ملخص الرسالة

لقد اهتم الباحثون اهتماما كبيرا بعلم الشبكات العصبية خلال السنوات القليلة الماضية. ذلك لأن هذا العلم قـــد اسهم فى تقديم حلول لمشاكل متعددة فى تطبيقات هامة مثل إدراك الأنمـــاط (Pattern Recognition) ، الذاكرة الترافقية (Associative Memory) والتحكم الآلي (Automatic Control) . من بـين هذه التطبيقات يهتم هذا البحث باستخدام الشبكات العصبية فى بنــــاء الحاكمـــات الآليـــة (Automatic Controllers) ويعرف هذا بعلم تحكم الشبكات العصبية (Neurocontrol) وقد تلقى هذا العلـــم ، هو الأخر ، اهتماما بالغا خلال السنوات القليلة الماضية من مهندسى التحكم الآلي.

نستخدم فى هذا البحث أسلوبا محددا من أساليب تحكم الشبكات العصبية وهو: بناء الحـــاكم الآلى باســـتخدام حاكم مرشد (Template Learning Neurocontrol Approach) . يقتضي هذا الأســـلوب استخدام حاكم مرشد (Template Controller) قادر على التحكم فى النظام المراد التحكم فيه آليـــا ، ويبنى الحاكم الآلي عن طريق محاكاة هذا الحاكم المرشد أثناء قيامه بالتحكم فى النظام . وقد اخترنا فى هذا البحـــث أذكى وأقوى حاكم مرشد عرف إلى الآن وهو المشغل الآدمـــي (Human Operator) . إن الغـــرض الأساسي من هذا البحث هو معرفة مدى إمكانية استخدام قدرات المشغل الآدمي فى بناء حاكمات آلية للأنظمـــة المعقدة التي تكون قادره على التحكم فى أدائها.

تتضمن هذه الطريقة ثلاثة مراحل لتصميم الحاكم الآلي وهى كما يلي :

- **مرحلة الملاحظة**: فى هذه المرحلة يقوم الحاكم المرشد بالتحكم فى أداء النظام وأثناء هذا يتم تسجيل ما يقـــوم بعمله من أفعال (Actions) مع الحالة المقابلة الموجود عليها النظام (Process State) فى صـــورة أمثلة للداخل الى والخارج من الحاكم المرشد.

- **مرحلة النمذجة** : ويتم فى هذه المرحلة استخدام البيانات المسجلة أثناء مرحلة الملاحظة واستخدامها فى تدريب (Training) نموذج للمشغل الآدمي والمبنى باستخدام شبكة عصبية . والهدف من هـــذه المرحلـــة هـــو إمكانية نمزجه دور المشغل الآدمي.

- **مرحلة اختبار النموذج** : تتم فى هذه المرحلة اختبار نموذج المشغل الآدمي وتقييم أدائه.

بناء على نتائج مرحلة الاختبار قد يتم المرور بمرحلة النمزجة مرة أخرى لتدريب النموذج اكثر أو حتى لتغـــيره . ويكرر هذا حتى الحصول على الأداء المطلوب من النموذج.

خلال هذا البحث ، تم تطبيق الأسلوب على نظامين محل دراسة وهما :

١. نظام التحكم فى مسار سفينة (Ship Steering Problem).

٢. نظام التحكم فى اتزان البنـــدول المقلـــوب (Inverted Pendulum or Pole balancing Problem)

وقد تم هذا عن طريق المحاكاة على الكمبيوتر . ثبت بالمحاكاة أن الحاكم الآلي الناتج قادر على التحكم فى الأنظمـــة المذكورة بأداء مقبول.

إن استخدام المشغل الآدمي كحاكم مرشد يمثل أسلوبا إيجابيا حيث يكون ، فى أغلب الأحيان ، قادر على قيـــادة الأنظمة الصعبة والهيمنة عليها . كما يعتبر هذا توريث لقدرات الإنسان للآلة والذي يعتبر من أهم أهداف علـــم الذكاء الاصطناعي . أيضا ، من إيجابيات هذا الأسلوب عدم الحاجة الى النموذج الرياضي للنظام المراد التحكـــم فيه مما يجعله مناسبا للأنظمة الغير خطية والمعقدة .

الشبكات العصبية التى استخدمت لبناء نموذج المشغل الآدمي هي شبكات التغذية الأمامية (Feed-Forward Neural Networks) وهى كالآتي :

١. (Radial Basis Function Networks) ، استخدم نوعان منها .

٢. (Multiiple Layer Perceptron Networks) ، استخدم نوعان منها.

بمقارنة النتائج وجدنا الآتي :

- كل من المشغل الآدمي ونموذجه قادر بمفرده على التحكم فى مسار السفينة والحفاظ على اتزان البنـــدول (لفترة محدودة ومناسبة).
- أداء النموذج كان أنعم (Smoother) الى حد ما من أداء المشغل الآدمي.
- النوع الأول من الشبكات العصبية كان أكثر ملائمة لنظام السفينة .
- النوع الثاني من الشبكات العصبية كان أكثر ملائمة لنظام البندول .

Summary

Neural networks have received attention in the research community in recent years. That networks have been shown to elegantly and powerfully realize solutions to problems in pattern recognition, associative memory, and database retrieval. Among neural networks application is : Neurocontrol, which is a dynamic research field that has attracted considerable attention from control engineering community in the last several years. The field searches the use of neural network in automatic controllers developing.

In this research the template learning neurocontrol approach is discussed as a new method for automatic controller design. In this method, a template controller, which is able to control the investigated system, is used to design a neurocontroller for that system. The template controller chosen, is the most intelligent controller known until now- *Human Operator(HO)*.The research showed that human operator factors can be utilized in building automatic controller for complex nonlinear systems. The method entails three phases in controller design process, they are:

Observation Phase: In this phase, the template controller is used to control the investigated system. The actions taken by the template together with the corresponding system states are registered in the form of input output examples.

Modeling Phase: In this phase, the registered data during the observation phase is used to train the *Human Operator Behavior Model(HOBM)* which is a neural network. The aim of this phase is the approximation of the HO role through the HOBM.

Testing Phase: In this phase the trained model (HOBM) is used to control the system and its performance is evaluated.

Based on the results of the testing phase the modeling phase may be revisited by doing more training, or even modifying the model . The process is repeated until the desired performance is obtained.

In our work, the method was applied on two selected benchmark control problems- ship steering problem and pole balancing problem through computer simulation. It's proved, through the simulation, that the resulting neurocontroller is able to control the system with accepted performance.

Using the HO as a template controller is one of the positive points of this method. This because, the HO is often expert in keeping the complex control system on the right track. Also, machine inheritance of human factors is one of the most important goals of the dynamic research field: *Artificial Intelligence,* this work can be considered as a step on the way. Another positive point for that method is that, there is no need for process mathematical model and its suitability for nonlinear systems.

The neural networks structures used in this work to implement the HOBM are feed forward networks and they are as follows:

- Radial Basis Function neural network (RBF): with the basis function segma parameter is a vector of n dimension, where n is the number of inputs to the network

- Radial Basis Function neural network (RBF): with the basis function segma parameter is a scalar.
- Multiple layers perceptors(MLP) feed forward neural network single hidden layer: with bipolar sigmoidal function as the activation function.
- Multiple layers Perceptrons(MLP) feed forward neural network two hidden layers:with bipolar sigmoidal function as the activation function.

Comparing the responses of the HO and the HOBM for both ship steering and pole balancing problems we found that:

- Both the neural controller and the HO are able to steer the ship and balance the pole(for limited time interval).
- The control signal provided by the neural controller is smoother than that provided by the HO.
- The best approximation for ship steering problem was achieved using the RBF vector-segma feed-forward network with single hidden layer, this structure seemed to be the best, among the discussed four structures, for that problem.
- The best approximation for pole balancing problem was achieved using the MLP feed-forward with single hidden layer, this structure seemed to be the best, among the discussed four structures, for that problem.

Table of Contents

Mansoura University
Faculty of Engineering
Computer & Systems Engineering Dept.

COMPUTER BASED MODEL FOR HUMAN OPERATOR AND ITS APPLICATIONS IN CONTROL SYSTEMS

thesis

Submitted in Partial Fulfillment of the Requirements for Degree of Master of Science

In

Automatic Control Engineering

by

Eng, Hamed Mohamed Hemeda

Under Supervision of

Dr.
Mohamed Sherief
Computer & Systems Engineering Dept.
Faculty of Eng. Mansoura University

Associate Prof. Dr
Mostafa M. Abd Elnaby
Electronics & Communication Dept.
Faculty of Eng. Mansoura University

Prof. Dr.

Y.M. Enab
Computer & Systems Engineering Dept.
Faculty of Eng. Mansoura University

1999

SUPERVISORS

Dissertation Tittle

COMPUTER BASED MODEL FOR HUMAN OPERATOR AND ITS APPLICATIONS IN CONTROL SYSTEMS

Researcher Name

Hamed Mohamed Hemeda

SUPERVISORS

Name	Position
Prof. Dr. Y.M. Enab [illegible]	Control & Computer Dept Faculty of Eng. Mansoura University
Associate Prof. Dr. M. Abd Elnaby [illegible]	Electronics & Communication Dept. Faculty of Eng. Tanta University
Dr. M. S. Mostafa [illegible]	Control & Computer Dept Faculty of Eng. Mansoura University

EXAMINATION COMMITTEE

Dissertation Tittle

COMPUTER BASED MODEL FOR HUMAN OPERATOR AND ITS APPLICATIONS IN CONTROL SYSTEMS

Researcher Name

Hamed Mohamed Hemeda

EXAMINATION COMMITTEE

Name	Position
Prof. Dr. Ebrahiem E. Zidan	Faculty of Eng. El-Zakaziek University
Prof. Dr. Fayez F. Gomaa	Computer & Systems Engineering Dept. Faculty of Eng. Mansoura University
Prof. Dr. Y.M. Enab	Computer & Systems Engineering Dept. Faculty of Eng. Mansoura University
Associate Prof. Dr M. Abd Elnaby	Electronic & Communication Dept Faculty of Eng. Tanta University

Examination Date : 6/9/ 1999

Signature

Name	Signature
Prof. Dr. Ebrahiem E. Zidan	[illegible]
Prof. Dr. Fayez F. Gomaa	[illegible]
Prof. Dr. Y.M. Enab	[illegible]
Associate Prof. Dr M. Abd Elnaby	Mustafa Mahmoud

ACKNOWLEDGMENT

Firstly, I wish to thank Prof. Dr. Y.M. Enab for his supervision and helpful advice. I would like to express my deep thanks to Associate Prof. Dr. M. Abd Elnaby and Dr. Mohamed Sherief Mostafa for their encouragement during the work. I like also to extend my sincere gratitude to all who helped me to complete this work by their guidance and support in times I needed help. Also, to all those assisted me in this thesis in Mansoura university or in Tanta university.

Summary

Neural networks have received attention in the research community in recent years. That networks have been shown to elegantly and powerfully realize solutions to problems in pattern recognition, associative memory, and database retrieval. Among neural networks application is : Neurocontrol, which is a dynamic research field that has attracted considerable attention from control engineering community in the last several years. The field searches the use of neural network in automatic controllers developing.

In this research the template learning neurocontrol approach is discussed as a new method for automatic controller design. In this method, a template controller, which is able to control the investigated system, is used to design a neurocontroller for that system. The template controller chosen, is the most intelligent controller known until now- *Human Operator(HO)*.The research showed that human operator factors can be utilized in building automatic controller for complex nonlinear systems. The method entails three phases in controller design process, they are:

Observation Phase: In this phase, the template controller is used to control the investigated system. The actions taken by the template together with the corresponding system states are registered in the form of input output examples.

Modeling Phase: In this phase, the registered data during the observation phase is used to train the *Human Operator Behavior Model(HOBM)* which is a neural network. The aim of this phase is the approximation of the HO role through the HOBM.

Testing Phase: In this phase the trained model (HOBM) is used to control the system and its performance is evaluated.

Based on the results of the testing phase the modeling phase may be revisited by doing more training, or even modifying the model . The process is repeated until the desired performance is obtained.

In our work, the method was applied on two selected benchmark control problems- ship steering problem and pole balancing problem through computer simulation. It's proved, through the simulation, that the resulting neurocontroller is able to control the system with accepted performance.

Using the HO as a template controller is one of the positive points of this method. This because, the HO is often expert in keeping the complex control system on the right track. Also, machine inheritance of human factors is one of the most important goals of the dynamic research field: *Artificial Intelligence,* this work can be considered as a step on the way. Another positive point for that method is that, there is no need for process mathematical model and its suitability for nonlinear systems.

The neural networks structures used in this work to implement the HOBM are feed forward networks and they are as follows:

- Radial Basis Function neural network (RBF): with the basis function segma parameter is a vector of n dimension, where n is the number of inputs to the network

- Radial Basis Function neural network (RBF): with the basis function segma parameter is a scalar.
- Multiple layers perceptors(MLP) feed forward neural network single hidden layer: with bipolar sigmoidal function as the activation function.
- Multiple layers Perceptrons(MLP) feed forward neural network two hidden layers:with bipolar sigmoidal function as the activation function.

Comparing the responses of the HO and the HOBM for both ship steering and pole balancing problems we found that:

- Both the neural controller and the HO are able to steer the ship and balance the pole(for limited time interval).
- The control signal provided by the neural controller is smoother than that provided by the HO.
- The best approximation for ship steering problem was achieved using the RBF vector-segma feed-forward network with single hidden layer, this structure seemed to be the best, among the discussed four structures, for that problem.
- The best approximation for pole balancing problem was achieved using the MLP feed-forward with single hidden layer, this structure seemed to be the best, among the discussed four structures, for that problem.

Table of Contents

CHAPTER 1

INTRODUCTION

Neural networks have received great attention in the research community in recent years. Then networks have been shown to elegantly and powerfully realize solutions to problems in pattern recognition, associative memory, database retrieval, and automatic control. In this chapter, we first introduce an overview of neural networks, then a survey of classical control will be introduced . The aim of this chapter is not to provide complete but enough information to be the basis of the following chapters

1.1 NEURAL NETWORK OVERVIEW

The term neural network is used to describe various topologies of highly interconnection of simple processing elements that offer an alternative to traditional approaches to computing. The Tactical Technology Office of the U.S. Defense Advanced Research Projects Agency(DARPA) recently completed a study of neural networks[1]. DPRPA was interested in determining the current status and potential usefulness of neural networks, and assessing the worth of investing more time, energy and money. The DARPA study reached the following conclusions[2].

- Neural networks offer important new approaches to information processing because of their adaptability to learn as well as their massive parallelism.
- Neural networks research has matured greatly since the perceptron of the 1950's. This maturation has three resources:(a) advancement in mathematical theories, (b)

development of new computer tools, and (c) increased understanding of neurobiology. There is a bridge forming between the biologically—oriented neural modelers and the artificial neural network modelers, and substantial progress will result from this alliance.

- Although the computer revolution of the last 20 years has played a critical role in the revival of neural network research by making it possible to undertake much work in mathematical theory. The limits of today's computing devices- inadequacies in storage, speed, and user flexibility- have restricted present neural network efforts.
- There have been significant demonstrations of neural network capabilities in vision, speech, signal processing, and robotics. Perhaps not to the scale that might be desired-due to a lack of funding of research-but the variety of problems addressed by neural networks is impressive.
- Hardware capabilities are limiting the development of important neural network applications. Clearly, if researchers are provided with improved simulation and implementation capabilities, the field of neural network will once again drift off into the wilderness.

1.1.1 Neural Networks vs. Traditional Computers

Robert Hecht_Nielson, of the Hecht_Nielson Neurocomputer corporation, defines a neural network as "a dynamical system with a topology of directed flow graph that can carry out information processing by means of its state response to continuos or episodic inputs "[3]. A neural network is fundamentally different from a traditional computer. Neural networks

- do not sequentially execute instructions nor do they contain memory for storing operating structure or data
- respond in parallel to a set of inputs
- are more concerned with transformations than algorithms and procedures
- do not contain only one or a few complicated computational devices, but are composed of a large number of simple devices oven doing little more than computing weighted sums.

Information in neural network is not stored in explicit memory locations, but is represented by the interconnection strengths and the current output of the neurons. Neural networks do not replaceme traditional computers, but are an entirely different class of computational devices capable of qualitatively different tasks. Neural networks provide a completely new unique way to look at information processing. To fully appreciate and understand neural networks they must not be cast into the traditional view of computing machines. They are vastly different, and must be viewed as such.

Also, while many neural networks are currently being simulated on computers, they should not be considered a new programming technique. Neural network applications are often explored via computer simulations, but these simulations are only tools for numerical analysis of the network. Given the complexity of neural network architectures, computer simulations are inefficient ways to implement neural network architectures. Neural networks hardware will have to be developed before neural networks are effectively implemented.

1.1.2 Basic Elements and Terminology

Fig. 1.1 shows the biological neuron model upon which neural networks are based[1]. Each neuron in the brain is composed of a *body*, one *axon*, and multitude of *dendrites*. The dendrites form a very fine "filamentary brush" surrounding the body of the neuron. The axons can be thought of a long thin tube that splits into branches terminating in little *endbulbs* almost touching the dendrites of other cells. The small gap between an endbulb and dendrite is called a *synapse*. The axon of a single neuron forms synaptic connections with many other neurons. Fig. 1.1 shows the body and the axon of one neuron interacting with the dendrites of a second neuron. Impulses propagate down the axon of a neuron and impinge upon the synapses, sending signals of various strengths down the dendrites of other neurons. The strength of a given signal is determined by the *efficiency* of the synaptic transmission. A particular neuron will send an impulse down its axon if sufficient signals from other neurons impinge upon its dendrites in a short period of time, called *the period of latent summation*. The signal impinging upon a dendrite may be either *inhibitory* or *excitatory*. A neuron will fire, i.e., sends an impulse down its axon, if its excitation exceeds its inhibition by critical amount, the *thresholds* of the neuron.

Fig. 1.2 shows an artificial neuron, refereed to hereafter simply as neuron, which models the behavior of the biological neuron. The body of the *jth* neuron will often be represented by a weighted linear sum, z_j, of the input signal followed by a linear or non linear function, $y_j=f(z_j)$. The function $f(z_j)$ is called the activation function, a function which uses the input value to determine the output activity of the neuron. This neuron description is termed a McCulloch-Pitts neuron. One typical case using the binary threshold function

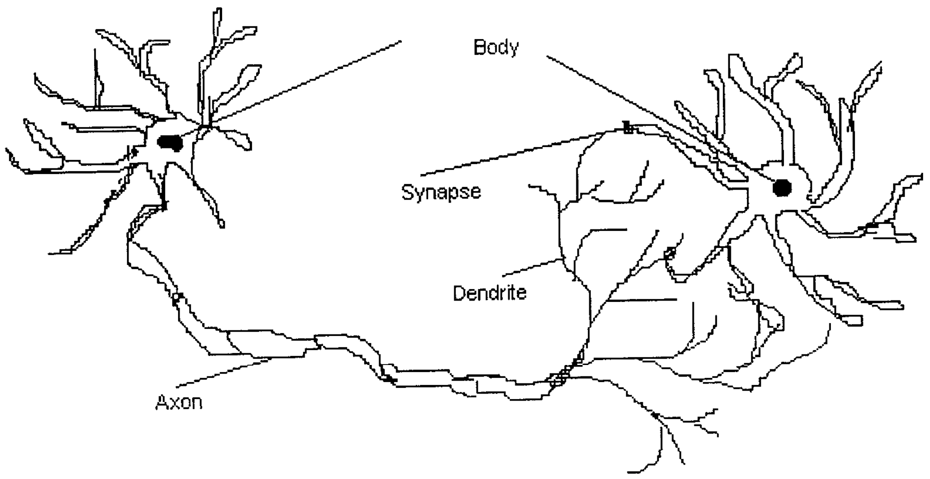

Fig. 1.1 Biological Neuron

$$y_j = f(z_j) = \begin{cases} ON & if\ z_j > threshold \\ OFF & if\ z_j < threshold \end{cases} \qquad (1.1)$$

McCulloch-Pitts type neuron using this binary threshold is refereed to in the literature as a perceptron. The perceptron is a basic element in may neural network architectures. Different neural networks use different *f(z$_j$)* functions, but the internal structure of the neuron, i.e., linear sum followed by a function *f(z$_j$)*, is common to most networks. Examples of functions, *f( )*, are linear function, non-linear sigmoidal functions, and threshold functions. The synaptic efficiencies will be represented by the interconnection weights, w_{ij}, from the ith neuron to the jth neuron. Resistors are often used for the interconnection weights in neural network hardware. The weights can be either positive (excitatory) or negative (inhibitory). Usually for a given network architecture the function *f ( )*, will be fixed, so varying the weights will allow the network to carry out different computation. The outputs of the neurons determine the current state of the network

Neural network will either have fixed weights or adaptable weights. Networks with adaptable weights use learning laws to adjust the values of the interconnection strengths. If the neural network is to use fixed weights, then the task to be accomplished must be will defined *a prior*. The weights will be determined explicitly from the description of the problem. Adaptable weights are essential if it is not known *a prior* what the correct weights should be. There are two types of learning: supervised and unsupervised[4]. *Supervised learning* occurs when the network is supplied with both the input values and the correct output values, and the network adjusts its weights based upon the error of the computed output. *Unsupervised learning* occurs when the network is only provided with the input values, and the network adjusts the interconnection strengths based solely on the input values and the current network output. *Learning laws* determine how the network is to adjust its weights using an error function or some other criteria. The learning law used is determined by the desired characteristics of the network.

Adaptable neural networks may have two phases to their operation. The first phase is the *training* of the network. The user provides the network with a "suitable" number of input patterns, and output patterns if required, and the network adjusts the interconnection strengths until the output is "close enough" to the correct output. There are some applications, using either supervised or unsupervised learning, where there will be no explicit training phase. The second phase is the *recall* or computation phase. The network is presented an input pattern and the network simply computes its output. When a network use unsupervised learning , it will sometimes continue to adjust the interconnection strengths during the recall phase. Usually, supervised networks will not do any learning during recall since that would require that the user know a priori the correct answer.

Neural network operates in either discreet time or continuos time depending upon the network architecture used[1]. Some networks use neurons which compute at synchronous and discreet intervals, others use asynchronous computations in continuous time. Typically, feed-forward networks use synchronous and discreet computations, while feed-back networks use asynchronous and continuos computations.

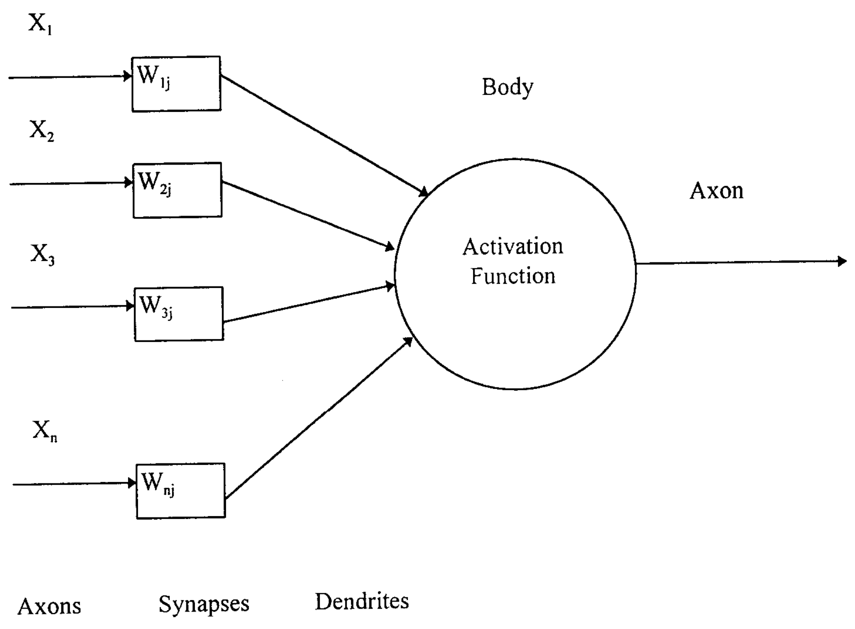

Fig. 1.2 The McCulloch-Pitts Neuron

1.1.3 Feed-Forward Neural Networks

Neural network architectures can be classified in two groups based upon their operation during the recall phase[1]. In *feed-forward* network the output of any given neuron can not be fed back to itself directly or indirectly through other neurons, and so its present output does not influence future outputs. In these networks computations are completed in single pass. When an input pattern is presented to the input terminals of a network the neurons in the first layer compute output values and pass these values onto the next layer. Each layer in turn receives input values from the previous layer, computes output values and passes these values onto the next layer. When the output values of the final layer are determined the computation ends. This rule can be broken only during the training phase or the learning portion of the recall phase, when the output of the neuron can be used to adjust its weights. Thus, influencing future outputs of the neuron. An example of this type of structure is the famous ADALINE[5,6,7]. Another popular feed-forward networks using feedback between layers during training are : Multiple Layers Perceptrons (MLPs) and Radial Basis Function (RBFs) [4]. Remember, that the operations during the computational portion of the recall phase are what determine the classification of the network. In a *feedback* network any given neuron is allowed to influence its own output directly through self feedback or indirectly through the influence it has, on other neurons from which it receives inputs.

1.1.4 Backpropagation

Technically, backpropagation is a specific learning law. The term is often used however to refer to a hierarchical network architecture that uses the backpropagation update algorithm for updating the weights of each layer based on the error presented at the network output. A backpropagation network always consists of at least three layers:an input layer, a hidden layer and an output layer. For sum application more than one hidden layer is used. The number of neurons in the input layer is equal to the number of inputs in each input pattern, and each of these neurons receives one of the inputs. The number of the neurons in the output layer are the output of the network. The number of neurons in the hidden layer is up to the network designer . Too many neurons will tend to make the network has difficulties dealing with new types of input patterns, e.g., making generalization . Too few neurons will not allow the network to make sufficient internal representations so that accurate maps from the input patterns to the output patterns can be made. Some researchers have proposed some general guidelines for choosing the number of hidden neurons, but no one has given strict rules for how many neurons to use.

In back propagation network there are no interconnections between neurons in the same layer. However, each neuron on the layer provides an input to each and every neuron on the next layer. That is, each input neuron will send its output to every neuron in the hidden layer. The input layer neurons will each have a single input and a single output, and will simply pass the value on their input to their output, without using the activation function. It's a valid remark that the input layer serves no functional purpose and could be omitted. Traditionally, the input layer has been included in the network discussed in the literature. Generally , the weights between the layers are initially small random values. This allows the network to train and function well. It's important to notice that the weights do not start with the same value, so that internal representations requiring non-symmetric weights can be obtained.

There are two distinct operations that take place during the training phase: the feed-forward computation and the updating of the weights based upon the error of the network output. The backpropagation network uses supervised learning so, the input/output patterns to be associated are both known. During recall phase only the feedforward computation, consisting of a single pass of the calculations, take place.

The feed-forward process involves presenting an input pattern to the input layer neurons which, passes the values onto the hidden layer. Each of the hidden layer neurons computes a weighted sum of its inputs, passes the sum through its activation function and presents the activation value to output layer. Each neuron in the output layer also computes a weighted sum of its inputs and passes the sum through the activation function , resulting in one of the output values for the network, although its not necessary. Some researchers are experimenting with different threshold and activation function for each of the different layers to get improved capabilities from the backpropagation network.

1.1.5 Feed-Forward Network Design- an Overview

Feed-forward network architecture must be designed probably for a particular data set to assure that the network will learn robustly and will be reasonably efficient[4]. The main questions in designing the architecture and then training the network are listed below.

1. How many layers of neurons should we use?
2. How many input nodes should we use?
3. How many neurons in the hidden layers should we use?
4. How many neurons should we use in the output layer?
5. What should the target(identifier) vector be?
6. How do we proceed to train the network?
7. How can we test to determine whether or not the network is probably trained?
8. How do we select parameters that speed up and improve the traning?
9. What should be the range of the weights and the network inputs and outputs?

Detailed answer for all these question is outside the scope of this overview but we introduce, briefly, answer for selected important question of them.

Hornik-Stinchomble-White provides the answer of question 1. He states that a hidden layer and an output layer of neurons are sufficient, provided that there are enough neurons in the hidden layer[8]. To reduce a large number of neurons in some situation, we may use two hidden layer but this is not necessary. There is also the question of what effects an extra layer of neurons in the middle can have. Is the training faster? Is the training better ? How does the answer of these questions depend upon the linearity or non-linearity of the data ?

Answer of question 2 can be given tentatively. The number N of input nodes must be the number N of features in the features vectors, so that , once a set of features is chosen, their number N is fixed. Answer of question 3 is difficult. A lot of research from different approaches has been done to find an answer, but a strict one has not been obtained. The number of neurons in the hidden layers in general is a problem and designer dependent.

Answer of question 7 which tells whether or not the training is satisfactory is that validation and verification testing of acceptance of the network training and the model(architecture) is needed. This involves using a training subset of the sample of example pairs and two other disjoint test subsets that are to be used for validation and verification but not for training. When there are sufficiently number of many examples, we may select 25% of them at random to save for validation, choose another 15% at random to serve as the final verification, and use the remanding 60% for training.

Answer of question 9, on the range of weights, is perhaps the easiest to answer for practical purposes. While the initial weights may be drawn from the interval [-0.5 , 0.5], the training may require that some weights out into the interval [-b,b] for some b>1. We want to avoid weight drift, however, where certain weights become large in magnitude and other weights compensate with opposite sign to cancel out its effects. Ideally the final weights should be in [-1 , 1] because the input and output do not exceed 1 in magnitude(assuming normalization is used which is the usual case) and the activation functions squash the summed values to within unit magnitude. In computational practice, though, the weights may be allowed to wander slightly. The range of the input features and output is typically taken to be [0 , 1] for unipolar activation functions and [-1 , 1] for bipolar ones.

1.2 SURVEY OF CLASSICAL CONTROL

Classical control theory can be divided into two main divisions: Linear control and nonlinear control. The classical controller (linear or non linear)can be designed to be

1. Optimal controller.
2. Robust controller
3. Adaptive controller.
4. A combination from the above three ones.

1.2.1 Linear Control

Linear control is concerned with the systems of the form

$$\dot{z} = Az + Bu \tag{1.2}$$

with state **z** and input **u** and measurable output

$$y = Cz \tag{1.3}$$

and controller of the form

$$u = -Kz + Mw \tag{1.4}$$

with **w** is the reference state, that is the state to which the plant to be brought with the help of the controller, **A**, **B**, **C** and **M** are matrices of appropriate dimensions. The goal of linear design is as follows:

- Stability of the closed-loop, that is, convergence back to an equilibrium point after being moved away from this point by disturbance.
- Optimality of closed-loop behavior in some user-defined sense.

For linear systems, local stability is identical with global stability, which is convergence to an equilibrium point from all points of the state space. A stability guided design is based on the analysis of closed-loop eigenvalues. The condition to be satisfied is that the real parts of all eigenvalues must be negative. Negative eigenvalues real parts are synonymous with the trajectory of the closed loop experiencing either damped oscillations or exponential overshoot-free convergence.

However with the stability condition satisfied, the controller is still under determined. There are two additional requirements:

1. A complete model of the closed-loop response (reference model)
2. Optimization

In using a reference model, the properties of a closed loop (consisting of the plant and feedback controller) can be satisfied in terms of eigenvalues. Because eigenvalues completely describe the behavior of a linear system, a path with arbitrary dynamic properties can be transformed, with the help of a feed back controller, into a closed loop with arbitrary different dynamic properties. This is true under some condition that, at least in principle, and with limited precision, are relatively frequently satisfied in practice. In theory, with appropriate sequence of control actions (dead_beat control), linear discrete systems with n state variables are guaranteed to be able to reach an arbitrary goal state with n steps. But the possibility of changing the dynamic properties of a closed loop can hardly be applied in practice.

Mansoura University
Faculty of Engineering
Computer & Systems Engineering Dept.

COMPUTER BASED MODEL FOR HUMAN OPERATOR AND ITS APPLICATIONS IN CONTROL SYSTEMS

thesis

Submitted in Partial Fulfillment of the Requirements for Degree of Master of Science

In

Automatic Control Engineering

by

Eng, Hamed Mohamed Hemeda

Under Supervision of

Dr.
Mohamed Sherief
Computer & Systems Engineering Dept.
Faculty of Eng. Mansoura University

Associate Prof. Dr
Mostafa M. Abd Elnaby
Electronics & Communication Dept.
Faculty of Eng. Mansoura University

Prof. Dr.

Y.M. Enab
Computer & Systems Engineering Dept.
Faculty of Eng. Mansoura University

1999

SUPERVISORS

Dissertation Tittle

COMPUTER BASED MODEL FOR HUMAN OPERATOR AND ITS APPLICATIONS IN CONTROL SYSTEMS

Researcher Name

Hamed Mohamed Hemeda

SUPERVISORS

Name	Position
Prof. Dr. Y.M. Enab [illegible]	Control & Computer Dept Faculty of Eng. Mansoura University
Associate Prof. Dr. M. Abd Elnaby [illegible]	Electronics & Communication Dept. Faculty of Eng. Tanta University
Dr. M. S. Mostafa [illegible]	Control & Computer Dept Faculty of Eng. Mansoura University

EXAMINATION COMMITTEE

Dissertation Tittle

COMPUTER BASED MODEL FOR HUMAN OPERATOR AND ITS APPLICATIONS IN CONTROL SYSTEMS

Researcher Name

Hamed Mohamed Hemeda

EXAMINATION COMMITTEE

Name	Position
Prof. Dr. Ebrahiem E. Zidan	Faculty of Eng. El-Zakaziek University
Prof. Dr. Fayez F. Gomaa	Computer & Systems Engineering Dept. Faculty of Eng. Mansoura University
Prof. Dr. Y.M. Enab	Computer & Systems Engineering Dept. Faculty of Eng. Mansoura University
Associate Prof. Dr M. Abd Elnaby	Electronic & Communication Dept Faculty of Eng. Tanta University

Examination Date : 6/9/ 1999

Signature

Name	Signature
Prof. Dr. Ebrahiem E. Zidan	[illegible]
Prof. Dr. Fayez F. Gomaa	[illegible]
Prof. Dr. Y.M. Enab	[illegible]
Associate Prof. Dr M. Abd Elnaby	[illegible]

ACKNOWLEDGMENT

Firstly, I wish to thank Prof. Dr. Y.M. Enab for his supervision and helpful advice. I would like to express my deep thanks to Associate Prof. Dr. M. Abd Elnaby and Dr. Mohamed Sherief Mostafa for their encouragement during the work. I like also to extend my sincere gratitude to all who helped me to complete this work by their guidance and support in times I needed help. Also, to all those assisted me in this thesis in Mansoura university or in Tanta university.

Summary

Neural networks have received attention in the research community in recent years. That networks have been shown to elegantly and powerfully realize solutions to problems in pattern recognition, associative memory, and database retrieval. Among neural networks application is : Neurocontrol, which is a dynamic research field that has attracted considerable attention from control engineering community in the last several years. The field searches the use of neural network in automatic controllers developing.

In this research the template learning neurocontrol approach is discussed as a new method for automatic controller design. In this method, a template controller, which is able to control the investigated system, is used to design a neurocontroller for that system. The template controller chosen, is the most intelligent controller known until now- *Human Operator(HO)*.The research showed that human operator factors can be utilized in building automatic controller for complex nonlinear systems. The method entails three phases in controller design process, they are:

Observation Phase: In this phase, the template controller is used to control the investigated system. The actions taken by the template together with the corresponding system states are registered in the form of input output examples.

Modeling Phase: In this phase, the registered data during the observation phase is used to train the *Human Operator Behavior Model(HOBM)* which is a neural network. The aim of this phase is the approximation of the HO role through the HOBM.

Testing Phase: In this phase the trained model (HOBM) is used to control the system and its performance is evaluated.

Based on the results of the testing phase the modeling phase may be revisited by doing more training, or even modifying the model . The process is repeated until the desired performance is obtained.

In our work, the method was applied on two selected benchmark control problems- ship steering problem and pole balancing problem through computer simulation. It's proved, through the simulation, that the resulting neurocontroller is able to control the system with accepted performance.

Using the HO as a template controller is one of the positive points of this method. This because, the HO is often expert in keeping the complex control system on the right track. Also, machine inheritance of human factors is one of the most important goals of the dynamic research field: *Artificial Intelligence,* this work can be considered as a step on the way. Another positive point for that method is that, there is no need for process mathematical model and its suitability for nonlinear systems.

The neural networks structures used in this work to implement the HOBM are feed forward networks and they are as follows:

- Radial Basis Function neural network (RBF): with the basis function segma parameter is a vector of n dimension, where n is the number of inputs to the network

- Radial Basis Function neural network (RBF): with the basis function segma parameter is a scalar.
- Multiple layers perceptors(MLP) feed forward neural network single hidden layer: with bipolar sigmoidal function as the activation function.
- Multiple layers Perceptrons(MLP) feed forward neural network two hidden layers:with bipolar sigmoidal function as the activation function.

Comparing the responses of the HO and the HOBM for both ship steering and pole balancing problems we found that:

- Both the neural controller and the HO are able to steer the ship and balance the pole(for limited time interval).
- The control signal provided by the neural controller is smoother than that provided by the HO.
- The best approximation for ship steering problem was achieved using the RBF vector-segma feed-forward network with single hidden layer, this structure seemed to be the best, among the discussed four structures, for that problem.
- The best approximation for pole balancing problem was achieved using the MLP feed-forward with single hidden layer, this structure seemed to be the best, among the discussed four structures, for that problem.

Table of Contents

CHAPTER 1

INTRODUCTION

Neural networks have received great attention in the research community in recent years. Then networks have been shown to elegantly and powerfully realize solutions to problems in pattern recognition, associative memory, database retrieval, and automatic control. In this chapter, we first introduce an overview of neural networks, then a survey of classical control will be introduced . The aim of this chapter is not to provide complete but enough information to be the basis of the following chapters

1.1 NEURAL NETWORK OVERVIEW

The term neural network is used to describe various topologies of highly interconnection of simple processing elements that offer an alternative to traditional approaches to computing. The Tactical Technology Office of the U.S. Defense Advanced Research Projects Agency(DARPA) recently completed a study of neural networks[1]. DPRPA was interested in determining the current status and potential usefulness of neural networks, and assessing the worth of investing more time, energy and money. The DARPA study reached the following conclusions[2].

- Neural networks offer important new approaches to information processing because of their adaptability to learn as well as their massive parallelism.
- Neural networks research has matured greatly since the perceptron of the 1950's. This maturation has three resources:(a) advancement in mathematical theories, (b)

development of new computer tools, and (c) increased understanding of neurobiology. There is a bridge forming between the biologically—oriented neural modelers and the artificial neural network modelers, and substantial progress will result from this alliance.

- Although the computer revolution of the last 20 years has played a critical role in the revival of neural network research by making it possible to undertake much work in mathematical theory. The limits of today's computing devices- inadequacies in storage, speed, and user flexibility- have restricted present neural network efforts.
- There have been significant demonstrations of neural network capabilities in vision, speech, signal processing, and robotics. Perhaps not to the scale that might be desired-due to a lack of funding of research-but the variety of problems addressed by neural networks is impressive.
- Hardware capabilities are limiting the development of important neural network applications. Clearly, if researchers are provided with improved simulation and implementation capabilities, the field of neural network will once again drift off into the wilderness.

1.1.1 Neural Networks vs. Traditional Computers

Robert Hecht_Nielson, of the Hecht_Nielson Neurocomputer corporation, defines a neural network as "a dynamical system with a topology of directed flow graph that can carry out information processing by means of its state response to continuos or episodic inputs "[3]. A neural network is fundamentally different from a traditional computer. Neural networks

- do not sequentially execute instructions nor do they contain memory for storing operating structure or data
- respond in parallel to a set of inputs
- are more concerned with transformations than algorithms and procedures
- do not contain only one or a few complicated computational devices, but are composed of a large number of simple devices oven doing little more than computing weighted sums.

Information in neural network is not stored in explicit memory locations, but is represented by the interconnection strengths and the current output of the neurons. Neural networks do not replaceme traditional computers, but are an entirely different class of computational devices capable of qualitatively different tasks. Neural networks provide a completely new unique way to look at information processing. To fully appreciate and understand neural networks they must not be cast into the traditional view of computing machines. They are vastly different, and must be viewed as such.

Also, while many neural networks are currently being simulated on computers, they should not be considered a new programming technique. Neural network applications are often explored via computer simulations, but these simulations are only tools for numerical analysis of the network. Given the complexity of neural network architectures, computer simulations are inefficient ways to implement neural network architectures. Neural networks hardware will have to be developed before neural networks are effectively implemented.

1.1.2 Basic Elements and Terminology

Fig. 1.1 shows the biological neuron model upon which neural networks are based[1]. Each neuron in the brain is composed of a *body*, one *axon*, and multitude of *dendrites*. The dendrites form a very fine "filamentary brush" surrounding the body of the neuron. The axons can be thought of a long thin tube that splits into branches terminating in little *endbulbs* almost touching the dendrites of other cells. The small gap between an endbulb and dendrite is called a *synapse*. The axon of a single neuron forms synaptic connections with many other neurons. Fig. 1.1 shows the body and the axon of one neuron interacting with the dendrites of a second neuron. Impulses propagate down the axon of a neuron and impinge upon the synapses, sending signals of various strengths down the dendrites of other neurons. The strength of a given signal is determined by the *efficiency* of the synaptic transmission. A particular neuron will send an impulse down its axon if sufficient signals from other neurons impinge upon its dendrites in a short period of time, called *the period of latent summation*. The signal impinging upon a dendrite may be either *inhibitory* or *excitatory*. A neuron will fire, i.e., sends an impulse down its axon, if its excitation exceeds its inhibition by critical amount, the *thresholds* of the neuron.

Fig. 1.2 shows an artificial neuron, refereed to hereafter simply as neuron, which models the behavior of the biological neuron. The body of the *jth* neuron will often be represented by a weighted linear sum, z_j, of the input signal followed by a linear or non linear function, $y_j=f(z_j)$. The function $f(z_j)$ is called the activation function, a function which uses the input value to determine the output activity of the neuron. This neuron description is termed a McCulloch-Pitts neuron. One typical case using the binary threshold function

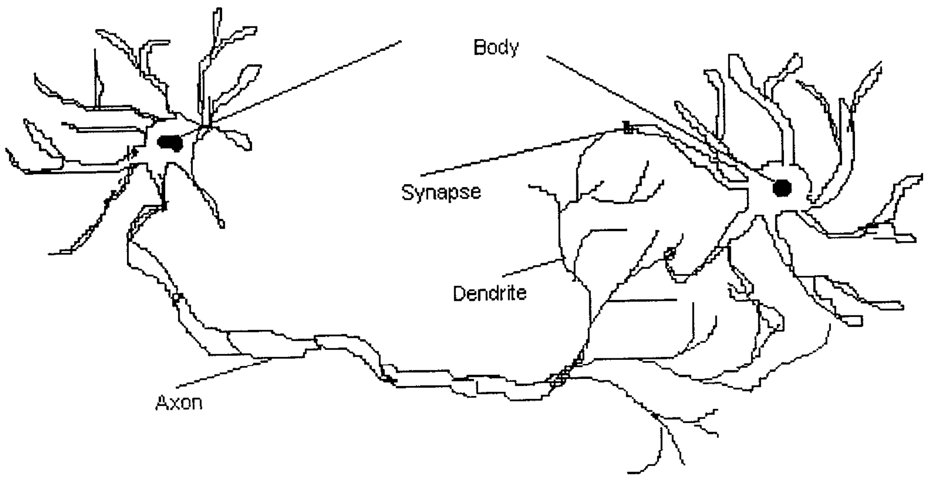

Fig. 1.1 Biological Neuron

$$y_j = f(z_j) = \begin{cases} ON & if\ z_j > threshold \\ OFF & if\ z_j < threshold \end{cases} \qquad (1.1)$$

McCulloch-Pitts type neuron using this binary threshold is refereed to in the literature as a perceptron. The perceptron is a basic element in may neural network architectures. Different neural networks use different *f(z$_j$)* functions, but the internal structure of the neuron, i.e., linear sum followed by a function *f(z$_j$)*, is common to most networks. Examples of functions, *f( )*, are linear function, non-linear sigmoidal functions, and threshold functions. The synaptic efficiencies will be represented by the interconnection weights, w_{ij}, from the ith neuron to the jth neuron. Resistors are often used for the interconnection weights in neural network hardware. The weights can be either positive (excitatory) or negative (inhibitory). Usually for a given network architecture the function *f ( )*, will be fixed, so varying the weights will allow the network to carry out different computation. The outputs of the neurons determine the current state of the network

Neural network will either have fixed weights or adaptable weights. Networks with adaptable weights use learning laws to adjust the values of the interconnection strengths. If the neural network is to use fixed weights, then the task to be accomplished must be will defined *a prior*. The weights will be determined explicitly from the description of the problem. Adaptable weights are essential if it is not known *a prior* what the correct weights should be. There are two types of learning: supervised and unsupervised[4]. *Supervised learning* occurs when the network is supplied with both the input values and the correct output values, and the network adjusts its weights based upon the error of the computed output. *Unsupervised learning* occurs when the network is only provided with the input values, and the network adjusts the interconnection strengths based solely on the input values and the current network output. *Learning laws* determine how the network is to adjust its weights using an error function or some other criteria. The learning law used is determined by the desired characteristics of the network.

Adaptable neural networks may have two phases to their operation. The first phase is the *training* of the network. The user provides the network with a "suitable" number of input patterns, and output patterns if required, and the network adjusts the interconnection strengths until the output is "close enough" to the correct output. There are some applications, using either supervised or unsupervised learning, where there will be no explicit training phase. The second phase is the *recall* or computation phase. The network is presented an input pattern and the network simply computes its output. When a network use unsupervised learning , it will sometimes continue to adjust the interconnection strengths during the recall phase. Usually, supervised networks will not do any learning during recall since that would require that the user know a priori the correct answer.

Neural network operates in either discreet time or continuos time depending upon the network architecture used[1]. Some networks use neurons which compute at synchronous and discreet intervals, others use asynchronous computations in continuous time. Typically, feed-forward networks use synchronous and discreet computations, while feed-back networks use asynchronous and continuos computations.

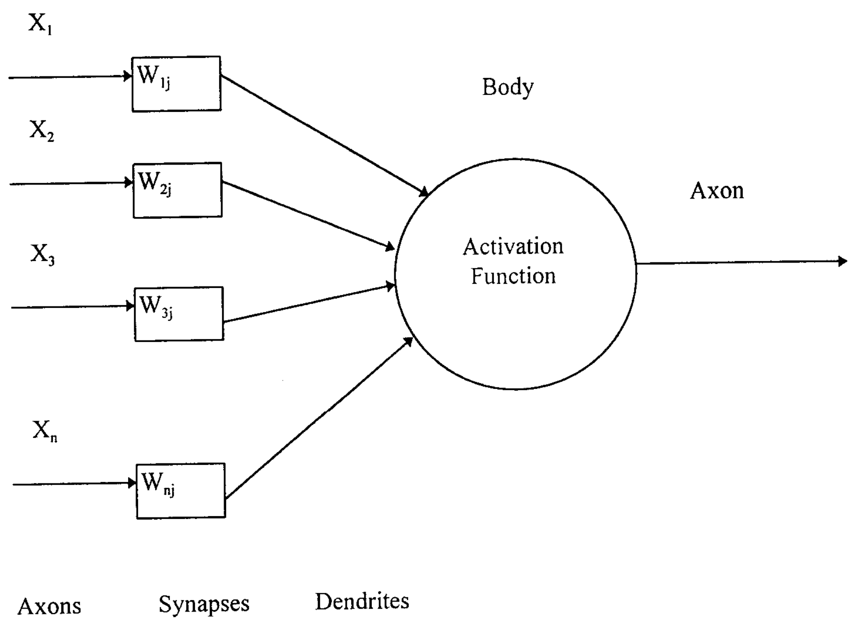

Fig. 1.2 The McCulloch-Pitts Neuron

1.1.3 Feed-Forward Neural Networks

Neural network architectures can be classified in two groups based upon their operation during the recall phase[1]. In *feed-forward* network the output of any given neuron can not be fed back to itself directly or indirectly through other neurons, and so its present output does not influence future outputs. In these networks computations are completed in single pass. When an input pattern is presented to the input terminals of a network the neurons in the first layer compute output values and pass these values onto the next layer. Each layer in turn receives input values from the previous layer, computes output values and passes these values onto the next layer. When the output values of the final layer are determined the computation ends. This rule can be broken only during the training phase or the learning portion of the recall phase, when the output of the neuron can be used to adjust its weights. Thus, influencing future outputs of the neuron. An example of this type of structure is the famous ADALINE[5,6,7]. Another popular feed-forward networks using feedback between layers during training are : Multiple Layers Perceptrons (MLPs) and Radial Basis Function (RBFs) [4]. Remember, that the operations during the computational portion of the recall phase are what determine the classification of the network. In a *feedback* network any given neuron is allowed to influence its own output directly through self feedback or indirectly through the influence it has, on other neurons from which it receives inputs.

1.1.4 Backpropagation

Technically, backpropagation is a specific learning law. The term is often used however to refer to a hierarchical network architecture that uses the backpropagation update algorithm for updating the weights of each layer based on the error presented at the network output. A backpropagation network always consists of at least three layers:an input layer, a hidden layer and an output layer. For sum application more than one hidden layer is used. The number of neurons in the input layer is equal to the number of inputs in each input pattern, and each of these neurons receives one of the inputs. The number of the neurons in the output layer are the output of the network. The number of neurons in the hidden layer is up to the network designer . Too many neurons will tend to make the network has difficulties dealing with new types of input patterns, e.g., making generalization . Too few neurons will not allow the network to make sufficient internal representations so that accurate maps from the input patterns to the output patterns can be made. Some researchers have proposed some general guidelines for choosing the number of hidden neurons, but no one has given strict rules for how many neurons to use.

In back propagation network there are no interconnections between neurons in the same layer. However, each neuron on the layer provides an input to each and every neuron on the next layer. That is, each input neuron will send its output to every neuron in the hidden layer. The input layer neurons will each have a single input and a single output, and will simply pass the value on their input to their output, without using the activation function. It's a valid remark that the input layer serves no functional purpose and could be omitted. Traditionally, the input layer has been included in the network discussed in the literature. Generally , the weights between the layers are initially small random values. This allows the network to train and function well. It's important to notice that the weights do not start with the same value, so that internal representations requiring non-symmetric weights can be obtained.

There are two distinct operations that take place during the training phase: the feed-forward computation and the updating of the weights based upon the error of the network output. The backpropagation network uses supervised learning so, the input/output patterns to be associated are both known. During recall phase only the feedforward computation, consisting of a single pass of the calculations, take place.

The feed-forward process involves presenting an input pattern to the input layer neurons which, passes the values onto the hidden layer. Each of the hidden layer neurons computes a weighted sum of its inputs, passes the sum through its activation function and presents the activation value to output layer. Each neuron in the output layer also computes a weighted sum of its inputs and passes the sum through the activation function , resulting in one of the output values for the network, although its not necessary. Some researchers are experimenting with different threshold and activation function for each of the different layers to get improved capabilities from the backpropagation network.

1.1.5 Feed-Forward Network Design- an Overview

Feed-forward network architecture must be designed probably for a particular data set to assure that the network will learn robustly and will be reasonably efficient[4]. The main questions in designing the architecture and then training the network are listed below.

1. How many layers of neurons should we use?
2. How many input nodes should we use?
3. How many neurons in the hidden layers should we use?
4. How many neurons should we use in the output layer?
5. What should the target(identifier) vector be?
6. How do we proceed to train the network?
7. How can we test to determine whether or not the network is probably trained?
8. How do we select parameters that speed up and improve the traning?
9. What should be the range of the weights and the network inputs and outputs?

Detailed answer for all these question is outside the scope of this overview but we introduce, briefly, answer for selected important question of them.

Hornik-Stinchomble-White provides the answer of question 1. He states that a hidden layer and an output layer of neurons are sufficient, provided that there are enough neurons in the hidden layer[8]. To reduce a large number of neurons in some situation, we may use two hidden layer but this is not necessary. There is also the question of what effects an extra layer of neurons in the middle can have. Is the training faster? Is the training better ? How does the answer of these questions depend upon the linearity or non-linearity of the data ?

Answer of question 2 can be given tentatively. The number N of input nodes must be the number N of features in the features vectors, so that , once a set of features is chosen, their number N is fixed. Answer of question 3 is difficult. A lot of research from different approaches has been done to find an answer, but a strict one has not been obtained. The number of neurons in the hidden layers in general is a problem and designer dependent.

Answer of question 7 which tells whether or not the training is satisfactory is that validation and verification testing of acceptance of the network training and the model(architecture) is needed. This involves using a training subset of the sample of example pairs and two other disjoint test subsets that are to be used for validation and verification but not for training. When there are sufficiently number of many examples, we may select 25% of them at random to save for validation, choose another 15% at random to serve as the final verification, and use the remanding 60% for training.

Answer of question 9, on the range of weights, is perhaps the easiest to answer for practical purposes. While the initial weights may be drawn from the interval [-0.5 , 0.5], the training may require that some weights out into the interval [-b,b] for some b>1. We want to avoid weight drift, however, where certain weights become large in magnitude and other weights compensate with opposite sign to cancel out its effects. Ideally the final weights should be in [-1 , 1] because the input and output do not exceed 1 in magnitude(assuming normalization is used which is the usual case) and the activation functions squash the summed values to within unit magnitude. In computational practice, though, the weights may be allowed to wander slightly. The range of the input features and output is typically taken to be [0 , 1] for unipolar activation functions and [-1 , 1] for bipolar ones.

1.2 SURVEY OF CLASSICAL CONTROL

Classical control theory can be divided into two main divisions: Linear control and nonlinear control. The classical controller (linear or non linear)can be designed to be

1. Optimal controller.
2. Robust controller
3. Adaptive controller.
4. A combination from the above three ones.

1.2.1 Linear Control

Linear control is concerned with the systems of the form

$$\dot{z} = Az + Bu \tag{1.2}$$

with state **z** and input **u** and measurable output

$$y = Cz \tag{1.3}$$

and controller of the form

$$u = -Kz + Mw \tag{1.4}$$

with **w** is the reference state, that is the state to which the plant to be brought with the help of the controller, **A**, **B**, **C** and **M** are matrices of appropriate dimensions. The goal of linear design is as follows:

- Stability of the closed-loop, that is, convergence back to an equilibrium point after being moved away from this point by disturbance.
- Optimality of closed-loop behavior in some user-defined sense.

For linear systems, local stability is identical with global stability, which is convergence to an equilibrium point from all points of the state space. A stability guided design is based on the analysis of closed-loop eigenvalues. The condition to be satisfied is that the real parts of all eigenvalues must be negative. Negative eigenvalues real parts are synonymous with the trajectory of the closed loop experiencing either damped oscillations or exponential overshoot-free convergence.

However with the stability condition satisfied, the controller is still under determined. There are two additional requirements:

1. A complete model of the closed-loop response (reference model)
2. Optimization

In using a reference model, the properties of a closed loop (consisting of the plant and feedback controller) can be satisfied in terms of eigenvalues. Because eigenvalues completely describe the behavior of a linear system, a path with arbitrary dynamic properties can be transformed, with the help of a feed back controller, into a closed loop with arbitrary different dynamic properties. This is true under some condition that, at least in principle, and with limited precision, are relatively frequently satisfied in practice. In theory, with appropriate sequence of control actions (dead_beat control), linear discrete systems with n state variables are guaranteed to be able to reach an arbitrary goal state with n steps. But the possibility of changing the dynamic properties of a closed loop can hardly be applied in practice.