ABSTRACT

COMPUTER AND INFORMATION SCIENCE

CRUM, MELVIN L.   B.S. MOREHOUSE COLLEGE, 1992

PREDICTING TENDENCIES: A NEURAL NETWORK APPROACH

Advisor: Dr. R. Srikanth

Thesis dated July, 1995

This study examines the use of Neural Networks to predict tendencies of a football team's offense based on plays executed on third downs. The examinations were processed by using neural networks to recognize patterns by training the networks with sample plays.

Several different architectures for neural network models are explored to determine how well patterns can be recognized utilizing one hidden layer and varying nodes at the layer. This problem is also studied using a Bayes Classifier.

PREDICTING TENDENCIES:

A NEURAL NETWORK APPROACH

A THESIS

SUBMITTED TO THE FACULTY OF CLARK ATLANTA UNIVERSITY

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR

THE DEGREE OF MASTER OF SCIENCE

BY

MELVIN L. CRUM

DEPARTMENT OF COMPUTER AND INFORMATION SCIENCE

ATLANTA, GEORGIA

JULY 1995

R= vi T= 45

ACKNOWLEDGEMENTS

Special thanks to my advisor Dr. Srikanth, committee members Dr. George and Dr. Perry, faculty of the Computer and Information Science Department, and Karl Lehenbauer and Mark Diekhans for the use of PARTEK.

TABLE OF CONTENTS

ACKNOWLEDGEMENTS ii

LIST OF TABLES v

LIST OF ILLUSTRATIONS vi

INTRODUCTION 1

Problem Definition 3

Motivation 4

Approach 4

Chapter

1. BACKGROUND INFORMATION 7

Pattern Recognition 7

Neural Networks 10

2. DATA EXTRACTION AND REPRESENTATION 16

3. DESIGNS AND IMPLEMENTATIONS 21

Multilayer Feedforward Neural Networks 21

Training Process 24

Bayes Classifier 27

4. TESTS AND EVALUATIONS 30

Multilayer Feedforward Models 30

Bayes Classifier 37

5. CONCLUSION 40

Future Work 41

Appendix

THIRD DOWN GENERATOR 43

REFERENCES 45

LIST OF TABLES

Table Page

1. Data Representation in the Data File 17
2. Unary Representation of Plays 18
3. Data File Representation 19
4. Breakdown of Plays in Data File 20
5. Performance of Networks 31

LIST OF ILLUSTRATIONS

Figure Page

1. A Pattern Recognition and Classification System 9

2. McCulloch-Pitts Model Neuron 11

3. Multilayer Feedforward Neural Network 22

4. Flowchart for Error Backpropagation Training Algorithm 25

5. A Bayes Classifier Model 28

6. Error Graph of the Network Models 32

7. Error Graph for 15 Neurons on the Hidden Layer . . 35

8. Error Graph for 20 Neurons on the Hidden Layer . . 35

9. Error Graph for 30 Neurons on the Hidden Layer . . 36

10. Error Graph for 40 Neurons on the Hidden Layer . . 36

11. Confusion Matrix for Bayes Classifier 38

INTRODUCTION

A common problem with the defense of football teams is understanding the offensive strategies of the opponent. Most defensive coordinators have the task of trying to recognize the play an offense is attempting to execute before the execution begins. Moreover, the third down plays of a football game are crucial to both the offense and the defense in determining which team will have possession of the football after these plays. The offense needs to convert on third down situations to move closer to the goal of scoring a field goal, a touchdown, or both. However, the defensive unit has to eliminate the successful conversion of the play execution by the offense in order to gain possession of the football and have the same chances of scoring as the opponent.

For the defense to accomplish the task of defending against a third down conversion, tendencies of the offense to execute plays based on the location of the football on the field must be compiled and analyzed in hopes of recognizing patterns in the third down plays. However, some offenses are very good at mixing up their play selections on third down in order to confuse the defense on what play to expect. This often causes a breakdown on the defense. For example, if the ball is on the 35 yard line, the offense needs three yards for a first down, and the football is located in the center of the field, then a choice of six different plays are at the disposal of the offense. The offense can run left, run middle, run right, pass left, pass middle, or pass right. Since three yards is not much to gain and most teams prefer to run on short yardage situations, the offense may elect to run the ball. This cuts the decision in half, but there are still three choices to consider.

The use of neural networks will serve as a mechanism that will attempt to give some insight as to what an offense will generally do under these circumstances. Once the training data is fed into the neural network models, then testing data will be used to analyze the performance of the trained models.

This thesis explores a multilayer neural network approach and a Bayesian approach using apriori probabilities to predict tendencies of opposing offensive teams. The results from the neural networks, depicted in graphs and tables, will show how well offensive tendencies can be predicted. This is achieved by training the neural networks with a subset of randomly chosen third down plays and testing them with the remainder of the input data introduced to the models for the first time. The results from the Bayes model will be illustrated in a matrix table suggesting which plays were correctly and incorrectly classified. A conclusion of whether or not a football team actually has tendencies that can be recognized from a neural network approach will be drawn from the performance of these models.

Problem Definition

This section succinctly states the problem at hand and relates it to a neural network solution. The layman's definition of the problem deals with defensive coaches deciding what defensive play to use in third down situations where the offense must convert in order to keep the football in their possession. By attempting to predict tendencies based on third down plays of a football team, using a neural network approach will determine if patterns in a team's game plan can be recognized and classified into predefined classes. If the neural network models are successful in recognizing tendencies of the offensive unit, then this will serve as a tool to assist coaches on how to better defend against the third down play of opponents given a history of third down plays executed by the opponent's offense.

Motivation

The motivation for predicting tendencies utilizing the tools of artificial intelligence was inspired by the concept of a machine with built in intelligence. It is a very interesting task to construct an architecture that has the capability to learn from its environment and make necessary changes (adaptation) to recognize certain situation based on past experiences. Coaches for the defensive unit of the football team make similar decision based on what is known about the opponents, given the situation on the current third down. The defensive coordinators express the need for some mechanism to assist them in preparation for determining possible plays of an offense. Having this kind of tool handy could potentially improve the efficiency of the defensive game plan.

Approach

The approach used here is a study of different neural network architectures that are trained from third down plays. The input to the networks consist of four variables, namely the yard line where the play started, number of yards to go for a first down, whether the ball is on the left, middle, or right side of the field, and what play was executed. Six hundred third down input data were created with known tendencies and stored in a data file. The reasoning behind this type of an approach was to be able to control the tendencies of a fictitious football team and observe how well the neural networks train and test on a subset of the data file.

This thesis attempts to offer a neural network approach to predicting tendencies of a football team. Chapter one offers some background information about the origins and some sound reasoning for experimenting with neural networks. This chapter also decomposes the architecture of a network and explain the components and their functions.

Chapter two provides the methodologies associated with extracting the data used for training and testing the neural network models. This empirical data was modeled after actual game statistics compiled by the athletic department for past games of Clark Atlanta University football team in the fall of 1994. Tables serve as visual aid to demonstrate the representation of the data used by the network models.

Chapter three covers the designs and implementations for the models used to evaluate the tendencies of the empirical data with predefined tendencies. Detailed documentation of the architectural designs, data flow, and training procedure is also presented in this chapter to uncover the internal processing of neural networks.

Chapter four presents the evaluation of tests on the models performance after presenting new data to the trained models. This chapter also makes an attempt to offer analytical deductions drawn from the illustrations documented therein.

Chapter five is partially devoted to summarizing the performance of the models and offering answers concerning the test results and expectations. The summarization focuses on design and implementation and what can be done to improve the performance capability of the neural networks. The rest of the chapter suggests some future work with the present models and exploration of an alternative approach to predicting tendencies.

CHAPTER 1

BACKGROUND INFORMATION

This chapter documents a general pattern recognition (PR) system which laid the groundwork for neural network models. Documentation about the early neural network architectures are also discussed from building blocks to their power system.

Pattern Recognition

Machine intelligence will dominate the technological industries in the 1990s because of the increasing applications that require knowledge for decision making processes. This knowledge can be acquired by the machines (computers) through the use of pattern recognition techniques which play a vital role as a component of intelligent systems and are used for data preprocessing and decision making. Pattern recognition is defined as the science that concerns the description of classification (recognition) of measurements (Schalkoff 1992). Most agree that this field of science is an important, useful, and rapidly developing technology that is applicable to many real world problems (Schalkoff 1992). Pattern recognition is a widely used and recognized discipline that is comprised of a broad body of loosely related knowledge and techniques rather than a single approach to a problem. In the past, two major approaches to pattern recognition were the statistical (decision theoretic) and the syntactic (structural) approaches. The statistical pattern recognition approach assumes that there is an underlying and quantifiable basis for statistics in generating patterns. In the syntactic pattern recognition approach, the structure itself provides the fundamental information for recognizing patterns. Recently, neural networks have provided a third approach that is widely used with the 'black box' implementation for pattern recognition algorithms. The algorithms are rigorous mathematical formulas that process the weights and inputs within the neurons of a neural network. This third approach is mainly concerned with what the inputs and outputs are, not on how the information fired to the next level is achieved. Firing refers to information channeled as output once exceeding the threshold value.

Prior to the actual algorithm that makes the decision as to what class the input data belongs, known as the classification process, important attributes of the input data are filtered or transformed via feature vectors. These vectors provide the computational data and are primarily found in statistical and neural pattern recognition models. Figure 1 shows a commonly used pattern recognition and classification model.

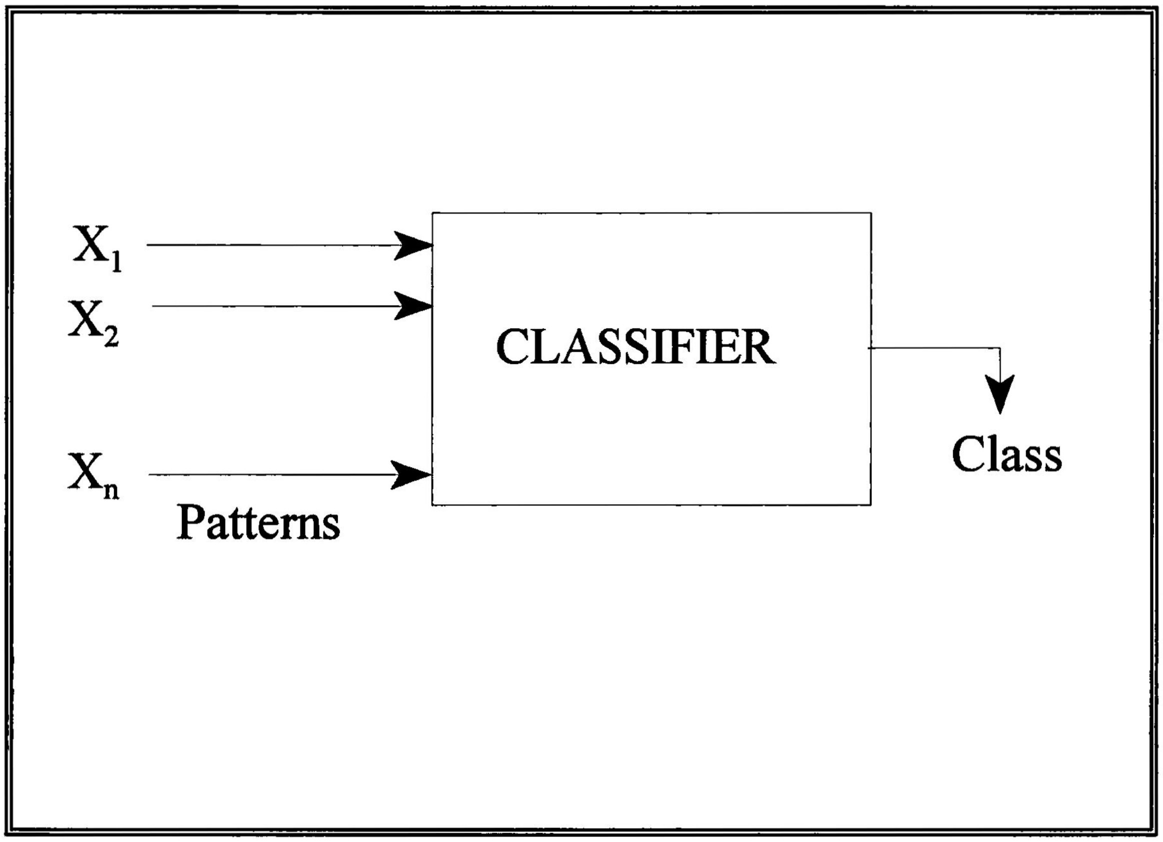


Figure 1. A Pattern Recognition and Classification System.

This model of a PR system accepts input patterns that are channeled through a feature extractor where measurements of the attributes are formed into feature vectors. These feature vectors are then fed into the classifier where a classification algorithm is performed on the vectors.

Information from the classifier is termed as the actual decision made by the algorithm.

One particular technique incorporated into the neural pattern recognition approach is the 'black box' structure. A 'black box' viewpoint treats problems from an input/output perspective by specifying two things: (1) an internal computation; and (2) a stimulus/response (S-R) based training set (e.g., a set of input/output pattern pairs). Neural pattern recognition seems to provide good examples of the black box structure where the emphasis is placed on the utilization of training data to achieve good pattern mapping. This technique is modeled after the human brain which is, for the purpose of present artificial intelligence implementations, a black box. Observation and emulation of intelligent behavior (including pattern recognition and classification) is achieved without a detailed set of algorithms that set limitations for the input/output characteristics of the brain. This thesis is primarily focused on utilizing neural network implementation in an attempt to offer a solution to the problem.

Neural Networks

The first formal definition of constructing a neuron model was introduced by McCulloch and Pitts in 1943. Their model of the simple neuron laid the groundwork for future research and developments. Part of the reason their work was important is that it addressed problems in both theoretical computer science and brain modeling. This neuron model was based upon considerations that were highly simplified versions of the biological model medically known as a nerve cell (Zurada 1992). Figure 2 shows the McCulloch-Pitts model of the neuron suggesting that a number of inputs are fed to the neuron via weighted links.

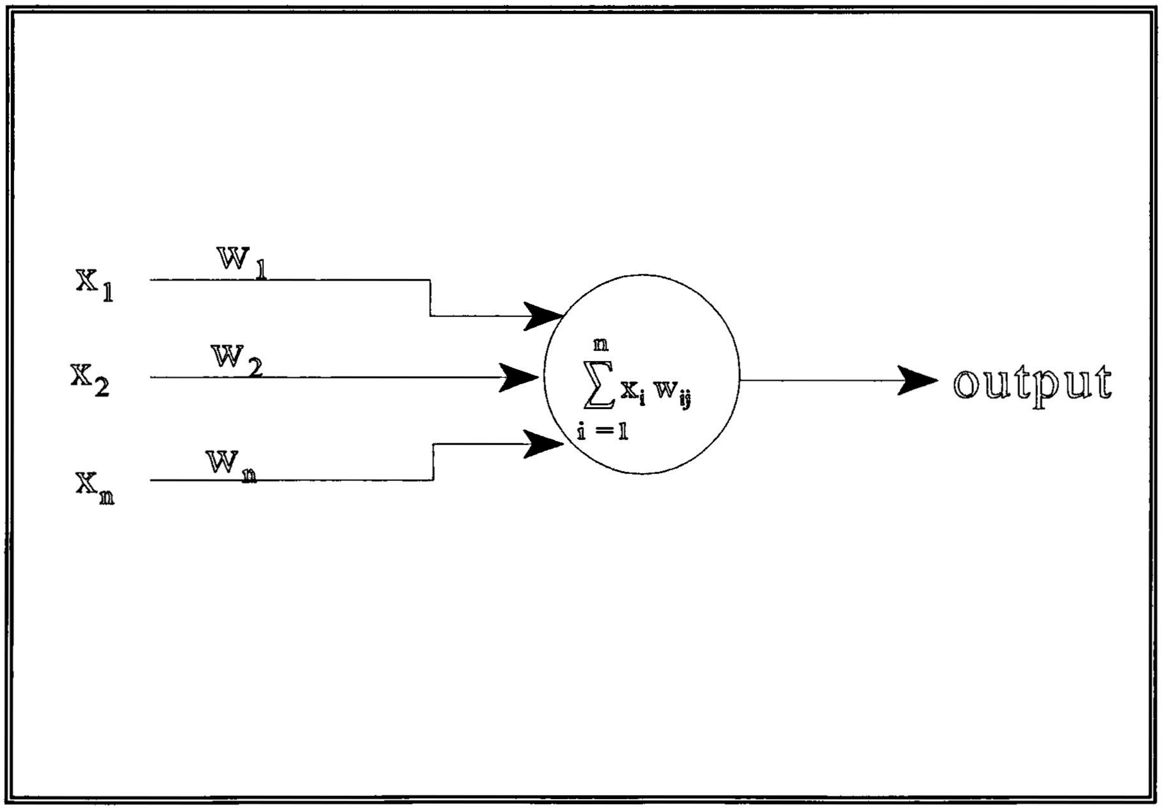


Figure 2. McCulloch-Pitts Model Neuron.

This early model was equipped with all the necessary elements to perform logic operations, and thus this unit could function as an arithmetic logic computing element. The summation of all the inputs multiplied by their corresponding weights is processed within the neuron and once the processed value reaches or exceeds some predetermined threshold, the new value is fired as output from the neuron.

Before proceeding further, an examination of the origins and important features of neural network is necessary. For years, one of the many goals of humankind has been to develop machines to perform all cumbersome and tedious tasks. The era finally came when levers, wheels, and pulleys were invented to move heavy objects for humans (Zurada 1992). Today engineers and scientists are trying to develop more machines that do not require much assistance from humans in order to perform certain tasks. These machines are now fully operational and are currently known as Neural Networks, which are machines built with some degree of intelligence for decision making applications. Neural network models are composed of many nonlinear computational processing elements generally known as neurons and weighted connections. Each neuron in a neural network collects the values from all the associated input connections, executes a predefined mathematical operation (typically a dot product followed by a nonlinear function) and produces a single output value (Sanchez-Sinencio and Lau 1992). There are several features that are important and apply to all neural networks:

- Each neuron acts independently of all others - each neuron's output relies only on its constantly available inputs from the input connections.
- Each neuron depends only on local information - the information that is provided by the connections is all a neuron needs to process.
- The large number of connections provide a large sum of redundancy and models a distributed representation.

The first two features above enable neural networks to operate efficiently in parallel enabling problems to be solved with great speed. The latter feature provides inherent fault tolerance that serves well in cases where damage to some neurons will not impair the overall performance of neural networks significantly (Lippmann 1987). Combining the three features with the proper arrangement of the neurons, the introduction of a nonlinear function, and the appropriate learning algorithm, neural networks are able to learn arbitrary nonlinear mappings. The neurons are known to perform in parallel and are constructed in pattern form reminiscent of the biological neural nets (Lippmann 1987). The neurons individually function rather slowly and imperfectly, but collectively their performance of tasks has not been matched by the largest of computers (Lippmann 1987). Neural networks explore many hypotheses simultaneously competing through the use of massively parallel architectures. These networks are constructed with many processing elements capable of computations linked together with variable weights (Lippmann 1987). This mode of processing differs from the von Neumann sequential computers which perform instructional programs sequentially.

The neural networks, broadly speaking, can be divided into two classes: those that involve some mechanism or algorithm for learning and those that do not. The networks that involve learning are sometimes referred to as backpropagation networks since the training (learning) algorithm used is called backpropagation (Sanchez-Sinencio and Lau 1992). The learning algorithm is used in the networks with supervised learning to achieve a minimal error at the output layer of the network. This has proven to be very useful in the adjusting of the weights and threshold values during the training process of neural networks.

The neural network models are typically specified by the network topology, characteristics of the neurons, and the training and learning algorithm. The neural network topology is the structure of the network based on the flow of information from one internal point to another. For example, a neural network whose input information propagates forward from neurons of a layer to neurons of the next layer without any connectional links leading back to the source layer is called a multilayer feedforward network. The neuron characteristics are described by the activation functions applied after the dot product has been computed from all the weight and input values propagated to that neuron. One such activation function is known as the sigmoidal function which is performed by the decision element on the activation value produced by the neuron's computational processing. Simply stated, the function class forces a *high* (active class) or *low* (inactive class) value from the neuron towards one of the output classes defined on the output layer. The training algorithm is constructed to handle the process of misclassifying input patterns either after every iteration or once all the input patterns have been processed. Arguably the most appealing process in the neural network is the learning algorithm used to update (change) the weights values that results in the capture of information which can be recalled at some later time.

In the next chapter, data collection and representation for neural network processing is explained in great detail. This chapter also discusses tendencies placed on the data.

CHAPTER 2

DATA EXTRACTION AND REPRESENTATION

This chapter documents how the data was formed based on actual third down plays of the Clark Atlanta University football team.

The initial data was extracted from statistics of four previous games played by Clark Atlanta University (CAU). A total of 24 plays were executed on third down and provided a good working model for representing the parameters associated with the inputs for the neural network. The problem found with this data from the games is that there was not enough third down plays to properly generalize the information present in the data. The network needs to be trained using many plays in order to allow the network to make more of a generalization about the information in the input data. The third down parameters used as input for the neural network are based upon position (yard line the play took place), yards to go (yards needed to complete a first down), and location of the football (left, right, or center hash mark) on the yard line. Table 1 shows a sample of the input data.

Table 1. Data Representation in the Data File

Yard line of Play	Yards to go	Position of ball
1	5	1
35	3	3
6	2	2
95	4	1

In Table 1, notice the position of the ball on the field (third column) is represented by numerics since the network only accepts numerical data as inputs. So, the numbers 1, 2, and 3 are associated with left, middle, and right respectively.

The output for each third down situation is represented by plays executed based on the inputs. Since the supervised training is the technique for the neural network learning, the output must be known for each input yielding a one-to-one mapping between the third down parameters and the play executed as a result of those parameters. The plays (outputs) that can be executed by a football team are as follow: run right, run middle, run left, pass right, pass middle, and pass left. These plays were mapped to a unary representation so that the class that represents the current input will be the only active class at a given time in the network during classification. Table 2 depicts the plays and their unary representation.

Table 2. Unary Representation of Plays

Plays	Unary number
run right	000001
run middle	000010
run left	000100
pass right	001000
pass middle	010000
pass left	100000

The unary representation is used for the plays in order to have one and only one of the six output classes active in the neural network after an input pattern is presented. The data that is presented to the network for classification is shown in Table 3 as a sample format of the actual data used to train and test the networks.

Table 3. Data File Representation

Yard line	Yards to go	Position	Play
35	7	2	001000
12	3	1	000001
66	8	3	100000
52	2	2	000010
9	9	2	001000
91	3	2	000010

The first three columns in Table 3 represent the third down parameters and column four is the associated play executed based on those parameters. Column 1 depicts what yard line the ball was on when the play was executed. The values ranged from 1 to 98 (2 yard line of the defense) where it was possible to have third down plays. Column 2 shows how many yards are needed for a first down and its values range from 1 to 10. When the values are between 1 and 5, the offense will run the ball in an attempt to gain first down yardage. Values ranging from 6 to 10 puts the offense in a position where the best option for first down conversion is to pass the ball. The third column provides values associated with the position of the ball on the hash marks. For example, 1 represents the ball was placed on the left hash mark of the yard line, 2 depicts the center hash mark, and 3 is associated with the right hash mark of the yard line. A sample third down based on Table 3 would read as follows: third down, ball on the 35 yard line with 7 yards to go for a first, and the ball is placed on the center hash mark. The play executed as a result of the parameters was a pass right. Table 4 gives a detailed outline of the how the data was generated for each class of plays and what percentage of times those plays exist in the data file.

Table 4. Breakdown of Plays in Data File

Play	Total number	Percentage
run right	216	36
run middle	108	18
run left	36	6
pass right	120	20
pass middle	72	12
pass left	48	8

The total number of input data generated for training and testing was six hundred. Three hundred of the input patterns were randomly chosen for training and the remaining were used for testing purposes.

The next chapter defines the neural network models used for processing the third down data. These models along with their training algorithm are used to find a mapping between the input data and output classes.

CHAPTER 3

DESIGNS AND IMPLEMENTATIONS

This chapter documents the designs and implementations associated with the Multilayer feedforward and Bayes classifier. The design advantages are discussed in detail in each section of the design models. The architectural foundations serve as explanations to how the models perceive and process their data.

Multilayer Feedforward Neural Networks

A multilayer feedforward architecture is defined to be *m* neurons receiving *n* inputs that feed information in only one direction (e.g., input to output) without any feedback pathways in the network (Sanchez-Sinencio and Lau 1992). The model used in this thesis for predicting tendencies consist of a two layer architecture with a single hidden layer and an output layer. The number of neurons in the hidden layer was varied from 15 to 40 in an attempt find an optimal size based on the results from training and testing the data.

The architecture of the multilayer network model is shown below in Figure 3.

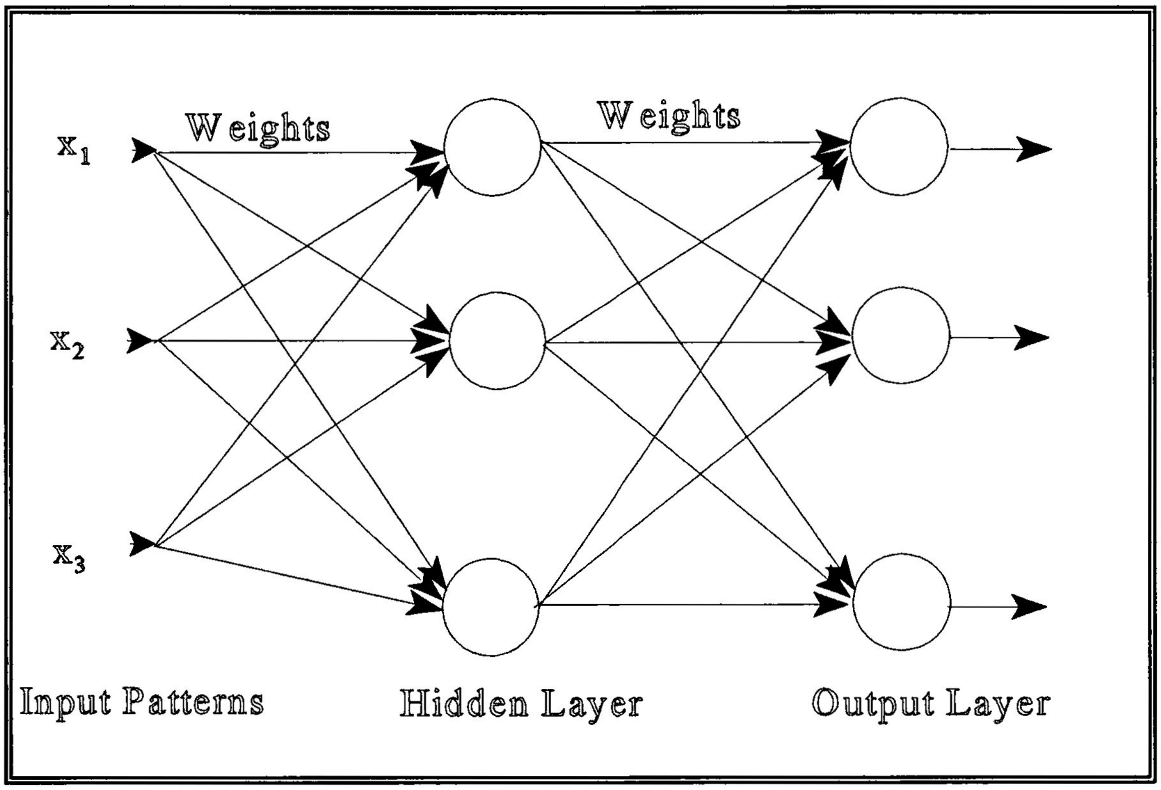


Figure 3. Multilayer Feedforward Neural Network.

In Figure 3, each x_i represents an input parameter (input parameters associated with a third down play) that feed the input to the hidden layer neurons for processing. These x_is are called input signals and their only function is to introduce the input patterns to the network. The weighted links (connections) have two main functions in a neural network. The first task is to provide forward pathways for the input patterns to travel through the network. This is accomplished by propagating the information from one neuron to another via the weighted links. The second task of the weighted links is to modulate (regulate) the amount of information passing between two neurons. Modulation is achieved by links with positive, negative, and zero values. When weighted links are assigned positive values, more information flow through the pathways than those assigned negative ones. Zero valued links are treated as if no connections are present between neurons.

Like the weighted links, neurons also have two important functions to carry out in a neural network. First, all the information needed by a neuron for processing is collected through the weighted input connections (weighted links attached to the neuron). The incoming signal is a linear combination (dot product) of the inputs and the associated weights on the links. Next, the neurons produce a single output value that is propagated to receiving neurons or serve as an output from the network. Prior to propagation, a nonlinear function is applied to map the computed value into a prespecified range. This function is commonly known as an activation function. When the two processes of neurons are combined, the complete internal processing of neurons are achieved which allow neural networks to map input patterns to output classes.

Training Process

The training algorithm used for the multilayer feedforward neural network models is error backpropagation (backprop) with supervised learning. This algorithm is an iterative gradient (slope) descent algorithm designed to minimize the mean square error between the actual output of the network and the desire output (Lippmann 1987). Figure 4 illustrates the flowchart of the error backpropagation training algorithm for the two layer network model.

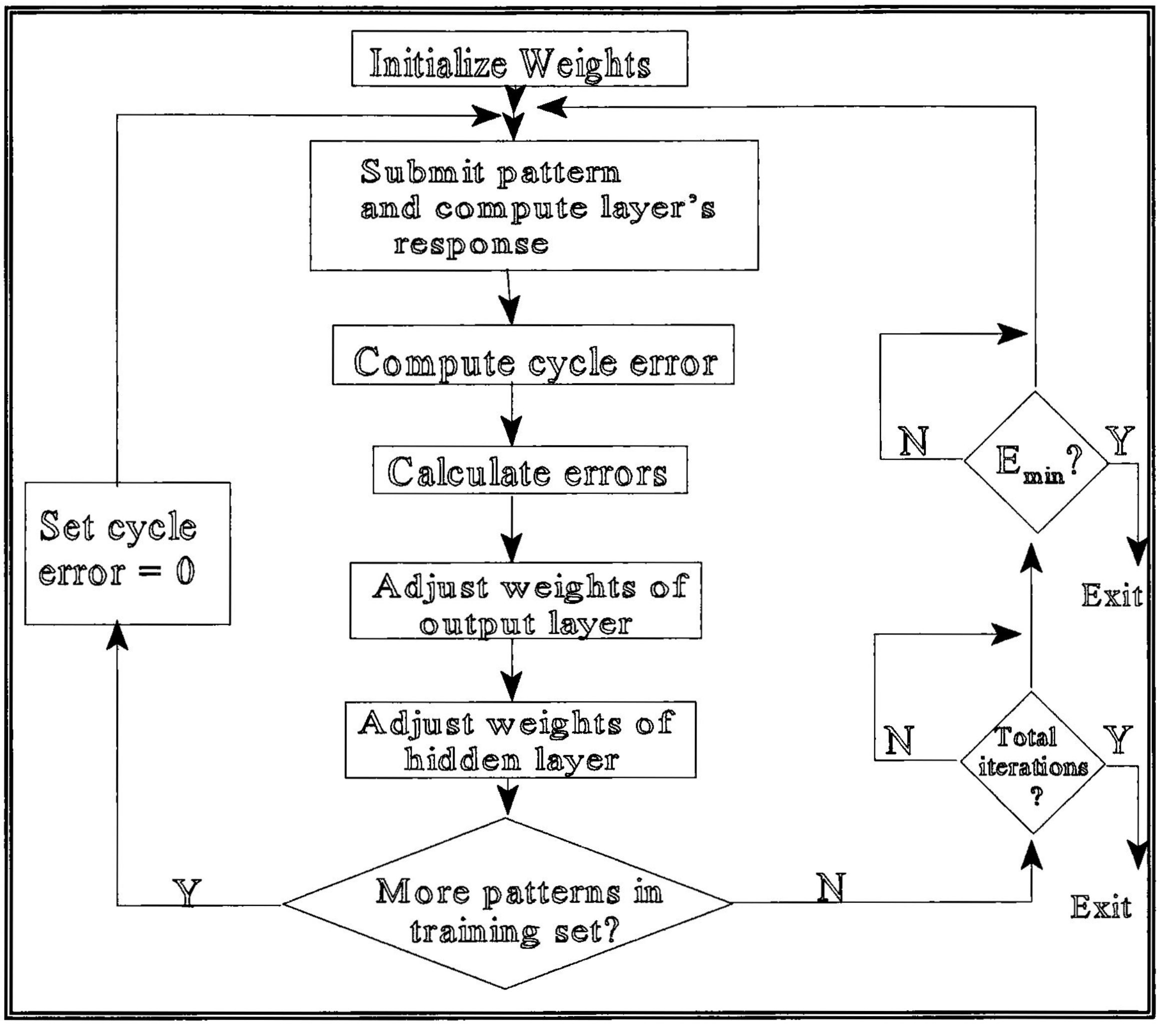

Initialize Weights
Submit pattern and compute layer's response
Compute cycle error
Calculate errors
Adjust weights of output layer
Adjust weights of hidden layer
More patterns in training set?
Set cycle error = 0
Y
N
E_{min}?
Total iterations ?
Exit
Exit

Figure 4. Flowchart for Error Backpropagation Training Algorithm.

The first step is to initialize weights to small random values. The choices of values are important since they control the initial starting point and thus the portion of solution space explored. In the second step, patterns are submitted to the network and the layer's response is computed. Results from the layer are propagated to the next layer or serve as output for the network. The third step in the training phase is computation of the cycle error. The total error computed over the pattern set is used as a termination condition once the value falls below an acceptable error value. Error calculations for the output and hidden layers are computed in step four. In this phase, supervised learning presents the desired output response and compares it to the actual output from the network. The distance between *desired* and *actual* responses serves as an error measure that is used to adjust the weights along the pathways. Steps five and six use the error measurements to adjust weights of the output and hidden layer via propagation back through the network. Once the modification to the weights are completed, step seven submits more patterns, or a new training cycle begins by initializing the cycle error to zero. The goal is to minimize the mean square error between desired and actual responses by finding a set of weights that determine the separable regions among the output classes. The next model uses information about the output classes to determine the best class for a input pattern.

Bayes Classifier

The Bayes classifier model is not examined and explored in great detail as the multilayer feedforward. The decision problem is viewed from probabilistic terms. The assumption is that all relevant prior probability values are known in advance (Duda and Hart 1973). The apriori probability is taken from the input data file and is produced based on the frequency of each play in the training data file. The individual likelihoods of plays executed are computed from each input pattern. Input parameters of the model have to be assumed statistically independent when using the Bayesian approach to pattern classification. This is because different plays were executed by the offense under the same third down conditions. The task is to construct a model capable of finding a general solution that will approximate the play and direction most likely to be used in a certain situation. Figure 5 illustrates the design of a Bayes classifier.

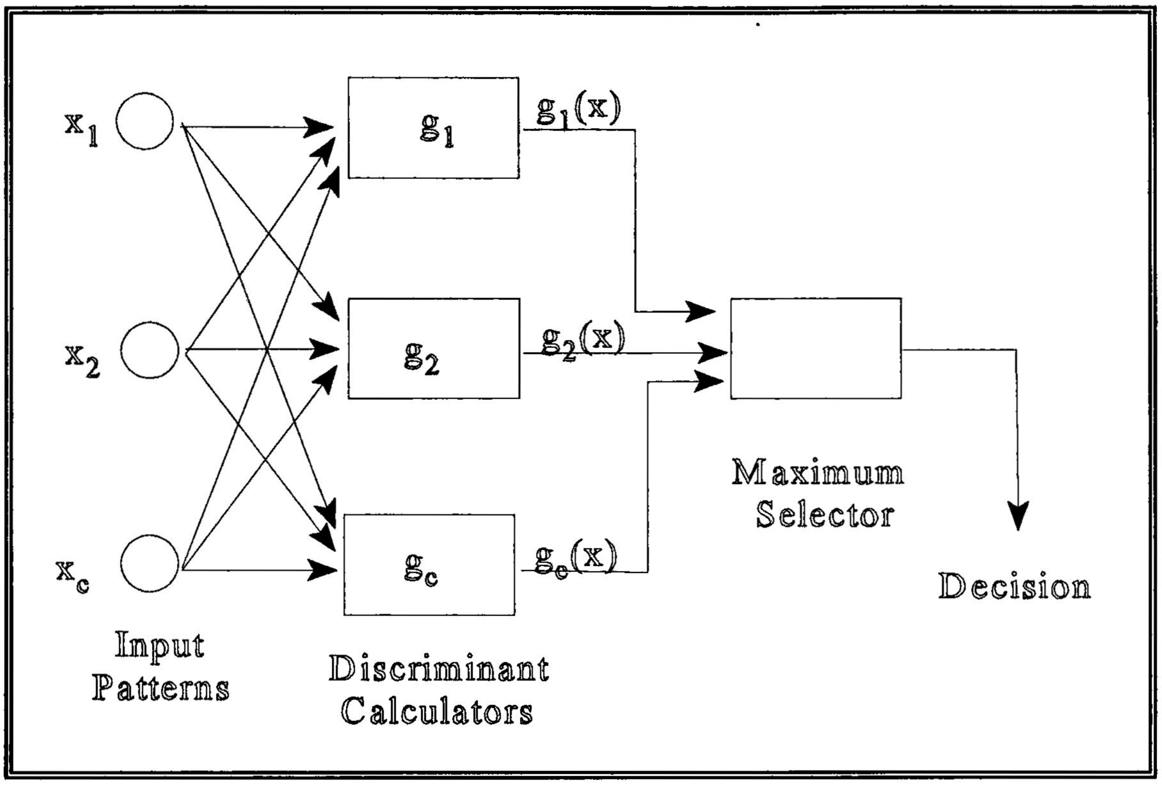


Figure 5. A Bayes Classifier Model.

In Figure 5, the discriminant calculators are used much like the neurons of a multilayer feedforward network. Their task is to map the input patterns to an output class. This is done by calculating the probability of each class and the probability that the current input pattern belongs to each of the classes. Next, a comparison is made to determine which of the discriminant functions produced the largest value. By choosing the value, the maximum selector provides the output class to which the input pattern is mapped. For example, if $g_1(x) > g_2(x)$, then the input pattern is mapped to the class represented by g_1.

The next chapter provides results from the experiments performed using the previously defined data and models. These results are discussed in an attempt to explain the performances of the models.

CHAPTER 4

TESTS AND EVALUATIONS

This chapter documents the results from testing the trained multilayer feedforward and Bayes classifier models. The results are examined and evaluated to determine how well the neural network models performed using the testing data.

Multilayer Feedforward Models

The objective for testing and evaluating the models was to find the optimal model that produced the lowest error rate during training. As a result, four models were used in this experiment and their results are depicted in a graph and a table.

The experiment was conducted using PARTEK a pattern analysis and recognition software package. PARTEK enabled the building of neural networks, data manipulation and visualization, and generation of solutions. In the first step, the data file was imported into PARTEK using a format file to transport data to a spreadsheet. Once the data had been collected into the spreadsheet, three hundred input patterns were randomly filtered into a training set. This training set was then used to train the network models.

Next, the multilayer feedforward models were designed by specifying an activation function, number of hidden layers, and neurons. PARTEK uses information about the data in order to set up the number of input signals and output neurons. Now that the data and models are constructed, training parameters have to be set for training the models. A number for total iterations is selected and training starts with close attention given to the minimum error. Once the minimum error reach a low plateau, the training is stopped and the results are recorded. Table 5 describes the performance of the four network models.

Table 5. Performance of Networks

Architecture	Percentage	Minimum Error
3x15x6	67.67	.150
3x20x6	68	.149
3x30x6	50.33	.167
3x40x6	50	.168

Figure 6 offers a mean square error graph representation for the performance of the four multilayer network models.

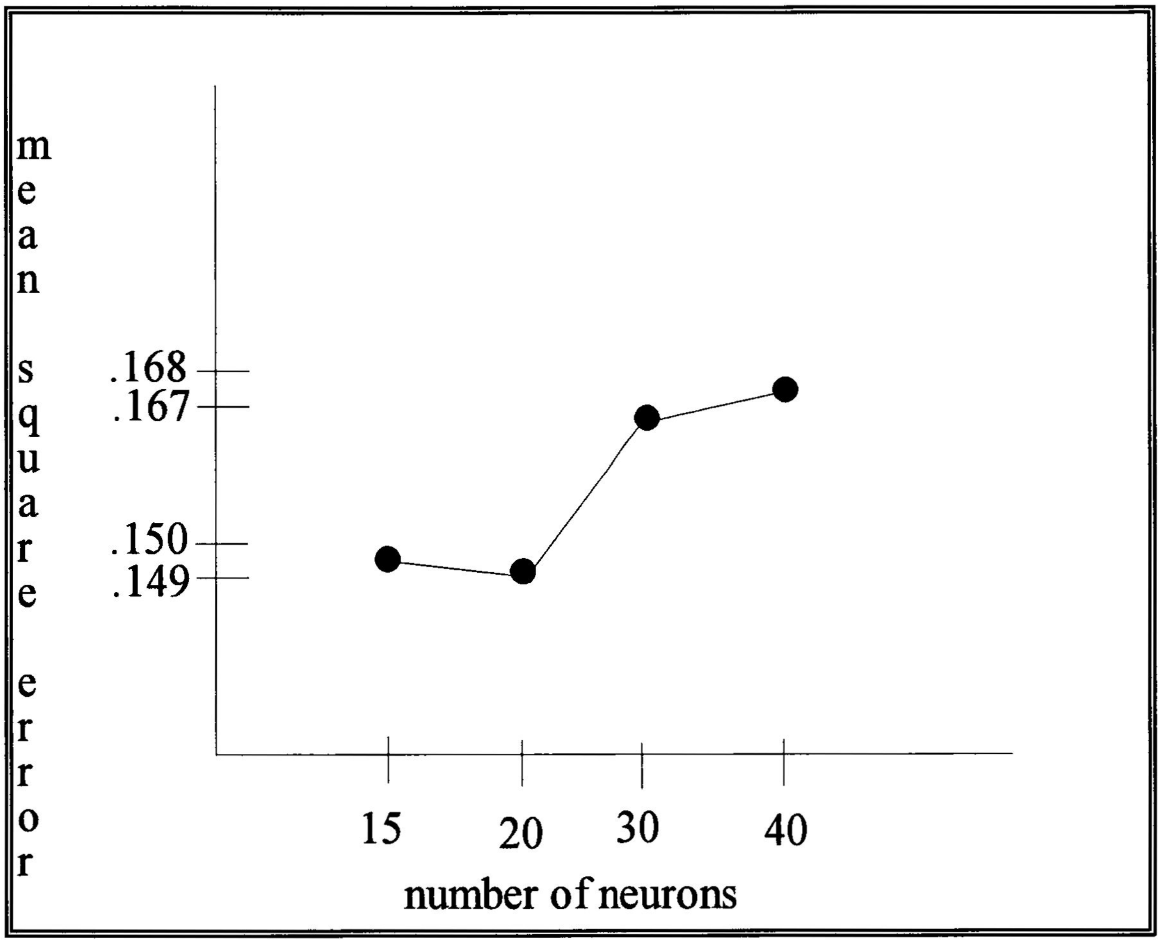


Figure 6. Error Graph of the Network Models.

Each of the neural network models were trained until the error reached a low plateau. The mean square error is plotted along the y axis and the number of neurons at the hidden layer is plotted on the x axis. This process suggested that the training algorithm minimized the error to a region that stabilized the learning process. In viewing the graph of Figure 6, the hidden layer constructed with 20 neurons performed the best over the training and testing data. This model was able to map 207 out of 300 input patterns to output classes yielding a 68% performance over the testing patterns. Given that the plays cannot be totally separated over the output classes, the networks made a very good generalization about the data as a whole.

The performance of the neural network models were in the range of 50% - 68%. This implies that out of 300 input patterns propagated through the networks, between 150 and 207 of the patterns were correctly mapped to the desired output classes. While the performance was not spectacular as the teams were able to mix up the play selections when the ball was placed on the center hash mark, the results suggest that this team did have tendencies and they could be recognized from the neural network approach. The results also show that input patterns and plays appearing in the training data more frequently had a very low misclassification rate as compared to the others. This implies that the more the network processes similar patterns, learning tends to shift the weights in favor of those patterns. The learning is recognized when rate of error becomes stable. For example, the error plots for the four multilayer feedforward network located in Figures 7 thru 10 illustrate the leveling off of the error that suggests training the network has reached a point where it can no longer effectively learn the input patterns.

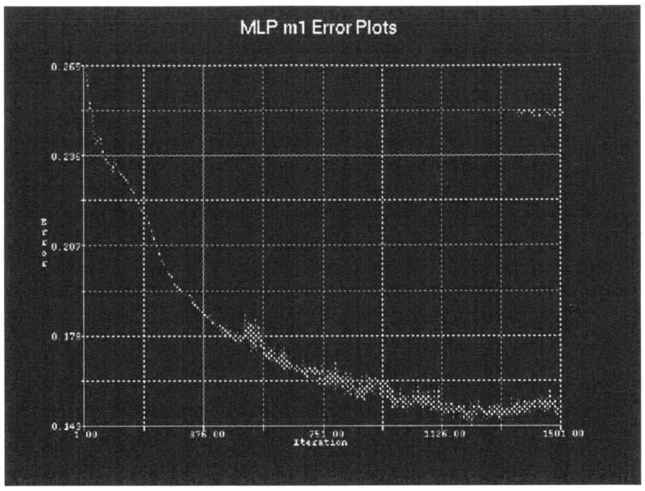


Figure 7. Error Graph for 15 Neurons on the Hidden Layer.

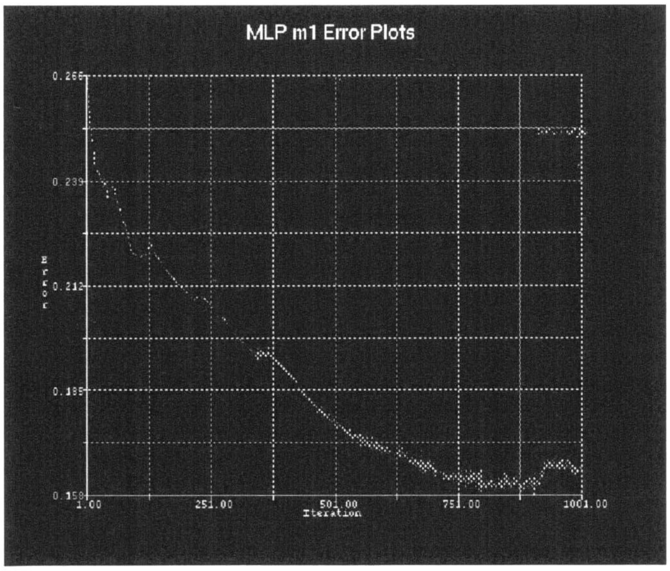


Figure 8. Error Graph for 20 Neurons on the Hidden Layer.

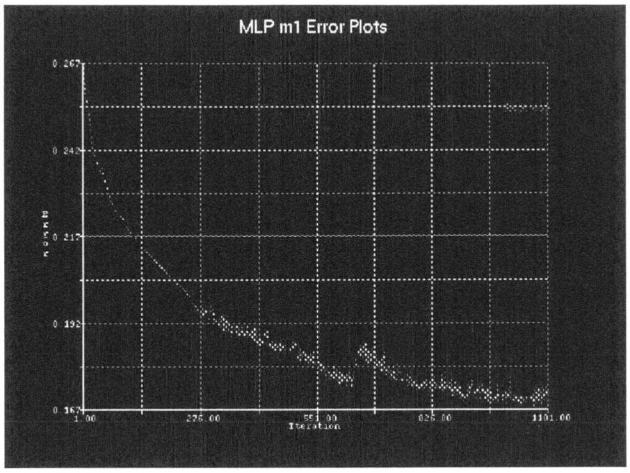


Figure 9. Error Graph for 30 Neurons on the Hidden Layer.

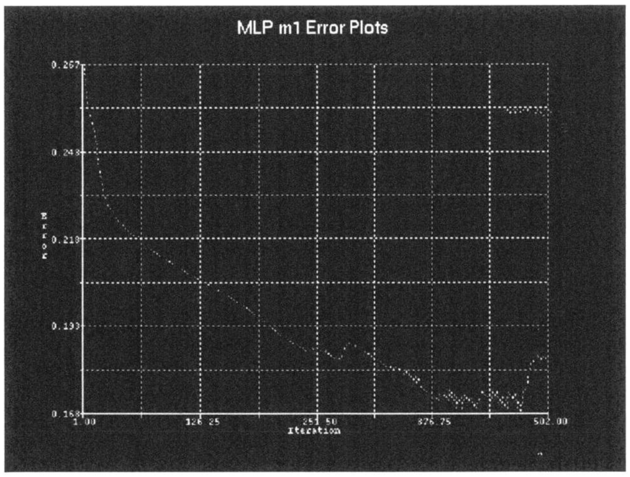
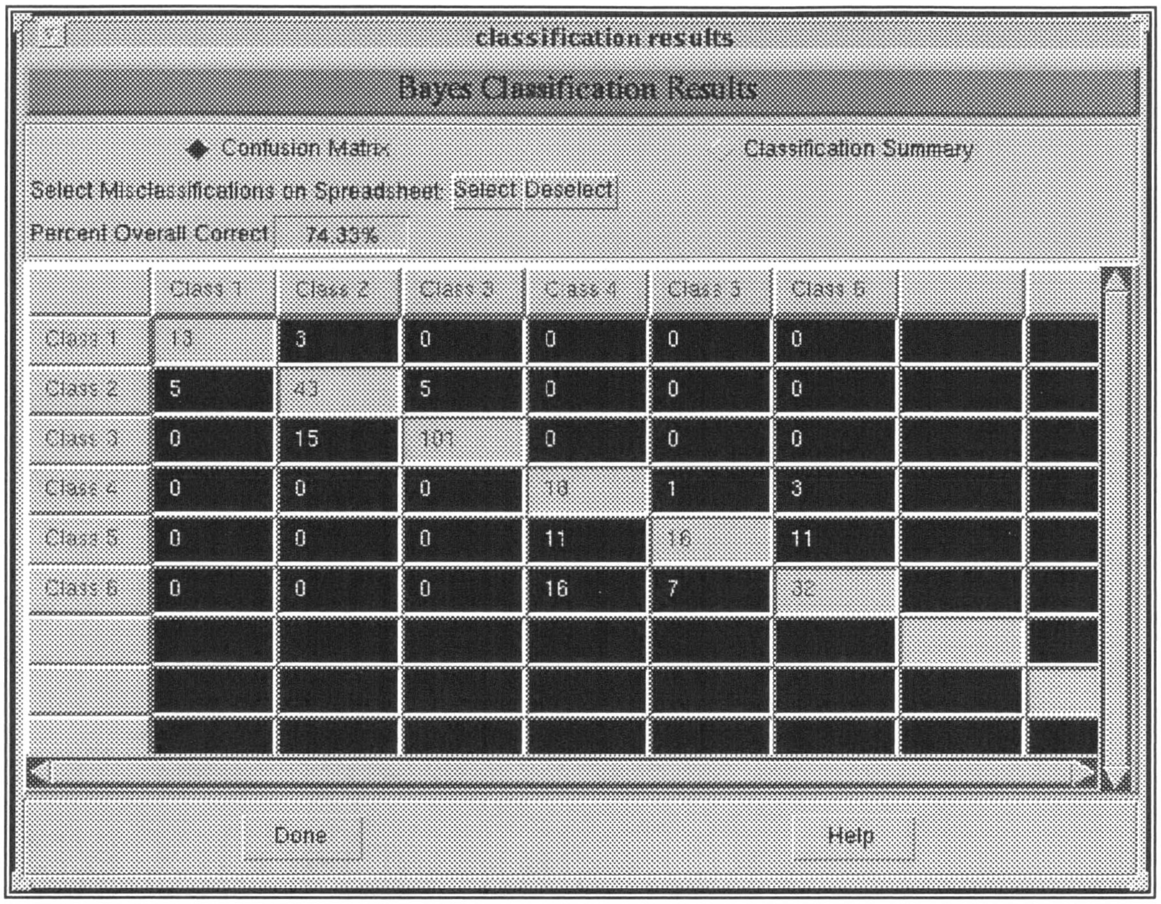


Figure 10. Error Graph for 40 Neurons on the Hidden Layer.

Bayes Classifier

The Bayes classifier provided a visual tool where the classification results are stored in a matrix format. The rows and columns combined represent the number of input patterns correctly and incorrectly mapped to the six output classes. The correct mappings are stored in the cells where the row and column numbers are identical (the diagonal). The other cells of the matrix depict misclassified input patterns. Figure 11 illustrates the resulting matrix and the percentage of input patterns correctly mapped to output classes.



Bayes Classification Results

◆ Confusion Matrix   Classification Summary

Select Misclassifications on Spreadsheet: Select Deselect

Percent Overall Correct: 74.33%

	Class 1	Class 2	Class 3	Class 4	Class 5	Class 6
Class 1	13	3	0	0	0	0
Class 2	5	43	5	0	0	0
Class 3	0	15	101	0	0	0
Class 4	0	0	0	18	1	3
Class 5	0	0	0	11	16	11
Class 6	0	0	0	16	7	32

Figure 11. Confusion Matrix for Bayes Classifier.

Both models used in this thesis gave good results in terms of the data provided for training and testing. The important concept to remember is that the goal is not to get 100% results, but rather construct and train the models to make a generalization over the entire data set. Though the results may not seem favorable, the defensive coordinators can have assistance in outlining the most successful plays an offense will likely execute in crucial situations.

The next chapter gives some final remarks about the data and models used for this thesis. Some suggestions for future work are also offered to further investigate predicting third down tendencies of a football team.

CHAPTER 5

CONCLUSION

Answering the question of whether a football team's offense has tendencies to execute certain plays on third down was not an easy task. However, by utilizing techniques presented in artificial intelligence, these tendencies were recognized. Neural network models enabled the input patterns to be classified into one of the six predefined classes where information about the members in the classes were somehow similar. The classifier of the neural networks played an important role in the classification of the input patterns into the output classes.

One factor pertaining to the play tendencies of the offense must be noted in order to understand the results produced by the networks. A team can execute different plays for the same situations which cannot be recognized based on the information from the inputs. For example, consider the situation where the ball is on the 45 yard line, 3 yard to go for a first down, and the ball is positioned in the center of the field. What play should be executed? This situation is ideal for the neural network models examined in Chapter 3 since they treat each input vector independently of all other inputs.

The set back with the multilayer feedforward models is determining the number of layers and neurons per layer during the design phase. There are no concrete methods documented for selecting a particular size for the network. As a result, testing the network with different number of neurons on the hidden layer is the only alternative approach.

Like the multilayer feedforward network, Bayes classifier demonstrated good performance on predicting tendencies of the third down situations. The known probabilities of the plays aided in selecting the best possible class for the input patterns. This model could be thought of as a tool for summarizing the play execution of an offense. A defensive coach could utilize the information provided to prepare the defensive strategies of crucial third downs late in the game.

Future Work

The research of predicting tendencies from a neural network approach is a very good building block for more experimental work. More tests need to be conducted on the network models in an attempt to locate better architectures.

This can be done by increasing the number of neurons on the hidden layer and training the network in order to evaluate the performance. Another alternative is to insert additional layers. Doing this will affect the training time of the network because the processing and weight adjustments involved with complex networks. However, the learning rate and momentum terms can be modified in order to speed up convergence for the networks and large training data sets. This would help the effort to cut cost of updating networks after each iteration of training cycles.

The data used in this thesis was constructed based on the actual third downs of Clark Atlanta University. This modeling gave good results since the tendencies of the plays were known before examinations began. However, to truly test the performances of design models for the this thesis, real data must be used to assess how well they can recognize patterns in third down plays.

Another parameter not mentioned in this thesis that can prove to be very important during a game is time. Factoring the amount of time left in a game into the equation for determining play selections could possibly shed more light on the tendencies of an offense. Using time may improve the mapping between third down parameters and play classes. This could in turn increase the prediction of what plays the offense will generally run in crucial third down situations.

APPENDIX

THIRD DOWN GENERATOR

```
#include <stdio.h>
#include <stdlib.h>

/* This program generates data to be used in PARTEK as
input/output data of neural network models.  The data is
created using a random number generator function.       */

main ()
{

   int loc, ytg, i, stop, ploc, pytg, post;
   FILE *fp;
   char cplay[7];

/* This function calls on the system to generate random
numbers                */

   srand(time(NULL));

/* An ascii file is opened to accept three inputs and the
one output data for the neural network.  The information is
entered by the user at the prompt. */

   if ((fp = fopen ("prod", "a+")) == NULL)
     printf ("this is an error\n");
   else
   {
      printf("Enter location ----------------->");
      scanf("%d", &ploc);
      printf("Enter yards to go -------------->");
      scanf("%d", &pytg);
      printf("Enter position -- -------------->");
      scanf("%d", &post);
      printf("Enter play --------------------->");
      scanf("%s", cplay);
      printf("Enter # of plays to generate --->");
      scanf("%d", &stop);
      for (i = 0; i < stop;)
```

```
 loc = rand() % ploc +1;
 ytg = rand() % pytg + 1;
 if ((ytg > 5) && (ytg <= 10))
 {
      i++;
      fprintf(fp, "%d,%d,%d,%s\n" ,loc, ytg, post,
      cplay);
 }
 if (ytg <= 5)
 {
     i++;
      fprintf(fp, "%d,%d,%d,%s\n" ,loc, ytg, post,
     cplay);
}
fclose(fp);
```

REFERENCES

Duda, Richard O., and Peter E. Hart. Pattern Classification and Scene Analysis. New York: John Wiley and Sons, 1973.

Fu, K. S. Sequential Methods in Pattern Recognition and Machine Learning. New York: Academic Press, 1968.

Lehenbauer, Karl and Mark Diekhans. "PARTEK Tutorial Version 1.1 1992" [manual].

Lippmann, Richard. An Introduction to Computing with Neural Nets. The Institute of Electrical and Electronics Engineers (April 1987): 4-22.

Rogers, Steven K. and Matthew Kabrisky. An Introduction to Biological and Artificial neural Networks for Pattern Recognition. SPIE Opticial Engineering Press Washington: 1990.

Schalkoff, Robert. Pattern Recognition Statistical, Structural, and Neural Approaches. New York: John Wiley and Sons, Inc., 1992.

Simpson, Patrick K. Foundations of Neural Networks. Edited by E. Sanchez-Sinencio, and C. Lau. Artificial Neural Networks Paradigms, Applications, and Hardware Implementation. New York: IEEE Press, 1992.

Werbos, Paul J. Links Between Artificial Neural Networks (ANN) and Statistical, Pattern Recognitions. Edited by L. N. Kanal, A. Rosenfeld. Vol 11, Machine Intelligence and Pattern Recognition. New York: 1991.

Wantanabe, Satosi. Methodologies of Pattern Recognition. New York: Academic Press, 1969.

Zurada, Jacek M. Introduction to Artificial Neural Systems. New York: West Publishing Company, 1992.