Revista Colombiana de Estadística
Nº s 23-24, 1991

STOCHASTIC NEURAL NETWORKS

Max Garzón[1]
Depart.of Math. Sicienes
Memphis State University

Luz Gloria Torres[2]
Depto. de Matemáticas y Est.
Universidad Nacional

Abstract. Artificial neural networks are brain-like models of parallel computations and cognitive phenomena. We sample some basic results about neural networks as they relate to stochastic and statistical processes. Given the explosive amount of material, only models bearing a stochastic component in the function or analysis are presented, such as Hopfield and feedforward nets, Boltzman machines and some recurrent networks. Basic algorithms for learning such as backpropagation and gradient descent are sketched. A handful of applications (associative memories, pattern recognition, time series forecast) are described. Finally, some current trends in the field are discussed.

1. Introduction.

Artificial neural networks are brain-like models of parallel computations and cognitive phenomena. They were introduced in the seminal paper by McCulloch and Pitts (1943), in

1 Partially supported by NSF grant DCR-901985.

2 The author acknowledges support while on leave from the Universidad Nacional de Colombia visiting Memphis State University.

the 1940's as ways of implementating logical gates with abstractions of real neurons, and again in the 1960's by Rosenblatt (1962), Grossberg (1988), Kohonen (1984), Early negative results by Minsky and Papert (1986), and the unavailability of massively parallel machines to run realistic simulations of the power of the model discouraged researchers from actively pursing further study of their capabilities and applications. Although several of today's main figures in the field (Grossberg, Kohonen, Marr) remained attracted to the idea, it wasn't until the publication of Hopfield's landmarks (1982), 1986 that the time was right for a rebirth into a third period of activity, this time with every indication to stay for good. Today, artificial neural networks are not only revolutionizing fundamental ideas in computer science and informatics, but they are becoming fundamental tools in fields as diverse as electrical engineering, optimization, medical diagnosis, image processing, robotics, machine learning, time series prediction, artificial intelligence, cognitive science, neurophysiology, and even music and poetry.

The last few years have seen an explosive growth of activity and publications in the field of neural networks. It is estimated that the neural network literature contains over 10.000 papers as of now (June 1991). IJCNN (The International Joint Conference on Neural Networks), the major conference in neural networks in North America, by itsel publishes about 2,000 pages of recent research every year, not to mention many other publications in specialized journals. The last four years have seen the formation of least five major neural network societies, at least five new journals devoted entirely to the subject, and major conferences with over 3,000 attendees in America, Asia and Europe. Therefore we feel justified in so drastically oversimplifying and selecting the material in the following sections. These figures should also give the reader an idea of how short this paper falls of being even a modest survey. It is intended as a motivation for statisticians and probabilists for the study of neural nets and their applications to and from the viewpoint of their fields of interest.

Since their beginning, statistics and probability have played an increasingly important role in neural networks , either in the form of stochastic components or as analytical tools. The literature in stochastic networks, however, is fragmented and inaccessible, largely due to the fact that they are still in the research stage. A neural network may be stochastic in two rather different ways.

First, the networks itself may update cells randomly. Second, probabilistic and statistical tools are being used in the analysis of a deterministic network. The purpose of this article is to give an elementary exposition of the most prominent models, results and applications of both types of stochastic neural networks. No proofs, not even formal statements are given, although many references are given to the literature where exact details can be found.

2. Definitions.

In this section we describe the deterministic models prior to the introduction of the stochastic model. There are many varieties of networks, but the ones introduced here are building prototypes for virtually every conceivable network.

Neural networks are artificial models of biological brains, and the basic inspiration of their desing follows closely their anatomical structure. A biological brain is essentially made up of neurons interconnected in very complex spacio-temporal patterns, the exact significance of which is unknown. Thus an artificial neural net consists of basic components called *neurons*, an abstraction of the biological entity herein simply referred to as a *cell* or *unit*. A cell is characterized by a set of *activation values* from a set and an *activation function* $f : A \to A$. Activation functions are also called *transfer functions*. The activation is a full description of the state of a cell at any instant of time. It can be either a discrete value (e.g. binary, from the set $B := \{0,1\}$) or a continuous quantity (usually a value from the set of real numbers R). Cells change activation values in time according to their activation functions, which take as arguments weighted sums of few values of other cells, called *neighboring* cells. The pattern of interconnections among the cells, usually referred to as *architecture* or *topology* of the network, is determined by a directed graph (digraph). The vertices of the digraph correspond to the units of the net and an arc (oriented edge) indicates a *link* or *connection* among the corresponding cells. Every arc is weighted by the analogue of a *synaptic strength*, a value of the same nature as the activations of the cells. Weighted arcs play a role analogous to that of the axons and dendrites of real neurons. Links with positive real-valued weights are called *excitatory* and negatively weighted links are called *inhibitory*. Throughout, we will only consider time flowing discretely. Important networks based on continuous time with continuous updates governed by differential equations have been developed, and they will be summarily described in section 5.

For the sake of future reference we record this discussion in the following definition.

DEFINITION 2.1. *An activation set A is any set with an additive-multiplicative structure. An activation function is a self-map $f:A \to A$ that fixes 0, i.e. $f(0) = 0$. A neural network is a triple $N = \langle D, A, \{f_i\}\rangle$ consisting of a digraph D, an activation set A and a family of activation functions f_i, one for each vertex i in D. The global dynamics of N is defined by equations (2) below.*

Examples of neural networks are presented in detail in the following subsections. Very commonly, examples have binary activation values, and threshold and sigmoid functions for activations functions. Usually all cells are of the same type (discrete or continuous activations), and, correspondingly, networks are classified as *discrete* or *continuous*. Sometimes networks are also classified by the type of interconnection digraph. For instance, if this digraph consists of layers of disconnected cells, each layer communicating only to cells in the next layer, it is called a *feedforward* (or *layered*) net. If the diagraph contains cycles, it is called a *recurrent* net. If it is a complete graph, i.e. all cells are interconnected, it is called a *fully recurrent* net.

More important than the anatomy of a network is its evolution in time. Typically, a network operates as follows. A cell i reads the activation value of all units j having a link into it and makes a weighted sum that integrates all these inputs into a single sum, the net-input of cells i. It then applies its characteristic activation function to find a new activation value. If this action is taken simultaneously by all cells, the update is called *parallel*. Other common update modes are *asynchronous parallel* (*all* cells update, maybe at different times), *sequential* (one cell updates at a time according to a given *schedule*, e.g. in a fixed cyclic order), *block sequential* (like the sequential update but with cells partitioned into blocks and the whole block updating all at once), *stochastic* (if the cells update randomly according to a certain distribution). Stochastic nets will be considered more in detail in the following sections.

In order to understand the behavior of a net as a whole it is convenient to consider the so-called global activation of the network. an assignment $x: V \to A$ of activations to each cell i of a neural network is called a *total state* or *configuration*. A configuration is a snap-shot of the state of the cells at a given instant of time. For example, **0** and **1** are configurations consisting of a 0 and a 1 at every cell, respectively. The set of all configurations is denoted C. The local dynamics of a neural network induces a global map given by

$$T : C \to C \tag{1}$$

$$T(x)_i := f_i \left(\sum_j w_{ij} x_j(t) \right); \tag{2}$$

for all cells i.

2.1. Feedforward nets.

The simplest example of a feedforward net is the *perceptron*. Historically the first neural (as opposed to McCulloch & Pits's neuronal) model, it was introduced by Rosenblatt (1962) in the late 1950's. The perceptron is a continuous net with a single layer of disconnected *input* cells, each of which has a link to an *output* cell. The activation function is a threshold function θ determined by a value b of the type

$$\theta(u) = \begin{cases} 1 & \text{if } u \geqslant b \\ 0 & \text{else.} \end{cases}$$

A particular case of this function is the two valued signum sgn where $b = 0$.

This makes a perceptron very analogous to a single brain cell: if the net-input is above a certain value b, the cell goes into a firing state (activation 1), else it remains idle (value 0). The net result is a global dynamics of the type

$$T(x)_i = \theta(\sum_j w_{ij} x_j) \tag{3}$$

Perceptrons were originally used for pattern recognition. The cells in the input layers are actually arranged as a rectangular array of sensors to which the inputs x_i are fed. With the appropriate weights loaded to tye synapses, it can provide binary decisions and classify input patterns into two categories corresponding to the binary states of the output neuron. For example, if the objects to be categorized are determined by the values of three basic parameters x_1,

x_2, x_3, they are represented by points in $\mathbb{R}^3$. These objects are *linearly separable* if one can find a plane π of equation $w_1 x + w_2 y + w_3 z = b$ in $\mathbb{R}^3$ that separates them in the sense that all points in one category are below π and the others are above it. In this case they are naturally clustered in two categories. A perceptron can thus be designed to automate the process of recognizing objects. The input cells receive the characteristic parameters of an object and are connected with weights w_i to the output cell which has threshold b.

Despite some other interesting applications, perceptrons have long been known to have a limited recognition ability. For instance, one cannot design a perceptron with appropriate real-valued weights to compute an XOR of two binary input values. Mathematical proofs and arguments by Minsky and Papert (1986) of these an other negative facts back in the 1960's dissuaded contemporary researchers from studying feedforward networks with hidden layers in the 1960's and 1970's.

If one allows so-called *hidden units*, however, they can compute logical functions. It is easy to verify that the following 2-layer network would do XORs. Feedforward nets can have, in general, an arbitrary number of layers. It can be proved that each additional layer provides, in fact, additional computational power.

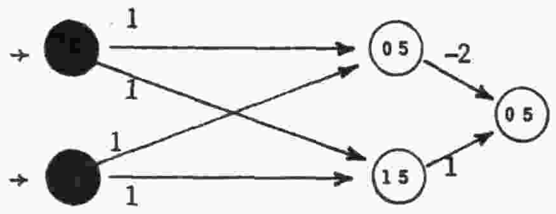

Figure 1
A perceptron for XOR

The update mode of a feedforward net is block sequential, with blocks in a layer simultaneously updated successively starting from the input layers toward the output (target) layer.

Feedforward nets are now very popular mainly due to the fact that a nice training algorithm, backpropagation, exists to successfuly train a network to find "by itself" a suitable set of weights for a given task, if such a set exists at all. Interesting feedforward nets of recent use have so far included most of the nets used in applications. However, recurrent nets are necessary for other applications.

2.2. Hopfield nets.

Hopfield nets are very useful examples of discrete recurrent nets. They were popularized by Hopfield, a neurobiologist and physicist, in a couple of widely read papers (Hopfield, 1982, 1986) that have been very influential in returning the attention of researchers to the untapped potential of neural net models after the perceptron's demise.

As with the perceptron, the activation functions of the Hopfield model are threshold functions. The weights are also real numbers. The activation values are binary values, say 0,1 (or equivalently, ±1 via the change of variable $x \to 2x-1$). The basic difference lies in the connections and the update mode. A hopfield network is fully connected and symmetric, i. e, every arc ij has an opposite arc ji with the same weight. Originally, Hopfield used the network as an associative memory, as described below in section 4.1. In this case cells are updated at random with the only requirement that each cell be updated, on the average, once every n (the number of cells) updates.

In the case of a feedforward net it is easy to decide when the network has obtained an answer as a result of its computation. One simply looks at the ouput cells after t iterations, the number of hidden layers in the net. In the case of a fully recurrent network it is no longer clear what to observe where. Associative memories require retrieval of certain binary patterns v^k associated with an initial configuration u^k of the network. The most natural for the net to return v is to initialize its cells to the values indicated by the binary vector u, and then repeatedly update the network in some predetermined fashion until a fixed point is reached. Thus the net has found an equilibrium state v, which is then regarded as the retrieved memory. The success of the method, of course, lies in *finding* suitable weights and in *guaranteeing* somehow that, under iteration, the stable states arrived at from a pattern u truly corresponds to the desired associated value v. These problems will be examined in section 4.1 together with the efficiency of networks in performing this task.

2.3. Stochastic neural networks.

The randomness in a Hopfield net lies in the uniform choice of cells in an update. Selected cells are updatedwith certainty. In a network with stochastic units, the transfer function is actually a probability distribution, a sigmoid function

$$f_i(u) = \frac{1}{1+e^{-\alpha u}} \,. \tag{4}$$

The networks is usually binary. The new activation of the unit is +1 with probability $f(net_{input}) = f_i(\sum w_{ij} x_i)$ and -1 otherwise, of course with the complementary probability. The update schedule can be randomized over all cells (recurrent nets) or just over particular cells in a layer (feedforward nets). Some of the units may be designated as input, others as output, and the rest as hidden units. Sometimes a so-called *bias* factor is introduced in equation (4), which multiplies the net-input before applying the transfer function.

The main motivation of a stochastic neural net is to provide a mechanism to implement a given probability distribution. Like the Hopfield network, one may want only certain output patterns occur with a certain distribution, despite the fact that the input patterns are fed randomly with, say, a uniform distribution. As with Hopfield network, the basic problem is to find a set of weights and a transfer function that allow the network to reach the desire equilibrium states with high probabilities, while others are reached with very low probability.

2.4. Boltzmann machines.

This model was originally introduced by Hinton and Sejnowski (Vol.1, 1986), as a continuous model of biological brains. It represents a departure from other models in two aspects. First, the updates are now made, though randomly, not as a function of local changes in neighboring cells, but rather as a function of *global* features of the entire network. Second, the distribution of updates varies according to a decreasing parameter τ called the *temperature* of the network. The model was inspired by the physical process of metallurgical annealing and introduced for optimization of real-valued functions.

Currently, the most common Boltzmann machine is basically a discrete and recurrent Hopfield network with hidden units stochastically updated. The weights w_{ij} on links ij between cells i and j are real-valued and symmetric, $w_{ij} = w_{ji}$. The update requires a random number generator and a temperature parameter. Initially, the temperature parameter is set to a high value. An *update* unit overseeing the entire network is in charge of the updating schedule. The overall goal of the update is to minimize a global quantity H called the *energy* of the network; for instance,

$$H(x) := -\frac{1}{2} \sum_{ij} w_{ij}\, x_i\, x_j .$$

For an update, a cell is randomly chosen and a random value ζ is obtained from the unit interval with a uniform probability distribution; the update actually changes the activation of a cell (from -1 to +1 or vice versa) in case the energy of the new configuration actually decreases (i.e. the change in energy is negative). However, if only this type of update is performed, the distribution of energy over configuration space may lead the net into local minima of H which are not global minima, it is thus necessary to introduce *annealing* updates, where the energy is actually allowed to increase, more rarely as time (temperature) progresses (decreases). The activation of a cell is thus switched in value if $\Delta H \geqslant 0$ and given by

$$e^{(\Delta H/\tau)} > \zeta ,$$

i.e. if the mean energy at temperature τ exceeds the random parameter ζ.

The foregoing examples are among the most important and useful neural networks. By themselves, they have little to of̲fer. The most interesting and remarkable properties of neural networks derive from their designer's ability to bypass the entire process of *analysis* and *programming* so characteristic of ordinary applications in artificial intelligence and symbolic computation. We consider this aspect in the following section.

3. Learning, generalization and degradation.

Learning in natural and artificial systems refers to the change in internal states of a system in order to cope with and adapt to changing conditions in the environment. Learning has long been considered one of the fundamental traits of living organisms. Orthodox AI (Artificial Intelli̲gence) created programs performing high-level tasks that when executed by humans appeared to be proof of intelligent behavior (chess-players, ping-pong players, even medical diagnosers like MYCIN, expert systems, etc.). The performance of these products is optimized by careful adjustements of key parameters obtained through analysis and experimentation with the programs, not typically *by the programs themselves*. There are of course interactive programs that can change these parameters by themselves on the run, but the nature of the changes requires a fixed framework that has to be entirely *known to the designer* in advance.

The internal state of a neural net consist not only of the entire patrern of activation of all cells (simetimes called a short-term memory), but more importantly, of the set of weights assigned to the connections (the long-term memory). What the network 'knows' is expressed in the strength of the synaptic connecions and the activations of its cells. Thus, to make a net work learn just requires to find a way to change its weights. To make it learn a significant task, say compute an XOR, requires finding the appropiate weights starting from a state of total 'ignorance' (i.e. from an initial set of random weights and configuration). This may appear difficult to achieve by a network itself. The remarkable fact, however, is that there exists a number of learning algorithms to modify the weights of a network in succesive stages so they converge often to the appropriate values for a given task.

There are various types of learning. The simplest is *supervised learning*. The task is represented by a number of input/output pairs (x^k, y^k) called *exemplars* or just data. A network learning phase for a neural net goes as follows.

1. The weights of the network are initially set to some predetermined values (say 0, or small random values);

2. The first x^1 is clamped on the input nodes. The net is then updated according to the current weights until some activations z^1 are read at the ouput nodes according to some criterion (e.g. in a feedforward net, right after the output layer is updated);

3. usually this output z^1 differs from what it ought to be, i.e. y^1. A predetermined calculation is made (by a teacher or supervisor) that will determine what changes are to be made to the weights to 'correct' the erronous output;

4. steps 1-4 are repeated for all remaining examplars.

Viewing a net as a vector of weights, a point in some high dimensional space, a learning algorithm provides a trajectory in weight-space. Given a particular task, one can associate with a specific set of weights some measure of *performance* (or error off the target). The goal of the algorithm is, ideally, to take the net to a global maximum of performance (or equivalently, a global minimum of the number of errors).

A good learning algorithm will, under a wide variety of inputs and initial conditions, make the net's set of weights converge to the appropiate values given sufficiently many exemplars. The goodness of the algoritm depends onthree main factors: how many exemplars are necessary, how general are the convergence conditions, and the speed and computational expense of updating weights. Establishing the convergence to appropriate weights is the most difficult part in designing a learning algorithm.

The various learning algorithms are distinguished in terms of the precise output criterion and the method to update weights. There are three basic types: *supervised* learning, *reinforcement* learning, and *self-organized* learning.

In supervised learning, a teacher (or critic, in reinforcement learning) provides immediate feedback and weights are corrected in proportion to the magnitude of the deviation of the net's response from the 'correct' response. Usually one assumes that the right set of weights exists, it is a matter of zeroing in on them using *exemplars*, i.e. pairs of desired inputs-outputs. For example, with a set of linearly separable data, one can start with zero weights for *perceptron learning*. For each pairs x^k, y^k, have the net output its value z^k, then change the weight from input cell j to the output cell *i à la* Rosenblatt, i.e. by

$$w_{ij}^{new} = w_{ij}^{old} + \Delta w_{ij} = w_{ij}^{old} + \eta(y_i^k \; z_i^k)x_i^k \qquad (5)$$

where η is a so-called *learning-rate*. This *perceptron learning rule* is very satisfying because Minsky and Papert proved a theorem stating that if the data x^k are linearly separable, then the weights w_{ij} converge to a net that produces the desired outputs from arbitrary initial weights. These results, however, only apply to perceptrons. The absense of similar strategies to effect weights changes in more complicated, even just feedforward nets with hidden units, was originally an obstacle for the advent of neural networks.

The reason behind the fascination for feedforward neural networks is *backpropagation*, the most basic 'general-purpose' learning algorithm. As such, it has been reinvented several times (Bryson & Ho 1969; Werbos 1974; Parker 1985;

Rumelhart, McLelland & Williams 1986). It is, in a way, a generalization of perceptron learning. The error function is usually the square error

$$E(w_i) = \frac{1}{2} \sum_{j,k} |y_j^k - z_j^k|^2 ,$$

where w_i denotes the vector of weigths into cell i. Proceeding backwards from the output layer, for each output cell i the weights from the previous layer w_{ij} are updated in the opposite direction of the gradient of the error function by

$$w_{ij}^{new} = w_{ij}^{old} + \Delta w_{ij} = w_{ij} - \eta \frac{\partial E}{\partial w_{ij}} \qquad (6)$$

(which also dependes on weights feeding into cells j at the previous level). This gradient makes sense for a differentiable transfer function, which is usually a logistic function approximating the threshild function at node i. After all weigth changes for all cells in all the hidden layers are determined, the *backward epoch* of the process is completed for the given pattern x^k, y^k. In the *forward phase*, all weights are updated. The whole process is then repeated for all remaining patterns. As the reader may suspect, this is a far more complicated and computationally expensive process, one that does not always necessarily converge to the right weights, even if they exist. Much effort has gone into analyzing convergence conditions and generalizing the method to work for recurrent networks, which tend to work better for a number of problems such as pattern completion, stereoscopic vision, and robot manipulators. The reader is referred to Hertz, Krogh and Palmer , (1991), for analysis and generalizations of the algorithm. Note that the nature of the learning makes it suitable for discrete and continuous networks.

There are other variations of the supervised learning paradigm. Important examples are, Hebbian learning (like in biology, weight changes are proportional to the correlation between firing of pre and post-synaptic units), reinforcement learning (only a 'right' or 'wrong', not the 'correct' value, are known for weight change), competitive learning (only one cell will fire at a time), gradient descent/ascent, and maximum likelihood estimation. The reader can find up-to-date references on these methods in Buntine (1991) and White (1989), along with a coherent general exposition of bayesian backpropagation in White (1989).

In unsupervised learning there is no teacher. The network must discover on its own correlations in order to categorize the data and produce outputs that exhibit a degree of self-organization. This is possible only where information is encoded as redundant data. For instance, the output may end up reflecting deviation from the average pattern (along one or several dimensions, like in principal component analysis), the propotype of a cluster with similar features, a trimmed-up encoding of the input (redundancy washed away), a feature extracted from the data, or a combination of some or all of these features. Unsupervised learning is useful with many-hidden layer feedforward networks where backpropagation is very slow and expensive, as a follow-up to supervised learning to allow for adaptation of the network. There are two basic types of learning rules, depending on how many cells are allowed to fire at the time. The exact description of these rules is too technical for this paper to describe. The reader can find additional detail and references in Hertz, Krogh and Palmer (1991).

Once a network has been trained it offers an advantage that gives it an edge over many traditional models. Its weights are 'set up', so to speak, to respond in a coherent, reasonable way to *unforeseen* exemplars. This feature is called *generalization*. As more and more exemplars are presented, continued learning may, on the other hand, lead to degradation of the learned responses to accomodate new patterns. Some neural models, such as Carpenter & Grossberg's ART models have the ability to retain previously learned models, a desirable feature in many applications,(Grossberg 1988). This degradation is not to be confused with so-called *graceful degradation*, i.e. the ability of the network to return meaningful (but somewhat degraded) answers even if some of its cells or synaptic links have becomes faulty or nonoperational.

4. Applications.

In this section we present a selection of some of the most notable uses of neural networks to date.

4.1. Associative memories.

As mentioned before, associative memories are systems in which pairs of patterns (u^i, v^i) are associated in such a way that, when presented a pattern u that resembles (with high probability) u^i, the system retrieves pattern v^i (with high probability). If $u^i = v^i$, the memory is *autoassociative*, otherwise it *is heteroassociative*.

Neural nets can be used to implement associative memories. What is required is that the vectors to be stored be fixed points (stable states), which means that the connection weights must be chosen so as to satisfy the equation

$$T(x)_i = f(\sum_j w_{i,j}\, x_j) = x_i,$$

for all cells i. In addition, the stable states must have an "attracting property", that is, if an input pattern is sufficiently similar to (e.g. a small Hamming distance away from) the stable state, it will end up in the stable state after succesive transitions, or at least in a state "near" the stable one(1). The former implies that an associative memory has an error correction property: if the number of errors of a given vector is such that the vector lies within an "appropriate" distance from a memory (stable vector), then total or partial error correction is obtained according to whether the vector reaches the memory or a state close to it after iterations of the system. Thus associative memories are very useful in recognition and reconstruction of images and in information retrieval when only a partial or degenerate input is given.

For a fixed network with n cells and discrete activations only finitely many memories can be stored in the network because there are only finitely many fixed points of the

(1) Here "nearness" is usually measured in terms of the Hamming distance, where the distance between two vectors is the number of different components.

network dynamics. The patterns of activation that evolve into the same fixed points x are the possible corrupted input signals that admit error-correction. They form the basin of attraction of x. There is therefore a trade-off between the number of memories to be stored and the robustness with which they can be retrieved. Memory will fade as the number of vectors to be memorized increases. For instance, some fixed points will not correspond to an actual pattern (*a spurious memory*); or a memory given as an initial condition will converge to a different fixed point (the pattern is *unstable*). The natural question arises as to how many vectors can be stably recalled with high probability for a given network on n cells.

The first answer to this question was provided by Hopfield (1982). It has been confirmed experimentally that when m memories are stored in n cells and m is kept below $0.15n$, spurious and unstable recall is very rare. Exact analytic results were later obtained by McEliece and coworkers (1987) for random errors. If $m < n/4 \log n$ then with very high probability all the memories will be stable. For $n/4 \log n < m < n/2 \log n$ still most memories will be stable with high probability. The question remains whether stable memories can attract all vectors with a positive radius ρn (in Hamming distance), for some $\rho > 0$. Komlos and Paturi (1988) further established convergence in at most $O(\log \log n)$ steps as well as the existence of a positive radius of attraction for $m < n/4 \log n$. They also established the existence of exponentially many states which are stable, although in a weaker sense defined in terms of energy. Exact descriptions of these results are beyond the scope of this survey - Komlos and Paturi (1988).

The associative memory problem can be regarded as a particular case of a more general problem dealt with in the next section.

4.2. Pattern classifiers.

Classification is a fundamental idea in many areas, particulary learning. It usually involves the partition of objects in question in a number of categories, usually finite. The problem of *pattern classification* consists in determining, with a minimum probability of error, to which of a finite number of possible categories an input pattern belongs. This problem is central to areas such as artificial intelligence, machine vision, image processing, speech processing, automatic mailing systems, and many others.

Traditional classifiers usually perform by finding the input with "maximum score" obtained in matching with an exemplar from the given category. In most cases these classifiers rely on strong assumptions made on the distribution of the input patterns so as to estimate the parameters that stay fixed during the process. The categorization of data is usually performed in such a way that the output will be classified as "high" upon input of a pattern if it is close to the corresponding exemplar in the category. The aim is to perform the task correctly even if a particular pattern is slightly distorted. For instance, pattern classifiers can be used as associative memories where the stable vectors or memories correspond to the categories of the network.

Neural networks are widely used in pattern classification. The advantage of neural networks is that, in contrast to statistical methods, they are adaptive, non-parametric and more flexible in the specification of underlying distributions.

The simplest pattern classifier is the perceptron mentioned in subsection 2.1. It decides if an input is above or below an hyperplane (in our case a line) forming two regions or categories separated by a decision boundary. If the data is not separable and their density functions overlap, the decision boundaries may oscillate. Approximate solutions can be found by considering the *least mean square* approach. This procedure minimizes the mean square error between the desired and the actual output of the perceptron. Weights are corrected according to the difference between the two patterns. The procedure works with other neural networks as well. By minimizing the probability of error classification, the network acts a *bayesian classifier*. The question now is whether or not the network approaches an optimal decision. Specht 1990 proves that if the sigmoid activation function, commonly used for *back propagation*, is replaced by an exponential function, virtually an optimal Bayesian classification is obtained.

Another example is the *winner-take-all* network. It consists of an input layer of cells connected to a fully interconnected set of binary cells. Weights are set up so that only one of the output units, called the *winner*, can fire at any time. It is usually the unit with the largest net input.

The *Hamming net*, like several other networks, implements a classical algorithm. The output of the network is the vector nearest in Hamming distance to the input vector. The result is obtained by actually implementing in neural primitives the *optimum minimum error classifier* algorithm, used to solve some problems in communications when binary signals are sent through memoryless channels. The Hamming net consists of two subnets, a feedforward net and a winner-take-all net, each containing two layers. The conecctions are as follows: In the lower net, input node j connects to next layer node i with weight w_{ij}; node i in this layer connects to a corresponding node i in the first layer of the upper net with weight 1; node j connects to node in the same layer with weight u_{ij}; node j in this layer connect to output node k with weight v_{kj}. The net effect of these two layers is as follows. The first subnet finds the difference between the number of elementos of the input pattern and the Hamming distance to the exemplar for each class. The winner-take all subnet then selects the maximum value yielding the corresponding class of the input vector.

The reader is referred to Lippmann (1987), for a more detailed study of pattern classifiers.

4.3. Time series.

A *time series* is a sequence of observations ordered in time. The series is *continuous* or *discrete* depending on whether the observations are made continuously or at discrete times. An example of a discrete series is the value of the stock of a certain company through time. The objectives in time series analysis include obtaining simple descriptive measures (if possible) of the main properties by plotting the data, validating empirically prior knowledge on its structure, and, most importantly, predicting as saccurately as possible future values of the series.

If the future behavior of a time series can be exactly predicted based on some knowledge of the past, then the time series is deterministic and no further study is required. Otherwise, it is a statistical time series. The series can be univariate or multivariate depending on whether the observations are taken on a single value or simultaneously on several values. A time series can be seen as a realization of a stochastic process. Unlike many processes handled by statistical theory, however, the special feature about time series as a random process is that observations are not independet and the temporal order is quite important. Time series apply to many fields. Much of the behavior encountered in social sciences, natural sciences, biology, physics and other sciences can be modeled in the form of time series. The apparently chaotic nature of data in areas such as economy and history has given an increasing interest to time series analysis.

Given observations $x_1, x_2, \ldots, x_t$, up to time t, we wish to predict the value at time $x_{t+1}, \ldots, x_{t+m}$, m a positive integer. One of the fundamental problems is the selection of a mathematical model to represent the process. Once the form of the model is found, it is a matter of using known values of the series to determine the unknown parameters in the model. For example, one of the most general techniques, the Box-Jenkins method assumes that there is a linear relationship between a number of consecutive terms of the series, of the form

$$x_{t+1} = a_o x_t + a_1 x_{t-1} + \ldots + a_p x_{t-p} + w_t,$$

where w_t is some random noise. Other models include ARMA, ARI

MA, etc. Most of these methods in time series forecasting rely on linear relationships among the variables.

Yet, most of the data obtained from the real world are nonlinear. Neural Networks offer an alternative model for time series forecast. There are several reasons why they may be better models. First, they are nonlinear models. Second, learning algorithms like backpropagation, if successful, will "discover" the best possible set of weights to fit the given data without previous knowledge even of their form. Third, they do not impose in advance too particular a restriction on the relationship between the various parameters represented in the hidden layers of the net. They are adaptive in the sense that the models change with the structure of the data if a training algorithms is in place.

A feedforward network may be used to predict future values for a time series as follows (Mehrotra, Mohan and Ranka 1990). The last *n* terms of the series can be used as input for the cells in the input layer. (The value of *n* requires judicious choice). The output layer will contain the value predicted by the network for the next term of the series. This predicted value can be either ignored (one-lag prediction) or used for further training (multi-lag prediction). This method has been used for prediction of sun-spots, (Mehrotra, Mohan and Ranka 1990). Initial segments of the known series can thus be used for training set for the network. The remaining values can be used to verify the goodness of the network. If the actual values of the series are known after some time (for instance in the case of sun-spots) they can be used for continued training. The network thus becomes an adaptive predictor of a series in which the parameters of the series are themselves changing with time. The problem now reduces to determining the most suitable value of n and the architecture of the network that will provide more accurate predictions that traditional models.

Stochastic recurrent networks have also been used to predict future values of a multivariate series. For example, in Kehagias, (1991) they have been applied to series whose val ues can be expressed over a finite alphabet, e.g. quantized speech waveforms. It is assumed that an unknown network is producing the values of the series. The problems of *prediction* and *classification* are then formulated as follows. Given outputs $y^1,\dots,y^t$ at the present time, and the inputs $x^1,\dots,$ x^{t+1} up to the immediate future, predict the maximum likelihood estimation of the ouput y^{t+1}. In formulas, using the corresponding capital for the corresponding random variables, one wants a value for Y^{t+1} that maximizes the function $Y_t(y)$ given by the conditional probability

$$prob(Y^{t+1} = y \mid Y^1 \dots Y^t = y^1 \dots y^t;\ X^1 \dots X^{t+1} = x^1 \dots x^{t+1})$$

In the classification problem, given the same but x^{t+1}, one asks for the networks N_i out of a number of candidates $N_1,\dots$ $\dots, N_k$ that has maximum posterior probability of producing $y^1,\dots,y^t$.

Although experimental results show succesful prediction of future values, in the sense that they were closer to exact values than the ones obtained with the Box-Jenkins method, there is not underlying theory that will support these results. A first step in this direction can be found in Kehagias (1991).

4.4. Computational models of physics.

Discrete neural networks as given in definition 2.1. are a generalization of a model, cellular automata, introduced by John von Neumann(2) as discrete computational versions of partial differential equations in the 1950's. Since then, cellular automata have developed into a very active field of research, mainly by physicists and computer scientist (Toffoli and Margolus 1987, Wolfram 1987). A cellular automaton can be defined as a neural network in which the underlying graph is homogeneous (it looks the same from every node) and all cells apply the same rule. Typical examples are the points of integer coordinates in real euclidean spaces connected in the usual grid manner by horizontal and vertical edges. Other examples are regular trees, and, more generally, Cayley graphs of arbitrary finitely generated groups. With binary states, common rules are the sum modulo 2 of a subset of the states of neighboring cells. Conway's game of LIFE is probably the most popular example of a cellular automaton.

Cellular automata provide *simply* defined *discrete* models of a wide variety of natural and complex phenomena hitherto thought to be undescribable by simple classical continuous or discrete systems. Examples include the Greenberg-Hasting models for the Belusov-Zhabotinsky reaction, the digital BBM (billiard ball) model for conservative mechanics, the HPP-GAS, TM-GAS, and FHP-GAS models of fluid dynamics for the Navier-Stokes equation, Ising systems, spin glasses,

(2) S. Ulam's seems to have made crucial suggestions on the type of model.

voting models of cooperative phenomena, and myriad other cellular automata rules currently under research. They have even been used as a primer example of artificial life and related issues (Langton 1991). The reader is referred to Gutowitz (1990). Toffoli and Margolus (1987) and Wolfram, (1987), for a detailed description of applications of cellular automata. Generally speaking, they can be regarded as imaginary miniature universes where the local rules play the roles of *physical* local laws and clusters of pixels in temporally stable configurations behave like objets in that universe.

Most common cellular automata models have been deterministic. They are closely related to deterministic neural networks (Gutowitz 1990 pp.431-440) for precise relationships between the two models). However, some recent models have been proposed concerning probabilistic models and inherently probabilistic phenomena such as quantum mechanical effects and even the entire universe itself (Gutowitz 1990). For exemple, the *deterministic* 2-dimensional model of sum modulo 2 has been found to possess quantitative properties akin to quantum mechanical properties in Vichniac, (1984). Richard Feynman (1982) considers very seriously the idea that the evolution of our physical universe may be governed by local rules of a cellular automaton type with stochastic spatiotemporal update depending not only on the immediate past, but also on the immediate future. Rujan (1987) has furthered this consideration to full blown models of quantum mechanics. Other probabilistic cellular automata rules have been examined in Bramson and Griffeath (1990).

5. Other applications.

The foregoing sections have barely touched upon the various neural models and their applications. They have emphasized discrete models. Continuous models have been studied, mainly by Grossberg (1988), Kohonen (1984) and Fukushima. The ART (Adaptive Resonance Theory) models of Grossberg and his collaborators have as activation values real numbers and the local evolution is governed by differential equations. They have been used with great success in many areas such as a pattern classification, adaptive resonance and neural modeling of brain functions.

Neural modeling is an interesting overlap with medicine and neurophysiology. Neural nets can be used as experimental full models of nervous systems of primitive organisms (including training and adaptation), models of parts of the human brain (hyppocampus, cortical areas, eye movement control mechanisms, language functions, etc.), models of learning and behavior, and so forth.

Industry has found a great deal of applications for neural network as well. We mention just a couple. A neural-net based systems has been designed and successfully outperforms other system for airport security in detection of plastic explosives. Neural-based pattern recognizers are in use to identity handwritten characters and zip codes for automatic sorting of mail in the postal service. This type of control system is beginning to find its way into homes in appliances, electronic devices, etc. Large companies have been founded in Europe, North America and Japan for research and development of an industry projected to have a volume of operations in the multibillion dollar range by 1993.

Neural models have also been found of interest as statistical tools, as indicates above. As their properties becomes better known, they are likely to become important tools in applied statistics as well. In turn, probability and statistics can be used to analyze the difficult problem of the longterm behavior of dynamics and learning by neural nets (White 1989). This interrelation is likely to increase as the interest in neural nets shifts from experimentation to a more systematic and analytical stage of development.

*

REFERENCES

D.H. Ackley, G.E. Hinton and T.J. Sejnowski, (1985) A learning algorithm for Boltzman Machines, *Cognitive Science* 9, 147-169.

R. Allen and J. Alspector, (1990) Learning of stable states in Stochastic Asymmetric Networks, *IEEE Transactions on Neural Networks*, 1:2, 233-238.

W.L. Buntine, (1991) Bayesian Back-Propagation, avaible as (Stanford U. Tr. from the neuroprose archive via ftp).

R. Feynman, (1982) Simulating physics with computer, *Int. J. Theor. Physics* 21:6/7, 467-488.

H. Gutowitz, ed. (1990) *Cellular automata: theory and experiment*. Proc. 3rd Workshop, *Physica D* 45:1-3.

M. Bramson and D. Griffearth, (1990) Flux and fixation in cyclic particle systems, TR. U. of Wisconsin, preprint.

S. Grossberg, (1988) Competitive Learning: From Interactive Activation to Adaptive Resonance, D. Waltz & J. A. Feldman (Eds.) *Connectionist Models and their Applications*, Ablex.

J. Hertz, A. Krogh, and R. Palmer, (1991) *Introduction to the theory of neural computation,* Santa Fe Institute studies in the science of complexity, Addison-Wesley.

J.J. Hopfield, (1982) Neural networks and physical systems with emergent collective computational abilities, Proc. Mat. Acad. Science 79, 2554-8.

J.J. Hopfield, (1986) Computing with neural circuits: a model, Science 233, 625-33.

A. Kehagias, (1991) Stochastic recurrent networks: Prediction and classification of time series, TR, Brown University.

T. Kohonen, (1984) Self organization and associative memory, New York, Springer Verlag.

T. Kohonen, (1988) An Introduction to Neural Computing, *Neural Networks* Vol. 1, 3-16.

J. Komlos and R. Paturi, (1988) Convergence results in an associative memory model, Neural Networks, Vol. 1, 239-250.

M. Langton, ed. (1989) "Artificial Life", Proc. first conference, MIT Press, Cambridge, MA, See also Proc. 2nd conference, (1991) Santa Fe Institute series on complexity, Addison-Wesley.

M. Li, K. Mehrotra, C.K. Mohan, and S. Ranka, (1990) Forecasting sunpots numbers using neural networks, *Proc. IEEE Symp. on Intelligent Control.* Sep.

R.P. Lippmann, (1987) An introduction to computing with neural nets, *IEEE ASSP* magazine, April. (1988) Reprinted in Vemuri, 36-54.

W.S. McCulloch & W. Pitts, (1943) A logical calculus of the ideas immanent in nervous activity, Bull. Math. Biophys. 5, 115-33.

R.J. Mc Eliece, E.C. Posner, E.R. Rodemic and S.S. Venkatesh, (1987) The capacity of the hopfield associative memory, *IEEE Trans. on Information Theory*, 33, 461-482.

J.L. McClelland and D.E. Rumelhart (Eds.) (1986) *Parallel Distributed Processing*, MIT Press, Cambridge, MA, Vols. 1,2.

M. Minsky and S. Papert, (1986) *Perceptrons*, MIT Press, Cambridge, MA, Vols. 1,2.

F. Patrick, (1990) Computational Complexity issues in Neural Associative Memories, Department of Computer Science, University of Helsinki, report C. 40.

F. Rosenblatt, (1962) *Principles of neurodynamics*, New York, Spartan.

P. Rujan, (1987) Cellular Automata and Statistical Mechanical models, *J. Statistical Physics* 49, 139-232.

D.E. Rumelhart, G. Hinton and R.J. Williams, (1986) Learning Internal Representations by Error Propagation, in McClelland and Rumelhart, 318-362.

T.J. Sejnowski and C.R. Rosenberg, (1987) Parallel Networks that the Learn to Pronounce English Text, *Complex Systems*, Vol. 1, 145-68.

D.F. Specht, (1990) Probabilistic Neural Networks, Neural Networks, Vol. 3, 109-118.

T. Toffoli and N. Margolus, (1987) *Cellular Automata Machines*, MIT Press, Cambridge, MA.

V. Vemuri, (1988) Artificial neural networks, theoretical con cepts, The Computer Society Press.

G. Vichniac, (1984) "Simulating Physics with Cellular Automata", *Physica* 10D, p.96.
Reprinted in Cellular Automata, *Proceedings of the Los Alamos Conference*, North Holland, Amsterdam.

H. White, (1989) Learning in artificial neural networks, a statistical perspective, Discussion paper 89-49. Economics, U. of California, San Diego.

H. White, (1989) Some asymptotic results for learning in sin gle hidden-layer feedforward neural models, *J.Amer. Stat. Ass*. 1003-1013.

S. Wolfram, (1987) Theory and Applications of Cellular Automata, World Publishing Co., Singapore.

*