DECISION SUPPORT SYSTEM FOR FERTILIZER USE

By

JIANHUA REN

Bachelor of Science

Shanxi Agricultural University

Taigu, China

1982

Submitted to the Faculty of the
Graduate College of the
Oklahoma State University
in partial fulfillment of
the requirements for
the Degree of
MASTER OF SCIENCE
December, 2000

DECISION SUPPORT SYSTEM FOR FERTILIZER USE

Thesis Approved:

[illegible]

Thesis Advisor

J Chandler

Hailin Zhang

[illegible]

Dean of the Graduate College

ACKNOWLEDGMENTS

I would like to take this opportunity to express my sincere gratitude and thanks to my major advisor, Dr. G. E. Hedrick, for his intelligent supervision, constructive guidance, inspiration and friendship. I wish to heartily express my appreciation to Dr. J. P. Chandler and Dr. H. L. Zhang for their assistance as committee members.

I would like to give my special appreciation to my wife, Yali, for her precious suggestions to my research, her strong encouragement at times of difficulty, love, understanding and support throughout my studies and during the preparation of this thesis. I would also like to thank my daughter, Jie, for her love and support during my studies.

Finally, I would like to thank the Department of Computer Science and the Department of Plant and Soil Sciences at Oklahoma State University for supporting me during my studies.

TABLE OF CONTENTS

Chapter Page

I. INTRODUCTION 1

1.1 Problem 1
1.2 Objective 2
1.3 Organization 2

II. LITERATURE REVIEW 4

2.1 Decision Support System 4
2.2 Expert System 6
2.3 Artificial Neural Network 9
2.4 Relation between ANN and Statistical Methods 15
2.5 Relational Database System and SQL 17

III. PROJECT DEVELOPMENT AND METHDOLOGY 21

3.1 Design of SFUDSS 1.0 21
3.2 Expert System Development 23
3.3 Knowledge Acquisition 23
3.4 ANN training algorithms 28
3.5 ANN design and implementation 35
3.6 Knowledge Representation 37
3.7 Access to Database 38
3.8 User friendly interface 39

IV. RESULTS AND DISCUSSION 40

4.1 Case for ANN training using GA 40
4.2 Case for ANN training using QuickProp 42
4.3 Comparison of GA with QuickProp 45
4.4 DBMS and knowledge base 50
4.5 Experts system testing 52
4.6 User friendly interface testing 54

V. CONCLUSION 59

5.1 Summary 59
5.2 Conclusion 59
5.3 Future Work 60

Chapter Page

BIBLIOGRAPHY 61

APPENDIXES 64

APPENDIX A – GLOSSARY 64
APPENDIX B – LIST OF ACRONYMS 67

LIST OF TABLES

Table Page

I. Nitrogen requirements for small grain crops 24

II. ANOVA for quadratic model for wheat yield response to nitrogen 25

III. ANOVA for linear model for wheat yield response to nitrogen 25

IV. Sensitivities to community size (m) of the genetic algorithm 40

V. Sensitivities to epochs of the genetic algorithm 40

VI. Sensitivities to mutation rate of the genetic algorithm 41

VII. Sensitivities to the architecture of ANN using genetic algorithm training 41

VIII. The best parameters combination of ANN training using GA for fertilizer use 42

IX. Sensitivities to architecture of ANN using QuickProp training 43

X. Sensitivities to learning rate of QuickProp algorithm for 1:2:1 ANN training 43

XI. Sensitivities to epoch size of QuickProp algorithm for 1:2:1 ANN 44

XII. Test data for Wheat Yield Response to Nitrogen 45

XIII. Test Results of GA for Wheat Yield Response to Nitrogen 46

XIV. Test Results of QuickProp for Wheat Yield Response to Nitrogen 47

LIST OF FIGURES

Figure Page

1. Three-layer Artificial Neural Network architecture 13
2. Design structure of FUDSS 1.0 22
3. Quadratic regression of wheat yield response to nitrogen 26
4. Linear regression of wheat yield response to nitrogen 26
5. Flowchart of Genetic Algorithm 29
6. Back propagation training interface 35
7. Performance of the Genetic Algorithm (1:2:1) ANN 47
8. Performance of the QuickProp algorithm (1:2:1) ANN 48
9. Performance of quadratic regression model 49
10. DBMS for Soil, Water, Forage Analytical Lab 50
11. Knowledge Base of nitrogen requirement for 6 main crops 51
12. Decision making using soil test database 52
13. Decision making using manual input 53
14. Query soil test results for customers 54
15. Main form of Decision Support System for Fertilizer Use 55
16. Menu system of DSS for Fertilizer Use 55
17. Message box reminds user about yield goal range 56
18. Message box shows sample situation 57
19. ComboBox lists all sample number for a customer 58
20. Form for printing daily reports 58

CHAPTER I

INTRODUCTION

1.1 Problem

Nutrient deficiencies in soil can be corrected simply by applying appropriate fertilizers, but it is neither practicable nor economic to attempt to eliminate all deficiencies and maximize crop production by massive applications of fertilizer. Also, application of excess nutrients can result in environmental risk. The appropriate amount of fertilizer, after all such consideration, is termed optimal, meaning that it works the best under all the relevant circumstances.

OSU Soil, Water and Forage Analytical Laboratory (SWFAL) established by the Oklahoma Cooperative Extension Service provides quality soil, forage and water testing for the state of Oklahoma. Each year over 35,000 samples are analyzed for thousands of farmers, ranchers, homeowners, government agencies, and researchers. In order to increase fertilizer use efficiency and to protect the environment, SWFAL gives farmers recommendations on the optimal rate of fertilizer applications based on their soil testing results and yield-goal demands.

It is practically impossible for fertility experts to work out thousands of recommendations manually, so computer software that can generate recommendations automatically is needed by the laboratory to support Oklahoma farmers in making agronomically, economically, and environmentally correct ground fertilizer application decisions. This software is called a Decision Support System (DSS). It consists of a user interface for communicating with a user, a database for storing soil test data, and an expert system for generating recommendations for optimal fertilizer use. The most important part in an expert system is knowledge acquisition. There are a few methods available to complete this task. Among them, Artificial Neural Network (ANN) has been adopted extensively.

1.2 Objectives

The objectives of the study are:

(1) To design and implement a DSS, named FUDSS 1.0 (Decision Support System for Fertilizer Use version 1.0), for OSU Soil, Water and Forage Analytical Laboratory to give farmers a decision making aid in managing fertilizer use.

(2) To investigate and compare the ability of two distinct ANN training algorithms in learning the rule of optimal fertilizer application.

1.3 Organization

In this chapter we introduce the problem in decision making on fertilizer use and the limitation of generating recommendations of optimal fertilizer application manually. We suggest the benefits of employing computer software to help generate recommendations. In chapter two we review the role and effect of using a decision support system as well as expert system as an aid of decision making. We describe the principle of ANN and the relationship between ANN and statistical methods in detail. The relational database system is discussed in this chapter as well. Chapter three presents concrete design and implementation of DSS for fertilizer use. Also, in this chapter, we expound the methods of knowledge acquisition and describe two distinct ANN training algorithms, a Genetic Algorithm (GA) and the QuickProp algorithm. Chapter four provides an analysis of different ANN training algorithms and compares their performance on learning the rule of optimal fertilizer use. The functions of an expert system with a user friendly interactive interface is tested in this chapter as well. Chapter five summarizes the study and draws conclusions.

CHAPTER II

LITERATURE REVIEW

2.1 Decision Support System

A Decision Support System (DSS) is a computer program that provides information, in a given domain of application by means of analytical decision models and access to a database in order to support a decision maker effectively in making decisions for complex and ill-structured (non-programmable) tasks (Klein, Michel, 1995).

Decision support systems (DDSs) help their users by answering complicated questions and providing information required for effective planning and organizing (Simon, H., 1980).

Historically oriented to business managers, a wider range of people use DSSs today to reach decisions in ambiguous and complex environments. DSSs provide current data to use in problem analysis as well as forecasts of future conditions. DSSs grant their users the ability to analyze alternative choices quantitatively. By modeling a complex set of circumstances, the analyst or decision maker can manipulate various parameters to assess the impact of diverse conditions.

The benefits of DSSs include improved decision making through better understanding of the organization, an increased number of decision alternatives examined, the ability to implement ad hoc analysis, and faster response to expected situations. DSSs are expected to result in improved communication, more effective teamwork, better control, and time and cost savings.

Traditionally, a full-featured decision support system consists of three major components:

(1) databases,

(2) model bases,

(3) user interfaces.

The databases provide access to internal and external data that may influence or be affected by decisions under consideration. DSSs also use data from the databases to form a baseline that statistical models used extrapolate (and otherwise project) from past to present to future conditions. The DSS also may use these data to determine or estimate parameters of models used for forecasting. Such a DSS would feature a wide ranging and large number of discrete but interrelated databases. These databases would be developed and managed by a diverse set of owners. Ideally, the DSS would access databases stored on a network of servers. This approach offers the most efficient method if the database contains non-replicated data, carries minimized redundancy, and pursues a principle of leaving the source database at the site of the owner for ease of maintenance and updating.

The DSS can be integrated with an Expert System (ES) to extend its capabilities (Efraim Turban, 1985). Its knowledge base provides information about the relationships among data that are too complex for a database to represent. The knowledge base consists of rules that can constrain possible solutions as well as alternative solutions and methods for evaluating them. Today, these relationships are increasingly housed within the model base and/or table-oriented databases.

A DSS model base has been defined to include an array of spreadsheets, simulation packages, forecasting tools, and statistical packages. While this framework continues in DSS constructions today, more attention is given to sets of alternative cause and effect models which integrate all of these types of tools. The user has access to models that have been developed previously and that can be reused.

The DSS must include a sophisticated user interface, which allows users to control which data and models to include in their analyses. The DSS must be designed to support the greater freedom users experience in manipulating data and processing information. A DSS should allow easy data and knowledge assembly from a variety of sources as well as using as inputs to previously developed models or models that currently analyze many alternatives and extensive data about each alternative. A high-quality DSS should utilize facilities to compare, contrast, and aggregate data in a wide range of graphical and tabular formats.

2.2 Expert Systems

An expert system is one of the most commonly used computer tools for making decisions. An expert system is defined to be "…computer system (hardware and software) that simulates human experts in a given area of specialization" (Castillo, Gutierrez, and Hadi, 1997). The reason expert systems became popular in the 1990's is because of the very high level of expertise required to address increasingly complex problems. These expert systems, if designed properly, contain an extremely high level of human expertise required to make potential life and death decisions. This, in turn, gives a high level of expertise to everybody who uses the expert system appropriately and responsibly.

Expert systems are a part of a field called artificial intelligence. The official beginning of artificial intelligence was at a convention at Dartmouth College in 1956 (Walker and Miller. 1986). A prediction made at this conference was that within the next 25 years everyone would be involved in recreational activities, while their computers back at the office would do all the work (Walker and Miller. 1986).

The development of artificial intelligence slowed during the 1960's and 1970's when programs that became more difficult and complex than originally thought were installed on computers lacking power and speed. Through these lean years, research continued and successful artificial intelligent programs or systems were developed in the early 1980's. The major advancement that spurred this new growth in artificial intelligence was the rapid growth and power of the modern computer. As a result, there are numerous types of artificial intelligent programs in use in commercial businesses and industry. The majority of the artificial intelligence and expert systems are being used in the business world, but increasingly there are significant examples in other fields (Castillo, Gutierrez, and Hadi, 1997).

Expert systems are based directly on knowledge created or derived from their human counterparts. This process is called knowledge engineering (Williams, 1986). Knowledge engineering involves four components:

(1) define the expert system domain,

(2) elicit desired information from human experts,

(3) incorporate that knowledge into acceptable form for the knowledge base, then

(4) test the system to evaluate its robustness and accuracy (Williams, 1986).

The knowledge base contains facts, rules, and relationships derived from expert experience relating to specific situations. Once the knowledge base has been established the computer-based expert system must simulate the thinking process of a human expert. The knowledge base must then use an inference engine to apply the knowledge base efficiently. The inference engine automatically identifies and evaluates any knowledge from the knowledge base that would be applicable to each given situation. This process of going through the knowledge base and the inference engine is not always easy. The knowledge engineer must incorporate not only general rules about a situation, but also the specialized rules associated with special circumstances. Additional information can be added as the errors are detected. The problem is that the errors may not be caught until the system has been sent to the public. For this reason, expert systems have become iterative processes using successive steps of multiple refinement. (Williams, 1986)

Expert systems offer several advantages to an organization that uses them. The expert system allows the design engineer to make decisions with the most current knowledge as long as the computer program is updated. If the expert system is not updated continuously, the system still can be useful. Updates normally involve minor changes in the decision tree or the addition of new processes. As a result, the updated version provides only a small change in the output. If the system is not updated over a period of time, then the small changes aggregate to make the system out of date. Expert systems are capable of outperforming their human counterparts that become bored, forgetful, or sometimes develop tunnel vision (Williams, 1986). The speed of the decision making process used in expert systems via the computer is far superior to a human. The limitation is that the computer is only as smart as the human who created the programs. Expert systems also are susceptible to any wrong information that is input. Human experts have two qualities that supersede the computer. Humans have considerable knowledge in specific areas and can have effective strategies for quickly sorting through knowledge when faced with problems (Williams, 1986). Clearly, the use of an expert system in association with a human expert can improve the decision making process.

2.3 Artificial Neural Networks

There is no universally accepted definition of an Artificial Neural Networks (ANN). But perhaps most people in this field would agree that an ANN is a network of many simple processors ("units"), each possibly having a small amount of local memory. The units are connected by communication channels ("connection"), which usually carry numeric (as opposed to symbolic) data, encoded by any of various means. The units operate only on their local data and on the inputs they receive via the connections. The restriction to local operations is often relaxed during training (Haykin, 1994).

Neural networks have a history that dates back to the early 1940's. The first mention of a neural network was in an article by McCulloch and Pitts in 1943 (McCulloch and Pitts, 1943). Their paper started the examination of how the human brain functions. They believed that the brain exists as a series of interacting parts, which evolve continuously, and whose sole function is dependent upon the connection of these intermediate parts (Wilson and Sharda, 1992). This paper was given credit for starting the research for processing systems of data based on the function of the human brain. As a result, their research was the first ever done on the neural network.

Donald Hebb, a psychologist, further advanced the science of the human brain with a new theory in 1949 (Hebb, 1949) that stated that the brain operates by a collection of neurons called assemblies. These neurons process the brain's impulses and convert them into an appropriate behavior. Hebb also believed that the neurons were only interconnected by what he termed self-organization. He believed that the interconnections between neurons helped strengthen the pathways involved in brain functions (Wilson and Sharda, 1992).

In 1957, a scientist named Rosenblatt made the next major stride in the neural network field by developing one of the first learning systems (Rosenblatt, 1958). The system was a neuron-like device called the "Perceptron." This was the first neuron type program or machine that could learn from data input.

The biggest break in the early history of the neural network occurred in 1960 when two electrical engineers, Widrow and Hoff, described the first neuron based computer (Widrow and Hoff, 1960). This neuron based computer is the foundation for the modern neural network training of computer programs (Wilson and Sharda, 1992).

An opposite view was published in a book called "Perceptrons" co-authored by Minsky and Papert in 1969 (Minsky and Papert, 1969). This sent ripples through the scientific community and they seriously disputed the earlier claims of Rosenblatt. A lack of technology further fueled this argument. Computers in the 1960's and 1970's were severely limited in speed and capacity. They were large but very slow by today's standards. As such they could not process problems of the complexity and sophistication posed by typical neural network approaches. Federal research money was temporarily directed away from neural network topics, resulting in a thirteen-year period where little or no additional development occurred (Wilson and Sharda, 1992).

In 1982, John Hopefield restarted neural network research and restored its credibility to the scientific community (Hopfield, 1982). His work proved that neural networks were able to solve various types of problems from simple to complex by applying a learning system program that was trained to learn.

In 1986, Rumelhart and McCelland derived a learning algorithm for Rosenblatt's Perceptron device (Rumelhart and McClelland, 1986). The algorithm was developed from the work performed by Rosenblatt as well as Windrow and Hoff in the late fifties and early sixties. This learning algorithm was termed a "back propagation" network. The back propagation algorithm had become the accepted standard process in training the multi-layer, feed-forward neural network. In the end, the perseverance of these scientists and engineers has given the technical community a very innovative way to solve modern complex problems (Wilson and Sharda, 1992).

In practice, the standard back propagation algorithm is too slow in converging, and it has a problem of falling into local minimum of the objective function. This encouraged considerable research on methods to accelerate the convergence of the algorithm (Hagan, et al., 1996).

The research on faster algorithms falls roughly into two categories. The first category involves the development of heuristic techniques, which include such ideas as varying the learning rate, using momentum and re-scaling variables. There are many different approaches for heuristic techniques, such as the variable learning rate backpropagation algorithm (Vogl, Mangis, etal., 1988), Delta-Bar-Delta algorithm (Jacobs, 1988), and QuickProp algorithm (Fahlman, 1988). Another category of research has focused on standard numerical optimization techniques, such as Conjugate Gradient algorithm and Levenberg-Marquardt algorithm (Hagan, etc., 1996).

The research on global minimum algorithms are most focused on simulated annealing algorithms (Ingber, 1993) and Genetic Algorithms (Holland, 1975). There are several ways to combine Genetic Algorithms with a neural network. So far, Genetic Algorithms have been used to:

(1) generate the weights of a neural network,

(2) generate the architecture of a neural network,

(3) generate both the architecture and the weights of a neural network simultaneously,

(4) analyze a neural network.

As soon as a genetic algorithm is used to generate the weights of a neural network, then the learning of an ANN is formulated as a weight optimization problem, usually using the inverse mean squared error as fitness measure.

On the simplest level, a neural network models a human brain with many simple elements (neurons) that work together in parallel. Figure 1 shows how the neural network attempts to simulate the human brain. The neurons in the human brain are setup in parallel and are connected to each other. A more complex look at the neurons would show multiple layers of these neurons that would be interconnected. Also Figure 1 illustrates how the computer based structure of neural network works. The neurons that make-up each layer are connected to each other to create a network of neurons.

The "**X**'s" in figure 1 represent the input layers of a neural network. The information is then spread throughout the network through the hidden layers that are represented with the "**h**'s". After the information is processed in the hidden layers, the information goes into the output layers defined as "**O**'s". The difference between the biological process of the brain and the artificial processes associated with a computer must be dealt with to produce an effective artificial neural network.

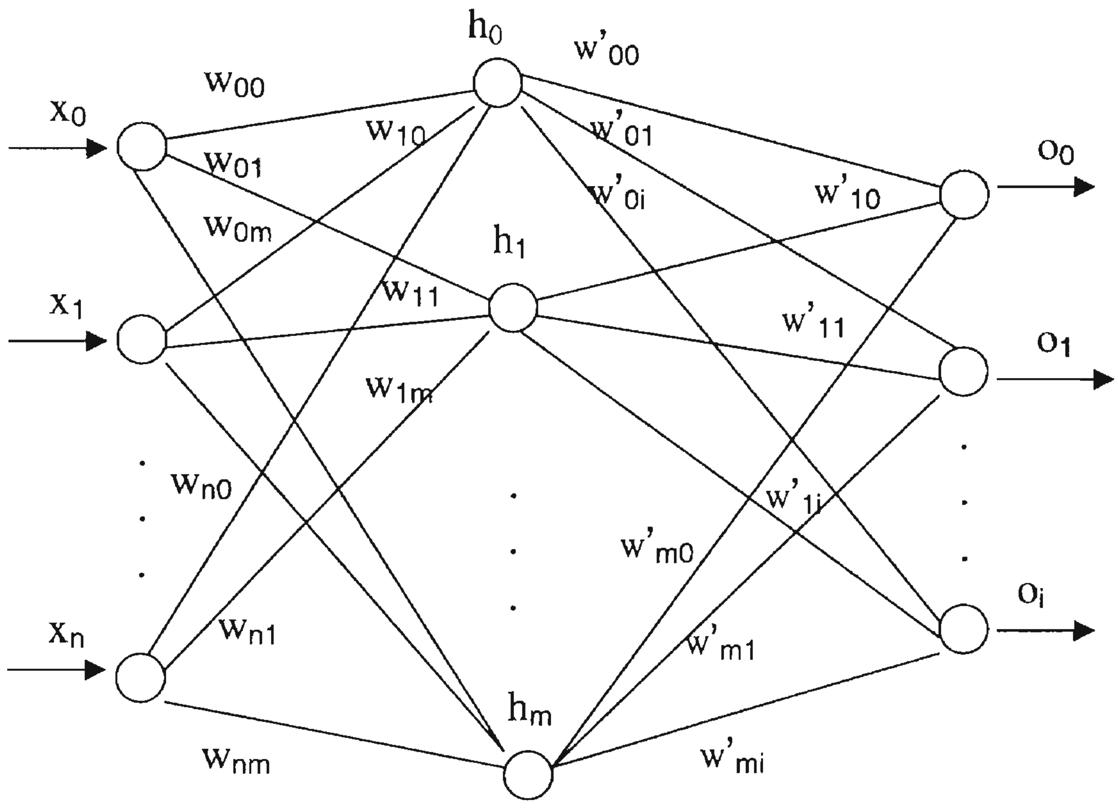


Figure 1. Three-layer Artificial Neural Network architecture

Neural networks can have one or more layers of neurons. The networks normally allow the user to have a range of 2-6 layers. The layers are separated into three different categories. The first layer of neurons in the network is termed the input layer. Input neurons are used to define the problem that is to be solved. The last layer of neurons is the output layer. The layers between the input and output layers are called hidden layers. Hidden layers are used by the neural network to process the inputs while predicting the outputs. These layers are responsible for all the numerical computing and pattern association that is needed to convert or model the input information into the desired output. The layers can be interconnected together, or only connected on each individual layer. The connections between all neurons carry weights that contain the knowledge of the system. The neurons then sum the obtained information from other surrounding neurons or external input stimuli (Wilson and Sharda, 1992). This information is then transferred by each neuron in a non-linear fashion ultimately to achieve a trained system. The neurons with the summed information will then internally process the information and distribute the information to the next appropriate neuron or to the external output. The exact way in which the neural network operates is based on the structure of the computer code (Hagan, Demuth and Beale, 1996).

Shih reported that most neural networks can be resolved with three-layer systems (Shih, 1994). The neural network program used in this research project has a default size of three layers. System processing speed may be able to increase with the addition of more neurons and/or hidden layers. In some cases, too few layers and neurons in the neural network can cause the network to stall and never reach the desired goal of 100% pattern recognition. Pattern recognition occurs when the neural network produces the same results as the target results that were input into the network by the user. The reason is that too few neurons force the network to place a large amount of information into an individual neuron resulting in a loss of efficiency. In the case of too many neurons, where there is so little information in an individual neuron, it is almost a waste of space.

This, in turn, will slow down the network.

A neural network can be trained by either supervised or unsupervised training (Wilson and Sharda, 1992). Supervised training requires the user to describe both the problem domain and the answer (desired output). This allows the neural network to learn that this particular input should result in this output. A control group is required to teach the network the appropriate learning response from the training. The control group that contains data with known inputs and outputs is used as a calibration instrument for the network. After the input is entered, the neural network process the inputs and the given output until the system can train itself to produce the desired output. Training starts when the network is given a problem and higher control neurons try to apply the hidden neurons to generate the expected answer. The specific learning algorithm that is used during the training determines how the neuron's interconnections are weighed. The algorithm will then correct the weights of the neurons due to offsets from the actual desired outputs. The network is continually updating the system until the desired outputs are achieved. In other words, after the calibration is completed, the system adjusts the weighted values of the neurons so that when a given input is received, the correct output is calculated (Gifford, 1997).

2.4 Relation between ANN and statistical methods

There is considerable overlap between the fields of neural networks and statistics. Statistics is concerned with data analysis. In neural network terminology, statistical inference means learning to generalize from noisy data. Some neural networks are not concerned with data analysis (e.g., those intended to model biological systems) and therefore have little to do with statistics. Some neural networks do not learn (e.g., Hopfield neural networks (Hopfiel, 1982)) and therefore have little to do with statistics. Some neural networks can learn successfully only from noise-free data (e.g., ART or the perceptron rule (Hagan, 1996)) and therefore would not be considered as statistical methods. But most neural networks that can learn to generalize effectively from noisy data are similar or identical to statistical methods, such as:

(1) Feed-forward nets with no hidden layer (including functional-link neural nets and higher-order neural nets) basically are generalized linear models.

(2) Feed-forward nets with one hidden layer are related closely to projection pursuit regression (PPR).

(3) Probabilistic neural nets are identical to kernel discriminant analysis. Kohonen nets for adaptive vector quantizations are very similar to K-means cluster analysis.

(4) Hebbian learning is closely related to principal component analysis (PCA).

Feed-forward nets are a subset of nonlinear regression and discrimination models. Statisticians have studied the properties of this general class, but did not consider the specific case of feed-forward neural nets before such networks were popularized in the neural network field. Still, many results from the statistical theory of nonlinear models apply directly to feed-forward nets, and the methods that commonly are used for fitting nonlinear models, such as various Levenberg-Marquardt and Conjugate Gradient algorithms, can be used to train feed-forward nets. The application of statistical theory to neural networks was explored in detail by Bishop (1995) and Ripley (1994). Among the many statistical concepts important to neural nets is the bias/variance trade-off in nonparametric estimation, discussed by Geman, Bienenstock, and Doursat, R. (1992).

While neural nets are often defined in terms of their algorithms or implementations, statistical methods are usually defined in terms of their results. The arithmetic mean (expected value), for example, can be computed by a (very simple) back propagation neural net, by applying the usual formula,

$$E(x) = \frac{\sum_{i=1}^{n} x_i}{n}$$

or by various other methods. The result is still an arithmetic mean regardless of how it is computed. A statistician would consider standard back propagation, Quickprop, and Levenberg-Marquardt to be different algorithms for implementing the same statistical model such as a feed-forward net. On the other hand, different training criteria, such as least squares and cross entropy, are viewed by statisticians as fundamentally different estimation methods with different statistical properties.

It is sometimes claimed that neural networks, unlike statistical models, require no distributional assumptions. In fact, neural networks involve exactly the same sort of distributional assumptions as statistical models (Bishop, 1995), but statisticians study the consequences and importance of these assumptions. For example, least-squares training methods are widely used by both statisticians and neural networkers.

2.5 Relational Database System and SQL

A database is a collection of information that exists over a long period of time, often many years. The term database refers to a collection of data that is managed by a database management system, also called a DBMS, or just database system (Ullman, 1997). A DBMS is expected to:

(1) Allow users to create new databases and specify their schema (logical structure of the data), using a specialized language called a data-definition language.

(2) Give users the ability to query the data and modify the data, using an appropriate language, often called a query language or data-manipulation language.

(3) Support the storage of very large amounts of data – gigabytes or more – over a long period of time, keeping it secure from accident or unauthorized use and allowing efficient access to the data for queries and database modifications.

(4) Control access to data from many users at once, without allowing the actions of one user to affect other users and without allowing simultaneous accesses to corrupt the data accidentally.

The first commercial database management system appeared in the late 1960's. It evolved from the systems, which provide some of expanded version of the above. File systems store data over a long period of time, and they allow the storage of large amounts of data. However, file systems generally do not guarantee that data cannot be lost if it is not backed up, and they do not support efficient access to data items whose location in a particular file is unknown. Further, file systems do not directly support item (2), a query language for the data in files. Their support for (1) – a schema for the data – is limited to the creation of directory structures for files. Finally, file systems do not satisfy (4). When they allow concurrent access to files by several users or processes, a file system generally does not prevent situations such as two users modifying the same file at about the same time, so the changes made by one user fail to appear in the file.

The first important applications of DBMS's were ones where data was composed of many small items, and many queries or modifications were made. Some of these applications include:

(1) Airline Reservation Systems,

(2) Banking Systems,

(3) Corporate Records.

The relational database model is the current database standard. The user/designer needs be concerned only with the logical view of the database; the details of physical storage, access paths, and data structures are managed by the (RDBMS). Therefore, relational database design becomes much simpler than hierarchical or network models. A relational database is perceived by the user as a collection of tables in which data are stored. Each table consists of series of row/column intersections. Tables (or relations) are related to each other by sharing a common entity characteristic. The relationship type is often shown in a relational schema. A table provides complete data and structural independence because it is a purely logical structure (Ullman, J. D. 1997).

The relational database model has the following advantages:

(1) Because the relational model achieves both data independence and structural independence, it becomes much easier to design the database and to manage its contents.

(2) Less programming effort is required because the relational database has a very powerful query language called Structured Query Language (SQL), which makes ad hoc queries possible.

The disadvantages of relational database model are: RDBMS requires both substantial hardware and substantial operating system overhead.

Today, most database systems are based on the relation model of data, which organizes information into tables. SQL is the language most often used in these systems. Structured Query Language (SQL) was developed in the early 1970's at IBM for use with relational databases. The language was standardized in 1986 by ANSI (American National Standards Institute). SQL has an updated standard adopted in 1992, called SQL-92 or SQL2. An important core of SQL is equivalent to the relational algebra, although there are many important features of SQL that go beyond what is found in the relational algebra, for example, aggregation (e.g., sums, counts) and database updates. The most commonly used relational database system both queries and modifies the database through SQL (Ullman,1997).

CHAPTER III

PROJECT DEVELOPMENT AND METHODOLOGY

3.1 Design of FUDSS 1.0

FUDSS 1.0 is designed as a decision making tool to help user to interpret the results of soil tests and to determine fertilizer application rates based on yield goals. For these purposes, FUDSS 1.0 characterizes an Intelligent Decision Support System (IDSS) that carries three components (Figure 2):

(1) The user friendly ***interactive interface***

The system through interface asks user following questions in natural language:

a) "What is your soil test sample number?"

b) "What kind of crop will you plant?"

c) "What is your yield goal?"

The interface should allow user change their selection at any time and show the answers quickly.

The outputs of FUDSS 1.0 are:

a) The interpretation of soil test results in table format

b) The recommendation of optimal fertilizer application rate based on yield goal

(2) ***An Expert System*** (ES)

Expert System determines the fertilizer application rate based on yield goal. It consists two parts:

a) *Knowledge Base* (KB) that contains the facts, rules, and relationships derived from fertility experts and soil scientists.

b) *Inference Engine* (IE) that automatically identifies and evaluates any knowledge from knowledge bases that would be applicable to each situation.

(3) ***A Database Management System*** (DBMS)

A database management system stores, accesses and queries the information, such as soil test results, customers' information, invoice, fertilizer products, and knowledge bases.

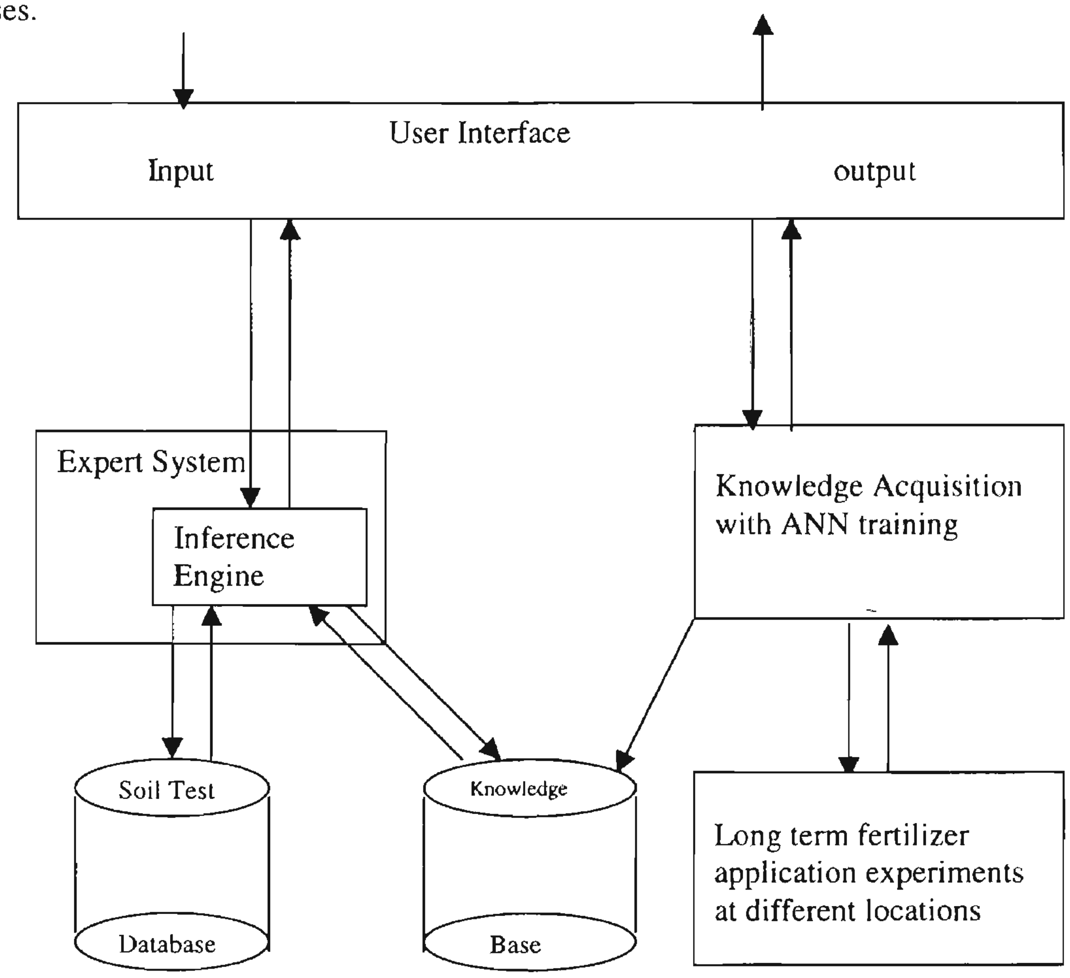

User Interface
Input
output
Expert System
Inference Engine
Knowledge Acquisition with ANN training
Soil Test
Database
Knowledge
Base
Long term fertilizer application experiments at different locations

Figure 2. Design structure of FUDSS 1.0

3.2 Expert System Development

It is difficult to decide whether or not to fertilize, or the amount of fertilizer to apply to a specified field in a farm. In order to help farmers to address this problem, the design of FUDSS 1.0 contains a program component that uses knowledge and procedures to deal with the complicated problems at a level of a professionally trained human, such as a fertility expert. To develop this expert system, the most important work is how to build a knowledge base that contains the scientific knowledge and experience in the soil fertility area of expertise. The first subtask in building a knowledge base is knowledge acquisition that refers to the process of collecting, organizing, and formalizing the information, concepts, and strategies. The second subtask is the representation of a large amount of knowledge in a fashion that permits its effective use. This is known as the knowledge representation problem, one of the hardest problems so far identified in Artificial Intelligence (AI) (Jackson 1986). These subtasks are discussed in detail below.

3.3 Knowledge Acquisition

Some knowledge extraction methods seek to collect knowledge from experts. But, some knowledge also exists in examples. That is, there is much that can be inferred from prior situations, then applied to new or similar circumstances.

To estimate fertilizer requirements, soil scientists usually carry out some exploratory experiments with fertilizers, which shows the importance of fertilizer applications for crop production. The results of fertilizer experiments could be represented in the form of regression equations that can be used to calculate estimates of yields for various fertilizer application rates (Colwell, 1994). However, the fertilizer requirements of crops vary, often markedly, from site to site and from year to year because of variations in the soil, seasonal growing conditions, agricultural practices, and so on. There is no general regression equation model that could apply to every region for estimating fertilizer requirements of crops. It is necessary that every region should carry out fertilizer experiments to find out the model that is applicable to its environment conditions. Based on many years' field calibrations conducted in Oklahoma, the relationships of crops' yield and fertilizer requirements in Oklahoma have been determined by OSU soil fertility scientists and reported as OSU extension facts F-2225 (Table 1). It presents the relation between nitrogen requirements for selected crop and yield goal.

Table 1. Nitrogen Requirements for Small Grain Crops

Yield Goal (bu/A)			Nitrogen
Wheat	**Barley**	**Oats**	**(lbs/A)**
15	20	25	30
20	25	35	40
30	35	55	60
40	50	70	80
50	60	90	100
60	75	105	125
70	90	125	155
80	100	140	185
100	125	175	240

Generally there is a non-linear relation between nitrogen requirements and crop yield. Unfortunately, it is hard to find a suitable non-linear regression equation to represent this relation sometime because the regression model and the individual regression coefficient is not significant at the level we deem appropriate (99%, or $\alpha = 0.01$) (Colwell, 1994).

From Table 1, the fitted regression line relating wheat yield and nitrogen requirements was computed using a quadratic regression model and is given by

$Y = -0.5286 + 0.5450X - 0.0005416X^2$. The residual estimate of variance is calculated as $s^2 = 1.36$ (lbs/acre). The value of F reaches 2391.45, which is significant at a level less than 0.0001 (Table 2). Thus null hypotheses on slope (H_0:$\beta = 0$) is rejected in favor of H_1:$\beta \neq 0$, with the conclusion that there is, indeed, statistically significant quadratic trend between nitrogen application and yield of wheat. From statistics, the quadratic regression equation for wheat yield response to nitrogen is acceptable (Myers, 1989).

Table 2 ANOVA for Quadratic Model of Wheat Yield Response to Nitrogen

Source	DF	Sum of Squares	Mean of Squares	F	$F_{0.01}$
Regression	2	6491.86	3245.93	2391.45	12.25
Residual	6	8.14	1.36		
Total	8	6500.00			

Table 3 ANOVA for Linear Model of Wheat Yield Response to Nitrogen

Source	DF	Sum of Squares	Mean of Squares	F	$F_{0.01}$
Regression	1	6446.39	6446.39	841.80	12.25
Residual	7	53.61	7.65		
Total	8	6500.00			

From Table 1, the fitted regression line relating wheat yield and nitrogen requirements was computed using a linear regression model and is given by $Y = 6.1099 + 0.4039X$. The residual estimate of variance is calculated as $s^2 = 7.66$ (lbs/acre). The value of F reaches 841.80, which is significant at a level less than 0.0001 (Table 3). Thus null hypotheses on slope (H_0:$\beta = 0$) is rejected in favor of H_1:$\beta \neq 0$, with the conclusion that there is, indeed, statistically significant quadratic trend between nitrogen application and yield of wheat.

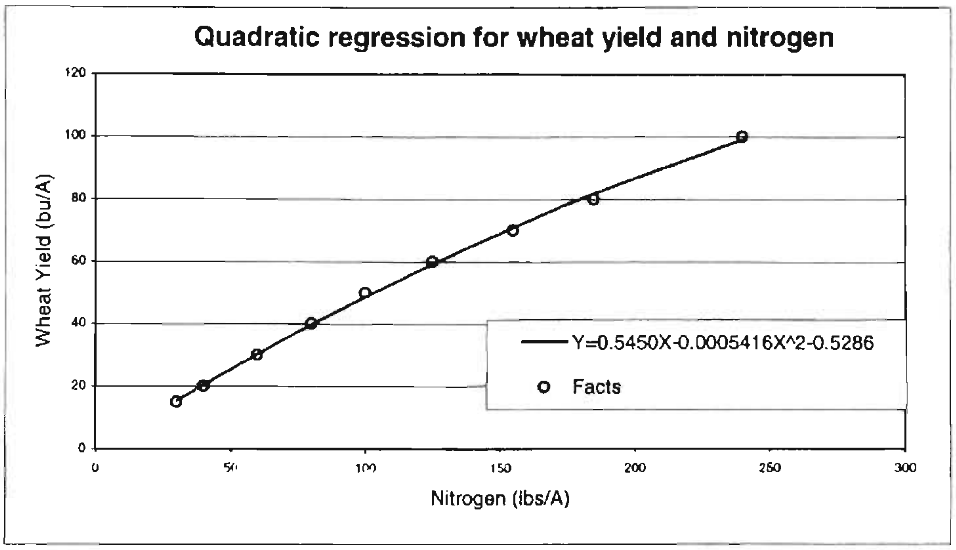


Figure 3. Quadratic regression of wheat yield response to nitrogen

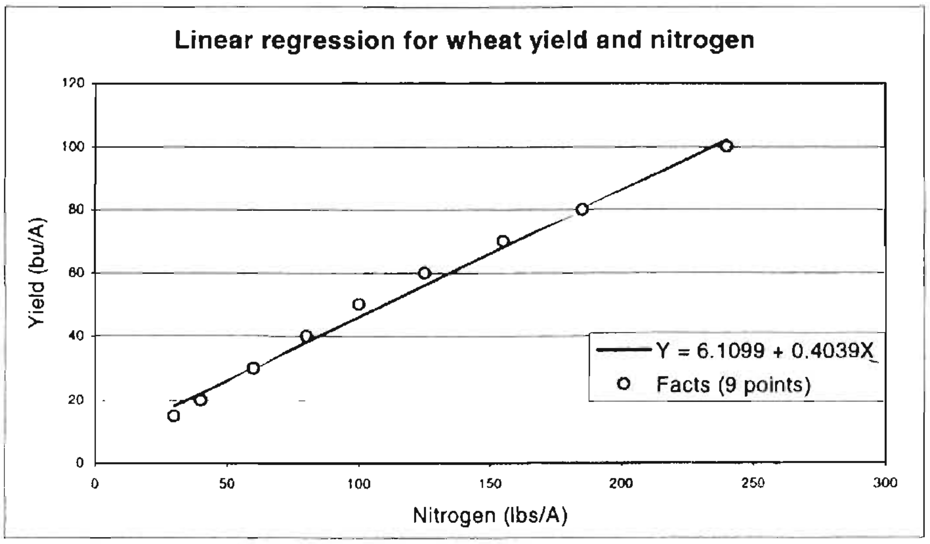


Figure 4. Linear regression of wheat yield response to nitrogen

Using runs test for models' fitting evaluation (Bennett and Franklin, 1954), the linear model gives a sum of squares of 53.605, a standard deviation of 2.767, and three runs of signs in the residuals. For a fit to nine data points (Table 1), the maximum number of runs is nine and expected number of runs is four or five for good fit. So the linear fit is not very good (Figure 4). The quadratic fit gives a sum of squares of 8.1438, a standard deviation of 1.165, and four runs. The fit is acceptable (Figure 3).

The method of regression in statistics describes the relation between nitrogen requirements and wheat yield as a quadratic equation or linear equation, and its prediction of the fertilizer application rate is based on the points that locate only on the line. As a result, some points that deviate from the line, but represent the values in fact, are inappropriately treated as errors and discarded.

One solution to this problem is to use chopping. This method employs slopes between every adjacent pair of points to do approximate predication, since the chopped but contiguous lines take a shape of zigzag line that approaches the curve formed by the values in fact. The disadvantage of the chopping method is that it cannot be modeled.

ANN is able to eliminate the disadvantages of both previous methods. It can learn patterns from examples and reproduce them as a prediction tool. This capability is important because there are many problems in which one knows what should be the correct result, but it is not possible to lay down a precise procedure or set of rules for finding the result. As with other predictive tools (such as statistics), it is important that ANN be trained with sufficient systematic variation. However, the difference between ANN and other predictive tools is that the former can handle more noise or unsystematic variation than the latter. Compared with other methods, ANN is more useful in knowledge acquisition. In this project, the data from long term soil fertility experiments conducted in Oklahoma (Table 1) were used to train an ANN in acquiring the knowledge about optimal fertilizer application. The trained ANN could be employed as a substitute for a non-linear regression model.

3.4 ANN Training Algorithms

1. Modified Genetic Algorithm Genetic algorithms were developed by John Holland in the 1970's (Holland, 1975). They are based on a Darwinian-type survival of the fittest strategy, whereby potential solutions to a problem compete and mate with each other in order to produce increasingly stronger individuals. Each individual in the population represents a potential solution to the problem that is to be solved. These individuals are represented in the genetic algorithm by means of a linear string, similar to the way genetic information in organisms is coded onto chromosomes. In GA terminology the members of a population therefore are referred as chromosomes. Chromosomes are assembled from a set of genes, that are generally characteristics belonging to a certain alphabet A. A chromosome can be thought of as a vector x consisting of l genes a_i:

$$x = (a_1, a_2, \ldots, a_l), a_i \in A_i$$

l is referred to as the length of the chromosome. Commonly, all alphabets in a chromosome are the same $A = A_1 = A_2 = \ldots = A_l$. The alphabets commonly used today are the cases of binary genes ($A = \{0, 1\}$), and real–valued genes (A = {Real number}), In the latter case, the real value can be stored in a gene by means of a fixed or floating point representation or by a conversion to an integer.

Figure 5 shows a flowchart representation of a simple genetic algorithm that depicts the evolution of population of individuals. A GA operates on a population of decision variable sets called " string," or " chromosomes." Each string made up of a series of characters, or called " genes." The characters represent a coding of the decision variable set. The values of the genes are chosen initially at random. Each possible value has an equal probability of being selected for each gene. The number of strings in simple

GA population can vary, but is typically in the range from 30 to 100 (Goldberg, 1989). After a population of strings is initialized, a series of three genetic operators are applied to the population: (1) selection, (2) crossover, and (3) mutation. Application of the mutation operator marks the completion of a GA cycle. A single GA cycle is also known as a "generation." A GA is usually allowed to run for a specified number of generations, or until some stopping criterion, such as convergence of the population to a single solution, is met.

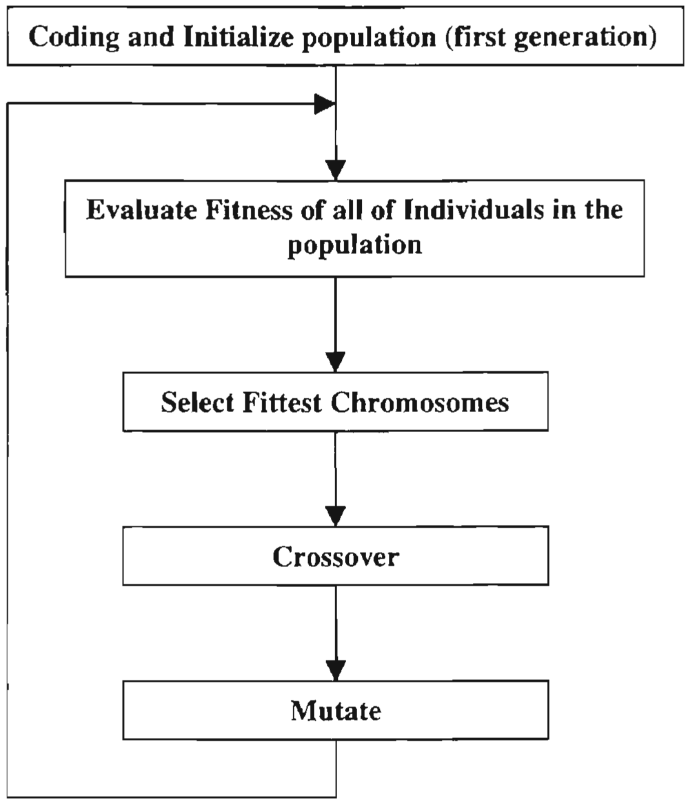

Coding and Initialize population (first generation)
Evaluate Fitness of all of Individuals in the population
Select Fittest Chromosomes
Crossover
Mutate

Figure 5. Flowchart of Genetic Algorithm

In this thesis we use a modified GA that gives offspring a chance to compete with its parents for survival and has a new coding method instead of binary string (Yang, 1997). The algorithm is summarized as follows:

Objective function:

$$f = f(x_1, x_2, \ldots, x_n) \qquad a_i \leq x_i \leq b_i, \; i = 1, 2, 3, \ldots, n$$

where x_i is the ith variable, and a_i and b_i are the limits of that variable.

Step 1) Initialization: randomly select m distinct points (community size) of that variable from the search space with equal probability. That is, generate a random number subject to a uniform (0, 1) distribution. Calculate the value of the first variable by

$$x_i = a_i + x_i(b_i - a_i)$$

Do the same for the other variables. These steps identify the first point in the search space. Repeat this procedure m times.

Step 2) Fitness Evaluation: Calculate the objective function values at the m points.

Step 3) Point ranking: Sort the m points in order of descending objective function value, so that the first point represents the worst and the last point represents the best.

Step 4) Assigning probabilities to each point: Each of the points is assigned a probability p_i, i = 1, 2, …, m, giving higher probabilities to the points with lower function value and lower probabilities to those with higher function values. Thus, the best point has the highest probability p_1, while the worst point has the lowest probability p_m. The other points have probabilities ranging from p_m to p_1. The following linear relation can be

$$p_i = p_m + \frac{(i-1)}{(m-1)}(p_1 - p_m), \qquad i = 1,2,3,\ldots\ldots, m-1$$

used. Probability is nonnegative and the total probability should sum to one.

Step 5) Selecting parents: randomly select two points from the m points according to the probability p_i. Make sure that the two points are not identical.

Step 6) Crossover: For each of the genes (variables), randomly select one value from the corresponding two selected points to construct a new point.

Step 7) Mutation: Occasionally, with a small probability p_m (mutation rate), alter the newly created point. To do this, for each of the genes of the newly created point, generate a random number r, if $p_m > r$, replace the value of that gene by another uniformly distributed (0, 1) random number.

Step 8) Repeat k times Step 5 through 7 so that m new points are generated.

Step 9) These new points produced by genetic methods represent the offspring population and are going to compete with their parents.

Step 10) Sort the newly created points into descending order.

Step 11) If the best point of the new generation (the last one) is not better than the best one of the old generation, then replace the worst point of the new generation by the best point of the old generation and re-sort them. This step is to ensure that the current best-so-far point in the community is always retained.

Step 12) Start from the second-best point of the new generation and compare it with the point in the same rank of the old generation. If the new point is better than the old one and is farther away from the best-so-far point, then keep the new on and discard the old one; then compare the rest until they are all finished. Go to Step 15; otherwise, go to Step 13.

Step 13) If the distance of the old point is farther away from the best-so-far point than the new one and has better fitness, then keep the old one and reject the new one and go to

Step 12 to screen others; otherwise, go to Step 14.

Step 14) If the distance of the new one from the best-so-far point, $d_{n.}$ times the objective function value of the old one, $f_{0.}$ is greater than the distance of the old one, $d_{0.}$ times the objective function value of the new one, f_n, (i.e., $d_n f_0 > d_0 f_n$), then select the new one and go to Step 12, otherwise, generate a random number. If it is greater than 0.5, then keep the old one and discard the new one and vice versa.

Step 15) Repeat Step 5 through 14 until either a predetermined iterative number or an acceptable objective function value is reached.

The neural network structure is employed to investigate the above algorithm and the program is written in the C^{++} language. The objective function has been restricted to:

$$S = \sqrt{\sum_{i=1}^{n} (y_i - x_i)^2}$$

where y is computed output, x is experimental output. The purpose of training the neural network is to minimize the objective function by adjusting the weights during the program iterations with the proposed algorithm. This strategy of minimizing the objective function value is maintained for all cases in this paper.

2. Quick-Propagation algorithm

Fahlman (1988) conducted a systematic empirical study of learning speed in backpropagation-like algorithms. He chose the encoder/decoder task as the standard test problem because of its importance to real-word applications. The most important result described in his paper is a new algorithm called "QuickProp" that provides significant speedup over standard backpropagation algorithms and is one of today's most frequently used adaptive learning paradigms.

The QuickProp algorithm described in this paper, is one of the more heuristic modifications to backpropagation. It assumes that error curve can be approximated by a parabola with a minimum point, and that the effect of each weight can be considered independently. The main formula of the QuickProp algorithm is built on a second order Taylor series expansion for each weight element. The second order Taylor series approximation can be expressed as:

$$f(x_k + \Delta x_k) \approx f(x_k) + f'(x_k)\Delta x_k + \frac{1}{2} f''(x_k)\Delta x_k^{\,2} \qquad (1)$$

To minimize $f(x_k + \Delta x_k)$, the first derivative of the function with respect to Δx_k should be zero. Therefore we have:

$$f'(x_k) + f''(x_k)\Delta x_k = 0 \qquad (2)$$

$$\Delta x_k = \frac{f'(x_k)}{f''(x_k)} \qquad (3)$$

If the second order information is not easily available, then a local approximation can be made using the first order derivatives:

$$f''(x_k) = \frac{f'(x_k) - f'(x_{k-1})}{x_k - x_{k-1}} = \frac{f'(x_k) - f'(x_{k-1})}{\Delta x_{k-1}} \qquad (4)$$

Substituting (4) in (3) then yields:

$$\Delta x_k = \frac{f'(x_k)}{f'(x_{k-1}) - f'(x_k)} \Delta x_{k-1} \qquad (5)$$

The weight update formula for the QuickProp algorithm corresponds exactly to the expression given in equation (5):

$$\Delta w_{ij}(k) = \frac{\frac{\partial E}{\partial w_{ij}}(k)}{\frac{\partial E}{\partial w_{ij}}(k-1) - \frac{\partial E}{\partial w_{ij}}(k)} \Delta w_{ij}(k-1) \tag{6}$$

Everything in equation (6) proceeds as in standard back propagation. To compute the weight update, the current gradient, the previous gradient and the previous weight step are needed. Using this main formula, the computation is straightforward for some cases. If the current gradient is somewhat smaller than the previous one, but in the same direction, the weight changes again in the same direction. If the current gradient is in the opposite direction from the previous one, the minimum point was crossed over, and the next step will be placed somewhere between the current and previous positions.

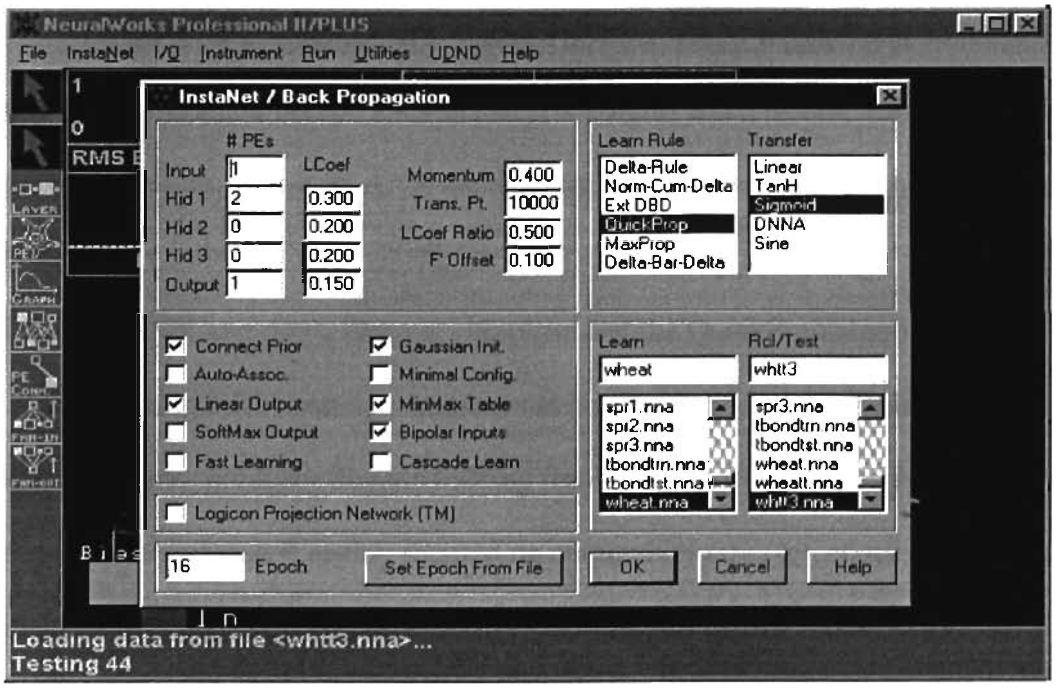
Although equation (6) is easy to compute, there are a few modifications necessary, due to violations of the above assumptions. If the numerator over the denominator in equation (6) is too large or undefined, then the formula should be modified to avoid divergence. In this case, a parameter called maximum growth factor can be used to control the step size of the weight. The other situations not handled by the main formula must be considered when the previous step size is zero, and there should be some changes in the current step. Also, the initial step size should be determined. The methods described in the thesis can result in an additional set of formulas in addition to the main formula:

$$\Delta w_{ij}(k) = \Delta w_{ij}(k-1)\mu \tag{7}$$

$$\Delta w_{ij}(k) = \Delta w_{ij}(k) - \alpha \frac{\partial E}{\partial w_{ij}}(k) \tag{8}$$

Equation (7) is applied when the magnitude of the current weight step is greater than μ times the magnitude of the previous weight step. Equation (8) adds a steepest descent component to the result computed from equation (6) or equation (7), α is the learning rate. All these modifications are required to make the QuickProp algorithm work faster than the standard backpropagation algorithm. The test of the algorithm will be reported in the following sections.

A commercial software package named NeuralWorks Professional II/PLUS is used to implement the QuickProp algorithm training, testing and generalization(Figure 6).


NeuralWorks Professional II/PLUS
File InstaNet I/O Instrument Run Utilities UDND Help
InstaNet / Back Propagation
OK
Cancel
Help
Set Epoch From File

	# PEs	LCoef
Input	1	
Hid 1	2	0.300
Hid 2	0	0.200
Hid 3	0	0.200
Output	1	0.150

Momentum	0.400
Trans. Pt.	10000
LCoef Ratio	0.500
F' Offset	0.100

Learn Rule: Delta-Rule, Norm-Cum-Delta, Ext DBD, [✓ QuickProp], MaxProp, Delta-Bar-Delta

Transfer: Linear, TanH, [✓ Sigmoid], DNNA, Sine

- [✓] Connect Prior
- [ ] Auto-Assoc.
- [✓] Linear Output
- [ ] SoftMax Output
- [ ] Fast Learning
- [✓] Gaussian Init.
- [ ] Minimal Config.
- [✓] MinMax Table
- [✓] Bipolar Inputs
- [ ] Cascade Learn
- [ ] Logicon Projection Network (TM)

16 Epoch

Learn: wheat — spr1.nna, spr2.nna, spr3.nna, tbondtrn.nna, tbondtst.nna, [✓ wheat.nna]

Rcl/Test: whtt3 — spr3.nna, tbondtrn.nna, tbondtst.nna, wheat.nna, wheatt.nna, [✓ whtt3.nna]

```
Loading data from file <whtt3.nna>...
Testing 44
```

Figure 6. Back propagation training interface

3.5 ANN Design and Implementation

1) Design of an ANN structure

A neural network is built of neurons. Neurons usually are arranged in layers, and the neurons in one layer often are connected to many neurons in other layers. Networks where data flows only in the forward direction are called feed-forward networks. They are very popular due to their relative simplicity and stability (Bharath and Drosen, 1994). In this paper, we used a multi-layer feed-forward neural network architecture. Based on the training data of crop yield response to nitrogen requirements (see Table 1), there is only one input neuron representing nitrogen and only one output neuron representing yield. When the number of input and output neurons is fixed. The number of neurons in hidden layers is variable. However, it is difficult to determine in advance the number of hidden layers and the number of neurons in each hidden layer. We made different trails, such as 1:1:1 net with 2 weights and 2 biases, 1:2:1 net with 4 weights and 3 biases, 1:3:1 net with 6 weights and 4 biases, 1:4:1 net with 8 weights and 5 biases, 1:5:1 net with 10 weights and 6 biases, so that an ANN architecture with the best performance could be achieved.

2) Training ANN by using a Genetic algorithm and the QuickProp algorithm

For training ANN, setting parameters is important since the combination of different parameter produces distinct training result. For a genetic algorithm, the investigation of parameters setting focused on community size (m), epochs (generations), and mutation rate (p_m). For the QuickProp algorithm, the effort to choose suitable parameters was made on learning rate and epoch size (the number of sets of training data presented to the network (learning cycles) between weight updates). After examination of parameter setting, the ANNs with best combination of parameters were saved and used for testing.

3) Test the ANN and generalization

The ANN is trained during the training phase, and then the testing phase serves as a way of measuring network performance. During this testing phase, the test cases are presented to ANN, and the ANN provides results. If the correct output (target) is known, the ANN performance could be measured. A major difference between training and testing is that in the test case, the weights in the network are not updated. A test program written in C/C++ for this thesis can read the test data into ANN and output its results. If testing results show that no over-fitting occurs, then the ANN has a good performance, and it could be a feasible model to predict the yield response to fertilizer application.

3.6 Knowledge Representation

Knowledge representation is a systematic way of codifying what knowledge exists about some subject area (or domain). It deals with the structuring, encoding, and storing of information so that its attributed meaning is clear. This representation of knowledge must be such that it can be stored and manipulated by a computer. There exist many methods for encoding knowledge, such as rules, mathematical equations, databases, spreadsheets, hypertext (Rauscher and Host 1990).

In return for the advantage of ANN, though, ANN does not explain its behavior. One cannot look at the knowledge that an ANN uses directly as he might study a set of rules. This is because ANNs do not contain any explicit representation of knowledge in symbolic form (Kamran and Mark 1993). In order to represent the knowledge that is stored in a perfectly trained ANN, a set of simulations to predict the fertilizer requirements based on the yield goal was implemented. Each input and its corresponding output from ANN is stored in a database, called a knowledge base to represent the knowledge of the relationships between soil fertility and crop yield. When a yield-goal is specified, the Inference Engine of the Expert System quickly queries this knowledge base to generate a recommendation of optimal fertilizer application.

3.7 Access to Database

When FUDSS 1.0 generates its recommendations for optimal fertilizer use, its expert system must access the database that stores soil testing results and knowledge base. However, the language, though used to develop the expert system cannot be accepted directly by a DBMS. Database systems have their own Structured Query Language (SQL) that is at present the most commonly used language in DBMS. In order to solve this problem a powerful language tool was employed in development of FUDSS 1.0. Microsoft Visual Basic 6.0 provides powerful support for database programming. Visual Basic 6.0 offers access to a variety of database systems, such as, Microsoft Access database, FoxPro database, dBase database, SQL Server and Oracle database. Because VB 6.0 embeds the Access database engine in its language, it is able to manipulate Access database without using Open Database Connectivity (ODBC). Microsoft Access 97 was chosen as the database management system for FUDSS 1.0 as well.

VB communicates with databases through the Data Control, Data Access Object, (DAO), and ActiveX Data Object (ADOs). Data control can read, modify, delete, and add records to databases. In this paper, Data Control was chosen as the connectivity between the database and the expert system.

In Visual Basic, the double quote is used with a SQL statement. One would think of an SQL statement as creating in essence a new "virtual" table from existing tables.

These tables do not really exist physically. However, for all practical purpose, VB acts as though they did. In VB terminology, a "virtual" table is called a Recordset, and SQL statements are said to create a Recordset. In standard relational database books, a "virtual" table is called a view. SQL also can be used in VB code with a statement of the form

Data1.RecordSource = " SELECT … FROM … WHERE …" to read data from databases (David I. Schneider, 1998). In this study, we used this kind of statement wherever we need to get data from database system.

3.8 User Friendly Interface

The FUDSS 1.0 system was built using, and runs on, the Microsoft Windows operating system. This means that the application has a graphic user interface (GUI) as the front end. The front end accepts the input data, and verifies the data that is valid. The GUI usually provides the user with several convenient data input and output formats, such as, windows, menus, mice, image, and voice. With GUI, the system is user-friendly, easy-to-learn and easy-to-use. This is very important for non-professional users. In order to develop this front-end interface, we used several tools provided by VB 6.0, such as forms, text boxes, list boxes, combo boxes, picture boxes, menu systems, and command buttons.

CHAPTER IV

RESULTS AND DISCUSSION

4.1 Case for ANN training using GA

The ten-run-average network error (standard deviation) does not increase with the community size. GA seems to prefer a smaller community size. Between the community size from 3 to 15, the best performance of GA is attributed to value of 5. If the community size is not too large or too small, there should be no significant difference on networks' performance (Table 4).

Table 4 Sensitivities to community size (m) of the genetic algorithm for 1:2:1 ANN training, ten-run-average network error (std), mutation rate $p_m = 0.1$, epochs = 100k

M	2	3	4	5	8	10	15	20
Std	1.4792	0.1476	0.0387	0.0239	0.0617	0.0599	0.0412	0.1204
Min.	< 0.016	< 0.016	2	3	6	10	17	25

Table 5 Sensitivities to epochs of the genetic algorithm for 1:2:1 ANN training, ten-run-best network error, mutation rate $p_m = 0.09$, community size m = 5

Epochs	500	1000	5k	10k	50k	100k	200k	500k
Std	0.1796	0.1117	0.0389	0.0284	0.0171	0.0135	0.0107	0.0105

Table 5 shows that the error decreases as the number of iterations increase. It is clear that the GA network converges very slowly and requires many iterations and a long execution time. In order to make the error acceptable, at least 200K iterations are required based on the simple case of wheat yield response to nitrogen. It takes around 25 minutes of run time on a Pentium 233 Personal Computer using the Microsoft Windows 95 operating system.

Table 6 shows that the proposed GA is sensitive to the mutation rate p_m. Selecting mutation rate carefully can improve the performance of a GA network significantly. In the case of wheat-nitrogen, the best performance is obtained when the mutation rate is 0.09. However, when p_m is set to 0.0 impling no mutation at all, all points in the community become the same in a short time before fitness is reached. Under this situation the network never converges resulting in failure to learn. Mutation is a subtle operation that requires minor adjustments for improving the performance of network.

Table 6 Sensitivities to mutation rate (p_m) of the genetic algorithm for1:2:1 ANN training, ten-run-best network error (std)

Epochs	P_m =0.00	p_m =0.05	p_m =0.08	p_m =0.09	p_m =0.10	p_m =0.20	p_m =0.30
100k	10.6753	0.0432	0.0156	0.0135	0.0239	0.0657	0.0575
200k	10.6753	0.0432	0.0145	0.0107	0.0236	0.0642	0.0551
Epochs	P_m =0.40	p_m =0.50	p_m =0.60	p_m =0.70	p_m =0.80	p_m =0.90	p_m =1.00
100k	0.1089	0.0556	0.0515	0.0401	0.0381	0.1048	0.1021
200k	0.0822	0.0437	0.0408	0.1149	0.0339	0.0768	0.1020

Table 7 Sensitivities to the architecture of ANN using genetic algorithm training, ten-run-best network error (std), mutation rate p_m = 0.09, community size = 5

Architecture	1:1:1	1:2:1	1:3:1	1:4:1	1:5:1
Epochs=50k	0.2181	0.0171	0.0255	0.0533	0.0294
Epochs=100k	0.2181	0.0135	0.0202	0.0433	0.0285
Epochs=200k	0.2181	0.0107	0.0137	0.0403	0.0249
Epochs=300k	0.2181	0.0106	0.0129	0.0389	0.0238

The changes in the number of nodes in the hidden layer result in the difference between training performances (Table 7). That does not mean that increasing the number of nodes in the hidden layer can lead to better performance. For example, if net architecture is set to 1:1:1 with 2 weights and 2 biases, all points in community become the same before the net has good fitness. If 1:2:1 net with 4 weights and 3 biases is taken as a net architecture, the best performance is achieved. However, when the number of nodes grows to 1:3:1 with 6 weights and 4 biases, and further to 1:4:1 with 8 weights and 5 biases, the performance of network dramatically goes down and the run times are doubled and redoubled. The training errors of 1:3:1, 1:4:1, and 1:5:1 net are all greater than the training error of 1:2:1 net. This phenomenon indicates lack of convergence of the GA. The number of parameters for each of 1:3:1, 1:4:1, and 1:5:1 net is greater than the number of data points (only 9 points).

Table 8 displays the best combination of parameters for GA training. The best parameter combination in the case we studied is: 1:2:1 for net structure with 7 parameters, 2 neurons in hidden layer, 5 points for community size, 0.09 of mutation rate, and 200k of iteration. Training ANN with this parameter combination has produced ideal results that are applied in fertilizer application prescription.

Table 8 The best combination of parameters of GA for fertilizer application

Architecture	Community size	Mutation rate	Epochs
1:2:1	5	0.09	>200k

4.2 Case for ANN training using QuickProp

The difference in performances is due to the application of distinct algorithms and the use of different nets with the same algorithm. Table 9 illustrates the results of the QuickProp algorithm. When nets are set to 1:2:1 with 4 weights and 3 biases and 1:4:1 with 8 weights and 5 biases, we have the best performance and the worst performance, respectively. Comparing GA with the QuickProp algorithm on the same net 1:1:1, the former provides a much better performance than the latter does. The training errors of 1:3:1, 1:4:1, and 1:5:1 net are all greater than the training error of 1:2:1 net. This phenomenon indicates lack of convergence of the QuickProp. The number of parameters for each of 1:3:1, 1:4:1, and 1:5:1 net is greater than the number of training cases (only 9 cases).

Back-propagation uses one important parameter to encourage learning. The learning rate, α, refers to the rate at which error modifies the weight. With large learning rates, a network may go through large oscillations during training. In fact, if the rates are too large, the network may never settle or converge. Smaller rates tend to be more stable. The theory of back-propagation requires rates approaching zero.

Table 9 Sensitivities to architecture of ANN using QuickProp training

Epochs\net	1:1:1	1:2:1	1:3:1	1:4:1	1:5:1
500	0.0516	0.0514	0.0451	0.0841	0.0420
1,000	0.0469	0.0309	0.0412	0.0771	0.0433
5,000	0.0431	0.0245	0.0348	0.0525	0.0344
10,000	0.0465	0.0240	0.0278	0.0542	0.0303
20,000	0.0342	0.0171	0.0226	0.0553	0.0208
30,000	0.0260	0.0153	0.0188	0.0562	0.0225
50,000	0.0183	0.0115	0.0207	0.0465	0.0129
100,000	0.0172	0.0084	0.0178	0.0406	0.0100
200,000	0.0178	0.0086	0.0178	0.0789	0.0105

Table 10 Sensitivities to learning rate of QuickProp algorithm for 1:2:1 ANN training

Epochs \ α	0.5	1.0	1.5	2.0	2.5
500	0.0514	0.0558	0.0645	0.0671	0.0692
1000	0.0309	0.0312	0.0417	0.0481	0.0512
5000	0.0245	0.0274	0.0417	0.0460	0.0506
10000	0.0240	0.0259	0.0398	0.0408	0.0487

Table 10 displays the results with moderate learning rates applied. When the learning rates are set in the range from 0.5 to 2.5 and the number of iterations is constrained between 5,000 and 10,000, the convergence is achieved and this situation is stable. At this stage, the best learning rate exists when the value of α equals 0.5.

In Table 11, two kinds of epoch size and 16 kinds of iterations are used to test the QuickProp algorithm. A QuickProp network uses cumulative learning. With this model, weight changes accumulate during the period of an epoch size before the weights are applied to the network. An epoch is the number of sets of training data presented to the network (learning cycle) between weight updates. An epoch size is a number less than or equal to the number of training vectors. Table 11 demonstrates that the QuickProp

Table 11 Sensitivities to epoch size of QuickProp for 1:2:1 ANN (training error)

Iterations	Epoch size=3	Epoch size=6	Epoch size=9
100	0.0295	0.0390	0.0337
200	0.0478	0.0356	0.0304
300	0.0270	0.0218	0.0313
400	0.0118	0.0197	0.0295
500	0.0342	0.0340	0.0311
600	0.0363	0.0305	0.0301
700	0.3950	0.0324	0.0296
800	0.0328	0.0346	0.0290
900	0.0359	0.0337	0.0295
1000	0.0252	0.0309	0.0304
5000	0.0096	0.0147	0.0263
10000	0.0170	0.0186	0.0191
21,000	0.0157	0.0181	0.0156
56,000	0.0145	0.0106	0.0102
61,000	0.0222	0.0144	0.0098
100,000	0.0137	0.0141	0.0094

algorithm is sensitive to epoch size, when it is trained with the data of nitrogen requirements based on wheat yield and epoch sizes are set to 3, 6 and 9, respectively. In general, this table exposes two situations. Under the first situation, when the number of iterations is less than or equal to 5000, the training performance with epoch size equal to

3 or 6 is better than the performance with epoch size equal to 9. Under the second situation, when the number of iterations is more than 5000, the latter is much better than the former. The fact shows that the convergence of the training error with epoch size equal to 9 is more stable than that with epoch size 3 or 6.

For the QuickProp algorithm, the desirable parameter combination could be 0.5 as learning rate, 9 as epoch size, and 1:2:1 with 4 weights and 3 biases as net architecture.

4.3 Comparison of GA with QuickProp

The following sections discuss several cases that could be used to measure the

Table 12 Test data for Wheat Yield Response to Nitrogen

nitrogen	wheat yield	nitrogen	wheat yield
32.00	16.00	139.00	64.90
37.00	18.40	145.00	66.90
44.00	21.80	149.00	68.20
49.00	24.30	153.00	69.50
55.00	27.40	157.00	70.80
57.00	28.40	162.00	72.40
63.00	31.50	168.00	74.30
68.00	34.10	174.00	76.30
74.00	37.20	179.00	77.90
77.00	38.80	185.00	80.00
82.00	41.30	189.00	81.30
86.00	43.30	194.00	83.10
89.00	44.70	198.00	84.50
93.00	46.60	204.00	86.60
97.00	48.40	209.00	88.40
102.00	50.70	212.00	89.50
106.00	52.40	217.00	91.40
111.00	54.50	225.00	94.40
116.00	56.50	229.00	95.90
122.00	58.80	232.00	97.00
128.00	61.00	235.00	98.10
135.00	63.50	239.00	99.60

performance of networks trained by different algorithms. In order to test ANN trained by different algorithms, we use the data in Table 12 as testing set, where nitrogen stands for test input and wheat yield is target output.

Table 14 and Figure 8 show the performance of the QuickProp algorithm; whereas, Table13 and Figure 7 display the performance of a Genetic algorithm. By analysis of these tables and figures we perceive that QuickProp, with an error of 0.0074, outperforms GA with error 0.0104. However, the difference between both errors is negligible. In other words, there is no significant difference in performance between these two networks trained with either QuickProp or GA

Table 13 Test Results of GA for Wheat Yield Response to Nitrogen

target	Calculated	target	calculated
16.00	16.54	64.90	64.75
18.40	19.09	66.90	66.98
21.80	22.62	68.20	68.43
24.30	25.12	69.50	69.87
27.40	28.09	70.80	71.28
28.40	29.07	72.40	73.03
31.50	31.99	74.30	75.10
34.10	34.39	76.30	77.15
37.20	37.23	77.90	78.84
38.80	38.64	80.00	80.85
41.30	40.95	81.30	82.19
43.30	42.78	83.10	83.87
44.70	44.14	84.50	85.22
46.60	45.93	86.60	87.24
48.40	47.69	88.40	88.95
50.70	49.87	89.50	89.98
52.40	51.58	91.40	91.71
54.50	53.68	94.40	94.53
56.50	55.75	95.90	95.96
58.80	58.18	97.00	97.05
61.00	60.55	98.10	98.15
63.50	63.25	99.60	99.63

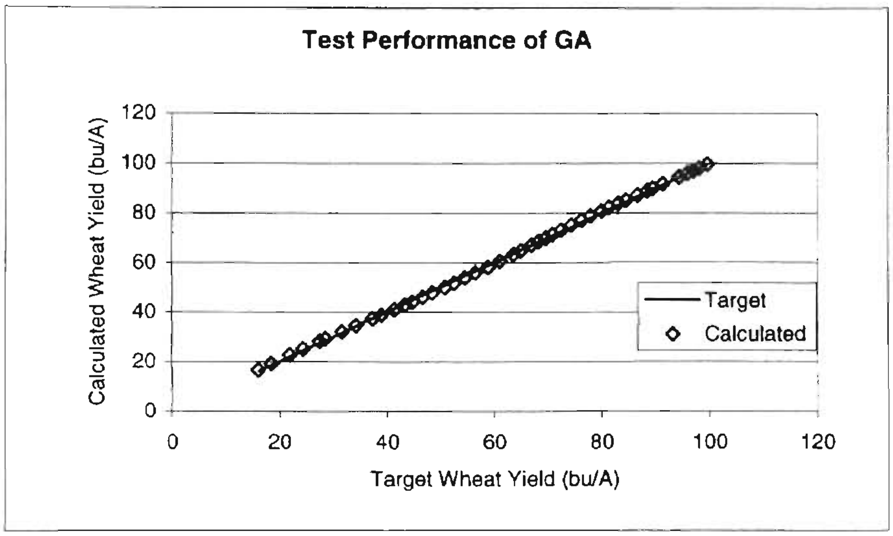


Figure 7. Performance of the Genetic Algorithm (1:2:1)ANN

Table 14 Test Results of QuickProp for Wheat Yield Response to Nitrogen

target	Calculated	target	calculated
16.00	16.22	64.90	64.61
18.40	18.82	66.90	66.78
21.80	22.42	68.20	68.20
24.30	24.96	69.50	69.60
27.40	27.98	70.80	70.98
28.40	28.98	72.40	72.68
31.50	31.94	74.30	74.71
34.10	34.38	76.30	76.71
37.20	37.25	77.90	78.38
38.80	38.67	80.00	80.38
41.30	41.00	81.30	81.71
43.30	42.84	83.10	83.39
44.70	44.21	84.50	84.75
46.60	46.00	86.60	86.81
48.40	47.77	88.40	88.55
50.70	49.93	89.50	89.61
52.40	51.64	91.40	91.40
54.50	53.73	94.40	94.33
56.50	55.77	95.90	95.84
58.80	58.17	97.00	96.98
61.00	60.50	98.10	98.14
63.50	63.14	99.60	99.71

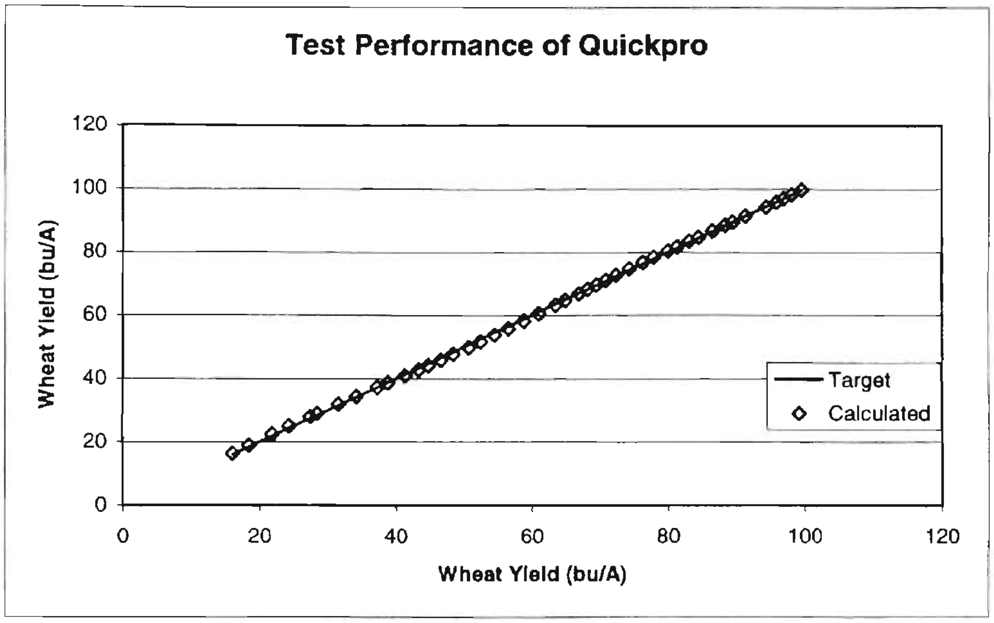


Figure 8. Performance of the QuickProp algorithm (1:2:1)ANN

The testing errors resulting from these two algorithms are low enough to satisfy soil scientists and farmers. The networks trained by both algorithms are qualified for prediction of optimal fertilizer application.

Figure 9 shows the testing performance of quadratic regression model. In this figure, the line represents the wheat yield from target and small squares represent the wheat yield calculated by $Y= -0.5286 + 0.5450X - 0.0005416X^2$ quadratic regression equation. This figure shows there are only 24 small squares falling on the target line that indicates 45% of points lost the fitness. The test error of quadratic model is 1.026, which is much greater than the test error of ANN (0.01). Because the values of target come from the experiments conducted at different locations in many years, they are considered as values with no error possibly caused by environment factors. However the deviation between target line and the values calculated by quadratic regression model is not acceptable. Hence, it is not a good choice to have quadratic regression be the forecast model for optimal fertilizer application. In contrast, ANN is able to learn the target line as exactly as possible, which makes it become desirable forecast model for optimal fertilizer application.

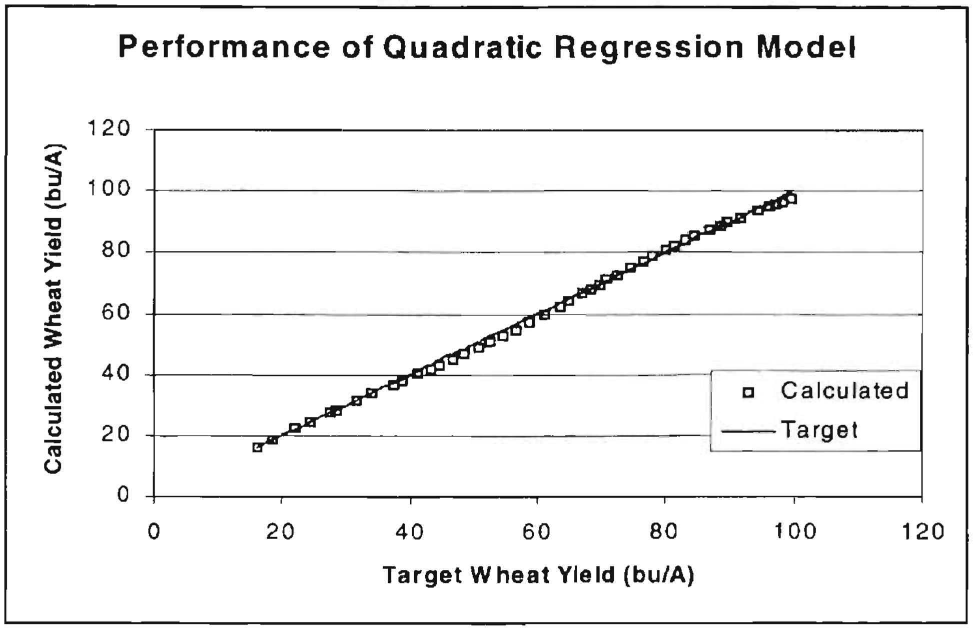


Figure 9. Performance of quadratic regression model

It is necessary to keep in mind that the data used as target for ANN learning should be error free if the training data set is small like the one in this study. If the target data is based on the experiment in a particular year or region, the ANN trained by these data will also contain errors. Consequently, the prediction from this ANN hardly is exactly correct.

The comparison of GA with QuickProp can be summarized as follows:

(1) The test errors of both algorithms are very small (about 0.01), therefore, they both are robust algorithms and could be applied in a variety of situations without problem.

(3) Using a small and simple training data set, GA takes about 25 times more run time

(4) than the QuickProp does to reach the same error standard. For large and complicated training sets, GA may not be acceptable due to its unacceptable run time.

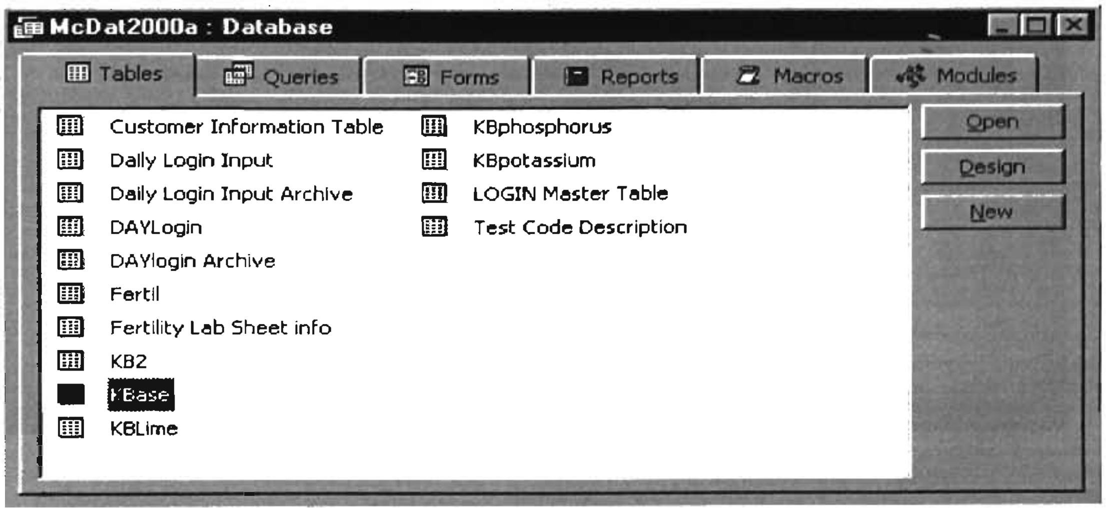
(5) For the simple case without complex errors, such as nitrogen requirements for wheat yield, the QuickProp algorithm easily outperforms GA considering training errors, epochs, and runtime.

Neuralworks professional II/PLUS provides an easily used interface and powerful computing ability. Thus, we chose this software for the training, testing and generalizing tool to build the knowledge base for the expert system for fertilizer use.

4.4 DBMS and Knowledge Base

A DBMS built for OSU SWFAL is used to store information, such as customer data, login information, fertility test results and a knowledge base for six main crops in Oklahoma (Figure 10). There are six tables most frequently accessed in DSS for fertilizer

use.



Customer Information Table
Daily Login Input
Daily Login Input Archive
DAYLogin
DAYlogin Archive
Fertil
Fertility Lab Sheet info
KB2
KBase
KBLime
KBphosphorus
KBpotassium
LOGIN Master Table
Test Code Description

Figure 10. DBMS for OSU Soil, Water, Forage Analytical Lab.

(1) Fertile table: used to store soil test results, including the contents of chemical nutrition, such as N, P, K, Ca, Mg, Fe, S, B, etc, and soil environments indexes, such as pH and BI.

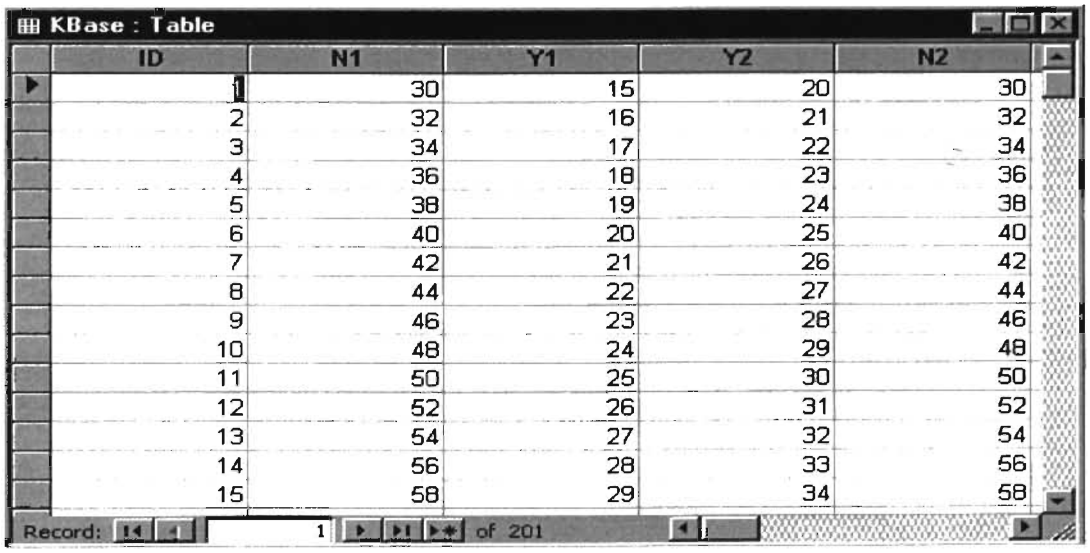
(2) LOGIN Master table: used to store login information, including customer code, sample number, login date, and so on.

(3) KBase table: used to store knowledge about six crops yield goal and their nitrogen requirements. The crops include wheat, barley, oat, sorghum, corn, and cotton. Nitrogen requirement is based on the yield goal for a specified field.

(4) KBphosphorus table: used to store knowledge of phosphorus requirements based on six crops yield goal.

(5) KBpotassium table: used to store knowledge about potassium requirements based on six crops yield goal.

(6) KBlime table: used to store knowledge of lime requirements for six crops.



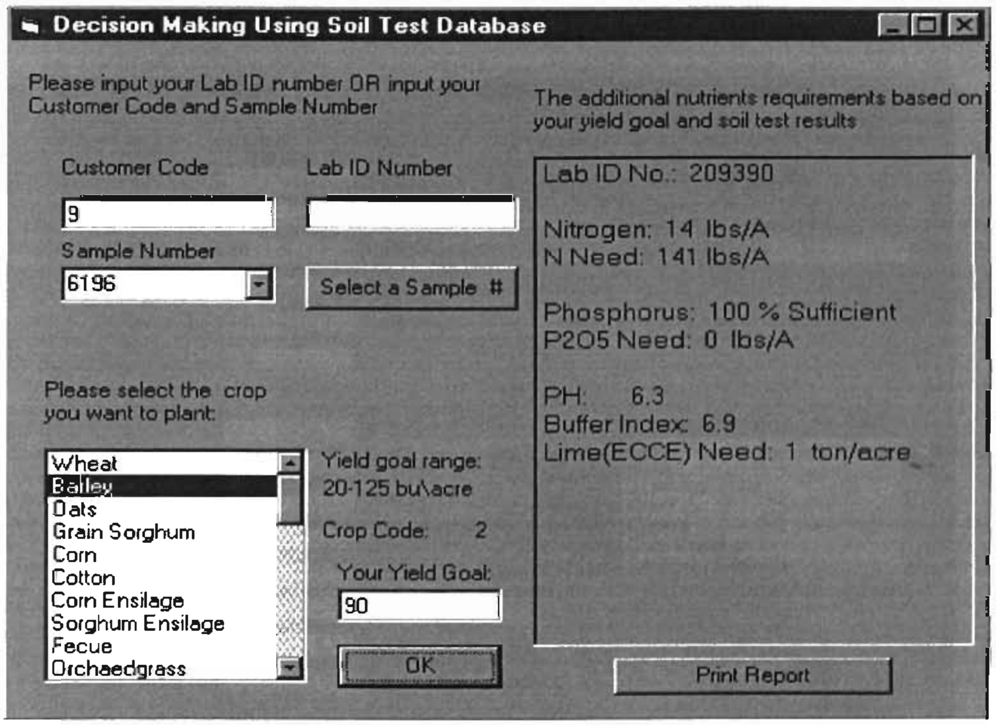
ID	N1	Y1	Y2	N2
1	30	15	20	30
2	32	16	21	32
3	34	17	22	34
4	36	18	23	36
5	38	19	24	38
6	40	20	25	40
7	42	21	26	42
8	44	22	27	44
9	46	23	28	46
10	48	24	29	48
11	50	25	30	50
12	52	26	31	52
13	54	27	32	54
14	56	28	33	56
15	58	29	34	58

Figure 11. Knowledge Base of nitrogen requirement for 7 main crops

Figure 11 gives an example of knowledge base for nitrogen requirements. In this figure Y1 stands for wheat yield goal, N1 for nitrogen requirements based on wheat yield goal. The nitrogen requirements come from a reverse use of a trained ANN using nitrogen as input and corresponding yield as output.

4.5 **Expert System Testing**

Figure 12 and Figure 13 indicate two alternative input methods to get the recommendation for fertilizer use. The first one works for the users who have soil sample tested in OSU SWFAL. These users could click "Use database input" menu entry. After


Decision Making Using Soil Test Database

Please input your Lab ID number OR input your Customer Code and Sample Number

Customer Code: 9

Lab ID Number:

Sample Number: 6196

Please select the crop you want to plant:

Wheat
Balley
Oats
Grain Sorghum
Corn
Cotton
Corn Ensilage
Sorghum Ensilage
Fecue
Orchaedgrass

Yield goal range: 20-125 bu\acre

Crop Code: 2

Your Yield Goal: 90

The additional nutrients requirements based on your yield goal and soil test results

Lab ID No.: 209390

Nitrogen: 14 lbs/A
N Need: 141 lbs/A

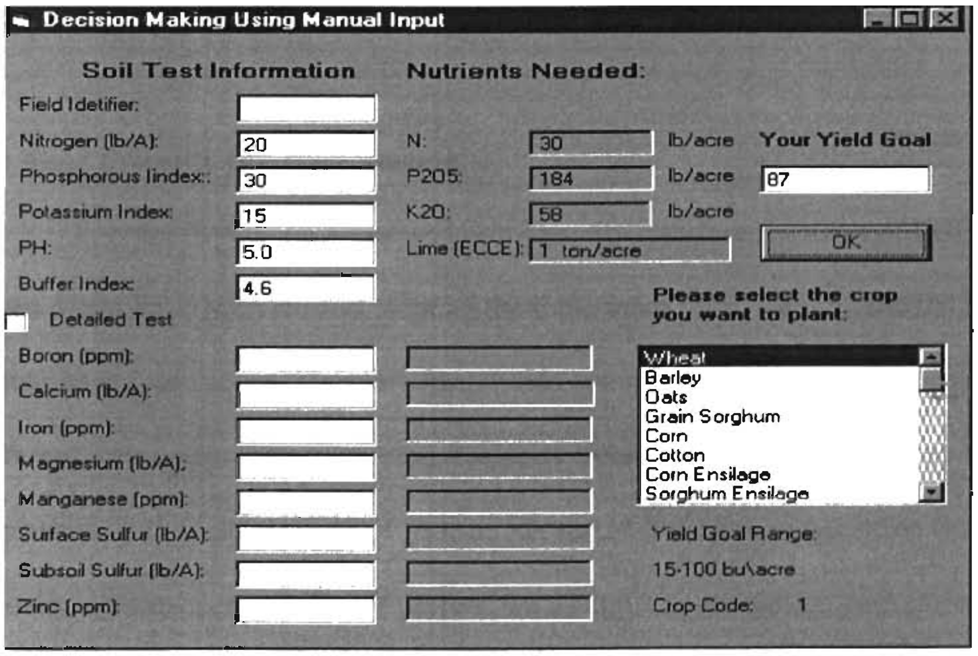
Phosphorus: 100 % Sufficient
P2O5 Need: 0 lbs/A

PH: 6.3
Buffer Index: 6.9
Lime(ECCE) Need: 1 ton/acre

Figure 12. Decision making using soil test database

they input their customer code, all their numbers of samples tested in OSU SWFAL appear in the ComboBox. These users could choose a sample number by highlighting it. Then, the user could highlight a crop he wants to plant and enter the desirable yield goal. After all the necessary information is input, the soil test results and the optimal rate of fertilizer use are displayed on the picture box. Also, the user has a choice to print out the report (Figure 12).

The second input method is for the user who had no samples tested in OSU SWFAL. The user is required to input the soil test results in Soil Test Information boxes first, then highlight the crop he wants to plant. After the yield goal is entered, the optimal rate of fertilizer use will show up in the picture box (Figure 13). In general, the second method usually is used in consultation by telephone.



Decision Making Using Manual Input

Soil Test Information

Field Idetifier:	
Nitrogen (lb/A):	20
Phosphorous Iindex::	30
Potassium Index:	15
PH:	5.0
Buffer Index:	4.6

☐ Detailed Test

Boron (ppm):		
Calcium (lb/A):		
Iron (ppm):		
Magnesium (lb/A):		
Manganese (ppm):		
Surface Sulfur (lb/A):		
Subsoil Sulfur (lb/A):		
Zinc (ppm):		

Nutrients Needed:

N:	30	lb/acre
P205:	184	lb/acre
K20:	58	lb/acre
Lime (ECCE):	1 ton/acre	

Your Yield Goal 87

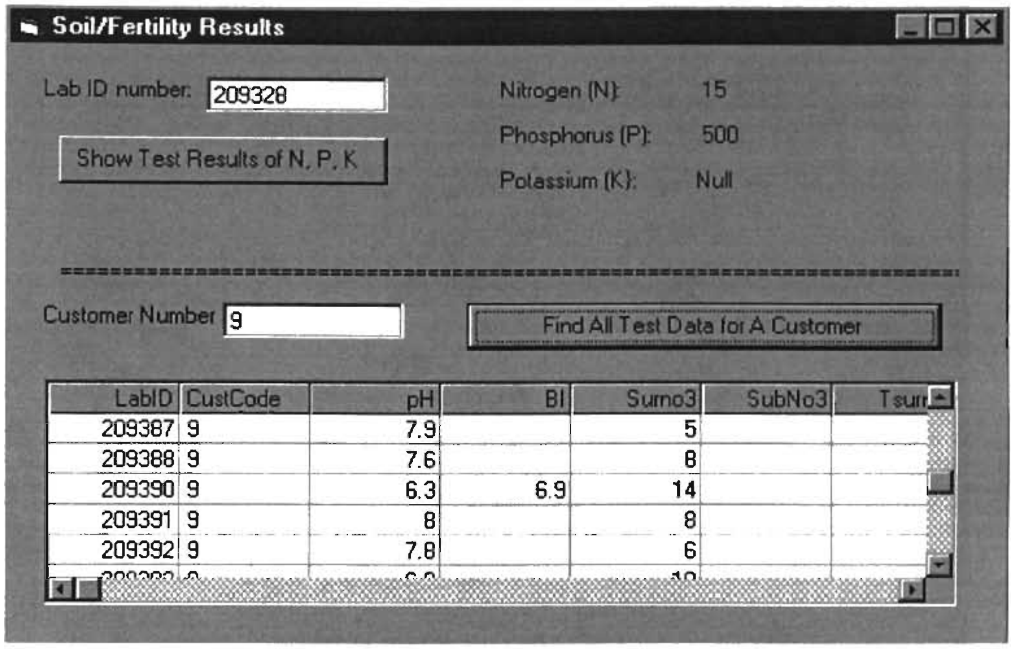
OK

Please select the crop you want to plant:

Wheat
Barley
Oats
Grain Sorghum
Corn
Cotton
Corn Ensilage
Sorghum Ensilage

Yield Goal Range:
15-100 bu\acre
Crop Code: 1

Figure 13. Decision making using manual input


Soil/Fertility Results
Show Test Results of N, P, K
Find All Test Data for A Customer

Lab ID number: 209328

Nitrogen (N): 15
Phosphorus (P): 500
Potassium (K): Null

Customer Number 9

LabID	CustCode	pH	BI	Sumo3	SubNo3	Tsurr
209387	9	7.9		5		
209388	9	7.6		8		
209390	9	6.3	6.9	14		
209391	9	8		8		
209392	9	7.8		6		

Figure 14. Query soil test results for a customer

4.6 User Friendly Interface Testing

In order to provide a neat and user-friendly interface for users of FUDSS 1.0, seven forms have been created. Among them the main form, named "frmDatMan," functions as a container. The main form holds one menu bar that leads the user to different forms. The menu system is a popular technique. A menu in VB can have many submenus up to four levels deep. A user can search and start an operation fast and audio-visually by using a menu. In this project, we design a menu system with three levels of submenus for organizing the operations. In the first level of submenus, user could select Data Entry, Report, Inquiry, and Accounting functions. If a user wants to make a decision for optimal fertilizer application, he should click Inquiry first then chose

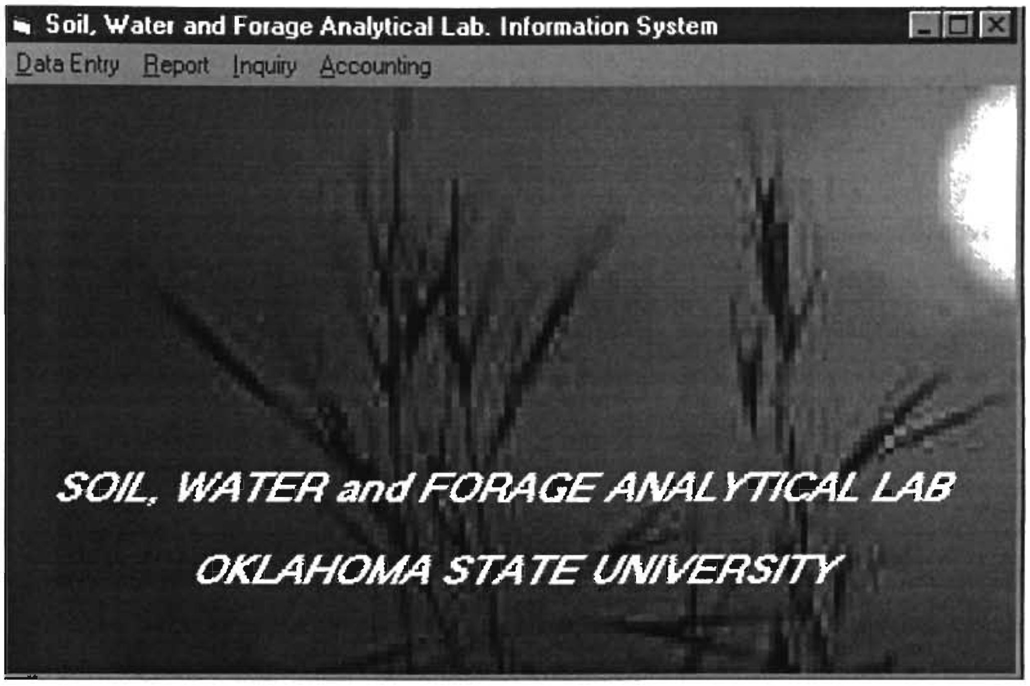

Soil, Water and Forage Analytical Lab. Information System
Data Entry
Report
Inquiry
Accounting
SOIL, WATER and FORAGE ANALYTICAL LAB
OKLAHOMA STATE UNIVERSITY

Figure 15. Main form of Decision Support System for Fertilizer Use

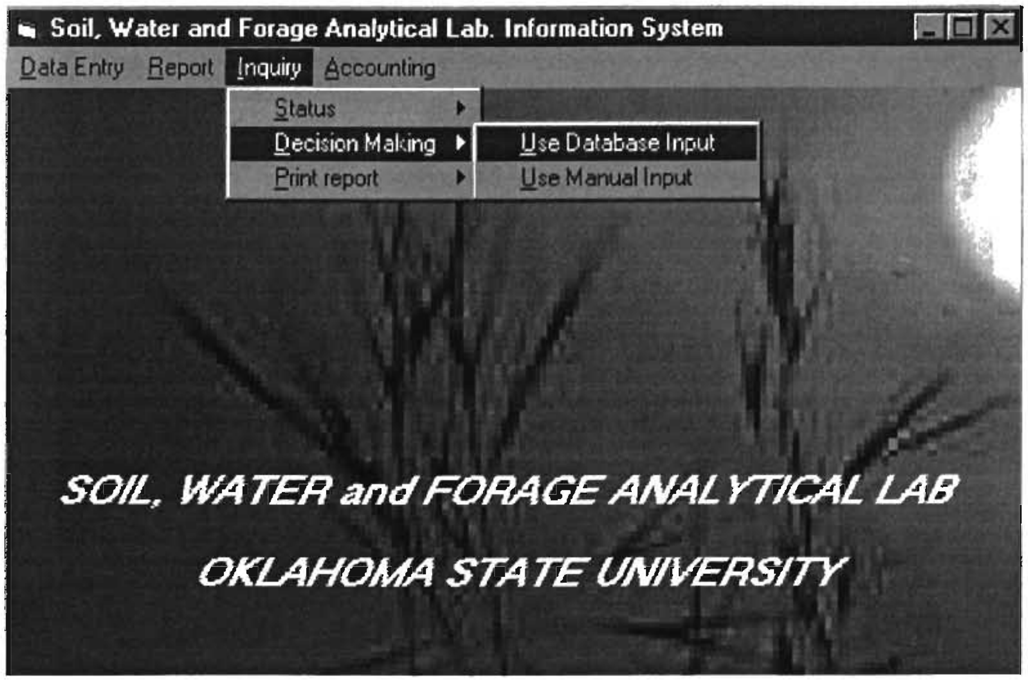

Soil, Water and Forage Analytical Lab. Information System
Data Entry
Report
Inquiry
Accounting
Status
Decision Making
Print report
Use Database Input
Use Manual Input
SOIL, WATER and FORAGE ANALYTICAL LAB
OKLAHOMA STATE UNIVERSITY

Figure 16. Menu system of DSS for Fertilizer Use

"Decision Making" in the second level under "Inquiry" group. From here, the user can get into the third level and select an input method, "Use Database Input" or "Use Manual Input" (Figure 16).

To avoid program errors due to improper input from careless users, several Message Boxes are introduced as reminders for users. For example, if a user inputs a yield goal of a crop beyond its range, a message appears in the Message Box to inform user to enter a correct value for the yield goal (Figure 17). "This sample is still in process" is another message example shown in Message Box to notify the user that his sample is still being processed (Figure 18).

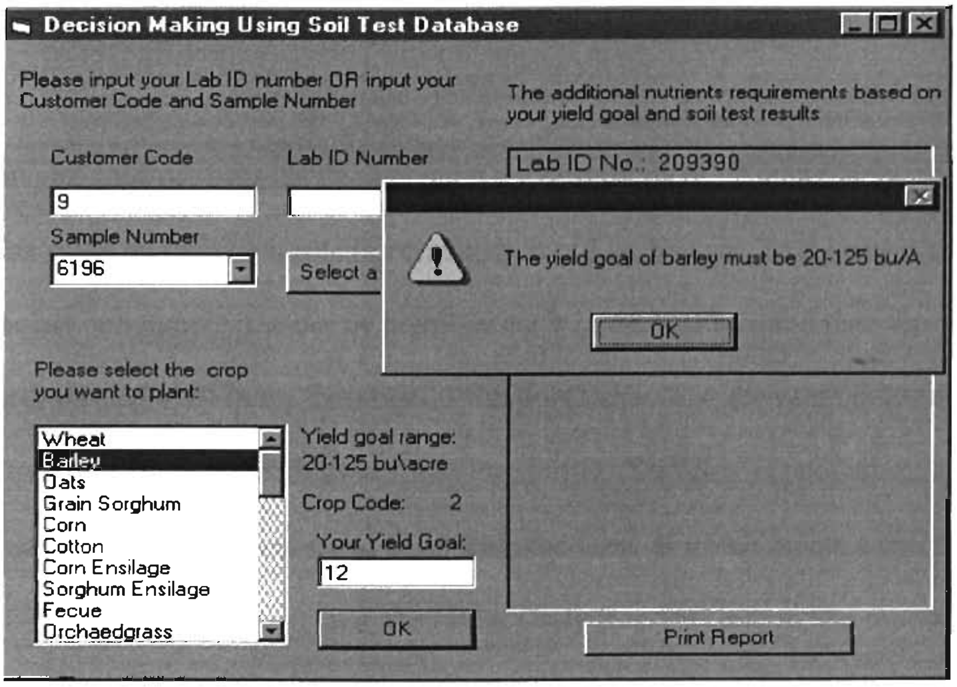
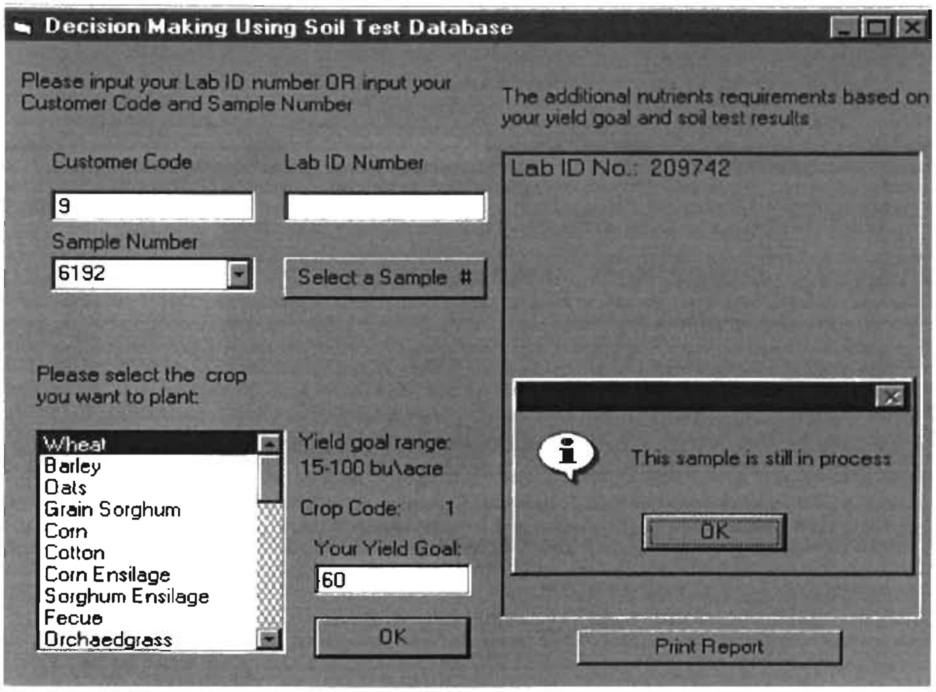

Decision Making Using Soil Test Database
Please input your Lab ID number OR input your Customer Code and Sample Number
The additional nutrients requirements based on your yield goal and soil test results
Customer Code
Lab ID Number
Lab ID No.: 209390
9
Sample Number
6196
Select a
The yield goal of barley must be 20-125 bu/A
OK
Please select the crop you want to plant:
Wheat
Barley
Oats
Grain Sorghum
Corn
Cotton
Corn Ensilage
Sorghum Ensilage
Fecue
Orchaedgrass
Yield goal range: 20-125 bu\acre
Crop Code: 2
Your Yield Goal:
12
OK
Print Report

Figure 17. Message box reminds user about yield goal range


Decision Making Using Soil Test Database

Please input your Lab ID number OR input your Customer Code and Sample Number

Customer Code: 9

Lab ID Number:

Sample Number: 6192

Please select the crop you want to plant:

Wheat
Barley
Oats
Grain Sorghum
Corn
Cotton
Corn Ensilage
Sorghum Ensilage
Fecue
Orchaedgrass

Yield goal range: 15-100 bu\acre

Crop Code: 1

Your Yield Goal: 60

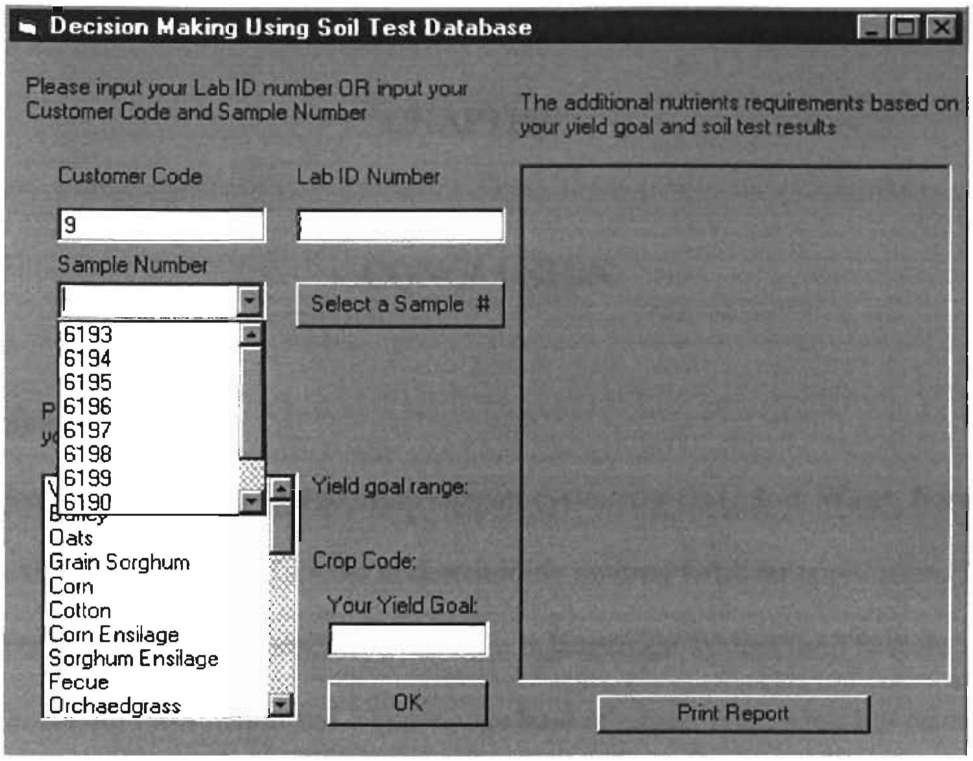
The additional nutrients requirements based on your yield goal and soil test results

Lab ID No.: 209742

This sample is still in process

Figure 18. Message box shows the sample situation

Several controls were used in the interface development. Among them, the ComboBox plays an important role in combination of List box and Text box. It allows user to choose one sample number by highlighting it in the List box and then the chosen number will be displayed in the Text box. With the ComboBox, the user could select a number from a lot of numbers in short time (Figure 19). Text Box is another important control that can be used to receive the input from the user. If a user inputs a specific login date into the Text Box and click the "Print Daily Reports" button, the reports for the samples logged in on that day will be printed out (Figure 20).


Decision Making Using Soil Test Database

Please input your Lab ID number OR input your Customer Code and Sample Number

The additional nutrients requirements based on your yield goal and soil test results

Customer Code: 9

Lab ID Number:

Sample Number:

6193
6194
6195
6196
6197
6198
6199
6190

Select a Sample #

Oats
Grain Sorghum
Corn
Cotton
Corn Ensilage
Sorghum Ensilage
Fecue
Orchaedgrass

Yield goal range:

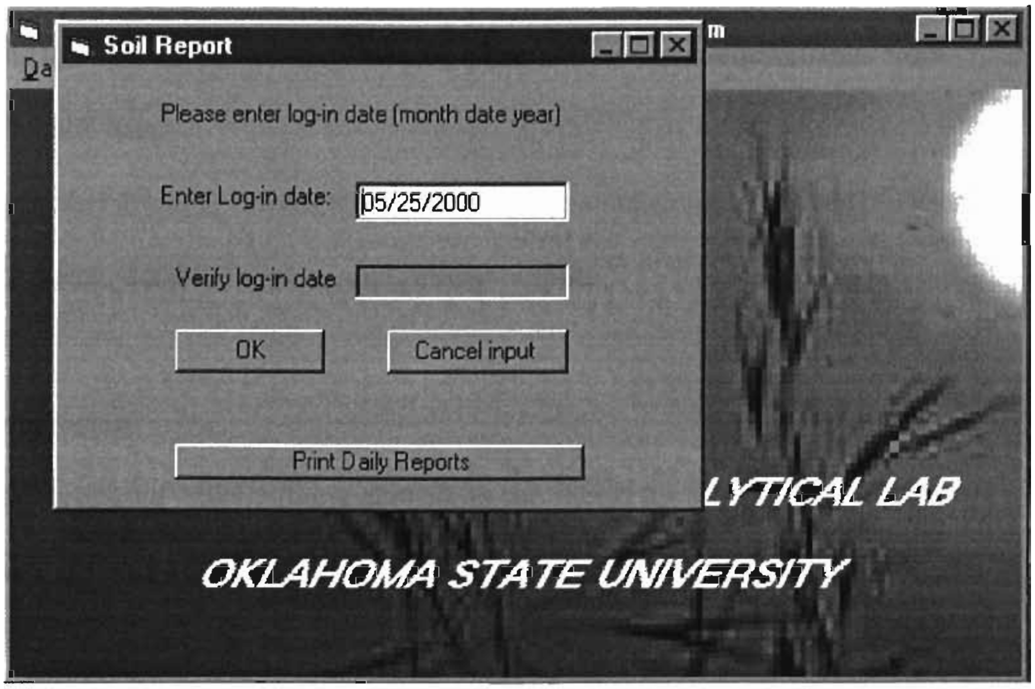
Crop Code:

Your Yield Goal:

OK

Print Report

Figure 19. ComboBox lists all sample numbers for one customer


Soil Report

Please enter log-in date (month date year)

Enter Log-in date: 05/25/2000

Verify log-in date

OK

Cancel input

Print Daily Reports

Figure 20. Form for printing daily reports

CHAPTER V

CONCLUSION

5.1 Summary

In this study, we built a decision support system for OSU Soil, Water, Forage Analytical laboratory to aid farmers in determining optimal fertilizer application. This system consists of two components, a database management system used to store soil test results, customer information and a knowledge base of expert system used to resolve optimal fertilizer application based on yield goal. In order to construct the knowledge base of the expert system, two artificial neural network training algorithms – QuickProp Algorithm and Genetic Algorithm – are investigated and compared on the ability of learning the relation model between crop yield and nitrogen requirements from experimental data.

A user-friendly interface is created for communications between the user and expert system, database system, and printed output.

5.2 Conclusion

(1) Both proposed ANN training algorithms are robust for learning the relation of crop yield and nitrogen requirements. Both outperform the linear regression model.

(2) The training with the Genetic Algorithm takes 25 times longer to run than the QuickProp Algorithm does when the same set of simple data for crop yield response

to nitrogen is applied.

(3) This study introduces a method of knowledge representation, which uses database tables to store input and corresponding output value from prediction of ANN. This method could simplify the task of expert system coding.

(4) This study provides a neat, user-friendly interface for the operation of Decision Support System.

(5) This study offers an automatic recommendation generator to assist Oklahoma farmers making decision in fertilizer use.

5.3 Future work

The decision support system created from this study only could determine how many pounds of nitrogen are needed per acre based on the yield goal of crops and how many pounds of phosphorus and potassium are needed depending on P test index and K test index. However, the fertilizer products no the market contain different amount of nitrogen, phosphorus, and potassium with different prices. It is hard for farmers to find the right mixture of fertilizer ingredients that meet the needs of nitrogen, phosphorus, and potassium requirements and cost the least. In order to solve this investment plan problem, integrating a Linear Programming model into this system is needed.

There are several ANN training algorithms including Delta-Bar-Delta algorithm, Conjugate Gradient algorithm, and Levenberg-Marquardt algorithm mentioned in the literature review of this thesis. It is necessary to investigate and compare them in modeling the optimal fertilizer application also.

BIBLIOGRAPHY

Bennett, C. A. and Franklin, N. L. (1954), Statistical Analysis in Chemistry and the Chemical Industry, John Wiley & Sons, INC. pp. 663-688.

Bhamidipati, K. (1998), SQL Programmer's Reference, Osborne/McGraw-Hill.

Bharath, R.and Drosen, J. (1994), Neural Network Computing, Windcrest/McGraw-Hill.

Bielawskti, L. and Lewand, R. (1988) Expert Syatems Development Building PC-Based Applications, Wellesley, MA: QED Information Sciences, Inc.

Bishop, C.M. (1995), Neural Networks for Pattern Recognition, Oxford: Oxford University Press.

Brian J. R. and Wayland, J. (1994), Using genetic algorithms to solve a mutiple objective groundwater pollution containment problem, Water Resource Research, Vol. 30, No 5, 1589-1603.

Castillo, E., Gutierrez, J. and Hadi, A. (1997), Expert Systems and Probabilistic Network Models, New York, NY: Springer-Verlag New York, Inc.

Colwell, J.D. (1994), Estimating Fertilizer Requirements: A Quantitative Approach, UK: Cab International, pp. 11-26.

Geman, S., Bienenstock, E. and Doursat, R. (1992), "Neural Networks and the Bias/Variance Dilemma", Neural Computation, 4, 1-58.

Gifford, M. (1997), "Net Results," Civil Engineering, (January). pp. 58-60.

Goldberg, D. E. (1989), Genetic Algorithms in Search, Optimization, and Machine Learning, Addison-Wesley, Reading, Mass.

Fahlman, S. E. (1989) "First learning variation on back-propagation: An empirical study," in Proceedings of the 1988 Connectionist Models Summer School, D. Touretzky, G. Hinton and T. Sejnowski, eds., San Mateo, CA: Morgan Kaufmann, pp. 38-51.

Hagan, M. T., Demuth, H. B., and Beale, M. (1996) Neural Network Design, PWS Publishing Company.

Haykin, S. (1994), Neural Networks: A Comprehensive Foundation, NY: Macmillan, p.2.

Hebb, D. O. (1949) The Organization of Behavior. New York: Wiley.

Holland, J. R. (1975), Adaptation in Natural and Artificial Systems, University of Michigan Press.

Hopfield, J. J. (1982) "Neural networks and physical systems with emergent collective computational abilities," Proceedings of the National Academy of Sciences, Vol. 79, pp. 2554-2558.

Jacobs, R. A. (1988) "Increased rates of convergence through learning rate adaptation," Neural Networks, Vol. 1, No. 4, pp 295-308.

Klein, R. Michel (1995), Knowledge-based Decision Support Systems: with Applications in Business 2nd ed. New York: Wiley, 1995.

McCulloch, W. and Pitts, W. (1943), "A logical calculus of the ideas immanent in nervous activity," Bulletin of Mathematical Biophysics., Vol. 5, pp. 115-133.

Minsky, M. and Papert, S. (1969), Perceptrons, Cambridge, MA: MIT Press.

Myers, Raymond H. (1989), Classical and Modern Regression with Application, Duxbury Press, pp. 8-39.

Parsaye, Kamran and Chignell, Mark. (1993), Intelligent Database Tools and Applications, John Wiley and Sons, Inc. pp. 192.

Rao, Valluru and Rao, H. (1995), C++ Neural Networks and Fuzzy Logic, New York, NY: MIS: Press.

Rauscher, H. M. and George, E. H. (1990), Hypertext and AI: A complementary combination for knowledge management. AI Applications 4: 58-61.

Ripley, B.D. (1994), "Neural Networks and Related Methods for Classification," Journal of the Royal Statistical Society, Series B, 56, 409-456.

Rosenblatt, F. (1958) "The perceptron: A probabilistic model for information storage and organization in the brain," Psychological Review, Vol. 65, pp. 386-408.

Rumelhart, D. E. and McClelland, J. L. eds. (1986), Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Vol. 1, Cambridge, MA: MIT Press.

Schneider, David I. (1998), An Introduction to Programming Using Visual Basic 5.0 (3 ed), Upper Saddle River, NJ: Prentice Hall.

Shih, Y. (1994), Neuralyst User's Guide (Vol.4), Pasadena, CA: Cheshire Engineering Corporation.

Sprague, Ralph H., and Eric, Carlson, (1982) Build Effective Decision Support Systems. Englewood Cliffs, N.J.: Prentice-Hall.

Turban, Efraim and Watkins, R. Paul (1986) Integrating Expert Systems and Decision S Support Systems, in Sprague, H. Ralph, Jr. and Watson, J. Hugh (eds.), Decision Support Systems: Putting Theory into Practice, pp. 138-152, N.J.: Prentice-Hall.

Vogl, T. P., Mangis, J. K., Zigler, A. K., Zink, W. T. and Alkon, D. L. (1988), "Accelerating the convergence of backpropagation method," Biological Cybernetics, vol. 69, pp. 256-264.

Ullman, J. D. and Widom, J. (1997), A First Course in Database Systems, Prentice Hall, Inc., a Simon and Schuster Company.

Walker, T. C. and Miller, R. K. (1986), Expert Systems, Madison, GA: SEAI, 1986.

Widrow, B. and Hoff, M. E. (1960), "Adaptive switching circuits," 1960 IRE WESCON Convention Record, New York: IRE Part 4, pp. 96-104.

Wilason, R. and Sharda, R. (1992), "Neural Networks: 'Brain Metaphors' of Information Processing Turns Heads in OR/MS Community," OR/MS Today, (August).

Williams, J. (1986), Expert Systems: Knowledge Engineering, Macmillan Publishing Co.

Yang, R. (1997), Application of Neural Networks and Genetic Algorithms to Modelling Flood Discharges and Urban Water Quality, Ph. D. Thesis, University of Manchester, England.

Zalzala, S. and Fleming, J. (1997), Genetic Algorithms in Engineering Systems, The Institution of Electrical Engineers, London, UK.

APPENDIX A

GLOSSARY

Artificial Intelligence (AI). A multidisciplinary subfield of computer science that blends cognitive science, psychology, philosophy, neurophysiology, engineering, robotics, and linguistics and applies them to the study of intelligent behavior. Much of the work in AI focuses on generating intelligent behavior in machines.

Artificial Neural Network (ANN) is a network of many simple processors ("unit"), each possibly having a small amount of local memory. The units are connected by communication channels ("connection"), which usually carry numeric (as opposed to symbolic) data, encoded by any of various means. The units operate only their local data and on the inputs they receive via the connections. The restriction to local operation is often relaxed during training.

Artificial Neuron (AN) is conceptual device that works as follows. It receives input on one or more input lines. It converts the inputs into a net input by calculating a weighted sum of inputs. And it responds to different net inputs by producing different net outputs based on relationship between the net input and the output. This relationship is called the transfer function of the AN.

Back Propagation (Back-prop) is a general purpose network paradigm. Use Back-prop for system modeling, prediction, classification, filtering, and many other general problems. Back-prop calculates an error between desired and actual output and propagates the error information back to each node in the network. The back-propagated error drives the learning at each node. Back-prop uses two important parameters to encourage learning. The first, Learning Rate (also known as Learning

Coefficient), refers to the rate at which errors modify the weights. The second, Momentum, says that if weights change in a certain direction than there is a tendency for weights to continue to change in that direction.

Front End. An interface program that operates between a user and a complex application program. Its operation usually makes the complexity of the application program less daunting and more understandable to the user.

Genetic Algorithm (GA). The genetic algorithm is a procedure based on the mechanics of nature selection and natural genetics, which combines an artificial survival of the fittest with genetic operators abstracted from nature. It has been applied to a number of problem including search, optimization and machine learning.

Inference Engine. See **reasoning engine**.

Information. A collection of data that has at least some level of organization, but it remains unevaluated in terms of a justified belief in its truth.

Intelligent Front End (IFE). See **front end**. A front end program that uses AI techniques to perform its task intelligently.

Knowledge Acquisition. The process of collecting, organizing, and formalizing knowledge from some source, usually human experts, for use in the computer program-also, in the general sense, learning.

Knowledge Base. A database that stores knowledge needed to solve problems in the specific domain.

Knowledge Representation. Formalizing and storing knowledge into some structure (often using symbols) so that those symbols can be automatically processed by machine (computer).

Knowledge. The representation of real-world constructs, such as facts, rules, events, heuristics, and perception-based performance. The body of truth, information, and principles acquired through education, training, and experience. Knowledge means being aware of facts, and it also means understanding how to apply that factual knowledge.

Microsoft Access is a relational database program that developed by Microsoft. MS-Access creates database programs to store, analyze, and report on information.

Principal component analysis (PCA). Principal Component Analysis (PCA) is a technique to find the directions in which a cloud of data points is stretched most. Directions of stretch in data represent most of the information in the data and are thus important to know. Knowing these directions allows us to store the data in a compressed form and later reconstruct the data with a minimal amount of distortion.

Projection pursuit regression (PPR). The PPR method implemented by Friedman and Tukey, (1974) is used to estimate a smooth function of several variables from noisy and scattered data. The estimate is a sum of smoothed one-dimensional projections of the variables. The term "projection" implies that we are looking at projected data and the term "pursuit" means to find a "good" projection for purpose of regression.

QuickProp. QuickProp, developed by Scott Fahlman at Carnegie Mellon University, uses a quadratic estimation heuristic to determine direction and step size. This learning rule may be much faster than standard back propagation.

Reasoning Engine. The component of a rule-based system that embodies the inference and control mechanisms. It selects rules to evaluate, interprets the rule's instructions, matches the rules promises to working memory elements, and carries out the action(s) specified in the rule. Also, referred to as an inference engine.

Relational Databases. A collection of related tables is called relational database.

Structured Query Language. (SQL, pronounced "sequel") is a nonprocedural language that is used to handle data in the modern-day relational database systems (RDBMS). It is a set of commends used by all programmers and application developers to access data in any database. SQL was developed by IBM in the 1970s with the intention of using E.F. Codd's model of storing data in relational databases.

User Interface. In our formulation, the user interface consists of the physical medium that the computer provides. This includes hardware and software.

APPENDIX B

ACRONYMS

AI – Artificial Intelligence

AN – Artificial Neuron

ANN – Artificial Neural Network

DBMS – Database Management System

DSS – Decision Support System

ES – Expert System

GA – Genetic Algorithm

GUI – Graphic User Interface

IE – Inference Engine

KB – Knowledge Base

MS – Microsoft

ODBC – Open Database Connectivity

OSU – Oklahoma State University

PCA – Principal component analysis

PPR – Projection pursuit regression

RDBMS – Relational Database Management System

SDK – Software Development Kit

FUDSS – Soil Fertilizer Use Decision Support System

SQL – Structured Query Language

SWFAL – OSU Soil, Water, and Forage Analytical Laboratory

VB – Visual Basic Language

VITA

Jianhua Ren

Candidate for the Degree of

Master of Science

Thesis: DECISION SUPPORT SYSTEM FOR FERTILIZER USE

Major Field: Computer Science

Biographical:

Education: Received Bachelor of Science degree in Agronomy from Shanxi Agricultural University, Shanxi, China in January 1982. Completed the requirements for the Master of Science degree with a major in Computer Science at Oklahoma State University in July, 2000.

Experience: Employed by Oklahoma State University, Soil, Water & Forage Analytical Laboratory as a software developer, 07/1999 to present; employed by Shanxi Academy of Agricultural Science of China, Crop Genetic Resources Institute as an assistant professor, 08/1987 to 01/1994; Employed by Shanxi Academy of Agricultural Science of China, Crop Genetic Resource Institute as research assistant, 01/1982 to 07/1987.