Artificial Intelligence in Insurance Profitability Models

by

Edward S. Whang

Submitted to the Department of Electrical Engineering and Computer Science

in Partial Fulfillment of the Requirements for the Degrees of

Bachelor of Science in Electrical Engineering and Computer Science

and Master of Engineering in Electrical Engineering and Computer Science

at the Massachusetts Institute of Technology

May 21, 1999

[June 1999]

Author [illegible]
Department of Electrical Engineering and Computer Science
May 21, 1999

Certified by [illegible]
David R. Karger
Thesis Supervisor

Accepted by [illegible]
Arthur C. Smith
Chairman, Department Committee on Graduate Theses

Artificial Intelligence in Insurance Profitability Models
by
Edward S. Whang

Submitted to the
Department of Electrical Engineering and Computer Science

May 21, 1999

In Partial Fulfillment of the Requirements for the Degrees of
Bachelor of Science in Electrical Engineering and Computer Science
and Master of Engineering in Electrical Engineering and Computer Science

ABSTRACT

Artificial intelligence techniques have been successfully applied to the underwriting of credit risk products such as credit cards and loans, but not to the underwriting of casualty risk. Techniques in data mining and neural networks provide an efficient and effective method for creating a decision-aiding tool for underwriters. A web-based back propagation program will increase the accessibility of the tool, which outperforms traditional statistical techniques such as multiple linear regression. The application will be implemented on publicly available German credit data, illustrating the benefits of neural networks in information technology.

Thesis Supervisor: David R. Karger

Table of Contents

Chapter 1
INTRODUCTION

1.1 Background

Workers' compensation laws dictate that employers should shoulder the costs of occupational disabilities without regard to any fault involved; by considering any such losses and costs of production, the laws relieve employers of liability from lawsuits involving negligence. Furthermore, by providing a prompt income for the victim regardless of fault, as well as a reduction in court delays and costs, workers' compensation laws encourage maximum employer interest in safety and promote a candid study of causes of accidents (U.S. Chamber of Commerce, 1996).

1.2 Existing System

American International Group, Inc., (AIG) offers workers' compensation insurance. Currently, underwriters utilize a variety of forms and checklists in order to make the decision of giving workers' compensation coverage for a company. With the satisfaction of a given set of underwriting criteria, the underwriter approves the coverage of the employer, and calculates the premium for coverage. Computational tools exist for the formulation of premiums; however, no technological system exists in aiding the decision process of the underwriter.

The strengths of the existing system lie in the experience and expertise of the underwriter and the reliable set of pre-determined criteria. However, with large amounts of data, a computer system could greatly aid the decision process of the human underwriter. Currently, artificial intelligence techniques have been successfully applied to the underwriting of credit risk products such as credit cards and loans. However, no research or publications exist on the applications of these techniques to the underwriting of casualty risk.

1.3 Objective

The objective of the thesis project is to study a variety of data mining algorithms and implement a web-based system for enhancing the underwriters' decision-making capabilities (Fig. 1.1). The system is tested on a publicly available dataset pertaining to German credit information, which provides 1000 examples of customer credit information and classifies the customer as either approved or rejected using a set of variables.

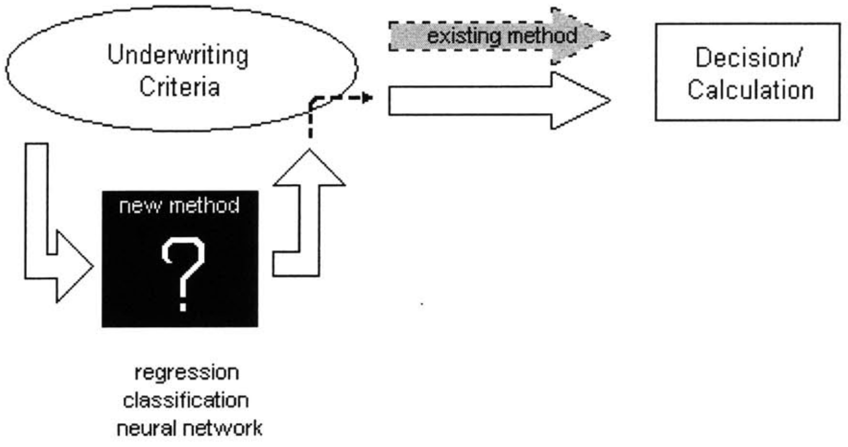


Figure 1.1: A simple diagram comparing the existing and proposed method of the underwriter's decision.

The paper compares, in specific, two methods of data analysis for this two method classification problem: multiple linear regression, which attempts to discover linear relationships between variables, and back propagation, a versatile, multi-purpose neural network algorithm. Each of these methods were implemented and trained using 800 examples of the dataset, and the remaining 200 were used to test the models. Due chiefly to the apparent non-linearity and non-uniformity of the data, back propagation works as a more effective learning method for the given data set. Back propagation yields a significantly more accurate prediction with a total error of 6.71%, while the regression model predicts the outcome with a total error of 16.48%.

Although creating and designing the back propagation network takes more time and effort, the benefits in accuracy outweigh the cost of production. The web-based back propagation simulator is a Java applet with the weights, categories, and values hard-coded into the application, and the actual applet must be re-designed in order to deal with different data. The training algorithm can take in any numerical text file as input and can simulate a variety of back propagation architectures; however, the parameters must be re-evaluated in order for the system to work correctly.

The paper is organized as follows. Chapter 2 provides a description of the data mining procedure. Chapter 3 describes the German credit dataset. Chapter 4 explores the theory of regression analysis, particularly simple and multiple linear regression, and Chapter 5 provides an analysis of the data using multiple linear regression. Chapter 6 provides an overview of neural networks. Chapter 7 explains the back propagation algorithm, and Chapter 8 describes the implementation of the web-based neural network algorithm. Finally, Chapter 9 compares the results of the linear regression analysis and the back propagation network.

Chapter 2

DATA MINING

Information technology has evolved radically over the past few decades. The rapid improvements in processing power and storage capabilities, coupled with the progression from centralized to distributed computing, gave birth to the need for efficient maintenance and manipulation of tremendous amounts of data. Data warehouses are a result of the need for maintaining large amounts of data. Data mining is the offspring of the necessity for improved manipulation of data. The following sections provide an analysis of the data mining process.

2.1 Data Mining Overview

Data mining, also known as knowledge discovery, is "the efficient discovery of valuable, non-obvious information from a large collection of data." (Bigus, 1996) The concept of data mining not only requires the discovery of new relationships within a set of data, but also necessitates that such a discovery is efficient, i.e., the benefits of finding these relationships outweigh the costs. Furthermore, the newly gleaned information must provide value and non-obviousness, yielding a competitive advantage by adding significant value to the decision-making process. The process of data mining is composed of three fundamental steps: data preparation, data mining algorithm selection and implementation, and data analysis (Bigus, 1996).

2.2 Data Preparation

. Before analysis is possible, the data must be acquired. However, simply having the data is inadequate -- the data must be of sufficient quality and quantity. The data might not be in a desirable format and may need to be converted into a flat file or placed in a database. Furthermore, the data may contain incorrect or inconsistent information. All these problems are tackled when preparing the data.

The initial step of data preparation is estimated to utilize anywhere from 50 percent to 80 percent of the resources in a data mining project. Data preparation consists of the cleansing, selection, pre-processing, and representation of data (Bigus, 1996).

2.2.1 Data Cleansing

When data warehouses were treated essentially as archival storage mechanisms, the consistency and correctness of data were not viewed as major issues. However, in modern data mining practices, the quality of data plays an integral role in the accurate manipulation of information. Missing and inaccurate values in the dataset are "cleansed" by a variety of methods. For example, rule-based techniques are used to evaluate each data item in terms of range, constraints, or relationships. If missing, incorrect, or outlying data is discovered, statistical methods can be used to replace these fields with neutral or corrected values. (Bigus, 1996)

2.2.2 Data Selection

The selection process includes the collection of relevant data and sub-fields appropriate to the task at hand. Because necessary data might be spread across a variety of incompatible databases, the useful data must be separated from the irrelevant data, and consolidated in a useable and accessible format. Determining what values and fields are pertinent to the task can be determined manually or by statistical or computational procedures. (Bigus, 1996)

2.2.3 Data Pre-processing

To efficiently manipulate the data, it must be pre-processed and refined. A variety of methods for enhancing data exist. For example, because quantity does not necessitate quality, a large number of data categories may not mean that an equal amount of content is supplied. Merging tables or columns in a database can reduce redundant information or convert it to a more relevant form. Furthermore, data can be transformed to appropriate values by normalization or scaling.

In the past, storage and processing constraints forced datasets to sacrifice the amount of information for the sake of minimizing storage space, as evidenced by the Y2K problem. Data fields may need to be combined or separated to represent a new and more relevant attribute; such computed attributes are calculated by methods such as ratios or products. Data transformation can also be performed through the normalization or scaling of fields. If the data mining algorithm requires specific value types or ranges, then the data can be linearly or logarithmically scaled. Furthermore, data strings and symbols can also be mapped to numerical values.

2.2.4 Data Representation

Data representation plays a major role in the correct operation of the data mining algorithm; not only does it ensure the correctness of values, but also influences the training time of neural networks. The two major types of data are continuous numeric values and discrete values. Continuous numeric values, such as gross income and prices, are often scaled and/or normalized to fit a specified range.

Discrete values can take the form of categories or numbers. Discrete categorical values such as {yes, no, unknown} questions, can be represented by numerical or

Boolean digits. Each category must take on a unique value, and if need be, ranking and hierarchical information can be preserved. Discrete numeric values can be represented as numerical or binary digits as well, each presenting different methods of distinguishing values and their relative distances. Binary strings are often preferable to numerical values due to the formatting constraints placed on the input data of the data mining algorithms.

The following methods - one-of-N codes, binary codes, and thermometer codes - each utilize bit strings in a unique fashion. In bit patterns, the relative distance between discrete binary values is measured by Hamming distance, the number of bits that differ between two strings, is used instead of Euclidean distance. The measured distance between bit strings could be used to represent an ordering among values.

A one-of-N code consists of a binary string with a length equal to the number of categories in the variable; each bit represents a distinct category, and this bit is set to 1 while all others are set to 0. For example, a three-category variable with values A, B, and C can be mapped to 100, 010, and 001, respectively. One-of-N codes are easily distinguishable, and any two values always maintain the same Hamming distance. However, if variables have a large number of categories, this method can be costly in terms of bit string length. Because the training time and learning efficiency of neural networks are proportional to the amount of input variables, a one-of-N encoding of one hundred categories which would require one hundred input variables would cripple the neural network. In order to alleviate this problem, the binary encoding scheme could be used.

In a binary code, each category is assigned a value between 1 and N; each category is mapped to its binary equivalent, represented by a binary string of length $\log_2 N$. For example, with four categories of A, B, C, and D, each value would be set to 00, 01, 10, and 11. The string length is dramatically reduced over one-of-N codes, but the Hamming distance has no meaning in this method. Using the example in the previous paragraph, a one-of-N encoding of one hundred categories would require 100 bits, while a binary encoding method would require a mere eight!

If the Hamming distance between values is significant, such as in a hierarchy or ordering of categories or values, then a thermometer code is the optimal choice. A thermometer code, like the one-of-N code, consists of a binary string of a length equal to the number of categories represented. However, Hamming distances are taken into account to create an increasing distance between values. For example, when measuring a four-category variable illustrating height, {short, medium, tall}, the values can be mapped to {001, 011, 111}. In this case, the Hamming distance between short and tall is two, while the Hamming distance between short and medium is only one. A more complex coding scheme could place larger distances between values depending on their perceived differences in value. (Bigus, 1996)

2.3 Algorithm Selection and Implementation

The second data mining step involves the selection and implementation of the data mining algorithm. The selection of the algorithm depends on the nature of the problem, along with the ease of development and deployment. Tasks such as associations, clustering, classification, modeling, and forecasting require different data mining algorithms, ranging from statistics to neural networks. The following table (Table

2.1) illustrates a few of the currently available data mining techniques along with examples of applications and algorithms.

Table 2.1: Data mining techniques

Analysis	Definition and Examples	Common Algorithms
Associations	Find trends across datasets to understand the natural patterns within. Example: market basket analysis.	Statistics
Clustering	A method for performing information-driven, unsupervised segmentation by finding cluster "centroids," i.e., grouping like things together. Example: market segmentation	Neural networks, statistics
Classification	Finding a discrimination/classification between certain things. Unlike clustering, it has specific boundaries and is supervised. Examples: Target marketing, quality control, risk assessment	Decision trees, neural networks
Modeling	Creating a simplified model using inputs and outputs, mainly used for prediction purposes. Examples: Rank/scoring customers, pricing models	Statistics, neural networks
Time-series forecasting	Using input and output data in the past to predict values at some future point. Unlike modeling, it is not a static one-time prediction, but instead it attempts to locate time trends between data points. Examples: stock market analysis, interest rate prediction, sales forecasting	Neural networks

2.4 Output Analysis

The final step in the data mining process is the analysis of the output of the data mining algorithm. Certain algorithms or implementations allow for simple output analysis, providing a graphical, user-friendly result. Other algorithms require expert knowledge in analyzing results, requiring repeated runs of the algorithm. Success depends not only on correct information extraction, but also on compact and coherent presentation of results.

Chapter 3

German Credit Data

3.1 German Credit Data Overview

The data used for regression and neural network analysis is a collection of German credit data, available at http://www.ncc.up.pt/liacc/ML/statlog/. Originally produced by Prof. Hans Hofmann of the Institut f'ur Statistik und "Okonometrie Universit" at Hamburg, the dataset consists of customers' financial and credit information and the resulting classification of customers as "good" or "bad."

3.2 Variable Analysis

The German credit dataset contains financial information about 1000 customers, 700 of which are classified as "good" and 300 of which are classified as "bad." In its original form, it includes twenty independent, input attributes that determine the classification of the customer: seven numerical attributes and thirteen categorical "multiple choice" attributes (Table 3.1). Although no missing data existed in the German credit dataset, various pre-processing methods were performed in order to generate appropriate values; in addition, data was pre-processed differently depending on the type of analysis required.

For the regression analysis, the categorical variables were each mapped to bit strings of specific length, i.e., given *n* categories, the variable was mapped to a bit string of length *n*. The numerical attributes were normalized by the following equation:

$$V_i' = \left(\frac{V_i - \overline{V}}{\sigma} \right)$$

where, V_i' is the normalized value, V_i is the original value, $\bar{V}$ is the mean of $V_1, \ldots, V_n$, and σ is the standard deviation of the values $V_1, \ldots, V_n$. By normalizing the data by this method, the normalized mean, $\bar{V}'$, becomes 0, and the normalized standard deviation, σ' becomes 1, loosely restricting every value of a variable to a limited range. The number of elements in the dataset increased to 61 input variables and one output variable.

The back propagation implementation also required the normalization of numeric values in the same method. Due to the constraints placed by the activation function and the learning algorithm, it was advantageous, if not necessary, to scale and normalize the data to a loose range. Categorical attributes were pre-processed in a different manner, however. Because each categorical variable allowed for only one selection, a unique binary string was assigned to each category. However, a one-of-N code would be costly, expanding the original number of inputs from 20 to 61 variables; as stated earlier, the training time and quality is proportional to the number of inputs. A binary code would maintain the distinct partition without such a drastic increase in input size, where each variable, given n categories, is mapped to a $[\log_2 n]$ bit string, where each bit represents a different input variable. For the neural network paradigm, after normalization and binary code mapping, the number of dependent variables becomes thirty-seven.

Table 3.1: Variable Description

Variable	Type	Value	Regression	Neural Network
Checking account status	categorical	amount < 0 DM	0 0 0 1	0 1
		0 <= amount < 200 DM	0 0 1 0	1 0
		amount >= 200 DM/salary assignments for at least one year	0 1 0 0	1 1
		no checking account	1 0 0 0	0 0
Duration in month	numerical		normalized	normalized
Credit history	categorical	no credits taken/all credits paid back duly	0 0 0 0 1	1 0 0
		all credits at this bank paid back duly	0 0 0 1 0	0 0 1
		existing credits paid back duly until now	0 0 1 0 0	0 1 0
		delay in paying off in the past	0 1 0 0 0	0 1 1
		critical accounts/ other credits existing (not at this bank)	1 0 0 0 0	0 0 0
Purpose	categorical	car (new)	0 0 0 0 0 0 0 0 0 0 1	1 0 1 0
		car (used)	0 0 0 0 0 0 0 0 0 1 0	1 0 0 1
		furniture/equipment	0 0 0 0 0 0 0 0 1 0 0	1 0 0 0
		radio/television	0 0 0 0 0 0 0 1 0 0 0	0 1 1 1
		domestic appliances	0 0 0 0 0 0 1 0 0 0 0	0 1 1 0
		repairs	0 0 0 0 0 1 0 0 0 0 0	0 1 0 1
		education	0 0 0 0 1 0 0 0 0 0 0	0 1 0 0
		vacation	0 0 0 1 0 0 0 0 0 0 0	0 0 1 1
		retraining	0 0 1 0 0 0 0 0 0 0 0	0 0 1 0
		business	0 1 0 0 0 0 0 0 0 0 0	0 0 0 1
		others	1 0 0 0 0 0 0 0 0 0 0	0 0 0 0
Credit amount	numerical		normalized	normalized
Savings account/bonds	categorical	amount < 100 DM	0 0 0 0 1	0 0 1
		100 <= amount < 500 DM	0 0 0 1 0	0 1 0
		500 <= amount < 1000 DM	0 0 1 0 0	0 1 1
		amount >= 1000 DM	0 1 0 0 0	1 0 0
		unknown/no savings account	1 0 0 0 0	0 0 0
Present employment since	categorical	unemployed	0 0 0 0 1	0 0 0
		duration < 1 year	0 0 0 1 0	0 0 1
		1 <= duration < 4 years	0 0 1 0 0	0 1 0
		4 <= duration < 7 years	0 1 0 0 0	0 1 1
		duration >= 7 years	1 0 0 0 0	1 0 0
Installment rate in %-age of disposable income	numerical		normalized	normalized

Table 3.1: Variable Description (cont.)

Variable/Description	Type	Value	Regression	Neural Network
Personal status and sex	categorical	male: divorced/separated	0 0 0 0 1	0 0 0
		female: divorced/separated/married	0 0 0 1 0	0 0 1
		male: single	0 0 1 0 0	0 1 0
		male: married/widowed	0 1 0 0 0	0 1 1
		female: single	1 0 0 0 0	1 0 0
Others debtors/guarantors	categorical	none	0 0 1	0 0
		co-applicant	0 1 0	0 1
		guarantor	1 0 0	1 0
Present residence since	numerical		normalized	normalized
Property	categorical	real estate	0 0 0 1	1 1
		building society savings agreement/life insurance	0 0 1 0	1 0
		car or other	0 1 0 0	0 1
		unknown/no property	1 0 0 0	0 0
Age in years	numerical		normalized	normalized
Other installment plans	categorical	bank	0 0 1	1 0
		stores	0 1 0	0 1
		none	1 0 0	0 0
Housing	categorical	rent	0 0 1	0 0
		own	0 1 0	0 1
		for free	1 0 0	1 0
Number of existing credits at this bank	numerical		normalized	normalized
Job	categorical	unemployed/unskilled - non-resident	0 0 0 1	0 0
		unskilled - resident	0 0 1 0	0 1
		skilled employee/official	0 1 0 0	1 0
		management/self-employed/highly qualified employee/officer	1 0 0 0	1 1
Number of people being liable to provide maintenance	numerical		normalized	normalized
Telephone	categorical	none	0	0
		yes, registed under the customers name	1	1
Foreign worker	categorical	yes	0	0
		no	1	1
Rating (dependent variable)	categorical	"good" score	0	0
		"bad" score	1	1

Chapter 4
REGRESSION

Regression analysis is a statistical technique first developed by Sir Francis Galton in the late nineteenth century to describe the statistical relationship between the heights of fathers and sons (Neter, Wasserman, Kutman, 1990). The following sections describe the functions and features of regression analysis.

4.1 Regression Overview

Regression analysis is a technique used to understand how one set of input variables, the independent variables, affects another set of output variables, the dependent variables. This project exclusively utilizes multiple linear regression, which attempts to find a linear relationship between the input and output variables. While a wide variety of regression algorithms exist in the field of statistics, linear regression still holds as the most popular form of regression analysis.

4.2 Simple Linear Regression

Simple linear regression deals with problems of two variables, the dependent variable, Y, and the independent variable, X, formally describing the tendency of Y to vary with X in a systematic fashion. The scattering of points around the curve representing this statistical relationship can be characterized by the existence of a probability distribution of Y for each value of X, whose means varies systematically with X (Fig 4.1).

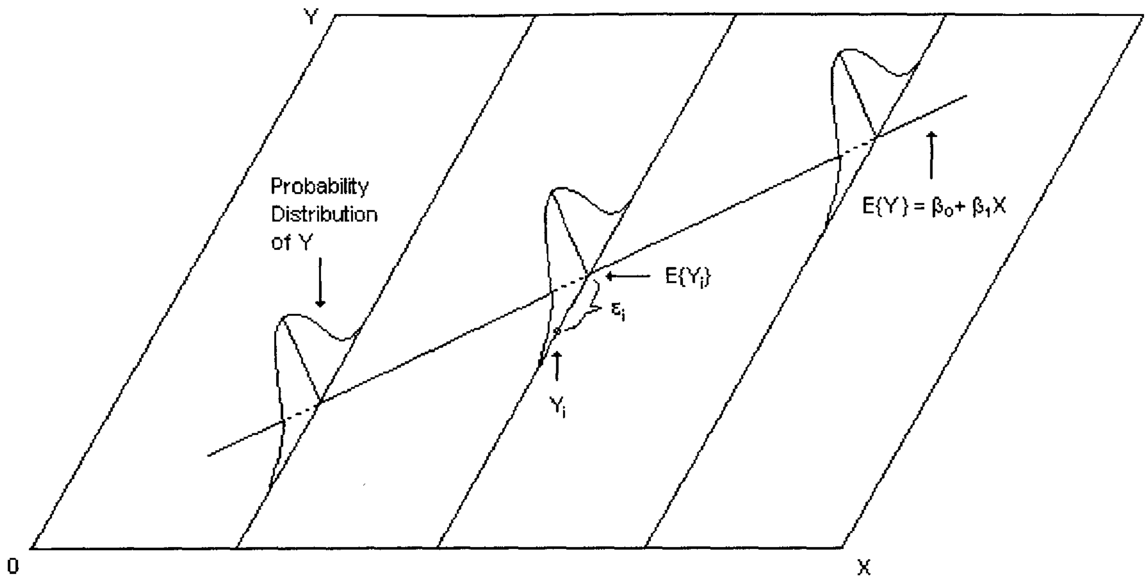


Figure 4.1: A simple linear regression model of dependent variable Y and independent variable X. (Neter, Wasserman, Kutman, 1990).

The relationship can be described by the first-order equation,

$$Y_i = \beta_0 + \beta_1 X_i + \varepsilon_i, \{\text{for } i = 1, \ldots, n\}$$

where, out of n observations, Y_i is the value of the dependent variable in the i-th observation, X_i is the value of the independent variable in the i-th observation, β_0 and β_1 are regression coefficients, and ε_i is the random error/disturbance term in the i-th observation.

Several assumptions are made in the regression model. 1) The dependent variable is a linear function of the independent variable. 2) The expected value of the error term is equal to zero, i.e., $E\{\varepsilon_i\} = 0$. 3) The error terms have the same variance and are uncorrelated, i.e., $\sigma^2\{\varepsilon_i\} = \sigma^2$ and $\sigma\{\varepsilon_i, \varepsilon_j\} = 0$ for all $i \neq 0$ (Neter, Wasserman, Kutman 1990).

From the second assumption of $E\{\varepsilon_i\} = 0$, it follows that:

$$E\{Yi\} = E\{\beta_0 + \beta_1 X_i + \varepsilon_i\} = \beta_0 + \beta_1 Xi + E\{\varepsilon_i\} = \beta_0 + \beta_1 X_i$$

Thus, the regression function is $E\{Yi\} = \beta_0 + \beta_1 X_i$. The task in any regression analysis is to estimate the coefficients β_0 and β_1, and this is commonly done by using the ordinary least squares estimator. Specifically, the goal of the ordinary least squares estimator is to minimize the sum of squared residuals, i.e., (actual - predicted)^2. In the method of least squares, b_0 and b_1 denote the estimators of β_0 and β_1 that minimize the sum of squared residuals.

If Q denotes the sum of squared residuals,

$$Q = \Sigma\,(Y_i - E\{Y_i\})^2 = \Sigma(Y_i - \beta_0 - \beta_1 X_i),$$

and the estimators b_0 and b_1 can be derived by solving the simultaneous equations provided by $\partial Q^2/\partial\beta_0 = 0$ and $\partial Q^2/\partial\beta_1 = 0$:

$$b_1 = [\Sigma(X_i - \overline{X})(Y_i - \overline{Y})]/\Sigma(X_i - \overline{X})^2, \text{ and}$$

$$b_0 = \overline{Y} - b_1\,\overline{X},$$

where $\overline{Y}$ and $\overline{X}$ are the means of Y_i and X_i, respectively, i.e.,

$$\overline{Y} = (\Sigma Y_i)/n, \text{ and}$$

$$\overline{X} = (\Sigma X_i)/n.$$

The estimated regression function can be expressed as follows:

$$\hat{Y}_i = b_0 + b_1 X_i$$

Regression analysis can be viewed in a new light using the analysis of variance (ANOVA) approach. In the ANOVA approach, three key values are calculated in order to provide some insight into the quality of the estimated regression function: the total sum of squares (SSTO), the error sum of squares (SSE), and the regression sum of squares (SSR). (Neter, Wasserman, Kutman 1990)

The variation among the observations of the dependent variable, Y_i, is often measured in terms of the deviation between the observations and the mean. The sum of the squared deviations is the SSTO:

$$SSTO = \sum \left(Y_i - \overline{Y}\right)^2 = \sum Y_i^2 - n\overline{Y}^2$$

Greater values of the SSTO indicate more variation in the observed Y_i values; conversely, if SSTO = 0, then all observations are them same and equal to the mean, i.e., there is no variation.

The variation of the uncertainty in the data can be measured by the variation of the observed Y_i around the regression function. The SSE is the sum of squared deviations between the estimated value and the observed value, which measures the variation of the data with the regression function:

$$SSE = \sum \left(Y_i - \hat{Y}_i\right)^2 = b_1\left[\sum \left(X_i - \overline{X}\right)\left(Y_i - \overline{Y}\right)\right]$$

Note that the SSE is the term that is minimized in the ordinary least squares method of estimation. The greater the value of the SSE, the greater the variation between the predicted and actual output values; if SSE = 0, the regression function is a perfect fit with all predicted data coinciding with observed data.

Finally, using the sum of squared deviations between the estimated value and the mean, the SSR can be calculated. The SSR measures the degree to which the regression relation between the independent and dependent variables account for the variability of the observed Y_i:

$$SSR = \sum\left(\hat{Y}_i - \overline{Y}\right)^2 = b_i^2 \sum\left(X_i - \overline{X}\right)^2$$

The greater the value of the SSR in relation to the SSTO, the more significant the effect of the regression relation in explaining the total variation in the observed Y_i values; if SSR = 0, all predicted values are equal to the mean. (Note that SSTO = SSR +SSE.)

Figure 4.2 gives a graphical illustration of the origins of the deviations that are squared and summed in order to calculate the above three ANOVA values.

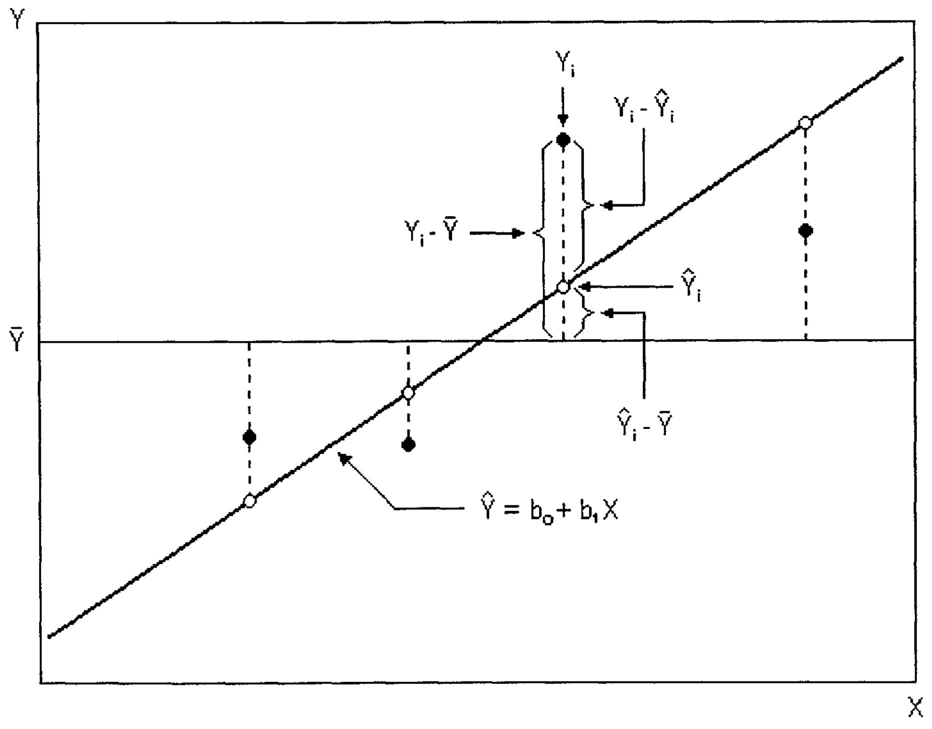


Figure 4.2: Illustration of the deviation values used to calculate SSTO, SSE, and SSR in the ANOVA approach. (Neter, Wasserman, Kutman, 1990)

Two major descriptive measures arise from the above three calculated values that illustrate the degree of linear relation between the independent and dependent variables. The SSTO indicates the variation in the observed dependent variable, Y_i, and thus describes the uncertainty in predicting Y, without taking into account the effect of the independent variable, X. The SSE, on the other hand, indicates the variation in Y_i with the effect of X in the regression model taken into account. Thus, the degree of the influence of X in predicting Y would be measured by r^2, known as the coefficient of determination:

$$r^2 = \frac{SSTO - SSE}{SSTO} = \frac{SSR}{SSTO} = 1 - \frac{SSE}{SSTO}$$

The r^2 value, which ranges from 0 to 1, inclusive, because $SSTO \geq SSE \geq 0$, describes the proportionate decrease in total variation due to the independent variable X. In other words, the closer r^2 is to 1, the greater the degree of linear association. (Neter, Wasserman, Kutman 1990)

The second descriptive measure is the Pearson product moment correlation coefficient, r, which is basically the square root of the r-squared value. However, the sign of r implies the direction of association; if $r < 0$, the regression line is negative, and if $r > 0$, the slope is positive. While the value of r^2 gives a more accurate measure of the extent of linear association, the correlation coefficient gives the direction (Judd, Smith, Kidder, 1991).

4.3 Multiple Linear Regression

Multiple linear regression analysis deals with models that have more than one independent input variable. This technique maps variables to a first-order equation with p-1 independent variables,

$$Yi = \beta 0 + \beta_1 X_{i1} + \beta_2 X_{i2} + \beta_3 X_{i3} + \ldots + \beta_{p-1} X_{i,p-1} + \varepsilon_i, \{\text{for } i = 1, \ldots, n\},$$

where, out of n observations, Y_i is the value of the dependent variable in the i-th observation, X_i, ..., $X_{i,p-1}$ are the values of the independent variables in the i-th observation, β_0, ..., β_{p-1} are regression coefficients, and ε_i is the random error/disturbance term in the i-th observation. Like the method of simple linear regression, the optimal values occur when the least squares are minimized, i.e., (actual - predicted)^2. Note that if p-1 = 1, the equation reduces to the simple linear regression equation; it is evident that simple linear regression is just a subset of multiple linear regression.

It is often simpler to display the information regarding the multiple linear regression model in matrix terms, allowing for a concise presentation of results. The above equation can be expressed as follows:

$$\underset{n\times 1}{\mathbf{Y}} = \underset{n\times p}{\mathbf{X}}\,\underset{p\times 1}{\boldsymbol{\beta}} + \underset{n\times 1}{\boldsymbol{\varepsilon}}$$

where:

$$\underset{n\times 1}{\mathbf{Y}} = \begin{bmatrix} Y_1 \\ Y_2 \\ Y_3 \\ M \\ Y_n \end{bmatrix} \qquad \underset{n\times p}{\mathbf{X}} = \begin{bmatrix} 1 & X_{11} & X_{12} & \Lambda & X_{1,p-1} \\ 1 & X_{21} & X_{22} & \Lambda & X_{2,p-1} \\ M & M & M & & M \\ 1 & X_{n1} & X_{n2} & \Lambda & X_{n,p-1} \end{bmatrix}$$

and

$$\underset{p\times 1}{\boldsymbol{\beta}} = \begin{bmatrix} \beta_0 \\ \beta_1 \\ \mathrm{M} \\ \beta_{p-1} \end{bmatrix} \qquad\qquad \underset{n\times 1}{\boldsymbol{\varepsilon}} = \begin{bmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \mathrm{M} \\ \varepsilon_3 \end{bmatrix}$$

Thus, the expectation and variance-covariance matrix of **Y** are the following:

$$\underset{n\times 1}{\mathbf{E}\{\mathbf{Y}\}} = \mathbf{X}\boldsymbol{\beta}$$

$$\underset{n\times n}{\boldsymbol{\sigma}^2\{\mathbf{Y}\}} = \sigma^2\mathbf{I}$$

where **I** is the identity matrix.

Given that the vector of estimated regression coefficients b_0, b_1, ..., b_{p-1} is the following:

$$\underset{p\times 1}{\mathbf{b}} = \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ \mathrm{M} \\ b_{p-1} \end{bmatrix}$$

which can be calculated to be the following:

$$\underset{p\times 1}{\mathbf{b}} = \underset{p\times p}{(\mathbf{X}^{\mathrm{T}}\mathbf{X})^{-1}}\, \underset{p\times 1}{\mathbf{X}^{\mathrm{T}}\mathbf{Y}}$$

where $\mathbf{X}^{\mathrm{T}}$ is the transpose of matrix **X**, i.e., the matrix that is obtained by exchanging the corresponding rows and columns of **X**. (Neter, Wasserman, Kutman 1990).

Given that the vector of the fitted values of Y and the vector of the residual terms are denoted by the following:

$$\underset{n\times 1}{\hat{\mathbf{Y}}} = \begin{bmatrix} \hat{Y}_1 \\ \hat{Y}_2 \\ M \\ \hat{Y}_n \end{bmatrix} \qquad \underset{n\times 1}{\mathbf{e}} = \begin{bmatrix} e_1 \\ e_2 \\ M \\ e_n \end{bmatrix}$$

The fitted values can be represented by:

$$\underset{n\times 1}{\hat{\mathbf{Y}}} = \mathbf{Xb}$$

and the residual error by:

$$\underset{n\times 1}{\mathbf{e}} = \mathbf{Y} - \hat{\mathbf{Y}} = \mathbf{Y} - \mathbf{Xb}$$

The vector of the fitted values can be represented in terms of **H**, the hat matrix, as follows:

$$\underset{n\times 1}{\hat{\mathbf{Y}}} = \mathbf{HY}$$

where:

$$\underset{n\times n}{\mathbf{H}} = \mathbf{X}(\mathbf{X}^{\mathrm{T}}\mathbf{X})^{-1}\mathbf{X}^{\mathrm{T}}$$

Using the ANOVA approach, the following values of SSTO, SSE, and SSR can be calculated. Their definitions are identical to that of simple linear regression, but they now account for multiple variables. Using the matrix notation adopted by multiple linear regression, the value can be calculated as follows:

$$SSTO = \mathbf{Y}^{\mathrm{T}}\left[\mathbf{I} - \left(\frac{1}{n}\right)\mathbf{J}\right]\mathbf{Y}$$

$$SSE = \mathbf{Y}^{\mathrm{T}}(\mathbf{I} - \mathbf{H})\mathbf{Y}$$

$$SSR = \mathbf{Y}^{\mathrm{T}}\left[\mathbf{H} - \left(\frac{1}{n}\right)\mathbf{J}\right]\mathbf{Y}$$

where J is simply a $n \times n$ square matrix with all elements set to 1.

The coefficient of multiple determination, much like the r^2 measure in simple linear regression, measures the proportionate decrease of total variation in Y due to the set of X variables. It is calculated in the same manner:

$$R^2 = \frac{SSTO - SSE}{SSTO} = \frac{SSR}{SSTO} = 1 - \frac{SSE}{SSTO}$$

However, a large R^2 does not always imply that the fitted model is effective; because adding more independent variables will only increase this coefficient. An adjusted variant, R_a^2, is sometimes used to take into account the number of independent variables:

$$R_a^2 = 1 - \left(\frac{n-1}{n-p}\right)\frac{SSE}{SSTO}$$

Finally, the coefficient of multiple correlation, R is calculated as the positive square root of R^2. Note that both R and R^2 reduce to r^2 and $|r|$, respectively, when there is only one independent variable, i.e., p - 1 = 1.

Chapter 5
Regression Analysis of German Credit Data

5.1 Analysis Overview

Using the Statistical Analysis System (SAS), a multiple linear regression analysis was performed on the German credit dataset. SAS, while providing a simple interface and language, is a powerful tool in statistical analysis. By means of procedures such as PROC REG and PROC MEANS, general regression analysis and statistical summary analysis are performed.

5.2 Results

Using SAS, a variety of information was obtained using a random sample of 800 elements of the German credit dataset -- the remaining 200 samples are used for the purpose of testing the results of the analysis. Table 5.1 displays the estimated regression coefficients that were calculated by SAS, consisting of the 62 estimated least squares coefficients and their respective error values. Note that each variable entry corresponds to each value element in Table 3.1. The values of zero in the parameter estimates indicate that SAS assessed that the variable is a linear combination of other variables.

Table 5.1: Regression Coefficients and Errors from Regression Analysis

Variable	Parameter Estimate (**b**)	Standard Error (**e**)
INTERCEP	0.953858	0.16151761
CHECK1	-0.265077	0.03910634
CHECK2	-0.178446	0.06262315
CHECK3	-0.051681	0.04091709
CHECK4	0	.
DUR	0.05936	0.02002784
HIST1	-0.261543	0.07902081
HIST2	-0.186439	0.08603248
HIST3	-0.162345	0.07775676
HIST4	0.01961	0.10242976
HIST5	0	.
PURP1	-0.234516	0.13692264
PURP2	-0.122461	0.05812375
PURP3	-0.324123	0.14848267
PURP4	0	.
PURP5	0.020663	0.06822443
PURP6	-0.063724	0.09865126
PURP7	-0.112023	0.13837061
PURP8	-0.140081	0.04197163
PURP9	-0.147308	0.04661147
PURP10	-0.242596	0.05674766
PURP11	0	.
AMOUNT	0.040814	0.02339552
SAVINGS1	-0.090618	0.04036039
SAVINGS2	-0.139786	0.06619855
SAVINGS3	-0.087282	0.06168551
SAVINGS4	-0.061028	0.04846472
SAVINGS5	0	.
EMPLOY1	-0.074085	0.07489536
EMPLOY2	-0.16227	0.07899142
EMPLOY3	-0.051881	0.07550533
EMPLOY4	-0.031204	0.07969374
EMPLOY5	0	.
INSTALL	0.055541	0.01641891
PERS1	0	.
PERS2	-0.097902	0.08326194
PERS3	-0.189535	0.07000684
PERS4	-0.085703	0.07169325
PERS5	0	.
DEBT1	-0.158718	0.06756255
DEBT2	0.13551	0.07755413
DEBT3	0	.
RESIDE	-0.000020082	0.01640222
PROP1	0.126025	0.06975242
PROP2	0.030386	0.03945545
PROP3	0.058581	0.04234147
PROP4	0	.
AGE	-0.025218	0.01731042
OTHER1	-0.039642	0.04375603
OTHER2	0.020814	0.08025967
OTHER3	0	.
HOUSING1	-0.108772	0.07927268
HOUSING2	-0.06007	0.04248605
HOUSING3	0	.
NUMCRED	0.026814	0.01812548
JOB1	0.115862	0.11571745
JOB2	0.084233	0.1141398
JOB3	0.074355	0.11751652
JOB4	0	.
NUMMAINT	0.02383	0.01574447
PHONE	-0.054393	0.03339677
FOREIGN	-0.155427	0.08224725

In Table 5.1, the **b** vector represents the set of coefficients in the estimated regression function, and ε represents the error vector of the difference between the actual Y and the predicted Y:

$$\underset{n\times 1}{\hat{\mathbf{Y}}} = \mathbf{Xb} \qquad\qquad \underset{n\times 1}{\mathbf{e}} = \mathbf{Y} - \hat{\mathbf{Y}}$$

Furthermore, the SAS regression output calculated the coefficient of multiple determination to be $R^2 = 0.2899$, and the adjusted coefficient of multiple determination to be $R_a^2 = 0.2445$.

The low R^2 value hints at the lack of success of the multiple linear regression analysis; a low coefficient of multiple determination, as stated previously, indicates that the independent variables have a small linear effect on the dependent variable. Table 5.2 compares the predicted and actual results of the test set, which consists of 139 actual "good" customers and 61 "bad" ones.

Table 5.2: Statistics of Multiple Linear Regression Results

"Good"	"Bad"	"Unclear"
119	31	50

Correct "good"	Correct "bad"	"Unclear" when "good"
69	2	41

Incorrect "good"	Incorrect "bad"	"Unclear" when "bad"
50	29	9

Error "good"	Error "bad"	Total Error
42.02%	93.55%	52.67%

Because this system is meant to aid the decision process of the user, instead of making definitive decisions, the output categories were adjusted from the original, solid divisions of output = 1 being a "bad/rejected" customer, and output = 0 being a "good/approved" customer. Thresholds were instead created, so that an output <= 0.3 denotes a "good" customer, an output >= 0.7 signifies a "bad" customer, and the values between would mean the output was "unclear," indicating that more information is required.

The above method produced very high errors, so the one-of-N code for categorical variables were altered to binary codes, so that both the back propagation algorithm and the multiple linear regression model used the same inputs. The number of input variables was reduced to 37, and the result of the regression analysis showed some improvement (Table 5.3). The parameter estimates and standard errors are also provided (Table 5.4)

Table 5.3: Statistics of Revised Regression Model

"Good"	"Bad"	"Unclear"
97	9	94

Correct "good"	Correct "bad"	"Unclear" when "good"
85	6	52

Incorrect "good"	Incorrect "bad"	"Unclear" when "bad"
12	3	42

Error "good"	Error "bad"	Total Error
12.37%	33.33%	16.48%

Table 5.4: Regression Coefficients and Errors from Regression Analysis

Variable	Parameter Estimate (**b**)	Standard Error (**e**)
INTERCEP	0.387985	0.11626179
CHECK1	0.120594	0.03231143
CHECK2	0.162378	0.03365603
DUR	0.065234	0.02051667
HIST1	0.316275	0.07935352
HIST2	0.119779	0.03901964
HIST3	0.129239	0.04686346
PURP1	-0.054106	0.05636436
PURP2	-0.008588	0.05658868
PURP3	0.054595	0.0340859
PURP4	-0.131973	0.03877627
AMOUNT	0.051088	0.02377912
SAVINGS1	-0.046931	0.0718103
SAVINGS2	-0.008562	0.04201105
SAVINGS3	0.075776	0.03577063
EMPLOY1	-0.121139	0.05448365
EMPLOY2	-0.102458	0.04096365
EMPLOY3	-0.082842	0.03719677
INSTALL	0.060848	0.01675113

Variable	Parameter Estimate (**b**)	Standard Error (**e**)
PERS1	0	.
PERS2	-0.093501	0.04437678
PERS3	0.029386	0.04513822
DEBT1	-0.113834	0.06834357
DEBT2	0.121388	0.07791696
RESIDE	0.000932	0.01655995
PROP1	-0.035917	0.03544979
PROP2	-0.064459	0.03687462
AGE	-0.025599	0.01745425
OTHER1	0.041629	0.04404666
OTHER2	0.06312	0.07419389
HOUSING1	-0.09168	0.06986913
HOUSING2	-0.068404	0.04312479
NUMCRED	0.032	0.01833828
JOB1	0.037931	0.05152164
JOB2	0.040573	0.0434784
NUMMAINT	0.027489	0.01594006
PHONE	-0.070933	0.03375397
FOREIGN	-0.131072	0.08375962

Although total error decreased to 16.48%, the number of "unclear" values accounts for nearly half the training set. This model used the same threshold as the previous, 61 input model.

Chapter 6

Neural Networks

6.1 Neural Networks Overview

Neural networks are models consisting of a mapping between a set of inputs and a set of outputs. The inputs and outputs are connected by parameters known as weights, which are adjusted at every pass of a set of inputs in order to minimize the error between the desired and the actual outputs. These inputs which are passed through the neural network is known as the training set; once the weights are tuned, the neural network need not refer to the training set ever again -- it has "learned" it.

A variety of neural network paradigms exist, and each is best suited for different tasks. Although there are hundreds of variations on the neural network algorithms, there are a few basic models and classes. In general, neural networks can be categorized by the following: the learning paradigms, the connection topology, and the processing functions (Bigus, 1996).

6.1.1 Learning Paradigms

Three basic learning paradigms separate the currently available neural networks: supervised learning, unsupervised learning, and reinforcement learning. The most common training approach, supervised learning, is most often used as a classification and prediction tool. Given a problem in the form of a dataset, the neural network algorithm produces a set of answers; the predicted answers are then compared to the actual answers, and weights are adjusted to ultimately minimize the difference between the predicted and the desired.

The unsupervised learning paradigm differs in the fact that, although it takes in a large amount of data to illustrate the problem, no predicted answers exist. It attempts to find relations and patterns between the input data, working well in problems of clustering. Also called self-organized learning, it takes in no direction in terms of correct or desired output.

Finally, in reinforcement learning, the dataset representing the problem is given as well. However, the answers are not immediately available, and feedback information is either inexact or unavailable in the present. This paradigm is optimal in cases of economic and control applications.

6.1.2 Network Topology

As stated earlier, another distinguishing characteristic of neural network algorithms is the network topology, i.e., the arrangement of the processing units and their interconnections. The three main categories of feedforward, limited recurrent, and fully recurrent networks determine how the input units, intermediate hidden units, and output units are arranged and connected. The feedforward network is used in situations where all the information is presented at once. The data is processed into input units, then through the hidden units, then through the output units; in between the connections are weights and activation functions to vary the values. To put it simply, the output of each layer is the input of the next.

Recurrent networks are used when current information is available but the sequence is important. Thus prior input records are factored with current records to produce an output. This type of network has connections between adjacent layers, and values flow between these layers until weights are stabilized. Limited recurrent networks only provide two-way connections between part of the layers, while fully recurrent networks supply them for all the layers. Fully recurrent networks are complex and can take an indeterminate amount of time to train -- each input pass must run until the weights stabilize, unlike feedforward networks, which simply run through the network and produce an output. The limited recurrent topology exists as a compromise between the simplicity of a feedforward network and the complexity of a fully recurrent one.

6.1.3 Processing Functions

There is a wide selection of processing functions available, but the best model depends on the data and the requirements. For the purpose of the thesis project, which is a two-category classification problem ("good" or "bad"), three common neural network architectures that are able to handle classification problems are compared. The following table (Table 6.1) summarizes the characteristics of these algorithms.

Table 6.1: Common Neural Network Classification Algorithms

Model	**Training Paradigm**	**Topology**	**Primary Function**
Back propagation	Supervised	Feedforward	Classification, modeling, time-series
Kohonen feature maps	Unsupervised	Feedforward	Clustering, classification

6.2 Neural Network Models

As seen in the table above, both of the algorithms can be used for classification purposes. However, each of these models are vastly different, each exhibiting unique characteristics. The following sections give a brief description of these neural network models.

6.2.1 Kohonen Feature Map

Kohonen feature maps use unsupervised learning in a feedforward topology. Used mainly for clustering and classification purposes, the basic Kohonen network has no hidden units, but has an input layer that is fully connected to the output layer. The distinguishing characteristic of the Kohonen map is that it has "neighborhoods", which take advantage of the relative proximity of nodes. The algorithm utilizes self-organization, where the output/Kohonen layer can be seen as a topological map; the output units rearrange themselves after every pass of the training set, grouping together to form clusters (Fig 6.1). The figure graphically illustrates the nature of the topological map -- the winning node and the adjacent nodes (the "neighborhood") rise in height to indicate the emergence of a cluster, much like a geographical contour map. The model is a competitive neural network, where each of the output units "compete" to become the "winner." The units in the output layer compete when given an input pattern until a "winner" is declared. The winning output unit, whose connection weights are closest to the input pattern, is allowed to adjust its connection weights toward the direction of the input pattern; furthermore, the "neighborhood" of units, the output units in close proximity to the winner, also gets their weights adjusted. As training progresses, output units align themselves in a manner such that a decreasing number of units get updated at every input pattern, i.e., the neighborhood size decreases. With sufficient training, the collection of similar units in neighborhoods becomes more distinct, and clustering information can be gleaned from the topological map.

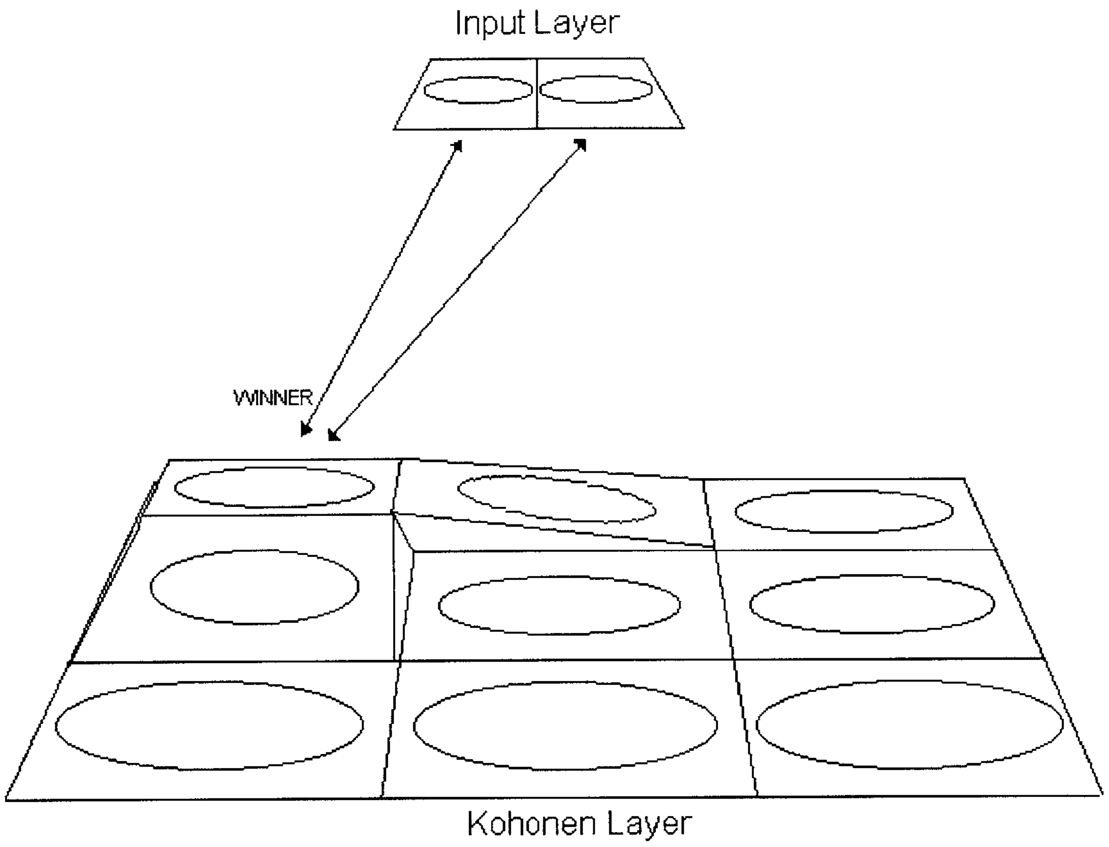


Figure 6.1: Topological map and clustering effect of the Kohonen feature map. The winner gets to adjust its weights.

6.2.2 Back Propagation

The back propagation neural network consists of a supervised learning scheme and feedforward network topology. As the most popular neural network algorithm, it is a powerful, multi-faceted tool, yet costly in terms of computational requirement. The basic algorithm is based on the simple optimization scheme of gradient descent, and there are a huge number of variants. However, the basic method is the most popular, due to its simplicity and its versatility.

The network consists of an input layer, an output layer, and any number of hidden layers in between, although a single hidden layer with a sufficient number of units can perform with a great deal of accuracy. The input pattern is propagated to the output units, and the result is compared to the provided, desired output. Using the error term (desired minus actual), the interconnecting weights are adjusted. Such a process is continued until the error is reduced to under a specified tolerance. (Fig. 6.2)

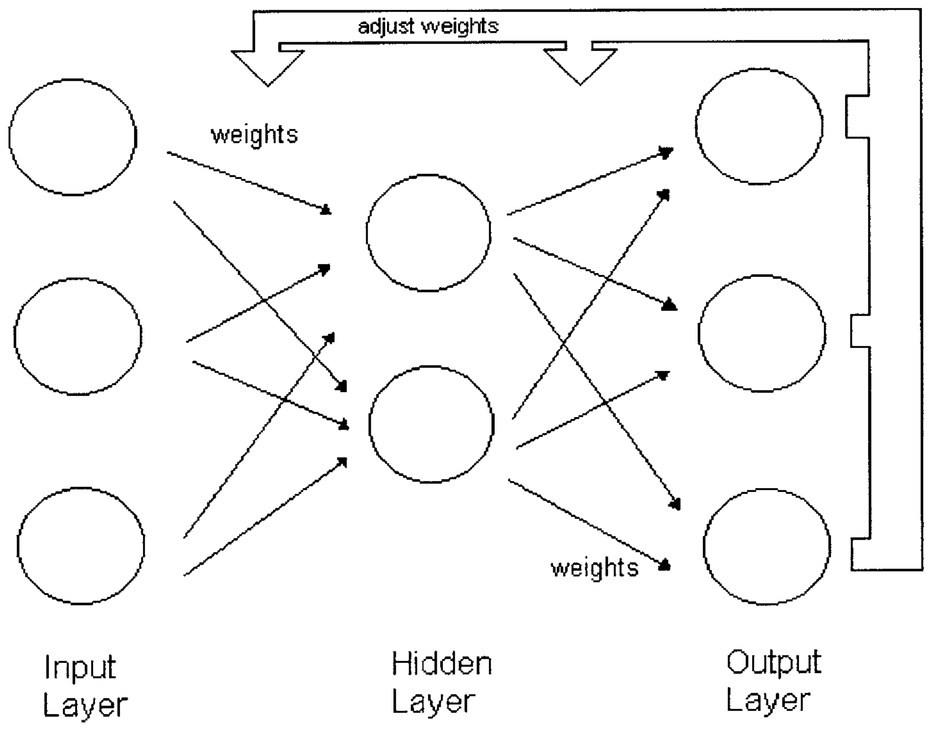


Figure 6.2: Back propagation network illustration.

The back propagation neural network uses a few learning parameters to control the training of the network. The learning rate is used to control the degree of change that could be made to the connection weights after each pass through the network. The momentum term is used to control possible oscillations in the weights, sometimes due to the local minima.

6.3 Neural Network Considerations

From the information above, it is obvious that a myriad of neural network models and architectures exist. However, like any tool, it must be selected and used for the appropriate purpose. The following are some factors that determine the most effective neural network model for a given situation.

The desired function, such as clustering, classification, or modeling, will help determine the appropriate model. For example, the Kohonen feature map is a clustering algorithm, so if the functional requirement were forecasting, such a method would be inappropriate. The input data format, such as binary, continuous, or time-series data, could also determine the model to use. The training speed may also play a factor in deciding upon the most effective architecture. If online learning is not an issue, the back propagation network may be more useful and accurate than another network that trains faster. (Bigus, 1996)

6.4 Neural Network Selection

The above architectures, the Kohonen feature map and the back propagation network, are able to handle the two-category classification problem dictated by the German credit dataset. However, the back propagation network seems best suited for the task at hand. Because the classifications are well defined, and outputs are supplied in the data set, the unsupervised learning paradigm of the Kohonen feature map is unsuitable for the situation. Although online training and learning is slow and inefficient with the back propagation neural network due to the heavy computational load that is required, web-based online testing takes far less processing power. The following chapter describes, in detail, the implementation and results of the back propagation neural network.

Chapter 7

Neural Network Analysis of German Credit Data

7.1 Analysis Overview

A stand-alone back propagation network was developed with the Java programming language. Using the same random sample of 800 entries in the German credit dataset that was used by the multiple regression analysis, the neural network was trained; the remaining 200 items were used to test the network.

7.2 Back Propagation Architecture

The back propagation neural network involves a feedforward topology and utilizes supervised learning. The German dataset is composed of 37 dependent input units, and thus the input layer consists of 37 neurons; the data set also has one binary output unit, so the output layer also has one neuron. Although the number of hidden layers between the input and output layers can be varied, any more than one layer caused overfitting and severely increased the training time. The number of units in the hidden layer is also variable, but the optimal number of hidden units for this system was around nine units. Although it is possible to design a network with a greater number of hidden units and layers, the network is limited by the quantity of data -- training time and the number of connections/weights are governed by the size of the dataset. (Swinger, 1996) More connections require more data. However, the 37-9-1 architecture of the network generalized the dataset sufficiently while maintaining a high degree of accuracy.

7.3 Back Propagation Implementation

There are four basic values that can be adjusted in the training of the neural network, other than the layer size and neuron quantities: the learning factor (β), error tolerance, noise factor, and momentum (α). The learning factor is the parameter used to adjust the weights between neurons. This value is used after the hidden layer and ouptut layers, and dictates whether the neural network learns in leaps or in small steps. The error tolerance is the error value (desired-actual) that is required for the neural network to stop its training. When the error of the network falls below this value, training is halted. The noise factor is the multiplier for the random number that is added to each input pass. The noise factor decreases at regular intervals so that it does not interfere with the convergence to the solution at the end of the training. The noise factor attempts to improve generalization and to overcome local minima. Finally, the momentum term is the multiplier used to calculate weight change; specifically, the weight change is the sum of the β*error*input and α*previous weight change. The momentum term is added in order to force weight changes to go in the same direction, attempting to keep the weights moving to overcome local minima.

The back propagation simulator works as follows:

Weights are initialized to low random values. Given an input pattern of vector **I**, the network multiplies each element of the input pattern (after adjusting for the noise factor) by the connection weights between all the input nodes and each hidden node, the vector **WI**, and then computes the sum, i.e., **I*WI**. This sum is transformed by the sigmoid activation function:

$$f(x) = 1/(1+e^{-x}).$$

Each element of the new vector of hidden unit outputs, **H**, is then multiplied by the connection weights between the hidden and output layers, the vector **WO**, and the sum is computed as **H*WO**. This sum is again transformed by the sigmoid activation function.

After being compared to the desired output, the error is calculated: desired - actual. If the sum of squared errors is less than the error tolerance, the process stops. Or else, weights are adjusted and the process repeats. This implementation was based upon the techniques and code described by Valluru and Hayagriva Rao.

7.4 Results

The back propagation network used a 37-9-1 architecture, i.e., 37 input units for each input variable, a single hidden layer of nine units, and an output layer of one node. The learning factor was set at 0.2, the error tolerance at 0.01, the momentum term at 0.001, and the noise factor at 0.1. These parameters were extensively tested and they appear to be the optimal values for the given architecture, producing a good generalization without over-training.

The German dataset was divided into a random sample of 800 rows for training out of the available 1000, with the remaining 200 used for testing purposes. After training the network, the connection weights were frozen, and the test cases were processed. The same thresholds are used to divide the predicted outputs as in the multiple linear regression analysis, so that an output <= 0.3 indicates a "good" customer, an output >= 0.7 represents a "bad" customer, and the values between are mapped to "unclear," requiring more information. The predicted versus the actual results are compared in Table 7.1. Originally, there are 139 cases where the actual output = 1 and the 61 cases where the output = 0.

Table 7.1: Statistics of Back Propagation Results

"Good"	"Bad"	"Unclear"
131	44	25

Correct "good"	Correct "bad"	"Unclear" when "good"
123	41	13

Incorrect "good"	Incorrect "bad"	"Unclear" when "bad"
8	3	12

Error "good"	Error "bad"	Total Error
6.11%	6.82%	6.71%

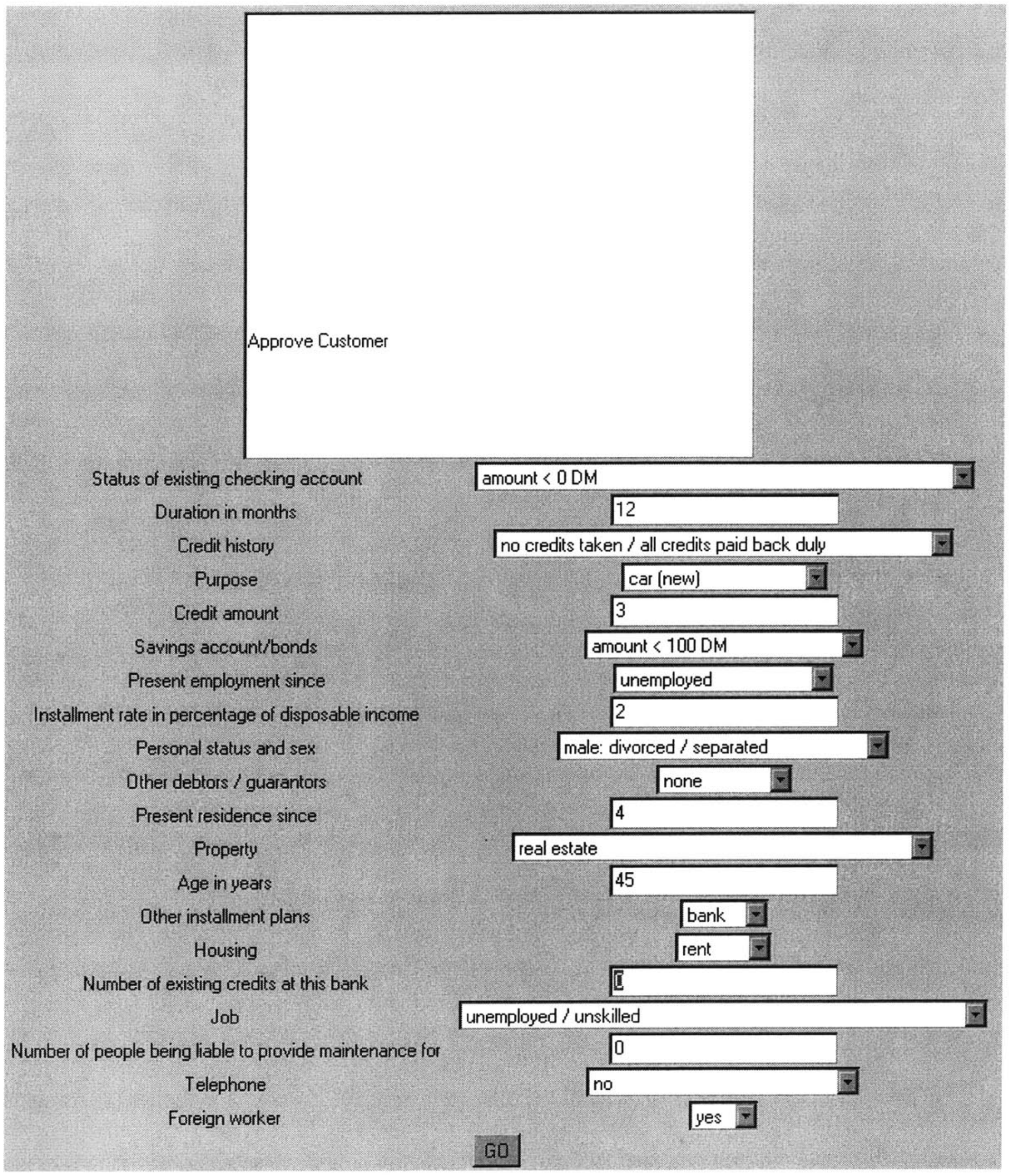
Chapter 8
Web-based Implementation

8.1 Analysis Overview

When deciding to implement the neural network, efficient online functionality was a requirement. Although back propagation may have slow learning and training times, a test-only version that has its connection weights frozen would produce predictions in a reasonable amount of time. Theses weights were obtained from the fully trained back propagation implemented above. The system was developed as a Java applet.

8.2 Results

The web-based back propagation simulator consists of 13 pull-down menus to represent the 13 categorical variables, and 7 text boxes to represent the numerical variables. A large text area is placed on top to display the output of the neural network (Fig. 8.1). After the user enters the information required and hits the "go" button, the input pattern is parsed as a string, after the numerical values are normalized; the categorical values pass bit strings. The string is passed through the neural network with frozen weights, and an output value is produced. If the output <= 0.3, the "good" customer status is indicated in the text area as "Approve Customer"; if the output >= 0.7, the "bad" customer status is indicated as "Reject Customer"; and finally, if the output value falls between the thresholds, the "unclear" status is displayed as "Need More Information."



Approve Customer

Status of existing checking account	amount < 0 DM
Duration in months	12
Credit history	no credits taken / all credits paid back duly
Purpose	car (new)
Credit amount	3
Savings account/bonds	amount < 100 DM
Present employment since	unemployed
Installment rate in percentage of disposable income	2
Personal status and sex	male: divorced / separated
Other debtors / guarantors	none
Present residence since	4
Property	real estate
Age in years	45
Other installment plans	bank
Housing	rent
Number of existing credits at this bank	0
Job	unemployed / unskilled
Number of people being liable to provide maintenance for	0
Telephone	no
Foreign worker	yes

Figure 8.1: Screen shot of web-based neural network simulator tailored for German credit information.

Chapter 9
Discussion

9.1 Result Comparison

The multiple linear regression was performed primarily as a benchmark for the data analysis, and performed poorly as an efficient predictor of customer credit rating -- the total error was 16.48% and nearly half the output was predicted as "unclear." Low R^2 and R_a^2 values of 0.2899 and 0.2445, respectively, suggest that there exist very little linear correlation between the input variables and the output variable. The non-linearity of the dataset may be the reason for the poor performance, along with the size and spread of the values. The back propagation neural network, on the other hand, performed much better, yielding a dramatically lower total error value of 6.71% in predicting outputs.

9.2 Future Considerations

The multiple linear regression model did not produce as successful an output as the back propagation network. However, different regression models could be used in the future to take into account the non-linearity of the dataset. Furthermore, the lack of sufficient data largely dictated the size and shape of the neural network; given enough data, a larger neural network with less constrained inputs could have been developed, perhaps leading to an even more successful model.

Although neural networks generally work better for large amounts of non-linear data, implementation of neural networks is never as straightforward as it is for statistical regression. While regression provides insight into the effects of each variable, neural networks work as a black box. If a successful output is all that is desired, this is fine; but if variable significance is an important factor, especially if the information is to be used to dictate the future gathering of data, neural networks fall short.

Neural networks are not easy to train and to build. Factors such as the number of layers, the number of hidden nodes, and the learning rate all depend on the problem that is being investigated. The design of a neural network involves a combination of trial-and-error and experience -- "neural-net training is an art." (Winston, 1993). However, the back propagation simulator was successfully implemented in this project, and neural networks are definitely the optimal choice.

ACKNOWLEDGEMENTS

A great deal of thanks go to the following:

God, for without Him this would never have been possible.

Prof. David Karger, for his infinite patience and understanding.

Patrick J. Sullivan, Minda Liu, Jeff Getis, and Ken Shih, for giving me the opportunity to do something worthwhile.

Donna Dietz, for the original idea.

Steve Snell, for supplying the help I always seem to need.

Saksiri Tanphaichitr, for the Java expertise.

Bibliography

Agresti, A. *Categorical Data Analysis.* New York: John Wiley & Sons, Inc., 1990.

Bigus, Joseph P. *Data Mining with Neural Networks.* New York: McGraw-Hill, 1996.

Draper, N. and Smith, H. *Applied Regression Analysis, Second Edition.* New York: John Wiley & Sons, Inc., 1981.

Holder, R.L. *Multiple Regression in Hydrology.* Wallingford, Oxfordshire: Institute of Hydrology, 1985.

Johnson, Richard A. and Wichern, Dean W. *Applied Multivariate Statistical Analysis, Third Ed.* Englewood Cliffs, NJ: Prentice-Hall, Inc., 1992.

Judd, Charles M., Eliot R. Smith, and Louise H. Kidder. *Research Methods in Social Relations.* 3rd ed. Fort Worth: Harcourt Brace Jovanovich, 1991.

Kennedy, Peter. *A Guide to Econometrics.* 3rd ed. Cambridge, MA: MIT Press, 1992.

Liu, Minda. Personal Inteview. Dec. 1, 1998.

McCullagh, P. and Nelder, J.A. *Generalized Linear Models.* London: Chapman and Hall, 1989.

Neter, John, William Wasserman, and Michael H. Kutner. *Applied Linear Statistical Models.* 3rd ed. Homewood, IL: Richard D. Irwin, Inc., 1990.

Potts, William J.E. *Data Mining Primer: Overview of Applications and Methods.* Cary, NC: SAS Institute, Inc., 1998.

Rao, Valluru B., Hayagriva B. *Neural Networks and Fuzzy Logic.* New York: MIS Press, 1995.

Rumelhart, David E., et al. *Parallel, Distributed Processing, Vol. 1: Foundations.* Cambridge, MA: MIT Press, 1986.

Sullivan, Patrick J. Personal Interview. Dec. 1, 1998.

Swingler, Kevin. *Applying Neural Networks: A Practical Guide.* London: Academic Press, 1996.

U.S. Chamber of Commerce. *Workers' Compensation Laws.* Washington D.C.: U.S. Chamber of Commerce, 1996.

Winston, Patrick Henry. *Artificial Intelligence*. 3rd ed. Reading, MA: Addison-Wesley, 1993.