Automated Construction of Generalized Additive Neural Networks for Predictive Data Mining

J.V. du Toit

Automated Construction of Generalized Additive Neural Networks for Predictive Data Mining

Jan Valentine du Toit

B.Sc. (Potchefstroomse Universiteit vir Christelike Hoër Onderwys)

B.Sc. Hons. (Potchefstroomse Universiteit vir Christelike Hoër Onderwys)

M.Sc. (Potchefstroomse Universiteit vir Christelike Hoër Onderwys)

YUNIBESITI YA BOKONE-BOPHIRIMA
NORTH-WEST UNIVERSITY
NOORDWES-UNIVERSITEIT

Thesis submitted in the School for Computer, Statistical and Mathematical Sciences at the Potchefstroom Campus of the North-West University in fulfilment of the requirements for the degree Doctor of Philosophy in Computer Science

Supervisor: Prof. D.A. de Waal

Potchefstroom

May 2006

Acknowledgements

Becoming a Doctor of Philosophy is the realization of a childhood dream to achieve something significant using the computer. This dream would not have materialized without the support and guidance of my supervisor, Prof. De Waal, and my parents.

Prof. De Waal introduced me to the academic world. His dedication and positive attitude motivated me to perform my best. During my years in graduate school he also became a mentor providing me with sound advice during our conversations in the Jonge Akker coffee shop. He supported me during the Ph.D. blues and always reminded me that defeat is not an option. Prof. De Waal not only provided me with opportunities to grow as a person, but also set a prime example of being a supervisor.

My parents introduced me to computers at the age of seven and encouraged me to explore this fascinating device. They gave me the opportunity of higher education, believed in me, and showed a keen interest in my studies.

I would also like to express my gratitude to: SAS Institute Inc. for providing me with Base SAS® and SAS® Enterprise Miner™ software used in computing all the results presented in this thesis. This work forms part of the research done at the North-West University within the TELKOM CoE research programme, funded by TELKOM, GRINTEK TELECOM and THRIP. Furthermore, my sincere thanks go to Ms. Cecilia van der Walt for the language editing, everyone who contributed to this thesis and finally, to my Heavenly Father, all honour and gratitude.

Abstract

In this thesis Generalized Additive Neural Networks (GANNs) are studied in the context of predictive Data Mining. A GANN is a novel neural network implementation of a Generalized Additive Model. Originally GANNs were constructed interactively by considering partial residual plots. This methodology involves subjective human judgment, is time consuming, and can result in sub-optimal results. The newly developed automated construction algorithm solves these difficulties by performing model selection based on an objective model selection criterion. Partial residual plots are only utilized after the best model is found to gain insight into the relationships between inputs and the target. Models are organized in a search tree with a greedy search procedure that identifies good models in a relatively short time. The automated construction algorithm, implemented in the powerful SAS® language, is nontrivial, effective, and comparable to other model selection methodologies found in the literature. This implementation, which is called AutoGANN, has a simple, intuitive, and user-friendly interface. The AutoGANN system is further extended with an approximation to Bayesian Model Averaging. This technique accounts for uncertainty about the variables that must be included in the model and uncertainty about the model structure. Model averaging utilizes in-sample model selection criteria and creates a combined model with better predictive ability than using any single model. In the field of Credit Scoring, the standard theory of scorecard building is not tampered with, but a pre-processing step is introduced to arrive at a more accurate scorecard that discriminates better between good and bad applicants. The pre-processing step exploits GANN models to achieve significant reductions in marginal and cumulative bad rates. The time it takes to develop a scorecard may be reduced by utilizing the automated construction algorithm.

Keywords: Akaike Information Criterion, AIC, automated construction algorithm, Bayesian Model Averaging, credit scoring, data mining, Generalized Additive Neural Network, GANN, Generalized Additive Model, GAM, interactive construction algorithm, model averaging, neural network, partial residual, predictive modeling, Schwarz Information Criterion, SBC.

Uittreksel

In hierdie proefskrif word Veralgemeende Additiewe Neurale Netwerke (VANN'e) bestudeer binne die konteks van voorspellende Data-ontginning. 'n VANN is 'n interessante neurale netwerkimplementering van 'n Veralgemeende Additiewe Model. VANN'e is oorspronklik interaktief gekonstrueer deur parsiële residu-grafieke te beskou. Hierdie metodologie behels subjektiewe menslike oordeel, is tydrowend en kan suboptimale resultate tot gevolg hê. Die nuut ontwikkelde outomatiese konstruksie-algoritme los hierdie probleme op deur modelpassing te doen wat gebaseer is op 'n objektiewe modelpassing-kriterium. Parsiële residu-grafieke word slegs gebruik nadat die beste model gevind is om insig te verkry in die verwantskappe tussen die invoere en die teiken. Modelle word georganiseer in 'n soekboom met 'n gretige soekprosedure wat goeie modelle identifiseer in 'n relatief kort tydperk. Die outomatiese konstruksie-algoritme wat in die kragtige SAS®-taal geïmplementeer is, is nie eenvoudig nie, is effektief en vergelykbaar met ander modelpassing-metodologieë wat in die literatuur aangetref word. Hierdie implementering, wat AutoGANN genoem word, het 'n eenvoudige, intuïtiewe en gebruiker-vriendelike koppelvlak. Die AutoGANN-stelsel word verder uitgebrei met 'n benadering tot die Bayes Model Gemiddelde. Hierdie tegniek bring onsekerheid oor die veranderlikes wat by die model ingesluit moet word asook onsekerheid oor die modelstruktuur in berekening. Die tegniek van modelgemiddeldes gebruik insteekproefmodelpassingskriteria en skep 'n gekombineerde model met beter voorspellingsvermoë as enige enkele model. In die veld van Kleinhandel Kredietrisiko word daar nie aan die teorie agter die skep van telkaarte verander nie, maar 'n voorverwerkingstap word geskep om 'n akkurater telkaart te verkry wat beter onderskei tussen goeie en slegte aansoekers. Die voorverwerkingstap maak gebruik van VANN-modelle om beduidende verminderings in rand- en kumulatiewe onreëlmatigheidkoerse te kry. Die tyd wat dit neem om 'n telkaart te ontwikkel kan verminder word deur van die outomatiese konstruksie-algoritme gebruik te maak.

Sleutelwoorde: Akaike Inligtingskriterium, AIC, outomatiese konstruksie-algoritme, Bayes Model Gemiddelde, Kleinhandel Kredietrisiko, Data-ontginning, Veralgemeende Additiewe Neurale Netwerk, VANN, Veralgemeende Additiewe Model, VAM, interaktiewe konstruksie-algoritme, modelgemiddelde, neurale netwerk, parsiële residu, voorspellende modellering, Schwarz Inligtingskriterium, SBC.

Contents

"We are drowning in information, but starving for knowledge."

John Naisbett

1 Introduction

Ever since data collection was invented by the Sumerian and Elam peoples living in the Tigris and Euphrates river basin some 5,500 years ago (using dried mud tablets marked with tax records), people have been trying to understand the meaning of, and get use from, collected data (Pyle, 1999). This data has led to theories, observations, and equations that describe the natural world and its laws. The ancient Chinese, Egyptians and later the Greeks measured the sides of right triangles and induced what is now known as the Pythagorean Theorem. Before them, people observed the movements of the moon, the sun, and the stars and created calendars to describe heavenly events. Without the assistance of computers, people have been analyzing data and looking for patterns before recorded history even began.

What started to change during the past few centuries, though, has been the systematizing of the mathematics and the creation of machines to facilitate the taking of measurements, their storage, and their analysis. This led to a growing data glut problem at the end of the previous century which has been brought to the worlds of science, business, and government. Contributing factors include the widespread introduction of bar codes for almost all commercial products, the computerization of many business and government transactions and advances in data collection tools ranging from scanned text and image platforms to satellite remote sensing systems. In addition, popular use of the World Wide Web as a global information system has created a tremendous amount of data and information. Data storage technology also advanced with faster, higher capacity, and cheaper storage devices, better database management systems, and data warehousing technology which allowed us to transform the data deluge into "mountains" of stored data.

Our capabilities for collecting and storing data of all kinds have far surpassed our abilities to analyze, summarize, and extract knowledge from this data. Traditional methods of data analysis such as spreadsheets and ad-hoc queries simply do not scale to handling very large data sets since they are based mainly on the human dealing directly with the data. These methods can create informative reports from data, but cannot analyze the contents of those reports to focus on important knowledge.

A new generation of intelligent tools for automated data mining and knowledge discovery is needed to deal with the data glut. These tools must have the ability to intelligently and automatically assist humans in analyzing the mountains of data for nuggets of useful knowledge. The emerging field of knowledge discovery in databases (KDD) deals with these techniques and tools. This need has been recognized by researchers in different areas, including artificial intelligence, data warehousing, on-line analysis processing, statistics, expert systems, and data visualization (Fayyad, Grinstein & Wierse, 2002). Growing interest in data mining and discovery in databases, combined with the realization that the specialists in these areas were not always aware of the state of the art in other areas, led to the organization of a series of workshops on knowledge discovery in databases. The last workshop held in 1994 was upgraded to the First International Conference on Knowledge Discovery and Data Mining the next year.

The notion of finding useful patterns (or nuggets of knowledge) in raw data has been given various names in the past, including knowledge mining from databases, data mining, knowledge extraction, information harvesting, information discovery, data analysis, pattern analysis, data pattern processing, and data archaeology. In 1989, the term *Knowledge Discovery in Databases* was coined by Piatetsky-Shapiro (2002) to refer to the broad process of finding knowledge in data, and to emphasize the "high-level" application of particular data mining methods. Statisticians, data analysts, and the MIS (Management Information Systems) community have commonly used the term data mining, while KDD has been mostly used by artificial intelligence and machine learning researchers.

In the next section, a formal definition of Knowledge Discovery in Databases is given and the different components of the definition are discussed. In Section 1.2, the broad KDD process is explained. One of the steps of KDD, data mining, is discussed in Section 1.3. Furthermore, an overview of data mining methods are given. The primary tasks of data mining are explained and the three main components of data mining algorithms are discussed. The important concept of a model is considered. Neural networks, probably the most common data mining method (Berry & Linoff, 1997), is considered and the motivation for this study is explained. In Section 1.4, the key objectives of the thesis are presented and an overview of the rest of the thesis is given.

1.1 Knowledge Discovery in Databases defined

Fayyad, Piatetsky-Shapiro & Smyth (1996) adopt the view that KDD refers to the overall process of discovering useful knowledge from data while data mining refers to the application of certain algorithms for extracting patterns from data without the additional steps of the KDD process. These steps are essential to ensure that useful information (knowledge) is derived from the data. Invalid patterns can be discovered by the blind application of data mining methods and therefore care must be taken to interpret patterns properly. According to Fayyad et al. (1996), KDD can be defined as follows.

Definition 1.1 (Knowledge Discovery in Databases) *Knowledge discovery in databases is the non-trivial process of identifying valid, novel, potentially useful, and ultimately understandable patterns in data.*

Subsequently, each term in the definition of KDD is explained in more detail.

- *Data* is a set of facts F (e.g., cases in a database).
- *Pattern* is an expression E in a language L describing facts in a subset F_E of F. E is called a pattern if it is simpler than the enumeration of all facts in F_E.
- *Process*: Usually in KDD, process is a multi-step procedure which involves data preparation, search for patterns, evaluation of knowledge, and refinement involving iteration after modification. The process must be non-trivial, that is, have some degree of search autonomy.
- *Validity*: The patterns that are discovered should be valid on new data with some degree of certainty. A measure of certainty is a function C mapping expressions in L to a partially or totally ordered measurement space M_C. An expression E in L about a subset $F_E \subset F$ can be assigned a certainty measure $c = C(E, F)$.
- *Novel*: The patterns must be novel (at least to the system). Novelty can be measured by considering changes in data (by comparing current values to previous or expected values) or knowledge (the relationship of a new finding to old ones). In general it is assumed that novelty can be measured by a function $N(E, F)$, which can be a boolean function or a measure of degree of novelty or unexpectedness.
- *Potentially Useful*: Potentially, the patterns should lead to some useful actions, as measured by some utility function. Such a function U maps expressions in L to a partially or totally ordered measure space M_U as in $u = U(E, F)$.
- *Ultimately Understandable*: One of the goals of KDD is to make patterns understandable to humans in order to facilitate a better understanding of the underlying data. This is difficult to

measure, but one frequent substitute is the simplicity measure. Several measures of simplicity exist, ranging from the purely syntactic (e.g., the size of the pattern in bits) to the semantic (e.g., how easy it is for humans to comprehend in some setting). It is assumed that this is measured, if possible, by a function S mapping expressions E in L to a partially or totally ordered measure space M_S by $s = S(E, F)$.

An important concept, called interestingness, is usually taken as an overall measure of pattern value, combining simplicity, usefulness, novelty, and validity. Some KDD systems have an explicit interestingness function given by $i = I(E, F, C, N, U, S)$ which maps expressions in L to a measure space M_I. Interestingness is indirectly defined by other systems via an ordering of the discovered patterns.

By using the notions listed above, knowledge can be defined as viewed from the narrow perspective of KDD. This definition is by no means philosophical or even the popular view. The purpose of this definition is to specify what an algorithm used in a KDD process may consider as knowledge.

Definition 1.2 (Knowledge) *A pattern $E \in L$ is called knowledge if for some user specified threshold $i \in M_I$, $I(E, F, C, N, U, S) > i$.*

This definition of knowledge is by no means absolute, it is purely user-oriented, and determined by whatever thresholds and functions the user chooses. As an example, one instantiation of this definition is to choose some thresholds $c \in M_C$, $s \in M_S$, and $u \in M_U$, and calling E knowledge if and only if $C(E, F) > c$ and $S(E, F) > s$ and $U(S, F) > u$.

By setting the thresholds appropriately, one can emphasize accurate predictors or useful patterns over others. There is an infinite space of how the mapping I can be defined and such decisions are left to the specifics of the domain and the user.

Definition 1.3 (Data Mining) *Data Mining is one step in the KDD process consisting of specific data mining algorithms that, under some acceptable computational efficiency limitations, produces a specific enumeration of patterns E_j over F.*

Often, the space of patterns is infinite and the enumeration of patterns involves a certain form of search in this space. Severe limits are placed by the computational efficiency constraints on the subspace that can be explored by the algorithm.

Definition 1.4 (KDD Process) *KDD process is the process of utilizing data mining algorithms (methods) to extract (identify) what is deemed knowledge according to the specification of thresholds and measures, using the database F along with any required preprocessing, subsampling, and transformations of F.*

The data mining step of the KDD process is mostly concerned with means by which patterns are extracted and enumerated from the data. Knowledge discovery involves the evaluation and possibly interpretation of the patterns to make decisions on what comprises knowledge and what does not. Furthermore, it also includes the choice of encoding schemes, preprocessing, sampling, and projections of the data before the data mining step. For a more practical account on data mining, refer to Cabena, Hadjinian, Stadler, Verhees & Zanasi (1997), Groth (1998), Weiss & Indurkhya (1998), and Bigus (1996).

Having defined Knowledge Discovery in Databases, a broad outline of its basic steps is subsequently presented.

1.2 The KDD process

The KDD process can be described as being interactive and iterative, involving many steps with several decisions being made by the user (for a more detailed account, refer to Adriaans & Zantinge (1996) and Brachman & Anand (1996)):

1. Develop an understanding of the goals of the end-user, the application domain, and the relevant prior knowledge.

2. Create the target data set by selecting a data set, or focusing on a subset of variables or data samples on which the discovery is to be performed.

3. Data preprocessing and cleaning. This step consists of basic operations such as the removal of noise or outliers if appropriate, gathering the necessary information to model or account for noise, deciding on strategies for handling absent data fields, accounting for time sequence information and known changes.

4. Data reduction and projection. Find useful features to represent the data depending on the goal of the task. The effective number of variables under consideration can be reduced by dimensionality reduction or transformation methods. Also, invariant representations for the data can be used.

5. Select the data mining task. Decide whether the goal of the KDD process is regression, classification, clustering, etc.

6. Choose the data mining algorithm. Select the method(s) to be used for searching for patterns in the data. This involves deciding which models and parameters may be suitable. The particular data mining method must be matched with the overall criteria of the KDD process.

7. Data mining. Search for patterns of interest in a particular representational form or a set of such representations. Examples are rules or trees, regression, clustering, etc.

8. Interpret the mined patterns. This includes the possible return to any of steps 1 to 7 for further iteration.

9. Consolidate the discovered knowledge. Document the knowledge, report it to interested parties, or incorporate this knowledge into the performance system. Included in this step is to check for and resolving potential conflicts with previously believed (or extracted) knowledge.

There may be numerous iterations in die KDD process and loops can occur between any two steps. Historically, most work on KDD has focused on step 7 - the data mining (Friedman, 1997); (Džeroski & Lavrač, 2001); (Han & Kamber, 2001). The other eight steps of the KDD process, however, are also very important for the successful application of KDD in practice.

In this thesis a contribution is made to steps 6 and 7. A recently developed data mining method is studied, evaluated, and automated. This newly developed automated algorithm can perform classification, regression, and feature selection with limited human intervention.

In the next section the data mining component which has received the most attention in the literature is considered. The objective is to present a unified overview of some of the most popular data mining methods currently in use. Often the data mining component of the KDD process involves repeated iterative application of particular data mining methods.

1.3 Overview of data mining methods

The terms *patterns* and *models* are used loosely throughout this chapter. A pattern can be seen as the instantiation of a model, e.g., $f(x) = 7x^2 + x + 1$ is a pattern whereas $f(x) = \alpha x^2 + \beta x + 1$ is considered a model. In this thesis, Generalized Additive Models (GAMs) implemented as neural networks, are studied. These implementations are called Generalized Additive Neural Networks (GANNs).

Data mining involves determining patterns from, or fitting models to observed data. These fitted models make up the inferred knowledge: whether or not the models reflect useful or interesting knowledge is part of the overall, interactive KDD process which usually requires subjective human judgment. In model fitting there are two primary mathematical formalisms used (Fayyad et al., 1996): a logical model is purely deterministic (e.g., $f(x) = \alpha x$), with no possibility of uncertainty in the modeling process, whereas the statistical approach allows for nondeterministic effects in the model (e.g., $f(x) = \alpha x + e$, where e could be a Gaussian random variable). The focus of this study is on the statistical/probabilistic approach to data mining which tends to be the most widely-used basis for practical data mining applications given the typical uncertainty about the precise nature of real-world data-generating processes.

This section begins by considering the primary tasks of data mining. Then it is shown that the data mining methods that perform these tasks consist of three primary algorithmic components:

model representation, model evaluation, and search. Also, the important concept of a model is considered in more detail. The section concludes by discussing one of the most popular data mining algorithms, namely neural networks. This technique forms the basis of the thesis.

1.3.1 Primary tasks of data mining

The two "high-level" principle goals of data mining in practice tend to be prediction (Armstrong, 1985) and description. With prediction, some variables or fields in the database are used to predict unknown or future values of other variables of interest. Description focuses on discovering human-interpretable patterns describing the data. The relative importance of prediction and description goals for specific data mining applications can vary significantly. However, description tends to be more important than prediction in the context of KDD. This is in contrast to machine learning (Witten & Frank, 2005) and pattern recognition applications (Ripley, 1996) where prediction is frequently the primary goal. By using the following primary data mining tasks, the goals of prediction and description can be accomplished.

- *Classification* is to learn a function that classifies (maps) a data item into one of several predefined classes.

- *Regression* is to learn a function which maps a data item to a real-valued prediction variable.

- *Clustering* is used for description where one attempts to identify a finite set of categories or clusters to describe the data. The clusters can be mutually exclusive and exhaustive, or consist of a more complex representation such as hierarchical or overlapping clusters.

- *Summarization* consists of methods for finding a compact description for a subset of data. These techniques are often applied to interactive exploratory data analysis and automated report generation.

- *Dependency Modeling* involves methods for finding a model which describes significant dependencies between variables. Models of dependency exist at two levels: the structural level and the quantitative level. The former level of the model specifies (often in a graphical form) which variables are locally dependent on each other, whereas the latter level of the model specifies the strengths of the dependencies using some numerical scale.

- *Change and Deviation Detection* methods focus on discovering the most significant changes in the data from previously measured or normative values.

Having identified the primary tasks of data mining, the next step is to create algorithms to solve them.

1.3.2 Components of data mining algorithms

In any data mining algorithm one can identify three primary components: model representation, model evaluation, and search.

- *Model Representation* is the language L which describes discoverable patterns. The representation must not be too limited, for no amount of training time or examples will then produce an accurate model for the data.

- *Model Evaluation* assesses how well a particular pattern (i.e. a model and its parameters) meet the criteria of the KDD process. Evaluation of the predictive accuracy is based on performing a cross validation or considering in-sample fit statistics. Descriptive quality is evaluated by considering predictive accuracy, novelty, utility, and understandability of the fitted model. Logical and statistical criteria can both be used for model evaluation. Two in-sample model selection criteria, AIC and SBC, are utilized by the new algorithm to evaluate different GANN patterns.

- *Search Method* has two components: parameter search and model search. With parameter search the algorithm must search for the parameters which optimize the model evaluation criteria given observed data and a fixed model representation. Relatively simple problems require no search, the optimal parameter estimates can be obtained in closed form. Typically, for more general models, a closed form solution is not available and therefore greedy iterative methods are commonly used like the gradient descent method of backpropagation for neural networks. Model search is implemented with a loop over the parameter search method: the model representation is changed so that a family of models are considered. For each specific model representation, the parameter search method is instantiated to assess the quality of that specific model. Implementations of model search methods tend to utilize heuristic search techniques since the size of the space of possible models often prohibits an exhaustive search and solutions with a closed form are not easily obtainable. The new technique developed in this thesis performs a search over all GANN models and organizes the models in a search tree. For each GANN model in the tree, parameter estimation is performed and the model is evaluated with a model selection criterion. This search continues until the search space is exhausted or the allowed time has elapsed. The best model found in the tree is then reported.

The important concept of a model provides a common basis for discussing data mining techniques and will be considered next.

1.3.3 Models

A model generates one or more output values for a given set of inputs. The process of data analysis if often the procedure of building an appropriate model for the data. A linear regression, for example, builds a model that is a line with the form $aX + bY + c = 0$ where a, b, and c are the parameters of the model and X and Y are variables. For a given value of X the line can be used to estimate a Y value. A linear regression is one of the simplest models available.

The existence of a model does not guarantee accurate results. Given any set of points, there is some line that "best" fits the points - even when there is no linear relationship between the values. Some models are good and some are bad. Evaluating the results of a model is a critical step in using and developing them.

The linear model just considered is an example of regression - finding the best form of a curve to fit a set of points. A model is of a broader scope and can be used for clustering, classification, and time-series analysis. Models create a common language for talking about data mining.

When models are created for data mining, there are some useful aspects to keep in mind. Two of the dangers of models are underfitting or overfitting the data. Directed and undirected data mining use models in slightly different ways. Some models can explain what is being done better than other models. Finally, some models are easier to apply than other models.

Underfitting and overfitting

Two common problems associated with models are underfitting and overfitting of the data. With overfitting, the model memorizes the data and predicts results based on idiosyncrasies in the particular data used for training. Consequently the model produces good results on the training set, but does not generalize to other data. Overfitting occurs for a number of reasons. Some modeling techniques readily memorize the data if the data set is too small. A simple example of this would be applying a linear regression technique to only two input data points. The regression finds the line that connects the two points exactly. Unfortunately, there is not enough data to determine whether the line is useful. A second reason for overfitting is that the target field is redundant. That is, another field or combination of fields contains the same information as the target field, possibly in another form.

Underfitting occurs when the resulting model does not match patterns of interest in the data. It is common when applying statistical techniques to data. A common cause of underfitting is the elimination of variables that have predictive power but are not included in the model. Another cause for underfitting is that the technique simply may not work well for the data in question.

Supervised versus unsupervised

The difference between supervised (directed) and unsupervised (undirected) data mining occurs when creating the data mining model. In supervised data mining, the target of the model is specified prior to creating the model. The created model then trains on examples where the known target provides feedback into refining the model. In an unsupervised model, the model itself determines its output. The analyst must determine what is interesting about the results. The resulting model can be applied to other data in both cases. In this study only supervised data mining is considered.

Explainability

For some applications, knowing why a particular model produces a particular result is not important. For other purposes, it can be quite insightful and significant. Some models are easier to understand than others. For instance, market basket analysis and decision trees produce clear sets of rules that make sense in English. Neural networks in general and clustering techniques at the other extreme provide little insight into why a particular model does what it does. GANNs discussed in this thesis, however, provide insight into the relationships between the input variables and the target.

Ease of applying the model

Another important aspect is the ease of applying the model to new records. Suppose the data is stored in a relational database, then a model that can be implemented using SQL statements is preferable to one that requires exporting the data into other tools. Vendors of data mining products are increasingly seeing the value of working with relational databases and other data stores. As a result, complex models are implemented using stored procedures in the database. These stored procedures are written in a computer language inside the database.

A wide variety of data mining methods exist. Examples are cluster detection, decision trees, rule induction, example-based methods, genetic algorithms, link analysis, market basket analysis, memory-based reasoning, nonlinear regression and classification methods, on-line analytic processing, probalistic graphical dependency models, relational learning models, and neural networks. A more detailed account of these methods can be found in Berry & Linoff (1997).

1.3.4 Neural networks

Neural networks are likely the most common data mining technique (Berry & Linoff, 1997). Some people even consider them synonymous with data mining. Neural networks (Freeman & Skapura, 1992); (Cheng & Titterington, 1994); (Zhang, Patuwo & Hu, 1998) represent simple models of neural interconnections in the human brain and are adapted for use on digital computers. In their most common form, they learn from a training set, generalizing patterns inside it for classification and prediction.

The first examples of these new systems appeared in the late 1950s from work done by Frank Rosenblatt on a device called the perceptron (Rosenblatt, 1962) and Bernard Widrow on a linear model called Adeline (Widrow & Hoff, 1960). Unfortunately, artificial neural network technology has not always enjoyed the status in the fields of engineering and computer science that it has gained in the neuroscience community. Early pessimism concerning the limited capability of the perceptron caused the field to languish from 1969 until the early 1980s. The appearance of the book, Perceptrons, by Minsky & Papert (1969) is often credited with causing the demise of this technology. Still, during those years research in this field continued. A modern renaissance of neural network technology took place in the 1980s when Rumelhart & McClelland (1986) published their influential research on parallel distributed processing.

New applications and new structures for neural networks are being investigated and appear frequently at various conferences and publications devoted to them. The GANN was introduced by Sarle (1994) when he discussed the relationships between neural networks and statistical models. Potts (1999) used this type of neural network to lessen the practical difficulties with the widespread application of artificial neural networks to predictive data mining. The latter two papers were the only research available on GANNs when this study commenced.

Potts proposed an interactive construction algorithm to build GANNs that requires human judgment to interpret partial residual plots. This judgment is subjective and can result in models that are suboptimal. Also, for a large number of variables this can become a daunting and time consuming task. In this thesis an algorithm to automate the construction of GANNs is developed. This new technique ensures objectivity by incorporating a model selection criterion to guide the model building process. With the automated construction algorithm, partial residual plots are not used primarily for model building, but to provide insight into the structure of the best model found. By automating the construction of GANNs, no human intervention is needed.

In the next section the main objectives and an overview of the rest of the thesis are presented.

1.4 Overview of the thesis

The main objectives of this thesis are to:

1. Show that a GANN, the neural network implementation of a GAM, has predictive power and consequently is worth studying. Furthermore, this type of neural network alleviates the "black box" perception of neural networks in general by providing insights into the relationships between inputs and the target.

2. Develop and implement an automated GANN construction algorithm based on a complete greedy search algorithm that finds "good" models in a reasonable time period. This implemented program must be able to perform GANN model selection and variable selection with results comparable to that of other linear and nonlinear model building techniques found in the literature and current data mining systems.

3. Extend the automated construction algorithm with model averaging to account for model uncertainty.

4. Demonstrate how accurate scorecards may be built using GANNs with potential time savings when the automated algorithm developed in (2) is used.

The rest of the thesis is organized as follows. In Chapter 2, smoothing is discussed which forms the basis of estimating additive models with the backfitting algorithm. GANNs and the relation of this type of model to GAMs are considered. The interactive construction algorithm and partial residual plot are explained. The predictive power of GANNs is demonstrated by two applications. First, an interactive constructed GANN model was entered into the coveted KDD Cup 2004 competition and results comparable to the best entries were obtained. Second, a GANN is built interactively to predict excess returns on the S&P 500. This model is compared to another model building technique and found to be superior.

The new automated construction algorithm for GANNs is discussed in Chapter 3. This technique is an extension of the interactive construction algorithm and based on a best-first search procedure. It is shown that the search for GANN models is complete. The automated algorithm which searches for the best GANN model is illustrated with the Kyphosis data set (Bell, Walker, O'Connor, Orrel & Tibshirani, 1989). Potts (2000) used the Boston Housing data set to construct a GANN model interactively. This model is compared to the best model found by the automated technique. For the Ozone data set (Breiman & Friedman, 1985) it is shown that the automated method gives results comparable to other model building techniques found in the literature. Finally, the implementation of the automated construction algorithm in the SAS® statistical language is discussed. This implementation is called AutoGANN.

Model selection criteria which objectively guide the selection of GANN models are discussed in Chapter 4. The history of the most prominent model selection criteria and two philosophical views on model selection criteria are described. Two information-based model selection criteria, AIC and SBC, are explained. They can be applied to GANN models. Bayesian model averaging (BMA) which accounts for model uncertainty is considered. In general, BMA has not been adopted in practice and an approximation to BMA is discussed. This approximation is implemented in the AutoGANN system. An example of model averaging using the SO_4 data set (Xiang, 2001) is discussed.

The performance of a GANN is compared to that of a logistic regression model on a home equity data set in Chapter 5. Logistic regression is an established method in the field of credit scoring, since it is relatively well understood and an explicit formula can be derived on which credit decisions may be based. The process of scorecard building using the automated construction algorithm is compared to standard scorecard building practice. It is shown that the usual time it takes to build a scorecard may be drastically reduced by using the automated construction technique. Also, more accurate scorecards may be obtained by utilizing the new automated methodology. Finally, Chapter 6 contains a summary of the contributions of this thesis and directions for future research.

"Discovery consists of seeing what everybody has seen and thinking what nobody has thought."

Albert von Szent-Gyorgyi

2 Generalized Additive Neural Networks

In many real-world applications, computers must be able to perform complex pattern recognition tasks. Since conventional sequential computers are not suited to this type of problem, features from the physiology of the brain are used as the basis of these processing models. As a result, the technology has come to be known as artificial neural networks (ANNs) or simply neural networks.

Neural networks have been applied to many real-world situations, among them medical diagnostics, speech recognition, flight control, product inspection, oilwell exploration, terrain classification, coin grading, machine tool controls, and financial forecasting (Gately, 1996); (Kaastra & Boyd, 1996). Financial areas where neural networks have found extensive applications include credit card fraud, bankruptcy prediction, mortgage applications, stock market prediction, real estate appraisal, and option pricing (Gately, 1996).

In this chapter a special type of neural network called a Generalized Additive Neural Network (GANN) is considered that provides the modeler with a new tool to predict the future. When this study commenced, little research on GANNs was available: two articles and a course on implementing neural networks in the SAS® programming language. The GANN was introduced by Warren Sarle in the early 1990s (Sarle, 1994) when he explained what neural networks are, translated neural network terminology into statistical terminology, and discussed the relationships between neural networks and statistical models. He showed that a nonlinear additive model can be implemented as a neural network. Will Potts (Potts, 1999); (Potts, 2000) used GANNs to lessen the practical difficulties with the widespread application of artificial neural networks to predictive data mining. Three of these difficulties are inscrutability, model selection (Zucchini, 2000); (Gallinari & Cibas, 1999); (Snyman, 1994); (Lee, 2000), and troublesome training.

Multilayer perceptrons are usually regarded as black boxes with respect to interpretation. The influence of a particular input on the target can depend in complicated ways on the values of the other inputs. In some applications, such as voice recognition, pure prediction is the goal; understanding how the inputs affect the prediction is not important. In many scientific applications, the opposite is true. To understand is the goal, and predictive power only validates the interpretive power of the model. This is the domain for formal statistical inference such as hypothesis testing and confidence intervals.

Some domains such as database marketing often have both goals. Scoring new cases is the main purpose of predictive modeling. However, some understanding, even informal, of the factors influencing the prediction can be helpful in determining how to market to segments of people likely to respond. Decisions about costly data acquisitions can also benefit from an understanding of the effects of the inputs. In credit scoring, the arcane characteristic of the model can have legal consequences. Creditors are required by the US Equal Credit Opportunity Act (Anonymous, 2006) to provide a statement of specific reasons why an adverse action was taken. A statement that the applicant failed to achieve the qualifying score on the creditor's scoring system is considered insufficient by the regulation.

The second practical difficulty with neural networks is the vast number of configurations from which to choose. Trial and error is the most reliable method for determining the best number of layers, number of units, number of inputs, type of activation functions, type of connections, etc.

The third practical difficulty is the computational effort that is required to optimize the large number of parameters in a typical neural network model. This is partially self-imposed by data analysts that often use inefficient optimization methods such as backpropagation. Even with an efficient algorithm, local minima are troublesome. Different starting values can converge to different, and sometimes faulty, solutions. Often, the best remedy is to have multiple runs from different random starting values.

GANNs have constraints on their architecture that reduce these difficulties. The effect of each input on the fitted model can be interpreted by using graphical methods. Partial residual plots can be used to determine visually the network complexity. With the addition of direct connections, GANNs can be initialized using Generalized Linear Models.

A GANN is the neural network implementation of a Generalized Additive Model (GAM) and forms the basis of this study. A discussion on GANNs would not be complete without considering GAMs and the backfitting algorithm for estimation of GAM models. In Section 2.1 smoothing is considered, which summarizes the trend of a response measurement as a function of one or more predictor measurements. The running-mean smoother is used to illustrate smoothing and the bias-variance trade-off is explained for deciding on the value of the smoothing parameter. In Section 2.2 additive models are discussed. The backfitting algorithm for estimating additive models is explained and an extension to additive models, the GAM, is discussed. A special type of smoother, the scatterplot smoother, is utilized by the backfitting algorithm. In Section 2.3 the GANN architecture and interactive construction algorithm are discussed. This methodology is utilized to build GANN models in the fields of physics and finance that will hopefully shed light on two very important issues. First, how efficient is the interactive construction methodology when trying to find a good model? Second, do GANNs provide the modeler with adequate predictive power? Answers to these two questions will provide an impetus for further research into this interesting topic. Quantum physics particles generated by high energy collisions are classified by a GANN in Section 2.4. Finally, a GANN is constructed to predict stock market excess returns in Section 2.5.

2.1 Smoothing

The linear model holds a central place in the toolbox of the applied statistician since it is simple in structure, elegant in its least-squares theory, and interpretable by its user. However, this type of model does not need to work alone. Recently, computers exploded in speed and size which allows the data analyst to augment the linear model with new methods that assume less and therefore potentially discover more. In Section 2.2 one of these new methods, the additive model (Hastie & Tibshirani, 1990), is described. This model is a generalization of the linear regression model. Basically, the linear function of an input is replaced with an unspecified smooth function. The additive model consists of a sum of such functions. The functions have no imposed parametric form and are estimated in an iterative manner by using scatterplot smoothers.

The estimated additive model consists of a function for each of the inputs. This is useful as a predictive model and can also help the data analyst to discover the appropriate shape of each of the input effects.

By assuming additivity of effects, the additive model retains some of the interpretability of the linear model. A fast computer is required to estimate the univariate functions and this estimation would have been computationally unthinkable thirty-five years ago. Fortunately, with the current computing power it is feasible on a personal computer. The additive model is a prime example of how the power of the computer can be used to free the data analyst from making unnecessarily rigid assumptions about the data.

A smoother is a tool that summarizes the trend of a response measurement Y as a function of one or more predictor measurements $X_1, \ldots, X_p$. An estimate of the trend is produced that is less variable than Y itself; hence the name smoother. A smoother has a nonparametric nature, it does not assume a rigid form for the dependence of Y on $X_1, \ldots, X_p$. Consequently, a smoother is often referred to as a tool for nonparametric regression. The running-mean (moving average) is a simple example of a smoother. On the other hand, a regression line is not strictly thought of as a smoother because of the rigid parametric form. The estimate produced by a smoother is called a smooth. The single predictor case is the most common and called scatterplot smoothing.

The Diabetes data set is utilized to illustrate scatterplot smoothing and comes from a study of the factors affecting patterns of insulin-dependent diabetes mellitus in children (Sockett, Daneman, Clarson & Ehrich, 1987). The objective of the study was to investigate the dependence of the level of serum C-peptide on various other factors to understand the patterns of residual insulin secretion. The response measurement is the logarithm of C-peptide concentration found at diagnosis and the predictor measurements are age and base deficit, a measure of acidity. These two predictors are a subset of the factors studied in Sockett et al. (1987).

Smoothers have two main purposes. The first purpose is description. With a scatterplot smoother the visual appearance of the scatterplot of Y versus X is enhanced. This helps the data analyst to pick out the trend in the plot. In Figure 2.1 a plot of log(C-peptide) versus age is shown. It appears that log(C-peptide) has a strong dependence on age and a scatterplot smoother can provide assistance in describing this relationship.

The second purpose of a smoother is to estimate the dependence of the mean of Y on the predictors, and consequently serve as a building block for the estimation of additive models.

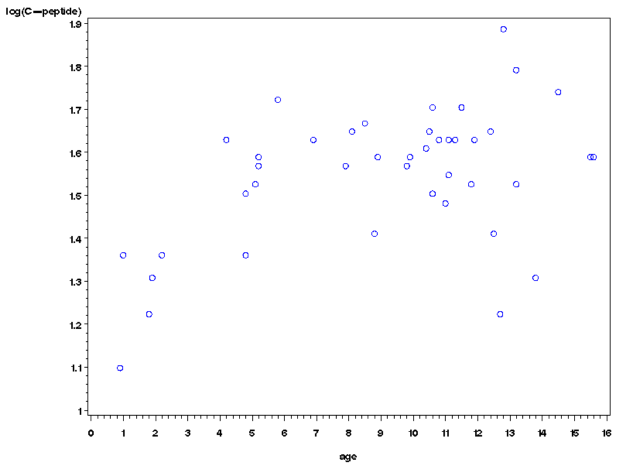


Figure 2.1: Scatterplot of *log(C-peptide)* versus *age*

Most smoothers perform local averaging, that is, averaging the Y-values of observations having predictor values close to a target value. The averaging is done within neighbourhoods around the target value. In scatterplot smoothing there are two main decisions to be made:

1. how the response values in each neighbourhood must be averaged and

2. how large to make the neighbourhoods.

The first decision concerns the type of smoother to use, because smoothers differ mainly in their method of averaging. The second decision is typically expressed in terms of an adjustable smoothing parameter. Intuitively, small neighbourhoods will produce an estimate with high variance but potentially low bias, and conversely for large neighbourhoods. As a result there is a fundamental trade-off between bias and variance, controlled by the smoothing parameter. The amount of smoothing is calibrated according to the number of equivalent degrees of freedom.

In the next section a formal definition of scatterplot smoothing is given.

2.1.1 Scatterplot smoothing defined

Suppose response measurements[1] $\mathbf{y} = (y_1, \ldots, y_n)^T$ exist at design points $\mathbf{x} = (x_1, \ldots, x_n)^T$. It is assumed that each of $\mathbf{y}$ and $\mathbf{x}$ represent measurements of variables Y and X. In most of the cases it is useful to think of Y, and sometimes X, as having been generated by some random mechanism, but this is not necessary for the current discussion. Also, the pairs (x_i, y_i) need not be a random sample from some joint distribution.

Since Y and X are noncategorical, not many replicates at any given value of X is expected. For convenience it is assumed that the data are sorted by X and for the present discussion, there are no tied X-values, that is, $x_1 < \ldots < x_n$. In the case of ties, weighted smoothers can be used.

A scatterplot smoother is defined as a function of $\mathbf{x}$ and $\mathbf{y}$ of which the result is a function s with the same domain as the values in $\mathbf{x} : s = \mathcal{S}(\mathbf{y}|\mathbf{x})$. Usually the set of instructions that defines $s(x_0)$, which is the function $\mathcal{S}(\mathbf{y}|\mathbf{x})$ evaluated at x_0, is defined for all $x_0 \in [-\infty, \infty]$. At other times, $s(x_0)$ is defined only at $x_1, \ldots, x_n$, the sample values of X. In the latter case some kind of interpolation must be done in order to obtain estimates at other X-values.

Hastie & Tibshirani (1990) discuss a number of scatterplot smoothers including bin smoothers, running-line smoothers, kernel smoothers, regression splines, cubic smoothing splines, locally-weighted running-line smoothers, and running-mean smoothers. The latter smoother is explained in more detail in the next section to illustrate the trade-off between bias and variance. Decisions about the complexity of models is governed by this important trade-off.

2.1.2 The running-mean smoother

Suppose the target value x_0 equals one of the $x_j s$, say x_i. If there are replicates at x_i, the average of the Y-values at x_i can be used for the estimate $s(x_i)$. Lets assume there are no replicates. In the latter case Y-values corresponding to X-values close to x_i are averaged. How are points close to x_i picked? A simple way is to choose x_i itself, as well as the k points to the left of x_i and the

[1]Note that $(y_1, \ldots, y_n)^T$ denotes the transpose of the vector $(y_1, \ldots, y_n)$.

k points to the right of x_i that are closest in X-value to x_i. This is called a symmetric nearest neighbourhood and the indices of these points are denoted by $N^S(x_i)$. The running-mean is then defined by

$$s(x_i) = ave_{j \in N^S(x_i)}(y_j).$$

When it is not possible to take k points to the left or right of x_i, as many points as possible are taken. A symmetric nearest neighbourhood can be formally defined as

$$N^S(x_i) = \{max(i-k,1), \ldots, i-1, i, i+1, \ldots, min(i+k,n)\}.$$

For target points x_0 other than the x_i in the sample it is not obvious how to define the symmetric nearest neighbours. One solution is to simply interpolate linearly between the fit of two values of X in the sample adjacent to x_0. Alternatively, symmetry can be ignored and the r closest points to x_0 can be taken, regardless of which side they are on. This procedure is called a nearest neighbourhood. Arbitrary values of x_0 are handled in a simple and clean way.

This smoother is also called a moving average, and is popular for evenly-spaced time-series data. The moving average smoother is valuable for theoretical calculation because of its simplicity, but in practice it does not work very well. It tends to be so wiggly that it hardly deserves the name smoother. Furthermore, it tends to flatten out trends near the endpoints and consequently can be severely biased. In Figure 2.2 a running-mean smooth with $k = 11$ or about 25% of the 43 observations is shown.

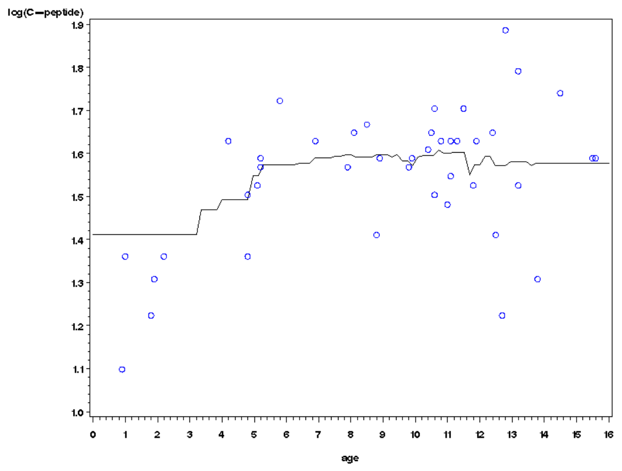


Figure 2.2: Smoother with 25% span

2.1.3 Smoothers for multiple predictors

So far a smoother for a single predictor has been discussed, that is, a scatterplot smoother. When more than one predictor is present, say $X_1, \ldots, X_p$, the problem is one of fitting a p-dimensional surface to Y. A simple, but very limited, estimate of the surface is provided by the multiple regression of Y on $X_1, \ldots, X_p$. Conceptually, it is easy to generalize the running-mean to this setting. This smoother requires a definition of nearest neighbourhood of a point in p-space. Here, nearest is determined by a distance measure and the most obvious choice is Euclidean distance. The notion of symmetric nearest neighbours is no longer meaningful when $p > 1$. After a neighbourhood is defined, the generalization of the running-mean estimates the surface at the target point by taking the average of the response values in the neighbourhood.

Hastie & Tibshirani (1990) argue that multi-predictor smoothers are not very useful for more than two or three predictors. These type of smoothers have many shortcomings, such as difficulty of interpretation and computation, which provides an impetus for studying additive models in this thesis. The shortcomings refer to the generic multivariate smoothers as described here and not to some of the adaptive multivariate nonparametric regression methods, which might also be termed surface smoothers. The latter were designed to overcome some of these objectionable aspects.

2.1.4 The bias-variance trade-off

In the previous sections, no formal relationship between the response Y and the predictor X is assumed. This assumption is now made to lay the groundwork for additive models. It is assumed that

$$Y = f(X) + \epsilon \tag{2.1}$$

where the expected value of ϵ, $E(\epsilon)$, is 0 and the variance of ϵ, $var(\epsilon)$, is σ^2. Also, assume that the errors ϵ are independent. Model (2.1) states that $E(Y|X = x) = f(x)$. In this formal setting, the goal of a scatterplot smoother is to estimate the function f. Note that the fitted functions are denoted by $\hat{f}$ rather than the s used in the previous sections. The running-mean can be seen as estimates of $E(Y|X = x)$ since this smoother is constructed by averaging Y-values corresponding to x-values close to a target value x_0. The averaging involves values of $f(x)$ close to $f(x_0)$. Since $E(\epsilon) = 0$, this implies $E\{\hat{f}(x_0)\} \approx f(x_0)$. For a cubic smoothing spline, it can be shown that as $n \to \infty$ and the smoothing parameter $\lambda \to 0$, under certain regularity conditions, $\hat{f}(x) \to f(x)$, where n denotes the number of design points and λ denotes the window-width, also known as the bandwidth. This says that as more and more data are obtained, the smoothing-spline estimate will converge to the true regression function $E(Y|X = x)$.

There is a fundamental trade-off between the bias and variance of the estimate in scatterplot smoothing. This trade-off is controlled by the smoothing parameter. In the case of the running-mean, the trade-off can be easily seen. The fitted running-mean smooth can be written as

$$\hat{f}_k(x_i) = \sum_{j \in N_k^S(x_i)} \frac{y_j}{2k+1} \tag{2.2}$$

with expectation

$$E\{\hat{f}_k(x_i)\} = \sum_{j \in N_k^S(x_i)} \frac{f(x_j)}{2k+1} \tag{2.3}$$

and variance

$$var\{\hat{f}_k(x_i)\} = \frac{\sigma^2}{2k+1}. \tag{2.4}$$

For ease of notation it is assumed that x_i is near the middle of the data so that $N_k^S(x_i)$ contains the full $2k+1$ points. From (2.3) and (2.4) it can be seen that increasing k decreases the variance but tends to increase the bias since the expectation $\sum_{j \in N_k^S(x_i)} f(x_j)/(2k+1)$ involves more terms with function values, $f(\cdot)$, different from $f(x_i)$. In a similar way, decreasing k increases the variance but tends to decrease the bias. This phenomenon is the trade-off between bias and variance, also encountered when adding or deleting terms from a linear regression model.

Figures 2.3, 2.4, and 2.5 show the running-mean smooths using 20%, 50%, and 80% of the 43 observations for the diabetes data. From these figures it can be seen that a larger percentage of observations produce smoother but flatter curves.

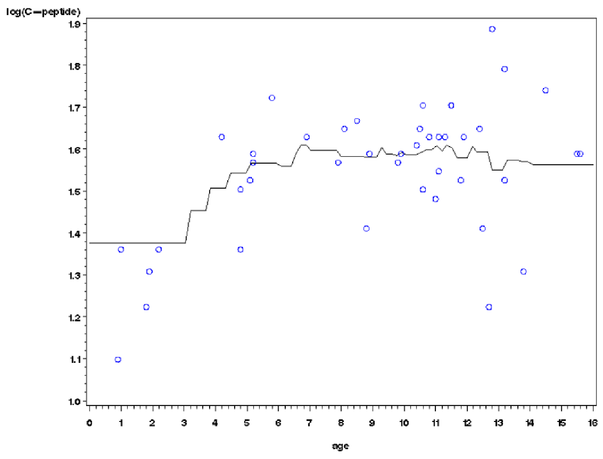


Figure 2.3: Smoother with 20% span

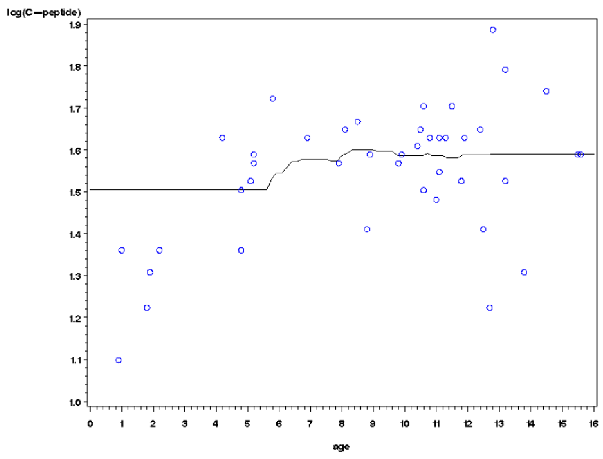


Figure 2.4: Smoother with 50% span

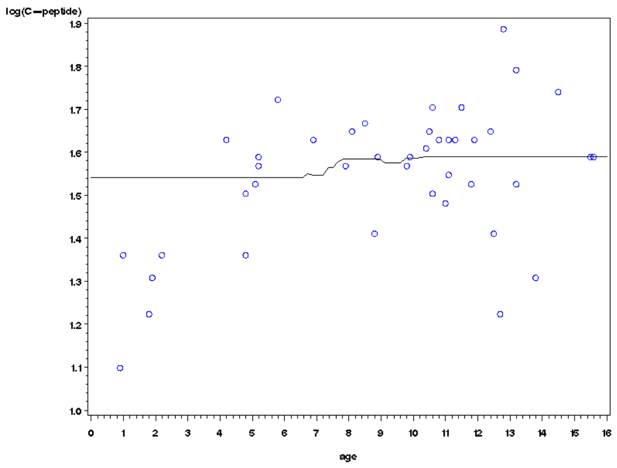


Figure 2.5: Smoother with 80% span

In the next section the additive model for multiple regression data is discussed as well as the backfitting algorithm for its estimation. This algorithm utilizes scatterplot smoothers to determine the functional form of the additive model.

2.2 Additive models

The additive model is a generalization of the usual linear regression model. It is important to outline the limitations of the linear model and why one might want to generalize it. A natural generalization would be an arbitrary regression surface. Unfortunately, there are problems with the estimation and interpretation of fully general regression surfaces and these problems lead one to restrict attention to additive models.

2.2.1 Multiple regression and linear models

With a multiple regression problem there are n observations on a response variable Y, denoted by $\mathbf{y} = (y_1, \ldots, y_n)^T$ measured at n design vectors $\mathbf{x}^i = (x_{i1}, \ldots, x_{ip})$. The points $\mathbf{x}^i$ may be chosen beforehand, or may be measurements of random variables X_j for $j = 1, \ldots, p$, or both. These two situations are not distinguished.

The goal is to model the dependence of Y on $X_1, \ldots, X_p$ for several reasons.

- Description. A model is utilized to describe the dependence of the response on the predictors so that more can be learned about the process that produces Y.
- Inference. The relative contributions of each of the predictors in explaining Y is assessed.
- Prediction. The data analyst wishes to predict Y for some set of values $X_1, \ldots, X_p$.

The standard tool utilized by the applied statistician for these purposes is the multiple linear regression model

$$Y = \alpha + \alpha_1 X_1 + \ldots + \alpha_p X_p + \epsilon \tag{2.5}$$

where $E(\epsilon) = 0$ and $var(\epsilon) = \sigma^2$. A strong assumption about the dependence of $E(Y)$ on $X_1, \ldots, X_p$ is made by the model, namely that the dependence is linear in each of the predictors. The linear regression model is extremely useful and convenient when this assumption holds, even roughly, since

- a simple description of the data is provided,
- the contribution of each predictor is summarized with a single coefficient, and
- a simple method is provided for predicting new observations.

The linear regression model can be generalized in many ways. Surface smoothers is one class of candidates and can be thought of as nonparametric estimates of the regression model

$$Y = f(X_1, \ldots, X_p) + \epsilon. \tag{2.6}$$

One problem with surface smoothers is choosing the shape of the neighbourhood that defines local in p dimensions. A more serious problem common to all surface smoothers has been called the curse of dimensionality by Bellman (1961). The difficulty is that neighbourhoods with a fixed number of points become less local as the dimensions increase.

A number of multivariate nonparametric regression techniques have been devised, partly in response to the dimensionality problem. Examples are recursive-partitioning regression and projection pursuit regression (Friedman & Stuetzle, 1981). When sufficient data is available, these models have good predictive power. Under suitable conditions they are all consistent for the true regression surface. Unfortunately, these methods all suffer from being difficult to interpret. Specifically, how is the effect of particular variables examined after a complicated surface has been fitted?

The interpretation problem emphasizes an important characteristic of the linear model that has made it so popular for statistical inference: the linear model is additive in the predictor affects. Once the linear model has been fitted, the predictor effects can be examined separately in the absence of interactions. Additive models retain this important feature by being additive in the predictor effects.

2.2.2 Additive models defined

The additive model is defined by

$$Y = \alpha + f_1(X_1) + \ldots + f_p(X_p) + \epsilon \tag{2.7}$$

where the errors ϵ are independent of the $X_j s$, $E(\epsilon) = 0$ and $var(\epsilon) = \sigma^2$. The $f_j s$ are unspecified univariate functions, one for each predictor. Implicit in the definition of additive models is that $E\{f_j(X_j)\} = 0$. If this were not the case there would be free constants in each of the functions.

With the additive model an important interpretive feature of the linear model is retained: the variation of the fitted response surface holding all but one predictor fixed does not depend on the values of the other predictors. This follows from the fact that each variable is represented separately in (2.7). Consequently, once the additive model is fitted to data, the p univariate functions can be plotted separately to examine the roles of the predictors in modeling the response. Unfortunately, such simplicity comes at a price; the additive model is almost always an approximation to the true regression surface but hopefully a useful one. When a linear regression model is fitted, it is not generally believed that the model is correct. Rather, it is believed that the model will be a good first order approximation to the true surface, and that the important predictors and their roles can be uncovered using the approximation. Additive models are more general approximations than linear regression models.

The estimated functions of an additive model correspond to the coefficients in linear regression. All the potential difficulties encountered in interpreting linear regression models apply to additive models and can be expected to be more severe. Care must be taken not to interpret functions for variables that are insignificant and not have them affect the important functions.

Next, the backfitting algorithm for estimating additive models is described.

2.2.3 Fitting additive models

The formulation and estimation of additive models can be approached in many ways. Hastie & Tibshirani (1990) discuss a number of methods including multiple linear regression, more general versions of multiple regression, regression splines, and smoothing splines. The most general method estimates the functions by an arbitrary smoother. The general backfitting algorithm enables the data analyst to fit an additive model using any regression-type fitting mechanism. The algorithm is an iterative fitting procedure and this is the price paid for the added generality.

A simple intuitive motivation for the backfitting algorithm is provided by conditional expectations. If the additive model (2.7) is correct, then for any k,

$$E(Y - \alpha - \sum_{j \neq k} f_j(X_j) | X_k) = f_k(X_k) \tag{2.8}$$

where α is the constant term. The conditional expectations (2.8) immediately suggest an iterative algorithm for computing all the $f_j s$, which is presented next in terms of data and arbitrary scatterplot smoothers $\mathcal{S}_j$.

1. Initialize: $\alpha = ave(y_i), f_j = f_j^0, j = 1, \ldots, p$
2. Cycle: $j = 1, \ldots, p, 1, \ldots, p, \ldots$
   $f_j = \mathcal{S}_j(\mathbf{y} - \alpha - \sum_{k \neq j} \mathbf{f}_k | \mathbf{x}_j)$
3. Continue 2. until the individual functions do not change.

When the univariate function f_j is being readjusted, the effects of all the other variables are removed from $\mathbf{y}$ before smoothing this partial residual[2] against x_j. This is only appropriate if all the functions are also correct, and therefore the iteration.

Initial functions (f_j^0) must be provided to start the algorithm and without prior knowledge of the functions, a sensible starting point might be the linear regression of $\mathbf{y}$ on the predictors. The

[2]Note that $(\mathbf{y} - \alpha - \sum_{k \neq j} \mathbf{f}_k | \mathbf{x}_j)$ denotes the partial residual in the backfitting algorithm.

backfitting algorithm is often nested within some bigger iteration, in which case the functions from the previous big iteration loop provide starting values. Hastie & Tibshirani (1990) discuss the convergence of the backfitting algorithm for a number of different types of smoothers. Although no proof of convergence exists for certain types of smoothers like locally-weighted running-line smoothers, their experience has been very promising and counter examples are hard to find.

The discussion so far deals with additive models where the mean of the response is modelled as an additive sum of the predictors. These type of models extend the linear regression model. In the next section an additive extension of the family of Generalized Linear Models is described. The latter type is a generalization of linear regression models. With Generalized Linear Models the predictor effects are assumed to be linear in the predictors, but the distribution of the responses and the link between the predictors and this distribution can be general.

2.2.4 Generalized Additive Models defined

Generalized Additive Models extend Generalized Linear Models in the same manner that the additive model extends the linear regression model.

The Generalized Linear Model (McCullagh & Nelder, 1989) is defined as

$$g_0^{-1}(E(Y)) = \alpha_0 + \alpha_1 X_1 + \ldots + \alpha_p X_p + \epsilon \tag{2.9}$$

where $E(\epsilon) = 0$ and $var(\epsilon) = \sigma^2$. A link function, g_0^{-1}, is used in (2.9) to constrain the range of the response values and is the inverse of the (neural network) activation function g_0. When the expected response is bounded between 0 and 1, such as probability, the logit link function given by

$$g_0^{-1}(E(Y)) = \ln\left(\frac{E(Y)}{1 - E(Y)}\right)$$

is appropriate. For an expected response that is bounded between -1 and 1, the hyperbolic tangent link function given by

$$g_0^{-1}(E(Y)) = 1 - \frac{2}{1 + \ln(2E(Y))}$$

can be used. A Generalized Additive Model (GAM) (Hastie & Tibshirani, 1987); (Wood, 2006) is defined as

$$g_0^{-1}(E(Y)) = \alpha + f_1(X_1) + \ldots + f_p(X_p) + \epsilon \tag{2.10}$$

where $E(\epsilon) = 0$ and $var(\epsilon) = \sigma^2$.

Multilayer perceptrons (MLPs) are the most widely used type of neural network for supervised prediction. Theoretically, MLPs are universal approximators that can model any continuous function (Ripley, 1996). For this reason, MLPs can be used as the univariate functions of GAMs. When GAMs are implemented as neural networks, backfitting is unnecessary, since any training method suitable for fitting MLPs can be utilized to simultaneously estimate the parameters of GANN models. As a result, the usual optimization and model complexity issues also apply to GANN models.

In the next section the GANN architecture and an iterative algorithm for constructing GANNs are presented. This methodology guides the modeler in visually deciding on the appropriate complexity of the individual univariate functions.

2.3 Interactive construction methodology

A MLP that has a single layer with h hidden neurons has the form

$$g_0^{-1}(E(y|\mathbf{x})) = w_0 + w_1\text{tanh}(w_{01} + \sum_{j=1}^{p} w_{j1}x_j) + \ldots + w_h\text{tanh}(w_{0h} + \sum_{j=1}^{p} w_{jh}x_j). \tag{2.11}$$

The link-transformed expected value of the target is expressed as a linear combination of non-linear functions of linear combinations of the inputs (2.11). The activation function used for the hidden layers in this case is the hyperbolic tangent function suggested by (Potts, 2000). This non-linear regression model has $h(p+1)+1$ unknown parameters (weights and biases). The parameters are estimated by numerically optimizing some suitable measure of fit to the training data such as the negative log likelihood.

The basic architecture for a GANN has a separate MLP with a single hidden layer of h units for each input variable, given by

$$f_j(x_j) = w_{1j}\text{tanh}(w_{01j} + w_{11j}x_j) + \ldots + w_{hj}\text{tanh}(w_{0hj} + w_{1hj}x_j).$$

The overall bias α absorbs the individual bias terms. Each individual univariate function has $3h$ parameters, where h could vary across inputs. The architecture can be enhanced to include an additional parameter for a direct connection (skip layer)

$$f_j(x_j) = w_{0j}x_j + w_{1j}\text{tanh}(w_{01j} + w_{11j}x_j) + \ldots + w_{hj}\text{tanh}(w_{0hj} + w_{1hj}x_j)$$

so that the Generalized Linear Model is a special case.

An example of a GANN with two inputs is shown in Figure 2.6. Each input has a skip layer; the first input has two nodes in the hidden layer and the second input has three nodes in the hidden layer. Nodes in the consolidation layer correspond to the univariate functions, and weights between these nodes and the output node are fixed at 1.0. In this example, the first univariate function is given by

$$f_1(x_1) = w_{01}x_1 + w_{11}\text{tanh}(w_{011} + w_{111}x_1) + w_{21}\text{tanh}(w_{021} + w_{121}x_1)$$

and the second univariate function is

$$f_2(x_2) = w_{02}x_2 + w_{12}\text{tanh}(w_{012} + w_{112}x_2) + w_{22}\text{tanh}(w_{022} + w_{122}x_2) + w_{32}\text{tanh}(w_{032} + w_{132}x_2).$$

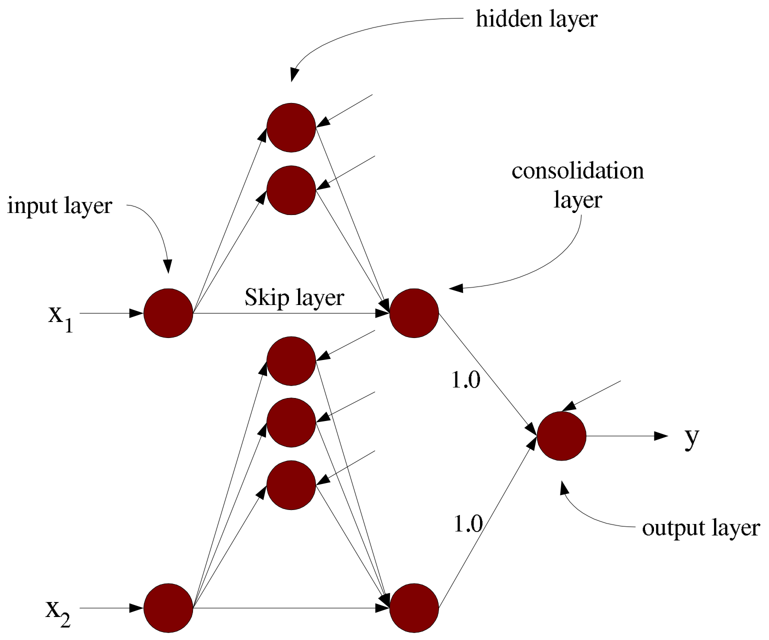


Figure 2.6: Generalized Additive Neural Network

For more than half a century, a variety of diagnostic plots have been used to asses nonlinear relationships between the target and input variables in multiple regression models. In general, there are two complementary approaches to examining the assumption of linearity: informal graphical methods (Cai & Tsai, 1999) and formal tests. Ezekiel (1924) introduced an informal graphical method that was termed the partial residual plot by Larsen & McCleary (1972). This method is still used frequently.

The visual diagnostics used to aid the model selection process for GANNs are plots of the fitted univariate functions, $\hat{f}_j(x_j)$, overlaid on the partial residuals

$$pr_j = g_0^{-1}(y) - \alpha - \sum_{l \neq j} \hat{f}_l(x_l) = (g_0^{-1}(y) - g_0^{-1}(\hat{y})) + \hat{f}_j(x_j)$$

versus the corresponding jth input[3].

With partial residuals the effect of the individual inputs, adjusted for the effect of the other inputs, can be investigated. The jth partial residual is the deviation between the actual values and that portion of the fitted model that does not involve x_j.

The construction process is started with a single neuron and a skip layer for each input instead of the linear model. The linear fit is only used for initialization. The effectiveness of partial residual plots for visualizing the underlying curve is discussed by Berk & Booth (1995). They demonstrated that the partial residuals based on a GAM fit are more reliable than those based on a linear fit. Also, starting with 4 parameters is common practice with GAM estimation.

The following set of instructions for constructing a GANN interactively (Potts, 1999) takes advantage of their constrained form to simplify optimization and model selection.

[3]When g_0^{-1} is nonlinear, a first order approximation is usually used: $pr_j = \frac{\partial g_0^{-1}(\hat{y})}{\partial y}(y - \hat{y}) + \hat{f}_j(x_j)$.

```
1. Construct a GANN with one neuron in the hidden layer and a skip layer for each input
   in the model. In this step the univariate functions are initialized to
   $f_j(x_{ji}) = w_{0j}x_{ji} + w_{1j}tanh(w_{01j} + w_{11}x_{ji})$.
   This gives 4 parameters for each input. Binary inputs only have a direct connection.
2. Fit a Generalized Linear Model to give initial estimates of $\alpha$ and the $w_{0j}$.
3. Initialize the remaining 3 parameters in each hidden layer as random values from a
   normal distribution with mean zero and variance equal to 0.1.
4. Fit the full GANN model.
5. Examine each of the fitted univariate functions overlaid on their partial residuals.
6. Prune (remove neurons) the hidden layers with apparently linear effects and grow (add
   neurons) the hidden layers where the nonlinear trend appears to be underfitted. If this
   step is repeated, the final estimates from previous fits can be used as starting values.
```

Algorithm 2.1: Interactive Construction Algorithm

In the next section the interactive construction algorithm is illustrated with an example from the quantum mechanics field. This example will provide insight into the effectiveness of the interactive construction methodology as well as the power that can be obtained by utilizing a GANN model.

2.4 Quantum Physics example

An informed list of the most far-reaching scientific developments of the 20th century is likely to include general relativity, big bang cosmology, the unraveling of the genetic code, evolutionary biology, and quantum mechanics (Kleppner & Jackiw, 2000). Among these, the latter is unique because of its profoundly radical quality.

Quantum mechanics was created to describe an abstract atomic world far removed from daily experience, but its impact on our daily lives could hardly be greater. The dramatic advances in biology, chemistry, medicine, and in essentially every other science could not have occurred without the tools made possible by quantum mechanics. The creation of quantum physics has transformed our world, bringing with it all the advantages and risks of a scientific revolution.

Max Planck created the quantum concept in 1900 when he hypothesized that the total energy of a vibrating system cannot be changed continuously. Instead, the energy must move from one value to another in discrete steps, or quanta, of energy. The idea of energy quanta was so markedly new that Planck let it lay fallow. Einstein recognized the implications of quantization for light in 1905, but even then the concept was so bizarre that there was little basis for progress. It took twenty more years and a fresh generation of physicists to create modern quantum theory.

Among the greatest feats of the revolution is that quantum mechanics has provided a quantitative theory of matter. Currently we understand essentially every detail of atomic structure; the Periodic Table has a simple and natural explanation and the vast arrays of spectral data fit into an elegant theoretical framework. With quantum theory the quantitative understanding of molecules, solids, liquids, conductors, and semiconductors is permitted. Quantum mechanics provides essential tools for every advanced technology and all of the sciences.

Quantum physics actually includes two entities. First, the theory of matter at the atomic level called quantum mechanics. With quantum mechanics we can understand and manipulate the material world. The second entity is the quantum theory of fields. The problem that motivated quantum field theory was the question of how an atom sends out rays of light as its electrons "jump" from excited states to the ground state. Quantum field theory is the theory of fields, not only electromagnetic fields, but other fields that were later discovered.

Each year the international KDD Cup knowledge discovery and data mining competition is held in conjunction with the Association for Computing Machinery's Knowledge Discovery and Data Mining (ACM KDD) Conference. Du Toit & De Waal (2004) took part in the competition which focused on a problem from the quantum mechanics field. A GANN was constructed interactively for the binary classification of atomic particles generated by high energy collisions. This competition, the development of the GANN model, and results obtained are subsequently discussed.

2.4.1 KDD Cup competition

KDD Cup is the most rigorous competition in the field of predictive technology and data mining with participants from all over the world competing to showcase their capabilities. The competition is open to all industries and hundreds of individuals from around the world participated in 2004. More than 500 teams from 49 countries registered for the competition and 65 teams entered submissions.

Two data sets were supplied by the organizers. The training set consists of 50,000 cases and the test set is of size 100,000. Each case has 78 attributes, indicated by $v1, \ldots, v78$, describing the route along which the particle traveled and a binary response which denotes the particle type (B or B-bar, indicated by 0 and 1 respectively). These two types are distributed 50% (B) and 50% (B-bar) in the training and test sets. No detail information was provided regarding the 78 attributes and what the B and B-bar particles represent. After considering descriptive statistics, variables $v47, v48, v49, v50$, and $v51$ were removed from the model since they contained only zeros and would not be useful in describing variation in the response.

In 2004, the competition focused on various performance measures for supervised classification where rules had to be designed that optimize a particular performance measure (e.g. accuracy, and area under the ROC curve, to be defined later). For the test set, multiple sets of predictions had to be submitted. Each set had to maximize the performance according to a particular measure. Four metrics were used to rank the entered models and are described next.

- *Accuracy* is the number of cases predicted correctly, divided by the total number of cases. An accuracy of 1 is a perfect prediction and accuracy near 0 is poor. The predictions used when measuring accuracy can be boolean or continuous. A threshold is associated with continuous predictions. Predictions above or equal to this threshold will be treated as class 1, and predictions below the threshold are treated as class 0.

- An *ROC curve* is a plot of the true positive rate versus the false positive rate as the prediction threshold sweeps through all the possible values. An area under the ROC curve of 1 is a perfect prediction. When the area is 0.5 it denotes a random prediction. An area below 0.5 indicates that there is a relationship between predicted values and truth, but in this case the model is backwards and predicts smaller values for positive cases.

- *Cross-entropy* measures how close predicted values are to target values. It is assumed the predicted values are probabilities on the interval [0, 1] that indicate the probability that the case belongs to class 1. Cross-entropy is calculated as

  $$\sum targ \cdot log(pred) + (1 - targ) \cdot log(pred)$$

  where *targ* is the target class (0 or 1) and *pred* is the predicted probability that the case is in class 1. To make cross-entropy independent of data set size, the mean cross-entropy is used. That is, the sum of the cross-entropy for each case divided by the total number of cases.

- *SLQ score* is a domain-specific metric devised by researchers at the Stanford Linear Accelerator (SLAC) to measure the performance of predictions that are made for certain kinds of particle physics problems. SLQ is used for models that make continuous predictions on the interval [0, 1]. This interval is divided into a series of bins. For this problem, 100 equally sized bins are used. The 100 bins are: 0.00 to 0.01, 0.01 to 0.02, ..., 0.98 to 0.99, and 0.99 to 1.00. The predictions are placed into the bins based on the predicted values. For example, if the model predicts a value of 0.025 for a case, that case will be placed in the third bin 0.02 to 0.03. In each bin SLQ keeps track of the number of true positives and true negatives. If bins have high purity, SLQ is maximized. That is, if all bins contain all 0s or all 1s. The latter situation is unlikely, so SLQ computes a score based on how pure the bins are. The score is computed as follows. Suppose a bin has 400 positives and 100 negatives. The error of this bin if positives are predicted is $\frac{100}{(400+100)} = 0.2$. The contribution of this bin to the total SLQ score is

  $$(1 - 2 \cdot error)^2 \cdot \frac{\text{total number of points in this bin}}{\text{total number of points in the data set}}.$$

Division by the size of the total data set normalizes the score so that it is more independent of the size of the data set. Multiplying by the total number of points in the bin gives more weight to bins that have more points, and less weight to bins with fewer points. The term

$$(1 - 2 \cdot error)^2$$

is maximized to 1 when all points in that bin are the same class. Finally, the sum over the 100 bins is maximized when the contribution of each bin (weighted by the number of points in each bin) is maximized.

The interactive construction algorithm was utilized in building a GANN model for the quantum physics problem. This methodology is described next.

2.4.2 Methodology

In the first step of the interactive construction algorithm, the process of growing and pruning is started with a single neuron plus a skip layer for each input. The binary inputs ($v30, v35, v36, v43$, and $v52$) are assigned only a skip layer. A Generalized Linear Model is fitted in the second step to give initial estimates of the constant term α and the w_{0j}. The remaining three parameters in each hidden layer are initialized as random values from a normal distribution with mean zero and variance equal to 0.1 in step 3. The full GANN model is estimated in step 4 and the fitted univariate functions overlaid on their partial residuals are examined in step 5. A total of 73 partial residual plots are created in step 5 and each one inspected visually to determine the appropriate bias-variance trade-off. Three of these diagrams are shown in Figures 2.7, 2.8, and 2.9.

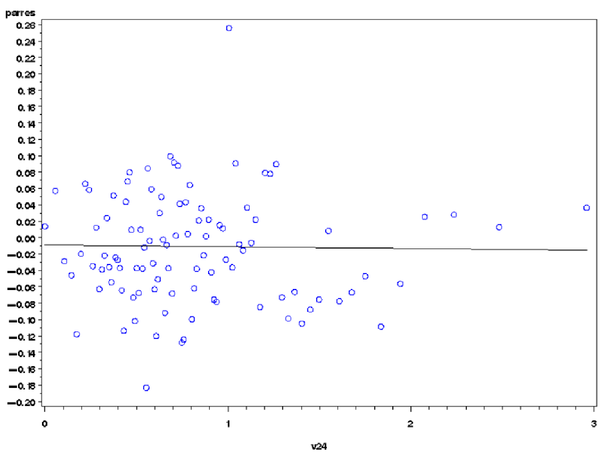


Figure 2.7: Partial residual plot for *v24*

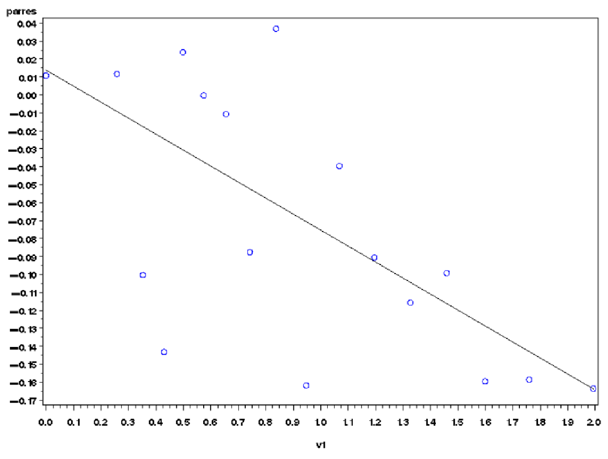


Figure 2.8: Partial residual plot for *v1*

To guide the modeler in deciding on the appropriate complexity of the univariate functions, the functions are shown as a fitted spline overlaid on the partial residuals. When the spline forms a horizontal or near horizontal line, it indicates that the univariate function is constant for the whole range of input values and consequently no contribution is made towards describing variation in the response. In this case the input can be removed from the model (e.g. Figure 2.7). A spline that forms a line with a significant positive or negative slope indicates a linear relationship between the input and the response. For these cases the input can be set to only a skip layer (e.g. Figure 2.8). A nonlinear relationship with the response is identified when the spline forms a curve and modeled by one or more neurons in the hidden layer of the input (e.g. Figure 2.9). Care must be taken not to add too many neurons, for this would create an univariate function with high variance and high bias.

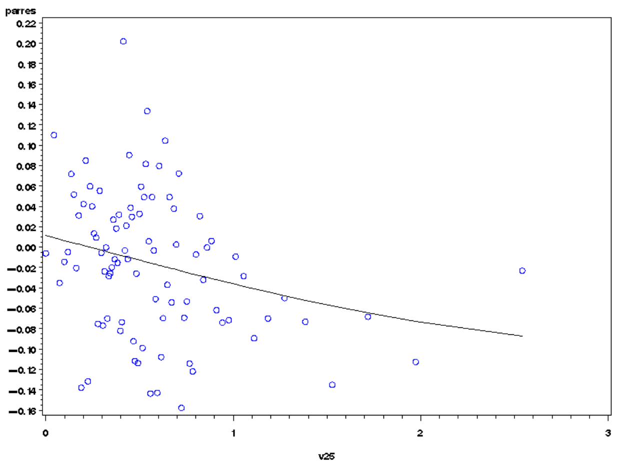


Figure 2.9: Partial residual plot for *v25*

In step 6 several architectural changes are made to the 73 inputs after examining their partial residual plots: 46 inputs are eliminated, 23 inputs are pruned back to a linear fit (a skip layer), 5 inputs are kept the same (a skip layer and one neuron), a single neuron is added to 3 inputs (resulting in a skip layer and two neurons), and two neurons are added to 1 input (creating a skip layer and three neurons). These changes are made in several iterations of steps 4, 5, and 6 of the interactive construction algorithm. Table 2.1 summarizes the final GANN model that was built. From this table it can be seen that inputs $v4, v6, v7, v8, v12, v39, v63, v69,$ and $v75$ were identified as having nonlinear relationships with the response.

Inputs removed	$v2, v3, v9, v10, v11, v16, v17, v18, v19, v20, v23, v24, v25, v26, v27, v28, v31, v32, v33, v34, v38, v47, v48, v49, v50, v51, v53, v54, v56, v57, v58, v59, v60, v61, v62, v64, v65, v67, v68, v70, v72, v73, v74, v76, v77, v78$
Linear fit (only skip layer)	$v1, v5, v13, v14, v15, v21, v22, v29, v30, v35, v36, v37, v40, v41, v42, v43, v44, v45, v46, v52, v55, v66, v71$
Skip layer with one neuron	$v4, v6, v7, v39, v75$
Skip layer with two neurons	$v8, v12, v69$
Skip layer with three neurons	$v63$

Table 2.1: Final GANN architecture

2.4.3 Results

The final GANN model (Table 2.1) was entered into KDD Cup 2004 and the same set of predictions were submitted for the different tasks. In Table 2.2 the results of the four metrics are shown for the top 10 teams.

Group	Accuracy		Area under ROC		Cross-entropy		SLQ score		Overall Rank
21	2	0.73187	1	0.83054	1	0.70949	1	0.33280	1.333
425	1	0.73255	2	0.82754	2	0.72456	2	0.32648	1.667
566	3	0.72775	3	0.82250	4	0.73001	3	0.31749	3.333
532	5	0.72744	6	0.81791	3	0.72798	6	0.30982	4.667
112	4	0.72745	4	0.82109	8	0.74371	4	0.31447	5.333
421	6	0.72522	5	0.81906	6	0.73634	5	0.31142	5.667
349	**7**	**0.72424**	**8**	**0.81412**	**7**	**0.74137**	**9**	**0.30217**	**7.333**
24	11	0.72060	7	0.81572	5	0.73583	7	0.30591	7.667
191	9	0.72304	9	0.81252	10	0.74632	8	0.30410	9.333
14	8	0.72332	10	0.81208	14	0.75432	*empty*	*empty*	10.667

Table 2.2: KDD Cup 2004 results for top 10 teams

Each team was assigned a group number that is given in the first column. For each measurement the rank of the metric and the value is given. No submission for a specific metric is indicated by *empty*. In the last column the overall rank of the team is shown. The organizers decided to use only the average of the accuracy rank, area under the ROC curve rank, and the cross-entropy rank to determine the overall rank since the given software that computed the SLQ score was faulty. Thus, the overall winner is the participant with the best average rank across all three metrics. The final GANN model developed in the previous section (represented by group 349) ranked seventh overall.

2.4.4 Conclusions

The GANN model proved its predictive power, as can be seen from the results given in Table 2.2 (accuracy is less than 0.7% worse than that of the winning team). As no interactions between variables were included in the model, the obtained results are exceptionally accurate considering that the winning submission used variable interactions to a large degree. This is surprising, as including variable interactions in a model are usually considered essential in obtaining a good model.

Figures 2.10, 2.11, 2.12, and 2.13 show four partial residual plots of the final GANN model with univariate functions of increasing complexity: a linear fit in Figure 2.10, a skip layer with one neuron in Figure 2.11, a skip layer with two neurons in Figure 2.12, and a skip layer with three neurons in Figure 2.13[4].

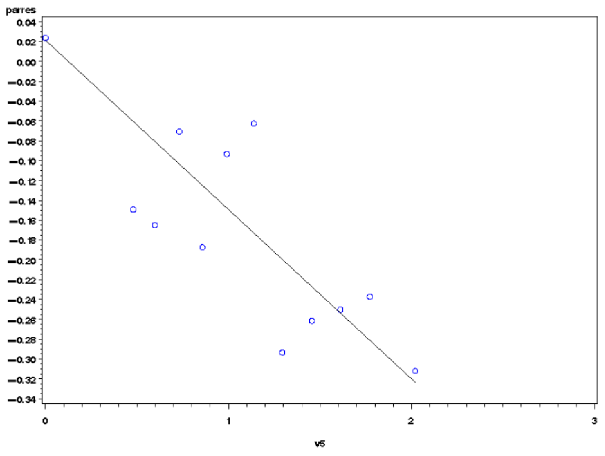


Figure 2.10: Partial residual plot for *v5*

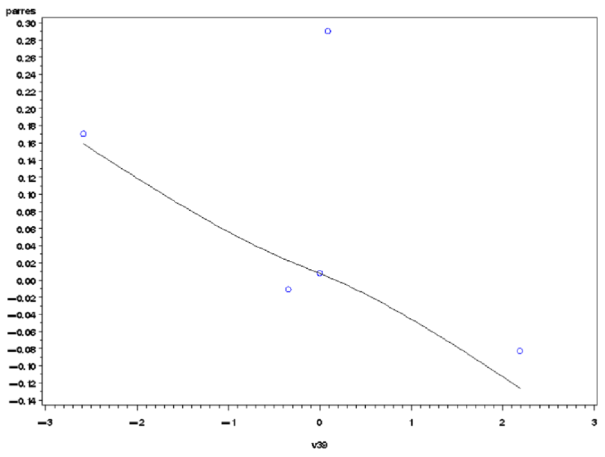


Figure 2.11: Partial residual plot for *v39*

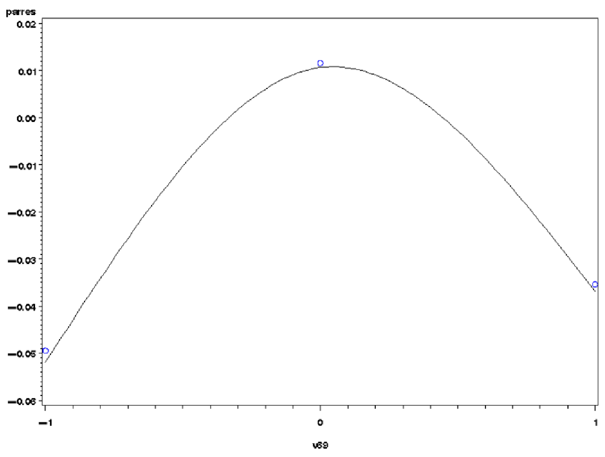


Figure 2.12: Partial residual plot for *v69*

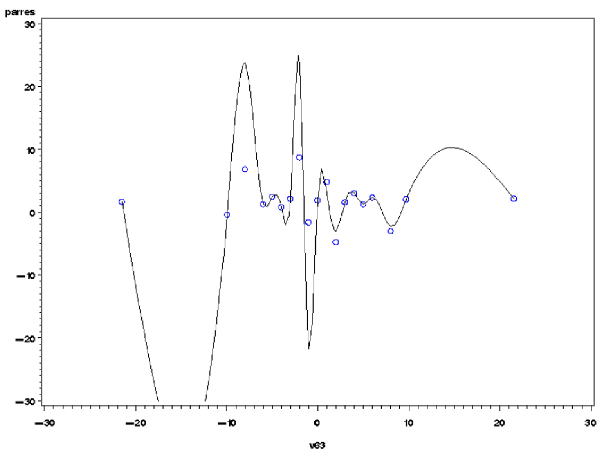


Figure 2.13: Partial residual plot for *v63*

The principle of parsimony was followed when the complexity of the GANN model had to be decided on[5]. In the context of model selection, the GANN model should not be made unnecessarily complex. From Table 2.1 it can be seen that many inputs were removed in the final GANN model. Only nine inputs have nonlinear relationships with the target and most of the inputs have linear relationships with the target. Consequently, the final GANN model can be called a near-linear model. The choice of a simple model may also alleviate the problem of over-fitting.

[4]Note that the data measurement scale of variable $v63$ was mistakenly set to nominal in the implementation which caused the fitted spline in Figure 2.13 to appear more complex than the fitted univariate function f_{63}. This phenomenon is caused by the implementation of a neural network in the SAS® programming language. To correct the fitted spline, the data measurement scale of $v63$ should be set to interval.

[5]This principle is also called Ockham's Razor and is attributed to the 14th-century English logician William of Ockham. Originally it stated that "Multiples should not be put forward without necessity."

On the downside, the interactive construction methodology uses human judgment to interpret partial residual plots which can be subjective and consequently result in suboptimal models. Also, for a large number of inputs this is a time consuming process which can cause the data analyst to experience dismay.

In the next section, financial and economic variables are used to predict excess stock returns with an interactively constructed near-linear GANN model.

2.5 Stock returns example

Prediction of the future has fascinated people since recorded history. Lately, accurately predicting stock returns have captivated the attention of both investors and analysts (Eakins & Stansell, 2003); (Jagric, 2003); (Olson & Mossman, 2003); (Qi, 1999); (Qi, 1996); (Racine, 2001); (Kaul, 1996); (Kanas & Yannopoulos, 2001). In the last decade and a half the characterization of stock return predictability (Avramov, 2002) is arguably the most hotly debated issue of empirical asset pricing. Its importance for understanding the nature of potential market inefficiencies and time-varying risk premia is generally agreed upon. As a result, the last two decades of financial research have seen a large number of articles documenting the ability of various variables to explain movements in conditional expected returns. Although a few people with aggressively negative attitudes exist, usually employing some type of data snooping argument, the literature is generally in favor of time-varying expected returns (Cremers, 2002).

As more off-the-shelf packages become available which support the generation of nonlinear models, the issue of utilizing nonlinear models for stock return prediction seems to be especially relevant. The current norm seems to suggest that nonlinear models is the tool to use since linear models have fallen out of favor. Furthermore, current wisdom seems to suggest that nonlinear models must necessarily be better than linear models, in all cases and for all applications. Qi (1999) recently concluded that a switching portfolio based on forecasts from a neural network generates considerably higher wealth than forecasts based on linear regression. Racine (2001) reviewed this work and could not replicate the stated results. He came to the conclusion that the superiority of a switching portfolio guided by neural networks over a switching portfolio guided by simple linear regression may be premature and then warns forecasters to proceed with caution. De Waal & Du Toit (2004) analyzed the benchmark data set with three goals in mind:

1. Replicate and verify some of the results obtained by (Qi, 1999) and (Racine, 2001).

2. Gain insight into the linear model versus nonlinear model issue for predicting stock returns by exploiting a GANN model.

3. Create a well-motivated, clear, and repeatable strategy that minimizes the use of hindsight to

obtain generated wealth superior to that of the linear model. This must be a simple strategy, so that a reasonably skilled investor can implement it, without the need to consult a highly specialized data analyst trained in neural network modeling.

The S&P 500 data set contains nine variables over which a search for a prediction model may be conducted. The variables are: *YSP*, dividend yield lagged one period, *EP*, earning-price ratio lagged one period, *PI(-2)*, year-on-year rate of inflation lagged two periods, *I1(-1)*, one-month T-bill rate lagged one period, *I1(-2)*, one-month T-bill rate lagged two periods, *I12(-1)*, one-year T-bill rate lagged one period, *I12(-2)*, one-year T-bill rate lagged two periods, *DIP*, year-on-year change in industrial output lagged two periods, and *DM*, year-on-year growth rate in narrow money stock (monetary growth rate) lagged two periods. The target variable is monthly excess returns (*EXRET*) on the S&P 500 over a period of 33 years (January 1960 to December 1992). A switching portfolio is created where the investor may either invest in stocks or in bonds. This investment decision is made at the end of each month, from month 73 (January 1966) to month 467 (November 1992). The initial value of the portfolio at the beginning of the forecasting period (month 73) is set to \$100. The goal is to maximize the accumulated wealth generated over the 33-year period. More details on this problem can be found in Pesaran & Timmerman (1995).

The linear model is used as the reference model as it should be independent of the system on which it is computed. To provide more insight into the importance of individual regressors over the forecasting period, weight estimates for the nine input variables and the output bias are calculated and plotted for each linear model from month 73 to month 468. This diagram is called the weight estimate plot, and inspection of the diagram shows that the linear model changes over time. This observation is confirmed by Pesaran & Timmerman (1995). In contrast to linear models, nonlinear models are not independent of the system on which it is computed. Different seeds, different optimization techniques and different stopping criteria may give vastly different results.

One of the aims is to improve on the linear model. Hopefully, the generated final wealth will also improve. A GANN model with lower MAPE (mean absolute percentage error), lower MAE (mean absolute error), higher CORR (correlation coefficient), lower RMSE (root mean squared error), and higher SIGN (fraction of correctly predicted signs) than the linear model is considered better than the linear model (De Waal & Du Toit, 2004). Consult Armstrong & Collopy (1992) for a more detailed account on error measures.

2.5.1 Methodology

The following guidelines to construct the GANN model interactively were followed and turned out to work reasonably well.

1. To restrict the number of possible models that need examination in steps 4, 5, and 6 of

the interactive construction algorithm, reduce the complexity of the univariate functions by limiting the number of neurons h for each variable to a small number. A maximum of two neurons per input was utilized to capture a possible nonlinear relationship between the input and the target.

2. The partial residual plots should be inspected in conjunction with information gained from examining the weight estimates (regression coefficients) generated by the linear model. When a variable is earmarked for deletion, this especially is the case. Furthermore, a variable should only be deleted from the model if evidence from the partial residual plot and weight estimate plot supports the removal.

3. After the linear model is constructed in step 2 of the interactive construction methodology, fit the most complex model possible by keeping to the maximum number of neurons decided on in guideline 1. Prune the hidden layers of variables that clearly show linear or constant effects. During the final iterations of the pruning and growing process, nonlinear effects can be optimized.

With the proposed approach, models that do not deviate too much from the linear model are studied. This implies that variables that are not important may be deleted from the model, univariate functions may be replaced with more complex univariate functions where the nonlinear trend appears to be underfitted and univariate functions that exhibit linear effects may be simplified.

The nine plots with univariate functions overlaid on the partial residuals are given in Figures 2.14 to 2.22 for month 467. Also, the typical reasoning for deciding on the complexity of the univariate functions in steps 5 and 6 of the interactive construction methodology is presented.

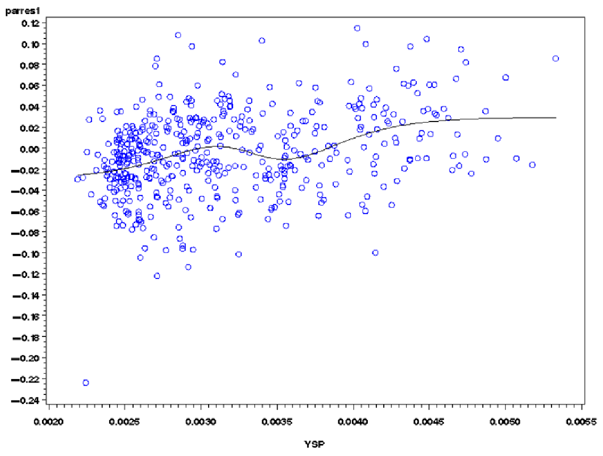


Figure 2.14: Partial residual plot for *YSP*

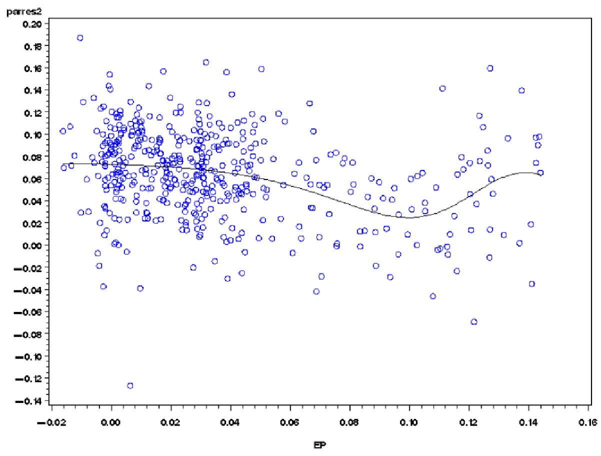


Figure 2.15: Partial residual plot for *EP*

The fitted univariate function in Figure 2.14 exhibits a reasonable fit and this variable will be further investigated after simplification of the model. This univariate function is kept unchanged. In Figure 2.15 the fitted spline overlaid on the partial residuals exhibits a reasonable fit and this variable will be examined again after simplification of the model. The corresponding univariate function is left unchanged.

A horizontal line overlaid on the partial residuals (Figure 2.16) also exhibits a reasonable fit. The weight estimates for *PI(-2)* are small, negative and nearly constant for all months, so it is removed from the model. In Figure 2.17 a line would also give a reasonable fit and the contribution of this variable to the model is simplified to a linear function.

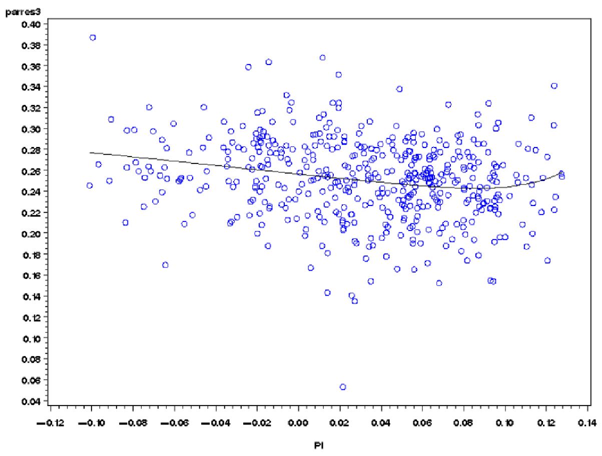


Figure 2.16: Partial residual plot for *PI(-2)*

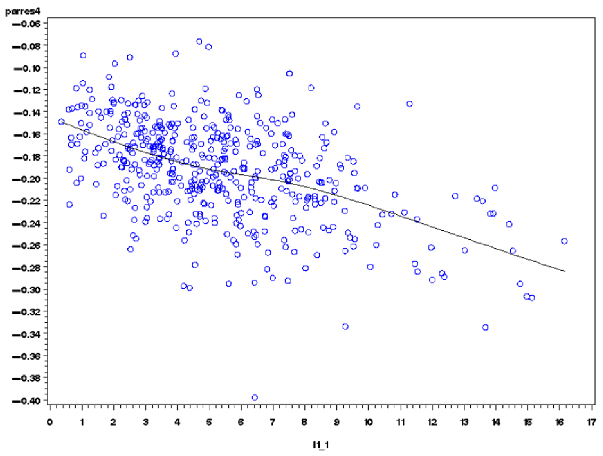


Figure 2.17: Partial residual plot for *I1(-1)*

A horizontal line would give a good fit in Figure 2.18. The weights for *I1(-1)* and *I1(-2)* are strongly negatively correlated with a correlation coefficient of -0.898. Consequently, *I1(-1)* or *I1(-2)* should be deleted as it is good modeling practice not to include such highly correlated variables in a model. *I1(-2)* was deleted from the model. The added complexity of the S-curve is not really needed in Figure 2.19 and the univariate function for this variable is simplified to a linear function.

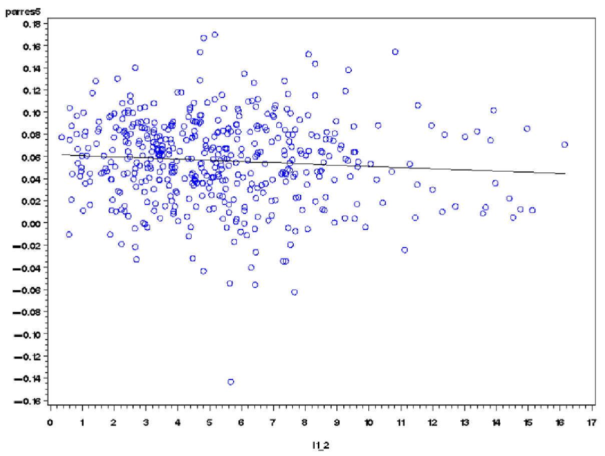


Figure 2.18: Partial residual plot for *I1(-2)*

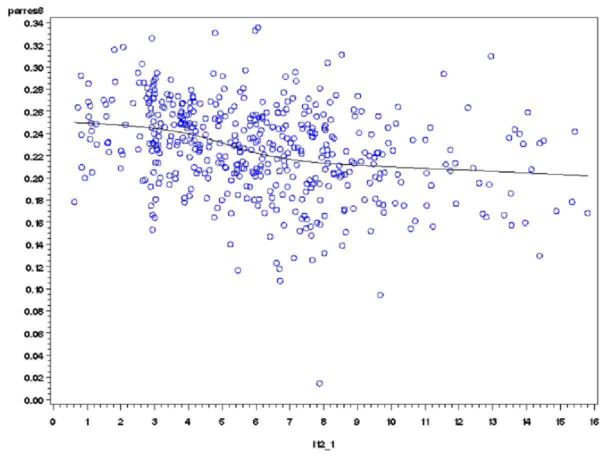


Figure 2.19: Partial residual plot for *I12(-1)*

An S-curve is not needed for a good fit (Figure 2.20) and a linear function is also assigned to the univariate function for this variable. In Figure 2.21 the S-curve exhibits a nonlinear effect and gives a good fit. The univariate function is therefore kept unchanged.

Finally, the fit in Figure 2.22 is reasonable, although a straight line would also give a good fit. The variable will be further investigated after simplification of the model.

The GANN model is updated by deleting variables *PI(-2)* and *I1(-2)*. Then the new weight estimates for the variables are computed and the partial residual plots are updated and inspected.

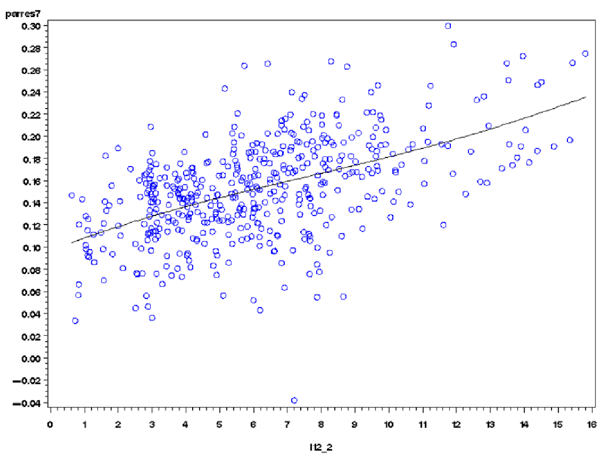


Figure 2.20: Partial residual plot for *I12(-2)*

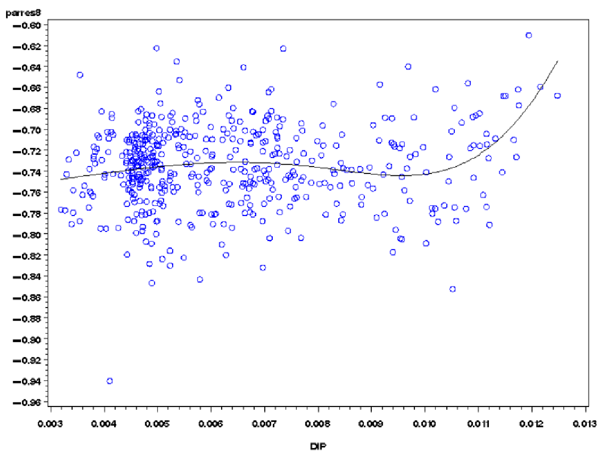


Figure 2.21: Partial residual plot for *DIP*

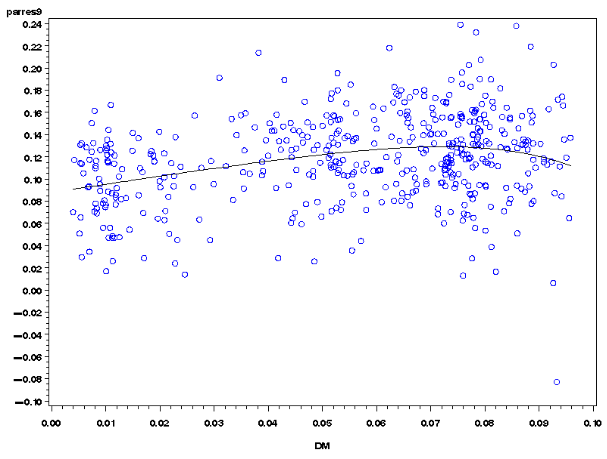


Figure 2.22: Partial residual plot for *DM*

After several architectural changes were made in step 6 of the interactive construction algorithm based on the partial residual plots and weight estimates, the final GANN model contains the following variables: *YSP, EP, I1(-1), I12(-1), I12(-2),* nlin(*DIP*) and *DM*, where nlin(*DIP*) indicates that the variable *DIP* (with a lag of two months) is a complex univariate function (with two neurons) capable of capturing nonlinear effects. The final partial residual plots are given in Figures 2.23 to 2.29.

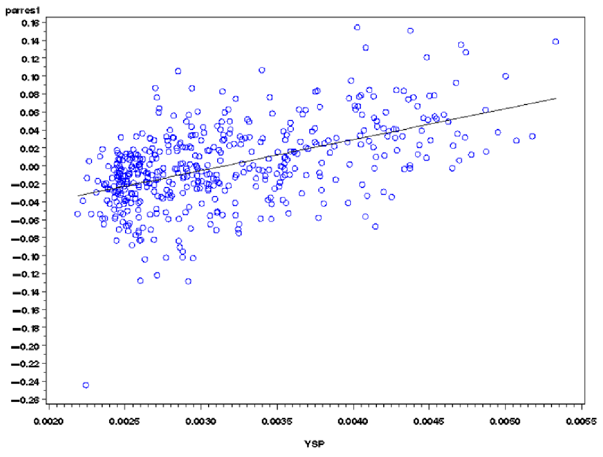


Figure 2.23: Partial residual plot for *YSP*

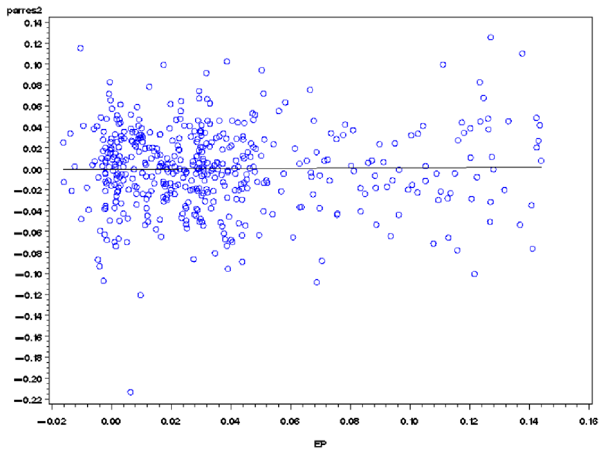


Figure 2.24: Partial residual plot for *EP*

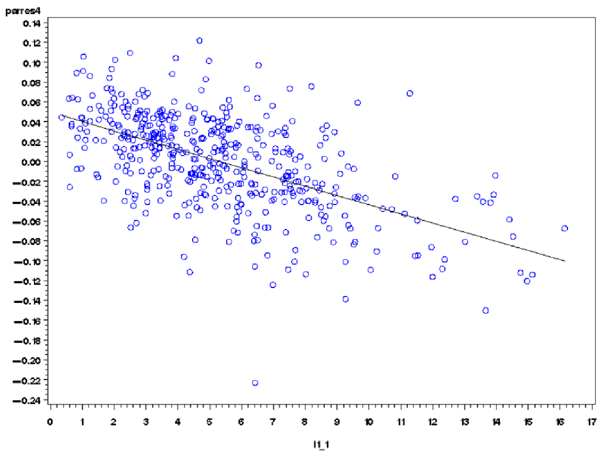


Figure 2.25: Partial residual plot for *I1(-1)*

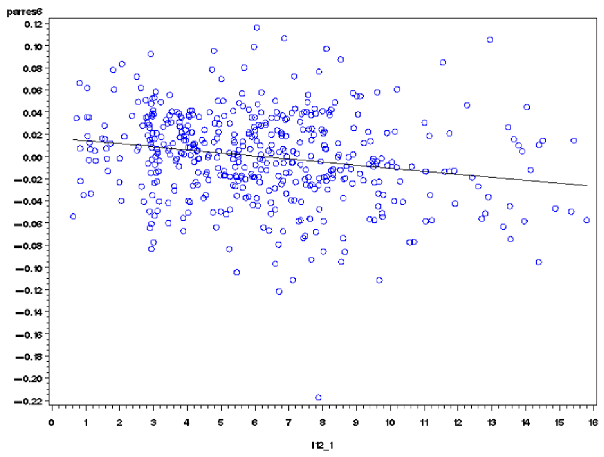


Figure 2.26: Partial residual plot for *I12(-1)*

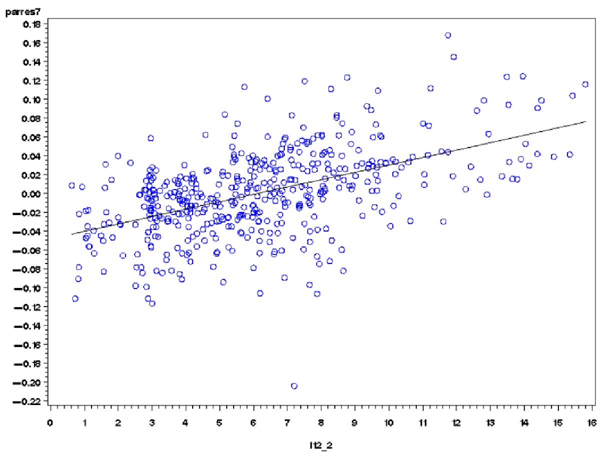


Figure 2.27: Partial residual plot for *I12(-2)*

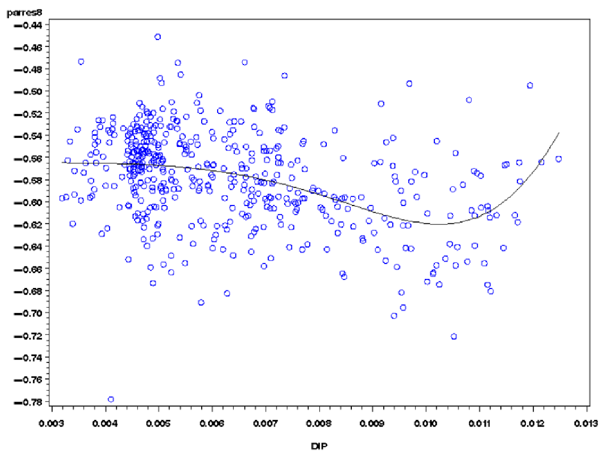


Figure 2.28: Partial residual plot for *DIP*

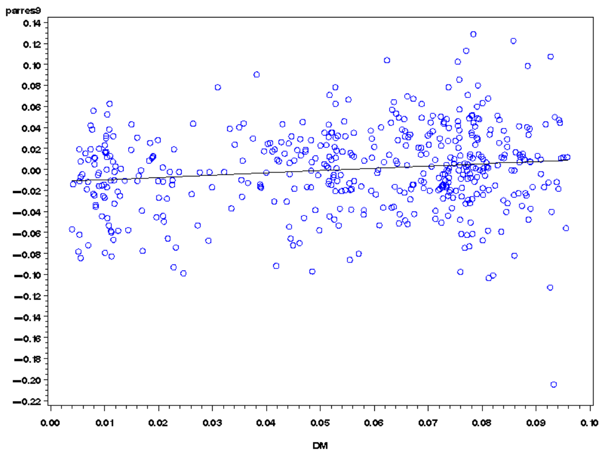


Figure 2.29: Partial residual plot for *DM*

Note again the near-linear model that was created (six variables with linear contributions and one variable with a nonlinear contribution).

2.5.2 Results

In Table 2.3 the in-sample model-fit measures are given for the linear model and the final GANN model. Note the slight improvements in RMSE, MAE, CORR, and SIGN. The biggest improvement (a 9% reduction) is in MAPE. The proof of the strategy is in the final wealth generated by this model. It is $9,915.38 and a improvement of 33% over that generated by the linear model.

The out-of-sample model-fit statistics for zero transaction costs compared to the linear model are given in Table 2.4.

Architecture	RMSE	MAE	MAPE	CORR	SIGN
Linear model	0.0360	0.0284	1.8622	0.3915	0.6546
GANN model	0.0355	0.0279	1.6772	0.4140	0.6696

Table 2.3: In-sample model-fit statistics

Architecture	Mean return (%)	Std. of return	Sharpe ratio	Final wealth ($)
Linear model	13.66	10.08	0.77	7,458.41
GANN model	14.49	9.63	0.89	9,915.38

Table 2.4: Measures of Risk and Returns

From all the models in Qi (1999) and Racine (2001), the model developed in the previous section is the only model with a risk measure (standard deviation of return) smaller (better) than that of the linear model. This measure provides confirmation that the developed strategy is superior to previously developed strategies resulting in a model with lower associated risk (which is reflected in the final wealth generated).

2.5.3 Conclusions

The interactive construction algorithm focused on developing a simple model with good in-sample fit statistics dominating the in-sample fit statistics of the linear model. This resulted in an increase in accumulated wealth. The strategy suggested by the interactive construction algorithm is simple enough for a reasonably skilled investor to implement.

On the other hand, a large number of partial residual plots must be inspected by the investor. For each month, at least nine plots must be inspected resulting in a minimum of $9 \times 396 = 3,564$ plots for the 33-year period. This process can be very time consuming and may discourage the investor to utilize the proposed methodology effectively.

2.6 Conclusions

Currently, nonlinear modeling procedures (such as neural networks) are becoming more available in the form of statistical and data mining packages. However, linear and near-linear models (e.g. GANNs) should not be discarded as useless and ineffective for the solving of complex multi-dimensional problems, as these models are in general easier to compute and interpret than nonlinear models. Nonlinear models come with their own set of problems, such as the curse of dimensionality, over parameterization and training difficulty that may not be straightforward to solve.

Using the GANN architecture helps to alleviate the black box perception of neural networks with respect to interpretation as the effect of each input variable on the fitted model can be interpreted using graphical methods. GANNs may outperform multilayer perceptrons with noisy data since training is simplified. The individual nonlinear effects of the inputs may be easier to learn in this constrained architecture than in the multilayer perceptron architecture.

The GANN model provides the data analyst with adequate predictive power as can be seen from the results given in Tables 2.2, 2.3, and 2.4. This favorable property motivates further investigation into the use of these type of models.

When GANNs are constructed interactively, human judgment is required to interpret the partial residual plots. For a large number of variables this can become a daunting and time consuming task. Also, human judgment is subjective which may result in the creation of models that are suboptimal. In the next chapter, a way forward is discussed. An objective approach is obtained by incorporating a formal measure of fit into the process. This leads to an automated method based on the search for models using model selection criteria. With this new approach, partial residual plots are not used primarily for model building, but as a tool to provide insight into the models constructed. Finally, no human interaction is needed while building the GANN models. This allows the modeler to spend more time on interpretation of the results.

"All science is concerned with the relationship of cause and effect. Each scientific discovery increases man's ability to predict the consequences of his actions and thus his ability to control future events."

Lawrence J. Peters

3 Automated Construction of Generalized Additive Neural Networks

Neural networks' powerful pattern recognition and flexible nonlinear modeling capabilities are the main reasons for their popularity. In contrast to various model-based prediction methods, neural networks are data driven without any restrictive assumptions concerning the functional relationships between the target variable and the input variables. This unique characteristic of neural networks is highly desirable in many situations where data are generally abundant but the underlying data generating mechanism is often unknown or untestable.

Although neural networks have been successfully used for numerous prediction applications, several issues in neural network model building still have to be solved. One of the most critical issues is how to select an appropriate network architecture for a prediction task at hand. Model selection is a nontrivial issue in traditional linear prediction applications and is a particularly tricky one for nonlinear models such as neural networks. Neural networks often suffer overfitting problems due to a typically large number of parameters to be estimated. That is, they fit in-sample (or training) data very well but predict poorly out-of-sample (or on the test set).

To alleviate the overfitting effect, two broad types of model selection approaches are often adopted in the neural network literature (Qi & Zhang, 2001). The first is the cross-validation-based approach (Stone, 1974) that divides the available data into three parts: training, validation, and test sets. The training and validation sets are used for neural network model building while the last part is for out-of-sample evaluation. The training set is used for parameter estimation in a number of alternative neural network specifications (e.g., networks with different numbers of inputs, and different numbers of middle layer units in a three-layer MLP). The trained network is evaluated using the validation set. The network model that performs best on the validation set is selected as the final prediction model. The validity and usefulness of the model is then checked using the test set. This out-of-sample model selection and testing procedure is generally quite effective in conquering the overfitting tendency of neural network models. However, it has several limitations. First, splitting the data may increase the variability of the estimates (Faraway, 1992). In addition, no general guidelines exist on how to split the data into three parts. Finally, data splitting requires a fairly large sample size. For applications with a small data set, this procedure reduces the already small number of training patterns and weakens the reliability of the model.

The second method uses in-sample model selection criteria. This approach relies solely on a certain in-sample criterion as a convenient computational shortcut, hoping that the in-sample criterion can assist in choosing the best prediction model between the available alternatives. In the next chapter the two most popular model selection criteria for linear and nonlinear model selection is discussed. These two criteria penalize large models that often tend to overfit and can be applied to Generalized Additive Neural Networks to guide the model selection process. For certain applications where the model structure and selection of variables are uncertain, the model selection criterion can be utilized to perform model averaging to obtain a more stable result. This technique is explained further in Chapter 4.

In the next section, an automated approach to the construction of GANNs is introduced. This new method is objective and relies only on a model selection criterion for model selection. No human interaction is needed for choosing the single best model. The operation of the automated construction algorithm is illustrated in Section 3.2 with the Kyphosis data set (Bell et al., 1989). In Section 3.3, the best GANN model found with the interactive construction methodology is compared with the best model found by the automated method. For this comparison, the Boston Housing data set (Harrison & Rubinfeld, 1978) is utilized. The Ozone data set (Breiman & Friedman, 1985) has been analyzed by several others in the nonparametric regression literature. The best model found by the automated approach using this meteorological data set is compared with best models found by other techniques in Section 3.4. Finally, in Section 3.5, the implementation of the automated construction algorithm in the powerful SAS® language is described. This implementation has a simple user interface and presents the modeler with a number of result windows. Insight can be obtained into the nature of the developed GANN models, by considering partial residual plots and fit statistics created by the system. Examples of output from the system are presented and a complexity analysis of the implementation is performed, given the structure of the search tree created by the automated construction algorithm.

3.1 Automated construction methodology

In order to describe the automated construction methodology of GANNs, a number of definitions are needed. They are subsequently presented and illustrated with the example GANN of Figure 3.1.

Definition 3.1 (Neural Network) *A neural network is a system composed of many simple processing elements operating in parallel of which the function is determined by network structure, connection strengths, and the processing performed at computing elements or nodes (DARPA, 1988).*

Definition 3.2 (GANN sub-architecture) *The neural network structure for a specific input.*

Each input of the GANN model has a certain sub-architecture which defines the functional form of the univariate function. The GANN sub-architectures for the three inputs in Figure 3.1 are the following. Input x_1 has an MLP (see Section 2.3) with no skip layer and one neuron in the hidden layer. Input x_2 has an MLP with a skip layer and three neurons in the hidden layer, and input x_3 has an MLP with a skip layer and two neurons in the hidden layer. To capture a more complex nonlinear relationship between an input and the target, nodes must be added to the hidden layer of the input.

Definition 3.3 (GANN sub-architecture identifier) *The symbol used to denote the sub-architecture for a specific input.*

In Table 3.1, ten standard GANN sub-architecture identifiers[1] are defined which proved to be adequate for this study on the automated construction of GANNs. These ten symbols are used throughout the rest of the thesis to describe GANN models. Sub-architecture identifier 0 indicates that the input is removed from the GANN model. A linear relationship between the input and the target is modeled by a 1 sub-architecture identifier and nonlinear relationships are captured by sub-architecture identifiers 2 to 9. By using these identifiers, GANN models can be described in a compact way using a standardized notation. The GANN sub-architecture identifier is also called the GANN sub-architecture in certain contexts.

In Figure 3.1 the GANN model has a GANN sub-architecture of 6 for input x_1, a GANN sub-architecture of 4 for input x_2, and a GANN sub-architecture of 3 for input x_3.

Definition 3.4 (GANN architecture identifier) *The list of GANN sub-architecture identifiers, $[identifier_1, identifier_2, \ldots, identifier_k]$, used to denote the architecture for a specific GANN*

[1]Xiang (2001) uses a similar notation to describe Generalized Additive Models.

model having k inputs, $x_1, x_2, \ldots, x_k$, where $identifier_i$ corresponds to the sub-architecture of input x_i, $i = 1, 2, \ldots, k$.

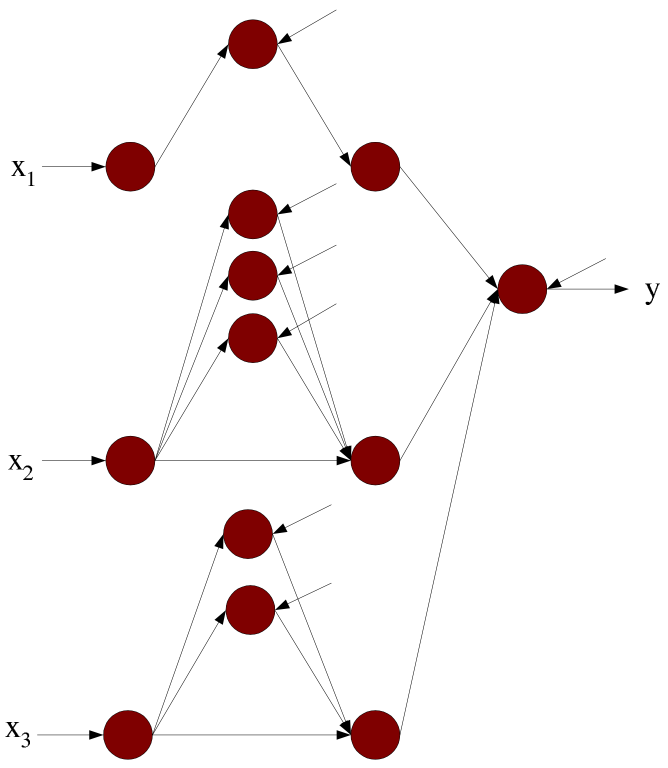


Figure 3.1: Generalized Additive Neural Network

GANN sub-architecture description	GANN sub-architecture identifier
Input removed from the model	0
MLP with a skip layer and no hidden nodes	1
MLP with a skip layer and one hidden node	2
MLP with a skip layer and two hidden nodes	3
MLP with a skip layer and three hidden nodes	4
MLP with a skip layer and four hidden nodes	5
MLP with no skip layer and one hidden node	6
MLP with no skip layer and two hidden nodes	7
MLP with no skip layer and three hidden nodes	8
MLP with no skip layer and four hidden nodes	9

Table 3.1: GANN sub-architecture identifiers

The GANN architecture identifier is also called the GANN architecture or GANN model in certain contexts. In Figure 3.1 the GANN architecture is [6,4,3].

Definition 3.5 (GANN architecture) *The combination of all GANN sub-architectures to form the complete GANN model.*

The GANN architecture in Figure 3.1 is the combination of the three sub-architectures of inputs x_1, x_2, and x_3.

Definition 3.6 (GANN sub-architecture identifier function) *A function, $sub(x_i)$, that returns the GANN sub-architecture identifier for a specific GANN model and input, x_i.*

For the example GANN, $sub(x_1) = 6$, $sub(x_2) = 4$, and $sub(x_3) = 3$.

Definition 3.7 (GANN sub-architecture space) *The set of all possible GANN sub-architectures.*

The GANN sub-architecture space defined by Table 3.1 is $\{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$. Occasionally the GANN sub-architecture space is called the GANN search space. In the latter case the GANN search space denotes the set of all possible GANN models determined by the GANN[2] sub-architecture space.

Given the definitions above, automation of the interactive construction process is based on the creation of a search space of possible GANN models and an effective search procedure to find the best model using some model selection criterion. The search space is organized in the form of a tree.

Algorithm 3.1, that utilizes the sub-architectures defined in Table 3.1, automates the interactive construction algorithm (Section 2.3) of Potts (1999). For the automated algorithm to be effective, a criterion has to be defined to order models from "good" to "bad". The insight that a model selection criterion such as SBC or AIC can be used to evaluate the usefulness/accuracy of a model makes automation of the interactive construction possible. When a validation data set is present, models are evaluated on the validation set which allow the algorithm to perform cross-validation. Furthermore, feature selection (Guyon & Elisseeff, 2003); (Blum & Langey, 1997) is automatically performed by the algorithm.

The automated construction algorithm (Du Toit & De Waal, 2003) utilizes a best-first search strategy (Rich & Knight, 1991). At each step of the best-first search process, the most promising node of those generated so far is selected (step 5). This is done by considering the model selection criterion value of each generated node. The chosen node is then expanded by applying steps 6 and 7 to generate its successors. All the new nodes are added to the set of nodes generated so far. Again the most promising node is selected for expansion and the process continues. The order in

[2]Note that when GANN models are the only models under discussion, the word GANN in the terms defined above is sometimes omitted.

which sub-architectures are defined in Table 3.1 influences the paths followed by best-first search. Usually a sub-set of the sub-architectures in Table 3.1 is chosen when a problem is solved to reduce the size of the search space.

1. Construct a GANN with one neuron in the hidden layer and a skip layer for each input in the model. In this step the univariate functions are initialized to
   $f_j(x_{ji}) = w_{0j}x_{ji} + w_{1j}tanh(w_{01j} + w_{11}x_{ji})$.
   This gives 4 parameters for each input. Binary inputs only have a direct connection.
2. Fit a Generalized Linear Model to give initial estimates of the constant term and the w_{0j}.
3. Initialize the remaining 3 parameters in each hidden layer to random values selected from a normal distribution with mean zero and variance equal to 0.1.
4. Fit the full GANN model. Evaluate this model with the model selection criterion, set the flag *expanded* to $false$ (model available for expansion) and denote it as the root of the tree.
5. Search the tree for the best GANN model m, based on the model selection criterion, where the flag *expanded* is $false$. If such a model is found, set the flag *expanded* to $true$ (model expanded). If no model can be found with the flag *expanded* set to $false$, search the tree for the best model, report this model and terminate.
6. For each input x_i of model m (identified in step 5): If $1 \leq$ sub(x_i) ≤ 9, create a node (GANN model) n with the sub-architecture of x_i set to sub(x_i) - 1 and the remaining sub-architectures of m unchanged. Check whether node n has been previously created in the tree. If not, evaluate n with the model selection criterion, add n as a child node to the parent node m, and set the flag *expanded* of node n to $false$.
7. For each input x_i of model m: If $0 \leq$ sub(x_i) ≤ 8, create a node n with the sub-architecture of x_i set to sub(x_i) + 1 and the remaining sub-architectures of m unchanged. Check whether node n has been previously created in the tree. If not, evaluate n with the model selection criterion, add n as a child node to the parent node m, and set the flag *expanded* of node n to $false$.
8. Go to step 5.

Algorithm 3.1: Automated Construction Algorithm

Two lists of nodes are needed to implement the best-first search procedure:

- Open: nodes that have been generated with the model selection criterion calculated, but have not yet been expanded (i.e., had their successors generated). These are nodes with the *expanded* flag set to $false$.

- Closed: nodes that have already been expanded (*expanded* flag set to $true$).

With best-first search, a heuristic function is needed to estimate the merits of each node that is generated. The model selection criterion is used for this purpose and enables the algorithm to search more promising paths first.

The actual operation of the automated algorithm is straightforward. It proceeds in steps. At each step, it picks the most promising of the nodes that have so far been generated but not expanded. It generates the successors of the chosen node, perform a check to determine whether any of the nodes have been generated before, applies the heuristic function to the successors, and adds the successors to the list of open nodes. By doing this check, it can be guaranteed that each node only appears once in the tree, although many nodes may point to a node as a predecessor. For efficiency reasons, a finite number of sub-architectures are chosen before attempting to solve any real problem. This guarantees a finite number of models to be searched.

Paths found by best-first search are likely to be shorter than those found with other methods, because best-first search always moves forward from the node that seems closest to the goal node (Winston, 1992). In this case, the goal node is the one with the lowest model selection criterion value. When the algorithm terminates, all possible models have been constructed, given the finite number of sub-architectures, making it a complete search strategy.

The automated construction of GANNs is illustrated next by an example that utilizes a data set from the medical field.

3.2 Medical example

Bell et al. (1989) studied multiple level thoracic and lumbar laminectomy, a corrective spinal surgery commonly performed on children for tumor and congenital or developmental abnormalities such as tethered cord, syrinx, and diastematomyelia. The incidence of postoperative deformity is unknown. The purpose of the study was to delineate the true incidence and nature of spinal deformities that follows the surgery and to assess the importance of age at the time of surgery, as well as the effect of the location and number of vertebrae levels decompressed. The Kyphosis data set in the study consists of retrospective measurements on 81 patients, one of the largest studies of this procedure to date.

The outcome of interest is the absence (0) or presence (1) of kyphosis, defined to be a forward flexion of the spine of at least 40 degrees from vertical. The predictors are AGE, the age in months at the time of operation, the starting ($STARTVERT$) and ending ($ENDVERT$) range of vertebrae levels involved in the operation, and the number of levels involved (*NUMVERT*). These last predictors are related by *NUMVERT = ENDVERT - STARTVERT + 1*. The goal of the analysis is to identify risk factors for kyphosis, and a natural approach is to model the prevalence of kyphosis as a function of the predictors. The medical investigator felt a priori that *NUMVERT* and *STARTVERT* would be more interpretable, so these are used in the analysis. For *STARTVERT*, the range 1 to 12 corresponds to the thoracic vertebrae, while 13 to 17 are the lumbar vertebrae. This dichotomy was considered an important one.

3.2.1 Methodology

For this example the GANN sub-architecture space is limited to $\{0, 1, 2, 3\}$ and the SBC criterion (defined for a least squares analysis) is used as the model selection criterion[3]:

$$SBC = T\log(MSE) + m\log(T).$$

Recall that T is the training sample size, MSE the mean squared error, and m the number of parameters in the model. The workings of the algorithm on this example is now explained in detail.

For each node created in the search tree, Table 3.2 contains the node number (NODE), SBC criterion value (SBC), GANN architecture identifier (ID), depth of node in the tree (DEPTH), and the node number of the parent node (PARENT)[4].

Steps 1 to 4 of the automated construction algorithm creates a [2,2,2] GANN model where the three input variables, x_1, x_2, and x_3 correspond to *AGE, STARTVERT,* and *NUMVERT*. The model is evaluated by the SBC criterion resulting in a value of 27.055. This model is then set as the root of the search tree (Figure 3.2) and indicated by node number 1. The associated flag *expanded* is set to $false$.

Step 5 identifies the root of the search tree ([2,2,2] model) as the best unexpanded GANN model created up to this point and denotes this node as node m. The *expanded* flag of m is set to $true$ which indicates that the node is (being) expanded.

In step 6, each input of model m is pruned by one sub-architecture level. First, a child node, n, is created with an [1,2,2] architecture. A check is made to determine whether node n has already been placed in the tree. Since there is no node with an architecture of [1,2,2] in the tree, this model is evaluated by the SBC criterion which returns a value of 15.256. The node is added to the tree (node number 2 in Table 3.2) and the associated *expanded* flag is set to $false$.

Next, child node n is constructed with a [2,1,2] architecture. A check is made to determine

[3]A better choice of model selection criterion for this small data set would be the AICc, but the SBC is chosen for illustration purposes.

[4]Note that Table 3.2 is composed of two separate tables to make it more readable.

whether node n exists in the search tree. This node has not been placed in the tree previously, consequently the model selection criterion is calculated which returns a value of 7.240 for the model. Node n is then added to the tree (node number 3) and the *expanded* flag of this node is set to $false$.

Then a child node, n, is built with a [2,2,1] GANN architecture. A check is made to determine if node n exists in the tree. Up to this stage no [2,2,1] GANN model was placed in the tree. A value of 9.311 is obtained when the model selection criterion is applied to the model. The node is added to the tree (node number 4) and the flag, *expanded* of this node is set to $false$.

Step 7 of the automated construction algorithm grows each input of model m ([2,2,2]) identified in step 5 by one sub-architecture level. This results in three new child nodes (number 5, 6, and 7) with architectures of [3,2,2], [2,3,2], and [2,2,3] respectively that are added to the search tree. The *expanded* flags of these nodes are set to $false$.

Step 8 returns to step 5 where the best unexpanded node is identified and set to node m. In this case it is node 3 ([2,1,2] architecture) with an SBC criterion value of 7.240. The process to prune (step 6) and grow (step 7) each input of node m is then repeated. Step 6 creates three new child nodes ([1,1,2], [2,0,2], and [2,1,1]). Step 7 adds two new child nodes to the tree, namely [3,1,2] and [2,1,3]. Note that the child node [2,2,2] is not added to the tree, since a node (the root) with the same architecture already exists.

The automated construction algorithm then continues until the search space is exhausted. In Figure 3.2, 63 GANN models are created[5]. At this stage all the models have been generated and a search for the best model in the tree is conducted. This model is then reported and the algorithm terminates.

3.2.2 Results

For the Kyphosis data set, the best GANN model found by the automated construction algorithm is a [0,1,0] GANN model with a SBC criterion value of -5.654. This model is the 25th node created in the search tree and can be found at a tree-depth of 5. Figure 3.3 shows the partial residual plot for $STARTVERT$ of the best model found. Figures 3.4, 3.5, and 3.6 show partial residual plots for the linear ([1,1,1]) GANN model.

From these plots it is not clear whether inputs AGE and $NUMVERT$ should be removed to obtain the best model. The SBC criterion guides the system in choosing the [0,1,0] model over the linear model. By utilizing a model selection criterion, the system is objective and automatic.

[5]There are four possible sub-architectures and three inputs which define a search space of $4^3 = 64$ GANN models. The constant model ([0,0,0]) is omitted since it is not possible to construct a neural network model in the SAS® programming language without any inputs.

3.2.3 Conclusions

The Kyphosis data set was used in Section 3.2 to illustrate the automated construction algorithm. The GANN model utilized to start the search in step 1 can have a large influence on the structure of the search tree.

It is common practice to start estimation of GAM models with four degrees of freedom smoothers (Berk & Booth, 1995). They showed that partial residuals based on a GAM fit are more reliable than those based on a linear fit. Partial residual plots are central to estimation of GANN models in the interactive construction algorithm. For this reason the algorithm is initialized with a skip layer and one neuron (four degrees of freedom) for each input in the model. The automated construction algorithm on the other hand uses a model selection criterion to construct a near-linear model (see Section 2.4.4). Here, partial residual plots are considered after the algorithm has terminated to gain insight into the relationships between inputs and the target of the best model found. Since partial residual plots are not used during model building, the automated construction algorithm is updated (Algorithm 3.2) to start with a linear model[6], i.e. a [1,1,1] architecture in step 1. Hopefully, a good near-linear model would be found in less time than initializing the search with a [2,2,2] architecture. Figure 3.7 and Table 3.3 contain the search tree after the analysis was repeated on the Kyphosis data set with a linear model ([1,1,1] architecture) at the root of the search tree. The best GANN model identified was the ninth node developed in the tree on a depth of 2. When a near-linear relationship exists between the inputs and the target, the latter result confirms the presumption that a good near-linear model can be found in less time when the search for this model commences at the linear model. When the automated construction algorithm starts with a nonlinear model, more nodes must be developed to arrive at a good near-linear model[7].

The following example taken from Potts (2000) illustrates the interactive construction of a GANN. The best model found interactively by Potts is then compared with the best model found by the automated construction algorithm.

3.3 Housing example

The Boston Housing data set was utilized by Harrison & Rubinfeld (1978) in a study to determine how various factors might affect the housing values for 506 census regions in the Boston Standard Statistical Metropolitan Area in 1970. Census tracts in this area which contained no housing units were excluded.

[6]Step 3 of the automated construction algorithm is not required when search is started from a linear model.

[7]An untested hypothesis is that most data sets contain mostly linear relationships between the input variables and the target with only a small percentage of nonlinear relationships. For the Housing and Stock Return examples, this is the case.

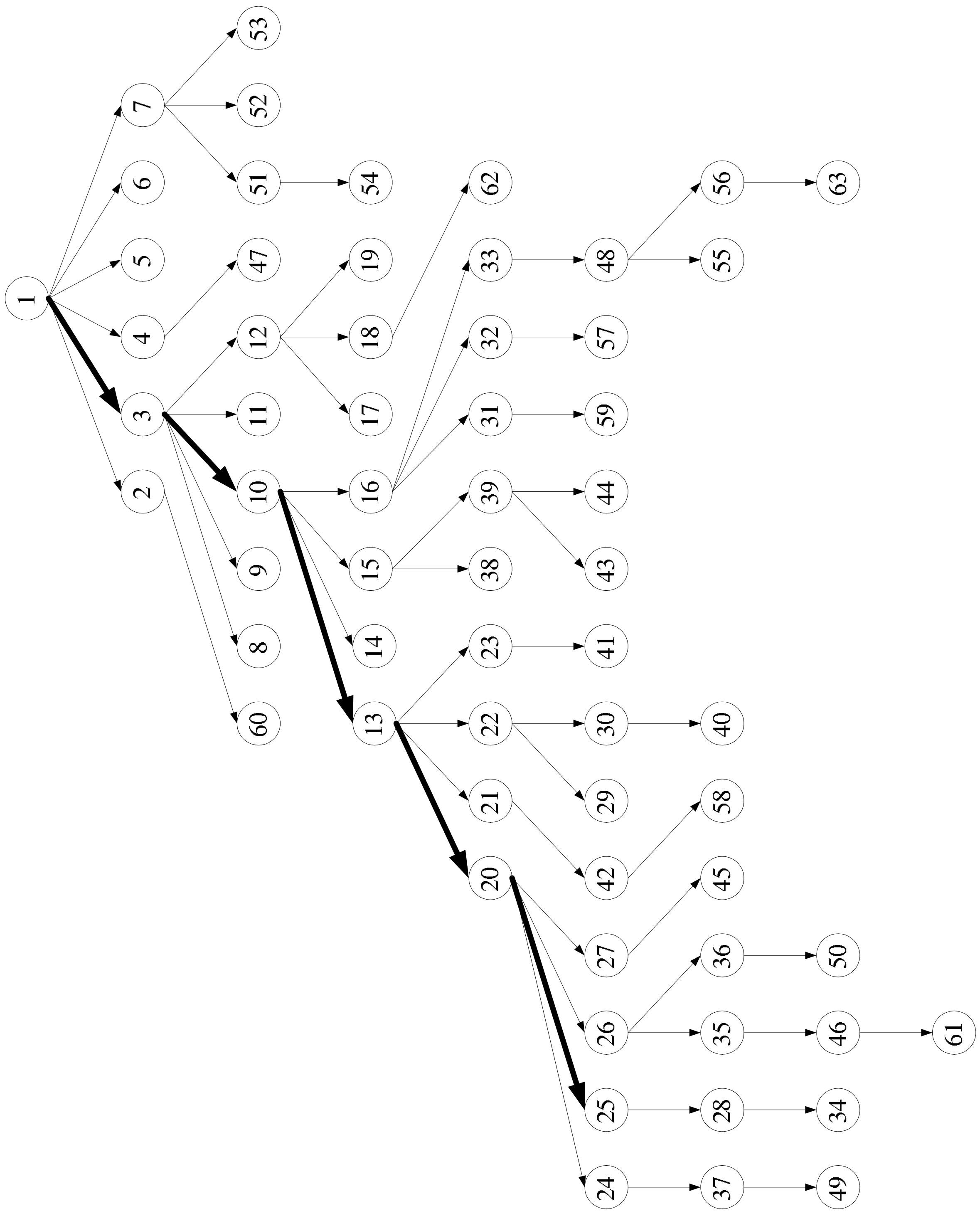


Figure 3.2: Search tree for Kyphosis example with [2,2,2] root node

The data originated from several sources, including the 1970 US Census and the Boston Metropolitan Area Planning Committee. Originally, interest was focussed on the impact of air pollution on the price of owner-occupied homes, but the data includes more than a dozen other potential explanatory variables, and the general question of what factors determine housing values (at least in Boston, 1970) is of interest. Cawley & Talbot (2002), Bakker & Heskes (2003), Belsley, Kuh & Welsch (1980) also analyzed the data.

NODE	SBC	ID	DEPTH	PARENT	NODE	SBC	ID	DEPTH	PARENT
1	27.055	[2,2,2]	0	0	33	10.887	[3,2,1]	4	16
2	15.256	[1,2,2]	1	1	34	13.728	[0,3,0]	7	28
3	7.240	[2,1,2]	1	1	35	7.781	[0,3,1]	6	26
4	9.311	[2,2,1]	1	1	36	12.505	[0,2,2]	6	26
5	23.323	[3,2,2]	1	1	37	11.390	[0,0,2]	6	24
6	18.927	[2,3,2]	1	1	38	17.110	[2,0,0]	4	15
7	13.037	[2,2,3]	1	1	39	5.564	[2,2,0]	4	15
8	8.194	[1,1,2]	2	3	40	16.622	[1,3,0]	6	30
9	28.594	[2,0,2]	2	3	41	16.966	[1,3,1]	5	23
10	-3.992	[2,1,1]	2	3	42	15.041	[1,0,2]	5	21
11	19.796	[3,1,2]	2	3	43	19.271	[3,2,0]	5	39
12	-3.937	[2,1,3]	2	3	44	16.403	[2,3,0]	5	39
13	-2.777	[1,1,1]	3	10	45	16.172	[0,1,3]	6	27
14	4.430	[2,0,1]	3	10	46	23.328	[0,3,2]	7	35
15	1.093	[2,1,0]	3	10	47	12.560	[2,3,1]	2	4
16	-0.539	[3,1,1]	3	10	48	13.259	[3,3,1]	5	33
17	17.563	[1,1,3]	3	12	49	20.647	[0,0,3]	7	37
18	26.729	[2,0,3]	3	12	50	26.990	[0,2,3]	7	36
19	31.532	[3,1,3]	3	12	51	6.391	[1,2,3]	2	7
20	-5.597	[0,1,1]	4	13	52	32.123	[3,2,3]	2	7
21	4.375	[1,0,1]	4	13	53	33.525	[2,3,3]	2	7
22	-2.838	[1,1,0]	4	13	54	41.349	[1,3,3]	3	51
23	3.446	[1,2,1]	4	13	55	29.323	[3,3,0]	6	48
24	0.572	[0,0,1]	5	20	56	30.875	[3,3,2]	6	48
25	**-5.654**	**[0,1,0]**	**5**	**20**	57	30.367	[3,0,0]	5	32
26	0.384	[0,2,1]	5	20	58	21.163	[1,0,3]	6	42
27	6.017	[0,1,2]	5	20	59	30.956	[3,0,2]	5	31
28	-0.069	[0,2,0]	6	25	60	23.761	[1,3,2]	2	2
29	12.135	[1,0,0]	5	22	61	32.478	[0,3,3]	8	46
30	3.148	[1,2,0]	5	22	62	36.457	[3,0,3]	4	18
31	15.079	[3,0,1]	4	16	63	37.735	[3,3,3]	7	56
32	14.371	[3,1,0]	4	16					

Table 3.2: Search tree for Kyphosis example with [2,2,2] root node

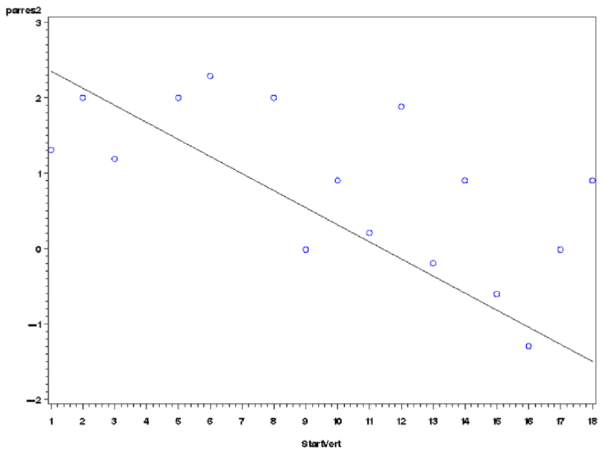


Figure 3.3: Partial residual plot for *STARTVERT*

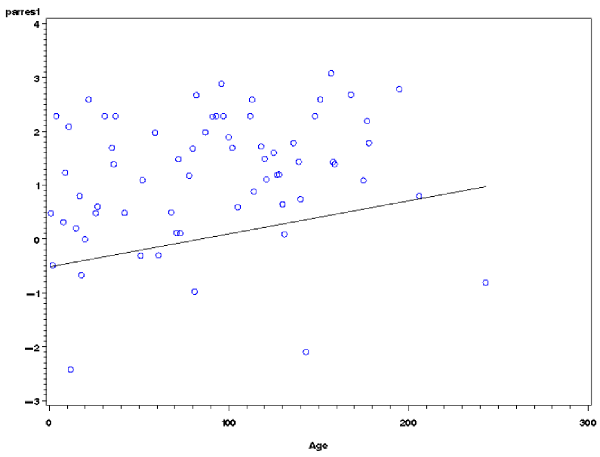


Figure 3.4: Partial residual plot for *AGE*

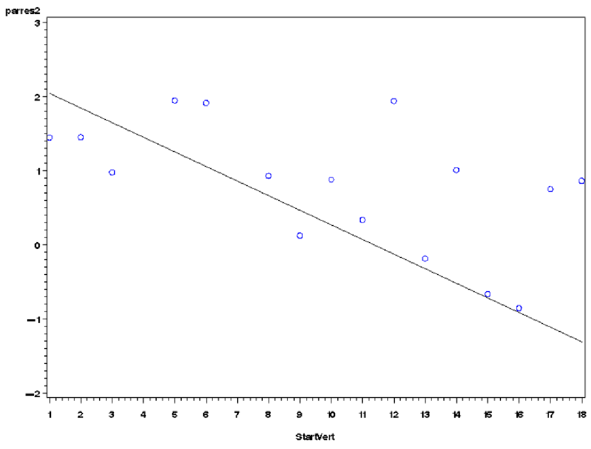


Figure 3.5: Partial residual plot for *STARTVERT*

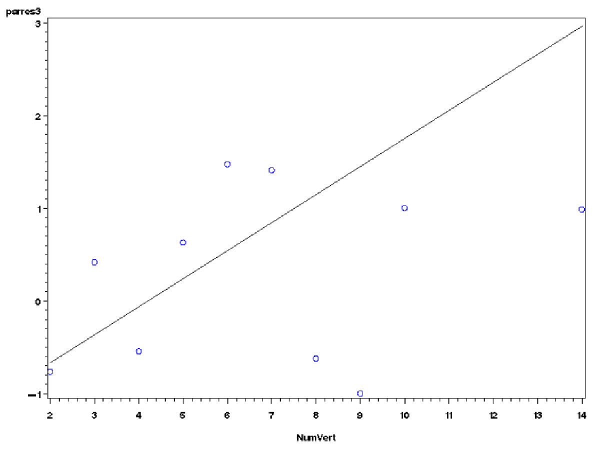


Figure 3.6: Partial residual plot for *NUMVERT*

The Boston Housing data set contains thirteen variables over which a search for a prediction model may be conducted. The variables are: *CRIM*, per capita crime rate by town, *ZN*, proportion of residential land zoned for lots over 25,000 square feet, *INDUS*, proportion of non-retail business acres per town, *CHAS*, Charles River dummy variable (1 if tract bounds river; 0 otherwise), *NOX*, Nitric oxides concentration (parts per 10 million), *RM*, average number of rooms per dwelling, *AGE*, proportion of owner-occupied units built prior to 1940, *DIS*, weighted distances to five Boston employment centres, *RAD*, index of accessibility to radial highways, *TAX*, full-value property-tax rate per \$10,000, *PTRAT*, pupil-teacher ratio by town, *B*, $1000(B_k - 0.63)^2$ where B_k is the proportion of blacks by town, and *LSTAT*, percent lower status of the population. The target variable is *MEDV*, the median value of owner-occupied homes in \$1,000s.

3.3.1 Methodology

In this section the interactive construction approach is described first and then the automated construction methodology.

Interactive Construction Methodology

In steps 1 to 4 of the interactive construction algorithm (Algorithm 2.1, Section 2.3), a Generalized Linear Model is fitted to provide initial estimates for the GANN.

The following partial residual plots (Figures 3.8 to 3.20) were obtained by Potts (2000), one partial residual plot for each variable after initial fitting of the linear model and GANN with a [2,2,2,2,2,2,2,2,2,2,2,2,2] architecture. Inspection of the partial residual plots alone is subjective and can result in the creation of suboptimal models.

1. Construct a GANN with a skip layer for each input in the model. In this step the univariate functions are initialized to $f_j(x_{ji}) = w_{0j}x_{ji}$. This gives 1 parameter for each input.
2. Fit a Generalized Linear Model to give initial estimates of the constant term and the w_{0j}.
3. Fit the full GANN model. Evaluate this model with the model selection criterion, set the flag *expanded* to $false$ (model available for expansion) and denote it as the root of the tree.
4. Search the tree for the best GANN model m, based on the model selection criterion, where the flag *expanded* is $false$. If such a model is found, set the flag *expanded* to $true$ (model expanded). If no model can be found with the flag *expanded* set to $false$, search the tree for the best model, report this model and terminate.
5. For each input x_i of model m (identified in step 4): If $1 \leq \text{sub}(x_i) \leq 9$, create a node (GANN model) n with the sub-architecture of x_i set to sub(x_i) - 1 and the remaining sub-architectures of m unchanged. Check whether node n has been previously created in the tree. If not, evaluate n with the model selection criterion, add n as a child node to the parent node m, and set the flag *expanded* of node n to $false$.
6. For each input x_i of model m: If $0 \leq \text{sub}(x_i) \leq 8$, create a node n with the sub-architecture of x_i set to $\text{sub}(x_i) + 1$ and the remaining sub-architectures of m unchanged. Check whether node n has been previously created in the tree. If not, evaluate n with the model selection criterion, add n as a child node to the parent node m, and set the flag *expanded* of node n to $false$.
7. Go to step 4.

Algorithm 3.2: Updated Automated Construction Algorithm

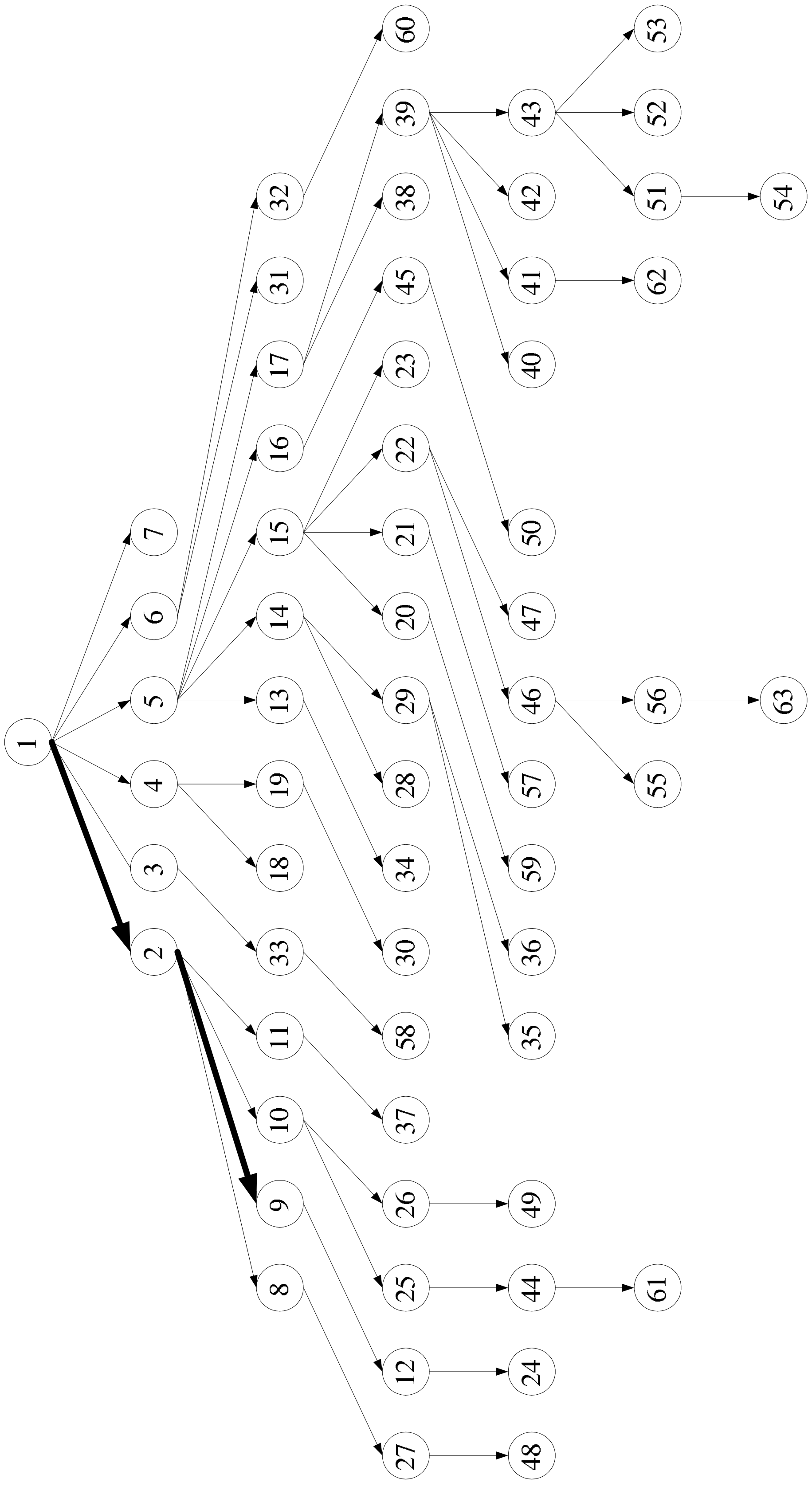


Figure 3.7: Search tree for Kyphosis example with [1,1,1] root node

NODE	SBC	ID	DEPTH	PARENT	NODE	SBC	ID	DEPTH	PARENT
1	-2.777	[1,1,1]	0	0	33	15.041	[1,0,2]	2	3
2	-5.597	[0,1,1]	1	1	34	28.594	[2,0,2]	3	13
3	4.375	[1,0,1]	1	1	35	19.271	[3,2,0]	4	29
4	-2.838	[1,1,0]	1	1	36	16.403	[2,3,0]	4	29
5	-3.992	[2,1,1]	1	1	37	16.172	[0,1,3]	3	11
6	3.446	[1,2,1]	1	1	38	27.055	[2,2,2]	3	17
7	8.194	[1,1,2]	1	1	39	-3.937	[2,1,3]	3	17
8	0.572	[0,0,1]	2	2	40	17.563	[1,1,3]	4	39
9	**-5.654**	**[0,1,0]**	**2**	**2**	41	26.729	[2,0,3]	4	39
10	0.384	[0,2,1]	2	2	42	31.532	[3,1,3]	4	39
11	6.017	[0,1,2]	2	2	43	13.037	[2,2,3]	4	39
12	-0.069	[0,2,0]	3	9	44	23.328	[0,3,2]	4	25
13	4.430	[2,0,1]	2	5	45	12.560	[2,3,1]	3	16
14	1.093	[2,1,0]	2	5	46	13.259	[3,3,1]	4	22
15	-0.539	[3,1,1]	2	5	47	23.323	[3,2,2]	4	22
16	9.311	[2,2,1]	2	5	48	20.647	[0,0,3]	4	27
17	7.240	[2,1,2]	2	5	49	26.990	[0,2,3]	4	26
18	12.135	[1,0,0]	2	4	50	18.927	[2,3,2]	4	45
19	3.148	[1,2,0]	2	4	51	6.391	[1,2,3]	5	43
20	15.079	[3,0,1]	3	15	52	32.123	[3,2,3]	5	43
21	14.371	[3,1,0]	3	15	53	33.525	[2,3,3]	5	43
22	10.887	[3,2,1]	3	15	54	41.349	[1,3,3]	6	51
23	19.796	[3,1,2]	3	15	55	29.323	[3,3,0]	5	46
24	13.728	[0,3,0]	4	12	56	30.875	[3,3,2]	5	46
25	7.781	[0,3,1]	3	10	57	30.367	[3,0,0]	4	21
26	12.505	[0,2,2]	3	10	58	21.163	[1,0,3]	3	33
27	11.390	[0,0,2]	3	8	59	30.956	[3,0,2]	4	20
28	17.110	[2,0,0]	3	14	60	23.761	[1,3,2]	3	32
29	5.564	[2,2,0]	3	14	61	32.478	[0,3,3]	5	44
30	16.622	[1,3,0]	3	19	62	36.457	[3,0,3]	5	41
31	16.966	[1,3,1]	2	6	63	37.735	[3,3,3]	6	56
32	15.256	[1,2,2]	2	6					

Table 3.3: Search tree for Kyphosis example with [1,1,1] root node

Having considered the partial residual plots and SBC, several architectural changes were made. Inputs *ZN, INDUS, CHAS, AGE,* and *B* (Figures 3.9, 3.10, 3.11, 3.14, and 3.19) make little or no contribution in describing variation in the target, and were removed. Four of the remaining eight inputs, namely *RAD, TAX, PTRAT,* and *LSTAT* were identified as having linear relationships with the target and were pruned back to a linear fit (Figures 3.16, 3.17, 3.18, and 3.20). Three inputs (*CRIM, NOX,* and *DIS*) had slight nonlinear relationships with the target and were kept at four degrees of freedom (Figures 3.8, 3.12, and 3.15). Finally, a neuron was added for one input (*RM,* Figure 3.13). These changes were made in several steps of pruning and refitting and resulted in the partial residual plots of Figures 3.21 to 3.28.

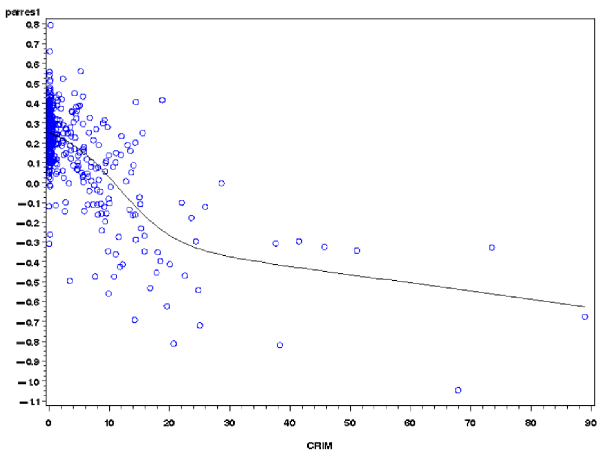


Figure 3.8: Partial residual plot for *CRIM*

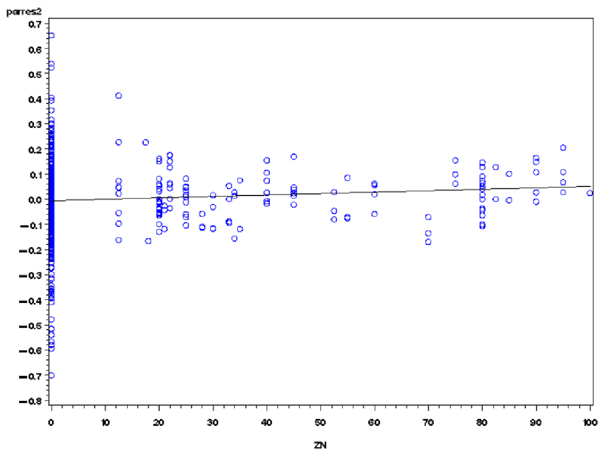


Figure 3.9: Partial residual plot for *ZN*

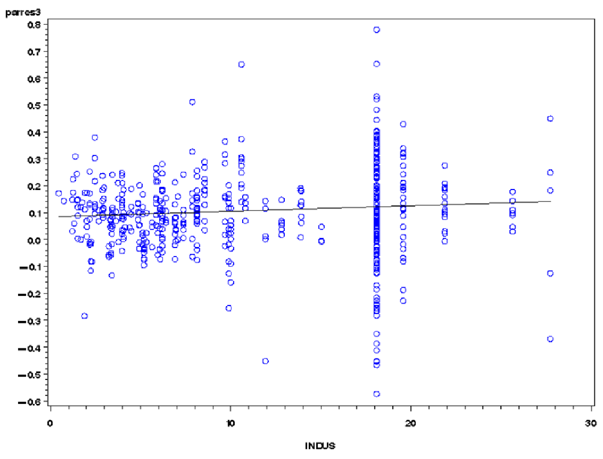


Figure 3.10: Partial residual plot for *INDUS*

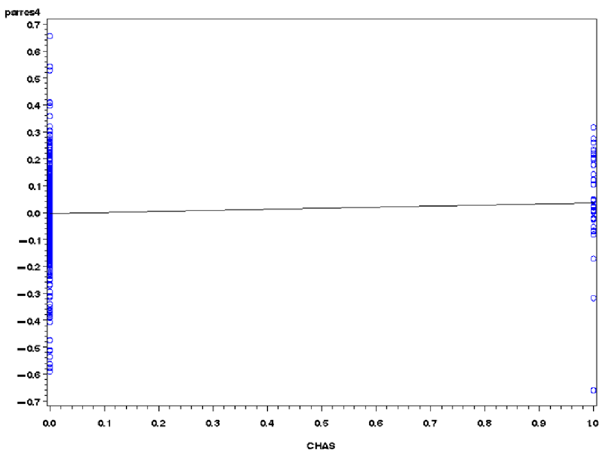


Figure 3.11: Partial residual plot for *CHAS*

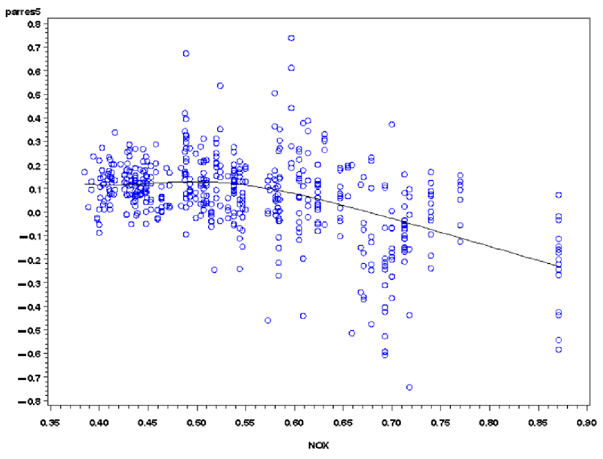


Figure 3.12: Partial residual plot for *NOX*

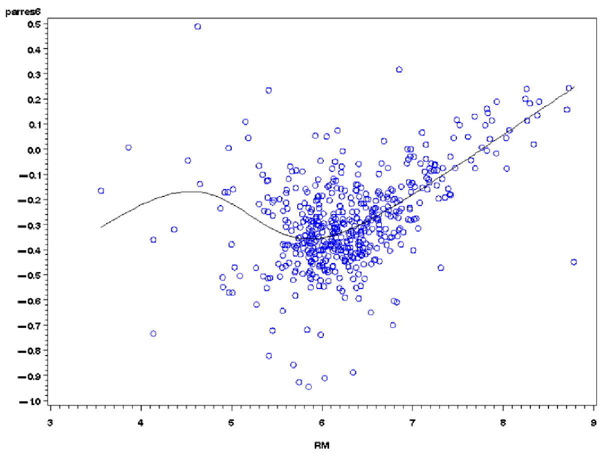


Figure 3.13: Partial residual plot for *RM*

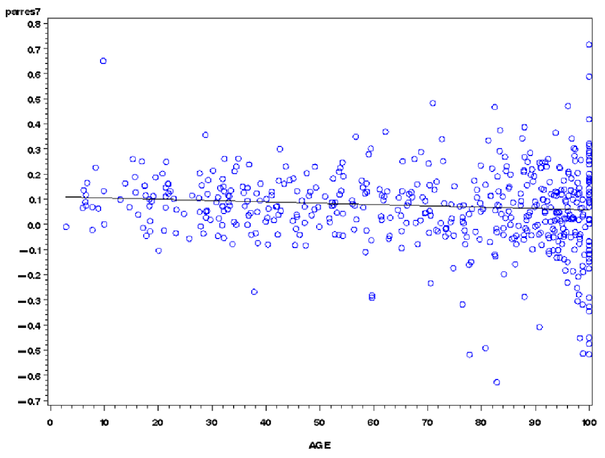


Figure 3.14: Partial residual plot for *AGE*

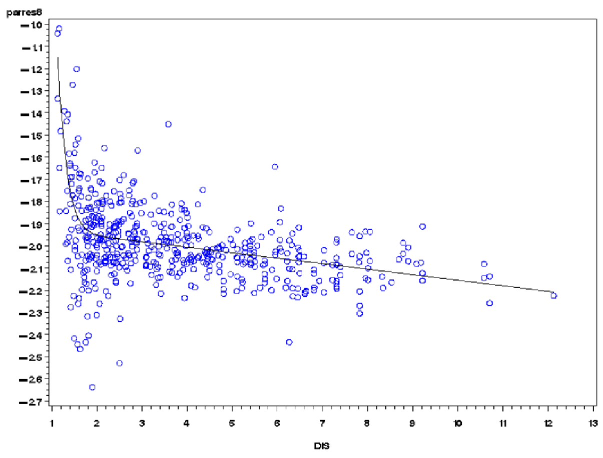


Figure 3.15: Partial residual plot for *DIS*

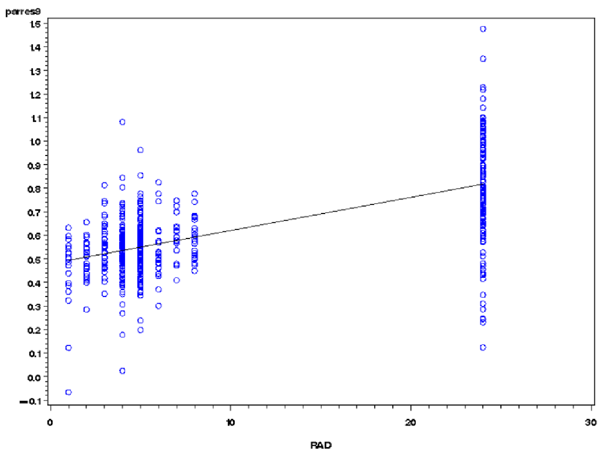


Figure 3.16: Partial residual plot for *RAD*

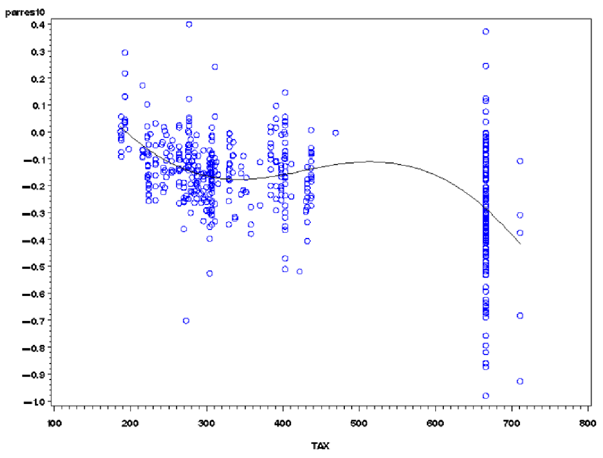


Figure 3.17: Partial residual plot for *TAX*

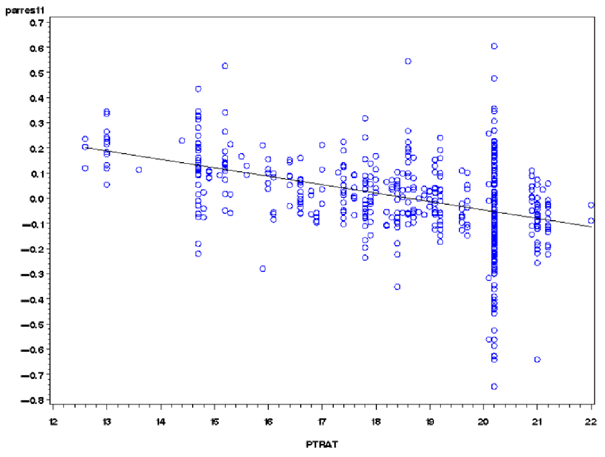


Figure 3.18: Partial residual plot for *PTRAT*

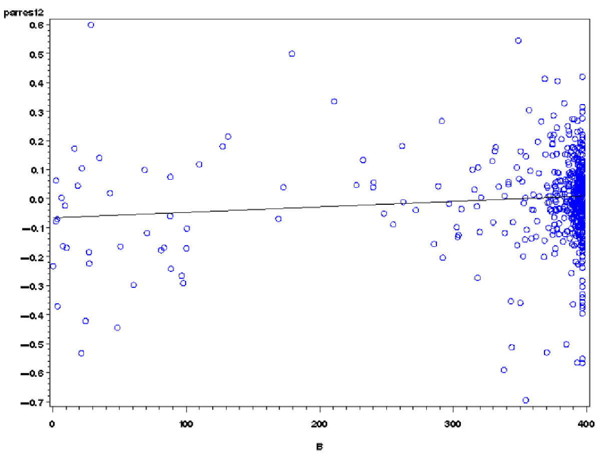


Figure 3.19: Partial residual plot for *B*

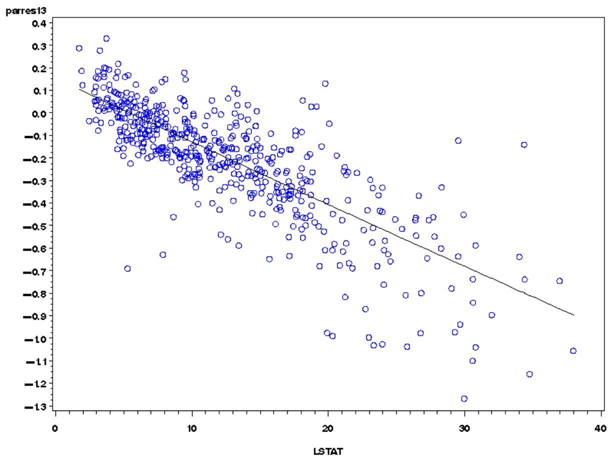
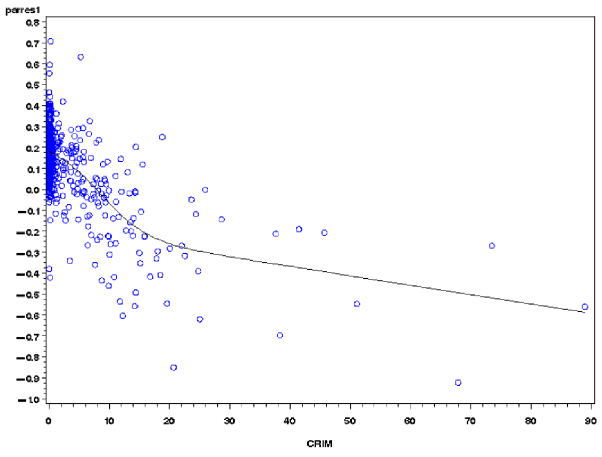


Figure 3.20: Partial residual plot for *LSTAT*



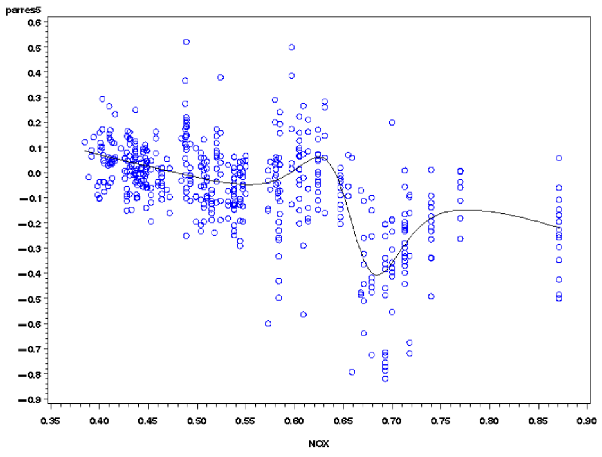
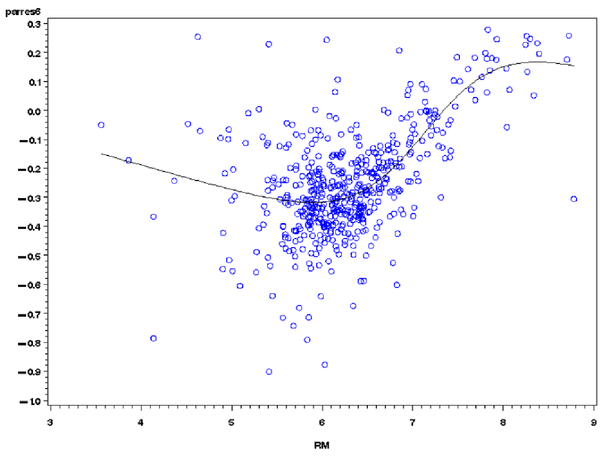


Figure 3.21: Partial residual plot for *CRIM* **Figure 3.22: Partial residual plot for *NOX***



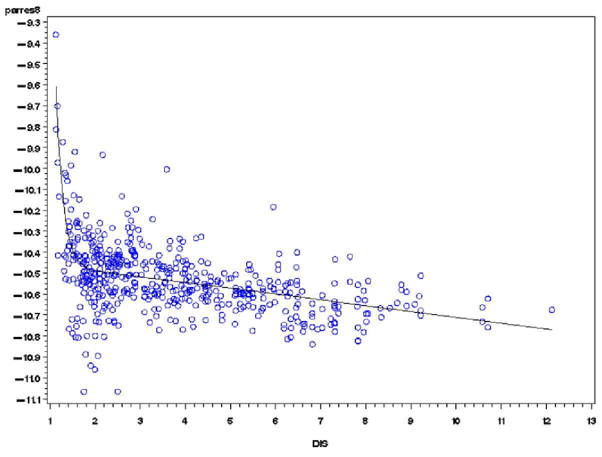


Figure 3.23: Partial residual plot for *RM* **Figure 3.24: Partial residual plot for *DIS***

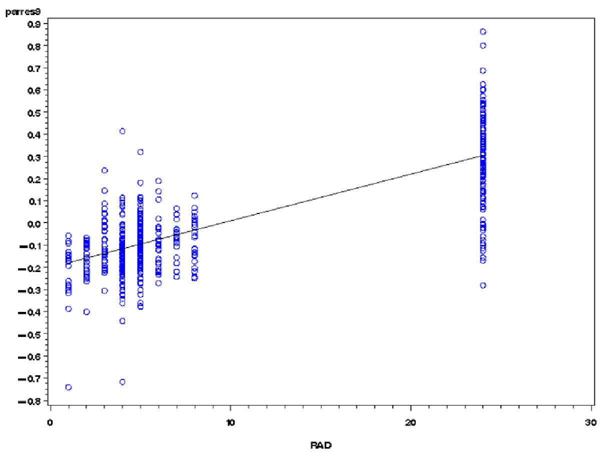


Figure 3.25: Partial residual plot for *RAD*

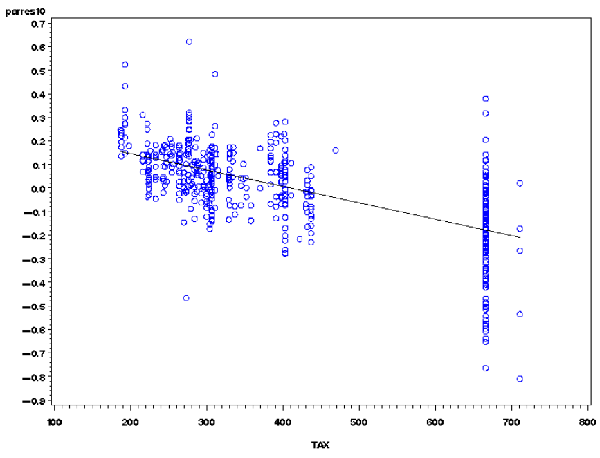


Figure 3.26: Partial residual plot for *TAX*

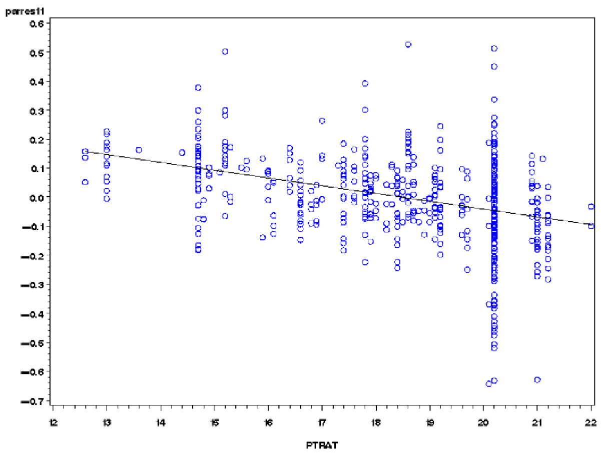


Figure 3.27: Partial residual plot for *PTRAT*

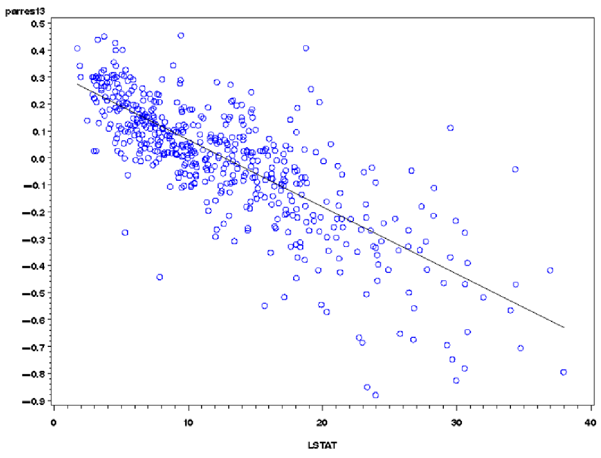


Figure 3.28: Partial residual plot for *LSTAT*

Potts (2000) performed model selection by considering the partial residual plots along with the SBC criterion which he defined[8] as

$$SBC = SSE + m\log(T).$$

The sum of squared errors (SSE) is defined as

$$SSE = \sum_{i=1}^{T} (y_i - \hat{y}_i)^2,$$

where y_i is the target values and $\hat{y}_i$ is the predicted target values.

Automated Construction Methodology

The automated construction algorithm is initialized with a linear model in step 1 and the sub-architecture space is limited to $\{0, 1, 2, 3\}$. There are four possible sub-architectures for twelve of the inputs and two possible sub-architectures for the binary input *CHAS*, resulting in a search space with a total size of $4^{12} \times 2^1 = 33,554,432$ GANN models.

[8]Note that this is not a standard definition of the SBC using least squares fit.

3.3.2 Results

The best models found by the interactive construction algorithm and the automated construction algorithm are displayed in Table 3.4.

	Architecture	SBC
Best model found interactively	[2,0,0,0,3,2,0,2,1,1,1,0,1]	343.0648361
Best model found by automated system	[1,0,0,0,2,2,0,2,1,1,1,0,1]	331.6428123

Table 3.4: Comparison of best models found

The best model was found by the automated construction methodology in under two minutes on a Pentium 4, 3.2 GHz Intel processor with 4 GB of RAM in SAS® Enterprise Miner™ 5.1. The search was continued for another 240 hours, but no better model could be found. More than ten models were found with SBC criterion values superior to that of the best model constructed utilizing the interactive construction methodology. The two models in Table 3.4 are nearly the same with only the first and fifth sub-architectures that differ.

The following four partial residual plots (Figures 3.29 to 3.32) show the difference between the best model found interactively (Figures 3.29 and 3.31) and the best model found by the automated system (Figures 3.30 and 3.32) for inputs one and five. It is clear that the model found by the automated system is simpler, as the sub-architectures for both variables have been simplified (variable one (*CRIM*) has been simplified to a linear contribution and variable five (*NOX*) to a slight non-linear contribution).

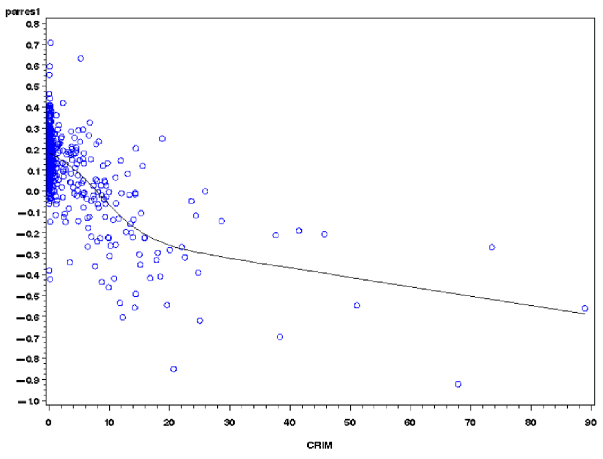


Figure 3.29: Partial residual plot for *CRIM*

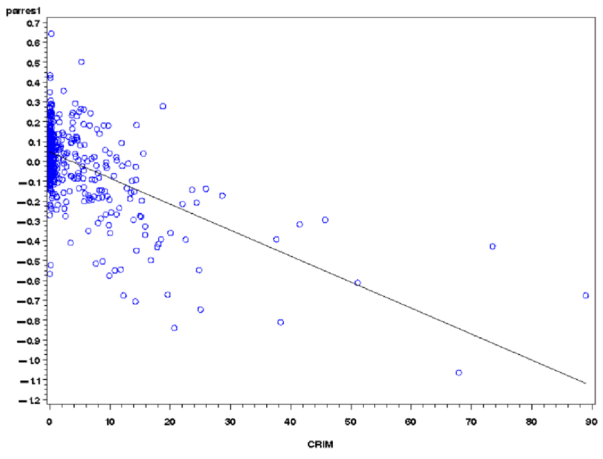


Figure 3.30: Partial residual plot for *CRIM*

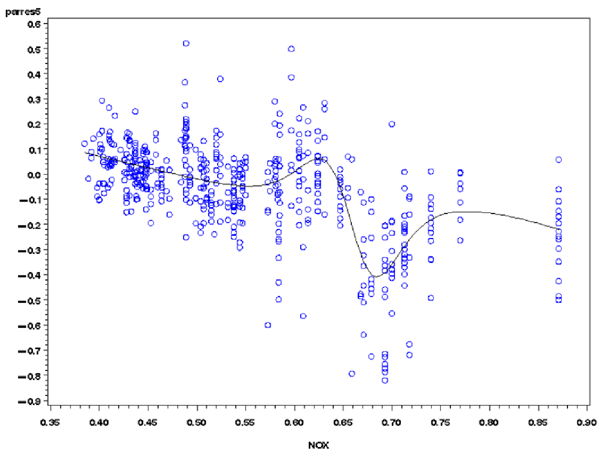


Figure 3.31: Partial residual plot for *NOX*

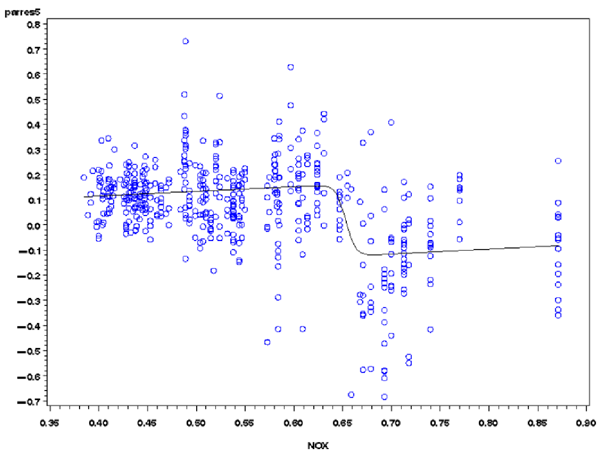


Figure 3.32: Partial residual plot for *NOX*

Although the mean squared error (MSE) and the sum of squared errors (SSE) of the model found by the automated approach are worse than the model found by the interactive approach (Table 3.5), the former model is more parsimonious. This model with fewer parameters is better able to generalize from new, unseen inputs. A model with too many parameters tends to overfit and its generalization capability is worse than a model with fewer parameters.

	MSE	SSE
Best model found interactively	9.225975	4373.112
Best model found by automated system	9.964614	4783.015

Table 3.5: Comparison of best models found

To verify the latter statement, an experiment was conducted to compare the in- and out-of-sample performance of the two models. The data set was randomly partitioned into two equally sized (training and test) sub-sets. The two models were trained on the training set and scored on the test set. This process was repeated one hundred times and the average mean-squared errors were determined for the two sets. The following results were obtained.

	MSE (training set)	MSE (test set)
Best model found interactively	10.352	16.373
Best model found by automated system	10.815	14.139

Table 3.6: Comparison of best models found

From Table 3.6 it can be seen that the out-of-sample performance of the simpler model found by the automated approach is superior to that of the more complex model constructed using the interactive approach.

3.3.3 Conclusions

The automated construction algorithm found a better model than the one found by Potts (2000) using the interactive construction methodology. This model was found in less than two minutes and is not only simpler in structure but has better out-of-sample performance. Furthermore, it is very unlikely that even the very skilled neural network modeler could arrive at the results of Potts (2000) using his interactive algorithm in less than two minutes. The automated algorithm therefore represents a substantial improvement over the method of Potts (1999).

In the next section, a meteorological data set is analyzed to compare the performance of the automated construction methodology to other model selection techniques found in the literature.

3.4 Meteorological example

The Ozone data of Breiman & Friedman (1985) has 330 observations of the response variable, groundlevel ozone (as a pollutant), and nine explanatory variables. Eight of the predictors are broadly meteorological by nature and the ninth is the day of year. *VH* is the altitude (m) at which the pressure is 500 millibars; *WIND*, the wind speed (mph) at Los Angeles International Airport (LAX); *HUMID*, the humidity (%) at LAX; *TEMP*, the temperature (degrees F) at Sandburg Air Force Base; *IBH*, the temperature inversion base height (feet); *DPG*, the pressure gradient (mm Hg) from LAX to Daggert; *IBT*, the inversion base temperature (degrees F) at LAX; *VIS*, the visibility (miles) at LAX; and *DOY*, the day of year.

3.4.1 Methodology

This data set has been analyzed by a number of people in the nonparametric regression literature (Lee, 1999), and so it can be used to compare the automated construction algorithm to other nonparametric regression techniques. Breiman & Friedman (1985) utilized this data set in their paper on Alternating Conditional Expectation (ACE). They used the estimated multiple correlation coefficient, R^2, as a goodness-of-fit measure and fit the model using a subset of only four variables (*TEMP, IBH, DPG,* and *VIS*) that were chosen by a stepwise algorithm. Hastie & Tibshirani (1986) fit a Generalized Additive Model to the data. Friedman & Silverman (1989) use TURBO to fit the data. Hawkins (1989) discussed the previous paper and fits the data with linear regression after using Box-Tidwell style transformations on the variables. Lee (1999) utilizes a MLP neural network model with three nodes in the hidden layer and five explanatory variables.

3.4.2 Results

The best GAM model found by the automated construction algorithm in less than 1 minute with a sub-architecture space of {0,1,2,3} is included in Table 3.7. This model was found on a Pentium

4, 3.2 GHz Intel processor with 4 GB of RAM in SAS® Enterprise Miner™ 5.1. The search was continued for another 10 hours but no better model could be found. Note that there are four possible sub-architectures for each of the nine inputs, resulting in a search space with a total size of $4^9 = 262,144$ GANN models.

Table 3.7 shows that all of the above methods have similar goodness-of-fit to the data. All the methods do manage to find a reasonable fit, but none is substantially better than the others.

Method	R^2
ACE, 4 variables	0.78
GAM	0.80
TURBO	0.80
Box-Tidwell	0.82
Neural Network	0.79
GAM (automated system)	0.80

Table 3.7: Comparison of competing methods

Hastie & Tibshirani (1990) performed a comparison of several methods on this data set in terms of variable selection. In addition to some of the above methods, they also included a stepwise algorithm for their GAM models, as well as a response to TURBO which they call BRUTO. The latter system is meant to do automatic variable selection and smoothing parameter selection. In Table 3.8 the variables chosen by the the models in each of these methods are displayed.

Method	*VH*	*WIND*	*HUMID*	*TEMP*	*IBH*	*DPG*	*IBT*	*VIS*	*DOY*
Stepwise ACE				X	X	X		X	
Stepwise GAM	X	X	X	X	X	X		X	X
TURBO	X			X	X	X		X	X
BRUTO				X	X	X		X	X
Neural Network	X		X			X	X		X
GAM (automated system)	X	X	X			X	X	X	X

Table 3.8: Comparison of variable selection

It is interesting to note that there is serious disagreement by the methods on which variables to select. This may be in part because the variables are highly correlated with one another other (Lee, 1999), so that different subsets may give similar predictions.

3.4.3 Conclusions

From Table 3.8 it can be seen that TURBO and BRUTO are largely in agreement with each other. It seems clear that some variable selection is necessary because of the high level of correlation between the explanatory variables, even if there is a difference of opinion about which subset is optimal.

Next, the prototype implementation of the automated system in the powerful SAS® programming language is described.

3.5 Implementing the automated construction algorithm

The SAS®[9] programming language is an integrated collection of data management, analysis, and reporting tools. The data management features allow the user to read and combine data files in many ways. The analysis capabilities extend from simple frequency distributions through to complex multivariate techniques. Finally, the reporting features give the user the capability of presenting data management and analysis results in a large number of formats.

The power of the SAS® System is that it is integrated. Data handled by its data management facilities can be used without modification by the analysis and reporting components. This feature allows the user to work with minimal concern for data formats and structures. The system is also integrated across different computing environments. A SAS® program written on an IBM-compatible personal computer will run with almost no modification on a mainframe or minicomputer. SAS® was chosen as the language to implement the automated construction algorithm. This implementation is called AutoGANN[10]. SAS® has a number of vertical solutions and AutoGANN was integrated with the Enterprise Miner™ solution. Currently, Enterprise Miner™ is the most complete and powerful data mining solution on the market and streamlines the entire data mining process from data access to model deployment by supporting all necessary tasks within a single, integrated solution, all while providing the flexibility for efficient workgroup collaborations (SAS Institute Inc., 2005).

The SAS® programming language has an experimental procedure for fitting GAMs using the backfitting algorithm. PROC GAM provides an array of powerful tools for data analysis, based on nonparametric regression and smoothing techniques. Very little has been written about it by SAS® users since the inception of SUGI (SAS® User's Group International Conference). Furthermore,

[9]SAS Institute Inc. was founded by Dr. Jim Goodnight and John Sall in 1976 and is currently the world's largest privately held software company (SAS Institute Inc., 2006). SAS® is leading in business intelligence software and services with customers at 40,000 sites. SAS® software is used by 96 of the top 100 FORTUNE Global 500 companies to manage and gain insights from vast amounts of data, resulting in faster, more accurate business decisions, more profitable relationships with customers and suppliers, compliance with governmental regulations, research breakthroughs and better products. SAS® has customers in 110 different countries and more than 2,200 universities utilize SAS® software. The worldwide revenue for 2005 was \$1.68 billion and 24% of revenue was reinvested in research and development in the same year.

[10]AutoNeural is an automated tool in Enterprise Miner™ that assists the user to find optimal configurations for a neural network model. AutoGANN was developed independent of AutoNeural more or less simultaneously.

PROC GAM is not implemented in Enterprise Miner™ as a modeling node. By implementing the automated construction algorithm in Enterprise Miner™, this void is filled from a neural network perspective. Research on the automated construction of GANNs and the implementation in SAS® not only extend work done by Sarle (1994) and Potts (1999), but AutoGANN provides the data analyst with a more user-friendly tool than PROC GAM.

Meta-programs can be designed with the Macro Language of Base SAS®, that is, programs that create and execute other programs. This powerful capability allows the program to create GANN source code in real time, execute it, and then assess the predictive power of these models. By automating the construction of GANN models in SAS®, they can be evaluated in a fraction of the time it would take a human to code the models by hand.

3.5.1 AutoGANN description

AutoGANN was implemented in the SAS® Macro Facility, a tool within Base SAS® software that enables the use of macros (Carpenter, 2004). The macro facility is, first and foremost, a source code generator that incorporates a macro processor to translate macro code into statements that can be utilized by SAS® and the macro language. The macro language provides a way to communicate with the macro processor. The macro language provides tools that allows the programmer to:

- communicate information between DATA and PROC steps
- dynamically create SAS® code after the user submits the program for execution
- execute DATA or PROC steps conditionally
- create code that is flexible and generalizable.

A DATA step enables the programmer to read raw data or other SAS® data sets and to create a SAS® data set, write a report, or write to an external file. A PROC step is part of a SAS® program that invokes a SAS® procedure.

Figure 3.33 indicates the four basic steps of the AutoGANN implementation. These steps are described next.

Initialize AutoGANN system

The first step reserves adequate main memory for execution of the program. Then the six parameters of AutoGANN are checked for consistency. These parameters are *Criterion, Start Architecture, Search Space, Partial Residual Plots, Time,* and *Number of Models.*

The *Criterion* parameter sets the model selection criterion that is used to evaluate different GANN models. A default value of *SBC_dev* defines the SBC criterion for a least squares analysis.

Other values include *SBC_like* where the likelihood of the model is utilized to calculate the SBC criterion value. Also, the Akaike information criterion, *AIC_dev* and *AIC_like*, is implemented for least squares and likelihood analysis. The GANN architecture of the root node in the search tree is defined by the *Start Architecture* parameter. *Linear* is the default start architecture and causes the search to begin with a linear model. To start the search with a [2,2,2] model, for example, the *Start Architecture* parameter should be set to 222 (without square brackets and commas). The sub-architecture space is defined by the *Search Space* parameter and has a default value of 0123. Creation of partial residual plots are turned on or off by the *Partial Residual Plots* parameter. The default option (*Yes*) is to create partial residual plots after the algorithm terminates. It is possible to force the algorithm to stop after a certain length of time has elapsed by setting the *Time* parameter - in this case the best model that was found up until termination of the algorithm is reported. A default value of *Initialize* causes the algorithm to execute steps 1 to 4, that is, terminate after the root node is created and evaluated. There are a number of different time settings ranging from 15 seconds up to 5 days. Finally, the *Number of Models* parameter sets the number of models that are used for model averaging. The latter technique that extends the automated construction algorithm is discussed in detail in the next chapter. A default value of 1 causes the algorithm to search for the single best model. Possible values range from 1 to 10. For a value larger than 1, the technique of model averaging is applied to find a more stable GANN model by averaging over the best chosen number of models found.

Inconsistent parameters may cause the system to malfunction. One example of an inconsistent parameter definition is when the algorithm is initialized with a three-input linear architecture when there are more than three inputs in the data set. In the latter case, an error message is generated and the program terminates.

Finally, after analyzing the input data set, source code for a skeletal GANN system is generated that can be configured to represent any GANN model in the search space. This skeletal GANN code is modified by the automated construction algorithm to create different GANN models in the search tree.

Execute automated construction algorithm

The automated construction algorithm is executed in the second step of the AutoGANN system. For simplicity, the tree of GANN models is maintained in a list data structure stored inside the computer's main memory. The list is sorted in increasing order of model selection criterion values so that the best model is always found at the start of the list. Extending the list to include new information is relatively easy with only minor changes to the source code. When the algorithm terminates, results are exported to an external file to allow other programs to utilize the results. Also, fit statistics for the best GANN model is calculated and presented. Score code is automatically generated and applied to a score data set if such a set is present. The size of problems AutoGANN can solve is only limited by the amount of memory available. Table 3.9 contains the activation and error functions currently implemented in AutoGANN.

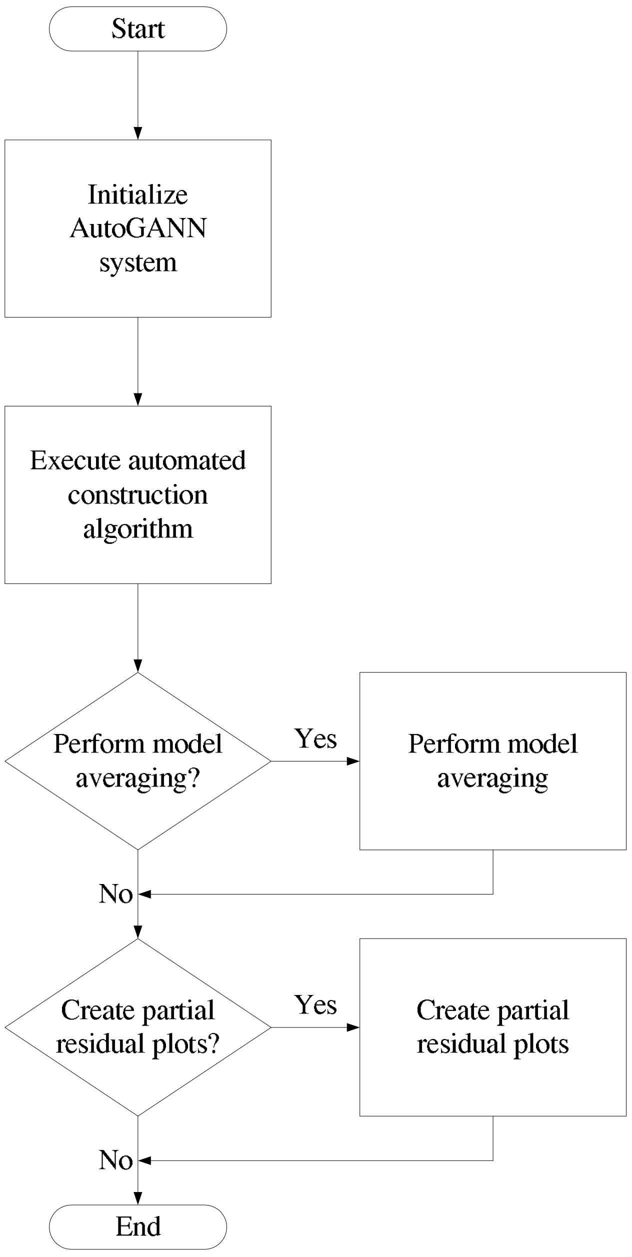

Start
Initialize AutoGANN system
Execute automated construction algorithm
Perform model averaging?
Yes
Perform model averaging
No
Create partial residual plots?
Yes
Create partial residual plots
No
End

Figure 3.33: AutoGANN flowchart

Activation Function	Link Function	Target Scale	Error function
Identity	Identity	Interval on $[-\infty, +\infty]$	Normal
Hyperbolic Tangent	Inverse Hyperbolic Tangent	Interval on [-1,1]	Normal
Exponential	Log	Nonnegative	Poisson
Multiple Logistic	Logit	Binary	Multiple Bernoulli

Table 3.9: Activation functions implemented in AutoGANN

The activation functions implemented in AutoGANN solve a large class of problems, but can be extended (see Potts (2000)).

Perform model averaging

When the user decides to construct a more stable GANN model than the single best model found with the automated construction algorithm, model averaging is performed in step three. This technique is discussed in Chapter 4.

Create partial residual plots

Partial residual plots are generated in step four. With model selection, partial residual plots of the best GANN model are created. For model averaging, partial residual plots of the combined GANN model are produced. In addition to the partial residuals and fitted splines, ticks are added to the plots to provide insight into the distribution of function values. Adding ticks is an extension to the method in which partial residual plots are created by Potts (2000). An example of this improved partial residual plot can be seen in Figure 3.39.

3.5.2 AutoGANN user interface

In Figure 3.34, the adjustable parameters of AutoGANN are displayed. The user interface is easy to use and intuitive with only six settings[11]. When the user decides to utilize default values, a linear model and partial residual plots are created. Normally, analysis of a data set is started with default parameter values and is then repeated with specific parameter settings.

Figure 3.35 illustrates a typical experiment where the Housing data set is analyzed by the AutoGANN system in Enterprise Miner™ 5.1. The four output windows obtained when the system terminates are shown in Figures 3.36 to 3.39. First a summary of the best model found is given in Figure 3.36. The user can view the sub-architecture as well as a description of each input. In Figure 3.37 the search tree is presented with enough information to reconstruct the search path. Models are sorted by model selection criterion values. The number of parameters (PARAMS) for each model is calculated and the *expanded* flag utilized by the automated construction algorithm is indicated by USED. A value of 0 for USED denotes a node available for expansion and a value of 1 denotes a node already expanded. Fit statistics of the best model are given in Figure 3.38. No validation and test sets were available, which explains the missing values in the table. Finally, the user can also inspect partial residual plots (Figure 3.39) created by the system by choosing the *View* option.

[11]Only the most important parameters can be adjusted to keep the system as simple as possible. Many parameter settings are chosen by the automated system in the background to relieve the user from making unnecessary decisions. In contrast to AutoGANN, the Regression Node in Enterprise Miner™ has 49 parameter settings.

Next an asymptotic analysis of the complexity of the AutoGANN system is presented.

3.5.3 AutoGANN complexity analysis

Often computer scientists must compare two algorithms to decide which runs faster or takes less memory. In general, there are two approaches to this task, namely benchmarking and a mathematical analysis of the algorithms (Russel & Norvig, 1995).

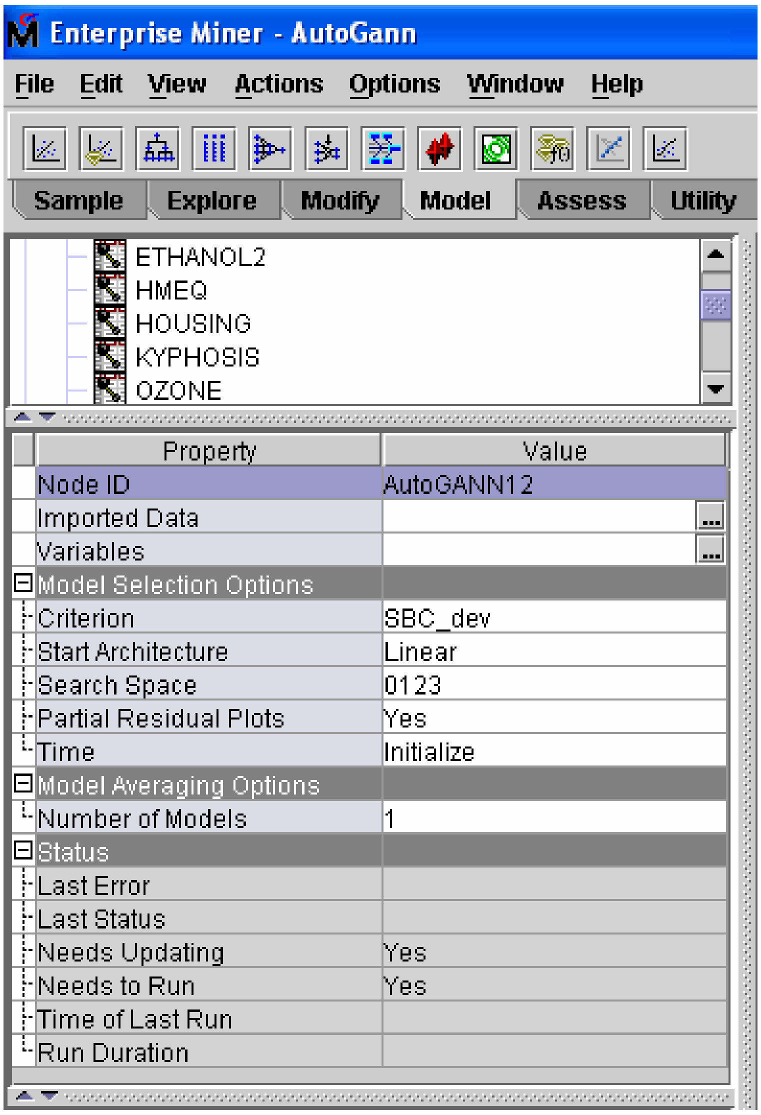
With benchmarking, the two algorithms are executed on a computer and a measurement is made to determine which is faster or which uses less memory. Unfortunately, a benchmark can be unsatisfactory because it is too specific. The performance of a particular program written in a particular language on a particular computer with a particular compiler and particular input data is measured. It can be difficult to predict how well the algorithm would do on a different compiler, computer, data set, or programmer from the single result that the benchmark provides.

A useful variant to benchmarking relies on a mathematical analysis of algorithms, independent of the particular implementation and input. This approach counts the number of operations of a particular kind performed. The first step in the analysis of the algorithm is to abstract over the input, to obtain some parameter or parameters that characterize the size of the input, called n.

The second step is to abstract over the implementation in order to find some measure that reflects the running time of the algorithm, but is not dependent on a particular compiler or computer. This could be the number of lines of code executed, or it could be more detailed, measuring the number of assignments, additions, array references, and branches executed by the algorithm. The $O()$ notation is used for an asymptotic analysis. As n asymptotically approaches infinity, an $O(n)$ algorithm is better than an $O(n^2)$ algorithm. A single benchmark figure could not confirm such a claim. Asymptotic analysis is the most widely used tool for analyzing algorithms.

AutoGANN is a meta-program that performs a best-first search over a search space of GANN models. Best-first search resembles depth-first search in the way it prefers to follow a single path all the way to the goal (best GANN model). When it hits a dead end, it will back up. The automated construction algorithm defines a finite sub-architecture space and the search space is therefore finite (because there is a finite number of inputs or variables). As the algorithm does not allow for duplicate states in the search space, the search space of possible models are therefore finite and the search strategy is thus also complete (will generate all possible GANN models) and optimal (the best solution (model) given that the restricted architectures will always be found if the search continues for long enough).

The worst-case time complexity for best-first search is $O(b^m)$, where m is the maximum depth of the search space and b the branching factor of the nodes (each node in the search space represents a GANN model). Since best-first search retains all nodes in memory, its space complexity is the same as its time complexity.



Enterprise Miner - AutoGann
File Edit View Actions Options Window Help
Sample Explore Modify Model Assess Utility
ETHANOL2
HMEQ
HOUSING
KYPHOSIS
OZONE

Property	Value
Node ID	AutoGANN12
Imported Data	...
Variables	...
⊟ Model Selection Options	
Criterion	SBC_dev
Start Architecture	Linear
Search Space	0123
Partial Residual Plots	Yes
Time	Initialize
⊟ Model Averaging Options	
Number of Models	1
⊟ Status	
Last Error	
Last Status	
Needs Updating	Yes
Needs to Run	Yes
Time of Last Run	
Run Duration	

Figure 3.34: AutoGANN parameters in Enterprise Miner™

When the automated construction algorithm commences, the branching factor is usually twice the number of inputs (each input is first pruned and then grown - see steps 6 and 7 of the automated construction algorithm). No duplicates are allowed in the search tree which causes the branching factor to decrease as search progresses deeper in the tree. The amount of reduction depends on the particular problem.

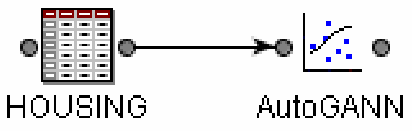
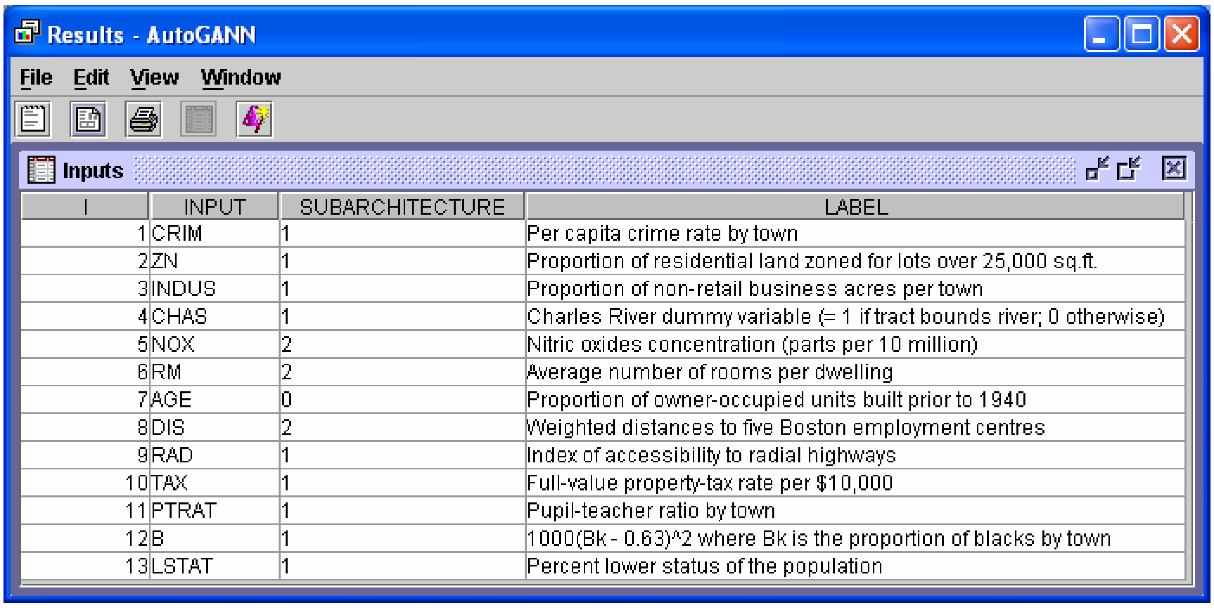
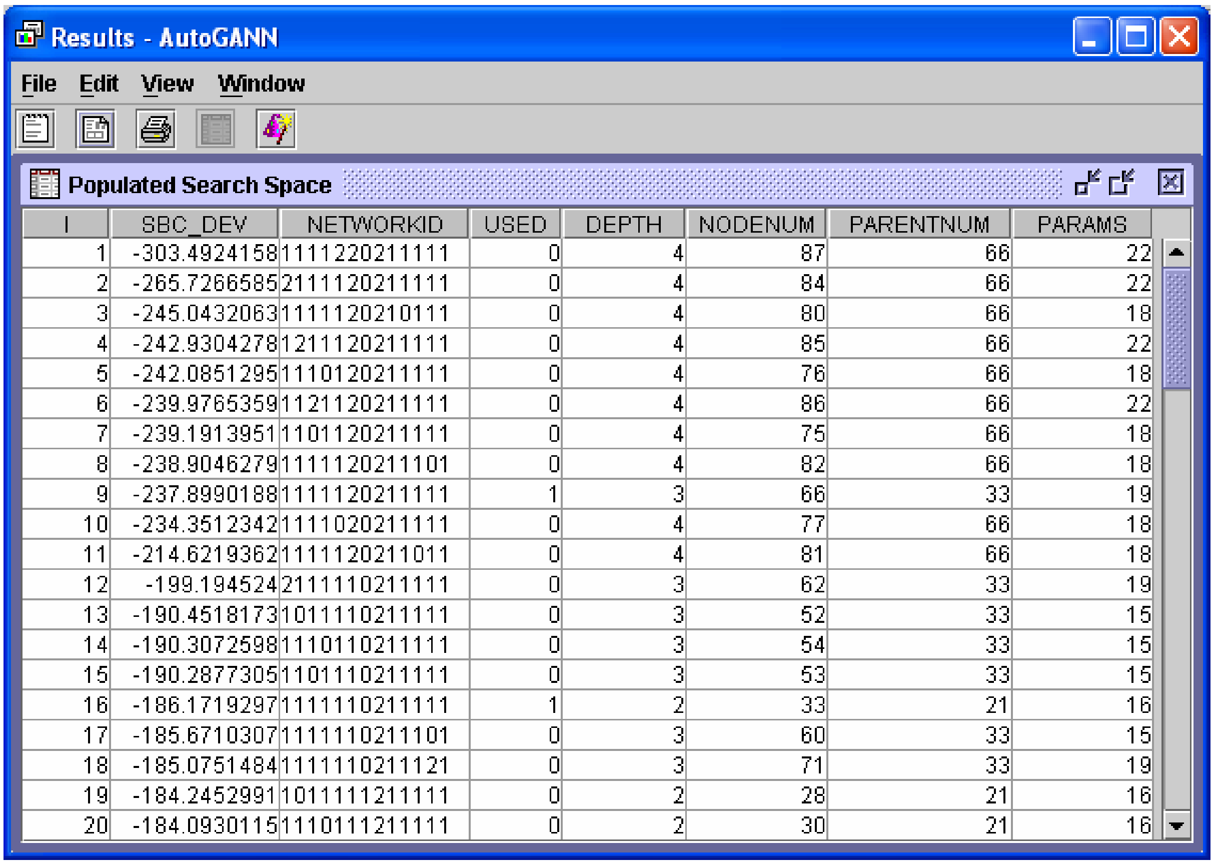


Figure 3.35: AutoGANN analysis of Housing data set



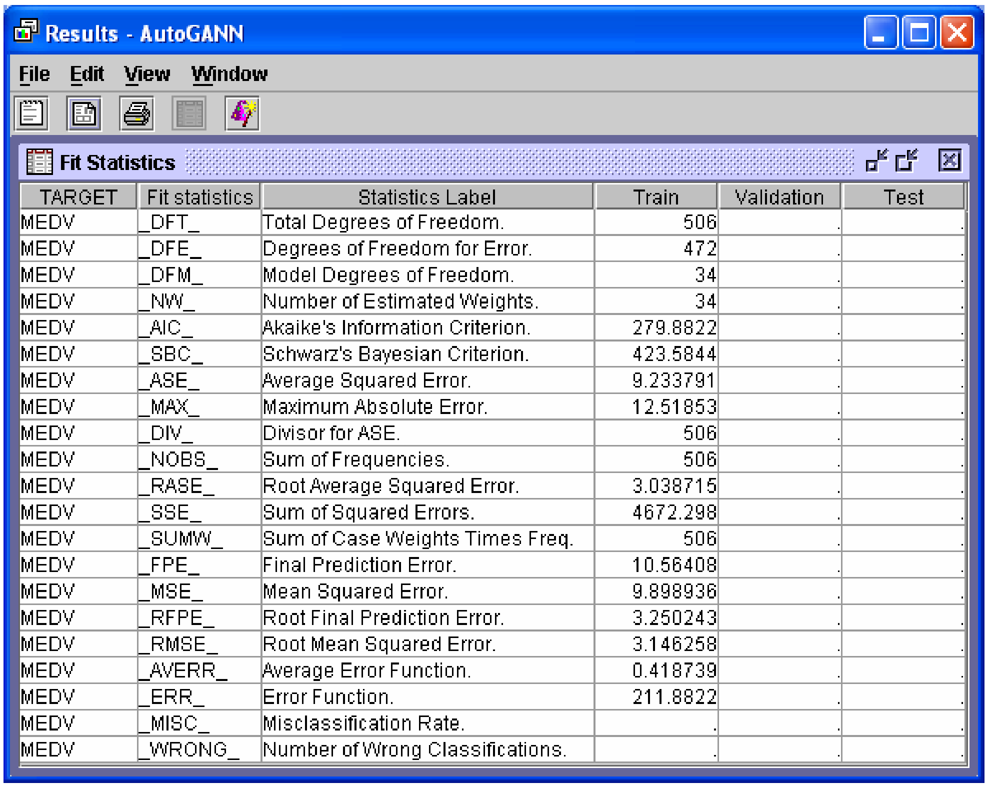
I	INPUT	SUBARCHITECTURE	LABEL
1	CRIM	1	Per capita crime rate by town
2	ZN	1	Proportion of residential land zoned for lots over 25,000 sq.ft.
3	INDUS	1	Proportion of non-retail business acres per town
4	CHAS	1	Charles River dummy variable (= 1 if tract bounds river; 0 otherwise)
5	NOX	2	Nitric oxides concentration (parts per 10 million)
6	RM	2	Average number of rooms per dwelling
7	AGE	0	Proportion of owner-occupied units built prior to 1940
8	DIS	2	Weighted distances to five Boston employment centres
9	RAD	1	Index of accessibility to radial highways
10	TAX	1	Full-value property-tax rate per $10,000
11	PTRAT	1	Pupil-teacher ratio by town
12	B	1	1000(Bk - 0.63)^2 where Bk is the proportion of blacks by town
13	LSTAT	1	Percent lower status of the population

Figure 3.36: AutoGANN output



I	SBC_DEV	NETWORKID	USED	DEPTH	NODENUM	PARENTNUM	PARAMS
1	-303.4924158	1111220211111	0	4	87	66	22
2	-265.7266585	2111120211111	0	4	84	66	22
3	-245.0432063	1111120210111	0	4	80	66	18
4	-242.9304278	1211120211111	0	4	85	66	22
5	-242.0851295	1110120211111	0	4	76	66	18
6	-239.9765359	1121120211111	0	4	86	66	22
7	-239.1913951	1101120211111	0	4	75	66	18
8	-238.9046279	1111120211101	0	4	82	66	18
9	-237.8990188	1111120211111	1	3	66	33	19
10	-234.3512342	1111020211111	0	4	77	66	18
11	-214.6219362	1111120211011	0	4	81	66	18
12	-199.194524	2111110211111	0	3	62	33	19
13	-190.4518173	1011110211111	0	3	52	33	15
14	-190.3072598	1110110211111	0	3	54	33	15
15	-190.2877305	1101110211111	0	3	53	33	15
16	-186.1719297	1111110211111	1	2	33	21	16
17	-185.6710307	1111110211101	0	3	60	33	15
18	-185.0751484	1111110211121	0	3	71	33	19
19	-184.2452991	1011111211111	0	2	28	21	16
20	-184.0930115	1110111211111	0	2	30	21	16

Figure 3.37: AutoGANN output



TARGET	Fit statistics	Statistics Label	Train	Validation	Test
MEDV	_DFT_	Total Degrees of Freedom.	506	.	.
MEDV	_DFE_	Degrees of Freedom for Error.	472	.	.
MEDV	_DFM_	Model Degrees of Freedom.	34	.	.
MEDV	_NW_	Number of Estimated Weights.	34	.	.
MEDV	_AIC_	Akaike's Information Criterion.	279.8822	.	.
MEDV	_SBC_	Schwarz's Bayesian Criterion.	423.5844	.	.
MEDV	_ASE_	Average Squared Error.	9.233791	.	.
MEDV	_MAX_	Maximum Absolute Error.	12.51853	.	.
MEDV	_DIV_	Divisor for ASE.	506	.	.
MEDV	_NOBS_	Sum of Frequencies.	506	.	.
MEDV	_RASE_	Root Average Squared Error.	3.038715	.	.
MEDV	_SSE_	Sum of Squared Errors.	4672.298	.	.
MEDV	_SUMW_	Sum of Case Weights Times Freq.	506	.	.
MEDV	_FPE_	Final Prediction Error.	10.56408	.	.
MEDV	_MSE_	Mean Squared Error.	9.898936	.	.
MEDV	_RFPE_	Root Final Prediction Error.	3.250243	.	.
MEDV	_RMSE_	Root Mean Squared Error.	3.146258	.	.
MEDV	_AVERR_	Average Error Function.	0.418739	.	.
MEDV	_ERR_	Error Function.	211.8822	.	.
MEDV	_MISC_	Misclassification Rate.	.	.	.
MEDV	_WRONG_	Number of Wrong Classifications.	.	.	.

Figure 3.38: AutoGANN output

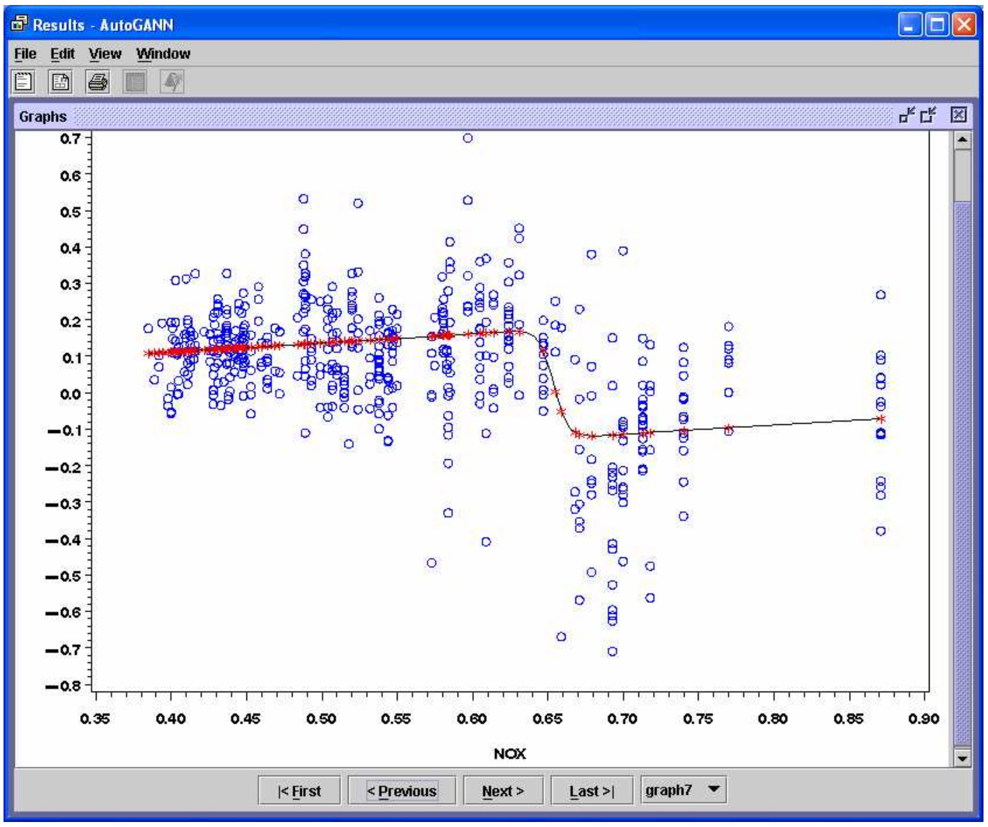


Figure 3.39: AutoGANN output

With the Kyphosis example in Section 3.2 there are three inputs. The root node in Figure 3.7 has a branching factor of six which decreases for nodes further down the tree. Also, when the search is started from a linear model, the maximum tree depth is $k(p-1)$, where k is the number of inputs and p is the size of the sub-architecture space.

3.6 Conclusions

In this chapter a new methodology was presented that automates the interactive construction algorithm discussed in Chapter 2. Although the interactive method has merit, two difficulties were identified. First, human judgment is required to assess the functional relationships between inputs and the target. This can lead to results influenced by personal opinion. Second, assessing the functional relationships for a large number of inputs can be a time consuming procedure which discourages the use of this technique.

The automated construction algorithm solves the difficulties mentioned above by incorporating an objective measure (model selection criterion) to guide search for the best GANN model. As a result, no human involvement is necessary during execution of the algorithm. The data analyst must only set the parameters of the algorithm beforehand and interpret the results when the algorithm terminates. The automated algorithm is capable of performing in-sample model selection as well as cross-validation. This best-first search technique is complete and optimal, given adequate time to evaluate candidate models. The three examples presented in this chapter revealed that the technique is nontrivial (medical example), powerful and effective (housing and meteorological examples), and produces results comparable to other nonlinear model selection techniques found in the literature (meteorological example). The algorithm is implemented in the powerful SAS® programming language with a simple, user-friendly, and intuitive user interface. A linear model can be constructed and interpreted with the default parameter settings. Also, output from the implementation can guide the data analyst to gain insight into the models developed.

In the next chapter model selection criteria and an extension to the automated construction algorithm are discussed. Model averaging produces a more stable GANN model than the single best model found. This technique is useful when a high degree of uncertainty about the model structure and choice of inputs is present.

"So far as the laws of mathematics refer to reality, they are not certain.
And so far as they are certain, they do not refer to reality."

Albert Einstein

4 Model Selection Criteria and Model Averaging for Generalized Additive Neural Networks

Recently, there has been a growing interest in the modeling of nonlinear relationships and a variety of test procedures for detecting nonlinearities have been developed. When the aim of analysis is prediction, however, it is not sufficient to uncover nonlinearities. These nonlinearities need to be described through an adequate nonlinear model. Unfortunately, for many applications the appropriate theory does not guide the process of model building by suggesting the relevant input variables or the correct functional form of the model.

In this chapter, model selection strategies based on statistical concepts are discussed. A statistical perspective is especially important for models like Generalized Additive Neural Networks, because the primary reason for applying such models is the lack of knowledge regarding an adequate functional form of the underlying model. Anders & Korn (1999) provide a systematic comparison of statistical selection strategies for neural network models and consider the concepts of hypothesis testing, information criteria, and cross validation methods. The application of these three methods in neural networks is discussed and they came to the conclusion that statistical analysis should become an integral part of neural network modeling.

When is a model regarded as good? Such a model will fit the data set under consideration well. Also, when more variables are added to the model, the apparent fit becomes better. One goal of model selection is to balance the increase in fit against the increase in model complexity. Perhaps a better defining quality of a good model is its performance on future data sets collected from the same process. A model that fits one of the data sets representing the process well should fit any other data set representing the process just as well. It is possible for a too complicated model that fits the current data set well to poorly fit subsequent data sets. Furthermore, a model too simple may not fit any of the data sets well.

How should a model be selected? After a probabilistic model has been proposed for an experiment, data can be collected, leading to a set of competing candidate models. The data analyst would then select some appropriate model from this set, where there may be more than one definition of "appropriate". One way to decide on the most appropriate model is to use model selection criteria. Results from simulation studies are often utilized to compare model selection criteria. However, it is a daunting task to assess subtle differences between performance results. Furthermore, no single model selection criterion will always be better than another; certain criteria perform best for specific model types.

In Section 4.1 a review of the most prominent model selection criteria is presented. The two opposing views on model selection are discussed in Section 4.2. These paradigms are represented by efficient and consistent criteria. In Sections 4.3 and 4.4 the two most widely used in-sample model selection criteria, AIC and SBC, are considered. Bayesian Model Averaging, which accounts for uncertainty about the variables that must be included in a model as well as the model structure, is explained in Section 4.5. Due to certain difficulties, this methodology has not been widely adopted in practice. In Section 4.6 a technique is considered that approximates Bayesian Model Averaging for Generalized Additive Models. This technique is implemented in the AutoGANN system and illustrated in Section 4.7 with an example from the field of exploration.

4.1 Historical overview

Much of the research on model selection criteria in the past has been concerned with univariate or multiple regression models (Hurvich & Tsai, 1989). The first model selection criterion to be widely used is the adjusted R-squared, R^2_{adj}, which still appears in many regression texts today. It is commonly known that R^2 always increases whenever a variable is added to the model. As a result, it will always recommend additional complexity without regarding the relative contribution to model fit. The R^2_{adj} model selection criteria attempts to correct for this always-increasing property. Other research on model selection criteria that appeared in the late 1960s and early 1970s are most notably Akaike's FPE (Akaike, 1969) and Mallow's Cp (Mallows, 1973). Approaches based on information theory appeared in the 1970s, with the landmark Akaike Information Criterion (Akaike, 1974), based on the Kullback-Leibler discrepancy (Kullback & Leibler, 1951). Much research on information theory appeared in the late 1970s when the Bayesian Information Criterion (BIC) (Akaike, 1978), the Schwarz Information Criterion (SIC) (Schwarz, 1978) the Hannan and Quinn (HQ) (Hannan & Quinn, 1979), FPEα (Bhansali & Downham, 1977), and GM (Geweke & Meese, 1981) were proposed. In the late 1980s, Hurvich & Tsai (1989) adapted Sugiura's 1978 results to create an improved small-sample unbiased estimator of the Kullback-Leibler discrepancy, AICc. The latter has proved itself to be one of the best model selection criteria.

4.2 Two model selection paradigms

In 1980, the notion of asymptotic efficiency as a paradigm for selecting the most appropriate model appeared in the literature. The SIC, HQ, and GM model selection criteria became associated with the notion of consistency. These two philosophies of model selection are described subsequently.

4.2.1 Efficient criteria

An assumption commonly made in both regression and time series is that the generating or true model is of infinite dimension, or that the set of candidate models does not contain the true model. The goal is then to choose one model that best approximates the true model from a set of finite-dimensional candidate models. The candidate model closest to the true model is assumed to be the appropriate choice. In order to be evaluated, the term "closest" requires some well-defined distance or information measure. A model selection criterion that chooses the model with minimum mean squared error distribution in large samples is said to be asymptotically efficient (Shibata, 1980). Examples of the latter notion are FPE, AIC, AICc, and Cp. When researchers believe that the system they study is infinitely complicated, or that there is no way to measure all the important variables, they choose models based on efficiency. Much research has been devoted to improve ("correct") efficient criteria for small-samples. Perhaps the best known corrected version is AICc (Sugiura, 1978); (Hurvich & Tsai, 1989).

The predictive ability of a candidate model is sometimes its most important attribute. PRESS (Allen, 1974) is an early selection criterion that modeled mean squared prediction error. Akaike's FPE also selects models that make good predictions. Both PRESS and FPE are efficient and it is worth noting that prediction and asymptotic efficiency are related (Shibata, 1980).

4.2.2 Consistent criteria

Many researchers assume that the true model is included in the set of candidate models and consequently of finite dimension. Under this assumption the goal of model selection is to correctly identify the true model from the list of candidate models. A model selection criterion that identifies the correct model asymptotically with a probability of one is said to be consistent. Examples of consistent criteria are SIC, HQ, and GM. In this case the researcher believes that all variables can be measured, and furthermore, that enough is known about the physical system being studied to identify the list of all important variables. To many statisticians these are strong assumptions, but they may hold in fields like physics, where there are large bodies of theories to justify, assuming the existence of a true model that belongs to the set of candidate models.

Many of the classic consistent model selection criteria are derived from asymptotic arguments. Less work has been done to find improvements to consistent criteria than to efficient criteria. This is due in part to the fact that the consistent criteria do not estimate some distance function or discrepancy.

There is little agreement on which philosophy is better; efficiency or consistency. The choice is highly subjective and depends on the individual researcher's assessment of the complexity and measureability of the modeling problem.

Information-based criteria such as AIC and SIC, which penalize large models that often tend to overfit, are the most widely used in-sample model selection criterion. These two model selection criteria are discussed next.

4.3 Akaike Information Criterion

The Akaike Information Criterion (AIC) is the most popular model selection criterion for linear and nonlinear model identification (Anders & Korn, 1999); (McQuarrie & Tsai, 1998). The AIC aims at finding, among the set of models that are considered, the best approximating model to the unknown true data generating process and will select the model with the fewest parameters which fits the data well. The AIC has been applied to many types of models, ranging from multiple regression (Shibata, 1981), selection of the order in autoregressive time series (Shibata, 1976), to neural networks (Anders & Korn, 1999).

When a neural network is thought of as an approximation to an underlying model and analyzed as being misspecified in the sense of White (1981), the AIC does not apply, since it assumes the model structure to be the true one. Fortunately, a generalization to the AIC for misspecified models has been proposed by Murata, Yoshizawa & Amari (1994) and called the Network Information Criterion (NIC). Both the AIC and NIC criteria were derived under the assumption of asymptotic normality of the maximum likelihood estimators. In the latter case, the penalty term will lead to minimal expected prediction errors. As a consequence, the criteria are not theoretically justified for over-parameterized networks, e.g. networks with irrelevant hidden units, even if the neural network model includes the true structure. Anders & Korn (1999) recommend the use of AIC instead of NIC when information criteria are to be employed. More information and worked out examples on the AIC can be found in (Linhart & Zucchini, 1986) and (Burnham & Anderson, 2002).

The AIC is defined as

$$\text{AIC} = -2\log(\mathcal{L}(\hat{\theta}|y)) + 2K \tag{4.1}$$

where $\log(\mathcal{L}(\hat{\theta}|y))$ denotes the natural logarithm of the likelihood function of the parameter vector θ, given the data y and K the number of estimable parameters in the approximating model.

In the special case of least squares estimation with normally distributed errors and constant variance, the AIC can be expressed as

$$\text{AIC} = n\log(\hat{\sigma}^2) + 2K \tag{4.2}$$

where

$$\hat{\sigma}^2 = \frac{\sum \hat{\epsilon}_i^2}{n} \quad \text{(the MLE of } \sigma^2\text{)},$$

ϵ_i are the estimated residuals for a particular candidate model, and K is the total number of estimated regression parameters, including the intercept and σ^2. Burnham & Anderson (2002) provide complete details on the conceptual and mathematical statistics of the derivation of AIC.

The first term in (4.1) measures the goodness-of-fit of the model to the data and tends to decrease as more parameters are added to the approximating model, while the second term ($2K$) becomes larger as more parameters are added to the approximating model. The latter term sets a penalty for model overfitting. The optimal model is selected when AIC is minimized. Consequently, AIC is a reasonable criterion, which balances model fitting and model parsimony. On the downside, the AIC often leads to a model with an unnecessarily large number of parameters.

Usually, AIC is positive, but it can be shifted by any additive constant and a shift can sometimes result in negative values of AIC. The model with the smallest AIC value is estimated to be "closest" to the truth and is the best approximation for the information in the data, relative to the other models being considered. Perhaps none of the models in the set are good, but AIC attempts to choose the best approximating model of those in the candidate set. Consequently, every effort must be made to assure that the set of models is well founded.

As an example, let candidate models g_1, g_2, g_3, and g_4 have AIC values of 4,400, 4,560, 4,380, and 4,415 respectively. Model g_3 would be selected as the best single model as the basis for inference because g_3 has the smallest AIC value. These AIC values are on a relative (additive) scale, and a value of say 4,000 could be subtracted from each AIC value resulting in the following rescaled AIC values: 400, 560, 380, and 415. Such rescaling does not change the rank of the models, nor the pairwise differences in the AIC values. It is not the absolute size of the AIC value that is important, but the relative values, and particularly the differences between the AIC values.

4.3.1 AIC differences

Since AIC is on a relative scale, Burnham & Anderson (2002) recommend computing the AIC differences

$$\triangle_i = \text{AIC}_i - \min \text{AIC}$$

over all candidate models in the set. The $\triangle_i$ values are easy to interpret and allow a comparison and ranking of candidate models and are also useful in computing Akaike weights (discussed in Section 4.3.3). The larger $\triangle_i$ is, the less plausible is the fitted model as being the best model for samples such as the data the modeler has. As a rule of thumb, $\triangle_i \leq 2$ for models that have substantial support. These models should receive consideration in making inferences. Models that have a $\triangle_i$ value of about 4 to 7 have considerably less support, while models with values of $\triangle_i > 10$ have essentially no support, and might be omitted from further consideration. In the latter case, those models fail to explain substantial variation in the data.

The best of the four models in the above example, model g_3, has $\triangle_3 = 4,380 - 4,380 = 0$. Models g_1, g_2, and g_4 have $\triangle_i$ values of 20, 180, and 35 respectively. From these $\triangle_i$ values it can be seen that model g_3 is best, followed in order by g_1, g_4, and finally g_2.

The individual AIC value, by itself, is not interpretable. AIC is only comparative, relative to other AIC values in the model set; thus the differences $\triangle_i$ are very important and useful.

4.3.2 Likelihood of a model

It is possible to quantify the plausibility of each model as being the actual Kullback-Leibler best model (Burnham & Anderson, 2002). In other words, given the possible data, the functional form of each model and the parameter values, Kullback-Leibler information[1] can be computed for each model in the set and the model best approximating full reality determined. To achieve this, the concept of likelihood of the parameters given both the data and the model, $\mathcal{L}(\theta|D, M_i)$, is extended to the concept of the likelihood of the model, given the data, namely $\mathcal{L}(M_i|D)$. The likelihood of model M_i, given the data, can be computed for each model in the set:

$$\mathcal{L}(M_i|D) \propto e^{-\frac{1}{2}\triangle_i}. \tag{4.3}$$

Akaike recommends the above $e^{-\frac{1}{2}\triangle_i}$ for the relative likelihood of the model, given the MLEs of model parameters based on the same data.

[1]Kullback-Leibler information is the distance between models f (full reality or truth) and g (approximating model).

4.3.3 Akaike weights

To better interpret the relative likelihood of a model, given the data and the set of R models, Burnham & Anderson (2002) normalize the $\mathcal{L}(M_i|D)$ to be a set of positive *Akaike weights*, w_i, that add to 1:

$$w_i = \frac{e^{-\frac{1}{2}\triangle_i}}{\sum_{r=1}^{R} e^{-\frac{1}{2}\triangle_r}}. \tag{4.4}$$

The weights, w_i, depend on the entire set. If a model is added or dropped during a post hoc analysis, the w_i must be recomputed for all the models in the newly defined set.

A given Akaike weight, w_i, is considered as the weight of evidence in favor of model i being the actual Kullback-Leibler best model for the situation at hand given that one of the models must be the Kullback-Leibler best model of that set of R models.

4.4 Schwarz Information Criterion

The SBC arises from a Bayesian viewpoint with equal prior probability on each model and very vague priors on the parameters, given the model. It was assumed that the purpose of the SBC-selected model was often simple prediction; as opposed to scientific understanding of the process or system under study.

Several forms of the Schwarz Information Criterion (SIC or SBC) have been proposed in the literature. The generic SBC is defined as follows (Burnham & Anderson, 2002):

$$\text{SBC} = -2\text{log}(\mathcal{L}(\hat{\theta}|y)) + K\text{log}(n). \tag{4.5}$$

In the special case of the Gaussian error model, the SBC can be expressed as

$$\text{SBC} = n\text{log}(\hat{\sigma}^2) + K\text{log}(n) \tag{4.6}$$

where

$$\hat{\sigma}^2 = \frac{\sum \hat{\epsilon}_i^2}{n} \quad \text{(the MLE of } \sigma^2\text{)},$$

ϵ_i are the estimated residuals for a particular candidate model, and K is the total number of estimated regression parameters, including the intercept and σ^2.

Expression (4.5) is very similar to (4.1) in that the SBC[2] is also composed of two parts with the same first term. The difference is in the penalty term. If $K > 7$, the SBC imposes a greater

[2]The SBC is sometimes confusingly abbreviated as BIC. The Bayesian Information Criterion (BIC) has the penalty term $K + K\text{log}(n)$ rather than $K\text{log}(n)$.

penalty for model complexity than the AIC. Hence the use of the SBC for model selection would result in a model of which the number of parameters is no greater than that chosen by AIC.

Expression (4.5) was developed independently by Schwarz (1978) and Rissanen (1978). Rissanen (1980) showed that the SBC gives a consistent estimate of the order of an AR model. The SBC is a consistent model selector if the true data generating model belongs to the finite-parameter family under investigation. Haughton (1989) showed for exponential families that the SBC selects models that tend to underfit if the latter assumption does not hold.

The same formula (4.4), which gives the Akaike weights from $\triangle AIC$, is used with $\triangle BIC$ to give the posterior probabilities of models $M_1, M_2, \ldots, M_R$ (Burnham & Anderson, 2002).

Having discussed model selection criteria, the technique of Bayesian Model Averaging which accounts for model uncertainty is subsequently considered.

4.5 Bayesian Model Averaging

Conditioning on a single selected model ignores uncertainty about the variables that must be included in the model and the model structure. This leads to underestimation of the uncertainty about quantities of interest (Madigan & Raftery, 1994). This underestimation can be large, as was shown by Regal & Hook (1991) and Miller (1984) in the contingency tables and regression contexts respectively. As a consequence it can lead to decisions that are too risky (Hodges, 1987).

The standard Bayesian formalism (Leamer, 1978) provides in principle a panacea for all these difficulties. Let $\triangle$ be the quantity of interest, such as a parameter, a future observation, or the utility of a course of action. The posterior distribution of $\triangle$, given data D is

$$\mathrm{pr}(\triangle|D) = \sum_{k=1}^{K} \mathrm{pr}(\triangle|M_k, D)\mathrm{pr}(M_k|D). \tag{4.7}$$

Expression (4.7) is an average of the posterior distributions under each of the models, weighted by their posterior model probabilities. $M_1, \ldots, M_K$ are the models considered and

$$\mathrm{pr}(M_k|D) = \frac{\mathrm{pr}(D|M_k)\mathrm{pr}(M_k)}{\sum_{l=1}^{K} \mathrm{pr}(D|M_l)\mathrm{pr}(M_l)} \tag{4.8}$$

where

$$\mathrm{pr}(D|M_k) = \int \mathrm{pr}(D|\theta_k, M_k)\mathrm{pr}(\theta_k|M_k)d\theta_k \tag{4.9}$$

is the marginal likelihood of model M_k, θ_k is the parameter vector of M_k, $\mathrm{pr}(\theta_k|M_k)$ is the prior distribution of θ_k, $\mathrm{pr}(D|\theta_k, M_k)$ is the likelihood, and $\mathrm{pr}(M_k)$ is the prior probability of M_k.

Averaging over all the models provides better predictive ability than using any single model M_j, as measured by a logarithmic scoring rule:

$$-E\left[\log\left\{\sum_{k=1}^{K}\mathrm{pr}(\triangle|M_k,D)\mathrm{pr}(M_k|D)\right\}\right] \leq -E[\log\{\mathrm{pr}(\triangle|M_j,D)\}] \quad (j=1,\ldots,K) \tag{4.10}$$

where $\triangle$ is the observable to be predicted and the expectation is with respect to $\sum_{k=1}^{K}\mathrm{pr}(\triangle|M_k,D)\mathrm{pr}(M_k|D)$. This result follows from the nonnegativity of the Kullback-Leibler information divergence.

The Bayesian model averaging (BMA) approach in general has not been adopted in practice due to many challenges involved (Hoeting, Madigan, Raftery & Volinsky, 1999). First, the posterior model probabilities $\mathrm{pr}(M_k|D)$ are hard to compute, since they involve the very high dimensional integrals in Expression (4.9) which typically do not exist in closed form. Second, the number of models in the sum of (4.7) can be huge, rendering exhaustive summation infeasible. Third, specification of $\mathrm{pr}(M_k)$, the prior distribution over competing models, is challenging and has received little attention. A number of researchers have investigated the problem of managing the summation in (4.7) for a large number of models. Hoeting et al. (1999) discuss the historical developments of BMA, provide an additional description of the challenges of carrying out BMA, and describe some solutions to these problems for a variety of model classes. Hoeting (n.d.) describes more recent work in this area.

In the next section an approximation to Bayesian Model Averaging is presented that is implemented in the AutoGANN system.

4.6 Approximating Bayesian Model Averaging for Generalized Additive Models

Lee (1999) reviewed a number of methods for estimating the integral of (4.9) and concluded that the SBC may be the most reliable way of estimating this quantity. The SBC defined as

$$SBC_i = \log(\mathcal{L}(\hat{\theta}|y)) - K_i\log(n) \tag{4.11}$$

is used. A noninformative prior is utilized that puts equal mass on each model, i.e. $P(M_i) = P(M_j)$ for all i and j. The SBC approximation to (4.8) then becomes

$$\mathrm{pr}(M_i|D) \approx \frac{P(D|M_i)}{\sum_j P(D|M_j)} \approx \frac{e^{SBC_i}}{\sum_j e^{SBC_j}}. \tag{4.12}$$

The Bayesian approach automatically takes care of the balance between improving fit and not overfitting, because additional variables that do not sufficiently improve the fit will dilute the posterior, causing a lower posterior probability for the model. This approach, in addition to being conceptually straightforward, also has the advantage that it can be used simultaneously on both the problem of choosing a subset of explanatory variables, and the problem of choosing the architecture for the neural network.

When the SBC defined as in (4.5) is used, (4.12) becomes

$$\text{pr}(M_i|D) \approx \frac{P(D|M_i)}{\sum_j P(D|M_j)} \approx \frac{e^{-SBC_i}}{\sum_j e^{-SBC_j}}. \tag{4.13}$$

Approximations to BMA is implemented for a number of model classes (Hoeting et al., 1999); (Hoeting, n.d.), but not for Generalized Additive Models. Lee's approximation is defined for Multilayer Perceptron (MLP) neural networks. Since a Generalized Additive Neural Network has a separate MLP with a single hidden layer of h units for each input variable (Potts, 1999), his approximation is also applicable to GANNs. Furthermore, GANNs is the neural network implementation of GAMs, so Lee's approximation to BMA is also applicable to GAMs.

A Generalized Additive Model (GAM) has the form

$$g_0^{-1}(E(y)) = \beta_0 + f_1(x_1) + f_2(x_2) + \ldots + f_j(x_j). \tag{4.14}$$

Define

$$E(\Lambda_k) = g_0(\beta_0 + f_1(x_1) + f_2(x_2) + \ldots + f_j(x_j)) \tag{4.15}$$

where $\Lambda_k = I(\triangle|M_k, D)$ and $I(\triangle|M_k, D)$ is an indicator function[3]. From (4.7) and (4.13),

$$\text{pr}(\triangle|D) \approx \sum_{i=1}^{K} \text{pr}(\triangle|M_i, D) \frac{e^{-SBC_i}}{\sum_j e^{-SBC_j}}. \tag{4.16}$$

Since

$$\text{pr}(\triangle|M_k, D) = E(\Lambda_k) \tag{4.17}$$

it follows that

$$\text{pr}(\triangle|D) \approx \sum_{i=1}^{K} E(\Lambda_i) \frac{e^{-SBC_i}}{\sum_j e^{-SBC_j}}. \tag{4.18}$$

Expression (4.18) is used as the approximation to BMA in the AutoGANN system and in the next section an example utilizing BMA is discussed.

4.7 Exploration example

The SO_4 data set analyzed by Xiang (2001) contains the deposition of sulfate at 179 sites in the United States in 1990. Each record contains the latitude and longitude of the site as well as the

[3]An indicator function $I(A)$ has a value of 1 when event A occurs and a value of 0 when event A does not occur.

sulfate deposition at the site measured in gram per square meter (g/m^2). Sulfate is a naturally occurring compound consisting of sulfur (S), and oxygen (O_4) atoms. Two well known regions included in the data set are the New York- and Atlanta regions at the East Coast.

In Figure 4.1 a scatterplot of the data is given. The x and y axes correspond to the longitude and latitude of the site. The height of the arrows indicates the amount of sulfate found at the site.

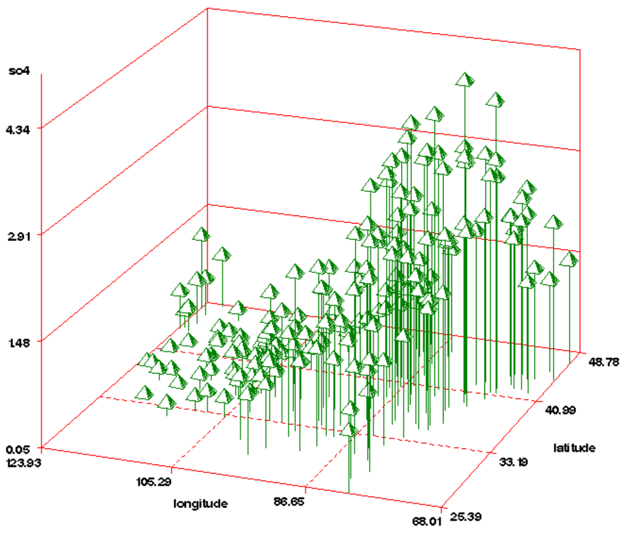


Figure 4.1: Scatterplot of SO_4 data set

Table 4.1 contains the model selection results after AutoGANN has terminated. The sub-architecture space was limited to $\{0, 1, 2, 3\}$ and the model selection criterion was set to the SBC defined for a least squares analysis. AutoGANN was executed on a Pentium 4, 3.2 GHz Intel processor with 4 GB of RAM in SAS® Enterprise Miner™ 5.1. The AutoGANN system evaluated the complete search space within 30 seconds. In Table 4.1 the first column (I) ranks the models in order of increasing SBC values found in the second column (SBC). The architecture identifier is given in the third column (ID) and the number of parameters for each model is presented in the fourth column (P). The best model found has a [2,2] architecture. The worst model in the search space, number 15, has a SBC value of -0.498 and a [3,0] architecture. The second best model has a [2,3] architecture.

I	SBC	ID	P	I	SBC	ID	P
1	-301.314	[2,2]	9	9	-210.281	[2,1]	6
2	-300.628	[2,3]	12	10	-195.319	[3,1]	9
3	-287.353	[3,2]	12	11	-188.249	[0,1]	2
4	-286.631	[3,3]	15	12	-184.948	[1,1]	3
5	-259.234	[0,2]	5	13	-11.485	[2,0]	5
6	-258.949	[1,2]	6	14	-9.041	[1,0]	2
7	-252.973	[0,3]	8	15	-0.498	[3,0]	8
8	-251.378	[1,3]	9				

Table 4.1: Model selection results

Figures 4.2 and 4.3 contain the partial residual plots for the *LONGITUDE* variable of the best and second best models found. The best model (Figure 4.2) has a less complex univariate function for *LONGITUDE* than the second best model (Figure 4.3). Since the partial residual plots of *LATITUDE* for the two models are nearly the same, they are omitted.

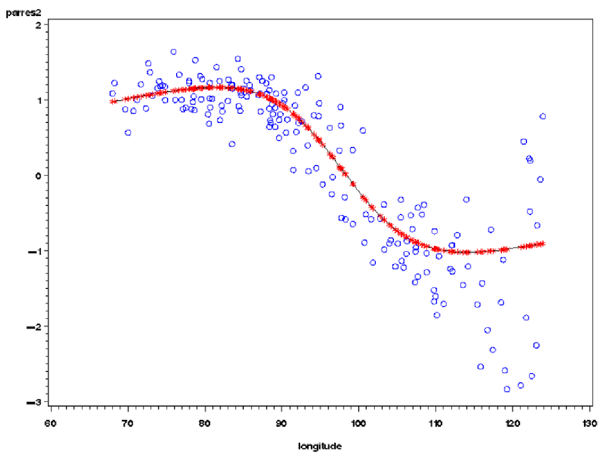


Figure 4.2: Partial residual plot for *LONGITUDE*

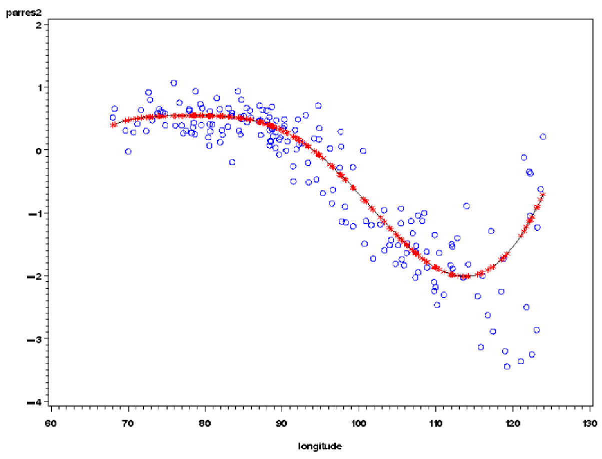


Figure 4.3: Partial residual plot for *LONGITUDE*

Figures 4.4 and 4.5 contain the surface plots of the best and second best models found. Again, it can be seen that the best model found (Figure 4.4) is less complex than the second best model (Figure 4.5) found.

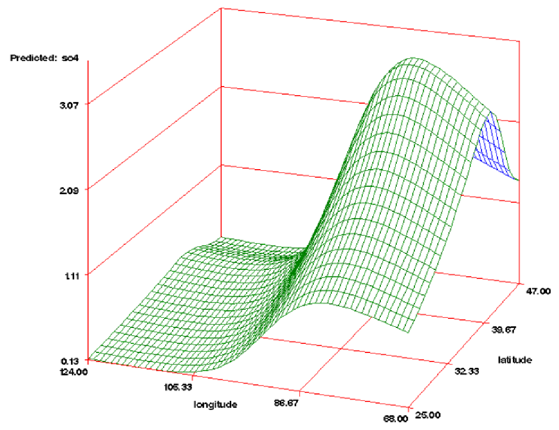


Figure 4.4: Surface plot for best model

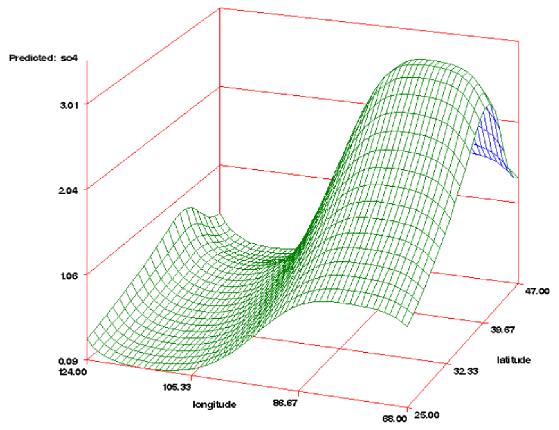


Figure 4.5: Surface plot for second best model

Next, model averaging is performed using the top 10 models found by AutoGANN. The results are given in Table 4.2. This table is the same as Table 4.1, except for one extra column (PR), the posterior probabilities of each model. The posterior probability of the best model is approximately 0.665 and the second best model is just under 0.335. For models 3 to 10 the posterior probabilities are very small and consequently these models do not contribute much to BMA. Since model averaging is performed using only the top 10 models, the posterior probabilities of all the other models are set to zero.

I	SBC	ID	P	PR	I	SBC	ID	P	PR
1	-301.314	[2,2]	9	0.66503	9	-210.281	[2,1]	6	1.94E-40
2	-300.628	[2,3]	12	0.33497	10	-195.319	[3,1]	9	6.16E-47
3	-287.353	[3,2]	12	5.75E-07	11	-188.249	[0,1]	2	0
4	-286.631	[3,3]	15	2.79E-07	12	-184.948	[1,1]	3	0
5	-259.234	[0,2]	5	3.53E-19	13	-11.485	[2,0]	5	0
6	-258.949	[1,2]	6	2.66E-19	14	-9.041	[1,0]	2	0
7	-252.973	[0,3]	8	6.74E-22	15	-0.498	[3,0]	8	0
8	-251.378	[1,3]	9	1.37E-22					

Table 4.2: Model averaging results

Figure 4.6 contains the partial residual plot of the combined model for *LONGITUDE*. This fitted spline is more complex than the one of Figure 4.2, but less complex than the fitted spline of Figure 4.3. Figure 4.7 contains the surface plot of the combined model. Again, this surface plot is more complex than the best model found (Figure 4.4), but less complex than the second best model (Figure 4.5).

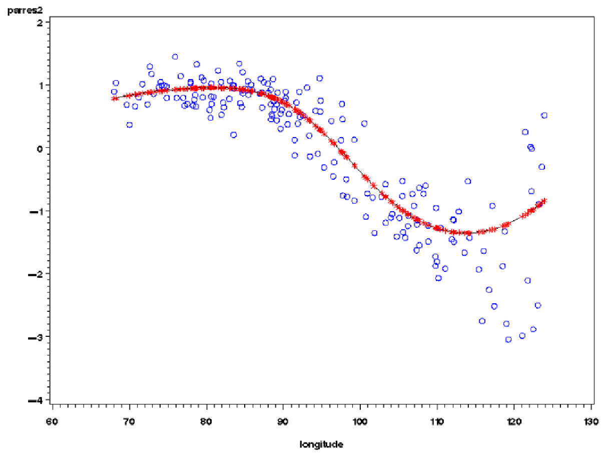


Figure 4.6: Partial residual plot for *LONGITUDE*

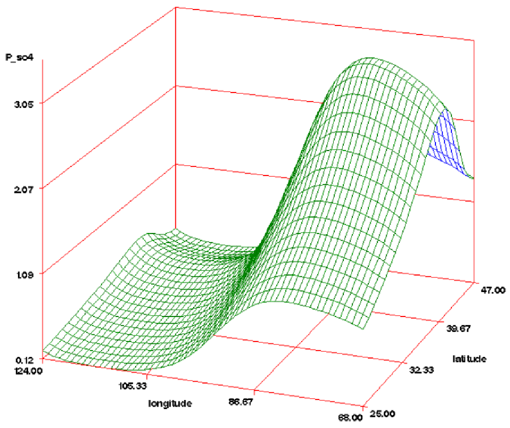


Figure 4.7: Surface plot for combined GANN model

Table 4.3 contains the average squared error (ASE) of the best model, second best model, and combined GANN model created with model averaging.

	ASE
Best model found	0.180484
Second best model found	0.180375
Model averaging	0.179154

Table 4.3: Model averaging results

From Table 4.3 it can be seen that the combined GANN model performed better than the best and second best GANN models found.

4.8 Conclusions

In this chapter the two most popular model selection criteria were discussed. The AIC and SBC can be used by the analyst to suggest the correct functional form of the Generalized Additive Neural Network. A statistical approach to model selection seems particularly relevant, since little is known about the correct functional form of the models. The AutoGANN program was supplemented with an approximation to the technique of Bayesian Model Averaging to provide better predictive ability than using any single model. Insight into this combined GANN model can be obtained by considering the partial residual plots, posterior probabilities, and fit statistics.

"The key to success in business is to know something that nobody else knows."

Aristotle Onassis

5 The Use of Generalized Additive Neural Networks in Credit Scoring

Credit scoring (Thomas, Edelman & Crook, 2002); (Mays, 2004); (Siddiqi, 2006); (McNab & Wynn, 2000); (SAS Institute Inc., n.d.) is a statistical method used to predict the probability that a loan applicant or existing borrower will default on the loan or become seriously delinquent (Mester, 1997). This method was introduced in the 1950s and is now widely used for consumer lending, especially credit cards, and mortgage lending. In business lending there has not been widespread application of the technique, but this is changing. The delay is caused in part by the fact that business loans typically differ substantially across borrowers which makes it harder to develop an accurate method of scoring. With the advent of new methodologies, increased computing power, and increased data availability, such scoring becomes possible. As a result, many banks are beginning to use scoring to evaluate small-business loan applications.

Credit scoring is a method that evaluates the credit risk of loan applications. Utilizing historical data and statistical techniques, credit scoring tries to isolate the effects of various applicant characteristics on defaults and delinquencies. The method creates a score that can be used by a bank to rank its loan applicants or borrowers in terms of risk. To build a scoring model, or scorecard, developers analyze historical data on the performance of previously made loans. This assists them in determining which borrower characteristics are useful in predicting whether the loan performed well or not. A well-designed model should assign a higher percentage of high scores to borrowers whose loans will perform well and a higher percentage of low scores to borrowers whose loans will not perform well. Unfortunately, no model is perfect, and some bad accounts will receive higher scores than some good accounts.

Information on the individuals applying for a loan is obtained from their loan applications and from credit bureaus. Factors such as the borrower's monthly income, outstanding debt, financial assets, how long the borrower has been in the same job, whether the borrower has defaulted or was ever delinquent on a previous loan, whether the borrower owns or rents a home, and the type of bank account the borrower has are all potential characteristics that may relate to loan performance and may end up being used in the scorecard.

A higher score in most scoring systems indicates a lower risk, and a lender sets a cutoff score based on the amount of risk he or she is willing to accept. Strictly adhering to the scorecard, the lender would approve applicants with scores above the cutoff and deny applicants with scores below. In practice, many lenders may take a closer look at applications near the cutoff before making the final credit decision.

A good scoring system cannot predict with certainty any individual loan's performance, but it should give a fairly accurate prediction of the likelihood that a loan applicant with certain characteristics will default. Developers need sufficient historical data to build a good scoring model. This data must reflect loan performance in periods of both good and bad economic conditions. New information on the credit performance of existing borrowers are frequently obtained and scorecards must be periodically rebuilt to keep the scoring models up to date.

In the field of credit scoring, logistic regression models (Kleinbaum, 1994); (Hosmer & Lemeshow, 1989) occupies a central position as it is relatively well understood and an explicit formula can be derived on which credit decisions may be based. Consequently, it is widely used in industry and has become the standard used by most companies. In the next section a comparison is made between the accuracy of a logistic regression model and that of a GANN model built on a home equity data set. The advantages of using a GANN model over a logistic regression model to build a scorecard is discussed. By utilizing the automated construction algorithm, the time it takes to build a scorecard may be drastically reduced without tampering with the traditional scorecard building methodology.

5.1 Credit risk example

Despite the fact that artificial neural networks may be more powerful than logistic regression, it is not widely used in credit scoring because, in general, it is perceived as a black box with respect to interpretation, and the absence of reasons why the neural network has reached it decisions may be unacceptable. The Equal Credit Opportunity Act of the United States of America (Anonymous,

2006) ensures that all consumers are given an equal chance to obtain credit. According to this act, a borrower has the right to know why the application for credit was rejected. The creditor must give a notice that gives the applicant specific reasons for the rejection. Acceptable reasons include: the borrower's income is too low, or the applicant is not employed long enough. Unacceptable reasons are: the borrower did not meet the minimum standards, or the applicant did not receive enough points on the credit-scoring system. Indefinite and vague reasons are illegal. As a result, obtaining regulatory approval for the use of neural networks to make credit decisions may be an important issue preventing the acceptance and wide use of neural networks in this environment. Fortunately, GANNs alleviates the difficulty mentioned above. Partial residual plots provide insight into the GANN model structure which makes the GANN a suitable model for building scorecards.

De Waal, Du Toit & De La Rey (2005) used a GANN to build a scorecard which is compared to a scorecard built using only logistic regression on a home equity data set analyzed by Wielenga, Lucas & Georges (1999). The aim is to predict whether an applicant will eventually default or be seriously delinquent on a loan that allows owners to borrow against the equity in their homes. The data set contains actual loan performance information for 5,960 recent home equity loans. Information that could identify the borrowers was removed.

The binary target variable (BAD) indicates whether an applicant eventually defaulted or was seriously delinquent and occurred in 1,189 cases (approximately 20%). There are 12 input variables: $REASON$, home improvement or debt consolidation, JOB, six occupational categories, $LOAN$, loan amount requested, $MORTDUE$, amount due to existing mortgage, $VALUE$, value of current property, $DEBTINC$, debt to income ratio, YOJ, years at present job, $DEROG$, number of derogatory reports, $CLNO$, number of trade lines, $DELINQ$, number of delinquent trade lines, $CLAGE$, age of oldest trade line in months, and $NINQ$, number of recent credit enquiries.

The data set consists of recent applicants granted credit and is split into training (67%) and validation (33%) data sets. There is a high percentage of missing values for $DEBTINC$ (20%) and consequently some method is needed to handle the missing values in the data set. The standard mean value imputation method for interval inputs was utilized[1]. Other variables with missing values are handled in a similar way. To illustrate the main difference between a logistic regression model and a GANN, the usual variable transformation step that accompanies the logistic regression is deliberately omitted. This transformation would handle nonlinear associations between inputs and the target.

5.1.1 Methodology

The logistic regression model is built using standard modeling practices. As the number of bad cases is substantially less than the number of good cases, over sampling might be considered. This

[1]Note that the newly created imputed variables have the prefix $IMP\_$.

is not done since the main motivation is to compare two modeling techniques on a high level and not to refine an existing model. The GANN is built using the same modeling practice just described.

A stepwise logistic regression is performed on the training data set with the missing values imputed. Standard significant levels of 0.05 are used. The variables *REASON* and *YOJ* are deleted from the model, and *JOB* is transformed into 5 variables using *N-1* dummy coding. The resulting model has 14 variables plus an intercept term.

5.1.2 Results

In Table 5.1 the events classification information indicates how well the model discriminates between bad and good applicants. From this table it can be seen that the logistic regression model has a 84% accuracy on the training set and a 83% accuracy on the validation set.

Data role	Target	False positive	False negative	True positive	True negative
Train	*BAD*	95	549	247	3101
Validate	*BAD*	43	282	111	1532

Table 5.1: Event Classification

The AutoGANN system is used to build a GANN in SAS® Enterprise Miner™. Table 5.2 shows the best GANN model found in 10 hours with the sub-architecture space set to {0,1,2,3,4,5,6,7,8,9} and the model selection criterion set to the SBC defined for a least squares regression. AutoGANN was executed on a Pentium 4, 3.2 GHz Intel processor with 4 GB of RAM in SAS® Enterprise Miner™ 5.1.

Variable	Sub-architecture	Sub-architecture description
LOAN	1	linear relationship with target
IMP_JOB	1	linear relationship with target
IMP_REASON	0	input removed
IMP_CLAGE	1	linear relationship with target
IMP_CLNO	1	linear relationship with target
IMP_DEBTINC	4	strong nonlinear relationship with target
IMP_DELINQ	1	linear relationship with target
IMP_DEROG	2	slight nonlinear relationship with target
IMP_MORTDUE	1	linear relationship with target
IMP_NINQ	1	linear relationship with target
IMP_VALUE	1	linear relationship with target
IMP_YOJ	0	input removed

Table 5.2: Generalized Additive Model results

The events classification information for this GANN is given in Table 5.3. The GANN model has a 89% accuracy on the training set and a 89% accuracy on the validation set.

Data role	Target	False positive	False negative	True positive	True negative
Train	*BAD*	165	292	504	3031
Validate	*BAD*	80	140	253	1495

Table 5.3: Event Classification

The best GANN model found has 23 degrees of freedom. Figures 5.1 to 5.9 contain the partial residual plots for the variables in the model. Partial residual plots for *IMP_REASON* and *IMP_YOJ* are not given, since they were deleted from the model. The partial residual plot for *IMP_JOB* is omitted because this variable contains six character based occupational classes, and partial residual plots can only be created for numerical variables.

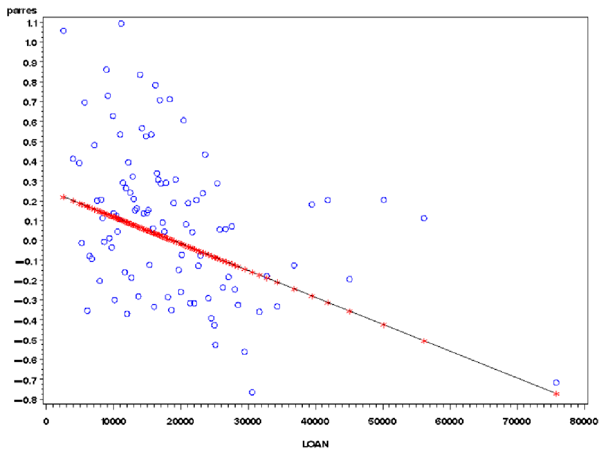


Figure 5.1: Partial residual plot for *LOAN*

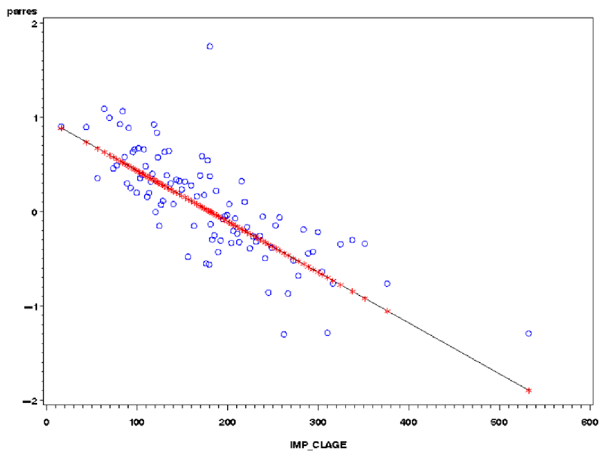


Figure 5.2: Partial residual plot for *IMP_CLAGE*

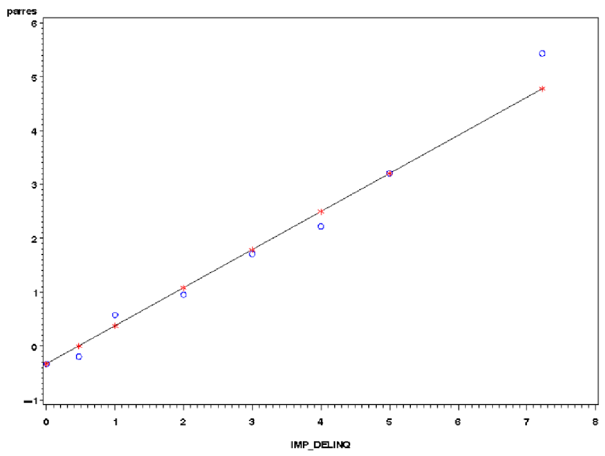


Figure 5.3: Partial residual plot for *IMP_DELINQ*

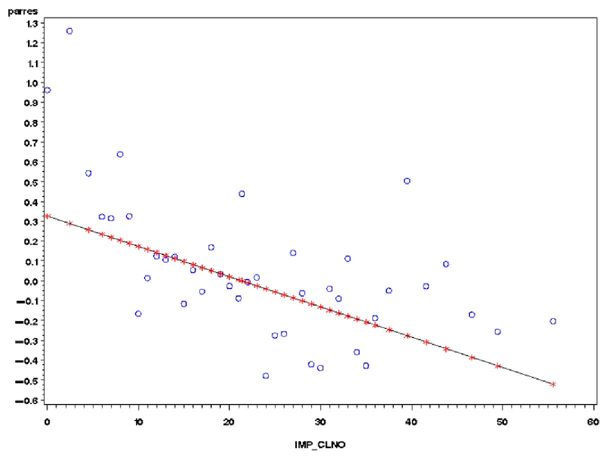


Figure 5.4: Partial residual plot for *IMP_CLNO*

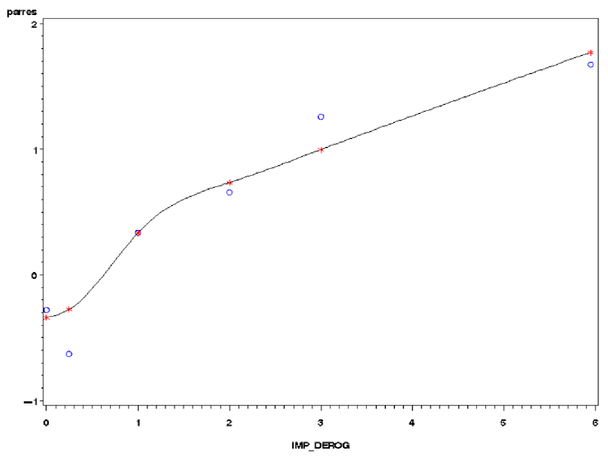


Figure 5.5: Partial residual plot for *IMP_DEROG*

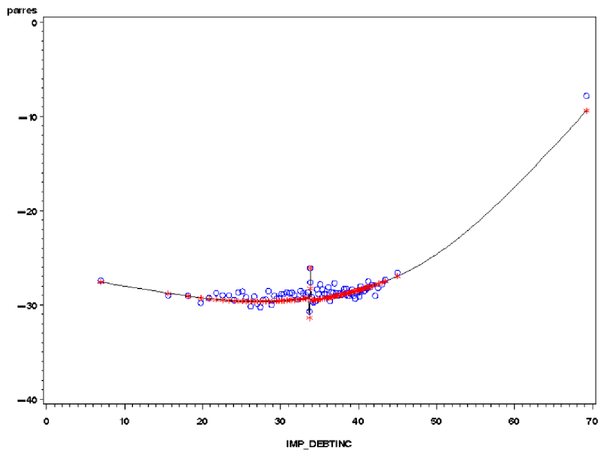


Figure 5.6: Partial residual plot for *IMP_DEBTINC*

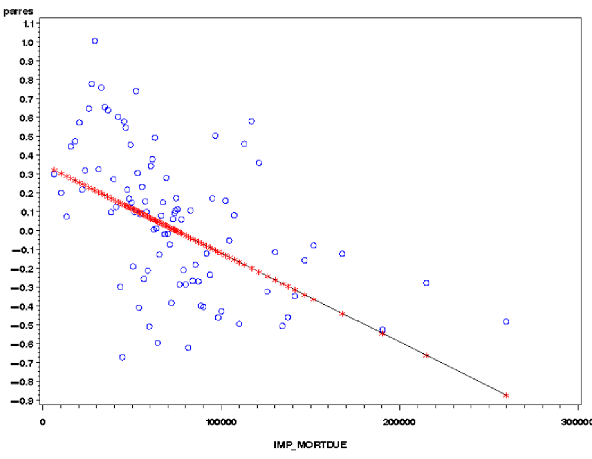


Figure 5.7: Partial residual plot for *IMP_MORTDUE*

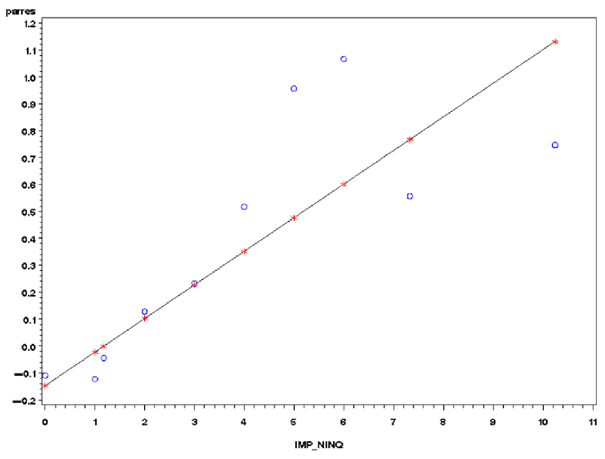


Figure 5.8: Partial residual plot for *IMP_NINQ*

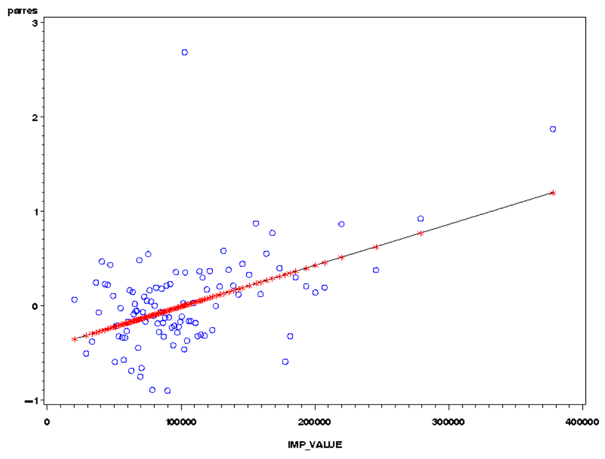


Figure 5.9: Partial residual plot for *IMP_VALUE*

It can be seen from Figures 5.5 and 5.6 that *IMP_DEROG* and *IMP_DEBTINC* exhibit nonlinear relationships with the target. An important aspect that needs consideration is whether the complexity of the nonlinear effects exhibited by the two variables is satisfactory given our domain knowledge of the given problem. The following insights can be gained from closer inspection of the partial residuals and fitted curve in Figure 5.5:

1. The curve is slightly nonlinear and it is worth investigating whether the added complexity of the nonlinear curve is really needed.

2. It is very unlikely that there can be 0.25 derogatory reports and thus one partial residual seems out of place. The value 0.25 is the result of substituting the missing values with the mean value. This partial residual influences the fitted curve and should be further investigated.

Inspection of Figure 5.6 provides the following insights:

1. At least three extreme points exist that dramatically influence the complexity of the curve. These points need further investigation as they may be regarded as extreme points.

2. The fitted curve is probably too complex and may need simplification so that the model will be able to generalize better on new data.

The remaining seven variables exhibit linear or slight nonlinear relationships with the target. These plots could be further investigated and the complexity of the univariate functions adjusted so as to achieve a reasonable fit.

The ROC Chart of Figure 5.10 reveals that the GANN (indicated by ImportModel) does significantly better than the logistic regression model. This result is to be expected as two nonlinear trends have been incorporated with the model. By considering Tables 5.1 and 5.3, it can be seen that the GANN is significantly better at discriminating between the bad and good applicants.

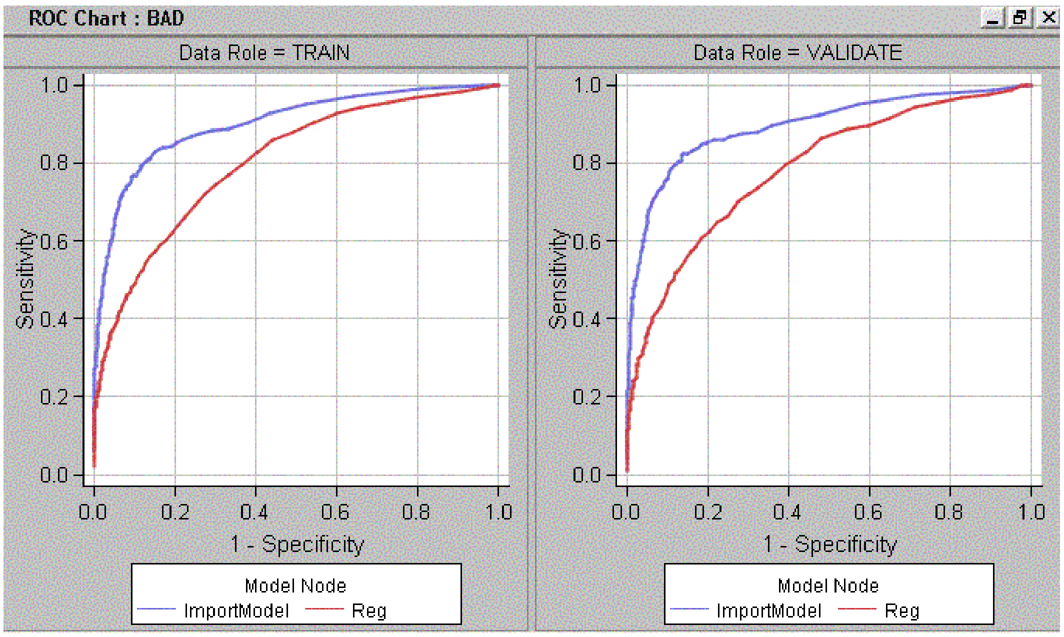


Figure 5.10: ROC Chart: BAD

The difference between the development of the two models is now further investigated to gain insight into the possibility of reducing the time it takes to build a scorecard.

5.1.3 Reducing scorecard development time with GANNs

From the description of the logistic regression model in a previous section, it is clear that at least one important step was omitted from the modeling process: variable transformation was not done. This step is usually done to model nonlinear trends in a more satisfactory way. From Figures 5.5 and 5.6 it is clear that the GANN incorporated these nonlinear trends into the computed model in a transparent way. Furthermore, it is not immediately clear what transformations must be done on the imputed variables to improve the logistic regression results so that similar results to that of the GANN can be obtained. At this stage the modeler has a choice: use the more accurate GANN or search for transformations to improve the logistic regression model. If the latter option is exercised, information gained from the GANN model may be used to improve the logistic regression model:

1. Variables deleted from the GANN may be investigated for deletion from the logistic regression model (interestingly, in this specific example the variables excluded from the logistic regression are exactly the same as those excluded from the GANN);

2. Variables having linear relationships with the target may be kept unchanged; and

3. Variables having nonlinear relationships may be candidates for transformation. Suitable transformations should be searched for to model the nonlinear relationships.

Quite some effort may be needed to arrive at the given GANN model presented earlier. The stated advantages may be negated by this effort as the time spent constructing the GANN model may have been spent on searching for suitable transformations for the logistic regression model giving similar results.

This argument may be valid if the GANN is constructed using an interactive approach relying on the inspection of partial residual plots. However, the AutoGANN system in Enterprise Miner™ (as described in Chapter 3) automates the construction of GANN models and very little or no user interaction is required while searching for the best GANN model. The AutoGANN system can therefore give substantial time savings while still resulting in an accurate model. The GANN presented in this chapter was constructed using the AutoGANN program, but with limited time allowed for the construction of the model. The sub-architecture space is also too complex (for example {0,1,2,3,4,5,6,7,8,9}) and could be simplified to for example {0,1,2,3} as in previous examples to speed up the model selection process. The model is therefore not optimal and may be further improved upon.

Given the benefits of constructing a GANN model, the process of building a scorecard with this type of model is discussed next.

5.1.4 Scorecard building with a GANN

It is possible to get access to the outputs of the univariate functions computed by the GANN model in Enterprise Miner™. These outputs may be regarded as the inputs to a classical scorecard building process that includes variable grouping and logistic regression on the weights of evidence. In building the scorecard, however, no variable transformations need to be considered as the GANN computed the transformations and it is now considered a pre-processing step. The building of the scorecard is completed using the outputs of the GANN univariate functions computed previously. In Table 5.4 the scorecard built the traditional way with only logistic regression and the new scorecard built by utilizing the GANN is given[2].

The two scorecards are very similar, except for *DEBTINC* that now has a wider range of scorecard points. This result is to be expected as *DEBTINC* was identified as the variable with the most complex nonlinear relationship with the target. The scorecard points have also been changed dramatically for this variable.

To compute the split values for the new scorecard, the original split values are run through the univariate functions giving "transformed" split values that are then used to build the scorecard. Note that the original variable groupings have been preserved and are transferred to the new scorecard, although the scorecard was actually built on the output of the GANN. Keeping the variable groupings unchanged is not optimal and groupings for variables exhibiting nonlinear relationships with the target should be reevaluated.

There is a change in scorecard points for the variable *LOAN* due to the fact that only four decimal places were allowed for split values (the "transformed" split values are on a completely different scale compared to original split values). Closer correlation of the groupings (and therefore scorecard points) can be achieved by allowing more precision for the "transformed" split values. In Tables 5.5 and 5.6 two extracts from the gains tables are given. The difference in accuracy between the two scorecards can be observed. From the 9th grouping that is highlighted, the following results can be inferred: with an approval rate of 77%, the bad rate for the improved scorecard is more than 2% lower than that of the original scorecard. The approval rate using the new scorecard can be increased from 77% to 82%, giving a similar bad rate to that of the old scorecard. These improvements obtained are significant and may provide huge monetary benefits to companies willing and able to exploit the power of GANNs.

5.2 Conclusions

As nonlinear models are better understood and become available in statistical and data mining systems, the move from linear models to nonlinear models is inevitable. There are, however,

[2]Note that Table 5.4 is composed of two separate tables to facilitate ease of comparison.

external constraints, such as the need for regulatory approval, that may hinder or temporarily delay the replacement of linear models by yet unproven, but potentially more powerful, nonlinear models such as Generalized Linear Models and neural networks.

Input	Attribute	Scorecard Points	Input	Attribute	Scorecard Points
Clage	.->120	40	Clage	.->120	38
Clage	120->180	50	Clage	120->180	49
Clage	180->240	57	Clage	180->240	58
Clage	240->	74	Clage	240->	74
Clno	.->9	34	Clno	.->9	34
Clno	9->13	60	Clno	9->13	60
Clno	13->18	59	Clno	13->18	59
Clno	18->24	55	Clno	18->24	55
Clno	24->28	54	Clno	24->28	54
Clno	28->.	49	Clno	28->.	49
Debtinc	.->30	96	Debtinc	.->30	106
Debtinc	30->35	29	Debtinc	30->35	95
Debtinc	35->40	82	Debtinc	35->40	84
Debtinc	40->.	54	Debtinc	40->.	22
Delinq	.->0.9	65	Delinq	.->0.9	65
Delinq	0.9->1.9	34	Delinq	0.9->1.9	34
Delinq	1.9->.	4	Delinq	1.9->.	4
Derog	.->1.9	55	Derog	.->1.9	55
Derog	1.9->.	15	Derog	1.9->.	17
Job	Sales	35	Job	Sales	36
Job	Self	44	Job	Self	44
Job	Mgr	49	Job	Mgr	49
Job	Other	50	Job	Other	50
Job	Profexe	58	Job	Profexe	58
Job	Office	62	Job	Office	61
Mortdue	.->49999	44	Mortdue	.->49999	41
Mortdue	49999->69999	52	Mortdue	49999->69999	53
Mortdue	69999->.	58	Mortdue	69999->.	60
Ninq	.->0.9	61	Ninq	.->0.9	60
Ninq	0.9->3.9	51	Ninq	0.9->3.9	51
Ninq	3.9->.	29	Ninq	3.9->.	32
Value	.->74332	55	Value	.->74332	55
Value	74332->107824	54	Value	74332->107824	55
Value	107824->.	49	Value	107824->.	48
Loan	.->5999	19	Loan	.->5999	27
Loan	5999->9999	50	Loan	5999->9999	51
Loan	9999->14999	53	Loan	9999->14999	50
Loan	14999->19999	55	Loan	14999->19999	60
Loan	19999->24999	61	Loan	19999->24999	55
Loan	24999->29999	59	Loan	24999->29999	65
Loan	29999->.	56	Loan	29999->.	55

Table 5.4: Scorecards Built with Logistic Regression and GANN

Score Range	Cumulative Count	Cumulative Number of Goods	Cumulative Number of Bads	Marginal Badrate	Cumulative Badrate	Approval Rate
626<=Score<641	14	14	0	0.00	0.00	0.71
612<=Score<626	101	99	2	2.30	1.98	5.13
597<=Score<612	240	236	4	1.44	1.67	12.20
583<=Score<597	472	465	7	1.29	1.48	23.98
569<=Score<583	651	635	16	5.03	2.46	33.08
554<=Score<569	850	813	37	10.55	4.35	43.19
540<=Score<554	1065	1007	58	9.77	5.45	54.12
525<=Score<540	1337	1244	93	12.87	6.96	67.94
511<=Score<525	**1516**	**1369**	**147**	**30.17**	**9.70**	**77.03**
497<=Score<511	1652	1465	187	29.41	11.32	83.94
482<=Score<497	1765	1525	240	46.90	13.60	89.68
468<=Score<482	1833	1550	283	63.24	15.44	93.14
453<=Score<468	1896	1569	327	69.84	17.25	96.34
439<=Score<453	1925	1575	350	79.31	18.18	97.82
425<=Score<439	1943	1575	368	100.00	18.94	98.73
410<=Score<425	1958	1575	383	100.00	19.56	99.49
396<=Score<410	1961	1575	386	100.00	19.68	99.64
381<=Score<396	1964	1575	389	100.00	19.81	99.80
367<=Score<381	1967	1575	392	100.00	19.93	99.95
353<=Score<367	1968	1575	393	100.00	19.97	100.00

Table 5.5: Gains Table for Logistic Regression

Score Range	Cumulative Count	Cumulative Number of Goods	Cumulative Number of Bads	Marginal Badrate	Cumulative Badrate	Approval Rate
632<=Score<648	23	22	1	4.35	4.35	1.17
617<=Score<632	129	126	3	1.89	2.33	6.55
602<=Score<617	316	310	6	1.60	1.90	16.06
586<=Score<602	619	605	14	2.64	2.26	31.45
571<=Score<586	891	861	30	5.88	3.37	45.27
556<=Score<571	1044	1003	41	7.19	3.93	53.05
540<=Score<556	1195	1140	55	9.27	4.60	60.72
525<=Score<540	1342	1268	74	12.93	5.51	68.19
510<=Score<525	**1483**	**1370**	**113**	**27.66**	**7.62**	**75.36**
495<=Score<510	1613	1452	161	36.92	9.98	81.96
479<=Score<495	1740	1523	217	44.09	12.47	88.41
464<=Score<479	1819	1545	274	72.15	15.06	92.43
449<=Score<464	1879	1563	316	70.00	16.82	95.48
433<=Score<449	1921	1573	348	76.19	18.12	97.61
418<=Score<433	1940	1575	365	89.47	18.81	98.58
403<=Score<418	1957	1575	382	100.00	19.52	99.44
387<=Score<403	1963	1575	388	100.00	19.77	99.75
372<=Score<387	1967	1575	392	100.00	19.93	99.95
342<=Score<357	1968	1575	393	100.00	19.97	100.00

Table 5.6: Gains Table for GANN

In this chapter, a way forward is sketched. The standard theory of scorecard building is not tampered with, but a pre-processing step is introduced to arrive at a more accurate scorecard that discriminates better between good and bad applicants. The pre-processing step exploits GANN models to achieve significant reductions in marginal and cumulative bad rates. Also, the time it takes to develop a scorecard may be reduced by utilizing the automated construction algorithm described in this thesis.

"The world is truly drowning in data, and much like we spent the last few thousand years figuring out how to navigate and build structures in the physical world, the explorers and builders of tomorrow will be figuring out our journey and navigation technology in the digital data universe."

Usama Fayyad

6 Conclusions

The convergence of computing and communication has created a society that feeds on information. Yet, most of the information is in a raw form, namely data. Technology makes it possible to capture and store vast quantities of data. The amount of data in the world is ever increasing. Inexpensive storage space makes it too easy to postpone decisions about what to do with all this data. If data is characterized as recorded facts, then information is the set of patterns, or expectations that underlie the data. Currently, there is a huge amount of information locked up in databases, information that is potentially important but has not yet been discovered, made explicit or taken advantage of. Finding trends, anomalies, and patterns in these data sets, and summarizing them with simple quantitative methods, is one of the main challenges of the information age - turning data into information and turning information into knowledge (Witten & Frank, 2005).

There has been substantial progress in data mining and machine learning. The synthesis of computing, statistics, machine learning, and information theory has created a solid science, with a firm mathematical base, and with very powerful tools. In this thesis some of the progress is presented. The basic theory of Generalized Additive Neural Networks is considered which leads to a powerful tool that automatically extracts models from data. GANNs are applied practically to search for patterns in data with positive results. Searching for patterns is not a particularly new endeavor. Statisticians, economists, forecasters, and communication engineers have long worked with the idea that patterns in data can be sought automatically, identified, validated, and used for prediction. What is new is the increase in opportunities for finding patterns in data. As the world grows in complexity, overwhelming us with the data it generates, data mining provides us with hope for elucidating the patterns that underlie it. This can lead to new insights and, in commercial settings, to competitive advantages.

In the next two sections the four contributions of this thesis and ideas for future research are discussed.

6.1 Contributions

The first contribution of this thesis is research on the interesting subject of Generalized Additive Neural Networks (Chapter 2). When the study commenced, only three sources existed, namely two papers (Sarle, 1994); (Potts, 1999) and a course on implementing neural networks in the SAS® programming language (Potts, 2000). This collection of work is extended (Chapters 2, 3, 4, and 5) and it is shown that GANNs (Section 2.3), the neural network implementation of Generalized Additive Models (Section 2.2), is powerful (Sections 2.4 and 2.5) and consequently worth studying. Also, a number of definitions are introduced to provide a common framework for discussing GANNs (Section 2.3). This new terminology on GANNs is utilized in the rest of the thesis. Two difficulties of the interactive construction methodology were identified which provided an impetus to automate the creation of GANN models. First, the interactive method is subjective, relying on a human to interpret partial residual plots. This examination can be influenced by opinion and result in suboptimal models. Second, considering a large number of partial residual plots is time consuming and can discourage the analyst in utilizing this methodology.

The second and main contribution of this thesis is the newly developed automated construction algorithm considered in Chapter 3. This novel methodology alleviates the data analyst from the painstaking process of examining a potentially large number of partial residual plots to create GANN models. Analysis of these plots forms the basis of the interactive construction algorithm (Potts, 1999). It is not a trivial exercise to automate human judgment required to interpret partial residual plots. Complex image processing and machine learning techniques could have been deployed, but the insight that a simple model selection criterion can be used exclusively to identify "good" models made the automated construction of GANNs possible. This new method allows the modeler to focus attention on interpretation of results obtained by the algorithm. The partial residual plot is not discarded as an ineffective tool, but used in a post-processing step to gain insight into relationships identified between input variables and the target. This step could guide the modeler to perform additional analysis on the data.

The main appealing qualities of the automated construction algorithm are its power, diversity of analysis, conciseness, and nontrivial execution. A greedy best-first search procedure (Section

3.1) identifies relatively good models in short time periods (Section 3.3). These models proved to have high predictive accuracy (Section 3.3) and are comparable to other models found in the literature (Section 3.4). With the automated algorithm, in-sample model selection (Section 3.2), cross-validation (Section 3.1), and feature selection (Section 3.4) can be performed. Individual models can be constructed and examined with a number of available fit statistics (Section 3.5.2) which make the interactive construction algorithm a special case of the automated methodology. Execution of the algorithm is nontrivial, but can be comprehended by the data analyst (Section 3.2).

AutoGANN, the implementation of the automated construction algorithm, has an elegant, simple, user-friendly, and intuitive user-interface (Section 3.5.2). This system relieves the user of making unnecessary decisions concerning the algorithm and provides a rich set of output results to guide the modeler. A linear model is constructed with default options. From this starting point the six parameters can be adjusted to perform a specific task. All models evaluated during the search are exported to enable other procedures to examine and utilize these models.

The third contribution of this thesis is the application of Bayesian Model Averaging to GAMs (Chapter 4). This technique accounts for uncertainty regarding variables that need to be included in the model and model structure. The development of methodologies to carry out model averaging is a rapidly growing area. Implementations of Bayesian Model Averaging could be found for a number of model classes (Hoeting, n.d.), but not for GAMs. The AutoGANN system is extended with an approximation to BMA (Section 4.6) which provides AutoGANN with better predictive ability than using any single GANN model. Model averaging is fully integrated within the system with the ability to examine partial residual plots and fit statistics of the combined model (Section 4.7), and to score a new data set.

The fourth contribution of this thesis is made to the field of credit scoring (Chapter 5). Although artificial neural networks may be more powerful than logistic regression, this type of model is not widely used in credit scoring. In general a neural network is perceived as a black box with respect to interpretation, and the absence of reasons why the neural network has reached its decisions may be unacceptable. The constrained architecture of a GANN lessens these difficulties. A GANN may be less powerful than a universal approximator (MLP), but insight into relationships between inputs and the target are provided by this type of model. A pre-processing step that uses GANN models is created to develop an improved scorecard that better discriminates between good and bad applicants (Section 5.1.4). This step does not influence the standard theory of scorecard building. Also, the AutoGANN system may reduce the time it takes to develop a scorecard (Section 5.1.3) by removing the responsibility of searching for suitable variable transformations from the data analyst.

Several directions for further research on GANNs are possible. These ideas are discussed in the next, final, section.

6.2 Future work

This thesis can be considered as the first in-depth research on GANNs and the automated construction of these models. Future work can be divided into theoretical and practical aspects.

On the theoretical side, the updated automated construction algorithm (Algorithm 3.2 in Section 3.2.3) could be extended to do feature selection before the algorithm commences or during execution of the algorithm. Effective feature selection procedures exist, for example stepwise regression techniques (Freund & Minton, 1979), that would ensure a good starting point (GANN model) for the search. Hopefully, with this added functionality, the automated construction algorithm can then continue to find "good" models much faster.

Research on the automated construction of GANN models utilizing Algorithms A and A* could be performed. Consider the evaluation function $f(n) = g(n) + h(n)$, where n is any state (GANN model) encountered in the search, $g(n)$ is the cost of n from the start state, and $h(n)$ is the heuristic estimate of the cost of going from n to the goal (best GANN model). If this evaluation function is used with the best-first search algorithm (Section 3.1), the result is called Algorithm A. When Algorithm A is used with an evaluation function in which $h(n)$ is less than or equal to the cost of the minimal path from n to the goal, the resulting search algorithm is called Algorithm A* (Luger, 2005). For the GANN system to utilize Algorithm A* effectively, a sound definition should be found for the $g(n)$ component of the evaluation function.

The response variable Y and the predictor variables $X_1, \ldots, X_p$ in regression analysis are often replaced by functions $\theta(Y)$ and $\phi_1(X_1), \ldots, \phi_p(X_p)$. Breiman & Friedman (1985) introduced the ACE procedure for estimating those functions θ^* and $\phi_1^*, \ldots, \phi_p^*$ that minimize $e^2 = E\{[\theta(Y) - \sum_{j=1}^{p} \phi_j(X_j)]^2\}/var[\theta(Y)]$, given only a sample $\{(y_k, x_{k1}, \ldots, x_{kp}), 1 \leq k \leq N\}$ and making minimal assumptions concerning the data distribution or form of the solution functions. A comparison between the automated construction algorithm and the ACE procedure (which both estimate the univariate functions) can be performed.

Generalized Additive Multi-Models (GAM-Ms) is a methodology based on the combination of prediction and classification procedures derived from different methods (Conversano, Siciliano & Mola, 2002); (Conversano & Mola, 2000); (Conversano, Siciliano & Mola, 2000*b*); (Conversano, Mola & Siciliano, 2000*a*). This methodology makes use of the backfitting algorithm (Section 2.2.3) for the calibration of the estimation provided by different kinds of models or smoothing functions. Generalized Additive Multi-mixture Models (GAM-MM) extend this approach to provide a class of models which combine a mixture of smoothers or models fitted to the data (Conversano, 2000). Research on the extension of the automated construction algorithm to include GAM-Ms and GAM-MMs could be conducted.

There are several ways to include interactions in the additive framework (Potts, 1999). New layers could be added for products of selected input variables. However, care must be taken since many interaction terms may degrade the interpretability of the model.

Decision trees can be considered an approximation to GAMs. Consequently, a detailed comparison between decision trees and GAMs could be performed.

Performance of AutoGANN on time series data (Brock & de Lima, 1996); (De Gooijer & Kumar, 1992) should be investigated. Other types of neural network architectures have been applied to time series data with mixed results (Faraway & Chatfield, 1998); (Dorffner, 1996); (Kaastra & Boyd, 1996); (Qi & Zhang, 2001).

On the practical side, the AutoGANN program is a prototype system with a number of possible enhancements. The basic data structure utilized by the system to keep track of GANN models is a list. Maintenance of the list becomes more time consuming as the number of models increase. Replacing the list with a binary tree (Collins, 1992) will make the system more effective.

At present, AutoGANN executes inside SAS® Enterprise Miner™. Normally in this environment, properties of the nodes must be set by hand before execution. Certain applications like time series analysis sometimes require the repeated execution of the modeling technique at each point in time. A batch execution facility in SAS® allows the AutoGANN system to be called repeatedly and configured at runtime by another program. Rewriting AutoGANN to operate in batch mode would allow more rigorous analysis.

SAS® released a new version of Enterprise Miner™ recently. This version has a number of improvements over the previous version which can benefit the AutoGANN system. Rewriting the program in Enterprise Miner™ 5.2 will result in a more user-friendly interface with more model fit statistics integrated with the system.

Finally, commercialization of the AutoGANN system is underway and at a sensitive stage at the time of writing this thesis. A number of organizations in South Africa have shown interest in the AutoGANN system. Currently the AutoGANN system has mainly been tested on relatively small experimental data sets. Commercializing AutoGANN entails more tests on large real world data sets.

Bibliography

Adriaans, P. & Zantinge, D. (1996), *Data Mining*, Addison-Wesley, England.

Akaike, H. (1969), 'Fitting autoregressive models for prediction', *Annals of the Institute of Statistical Mathematics* **21**, 243–247.

Akaike, H. (1974), 'A new look at the statistical model identification', *IEEE Transactions on Automatic Control* **AC-19**(6), 716–723.

Akaike, H. (1978), 'A bayesian analysis of the minimum aic procedure', *Annals of the Institute of Statistical Mathematics* **30, Part A**, 9–14.

Allen, D. M. (1974), 'The relationship between variable selection and data augmentation and a method for prediction', *Technometrics* **16**, 125–127.

Anders, U. & Korn, O. (1999), 'Model selection in neural networks', *Neural Networks* **12**, 309–323.

Anonymous (2006), Equal Credit Opportunity Act, http://www.cardreport.com/laws/ecoa.html, Date of access: 7 August 2006.

Armstrong, J. S. (1985), *Long-range forecasting: From crystal ball to computer*, John Wiley & Sons, New York.

Armstrong, J. S. & Collopy, F. (1992), 'Error measures for generalizing about forecasting methods: Empirical comparisons', *International Journal of Forecasting* **8**, 69–80.

Avramov, D. (2002), 'Stock return predictability and model uncertainty', *Journal of Financial Economics* **64**, 423–458.

Bakker, B. & Heskes, T. (2003), 'Clustering ensembles of neural network models', *Neural Networks* **16**, 261–269.

Bell, D., Walker, J., O'Connor, G., Orrel, J. & Tibshirani, R. J. (1989), 'Spinal deformation following multi-level thoracic and lumbar laminectomy in children', Submitted for publication.

Bellman, R. E. (1961), *Adaptive Control Processes: A Guided Tour*, Princeton University Press, Princeton, NJ.

Belsley, D. A., Kuh, E. & Welsch, R. E. (1980), *Regression Diagnostics: Identifying Influential Data and Sources of Collinearity*, Wiley Series In Probability and Mathematical Statistics, John Wiley & Sons, Inc., New York.

Berk, K. N. & Booth, D. E. (1995), 'Seeing a curve in multiple regression', *Technometrics* **37**(4), 385–398.

Berry, M. J. A. & Linoff, G. (1997), *Data Mining Techniques For Marketing, Sales, and Customer Support*, John Wiley & Sons, Inc., New York.

Bhansali, R. J. & Downham, D. Y. (1977), 'Some properties of the order of an autoregressive model selected by a generalization of akaike's epf criterion', *Biometrika* **64**(3), 547–551.

Bigus, J. P. (1996), *Data Mining with Neural Networks: Solving Business Problems - from Application Development to Decision Support*, McGraw-Hill, New York.

Blum, A. L. & Langey, P. (1997), 'Selection of relevant features and examples in machine learning', *Artificial Intelligence* **97**, 245–271.

Brachman, R. J. & Anand, T. (1996), The process of knowledge discovery in databases, *in* U. M. Fayyad, G. Piatetsky-Shapiro, P. Smyth & R. Uthurusamy, eds, 'Advances in Knowledge Discovery and Data Mining', AAAI Press / The MIT Press, Menlo Park, California, pp. 37–57.

Breiman, L. & Friedman, J. H. (1985), 'Estimating optimal transformations for multiple regression and correlation', *Journal of the American Statistical Association* **80**(391), 580–619.

Brock, W. A. & de Lima, P. J. F. (1996), Nonlinear time series, complexity theory, and finance, *in* G. S. Maddala & C. R. Rao, eds, 'Handbook of Statistics', Vol. 14 of *Statistical Methods in Finance*, North-Holland, Amsterdam, pp. 317–361.

Burnham, K. P. & Anderson, D. R. (2002), *Model Selection and Multimodel Inference: A Practical Information-Theoretic Approach*, 2nd edn, Springer, New York.

Cabena, P., Hadjinian, P., Stadler, R., Verhees, J. & Zanasi, A. (1997), *Discovering Data Mining: From Concept to Implementation*, Prentice Hall Ptr, London.

Cai, Z. & Tsai, C. (1999), 'Diagnostics for nonlinearity in generalized linear models', *Computational Statistics and Data Analysis* **29**, 445–469.

Carpenter, A. (2004), *Carpenter's Complete Guide to the SAS® Macro Language*, 2nd edn, SAS Institute Inc., Cary, NC.

Cawley, G. C. & Talbot, N. L. C. (2002), 'Improved sparse least-squares support vector machines', *Neurocomputing* **48**, 1025–1031.

Cheng, B. & Titterington, D. M. (1994), 'Neural networks: A review from a statistical perspective', *Statistical Science* **9**(1), 2–54.

Collins, W. J. (1992), *Data Structures: An Object-Oriented Approach*, Addison-Wesley Publishing Company, New York.

Conversano, C. (2000), Generalized additive multi-mixture models: some applications, Unpublished slides.

Conversano, C. & Mola, F. (2000), Semi-parametric models for data mining, *in* J. G. Bethlehem & P. G. M. van der Heijden, eds, 'Proceedings in Computational Statistics', Physica-Verlag, Heidelberg, pp. 241–246.

Conversano, C., Mola, F. & Siciliano, R. (2000*a*), Generalized additive multi-model for classification and prediction, *in* H. A. L. Kiers, J. P. Rasson, P. J. F. Groenen & M. Schader, eds, 'Data Analysis, Classification, and Related Methods', Springer, Berlin, pp. 205–210.

Conversano, C., Siciliano, R. & Mola, F. (2000*b*), Supervised classifier combination through generalized additive multi-model, *in* J. Kittler & F. Roli, eds, 'Proceedings of the First International Workshop on Multiple Classifier Systems', Lecture Notes in Computer Science, Springer, Berlin, pp. 167–176.

Conversano, C., Siciliano, R. & Mola, F. (2002), 'Generalized additive multi-mixture model for data mining', *Computational Statistics & Data Analysis* **38**, 487–500.

Cremers, K. J. M. (2002), 'Stock return predictability: A bayesian model selection perspective', *The Review of Financial Studies* **15**(4), 1223–1249.

DARPA (1988), *Defense Advanced Research Projects Agency: Neural Network Study*, AFCEA International Press, Fairfax, VA.

De Gooijer, J. G. & Kumar, K. (1992), 'Some recent developments in non-linear time series modelling, testing, and forecasting', *International Journal of Forecasting* **8**, 135–156.

De Waal, D. A. & Du Toit, J. V. (2004), Near-linear predictability of stock returns using financial and economic variables, Technical Report FABWI-N-RA/RKW:2004-134, North-West University, South Africa.

De Waal, D. A., Du Toit, J. V. & De La Rey, T. (2005), An investigation into the use of generalized additive neural networks in credit scoring, *in* 'Proceedings of Credit Scoring & Credit Control IX, Edinburgh, Scotland', Pollock Halls, University of Edinburgh, Scotland.

Dorffner, G. (1996), 'Neural networks for time series processing', *Neural Network World* **6**(4), 447–468.

Du Toit, J. V. & De Waal, D. A. (2003), Automated construction of generalized additive neural networks for predictive data mining, *in* J. Mende & I. Sanders, eds, 'Proceedings of the 33rd annual conference of the South African Computer Lecturer's Association'.

Du Toit, J. V. & De Waal, D. A. (2004), Classifying quantum physics particles generated by high energy collisions using generalized additive neural networks, Technical Report FABWI-N-RA/RKW:2004-140, North-West University, South Africa.

Džeroski, S. & Lavrač, N., eds (2001), *Relational Data Mining*, Springer, Berlin.

Eakins, S. G. & Stansell, S. R. (2003), 'Can value-based stock selection criteria yield superior risk-adjusted returns: an application of neural networks', *International Review of Financial Analysis* **12**(1), 83–97.

Ezekiel, M. (1924), 'A method for handling curvilinear correlation for any number of variables', *Journal of the American Statistical Association* **19**(148), 431–453.

Faraway, J. & Chatfield, C. (1998), 'Time series forecasting with neural networks: a comparative study using the airline data', *Applied Statistics* **47**(2), 231–250.

Faraway, J. J. (1992), 'On the cost of data analysis', *Journal of Computational and Graphical Statistics* **1**(3), 213–229.

Fayyad, U. M., Grinstein, G. G. & Wierse, A., eds (2002), *Information Visualization in Data Mining and Knowledge Discovery*, The Morgan Kaufmann Series in Data Management Systems, Morgan Kaufmann Publishers, San Francisco.

Fayyad, U. M., Piatetsky-Shapiro, G. & Smyth, P. (1996), From data mining to knowledge discovery: An overview, *in* U. M. Fayyad, G. Piatetsky-Shapiro, P. Smyth & R. Uthurusamy, eds, 'Advances in Knowledge Discovery and Data Mining', AAAI Press / The MIT Press, Menlo Park, California, pp. 1–34.

Freeman, J. A. & Skapura, D. M. (1992), *Neural Networks: Algorithms, Applications, and Programming Techniques*, Computation and Neural System Series, Addison-Wesley, Massachusetts.

Freund, R. J. & Minton, P. D. (1979), *Regression Methods: A Tool for Data Analysis*, Vol. 30 of *Statistics: Textbooks and Monographs*, Marcel Dekker, Inc., New York.

Friedman, J. H. (1997), Data mining and statistics: what's the connection?, http://www-stat.stanford.edu/∼jhf/ftp/dm-stat.ps, Date of access: 22 April 2006.

Friedman, J. H. & Silverman, B. W. (1989), 'Flexible parsimonious smoothing and additive modeling (with discussion)', *Technometrics* **31**, 3–39.

Friedman, J. H. & Stuetzle, W. (1981), 'Projection pursuit regression', *Journal of the American Statistical Association* **76**(376), 817–823.

Gallinari, P. & Cibas, T. (1999), 'Practical complexity control in multilayer perceptrons', *Signal Processing* **74**, 29–46.

Gately, E. (1996), *Neural Networks for Financial Forecasting*, Wiley Trader's Advantage Series, John Wiley & Sons, Inc., New York.

Geweke, J. & Meese, R. (1981), 'Estimating regression models of finite but unknown order', *International Economic Review* **22**, 55–70.

Groth, R. (1998), *Data Mining: A Hands-On Approach for Business Professionals*, The Data Warehousing Institute Series, Prentice Hall PTR.

Guyon, I. & Elisseeff, A. (2003), 'An introduction to variable and feature selection', *Journal of Machine Learning Research* **3**, 1157–1182.

Han, J. & Kamber, M. (2001), *Data Mining: Concepts and Techniques*, The Morgan Kaufmann Series in Data Management Systems, Morgan Kaufmann Publishers, San Francisco.

Hannan, E. J. & Quinn, B. G. (1979), 'The determination of the order of an autoregression', *Journal of the Royal Statistical Society* **41**(2), 190–195.

Harrison, D. & Rubinfeld, D. L. (1978), 'Hedonic housing prices and the demand for clean air', *Journal of Environmental Economics and Management* **5**, 81–102.

Hastie, T. J. & Tibshirani, R. J. (1986), 'Generalized additive models', *Statistical Science* **1**(3), 297–318.

Hastie, T. J. & Tibshirani, R. J. (1987), 'Generalized additive models: Some applications', *Journal of the American Statistical Association* **82**(398), 371–386.

Hastie, T. J. & Tibshirani, R. J. (1990), *Generalized Additive Models*, Vol. 43 of *Monographs on Statistics and Applied Probability*, Chapman and Hall, London.

Haughton, D. (1989), 'Size of the error in the choice of a model to fit data from an exponential family', *Sankhyā, Series A* **51**, 45–58.

Hawkins, D. (1989), 'Discussion of flexible parsimonious smoothing and additive modeling', *Technometrics* **31**, 3–39.

Hodges, J. S. (1987), 'Uncertainty, policy analysis and statistics', *Statistical Science* **2**(3), 259–275.

Hoeting, J. A. (n.d.), Methodology for Bayesian Model Averaging: An Update, Colorado State University, http://www.stat.colostate.edu/∼nsu/starmap/jah.ibcbma.pdf, Date of access: 22 April 2006.

Hoeting, J. A., Madigan, D., Raftery, A. E. & Volinsky, C. T. (1999), 'Bayesian model averaging: A tutorial', *Statistical Science* **14**(4), 382–417.

Hosmer, D. W. & Lemeshow, S. (1989), *Applied logistic regression*, Wiley series in probability and mathematical statistics, Wiley, New York.

Hurvich, C. M. & Tsai, C. (1989), 'Regression and time series model selection in small samples', *Biometrika* **76**(2), 297–307.

Jagric, T. (2003), 'A nonlinear approach to forecasting with leading economic indicators', *Studies in Nonlinear Dynamics & Econometrics, Article 4.*

Kaastra, I. & Boyd, M. (1996), 'Designing a neural network for forecasting financial and economic time series', *Neurocomputing* **10**, 215–236.

Kanas, A. & Yannopoulos, A. (2001), 'Comparing linear and nonlinear forecasts for stock returns', *International Review of Economics and Finance* **10**, 383–398.

Kaul, G. (1996), Predictable components in stock returns, *in* G. S. Maddala & C. R. Rao, eds, 'Handbook of Statistics', Vol. 14 of *Statistical Methods in Finance*, North-Holland, Amsterdam, pp. 269–296.

Kleinbaum, D. G. (1994), *Logistic regression: a self-learning text*, Springer series in statistics, Springer, New York.

Kleppner, D. & Jackiw, R. (2000), 'One hundred years of quantum physics', *Science, New Series* **289**(5481), 893–898.

Kullback, S. & Leibler, R. A. (1951), 'On information and sufficiency', *The Annals of Mathematical Statistics* **22**(1), 79–86.

Larsen, W. A. & McCleary, S. J. (1972), 'The use of partial residual plots in regression analysis', *Technometrics* **14**(3), 781–790.

Leamer, E. E. (1978), *Specification Searches: Ad Hoc Inference with Nonexperimental Data*, Wiley Series in Probability and Mathematical Statistics, John Wiley & Sons, New York.

Lee, H. K. H. (1999), Model Selection and Model Averaging for Neural Networks, PhD thesis, Department of Statistics, Carnegie Mellon University.

Lee, H. K. H. (2000), Model Selection for Neural Network Classification, http://citeseer.ist.psu.edu/lee00model.html, Date of access: 22 April 2006.

Linhart, H. & Zucchini, W. (1986), *Model Selection*, Wiley series in Probability and Mathematical Statistics, John Wiley & Sons, New York.

Luger, G. F. (2005), *Artificial Intelligence: Structures and Strategies for Complex Problem Solving*, 5th edn, Addison-Wesley, London.

Madigan, D. & Raftery, A. E. (1994), 'Model selection and accounting for model uncertainty in graphical models using occam's window', *Journal of the American Statistical Association* **89**(428), 1535–1546.

Mallows, C. L. (1973), 'Some comments on cp', *Technometrics* **15**, 661–675.

Mays, E. (2004), *Credit Scoring for Risk Managers: The Handbook for Lenders*, Thomson, Australia.

McCullagh, P. & Nelder, J. A. (1989), *Generalized Linear Models*, Vol. 37 of *Monographs on Statistics and Applied Probability*, 2nd edn, Chapman and Hall, London.

McNab, H. & Wynn, A. (2000), *Principles and practice of consumer credit risk management*, Financial World Publishing, Canterbury.

McQuarrie, A. D. R. & Tsai, C. (1998), *Regression and Time Series Model Selection*, World Scientific, Singapore.

Mester, L. J. (1997), 'What's the point of credit scoring?', *Business Review*.

Miller, A. J. (1984), 'Selection of subsets of regression variables', *Journal of the Royal Statistical Society, Series A* **147**(3), 389–425.

Minsky, M. & Papert, S. (1969), *Perceptrons*, MIT Press, Cambridge, MA.

Murata, N., Yoshizawa, S. & Amari, S. (1994), 'Network information criterion: Determining the number of hidden units for an artificial neural network model', *IEEE Transactions on Neural Networks* **5**(6), 865–872.

Olson, D. & Mossman, C. (2003), 'Neural network forecasts of canadian stock returns using accounting ratios', *International Journal of Forecasting* **19**(3), 453–465.

Pesaran, M. H. & Timmerman, A. (1995), 'Predictability of stock returns: Robustness and economic significance', *The Journal of Finance* **50**(4), 1201–1228.

Piatetsky-Shapiro, G. (2002), Who invented the term Data Mining?, http://www.kdnuggets.com, Date of access: 22 April 2006.

Potts, W. J. E. (1999), Generalized additive neural networks, *in* 'Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining', pp. 194–200.

Potts, W. J. E. (2000), *Neural Network Modeling Course Notes*, SAS Institute Inc., Cary, NC.

Pyle, D. (1999), *Data Preparation for Data Mining*, Morgan Kaufmann Publishers, Inc., San Francisco.

Qi, M. (1996), Financial applications of artificial neural networks, *in* G. S. Maddala & C. R. Rao, eds, 'Handbook of Statistics', Vol. 14 of *Statistical Methods in Finance*, North-Holland, Amsterdam, pp. 529–552.

Qi, M. (1999), 'Nonlinear predictability of stock returns using financial and economic variables', *Journal of Business & Economic Statistics* **17**(4), 419–429.

Qi, M. & Zhang, G. P. (2001), 'An investigation of model selection criteria for neural network time series forecasting', *European Journal of Operational Research* **132**, 666–680.

Racine, J. (2001), 'On the nonlinear predictability of stock returns using financial and economic variables', *Journal of Business & Economic Statistics* **19**(3), 380–382.

Regal, R. R. & Hook, E. B. (1991), 'The effects of model selection on confidence intervals for the size of a closed population', *Statistics In Medicine* **10**, 717–721.

Rich, E. & Knight, K. (1991), *Artificial Intelligence*, 2nd edn, McGraw-Hill, Inc., New York.

Ripley, B. D. (1996), *Pattern Recognition and Neural Networks*, Cambridge University Press, Cambridge, United Kingdom.

Rissanen, J. (1978), 'Modelling by shortest data description', *Automatica* **14**, 465–471.

Rissanen, J. (1980), Consistent order-estimates of autoregressive processes by shortest description of data, *in* E. O. Jacobs, M. Davis, M. Dempster, C. Harris & P. Parks, eds, 'Analysis and Optimization of Stochastic Systems', Academic Press, New York, pp. 451–461.

Rosenblatt, F. (1962), *Principles of Neurodynamics*, Spartan Books, Washington, DC.

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

Russel, S. & Norvig, P. (1995), *Artificial Intelligence: A Modern Approach*, Prentice-Hall International, Inc., London.

Sarle, W. S. (1994), Neural networks and statistical models, *in* 'Proceedings of the Ninteenth Annual SAS® Users Group International Conference'.

SAS Institute Inc. (2005), SAS® Enterprise Miner™ 5.1 Fact Sheet, http://www.sas.com/technologies/analytics/datamining/miner/factsheet.pdf, Date of access: 22 April 2006.

SAS Institute Inc. (2006), SAS® Corporate Statistics, http://www.sas.com, Date of access: 22 April 2006.

SAS Institute Inc. (n.d.), Building Consumer Credit Scoring Models with Enterprise Miner™, SAS Institute Inc., Cary, NC.

Schwarz, G. (1978), 'Estimating the dimension of a model', *The Annals of Statistics* **6**(2), 461–464.

Shibata, R. (1976), 'Selection of the order of an autoregressive model by akaike's information criterion', *Biometrika* **63**(1), 117–126.

Shibata, R. (1980), 'Asymptotically efficient selection of the order of the model for estimating parameters of a linear process', *The Annals of Statistics* **8**(1), 147–164.

Shibata, R. (1981), 'An optimal selection of regression variables', *Biometrika* **68**(1), 45–54.

Siddiqi, N. (2006), *Credit Risk Scorecards*, John Wiley & Sons, Inc., Canada.

Snyman, J. L. J. (1994), Model Selection and Estimation in Multiple Linear Regression, PhD thesis, Potchefstroom University for Christian Higher Education, South Africa.

Sockett, E. B., Daneman, D., Clarson, C. & Ehrich, R. M. (1987), 'Factors affecting and patterns of residual insulin secretion during the first year of type i (insulin dependent) diabetes mellitus in children', *Diabetologia* **30**, 453–459.

Stone, M. (1974), 'Cross-validatory choice and assessment of statistical predictions', *Journal of the Royal Statistical Society B* **36**(2), 111–147.

Sugiura, N. (1978), 'Further analysis of the data by akaike's information criterion and the finite corrections.', *Communications in Statistics - Theory and Methods* **7**, 13–26.

Thomas, L. C., Edelman, D. B. & Crook, J. N. (2002), *Credit scoring and its applications*, Society for Industrial and Applied Mathematics, Philadelphia.

Weiss, S. M. & Indurkhya, N. (1998), *Predictive Data Mining: a practical guide*, Morgan Kaufmann Publishers, Inc., San Francisco.

White, H. (1981), 'Consequences and detection of misspecified nonlinear regression models', *Journal of the American Statistical Association* **76**(374), 419–433.

Widrow, B. & Hoff, M. E. (1960), 'Adaptive switching circuits', *IRE WESCON Convention Record* **4**, 96–104.

Wielenga, D., Lucas, B. & Georges, J. (1999), *Enterprise Miner™: Applying Data Mining Techniques Course Notes*, SAS Institute Inc., Cary, NC.

Winston, P. H. (1992), *Artificial Intelligence*, 3rd edn, Addison-Wesley, Massachusetts.

Witten, I. H. & Frank, E. (2005), *Data Mining: Practical Machine Learning Tools and Techniques*, 2nd edn, Morgan Kaufmann Publishers, San Francisco.

Wood, S. N. (2006), *Generalized Additive Models: An introduction with R*, Texts in Statistical Science, Chapman & Hall/CRC, London.

Xiang, D. (2001), Fitting generalized additive models with the gam procedure, *in* 'SUGI26 Conference Proceedings', SAS Institute Inc., Cary, NC.

Zhang, G., Patuwo, B. E. & Hu, M. Y. (1998), 'Forecasting with artificial neural networks: The state of the art', *International Journal of Forecasting* **14**, 35–62.

Zucchini, W. (2000), 'An introduction to model selection', *Journal of Mathematical Psychology* **44**, 41–61.