B. Tech. Seminar Report

on

Artificial Neural Networks and It's Applications

Submitted in partial fulfillment of the requirements

for the degree of

Bachelor of Technology

by

KSH. RAKHESH SINGH

96007032

under the guidance of

Prof. K. V. V. MURTHY

Department of Electrical Engineering

Indian Institute of Technology

Bombay

March 1999

ACKNOWLEDGEMENT

I would like to express my sincere gratitude towards my guide, Prof. K. V. V. Murthy, for guiding me at every stage of the seminar and for being a continuous source of encouragement.

Date: March 22,1999

Ksh. Rakhesh Singh

(96007032)

ABSTRACT

Artificial neural networks are parallel computational models comprised of densely interconnected adaptive processing units. A very important feature of these networks is their adaptive nature, where "learning by example" replaces "programming" in solving problems. This feature makes such computational models very appealing in application domains where one has little or incomplete understanding of the problem to be solved but where training data is easily available. Artificial neural networks are viable computational models for a wide variety of problems. These include pattern classification, speech synthesis and recognition, function approximation, image compression, associative memory, forecasting, optimization, nonlinear system modeling, and control. The most wonderful thing is Neural networks could do certain things which you simply couldn't reprogrammed computing.

Nomenclature

d	Desired output of a neuron
E	Squared error in the output of a neuron
f	Activation function of a neuron
o	Output space of an artificial neuron
T	Threshold value
W	Connection or Weight matrix
w_{mn}	Weight due to n^{th} input to m^{th} neuron
x	Input space of an artificial neuron
α&c	Learning constant
Γ	Activation matrix

Contents

List of Figures

Chapter 1

Introduction

When interest in neural networks revived some decades ago, few people believed that such systems would ever be of any use. Computers worked too well: it was felt that they could be programmed to perform any desired task. Clearly, the fashion has changed. Now, limitations of current computers in solving many problems involving difficult to define rules or complex pattern recognition are widely recognized: if anything, expectations for neural networks may be too high. The problem is no longer to convince anyone that neural networks might be useful, but rather to incorporate such networks into systems that solve real-world problems economically.

Neural networks are inspired by biological systems where large numbers of neurons, that individually function rather slowly and imperfectly, collectively perform tasks that even the largest computers have not been able to match. They are made of many simple processors connected to one another by variable memory elements whose weights are adjusted by experience. They differ from the now standard Von Neumann computer in that they characteristically process information in a manner that is highly parallel than serial, and that they learn (memory element weights and threshold are adjusted by experience) so that to a certain extent they can be said to program themselves. They differ from the usual artificial intelligence systems in that (since neural networks can learn) the solution of the real-world problems requires much less of the expensive and elaborate program-

ming and knowledge engineering required for such artificial intelligence products as ruled-based expert systems.

In their current state, neural networks are probably best at problems related to pattern recognition. Some existing neural network systems can efficiently and rapidly learn to separate enormously complex decision spaces. Products that recognize characters, assembly line parts or signatures, that make complex decisions mimicking or improving on human experts that can diagnose engine or assembly line problems are already fielded.

Chapter 2

Fundamentals of Neural Networks

2.1 What is Neural Network?

Neural Networks use a set of processing elements (or nodes) analogous to neurons in the brain. Hence the name, neural networks. These processing elements are interconnected in a network that can then identify patterns in data as it is exposed to the data. In a sense, the network learns from experience just as people do. This distinguishes neural networks from traditional computing programs, that simply follow instructions in a fixed sequential order.

In the simplest case, it has three layers: input layer, hidden layer and output layer. It is the hidden layer that performs much of the work of the network. The output layer, for example, predict sales (output) based on past sales, price and season (inputs).

More on the Hidden Layer

Each node in the hidden layer is fully connected to the inputs. That means what is learned in a hidden node is based on all the inputs taken together. This hidden layer is where the network learns interdependencies in the model. Simply speaking a weighted sum is performed. If we assume that there are n inputs in the input layer and denote it's weights by w_1, w_2, ..., w_n and inputs by x_1, x_2, ... x_n. Then, x_1 times w_1 plus x_2 times w_2 on through x_n and w_n. This weighted sum is calculated for each hidden node and compared with a threshold value of the node

and if it exceeds, then the corresponding output will be active, otherwise, it will remain inactive.

Where does the Network gets it's weights from?

This we will consider in Network Learning.

The "artificial neuron" is the basic building block of an artificial neural network. We will consider some of it's models below.

2.2 McCulloch-Pitts Neuron Model

The Figure 1 shows McCulloch-Pitts model of a neuron. In the Figure, inputs x_i, for i=1,2,...,n, are 0 or 1, depending on the absence or presence of the input impulse at instant k. The neuron's output signal o at the instant k+1 is given by taking weighted mean of the input at the instant k and comparing it with the threshold T, as follows:

$$o^{k+1} = \begin{cases} 1, & \sum_{i=1}^{n} w_i {x_i}^k \geq \mathrm{T} \\ 0, & \sum_{i=1}^{n} w_i {x_i}^k < T \end{cases} \tag{2.1}$$

where w_i is the weight of the i^{th} input x_i.

2.3 Main Artificial Neuron Model

A general neuron symbol is shown in Figure 2. The neuron output signal o is given by following relationship

$$o = f(\mathbf{w}^t\mathbf{x}) = f(\sum_{i=1}^{n} w_i x_i) \tag{2.2}$$

where $\mathbf{w}$ is a *weight vector* defined as $\mathbf{w}=[w_1 w_2 .... w_n]^t$ and $\mathbf{x}$ is the input vector $\mathbf{x} = [x_1 x_2 ..... x_n]^t$. The function $f(\mathbf{w}^t\mathbf{x})$ is as an *activation function*. Its domain is the set of activation values, *net*, of the neuron model and is defined as

$$net = \mathbf{w}^t\mathbf{x}. \tag{2.3}$$

Here, the output is given by the value of the activation function at the *net* and *net* is given by the weighted mean of the inputs as in equation 2.3. Typical activation functions used are

$$f(net) = (2/(1 + exp(-\lambda net))) - 1 \tag{2.4}$$

for continuous case where $\lambda > 0$ is proportional to the neuron gain determining the steepness of the continuous function f(*net*) near $net = 0$. When $\lambda \to \infty$, it gives the relation,

$$f(net) = \begin{cases} 1, & net > 0 \\ -1, & net < 0 \end{cases} \tag{2.5}$$

for discrete case. The continuous activation function is shown in Figure 3(a) for various Λ. For unipolar continuous case,

$$f(net) = 1/(1 + exp(-\lambda net)) \tag{2.6}$$

and for discrete case,

$$f(net) = \begin{cases} 1, & net > 0 \\ 0, & net \leq 0 \end{cases} \tag{2.7}$$

Corresponding figure is shown in Figure 3(b).

Given a layer of m neurons, their output values $o_1, o_2, ..., o_m$ can be arranged in a layer's output voltage $\mathbf{o} = [o_1, o_2, ..., o_m]^t$ where o_i is the output signal of the i^{th} neuron. The domains of the vector o are defined in m-dimensional as follows for i = 1.2.....m (Hecht-Nielson 1990). For bipolar case,

$$(-1, 1)^m = [\mathbf{o} \epsilon \mathbf{R}^m, o_i \epsilon (-1, 1)]. \tag{2.8}$$

For unipolar case,

$$(0, 1)^m = [\mathbf{o} \epsilon \mathbf{R}^m, o_i \epsilon (0, 1)]. \tag{2.9}$$

2.4 Models of Artificial Neural Networks

The network models can be broadly classified into two.

2.4.1 Feedforward Network

Activation value and output for the i^{th} neuron is given by

$$net_i = \sum_{j=1}^{n} w_{ij} x_j, \tag{2.10}$$

$$and, o_i = f(\mathbf{w}_i^t \mathbf{x}) \tag{2.11}$$

for i = 1.2.....m where $\mathbf{w}_i = [w_{i1} w_{i2} .... w_{in}]^t$ where w_{ij} is the weight due to j^{th} input on the i^{th} neuron. Introducing the nonlinear matrix operator Γ, the mapping of the input space $\mathbf{x}$ to the output vector $\mathbf{o}$ is expressed as

$$\mathbf{o} = \Gamma[\mathbf{W}\mathbf{x}] \tag{2.12}$$

where nonlinear matrix operator Γ is given by

$$\Gamma() = \begin{bmatrix} f() & 0 & \cdots & 0 \\ 0 & f() & \cdots & 0 \\ \vdots & \vdots & \cdots & \vdots \\ 0 & 0 & \cdots & f() \end{bmatrix}$$

and f() denotes the activation function. The matrix $\mathbf{W}$ is given by

$$\mathbf{W} = \begin{bmatrix} w_{11} & w_{12} & \cdots & w_{1n} \\ w_{21} & w_{22} & \cdots & w_{2n} \\ \vdots & \vdots & \cdots & \vdots \\ w_{m1} & w_{m2} & \cdots & w_{mn} \end{bmatrix}$$

The mapping of an input pattern into an output pattern as shown in Figure 4 is of the feed forward and instantaneous type, since it involves no time delay between the input $\mathbf{x}$, and the output $\mathbf{o}$. We can rewrite in the explicit form involving time t as

$$\mathbf{o}(t) = \Gamma[\mathbf{W}\mathbf{x}(t)]. \tag{2.13}$$

Even though the feed forward network has no explicit feedback connection when $\mathbf{x}(t)$ is mapped into $\mathbf{o}(t)$, the output values are often compared with "teacher's" information, which provides the desired output value, and also an error signal can be employed for adapting the network's weights.

2.4.2 Feedback Network

The mapping of the output to $\mathbf{o}(t)$ into $\mathbf{o}(t+\Delta)$ can be written as

$$\mathbf{o}(t+\Delta) = \Gamma[\mathbf{W}\mathbf{o}(t)] \tag{2.14}$$

The time Δ elapsed between t and $t+\Delta$ is introduced by the delay elements in the feedback loop as shown in Figure 5(b). Initialize $\mathbf{o}(0) = \mathbf{x}(0)$. The input is then removed and the system remains autonomous for $t>0$. It finds an equilibrium state after some transition. For a discrete neural system, the performance at time $k\Delta$ is given by

$$o^{k+1} = \Gamma[wo^k], \tag{2.15}$$

for k=1.2..... where k is the instant number. Refer to Figure 5(a).

2.5 Neural Processing

Process of computation of o for a given x performed by the network is known as *Recall*. Recall corresponds to the decoding of the stored content which may have been encoded in the network previously.

2.5.1 Association

Assume that a set of patterns can be stored in a network. Later if the network is presented with a pattern similar to a number of the stored set, it may associate the input with the closest stored pattern. This process is called *Autoassociation*. Typically, a degraded input pattern serves as a cue for retrieval of it's original form.

In *heteroassociative* processing, the associations between the pairs of patterns are stored as in Figure 6.

2.5.2 Classification

Let us assume that a set of input patterns is divided into a number of classes, or categories. In response to an input pattern from the set, the classifier is supposed

to recall the information regarding class membership of the input pattern.

If the network's response is the class number but the input pattern does not exactly correspond to any of the patterns in the set, the processing is called *Recognition*. Refer Figure 7.

2.6 Neural Network Learning Rules

The Neural networks are adaptive. It's weights are modifiable depending on the input signal it receives, it's output values, and the associated teacher response. Here, we will consider some of the learning rule for the network.

2.6.1 Hebbian Learning Rule

Some cases, teacher signal not available and no error information, neuron modify it's weights based on the input/output. Here weight vector $\mathbf{w}_i = [w_{i1} w_{i2} ... w_{in}]^t$ increases in proportion to the product of input $\mathbf{x}$ and learning signal $\mathbf{r}$. In Hebbian Learning

$$\mathbf{r} = f(\mathbf{w}^t\mathbf{x}) \tag{2.16}$$

$$or. \quad \Delta w_i = cf(\mathbf{w}^t\mathbf{x})\mathbf{x}. \tag{2.17}$$

Here, Δw_i denotes the amount by which i_{th} neuron's weight is updated. For single weight,

$$\Delta w_{ij} = co_i x_j. \tag{2.18}$$

for j = 1.2.....n. It requires weight initialization at small random values around $w_i = 0$ prior to learning (Hebb 1949). The rule states that if the cross product of the input and output, or correlation term $o_i x_j$ is positive, this result in the increase in the weight; otherwise it decreases. It represents a purely feed forward, unsupervised learning. See Figure 8(a).

2.6.2 Delta Learning

In this learning, teacher's signal is given by

$$r = [d_i - f(\mathbf{w}_i^t\mathbf{x})]f'(\mathbf{w}_i^t\mathbf{x}) \tag{2.19}$$

which can be derived from the condition of the least squared error between o_i and d_i. Squared error is given by

$$E = 1/2(d_i - o_i)^2 \tag{2.20}$$

$$or,\ E = 1/2[d_i - f(\mathbf{w}_i^t\mathbf{x})]^2 \tag{2.21}$$

$$or,\ \nabla E = -(d_i - o_i)f'(\mathbf{w}_i^t\mathbf{x})\mathbf{x} \tag{2.22}$$

The components of the gradient are

$$\partial E/\partial w_{ij} = -(d_i - o_i)f'(\mathbf{w}_i^t\mathbf{x})x_j, \tag{2.23}$$

j= 1,2,.....,n. Minimization of error requires the weight changes to be in the negative gradient direction, hence, Δw_i = -$\eta\nabla$E where η is a positive constant. Therefore the weight adjustment becomes

$$\Delta\mathbf{w}_i = c(d_i - o_i)f'(net_i)\mathbf{x}. \tag{2.24}$$

The weights are initialized to arbitrary values in this method of training (McClelland and Rumelhert 1986). Refer Figure 8(b). The extension of Delta Learning to multilayer networks give the backpropagation algorithm.

2.6.3 Winner-Take-All Learning Rule

This rule is an example of competitive learning and it is used for unsupervised learning (Hecht-Nielson 1987). Neurons in the layer, say m'th, has the maximum response due some input **x**. This neuron is declared as the *winner*. As a result of this learning event, the weight vector $w_m = [w_{m1}w_{m2}...w_{mn}]^t$ containing weights highlighted in the figure 7(c) is the only one adjusted in the given unsupervised learning. It's increment is given by

$$\Delta\mathbf{w}_m = \alpha(\mathbf{x} - \mathbf{w}_m) \tag{2.25}$$

where $\alpha > 0$ is a small learning constant, typically decreasing as learning progresses. The winner selection is based on the following criterion of maximum activation among all p neurons participating in a competition:

$$\mathbf{w}_m^t \mathbf{x} = \max_{i=1,2,\ldots,p} (\mathbf{w}_i^t \mathbf{x}) \tag{2.26}$$

This corresponds to finding the weight vector that is closest to the input $\mathbf{x}$. The rule then reduces to incrementing $\mathbf{w}_m$ by a fraction $\mathbf{x} - \mathbf{w_m}$. Note that only the winning neuron fan-in weight vector is adjusted. See Figure 8(c).

2.7 Representations in Connectionist Models

Four kinds of representations are possible in a net (see figure 9).

First, information can be completely local: for a given set of stimuli, a single unit in the hidden layer becomes active and passes information to the output layer. This hidden unit might be thought of as passing categorical information to the output, recognizing a set of features. This representation might be thought as "symbolic". The difference between a localist net (as in the standard AI approach) and a hidden unit net with local units is that the latter can learn the features of the input domain that determine the input for the local hidden units.

In the second kind of local representation, hidden units are connected to a small subset of the input or output units that pick out some similar feature (e.g., they may represent a set of vowels or set of verbs) that the network has discovered can be "used" for correct categorization. This type of representation is local in the hidden layer because of clustering over some set of symbols (or features) in the input (or output) representation. It might be thought of as a "supersymbol".

In the third kind of representation, some set of hidden units is uniformly connected to some set of inputs or output units. In such a *distributed representation* a single input causes many hidden units to become activated. This kind of representation has also been referred to as "subsymbolic" (Smolensky, 1988), presumably because many hidden units are involved in coding information in the input or

the output. These features in the input has been chosen a priori and are not modifiable.

Fourth and the last, distributed representations may be created by a set of hidden units that are not fixed a priori. Such nets can encode an unknown feature representation, with each hidden unit responding to one or more aspects of features in the input representation. If the input representation is local (one unit per symbol) then the subset of hidden units can be said to be "distributing" the symbol into a set of hidden unit features.

What distinguishes connectionist representations from the other kinds of representation? Distributed models differ from localist one in which feature representations for the input are chosen a priori (as they are in AI) in that the relation between the inputs and features is made accessible to the learning rule. So the real difference between the symbolic and the connectionist approach is not in the type of representation but in the distributed representations commitment to learning. In a sense, the usual representational question has been turned on it's head: Instead of asking what the representation for a cup or a chair is, connectionists want to know under what conditions a useful representation for a cup or chair would be learned.

Figure 1 McCulloch-Pitts model

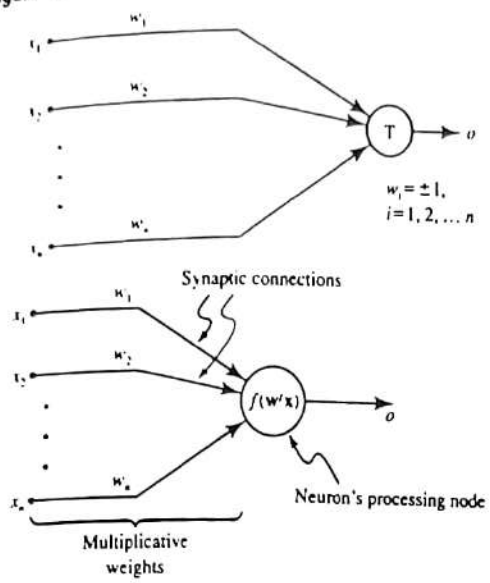


Figure 2 General symbol of neuron consisting of processing node and synaptic connections.

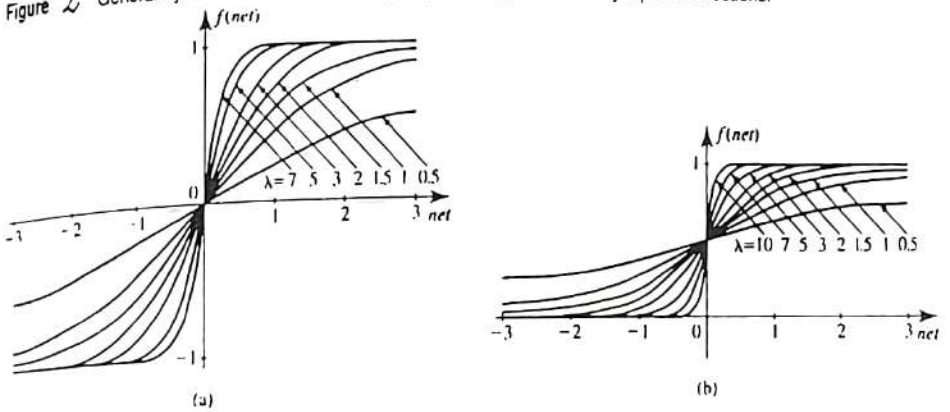


Figure 3 Activation functions of a neuron: (a) bipolar continuous and (b) unipolar continuous.

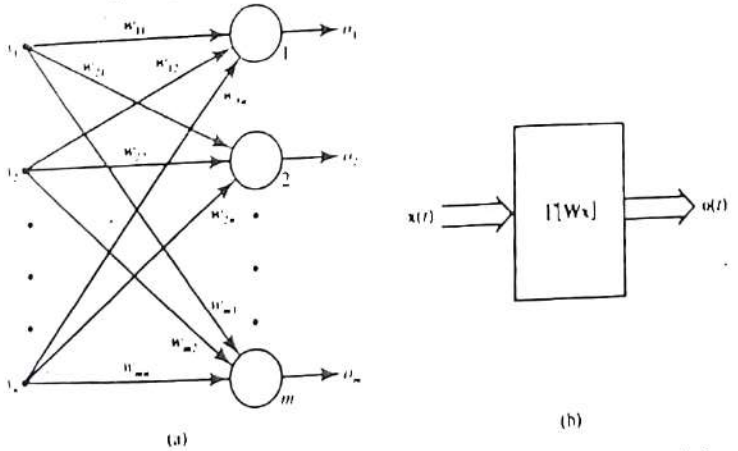


Figure 4 Single-layer feedforward network: (a) interconnection scheme and (b) block diagram.

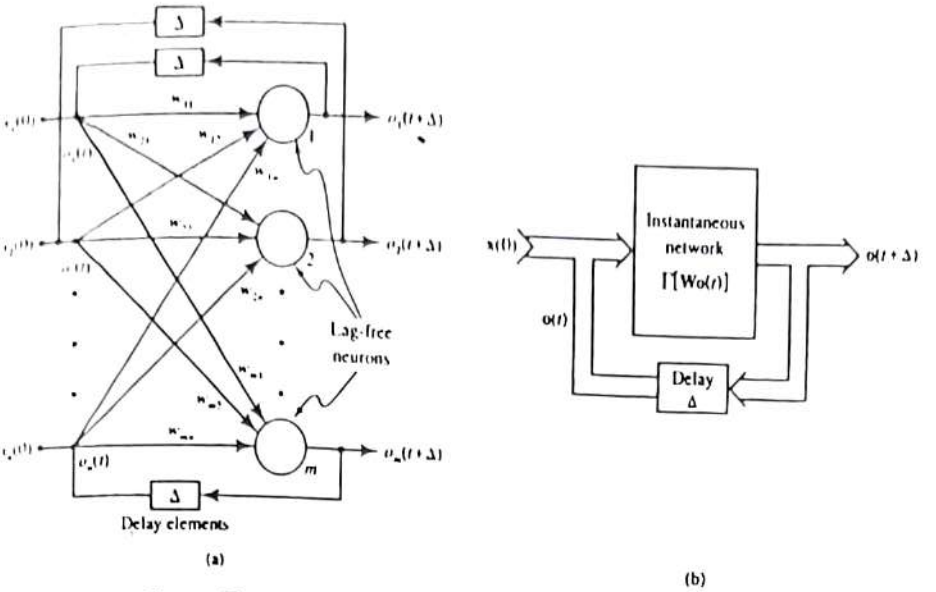


Figure 5 Single-layer discrete-time feedback network: (a) interconnection scheme and (b) block diagram.

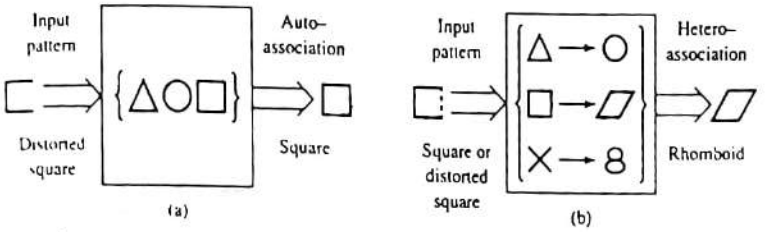


Figure 6 Association response: (a) autoassociation and (b) heteroassociation.

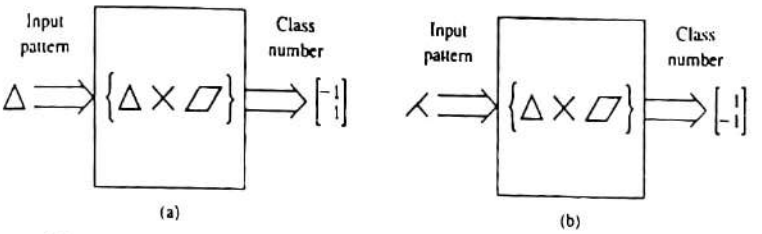


Figure 7 Classification response: (a) classification and (b) recognition.

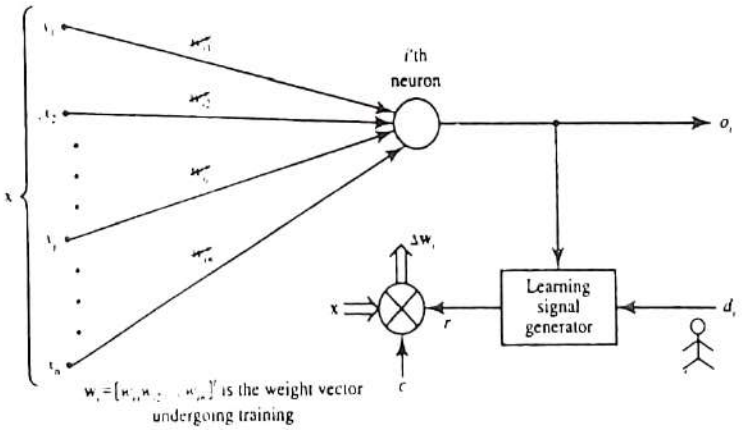


Figure 8a Illustration for weight learning rules (*d*, provided only for supervised learning mode).

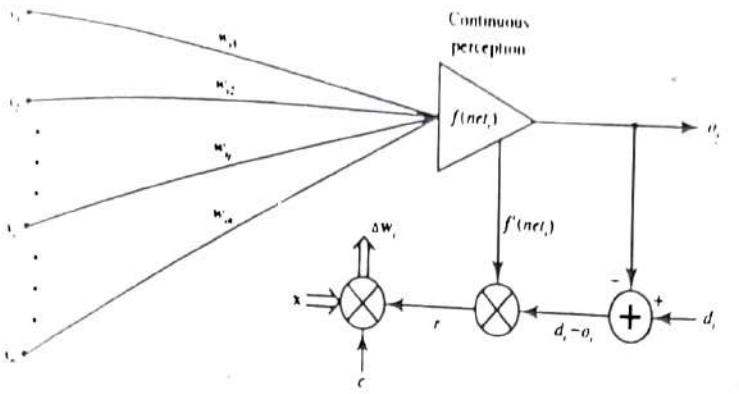
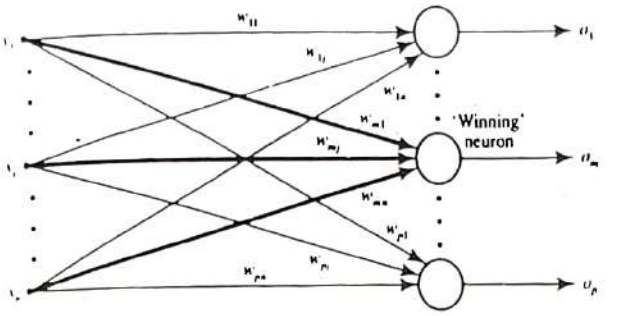


Figure 8b Delta learning rule.



(adjusted weights are highlighted)

Figure 8c Competitive unsupervised "winner-take-all" learning rule.

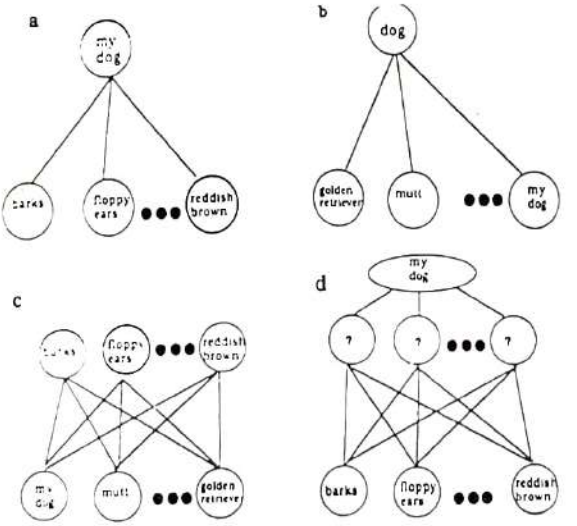


Figure 9 Examples of various sorts of local and distributed representations. (a) First example shows a completly local representation, (b) the next example is a local representation of a category, "dog", (c) the third case shows a distributed representation, where a number of hidden units can code for a single input, (d) and finally in the fourth case also shows a distributed representation, however with hidden units which can code for possibly arbitrary feature combinations in order to represent the concept.

Chapter 3

Expert System for Medical Diagnosis

3.1 Introduction

The conventional approach to building an expert system requires a human expert to formulate the IF-THEN rules by which the input data can be analyzed. This "rule-based" approach, although it provides an impressive array of tools for knowledge representation, has unfortunately not proved useful for study of learning. A learning should be able to acquire and update it's rule automatically on the basis of it's existing rules while learning to solve new problems. In practice, however, the existing rules are often unable to handle new problems, which make the entire system very "brittle". The programmer must then typically introduce new rules into the system and test their interactions thereby creating a "bottleneck". This need to keep intervening and revising on the part of the programmer does not seem to be leading in the direction of a system that can learn autonomously. The recurrent difficulties are all related to specifying the size, nature and interdependencies of the rules whenever either the task or the rules change (Langley 1983).

Among the several approaches to automated diagnosis, knowledge-based systems using Bayesian approach is also widely used. The approach based on Bayes theorem involves comparisons of relative likelihoods of different diagnosis. Symp-

toms of diseases have to be assumed to be statistically independent in the Bayesian approach. This assumption underlying the Bayes theorem, however, does not apply widely in diagnostic practice. This is because several symptoms often arise due to same organic cause.

An interesting alternative is **Connectionism** approach which consists of some old ideas about representation, spreading activation and associative memory all packaged in something that looks like a cartoon brain with cartoon neurons, connected by cartoon synapses. What would be missing here is the explanation function of the result.

3.2 Connectionist Expert System for Diagnosis

Let us take a look at the training and recall phases of error back-propagation-trained networks. Assume that a feed forward layered network is trained using the training vector pairs $(z_1, d_1), (z_2, d_2)...(z_p, d_p)$. Recognition is tested with the input being $z_i(1 \leq i \leq p)$ corrupted by noise. The network is expected to reproduce o_i at it's output in spite of the presence of noise. If no noise has been added, the network performs either a simple classification or association task.

If the trained network is tested with an input substantially different from any of the training set members, the expected response is supposed to solve the generalization problem. The generalization of the knowledge of the domain, which the network has learned during training, should cause it to respond correctly to any unseen before new input vector. Typical examples of generalization are diagnosis and prediction.

Let us take a look at how a connectionist expert system medical diagnosis can be built (Gallant 1988; Hripcsak 1988). A block diagram of an example expert system is shown in Figure 1. Input nodes take the information about the selected symptoms, test results, or a relevant medical history of the patient. If the answer to the symptom question is yes, or -1, if the answer is no. The unknown answer should thus be made 0 to eliminate the effect of the missing parameter on the

conclusion. With the I input nodes, the network is capable of handling that many binary or numerical disease or patient data. The number of identifiable diseases can be made equal to the number output nodes K.

Special care must be taken when choosing the number hidden nodes. Too low a number of hidden nodes J can cause difficulties in mapping I inputs into K outputs; too large a value for J will increase unnecessarily the learning and diagnosis times and/or cause uncertainty of the training objective. In general, for J larger than needed, weights become more difficult to estimate reliably from the training data.

Let us review an example of a connectionist expert system for fault diagnosis of an automobile engine (Marko et al). The input and output data are controlled by and monitored by an electronic engine control computer. Waveforms of analog/digital data such as those shown in Figure 11(a) are obtained first from an engine without faults. These multichannel data serve as a reference and they are used for training of the expert system modeling the fault-free operation of an engine. Figure 11(b) displays signals with a defective spark plug on cylinder 4, which causes a misfire at this cylinder in each engine cycle. Figure 11(c) displays the data from an engine with a defective fuel injector. In particular in this case, the fault can only be identified from simultaneous comparison of many traces of data. The task of defective engine diagnosis has been solved by a back-propagation trained single-hidden layer network. The connectionist expert system for engine diagnosis makes it possible to identify 26 different faults such as a shorted plug, an open plug, a broken fuel injector, etc. The training set consist of 16 sets of data for each failure, each of the sets representing a single engine cycle. A total of (16×26) data vectors with 52 elements in each vector has been used for training .

The described neural network-based expert system needs 10 minutes of training time on the NESTOR NDS-100 computer. It attains 100% fault recognition on the test data set. Low learning rates, a number hidden units equal to twice the number inputs, and a randomized presentation order within the training set have been used in the initial development phase. To improve the speed, the number of

hidden units are then decreased to less than the number of inputs. As a result, the learning time decreased five times while maintaining 100% accuracy on the test set.

3.3 Expert System for Skin Diagnosis

The expert system called DESKNET has been designed for instructions of medical students in the diagnosis of papulosquamous skin diseases. These are disease that exhibit bumpiness or scaliness of the skin (Yoon et al 1989). The developing diagnosing skills rather than for replacing the doctor's diagnosis. The expert system uses multilayer feed-forward network and employs the 96-20-10 architecture with a single hidden layer and continuous unipolar neurons.

Input for the system consists of the following skin diseases symptoms and their parameters: location, distribution, shape, arrangement, pattern, number of lesions, presence of an active border, amount of scale, elevation of papuls, color, altered pigmentation, itching, pustules, lymphadenopathy, palmar thickening, results of microscopic examination, the presence of herald patch, and the result of the dermatology test called KOH. In addition to the current symptom-generated data from the above list, the duration of skin lesions in days and weeks are also represented at the input. Inputs of values 0 and 1 are fed to the network signifying the absence or presence of the symptoms and their respective durations. The output neurons used in the local representation mode are indicative of the following 10 diseases diagnosed: psoriasis, pityriasis rubra pilaris, lichen planus, pityriasis rosea, tinea versicolor, dermatophytosis, cutaneous T-cell lymphoma, secondary syphilis, chronic contact dermatitis, and seborrheic dermatitis.

The training data from the DESKNET system consisted of input specifications of 10 model diseases collected from 250 patients. If specific symptoms or their parameters are not known, the input was coded as 0.5 to remove it's effects. The network was trained using the standard error back-propagation algorithm.

in connectionist model, accurate justification of a hypothesis, involves all nodes

of the network. If the justification of a conclusion is expected from a discussed system, analyzing the effect of a single input, or selected group of inputs, is very difficult.

However, some explanation capability exists within a trained connectionist expert system. Despite it's limitations, this capability can offer certain interpretations to the user, and also reveals those input variables that are more important and are likely to contribute to the decisions. For the local representation network, one of the output nodes is activated stronger than any of the remaining ones. Weights leading to this node with relatively large magnitudes are therefore contributing more to the decisions than the remaining weights, each considered separately. The positive and negative signs of relatively large weights can be looked at as indicative of positive and negative contributions, respectively. An additional consideration is that the hidden layer re-represents the input data and the outputs of the hidden nodes are neither symptoms nor diagnostic decisions. In a network with a single layer, internal data representation provided by the hidden layer rather than the original input data is used for generating the final decision. Internal representation can, however, offer further heuristic interpretations of the knowledge processing within the system.

In the case of the DESKNET expert system, which uses 20 hidden-layer nodes, the total of 20 hidden-layer factors attributed to the level of hidden node responses the discrimination capability for the set of 10 diseases. The factors are transmitted by the large weights connected to the strongest activated output neuron of the output layer. With reference to the Figure 3, such weights can be labeled $w_{k_I j_1}, w_{k_I j_2}, ..., wk_I j_l$, where k_I is the output node number indicating the disease diagnosed, and there are l internal factors supporting it.

An example of the explanation potential of DESKNET is shown in Figure 12. The figure shows only the weights that are of relative importance for diagnosing a dermatophytosis disease. It can be seen that, although unnamed, internal factors numbered 13 and 14 are the two strongest in causing disease number 5. Thus for this example case, we have $k_I = 5, j_1 = 13, j_2 = 14$, and l=2. Relatively large

input weights now need to be found for each of the hidden nodes $j_1, j_2, \ldots\ldots j_l$. Input nodes 1, 10, and 36 and 6, 10, 36, and 71 were identified as those that are affecting the internal factors of 13 and 14, respectively. Consequently, symptoms and parameters numbered 1, 6, 10, 36, and 71 were found to be domain importance for the diagnosis of dermatophytosis.

To determine internal decision factors, the largest and smallest weights of the output layer have been selected among all weights $w_{k_{ij}}$ forj=1.2.....J, for a given diagnosis k_I. The maximum difference between the largest and smallest weight constitutes the weight range. The range is then divided by 6 to estimate the standard deviation of the weights. The doubled the standard is used as the cutoff for discrimination of relatively large weights. For the data of Figure 4 with diagnosed disease 5, or $k_I = 5$, weights with the largest magnitudes are $w_{5,13} = 2.86, w_{5,14} = 2.71, w_{5,2} = -2.68, w_{5,6} = -3.46, w_{5,10} = -2.38$, and $w_{5,17} = -3.31$. The range for this set of weights is $w_{5,13} - w_{5,6} = 6.32$ This yields the standard deviation of approximately 1.05, and the cutoff points equal to $+2\times(1.05) = +2.1$ for the selection of relatively large weights. Therefore, only two weights, $w_{5,13}$ and $w_{5,14}$ are found to support the decision of node $k_l = 5$. The discussion indicates the presence of the following symptoms and their parameters for this particular disease: lesions of weeks of duration, minimal itching, positive KOH test, lesions on feet, minimal increase pigmentation, and microscopic evaluation for pseudohyphae. The profile produced by this methodology rather closely matches the probability profile provided by the domain expert.

It should be noted that the discussed method of explanation extraction from the expert system is rather heuristic and somewhat inconclusive. Indeed, factors influencing the decision are usually distributed within the network in a complex, superimposed way. Thus, only fairly general and somewhat tentative conclusions can be drawn from the search for the largest and smallest weights connected to the neurons with the strongest responses within the layer.

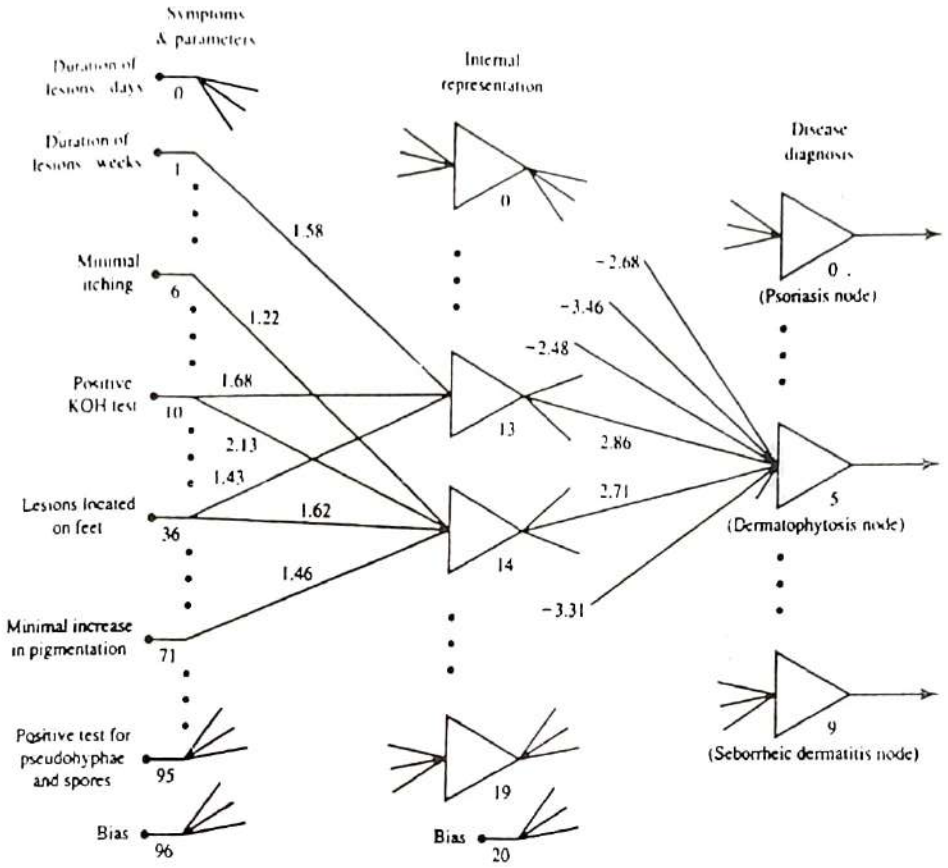


Figure 12. Explanation of dermatophytosis diagnosis using the DESKNET expert system. [Adapted from *Journal of Neural Network Computing*, New York: Auerbach Publishers, © Warren, Gorham & Lamont, Inc., used with permission, from Yoon et al. (1989).]

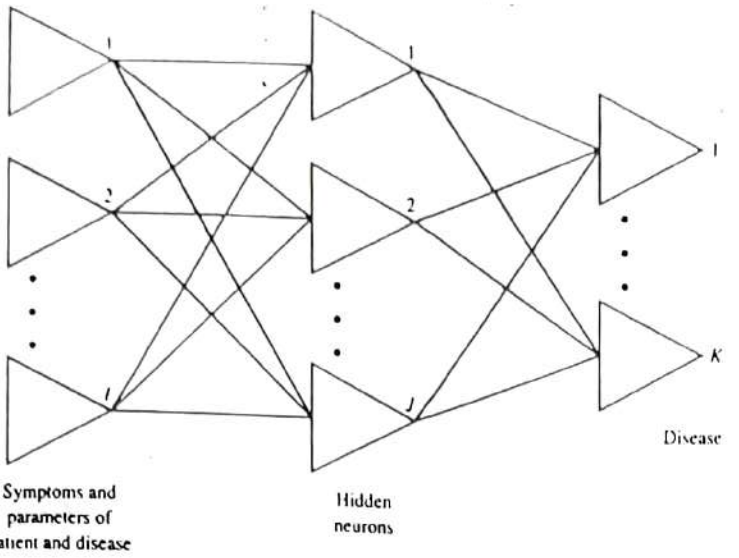


Figure 10 Connectionist expert system for diagnosis.

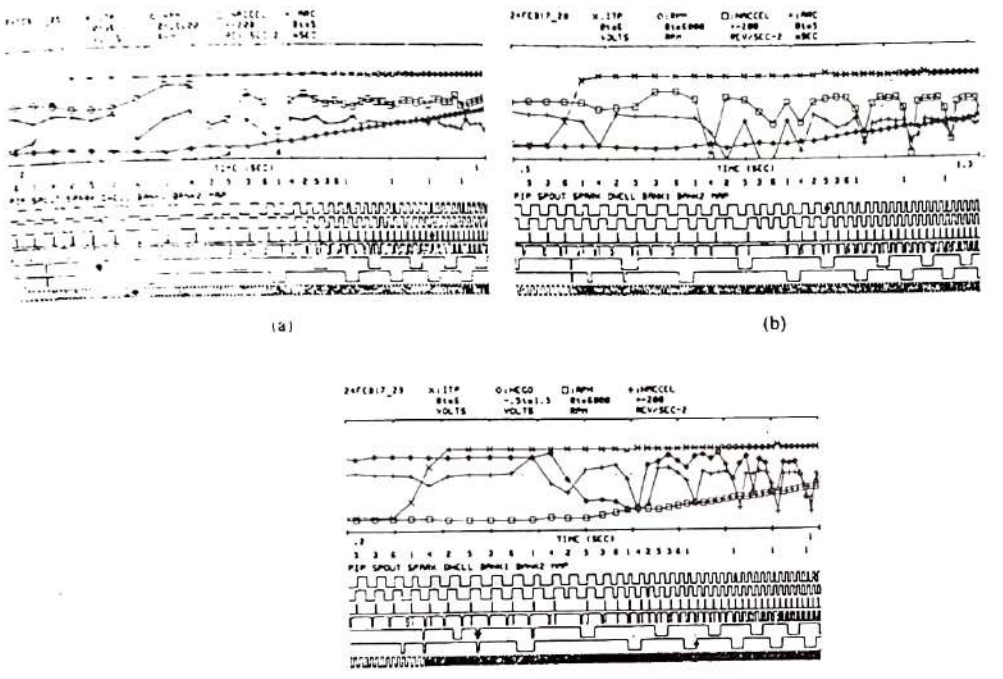


Figure 11 Automobile engine diagnostic data: (a) no-fault engine, (b) defective spark plug 4, and (c) defective fuel injector. [Source: Marko et al. (1989). © IEEE; reprinted with permission.]

Chapter 4

Pattern Recognition of Handwritten Digits

4.1 Handwritten digit recognition

4.1.1 Problem Statement

Let us look at objects that are drastically less structured than standard printed characters. In this group of problems, character recognition of handwritten signals represent an important example of a realistic, yet difficult, benchmark recognition task. Let us look at the neural network learning algorithms and approaches that first preprocess and then map handwritten digit character images to one of the ten categories. As we will see, neural network classifiers often yield comparable or better accuracy and, more importantly, require far less development time compared to conventional classifiers. Typical digit examples used for this project are shown in Figure 13. The digits are written by many different people using a great variety of sizes, writing styles, instruments, and with a widely varying amount of care. It may be noted that many of the digits are poorly formed and are hard to classify, even for a human. The captured digits are first normalized to fill an area consisting of 40×60 black and white pixels. The resulting normalized patterns are presented to a neural network after being reduced to 16×16 pixel images.

The general strategy for handwritten character processing is to extract fea-

tures from the images and then perform the classification based on the resultant feature map (Graf, Jackel, and Hubbard 1988). As a first step, a digital computer equipped with the frame-grab hardware captures a video image of a handwritten character. The computer thresholds the gray-level image into black and white pixels. This stage can be termed as image "capturing". The 16×16 scaled character is scaled both horizontally and vertically to fill a 16×16 pixel block. The 16×16 scaled character format appears to be of adequate resolution for the set of 10 digits.

4.2 Recognition Based on Handwritten Character Skeletonization

The image capturing and scaling described above is followed "skeletonization". Since the linewidth does not carries any any information about the character. the skeletonization removes pixel such that only the backbone of the character is remained. Skeletonization is shown in Figure 14(a) where the gray area represents the original digit 3, and the black area is it's skeleton. It is implemented through scanning of 5×5 pixel window template across the entire image. There are twenty of the 25-bit window templates on the neural network chip. One of them is shown in Figure 14(b). If the match of the image pixels to any of the templates exceeds the preset threshold, then the center pixel of the character bit map is deleted. In this way the entire 16×16 pixel image is scanned. The templates were crafted by network designers and examples of skeletons resulting from single scanning through the image using the selected windows are shown in Figure 15. The gray pixels shown in the figure are removed after the scanning.

It is followed by feature extraction. In this skeleton is presented to 7×7 feature extracting pixel window templates. Here also the chip stores twenty of the templates. It is shown in Figure 16. Feature extracting basically checks the presence of oriented lines, oriented line ends, and arcs in the image skeleton. Examples of the extracted features of the line end stops and horizontal lines are shown in

Figure 16(b), (c), (d). Whenever a feature template match the preset threshold, a 1 is set in a blank feature map for that corresponding feature. A feature map is produced for every 20 templates and maps of such features for every skeleton are OR*ed*. The process terminates when the 20 skeletons are mapped into 20 different 16×16 feature images. Then the feature map is compressed into 3×3 array resulting in 180 feature entries. Then these entries are classified into one of the 10 categories of digits. Several classification algorithms are used to test on digits taken from U.S post office base and the overall result gives 95% accuracy for hastily written digits and 99% accuracy for carefully written digits (Jackel et al. 1988). Note that in this approach neural network learning is replaced by using intuitively produced weight matrices associated with each window template, can be termed rather unconventional.

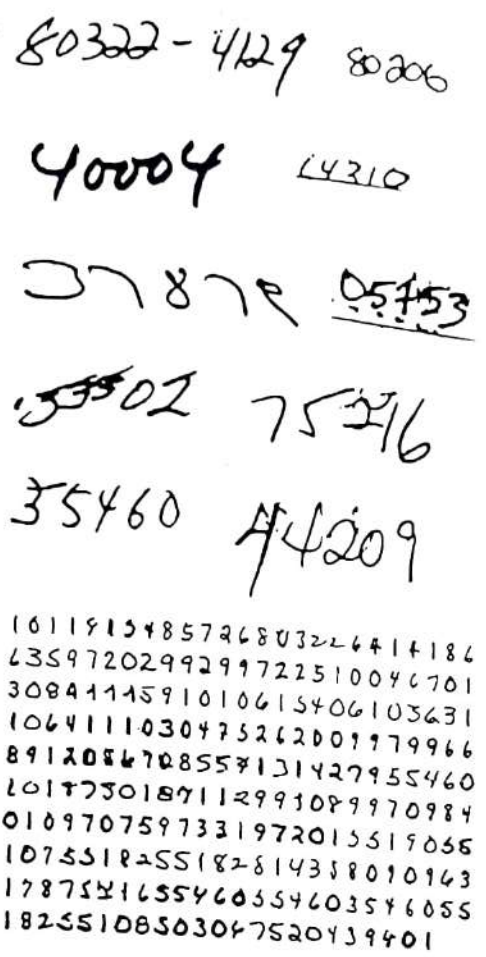
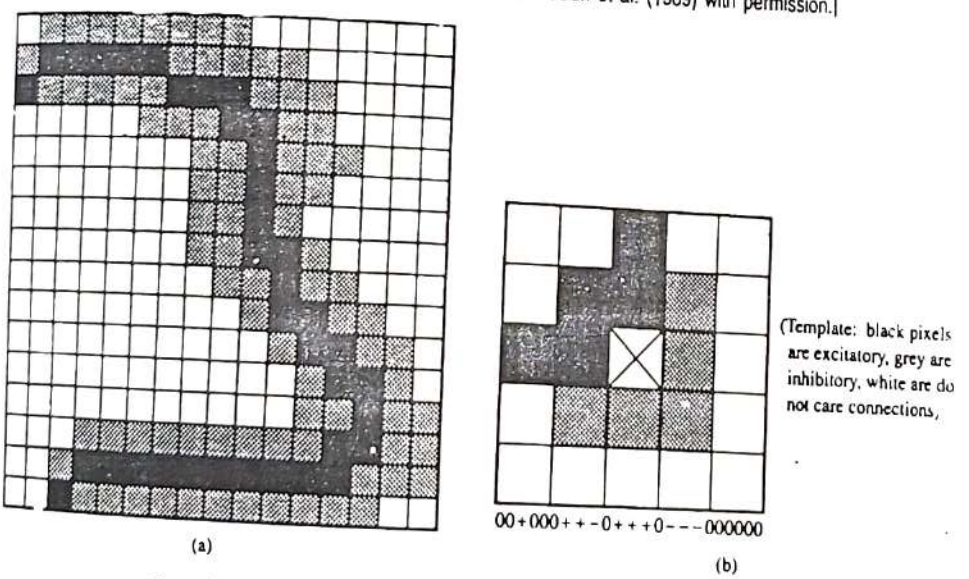


Figure 13 Examples of handwritten zip codes (top) and normalized digits from the training / test database (bottom). [© MIT Press; reprinted from LeCun et al. (1989) with permission.]



(a) (b)

(Template: black pixels are excitatory, grey are inhibitory, white are do not care connections.

Figure 14 Skeletonization of handwritten characters: (a) pixel image of digit 3 and (b) example of one of the window templates used for line thinning. [©*IEEE*; reprinted from Graf, Jackel, and Hubbard (1988) with permission]

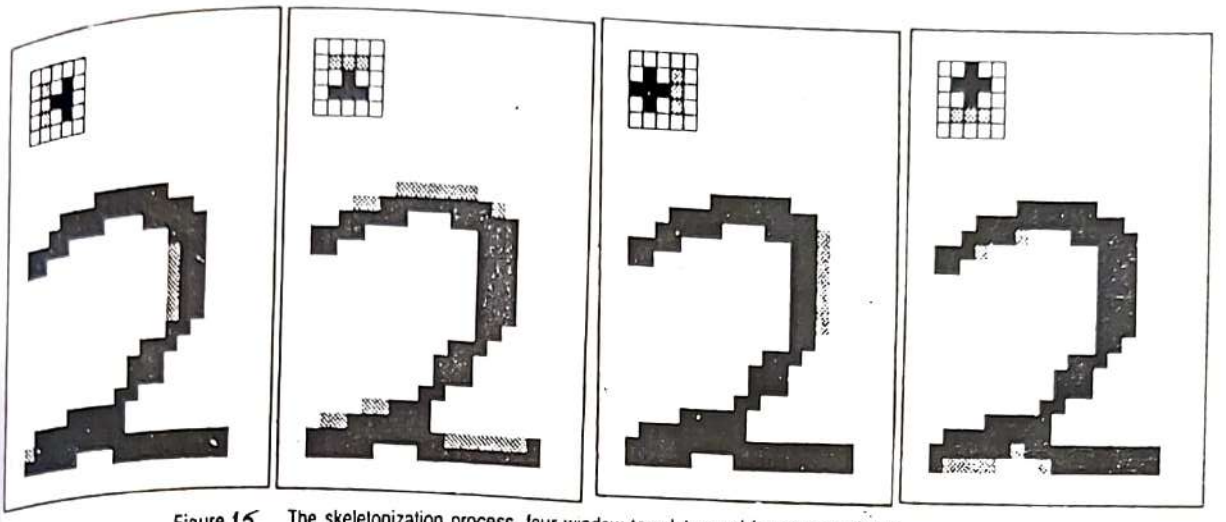
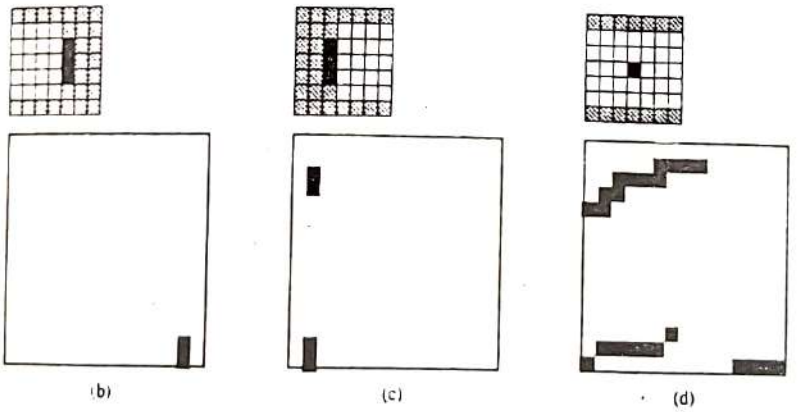


Figure 15 The skeletonization process, four window templates and four passes shown. (Templates: black pixels are excitatory, gray are inhibitory, white are do-not-care connections; digits: gray pixels are deleted by each template.) [©*IEEE*; adapted from Jackel et al. (1988) with permission.]

(a)



(b) (c) (d)

Figure 16 Examples of feature extraction: (a) skeletonized image for future extraction, (b) and (c) features of line endstops, and (d) features of horizontal lines. (Templates: black pixels are excitatory, gray are inhibitory, white are do-not-care connections.) [©*IEEE*; reprinted from Jackel et al. (1988) with permission.]

Chapter 5

Recovery of Temporal Information from Static Images of Handwriting

5.1 Introduction

The problem of off-line handwritten character recognition has eluded an adequate solution for several decades. Although researchers working in the area of on-line recognition have greater success, the possibility of extracting on-line information from static images has not been fully explored. We [2] suggest that this task requires that we break away from the traditional thresholding and thinning techniques, we will provide a framework for such analysis. We show how the temporal clues can reliably be extracted from this framework and how many of the seemingly ambiguous situations can be resolved by the derived clues and our knowledge of the writing process.

5.2 The Character Recognition Problem

The field of character recognition can be divided into two classes, off-line recognition and on-line recognition. Off-line recognition typically involves the analysis

of an intensity image obtained by scanning a prewritten sample of text. The off-line recognition of hand generated text typically exhibits a much less attractive cost/benefit ratio. It must deal with a very large set of perceptually equivalent, yet structurally different symbols whose construction is influenced by factors such as writer training, the condition of writing implement and surface, and writer's state of mind. In On-line recognition, it considers the motion of the stylus i.e. the x-y coordinates of the stylus as a function of time, and possibly related parameters such as the pressure, velocity and acceleration. The writer typically produces the strokes on an electrostatic and electromagnetic tablet and a series of coordinate pairs is recorded along with the meaningful events such as pen up or pen down. Recognition is performed on this 1-D signal (often after some preprocessing) through the use of traditional pattern matching techniques. On line recognition has produced much better results than off-line recognition. Unfortunately, the fact that the temporal information must be obtained during the writing process makes on-line recognition impractical for many practical applications.

5.3 The Temporal Recovery Problem

The claim that the increase in recognition performance is directly associated with availability of temporal information is supported by experiments in which on-line data is converted into equivalent off-line data and a deterioration in recognition performance is observed (E. Mandler, R. Oed, and W. Doster (1985)). By recovering temporal information from static handwritten images, then on-line techniques can be applied to problems such as recognition of handwritten text as well as signature verification, recognition of line drawings, and recognition of gestures. If traditional skeleton representation techniques are applied in isolation to the problem of temporal recovery, it is easily shown that many ambiguities will remain, especially in samples which include retraced strokes, loops or multiple strokes (Figure 1'). The experience of professional forensic document examiners assures us, however, that many of these ambiguities can be resolved by careful consideration

of the writing process and of local and global clues within the sample. Therefore, we [2] break away from typical document processing methodologies, and treat the extraction of on line information from static images as a partial "recovery" of isolated clues which provide evidence about the motion of the writing instrument both in time and space.

5.4 The Stroke Recovery Platform

Stroke cross sections are derived from image data and grouped into strokes based on properties such as width and intensity variation. Junctions are interpreted using knowledge of the handwriting process and a graph is built representing stroke segments, intersections, and endpoints.

5.5 Recovery of Temporal Clues by Detailed Examination

Assume stroke segments are delineated, and have tentative estimates of the path traced by the writing instrument between junctions, as described above.

5.5.1 Global Clues

The global information which is considered derived from general assumptions made about the handwriting process. They can be used to set a framework for the interpretation of the local clues.

Relative Position

If the strokes are to be interpreted as latin text, a progression from left to right across the typing surface is observed.

Effort

The general study of the handwriting process indicates that the ordering of strokes is influenced by an attempt to minimize the amount of energy required to produce them. For example, lifting the pen is conserved by retraces in script writing and

by endpoint features such as feathering and hooking.

5.5.2 Regional Clues

Feature consistency

Samples which are produced by the same author exhibit consistencies which are influenced by training and personal differences such as handedness.

Stroke properties

Other clues may include uniformity through an intersection to reveal temporal ordering (Figure 2‘) and local stroke properties, such as stroke width, to detect retraced or merged segments. These type of clues allow hypothesis to propagate between neighboring stroke segments.

5.6 Local/substroke clues

The third and most constraining class of information involves a set local clues that are available directly from the static image of the writing sample (Figure 3‘).

5.6.1 Striations

Striations are small marks in the interior of a stroke segment caused by dirt in or damage to the ball housing which prevents ink from reaching the paper. Figure 4‘ shows a stroke from figure 3‘f, the derived cross sections and the recovered striation. Along any curved stroke segment, a striation is characterized by a progression of skewed profiles, possibly including profiles with interior local minima. These profiles are tracked and position of the striation estimated.

5.6.2 End points and junction clues

Intensity variations

Ink deposits, or lack thereof, often occur at endpoints due to the fact that most of the modern writing instruments require the instrument to be in motion in order to deposit the residue evenly on the surface. Ball point pens, for example, will be

nearly void of ink on first contact with the writing surface, primarily due to the nature of fast drying inks. This feature can be used to distinguish between the start and end of a stroke, as shown in Figure 3'a and c.

Feathering

During the writing process, the continuous motion of the writing instrument may prevent it from leaving the writing surface cleanly. This produces a *feathering* effect as shown in Figures 3'a, b and c, in which the trajectory appears to fade and taper, often in a direction which is consistent with minimum energy path to the placement of the next stroke. It has been observed that that writers typically take greater care to place the instrument at the beginning of a stroke, so that feathering is typically most prominent at the point where the instrument leaves the page. When the text is produced rapidly, however, feathering has been observed at the beginnings and ends of the strokes (Figure 3').

Hooking

It has also been observed that the writer anticipate the placement of the next stroke before finishing the current stroke. By doing this, we find the instrument may actually begin to move in the direction of the next stroke before leaving the page, thus producing a hooking effect. Figure 5' as well as many earlier figures provide the examples of hooking effect. Figure 6' also shows how the ordering of strokes can be disambiguated by considering the light starting point, hook and right-left heuristics. We also observe that as with feathering, hooking is typically more pronounced at the end of a stroke than at the beginning.

5.7 Local segment analysis

To examine the ends of a stroke segment, we replace the computed scan lines emanating from each contour pixel with pseudo scan lines which minimize the local curvature (Figure 7'a). Noting that the original scan lines are precise only along the isolated segments and a smooth path can be generated across the junction and cross sections approximated (figure 7'). For more complex junctions, such as

merges or high curvature points, we can take the advantage of the stability of the outer counters entering the junction.

5.8 Pressures, velocities and acceleration

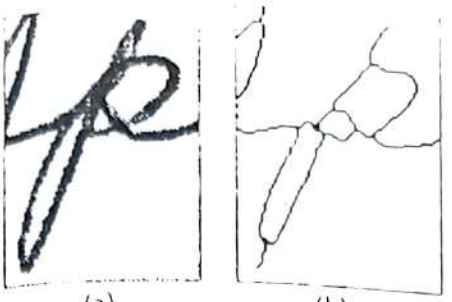
If we consider the typical grasp of a writing instrument, we note that the instrument is inclined towards the writer, and orthogonal to the writing direction. With this configuration, we find that a greater normal force is exerted on the writing surface when the instrument is being pulled downward during a downstroke than it is being pushed upward during an upstroke. It is observed that the average stroke width and intensity vary with the normal force generated by the pen, but appear to vary little with moderate changes in velocity or acceleration, as long as the pressure remains constant. By comparing segment pairs in long strokes such as l's, the upstroke and downstroke are distinguished as in Figure 8.

5.9 System control

The system allow varying amounts of the emphasis to be placed on the clues depending on the nature of the domain. A table of properties and clues, and associated reliability and emphasis weights will be provided a priori. It also allows the emphasis weights to be set or modified during the analysis.

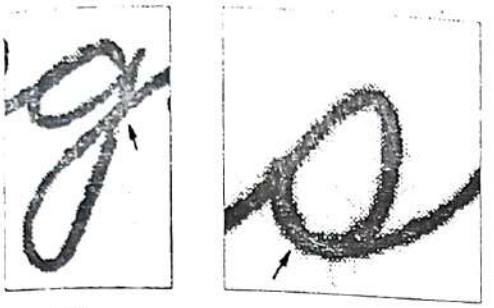
5.10 Result

Figure 9 shows a hand-printed zip code. For the "9", a striation at the top indicates the direction of the loop, and the direction of the stem is in continuation. For the "6", the light start point is located and indicates the correct direction. The "2" is correctly identified by propagating the ending hook back through the stroke and the "3" is disambiguated by the visible striation along the last stroke segment. No interpretation is make for the "0" although a hook is visible to the human observer.



(a) (b)

Figure 1: Ambiguous temporal orderings are produced if only "good continuation" properties are considered. Much of the information necessary to derive a correct ordering is destroyed by standard skeletonization.



(a) (b)

Figure 2: (a) shows the gray level consistency through an intersection, and in (b) the slight difference in width along a retraced stroke is visible.

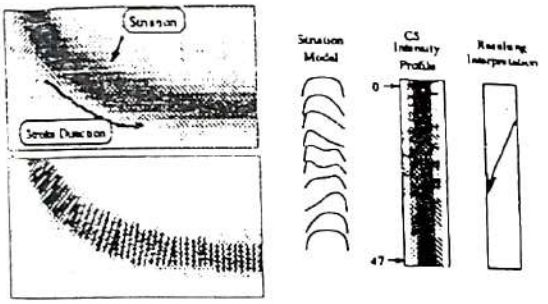
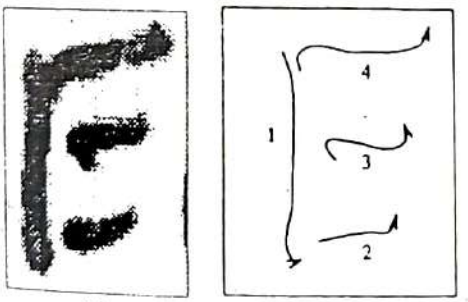
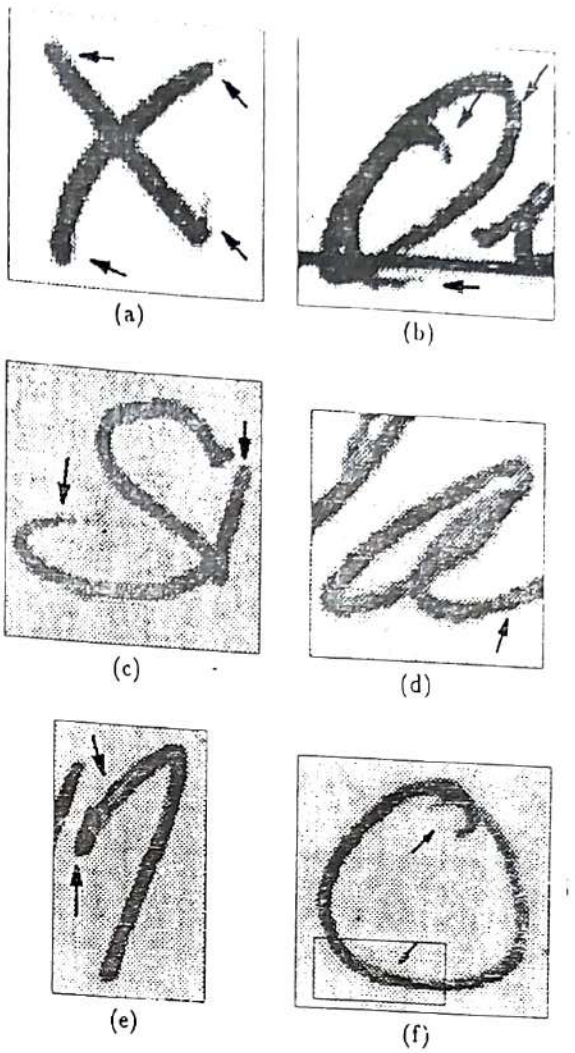


Figure 4: The cross sections computed from a subimage of Figure 3f and the resulting interpretation



(a) (b)

Figure 6: An ambiguous E and its correct interpretation based on endpoint information



(a) (b) (c) (d) (e) (f)

Figure 3: Samples of handwritten text: Clues are provided by the endpoints in (a), (b), (c) and (e); the striations in (b), (d), (e) and (f); and the partial retraces which are evident in (b), (d) and (e).

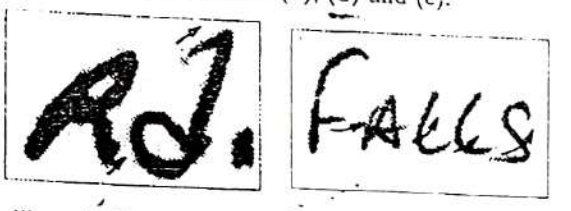
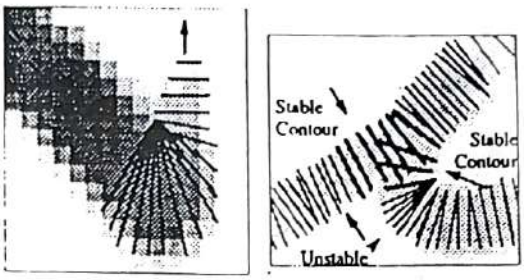
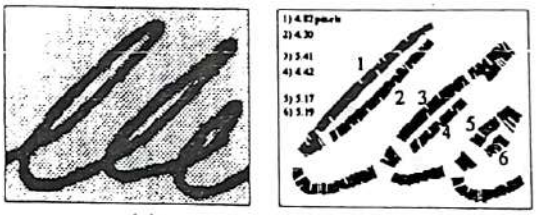


Figure 5: Examples showing feathering and hooking



(a) (b)

Figure 7: a) The pseudo scan lines generated for an endpoint in Figure 3a. b) Light cross sections are provided automatically during initial analysis, dark cross section are generated from the approaching contours.



(a) (b)

Figure 8: Relative forces: (a) The original image. (b) The recovered strokes. The cross sections of the long downstrokes range from 10% to 20% wider then those of the corresponding upstrokes.

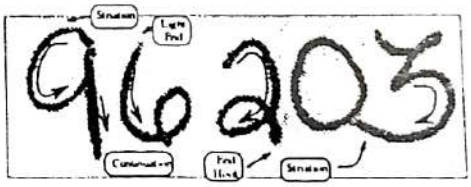


Figure 9: A hand-printed sample with the derived local clues labeled and resulting temporal interpretations indicated by the arrows.

Chapter 6

Conclusion

In spite of great progress, some of the fundamental principles of human speech recognition are still an enigma. The prodigious feats of a human who can pick out and understand in noisy environment a speaker of his native tongue cannot be mimicked by any machine at present. In visual perception the abilities of some humans are even more enigmatic. Take for instance the art of cartoonist, who with a few strokes can portray a faithful image of a face that all of us can immediately recognize. While the reader can follow my thoughts, his attention can easily wonder to many other thoughts in rapid succession, and return to this article. It is this rapid change of thoughts and mental processes that is so alien to algorithms and machines. However, one expects, that the pattern recognition ability coupled with, and feeding back and forth to rule-based systems will finally result in machines that share our ability to learn and duplicate our processes of reasoning machines that might be said to think. And, just as the 20^{th} century is the century of automobiles, airplanes, telephones and computers, the 21^{st} will be the century of intelligent machines. We will not only learn to live with these machines, but, indeed, will wonder, one day, how we ever lived without them.

REFERENCES

[1]. Antognettit P. and Milutinovic V., *Neural Networks Concepts, Applications, and Implementations*, Parentice Hall, New Jersey, 1991.

[2]. Doermann, D. S. and Rosenfield, A., "Recovery of Temporal Information from Static Images of Handwriting", In *Proc. of IEEE Intl. Conf on Comp. Vis. and Pat. Rec.*, pages 162-168, 1992.

[3]. Hanson, S. J. and Bure, D. J., "Learning and Representation in Connectionist Networks", Neural Networks Theory and Apllications, Vol. 1, pages 169-208, 1989.

[4]Hassoun, M. H., *Fundamentals of Artificial Neural Networks*, Parentice Hall of India, New Delhi, pages 1-30, 1995.

[5]. Julesz, B., "Early Vision, Focal Attentions, and Neural Nets", Neural Networks-Theory and Applications, Vol 1, pages 209-216, 1990.

[6]. Pan, J. C. and Lee S., "Off-line tracing and representation of signatures". In *Proceedings of Computer Vision and Pattern Recognition*, pages 679-680, 1991.

[7]. Swain, A. K and Subudhi B., " Neural Network and Intelligent Control". *EFY*, Vol. 28, pages 72-80, Oct. 1969.

[8]. Zurada, M. J., *Introduction to Artificial Neural Systems*, Jaico Publishing House, Bombay, 1997.