**İSTANBUL TECHNICAL UNIVERSITY ★ INSTITUTE OF SCIENCE AND TECHNOLOGY**

# A REAL-TIME OPTICAL CHARACTER RECOGNITON SYSTEM

**M.Sc. Thesis by**
**Tolga OVATMAN, B.Sc.**
**(504031527)**

| | | |
|---|---|---|
| **Date of Submission** | **:** | **09.05.2005** |
| **Date of defence examination** | **:** | **02.06.2005** |
| **Supervisor (Chairman)** | **:** | **Asst. Prof. Dr. Osman Kaan EROL** |
| **Members of the examining committee** | **:** | **Assoc. Prof. Dr. Coşkun SÖNMEZ** |
| | | **Asst. Prof. Dr. Işın ERER** |

**JUNE 2005**

## PREFACE

First of all, I would like to thank my supervisor Asst. Prof. Dr. Osman Kaan Erol for his support and encouragement. I would like to thank Fatih Kahraman and Binnur Kurt for their uninterrupted and patient helpful behavior. I would like to thank my research assistant colleagues and room mates for their support. I would like to thank all my friends for their confidence in me and dormitory room mates for bringing me the spirit of determination. I would like to thank all my teachers for teaching me to this point. Finally, with all my respect and love, I would like to thank my family for making me what I am.

June 2005 Tolga OVATMAN

**TABLE OF CONTENTS**

**ABBREVIATIONS**

**ADALINE** : Adaptive Linear Element
**AFSA** : Armed Forces Security Agency
**ANN** : Artificial Neural Network
**API** : Application Programming Interface
**ART** : Adaptive Resonance Theory
**ASCII** : American Standard Code for Information Interchange
**CTS** : Clear To Send
**GA** : Genetic Algorithm
**GUI** : Graphical User Interface
**IBM** : International Business Machines Corporation
**IMR** : Intelligent Machines Research Corporation
**L1** : Layer 1
**L2** : Layer 2
**LMS** : Least Mean Square
**LVQ** : Learning Vector Quantization
**MLP** : Multilayer Perceptron
**MSE** : Mean Squared Error
**NSA** : National Security Agency
**OCR** : Optical Character Recognition
**OCV** : Optical Character Verification
**PC** : Perconal Computer
**PCA** : Principle Component Analysis
**ROI** : Region of Interest
**SGA** : Simple Genetic Algorithm
**SOM** : Self Organizing Map
**USB** : Universal Serial Bus
**XOR** : Exclusive 'OR'

## LIST OF TABLES

## LIST OF FIGURES

## LIST OF SYMBOLS

| | |
|---|---|
| $_{i}\mathbf{w}$ | : Weight vector of $i^{th}$ layer |
| $\mathbf{w_{ij}}$ | : Weight value of the connection from neuron j to neuron i |
| $\mathbf{y_i}$ | : Output of the $i^{th}$ neuron at output layer |
| $\mathbf{a^i}$ , $\mathbf{a_i}$ | : Output from layer i |
| $\mathbf{p_i}$ | : $i^{th}$ input |
| **a(t)** | : Output of neuron at time t |
| **p(t)** | : Input to neuron at time t |
| **n** | : Input to the neurons transfer function |
| **n(t)** | : Input to the neurons transfer function at time t |
| $\mathbf{t_i}$ | : Target for the $i^{th}$ output |
| $\rho$ | : Vigiliance parameter of ART |
| $\mathbf{W^{i:j}}$ | : Weight vector between layer i and layer j |
| $\mathbf{b_i}$ | : Bias of the $I^{th}$ neuron |
| **e** | : Total error value |
| $_{i}\mathbf{e}$ | : Error value for the $I^{th}$ layer |
| **x** | : Connection weight-bias weight combined weight vector of a neuron |
| **z** | : Normal input-bias input combined input vector of a neuron |
| $\wedge$ | : Approximation of a value or a function |
| $\nabla$ | : Gradient of a function |
| $\Delta$ | : Difference of a value |
| **E[ ]** | : Expected value |
| $\mathbf{v^T}$ | : Transpose of vector v |
| **s** | : Sensitivity value |
| $_{i}\mathbf{s}$ | : Sensitivity value for layer i |
| $\varepsilon$ | : Signal decay rate parameter |
| $\mathbf{p^+}$ | : 'p' as an excitatory input |
| $\mathbf{p^-}$ | : 'p' as an inhibitory input |
| $\mathbf{I^0}$ | : Unnormalized inputs to ART |
| **I** | : Normalized inputs to ART |
| $F_\theta(x)$ | : Thresholding function |
| **N(x)** | : Normalizing function |
| $\alpha$ | : Learning rate |
| $\mathbf{i^*}$ | : Winning neuron |
| $\|\ \|$ | : Norm of a vector |
| $^{m}\mathbf{O}$ | : Chromosome 'O' is mutated |
| **P(t)** | : Population at time t |
| **w** | : Computational cost of weighting an input to a neuron |
| **n** | : Computational cost of a normal neuron operation (weights ignored) |
| **sn** | : Computational cost of a shunting neuron operation (weights ignored) |
| **cn** | : Computational cost of a competitive neuron operation (weights ign.) |
| **~** | : Approximate value |
| $\tau$ | : Time variable |

# GERÇEK ZAMANLI BİR OPTİK KARAKTER TANIMA SİSTEMİ

## ÖZET

Optik karakter tanıma sistemleri endüstriyel ve/veya günlük hayatta kullanılan birçok akıllı sistemden biridir. Basım endüstrisi, optik karakter tanıma sistemlerinin kalite kontrolü ve geri bildirimi amacıyla kullanıldığı, bahsedilen alanlardan biridir. Hazırlanan tez çalışması basım endüstrisinde kullanılacak bir optik karakter tanıma ve doğrulama sisteminin tasarım, geliştirme ve performans optimizasyonu çalışmaları üzerine eğilmektedir.

Optik karakter tanıma konusu ABD'de 1950'li yılların başında geliştirilmiş ve belge sayısallaştırma amaçları için kullanılmaya başlanmıştır. Optik karakter tanımayı günlük hayatta kullanmak ülkemizde, 1960-1970'li yıllardan beri posta servislerinde bu tür sistemleri kullanan yabancı birçok ülkeye göre pek yaygın bir konu değildir.

Optik karakter tanıma aslında dış dünyadan bir nakledici ile alınan bilgilerin işlenerek sayısal veriye dönüştürüldüğü bir örüntü tanıma işlemidir. İşleme süreci tanıma operasyonuna hazırlık amaçlı yürütülen görüntü işleme ve bölütleme operasyonlarını ve birçok yapay zeka yönteminin kullanıldığı tanıma operasyonunu içerir. Yapay sinir ağları optik karakter tanıma sistemlerinin tanıma motorlarında en yaygın kullanılan yöntemlerden biridir.

Tez çalışmaları dahilinde hazırlanan sistem zamanın kritik ölçüt olduğu bir basım otomasyon sisteminde kalite kontrolünde kullanılmak üzere tasarlanmıştır. Söz konusu otomasyon sisteminde mekanik taşıyıcı sistem ile dijital basım sistemi arasında oluşan senkronizasyon yuşmazlıklarından dolayı zaman zaman hatalı sonuçlar doğabilmektedir. Hatalı basılan ürünleri ayıklamak için otomasyon sistemine bir kontrol sisteminin gömülmesi kaçınılmazdır. Bu kontrol sistemi basılan sadece rakamlar içeren veriyi okumalı ve elde bulunan verilerle uyumunu kontrol etmelidir.

Otomasyon sisteminin çalışma hızı nedeniyle bu amaçla bir sayısal sistemin geliştirilmesi kaçınılmazdır. Bu noktada optik karakter tanıma sürece dahil edilerek sayısal kameraların ve senkronizasyon üniteleri yardımıyla bir optik karakter tanıma sistemi kullanılan otomasyon sistemine entegre edilmiştir. Optik karakter tanıma sistemi basılan ortam sürekli çıktılar ürettiğinden gerçek zamanlı çalışmak zorundadır. Bütün bu sebeplerden ötürü optik karakter doğrulama sisteminin karakter tanıma motoru öngörülen bir başarım oranının altına düşmeyerek saniyede yüzlerce karakter tanıyacak şekilde çalışması için optimize edilmelidir.

Yapay sinir ağları sistemin yüksek başarım oranları ve esnek yapıları nedeniyle karakter tanıma motorunda kullanılmak üzere tercih edilmiştir. Optik karakter doğrulama sisteminin ihtiyacı olan performans-başarım oranı sistemin karakter tanıma motorunda kullanılan yapay sinir ağları için bilinen bir ikilemdir. Bu ikilemi aşmak amacıyla her sınıflayıcı yapay sinir ağı türünden bir örnek seçilerek

gerçekleştirilmiş ve soruna ne derece çözüm getirdiği incelenmiştir. Tez çalışması boyunca incelenen her yapay sinir ağı ayrıca başka kullanım alanlarda öne çıkan kendine has özellikler de taşır. Ayrıca yenilikçi bir çözüm olarak evrimsel hesaplama yöntemleri de en iyi performansı sağlayacak şekilde budanarak özel olarak tasarlanmış bir yapay sinir ağının eğitiminde kullanılmıştır.

Tez çalışmalarının deneysel kısmında söz edilen yöntemler gerçekleştirilmiş ve gerçek sistemde test edilerek performansları doğrulanmıştır. Deneylerin sonuçları sistem için önemli olan başarım oranı ve hesap maliyeti açısından incelenmiştir. Böylece her bölümde gerçeklenen yapay sinir ağı yapısı ve kabul ettiği girdi türüyle takdim edilmiş ve başarım oranıyla hesap maliyeti tartışılarak avantaj ve dezavantajları kullanılması olası durumları öngörme amacıyla ortaya konmuştur.

Son olarak yapılan deneysel çalışmalarda kullanılan yapay sinir ağları ve yöntemler arasından tasarlanan sistemde kullanılmaya en elverişli olanına ulaşmak amacıyla bir karşılaştırma yapılmıştır. Bunun ötesinde, sistem yenilemeleri veya gelecekte yapılabilecek olası çalışmalar tartışılarak tercih edilmeyen yöntemlerin etkin olarak kullanılabileceği durumlar ortaya konmuştur. Tez çalışmasının sonunda sistemde kullanılmaya uygun birçok yapay sinir ağı sınıflayıcısı ile çalışılmış ve bir gerçek zaman sisteminin karakter tanıma motorundaki yerleri üzerine fikir yürütülmüştür.

## A REAL-TIME OPTICAL CHARACTER RECOGNITION SYSTEM

## SUMMARY

Optical character recognition systems are one of many intelligent systems which are used in industry and/or daily work. Printing industry is one of these areas where OCR systems are used for quality control and feedback. The thesis is mainly about the studies on designing, developing, performance optimization of a real-time optical character recognition and verification system which is going to be used for practical issues in printing industry.

The subject of OCR itself is developed in the beginning of 1950s in USA and is still being used widely in many areas as a tool for document digitalization. Embedding OCR to daily work is not a common topic in our country with respect to other countries where OCR systems is being used in postal services since 1960s-1970s.

OCR is actually a pattern recognition task where the images that are recieved from the real world by the help of a transducer are processed and transformed into the digital data. The mentioned process consist image processing and segmentation in order to prepare the image for the recognition process and the recognition process where various artificial intelligence methods are used. Artificial neural networks are one of the most common methods used for recognition engines of the OCR systems.

The prepared system is actually a solution for the need of a quality control system in a printing automation system where time is the critic measure. In the automation system there may exist printing problems based on the synchronization problem between the mechanical conveyor system and the digital printing system. In order to eliminate the erroneous printed media there must be a control mechanism embedded to automation system. This control system ought to read the printed data, which are only numeric in this case, and control it with the data in hand to detect an erroneous operation.

Artificial neural networks are chosen to be used in the character recognition engine of the system because of their high success rates and flexible structures. Because of the automation system's working speed a digital system is mandatory to be developed. At this point OCR comes into action where digital cameras and synchronization units are used to embed the OCR system to the present automation system. The OCR system has to work real time where the printing media continously flowing thus a batch processing system is not probable. Because of these reasons the character recognition engine of the optical character verification system must be optimized to recognize hundreds of characters in a second and do not perform below a predetermined value of success rate.

The performance-success rate that the OCV system suffers from is a well-known dilemma for artificial neural networks which are used in the character recognition engine. In order to overcome this dilemma a neural network from each type of

classifier neural networks is selected and realized in order to investigate how much it brings a solution to the problem. The neural networks that are experimented through the thesis study also has different authentic properties which brings them forward in another area for usage. Also an innovating concept, evolutionary computation, is used to train a specially designed network which is optimized for providing the best performance by pruning.

In the experimental part of the thesis these mentioned concepts are realized and their performance is verified by being experimented in the real world system. The results of the experiments are examined in a manner that considers the two important subjects about the neural network, which are the success rate and the computational cost of the neural network. So in each part the realized neural network is introduced by its structure and the inputs that it accepts, and then the sucees rate and the computational cost of the network is argued to reveal the advantages and disadvantages of the neural network and foresee the situation that it might be suitable to use them.

Finally at the final part the comparison between the experimented neural networks and methods is introduced and the most appropriate method choice to be used in current is system is argued. Beyond this, an update to system or future work is brought up in order to use the each remaning method effectively with the system. By the end of the thesis study most of the possible neural classifiers have been experimented and their place in the recognition engine of a real-time system has been argued.

## 1. INTRODUCTION

Almost all of the artificial intelligence concepts are based on biological inspirations. Artificial immune systems model the human immunity system where genetic algorithms work by modelling the evolutionary concepts introduced by Charles Darwin. The artificial neural networks are an example of this concept since they model the neural system of the human body. Even the hardware that is used in intelligent systems is based on biological models like a digital camera which represents a part of the human eye.

Biology is not an awkward source for inspiration since the main object of artificial intelligence is building systems that can replace the manpower or supply manpower where the human abilities are somehow insufficient. The starting point of an effort to fullfill this need is the modeling of the current well working systems which are the human body and biological facts for sure.

On a time where the computational abilities of the computers are growing with exponential rates, the expectations of the industry from the computer systems are changing from the classical application software systems that can be used in service duties to intelligent systems which can replace manpower.

Printing industry is one of these industries that need the artificial intelligence concept in order to increase their throughput. Quality control and verification of the printed media is an important issue for printing industry. The synchroniztion problem between the mechanical automation system and the digital printing system sometimes yields to unwanted results that affect the quality of the work and decrease the throughput of the system.

Optical character recognition (OCR) systems are used in printing industry to automatically and precisely control the printed media and help to eliminate the affected products of the printing system. This is often done real time where the production rate is important. The real time OCR concept comes from the limited time

where the printed media should be digitalized immediately before automation system presents new inputs to OCR system.

The studies that are performed in this thesis consist of two parts, when classified by the type of studies. Before beginning to introduce the thesis studies general information about OCR and pattern recognition is introduced in Chapter 2.

The first part of the thesis studies is the realization phase of the system where a real-time optical character recognition and verification system is designed and implemented by examining the current system that is used in the printing house and the defects of the current system. After these defects and the limitations of the system is examined, the new system design is introduced including the choices for hardware and software infrastructure of the system and why they have been chosen.

The first part is relatively superficial since it is the infrastructure for the experimental part of the thesis. In Chapter 3 the system that has been designed is introduced. And as a complementary part between the two parts, the infrastructure of the recognition subsystem that is supplied by the designed system is examined in Chapter 4.

The second part of the studies investigates ways to reduce the processing time and increase the success rate of the recognition system by experimenting different artificial neural network approaches in the implemented real-time OCR system. Before the application of these approaches, each approach is examined in theroy in Chapter 5 and the main reasons behind using these approaches is explained.

Finally in Chapter 6 the experimental studies that are peformed in order to increase the throughput of the system are explained in detail. The inputs and results are proven to be vaild for practice rather then sticking to theory, since the system that is used for experimentation is a real system.

In the final chapter the results that are obtained through the thesis study are debated and some ideas for the future work that can be performed to optimize the system are introduced.

Looking at the big picture, the thesis is mainly about designing, implementing and optimizing the performance of an intelligent system which uses artificial neural networks to perform the pattern classification over a predetermined set of patterns. Designing and developing a real-time intelligent system considering all the software and hardware requirements make the thesis work worthy for practical issues while

the implementation and experimentation of different neural network approaches in a real world pattern recognition environment make it worthy for theoretical issues. Combination of these efforts hopefully reveals a remarkable engineering thesis study.

## 2. OPTICAL CHARACTER RECOGNITON

The studies that are introduced in the thesis are performed around the topic of OCR and pattern recognition. At this point it is necessary to examine the fundamentals of OCR and pattern recognition in order to build a priori knowledge for the thesis.

### 2.1 History of OCR

Optical character recognition, usually abbreviated as OCR, involves computer systems designed to translate images of typewritten text into machine-editable text or to translate pictures of characters into a standard encoding scheme representing them (ASCII or Unicode). OCR began as a field of research in artificial intelligence and machine vision.

Optical character recognition (using optical techniques such as mirrors and lenses) and digital character recognition (using scanners and computer algorithms) were originally considered separate fields. Since very few applications survive that use true optical techniques the optical character recognition term has now been broadened to cover digital character recognition as well.

In 1950, David Shepard, a cryptanalyst at AFSA, the forerunner of the United States National Security Agency (NSA), was asked by Frank Rowlett, to work with Dr. Louis Tordella to recommend data automation procedures for the Agency. This included the problem of converting printed messages into machine language for computer processing. Shepard decided it must be possible to build a machine to do this, and, with the help of Harvey Cook, built "Gismo" in his attic during evenings and weekends. Shepard then founded Intelligent Machines Research Corporation (IMR), which went on to deliver the world's first several OCR systems used in commercial operation. While both Gismo and the later IMR systems used image analysis, as opposed to character matching, and could accept some font variation, Gismo was limited to reasonably close vertical registration, whereas the following commercial IMR scanners analyzed characters anywhere in the scanned field, a practical necessity on real world documents. The first commercial system was

installed at the Readers Digest in 1955, which, many years later, was donated by Readers Digest to the Smithsonian, where it was put on display. The second system was sold to the Standard Oil Company of California for reading credit card imprints for billing purposes, with many more systems sold to other oil companies. Other systems sold by IMR during the late 1950's were a bill stub reader to the Ohio Bell Telephone Company and a page scanner to the U.S. Air Force for reading and transmitting by teletype typewritten messages. IBM and others were later licensed on Shepard's OCR patents.

The United States Postal Service has been using OCR machines to sort mail since 1965 based on technology devised primarily by the prolific inventor Jacob Rabinow. Canada Post has been using OCR systems since 1971. OCR systems read the name and address of the addressee at the first mechanized sorting center, and print a routing bar code on the envelope based on the postal code. After that the letters need only be sorted at later centers by less expensive sorters which need only read the bar code. To avoid interference with the human-readable address field which can be located anywhere on the letter, special ink is used that is clearly visible under UV light. This ink looks orange in normal lighting conditions. Envelopes marked with the machine readable bar code may then be processed.

### 2.2 How OCR works

Character recognition systems are basically pattern recognition systems which operate on alphanumeric data. Thus, examining the basics of a typical pattern recognition system leads to understanding character recognition.

Most of the times, it is easy for a human to recognize a printed number or letter, if the letter is not badly deformed. But if it is tried to be modeled with a digital system it can easily be seen that the task is much harder than it seems. The process of recognition has several intermediate stages starting with a real image from a world and ending with a decision of class membership to which the image belongs. Each sub-stage has its own problems and a bad approach to any of them reduces the success of the recognition to the level of the erroneous stage.

A detailed decomposition of a pattern recognition problem is shown in Figure 2.1 [34]. In order for the system to analyze the real world scene, the system usually uses some form of a transducer which is generally a camera or a scanner. The camera

yields a two dimensional array of numbers each representing the quantized amount of light or brightness of the real world scene. After the image has been obtained from the transducer the first computational step arrives which consists of segmenting the image into meaningful objects. Secondly the noise and/or deformation on the image is removed as much as possible. The next step is feature extraction which consist selection of important features that represents the data. Final stage is concerned with classification of the data obtained from the image into one or more categories. It is meaningful now to consider these steps in more detail.

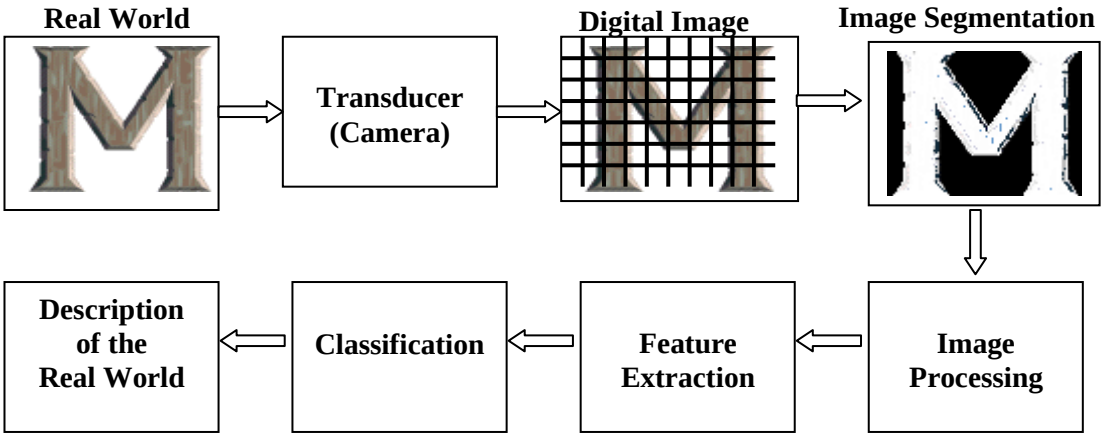



**Figure 2.1** Sub-stages of pattern recognition

***Transducers:***

A transducer is a transformer between the real world and the computer world. It is usually a sensor or array of sensors that measures the amount of light and/or colors on a specified location and transforms it into a digital pattern called an image. An image holds a matrix of numbers that represent the image. The atomic unit of an image is pixel. Pixels hold the value of an atomic region of the image by quantizing the analog data that has been measured. The quantization precision determines the quality of the image. The transducer is generally presented with the problem, in which the patterns that are acquired by a camera, scanner or maybe a photocell are presented to recognition system.

***Preprocessing:***

Preprocessing stands for a family of procedures for smoothing, enhancing, filtering, cleaning-up and normalizing a digital image so that succeeding algorithms can be applied to image more accurately and simply. A wide range of filters and methods are present for image processing and choosing one of them is problem specific.

***Feature Extraction:***

Feature extraction is the name given to a family of procedures or measuring the relevant shape information contained in a pattern so that the task of classifying the pattern is made easily by a formal procedure. For example in character recognition, a typical feature might be the height-to-width ratio of the letter. Such a feature would be useful in differentiating between a 'W' and an 'I' in some machine fonts where 'W' is much wider than 'I'. On the other hand this feature would be quite useless in distinguishing between 'E' and an 'F'. The task of designing a feature extractor is one of finding as few features as possible that adequately differentiate the patterns in a particular application into their corresponding pattern classes. Feature extraction may also be done problem specific but there exists many feature extraction algorithms for known problems like mentioned width-height ratio, or extraction of character curves. There also exist strong algorithms like principle component analysis (PCA) that can be applied to any set of data to find out the best features that represents data.

***Classification:***

Classification is the process of making decisions concerning the class membership of a pattern in question. The task in any given situation is to design a decision rule that is easy to compute and will minimize the probability of misclassification relative to the power of the feature extraction scheme employed. In this part of recognition many methods exists to be used such as template matching, statistical methods and neural networks.

# 3. FEATURES OF THE REAL-TIME OCR SYSTEM

As a real world problem, the designed system is a consequence of a project that has been prepared for a company in card printing industry. Hence the motivation behind designing the system and donating it with various technological concepts is the necessity for a solution in a real world problem. As will be explained later, since the problem is too complex for a human to perform, computers and artificial intelligence is being used to bring a solution to the problem. In this chapter the main features and the design process of the system that lies in the core of the theoretical studies is explained.

## 3.1 Necessity of the Designed System

The real-time OCR system is decided to be realized to cover the industrial necessities. Hence, the requirement for the system and the place of OCR in printing industry is vital for the motivation of designing such a system.

### 3.1.1 OCR in Industry

OCR systems are mostly being used in offices for the task of digitalizing printed documents. In industry, OCR concepts are used for the same basic problem: digitalizing printed data. But this time the data doesn't only stand for documents with masses of characters, it is more excessively being used to notice special subsets of characters on different media and process them. Since in industrial processes it is not able to get high quality scanned images, most of the time industrial OCR systems deal with nosiy or depraved data. Reading number plates of cars with camera systems is one of the areas of research for this kind of applications [17] [18].

Since the designed system proposes a solution for a problem in printing industry, the area of concentration is OCR applications for printing industry. In printing industry, OCR systems are used for quality control. By using an OCR system, the printing faults are detected as a feedback to the operators. This quality control becomes very important if batch printing process is performed and the need for accurate printing is needed.

In printing industry, the printer automation systems are used widely and these systems generally have no quality feedback units that will return the faulty printed media back to the system operator. These automation systems need real time quality control systems that will work synchronously with the automation system in order to work ineffective with respect to the changes of printing speed. The designed system basically is built to overcome this problem.

### 3.1.2 Current Printing Automation System

Current system in the plant simply performs the printing groups of numbers on cards. The system includes a conveyor band in order for the cards to be feed to system. Over this conveyor five printing heads are positioned parallel as seen in Figure 3.4 to print numbers on cards. These five printing heads successively print the labels that are read from a database. These heads reach the database over controllers that are driven by a computer system. The printing heads print the numbers on the cards, working synchronously with the conveyor system so that the cards may be presented to the system asychronously.

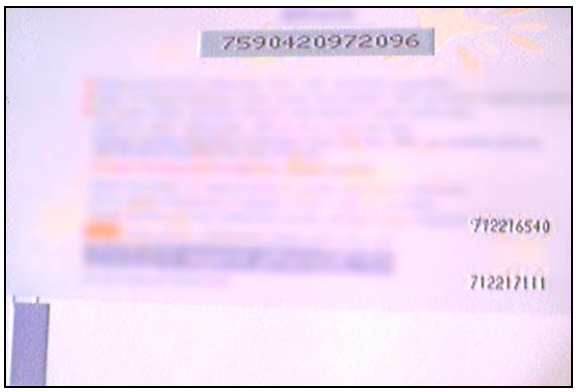



**Figure 3.1** A sample number card

The cards that the numbers will be printed on, is the same size of a credit card. These cards may contain groups of numbers as seen in Figure 3.1 that has predefined number of digits. The location of the groups of numbers on the card may vary through time but not through a session of printing process. The system prints five cards at the same time and has the ability to perform the printing process in different speed levels. The printing speed of the system varies from 5 cards per second to 25 cards per second.

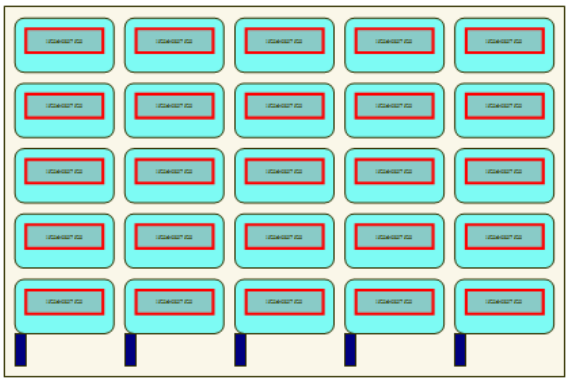


**Figure 3.2** A sample card sheet

The cards are not presented individually to the system. In order to keep the batch printing well-ordered, cards are presented to the system printed on sheets of media. After the number printing process is performed on a sheet, the cards are cut-out from sheets to be distributed individually. Each sheet of media includes 25 cards. Cards are placed on sheets as 5x5 card matrices as seen in Figure 3.2. Also the sheets have tags, printed on the bottom of them, in order to perform the synchronization process.

### 3.1.3 Defects of the Current System

The printing automation system is unable to perform a quality control operation over the printed cards. The system is unable to read the printed numbers on the cards and match them with what should be printed on the card being controlled. Because of the lack of this ability it becomes impossible to check if the numbers are printed accurately on the cards and stop the printing process if too much defects occur.

This control becomes very important when thousands of cards have to be printed accurately with a 25 cards per second speed. The batch processing may cause catastrophic results if the printing system starts to perform deterministic errors.

The probability of erroneous printing is more than an improbable theory, but a real possibility that occurs more frequently when the conveyor works with higher speeds. The printing automation system may print a number in the database again in the place of its successor in such speeds. Moreover the system may skip to print a number in arbitrary times that results a blank card to be printed.

All these mentioned errors are usual errors in these kinds of automation systems that appear undeterministically through time. In the current system the errors start to

happen more frequently in higher conveyor speeds and it becomes very hard to pick the sheets that have erronous cards inside.

### 3.1.4 Properties of the Target System

The main task of the system that will be designed to solve these problems is to detect the sheets that have erroneous cards inside and log these cards to the system operator in order to be picked from others precisely.

The errors of the system will be detected by accessing the database that keeps the numbers to be printed and check these with the printed numbers on the card. The printing heads keeps a predetermined order while printing the numbers in the database on the cards. There are a number of predetermined printing sequence schemes present in automation system, and the system to be designed has to work in the same sense while accessing the database.

In order to read the numbers on the cards, a camera system with high speed transfer rates has to be used. On the images acquired from the cameras, OCR operation will be performed and the verification of the number that should be printed on the card will be performed regarding the database of numbers. So, the system that will perform these operations may also be called as optical vharacter verification (OCV) system.

## 3.2 Infrastructure of the Real-Time Optical Character Verification System

Another synchronization issue is about triggering the OCV system to perform the OCR operation. This synchronizaion process can be done using the tags at the bottom of the sheets, the same way the automation system does.

With the synchronisation issue, comes another concept that drives the system to perform its operation in a limited time. This is very important in order for the OCV system to work properly in this real world problem. This makes the system a real-time OCV system, since the real-time systems are the ones that have limited time to perform its operations.

### 3.2.1 Limitations of the System

***Financial Limitations:***

Financial limitations are important in order to make the system more feasible with respect to similar systems in industry. In order to keep the budget low, the system has to work with cameras of poor quality which are called web-cam quality in the market. Also the yield of using a multi-processor mainframe system to perform the OCV operation real-time is obvious. But again to get the best performance from the lowest budget, a single home PC will be used to perform the real-time OCV operation.

***Performance Limitations:***

A more important issue is making the OCV system work real-time. In order to perform this, the system has to recognize approximately 325[1] digits in a second. In other words, OCV system has nearly 3.1 ms in order to cut out the digit, recognize the charater, and verify it.

Luckily, a little flexibility in verifying the numbers that are recognized is present. This flexibility comes from the randomness of the numbers that are being printed on the cards. In each card, the printed number is independent from the other numbers that are printed on cards. The numbers are prepared more than randomly, providing in each consecutive number more than %50 of the digits has been changed. By this way a one or two digit error in OCR process doesn't affect OCV process fatally.

### 3.2.2 Infrastructure of the System

#### 3.2.2.1 Hardware Infrastructure

There are three main hardware units in the system. These units and their technical specifications are:

***Cameras:***

The first important issue for the selected camera is its price. In order to make the system more affordable a webcam class camera must be chosen. This requirement becomes more obvious when it is remembered that three cameras will work at the same time to capture images of desired quality.

[1] Defaultly the cards have 13 digit numbers printed on them. Target is to read one plate per second that means 25 cards. A total number of 25x13=325 digits must be recognized in a second.

**Figure 3.3** Firewire camera of the system

The system needs to capture images within small time fragments. In this aspect the transfer rate of the camera becomes important. In order to fulfill this request an IEEE1394[2] compatible firewire camera with 400mb/s transfer rate is chosen , since this one is the highest transfer rate for webcam cameras after USB 2.0 standard.

Firewire cameras are still a matter of choice because of their potential of running more than one cameras at the same time in a PC. The support for multiple USB2.0 cameras were still in development at the time the project was being developed, so a firewire camera was chosen as the most appropriate unit.

One of the most important issues in camera selection was its shutter speed. The shutter speed of the camera determines the time that the shutter of the camera remains open, when taking snapshots from the image source. This time is very important since the image source is a dynamic system which changes state continously. In order to capture unblurred images from the source, the shutter speed must be able to work in very low time fragments. This time fragment is about 600ms in the maximum conveyor speed.

***Synchronization unit:***

Synchronization unit is nothing but a simple tone sensor that watches the bottom of the card sheets continously and sends a signal through the serial port of the PC when a change in the tone of the sheet flowing under it occurs. This signal is sent through the CTS channel of the standard RS232 sommunication port of the PC. The CTS signal is handled in the software to trigger the image capturing and OCR process.

[2] More info on IEEE1394 standard can be found in http://grouper.ieee.org/groups/1394/c/

***Software PC:***

Software PC is a Standard IBM PC with a IEEE1394 interface card attached to it in order to communicate with the firewire cameras , and its serial port is occupied by the synchronization unit in order to listen the triggering events.

#### 3.2.2.2 Software Infrastructure

In order to drive the system effectively, the most suitable programming language was C++ because of its object oriented structure and flexibility issues in memory management. The programming platform is chosen as Microsoft Visual C++ in order to use the supplied firewire API from the camera. Finally MFC is used for windows integration and GUI since it provides an enhanced API of wide range of functions.

## 3.3 General Structure of the System

Following the technical needs for the real-time OCV system, the general architecture of the designed system can be seen in Figure 3.4. There exist three cameras to capture frames from the cards that flow under them, a synchronization unit to synchronize the system with the conveyor and a PC to run the system software. Each part will be examined in more detail in later chapters. The designed system will work as follows.

Before the conveyor system is started to run, the operator of the system has to initialize the system so that it can work properly with the different types of cards. In this section, using the system software, the operator simply puts a dummy sheet under the cameras and takes a snapshot to select the “region of interest”s (ROIs). These ROIs are the positions on the card that contains the number groups that are desired to be verified with the database. After selecting the ROIs, the operator inputs other parameters of the system from the GUI of the system software. These parameters are examined more specifically in later sections.

After the initialization of the system, the conveyor system is started to run. As the system continues to run, the numbers are printed on cards and the sheet of cards continue to flow under the synchronization unit and the cameras consequtively. When the synchronization unit is triggered by the tags on the sheets, it sends a signal to the system software and the software triggers the cameras. When triggered, the

cameras take a snapshot from the cards and send them back to the system software so that the software can operate on them.

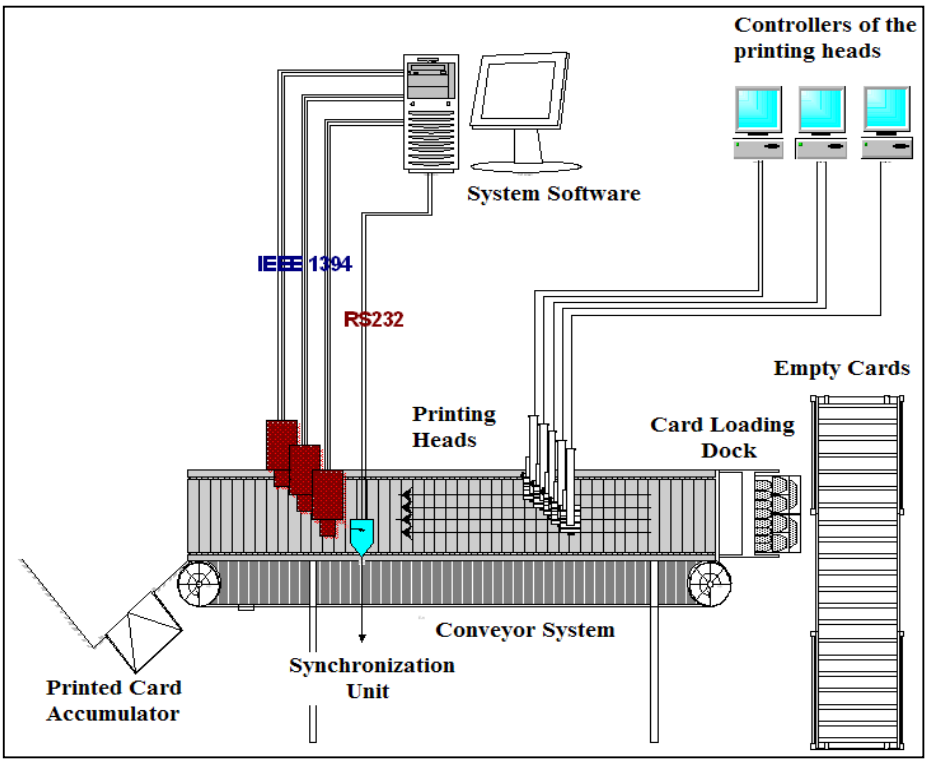



**Figure 3.4** General structure of the OCV system

The software then operates on the acquired images from the camera. Software extracts the numbers that are printed on the cards and the AI unit of the system digitalizes the images of each digit by presenting them to the patttern recognition sub-system, which is the main topic of the thesis.

The digitalized numbers are then compared with the numbers in the system's number database which is transferred to memory at the initialization section of the OCV software. The OCV system compares the numbers in a special way, which will be explained later, and then logs the results in a file. Also a warning signal is output to the operator so that the system's actions can be observed better.

The system repeats the recognition and verification actions after taking snapshots each time a signal is recieved from the serial port from the synchronization unit. This way the OCV system continues to work until the numbers in the database reaches an end, so does the printing process.

The algorithm of the system operation can be summarized as follows:

1. Initialize the system
    1.1. Activate cameras
    1.2. Select ROIs
    1.3. Input other necessary information
    1.4. Start a session by selecting a number database file
2. Start OCV process.
    2.1. Wait a signal from the synchronization unit.
    2.2. When signal comes get snapshots from cameras
    2.3. Extract the digit images from the snapshots
    2.4. Digitalize the digits of each number by presenting them to recognition system
    2.5. Compare the numbers with database
3. End when all the numbers in the database are finished

## 4. INFRASTRUCTURE OF THE SYSTEM AI

The main topic of the thesis is the experimental studies over the character recognition problem of the system. But why is the system has a character recognition problem while there exists well known solutions for optical character recognition? In this chapter the answer to this question will be given.

### 4.1 Cameras

Camera based problems arises because of the image quality of the camera. The camera is a webcam class camera and cannot produce high quality images that has sufficient resolution for a successful character recognition system.

A more important issue is the noise issue that makes it harder to segment the characters in the image. Because of the dot matrix font characteristics most of the time there exists disunities within a character. Because of these background interferences over the real characters, the image segmentation results produces characters that are partly out of trim. Even sometimes as the segmentation result, some part of the characters are connected with narrow bit sequences.

All of these problems occur because of the low image quality of the camera , but again it is the most appropriate camera regarding the limitations of the system which are mentioned in the earlier chapter.

### 4.2 Performance Necessity

For sure, this is the real problem when designing the system. OCR systems generally do not have a time limitation while they are working. But in this case hundreds of characters must be recognized and verified within a second. That keeps the recognition system from running too complex operations on the given data.

Even if it is said that the resolution of the input images are not high enough for a possible successful character recognition , the time limitation even draws the system to use a more feature reduced (images with less resolution) input sets.

The performance problem of the system forms the essential effort that has been spent for the studies of the thesis. In the presence of the minimization problem of the character recognition system, the neural network approach for the character recognition is tried to be passed beyond the performance-speed dilemma.

There also existed many other problems than character recognition problems which is as hard as the character recognition problems. These problems mostly contain adaptibility problems that tends to limit the developers flexibility to work with different kinds of tools. Because that these problems are not relevant with the thesis they will not be mentioned anymore.

### 4.3 Character Recognition Subsystem

The character recognition engine of the system mainly consists of two main parts as the usual OCR systems do. These parts are segmentation and recognition parts that take out the numbers in the captured image and digitalize them. The other parts that must be examined to fully understand the study is the data sets and how they are obtained and a brief summary of the techniques that are realized through the study.

#### 4.3.1 Segmentation

Segmentation is briefly the process of seperating the characters from each other and dividing the image to pieces that contain the characters. In order to segment the characters on the card precisely, the first mechanism that seperates the interested characters from the rest of the background is the selection of ROIs in the GUI of the system software. By selecting the ROIs, the uniterested sections of the card are discarded. So only one or two layers of solid background color(which is noisy in practice because of the quality of the webcam) and the characters that are intended to be digitalized remains.

Second phase of segmentation is the real segmentation process which will simply take out the numbers from the background color. Here, a vector quantization[3] operation is done to determine the numbers over the background and cut out these numbers. In vector quantization, the vectors that are close to each other are grouped into one vector so that the input vector space is quantized into a small number of vectors at the end of the process. Here the quantization operation groups the noisy

[3] More info on vector quantization can be found in [2]

background colors into one or two vectors (according to the number of background layers) and gropus the color of the printed characters into one vector.

As mentioned before, vector quantization is a simple and fast process, and performance is important in recognization engine so that it is the most appropriate technique to be used in segmentation. But it sometimes erronously quantizes the images if the input is too noisy or the printed character has too many disconnections inside.

### 4.3.2 Recognition Engine

After the image has been segmented and the input characters are transformed into standardized images, the process of digitalization begins. The standardized images are binary images that has ‘0’s inside for the background and ‘1’s for the character data. This image data must be interpreted to predetermined classes and digitalized this way. This classification operation is nothing but a pattern recognition problem which leads the problem of character recognition to be solved by neural networks.

Neural networks are the technology that has the widest area of application in pattern recognition and especially character recognition tasks , since they are able to be trained to recognize new patterns. Another reason to use neural networks is their stability agianst noisy and deformed data which fully overlaps with system needs. The properties of neural networks and why they have been chosen is examined deeper in the next chapter.

There are many kinds of neural networks which bring the system new abilities and at the same time limit some other abilities. In the thesis study the characteristic neural networks are applied to the real world system and the performance they provide in practice is obtained in this aspect.

## 4.4 Infrastructure of the Experimental Studies

Over the constructed system that has been explained in the former sections the performance improvement studies over the realized system is performed. These studies are performed over the real data that are obtained from the running system and the implemented solutions are tested on the real system.

### 4.4.1 Data Set

As argued before the data set of the recognition system is the output of the segmentation subsystem which are binary images that present ‘0’ for background and ‘1’ for an atomic piece of the character. These images are then presented to the neural network as input data in order for the neural network to classify them.

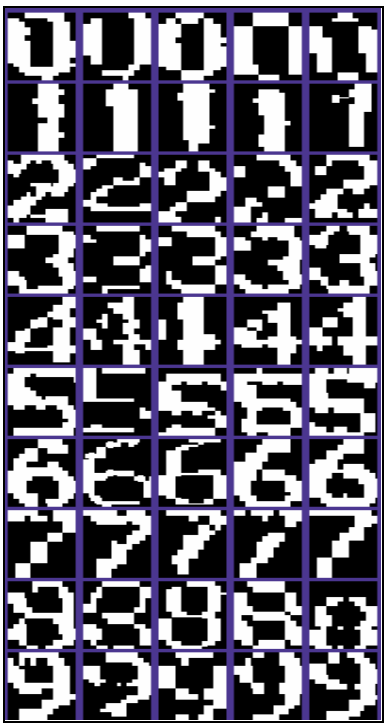


**Figure 4.1** Samples from the data set

As a result of the noisy images from the camera , the outputs for the vector quantization may result in disturbed images as can be seen in the Figure 4.1. These disturbed images are expected to be eliminated as a result of the neural network operation.

The input images are 24 bits in width and 36 bits in weight so that a total number of 864 bits of data vectors are presented to the neural network to operate. In most of the cases neural network gets too big in order to process all the 864 bits of the image. Since the performance is deadly important for the system, the input vectors are summarized or transformed into another vector space in order to keep the neural network smaller. These modifications on the data is explained in each technique to be applied in Chapter 6.

### 4.4.2 Techniques to Be Applied

There are many neural networks in literature that are used for different proposes. In our case the probem is very specific, in fact it is more specific than recognizing characters. Only the numeric characters of a single font is going to be recognized which keeps the number of output classes very small. There are actually 10 classes as long as there are 10 different digits present.

So the techniques that are selected must be able to deal with the problem of overlapping input classes. Another problem is the size of the network which must be kept as small as possible in order for the conveyor to perform in maximum speeds.

Applied techniques involve many neural network types ranging from supervised neural networks to unsupervised neural networks and also a mixture of them. Also evolutionary methods are applied to train neural networks by the means of soft computing techniques. For each neural network input vector space is transformed into an appropriate new input vector space so that the neural network operates with maximum throughput.

After each applied technique, the performance cost of the neural network is calculated and the advantages and disadvantages of the network is argued. This way the most appropriate recognition engine is found out as the result of the study.

The following section covers the theoretical background of the techniques that are used in the thesis study. By understanding the underlying concepts of the applied techniques, the explanations for the reasons that are behind the applied technique in question will be understood effectively.

# 5. THEORETICAL BACKGROUND

All of the techniques that has been applied in order to solve the performance-speed dilemma has strong backgrounds in the artificial intelligence and neural network history. These techniques have proven qualities and being used in many pattern recognition tasks. In this chapter these techniques that are used in expreimental studies are going to be explained in order to explain the reasons behind choosing these systems to be used in recognition engine of the OCV system.

## 5.1 Artificial Neural Networks

Artificial Neural Networks (ANNs) are information processing systems that brings the computer to act more effectively on many tasks such as pattern matching, optimization and data clustering. The very first attempts in constructing an ANN model has begun in 1940s [19]. Just like any new technology ANNs have been boosted up by new approaches and lost its popularity from time to time. From the beginning of 1980's hey have started to being used excessively.

### 5.1.1 Biological Inspiration

Like most of the concepts in artificial intelligence, ANNs are based on a biological model. Human nerveous system is responsible from the data processing and communication in human body. Its atomic elements are the neurons which performs the basic communication operations by using chemical reactions. They also perform the information processing in order to decide passing the messages through themselves. The brain is also composed of a gigantic complex neuron network.

Human brain consists of approximately $10^{11}$ neurons , each having approximately $10^{4}$ connections. As we all know, human brain has the ability to learn and to store huge mass of data. It shall also be possible to gain such functionality for computers if human nerveous system is modeled well enough.

In order to model precisely the way neuron works, it must be examined properly. A neuron, as seen in the Figure 5.1, consists of three parts: nucleus, axon and

dendrides. Data flow begins with a neuron receiving some signals by the dendrides over its nucleus, with the help of the synapses (connections) between dendrides of the neurons. After the signals are received, they are gathered in the nucleus and the received cell fires (activates) if the electrical potential of the signals are above a certain threshold.

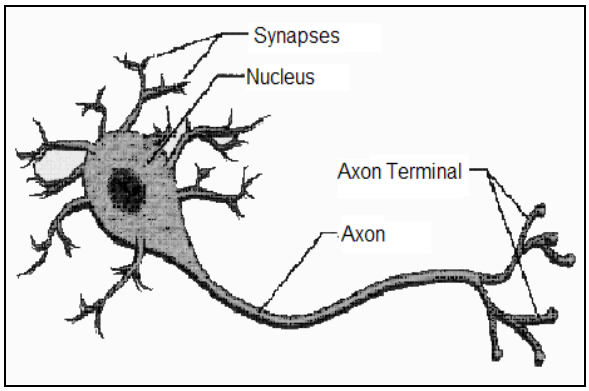



**Figure 5.1** A neuron cell

If the cell fires, it sends a signal through its axon to the dendrides at the end of the axon and the signal is sent to other neurons by the help of the synapses between neurons.

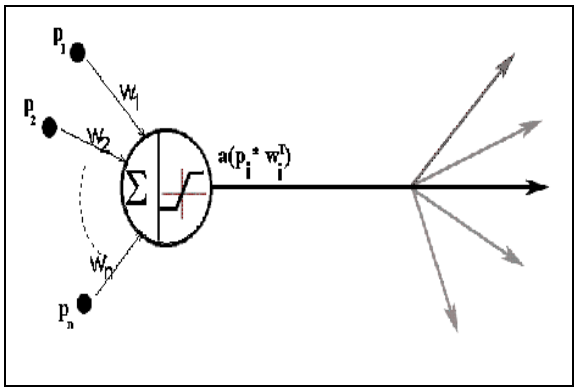



**Figure 5.2** A processing element

In Figure 5.2, a model that has been inspired by the neuron cell is shown, proposed by MCC Pitts [19] which is also known as M-P neuron model. In this model the processing elements calculates the weighted sum of its inputs (5.1) and decides to fire an output using a unit step function (5.2).

$$y_i(t+1) = a\left(\sum_{j=1}^{m} w_{ij}\, p_j(t) - \theta_i\right) \tag{5.1}$$

$$a(f) = \begin{cases} 1 & ,if\ f \geq 0 \\ 0 & ,otherwise \end{cases} \qquad (5.2)$$

In the equations above $y_i(t+1)$ is the output of a neuron at time t+1 which is formed by applying the transfer function of the element to the weighted sum of element's inputs which is calculated by summing the multiplication of each input value '$p_j$' with its corresponding weight '$w_{ij}$' of $j^{th}$ input to the $i^{th}$ element. After this summation the bias value is also added to the weighted sum and the result is presented to the transfer function. The transfer function in (5.2) is hard limiting function which outputs '1' if the weighted sum is positive and '0' if negative or zero.

The important element here is the weights that decide how much each input will affect to the final signal level in the processing element. After the input is obtained, the function that acts on the input, here a unit step function determines the behavior of the processing element and calculates the output signal.

Briefly, an ANN is a parallel distributed information processing structure with the following characteristics:

- It is a mathematical model inspired by biological facts
- Highly interconnected processing elements builds up the structure
- Knowledge is hold in the weights
- The behavior of each processing element is determined locally
- It has the ability to learn, recall and generalize data by tuning its weight values.
- It is a distributed system, so no processing element holds specific knowledge but the system performs the given task

Taking a closer look at the elements of ANNs, the important concepts of a neuron model are processing elements, connections and learning rules.

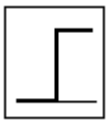
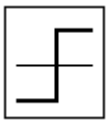
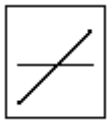
### 5.1.2 Processing Elements

As mentioned before, the transfer function of the processing element is the main mechanism to determine how the inputs of the artificial neuron are carried through it, to its output. In this manner, a group of functions are used commonly as being named

"activation function" or "transfer function". A summary of these functions can be found in the Table 1.

Each processing element may have a special type of input which is called a "bias". This kind of input always has a constant value which equals generally '1' and sometimes '-1'. The bias is used to shift the response function of the neural network which is described in Section 5.3 in more detail.

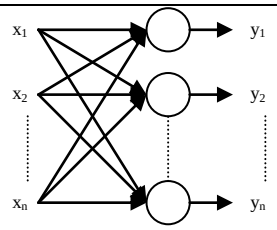
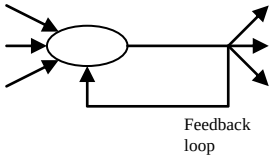
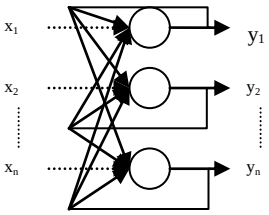
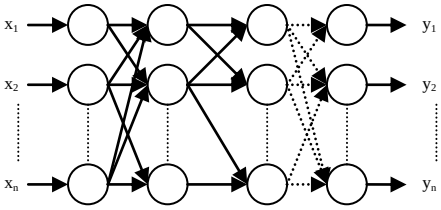
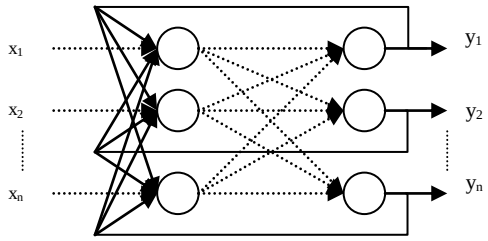
**Table 1** Some transfer functions

| **Name** | **Input/Output Relation** | **Icon** |
|---|---|---|
| Hard Limit | $a(y_i) = 0 \quad y_i < 0$<br>$a(y_i) = 1 \quad y_i \geq 0$ |  |
| Symmetrical Hard Limit | $a(y_i) = -1 \quad y_i < 0$<br>$a(y_i) = +1 \quad y_i \geq 0$ |  |
| Linear | $a(y_i) = y_i$ |  |
| Saturating Linear | $a(y_i) = 0 \quad y_i < 0$<br>$a(y_i) = y_i \quad 0 \leq y_i \leq 1$<br>$a(y_i) = 1 \quad y_i > 1$ |  |
| Symmetric Saturating Linear | $a(y_i) = -1 \quad y_i < -1$<br>$a(y_i) = y_i \quad -1 \leq y_i \leq 1$<br>$a(y_i) = +1 \quad y_i > 1$ |  |
| Log-sigmoid | $a(y_i) = \frac{1}{1+e^{-y_i}}$ |  |
| Hyperbolic Tangent Sigmoid | $a(y_i) = \frac{e^{y_i} - e^{-y_i}}{e^{y_i} + e^{-y_i}}$ |  |
| Positive Linear | $a(y_i) = 0 \quad y_i < 0$<br>$a(y_i) = y_i \quad y_i \geq 0$ |  |
| Competitive | $a(y_i) = 1 \quad$ *neuron with* $\max y_i$<br>$a(y_i) = 0 \quad$ *other neurons* |  |

### 5.1.3 Connections

As mentioned before, each neuron in an ANN doesn't hold any specific knowledge but ANN functions well as a whole. This condition emerges the result that the interconnection style of the artificial neurons in the network is very important and affetcts excessively the behavior of the whole system. Table 2 shows the most common organization types of artificial neurons in a neural network.

**Table 2** Five basic network connection geometries

| | | |
|---|---|---|
|  |  |  |
| Single-layer feedforward | Single node with feedback | Single layer recurrent |
|  |  | |
| Multilayer feedforward | Multilayer recurrent | |

Here a new concept is introduced about the structure of an ANN which is called a layer. A layer is the concept of logically grouped artificial neurons whose output can be grouped to be handled and whose inputs are also come from a logically grouped data source. This source can be another layer of artificial neurons or may be the data that comes from the inputs. The layer that handles those inputs is called the "input layer" and the layer that gives the final output of the ANN is called "output layer". Other layers which feed other layers and fed by other layers are called "hidden layers".

There are no certain rules in connection of the neurons so that an artificial neuron may skip a number of layers and connect with another neuron. A neuron may also

ignore the neuron hierarchy and make a connection with itself or a neuron in one of the former layers that feeds the neuron's layer. ANNs with these types of connections are named feedback networks since a layer may feed a layer that feeds it. ANNs with no such connections are called feedforward networks. Feedback networks which hold closed loops inside are called recurrent networks.

Recurrent networks may have special types of neurons that use a differential equation to carry its input to the output through time. These types of neurons are called integrators. An integrator neuron is shown in the Figure 5.3 and its function carrying the niput through time is shown in (5.3) where a(t) is the output of the neuron at time t, a(0) is the initial condition p(t) is the input to the neuron at time t. An example to the integrator neuron is explained in Section 5.4.

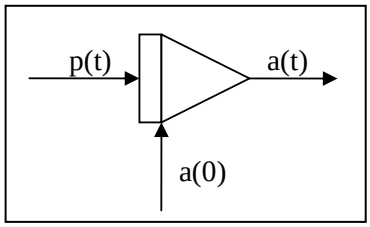



**Figure 5.3** An integrator neuron

$$a(t) = \int_0^t p(\tau)d\tau + a(0) \tag{5.3}$$

In this type of recurrent networks where inputs are carried through time, there exist two kinds of inputs which are called "excitatory" and "inhibitory". An excitatory input causes the output of the neuron to increase through time and the inhibitory input causes the output of the neuron to decrease through time.

Another important type of connection is lateral feedback connection in which a neuron is connected to another neuron in the same layer. One of the important structures that include lateral feed backs is the on-center off-surround structure which will be discussed later.

Different structures need different mechanisms to tune the ANNs behavior. This tuning mechanism is called learning.

### 5.1.4 Learning Rules

Learning is mentioned in two ways for ANNs. The first kind is parameter learning which is a procedure for modifiying the weights and biases of a network. The second

type is structure that is a procedure for modifiying the connection structure of the network. The latter type is a new concept and discussed later.

The purpose of parameter learning is to train network in order to perform some task. The training process is the modification of the weights of the network. There are two types of parameter learning which are mostly used in supervised and unsupervised learning.

In neural network learning, the concept of the presentation of a training input to the neural network is called iteration. For each input that is going to be learned by network, a training iteration is performed by presenting the input to the network and updating the weights according to the error rate. When all the samples in the input set are finished an "epoch" is said to be completed. An epoch is the presentation of all the input vectors once in the training set iteratively to the neural network. It may take many epochs to complete a training session, until the desired error rate is reached.

***Supervised Learning:***

In supervised learning, the modification of the weights of the system is based on the target values which must be obtained when certain inputs are presented to the system. As seen in Figure 5.4 the trainer must have some desired values called targets that are paired with certain input vectors in his hand. These pairs can be shown as

$\{p_1,t_1\}\ \{p_2,t_2\}\ \ldots.\ \{p_i,t_i\}$ (5.4)

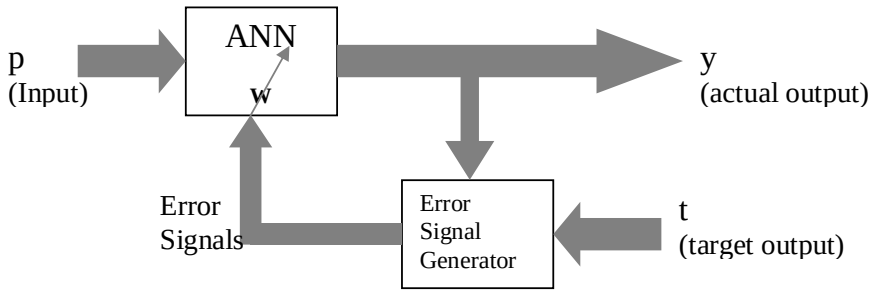



**Figure 5.4** Supervised learning diagram

Each target with its input pattern is shown with an indice. During the training process each input pattern in the training set is presened to the system and the weights updated regarding the difference between current output and target output.

There are many methods to update the weights according to the difference between target and output which is called error. Some of these methods will be examined in later chapters.

There is also a special type of supervised learning which is called reinforcement learning in which the feedback data for the output is only given as a ‘true’ or ‘false’ statement.

***Unsupervised Learning:***

In unsupervised learning there is no outer criterion that tells the network how its output should be. The network must find out the clusters and pattern classes by itself. While discovering the features of the presented patterns, the network undergoes some changes in its parameters. This process is called self organizing. Examples of this type of learning are introduced in later sections.

## 5.2 Pattern Classification and Neural Networks

Early pattern classification research performed in the ‘60s and ‘70s focused on asymptotic(infinite training data) properties of classifiers, on demonstrating convergence of density estimators and on providing bounds for error rates. Many researchers studied parametric Bayesian classifiers where the form of input distribution is assumed to be known and parameters of distributions are estimated using techniques that require simultaneous access to all training data.

For recent research, more attention is being paid to practical issues as pattern classification techniques are being applied to speech, vision, robotics and artificial intelligence applications where real-time response with complex real-world data is necessity, just like in the OCV system that has been designed. This has led to an emphasis on robust, adaptive, non-parametric classifiers that can be implemented on parallel hardware.

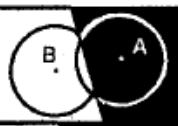
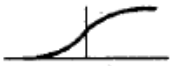
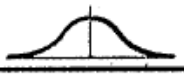
Adaptive non-parametric neural-net classifiers work well for many real-world problems. These classifiers differ in their ability to use unsupervised training data and in the case with which internal operations can be understood and interpreted to determine what input features contribute to classification performance [1].

The goal of pattern classification is to assign input patterns to a finite number, M, of classes. Input patterns can be viewed as points in the multidimensional space defined

by the input feature measurements. The purpose of a pattern classifier is to partition the multidimensional space into decision regions that indicate to which class any input belongs.

Good classification performance requires selection of effective features and also selection of a classifier that can make good use of those features with limited training data, memory, and computing. In addition, input features must be extracted automatically, allowing to effect input parameters as they would do in practical applications where extensive hand-tuning is normally impossible.

**Table 3** Types of neural network classifiers

| Group | Decision Region | Computing Element | Representative Classifiers |
|---|---|---|---|
| Probabilistic |   | Distribution Dependent | Gaussian Mixture |
| Hyperplane |  | Sigmoid  | Multi-Layer Perceptron, Boltzmann Machine |
| Receptive Fields (Kernel) |  | Kernel  | Method of Potential Functions, CMAC |
| Exemplar |  | Euclidean Norm  | K-Nearest Neighbor, LVQ |

Supervised training, unsupervised training, or combined unsupervised/supervised training can be used to train neural net classification and clustering algorithms.

Practical differences between classifiers and internal differences in how classifiers form decision regions lead to the taxonomy of classifiers containing four main groups as seen Table 3. In the thesis study, members of these groups is realized and applied to solve the problem.

***Probabilistic Classifiers:***

Probabilistic classifiers assume a priori probability distributions such as Gaussian or Gaussian mixture distributions for input features [20][21]. Parameters of distributions are typically estimated using supervised training where all training data is assumed to be available simultaneously.

***Hyperplane classifiers:***

Hyperplane classifiers form complex decision regions using nodes that form hyperplane decision boundaries in the space spanned by the inputs. Nodes typically form a weighted sum of the inputs and pass this sum through a sigmoid nonlinearity. These classifiers have low memory and computation requirements during classification but may require long training times and/or complex training algorithms. They include multilayer perceptrons trained with backpropagation.

***Kernel Classifiers:***

Kernel or receptive field classifiers create complex decision regions from kernel-function nodes that form overlapping receptive fields. Kernel-function nodes use a kernel function which provides the strongest output when the input is near a node's centroid.

Kernel classifiers train relatively rapidly, can use combined unsupervised/supervised training, and have intermediate memory and computation requirements. Neural net kernel classifiers include map based approaches that use arrays of nodes which compute kernel functions [22][23].

***Exemplar Classifiers:***

Exemplar classifiers perform classification based ın the identity of the training examples, or exemplars, that are nearest to the input. Nearest neighbor can be determined using exemplar nodes that are similar to the kernel function nodes. Exemplar classifiers train rapidly but many require large amounts of memory and computation time for classification. These include Learning Vector Quantizer and Adaptive Resonance Theory classifiers.

Classifiers from these four groups often provide similar low error rates but differ dramatically in practical characteristics. In addition to providing reduced error rates over older approaches, these classifiers provide trade-offs in memory and computation requirements.

## 5.3 Perceptron

Perceptron is the pioneer network model that was developed with one of the first neuron models designed. The structure of perceptron consists of one layer of neurons which adds its inputs and fires an output if the sum is above a threshold value. Each

neuron should have a bias value and a weight value for each of its inputs in order to fine tune its characteristic behaviour.

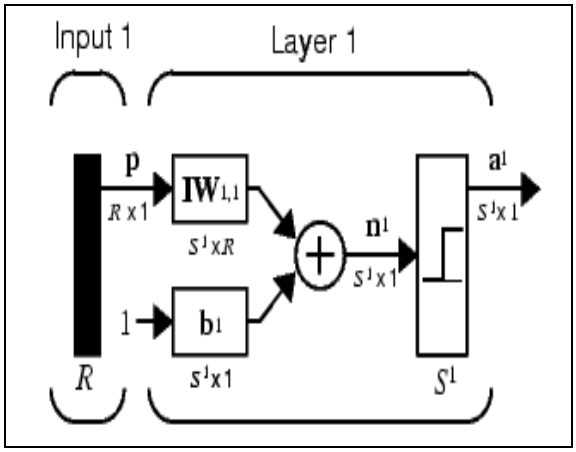



**Figure 5.5** A single-neuroned perceptron

As seen in the Figure 5.5, the main decision mechanism which determines the output of the perceptron is its transfer function which is generally chosen as hardlim for perceptron.

$$a = \text{hardlim} ( Wp + b ) \tag{5.5}$$

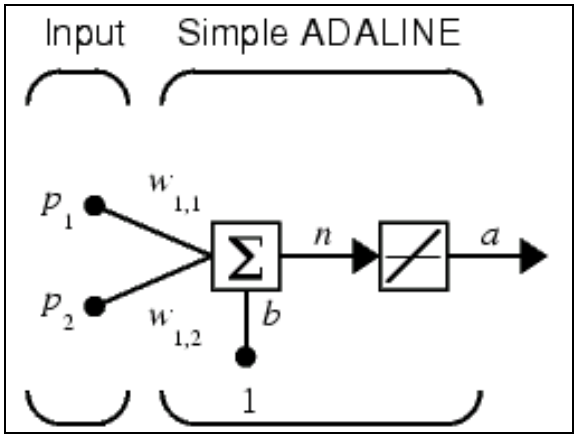



**Figure 5.6** ADALINE with two inputs

If a two input simple ADALINE (Figure 5.6) which has a very similar architecture with perceptron, is examined it is seen that its output is determined regarding the formula

$$a = \text{purelin} ( w_{1,1}\, p_1 + w_{1,2}\, p_2 + b ) \tag{5.6}$$

This means that a neuron in a neural network is activated if the summation of bias and the inner product of weight matrice and inputs are compared with the threshold value of the neuron's transfer function. When the transfer function is combined with the input the neuron will have the equation in (5.7).

$$w_{1,1}\, p_1 + w_{1,2}\, p_2 + b = 0 \tag{5.7}$$

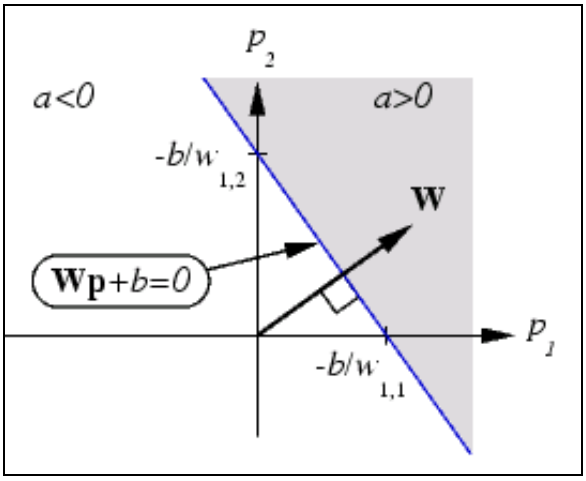



**Figure 5.7** Decision boundary of ADALINE

From the Figure 5.6 and equations (5.5) and (5.6) the graph in Figure 5.7 can be plotted. The graph in the figure accepts the values greater than the line produced by the equation (5.7). This line is said to be the decision boundary of the neuron. If the point given by the inputs falls beyond this decision boundary it is said that the given pattern is recognized.

This decision boundary changes dimension while the number of inputs gets bigger. If a three dimensional input vector is presented to the neuron, the decision boundary will be determined by three coordinates which will turn the decision boundary into a plane rather than a line.

### 5.3.1 Perceptron Learning Rule

The learning for the perceptron is realized by presenting a number of input and target vectors as if it had been discussed in the previous section. These pairs are presented in (5.4)

Since perceptron is one of the basic neural networks, its learning rule can be constructed in a basic sense. When the perceptron is initialized, its initial weights are

initialized randomly. That is because in a training process the network does not know about the nature of the problem.

After the weight vectors are initialized, an initial decision boundary is formed randomly. For a while the neuron can be accepted as biasless so that its decision boundary passes through the origin.

Accepting a strategy that each training input vector is to the network and the weights are updated according to the error rate of the network, the training rule can be accepted as

$$w^{new} = w^{old} + ep \tag{5.8}$$

where

$$e = t-a \tag{5.9}$$

Here "e" stands for the error rate which is most basically calculated as the difference between the target value and the current output of the system. As long as the bias is nothing but an input that is always accepted as '1', its weight should also be updated regarding the rule. That means,

$$b^{new} = b^{old} + e \tag{5.10}$$

Using this rule each of the perceptron's neurons can be updated. By this way the perceptron learning rule can be generalized as

$$_iw^{new} = {}_iw^{old} + {}_iep \tag{5.11}$$
$$_ib^{new} = {}_ib^{old} + {}_ie \tag{5.12}$$

The perceptron learning rule is proven to converge in linearly seperable problems. Unforunately many problems are not linearly seperable so more complex structures are needed.

### 5.3.2 LMS Algorithm

The perceptron learning rule is very straightforward and may be incapable in some cases it may not converge to desired outputs. Rather it can oscillate between static values and may never converge. Also it may not work properly for other similar Networks like ADALINE. In order to comeover these problems more complex

methods including the statistical analysis of the mean square error of the outputs are used to define a better algorithm for perceptron.

Recalling the notation used before, the inputs and outputs are described in (5.1).

For the sake of notation simplification the weight and the input vectors are also notated as

$$x = \begin{bmatrix} {}_1w \\ b \end{bmatrix} \tag{5.13}$$

$$z = \begin{bmatrix} p \\ 1 \end{bmatrix} \tag{5.14}$$

where ${}_1w$ is used for all the weights in the networks input layer. Thus the output of the network yields from the form

$$a = {}_1w^T p + b \tag{5.15}$$

to the form

$$a = x^T z \tag{5.16}$$

Using this notation the mean square error of the outputs can be defined as:

$$F(x) = E[e^2] = E[(t-a)^2] = E[(t-x^T z)^2] \tag{5.17}$$

In LMS algorithm the mean square error of the network is given by

$$F(x) = (t(k) - a(k))^2 = e^2(k) \tag{5.18}$$

where the expectation of squared error is replaced by the squared error at iteration k. This definition of LMS learning rule uses an approximate to steepest descent[4] algorithm algorithm to calculate local minimum. It is said to be an approximation because it doesn't use the original statistical quantities of the error rate. It uses (5.18) instead of (5.17). Using (5.17), the gradient squared error at iteration k can be written as

$$\hat{\nabla} F(x) = \nabla e^2(k) = -2e(k)z(k) \tag{5.19}$$

[4] More info on steepest descent algorithm can be found in [3]

As it is obvious, this approximation lets us use the multiplication of error vector with input vector instead of the real gradient which is not convenient to calculate. By this way the new values for the weight vectors can be calculated as

$$_1w(k+1) = {}_1w(k)+2\,\alpha\, e(k)p(k) \tag{5.20}$$

and for the biases

$$b(k+1) = b(k) + 2\,\alpha\, e(k) \tag{5.21}$$

These equations are known to be the LMS rule which is also known as delta learning rule or widrow-hoff learning rule in literature. Here the 'α' is the learning rate used to tune up the speed-sensitivity trade-off. This rule will be the bas efor the backpropagation rule which is very important in neural network training.

### 5.3.3 Multilayer Perceptron

Taking the perceptron one step further the multilayered feedforward neural networks have the ability to solve the problems which are not linearly seperable. This is done by adding a group of neurons which is called a "layer" between the input and the neurons which decide the behaviour of the network. This layer is called a hidden layer and as it is seen in Figure 5.9. There may be many hidden layers in a neural network.

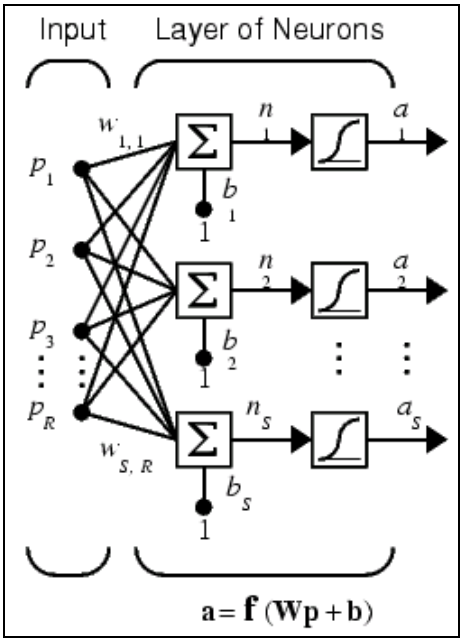



**Figure 5.8** A Layer of a feedforward network

The hidden layer is said to transform the input space into a new image space which also transforms the problem to a linearly seperable problem. Given as an example, a three layered neural network model that solves the XOR problem[5] can clearly illustrate the hidden layer's effect.

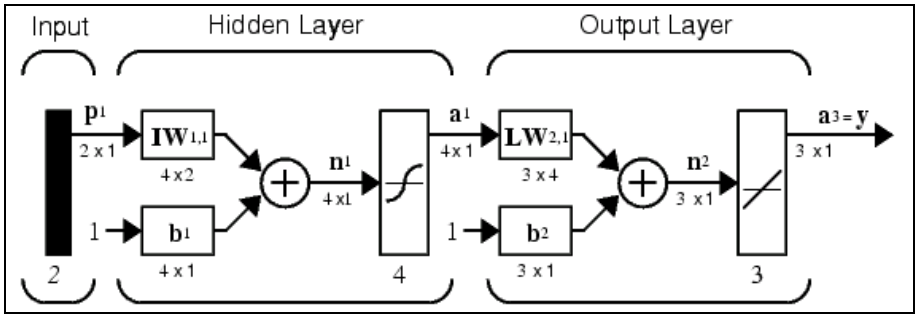



**Figure 5.9** A Multi-Layer feedforward neural network

### 5.3.4 Backpropagation

After the development of backpropagation algorithm, it has evolved the modeling and processing of many quantitive phenomena that uses neural networks. Backpropagation learning algorithm simply consists of two parts

- Propagation of the input pattern to the network.
- Backpropagation of the error rate back from the output to the input.

In the first step, a training pattern is presented to the network and each output is calculated by

$$_{m+1}a = {}_{m+1}f\,(_{m+1}w\,{}_{m}a + {}_{m+1}b) \tag{5.22}$$

where m indicates the number of layers

Propagating the input through the network, the performance index of the current weights are calculated by the mean square error, which is the natural result of being the generalized form of the LMS algorithm. The backpropagation algorithm should adjust the weights in order to minimize the error rate

$$F(x)=E[e^2] = E\,[(t-a)^2] \tag{5.23}$$

which can be generalized for multiple outputs as

$$F(x)=E[e^Te] = E[(t-a)^T(t-a)] \tag{5.24}$$

[5] Solution of this problem can be found in [16]

With the LMS algorithm the mean square eroor is approximated by

$$\hat{F}(x) = (t(k) - a(k))^T (t(k) - a(k)) = e^T(k)e(k) \quad (5.25)$$

This rule is the same as the one in single layered case. Now it is modified using the chain rule and partial derivatives and generalized in the form

$$_m w(k+1) = {}_m w(k) - \alpha s_m ({}_{m-1}a)^T \quad (5.26)$$

$$_m b(k+1) = {}_m b(k) - \alpha s_m \quad (5.27)$$

that means for each element

$$\Delta w_{iq} = \alpha s_i a_q \quad (5.28)$$

Here, 's' is the sensitivity of each neuron to changes. That means the rate of change in the neuron for a particular amount of error. And $a_q$ is the input to the $q^{th}$ layer applied to function a().

In the second step the error is distributed to all of the neurons in the network. That is nothing but backpropagating sensitivities ('s') through the network so that each of neurons should know how much they have to change. This sensitivity is calculated as

$$s = [t_i - a_i][a'(inp_i)] \quad (5.29)$$

where 'i' is the neuron number in the output layer. Looking the big picture of the backpropagation training, a generalized algorithm can be constructed as follows.

1. Initialize the weight vectors of the network.
2. Apply $[p_i]$ ($i^{th}$ pattern's input vector) to the network, and calculate each pattern's output vector $[a_i]$
3. Compute the error values for the ouput layer using (5.29)
4. Propagate the error backwards to update the weights
$$\Delta_M w_{ij} = \alpha_{M-1} S_{M-1} a$$
$$_{M-1}s = a'({}_{M-1}inp)[{}_{M-1}w]^T [{}_{M-1}s]$$
5. Cycle through the set of training data
6. If total error is accaptable, then quit.

The multilayer perceptrons are said to converge for almost all functions. But enough number of neurons and layers are required in order for the network to converge. There are no known decision criterias for indicating the number of layers and

neurons in MLPs but it is said to be well determined by performing experiments. Similar constructive proofs, developed independently [24][25][26] , demonstrated that two hidden layers are sufficient to form arbitrary decision regions using multilayer perceptrons.

More recent work demonstrated that multilayer perceptrons with only one hidden layer could form complex disjoint and convex decision regions [27][28]. Also a proof is presented in [29] demonstrates that continous non-linear mappings can be closely approximated using sigmoidal non-linearities and multilayer perceptrons with only one hidden layer.

Another issue is the optimization method decision in which there have been many proposed algorithms. Among them the most effective ones are those which aim to minimize the error rate. Steepest descent is usually regarded as a poor strategy; so many other methods are available for solving optimization problem.

MLP is stil being widely used in pattern calssification tasks with varying structures because in theory they are able to converge for almost all functions.

### 5.4 Adaptive Resonance Theory

Before examining the ART model some basic components must be explained in order to explain ART model effectively.

#### 5.4.1 Shunting Activation Model

The shunting activation model which is also known as multiplicative activation is an important generalization of the additive activation equation [33].

In a basic additive dynamic model, there exists neurons that work with the principle of leaky integrator which decays its output exponentially through time if no excitatory input is presented to the system. The basic model for a leaky integrator [7] is shown in Figure 5.10.

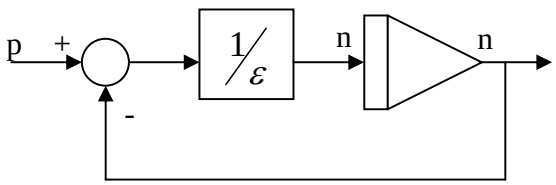



**Figure 5.10** The Leaky Integrator

$$\varepsilon \frac{dn(t)}{dt} = -n(t) + p(t) \tag{5.30}$$

Starting out with (5.30), if the input p(t) of the leaky integrator is constant and the initial condition is n(0)=0, the equation will be in the form

$$n(t) = p(1 - e^{-t/\varepsilon}) \tag{5.31}$$

The graph of this function for p=1 and $\varepsilon = 1$, looks like

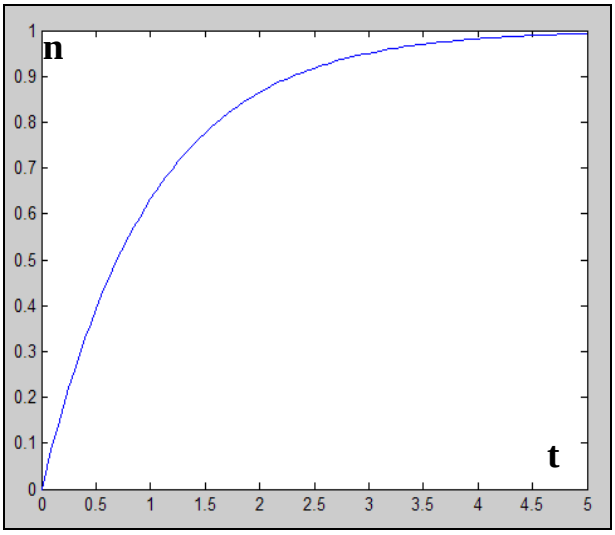



**Figure 5.11** The Graph of the leaky integrator's equation

In leaky integrator the input, given by p is directed to the output decayed through time which is determined by the constant $\mathcal{E}$. In other words, in a time interval determined by $\mathcal{E}$, the leaky integrator reaches to its constant input level.

In shunting model, the leaky integrator structure is generalized, and the ability to control the response boundaries of the neuron is controlled by the added elements to the neuron which can be seen in the Figure 5.12.

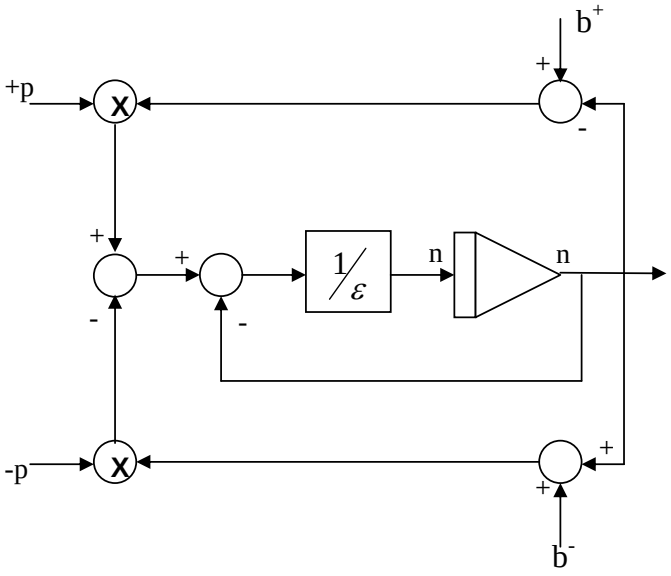



**Figure 5.12** Shunting model neuron

In shunting model the equation of the neuron is given by

$$\varepsilon \frac{dn(t)}{dt} = -n(t) + (b^+ - n(t))p^+ - (b^+ + n(t))p^- \qquad (5.32)$$

In equation (5.32) $p^+$ and $p^-$ inputs are called excitatory and inhibitory inputs, which cause the response of the neuron to decrease and to increase respectively.

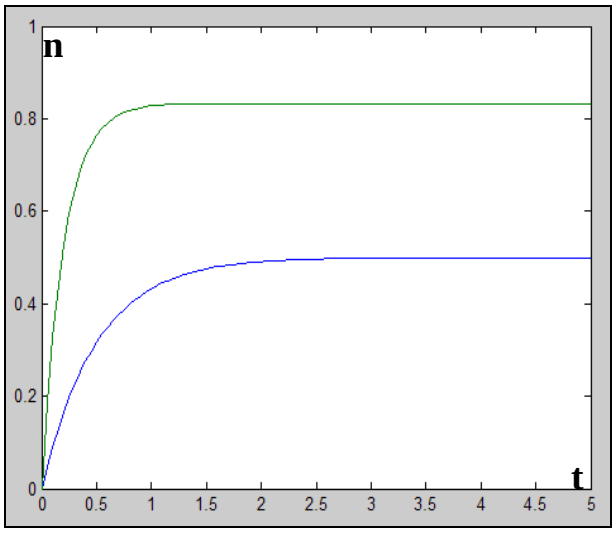



**Figure 5.13** Response of the shunting neuron

The biases $b^+$ and $b^-$ are also non-negative constants used to determine the upper and lower boundaries of the neuron response. Examining the response graphic of this neuron modal for common values $b^+=1$, $b^-=0, \varepsilon = 1$, $p^-=0$ ; and respectively $p^+=1$,

$p^+$=5 for the blue and red ones, the effects of the elements in the model can easily be understood.

This model is an important part of the competitive network model, because by the usage of nonlinear gain control the input to the network will be normalized and by this way the relative intensities will be obtained.

### 5.4.2 Competitive Networks

In a competitve network model, layers compete neurons compete with each other to determine a winner. In these types of neural networks the winner take all principle activates the winner neuron and this neuron indicates the stored prototype pattern which is most likely to the input pattern.

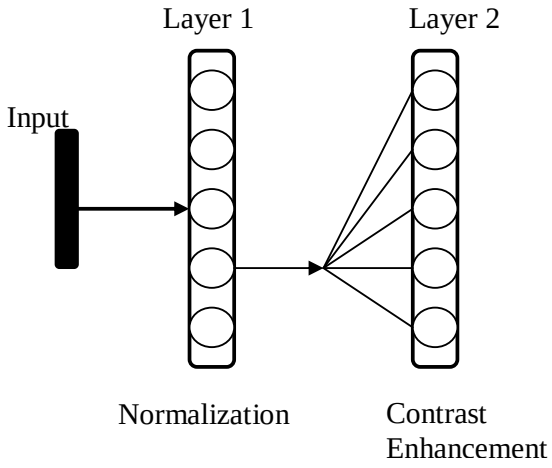



**Figure 5.14** Grossberg network

In a grossberg network[8] seen in Figure 5.14, which is also a type of competitive network, the competition is supported by the normalization and contrast enhancement functions which are originally the functions of the inspired biological model "retina"[6]. By this way the mutual differences can be dealt with and a more robust decision can be made.

[6] Actually the inspiration for the grossberg model and ART is the human vision system [8]. The very first citations about ART network is printed in "IEEE Applied Optics" magazine.

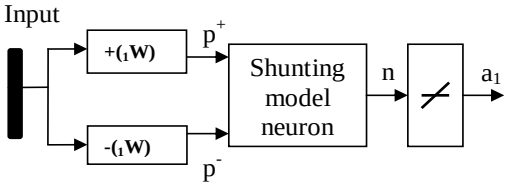



**Figure 5.15** Grossberg layer 1 neuron model

Simply describing, Layer 1(L1) of the network performs the normalization task and uses shunting model in Figure 5.15. This layer performs the normalization, using on-center off-surround connections from the input to the input layer neurons. Each input is connected on-center as an excitatory input to its corresponding neuron in input layer and connected off-surround as an inhibitory input to other neurons in input layer. This connection diagram can also be seen in Figure 5.16. By this way each input effect others and decays excitatory inputs' pull-up effect with an amount determined by the weights.

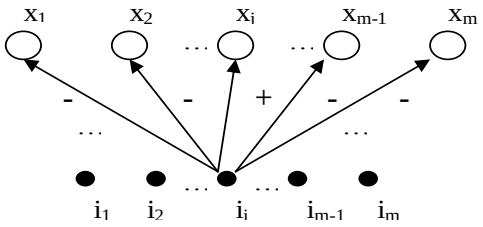



**Figure 5.16** On-center off-surround connections

In layer2(L2) , there exists recurrent connections as seen in Figure 5.17 in addition to L1's structure in order to store presented patterns even after they had been removed from the system. L2 also performs contrast enhancement mechanism in order to select the most likely pattern to the stored pattern in a very similar way to winner-take-all. The difference is the losing neurons' output starts to decay in contrat enhancement instead of immidieatly being set to '0'.

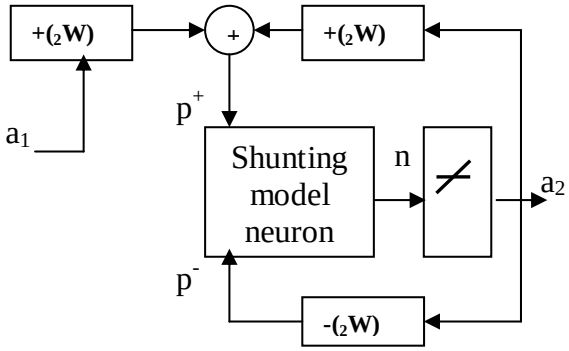



**Figure 5.17** Grossberg layer 2 neuron model

The main drawback of the Grossberg network is its instability in learning. This instability is the result of two factors. The first one is the affect of mutual comparison. As a result of mutual comparison, a poor pattern among worse patterns is likely to be chosen as a match for an irrelevant stored pattern. The second problem is the tendency of the network to overwrite recently learnt patterns.

### 5.4.3 Basic ART Network

It was the analysis of the competitive networks' instability (ref limlin) that led the researchers to develop ART system. ART model is developed ins such a way that they are stable enough to preserve significant former data , yet remain adaptable enough to incorporate new information whenever it might appear.

ART networks overcome stability-plasticity dilemma by the introduction of a vigiliance parameter '$\rho$'. This parameter changes the behaviour of the network in such a way that the network becomes more stable for new patterns or vice versa. The vigiliance parameter changes between '0' and '1' so that as it gets closer to '0' the categorization performance of the network gets coarse. That means the network prefers to classify the new patterns with existing ones instead of creating a new cluster. If the situation is inversed, the vigiliance parameter gets closer to '1', the network gets very sensitive to new patterns. It produces a new cluster for most of the patterns presented.

Having a look at the Figure 5.18, a new outstar connection and an orienting subsystem is added to Grossberg network's structure to form ART model. In order to understand how ART models work it is necessary to examine each component of the architecture.

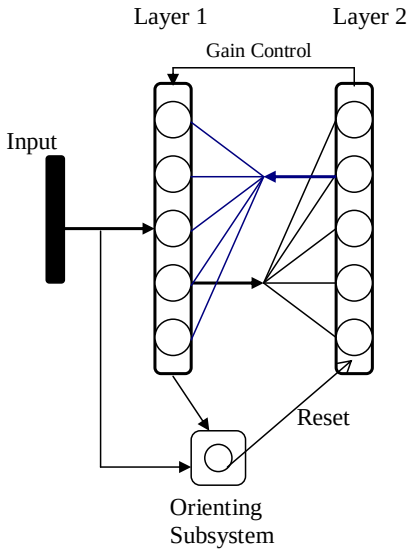



**Figure 5.18** Basic ART architecture

***Layer 1 of ART:***

The main task of L1 is to compare the presented input pattern with the expecation pattern that comes from L2. If the patterns are not close enough, L1 triggers orienting subsystem to cause a 'reset' in L2. If the patterns are close enough, the base pattern and the new pattern is combined using 'AND' operation or addition. This way the new pattern is joined to the existing one.

The difference of ART L1 from the Grossberg network is that ART always accepts binary inputs and doesn't perform normalization therefore there exists no on-center off-surround connections in L1 of ART network.

***Layer2 of ART:***

L2 of ART is almost identical with the Grossberg network's L2. It performs a winner-take-all operation between its neurons, hence input patterns, and nonzeroes the winner neuron's output. By this way L2 selects the most resembling patterns and makes the corresponding neuron return a non-zero output.

There are two differences between L2 of Grossberg network and ART. The most important difference is the reset signal from the orienting subsystem that causes the neuron to stop functioning until a suitable pattern is selected from the remaining patterns. The second difference is the secondary transfer function of the ART which is applied to the output of the first transfer function which transforms its output to binary data. Therefore this function is chosen as 'hardlim'.

***Orienting Subsystem:***

Orienting subsystem plays an important role in ART's behaviour. It simply decides if the pattern chosen by L2 is close enough to the input pattern. In this system '$\rho$' comes into action and if the pattern doesn't match the criteria a reset signal is sent to the corresponding neuron and that neuron stops actig until a pattern which is close enough to the input pattern is found among the remaining neurons. The structure of the system is as in Figure 5.19.

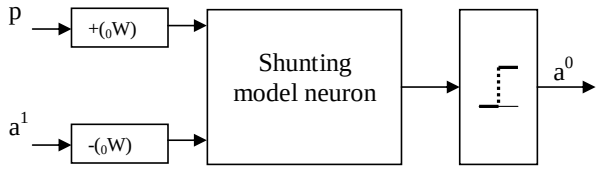



**Figure 5.19** Structure of Orienting Subsystem

Here p is the input coming from the original pattern and $a^1$ is the output of the L1. The output of the subsystem is shown with $a^0$.

### 5.4.4 ART 1

The ART1 network has two seperate learning laws for L1-L2 and L2-L1 connections. L1-L2 connections use a type of instar learning rule because it has to select a pattern between the stroed patterns according to the input pattern. L2-L1's task is the reversed of the L1-L2's so that it must recall the selected original pattern and present it to the L1. This is done via outstar learning law.

As a result when the pattern selected by the input pattern matches with the input pattern, the system is said to be in resonance and the layer weights are updated at the same time. Summarizing the ART1 algorithm, the principle can be understood better.

1. Initialization occurs. L2 is not activated.

2. Since L2 is not activated, for the first pattern, the output of L1 equals $a^1$=p

3. The pattern is carried to L2 by the equation $W^{1:2}a^1$; where $W^{1:2}$ represents weights between L1 and L2. After that, the neuron with the largest output according to the function

$$a_i^{\,2} = \begin{Bmatrix} 1, & if\ ((w_i^{1:2})a^1 = \max[(W^{1:2})^T a^1]) \\ 0, & otherwise \end{Bmatrix}$$

is activated where $w_i$ is the $i^{th}$ neuron's weights adn $a^1$ is the output of the first layer.

4. L2-L1 expectation is computed by $W^{2:1}a^2$ where $W^{2:1}$ : L2-L1 weights.

5. Expectation is added to L1 pattern by 'AND'ing here. $a^1 = p \cap w_j^{2:1}$

6. Next, orienting subsystem determines if the match is close enough.

$$a^0 = \begin{cases} 1, & if\ \|a^1\|^2 / \|p\|^2 < \rho \\ 0, & otherwise \end{cases}$$

7. If $a^0=1$ , reset the corresponding neuron and restart the process.

8. Else, resonance has occured so update the weights of selected pattern's column.

9. Remove the input pattern and restore all resetted neurons in former steps.

The ART1 networkis designed as a real-time system. All the weight updates and other mechanisms are described as differential equations so that ART1 can be realized using analog circuits (ref). The ART1 network performs very well with binary input pattern, but it is very sensitive to noise. Also it must have binary patterns as inputs in order to work properly. ART2 network is designed to overcome this problem.

### 5.4.5 ART 2

ART2 is a system that implements ART1's functionality over analog signals and performs classification tasks over them. There also exists ART2-A which performs ART2's tasks three of four times faster.

ART2 brings a third layer in front of input layer which analyses and normalizes input patterns. This layer performs normalization using the equation (5.33) where $I^0$ is the original signal and I is the output of the layer 0 which also becomes input to the core ART network.

$$I = N(P_\theta(N(I^0))) \tag{5.33}$$

N(x) is a function that satisfies

$$N(x) = \frac{x}{\|x\|} \tag{5.34}$$

and $F_\theta(x_i)$ satisfies

$$F_\theta(x_i) = \begin{cases} x_i, & if\ x_i > \theta \\ 0, & otherwise \end{cases} \quad (5.35)$$

Here the threshold is chosen as

$$0 < \theta \leq M \quad (5.36)$$

where M is the dimension of the input vector. There are also some small modifications on L2 and orienting subsystem. These modifications can be found in [10]. The importance of ART2 comes from its ability to handle analog signals which is not possible in ART1.

## 5.5 Self Organizing Feature Maps

### 5.5.1 Kohonen's Learning Rule

Tuevo Kohonen introduced a learning rule to be used in competitive networks in [11]. This is a rule that is similar to the instar [30][31] learning rule. According to the rule, the winner neuron of the competition ($j=i^*$) updates its weights according to the following rule

$$\Delta w_i = \begin{cases} (1-\alpha)w_i(t-1) + \alpha p(t) & ,if\ i = i^* \\ 0 & ,otherwise \end{cases} \quad (5.37)$$

where $\Delta w_i$ is the $w_i^{new}$-$w_i^{old}$ and $\alpha$ is the learning rate. Using this rule, the change in the change in the weights can be seen in the graph. It is obvious that the old weight vector move towards the input vector with a rate determined by $\alpha$.

Using this rule the weight vectors will move closer to the pattern groups in each iteration and eventually each vector points at a different cluster of input vectors as seen in Figure 5.20. After the necessary number of input patterns are presented to the network, the wieghts will remain stable at the point thet classifies the input vectors as seen in the Figure 5.21.

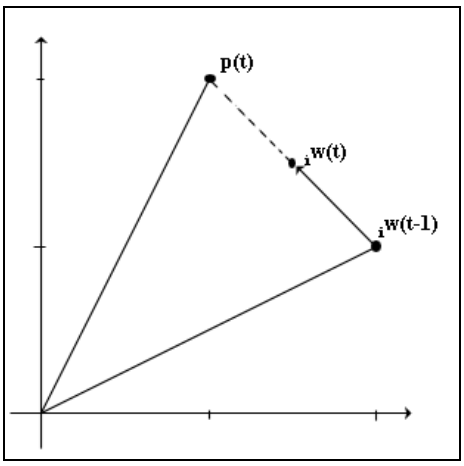



**Figure 5.20** Vector changes in Kohonen learning rule

The learning rate parameter of the competitive learning acts as a decision maker for the trade-off between the speed of learning and the stability of the final weight vectors. A learning rate closer to zero will result in a slow learning process, however after the vector has converged it remains stable pointing to the input cluster. If the rate is closer to '1' on the other hand the weight vector will converge faster but at the end will be unstable and continue to oscillate as different vectors in the cluster are presented to the network.

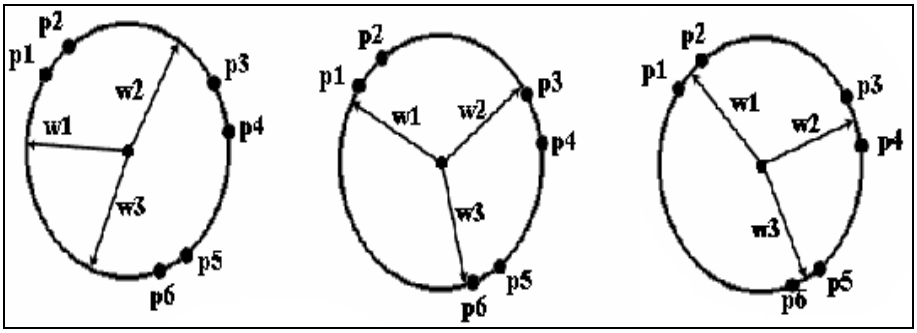



**Figure 5.21** Representation of Kohonen learning rule

There are also some other problems like dead weight vectors that points to no patterns. This happens when a vector is initialized far from the input patterns. Another problem occurs when the input patterns are very close to each other. This causes the vectors to slide to other clusters and hence trigger a change in the current clustering.

### 5.5.2 Basic SOM Network

The most obvious difference of SOM from other networks is the role of the topology in Networks functionality. In SOM the neurons in a layer can be lined up in n-dimensions according to the input's attributes. A two dimensional example layer can be seen in Figure 5.22.

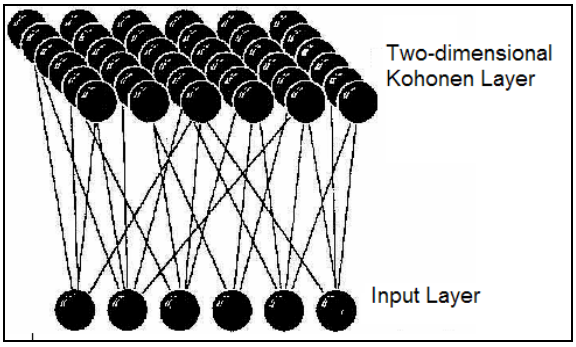



**Figure 5.22** A SOM

After the Kohonen layer has been formed, it will have lateral feedback connections which are called as "distance"s between the neurons. Each distance is originally a weight that corresponds to the correlation between the neurons. After the layers are formed the input pattern is presented to the network and a winning neuron is determined using competitive neural network concepts. Next, the winning neuron's weights are updated using the Kohonen's learning rule.

At that point, another important issue for SOM is reached which is called neighborhood of a neuron. Neighbors of a neuron is the neurons that are located ohysically near a neuron in a layer. The neighborhood also has levels and a neuron's neighborhood is determined regarding which neighbors shall be included with respect to their neighborhood levels.

Taking one step backwards, after the winning neuron's weights are updated, the wieghts of the neurons that remain in the winning neuron's neighborhood are also updated. There are many mechanisms that are used to determine the neighbors of a neuron and the intensity that the change in the weights can be carried to neighboring neurons.

As a result, the neurons in the SOM's output layer is physically organized according to the input patterns. Some organizations for related inputs can be seen in Figure 5.23.

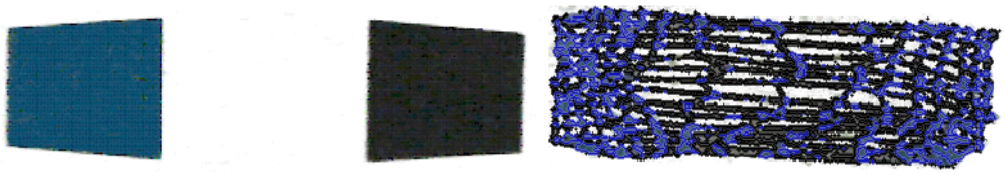


**Figure 5.23** Kohonen layer of a SOM arranged with an input

### 5.6 Learning Vector Quantization(LVQ)

LVQ networks are hybrid systems that both include unsupervised and supervised learning. This way, the pattern classification power of the unsupervised networks can be joined with the clustering power of the supervised learning.

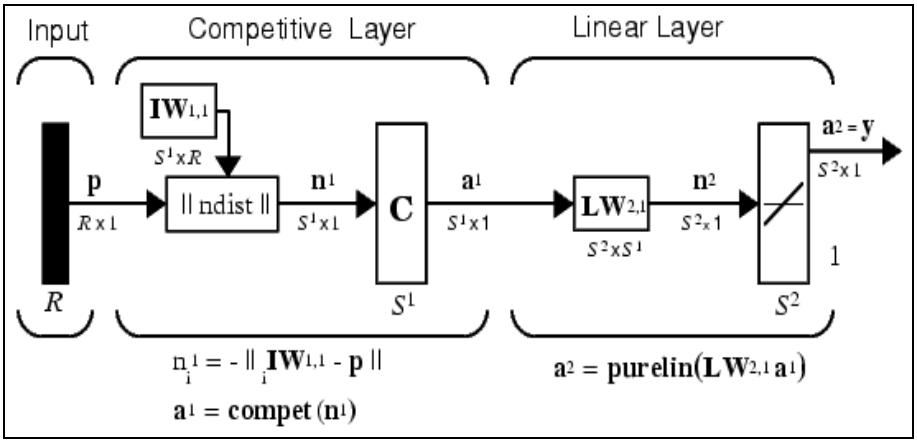



**Figure 5.24** Structure of a LVO network

In a LVQ network there exists two layers as seen in Figure 5.24. The first layer is the competitive layer which classifies inputs according to the similarity between them. This layer works unsupervised. The first layer of the LVQ is responsible of classifying regions from the input space, discarding the similarity between these regions. The inputs to the first layer of LVQ is determined by (5.38).

$$n' = \begin{bmatrix} \left\| w_1' - p \right\| \\ \left\| w_2' - p \right\| \\ \vdots \\ \left\| w_i' - p \right\| \end{bmatrix} \tag{5.38}$$

With these inputs, the need of normalization is eliminated because the length of the distance vector between the weight vectors and input is calculated automatically before the first layer.The first layer runs a competition among its inputs and the neuron whose weight vector is closest to the input vector will outpt a ‘1’ and the other neurons will output ‘0’.

Each winning neuron in the L1 indicates a subclass for the L2. L2’s task is to combine these subclasses into the supervised class.

***LVQ learning:***

LVQ Networks has a simple learning rule. As discussed above, the LVQ network has two layers, one for classifying inputs and other for grouping classes. So when the learning starts first layer runs a competition between the output neurons of L1 to determine a winning neuron, and the winner is stated as i*. The i* element of the layer 1 is set to ‘1’.

After the competition, the winning neuron is carried to the second layer and classified as class k*. If p is classified correctly the weights of the winning neuron is moved towards p using Kohonen rule.

$$_{i*}w^{1}(t) = {}_{i*}w^{1}(t-1) + \alpha\,(p(t) - {}_{i*}w'(t-1)) \tag{5.39}$$

If p was classified incorretly, then the wrong classification has been occured and the weights are moved away from the p by the following rule:

$$_{i*}w^{1}(t) = {}_{i*}w^{1}(t-1) - \alpha\,(p(t) - {}_{i*}w'(t-1)) \tag{5.40}$$

At each iteration of the training process, an input vector is presented to the network and the weights of the network is adjusted.

The LVQ classifiers typically have error rates that are similar to those of backpropagation classifiers but often train faster, and require more memory and computation time during classification.

## 5.7 Genetic Algorithms in Neural Network Training

Mentioned before, the training of a neural network is nothing but an optimization process of the network weights. In this aspect genetic algorithms (GAs) are rational choices to be applied in neural network training.

### 5.7.1 What are Genetic Algorithms

The GA is a stochastic global search method that mimics the metaphor of natural biological evolution. GAs operate on a population of potential solutions applying the principle of survival of the fittest to produce (hopefully) better and better approximations to a solution. At each generation, a new set of approximations is created by the process of selecting individuals according to their level of fitness in the problem domain and breeding them together using operators borrowed from natural genetics. This process leads to the evolution of populations of individuals that are better suited to their environment than the individuals that they were created from, just as in natural adaptation. Individuals, or current approximations, are encoded as strings, chromosomes, composed over some alphabet(s), so that the genotypes (chromosome values) are uniquely mapped onto the decision variable (phenotypic) domain. The most commonly used representation in GAs is the binary alphabet {0, 1} although other representations can be used, for example ternary, integer, real-valued etc. For example, a problem with two variables, x1 and x2, may be mapped onto the chromosome structure in the following way:

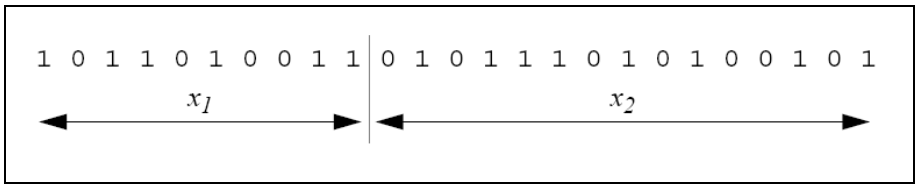



**Figure 5.25** A chromosome in a GA

where x1 is encoded with 10 bits and x2 with 15 bits, possibly reflecting the level of accuracy or range of the individual decision variables. Examining the chromosome string in isolation yields no information about the problem trying to be solved. It is only with the decoding of the chromosome into its phenotypic values that any meaning can be applied to the representation. However, as described below, the search process will operate on this encoding of the decision variables, rather than the decision variables themselves, except, of course, where real-valued genes are used.

Having decoded the chromosome representation into the decision variable domain, it is possible to assess the performance, or fitness, of individual members of a population. This is done through an objective function that characterizes an individual's performance in the problem domain. In the natural world, this would be an individual's ability to survive in its present environment. Thus, the objective

function establishes the basis for selection of pairs of individuals that will be mated together during reproduction. During the reproduction phase, each individual is assigned a fitness value derived from its raw performance measure given by the objective function. This value is used in the selection to bias towards more fit individuals. Highly fit individuals, relative to the whole population, have a high probability of being selected for mating whereas less fit individuals have a correspondingly low probability of being selected. Once the individuals have been assigned a fitness value, they can be chosen from the population, with a probability according to their relative fitness, and recombined to produce the next generation. Genetic operators manipulate the characters (genes) of the chromosomes directly, using the assumption that certain individual's gene codes, on average, produce fitter individuals. The recombination operator is used to exchange genetic information between pairs, or larger groups, of individuals. The simplest recombination operator is that of single-point crossover. Considering the two parent binary strings:

$$P_1 = 1\ 0\ 0\ 1\ 0\ 1\ 1\ 0$$
$$P_2 = 1\ 0\ 1\ 1\ 1\ 0\ 0\ 0$$

If an integer position, i, is selected uniformly at random between 1 and the string length, l, minus one [1, l-1], and the genetic information exchanged between the individuals about this point, then two new offspring strings are produced. The two offspring below are produced when the crossover point i = 5 is selected,

$$O_1 = 1\ 0\ 0\ 1\ 0\ 0\ 0\ 0$$
$$O_2 = 1\ 0\ 1\ 1\ 1\ 1\ 1\ 0$$

This crossover operation is not necessarily performed on all strings in the population. Instead, it is applied with a probability p(x) when the pairs are chosen for breeding. A further genetic operator, called mutation, is then applied to the new chromosomes, again with a set probability p(m). Mutation causes the individual genetic representation to be changed according to some probabilistic rule. In the binary string representation, mutation will cause a single bit to change its state '0' to '1' or '1' to '0'. So, for example, mutating the fourth bit of O1 leads to the new string,

$$^{m}O_1 = 1\ 0\ 0\ 0\ 0\ 0\ 0\ 0$$

Mutation is generally considered to be a background operator that ensures that the probability of searching a particular subspace of the problem space is never zero.

This has the effect of tending to inhibit the possibility of converging to a local optimum, rather than the global optimum. After recombination and mutation, the individual strings are then, if necessary, decoded, the objective function evaluated, a fitness value assigned to each individual and individuals selected for mating according to their fitness, and so the process continues through subsequent generations. In this way, the average performance of individuals in a population is expected to increase, as good individuals are preserved and bred with one another and the less fit individuals die out. The GA is terminated when some criteria are satisfied, for example a certain number of generations, a mean deviation in the population, or when a particular point in the search space is encountered.

### 5.7.2 Genetic Algorithms Versus Traditional Methods

From the above discussion, it can be seen that the GA differs substantially from more traditional search and optimization methods. The four most significant differences are:

- GAs search a population of points in parallel, not a single point.
- GAs do not require derivative information or other auxiliary knowledge; only the objective function and corresponding fitness levels influence the directions of search.
- GAs use probabilistic transition rules, not deterministic ones.
- GAs work on an encoding of the parameter set rather than the parameter set itself (except in where real-valued individuals are used).

It is important to note that the GA provides a number of potential solutions to a given problem and the choice of final solution is left to the user. In cases where a particular problem does not have one individual solution the GA is potentially useful for identifying these alternative solutions simultaneously.

### 5.7.3 Major Elements of the Genetic Algorithm

The simple genetic algorithm (SGA) is described by Goldberg [32] and is used here to illustrate the basic components of the GA. A pseudo-code outline of the SGA is shown in below. The population at time t is represented by the time-dependent variable P, with the initial population of random estimates being P(0).

```
procedure GA

begin

    t = 0;
    initialize P(t);
    evaluate P(t);
    while not finished do

    begin

        t = t + 1;
        select P(t) from P(t-1);
        reproduce pairs in P(t);
        evaluate P(t);

    end

end.
```

### 5.7.4 Genetic Algorithms and Neural Networks

Genetic algorithms are widely claimed to perform well in finding global optimums for optimization problems, so it is worth investigating whether they are also capable of training neural networks well. In order to apply genetic algorithms for neural network training, a suitable representation for these networks must be found, which can be stored in the form or chromosomes. Furthermore, a problem specific fitness function will need to be defined. Based on the practicular fitness function and chromosomes, the genetic algorithm can assess the performance of each neural network represented by those chromosomes, and have them evolve towards an acceptable solution.

The GAs and neural networks can be combined in several ways. So far, GAs have been mostly used to:

- Generate the weights of a neural network [14],
- Generate the architecture of a neural network [15],
- Generate both the architecture and the weights of a neural network simultaneously,
- Analyse a neural network.

#### 5.7.4.1 Genetic Algorithms in Neural Network Stucture Training

Neural network structure, in other words the number of nodes the formation and number of connections between them has a significant impact on the performance of a neural network and its training ability. The density of connections in a neural network determines its ability to store information. If a neural network does not have enough connections between nodes, the training algorithm may never converge; the neural network is not able to approximate the desired function. On the other hand, in a densely connected network, overfitting may occur. Overfitting is a problem with statistical models where too many parameters are present. This is a bad situation because instead of learning to approximate the function present in the data, the network can simply memorize each training example. Noise in the training data is then learned as part of the function, often destroying a network's ability to generalize.

Generating the architecture of a neural network with or without simultaneously generating its weights, has proven to be a very difficult task, yielding only moderate results on relatively small problems. Every individual in the population codes a neural network structure, with or without its weights. During evaluation, a chromosome is first translated into a neural network structure. When the weights are not coded in the chromosome, initial weights are usually generated randomly. A training stage follows, generally being a fixed number of backpropagation steps (requiring a training set) or a second genetic algorithm. Finally, a test set is used to determine the fitness of the network. Usually, the fitness function of the network incorporates a measure for the complexity of the particular network, in order to give the GA a preference for smaller networks.

#### 5.7.4.2 Genetic Algorithms in Neural Netork Weight Training

Since the network structure, including the neuron’s target function, is predefined and remains identical throughout the process of the GA, it is not necessary to code any architectural information into the chromosome. Thus it is possible to define a network solely by its set of weights, in which the thresholds of the neurons are included. So instead of evolving the parameters of the network, the parameters are now constant from one generation to the next and the GA is used to evolve neural network's weights. The genetic algorithm is now the training algorithm. A complete set of weights is coded in a (binary or real number) string, which has an associated fitness indicating its effectivenes. For example, the fitness can be simply given by -e,

where ‘e’ is the value of the cost function for the set of weights. Starting with a random population of such strings, successive generations are constructed using genetic operators to evolve new fitter individuals (weights) out of the old ones which are more likely to survive. In principle the crossover operation can bring together good building blocks found by chance in different members of the population. Unlike the backpropagation learning rule, GAs perform a global search and are thus not easily fooled by local minima.

# 6. APPROACHES FOR THE RECOGNITON ENGINE

## 6.1 Multilayer Perceptron and Backpropagation

Multilayer perceptron (MLP) is one of the most essential techniques that is used in pattern recognition tasks. Backpropagation is the common training method that is used with MLP. These issues are discussed in detail in Section 5.3. Since the MLP is still being used in most of the pattern recognition tasks it will be rational to use the MLP in the OCR sub-system regarding its simplicity and success rate.

On the other hand, since the input vectors that are planned to be presented to MLP are very large in size a feature reduction operation is inevitable. During the experiments this feature reduction operation is done in two ways. The results that are obtained using these two ways are examined in this section.

### 6.1.1 MLP with Projectioned Inputs

***Inputs:***

Mentioned above there exists 864 pixels in each input image that will tremendously grow the size of the MLP. For each digit printed, handling 864 pixels seperately and trying to handle at least 200 images in a second like this seems to be a naïve idea. To overcome this problem a feature reduction operation over these inputs must be applied to the problem.

The first feature reduction operation is performed by counting the full pixel numbers in each image and reforming the inputs according to the distributions of these pixels. The mentiond distributions are formed with respect to the horizontal and vertical coordinates of the image. In other words the image is projectioned over the horizontal and vertical directions in order to count how many pixels exist in each column and row of the image. This projection method is summarized in Figure 6.1.

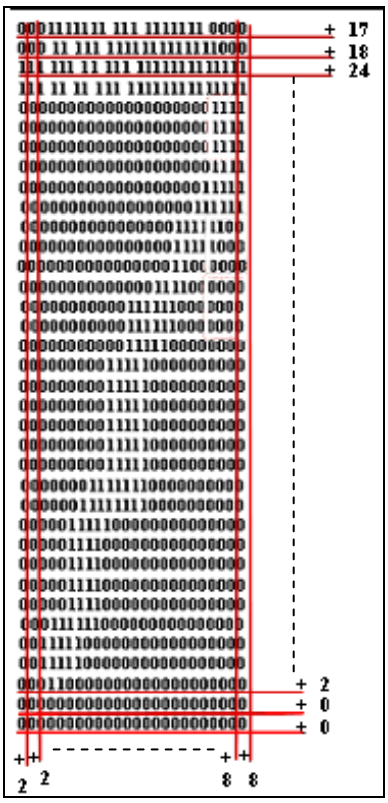



**Figure 6.1** Image projection method

Using this method, horizontal and vertical projections of pixel distributions of the image is obtained for every image in the training set. Since the images have 36 rows and 24 columns the projectioned inputs has 60 features inside. The sample projectioned pixel distribution of each number is presented below in Table 4.

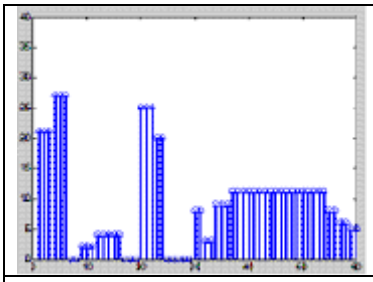
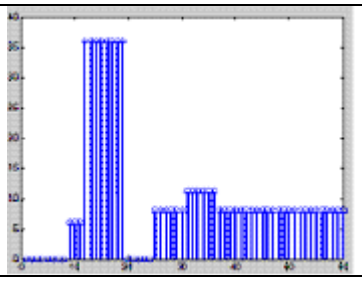
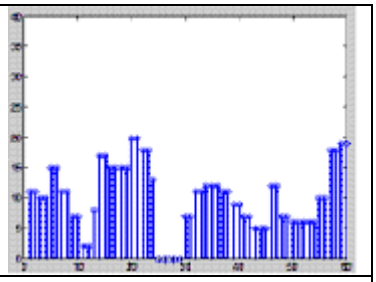
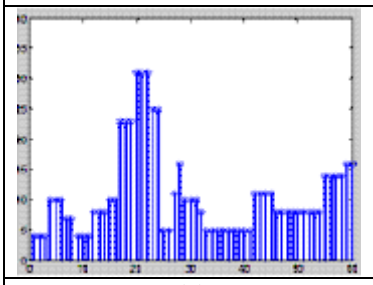
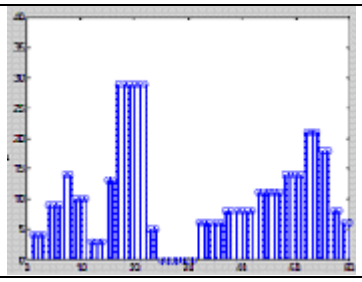
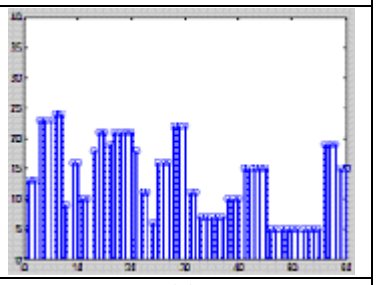
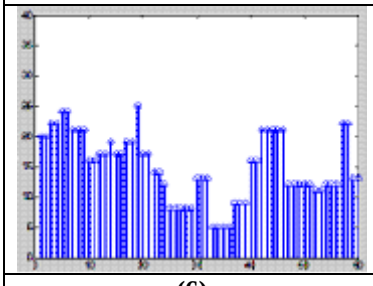
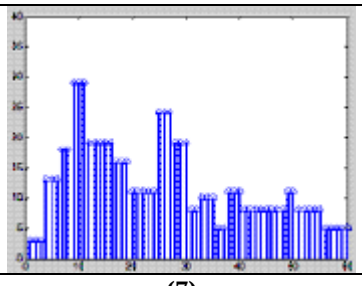
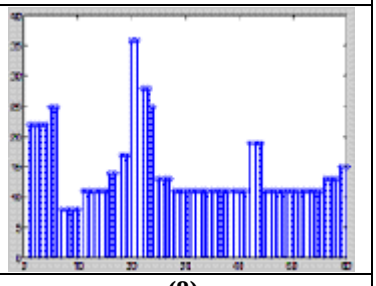
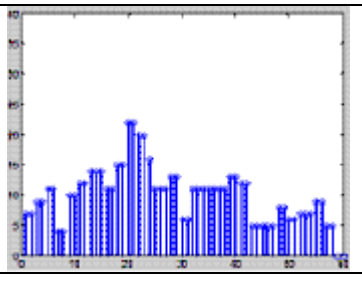
The first MLP experiments will be run over those input data. This way the number of neurons in input layer of the MLP is reduced to 60 and also the number of connections between input layer and hidden layer is reduced.

***The network and its performance:***

There is no exact definition for choosing the number of layers and number of neurons in the hidden layer of MLP, but on the other hand as mentioned before an MLP with one hidden layer and sigmoidal transfer functions is proven to solve a large number of problems in practice. This is aprropriate since a network as small as possible is desired in the real time system.

Another entity that is important for the neural network design is the number of neurons in the hidden layer. A proven technique doesn't exist for determining this

**Table 4** Projections samples of input numbers

| | | |
|---|---|---|
|  |  |  |
| **(0)** | **(1)** | **(2)** |
|  |  |  |
| **(3)** | **(4)** | **(5)** |
|  |  |  |
| **(6)** | **(7)** | **(8)** |
| |  | |
| **(9)** | | |

number but there exists a common sense that the number of neurons in the hidden layer should be equal to the number of inputs. This sense is also proven in the following experiments that the rate of succes doesn't increase linearly with the number of neurons in the input layer. In hidden layer the neurons have hyperbolic tangent sigmoid transfer functions, so that the inputs of the network can be pulled into different directions (1 , -1).

The output layer of the network has 10 neurons, each for a class of numbers. Each neuron in the output layer is responsible for being fired when the presented input

pattern is member of that class. A winner-take-all algorithm is applied to the outputs after the network finishes its calculations. In order to achieve this, each neuron in the output layer has log-sigmoid functions so that the outputs that are not similar to inputs are pulled to ‘0’ level and the output of neurons that is similar to the input is pulled near ‘1’. By getting the biggest output as the winner neuron the class that the input is most similar to is chosen.

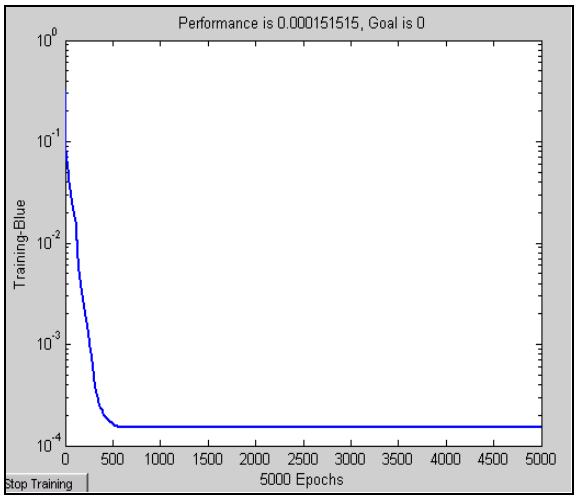



**Figure 6.2** Learning curve of MLP

The MLP structure with 30, 60 and 90 neurons in hidden layer is trained with scaled conjugate gradient training[7] method. The success rates for each of these networks are measured as %83.64 %85.29 and %90. The training curve for the network with 60 neurons in hidden layer is seen in Figure 6.2. It is seen that it converges to solution in 500 epochs.

It is not feasible to use the 90 neuroned structure since the %5 succes rate costs the number of connections to be tripled. It is more aprropriate to use the 30 neuroned structure but the experiments are achieved with 60 neuroned structure, in order to realize the concept of having the same number of neurons in hidden layer with the number of inputs, mentioned above.

Figure 6.3 is the results that are obtained from the test set of the trained neural network. This ladder formed output graph’s each step is a class of numbers starting from ‘0’ and ending with ‘9’. So it can be said that each disordered line in the ladder is an erroneous output. It is seen that most of the erroneous results are obtained

[7] Examine ‘trainscg’ method in MATLAB, more info can be found in [6]

between 8-3 and 6-0. This is an expected result since the projections of 8-3 and 0-6 look alike. The projection of pixel distributions concept for feature reduction yield in an accaptable result since the number of connections it contains is relatively small with the other concepts that are going to be examined in later sections.

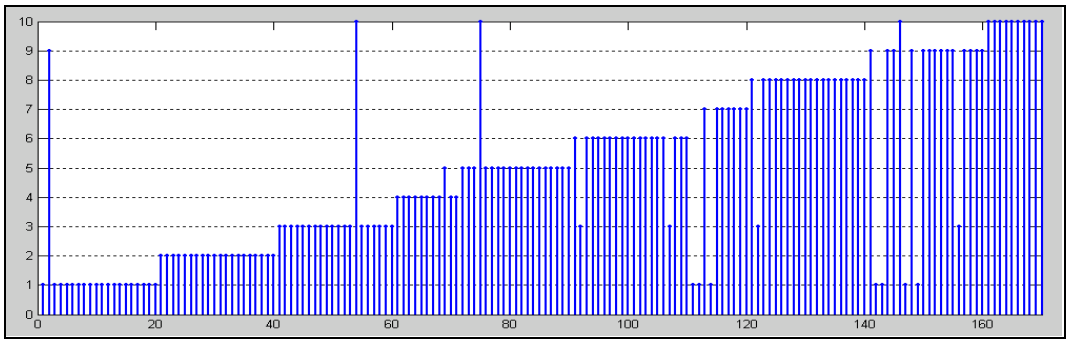



**Figure 6.3** Results of MLP

***Computational cost:***

As mentioned before the feedforward neural network has 60 inputs and 60 neurons in its hidden layer hence the number of connections between inputs and hidden layer is

$$60*60 = 3600\ w \tag{6.1}$$

There exists 10 neurons in the output layer so that

$$60*10 = 600\ w \tag{6.2}$$

is the number of connections between the hidden layer and the output layer. Including the number of neurons in each layer the total computational cost of the neural network becomes

$$4200\ w + 70\ n \tag{6.3}$$

where 70 is the total number of neurons that neural network has.

MLP is suitable for being used in the real-time system since it has a satisfactory performance and cost rates. On this account the MLP formed the neural network infrastructure of the prototype of the OCV system. The system performed approximately 2 plates of cards in one second in prototype system so that MLP performed well enough for being used in the test and pioneering phases of the system.

### 6.1.2 Multi MLP

An approach for increasing the success rate of the neural network is introduced in [4]. This approach defines a smaller neural network for each number digit. An input digit is presented each of these ten networks, each responsible for recognizing a number and the network that recognizes the input digit resembles the class of that number.

Smaller neural networks in this approach are again one hidden layered MLPs which have 30 neurons in hidden layer and 1 neuron in output layer. Those networks have the same type of transfer functions that are examined in previous part.

The same type of projectioned image inputs are used as inputs to multi MLPs and a success rate of %90 achieved. This way the %83 success rate is raised up to %90 but for each number, ten neural networks are needed to be run in each case. Since the input numbers are randonly distributed it can be stated that an avarage of 5 neural networks is needed for each digit. Using these as a priori the operational cost of the multi MLPs become

$$5* ( (60*30) w + (30 * 10) w + 40 n ) = 10500 w + 200n \quad (6.4)$$

where the operational cost of the single MLP with the same amount of success rate which has 90 neurons in hidden layer is

$$(60*90) w + (90 * 10) w + 100 n = 6300 w + 100 n \quad (6.5)$$

It is clear that the usage of multi MLP do not provide as good results as single MLPs so it is meaningless to use multi MLP in the OCR sub-system.

### 6.1.3 PCA and MLP

Actually, feature reduction is an important concept in pattern recognition. One of the strongest algorithms that perform this operation is principle component analysis (PCA)[8], which finds the eigenvectors that represent the data best. Instead of reinventing the steel, PCA can be used for discovering the strongest components of the input data. MATLAB[9] is used to find the principle component matrice of the input data and a new neural network constructed using this data.

[8] More info on PCA can be found in [5]
[9] See 'prepca' built-in function

***Inputs:***

The inputs of the new network are the outputs of the PCA. After the input vectors have been presented to PCA an input matrice of 116 features is obtained. It is obvious that we have more features in hand from PCA than the ones we obtained by projecting the images. Although this may seem a little disappointing the features that are obtained by PCA will definately represent the input data better and more successful results will be obtained.

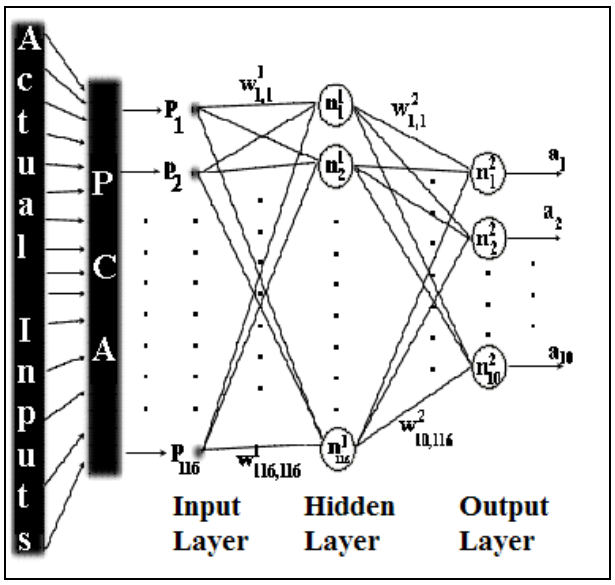



**Figure 6.4** MLP and PCA

***The network and its performance:***

The new neural network is no different than the network that has been constructed in the first part. The only difference is the neuron number in the hidden number which is equal to the number of inputs. Since the PCA has reduced the size of the input vectors to 116 there will be 116 neurons in the hidden layer and the connections between the input layer and the hidden layer will contain more connections.

There is also an additional operation before the feeature reduced inputs are presented to the network. This is the transformation of actual 864 featured input vectors to the 116 featured ones. This is done via the component matrice obtained from the PCA that is applied to the training input vectors.

The training performance of the new network is shown in Figure 6.5. It is obvious that the network has converged to the final result in 300 epochs which is a slightly better performance with respect to the first network introduced in first part.

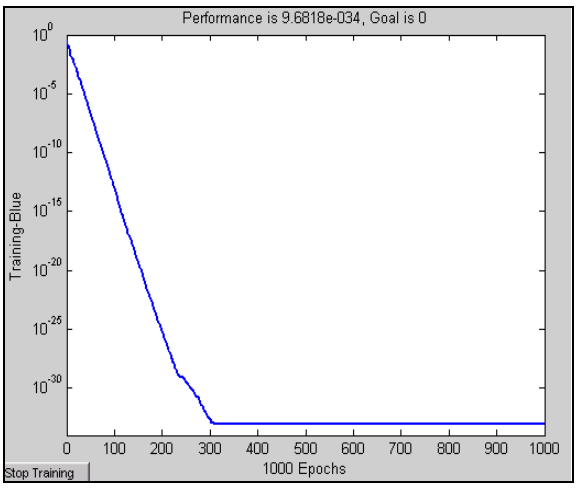



**Figure 6.5** Learning curve of MLP and PCA

After the network is trained with the inputs obtained from PCA the test set that is presented to the network and a success rate of %98.8 is performed. As seen in Figure 6.6 the network only has 2 erroneous results among 170 test inputs. These ones are the routine 8-3 confusion and a 7-3 confusion which is actually an intresting result.

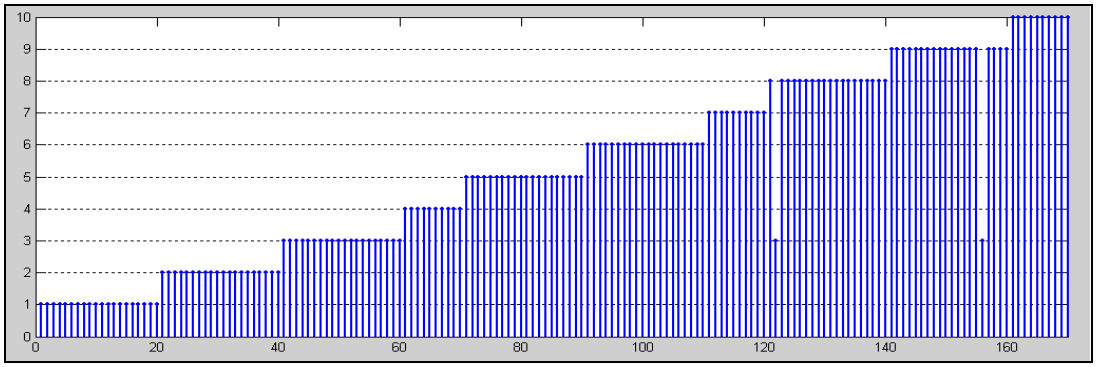


**Figure 6.6** Results of MLP and PCA

***Computational cost:***

The cost behind the enormous %98.8 rate is kind of a proof of performance-cost trade-off. Actually the neural network that has the same kind of structure with different number of neurons in the hidden layer has the computational cost of

$$(116*116)\,w + (116*10)\,w + (116+10)\,n + P = 14616\,w + 126n + P \qquad (6.6)$$

which is nearly 3 times larger than the MLP with 60 features inputs. Even if somehow the features from the PCA is reduced to 60, and the neural networks are

equalized in size there still exist a computational cost of PCA, which is denoted by ‘P’. This PCA cost consist the multiplication of component vector sized 864x60 and real input vector sized 864 to find the principle component values of the input image. This operation is actually

$$864*60= 51800 \quad (6.7)$$

multiplication operations and a number of addition operations which can be discarded here. The connection cost that are denoted by ‘w’ is nothing by the multiplication operation of two floating point numbers since the weighting operation in neural network connections is nothing bu multiplication. Therefore the additional 51800 multiplication operations can be seen as an additional 51800w to the neural network which means putting a computational cost nearly 12 times larger than the actual neural network operation which is inacceptable in our situation.

As seen in the microenvironment of MLP neural network efforts the performance-cost dilemma is always preserved. As the succes rate goes up the computational cost of the system also increases tremendously. This result will possibly be obtained at the end of the alternative approaches have been tried in the latter sections but the idea behind trying different techniques is performing the other advantages of alternative Networks and trying the catch the most appropriate performance-cost rate.

As mentioned earlier the computational cost hence the system speed is more important for OCV system than the success rate. %85 success rate is accaptable in designed system where the computational cost of the first MLP is nearly the lowest boundary for the interval of success that is determined for the system. Because of this reason the MLP is included in the prorotype system and is being used as the recognition system of the first version of the OCV software.

### 6.2 ART 1

The ART1 network [9] was appropriate for recognizing the number since it is known to be a good unsupervised classifier that models the vision system of the human body. It accepts binary data as input and groups the input data into classes real time. Real-time operation capability and the property of accepting binary data as input patterns make ART1 work well with the OCR sub-system.

Another important property of ART1 network is the vigilance parameter which can be altered in order for the ART1 network to tune its response for different inputs. Whenever ART1's classification performance decreases the vigilance parameter can be altered for the network to yield better results.

***Inputs:***

Since the network's real-time performance is important, it is not a very good idea to present ART1 the 864 pixels of the image as input so a preprocessing is necessary to reduce the size of the input vectors to the network. This size reduction must keep the basic information about the image but at the same time shrink the image as small as possible so that the size of the ART1 network gets smaller.

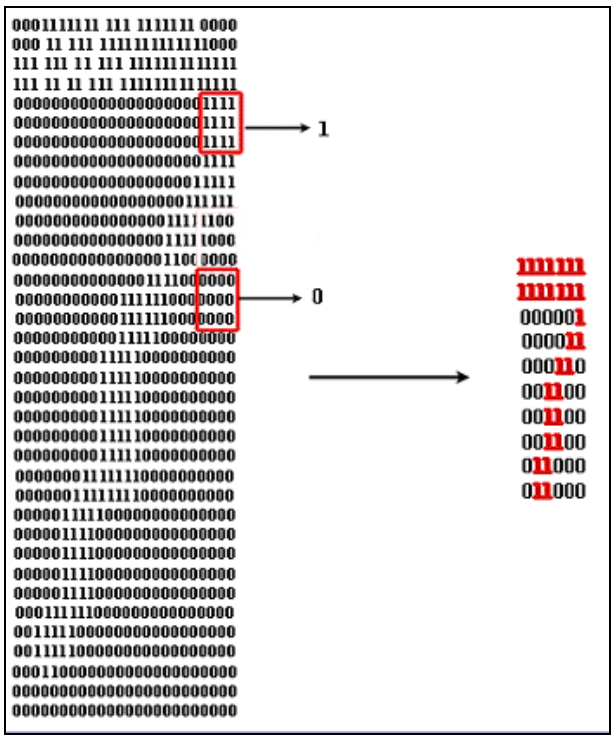



**Figure 6.7** Image shrinking operation

The size reduction operation is summarized in Figure 6.7. In size reduction operation the pixels in the real image are grouped into 4x3 sized boxes in order to form the new pixels in the shrinked image. In the shrinked image these 12 pixels stands for a '1' or a '0' according to the number of '1' it has inside. If it has more than a predefined number of '1's inside this box will be presented by a '1' in the shrinked image else it will stand for a '0'.

By applying this feature reduction operation the size of the input image reduces to 6x12 from 24x36 hence the size of the input vectors is reduced to 72 from 864. As seen in Figure 6.7 the reduction operation doesn't affect the overall look of the input image so it can be used instead of the real images to feed the ART1.

***The network and its performance:***

Designed ART1 network has the properties of the ART1 networks described in Section 5.4 and use the same mechanism to classify the inputs that are presented to it. It simply accepts the inputs via L1 and presents them to L2 to choose a former class that has been formed by predecessor inputs. The class is recalled form L2 to L1 and the similarity between the input and the chosen class is measured. The similarity is presented by a ratio between 0-1. I fthis ratio is smaller than the vigilance paramter orienting subsystem is triggered and the class is deactivated in L2 until the next input is presented. The current input is then matched with another input.

The ART1 network is prepared in MATLAB and the experiments are run with a test set of 700 different inputs which are obtained from the real system. As a result of the ART1 network the best performance is obtained with the vigilance parameter '0.3'. With this vigilance paramter the ART1 network has shown a %95 success in seperating the input classes from each other.

But in the other hand its clustering performance was not so successful. ART1 network output 27 different classes with its best performance while it should output 10. This problem is a common one in unsupervised networks, since there are no directions in order to teach the network how to classify the input patterns, the ART1 can arbitrarily divide an output class regarding the simillarities inside that class. In order to understand this performance better, Table 5 shows the output of ART1 for '0' class.

**Table 5** ART1 Classification of 70 '0' sample inputs

| **Class** | **Pattern** |
|---|---|
| 1 | 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 24, 25, 27, 30, 32, 33, 35, 36, 37, 40, 41, 42, 43, 50, 68 |
| 2 | 23, 26, 28, 29, 31, 34, 38, 39, 44, 45, 46, 47, 48, 49, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 69, 70, 71, 262, 265 |

The input patterns numbered between 0-70 were all ‘0’s in the test set for this output. As it can be seen there is only 71, 262 and 265 numbered patterns that are not standing for ‘0’s in this classification. But there exists 2 classes that represent number ‘0’. This situation must be handled in order to use the ART1 system in OCR sub-system. That means an extra mechanism for grouping the classes that are output by the ART1 system is needed. This clustering mechanism will wield to an extra cost in real-time system which is an unwanted situation for the performance of the system.

***Computational Cost:***

It is time to compute the cost of the ART1 system in words of sonnections and neurons. In order to compute this cost each layer of the ART1 network must be analyzed seperately. In this manner three different mechanisms exist to be analyzed.

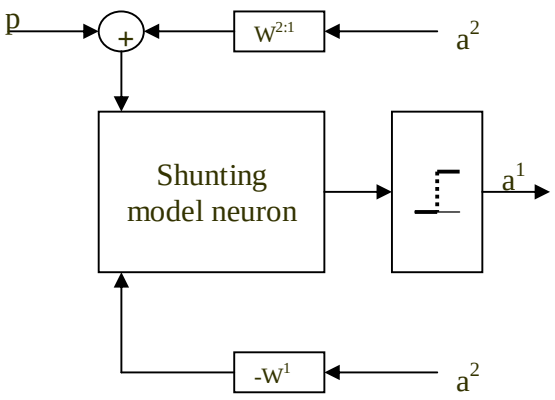



**Figure 6.8** A neuron in layer 1 of ART 1

These are:

*Layer 1 of ART1*

As seen in Figure 6.8 the L1 of the ART1 has the connections coming from L2 that are summed up with inputs to be presented to the L1 neurons. These L2 connections represent a class that is stored in L2 so it has 72 connections and 72 summation opertions which are nearly 1/3 in cost of multiplication. So the upper side of figure has

$$72+(72*1/3) = 96 \quad (6.8)$$

connection cost.

The lower side is another connection that is fed to L1 with negated L1 weights in order to provide inhibitory effect, compute the difference between the selected class and input pattern. This is a cost of 72 connections. Finally each neuron in this layer has

$$96 + 72 = 168 \quad (6.9)$$

connections.

In our network there exist 72 neurons in L1 so that:

$$72*168 = 12096 \quad (6.10)$$

connections and 72 shunting neuron operations exists , that is 12096w + 72sn

*Layer2 of ART1*

Each neuron in this layer has

$$72 + 72 + 24^{\varphi} = 168 \quad (6.11)$$

connection weight as seen on the upper side of the figure.

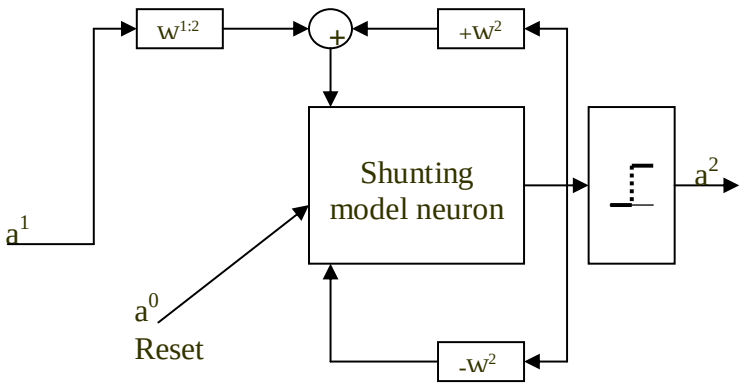



**Figure 6.9** A neuron in layer 2 of ART 1

Also a neuron has 72 more feedback connections as seen in the lower side.

L2 has variant neuron number, each representing a class in output so a precise number for neurons in L2 does not exist.

The determination of the number of neurons in L2 is a hard work. When the system is running there can be only 1 neuron activated in L2 that means a hit is catched in

[φ] Summation cost is aceepted nearly 1/3 of floating point multiplication

the first class choice of the network. On the other side there can be a number of trials since the best neuron is found which can maximum be the number of classes in L2.

The best case will be taken into account since the performance is not too well even for the best performance, that is

$$168w+1sn \tag{6.12}$$

*Orienting Subsystem*

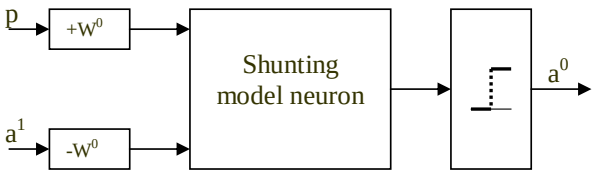



**Figure 6.10** A neuron in orienting subsystem of ART 1

In orienting system an undeterminable number of runs exist since it is not possible to foresee how many times it can send a signal to L2. Again the best case is taken into acoount where as seen in Figure 6.10 layer has

$$72+72=144 \tag{6.13}$$

connections present. So the cost of this layer becomes

$$144w+1sn \tag{6.14}$$

As a whole, the overall operational cost of the system is

$$12096w + 72sn + 168w+1sn + 144w+1sn = 12408w + 74\ sn + C \tag{6.15}$$

where ‘C’ is the clustering cost since the system outputs 27 different classes while it should output 10.

The biggest advantage of ART1 is its unsupervised working principle. By using ART1 no training for changes in input sets are required so that a huge amount of work for system update is eliminated.

Another advantage is its shunting model neurons which is also a disadvantage in operational cost as well. The shunting model neurons can be modeled as analog circuits because of their activation function's characteristics. By this way the prepared ANN can be realized with an analog hardware and can be embedded to the system. This way paralel computation can be supplied and the system will become more robust and the performance will increase.

On the other hand ART1 needs extra clustering operation in order to work properly hence need more computational power which is a critic source for the real-time system. Also the shunting model neurons need more computational power as mentioned before. These disadvantages are balanced by the properties presented above.

### 6.3 SOM

SOM was one of the mostly used ANNs in recognition system. Its power comes from the modeling of the input vector space over its neurons physically. This physical modeling is done by modifying the so called distances of the neurons. When the neurons are plotted using these logical distances, the output patterns are visualized.

On the other hand, just like unsupervised network ART1 in the former section, SOM also lacks clustering ability since there exist no outer directions feed into it. Because of this after the resulting maps are created by the SOM, there still exists a need for a clustering mechanism that transforms the neuron maps to clusters.

***Inputs:***

In order to keep the neuron map size small, the feature reduction mechanism that is introduced in the former section is used. By this way instead of creating a map of 24x36 neurons, a map with 6x12 neurons is created and the connections of this network are modified.

As examined in detail in Section 5.5, SOM works in such a way that it physically modifies the distribution of the neurons in its feature map so that the map holds the general distribution of the input data over the n-dimensional space of the data. In OCV system each input is a 72 dimensional vector so that the resulting map will be a 72 dimensional neuron space that each point in a space represents a possible distribution of the pixels in the input image.

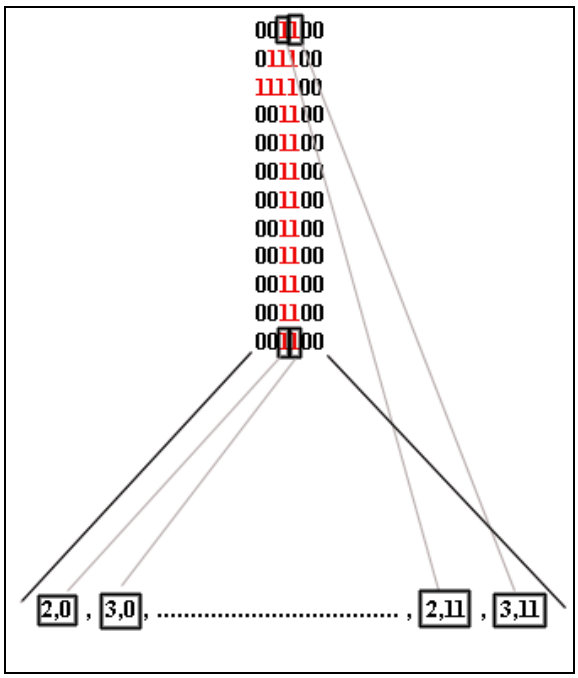



**Figure 6.11** Transformation of inputs for SOM

An output space that has 72 dimensions is hard to visualize and work on. Also the input data in hand is made of two dimensional pixel arrays, each holding the data if the pixel with the specified indice has a black pixel or a white pixel in it. This approach is very similar to the SOM approach. If the input is modified in such a way that each white pixel's x and y coordinates are fed into the network the resulting neuron map would represent the most common x and y coordinates in an input set.

In order to get the full pixel's coordinates each input vector in the test set is transformed into coordinate inputs as seen in the Figure 6.11. The transformation operation is simply searching for '1' in the input images starting from the bottom-left of the image and accepting this coordinate as (0,0). Each time a '1' is found in the image, a new coordinate couple is added to the new input vector of the map. This way each class of number is transformed separately into a new input vector and the test is applied on ten different SOMs each one is for keeping a number class.

***Network and its Performance:***

As explained in Section 5.5, SOM networks uses special types of connections called distances between its neurons to represent the input data that is presented to it via input neurons. In the tested SOM there are two input neurons since at each iteration

the pixel coordinates are fed into the network. As seen in the image Kohonen layer includes 72 neurons arranged as a 6x12 matrice just like the input images.

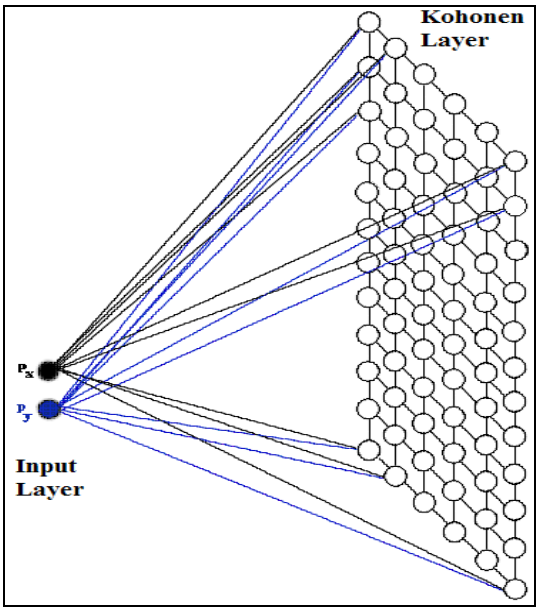


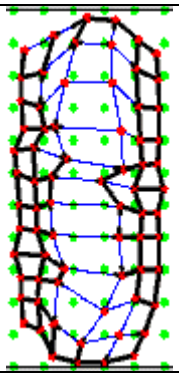
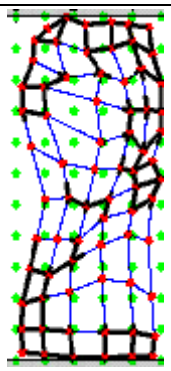
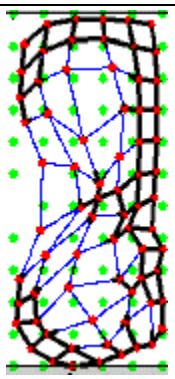
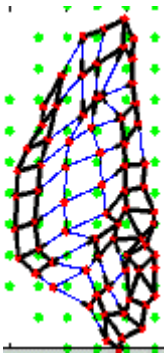
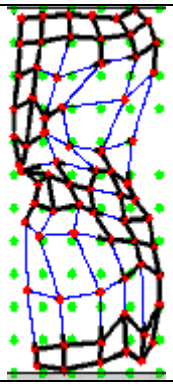
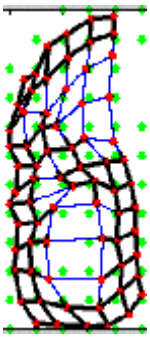
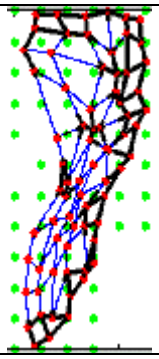
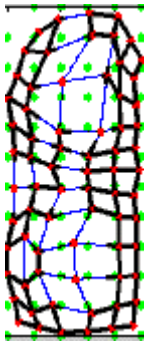
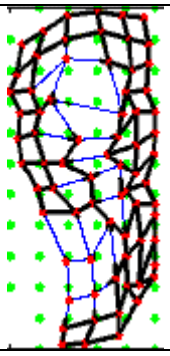

**Figure 6.12** SOM that has been designed

Ten different SOM is used in order to see each number's clustering separately. After feeding approximately 50 test inputs for each SOM the neurons in the final map are rearranged so that each map physically holds the data that is gathered form the input vectors.

As a result the following ten maps are obtained seen in Table 6. In each map every connection is showed with a link that is proportional to the value of the weight of the connection. If the weight is bigger the connection between two neurons is longer and if the weight is relatively small the length of the connection is shorter. Also the shortest connectios are showed boldly in order to see the physical arrangement of the neurons more clearly.

The resulting maps are satisfactory since it is seen that each corresponding map's neurons are arranged in a way representing the data. Like the former network ART the only task left is to transform those maps into classes with a supplementary mechanism.

**Table 6** Formation of SOMs after the input presentation

| | | |
|---|---|---|
|  |  |  |
| (0) | (1) | (2) |
|  |  |  |
| (3) | (4) | (5) |
|  |  |  |
| (6) | (7) | (8) |
|  | | |
| (9) | | |

The supplementary mechanism means a new layer to the network that is feed by the outputs of the Kohonen layer and gives out the classes for the corresponding maps. This layer can be achieved by using the self organizing motor maps that exactly puts a new layer of neurons in front of the map that reduces the number of neurons to the output class size so that each separate map distribution may yield in a neuron in the motor map's output layer. This mechanism is explained in detail in [12].

The wise way is to use another more common network which has a layer using Kohonen learning rule just like in SOM and a new supervised layer to classify the unsupervised layer's outputs. This network is called LVQ and the tests that are run with it is revealed in the next section.

***Computational Cost:***

The working principle of SOM is simpler than ART's, so that it is easier to calculate the computational cost of SOM. As easily seen in Figure 6.12 there exists 2 neurons in the input layer fully connected with the Kohonen layer. This structure forms

$$2*72=144 \tag{6.16}$$

connections between the input layer and the Kohonen layer. Also the Kohonen layer has distance connections between neurons. Since, in each row there exists 5 connections and there are 12 rows; and in each column there exists 11 connections and there are 6 columns, there are

$$(5*12)+(11*6)=126 \tag{6.17}$$

connections in Kohonen layer of the SOM.

There exist 72 neurons in the Kohonen layer so that the operation cost of the Kohonen layer so the SOM's total cost for each coordinate is

$$144\ w + 126\ w + 72\ n = 270\ w + 72n \tag{6.18}$$

In each image there exist approximately 40 white pixels. Regarding this, the operational cost for each input for the SOM multiplies by 40 and becomes

$$10800\ w + 2880n + C \tag{6.19}$$

where 'C' is the classification cost of the output.

SOM constructs the base for the LVQ networks in the next section. By testing the SOM and obtaining successful results the achievement of the LVQ is understood

better. Also SOM brings visual evidence to the working mechanism of the neural networks and hence is an important point in discovering the abilities of the neural networks.

### 6.4 Learning Vector Quantization

LVQ network is a hybrid network that uses both unsupervised and supervised layers to classify input patterns [13]. By having a structure like this, LVQ networks claim to overcome the clustering problem of the unsupervised networks.

LVQ networks' first layer simply seperate the input patterns from each other and group a number of output classes. The number of these classes may not always be equal to the desired number of classes. This problem is also faced in ART1 networks in which there existed 27 classes while it should only have 10. The second layer of the network is a supervised network and performs the combination or seperation of the output classes from the first layer in order to match the supervised output class number.

Having these properties the LVQ network seems to be appropriate for combine the success rate of the unsupervised networks and performance of supervised networks.

***Inputs:***

Since the LVQ network doesn't need specialized inputs because of its structure unlike ART and SOM, the projection features of the input images can be used without doubt just like in Section 5.6. There are nearly 700 inputs are used for training and 170 for testing the LVQ network.

***The network and its Performance:***

The LVQ network has 60-dimensional input vectors presented to it and in paralel has 60 neurons in its first layer. It uses Kohonen learning rule in its first layer and uses the distance between the input and the weights in order to achieve normalization. After the normalization, the competitive layer decides which class the input belongs or else if the input vector is a new class itself and output the result to second layer.

Second layer runs supervised learning to match the classes that are determined by the output of L1 with the classes that user defined. With the help of this operation the classes that are divided by the first layer which are actually members of the same target class are merged.

By using LVQ a success rate of %94.70 is obtained from the test input. This rate is not as good as the result of PCA and MLP. But with a %5 error rate, there might appear only one erroneous number recognition in every two number cards which is actually a very satisfactory result.

As seen in Figure 6.13 most of the erroneous results are seen between 3-8 , which can be reduced by controlling an extra feature to be sure if the recognized number is 3. By using an extra mechanism like this the success rate of the LVQ network will become larger, making it possible to use in the real time system if the performance is also satisfactory enough.

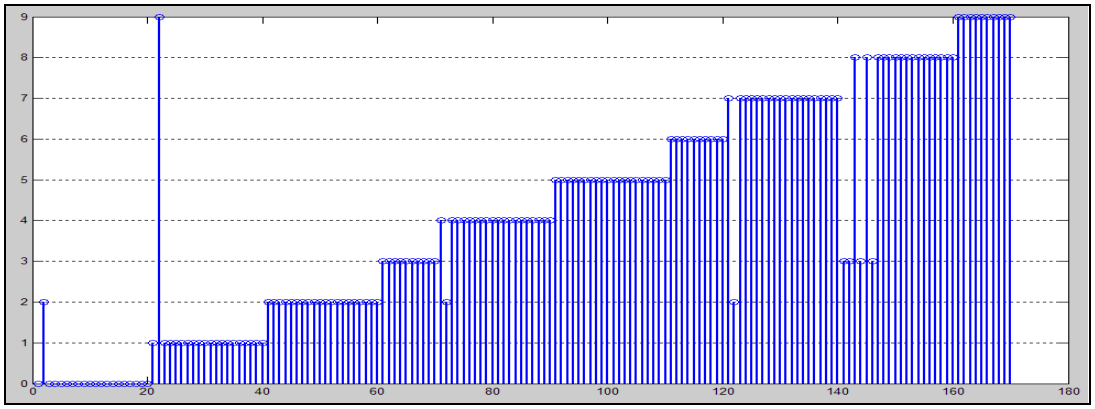



**Figure 6.13** Results of LVQ

***Computational cost:***

The number of neurons in the first layer of the network can be determined to maximize the performance. In order to keep the computational requirement of the network low enough the network must have as few neurons as possible in its first layer. Without affecting the success rate much the network works good enough with 20 neurons in its first layer. Actually the results above are obtained with a network that has 20 neurons in its first layer.

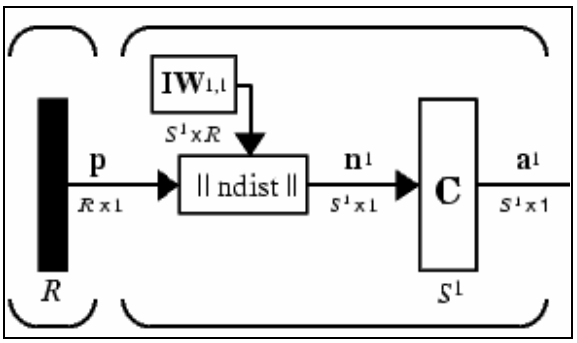



**Figure 6.14** Unsupervised layer 1 of LVQ

But before the input is directly feed into the first layer it goes through a normalization operation,as seen in Figure 6.14, which is done by using the euclidean distance between the weights and the input vector. This operation is nearly 2,6 times the normal connection operation[10] which is a simple multiplication. So that the cost of the first layer becomes

2,6 * (60*20) w + 20 c= 3120 w + 20 cn (6.20)

where cn is the competitive neuron cost of the neurons in the layer.

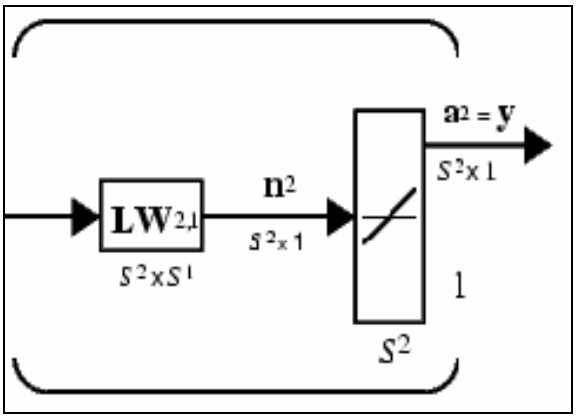



**Figure 6.15** Supervised layer 2 of LVQ

The second layer seen in Figure 6.15 is nothing but a simple supervised learning layer which is fed by the 20 neurons fully interconnected with the 10 neuroned second layer. This makes the second layer's operational cost:

(20*10) w + 10 n = 200w + 10 n (6.21)

[10] See 'dist' function in MATLAB

which makes the final cost of the network :

$$3320\ w + 20\ cn + 10\ n \tag{6.22}$$

which yields a cost nearly equal with the cost of the MLP.

Thus the LVQ network brings the best performance-cost rate among all the experimented networks so far. This is because of the hybrid approach of the LVQ network which uses the good sides of the supervised and unsupervised networks in order to form a new network that leaves the operation of seperating the input vectors to unsupervised Kohonen layer and operation of clustering the raw output classes of this layer to supervised layer.

## 6.5 Genetic Algorithms and Neural Networks

The idea behind using genetic algorithms in training neural networks is the optimization of the weight space of the neural network. As stated before, the training of the neural networks is nothing but finding the global minimas for error rates in the n-dimensional space constructed by the weights of the neural networks. This is just an optimization problem and solved by optimization functions like steepest descent and gradient descent algorithms.

Mentioned in Section 5.6 the genetic algorithms are the new generation global optimization functions. In this manner it is not a bad idea to use the genetic algorithms in training neural networks and finding the weights of a network that solves the stated problem best. Also by using genetic algorithms, a predefined structure for the training alogrithm is not necessary. This provides the reduction of the layer weights for improving the performance of the networks and finding weights of the network optimizing the success rate of the network.

***Inputs:***

A feedforward neural network will be used to be trained with GA. In order to keep the network size smaller the feature reduced inputs that are used in Section 6.1 will be used as inputs. There are 60 features that exist in these input vectors thus the first step in reducing the network size is taken.

***Genetic Operators :***

The basic operators and major elements of the genetic algorithm that is sued the train the network is as follows:

Individuals: Each chromosome(individual) in the genetic algorithm is made up of weights in the neural network. In order to easily interprete the weights that are coded in the chromosomes the weights and the biases of the network is coded seperately with their order in the neural network as seen in Figure 6.16. The chromosomes are coded real valued since this type of coding is more suitable for the nature of the problem.

Mutation: By mutation a random gene in a chromosome is changed with a random value. That mechanism ensures that the genetic algorithm is searching the problem space more messy and helps the algorithm avoid local minimas. Various mutation rates are used through the experiments but the best performance is obtained with a mutation rate of 0,6. It is natural for the mutation rate to be so high because there is a big number of variables in the search space that the algorithm runs on.

Reproduction: As the crossover algorithm of the chromosomes, multi-point crossover operator is used. By using multi-point crossover operator the weights of the two parent neural networks are fully blended to form a new breed. A single point operator doesn't provide good performance since the resulting neural network keeps too much of its parents' charasteristics. When the multi-point crossover is used a unique new breed is produced which inherits form its parents but is not completely a part of one of them.

Recombination: The heart of the recombination is the selection operator. In order to perform the selection, the tournament selection is used where a number of individuals are selected from the population randomly and the one with the better fitness is placed in the resulting population. The tournament selection that is used, selects two individuals from the population and runs a competition on them to select the one with the better fitness.

Fitness: The fitness of a neural network is naturally the success rate of the neural network over a defined number of inputs. In order to measure the success rate of the network the mean square error of the test inputs are calculated using

$$mse = \frac{1}{Q}\sum_{k=1}^{Q}(t(k)-a(k))^2 \tag{6.23}$$

where Q is the total number of outputs , t(k) is the target for output k and a(k) is tha actual output for output k.

***The network and its performance:***

The neural network that is going to be trained using the genetic algorithms must be a specially designed neural network. That is because the idea behind using genetic algorithms is training the neural networks that do not resemble any kind of neural networks so that it is hard to run a training algorithm on them.

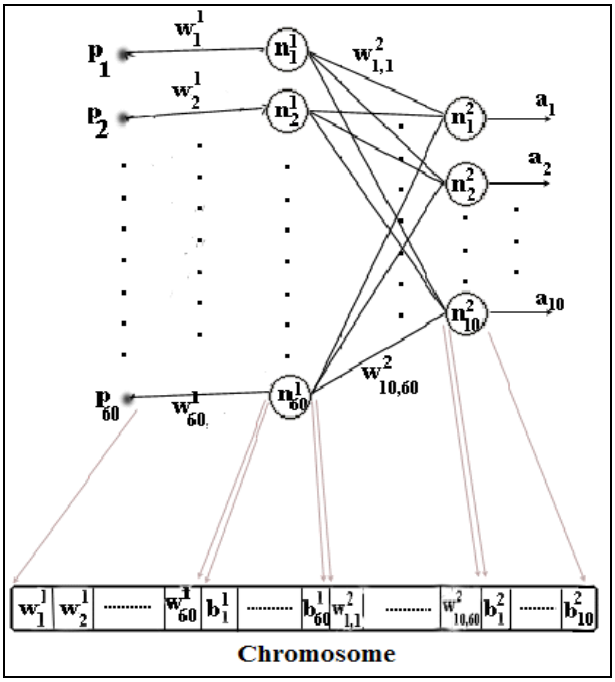



**Figure 6.16** Coding of neural network to be used with GA into a chromosome

Thus, the neural network that is going to be trained must have as small connections as possible but also have the potential to perform successful operation over the input

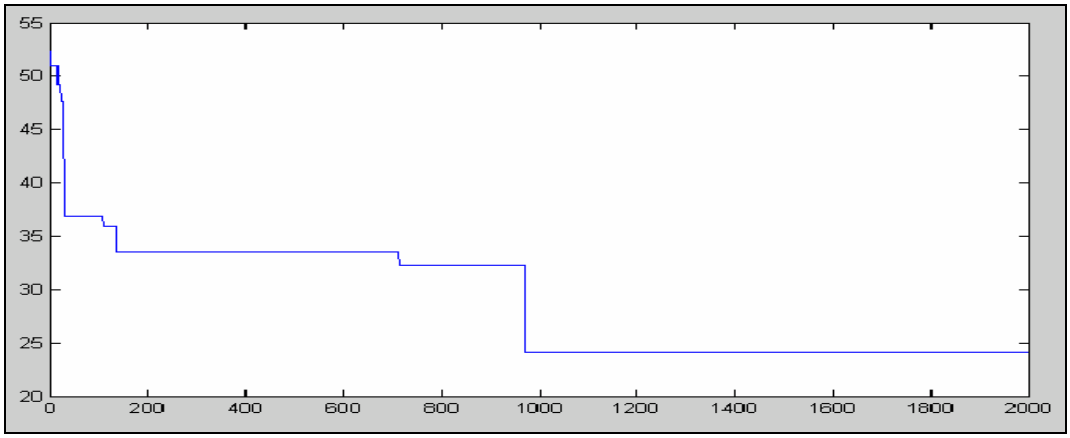



**Figure 6.17** Succes rates of best individual in each generation of GA

data. In order the perform this, the most costly part of the feedforward neural network that is used with backpropagation is pruned. As seen in Figure 6.16 the connections between the input layer and the hidden layer is pruned in such a way that each input is connected to the hidden layer with only one connection. By this way the number of connections in the layer with the largest connection size is reduced with a rate of nearly 1/60.

The chromosomes are coded in a way to resemble this structure so when a chromosome is taken into account for calculating fitness the corresponding neural network is formed easily. After a typical genetic run with 2000 iterations 0,6 mutation rate and with the tournament selection method the neural network performed %75 success rate with the training set and %70 success rate with the test set. This performance is foreseeable since many connections of the network that has performed %85 are missing from the network. But it can be seen as a success to reduce the success rate %15 while the %85 of the connections is missing. The error percentage of the neural network over iterations can be seen in the Figure 6.17. It is seen that the genetic algorithm actually converged to the result at the end of the $1000^{th}$ generation.

***Computational cost of the network:***

The network simply has 60 connections between the input layer and the hidden layer and 600 connections between the 60 neuroned hidden layer and 10 neuroned output layer. This makes the total computational cost of the neural network

$$660\ w + 70\ n \tag{6.24}$$

which is the smallest value that is obtained thorugh the experiments.

The genetic algorithm performed well enough for a relatively very small network. The %70 performance is the worst among the other neural networks but the performance-cost rate of the neural network is the largest among the other networks. This makes the network usable in applications where the error rate is not as important as speed. It is a foreseeable result that the success rate of the network will fall when the conveyor runs at higher speeds, so that this one is the appropriate network for higher conveyor speeds.

## 7. CONCLUSION AND FUTURE WORK

At the end of the exhausting studies for building an intelligent system that can be used in real-time printing industry there are many practical and theoretical consequences that are reached through the study. Dividing the thesis work in two parts just like in section one the arguements about the experiences that are earned can be performed in a more effective way.

In the first part, many important practical experiences about designing and constructing a real-time system, being limited by many obstacles, has been earned. Many unexpected problems occur before and during the development of the system and solving problems that are faced without any prevision is one of the most important concepts of engineering. Driving the synchronization unit over the serial port is one of these many problems. Many solutions are produced trying to drive the synchroniztion unit that uses the serial port, including win32 API[11] and MATLAB engine[12].

Another important issue about the first part is the experience of building a multi-camera system and processing the images that are acquired by the camera. Many important concepts about firewire camera driving and image processing is experimented throughout the thesis study and many possible problems are solved during these studies. These problems include acquiring images from the cameras and image segmentation techniques in order to preprocess the input data for the recognition engine.

As a summary for the first part, the outcome of developing an intelligent system with a team has built the practical infrastructure for the artificial experiments that are run through the thesis studies. Moreover it has been an excellent source for gaining experience on building a real-time system.

[11] See 'CreateFile'method in win32 API Reference at
http://msdn.microsoft.com/library/default.asp?url=/library/en-us/fileio/fs/createfile.asp
[12] See 'serial' function in external interfaces reference of MATLAB

Second part of the thesis was the theoretical part and include the performance-cost rate analysis of many neural networks and training techniques that are used. All these techniques that have been experimented and their performance-cost rates are summarized in Table 7.

**Table 7** Comparison of different ANNs

| **Method** | **Cost** | **~Performance** | **~# of operations** |
|---|---|---|---|
| MLP | 4200w + 70n | %85 | 13k |
| MLP+PCA | 14000w + 70n + P | %99 | 142k |
| ART1 | 11000w + 145n + C | %95 | 46k |
| LVQ | 9200w + 30n | %95 | 30k |
| GA + MLP | 660w + 70n | %70 | 2,5k |

Looking at Table 7 the performance-success rate trade-off that has been discussed in Section 5.2 is reached at the end of the experiments. The network with the best performance is the one that has been trained with the genetic algorithms since its design is very permissive. It has a %70 success rate which is not a promising rate for character verification but this network provides a solution for very high conveyor speeds where the success rate is not important much.

The network with the most promising performance-success rate is the multi layer perceptron trained with backpropagation. It has a simple feedforward structure and simply does its job with an accaptable success rate. As mentioned before this network is chosen for the current version of the OCV system and performs well in character verification task with moderate conveyor speeds.

The three networks that are experimented, provide high success rates but low neural network performances for a real-time system. The only supervied network among these is the one with the best performance. LVQ network is the most appropriate network when the success rate is very important perhaps to be used in logging of numbers printed on cards. But in order to use LVQ a slow conveyor speed is essential.

The unsupervised networks naturally performed worse since they have to learn to classify data by themselves. Hence they are obsolete in current system design. But in case of an hardware modification and system upgrade, with the use of the specialized hardware, they are the most appropriate choice because they omit the training

process which is essential when using supervised neural networks. They also lack a clustering system that merges the output classes of the neural network which is a subject to be considered when they are used.

As the conclusion of the theoretical work the multilayer perceptron that is trained with backpropagation is integrated to the character recognition system and is started to be used. Looking at the future aspects for the system the presentation of a catalog with diffrent performance-success rates to the user with system software seems to be a versatile solution for the OCV system. Also design of an highly paralel analog circuit with an ART 1 network running remains as a high level and extensive work for the future.

As a general approach and future work for all of the neural networks that are supposed to be used in recognition engine, neural network pruning techniques can be used in order to reduce the number of connections in the neural network by omitting ineffective connections over the output. The smaller the size of the network the more performance it performs. Another technique for constrcting smaller neural networks is using genetic programming [15] with a penalty function for network size. This way the network with the best success rate and the smallest size will be evolved among a number of neural networks.

One of the bottlenecks in character ecognition is the segmentation performance of the system. Even if the network with the highest proven recognition rates are introduced to the racognition engine it would be useless if the segmentation performance is insufficient. So one of the most important studies to construct a better system is to design a better segmentation algorithm. After a good segmentation has been performed less number of features that represent the input data better can be extracted from the acquired images.

It also must not be forgotten that the way to perform a better segmentation is a camera that can aquire better images from the system. Also a multi processing computer system will be able to run more complex neural network real-time that will increase the performance of the overall system. As usual, hardware updates are the shortest but expensive paths leading to solution of the performance problem.

For an overall conclusion neural networks are for sure the best solutions for character recognition which can perform in the level of %99. For the designed real-time

system, if used alone, they can be used when the time requirement is in moderate level. They work well enough for character verification where a success rate over %80 is acceptable. But if the requirement goes up over %90, hardware updates or usage of cooperative software techniques are required.

The studies that are performed during the thesis was a complete computer engineering challange where the design of the hardware and software units, develeopment and deployment of them, performance considerations and quality enhancements over almost all parts of the intelligent system have been performed. Therefore the study took its place among the milestones of the occupational and academic studies of the author as it is supposed to be.

## REFERENCES


**[1] R. P. Lippmann ,** "Pattern Classification Using Neural Networks", IEEE Communications Magazine , November 1989

**[2] R. M. Gray,** ``Vector Quantization," *IEEE ASSP Magazine*, pp. 4--29, April 1984.

**[3] G. Arfken,** "The Method of Steepest Descents." 7.4 in *Mathematical Methods for Physicists, 3rd ed.* Orlando, FL: Academic Press, pp. 428-436, 1985

**[4] S. Lucas, Z.Zhao, G. Cawley, P.Noakes,** "Pattern Recognition with the Decomposed Multilayer Perceptron",Electronic Letters, Vol.29 No.5,March 1993

**[5] I.T. Jolliffe,** "Principle Component Analysis", Spring-Verlag, New York, 1986.

**[6] Martin F. Møller,** "A Scaled Conjugate Gradient Algorithm for Fast Supervised Learning", November 1990

**[7] M.T. Hagan, H.B. Demuth, M. Beale,** "Neural Network Design", PWS Publishing Company, 1996

**[8] S.Grossberg, E.Mingolla, D.Todorovic,** "A neural network architecture for preattentive vision", IEEE Transactions on Biomedical Engineering, vol.36 no.1 January 1989

**[9] G.A. Carpenter, S.Grossberg,** "A massively parallel architecture for a self organizing neural pattern recognition machine", Computer Vision, Graphics, and Image Processing, vol.37 1987

**[10] G.A. Carpenter, S.Grossberg,** "ART2:Self Organization of Stable Category Recognition Codes for Analog Input Patterns", Applied Optics, vol.26 no.23 1987

**[11] T. Kohonen,** "Self Organized Formation of Topologically Correct Feature Maps", Biological Cybernetics 43,59-69, 1982

**[12] H. Ritter, T. Martinez, K. Schulten,** "Neural Computation and Self Organizing Maps:An Introduction", Reading, MA: Addison-Wesley, 1992

**[13] T. Kohonen,** "Self Organization and Associative Memory", New York:Springer-Verlag, 1989

**[14] S.A. Harp, T. Samad,** "Genetic Synthesis of Neural Network Architecture", Handbook of Genetic Algorithms, 202-221, New York: Van Nostrand Reinhold, 1991

**[15] J.R. Koza, J.P. Rice,** "Genetic Generation of Both the Weights and Architecture for a Neural Network", International Joint Conference on Neural Networks, {IJCNN}-91, 397-404, 1991

**[16] C.T. Lin** , "Neural fuzzy control systems with structure and parameter learning", Singapore ; River Edge, NJ : World Scientific, pp. 235 ,1994

**[17] J. Barroso, A. Rafael, E. L. Dagless, and J. Bulas-Cruz,** "Number plate reading using computer vision", *IEEE International Symposium on Industrial Electronics*, 1997.

**[18] P.G. Williams , H.R. Kirby , F.O. Montgomery , R.D.Boyle** , "Evaluation of video-recognition equipment for number plate matching", 2nd International conference on Road Traffic Monitoring. Institute of Electrical Engineers , February 1989.

**[19] W. McCulloch, W. Pitts ,** "A logical calculus of the ideas immanent in nervous activity" Bulletein of Mathematical Biophysics, vol. 5, pp. 115-133, 1943

**[20] R. O. Duda and P. E. Hart,** "Pattern Classification and Scene Analysis", NY:John Wiley and Sons, 1973.

**[21] K. Fukunaga,** "Introduction to Statistical Pattern Recognition", NY: Academic Press, 1972.

**[22] J. Moody , C. Darken,** "Fast Learning in Networks of Locally-Tuned Processing Units," Tech. Rep. YALEU/DCS/RR-654, Yale Computer Science Department, New Haven, CT. Oct. 1988.

**[23] A. Rojer , E. Schwartz,** "A Multiple-Map Model for Pattern Classification." Neural Computation, vol. 1(1), pp. 104-1 15, 1989

**[24] R. P. Lippmann,** "An Introduction to Computing with Neural Nets," IEEE ASSP Mag., vol. 4 (2). pp. 4-22, Apr. 1987

**[25] S. J Hanson , D. J. Burr,** "Knowledge Representation in Connectionist Networks," Tech. Rep., Bell Communications Research, Morristown, New Jersey, Feb. 1987.

**[26] I. D. Longstaff , J. F. Cross,** "A Pattern Recognition Approach to Understanding the Multi-Layer Perceptron," Memo. 3,936, Royal Signals and Radar Establishment, July 1986.

**[27] W. M. Huang and R. P. Lippmann,** "Neural Net and Traditional Classifiers," Neural Info. Processing Syst.. D. Anderson, ed., pp. 387-396, NY: American Institute of Physics, 1988.

**[28] A. Wieland and R. Leighton,** "Geometric Analysis of Neural Network Capabilities," IEEE lst Int'l. conf on Neural Networks. pp. 111-385. June 1987.

**[29] G. Cybenko,** "Approximation by Superpositions of a Sigmoidal Function." Mathematics of Control, Signals. and Systems, 2(4), 1989

**[30] S. Grossberg** , "Adaptive pattern classification and universal recoding : I. Parallel development and coding of neural feature detectors" Biological Cybernetics , 23:121-134 , 1976

**[31] S. Grossberg ,** "Adaptive pattern classification and universal recoding : II. Feedback, expectation, olfaction, illusions" Biological Cybernetics , 23:187-202 , 1976

**[32] D. E. Goldberg,** "Genetic Algorithms in Search, Optimization and Machine Learning", Addison Wesley Publishing Company, January 1989.

**[33] S. Grossberg ,** "Nonlinear neural networks: Principles, mechanisms and architectures", Neural Networks 1:17-61 , 1988

**[34] G. Toussaint ,** "Pattern Recognition Lecture notes: Chapter 1 - An Introduction to pattern recognition", http://cgm.cs.mcgill.ca/~godfried/teaching/pr-notes/

## CURRICULUM VITAE

Tolga Ovatman was born in Bursa, on the 13th of June, 1981. He received his B.Sc. degree from Hacettepe University, Department of Computer Engineering in 2003. He has been working at İstanbul Technical University Department of Computer Engineering as a research and teaching assistant since then. His areas of interest include but not limited to: artificial neural networks, evolutionary computation and intelligent systems.