c) the second modeling method, and (d) the third modeling method.

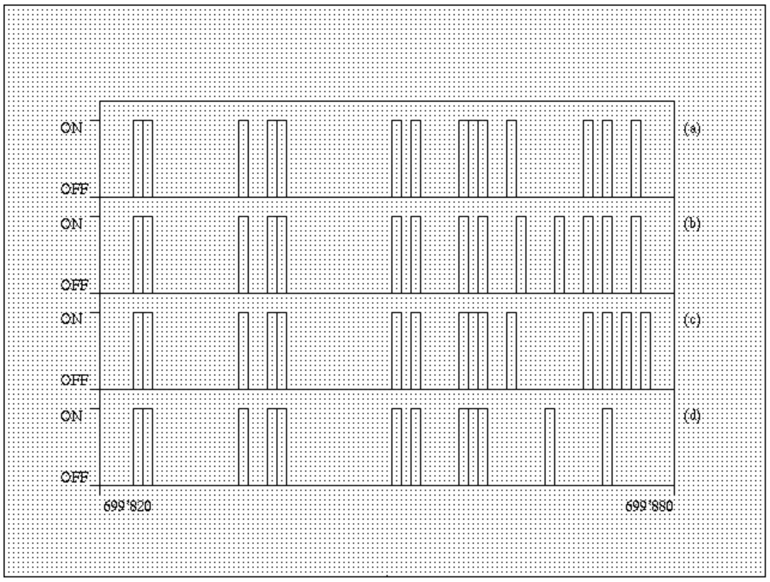

Figure 5.4: Results of modeling the ON-OFF traffic pattern generated by single intermittency map for the initial condition $x_0 = 0.3$. The horizontal axis displays the discrete time while the vertical axis displays the normalized version of packet generation at a peak rate. Results are shown for (a) the artificial traffic pattern, (b) the first modeling method, (c) the second modeling method, and (d) the third modeling method.

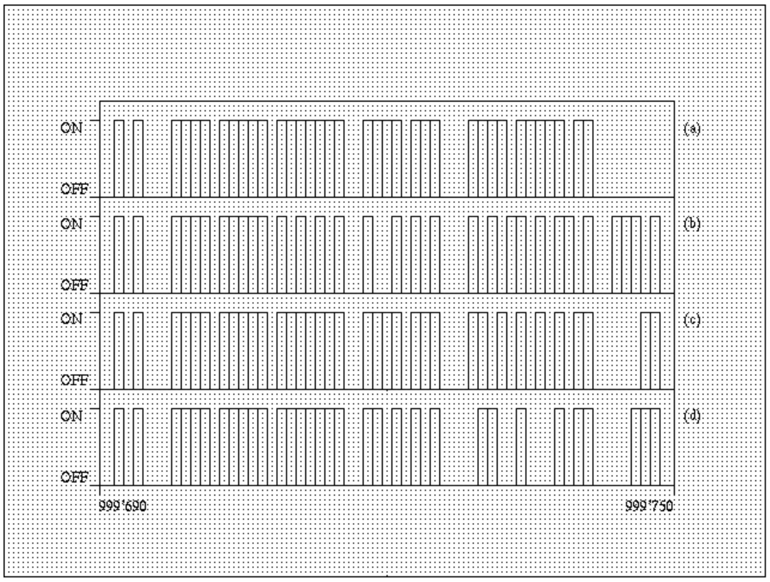

Figure 5.5: Results of modeling the ON-OFF traffic pattern generated by double intermittency map for the initial condition $x_0 = 0.1$. The horizontal axis displays the discrete time while the vertical axis displays the normalized version of packet generation at a peak rate. Results are shown for (a) the artificial traffic pattern, (b) the first modeling method, (c) the second modeling method, and (d) the third modeling method.

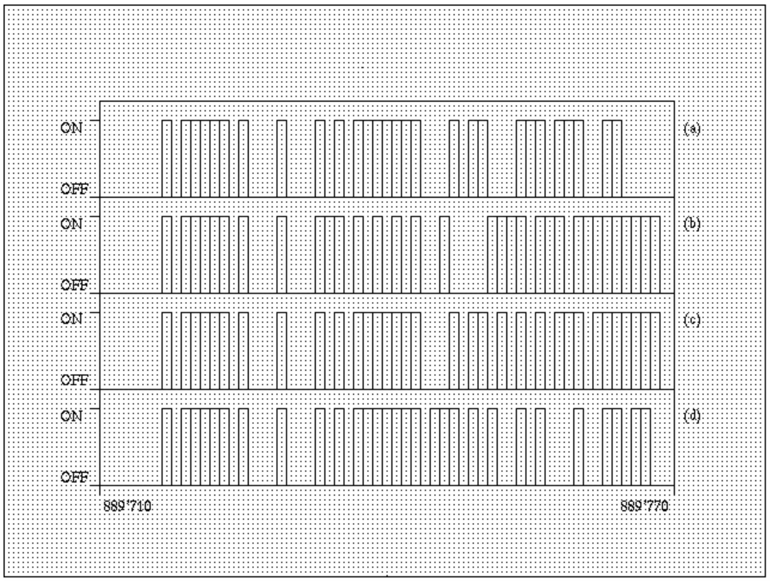

Figure 5.6: Results of modeling the ON-OFF traffic pattern generated by double intermittency map for the initial condition $x_0 = 0.3$. The horizontal axis displays the discrete time while the vertical axis displays the normalized version of packet generation at a peak rate. Results are shown for (a) the artificial traffic pattern, (b) the first modeling method, (c) the second modeling method, and (d) the third modeling method.

5.3 Modeling Aggregated Bursty Traffic at the Gateway Level

The following section illustrates the application of neural networks in modeling an aggregated level of bursty traffic. In order to be able to access an aggregated traffic pattern, a system consisting of 100 individual sources and a gateway as indicated in figure 5.7 is considered. This might be realized as an example of a real network with a number of work station nodes which send their packets to the network gateway. The traffic pattern might include a variety of different packets such as telnet, rlogin, ftp, mail, etc. The arrived packets are stored in a relatively large size buffer before being forwarded to the corresponding destinations. In order to be able to simulate the real network, each individual source is replaced by an artificial traffic generator following an ON-OFF pattern. The aggregated traffic, hence, can be considered to be self-similar. It is important to note that the objective here is merely predicting the traffic pattern arrived at the gateway.

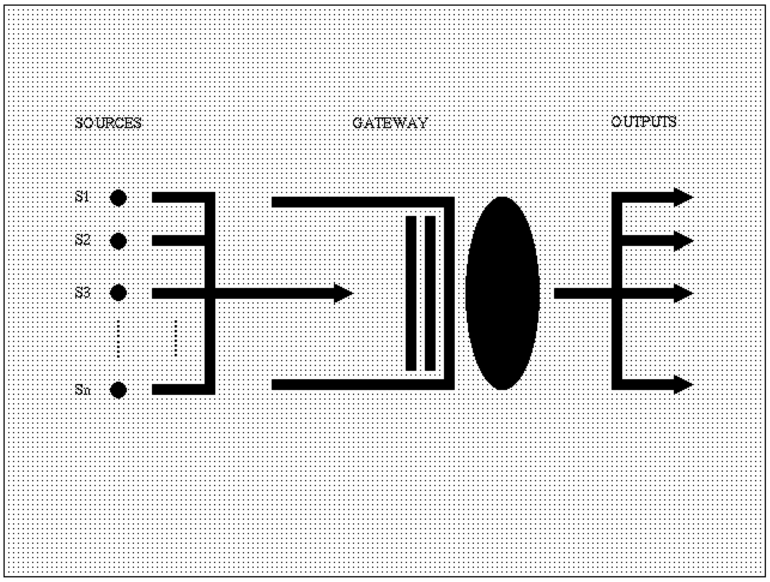

Figure 5.7: A sample network used to demonstrate the modeling power of neural networks for modeling aggregated level bursty traffic.

Same as before, a fixed structure perceptron neural network is used for the task of modeling. The network consists of an input layer with eight neurons, three hidden layers with twenty neurons in each layer, and an output layer with one neuron. This is the same structure as indicated in figure 5.1. The inputs of the network are eight consecutive samples of the traffic pattern and the output of the network is the ninth sample which is supposed to be predicted.

The traffic pattern of each source is obtained from the double intermittency map and is distinguished from other sources by assigning a different threshold value to the corresponding map. Figure 5.8 shows the result of neural network modeling task. Again, the familiar tracking period followed by a divergent behavior is observed. The only difference is that self-similarity increases the speed of convergence at the aggregated level. In case of figure 5.8 the neural network learning algorithm converges after 280'000 iterations approximately and is able to follow the real pattern for the next 55 arrivals.

Knowing that a statistically self-similar traffic pattern exhibits a fractal-like behavior

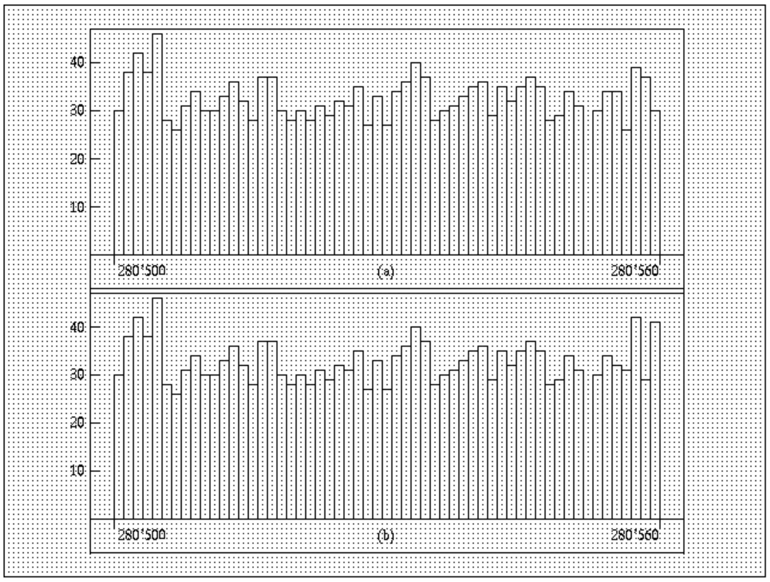

Figure 5.8: Neural network aggregated gateway level traffic modeling results.

in the sense that aggregating streams of such traffic pattern typically intensifies burstiness instead of smoothing it and reminding its serious impacts on design, control, and analysis of high speed packet based networks, the observed result is a very interesting one as it shows self-similarity provides an extra source of information that can be interpreted as some kind of correlation among the generated traffic patterns.

The conclusion is that the simple nonlinear dynamic of neural networks is able to implicitly capture self-similarity and hence neural networks may be viewed as suitable generators of self-similar traffic.

Chapter 6

A Study of Loss and Latency in a Multiplexed Queuing System: An Application of the Modeling Scheme

6.1 Introduction

Reducing packet loss in queuing systems is one of the most important issues in the design of traffic control algorithms. Reducing packet loss is equivalent to improving efficiency as it is usually considered as the performance evaluation tool. For the systems consisting of more than one source, there is another major issue worth considering known as fairness. Fairness basically means that every single source should be able to take advantage of a fair portion of the shared available resources such as buffer space or bandwidth. The combination of buffer management and scheduling algorithms specifies the fairness and the efficiency of a multiple source queuing system. It is important to note that there is a trade off between the fairness and efficiency issues as quality upgrade in one leads to the quality degradation in another. For the study of a multiple source queuing system, there are two independent major factors that have to be investigated. These are buffer management and scheduling. While scheduling specifies the server bandwidth sharing algorithm, buffer management deals with sharing the available buffer space among present sources.

In this study, two different scheduling algorithms are considered. These are namely Fixed Time Division Multiplexing (FTDM) and Statistical Time Division Multiplexing (STDM). In FTDM each source takes advantage of a fair portion of the server bandwidth [1] and there is no bandwidth sharing, while in STDM the unused portion of the bandwidth assigned to each source might be used to transmit packets generated by other sources. [2]

[1] Server bandwidth is also known as service rate.

[2] FTDM is used for ATM switching systems with a number of Virtual Paths while STDM is used in ATM queuing systems with a number of Virtual Channels.

There are a number of different buffer management algorithms studied in literature. [3] The first and probably the simplest method is complete sharing in which the buffer space is shared among all the existing sources without enforcing any capacity allocation mechanism. This method introduces an unfair consumption of the buffer space by greedy sources while providing the lowest loss rates. The second method is called complete partitioning in which the available capacity of the buffer is shared equally among the existing sources. This method has the best fairness characteristic while it greatly suffers from efficiency degradation by introducing the highest loss rate. In presence of a specified scheduling algorithm, a threshold buffer management algorithm is introduced as the third alternative solution. It might be considered as a solution between the above explained two extremes. In a threshold method, each source has its own fixed portion of the buffer space which can only be used for buffering packets generated by that specific source. There is also an additional shared portion of the buffer which is completely shared among the existing sources. This method is also called partial sharing.

A dynamic buffer management algorithm is classified under the threshold methods with the ability to adjust the buffer size of each source dynamically. It should be able to offer an acceptable solution considering both fairness and efficiency issues, i.e., the algorithm should assign a fair portion of the buffer space to each source with the ability to dynamically adjust the buffer space partitions.

The first application of this chapter is, in fact, considered as a dynamic buffer management algorithm relying on the modeling power of neural networks to adjust the portion of the buffer space assigned to each source dynamically. Relying on the prediction power of neural networks, it should be able to perform as good as if not better than the studied threshold algorithms.

Estimating and reducing packet latency is another major design issue in computer communication networks. There are a number of factors that introduce delay in network services. Different delay types may be classified under processing, propagation, multiplexing, and queuing categories. The main objective of packet scheduling methods is then to come up with a solution for predicting and reducing delay while efficiently utilizing network

[3] Please refer to [51] through [55] for more information.

resources. The performance of a packet scheduling method is then specified by the degree to which it satisfies the following requirements:

- Efficiency: Efficiently utilizing network resources, the algorithm should be able to provide bounded delays for a large number of packets.

- Flexibility: The algorithm be flexible enough to satisfy a diverse set of delay requirements.
- Predictability: The scheduling algorithm should be able to introduce a reliable method of estimating the latency for the transmitted/arrived packets.
- Complexity: The complexity of the scheduling algorithm must be kept small as it needs to perform at the speed of transmission links.

The second application of this chapter introduces a neural based approach for predicting the queuing delay as the dominant delay factor observed in a finite buffer used as an interface for transmitting a number of packets generated by a number of sources in a multiple source system. Hence, it might be considered as a part of packet scheduling algorithm that is able to estimate packet latency.

The current chapter outline is as follows. First the multiple source system used for the application task is described, then the applications of the neural network prediction scheme, namely loss prevention, and packet latency prediction, are discussed in detail. The queuing system consists of a number of sources with each source representing one or a number of packet generators. While loss prevention is approached by relying on a dynamic buffer management algorithm using the neural network prediction scheme, packet latency prediction is addressed by counting on the prediction power of neural networks directly.

6.2 The Multiple Source System

This section describes the multiple source system as the infrastructure of the following applications. The multiple source system consists of a number of sources with each source assumed to be generating packets with a pattern following the double intermittency map packet generation model described in previous chapters. By using different initial conditions and/or different threshold values, different traffic patterns are obtained for different sources.

This section provides a brief discussion viewing each source and the corresponding buffer as a separate queuing system. This is an appearance of complete partitioning buffer management scheme employing FTDM scheduling service at the output of the queue. The discussion may also be directly applied to the case of complete sharing buffer management scheme employing FTDM scheduling service at the output of the queue or may be slightly changed to apply to the case of complete/partial partitioning buffer management scheme employing STDM scheduling service at the output of the queue. [4]

There is a finite capacity buffer corresponding to each source which keeps the generated packets before they get transmitted. The occupancy of each buffer is determined by the flow of cells from the corresponding source and the rate at which the cells are serviced. In this model a queue is identified by its buffer capacity C_{max}, and its server capacity O_{max}. In each queue, the generation rate is compared with the service rate to determine whether the size of the queue is increasing or decreasing as well as whether the queue is losing cells.

Here the challenge is the dynamic assignment of the buffer space such that the probability of loss is minimized. It is also desirable to be able to predict the service time of each packet. Using the following notation,

- $I(i,k)$: The input rate of the i-th buffer at time k.
- $O(i,k)$: The output rate of the i-th buffer at time k.
- $Q(i,k)$: The queuing rate of the i-th buffer at time k.
- $L(i,k)$: The loss rate of the i-th buffer at time k.
- $C(i,k)$: The queue size of the i-th buffer at time k.

the state of the queue for each buffer is specified by

$$I(i,k) = O(i,k) + Q(i,k) + L(i,k) \tag{6.1}$$

[4] The model may be considered as the so-called burst scale queuing component of an ATM queuing system with a number of Virtual Channels (VCs) with each VC belonging to a traffic source. Please refer to [19] for a full discussion.

at any instant of time as shown in figure 6.1. Note that besides $Q(i,k)$ values which could be positive or negative, all the other values are always positive. Originally, all of the queues

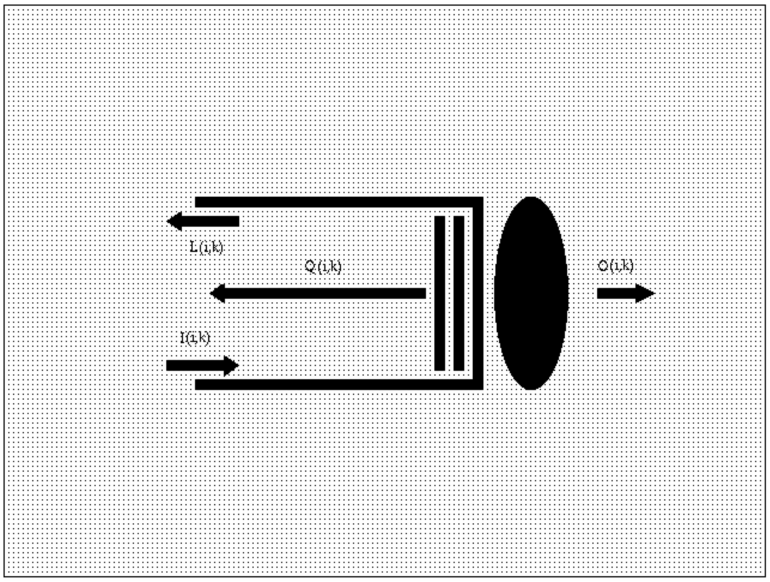

Figure 6.1: The queuing diagram of the i-th source at time k.

are empty. A queue begins to form when the buffer input rate exceeds the service rate. Hence, the queue rate $Q(i,k)$ and the loss rate $L(i,k)$ remain zero as far as the input rate is less than or equal the service rate, i.e.,

$$O(i,k) = \begin{cases} I(i,k) & : \quad I(i,k) < O_{max} \\ O_{max} & : \quad I(i,k) = O_{max} \end{cases} \tag{6.2}$$

The queue size $C(i,k)$ begins to increase as soon as the input rate exceeds the service rate O_{max}. While the queue is not empty, the output rate is always equal to the queue server capacity and the total queuing rate is the difference between the input rate and queue server capacity. The loss rate is zero at this stage.

The queue size keeps increasing and finally becomes full if the input rate remains higher than the queue server capacity. In that situation, the queuing rate is zero and the excess input rate is the cell loss rate as,

$$L(i,k) = I(i,k) - O(i,k) \tag{6.3}$$

with $O(i,k) = O_{max}$. The effect of a change in the input rate is not immediately appeared if there are packets in the queue waiting to be transmitted. It is the queuing rate which changes according to

$$Q(i,k+1) = Q(i,k) + I(i,k+1) - I(i,k) \tag{6.4}$$

The queue size begins to decrease in size when the input rate becomes less than the server capacity, i.e., $I(i,k) < O_{max}$, and the queuing rate goes below zero as the result, i.e., $Q(i,k) < 0$. The queue becomes empty if this situation lasts. The output rate is obtained from the following equation,

$$O(i,k) = \begin{cases} O_{max} & : \quad C(i,k) \geq 0 \\ I(i,k) & : \quad C(i,k) = 0 \end{cases} \tag{6.5}$$

After providing a brief queuing analysis for individual queues, now it is time to take a look at the system from a high level point of view. In the following two sections, it is assumed that a number of sources are sharing the total available buffer space. The traffic pattern of each source includes the packets generated by a number of ON-OFF models. An ON-OFF source model is generating traffic at a peak rate when it is active and it becomes active as soon as the state variable of the describing chaotic map goes beyond the threshold value. The source becomes passive as soon as the state variable goes below the threshold. The double intermittency map is chosen to be the chaotic map used for the packet generation as it generates a self-similar traffic pattern. [5] Each source traffic pattern is separated from the others by choosing a different initial condition.

6.3 Loss Prevention

Consider a system which consists of a number of sources sharing the space available in a central buffer and generating packets following an ON-OFF source model. Figure 6.2 shows the structure of a multiple source queuing system. In order to show the performance of the modeling approach, four different buffer management scenarios are compared together in presence of FTDM and STDM scheduling algorithms.

[5] Please see chapter 4 for details.

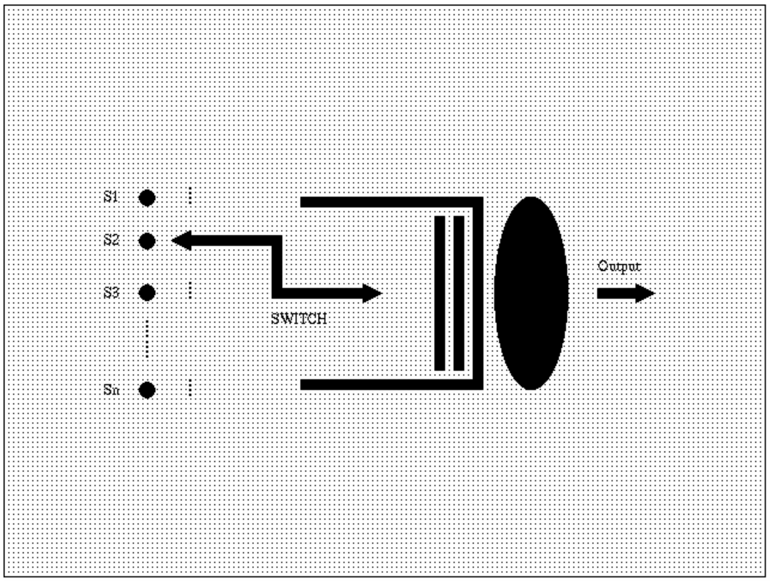

Figure 6.2: The structure of a multiple source queuing system.

- The first scenario happens when complete sharing mechanism is enforced, i.e., there is only one queue for all of the sources. This is a simple queuing mechanism in which all of the generated packets are directly sent to the central buffer and wait there until getting transmitted. This method has the advantage that the buffer is shared among all the sources, hence the maximum efficiency for the buffer space is obtained. The drawback of the method is that the space may not be used fairly. Hence, a source with a high output rate is able to consume a big portion of the buffer space and cause the queue to overflow. It is worth mentioning that the results of FTDM and STDM are the same as there is only one buffer in the system.

- The second method is a simple implementation of complete partitioning scheme in presence of FTDM and STDM in which the capacity of the central buffer is distributed equally among the sources. The most important characteristic of the method is that the buffer space is distributed fairly. FTDM is suffering from a possible low efficiency rate compare to STDM, i.e., sources with lower generation rate may not use the whole portion of the bandwidth assigned to them while sources with higher generation rate are having packets ready to be transmitted. This is not happening when STDM is employed as the unused portion of the server bandwidth is used if there is any packet ready to be transmitted.

- The third method is a simple implementation of partial sharing scheme that has three equal portions for the three sources with an additional shared portion that can be shared among all the sources. This is also called partial sharing.

- The fourth method is the dynamic assignment of the buffer space relying on the results obtained from the perceptron network prediction algorithm, i.e., adjusting the buffer space according to the packet generation pattern of each source. This is a generalization of the third method keeping the shared portion size fixed and adjusting the buffer space size of each source dynamically.

It is important to mention that for the last three methods, there is a separate queue for each source which holds the packets generated by that source. The difference between the third and the fourth scenario is that in the third scenario the buffer space assigned to each source is fixed and each source is able to send its generated packets either to its own buffer or the shared buffer if space is available while in the fourth scenario the portion of the buffer space assigned to the source with a higher packet generation rate is increased in case other sources are not generating enough packets to use their assigned share of the buffer space. This is a dynamic buffer management algorithm which has the potential to perform better than the existing buffer management algorithms as it relies on predicted future information.

In order to investigate the performance of the method, a triple source system is used. The traffic patterns of the first, second, and third source consist of an artificial traffic pattern generated by 30, 40, and 50 individual double intermittency map packet generators respectively. The traffic generated by each source is collected and sent to the corresponding buffer in a round robin manner. It is specially important to note that there is a slight difference among the number of packets generated by each source as the result of having a different number of ON-OFF packet generators per source. In order to evaluate the performance of different methods, the over all as well as per source loss probability of the system for different choices of buffer size with a fixed service rate are compared together. The buffer space can be shared among all of the sources or may be divided into equal portions for individual source usage. The server bandwidth may also be used according to FTDM or STDM scheduling mechanisms.

Figures 6.3 through 6.6 show the total and single source packet loss probability versus packet size diagram for the triple source queuing system in presence of FTDM and STDM scheduling algorithms. The single source is the source with the lowest generation rate to compare the fairness of different schemes. The simulation results have been obtained from an iterative algorithm with a total number of ten million iterations per choice of buffer size. Applying a continuous learning algorithm, the fixed structure neural network has been able to follow the traffic pattern within the specified error range between 20 and 30 times covering an average of fifty samples per time.

This is worth mentioning that the performance of different methods are very differ-

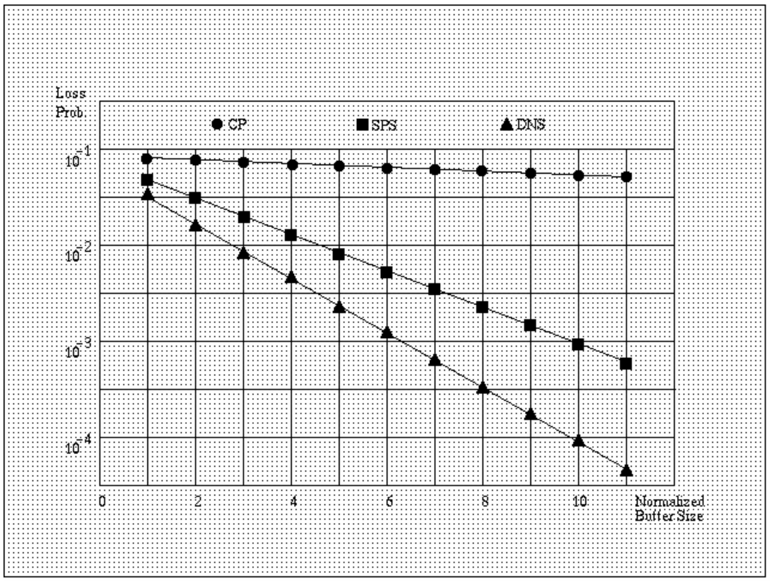

Figure 6.3: The total packet loss probability versus buffer size diagram for the triple source queuing system using complete partitioning (CP), static partial sharing (SPS), and dynamic neural sharing (DNS) in presence of FTDM scheduling algorithm.

ent as the result of applying different methods for traffic management of a heavily utilized system. Tables 6.1 through 6.4 summarize the results for different cases. It is clearly observed from the figures and the tables that for both FTDM and STDM using neural sharing scheme, the total loss rate compared to complete partitioning scheme as well as per source loss rate compared to complete sharing scheme are reduced. The results may be interpreted as a sign that the neural sharing scheme has come up with a solution in between the two extreme cases. Comparing the results for static partial sharing and neural dynamic sharing show the higher efficiency of the latter method. This is a significant improvement compare to the other three schemes.

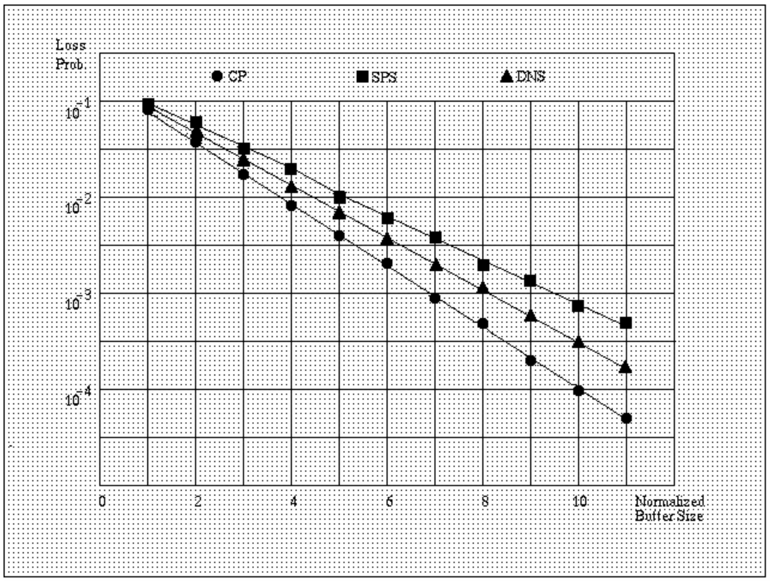

Figure 6.4: The single source packet loss probability versus buffer size diagram for the triple source queuing system using complete partitioning (CP), static partial sharing (SPS), and dynamic neural sharing (DNS) in presence of FTDM scheduling algorithm.

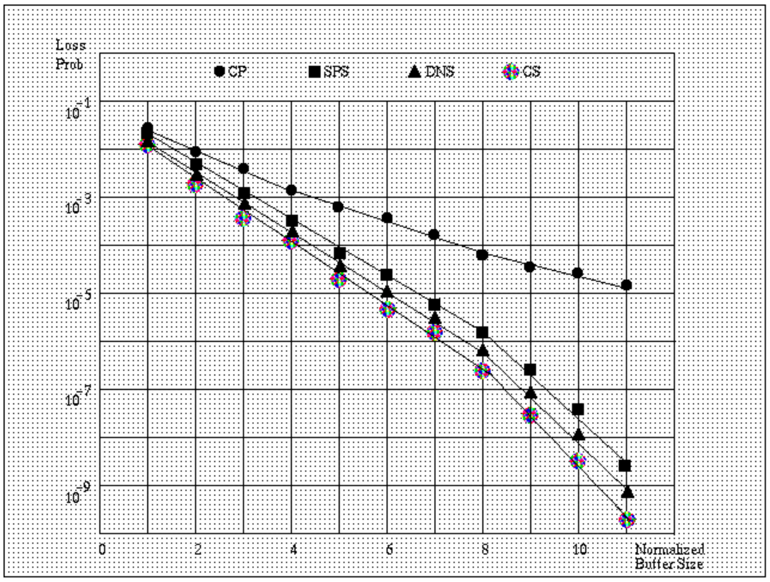

Figure 6.5: The total packet loss probability versus buffer size diagram for the triple source queuing system using complete partitioning (CP), static partial sharing (SPS), dynamic neural sharing (DNS), and complete sharing (CS) in presence of STDM scheduling algorithm.

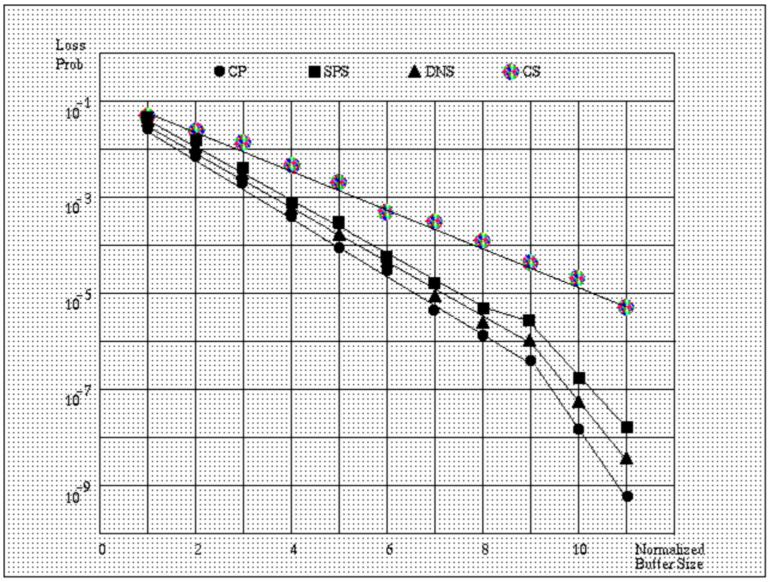

Figure 6.6: The single source packet loss probability versus buffer size diagram for the triple source queuing system using complete partitioning (CP), static partial sharing (SPS), dynamic neural sharing (DNS), and complete sharing (CS) in presence of STDM scheduling algorithm.

6.4 Packet Latency Estimation

In this section, the objective is to estimate the queuing delay of the packets generated by a number of traffic sources. This includes the time each packet spends in the queue before getting transmitted.

Again, consider the above triple source system sharing the buffer space of the central buffer following complete sharing scheme in presence of STDM scheduling scheme. For the case of packet latency estimation, the system load varies based on the value of service rate. The case has been chosen as the objective here is to determine the queuing delay of the generated packets. Supposing each packet carries a sequence number indicating

Buffer Size	CP	SPS	DNS
1	75378473.612	54914225.332	34914225.332
2	60689921.778	27958980.093	13958980.093
3	54305821.427	16698674.616	8576534.5123
4	51372262.818	10425478.992	5387542.1234
5	50131819.132	6749892.3056	2987364.5464
6	49526991.133	4412441.4656	1084565.8356
7	49255245.952	2942719.9642	834637.12234
8	49127329.703	1957200.4934	359347.74654
9	49058664.227	1334350.5099	234873.84956
10	49042049.109	923028.16957	91876.867263

Table 6.1: Total loss rates of complete partitioning (CP), static partial sharing (SPS), and dynamic neural sharing (DNS) buffer management schemes for different values of normalized buffer size in presence of FTDM scheduling algorithm. The numbers have been scaled as multiples of 10^{-9}.

Buffer Size	CP	SPS	DNS
1	96340229.048	100278284.02	98634818.723
2	45479959.816	52202567.328	49769278.243
3	21006347.153	28688742.339	2478984.3453
4	9447082.3529	16219350.050	1226525.0455
5	4470073.1877	9612488.8520	718951.92434
6	2079872.2546	5709319.8577	424686.32541
7	995001.51426	3450275.0697	255726.84655
8	466876.51958	2073420.1985	129477.63546
9	224865.93393	1304089.8296	684893.89456
10	105160.08544	832529.11289	319473.58930

Table 6.2: The most passive individual source loss rates of complete partitioning (CP), static partial sharing (SPS), and dynamic neural sharing (DNS) buffer management schemes for different values of normalized buffer size in presence of FTDM scheduling algorithm. The numbers have been scaled as multiples of 10^{-9}.

Buffer Size	CP	SPS	DNS	CS
1	22207551.433	15675391.040	13672845.258	10634994.258
2	9700136.1712	3733455.6678	2916742.0964	2331664.0964
3	4223896.4689	1005909.2285	854583.95892	551953.95729
4	1896600.3233	287436.89041	193735.64354	131071.11699
5	889360.03022	84652.809821	44893.567658	29872.055583
6	418606.70002	25171.393026	103967.98475	8230.2708137
7	204657.36078	6470.4656063	4167.4307543	2182.2065425
8	95144.171287	1917.3819851	865.49856558	328.31883366
9	54367.649641	895.23243764	99.479658436	34.263233637
10	32100.486624	31.512258787	11.498159349	3.4476532659

Table 6.3: Total loss rates of complete partitioning (CP), static partial sharing (SPS), dynamic neural sharing (DNS), and complete sharing (CS) buffer management schemes for different values of normalized buffer size in presence of STDM scheduling algorithm. The numbers have been scaled as multiples of 10^{-9}.

Buffer Size	CP	SPS	DNS	CS
1	42535754.843	47479197.017	44341454.652	71184457.380
2	9326704.8577	11553695.563	9942365.5625	29646347.308
2	9326704.8577	3096432.5116	2765457.0677	11746216.422
3	2207021.0677	887887.78245	708947.74566	4670319.9752
4	523721.70361	260234.48751	208123.47657	1937850.9118
5	119444.83561	78268.858814	55876.823679	759557.72958
6	32921.031207	19744.724449	9734.839460	309435.540656
7	8727.5883075	5581.5089305	4098.489654	124632.676795
8	1313.9145314	3063.5341416	1005.845673	57147.5016042
9	440.64532277	126.03415534	812.4856636	25280.3508996
10	170.48058589	10.503427678	56.86257354	9957.28637678

Table 6.4: The most passive individual source loss rates of complete partitioning (CP), static partial sharing (SPS), dynamic neural sharing (DNS), and complete sharing (CS) buffer management schemes for different values of normalized buffer size in presence of STDM scheduling algorithm. The numbers have been scaled as multiples of 10^{-9}.

the order it was sent to the central buffer, the objective is then to predict the average number of time units a packet spends in the queue before leaving the buffer. The task is approached by applying the neural network modeling scheme to predict the total number of generated packets. Knowing the buffer service rate, average latency can be calculated directly from the arrival rate of the buffer. [6] This, in deed, is a problem of predicting the state of a queue. Again the prediction tool is the fixed structure neural network that is supposed to learn the dynamic of the arrival pattern of the buffer.

The real and estimated average latency versus service time diagram for the triple source queuing system is shown in figure 6.7. The typical system consists of 120 sources generating traffic according to an ON-OFF pattern and sending the generated packets to a central buffer. The buffer size is assumed to be fixed, large enough to prevent having any loss. The average latency has been calculated over all of the time periods in which the neural network is able to follow the arrival pattern of the central buffer.

All of the numerical results are shown in table 6.5. As it can be observed from the

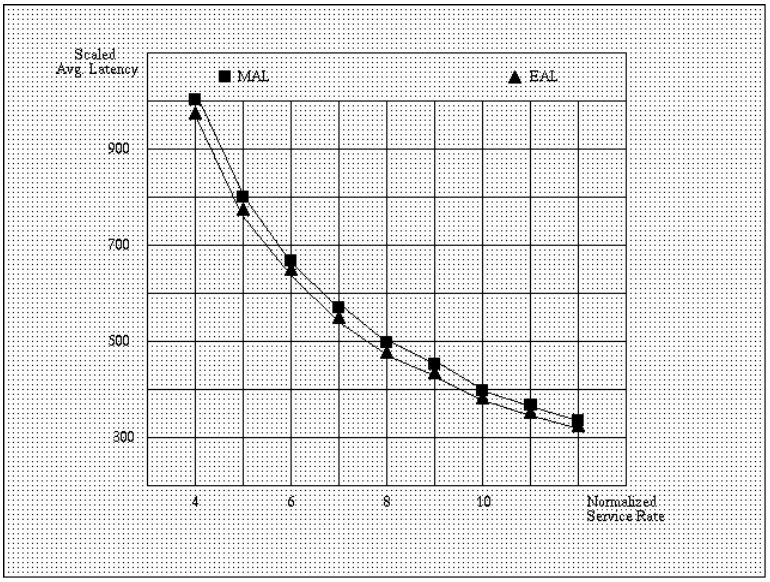

Figure 6.7: The estimated average latency (EAL), and measured average latency (MAL) of the packets versus service rate diagram for the triple source queuing system. The complete sharing buffer management and STDM scheduling algorithms have been employed.

figure 6.7 and, table 6.5 the estimation results are quite acceptable within the 3% error

[6] This is the queuing latency, which is the dominant latency factor in most of the cases.

Service Rate	EAL	MAL	EE
3	1943.87147	2002.86700	0.029456
4	1295.79382	1335.05957	0.029411
5	971.678529	1001.19972	0.029486
6	777.261471	800.867677	0.029476
7	647.631176	667.269205	0.029430
8	554.964118	571.847255	0.029524
9	485.629706	500.377443	0.029473
10	431.560294	444.644920	0.029427
11	388.334412	400.133652	0.029488
12	352.616471	363.456176	0.029824
13	322.565294	332.115360	0.028755
14	3.20676554	3.48024167	0.078580
15	0.46852956	0.57054254	0.178800
16	0.16794165	0.20360623	0.175166

Table 6.5: The estimated average latency (EAL), measured average latency (MAL), and the estimation error (EE) for different values of normalized service rate of the traffic pattern of multiple source system. The complete sharing buffer management and STDM scheduling algorithms have been employed.

range as far as the averaging period is long enough in order for the neural network to be able to follow the traffic pattern a number of times within the specified error bounds and as far as the buffer service rate does not exceed an existing threshold value. As a matter of fact, it is observed that the average packet latency drops very sharply with an order of ten or more choosing a value beyond the threshold value. As the result of having very small average latencies, the neural network latency estimation findings are not acceptable for the service rate values beyond the threshold value. The value of the threshold generally depends on the dynamics of the system and for the triple source system is the normalized value 13. Remind that for this complete sharing case, the modeling scheme relies on the combined dynamics of all of the sources to achieve the latency estimation as there is only one queue in the system. It is important to also note that the same qualitative approach may be used in case of having separate queues.

6.5 General Experimental Observations

In the following, I briefly mention some of the practical findings in the implementation of the individual source and aggregated level modeling algorithms as well as buffer management and latency prediction applications.

It is important to note that most of the findings here are the direct conclusion of dealing with complicated nonlinear dynamics. The first specific observation explained by the chaotic nature of traffic, is that although it is possible to reach a very small error at some steps during the learning phase, if the neural network error is studied for further samples, it is observed that this error begins to grow as time proceeds. The observation can be explained by reminding that the traffic patterns with self-similar characteristics have been generated by a chaotic map. [7] In this case, since the neural network attempts to model a chaotic system, it becomes chaotic itself. Hence, the small error may be considered as a small difference between two close initial conditions for the desired output and the network output and as a characteristic of a chaotic system, the error will begin to grow very soon which is nothing but the high sensitivity to the variations of initial conditions. As a matter of fact, this can be interpreted as a good sign for the network trained to model a chaotic map and it has become chaotic itself. One way of relieving the effect of having an error component which grows with time is to repeat the learning phase followed by the recalling phase periodically, otherwise the error exceeds the acceptable band and causes the network to create unreliable samples. In practice after the first learning phase with almost a few hundred thousand examples, the network is able to predict less than one hundred samples and then the learning procedure has to be repeated for the same number of examples as in the previous recalling phase and so on to have reliable results. This approach has been used to improve the loss performance of neural buffer management scheme.

Based on the same line of reasoning, all of the convergence results are affected strongly by the choice of initial conditions. It can be said that the initial values of the parameters play a crucial role in the convergence of the algorithm. It is even possible to have a divergent algorithm, if the initial values of the parameters are chosen unsuitably. As a practical result, it has been observed that setting the initial values of the parameters to $w_{ji}(0) = 0.01 \quad \forall i, j$ leads to have the best results. [8]

[7] See [3] and its corresponding references for a detailed discussion.
[8] This is very close to the reasoning line provided in chapter 3.

Finally, it has to be mentioned that the choice of initial conditions has another important impact on the performance of the application. The issue can be addressed by saying that in case of loss prevention application, it is very important for the algorithm to be able to predict the traffic pattern generated by all the sources. The very complicated and wild behavior of a self-similar traffic pattern generally leads to have an out-of-sync prediction power for different initial conditions, i.e., the acceptable error bounds are not reached after having the same number of iterations for all the sources. The only way of dealing with this problem is to choose the initial conditions suitably and find the time intervals that all the traffic patterns can be predicted by the neural network or to continue the training procedures for all of the sources until all of the neural networks are able to follow the corresponding traffic pattern.

It is also important to note that the learning algorithm is time consuming because of the rich dynamic of the traffic pattern that the neural network is trying to learn. In deed, it takes hundreds of thousands of samples for the neural network in order to be trained. The situation is even worse in case of measuring the loss performance as the procedure has to be repeated a few ten times.

Chapter 7

Conclusion, Future Research Issues, and Open Areas

This final short chapter highlights the topics covered in this research and introduces a number of potential aspects as the scope for some future work.

7.1 Accomplishment Survey

In this report, the applications of deterministic chaotic maps and perceptron neural networks to model traffic sources in packet based computer communication networks were investigated. The motivation was in part coming from the widespread application of computer communication networks in different fields of science, engineering and economy as a distributed computing environment or source of information and also from the recent measurement studies indicating the presence of significant statistical features in packet traffic belong to the fractal nature of the processes rather than their stochastic nature.

The report illustrated the required background about the tools being used, i.e., chaotic systems theory and perceptron neural networks, as well as the statistical features of the measured traffic understanding the packet traffic patterns are very irregular and bursty. It then outlined a potential performance analysis approach based on simple nonlinear modeling tools namely chaotic maps and neural networks to capture the complexity of the traffic significance of fractal properties. The report showed that the above mentioned simple nonlinear tools were able to capture several of the fractal properties observed in artificial and actual data. The most important conclusion of the report was that although chaotic maps provide a concise description of complex packet traffic phenomena, there are considerable analytical difficulties in their application such as the calculation of associated invariant densities.

On the other hand, the fact that neural networks rely on the information available in a number of samples made them more practical for the task of modeling as they did not need to cope with analytical complexities involved with the chaotic systems' approach. The report also investigated two specific applications, packet loss reduction and packet latency estimation in a typical multi-source computer communication system. Both applications relied on a neural based scheme. While the first application made use of the scheme to dynamically manage the available buffer space, the second application employed the scheme to estimate the packet latency average of the arrival pattern of the buffer. This could be interpreted as a very good sign to indicate the modeling power of the investigated tools.

The final conclusion was that chaotic maps and neural networks can provide practical tools to accurately model transient and hopefully steady state behavior of packet traffic generators despite encountering considerable analytical and numerical difficulties.

7.2 Open Areas and Research Issues

The remainder of this section discusses the scope for additional work and introduces a number of open areas as potential research topics.

- Multi-Dimensional Invariant Densities.

  Ultimately, the usefulness of chaotic maps approach is determined by the feasibility of solving for the multi-dimensional invariant densities. Thus, there is a scope for much fundamental work in the development of suitable analytical and numerical methods. As an interesting approach, the recently developed matrix geometric techniques as a generalization of the Frobenius-Perron operator may be mentioned which have an accelerated convergence to the solution of invariant densities [26].

- Investigating Other Chaotic Maps.

  As it was mentioned earlier, a key aspect of chaotic modeling approach is to use suitable chaotic maps such that the generated traffic patterns match those of measured actual traffic. Investigating additional maps besides piecewise linear maps, single and double intermittency maps, in order to find more realistic models of active periods is another open research area. I originally studied the traffic patterns generated by two suitable candidates namely logistic map described earlier by the state equation

$$x_{n+1} = 4Wx_n(1 - x_n)$$

and the Henon map with the describing equations

$$\begin{aligned} x_{n+1} &= 1 - Ax_n^2 + y_n \\ y_{n+1} &= Bx_n \end{aligned}$$

as an application of 2-D chaotic maps in modeling bursty traffic. Using multi-dimensional maps is specifically useful when dealing with modeling benchmarks of correlated dynamics.

- Other Source Source Interpretation.

  There is also a possibility of having other source interpretations which may be more suitable for some applications. For example, it is possible to link different packet generation patterns to cross different thresholds, or to use continuous time systems or directly generate inter-arrival times from dynamical systems.

- Multiplexing of Sources.

  Multiplexing of sources is another open issue which is worth discussing. So far, the output of a map is used to match a traffic pattern which may either correspond to a single source or a number of sources. An extension is to consider the multiplexing and splitting of traffic streams where each source is either modeled by a chaotic map or a neural network as I have done for the case of dynamic buffer management. This issue can, in fact, introduce a very elegant application in ATM networks supporting a number of virtual channels.

- Performance Analysis.

  Performance analysis is another aspect of further work in this area. It basically means the analysis of queuing systems in which the randomness in the arrival or service characteristics are modeled using chaotic maps. Additional work is required on the modeling and representation of systems of practical interest as well as investigation of numerical and analytical techniques to calculate performance measures of transient analysis. Histogram analysis can be used as a very good evaluation tool to show that real measured data and the generated data from chaotic models have the same statistical features.

- Artificial Load Generation.

  Artificial load generation is one of the other elegant applications of the chaotic maps and neural networks in this area. The approaches allow the generation of complex traffic patterns using relatively simple tools that can be incorporated into traffic streams. There is a scope to investigate suitable maps, networks and efficient software implementation. As it was mentioned, I have used some simple chaotic maps such as single and double intermittency maps to generate artificial traffic. Then the statistical features of generated traffic may be compared to those of real traffic in order to show that there are some match points for the generated traffic. This is another way of evaluating the performance of the method. It is also very useful in studying aggregated traffic patterns following self-similar structures. As a very good example, the case of self-similar gateway traffic studied in section 5.4 may be mentioned.

- Performance Improvement of Neural Algorithms.

  Improving the convergence speed of the neural network learning algorithms in case of modeling bursty traffic with self-similar characteristics is another open area worth paying attention. Knowing the gradient based learning algorithms are suffering from a performance degradation problem near the convergence point, it is desired to make use of alternative or hybrid approaches to increase the speed of convergence near the optimum point. This, in fact, is the application of optimal control as it has been the case for different perceptron networks.

- Other Structures of Neural Networks.

  Other structures of neural networks as discussed in chapter 2 seem to be valuable potential tools for predicting the behavior of the ON-OFF source models. This, in particular, includes Hopefield and Kohonen structures as alternative solutions for the task of modeling. As another alternative, the use of networks with feedback connections in their structure might lead to have a better performance although it makes the modeling procedure more complicated.

- Indirect Use of Neural Networks for Estimating Traffic Parameters.

  Neural networks are able to learn nonlinear dynamics as it was mentioned previously. As the conclusion, they might be used to estimate parameters such as fractal dimension in case of chaotic modeling, or Hurst parameter in case of stochastic modeling of self-similar traffic patterns. This is an indirect way of using the power of neural networks to make the promising approaches more practical. The method is specially useful for replacing the very complicated numerical algorithms with a simple repeatetive learning function.

- Packet Loss Control.

  Packet loss control is one of the most important issues of packet traffic networks. Reducing the loss rate has always been one of the most important objectives of traffic management as it has a great impact on different applications over the computer communication networks. As an example, a case of an ATM-based B-ISDN switch may be mentioned in which losing a number of packets might lead to have an unacceptable performance level. Although, there have been a number of different approaches introduced in literature using the prediction power of neural networks to dynamically manage aggregated level packet traffic can introduce a novel scheme with a higher level of performance.

- Packet Latency Estimation/Reduction.

  Finally, packet latency is another important issue of packet traffic networks. Again, reducing the latency has been one of the most important objectives of traffic management as it has a great impact on different applications over the computer communication networks. As an example, a case of a real-time application such as lecture on demand might be mentioned in which having a long latency always degrades the performance and in general is not acceptable. As another example, the case of accessing a web site might be mentioned in which having a long latency is not desirable at all, specially, in case of animation. The neural network modeling scheme can play an important role to predict the packet latency as well as reducing it by combing with the packet loss control scheme at the aggregated level as described in the prior topic.

Reference List

[1] A.Erramilli, R.P.Singh, P.Pruthi, "Chaotic Maps as Models of Packet Traffic.", ITC Vol. 14, PP. 329-338, 1994.

[2] W.E.Leland, W.Willinger, M.S.Taqqu, D.V.Willson, "Statistical Analysis and Stochastic Modeling of Self-Similar Datatraffic", ITC Vol. 14, PP. 319-328, 1994.

[3] A.Erramilli, J.Gordon, W.Willinger, "Applications of Fractals in Engineering for Realistic Traffic Processes.", ITC Vol. 14, PP. 35-44, 1994.

[4] W.E.Leland, W.Willinger, M.S.Taqqu, D.V.Willson, "On the Self-Similar Nature of Ethernet Traffic", IEEE/ACM Trans. on Networking, Vol. 2, No. 1, PP. 1-15, Feb. 1994.

[5] A.Erramilli, R.P.Singh, "An Application of Deterministic Chaotic Maps to Model Packet Traffic.", Bellcore Technical Memorandum, 1992.

[6] V.Ramaswami, "Traffic Performance Modeling for Packet Communication: Whence, Where, and Whither", Proc. of Australian Teletraffic Seminar, Nov. 1988.

[7] J.Beran, R.Sherman, M.S.Taqqu, W.Willinger, "Variable Bit Rate Video Traffic and Long Range Dependence", IEEE/ACM Trans. on Networking, Vol. 2, No. 3, Apr. 1994.

[8] W.Willinger, M.Taqqu, "Self-Similarity through High-Variability: Statistical Analysis of Ethernet LAN Traffic at the Source Level", IEEE/ACM Trans. on Networking, Vol. 5, No. 1, Feb. 1997.

[9] L.S.Liebovitch, T.I.Toth, "Ion Channel Kinetics: Random or Deterministic Process", Biophysical Journal, Vol. 57, PP. 317a-327a, 1990.

[10] K.M.Hellstern, P.Wirth, "Traffic Models for ISDN Data Users: Office Automation Application", Proc. ITC-13, Denmark, 1991.

[11] A.Erramilli, D.Gosby, W.Willinger, "Engineering for Realistic Traffic: A Fractal Analysis of Burstiness", Proc. of ITC Special Congress, India, 1993.

[12] H.J.Fowler, W.E.Leland, "Local Area Network Traffic Characteristics, with Implications for Broadband Network Congestion Management", IEEE JSAC, Vol. 9, No. 7, Sep. 1991.

[13] K.Sriram, W.Whitt, "Characterizing Superposition Arrival Processes in Packet Multiplexers for Voice and Data", IEEE JSAC, Vol. SAC-4, No. 6, Sep. 1986.

[14] H.Heffes, D.M.Lucantoni, "A Markov Modulated Characterization of Packetized Voice and Data Traffic and Related Statistical Multiplexer Performance.", IEEE JSAC, Vol. 9, No. 7, Sep. 1991.

[15] D.E.Duffy, W.Willinger, "Statistical Analysis of CCSN/SS7 Traffic Data from Working CCS Subnetworks", IEEE JSAC, 1994.

[16] V.A.Bolotin, "Modeling Call Holding Time Distributions for CCS Network Design and Performance Analysis", Preprint 1993.

[17] D.E.Smith, "On the Holding Times of Data Calls", Bellcore Technical Memorandum, 1986.

[18] B.Mandelbort, "The Fractal Geometry of Nature", Freeman, NY, 1983.

[19] J.M.Pitts, L.G.Cuthbert, M.Bocci, E.M.Scharf, "An Accelerated Simulation Technique for Modeling Burst Scale Queuing Behavior in ATM", ITC Vol. 14, PP. 777-786, 1994.

[20] E.Didelet, B.Dubuisson, D.Stern, "A Neural Technique Approach to Network Traffic Management", ITC Vol. 14, PP. 1321-1330, 1994.

[21] I.Norros, "Studies on a Model for Connectionless Traffic Based on Fractional Brownian Motion", ICOST24TD(92)041, 1992.

[22] G.E.P.Box, J.M.Jenkins, "Time Series Analysis: Forecasting and Control", SF CA, Holden day, 1976.

[23] -, "Self-Similarity in High Speed Packet Traffic: Analysis and Modeling of Ethernet Measurements", Statistical Science, 1994.

[24] B.Mandelbort, M.S.Taqqu, "Robust R/S Analysis of Long Run Serial Correlation", Proc. 42nd Edition ISI, PP. 69-99, 1979.

[25] B.Mandelbort, W. Van Ness, "Fractional Brownian Motions, Fractional Noises and Applications.", SIAM Rev., Vol. 10, PP. 422-437, 1968.

[26] F.Y.Hunt, W.Miller, "On the Approximation of Invariant Measures", J. of Statistical Physics, Vol. 66, 1992.

[27] A.Lasota, M.C.Mackey, "Chaos, Fractals, and Noise: Stochastic Aspects of Dynamics.", Second Edition, Springer-Verlag, 1994.

[28] T.Y.Li, J.A.Yorke, "Ergodic Maps on [0,1] and Nonlinear Pseudorandom Generators.", Nonlinear Analysis 2, PP. 473-481, 1978.

[29] H.G.Schuster, "Deterministic Chaos: An Introduction", Second Edition, VCH, 1988.

[30] H.Yousefizadeh, M.Shafiee, A.Ziloochian, "Chaotic Arrays Modeling with Neural Networks.", Proc. of the Iranian Conference of Electrical Eng., Vol.4, PP. 667- 679, May 1993.

[31] P.G.J.Lisboa, "Neural Networks Current Applications", Chapman & Hall Co., 1992.

[32] T.S.Parker, L.O.Chua, "Chaos: A Tutorial for Engineers", Proc. of IEEE, Vol. 75, No. 8, Aug. 1987.

[33] A.R.Vazquez, J.L.Heurtas, L.O.Chua, "Chaos from Switched Capacitor Circuits.", Proc. of IEEE, Vol. 75, No. 8, Aug. 1987.

[34] K.Hockett, P.Holmes, "Nonlinear Oscillators, Iterated Maps, Symbolic Dynamics, and Knotted Orbits." Proc. of IEEE, Vol. 75, No. 8, Aug. 1987.

[35] E.A.Jonckheere, "Course Notes on Chaotic Systems.", Dept. of EE-Systems., Univ. of Southern California, Dec. 1993.

[36] J.Guckenheimer, P.Holmes, "Nonlinear Oscillations, Dynamical Systems, and Bifurcation of Vector Fields.", Springer-Verlag Co., 1985.

[37] S.N.Rasband, "Chaotic Dynamics of Nonlinear Systems.", J.Wiley & Sons Co., 1989.

[38] P.R.Halmos, "Measure Theory.", D.Van Nostrand Co. 1950.

[39] R.Mane, "Ergodic Theory and Differential Dynamics.", Springer-Verlag Co., 1987.

[40] T.S.Parker, L.O.Chua, “Practical Numerical Algorithms for Chaotic Systems.”, NY : Springer-Verlag in Progress.

[41] T.Saito, “An Approach toward Higher Dimensional Chaos Generator.”, IEEE Trans. on Circuit & Syst. Vol.37, No.3, 1990.

[42] T.Saito, “Reality of Chaos in Four-Dimensional Circuits.”, IEEE Trans. on Circuit & Syst., Vol. 38, No.12, 1991.

[43] M.Maretti, “Discrete Dynamical Systems and Chaos”, Longman Scientific & Technical, 1992.

[44] J.Banks, J.Brooks, “ On Devaney’s Definition of Chaos.”, The American Mathematical Monthly, Vol. 99, No. 4, PP. 332-334, 1992.

[45] H.Peitgen, D.Saupe, “Chaos and Fractals: New Frontiers of Science”, Springer-Verlag 1992.

[46] R.L.Devaney, “An Introduction to Chaotic Dynamical Systems.”, Second Edition, Addison-Wesley, 1989.

[47] A.Van Ooyen, B.Neihuis, ”Improving the Convergence of Back Propagation Algorithm.”, Neural Networks, Vol.5, No.3, 1992.

[48] H.Guo, S.Gelfland, ”Analysis of Gradient Descent Learning Algorithms for Multilayer Feedforward Neural Networks.”, IEEE Trans. on Circuit & Syst., Vol. 38, No.8, Aug. 1991.

[49] H.Yang, T.Dillon, ”Convergence of Self-Organizing Neural Algorithms.”, Neural Networks, Vol.5, No.3, 1992.

[50] G.Mirchandani, W.Cao, ”On Hidden Nodes for Neural Nets.”, IEEE Trans. on Circuit & Syst., Vol.36, No.5, 1989.

[51] A.Lin, J.Silvester, “Priority Queuing Strategies and Buffer Allocation Protocols in Traffic Control at an ATM Integrated Broadband Switching System”, IEEE JSAC, Vol. 9, No. 9, Dec. 1991.

[52] M.Gerla, L.Kleinrock, “Flow Control: A Comparative Survey”, IEEE Trans. on Commun., Vol. COM-28, No. 4, Apr. 1980.

[53] F.Kamoun, L.Kleinrock, "Analysis of Shared Storage in a Computer Network Node Environment under General Traffic Conditions", IEEE Trans. on Commun., Vol. COM-28, No. 7, Jul. 1980.

[54] P.Kermani, L.Kleinrock, "Virtual Cut-Through: A New Computer Communication Switching Technique", Computer Networks, Vol.3, 1979.

[55] G.Gallasi, C.Rigolio, "ATM Bandwidth Assignment and Bandwidth Enforcement Policies", Proc. of IEEE GLOBECOM'89, Dec. 1987.