Neural Networks in π^0 Discrimination

M. Sc Thesis
in
Engineering Physics
University of Gaziantep

Supervisor
Assist. Prof. Dr. Ayda BEDDALL

By
Yılmaz DURMAZ
January 2006

Approval of the Graduate School of Natural and Applied Sciences.

Prof. Dr. Sadettin ÖZYAZICI
Director

I certify that this thesis satisfies all the requirements as a thesis for the degree of Master of Science.

Prof. Dr. Zihni ÖZTÜRK
Head of Department

This is to certify that we have read this thesis and that in our opinion it is fully adequate, in scope and quality, as a thesis for the degree of Master of Science.

Assist. Prof. Dr. Ayda BEDDALL
Major Supervisor

Examining Committee Members:

Prof. Dr. Asker Ali ABİYEV _____

Assist. Prof. Dr. Humbat AHMEDOV _____

Assist. Prof. Dr. Andrew BEDDALL _____

Assist. Prof. Dr. Ayda BEDDALL _____

Assist. Prof. Dr. Mustafa YILMAZ _____

Abstract

Neural Networks in π^0 Discrimination

DURMAZ, Yılmaz
M. Sc in Engineering Physics
Supervisor: Assist. Prof. Dr. Ayda BEDDALL
January 2006, 46 pages

In this study the basics of artificial neural networks are investigated and a neural network is employed in the discrimination of signal from background for pion candidates from the ALEPH experiment. The performance of the neural network is found to compare well with that of a standard and a PDE method.

Key words: Artificial Neural Networks, ANN, NN, neuron, signal discrimination, neutral pion.

Özet

π^0 Seçilmesinde Yapay Sinir Ağlarının Kullanılması

DURMAZ, Yılmaz
Yüksek Lisans Tezi, Fizik Mühendisliği
Tez Yöneticisi: Y. Doç. Dr. Ayda BEDDALL
Ocak 2006, 46 sayfa

Bu çalışmada, yapay sinir ağlarının temel özellikleri araştırılarak ALEPH deneyinden alınan yüksek enerjili parçacık verileri içerisindeki π^0 sinyallerinin, dedektör ve diğer parçacıklardan kaynaklanan arka plandaki gürültü sinyalleri arasından seçilmesinde kullanılmıştır. Yapay sinir ağları, standart metod ve PDE metodu ile karşılaştırılmış ve başarılı olduğu gözlenmiştir.

Anahtar kelimeler: Yapay sinir ağları, nöron, sinyal bulma, yüksüz pion.

Acknowledgements

I would like to thank Assist. Prof. Dr. Ayda BEDDALL and Assist. Prof. Dr. Andrew BEDDALL for their supervisory and support, Ahmet BİNGÜL and Zekeriya UYSAL for their help for programming and writing.

My special thanks for my family who gave their personal support during my study.

Table of Contents

Chapter

List of Figures

Figure

Chapter 1

Introduction

In High Energy collisions of sub-atomic particles results in events containing both charged and uncharged particles. An example reconstructed event from ALEPH detector is shown in Figure 1.1

Charged particles leave tracks in detectors and this makes their identification process easy in general. Their charge, momentum and energies can be calculated by examining their tracks.

But uncharged particles can only be traced by their energies collected at calorimeters, with a poor energy resolution. Or they can only be traced by their daughter particles such as $\omega \rightarrow \pi^+\pi^-\pi^0$.

The neutral pion will mainly decay into two photons, $\pi^0 \rightarrow \gamma\gamma$, and is identified from invariant mass spectra. The invariant mass spectra shows the neutral pion signal as a peak on a combinatorial background, Figure 1.2. To improve signal purity some combinatorial background can be removed by selecting the signal from a mass window as indicated in the figure.

Further improvements can be gained by employing multi-variate discrimination techniques. In High Energy Physics a number of techniques are presented in literature, commonly they include artificial neural networks and probability density estimators.

Artificial Neural Networks were used in HEP analyses in many areas, with a suspicion because NN is considered as a blackbox. However, NN can compete with present techniques in the area.

This study focussed on the optimization of π^0 signals from the ALEPH detector.

In Chapter 2, history and a brief introduction to Artificial Neural Networks is given. Chapter 3 deals with some important properties of NN. A preliminary study is conducted in Chapter 4. In Chapter 5, a neural network is employed in the discrimination of signal and background for ALEPH data. Finally the summary and conclusion of the thesis is given in Chapter 6.

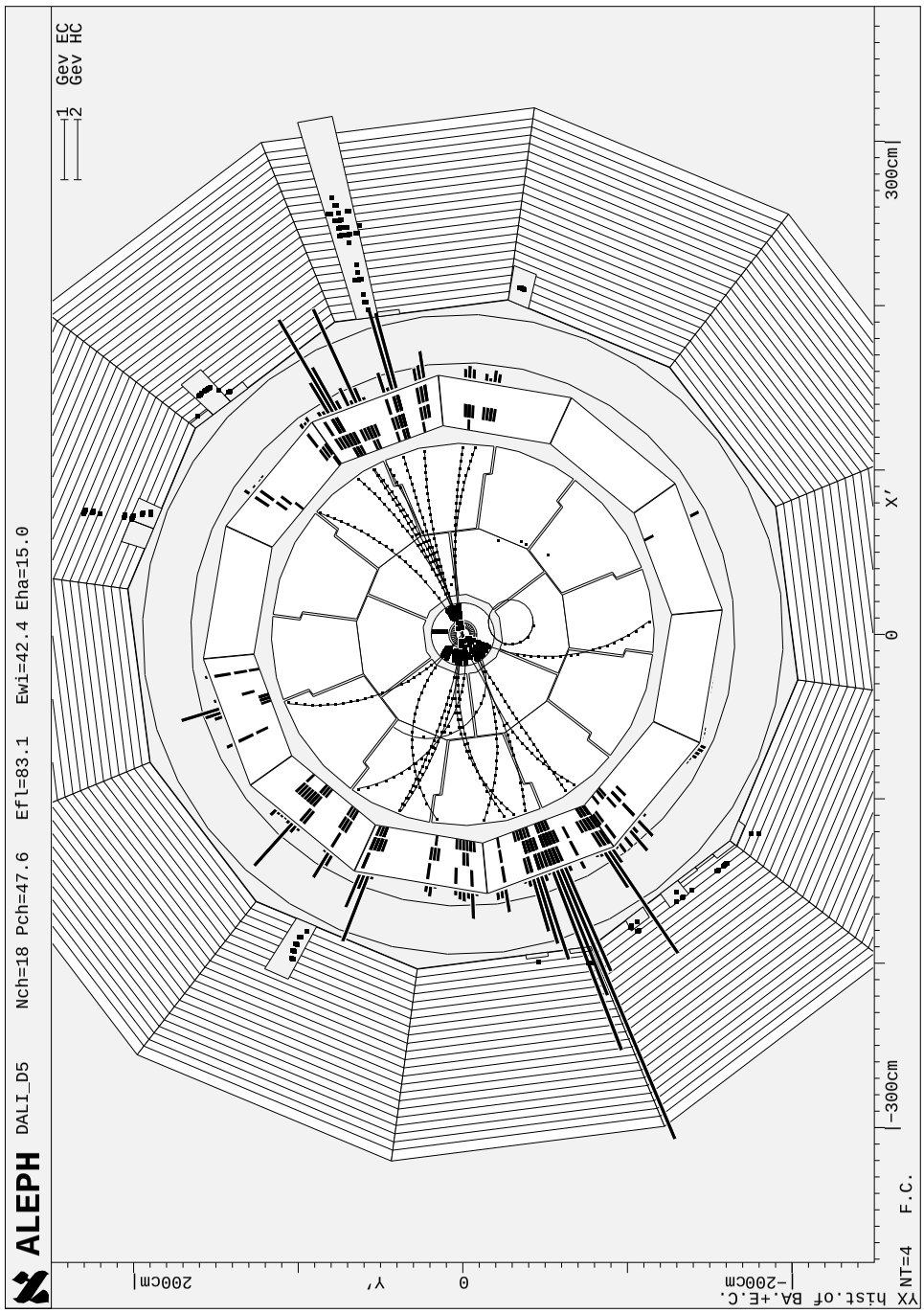


Figure 1.1: Reconstruction of a typical hadronic event in the ALEPH detector.

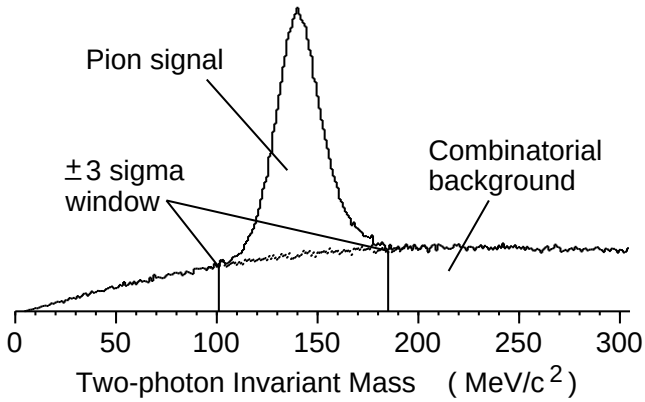


Figure 1.2: Two-photon invariant mass spectra.

Chapter 2

Introduction to Artificial Neural Networks

2.1 Introduction

Interest in Artificial Neural Networks (shortly ANN or NN) began in the early 40s after first researches on the working principles of neural structure of brain. Development of ANN was conducted to try both to understand the brain and to emulate some of its strengths.

Early studies on NN had great interest in both computer and electronics areas. Many theories were developed.

Up to 1969, studies continued on NN. Then, since limitations on computer based works, researchers lost their interest on this field. Development of NN almost stopped. Only a few researchers continued their studies. In the early 80s, with new achievements made by researchers who remained active during this lean time and developments on computer technology, interests on NN began to rise again in many areas, ranging from economics to high energy physics.

In high energy physics, NN was active in the late 80s and early 90s, and applied on areas such as signal-background separation, mass reconstruction, detector level triggering. But computer resources such as memory, speed for simulations of NN was not enough, researches on this area was slow. And again with the development of high performance systems, NN became popular on both software and hardware levels.

Following chapters includes a general introduction to NN with its basic properties. More detailed information and programming techniques can be found in [1], [2] and [3].

2.2 History of ANN

A summary of major researches on ANN is as follows:

- McCulloch and Pitts, 1943

  It is said that the modern age of neural networks began with the work of Warren S. McCulloch and Walter Pitts in 1943. They presented five assumptions governing the operation of neurons. These describe what is known as the McCulloch&Pitts neuron. There is no training of these neurons; however, they can act as certain logic functions. The McCulloch&Pitts neuron model laid the foundation for future developments in neural networks.

- Hebb, 1949

  In the paper by Donald Hebb in 1946, he describes a *learning process* that was postulated from a neurobiological viewpoint. Hebb stated that information is stored in the connections of neurons and postulated a learning strategy for adjustment of the connection weights. This was the first time a learning rule was presented that allowed for the adjustment of the synaptic weights, which has had a major impact on later work on this field.

- Rosenblatt, 1958

  In 1958, the original concept of the *perceptron* was developed by Frank Rosenblatt. It was the first precisely defined, computationally oriented neural network that attracted much intention by engineers because of its complex adaptive behavior. It was a machine that could be trained to classify certain patterns.

- Widrow and Hoff, 1960

  The Adaline (*Ada*ptive *Lin*ear *e*lement) of Bernard Widrow, trained by the least-mean-square (LMS) learning rule, closely resembles Rosenblatt's perceptron. The Adaline was extended to many Adalines (Madaline). There were many applications of the Adaline and Madaline, for example, in adaptive control and pattern recognition.

- Minsky and Papert, 1969

  Marvin Minsky and Seymour Papert slowed down neural network research in 1969 when they pointed out in their book, *Perceptrons*, that single-layer neural networks have limited capabilities. Rosenblatt had

studied architectures with more than one layer of neurons and believed they could overcome the limitations of the simple perceptron; however, there was no learning rule known at the time. Minsky and Papert doubted that one could be found.

- Werbos, 1974

  The first description of the *back-propagation algorithm* for training multi-layer feed-forward perceptrons was given by Werbos in 1974.

- Hopfield, 1982

  It is said that the *modern age* of neural networks began with the publication of the paper by John Hopfield (Nobel laureate in physics). Hopfield presented a sophisticated and comprehensive description of the working of a *recurrent* neural network and what it could actually do. The network can store information in a dynamically stable environment and is able to perform the function of data storage and retrieval. Given a noisy input to the network, the associated pattern stored in memory can be properly retrieved in spite of incomplete (corrupted) version that was presented to the network.

- Kohonen, 1982

  Teuvo Kohonen presented the self-organizing feature map in 1982. it is an unsupervised, competitive learning, clustering network in which only one neuron (or only one neuron in a group) is 'on' at a time.

Besides these main historical developments, there were many researches on this area i.e. learning methods to train multi-layered networks.

2.3 Biological Neurons

The fundamental processing element of a neural network is a neuron. We are born with about 100 billion neurons. A neuron may connect with up to 200,000 other neurons. This building block of human awareness encompasses a few general capabilities. Neurons provide us with the abilities to remember, think, and apply previous experiences to our action.

All natural neurons have four basic components: dendrites, soma, axon, and synapses. These can be seen in Figure 2.1. Basically, a biological neuron receives inputs from other neurons by dendrites, combines them in some way in its body, performs a generally nonlinear operation on the result, and then outputs the final result to its axon which distributes signal to other neurons.

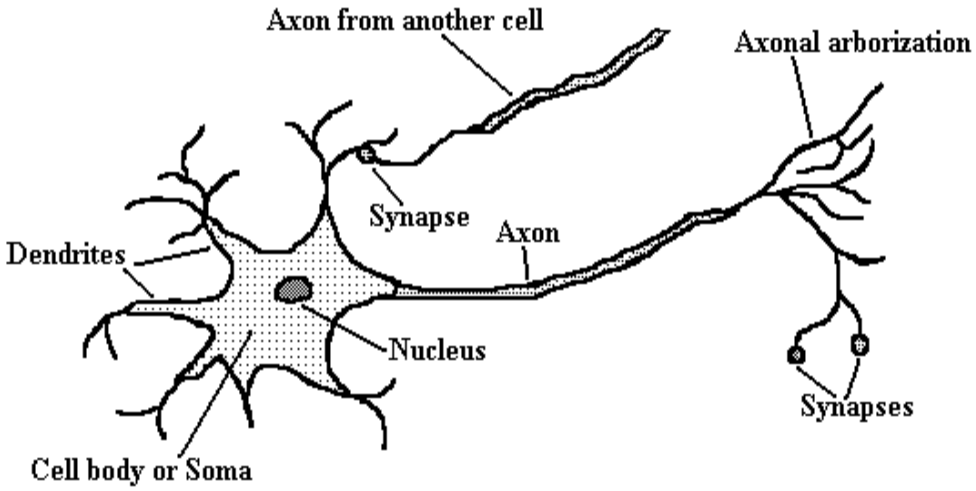
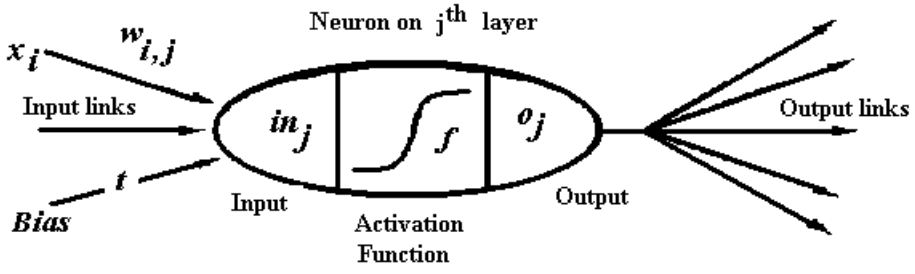


Figure 2.1: A biological neuron

2.4 Artificial Neurons

ANNs have been developed as generalizations of mathematical model of human cognition or neural biology, based on assumptions that:

- Information processing occurs at many simple elements called neurons.
- Signals are passed between neurons over connection links.
- Each connection link has an associated weight, which, in a typical neural net, multiplies the signal transmitted.
- Each neuron applies an activation function (usually nonlinear) to its net input (sum of the weighted signals) to determine its output signal.



$$\text{input to } j^{th} \text{ neuron} : \quad in_j = \sum w_{i,j} x_i + t$$

$$\text{output of } j^{th} \text{ neuron} : \quad o_j = f(in_j)$$

Figure 2.2: An artificial neuron

This can be seen in Figure 2.2. A neuron accepts inputs x_i, each associated with a weight $w_{i,j}$. Input to neuron is calculated as $in_j = \sum w_{i,j}x_i$, and processed with an activation function f, which is generally taken as a Sigmoid type. Output o_j is then distributed by links to other neurons.

2.5 Mathematical Methods vs. NN

There are some simple relations between mathematical methods and NN. It can be shown by some binary logics AND and OR.

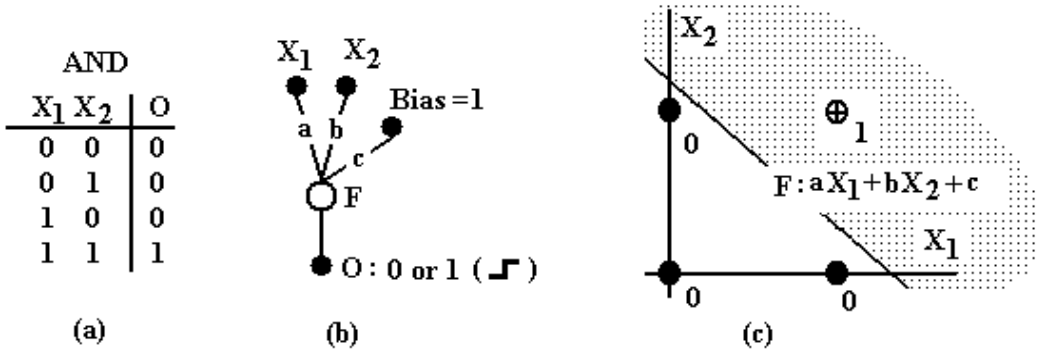


Figure 2.3: The AND function

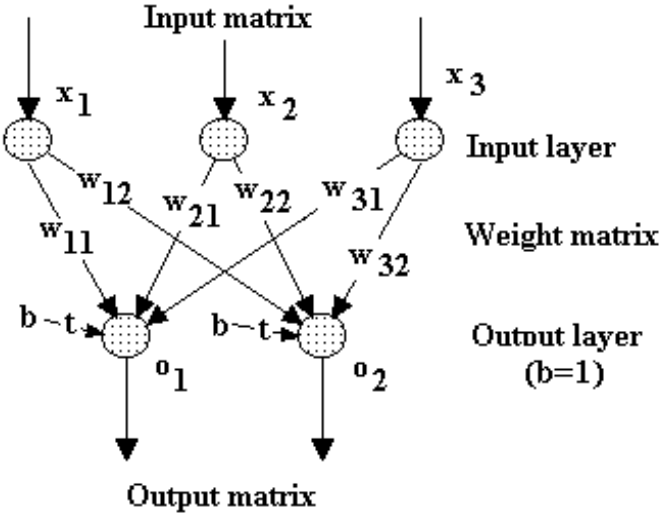


Figure 2.4: A single layer NN

2.5.1 AND Operation and Single Layer NN

AND logical operation is represented by two input variables and one output variable. Truth table and graphic of this function is as shown in Figure 2.3. From (c), it is shown that a line having equation $F = ax_1 + bx_2 + c = 0$ perfectly separates 0 and 1 outputs. It is also represented as in (b), which is much like a single layer NN with two inputs and one output as shown in Figure 2.4. Corresponding variables are as follows: $a \rightarrow w_{11}$, $b \rightarrow w_{21}$, $c \rightarrow w_{31}$, $bias \rightarrow x_3$ (bias always equal to 1) and finally $F(stepfunction) \rightarrow o_1$

In this single layer NN, calculation is $o_j = f(in_j)$; $in_j = \sum w_{ij}x_i + bt$, where $b = 1$, and f is a step or Sigmoid function.

This is the simple approximation of single-layered NN from a mathematical method. A more complex example describes why a multi-layered NN is needed.

2.5.2 XOR Operation and Multi-Layer NN

XOR logical operation is represented by two inputs and one outputs as in AND operation. Truth table and graphic of this function is as shown in Figure 2.5.

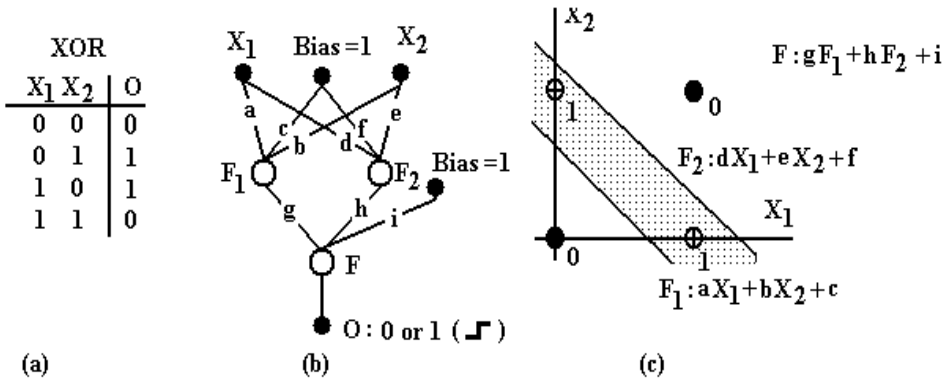


Figure 2.5: The XOR function

As seen a single function is not enough to separate output of XOR. At least two functions are needed which are subsequently fed into another function. From (c), it is shown that two lines having equation $F_1 = ax_1+bx_2+c = 0$ and $F_2 = dx_1+ex_2+f = 0$ which are fed into output function by $F = gF_1+hF_2+i$. F perfectly separates 0 and 1 outputs. It is also represented as in (b), which is much like a multi-layered NN with two inputs, one output and two hidden units as shown in Figure 2.6. Corresponding variables are as follows: $x_1 \rightarrow x_{11}$, $x_2 \rightarrow x_{12}$, $bias \rightarrow x_{13} = 1$, $a \rightarrow w_{111}$, $b \rightarrow w_{121}$, $c \rightarrow w_{131}$, $d \rightarrow w_{112}$, $e \rightarrow w_{122}$, $f \rightarrow w_{132}$, $F_1(step\ function) \rightarrow x_{21}$, $F_2(step\ function) \rightarrow x_{22}$, $bias \rightarrow x_{23}1$,

$g \rightarrow w_{211}$, $h \rightarrow w_{221}$, $f \rightarrow w_{231}$ and finally $F(step\ function) \rightarrow o_1$.

This second example introduces the hidden layer concept which is used in more complex problems to train a NN more accurately. Here $o_j^h = f(\sum(w_{ij}x_i)+t)$ and $o_k^{out} = f(\sum(w_{jk}o_j^h + t))$.

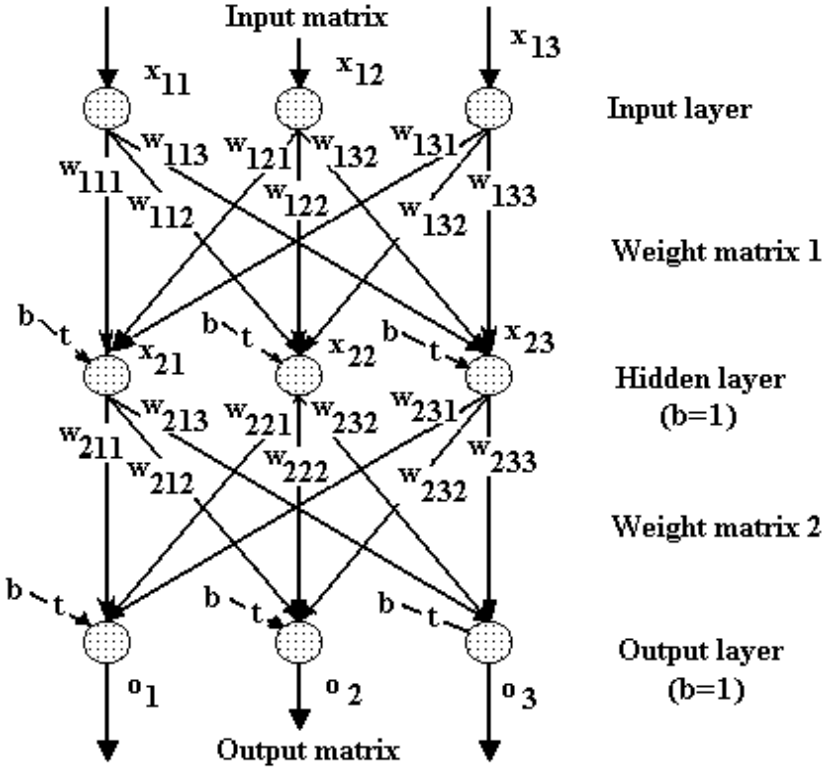


Figure 2.6: A multi-layered NN

This is the simple steps to move in to the field of NN. A NN is simple modifications of standard methods as seen. But NN has more advantages over standard methods.

Here, as an activation function, a step function is used in NN approach. Solving the mathematical method in such applications is sometimes, but not always, quite easy. Both NN and standard methods won't generally suit in more complex separation problems with step function. In general Sigmoid functions are used in NN to solve the problems of step functions.

2.6 Activation Functions

An activation function determines whether a neuron will fire or not for step functions. But step functions are only useful if the network is not complex to solve.

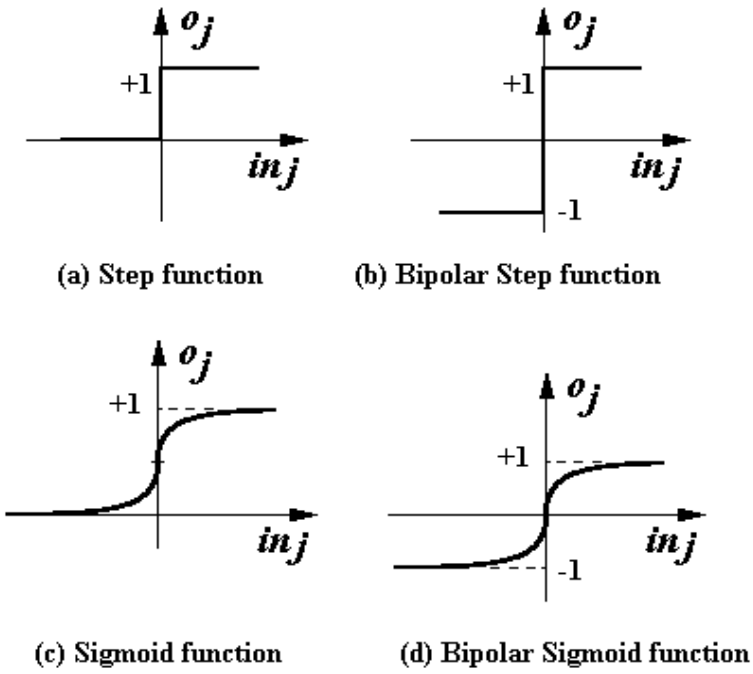


Figure 2.7: Some activation functions

Some functions used in NN applications are shown in Figure 2.7. Use of these functions depends on the updating method of NN.

A step function is a discrete function with a threshold value, $f(x) = ax + t$. Disadvantage of step functions for most NN applications is that it is not differentiable at x=0. This makes it difficult to develop a training algorithm for complex NN structures. Other two functions are, much like step function in their special ranges, *exponential Sigmoid function* $(1/(1 + exp(ax + t))$ in [0,1] interval) or *hyperbolic tangential function* $(tanh(ax + t)$ in [-1,1] interval) also called bipolar Sigmoid. It is also possible to use (1+ tanh(ax+t))/2 for [0,1] interval.

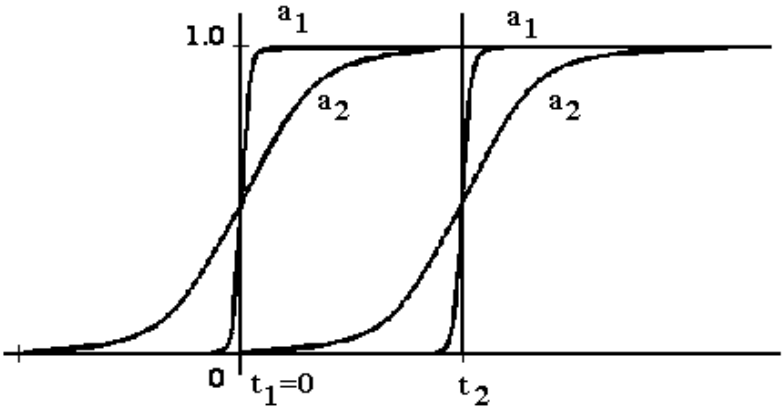


Figure 2.8: The Sigmoid function

In Figure 2.8, $a_1 > a_2$. For a Sigmoid function, increasing parameter a will approximate the function to step function. Parameter t is the threshold value and shifts the function right or left.

In NN, "$ax + t$ " corresponds to the total weighted input of any neuron, and written as $in_j = \sum(w_{ij}x_i + t)$. In this manner, one can say that training of NN is a search for a and t for all neurons, driven by data itself.

2.7 NN Structure

NNs are grouped according to data flow and connections between each pair of neurons or layers. There are two main categories for connections:

- Fully connected : Each neuron on the first layer is connected to all neurons on the second layer.
- Partially connected: A neuron on the first layer does not have to be connected to all neurons on the second layer.

And two main categories for flow of data:

- Feed forward: The neurons on the first layer send their output to the neurons on the second layer, but they do not receive any input back form the neurons on the second layer.
- Bi-directional: There is another set of connections carrying the output of the neurons on the second layer into neurons on the first layer.

Feed forward and bi-directional connections could be fully or partially connected.

2.8 Learning Methods

What makes NN different from mathematical methods is its learning ability. Learning means, a NN changes its parameters during training session. This change is determined by data itself. There are two types of learning methods:

- Supervised: NN is trained for a known input-output data set. Input set is given to NN and its response is compared to desired output set. This type of learning is used to solve similar problems with NN. NN is trained on an example data set and applied to other sets of data having same properties.
- Unsupervised: NN doesn't know the output and tries to find the similarities of data itself. By this method, a NN can reveal unknown properties of data.

2.9 Updating Methods

Whenever a new data is given to NN, it tries to adapt this new data to previous experience, changing its parameters. To make this change effective, one must take care not to loose previous experience of NN while keeping balance for all data.

There are many methods to do this. One of them is Back Propagation (see Appendix A), which generally uses a squared error between NN output and desired output. BP method is applied for supervised learning. First calculation is done for NNs output versus desired output. And this error is used to propagate the error over all neurons in all layers. After each calculation, weights of connections are updated. Updating procedure is continued until the error is small enough, or until a desired training epochs is reached.

There are many other methods for supervised learning: Manhattan Updating, Langevin updating, Conjugate Gradient Search etc. Some of them are based on BP method and increases the performance of BP.

2.10 Application Areas

NN is spread in many different areas. In general, for example, NN can be used in applications for signal processing, speech recognition and production, and pattern recognition such as handwritten characters.

Also there are many examples of NN applications in High Energy Physics. Some of them are mass reconstruction, track finding, off-line feature recognition and on-line triggers.

2.11 So Why ANN?

- Adaptive learning: An ability to learn how to do tasks based on the data given for training or initial experience.
- Self-Organization: An ANN can create its own organization or representation of the information it receives during learning time.
- Real Time Operation: ANN computations may be carried out in parallel, and special hardware devices are being designed and manufactured which take advantage of this capability.
- Fault Tolerance via Redundant Information Coding: Partial destruction of a network leads to the corresponding degradation of performance.

However, some network capabilities may be retained even with major network damage

Chapter 3

NN Requirements - Topology and Methods

3.1 Requirements

Before starting to use NN, it should be decided what type of neural network will be used. To make such decision, one must know the variables used to construct a net. These are:

- Input dimension: the number of nodes in input layer
- Output dimension: the number of nodes in output layer
- Processing elements: the number of hidden layers and hidden nodes
- Neuron activity: activation functions that neurons use
- Updating: updating method and its error function

3.2 Input Dimension

Number of nodes in input layer determines the input dimension of the problem. In other words, this is the input variables of the problem. For example, $f(x, y) = x^2 + y + c$ has two input variables x and y, hence its input dimension is 2, and required number of nodes is thus 2.

In the high energy physics application presented in Chapter 5 the input variables are the χ from a mass constraint, angle between pion daughter particles, and the energy of the pion.

A 3 dimensional input layer is shown in Figure 3.1.

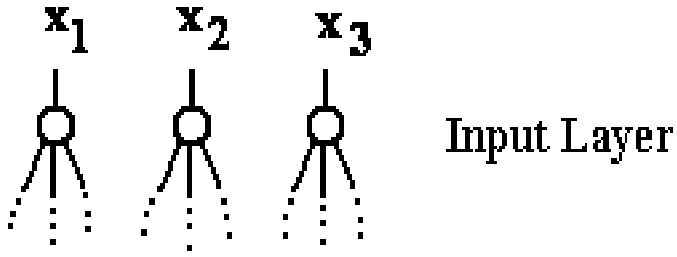


Figure 3.1: Input layer

3.3 Output Dimension

Number of nodes in output layer determines the output dimension of the problem. In general, since we deal with a function of variables, output dimension 1, as in $f(x, y) = x^2 + y + c$, output is f itself.

But neural network has the advantage that it can learn 1 or more functions at the same time with its internal structure. A possible configuration of output layer is shown in Figure 3.2 with 3 outputs.

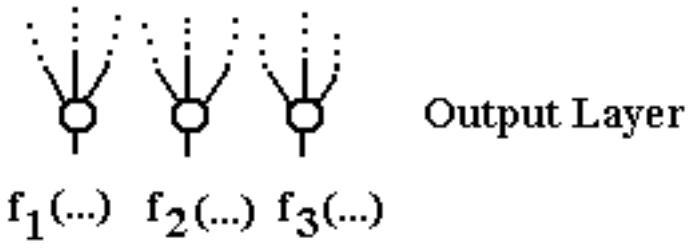


Figure 3.2: Output layer

In HEP data analysis, one of these outputs is the type of particle expressed in terms of 1 or 0, meaning signal or noise (background). The network can be trained for two different particles at the same time, by making 2 dimensions in its output.

3.4 Processing Elements

The number of hidden layers and hidden nodes determines the output of the network. These elements can be considered as the main part of a network, because the analysis of the data is improved at these levels.

3.4.1 Number of Hidden Layers

Without any layers a neural network will only separate the space into two. The first hidden layer will create hyper-surfaces around the data, either enclosing or separating the data types dividing this hyperspace into two. It is also possible that these hyper surfaces can make 1 or more hyper volumes including one type of data. This is dependent on data distribution on space and number of nodes in that layer. The second layer will process the output of the first layer and cluster the data to increase the resolution. This can be shown in Figure 3.3. Adding more layers will only increase clustering.

For many applications, one layer is enough and adding more will result in wrong generalization of data.

The effect of adding more hidden layers is shown in Figure 3.3

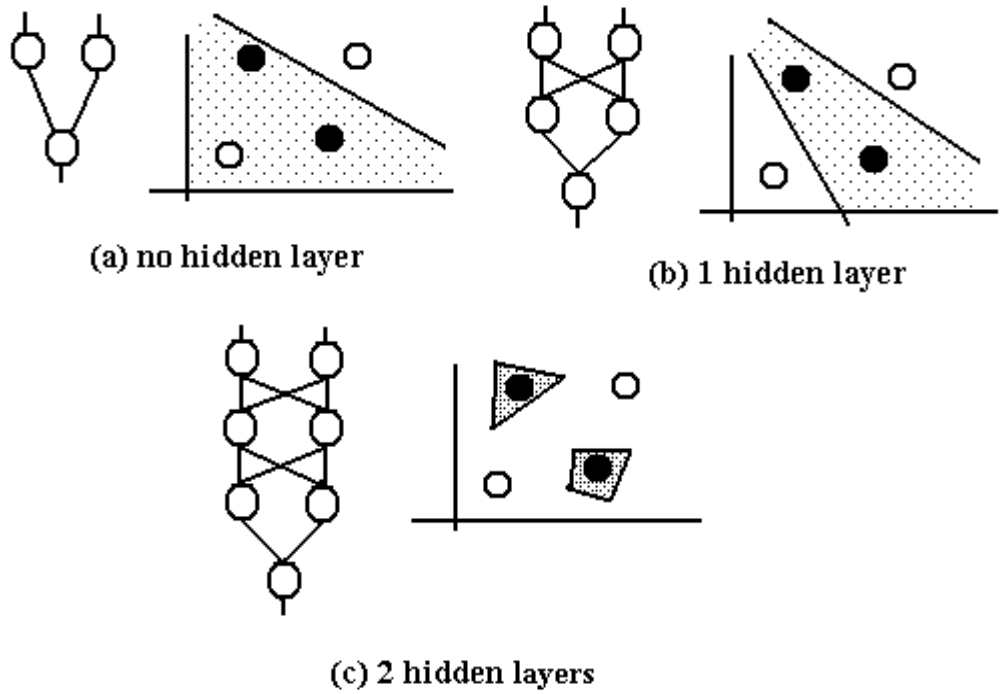


Figure 3.3: Hidden layers

3.4.2 Number of Hidden Nodes

Each node in the layers will make a hyper-surface around the data given to that layer, and divide the space region into two for this data. Combinations of these surfaces either separate the space more accurately or enclose the data in a region.

Selection of the number of nodes will require attention. For any number of

nodes space separation may occur, but a minimum number of nodes is required to make an enclosure.

In Figure 3.4 and 3.5, 1 to n nodes (here n is the input dimension) will make a separation. n+1 and $2n$ are the minimum number of nodes to make an enclosure. n+1 has the ability to make a triangular shape and $2n$ has a rectangular shape.

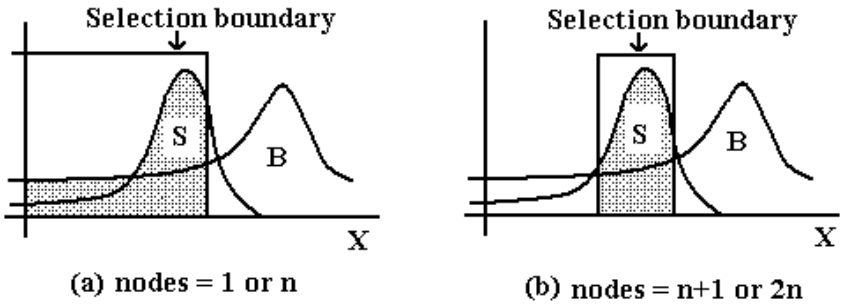


Figure 3.4: Effect of hidden nodes in 1 dimension

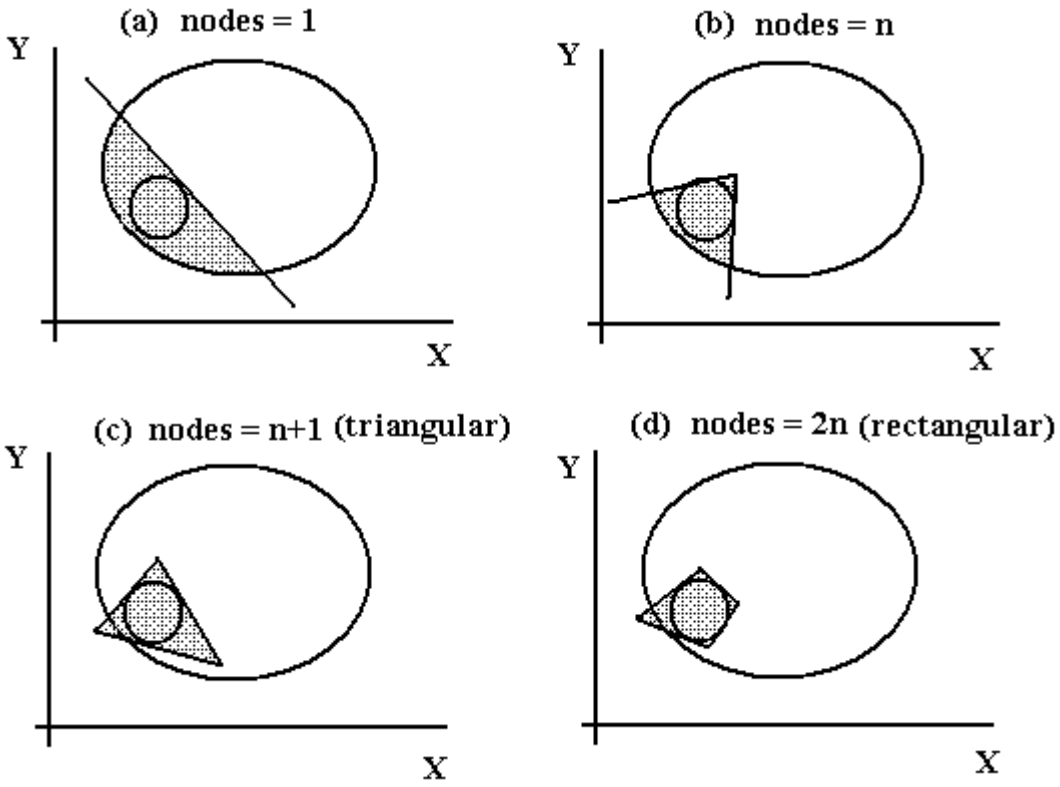


Figure 3.5: Effect of hidden nodes in 2 dimension

3.5 Neuron Activity

This is the activation function of the neuron. It may be a step function or Sigmoid type function depending on the type of the study. A simple network can be trained by using step functions as in Hebbian learning or perceptron learning methods. More complex training methods prefer Sigmoid type functions, or at least functions having a continuous first derivative.

3.6 Updating

There are several methods mentioned in Chapter 2. If one wants to have better results and avoid training errors like local minima problems, advanced techniques can be used. But back-propagation is suitable for most of application for its power and simplicity.

The problem here is that the method may end-up either in global minimum or local minima in the error surface, as seen in Figure 3.6. There are many other methods using algorithms to avoid this local minima problem. Back-propagation, unfortunately can not avoid getting stuck into local minima (if any and if occurs). But even in this situation, NN may give good results. It is also helpful to know that some initial training parameters can also lead BP to global minima.

Two of these initial parameters are the initial weights and learning parameter. These two parameters are important, since they change the convergence. Choosing too low makes training longer and increases the chance to get stuck into a local minimum, while choosing too high moves it too far from global minimum. Suggested initial values are random assignments between $\pm$ 1.0 for weights and near 0.5 for the learning parameter.

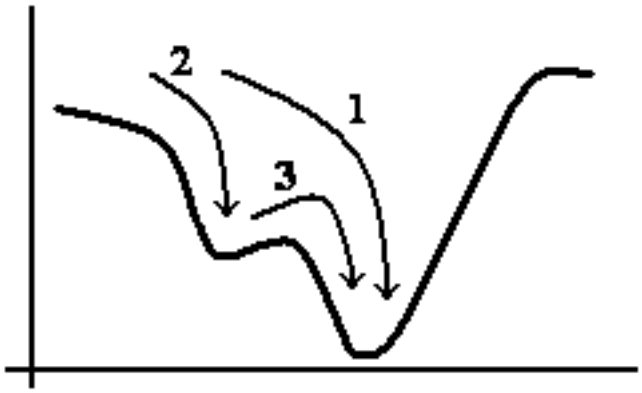


Figure 3.6: Error surface

3.7 Training Outcomes

There three basic conditions the NN may end up in. These are:

- Successfully trained: At the end of training the NN is stable and successful.
- Under-trained : The NN is too simple, or is terminated before it has completed training.
- Over-trained : The NN is too complex, it has learnt the training data in too much detail loosing its generalization ability.

Chapter 4

Preliminary Study

4.1 Preparation

In this chapter, some tests will be conducted to demonstrate the working principles and performance of Neural Networks. Behaviour of NN is investigated by applying different topologies and different data distributions.

As the programming environment, FORTRAN programming language is used and JETNET [4] package (a fortran implementation of NN methods, see Appendix B) is employed for NN.

Since this thesis is concentrated on basic properties, topological study will be limited to only the number of nodes and not to the number of layers. Only a Sigmoid function will be used as the activation function for each node and standard Back-Propagation method will be used to update NN. Input dimension will be 2 and output dimension will be 1. Because of the simplicity of data, only 1 hidden layer will be used.

Data distributions ($Ndata$) will be varied from low statistics (2000) to high statistics (up to 100000) and the initial signal purities will be varied from 10% to 60%

Minimum number of nodes is discussed in Section 3.4.2. In this part it is chosen to be $2n + 1$ (here $n = 2$, then $Nnodes = 5$) except the analysis part for nodes. In node analysis, number of nodes changed from 1 to 10.

The NN is trained for 1000 epochs, this is seen to be enough for all studies presented in this chapter.

After discrimination of signal from background, performance is measured in terms of the product of the selected signal efficiency E and signal purity P;the aim of the discrimination is to maximise this quantity (see Appendix C). The results from the NN are compared to those from a simple standard discrimination method based on cuts to the distributions.

4.1.1 Uniform Data

In this part, two uniformly distributed hypercubes (here squares - two dimension) are generated, one for signal and one for background. The background distribution is separated from the signal distribution by a distance according to the formula derived below. Generated data will be examined by using both a standard method and neural network, both explained below.

In this part, NN is trained on a simple data distribution that can be discriminated easily and perfectly with simple cuts. Given a suitable topology the NN is expected to be able to employ the same cuts and therefore perform equally well.

Preparing Data

Figure 4.1 shows 1 and 2 dimensional distributions. Signal and Background are overlapped at a distance d. d is chosen such that when all Background is rejected, while some of the Signal also rejected, remaining Signal is the half of the total initial Signal expressed as $S/S_0 = 1/2$, making the $E \times P = 1/2$.

for 1 dimension, d is then, $S/S_0 = (b-d)/b = 1/2 \rightarrow d = b(1-1/2) = b/2$

for 2 dimension, d is then, $S/S_0 = (b-d)^2/b^2 = 1/2 \rightarrow d = b(1-1/\sqrt[2]{2})$

These calculations will result in for n dimension $d = b(1-1/\sqrt[n]{2})$

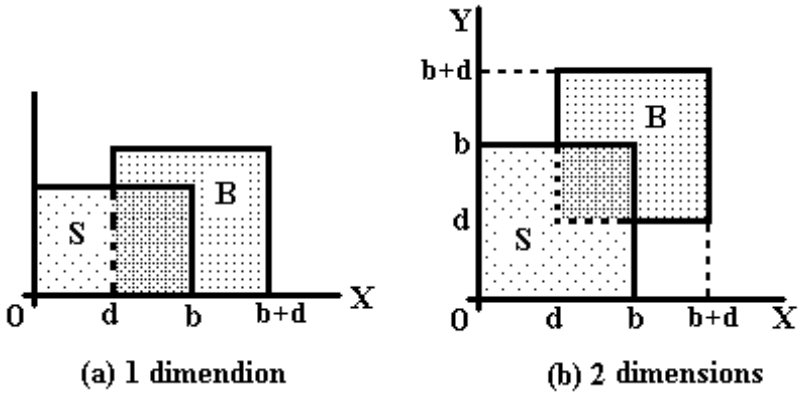


Figure 4.1: Generating uniform data.

Knowing this fact, two data distributions are generated by a random number generator with a mean value 1, multiplied by the factor b and second data is shifted by d to make the background distribution.

Discrimination: Standard Method

Standard Method is chosen as follows:

- select all data in the region 0 to b, both Signal and Background.
- starting at the upper edge at b towards d, all Signal and Background in a square area having diagonal length r are rejected, again both Signal and Background, Figure 4.2, and then calculate $E \times P$ value.
- this procedure is followed until an optimum r value and thus a maximum $E \times P$ value is found or d point is reached. The search is only 1 dimensional for simplicity.

The search for optimal r is meaningless for $0 \rightarrow d$ and for $b \rightarrow b+d$. At d, half of S and all B is rejected where $E \times P = 1/2$. moving back to 0 will decrease the E and thus $E \times P$ will decrease. At b, all of S and half of B is selected where $E = 1$ and $E \times P$ depends the initial signal purity P_0. Moving away from b will decrease the P and thus $E \times P$ will decrease.

The initial signal purity will determine the point where $E \times P$ is maximum. Actually there are two peak point we can get for $E \times P$, one at d and other at b. we already show that at d, $E \times P = 1/2$. At b,

$$E \times P = \frac{S_0}{S_0} \frac{S_0}{S_0 + (B_0/2)}$$

$$S_0 = P_0 \,;\; B_0 = 1 - P_0$$

$$E \times P = \frac{P_0}{P_0} \frac{P_0}{P_0 + (1 - P_0)/2}$$

$$E \times P = \frac{2P_0}{(1 + P_0)}$$

For $P_0 = 1/3$, $E \times P = 1/2$ for b, which makes a decision point whether to choose d or b.

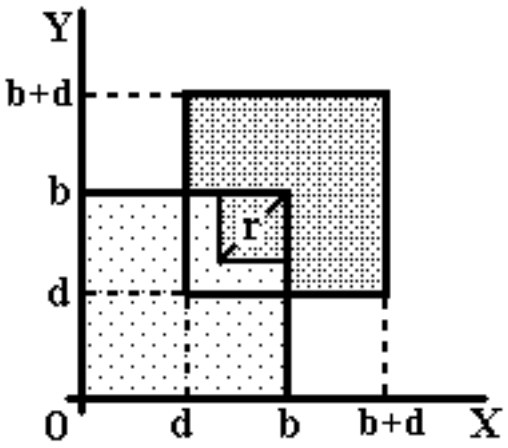


Figure 4.2: Standard method for uniform data discrimination.

Discrimination: Neural Network

NN response to uniform data type will be as seen in Figure 4.3. Procedure is the same as Standard Method.

- search through the NN response, O_{nn}, with the cut variable t.
- for any t, select all data that matches $O_{nn} > t$, and calculate $E \times P$ value.
- continue until an optimum t value, which makes $E \times P$ value maximum, is found. This search is also only 1 dimensional since the output of the NN is one-dimensional.

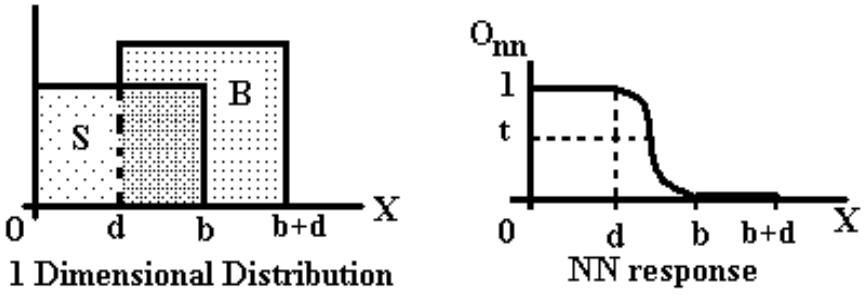


Figure 4.3: NN response to uniform data.

4.1.2 Gaussian Data

In this section, two gaussian hyper-spheres (here two circles in two dimension) are used, again one for Signal and one for Background types.

Preparing Data

In this section, two different gaussian data distributions having different standard deviations are examined. But generation of these data has the same basics with the uniform data study.

Two gaussian distributed data is generated, one having its center at $b/2$ as Signal and other at $b/2 + d$ as Background. d is calculated in the same way found in uniform data study.

Discrimination: Standard Method

For gaussian data type, one must assume that centers of the spheres are known, then the following procedure is conducted:

- for each data point, calculate the distances to the centers of Signal and Background spheres. At point P, $r_1 = \sqrt[2]{\sum(P(x_i) - o_1(x_i))^2}$ and $r_2 = \sqrt[2]{\sum(P(x_i) - o_2(x_i))^2}$, Figure 4.4.
- select all data matching $r_1 < kr_2$ (all possible Signal candidates) and calculate $E \times P$ value
- search for an optimal k value that makes $E \times P$ value maximum.

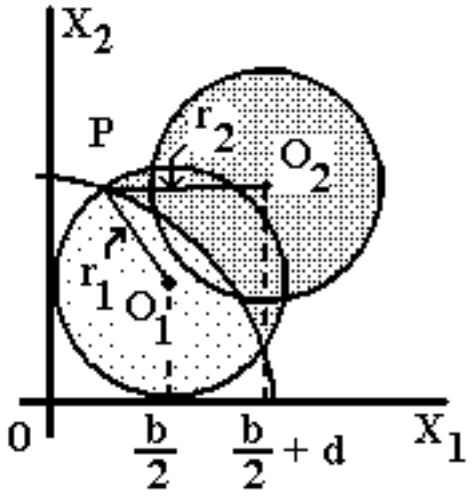


Figure 4.4: Standard method for Gaussian data discrimination.

Discrimination: Neural Network

The procedure for gaussian data type is the same as uniform data, since the response of NN is in 1 dimension. Search for optimal r value that makes $E \times P$ value maximum.

4.2 Results

Figure 4.5 and Figure 4.6 show the distribution of the data for both uniform and gaussian types for 20% initial signal purity. As the deviation of Background increases for gaussian data, its tails get wider and its peak decreases.

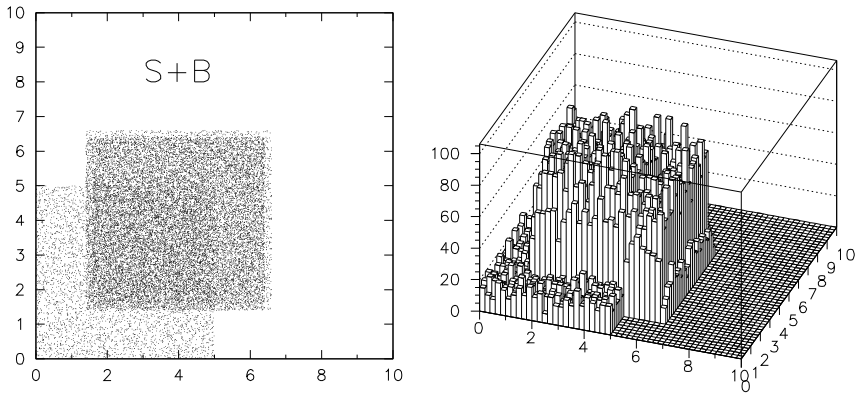


Figure 4.5: Uniform data - distribution.

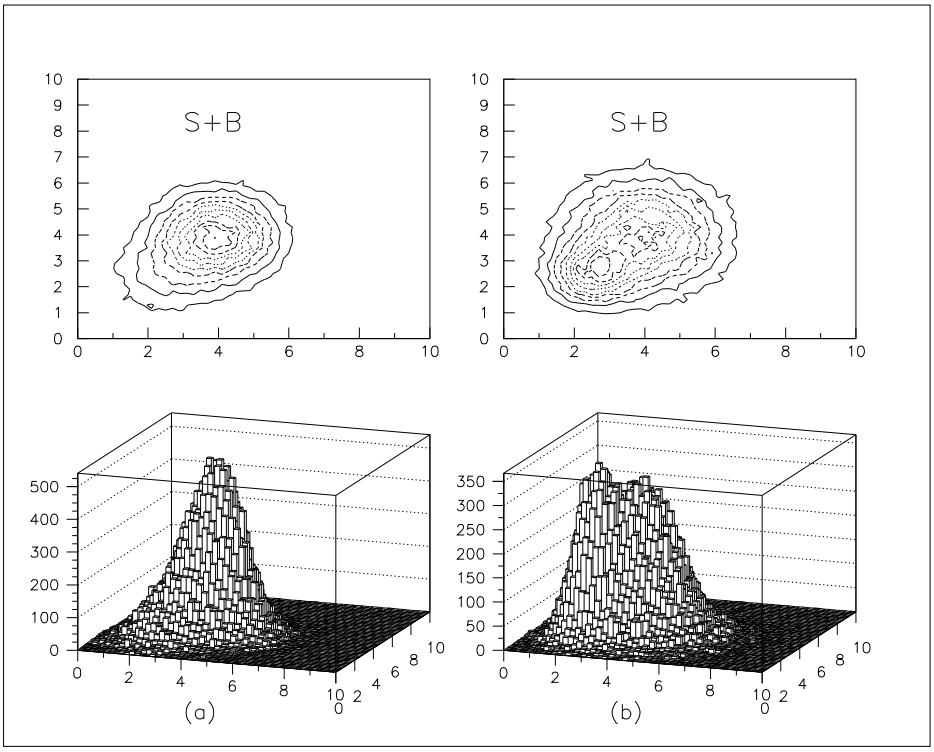


Figure 4.6: Gaussian data - distribution: (a) $\sigma_S = 1.0$, $\sigma_B = 1.0$, (b) $\sigma_S = 0.7$, $\sigma_B = 1.3$).

4.2.1 Uniform Data

Figure 4.7 shows the effect of changing the number of data. NN performs as well as the SM for high statistical data. Here data increases according to the form $Ndata = 2000(Data\,scale)^2$ (a programming choice to scan a wide data region with a few number of parameters). Figure 4.8 shows the effect of changing the initial signal purity. Both NN and SM works well. Figure 4.9 shows the effect of changing topology of NN, here number of nodes. Increase in the number of nodes result in a significant increase in $E \times P$.

4.2.2 Gaussian Data

For gaussian data type, figures show that NN works better than the selected SM technique. Here two data distributions are used having different S and B standard deviation. Distribution (a) includes data with $\sigma_S = 1.0$, $\sigma_B = 1.0$, and distribution (b) includes data with $\sigma_S = 0.7$, $\sigma_B = 1.3$.

Figure 4.10 shows the effect of changing the number of data. Here data increased according to this formula $Ndata = 2000(Data\,scale)^2$. NN performs better than SM for low statistical data. For high statistical data, NN shows a little decrease in its performance, and also a decrease in the error. This is because the distribution of data gets very complex at the decision boundary for high statistical data.

Figure 4.11 shows the effect of changing the initial signal purity. NN performance is better than SM. Figure 4.12 shows the effect of changing topology of NN, here number of nodes. Increase in the number of nodes result in a significant increase in $E \times P$ of NN results.

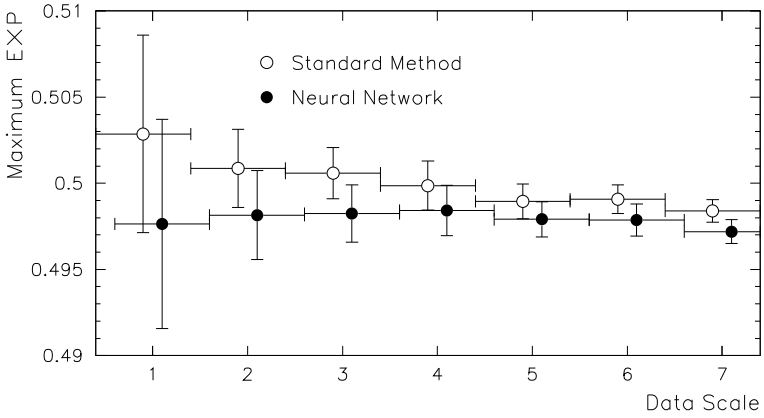


Figure 4.7: Uniform data - results for number of data.

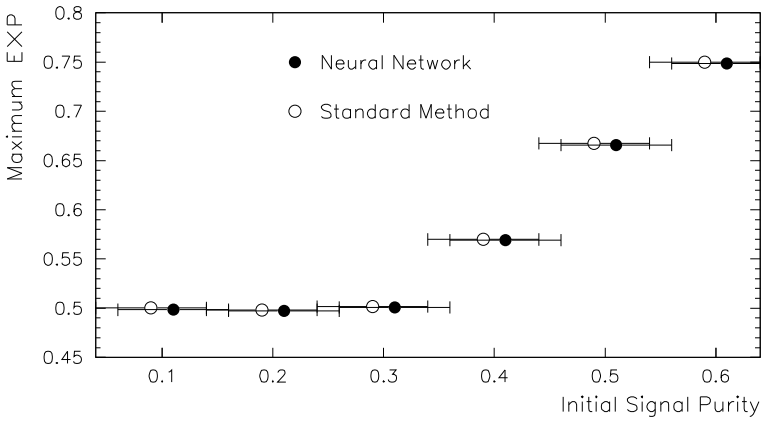


Figure 4.8: Uniform data - results for initial signal purity.

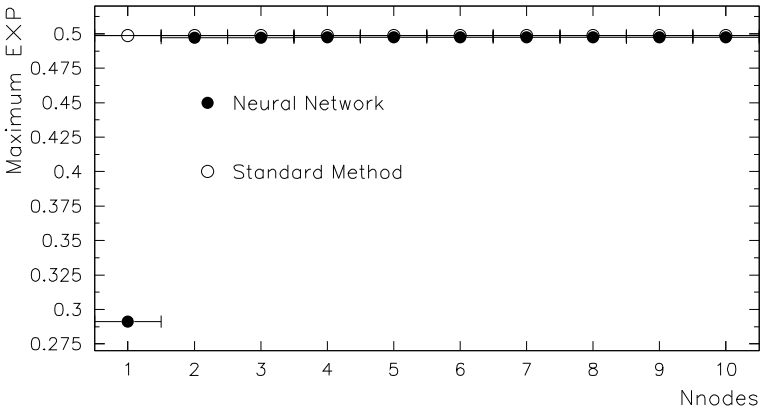


Figure 4.9: Uniform data - results for number of nodes.

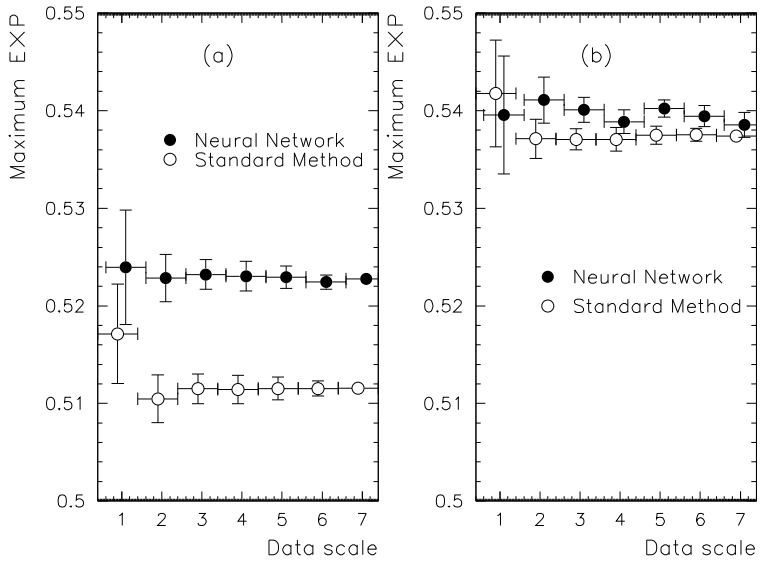


Figure 4.10: Gaussian data - results for number of data: (a) $\sigma_S = 1.0$, $\sigma_B = 1.0$, (b) $\sigma_S = 0.7$, $\sigma_B = 1.3$).

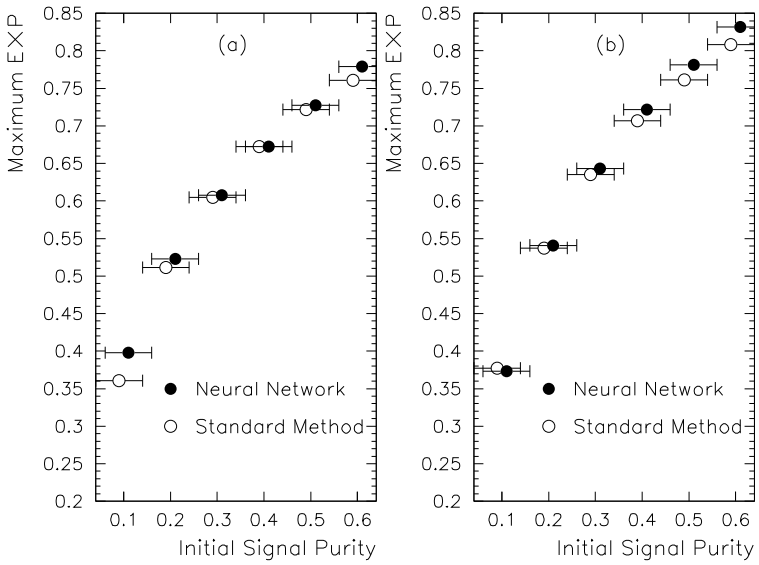


Figure 4.11: Gaussian data - results for initial signal purity: (a) $\sigma_S = 1.0$, $\sigma_B = 1.0$, (b) $\sigma_S = 0.7$, $\sigma_B = 1.3$).

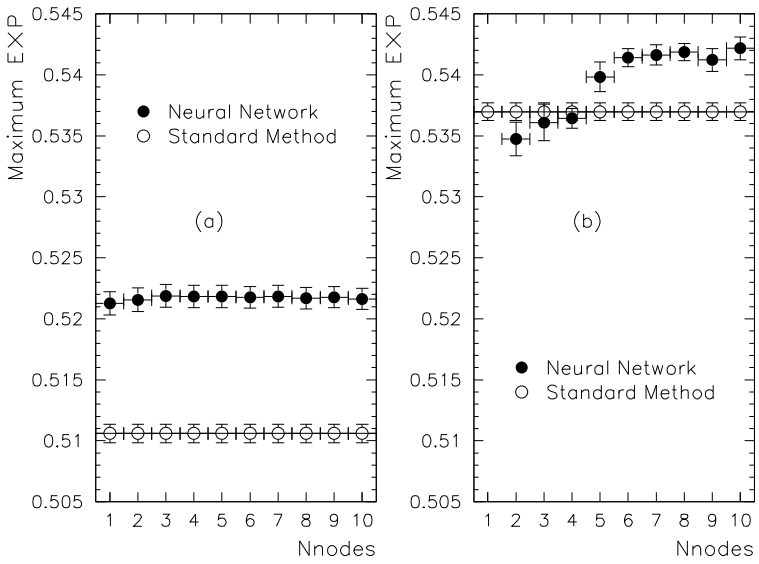


Figure 4.12: Gaussian data - results for number of nodes: (a) $\sigma_S = 1.0$, $\sigma_B = 1.0$, (b) $\sigma_S = 0.7$, $\sigma_B = 1.3$).

Chapter 5

π^0 Study

In this chapter a neural network is employed in the discrimination of π^0 signal from background.

5.1 Data and Topology

The data used is Monte-Carlo data from the ALEPH experiment where four-million hadronic decays of the Z^0 are simulated by the Jetset program and passed through a full detector simulation. Photons whose energy and position are measured in the electromagnet calorimeter are combined in pairs to reconstruct the decay $\pi^0 \rightarrow \gamma + \gamma$. Pion candidates are initially selected from a $\pm 3\sigma$mass window, Figure 1.2, with a purity of about 32%; at this stage we define the efficiency to be 100%. The aim of the neural network is to further discriminate between the signal and background to improve the product of the efficiency E and purity P. For this, three discriminants can be used; the first is a χ value from a constraint on the mass of the π^0 candidate, true π^0s will tend to have a smaller χ than false candidates. The second is the opening angle between the photons in the lab frame, again this value is generally smaller for true π^0s. A third discriminant is the energy of the π^0 candidate, this may help to further separate signal from background as the two exhibit different kinematics.

The data files, separated into 20 training and 20 test data sets, contain χ, Θ_{12} and E_{12} of the reconstructed π^0 candidates, and a signal tag for Signal and Background. The neural network is trained on the training sets, and the optimal threshold value t is searched for using the test data sets. The mean and standard errors are calculated from the 20 results. Figure 5.1 shows the distributions of χ and Θ_{12} which are used in the 2D analyse.

NN is chosen with its most common and simple topology and properties, as before in preliminary work. Fully connected and feed-forward network is selected, all activations functions are chosen to be exponential Sigmoid and the

error function is $E = \frac{1}{2}\sum(t_i - o_i)^2$

5.2 Comparison of different discrimination methods

Data is analyzed with different discrimination methods:

(1) A *standard* method where the signal is selected from within an ellipsoid whose shape is optimised,

(2) A *histogrammed* probability density estimation (PDE) method based on simple histogramming

(3) A *smoothed histogrammed* probability density estimation (PDE) method based on smoothed histogramming

(4) A *neural network* method.

The first three analyses are conducted by my colleagues.

Analysis of the data has the same procedure as for the studies of Chapter 4. The aim is to find maximum $E \times P$ value for different number of nodes and for different number of data. Timings for each method are also recorded.

Figure 5.2 and 5.3 show the changes in $E \times P$ for different number of nodes in $2D$ and $3D$. The number of pion candidates, $Ndata$, increases with $Data\,scale$ according to the form: $Ndata = 5 \times 10^{Data\,scale}$. From these figures it is seen that the NN improves the $E \times P$ from an initial value of about 32% to about 39% in 2-dimensions and to about 40% in 3-dimensions. In 2-dimensions there is little sensitivity to the number of nodes, however in 3-dimensions higher number of nodes give slightly greater performance.

Figure 5.4 and 5.5 show a comparison of different discrimination techniques. From these figures it is seen that all techniques give similar results with the exception of the standard method that does not perform well in three dimensions. The smoothed histogram PDE method and the NN method perform relatively well in the case of low statistics, this indicates that the NN is effective at smoothing data of low statistics.

Figures 5.6 and 5.7 show the comparison of total run times for each method. For all methods the total run time is seen to increase with the number of data. In 2-dimensions the NN is seen to be significantly slower than the other methods, however, with an increase to 3-dimensions the NN run time is comparable with those of the other methods. This is explained in Appendix D.

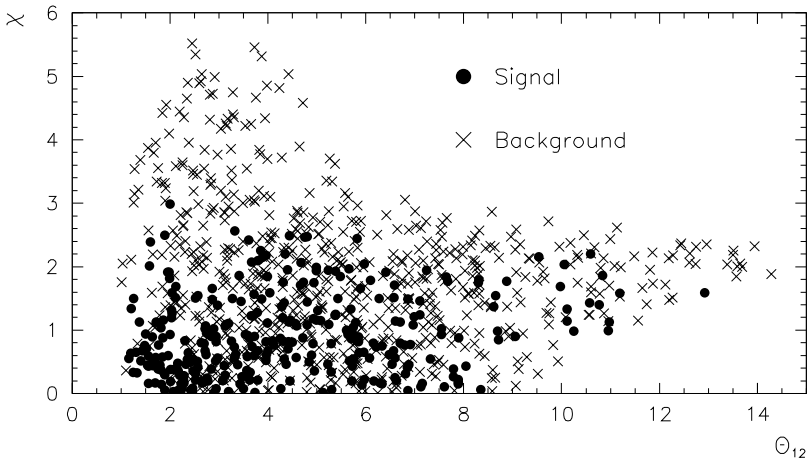


Figure 5.1: χ vs. Θ_{12} distribution. Signals tend to have lower χ and Θ_{12} values.

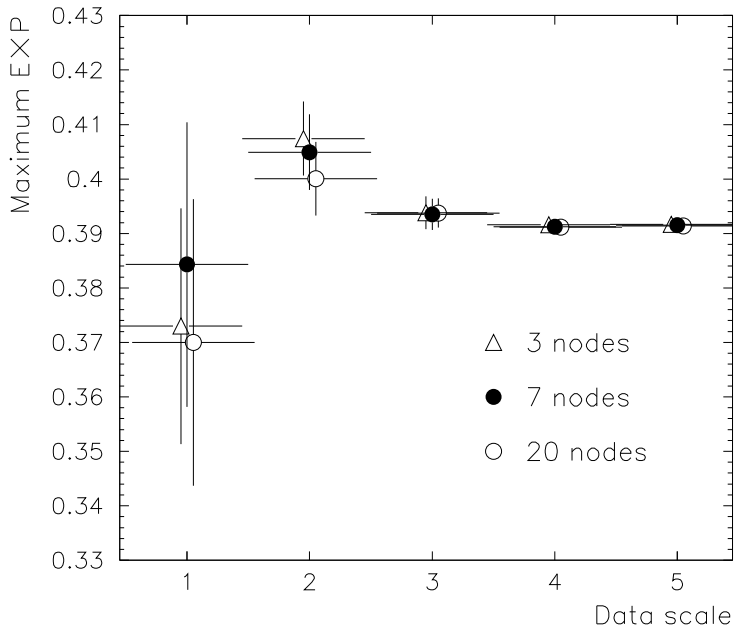


Figure 5.2: NN response to 2D data for different nodes.

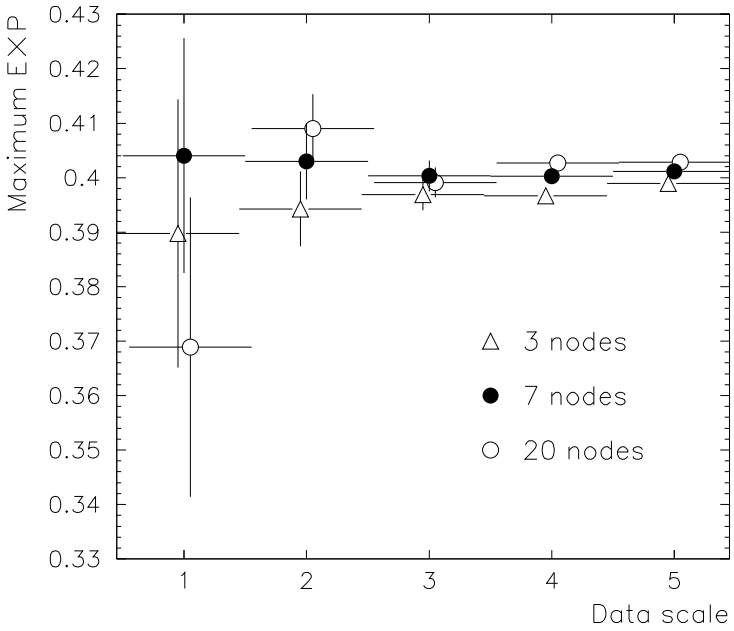


Figure 5.3: NN response to 3D data for different nodes.

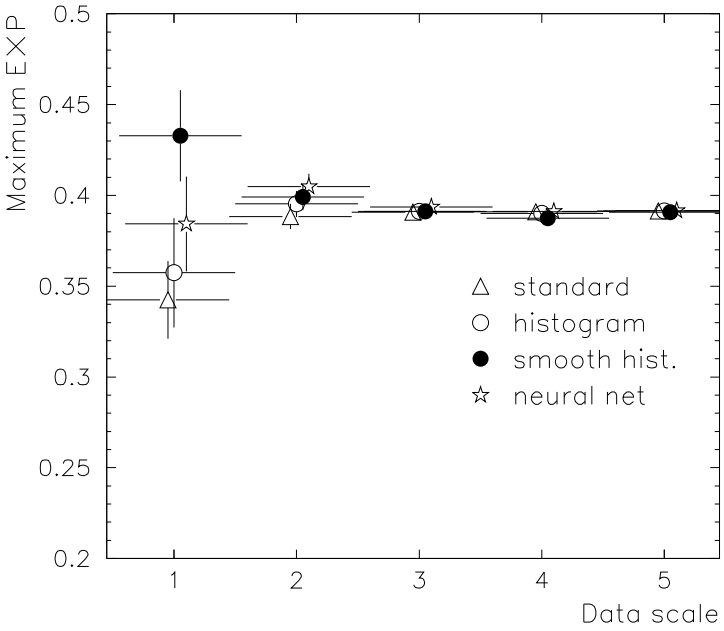


Figure 5.4: Comparison of discrimination techniques in 2D.

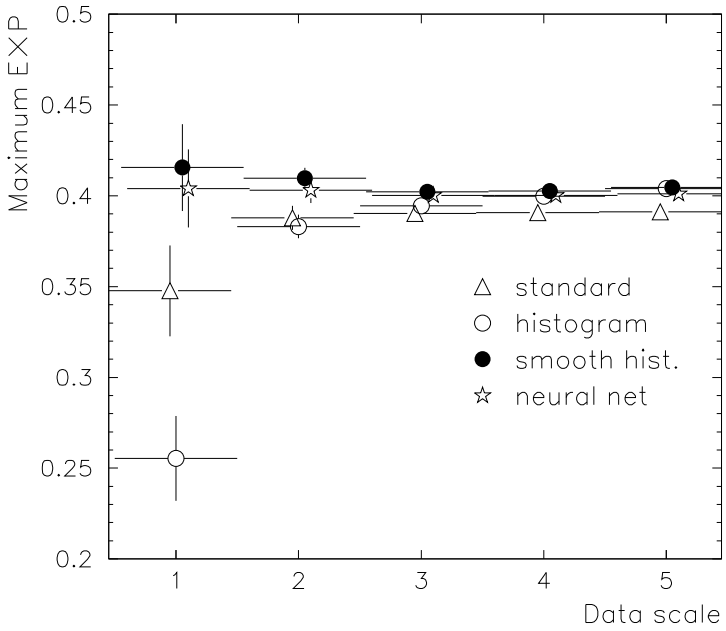


Figure 5.5: Comparison of discrimination techniques in 3D.

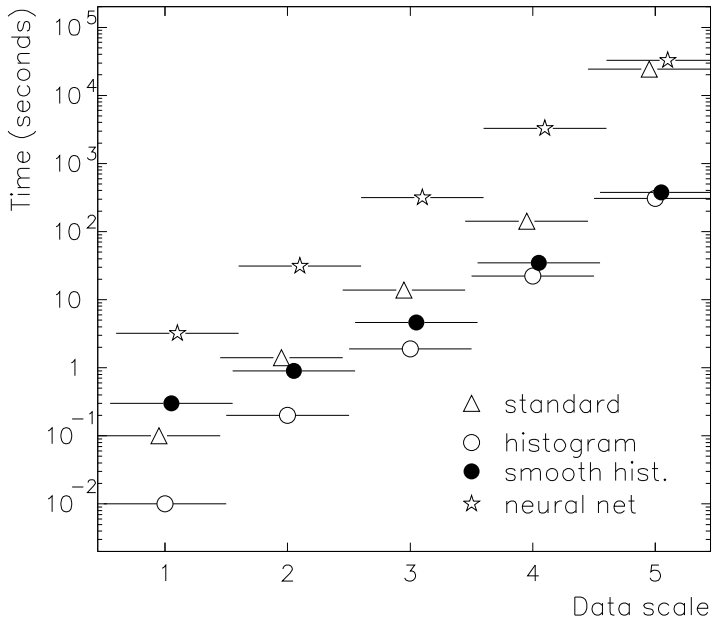


Figure 5.6: Time required for training in 2D.

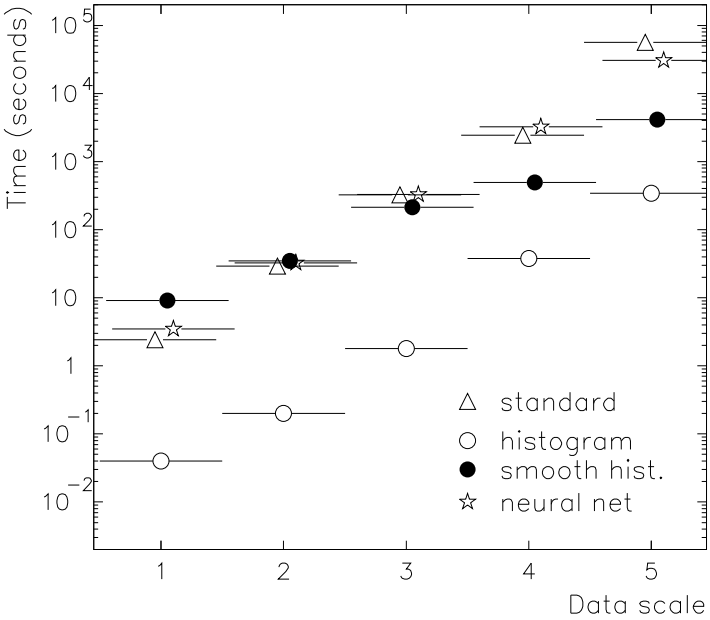


Figure 5.7: Time required for training in 3D.

Chapter 6

Summary and Conclusion

Artificial Neural Networks have been used in many areas including speech recognition, character recognition, data acquisition and data analysis etc. This study has shown that NN can give a comparable performance to other methods in the discrimination of signal and background. Some advantages of the NN can be noted: while a standard cut methods needs some analysis of the data (choice of the shape of the discriminating function), the NN needs no such analysis and so can be employed quickly. The NN is seen to perform well with data of low statistics as smoothing is inherent to the NN. The NN scales easily to high dimensions (simply by increasing the number of input nodes) without the geometric increase in run times experienced by other methods. At lower dimensions the NN is seen to be much slower than the other methods but is expected to be faster for high dimensional data.

Although no optimization, other than number of nodes, is studied, this study showed that NN is a good discriminator in its simplest form even with a low number of nodes.

List of References

[1] Ham, F. M., & Kostanic, I. (2001) , *Principles of Neurocomputing for Science and Engineering*, Mc Grav Hill.

[2] Fausett, L. V. (1994), *Fundamentals Of Neural Networks: Architectures, Algorithms and Applications*, Prentice Hall.

[3] Freeman,J. A., & Skapura, D. M. (1991), *Neural Networks: Algorithms, Applications and Programming Techniques*, Addison Wesley.

[4] Peterson, C., & Rognvaldsson, T., & Lonnblad, L. (1993) *JETNET 3.0 - A Versatile Artificial Neural Network Package*, Lund University Preprint LU-TP 93-29,

Appendix A

Back-Propagation algorithm

Back-Propagation, based on the method known as Delta Rule, is a very common and simple method for updating a NN.

A.1 Derivation

Let's first choose our notations. Any input and output to neurons are,

$$x_{i\rightarrow nodes}^{n\rightarrow layers}$$

and weights in every connection are,

$$w_{i,j\rightarrow between\,nodes}^{n-1,n\rightarrow between\,layers}$$

where $n = 0 \rightarrow m$ and $i, j = 1, l$. $n = 0$ is input layer and $n = m$ is output layer.

$$net_j^n = \sum w_{i,j}^{n-1,n} x_i^{n-1} + \theta_j^{n:-1,n}$$

$$x_j^n = f(net_j^n)$$

where f is a sigmoid function.

The error to be used is,

$$E^m = \frac{1}{2}\sum (t^m - x_j^m)^2)$$

where t^m is the desired output.

Update on output weights is

$$\Delta w_{i,j}^{m,m-1} = -\frac{\delta E^m}{\delta w_{i,j}^{m-1,m}}$$

$$= -\frac{\delta E^m}{\delta x_j^m}\frac{\delta x_j^m}{\delta net_j^m}\frac{\delta net_j^m}{\delta w_{i,j}^{m-1,m}}$$

$$= ((t_j^m - x_j^m))x_j'^m)(\frac{\delta}{\delta w_{i,j}^{m-1,m}}(\sum w_{i,j}^{m-1,m} x_i^m))$$

$$= ((t_j^m - x_j^m))x'^m_j)x_i^m$$

where the first term is called "delta".

$$\delta_j^m = x'^m_j(t_j^m - x_j^m)$$

and finally

$$w_{i,j}^{m-1,m}(new) = w_{i,j}^{m-1,m}(old) + \Delta w_{i,j}^{m-1,m}$$

where

$$\Delta w_{i,j}^{m-1,m} = \eta \delta_j^m x_i^m$$

here η is the learning parameter.

The weight updates for hidden nodes can be computed from the error of output node. Error can be expressed as,

$$E^m = \frac{1}{2}\sum(t^m - x_k^m)^2) = \frac{1}{2}\sum(t^m - f(net_k^n))^2$$

$$= \frac{1}{2}\sum(t^m - f(\sum w_{j,k}^{m-1,m} x_j^{m-1} + \theta_k))^2$$

$$= \frac{1}{2}\sum(t^m - f(\sum w_{j,k}^{m-1,m} f(net_j^{m-1})) + \theta_k))^2$$

and

$$\Delta w_{i,j}^{m-2,m-1} = -\frac{\delta E^m}{\delta w_{i,j}^{m-2,m-1}}$$

$$= -\frac{\delta E^m}{\delta x_k^m}\frac{\delta x_k^m}{\delta net_k^m}\frac{\delta net_k^m}{\delta x_j^{m-1}}\frac{\delta x_j^{m-1}}{\delta net_j^{m-1}}\frac{\delta net_j^{m-1}}{\delta w_{i,j}^{m-2,m-1}}$$

$$= (\sum(t_k^m - x_k^m))x'^m_k w_{j,k}^{m-1,m} f'(net_j^{m-1}))x_i^{m-1}$$

where the first term is called "delta" again.

$$\delta_j^{m-1} = \sum(t_k^m - x_k^m))x'^m_k w_{j,k}^{m-1,m} f'(net_j^{m-1})$$

$$\delta_j^{m-1} = f'(net_j^{m-1}) \sum \delta_k^m w_{j,k}^{m-1,m}$$

this equation gives a relation between δ terms for neighbouring layers ($\delta_i^{m-1} \Leftrightarrow \delta_j^m$). And finally

$$w_{i,j}^{m-2,m-1}(new) = w_{i,j}^m(old) + \Delta w_{i,j}^m$$

where

$$\Delta w_{i,j}^{m-2,m-1} = \eta \delta_j^{m-1} x_i^{m-1}$$

This is the error for the layer connected to the output layer.

The threshold can be given as another weight on a connection whose input is always equal to 1, and starting index for i is 0,

$$\theta_j^{n-1,n} = w_{0,j}^{n-1,n}\ ;\ x_0^{n-1} = 1$$

A.2 Procedure

(1) initialize all weights and thresholds randomly

(2) give n^{th} data to network

(3) calculate net_j^n and x_j^n for all neurons on layers.

(4) calculate derivatives x'^n_j, and δ of any layer starting from output to first hidden layer.

(5) calculate all weight and threshold changes, $\Delta w_{i,j}^{n-1,n}$ between any layer.

(6) apply changes to all weights and thresholds

(7) until a predefined termination condition reached, repeat steps (3) to (6) for $(n+1)^{th}$ data

Appendix B

JETNET

JETNET is a fortran77 subroutine collection of common NN training algorithms.

To use this package, one must know the parameters to initialize and how to use training and testing subroutines in his programs. JETNET program and a manual can be found at its download page,

- http://www.thep.lu.se/ftp/pub/LundPrograms/Jetnet/

Although all parameters are explained in the program itself, and some has initial values, it is useful to explain important parameters here.

- `MSTJN(1)` : number of layer in net. default is 3 and maximum 9 hidden layers.
- `MSTJN(3)` : overall transfer function used in net. `IGFN(I)` can be used to give each layer a different transfer function (compatible with the method, default $g(x) = 1/(1 + exp(-2x))$).
- `MSTJN(4)` : error measure for updating method. default $E = 0.5(o - t)^2$ (summed square error).
- `MSTJN(5)` : updating procedure. default is standard Back-Propagation method.
- `MSTJN(10+I)` : number of nodes in layer I (I=0 is the input layer).
- `PARJN(1)` : learning parameter eta. `ETAL(I)` can be used to give different values for different layers.
- `PARJN(4)` : width of initial weights (filled randomly by JETNET).

When all required parameters set correctly, JETNET is initialized by calling "`JNINIT`" subroutine. After initialization, data must be given to JETNET

by using `OIN(I)` and `OUT(I)`. `OIN(I)` is the input and `OUT(I)` is the output of the JETNET. Training and testing is done by calling `JNTRAL` and `JNTEST` subroutines. `OUT(I)` is changed during training, so it must be reentered for each data. An example training and testing for 2 input 1 output is:

- Training

```
DO data=1,10     !  10 data in training set
OIN(1)=x(1)      !  first input
OIN(2)=x(2)      !  second input
OUT(1)=y(1)      !  output
CALL JNTRAL      !  call training
response=OUT(1)  !  response of network
END DO
```

- Testing

```
DO data=1,10     !  10 data in test set
OIN(1)=x(1)      !  first input
OIN(2)=x(2)      !  second input
CALL JNTEST      !  call training
response=OUT(1)  !  response of network
END DO
```

Appendix C

The Performance Measure, $E \times P$

Performance of discrimination methods is measured by their ability to improve the efficiency and purity of studied data.

Signal efficiency E is defined by the ratio of the number of selected signals to the number of initial signals:

$$E = \frac{S}{S_0}$$

Signal purity P is defined by the ratio of the number of selected signals to the total number of selected signal and selected background:

$$P = \frac{S}{S+B}$$

Initial efficiency and initial purity are:

$$E_0 = \frac{S_0}{S_0} = 1$$

$$P_0 = \frac{S_0}{S_0 + B_0} = \textit{Initial Signal Fraction}$$

Improving one of these numbers will result in a decrease in other. In this study the maximization of the product $E \times P$ satisfies the requirement that the selected candidates have both reasonable efficiency and purity. $E \times P$ value is initially equal to P_0 since $S_0 = 1$.

Appendix D

NN Run-Time

One of the characteristic properties of a neural network is that it can be very fast on processing very complex type data depending on its topology. Actually the time-consuming part of NN training calculation of the outputs of all nodes and the updating of the weights in all connections between all layers.

The running time of the NN is proportional to the total connections between layers. Lets consider a 3-layered NN, with i nodes in the input layer which determines the input dimension, j nodes in hidden layer and k nodes in output layer. if we denote total connection with C then:

$$C = (i \times j) + (j \times k)$$

If one increases only the input dimension by a it will be

$$C_{new} = ((i + a) \times j) + (j \times k)$$

and difference

$$\Delta C = C_{new} - C_{old} = a \times j$$

which directly relates to the increase of total run time

$$t_{new} - t_{old} \propto \ C_{new} - C_{old} = a \times j$$

$$\Delta t \propto \ \Delta C$$

which is a linear function depending on the change of the input dimension.

The effect of changing the number of nodes in any layer on the time for any topology of NN can be estimated using the same method.

Most of the standard methods used in data analysis other than NN, uses a variable search of each dimension separately. $[a, b]$ interval is divided into n equal partitions, then an optimal value of r is searched in this interval, each n partition takes t' time complete, and a total run required is $t = n \times t'$.

Each dimension requires an independent $r_{optimum}(dimension)$ and a division nt. For example, in 2 dimension n and m are the intervals and total time

required is $t = n \times m \times t'$. In 3 dimension, it will be $t = n \times m \times o \times t'$ for divisions n, m and o. If the steps are taken to be equal, then the total time can be expressed in terms of the power of dimensions; $t = t' \times n^{dimension}$.

At low dimensions NN are seen to be slow compared to standard methods, however due to the linear increase of NN running time with the input number, at high dimensions the NN is expected to be faster than standard methods.