The Use of Backpropagating Neural Networks in Coordination with Logistic Spline Coefficients to Obtain the Change-Point

by

Elisa M. Maldonado-Colberg

A thesis submitted in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE
in
Mathematics - Scientific Computation

UNIVERSITY OF PUERTO RICO
MAYAGÜEZ CAMPUS
2006

Approved by:

_____	_____
Robert Acar, PhD Member, Graduate Committee	Date
_____	_____
Pedro Vásquez-Urbano, PhD Member, Graduate Committee	Date
_____	_____
Daniel L. McGee, PhD President, Graduate Committee	Date
_____	_____
Jaime Seguel, PhD Representative of Graduate Studies	Date
_____	_____
Luis F. Cáceres-Duque, PhD Chairperson of the Department	Date

ABSTRACT

This work presents an algorithm that uses backpropagating neural networks and their ability to find logistic regression coefficients as a useful tool that may outperform other conventional methods for finding the change-point of a data set. To seek this objective, we first demonstrate the capacity of a one-layered neural network to find logistic regression coefficients. Then, we add another layer to the network in order to determine if two-layer discriminants would allow us to approximate the change-point. We then develop an algorithm based on hidden layer coefficients of neural networks that allows us to assess change-points based on when a neural network switches from one "neuron" to another.

RESUMEN

Este trabajo presenta un algoritmo que utiliza redes neurales artificiales de retropropagación y su habilidad para encontrar coeficientes de regresión logística como herramienta útil que puede superar métodos convencionales para hallar el punto de cambio en un conjunto de datos. Para lograr este objetivo, primero se demuestra la capacidad que tiene una red neural de un nivel para encontrar coeficientes de regresión logística. Entonces, añadimos otro nivel adicional a la red neural para determinar si los dos discriminantes que ésta calculó nos aproximan al punto de cambio. Luego, se creó un algoritmo basado en coeficientes de un nivel escondido de la red que nos ayudará a encontrar los puntos de cambio basándonos en el momento en que la red se mueve de una neurona a otra.

To my parents and siblings, whose love, support and patience have taught me that every good thing in life requires sacrifice. Thank you for teaching me the value of hard work and perseverance. All my personal and professional achievements would have not been possible without your inspiration, love and support.

ACKNOWLEDGEMENTS

I want to start giving a special recognition and acknowledgement to my adviser Dr. Daniel McGee, who besides being my mentor and adviser, he became a sincere and true friend who believed in me, even when there were moments that I felt tired and frustrated. In addition, I would to thank Dr. Pedro Vásquez-Urbano and Dr. Robert Acar for being part of my graduate committee. Your detailed scrutiny and contributions to this work were very valuable and truly appreciated.

The grant from NSF PRECISE provided most of the funding and the resources for the development of this research.

And last, but not least, I would like to thank again my family, for their unconditional love, patience and support.

Table of Contents

ABSTRACT ..... **II**

RESUMEN ..... **III**

ACKNOWLEDGEMENTS ..... **V**

TABLE OF CONTENTS ..... **VI**

TABLE LIST ..... **VIII**

FIGURE LIST ..... **IX**

1 INTRODUCTION **2**

1.1 LITERATURE REVIEW 2
1.1.1 History of Artificial Neural Networks *2*
1.1.2 Cumulative Sum to Find the Change-point *5*
1.1.3 Minimum Message Length Method to Find the Change-point *5*
1.1.4 Classification and Regression Trees *6*
1.2 RELEVANCE OF THE RESEARCH PRESENTED IN THIS DISSERTATION 7
1.3 SUMMARY OF FOLLOWING CHAPTERS 8

2 THEORETICAL BACKGROUND **10**

2.1 WHAT IS AN ARTIFICIAL NEURAL NETWORK? 10
2.2 HOW DO ARTIFICIAL NEURAL NETWORKS WORK? 13
2.3 SINGLE LAYER NEURAL NETWORKS 15
2.4 LOGISTIC REGRESSION AND MAXIMUM LIKELIHOOD EQUATIONS 21
2.5 NEURAL NETWORKS WITH TWO HIDDEN LAYERS 24
2.6 DOUBLE LAYER NEURAL NETWORKS AND THE CHANGE-POINT 29
2.6.1 Interpretation of the First Hidden Layer *29*
2.6.2 Interpretation of the Second Hidden Layer *32*
2.7 NOTES ON OBTAINING THE COEFFICIENTS 33

3 MULTI-LAYER NEURAL NETWORKS AND THE BACKPROPAGATION ALGORITHM **33**

3.1 OVERVIEW OF THE TRAINING PROCESS 33
3.2 MATHEMATICAL FORMULATION FOR UPDATING COEFFICIENTS 38
3.3 APPLYING NEURAL NETWORKS 40

4 SINGLE LAYER ARTIFICIAL NEURAL NETWORK AND LOGISTIC REGRESSION **42**

4.1 CART, LOGISTIC REGRESSION AND SINGLE LAYER NEURAL NETWORKS 42
4.2 EXPERIMENT USING LOGISTIC REGRESSION TO OBTAIN LEARNING PARAMETERS FOR NEURAL NETWORKS 44
4.2.1 Methodology *44*
4.2.2 Results *44*
4.2.3 Conclusion *46*

5 THE CHANGE-POINT PROBLEM AND DISCRIMINANTS 46

5.1 INTRODUCTION TO DISCRIMINANTS 47
5.2 PRELIMINARY EXPERIMENT - DISCRIMINANTS WITH LOGISTIC REGRESSION, ONE-LAYER NEURAL NETWORKS AND TWO-LAYER NEURAL NETWORKS 49

6 THE CHANGE-POINT PROBLEM – NEW RESEARCH WITH FUZZY DATA 54

6.1 INTRODUCTION 54
6.2 METHODOLOGY 55
6.2.1 Algorithm to Find the Change-point *55*
6.2.2 Data Sets to be Tested *56*
6.3 RESULTS 59
6.3.1 Data Set 1 ..... *59*
6.3.2 Data Set 2 ..... *60*
6.3.3 Data Set 3 ..... *62*
6.3.4 Data Set 4 ..... *64*
6.3.5 Data Set 5 ..... *66*
6.3.6 Data Set 6 ..... *67*
6.3.7 Conclusion and Summary *69*

7 CONCLUSIONS AND FUTURE WORK 70

Table List

Tables **Page**

Table 4-1 Logistic Regression vs. Artificial Neural Networks..... 43
Table 4-2 Logistic Regression vs. Artificial Neural Networks (Cont.) 43
Table 4-3 Comparison Between the Performance of Logistic Regression and a Single Layered Neural Network..... 45
Table 4-4 Second Comparison Between the Performance of Logistic Regression and a Single Layered Neural Network..... 46

Figure List

Figures **Page**

Figure 1.1 Example of CART Tree 7
Figure 2.1 Neural Network's Goal – Example 1 18
Figure 2.2 Neural Network's Goal – Example 1 (Cont.) 19
Figure 2.3 Neural Network's Goal – Example 1 (Cont.) 19
Figure 2.4 Two-Layered Neural Network's Goal - Sample Population 25
Figure 2.5 Discriminants Found By Two-Layered Network 25
Figure 2.6 Regions Created by Neurons on First Hidden Layer 30
Figure 2.7 Example of Output from Two Neurons in First Hidden Layer 31
Figure 3.1 Simple Two-Layered Neural Network 38
Figure 5.1 Simple Death Data - Obvious Case 49
Figure 5.2 Risk from Logistic and Single Layer Neural Network 50
Figure 5.3 Risk from Neural Network 51
Figure 5.4 Simple Death Data – Blended Data 52
Figure 5.5 Risk from Neural Network 53
Figure 6.1 Two-Layered Network - Algorithm to Find Change-Point 55
Figure 6.2 Data Set 1 59
Figure 6.3 Change-point of Data Set 1 60
Figure 6.4 Data Set 2 61
Figure 6.5 Change-point of Data Set 2 62
Figure 6.6 Data Set 3 63
Figure 6.7 Change-point of Data Set 3 64
Figure 6.8 Data Set 4 65
Figure 6.9 Change-point of Data Set 4 66
Figure 6.10 Data Set 5 66
Figure 6.11 Change-point of Data Set 5 67
Figure 6.12 Data Set 6 68
Figure 6.13 Change-point of Data Set 6 69

1 INTRODUCTION

A significant barrier in modeling data sets is the existence of fundamental changes in the behavior of the data. For example, when modeling the probability of dying in an automobile accident based on age, the probability of dying in an automobile accident increases as the age of the person increases until the age of 23 when the probability decreases as age increases. Hence 23 would be the change-point of the data and it would be appropriate to use one model to describe the data when the age is less than 23 and another model to describe the data when the age is greater than 23. Our research focuses on the use of artificial neural networks and their associated logistic regression coefficients to determine the existence and the location of a change-point.

Two-layer backpropagating neural networks can be interpreted as finding various logistic regression equations in the first level and finding an appropriate balance between these in the second. Our intention is to associate sub-regions of the domain with neurons of the first layer of the neural network and thereby obtain a change-point for the population.

1.1 Literature Review

1.1.1 History of Artificial Neural Networks

The first step toward artificial neural networks came in 1943 when Warren McCulloch, a neurophysiologist, and a young mathematician, Walter Pitts, wrote a paper on how neurons

might work [Alderman, 2000]. They modeled a simple neural network with electrical circuits.

In 1949, Donald Hebb wrote the book "The Organization of Behavior" reinforcing the concept of neurons and how they work. It pointed out that neural pathways are strengthened each time that they are used.

As computer technology became more complex in the 1950s, it became possible to begin modeling the rudiments of the theories concerning human thought. Nathanial Rochester from the IBM research laboratories led the first effort to simulate a neural network. Although a first attempt failed, subsequent attempts were successful. It was during this time that traditional computing began to flower and, as it did, the emphasis in computing left the neural research in the background.

In 1956, the Dartmouth Summer Research Project on Artificial Intelligence provided a boost to both artificial intelligence and neural networks. One of the outcomes of this process was to stimulate research in both the intelligence side, AI, as it is known throughout the industry, and in the much lower level neural processing part of the brain.

In the years following the Dartmouth Project, John von Neumann suggested imitating simple neuron functions by using telegraph relays or vacuum tubes. Also, Frank Rosenblatt, a neuro-biologist of Cornell, began work on the Perceptron. He was intrigued with the operation of

the eye of a fly. Much of the processing which tells a fly to flee is done in its eye. The Perceptron, which resulted from this research, was built in hardware and is the oldest neural network still in use today [Fausett, 1994].

In 1959, Bernard Widrow and Marcian Hoff of Stanford developed models they called ADALINE and MADALINE. These models were named for their use of Multiple ADAptive LINEar Elements. MADALINE was the first neural network to be applied to a real world problem. It is an adaptive filter which eliminates echoes on phone lines. This neural network is still in commercial use.

John Hopfield of Caltech presented a paper to the national Academy of Sciences in 1982 where he stated that artificial neural networks were not used only to model brains, but to create useful devices. With clarity and mathematical analysis, he showed how such networks could work and what they could do.

By 1985 the American Institute of Physics began what has become an annual meeting - Neural Networks for Computing.

The 1990 US Department of Defense Small Business Innovation Research Program named 16 topics which specifically targeted neural networks with an additional 13 mentioning the possible use of neural networks.

Today, neural networks discussions are occurring everywhere. Their promise seems very bright as nature itself is the proof that this kind of thing works.

1.1.2 Cumulative Sum to Find the Change-point

A simple cumulative sum type has been often used to solve the change-point problem. A conditional test of no change against change is used and compared with a likelihood ratio test. The estimation of the change-point is also considered, using simple statistics, and the method is shown to be asymptotically equivalent to the maximum likelihood estimator in certain circumstances and almost equivalent in others.

Suppose that there is a sequence of independent zero-one random variables and that there is a change in the distribution at some unknown point. More precisely we suppose that $X_1, ...,X_T$ are independent random variables with

$$\left.\begin{array}{l} pr(X_i = 1) = 1 - pr(X_i = 0), \\ pr(X_i = 1) = \theta_0, (i \le \tau) \\ pr(X_i = 1) = \theta_1, (i > \tau) \end{array}\right\}, 1 \le \tau \le T,$$

The value of the parameter τ is known as the change-point. If $1 \le \tau \le T$, there has been a change in the sequence and if $\tau = T$ then there has been no change in the sequence [Hinkley, 1970].

1.1.3 Minimum Message Length Method to Find the Change-point

The Minimum Message Length (MML) principle is an invariant Bayesian point estimation technique based on information theory [Smith, 1975]. MML selects regions from the parameter space which contain models that can justify themselves with high posterior probability mass.

The MML method is especially useful when many change-points are being estimated and on large data sets.

Previous work on coding change-point parameters in the MML framework has resulted in analytical approximations which treat the change-point as a continuous parameter or avoid stating them altogether [Pettitt, 1979]. These approaches work well in practice. However, change-points are realized as discrete parameters since they partition a data sample.

1.1.4 Classification and Regression Trees

Classification and Regression Trees (CART) analysis is a form of binary recursive partitioning. At each node, a review of all independent variables is made and for each independent variable, a review of all values at which the population can be divided is conducted. The population is then split into two sub-nodes based on which independent variable and which splitting point for that independent variable will allow two separate logistic models operating on the two subpopulations to successfully predict the dependent variable over the entire population.

This procedure can be recursively continued until the population is divided into sufficient subpopulations to allow accurate modeling on all of them.

1.2 Relevance of the Research Presented in This Dissertation

CART uses logistic regression to subdivide a population along a grid where the grid consists of constant values for the independent variables. For example, if there are two risk factors of death, the following could be a tree obtained from CART:

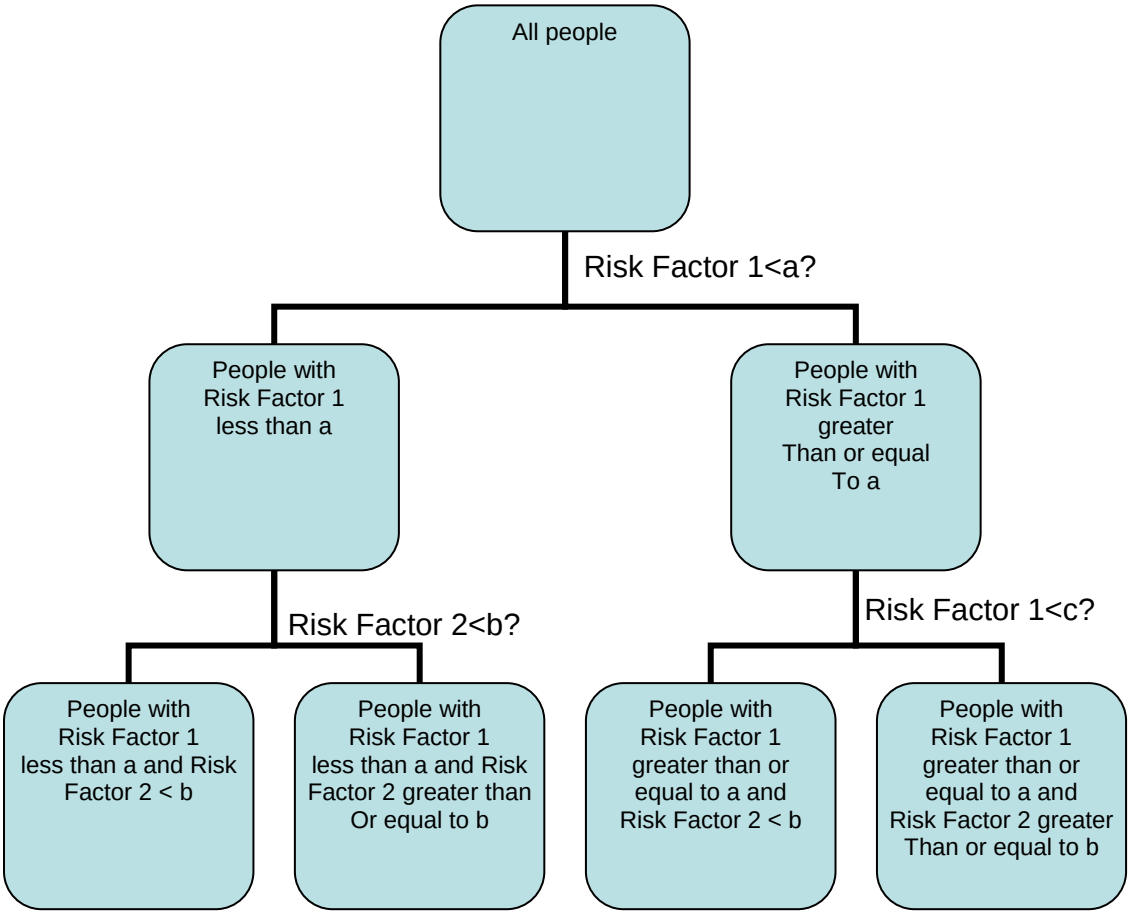


Figure 1.1 Example of CART Tree

The gridlines for the separation are obtained based on success of the logistic model in successfully dividing the population associated with each node of the tree into two subpopulations. It should be noted that risk factor 1<a is used in the first level of the tree and risk factor 1<c is used in the second level of the tree. The basis for each division is which independent variable and which value for that independent variable will provide two subpopulations which when modeled with two distinct logistic models will provide the best overall predictability for the population. In the tree above, if four separate logistic models are to be used on four subpopulations, the best results can be obtained through the divisions delineated by the bottom level of the tree obtained by CART

Each neuron in a neural network can use the same equation as is used in logistic regression. Our research has found that neural networks of two layers can also subdivide populations based on the success of the logistic function in predicting the dependent variable. We have found initial success with a new methodology which is very similar to CART in terms of a subdivision based on logistic equations but allows partitions that are non-constant hyperplanes. CART restricts subdivisions to constant values of independent variables.

1.3 Summary of Following Chapters

Chapter 1 gives an introduction to this work's objective and methodology. In addition, some of the research previously done is briefly discussed in the literature review section.

Then, chapter 2 gives a background on artificial networks definition, their purpose, functionality, different learning methods and a comparison between artificial neural networks and logistic regression. Also, examples of neural networks predicting output classes based on input characteristics are given in this chapter.

The artificial neural network's backpropagation algorithm is presented in Chapter 3.

Chapter 4 contains definitions, examples and experiments related to discriminants and their role in pattern classification using artificial neural networks.

The change-point problem methodology and experiments are discussed in chapter 5 and chapter 6, respectively.

Conclusions and future work are presented in chapter 7.

2 Theoretical Background

2.1 What is an Artificial Neural Network?

The following are some definitions of neural networks:

- An interconnected assembly of simple processing elements, units or nodes, whose functionality is loosely based on the animal brain. The processing ability of the network is stored in the inter-unit connection strengths, or weights, obtained by a process of adaptation to, or learning from, a set of training patterns. www.inproteomics.com/nwglosno.html
- A representation of a human brain similar in that both a brain and a neural network consist of input neurons gathering information from an external environment, synapses which interlace the input neurons' information in complex but fairly predictable patterns, and output neurons which turn the patterns of the synapses into actions made on the external environment. www.krl.caltech.edu/~charles/alife-game/glossary.html
- A type of statistical computer program which classifies large and complex data sets by grouping cases together in a way similar to the human brain. www.audiencedialogue.org/gloss-stats.html
- A computational method for optimizing for a desired property based on previous learning cycles (training). www.genpromag.com/Glossary~LETTER~N.html
- A machine-learning technique that simulates a network of communicating nerve cells. www.nature.com/nrg/journal/v5/n4/glossary/nrg1315_glossary.html
- A member of a class of software that is "trained" by presenting it examples of input and the corresponding desired output. For example, the input might be a magnetic anomaly and the required output the depth to the source of that anomally. www.geop.itu.edu.tr/~onur/seis/dic/gravmag.html

As can be seen from the variety of definitions, artificial neural networks are a varied and diverse set of information processing tools. To the author’s knowledge, there is no mathematical definition for what makes up the family of functions and processes that are

loosely labeled “neural networks.” To be classified as a neural network, there is generally an aspect of the process that replicates the behavior of biological neurons. They most frequently incorporate a “learning” or “training” process.

In this process, known data is used to adjust coefficients of neural networks and if the training process is successful, the neural network will then operate as a function capable of replicating the actual output for all of the known inputs. While these are common characteristics of neural networks, they are not definitions.

In our research, we use a feedforward backpropagating neural network. These do have a precise structure and a precise set of associated mathematical formulae. In sections 2.3 through 2.5, we will present the structure of feedforward backpropagating neural networks and the associated mathematical formulae. We will also present in sections 2.3 through 2.5 examples of input and output associated with feedforward backpropagating neural networks. In this section we simply wish to provide some background on neural networks.

In order to learn by example neural networks generally need to be configured for a specific application, such as pattern recognition or data classification through a learning process. Similarly to biological systems, artificial neural networks’ learning process involves adjustments to the synaptic connections (or weights) that exist between the neurons.

There are two major categories of artificial neural networks:

Fixed Networks in which the weights cannot be changed. In such networks, the weights are fixed according to the problem to solve.

Adaptive Networks which are able to change the weights within the neurons' connections. This type of network is used throughout this research thesis work.

The learning methods used for adaptive neural networks can be classified into two major categories:

Supervised learning which incorporates an external source of information, so that each output unit is told what its desired response to input signals ought to be. Paradigms of supervised learning include error-correction learning, reinforcement learning and stochastic learning.

An important issue concerning supervised learning is the problem of error convergence, or the minimization of error between the desired and computed unit values [Draper et al., 1998]. The aim is to determine a set of weights which minimizes the error.

Unsupervised learning uses no external source of information and is based upon only local information. It self-organizes data presented to the network and detects their common collective properties.

The behavior of an ANN (Artificial Neural Network) depends on both the weights and the activation function (transfer function) that is specified. The activation function is the most common way to replicate the activation of a biological neuron. This function typically falls into one of three categories: linear, threshold and sigmoid.

Linear: the output activity is proportional to the total weighted output.

Threshold: the output is set at one of two levels, depending on whether the total input is greater than or less than some threshold value.

Sigmoid: the output varies continuously but not linearly as the input changes. Sigmoid have a greater resemblance to real neurons than do linear or threshold units. They are useful in that they can approximate a threshold function while providing differentiability and other useful mathematical properties.

2.2 How Do Artificial Neural Networks Work?

Henceforth, when we use the term neural network we will be referring to a feedforward backpropagating neural network as these are the neural networks used in our experiments.

To make an artificial neural network perform a specific task, it should be decided how the units are connected to one another and the weights on the connections must be set

appropriately. These connections determine whether it is possible for one unit to influence another. The weights specify the strength of the influence.

For example, teaching a network to perform a particular task can be achieved by:

1. Presenting the network with training examples, which consist of a pattern of activities for the input units together with the desired or known output associated with this input. This is most often done by finding actual data where the output for given inputs is known . However, it can also be done by prescribing a desired set of outputs with a corresponding set of inputs or vice versa.
2. Determining how closely the actual output of the network matches the desired output.
3. Changing the weight of each connection so that the network produces a better approximation of the desired output.

As step 3 suggests, for a network to perform some task, we must adjust the weights of each unit in such a way that the error between the desired output and the actual output is reduced. This process requires that the neural network compute the error derivative of the weights. The backpropagation algorithm is the most widely used method for determining this derivative and the method used in this research work.

The algorithm computes each derivative by first computing the rate at which the error changes as the activity level of a unit is changed. For output units, this is simply the difference between the actual and the desired output. In multiple layer networks, to compute

this rate for a hidden unit in the layer just before the output, all the weights between that hidden unit and the output units need to be identified. Next, those weights are multiplied by the rate of error change of those output units and then, those products are added. This sum equals the rate at which the error changes for the chosen hidden unit. After calculating the rates in the hidden layer just before the output layer, the rate can be computed similarly for other layers, moving from layer to layer in a direction opposite to the way activities propagate through the network. The error derivative of the weights is the product of the rate at which the error changes and the activity through the incoming connection.

2.3 Single Layer Neural Networks

A single layer neural network has one layer of connection weights. Its input is frequently a set of characteristics for an object and its output is frequently whether the object associated with the input belongs to a particular class

In the typical single layer net, each output unit corresponds to a particular category to which an input vector may or may not belong. In addition, the weights of one output unit do not influence or affect the weights for other output units.

In general, a neural network's goal is to find a function which receives a vector as an input and returns a boolean value. The input vector's components are, generally, features or traits

and the function's output tells that the object belongs (or not) to a certain class of objects or endpoints.

The processing elements of neural networks are often sigmoid functions such as:

$$\sigma(x) = \frac{1}{1 + e^{-x}} \qquad (2.1)$$

in which the function's output approaches 1 if an object does belong to a certain class. Otherwise, the output approaches 0.

Neural networks are most often used with relatively clean data where their purpose is to determine classification or to describe data which can be easily and clearly classified into different sets. Graphically, the sets or populations are separated by a noticeable gap.

However, it is difficult to classify elements of different sets where the data is partially or completely mixed up. Graphically, no noticeable gap is shown between different sets of elements. Neural networks are often being used for "fuzzy data" sets as well where degree of belonging to a set is measured. This "fuzzy belonging" can be perceived as the probability that an item belongs to a certain set. When using neural networks with "fuzzy data", it is often desirable that the output from the neural network not be a boolean value but rather a number between 0 and 1 that indicates the degree of belonging to a set.

Example:

A possible goal for a single layer neural network would be to find a function that inputs a vector (x,y), where x and y are characteristics of a person and the output would be 0 if a person is sick and 1 if a person is healthy.

To accomplish this goal, first, we need to gather known data in order to train the neural network. Let us assume that we sample 8 people, obtain the values for *(x,y)* for these individuals and determine whether they are healthy or not. We will assume that we obtained the following data:

Healthy People (2.8, 1.8), (3.2, 3), (4.8, 4.2), (6.2, 5.1)

Sick People: (1, 4.7), (1.8, 6.3), (4.2, 7.6), (5, 8.9)

To visualize the goal of a single layer neural network, it helps to place these data on a grid:

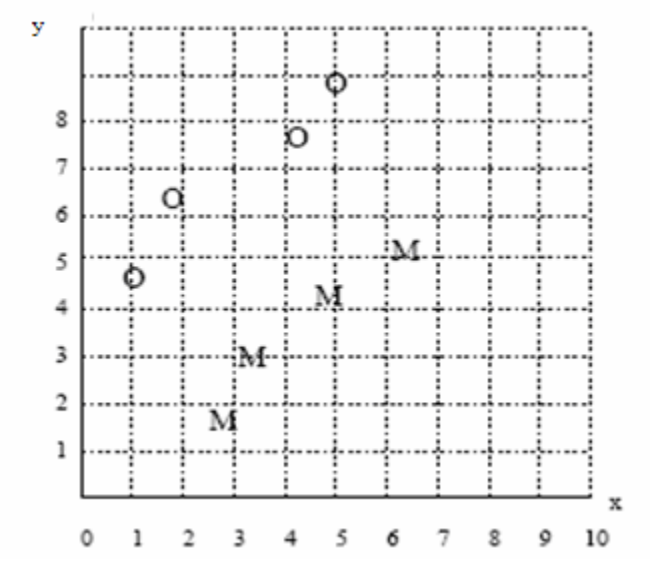


Figure 2.1 Neural Network's Goal – Example 1

M = healthy individual

O = not healthy individual

As the neural network processes the data set, it will divide the x-y plane into regions where the healthy individuals can be distinguished from the non-healthy individuals. The neural network will identify the different regions by calculating a discriminant, which is a boundary that helps to classify the data into different groups. In this case, there are many possible discriminants, however we will design a neural network which uses the line $x-y=-1$ to divide our population. Observing the graph below, it should be clear that this line will divide the plane into healthy and sick regions that are consistent with our 8 known data points.

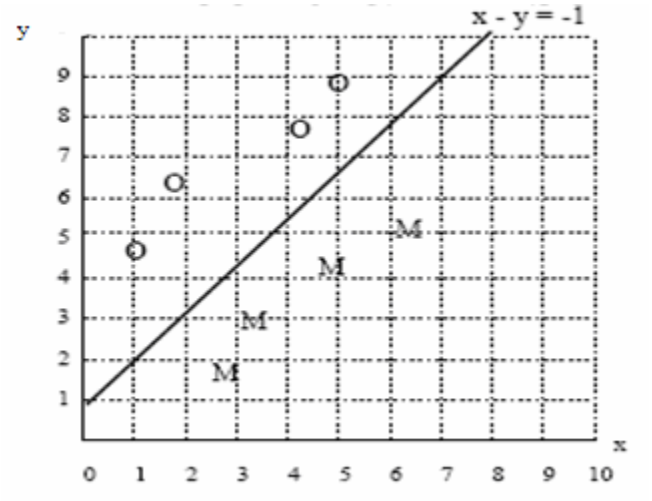


Figure 2.2 Neural Network's Goal – Example 1 (Cont.)

The following graph identifies which side of the discriminant is associated with healthy people. Our goal for the neural network we design will be to receive the characteristics *(x, y)* of a person and to return one if *(x,y)* are on the shaded side and zero if *(x,y)* are on the unshaded side.

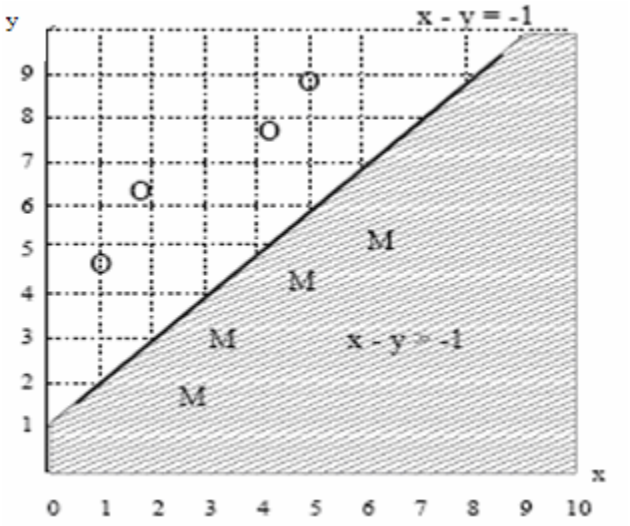
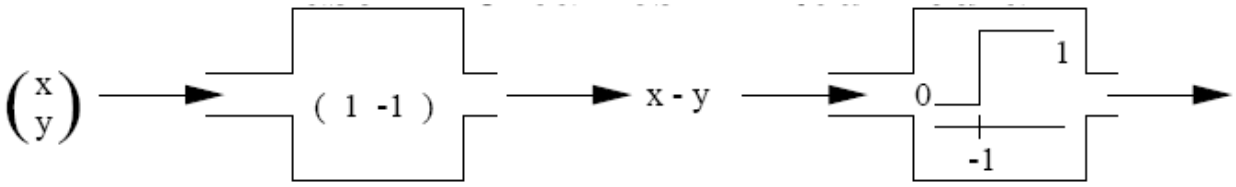


Figure 2.3 Neural Network's Goal – Example 1 (Cont.)

The following neural network will do precisely that.



(x,y) are inputted into the coefficient level and $(1\ -1)\begin{pmatrix} x \\ y \end{pmatrix} = x - y$ is output. This output $x - y$ is then sent to the activation function which is a threshold function that returns zero if the input is less than -1 and 1 if the input is greater than -1. Hence, the activation receives the input $x - y$ and returns 0 if $x - y < -1$ and returns one if $x - y > -1$. As this was the goal for our neural network, we have now designed a neural network consistent with the geometric goals we outlined.

It should be noted that for mathematical reasons, it is convenient that the activation function switches from 0 to 1 when the input is 0. Hence, the single layer neural networks that we use in our experiments would have input <x,y,1>. The row vector with which it would be multiplied would be (1 -1 1) and the output before the activation function would be *x-y+1*. This would be input into the activation function would activate when this quantity is greater

than zero. Thus, activation would occur when $x-y+1 > 0$ which is the same as $x-y > -1$ indicated in our above example.

Normally, healthy and sick people are intermingled and a clean division such as that presented in this example is impossible. In that case, an optimal discriminant is sought that will best allow prediction of the dependent variable based on the independent variables entered into the neural network.

2.4 Logistic Regression and Maximum Likelihood Equations

Logistic regression has the same intuitive goal as a single layer neural network. However, the formulation of the problem is very different. Assuming there is a dependent variable x and a dependent variable $y, 0 \le y \le 1$ and a set of known data points (x,y) associated with these data, the goal of logistic regression is to find the 'a' and 'b' values that best fit the known data points using the logistic equation:

$$y = \pi(x) = \frac{e^{-(ax+b)}}{1 + e^{-(ax+b)}} \qquad (2.2)$$

Logistic regression finds *'a'* and *'b'* using maximum likelihood equations. If (x_i, y_i) represents the i^{th} known data point, the *likelihood* of outcome y given data x is a conditional probability and it is given by the formula:

$$\text{Likelihood of } y_i \text{ given } x_i = \zeta(x_i, y_i) = \pi(x_i)^{y_i}[1-\pi(x_i)]^{1-y_i} \qquad (2.3)$$

Since the observations are assumed to be independent, the likelihood function is obtained as the product of the terms given in equation 2.3 over the n known data points. If we let $\beta = \begin{pmatrix} a \\ b \end{pmatrix}$, the likelihood function will depend on the parameters of β and is defined as

$$l(\beta) = \prod_{i=1}^{n} \zeta(x_i, y_i) \qquad (2.4)$$

The principle of *maximum likelihood* states that we use as our estimate of β the value that maximizes the expression in equation 2.4. However, it is mathematically easier to work with the log of equation 2.4 [Hosmer et al., 1989]. This expression, the *log likelihood*, is defined as:

$$L(\beta) = \ln[l(\beta)] = \sum_{i=1}^{n} \{y_i \ln[\pi(x_i)] + (1-y_i)\ln[1-\pi(x_i)]\} \qquad (2.5)$$

To find the value of β that maximizes $L(\beta)$ we differentiate it with respect to β_0 and β_1. Then, we set the resulting expressions equal to zero. These equations are as follows:

$$\sum_{i=1}^{n} [y_i - \pi(x_i)] = 0 \quad (2.6)$$

and

$$\sum_{i=1}^{n} x_i [y_i - \pi(x_i)] = 0. \quad (2.7)$$

On the other hand, the goal for the neural network is to find the values of 'a' and 'b' that best fit the data for equation 2.1 using the *steepest descent method*. That is:

$$a_{k+1} = a_k - \alpha \frac{\partial E}{\partial a} \qquad (2.8)$$

$$b_{k+1} = b_k - \alpha \frac{\partial E}{\partial b} \qquad (2.9)$$

where E= .5(real-output)². The coefficient α is a real, positive number between zero and one and is referred to as the learning coefficient. It may be changed after a course of iterations in order to reduce the error.

Then, another update is done to *a* and *b* values providing some smoothing, so that the impact of a given iteration would be balanced by the changes of previous iterations. This is done by using the "smoothing" parameter μ , which is a positive real number between zero and one. The results for *a* and *b* right after the *k+1* iteration would be:

$$a_{k+1} = a_k + \mu(-\alpha \frac{\partial E_k}{\partial a_k}) + (1-\mu)(-\alpha \frac{\partial E_{k-1}}{\partial a_{k-1}}) \qquad (2.10)$$

$$b_{k+1} = b_k + \mu(-\alpha \frac{\partial E_k}{\partial b_k}) + (1-\mu)(-\alpha \frac{\partial E_{k-1}}{\partial b_{k-1}}) \qquad (2.11)$$

The goal for logistic regression and a single layer neural network are precisely the same although their respective techniques differ. This fact can be observed, if we substitute

$$u = e^{-(\vec{a}\vec{x}+b)} \qquad (2.12)$$

into both logistic regression and sigmoid function equations.

2.5 Neural Networks with Two Hidden Layers

To understand how neural networks with two hidden layers work, let us consider the following situation where we have graphed mangoes and other fruit based on their characteristics *(x,y).* Our goal is to design a neural network that can distinguish between mangoes and other fruit.

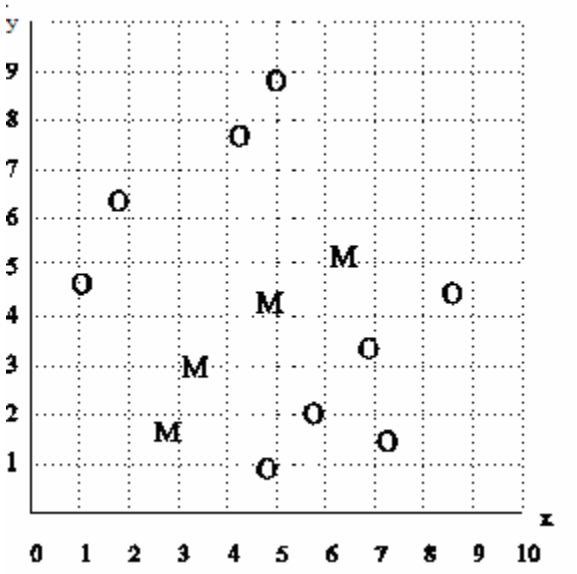


Figure 2.4 Two-Layered Neural Network's Goal - Sample Population

In this situation we can see that two lines are needed to separate the two distinct objects contained in the graph.

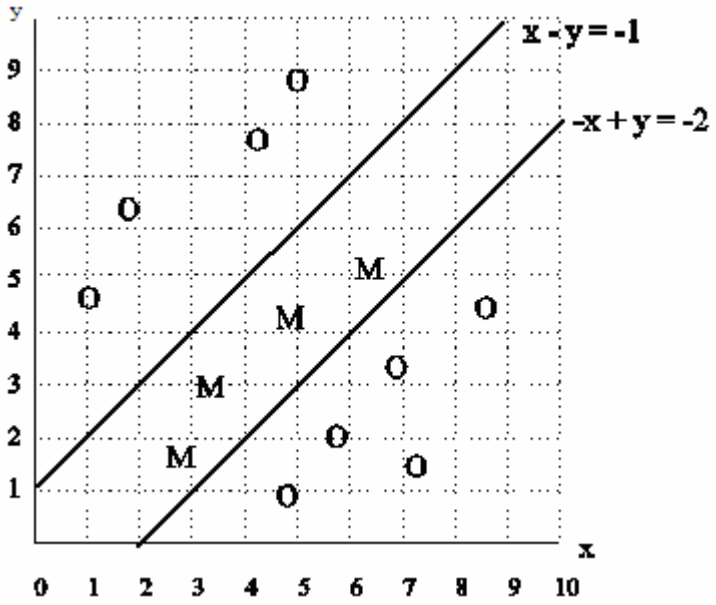


Figure 2.5 Discriminants Found By Two-Layered Network

Our goal is to design a function that will receive *(x,y)* and will return 1 if this *(x,y)* is in the mango region and will otherwise return 0. The following neural network will accomplish this goal:

$$(x,y,1)\rightarrow\left\{\begin{array}{l}\begin{pmatrix}1\\-1\\1\end{pmatrix}\rightarrow x+-y+1\xrightarrow{sigmoid}\delta_1\\ \\ \begin{pmatrix}-1\\1\\2\end{pmatrix}\rightarrow -x+y+2\xrightarrow{sigmoid}\delta_2\end{array}\right\}\rightarrow\ldots$$

First Hidden Layer

$$\ldots\rightarrow\begin{pmatrix}\delta_1\\ \delta_2\\ 1\end{pmatrix}\rightarrow\begin{pmatrix}2\\2\\-3\end{pmatrix}\rightarrow 2\delta_1+2\delta_2+-3\xrightarrow{sigmoid}z$$

Second Hidden Layer

$sigmoid \equiv \sigma(x) = \frac{1}{1+e^{-10000x}}$ which emulates a threshold function similar to that shown above except it is activated for input greater than zero and deactivated for input less than zero. To see how this neural network accomplishes our goal, we will enter a representative point from each region.

Leftmost Region:

Let us input *(x,y)=(1,3)* into the neural network. Remember that the output from our sigmoid is approximately zero for negative numbers and approximately 1 for positive numbers not close to zero. The following reflects the processing of the neural network:

$$(1,3,1) \rightarrow \left\{ \begin{array}{l} \begin{pmatrix} 1 \\ -1 \\ 1 \end{pmatrix} \rightarrow 1+-3+1 \xrightarrow{sigmoid} 0 \\ \begin{pmatrix} -1 \\ 1 \\ 2 \end{pmatrix} \rightarrow -1+3+2 \xrightarrow{sigmoid} 1 \end{array} \right\} \rightarrow \ldots$$

First Hidden Layer

$$\ldots \rightarrow \begin{pmatrix} 0 \\ 1 \\ 1 \end{pmatrix} \rightarrow \begin{pmatrix} 2 \\ 2 \\ -3 \end{pmatrix} \rightarrow 2*0+2*1+-3=-1 \xrightarrow{sigmoid} 0$$

Second Hidden Layer

Hence, the neural network would correctly identify that a point in the leftmost region should not be classified as a mango.

Central Region:

Let us input *(x,y)=(2,2)* into the neural network. The following reflects the processing of the neural network:

$$(2,2,1) \rightarrow \left\{ \begin{array}{l} \begin{pmatrix} 1 \\ -1 \\ 1 \end{pmatrix} \rightarrow 2+-2+1 \xrightarrow{sigmoid} 1 \\ \begin{pmatrix} -1 \\ 1 \\ 2 \end{pmatrix} \rightarrow -1+3+2 \xrightarrow{sigmoid} 1 \end{array} \right\} \rightarrow \ldots$$

First Hidden Layer

$$\ldots \rightarrow \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} \rightarrow \begin{pmatrix} 2 \\ 2 \\ -3 \end{pmatrix} \rightarrow 2*1+2*1+-3=1 \xrightarrow{sigmoid} 1$$

Second Hidden Layer

Hence, the neural network would correctly identify that a point in the central region should be classified as a mango.

Rightmost Region:

Let us input *(x,y)=(6,1)* into the neural network. The following reflects the processing of the neural network:

$$(6,1,1) \rightarrow \left\{ \begin{array}{l} \begin{pmatrix} -1 \\ 1 \\ -1 \end{pmatrix} \rightarrow -6+1+-1 \xrightarrow{sigmoid} 0 \\ \begin{pmatrix} -1 \\ 1 \\ 2 \end{pmatrix} \rightarrow -6+1+2 \xrightarrow{sigmoid} 0 \end{array} \right\} \rightarrow \ldots$$

First Hidden Layer

$$\ldots \rightarrow \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \rightarrow \begin{pmatrix} 2 \\ 2 \\ -3 \end{pmatrix} \rightarrow 2*0+2*0+-3=-3 \xrightarrow{sigmoid} 0$$

Second Hidden Layer

Hence, the neural network would correctly identify that a point in the rightmost region should not be classified as a mango.

Generalization:

In general, the first hidden layer will produce two expressions of the form $ax+by+c$. These are then passed into the activation function which outputs 0 for (x,y) on one side of the line and outputs 1 for points on the other side of the line. These two outputs from the first hidden layer (we often refer to them as output from the two neurons of the first hidden layer) will then be passed into the second hidden layer. The second hidden layer will use a linear combination of the outputs from the 2 neurons in the first hidden layer to determine if both neurons of the first hidden layer are activated, one of the first layer neurons is activated, or none of the first layer neurons is activated. By selecting the coefficients of the second hidden layer well, the neural network can select which scenarios will trigger the activation function associated with the second hidden layer. In this way, the second layer can classify an object based on which neurons of the first layer are activated or not activated.

2.6 Double Layer Neural Networks and the Change-point

2.6.1 Interpretation of the First Hidden Layer

As we have seen in the previous subsection, the first hidden layer of the neural network will produce 2 lines. In the previous section, the two lines were parallel producing three regions. However, in general, two neurons in the first hidden layer will produce four regions in the xy plane (see the example below).

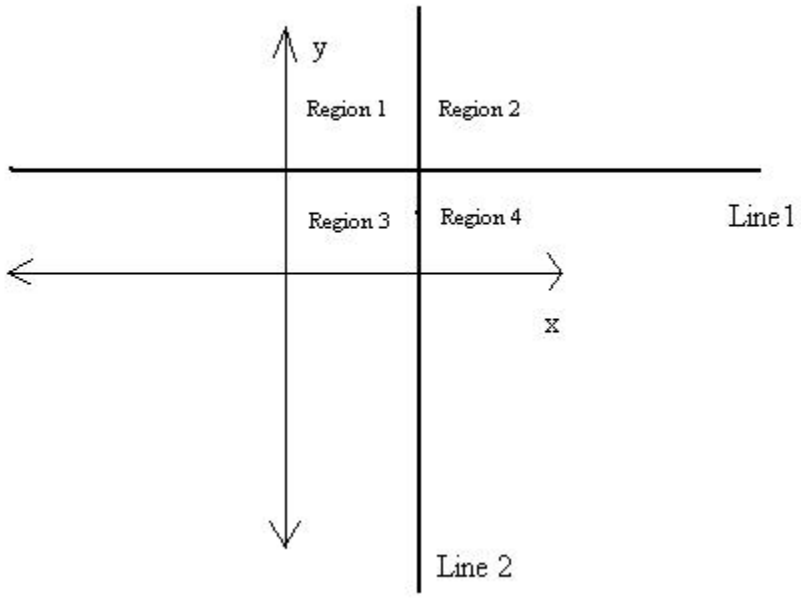


Figure 2.6 Regions Created by Neurons on First Hidden Layer

We consider δ_1 and δ_2 to be the outputs from the two neurons in the first hidden layer. If the activation function associated with the neuron is a threshold function or a very steep sigmoid, δ_1 and δ_2 will be approximately equal to zero on one side of the neuron associated line and approximately equal to one on the other side. If Line 1 is associated with δ_1 and line 2 is associated with δ_2, the output (δ_1, δ_2) will be distributed as follows:

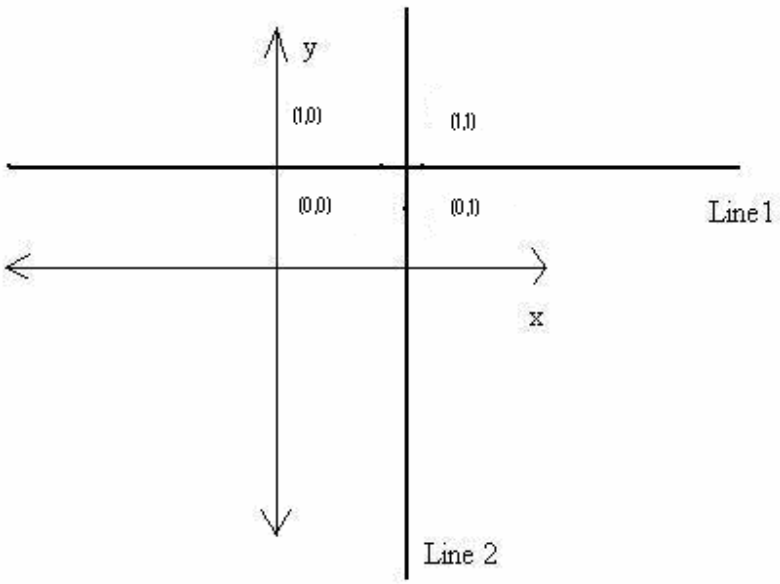


Figure 2.7 Example of Output from Two Neurons in First Hidden Layer

If the activation function associated with the neuron is a moderately inclined sigmoid, δ_1 and δ_2 will be less than 0.5 on one side of the neuron associated line and greater than 0.5 on the other side. The general trend noted in the above diagram will remain the same, however the values will gradually rise from 0 to 1 instead of jumping immediately upon reaching the neuron associated line.

2.6.2 Interpretation of the Second Hidden Layer

The second layer of the neural network will accept as inputs the values of δ_1 and δ_2 that were calculated within the first layer and that were used to divide the x-y plane into four regions.

Now, the network will assign weights (b_{1j}) to the values of δ_1 and δ_2 . The linear combination of $b_{11}\delta_1 + b_{12}\delta_2 + b_{13}$ is then input into the activation function associated with the second hidden layer.

We then divide all *(x,y)* into two classes: those *(x,y)* where the first neuron contributes more to the final output of the second activation function. I.e., $|b_{11}\delta_1| > |b_{12}\delta_2|$ and those points where the second neuron contributes more to the second activation function, $|b_{11}\delta_1| < |b_{12}\delta_2|$. The border between these two regions where $|b_{11}\delta_1| = |b_{12}\delta_2|$ will be a candidate to define where the nature of the population undergoes a fundamental change.

If one neuron has a greater impact than the other for the entire domain, then this technique will not divide the population into two parts. This may mean that there is no significant change-point in the population or that another technique to find the change-point is necessary. When the activation function is a threshold function, the diagram shows that the set of points where $|b_{11}\delta_1| = |b_{12}\delta_2|$ can form a region. However, our research uses sigmoid functions of the

form $\sigma(x) = \frac{1}{1 + e^{-const*x}}, const \neq 0$. In this case, unless $b_{11} = 0$ and $b_{12} = 0$ the set of points where $|b_{11}\delta_1| = |b_{12}\delta_2|$ will not form a region but a curve.

It should also be noted that our experiments are done with moderately inclined sigmoids so that the values of δ_1 and δ_2 gradually rise from zero to one and are not the threshold portrayed in parts of this section.

2.7 Notes on Obtaining the Coefficients

This section did not show how to find the coefficients that will make a neural network function. It simply showed how a neural network will correctly work if the coefficients associated with its neurons are well chosen. In chapter 4, we will demonstrate how the coefficients that will make a neural network function may be obtained.

3 Multi-layer Neural Networks and the Backpropagation Algorithm

3.1 Overview of the Training Process

In the previous section, we showed that a neural network receives as input certain characteristics of an object and outputs the whether an object belongs to a certain class. In

the examples of the previous sections, the inputs were characteristics of fruits and the output was whether the fruit was a mango. It is worth mentioning that if the output is not boolean, it can be used to represent the probability that an object belongs to a class.

To accomplish this assessment, the input is passed through hidden layers of the neural network that contain coefficients and activation functions. If these coefficients and activation functions are well chosen and there are sufficient neurons and hidden layers in the neural network, the expectation is that the neural network will output an accurate probability that the object associated with the given inputs belongs to the class associated with the output.

To obtain an overview of how this is done, let us consider a neural network that receives two inputs (x,y) and a single output z . The inputs (x,y) represent characteristics of an object. The output z is equal to zero if the object associated with the input is not a member of the output class and is equal to one if the object associated with the input is a member. Output between 0 and 1 is considered to represent probability that an object belongs to the output class. We will assume that the neural network has two hidden layers with coefficients a_{ij} in the first hidden layer and b_j in the second hidden layer. Hence, the purpose of the training process is to find the optimal set of values of a_{ij} in the first hidden layer and b_j in the second hidden layer that will allow the neural network to receive characteristics (x,y) and determine whether they represent a member of the class associated with the output z. A precise outline of the mathematics will follow in the next section. This section is simply to provide an overview of the training algorithm.

Let's suppose that we have two data points with which to train our neural network:

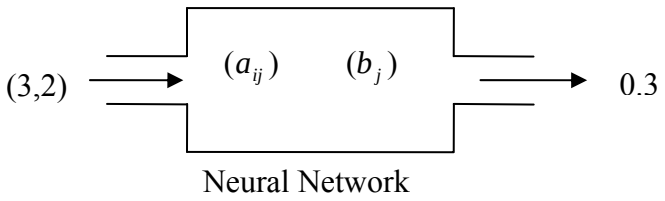
(x,y) = (3,2) for an object not in the class. I.e., *(3,2,0)* is the data point.

(x,y) = (5,3) for an object in the class. I.e., *(5,3,1)* is the data point.

Our goal is to train the neural network to output 0 when the input is *(3,2)* and to output 1 when the input is (5,3). As there are no data for other points, the output with other inputs will not be a factor in our training process. In order to train our neural network we perform the following steps.

Step 1: Initialize a_{ij} and b_j to random values between 0 and 1.

Step 2: Enter the first data point (3,2,0) and obtain the output from the neural network (in this case we assume that the output from the neural network is 0.3 while the actual output should be 0):



Step 3. Find the partial derivatives of the error with respect to all hidden coefficients.

If P = Desired Output, then the error (using the *l-2* norm) $E = \frac{1}{2}(P - z)^2$ = $\frac{1}{2}(0 - 0.3)^2 = 0.045$

The mathematics will be presented in the following section. However, we will now assume that the calculations for $\frac{\partial E}{\partial a_{ij}}$ and $\frac{\partial E}{\partial b_j}$ for all coefficients in the first and second hidden layers of the net have been obtained.

Step 4: Update the coefficients using a steepest descent approach so that each coefficient is slightly altered to diminish the size of the error.

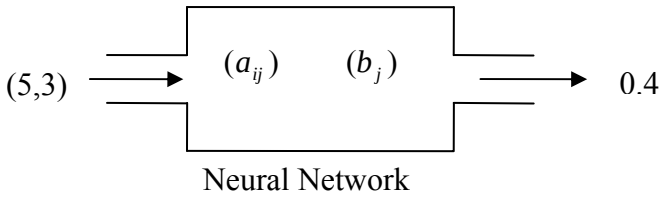
$$a_{ij} = a_{ij} - \alpha \frac{\partial E}{\partial a_{ij}}$$

$$b_j = b_j - \alpha \frac{\partial E}{\partial b_j}$$

The coefficient α is positive and is referred to as the learning coefficient. We generally started with $\alpha = .05$ and reduced its value over the course of our iterations.

Step 5: Repeats steps 2 through 4 with each of the remaining data points in the training set. In our case there is only one more training point.

Step 2 – Second Iteration



Step 3 – Second Iteration

Find the partial derivatives of the error with respect to all hidden coefficients

P = Desired Output, then the error $E = \frac{1}{2}(P - z)^2 = \frac{1}{2}(1 - 0.4)^2 = 0.18$

.

Step 4 – Second Iteration

Update the coefficients.

$$a_{ij} = a_{ij} - \alpha \frac{\partial E}{\partial a_{ij}}$$

$$b_j = b_j - \alpha \frac{\partial E}{\partial b_j}$$

Step 6: Repeat steps 2 through 5 until the Error is ~0 for all data points or until the sum of the errors over all data points ceases to diminish with successive iterations.

It is worth noting that we used some smoothing features when updating parameters so that the impact of each given update was smoothed with previous updates. However, the general format for our training algorithm was as indicated in this section.

3.2 Mathematical Formulation for Updating Coefficients

The following is the diagram for the simple two-layer neural network with sigmoidal activation functions, two inputs and one output that we used for our experiments with multi-layer neural networks.

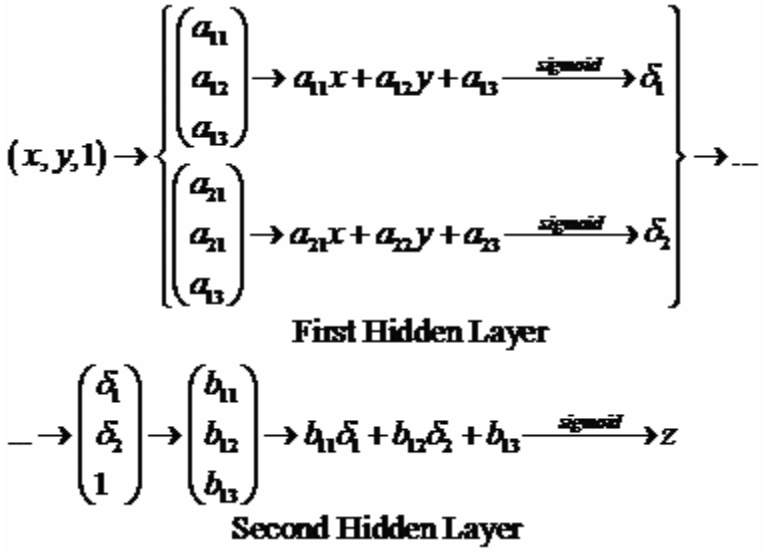


Figure 3.1 Simple Two-Layered Neural Network

(x,y) = Input,
z = Output ,
P = Desired Output,

$$E = \frac{1}{2}(P - z)^2 \text{ and}$$

$$sigmoid \equiv \sigma(x) = \frac{1}{1 + e^{-x}}$$

For multi layered networks, we must propagate the error back from the output node. The algorithm starts moving from layer to layer in a direction opposite to the way activities propagate through the network (from input to output).

Using the chain rule, we can expand the error of a hidden unit in terms of its posterior nodes:

1) Solving for $\frac{\partial E}{\partial b_{1j}}$

$$\frac{\partial E}{\partial b_{1j}} = \frac{\partial E}{\partial z}\frac{\partial z}{\partial b_{1j}}$$

$$E = \frac{1}{2}\left(P - z\right)^2 \rightarrow \frac{\partial E}{\partial z} = (P - z)(-1)$$

$$z = \sigma(b_{11}\delta_1 + b_{12}\delta_2 + b_{13} * 1) \rightarrow \begin{array}{l} \frac{\partial z}{\partial b_{1j}} = \sigma'(b_{11}\delta_1 + b_{12}\delta_2 + b_{13}) * (\delta_j),\ j = 1,2 \\ \frac{\partial z}{\partial b_{1j}} = \sigma'(b_{11}\delta_1 + b_{12}\delta_2 + b_{13}) * (1),\ j = 3 \end{array}$$

The error derivative of the weights is the product of the rate at which the error changes and the activity through the incoming connection.

2) Solving for $\frac{\partial E_k}{\partial a_{ij}}$,

$$\frac{\partial E}{\partial a_{ij}} = \frac{\partial E}{\partial \delta_i}\frac{\partial \delta_i}{\partial a_{ij}}$$

$$\frac{\partial E}{\partial \delta_i} = \frac{\partial E}{\partial z}\frac{\partial z}{\partial \delta_i}$$

$$E = \frac{1}{2}\left(P - z\right)^2 \rightarrow \frac{\partial E}{\partial z} = (P - z)(-1)$$

$$z = \sigma(b_{11}\delta_1 + b_{12}\delta_2 + b_{13} * 1) \rightarrow$$

$$\frac{\partial z}{\partial \delta_i} = \sigma'(b_{11}\delta_1 + b_{12}\delta_2 + b_{13}) * (b_{1i}),$$

$$\delta_i = \sigma(a_{i1}x + a_{i2}y + a_{i3}) \rightarrow \begin{cases} \frac{\partial \delta_i}{\partial a_{ij}} = \sigma'(a_{i1}x + a_{i2}y + a_{i3}) * (x),\ j = 1 \\ \frac{\partial \delta_i}{\partial a_{ij}} = \sigma'(a_{i1}x + a_{i2}y + a_{i3}) * (y),\ j = 2 \\ \frac{\partial \delta_i}{\partial a_{ij}} = \sigma'(a_{i1}x + a_{i2}y + a_{i3}) * (1),\ j = 3 \end{cases}$$

With the derivative of the error with respect to all of the hidden nodes of the neural network calculated, the training algorithm defined in the previous subsection of this chapter was used to continue updating coefficients until convergence at an optimal set of values for the coefficients of the neural network was achieved.

3.3 Applying Neural Networks

In a previous chapter, we demonstrated how backpropagating neural networks can be used to identify objects. In this chapter we demonstrated how to train neural networks to identify

objects. Normally, if we were going to use neural networks of this kind, we would first train them with known data and then apply the trained neural network with fixed coefficients to unknown data.

Our application however is a bit different. We are not interested in the capability of neural networks to classify data. In the process of training a neural network, coefficients will be obtained for all of the hidden layers. The outputs from the first hidden layer are $\delta_i = \sigma(a_{i1}x + a_{i2}y + a_{i3})$, i = 1,2 and the output from the second hidden layer is $z = \sigma(b_{11}\delta_1 + b_{12}\delta_2 + b_{13} * 1)$. We refer to $\delta_i = \sigma(a_{i1}x + a_{i2}y + a_{i3})$ as neurons. After training a neural network, each point *(x,y)* in the domain will activate one of the two neurons more strongly than the other with respect to its effect on the output *z*. This is determined by observing whether $b_{11}\delta_1$ or $b_{12}\delta_2$ is greater for a given point *(x,y)*. The reasoning behind this conclusion is that $z = \sigma(b_{11}\delta_1 + b_{12}\delta_2 + b_{13} * 1)$. Hence if $b_{11}\delta_1$ is greater than $b_{12}\delta_2$ then the first neuron is effecting the output *z* more strongly than the second and we will associate that data point *(x,y)* with the first neuron. Alternatively, if $b_{11}\delta_1$ is less than $b_{12}\delta_2$ then the first neuron is affecting the output *z* less strongly than the second and we will associate that data point *(x,y)* with the second neuron. In this way, each point of the domain will be associated with a neuron of the neural network and the domain can be divided into sub-regions based on associations with neurons of the trained neural network.

Our goal is to see if these divisions provide insight into the change-point problem.

4 Single Layer Artificial Neural Network and Logistic Regression

4.1 CART, Logistic Regression and Single Layer Neural Networks

The CART Method uses the logistic model to divide populations into two subpopulations by dividing the population along a constant value of one of the independent variables. Our goal in this section is to verify that our single layer neural networks will achieve the same goals as logistic models. It should be noted, that a single layer neural network is far less efficient than logistic regression in achieving the same goal. However, the neurons of a single layer neural network can be incorporated into a double layer neural network and for that reason they interest us.

Artificial neural networks are algorithms that can be used to perform nonlinear statistical modeling and provide a new alternative to logistic regression, the most commonly used method for developing predictive models.

Compared to logistic regression, artificial neural networks require less formal statistical training. They have the ability to implicitly detect complex nonlinear relationships between

dependent and independent variables and to detect all possible interactions between predictor variables. In addition, multiple algorithms can be used with artificial neural networks.

Despite the differences between the two, both methods share the same goal, as it was discussed in chapter 2.

Table 4-1 Logistic Regression vs. Artificial Neural Networks

Logistic Regression	Artificial Neural Networks
$\hat{\pi}(x) = \frac{\exp(\beta \bullet x)}{1 + \exp(\beta \bullet x)}$	$\hat{\pi}(x) = \frac{1}{1 + \exp((-)\beta \bullet x)}$

If we multiply the logistic regression equation by $\frac{\exp(-\beta \bullet x)}{\exp(-\beta \bullet x)}$, the formulas are the same. Thus, analyzing the same population of data with these two methods, should yield the same results.

Table 4-2 Logistic Regression vs. Artificial Neural Networks (Cont.)

Logistic Regression	Neural Networks
$l(\beta) = \prod_{i=1}^{n} (\pi(\hat{x}))^{y_i} (1 - \pi(\hat{x}_i))^{1-y_i}$	$E(\beta) = (y_i - \pi(x_i))^2$
$\ln(l(\beta)) = \sum_{i=1}^{n} \{y_i \ln(\pi(\hat{x}_i)) + (1 - y_i) \ln(1 - \pi(\hat{x}_i))\}$	Obtain $\nabla E(\beta)$.
$\nabla \ln(l(\beta)) = \sum_{i=1}^{n} \hat{x}_i (y_i - \pi(\hat{x}_i))$	$\beta_{new} = \beta - \lambda \nabla E(\beta)$, λ = small updating coefficient
$\nabla \ln(l(\beta)) = 0$	Repeat until convergence with β

4.2 Experiment Using Logistic Regression to Obtain Learning Parameters for Neural Networks

4.2.1 Methodology

The objective of the experiment was to use the fact that logistic regression is a much more precise solution for coefficients of 1 level neural networks in order to obtain parameters that will optimize the performance of a single layer neural network programmed in C code. We will then use these as starting parameters for our neural networks of two layers.

In this experiment, one hundred data pairs (x_i, y_i) were generated. The independent variable x was uniformly distributed from zero to six. The dependent variable y had probability of P(y=1|x) equal to 0 for $x<3$ and P(y=1|x) equal to .5+(x/12) for $x \geq 3$.

This procedure was repeated five times to generate five data sets with 100 elements in each one. Logistic regression was performed on each data set to obtain the optimal values of a and b. these values were then compared with neural networks that used a variety of different learning parameters.

4.2.2 Results

Table 4-3 shows that the results obtained by logistic regression where the maximum likelihood equations were used to find a and b in $y = \frac{e^{-(ax+b)}}{1 + e^{-(ax+b)}}$ and neural networks

with parameters of $\alpha = .5$ and $\mu = 1$ (learning and smoothing parameters; see chapter 2, equations 2.8, 2.9, 2.10, 2,11).were used with 100,000 repetitions of the training algorithm to find the same parameters. As was expected, the results were very similar.

Table 4-3 Comparison Between the Performance of Logistic Regression and a Single Layered Neural Network.

Coefficients 'a' and 'b' Found Using Logistic Regression	Coefficients 'a' and 'b' Found Using Neural Networks
0.772733,0.162439	0.738468,0.156479
0.676728,0.076428	0.670548,0.079042
0.687248,0.021124	0.681357,0.021162
0.806840,0.090810	0.806483,0.090825
0.657486,0.002503	0.643200,0.010918

Table 4-4 shows that the results obtained by logistic regression where the maximum likelihood equations were used to find a and b in $y = \frac{e^{-(ax+b)}}{1 + e^{-(ax+b)}}$ and neural networks with the initial value of $\alpha = .1$ which then was reduced by 10 percent for every 10000 iterations. A constant value of $\mu = .8$ was used. This was then run until no discernible difference was detected between successive iterations on the entire data set. With this methodology, the results were almost identical.

Table 4-4 Second Comparison Between the Performance of Logistic Regression and a Single Layered Neural Network.

Coefficients 'a' and 'b' Found Using Logistic Regression	Coefficients 'a' and 'b' Found Using Neural Networks
0.772733,0.162439	0.7727314,0.162521
0.676728,0.076428	0.676032,0.076512
0.687248,0.021124	0.687427,0.021436
0.806840,0.090810	0.806941,0.090803
0.657486,0.002503	0.657512,0.002494

4.2.3 Conclusion

In conclusion, the neurons of a neural network do in fact replicate the work of logistic regression, if the training of the neural networks is well done. Logistic regression will obtain these coefficients much more efficiently and precisely. However, while not efficient, these neurons will replicate the performance of logistic regression. CART consists of separate logistic regression models along a grid of subdivisions of the population. Hence, when we proceed with two-layer neural networks, the neurons of the first layer will replicate a layer of logistic regression models.

It remains to be seen whether the second level will be able to use the first level to find a change-point.

5 The Change-point Problem and Discriminants

5.1 Introduction to Discriminants

A discriminant is a boundary of a mathematical model that allows us to classify data into groups (e.g. healthy vs. not healthy). 'Discriminant' simply means that it has the ability to discriminate between two classes or sets of data [Fausett, 1994]. There are linear and non-linear discriminants.

Linear discriminants are widely used today in many application domains, including the modeling of various types of biological data. This type of discriminants can be obtained using artificial networks with supervised learning, which is used very frequently in traditional statistics and computer sciences. Also, they are widely used currently in many application domains, including the modeling of various types of biological or medical data.

For example, if we analyze a dataset of people and each single datum represents a person, each person (datum) has characteristics that can determine whether that person is healthy or not. That would be:

Person_1 = {characteristic_1, characteristic_2, …, characteristic_N}

Applying simple classification rules to the Person_1 datum could be done by comparing each characteristic with a particular value in the artificial neural network training set.

If characteristic_1 < pre-set value for characteristic_1 in training set and

characteristic_2 < pre-set value for characteristic_2 in training set and

.

.

.

characteristic_N< pre-set value for characteristic_N in training set

then, Person_1 is healthy.

Alternatively, Person_1 is not healthy.

The limitation of CART is that it requires that the discriminant occur at constant values for independent variables as was indicated in the above example.

Realistically, classification problems usually have too many characteristics that affect each single datum for so simple a scheme to accurately achieve a change-point. A more realistic approach is to use characteristics in order to find a hyper-plane such that:

If datum (Person_1) lies one side of the hyperplane, then the person is healthy.

If datum (Person_1) lies on the other side of the hyperplane, then the person is not healthy.

5.2 Preliminary Experiment - Discriminants with Logistic Regression, One-Layer Neural Networks and Two-Layer Neural Networks

To capture the shortcomings of Logistic Regression and neural networks with one hidden layer, a simple set of data with two risk factors. If either risk factor 1 or risk factor 2 is greater than 4, then the risk of death is equal to 1. Otherwise the risk is equal to 0. This is a case where the risk and not at risk groups are clearly separated and hence a clear cut and well defined discriminant exists. The question is what techniques will find it.

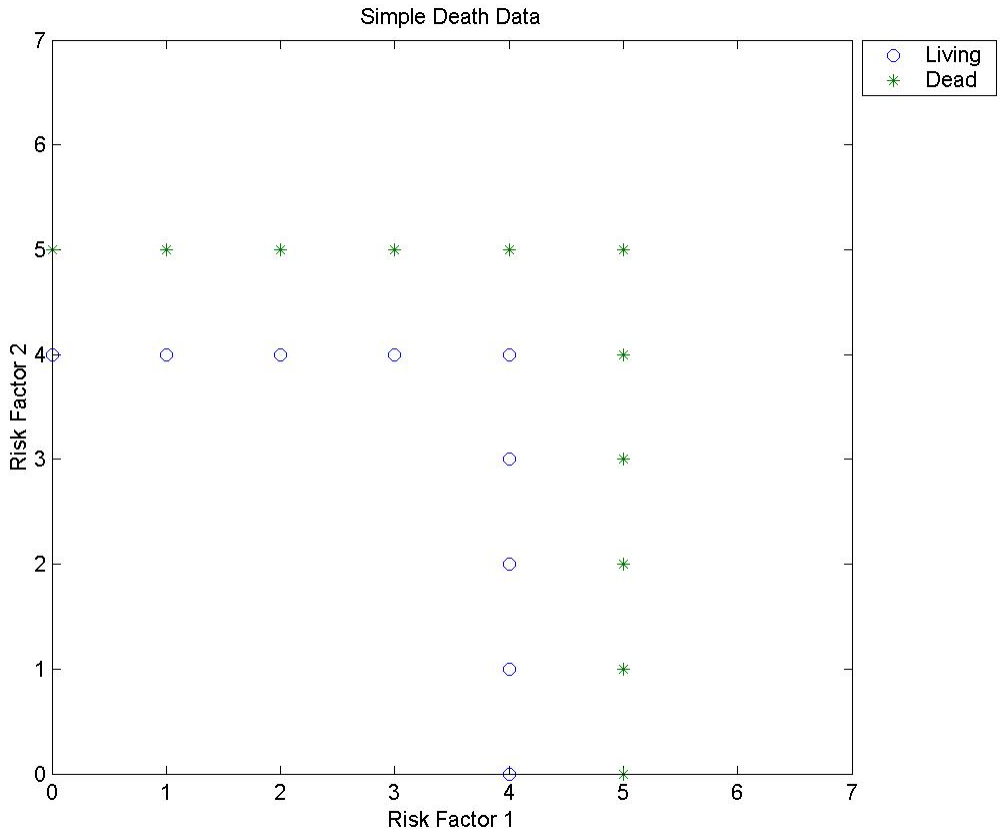
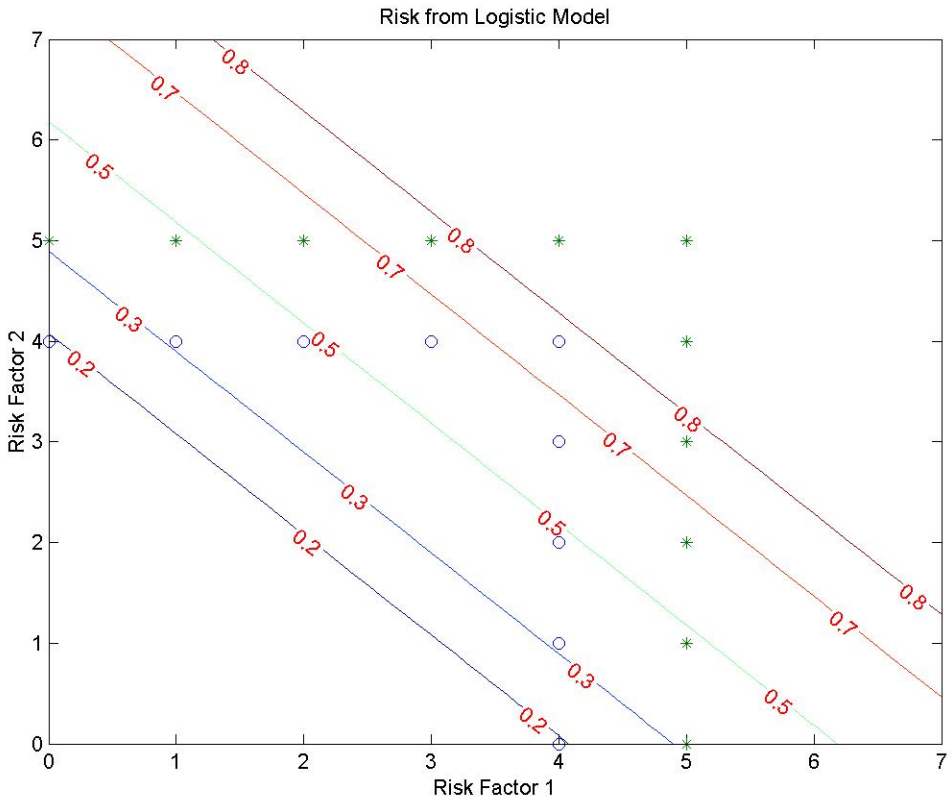


Figure 5.1 Simple Death Data - Obvious Case



.

Figure 5.2 Risk from Logistic and Single Layer Neural Network

Figure 5.2 presents the contour lines of constant risk for single layer neural networks and logistic regression. A single-layer neural network and logistic regression produced the same results, hence only one is presented. If we consider a risk of 50 percent as the discriminant that separates the two populations, it is clear that neither a single layer neural network nor logistic regression have captured the nature of the risk.

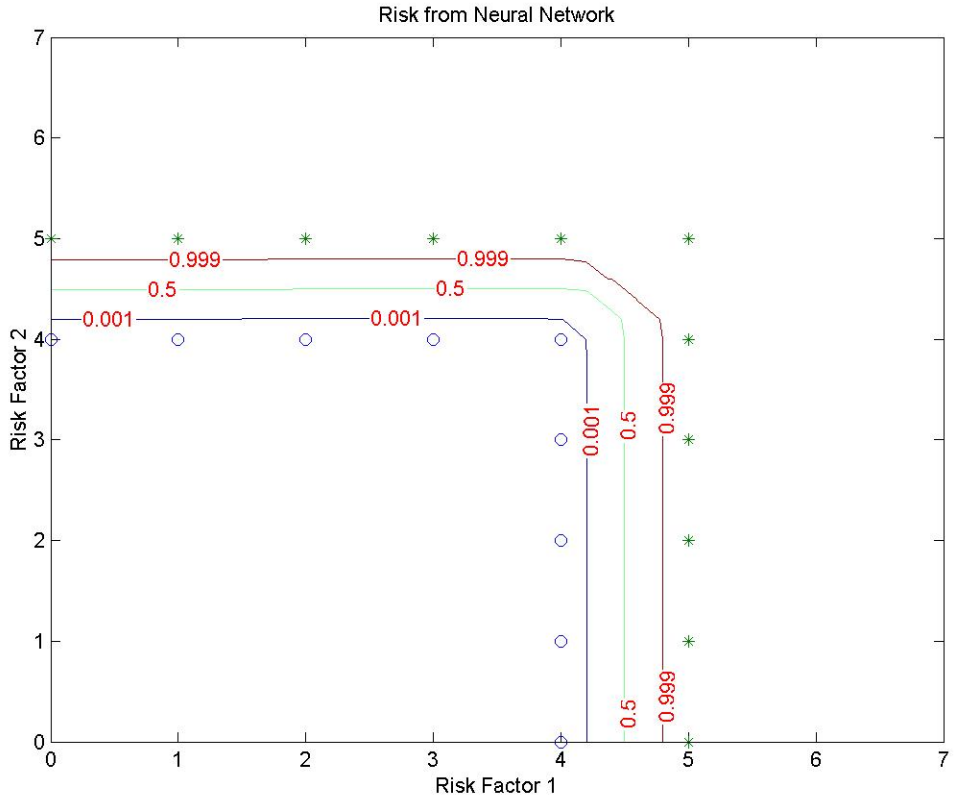


Figure 5.3 Risk from Neural Network

Figure 5.3 presents the contour lines of constant risk for a double layer neural network.

In contrast to logistic regression behavior and a single layer neural network, the double layer neural network adapts itself to the population's nature. The contours show how the net outputs the region in which an individual has 50% of living or dying. Obviously, it is providing a model that fits the data far better than the logistic regression's model.

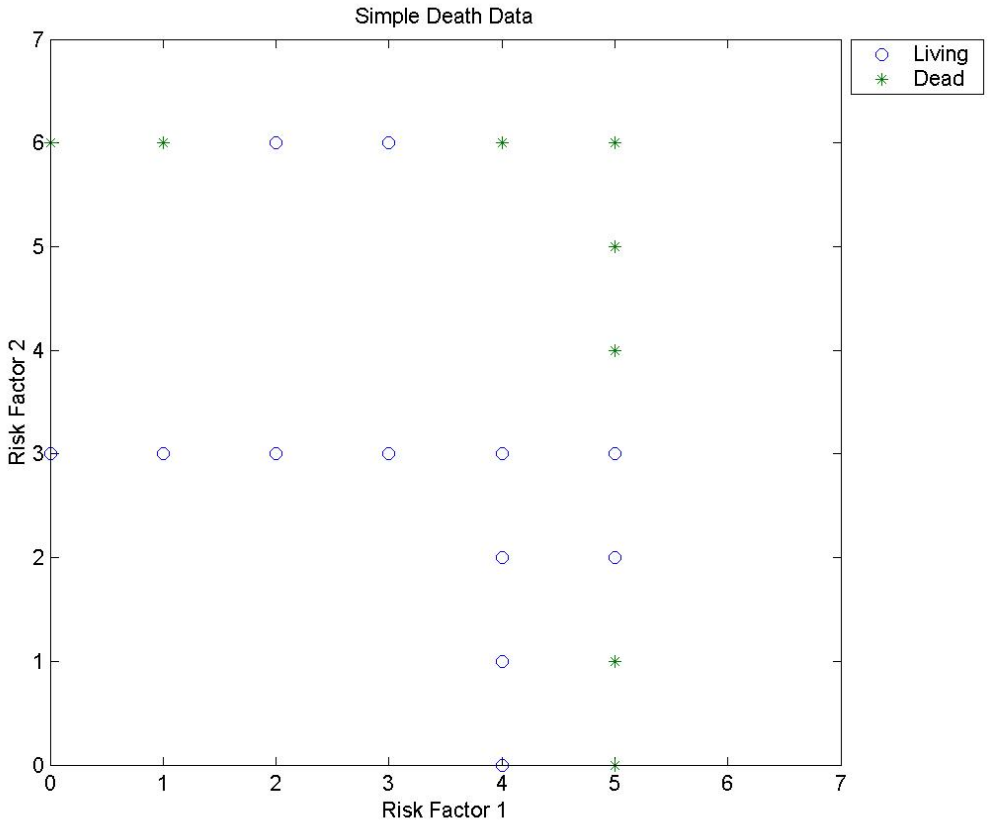


Figure 5.4 Simple Death Data – Blended Data

As it is unrealistic to expect data so clearly separated, Figure 5.4 represents a population that has the same general propensities as the last data set. However, the individuals that will survive are blended with the individuals that will die due to the risk factors.

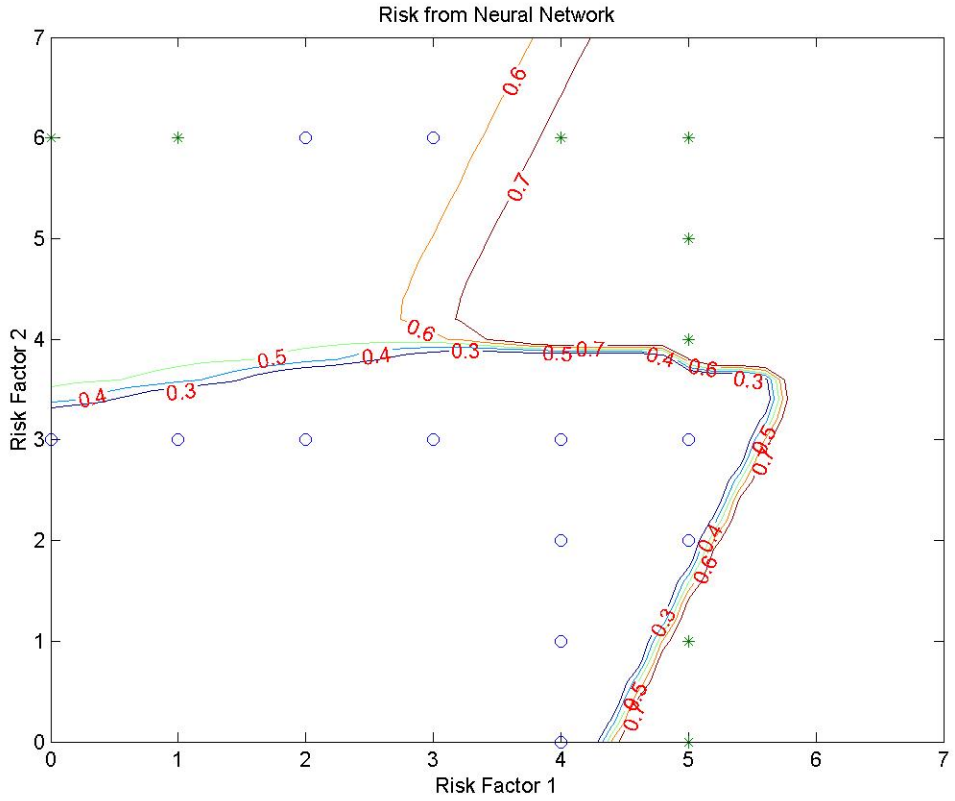


Figure 5.5 Risk from Neural Network

Figure 5.5 shows the contours of constant risk that a double layer neural network produced with this blended death data. It can be seen in this case that that data was not blended enough and correspondingly, the neural network found a different discriminant for the population. It is hence worth noting that the artificial neural network reacts to each datum as it processes it. In this case, the net is detecting a discriminant at risk factor 1= 4.5 (line that crosses risk factor 1 axis at 4.5) and also, it is detecting a discriminant at risk factor 2 = 4.5 (line that crosses risk factor 2 axis at 4.5). In addition, it is reacting to the data by treating as outliers those surviving individuals that are blended with those that will die. Hence, sparse data can

be very problematic with neural networks as it will react strongly to outliers. As sparse data is a problem with any modeling technique, this is not surprising but worthy of attention.

6 The Change-point Problem – New Research with Fuzzy Data

6.1 Introduction

When seeking to model a population, it is often found that there are in fact two distinct populations with distinct behaviors. Trying to model such populations with a single model is difficult; therefore, we need to separate those populations and model them separately.

Because of this, the change-point problem consists of finding a transfer point in which the population undergoes a significant change. Once the change-point has been identified, the two subpopulations can be modeled separately with whatever modeling scheme is deemed appropriate.

The most common techniques for obtaining change-points are sequential searches or Monte Carlo sampling techniques based on a maximization of the maximum likelihood equations [James et al., 1987].

6.2 Methodology

6.2.1 Algorithm to Find the Change-point.

To find the change-point a simple two-layer neural network with input *(x,y)* and output *z* was created with two "neurons" in the first layer and one "neuron" in the second layer.

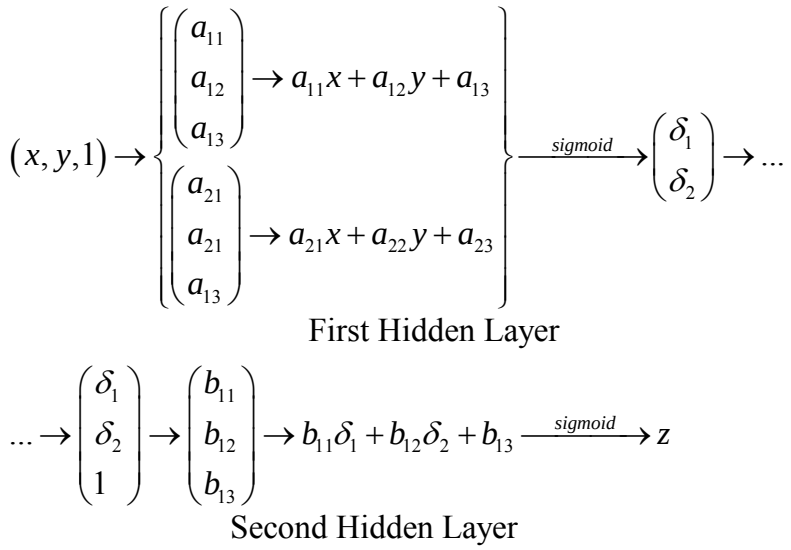


Figure 6.1 Two-Layered Network - Algorithm to Find Change-Point

The experiment was to determine if the coefficients associated with the hidden layers of the neural network would contain the information where a fundamental change results in the data set. Our hypothesis is that the population can be divided into two subpopulations: The first is the set of *(x,y)* where the magnitude of $b_{11}\delta_1$ is greater than the magnitude of $b_{12}\delta_2$ and the second subpopulation is the set of *(x,y)* where the magnitude of $b_{11}\delta_1$ is less than the magnitude of $b_{12}\delta_2$. The change-point will be the boundary of these two populations where

the magnitude of $b_{11}\delta_1$ is equal to the magnitude of $b_{12}\delta_2$. However, if one neuron has a greater impact than the other for the entire domain, then this technique will not divide the population into two parts. Probably, this means that there is no significant change-point in the population or that another technique is needed to find the change-point.

6.2.2 Data Sets to be Tested

For this preliminary set of experiments we created functions $z = f(x,y)$ above the region $(x, y) \in [0,4] \times [0,4]$.

The functions f are all continuous and fuzzy however they all have a well defined change-point where the nature of the function changes.

Data set 1: The function $z=f(x,y)$ is defined by

$$z = \begin{cases} .25(x+y) & (x+y) \le 4 \\ 2-.25(x+y) & (x+y) > 4 \end{cases}, \quad (x,y) \in [0,4] \times [0,4]$$

Data set 2: The function $z=f(x,y)$ is defined by

$$z = \begin{Bmatrix} 0.1(x+y) & (x+y) \le 4 \\ \dfrac{3(x+y)-4}{20} & (x+y) > 4 \end{Bmatrix},$$

Data set 3: The function $z=f(x,y)$ is defined by

$$z = \begin{Bmatrix} .5(x+y)\_ & (x+y) \le 2 \\ \dfrac{8-(x+y)}{6} & (x+y) > 2 \end{Bmatrix},$$
$$(x,y) \in [0,4] \times [0,4]$$

Data set 4: The function $z=f(x,y)$ is defined by

$$z = \begin{Bmatrix} 0.2(x+y) & (x+y) \le 2 \\ \dfrac{(x+y)+2}{10} & (x+y) > 2 \end{Bmatrix},$$
$$(x,y) \in [0,4] \times [0,4]$$

Data set 5: The function $z=f(x,y)$ is defined by

$$z = \left\{ \begin{array}{ll} \dfrac{d1}{d1+d2} & (x+y) \le 2 \\ \dfrac{d3}{d3+d2} & (x+y) > 2 \end{array} \right\},$$
$$(x,y) \in [0,4] \times [0,4] \text{ where}$$

d1= distance from *(x,y)* to the line *y = -2x*

d2= distance from *(x,y)* to the line *x+y = 2*

d3= distance from *(x,y)* to the line *y = -0.5x+6.*

Data set 6: The function *z=f(x,y)* is defined by

$$z = \left\{ \begin{array}{ll} \dfrac{0.4*d1}{d1+d2} & (x+y) \le 2 \\ \dfrac{0.4*d3+d2}{d3+d2} & (x+y) > 2 \end{array} \right\},$$
$$(x,y) \in [0,4] \times [0,4] \text{ where}$$

d1= distance from *(x,y)* to the line *y = -2x*

d2= distance from *(x,y)* to the line *x+y = 2*

d3= distance from *(x,y)* to the line *y = -0.5x+6.*

By comparing the actual change-point where the neural network switches from one primary activation function to another primary activation function, we can determine the capability of neural networks to obtain change-points.

From each of these six datasets, 10,000 points in the region $(x, y) \in [0,4] \times [0,4]$ were generated and used to train the neural network. In order to model these datasets, the neural network may switch from one primary activation function to another primary activation function. By observing where these changes in the primary activation function occur, we hope to obtain insight into the change-point problem.

6.3 Results

6.3.1 Data Set 1

The visualization of data set one is as follows:

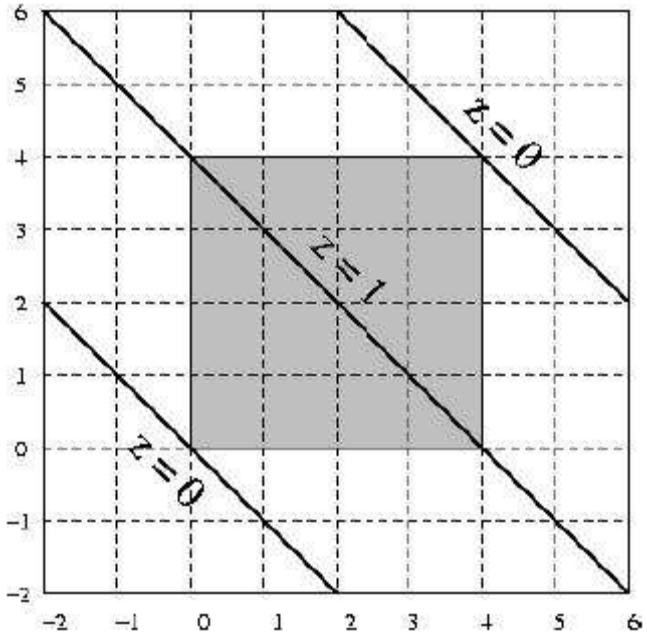


Figure 6.2 Data Set 1

The following diagram indicates where the neural networks switched from activation function 1 to activation function 2. The shaded region indicates where ∂_1 is the strongest indicator of z and the white region is where ∂_2 is the strongest indicator of z.

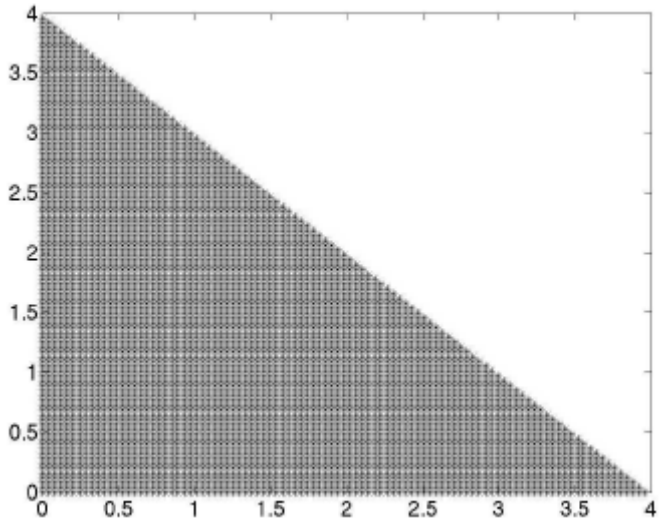


Figure 6.3 Change-point of Data Set 1

It is clear from observing the two populations that the neural network clearly identifies the change-point.

6.3.2 Data Set 2

The visualization of data set two is as follows:

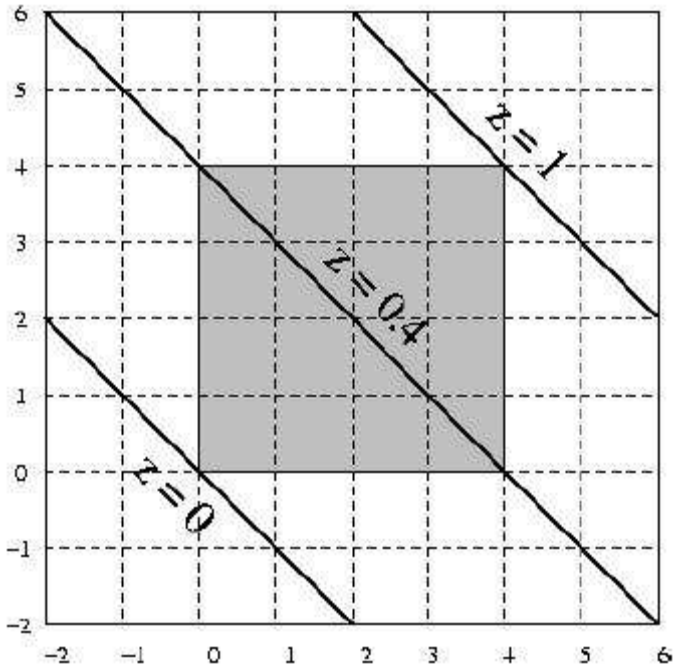


Figure 6.4 Data Set 2

The following diagram indicates where the neural networks switched from activation function 1 to activation function 2. The shaded region indicates where ∂_1 is the strongest indicator of z and the white region is where ∂_2 is the strongest indicator of z.

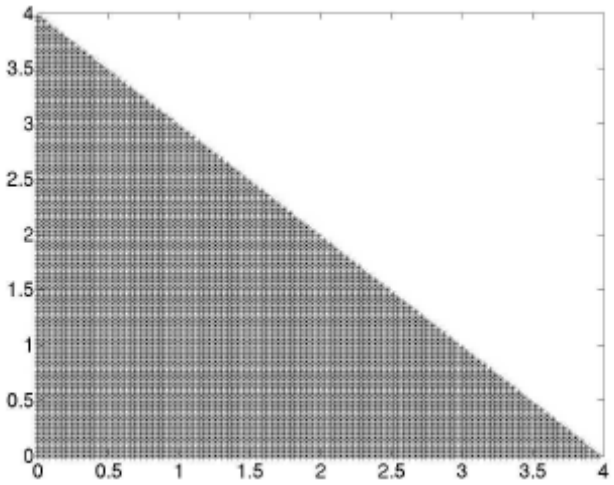


Figure 6.5 Change-point of Data Set 2

It is clear from observing the two populations that the neural network clearly identifies the change-point.

6.3.3 Data Set 3

The visualization of data set three is as follows:

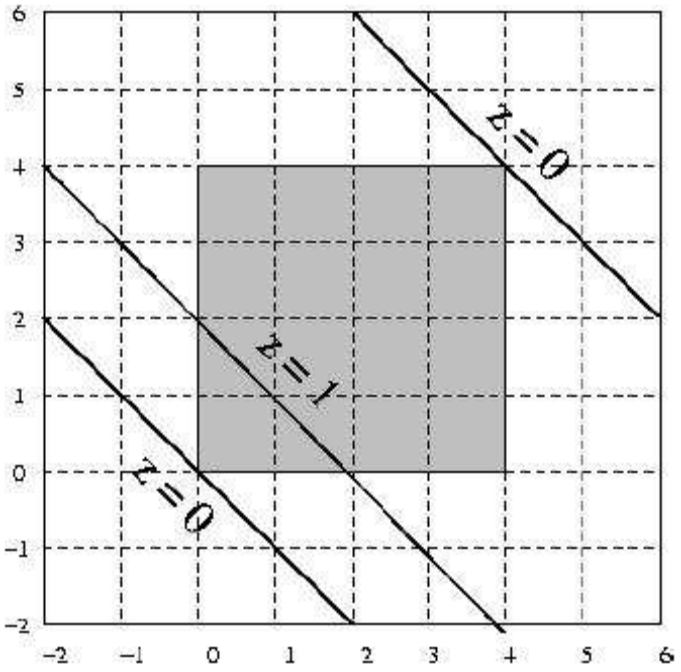


Figure 6.6 Data Set 3

The following diagram indicates where the neural networks switched from activation function 1 to activation function 2. The shaded region indicates where ∂_1 is the strongest indicator of z and the white region is where ∂_2 is the strongest indicator of z.

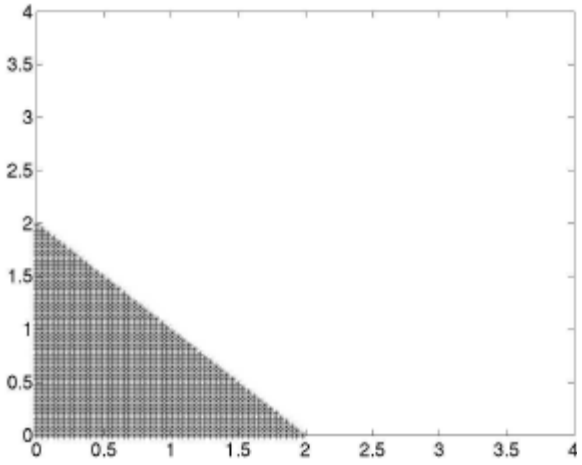


Figure 6.7 Change-point of Data Set 3

It is clear from observing the two populations that the neural network clearly identifies the change-point.

6.3.4 Data Set 4

The visualization of data set four is as follows:

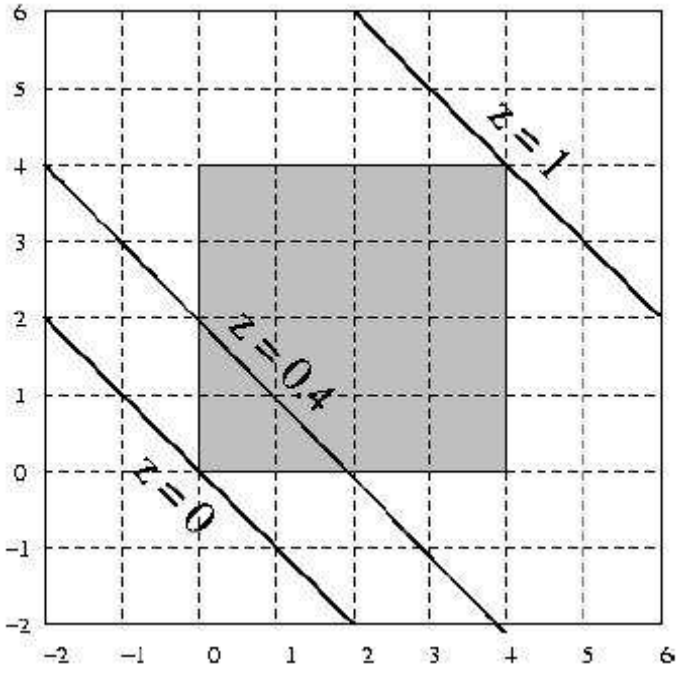


Figure 6.8 Data Set 4

The following diagram indicates where the neural networks switched from activation function 1 to activation function 2. The shaded region indicates where ∂_1 is the strongest indicator of z and the white region is where ∂_2 is the strongest indicator of z.

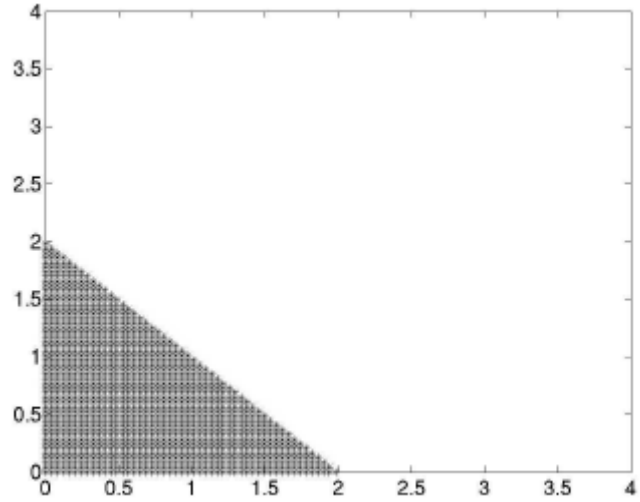


Figure 6.9 Change-point of Data Set 4

It is clear from observing the two populations that the neural network clearly identifies the change-point.

6.3.5 Data Set 5

The visualization of data set five is as follows:

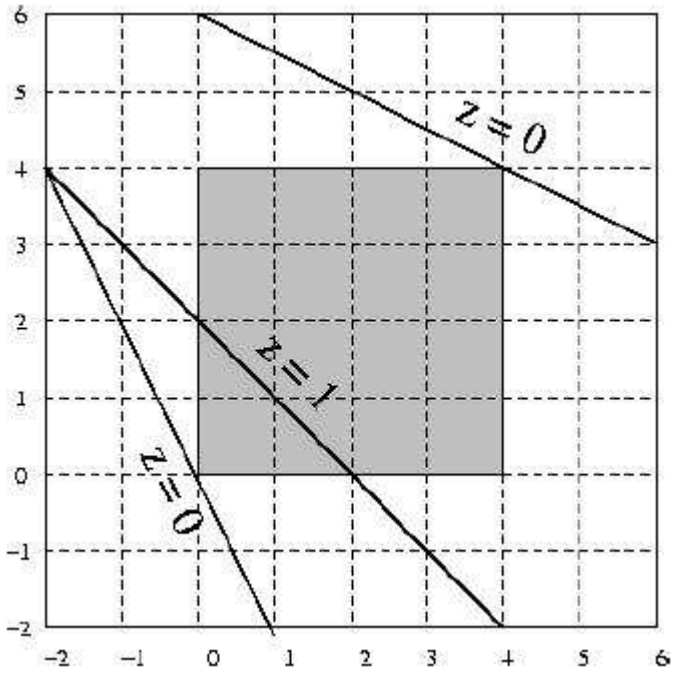


Figure 6.10 Data Set 5

The following diagram indicates where the neural networks switched from activation function 1 to activation function 2. The shaded region indicates where ∂_1 is the strongest indicator of z and the white region is where ∂_2 is the strongest indicator of z.

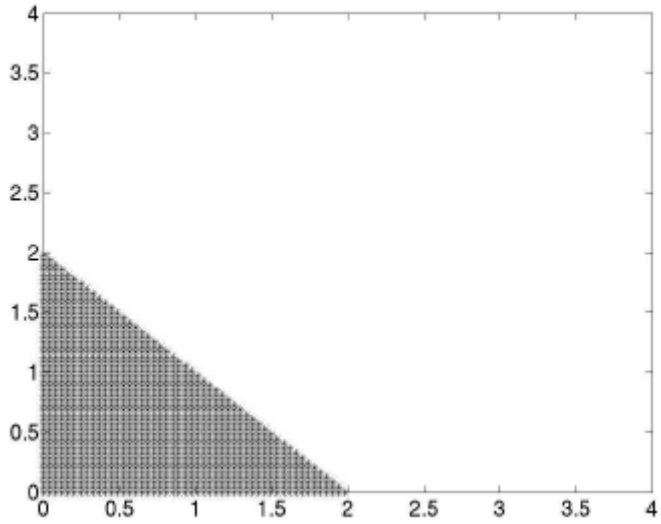


Figure 6.11 Change-point of Data Set 5

It is clear from observing the two populations that the neural network clearly identifies the change-point.

6.3.6 Data Set 6

The visualization of data set six is as follows:

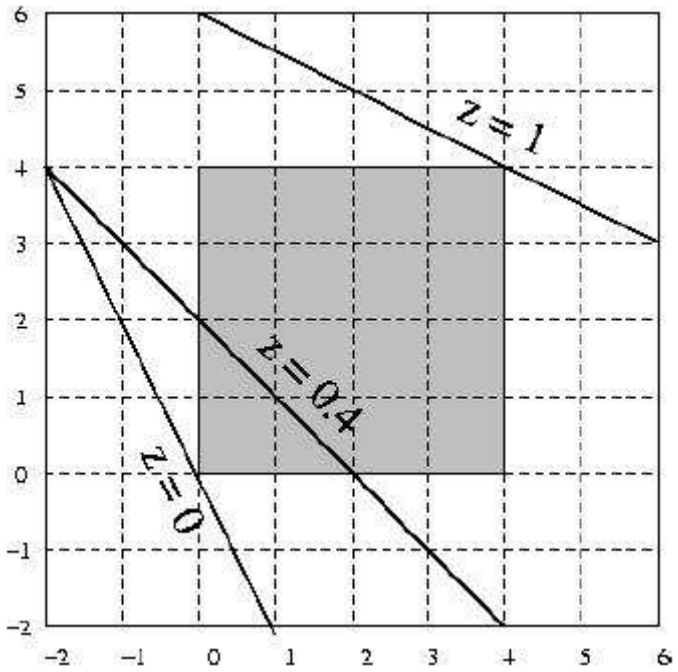


Figure 6.12 Data Set 6

The following diagram indicates where the neural networks switched from activation function 1 to activation function 2. The shaded region indicates where ∂_1 is the strongest indicator of *z* and the white region is where ∂_2 is the strongest indicator of *z*.

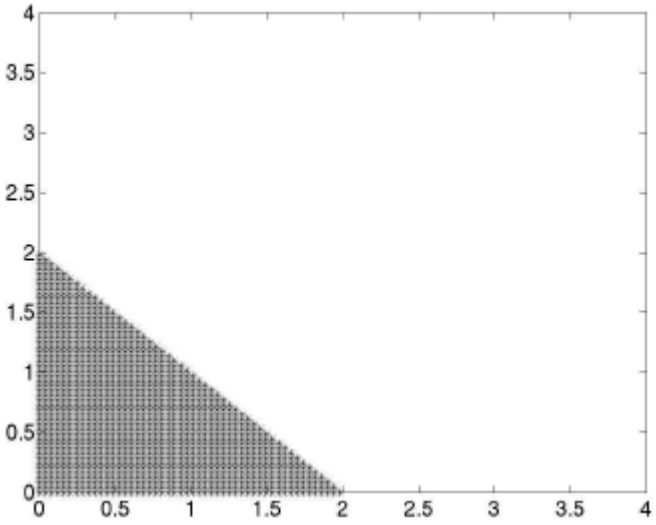


Figure 6.13 Change-point of Data Set 6

It is clear from observing the two populations that the neural network clearly identifies the change-point.

6.3.7 Conclusion and Summary

For all of these data sets, the neural networks captured the change-point of the population precisely and without difficulty.

The change-points of the data sets were not complicated. However, it is an important first step in our research to have verified the capabilities of neural networks to find change-points in a simple setting.

7 Conclusions and Future Work

Simulations with two variables have verified the effectiveness of backpropagating neural networks to quickly approximate change-points in simple simulations involving 3 variables.

The artificial neural network does quite well finding the change-point with a wide variety of simulated populations. However, the network is very sensitive as it reacts to each and every individual data point that may be outlier. Hence, sparse data can be problematic.

The neurons of neural networks may provide clues as to how to stratify populations as a function of multiple variables. If an ideal stratification can be found, then logistic regression can be applied to each subpopulation. Thus, we may be able to achieve the versatility of neural networks coupled with the precision and interpretability of logistic regression. CART achieves this, however, only along a grid.

Each neuron in a neural network can use the same equation as is used in CART. But, CART restricts subdivisions to constant values of independent variables. Our initial research has found that neural networks of two layers can also subdivide populations based on the success of the logistic function in a manner similar to CART. However non constant partitions can be obtained with our method.

Future work with our new change-point algorithm will be to use the same two level neural networks with more challenging change-points. After this, the following step would be to expand to three level neural networks and discontinuous change-points using the same basic principle that where the neural network switches from one "neuron" to another can be linked to a change-point of a population.

REFERENCES

Alderman, Michael H., "Measures and Meanings of Blood Pressure", *The Lancet*, Jan 15, 2000

Canneb, P.L., "A Simulation Study of One- and Two-Sample Kolmogorov-Smirnov Statistics with a Particular Weight Function", *Journal of American Statistics Association*, vol. 70, pp. 209-211.

Carlin, B.P., Gelfand, A.E. and Smith, A.F.M., "Hierarchical Bayesian Analysis of Change Point Problems", *Applied Statistics*, vol. 41, pp. 389-405, 1992

Cobb, G. W., "The Problem of the Nile: Conditional Solution to the Change-Point Problem", *Biometrika,* vol. 65, pp. 243-51, 1978

Draper, N.R. and Smith, H., "Applied Regression Analysis", John Wiley and Sons, 1998

Durazo-Arvizu R., McGee D., Li Z, and Cooper R., "Index-Mortality Relationship: A Case Study"; *Journal of the American Statistical Association*, Vol. 92, No.440, 1997

Fausett, L., "Fundamentals of Neural Networks", 1994

Hinkley, D.V., "Inference about the Change Point from Cumulative Sum Tests", *Biometrika,* vol. 53, pp. 509-523, 1970

Hosmer, D.W. and Lemeshow, S., "Applied Logistic Regression", 1989

James, B., James, K.L. and Siegmund, D., "Test for a Change Point", *Biometrika*, vol. 74, pp. 71-83, 1987

Moreno E., Casella G. and Garcia-Ferrer, A., "An Objective Bayesian Analysis of the Change Point Problem", 2004

Müller, H.G., "Change-Points in Nonparametric Regression Analysis", *Annals of Statistics*, vol. 20, pp. 737-761, 1992

Pettitt A. N., "A Non-Parametric Approach to the Change Point Problem", *Applied Statistics*. vol. 28, pp. 126-135, 1979

Smith, A. F. M., “A Bayesian Approach to Inference About a Change Point in a Sequence of Random Variables”, *Biometrika*, vol. 62, pp. 407-416, 1975