SELF-ORGANIZING DATA MINING TECHNIQUES IN MODEL BASED SIMULATION GAMES FOR BUSINESS TRAINING AND EDUCATION

Authors: Mihail Motzev[1], Frank Lemke[2]

Abstract: Many problems exist in model building, identification, pattern recognition, approximation and extrapolation. To address these problems new techniques in data mining, like artificial neural networks, decision trees, genetic algorithms etc. have been developed and applied to analyze existing massive amounts of data and extract useful information. This paper presents a hybrid approach based on self-organizing data mining and concentrates on predictive models for business games and simulations. The results show that it is able to develop even complex models reliably and achieves lower overall error rates than state-of-the-art methods. The paper presents some of the results from international research done in Europe, Australia and most recently at Walla Walla University in College Place, Washington, USA.

Keywords: models; model-based simulations and business games; group method of data handling (GMDH); self-organizing data mining (SODM); multi-layer net of active neurons (MLNAN).

JEL: M42

1. INTRODUCTION

Different types of models form the basis for any decision. They support and assist decision makers and help them prepare better, more cost-effective alternatives for each decision. Models make it possible to identify the structure and the functions of the system of interest, which leads to a deeper and better understanding of the problem. Also, models can be analyzed more easily (in general), faster and economically than the original problem. Eventually, models help to predict what the system can expect in the future and make it possible to run simulation experiments with the system, as well as to apply "what-if" analysis.

Decision making in simulations and business games involves predictions at many stages of the process about different, unknown variables. Players receive a description of an imaginary business and an imaginary environment and make decisions – on price, advertising, production targets, etc. – about how their company should be run. These decisions are compared with a model, which determines how well they have fared and so forth.

[1] Mihail Motzev – Professor at Walla Walla University School of Business, College Place WA, USA, Mihail.Motzev@wallawalla.edu
[2] Frank Lemke –Owner and Managing Director at KnowledgeMiner Software, Berlin, Germany, Frank.Lemke@knowledgeminer.eu

A “model” in this sense is a set of rules which state that if a certain decision is taken then a certain result will follow. *Elgood (2005)* pointed out its importance, especially for the model-based games: “Models are also core features of other types of game, but in this type – named for them – they have special prominence. Players submit their decisions to the same model time after time: its presence is ubiquitous.”

Developing a good, accurate model for analysis and/or predictions is very important element in decision making. Unfortunately, there are many problems in model building, both in identification and estimation, such as overfitting, autocorrelation, multicollinearity, non-stationary data, small-sample size of observations, and so on. To address these problems new techniques like artificial neural networks, genetic algorithms, support vector machines, and others have been developed and applied to analyze existing massive amounts of data and extract useful information.

2. BUSINESS INTELLIGENCE AND ANALYTICS

In a 1958 article, IBM researcher *Hans Peter Luhn* used the term ***Business Intelligence (BI)*** and he defined intelligence as: "*the ability to apprehend the interrelationships of presented facts in such a way as to guide action towards a desired goal*" (*Luhn, 1958*). Later on, in 1989 *Howard Dresner* proposed *BI* as an umbrella term to describe "concepts and methods to improve business decision making by using fact-based support systems" (*cited in Power, 2007*). Today, *BI* usually refers to skills, processes, technologies, applications and practices used to support decision making.

In this paper we are using a general definition, provided by online computer dictionary[1]:

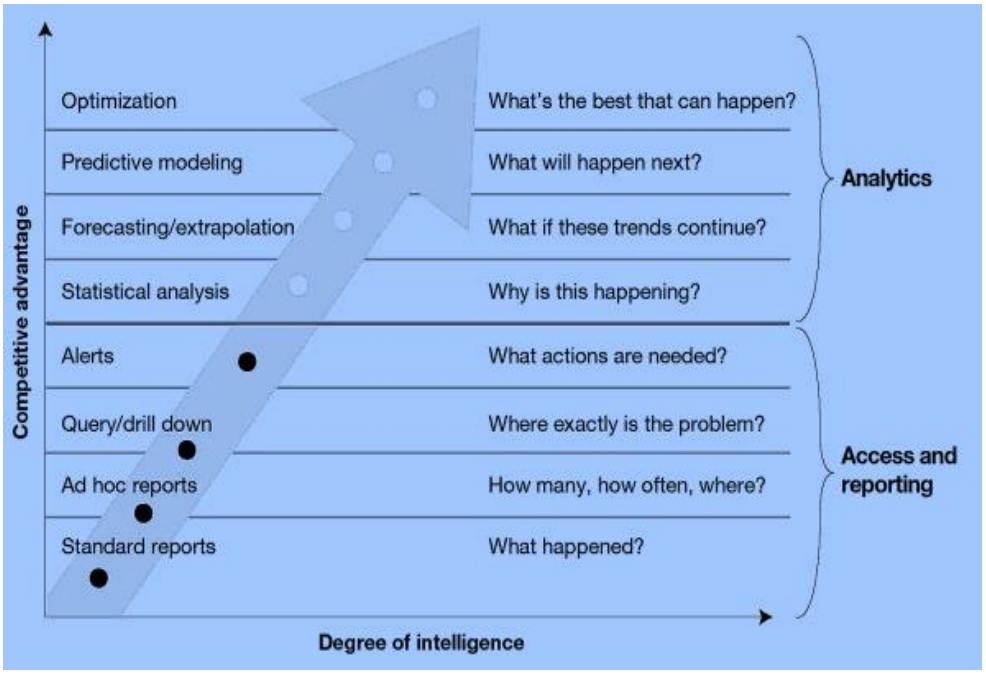
Business intelligence (BI) is a broad category of applications and technologies for gathering, storing, analyzing, and providing access to data to help enterprise users make better business decisions. BI applications include the activities of decision support systems, query and reporting, online analytical processing (OLAP), statistical analysis, forecasting, and data mining.

BI technologies provide historical, current, and predictive views of business operations. Common functions of Business Intelligence technologies are reporting, online analytical processing, analytics, data mining, business performance management, benchmarking, text mining, and predictive analytics (Fig. 1).

According to *Davenport and Harris (2007) BI* includes both data access and reporting, and analytics and in general, *Business Analytics (BA)* should be an element of *BI*, yet, there are different opinions and usually *BA* refers to the skills, technologies, applications and practices for continuous iterative exploration and investigation of past business performance to gain insight and drive business planning. In contrast with *business intelligence, business analytics* focuses on developing new insights and understanding of business performance whereas *business intelligence* traditionally focuses on using a consistent set of metrics to both measure past performance and guide business planning (see *Beller and Barnett, 2009*).

[1] http://searchdatamanagement.techtarget.com/sDefinition/0,,sid91_gci213571,00.html

Fig. 1 Business intelligence and analytics (Source: *Davenport and Harris, 2007*)



According to *Davenport and Harris (2007)* and others *(Shmueli et al., 2007)* business analytics can make extensive use of data, statistical and quantitative analysis, explanatory and predictive modeling, and fact-based management to drive decision making. Business intelligence is querying, reporting, OLAP, and "alerts", i.e. tools which can answer questions like: what happened, how many, how often, where, what actions are needed. Business analytics can answer more sophisticated questions like: why is this happening; what if these trends continue (i.e. what-if analysis & forecasting); what will happen next (i.e. prediction); what is the best that can happen (i.e. optimization) and so on (see Fig. 1.)

We can summarize that *BI* usually are related to *information systems* and *software applications* whilst *BA* is considered to be more oriented to *analytics*. No matter what is the most precise definition, both of them have applications in business forecasting and model building and here we are discussing only this function, often refer to as *predictive analytics.*

Predictive analytics encompasses a variety of techniques from statistics, data mining and game theory that analyze current and historical facts to make predictions about future events (see *Nyce, 2007*). In business, predictive models exploit patterns found in historical and transactional data to identify risks and opportunities. Models capture relationships among many factors to allow assessment of risk or potential associated with a particular set of conditions, guiding decision making for candidate transactions.

Generally, predictive analytics is used to mean *predictive modeling*, scoring of predictive models, and *forecasting*. However, people are increasingly using the term to describe related analytical disciplines, such as descriptive modeling and decision modeling or optimization. These disciplines also involve rigorous data analysis, and are widely used in business for segmentation and decision making but have different purposes and the statistical techniques underlying them vary.

The approaches and techniques used to conduct predictive analytics can broadly be grouped into *regression techniques* and *machine learning techniques*, based on *artificial neural networks*, *genetic algorithms* and other intelligent techniques.

3. PREDICTIVE ANALYTICS AND DATA MINING

The simplest definition of ***analytics*** is "the science of analysis". A simple and practical definition, however, would be how an entity (i.e., business) arrives at an optimal or realistic decision based on existing data. Business managers may choose to make decisions based on past experiences or rules of thumb, or there might be other qualitative aspects to decision making; but unless there are data involved in the process, it would not be considered analytics.

Common applications of analytics include the study of business data using statistical analysis in order to discover and understand historical patterns with an eye to predicting and improving business performance in the future. Also, some people use the term to denote the use of mathematics in business. Others hold that field of analytics includes the use of Operations Research, Statistics and Probability. However, it would be erroneous to limit the field of analytics to only statistics and mathematics.

As we mentioned already, *analytics* means "*the extensive use of data, statistical and quantitative analysis, explanatory and predictive models, and fact-based management to drive decisions and actions*" *(Davenport and Harris, 2007)*. Analytics are a subset of what has come to be called *business inte1ligenci* a set of technologies and processes that use data to understand and analyze business performance. Each of these approaches addresses a range of questions about an organization's business activities. The questions that analytics can answer represent the higher-value and more proactive end of this spectrum as it shown on Fig. 2.

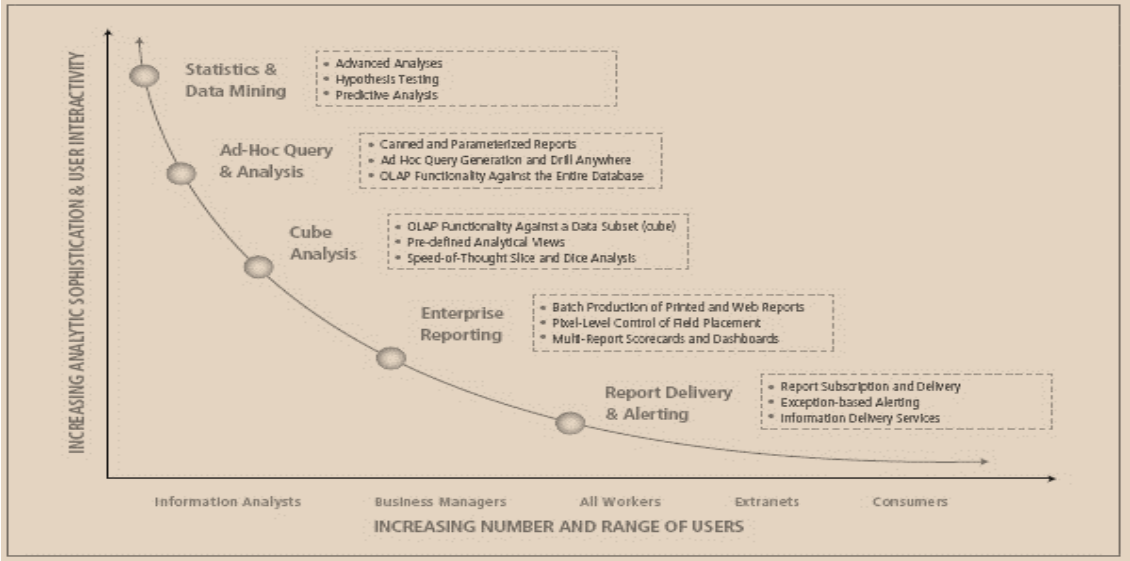
Analytics closely resembles *statistical analysis* and *data mining*, but tends to be based on modeling involving extensive computation. Some fields within the area of analytics are enterprise decision management, marketing analytics, predictive science, strategy science, credit risk analysis and fraud analytics. In this paper we will concentrate and discuss the most important techniques which could be used in model building for simulations and serious gaming.

It may sound curiously, but as it often happened in new concepts, definitions about data mining also vary. *Turban and Aronson (2001)* define ***Data mining*** as a "term used to describe knowledge discovery in databases. It includes tasks known as knowledge extraction, data archaeology, data exploration, data pattern processing, data dredging, and information harvesting". *Berry and Linoff (2000)* give more precise explanation:

Data mining is the process of exploration and analysis (by automatic or semi-automatic means) *of large quantities of data in order to discover meaningful patterns and rules.*

Second definition is better because it puts emphasis on large quantities of data (it's correct since data volumes continue to increase) and also on that the patterns and rules to be found ought to be meaningful.

Fig. 2 Business analytics (Source: *Davenport and Harris, 2007*)



The phrase ”by automatic or semi-automatic means” we put in brackets not because it is untrue (without automation it would be impossible to mine the huge quantities of data being generated today, as we’ll discuss it further on), but because as authors mentioned, we feel there has come to be too much focus on the automatic techniques and not enough on the exploration and analysis. In this article, we will discuss *data mining* in the context of this definition as *the process of extracting meaningful patterns from large amounts of data.*

Humans have been "manually" extracting patterns from data for centuries, but the increasing volume of data in modern times has called for more automated approaches. Current situation with the abundance of data, coupled with the need for powerful data analysis tools, has been described long time ago by *Finlay*: “*Without an efficient means of filtering and aggregating data, a manager could be* ***data rich yet information poor***” *(cited in Lucey, 1991)*.

Main characteristics are the fast-growing, tremendous amount of data (collected and stored in large and numerous data repositories), which has far exceeded our human ability for comprehension without powerful tools. Second, these data archives are seldom visited and, as a result, third, important decision are often made based not on the information-rich data stored in data repositories, but rather on a decision maker intuition. This happened because the decision maker does not have the tools to extract the valuable knowledge embedded in the vast amounts of data.

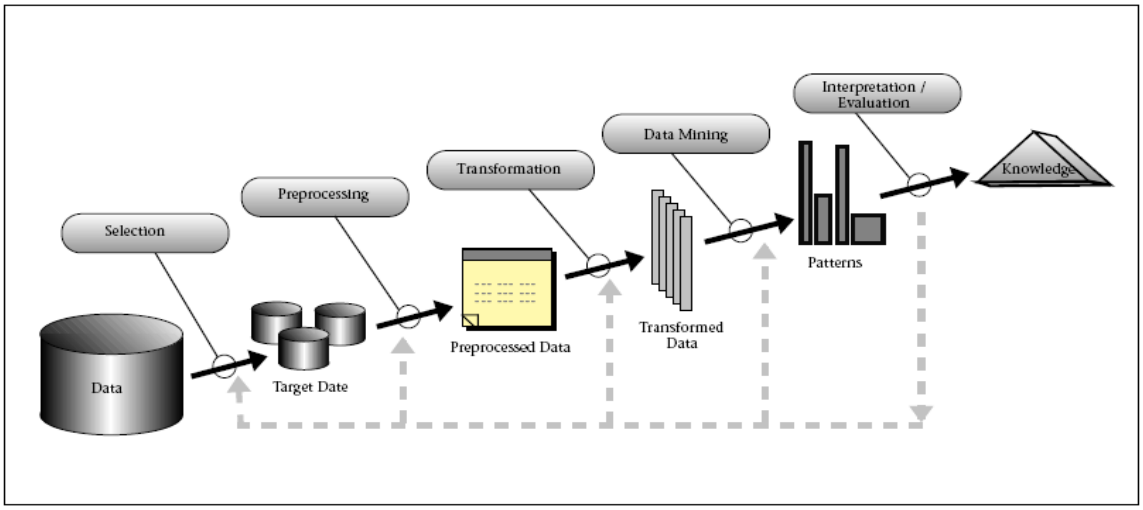
Early methods of identifying patterns in data include Bayes' theorem (1700s) and Regression analysis (1800s). The proliferation, ubiquity and increasing power of computer technology has increased data collection and storage. As data sets have grown in size and complexity, direct hands-on data analysis has increasingly been augmented with indirect, automatic data processing. This has been aided by other discoveries in computer science, such as neural networks, clustering, genetic algorithms (1950s), decision trees (1960s) support vector machines (1980s) and others.

The urgent need for a new generation of computational theories and tools to assist humans in extracting useful information (knowledge) from the rapidly growing volumes of digital data was pointed out yet in the 90ties of the last century (*Fayyad et al. 1996*). There are several reasons, which can be cited in support of the growing popularity of data mining today (see *Marakas, 2003*):

- The single greatest reason is the ever-increasing volume of data that require processing. The amount of data accumulated each day by businesses and organizations varies according to function and objective. A year 2000 report from the GTE research center suggests that scientific and academic organizations store approximately 1 terabyte of new data each day, even though the academic community is not the leading supplier of new data worldwide;
- Another reason for the growing popularity is an increasing awareness of the inadequacy of the human brain to process data, particularly in situations involving multi-factorial dependencies or correlations. Our biases formed by previous experience in data analysis often hold us hostage. As such, our objectivity in data analysis scenarios is often suspect;
- Finally, a third reason for the growing popularity of data mining is the increasing affordability of machine learning. An automated data mining system can operate at a much lower cost than an army of highly trained (and paid) professional statisticians. Although data mining does not entirely eliminate human participation in problem solving, it significantly simplifies the tasks and allows humans to better manage the process.

An increasingly common synonym for data mining techniques is knowledge data discovery (or ***Knowledge Discovery in Databases – KDD***). It should be noted that *KDD* applies to all activities and processes associated with discovering useful knowledge from aggregate data. Using a combination of techniques including statistical analysis, neural and fuzzy logic, multidimensional analysis, data visualization, and intelligent agents, *KDD* can discover highly useful and informative patterns within the data that can be used to develop predictive models of behavior or consequences in a wide variety of knowledge domains.

Fig. 3 Knowledge discovery in databases process (Source: *Fayyad et al., 1996, p.41*)



The need for distinction between the *KDD* process and the data-mining step (within the process – see Fig. 3) was mentioned in the very early articles in this area. According to *Fayyad et al. (1996, p.39)* "*Data mining is a step in the KDD process that consists of applying data analysis and discovery algorithms that produce a particular enumeration of patterns (or models) over the data."*

KDD refers to the overall process of discovering useful knowledge from data, and data mining refers to a particular step in this process. ***Data mining* (*DM*)** is the application of specific algorithms for extracting patterns from data. The additional steps in the *KDD* process, such as data preparation, data selection, data cleaning, incorporation of appropriate prior knowledge, and proper interpretation of the results of mining, are essential to ensure that useful knowledge is derived from the data. Blind application of data-mining methods, criticized as data dredging in the statistical literature, can be a dangerous activity, easily leading to the discovery of meaningless and invalid patterns.

Data mining involves the following activities in extracting meaningful new information from the data: *Classification, Estimation, Prediction, Affinity grouping or association rules, Clustering, Description and visualization* (see *Berry and Linoff, 2000*).

There is a discussion that there should not be a separate heading for prediction, because prediction can be thought of as classification or estimation. The difference is one of emphasis - when data mining is used to classify a phone line as primarily used for Internet access or a credit card transaction as fraudulent, we do not expect to be able to go back later to see if the classification was correct. The classification may be correct or incorrect, but the uncertainty is due only to incomplete knowledge. The computer is or is not used primarily for online business and the credit card transaction is or is not fraudulent. With enough effort, it is possible to check. Predictive task is different because data records are classified according to estimated future value. With prediction, the only way to check the accuracy of the classification is to wait and see.

As a matter of fact any of the techniques used for classification and estimation can be adapted for use in prediction by using training examples where the value of the variable to be predicted is already known, along with historical data for those examples. The historical data is used, to build a model that explains the current observed behavior. When this model is applied to current inputs, the result is a prediction of future behavior.

As we defined above, *Data Mining* is the process of examining large amounts of data in search of hidden patterns and predictive information (mostly in an automated manner), which allows organizations to make better decisions. Data Mining uses database technology, modeling techniques, statistical analysis and machine learning to find hidden patterns and make predictions which elude all but the most expert users and generate scoring or predictive models based on actual historical data.

How does data mining find information that business users and analysts did not already know? How does it find information about what is likely to happen next? In general, data mining platforms assist and automate the process of building and training highly sophisticated data mining models, and applying these models to larger datasets.

White paper published by *MicroStrategy (2005, pp. 162-173)* describes in details the data mining process and as an example, we'll suppose that a credit card company plans to develop a promotional campaign to recruit new customers:

1. Create a predictive model from a data sample – A sample dataset of customers who have responded to past promotional campaigns is extracted from the company data base. This sample contains customer characteristics and trends that potentially can be used to predict "responsiveness" like: Where do they live? What gender are they? What age range do they fall in? What is their income range? What is their marital status? What is their level of education? What are their past purchases? Have they responded to past campaign? What is their credit history?

Advanced statistical and mathematical techniques (like regression analysis and machine learning algorithms) are used to identify the significant characteristics and trends in predicting responsiveness, and a predictive model is created using these as inputs. Note that often only a small subset of all characteristics and trends in the sample dataset are generally used in the model.

2. Train the model against datasets with known results – The new predictive model is applied to additional data samples with known outcomes to validate whether the model is reasonably successful at predicting the known results. In this example it would be data based on historical campaign responses. This gives a good indication of the accuracy of the model. It can then be further trained using these samples to improve its accuracy.

3. Apply the model against a new dataset with an unknown outcome – Once the predictive model is validated against the known data, it is used for scoring, which is defined as the application of a data mining model to forecast an outcome. In the current example, the predictive model is applied to the new customer/prospect database to predict the likelihood of a customer responding to the marketing campaign and will generate a score for each customer that indicates his or her likelihood to respond. This score can be a simple binary result, such as Yes/No, or it could be a number indicating the propensity or confidence in that customer responding, say "97%." In both cases the end result will yield those customers that have a high probability of responding to the marketing promotion.

In addition to major techniques that are found in most comprehensive data mining tools (like *decision trees, neural networks* and *clustering*), another approach, which showed great promise (commonly referred to as *Group Method of Data Handling*) will be discussed and a unique hybrid technique in predictive modeling and business forecasting well be described in details in the next sections.

4. DATA MINING TECHNIQUES

Because data mining is viewed as a technical subject, people often get the notion that mastering data mining is largely a matter of studying advanced algorithms and learning the techniques for applying them. This technical understanding is actually only one small component of the mastery one seeks. It is, however, a very important one! Without at least a high-level understanding of the most important data mining algorithms, user will not be able to understand when one technique is called for and when another

would be more suitable. Users also need to understand what is going on inside a model in order to understand how best to prepare the model set used to build it and how to use various model parameters to improve results.

Usually, the level of understanding needed to make good use of data mining algorithms does not require detailed study of machine learning or statistics. In fact only a basic understanding of the principal algorithms is essential for anyone wishing to master the art of data mining. First of all, each user should distinguish between data mining techniques and the algorithms used to implement them. The term *technique* refers to a conceptual approach to extracting information from data. An *algorithm* is a step-by-step details of a particular way of implementing a technique. For example, automatic cluster detection is a technique that can be implemented using self-organizing maps, simple k-means, Gaussian k-means, and a number of other algorithms.

Data mining is a method of searching data for unexpected patterns or relationships using a variety of tools and algorithms. *Fayyad et al. (1996)* identified six tasks as follow:

- *classification*: learning a function that maps (classifies) a data item into one of several predefined classes;
- *regression*: learning a function that maps a data item into a real-valued prediction variable;
- *clustering*: identifying a finite set of categories or clusters to describe the data;
- *summarization*: finding a compact description for a subset of data;
- *dependency modeling*: finding a model that describes significant dependencies between variables;
- *change and deviation detection*: discovering the most significant changes in the data from previously measured or normative values.

Data Mining uses database technologyes, modeling techniques, statistical analysis and machine learning to find hidden patterns of relationships, generate forecasting models based on actual historical data and make predictions. The purpose of data mining platforms is to assist and automate the process of building and training highly sophisticated models, and applying these models to make predictions, perform what-if analysis and simulations, which support and help making better decisions.

Each data mining technique has many different algorithms and implementations – in fact, almost every tool has some nuance that makes its implementation a little different from the next tool. In spite of this, just as it is possible to learn how to drive a car, and generalize it to any car, it is possible to learn how to use neural networks or decisions trees, and generalize to any tool. It is also true that data mining has a broad reach, but it is not possible to cover every algorithm used in the business world. And, in addition as many authors mention it, the minor variations have much less effect on data mining results than other issues, such as preparing the data and building the right models.

According to *Fayyad et al. (1996)* the major groups of techniques that are found in most comprehensive data mining tools are ***clustering***, ***decision trees*** and ***artificial neural networks (ANNs)***.

4.1. Clustering

There are many mathematical approaches to finding clusters in data (Fig. 4), and whole text books are devoted to the subject. Some methods, called divisive methods, start by considering all records to be part of one big cluster. That cluster is then split into two or more smaller clusters, which are themselves split until eventually each record has a cluster all to itself. At each step in the process, some measure of the value of the splits is recorded so that the best set of clusters can be chosen at the end. Other methods, called agglomerative methods, start with each record occupying a separate cluster, and iteratively combine clusters until there is one big one containing all the records. There are also self-organizing maps, a specialized form of neural network that can be used for cluster detection.

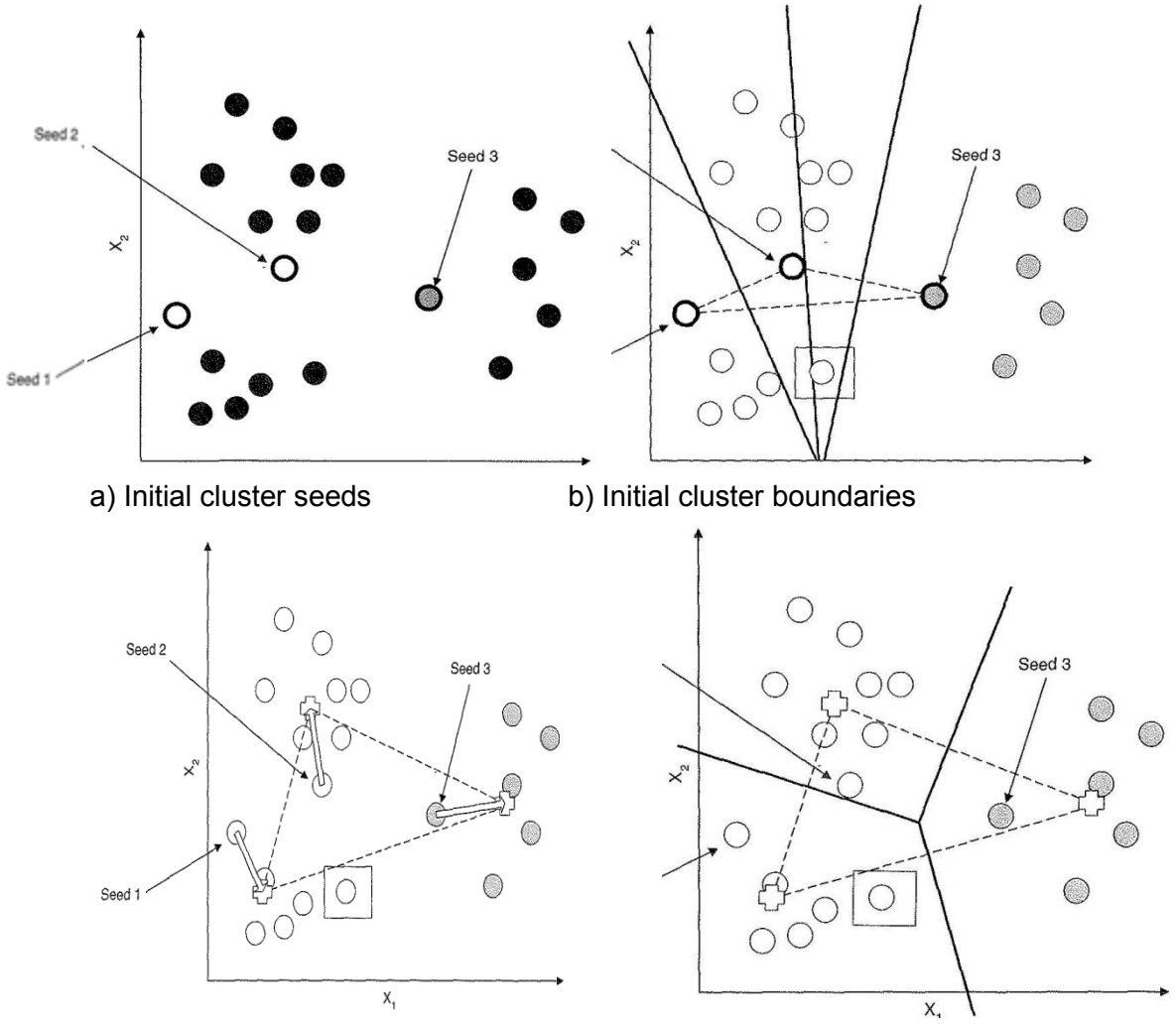
For example, *K-means* is a clustering algorithm, which is available in a wide variety of commercial DM tools and is more easily explained than most. It works best when the input data is primarily numeric. Consider an analysis of supermarket shopping behavior based on loyalty card data. Simply take each customer and create a field for the total amount purchased in various departments in the supermarket over the course of some period of time-diary, meat, cereal, fresh produce, and so on. This data is all numeric, so *K-means* clustering can work with it quite easily and the algorithm will find clusters of customers with similar purchasing patterns.

This algorithm divides a data set into a predetermined number of clusters. That number is the “k” in the phrase *k means*. A mean is, of course, just what a statistician calls an average. In this case it refers to the average location of all of the members of a particular cluster. But what does it mean to say that cluster members have a location when they are records from a database?

The answer comes from geometry. To form clusters, each record is mapped to a point in “record space.” The space has as many dimensions as there are fields in the records. The value of each field is interpreted as a distance from the origin along the corresponding axis of the space.

In order for this geometric interpretation to be useful, the fields must all be converted into numbers and the numbers must be normalized so that a change in one dimension is comparable to a change in another. Records are assigned to clusters through an iterative process (see Fig. 4) that starts with clusters centered at essentially random locations in the record space and moves the cluster *centroids* (another name for the cluster means) around until each one is actually at the center of some cluster of records. This process is best illustrated through diagrams. For ease of drawing, we show the process in two dimensions, but bear in mind that in practice the record space will have many more dimensions, because there will be a different dimension for each field in the records. And we don’t need to worry about drawing the clusters, because there are other ways of understanding them.

Fig. 4 Automatic Cluster Detection



a) Initial cluster seeds b) Initial cluster boundaries

c) After one iteration d) New cluster assignments

Because automatic cluster detection is an undirected technique, it can be applied without prior knowledge of the structure to be discovered. On the other hand, since the clusters that are automatically detected have no natural interpretation other than that, for a given mapping of records to a geometric coordinate system, some records are close to one another, it can be hard to put the results to practical use.

By choosing different distance measures, automatic clustering can be applied to almost any kind of data. For instance, there are measures of the distance between two passages of text that can be used to cluster newspaper articles into subject groups. Most clustering software, however, uses the Euclidean distance formula we all once learned in school – the one where you take the square root of the sum of the squares of the displacements along each axis. That means that non-numeric variables must be transformed and scaled before they can take part in the clustering. Depending on how these transformations are done, the categorical variables may dominate the clustering or be completely ignored.

A strength of automatic cluster detection is that it is an undirected knowledge discovery technique. Every strength has a corresponding weakness. When you don’t know

what you are looking for, you may not recognize it when you find it! The clusters generated by the automated clustering algorithms (whether k-means or any other algorithm) are not guaranteed to have any practical value. Once the clusters have been created, it is up to user to interpret them. There are several approaches to understanding clusters. Three that we use frequently are:

- Building a decision tree with the cluster label as the target variable and using it to derive rules explaining how to assign new records to the correct cluster.
- Using visualization to see how the clusters are affected by changes in the input variables.
- Examining the differences in the distributions of variables from cluster to cluster, one variable at a time.

When to use Cluster Detection – we should use cluster detection when we suspect that there are natural groupings that may represent groups of customers or products that have a lot in common with each other. These may turn out to be naturally occurring customer segments for which customized marketing approaches are justified. More generally, clustering is often useful when there are many competing patterns in the *data* making it hard to spot any single pattern. Creating clusters of similar records reduces the complexity within clusters so that other data mining techniques are more likely to succeed.

4.2. Decision Trees

Decision trees are a wonderfully versatile tool for data mining. Decision trees seem to come in nearly as many varieties as actual trees in a tropical rain forest. And, like deciduous and coniferous trees, there are two main types of decision trees:

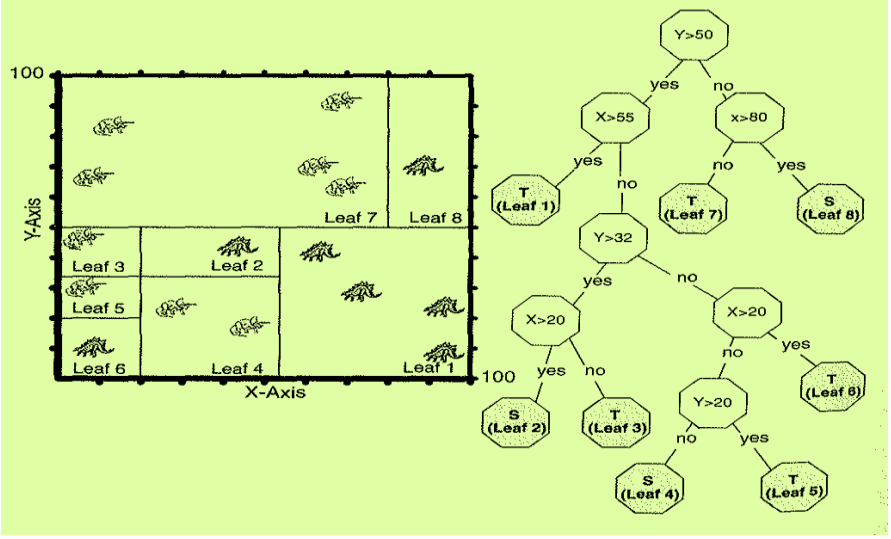
Classification trees label records and assign them to the proper class. They can also provide the confidence that the classification is correct. In this case, the classification tree reports the class probability, which is the confidence that a record is in a given class.

Regression trees estimate the value of a target variable that takes on numeric values. So, a regression tree might calculate the amount that a donor will contribute or the expected size of claims made by an insured person.

All of these trees have the same structure. When a tree model is applied to data, each record flows through the tree along a path determined by a series of tests such as “is field 3 greater than 27?” until the record reaches a leaf or terminal node of the tree. There it is given a class label based on the class of the records that reached that node in the training set or, in the case of regression trees, assigned a value based on the mean (or other mathematical function) of the values that reached that leaf in the training set.

Various decision tree algorithms exist, which produce trees that differ from one another in the number of splits allowed at each level of the tree, how those splits are chosen when the tree is built, and how the tree growth is limited to prevent overfitting. Although these variations have led to many a doctoral thesis, for our purposes they are not very interesting. Today’s DM software tools typically allow the user to choose among several splitting criteria and pruning rules, and to control parameters such as minimum node size and maximum tree depth allowing one to approximate any of these algorithms.

Fig. 5 Decision tree cuts the space into boxes (Source: *Berry and Linoff, 2000, p.112*)



The discussion on clustering described how the fields in a record can be viewed as the coordinates of that record in a multidimensional record space. That geometric way of thinking is useful when talking about decision trees as well. Each branch of a decision tree is a test on a single variable that cuts the space into two or more pieces. For concreteness and simplicity, let's consider a simple example where there are only two input variables, X and Y. These variables take on values from 0 to 100. Each split in the tree is constrained to be binary That is to say, at every node in the tree, a record will go either left or right based on some test of either X or Y.

In Fig. 5 a decision tree has been grown until every box is completely pure in the sense that it contains only one species of dinosaur. Such a tree is fine as a description of this particular arrangement of stegosauruses and triceratopses, but is unlikely to do a good job of classifying another similar set of prehistoric reptiles.

For some training sets, it is possible to build a decision tree that correctly classifies every single record. This is possible when the training set contains no examples of records whose input variables all have the same values, but whose target variables belong to different classes. Although such a tree provides a good description of the training data, it is unlikely to generalize to new data sets. That is why the test set is used to prune the tree once it has been grown using the training set.

Why? A tree that precisely describes the data from which it was derived is unlikely to generalize well to another sample drawn from the same population. This problem is known as *overfitting, a* topic we will return to in next and other Sections. But ignoring that for the moment, how would we use this tree to classify an unknown dinosaur for which X=40 and Y=75? Starting at the root node, we go to the right because the Y-value is greater than 50. Then, since the X-value is not greater than 80, we classify the unknown dinosaur as a Triceratops. Equivalently, by looking at the box chart we can see that the point (40, 75) is clearly in a box containing only triceratopses.

Decision trees are built through a process known as recursive partitioning. Recursive partitioning is an iterative process of splitting the data up into partitions – and then splitting it up some more. Initially, all of the records in the training set, i.e. the

preclassified records that are used to determine the structure of the tree, are together in one big box. The algorithm then is breaking up the data, using every possible binary split on every field. So, if age takes on 72 values, from 18 to 90, then one split is everyone who is 18 and everyone older than 18. Another is everyone who is 18 or 19, and everyone who is 20 or older. And so on. The algorithm chooses the split that partitions the data into two parts that are purer than the original. This splitting or partitioning procedure is then applied to each of the new boxes. The process continues until no more useful splits can be found.

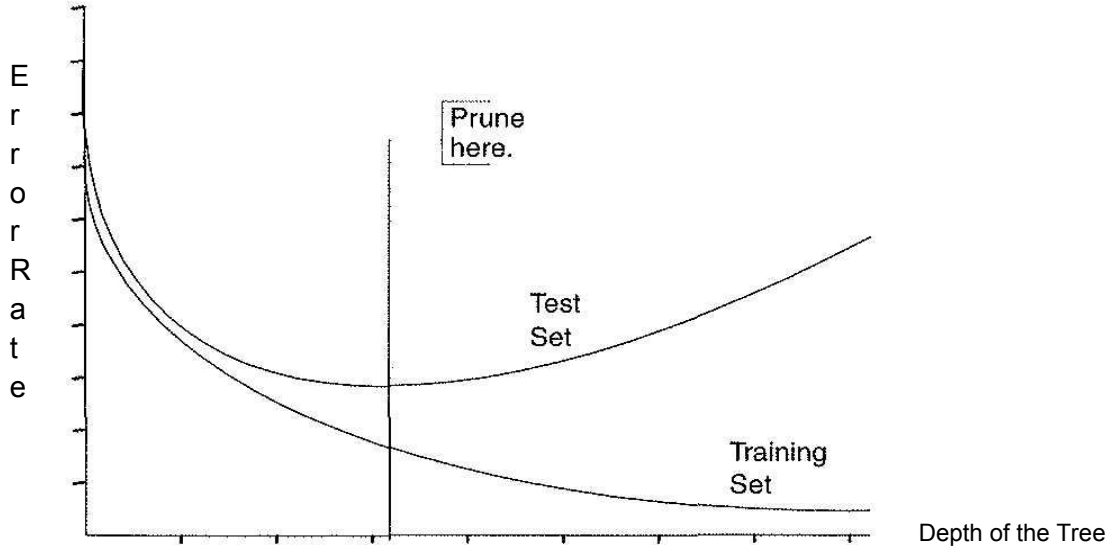
The graph in Fig. 6 shows how to make a pruning decision when data is plentiful (as is rarely the case in the academic environments where algorithms are developed, but is frequently true in the commercial world where they are applied) using an approach, which bases the pruning decision on the actual performance of the tree. The performance of the tree and all of its subtrees is measured on a separate set of preclassified data, called the test set (see next section and Fig. 12). With a single test set, the algorithm can prune back to the subtree that minimizes the error on the test set. With multiple test sets, we can even more directly address the issue of model generality by selecting the subtree that performs most consistently across several test sets.

It is important to point out some of the consequences of choosing Decision Trees. First, because every split in a decision tree is a test on a single variable, they can never discover rules that involve a relationship between variables. This puts a responsibility on the researcher to add derived variables to express relationships that are important.

For example, a loan database is likely to have fields for the initial amount of the loan and the remaining balance, but neither of these fields is likely to have much predictive value in isolation. The ratio of the outstanding balance to the initial amount carries much more helpful information, but a decision tree will never discover a single rule based on this ratio unless it is included as a separate variable.

One advantage to the way decision trees treat numeric inputs is that they are not sensitive to scale differences between the inputs, nor to outliers and skewed distributions. This means that data preparation is less of a burden with decision trees than it is with *ANNs* and *k-means* clustering.

Fig. 6 Error rate on training set and test set as tree complexity increases



The handling of categorical variables can also cause problems. Depending on the particular algorithm employed, categorical variables may be split on every value taken on by the variable, leading to a very bushy tree that soon runs out of records on which to base further splits. Other algorithms find ways to group class labels into a small number of larger classes by combining classes that yield similar splits. Since the number of possible groupings grows very large, very fast as the number of classes grows, an exhaustive search of all combinations quickly becomes impractical. Software products use various shortcuts to pare down the space to be searched, but the clustering process can still be quite time consuming.

Decision trees are error-prone when the number of training examples per class gets small. This can happen rather quickly in a tree with many levels and/or many branches per node because trees are very sensitive to the density of the outcomes.

Decision-tree building algorithms put the field that does the best job of splitting at the root node of the tree (and the same field may appear at other levels in the tree as well). It is not uncommon for decision trees to be used for no other purpose than prioritizing the independent variables. That is, using a decision tree, it is possible to pick the most important variables for predicting a particular outcome because these variables are chosen for splitting high in the tree.

Another useful consequence of the way that important variables float to the top is that it becomes very easy to spot input variables that are doing *too* good a job of prediction because they encode knowledge of the outcome that is available in the training data, but would not be available in the field. There are many amusing examples of this, such as discovering that people with nonzero account numbers were the most likely to respond to an offer of credit – less than surprising since account numbers are assigned only after the application has been processed.

Decision tree methods are often chosen for their ability to generate understandable rules, but this ability can be overstated. It is certainly true that for any particular classified record, it is easy to simply trace the path from the root to the leaf where that record landed in order to generate the rule that led to the classification – and most decision tree tools have this capability. Many software products can output a tree as a list of rules in SQL, pseudocode, or pseudo-English. However, a large complex decision tree may contain hundreds or thousands of leaves. Such a tree is hardly more likely than an neural network to communicate anything intelligible about the problem as a whole.

When to use Decision Trees – Decision-tree methods are a good choice when the data mining task is classification of records or prediction of outcomes. We should use decision trees when the goal is to assign each record to one of a few broad categories. Decision trees are also a natural choice when the goal is to generate rules that can be easily understood, explained, and translated into SQL or a natural language.

4.3. Artificial Neural Networks

Artificial Neural Networks (ANNs) are at once the most widely known and the least understood of the major data mining techniques. Much of the confusion stems from overreliance on the metaphor of the brain that gives the technique its name. The people

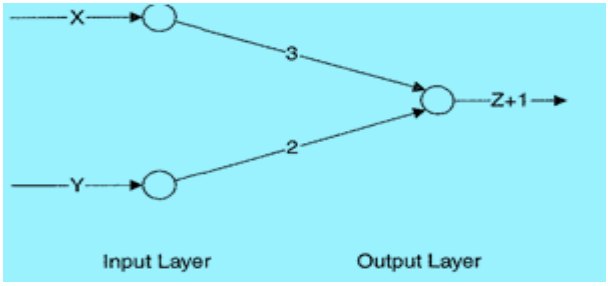
who invented ***ANNs*** were not statisticians or data analysts. They were machine learning researchers interested in mimicking the behavior of natural neural networks such as those found inside of fruit flies, earthworms, and human beings The vocabulary these machine learning and artificial intelligence researchers used to describe their work – “perceptrons”, ”neurons”, “learning”, and the like – led to a romantic and anthropomorphic impression of neural networks among the general public and to deep distrust among statisticians and analysts. Depending on your own background, you may be either delighted or disappointed to learn that, whatever the original intentions of the early neural networkers, from a data mining perspective, neural networks are just another way of fitting a model to observed historical data in order to be able to make classifications or predictions.

To illustrate this point and to introduce the various components of a neural network, it is worth noting that standard linear regression models and many other functions equally devoid of mystery can easily be drawn as neural network diagrams. Take for example, the function z= 3x+2y-l. There are two variable inputs, x and y. For any values of x and y, the function will return a value for z. It might be that this function is a model, based on many observed values of x, y, and z that is now being used to predict values for z given new, previously unobserved values of x and y. If so, it is a predictive model just as surely as anything created by a data mining software package. In fact, this particular predictive model is represented by the simple neural network in Fig. 7.

In neural network terminology, this network has an *input layer* and an *output layer.* Each of the inputs x and y gets its own *unit,* or network node. In general, it is not the actual values of the input variables that are fed into the input layer, but some transformation of them. Each input unit is connected to the output unit with a *weight.* In this case, the weights are the coefficients 3 and 2. Inside the output unit, the input weights are combined using a *combination function* (typically summation, as in this case) and then passed to a *transfer function,* the result of which is the output of the network. Together, the combination function and the transfer function make up the unit’s *activation function.* The value produced by the output node’s activation function is usually some transformation of the actual desired output. In this case, the network outputs z+l rather than *z.* Just as some function is applied to the input variables in order to generate suitable inputs to the neural network, some function of the network’s output is required to translate it back to the actual range of the target variable.

Most neural networks are not as simple as the one in Fig. 7. There is usually one, but sometimes more than one, additional layer of units between the input layer and the output layer. These layers are called *hidden layers* and the units in them are *hidden units.* Fig. 8 shows a neural network similar to the one in Fig. 7 but with a hidden layer. With the addition of the hidden layer, the function represented by the network is no longer a simple combination of its inputs. The output value is now calculated by feeding the weights coming from the two hidden units to the activation function of the output unit. The weights produced by the hidden units are themselves functions of the input units, each of which is connected to both units of the hidden layer. All this gets pretty complicated, pretty quickly, which is why no one ever actually writes out a neural network as an equation. The point is, though, it could be done!

Fig. 7 Neural Network representation of z=3x+2y-1



The network illustrated here is a feed-forward network with a hidden layer. By feed-forward, we mean that data enters at the input nodes and exits at the output nodes without ever looping back on itself. Networks like this are also called *multilayer perceptrons.* If there is a “standard” neural network, it is the fully connected, feed-forward network with one hidden layer and a single- node output layer, but there are many, many variations. Often, there are multiple nodes in the output layer, each estimating the probability of a separate class of the target variable. Sometimes there is more than one hidden layer. Sometimes there are direct links from inputs to outputs that skip the hidden layer. There are neural network architectures that include loops and ones where the inputs arrive in waves, not all at the same time. Usually, when neural networks are discussed, we are referring to fully connected, feed-forward, multilayer perceptrons.

Inside each unit of a neural network, there is an activation function that consists of a combination function and a transfer function. The combination function is nearly always the weighted sum of the inputs. Transfer functions come in many more flavors.

The graph in Fig. 9 shows a linear transfer function illustrating the neural network drawn in Fig. 8, which represents a linear function. More commonly, the transfer function is *sigmoidal* (S-shaped) or bell-shaped. The bell-shaped transfer functions are called *radial basis functions.* Common sigmoidal transfer functions are the arctangent, the hyperbolic tangent, and the logistic. The nice thing about these S-shaped and bell-shaped functions is that any curve, no matter how wavy, can be created by adding together enough S-shaped or bell-shaped curves. In fact, multilayer perceptrons with sigmoidal transfer functions and radial basis networks are both *universal approximators,* meaning that they can theoretically approximate any continuous function to any degree of accuracy. Of course, theory does not guarantee that we can actually find the right neural network to approximate any particular function in a finite amount of time, but it’s nice to know it and also that decision trees are not universal approximators.

Fig. 8 Artificial Neural Network with a hidden layer

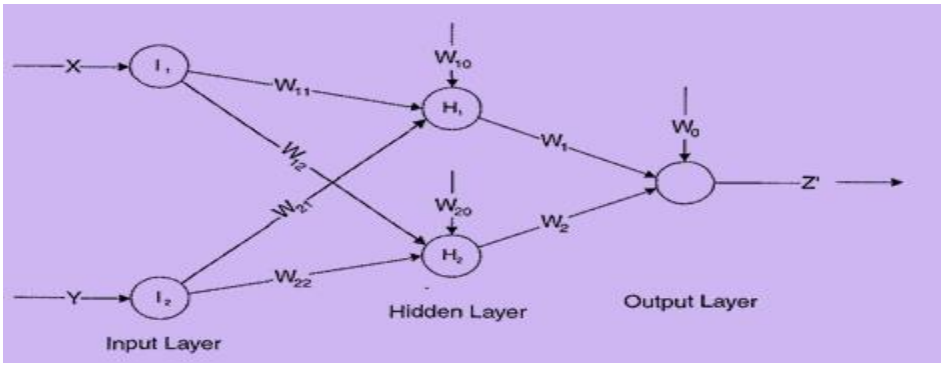
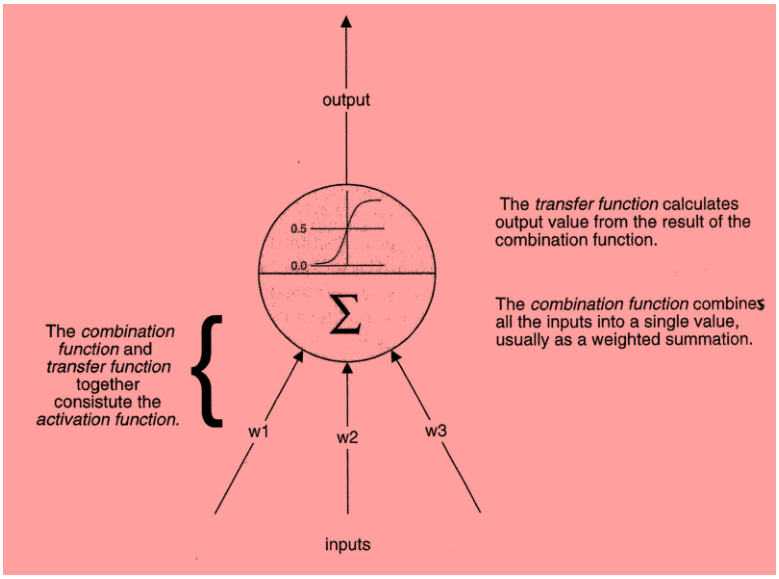


Fig. 9 Linear transfer function



The sigmoidal transfer functions used in the classic multilayer perceptrons have several nice properties. The shape of the curve means that no matter how extreme the input values, the output value is always constrained to a known range (–1 to 1 for the hyperbolic tangent and the arctangent, 0 to 1 for the logistic). For moderate input values, the slope of the curve is nearly constant. Within this range, the sigmoid function is almost linear and exhibits almost-linear behavior. As the weights get larger, the response becomes less and less linear as it takes a larger and larger change in the input to cause a small change in the output. This behavior corresponds to a gradual movement from a linear model to a nonlinear model as the inputs become extreme.

Training a neural network is the process of setting the weights on the inputs of each of the units in such a way that the network best approximates the underlying function, or put in data mining terms, does the best job of predicting the target variable. This is an optimization problem and there are whole textbooks dedicated to optimization, but in broad outline most software packages for building neural network models use some variation of the technique known as *backpropagation.* The term *backpropagation* refers to any method of training a neural network that involves comparing the expected result for a given set of inputs to the output of the network during a training run, and feeding that difference back through the network to adjust the weights. In general, most of the neural networks in use today are trained using backpropagation. However, the original backpropagation networks popularized in the 1980s used an optimization method called *steepest descent* to correct the network weights. This turns out to be inefficient and is now generally replaced by other algorithms such as conjugate gradient or modified Newton. Some writers reserve the term “backpropagation networks” for the earlier, less efficient variety and coin new terms for each combination of error estimate and optimization method. This could be somehow confusing, so we’ll call all “backpropagation methods”.

Training a backpropagation neural network has three steps:

1. The network gets a training instance and, using the existing weights in the network, it calculates the output or outputs for the instance.

2. Backpropagation then calculates the error, by taking the difference between the calculated result and the expected (actual result).

3. The error is used to adjust the weights (this is referred to as feeding the error back through the network).

Using the error measure to adjust the weights is the critical part of any back-propagation algorithm. In classic backpropagation, each unit is assigned a specific responsibility for the error. For instance, in the output layer, one unit is responsible for the whole error. This unit then assigns a responsibility for part of the error to each of its inputs, which come from units in the hidden layer, and so on, if there is more than one hidden layer. The specific mechanism is not important. Suffice it to say that it is a complicated mathematical procedure that requires taking partial derivatives of the transfer function. More recent techniques adjust all the weights at once, which is one of the things that make them more efficient.

Given the error, how does a unit adjust its weights? It starts by measuring how sensitive its output is to each of its inputs. That is, it estimates whether changing the weight on each input would increase or decrease the error. The unit then adjusts each weight to reduce, but not eliminate, the error. The adjustments for each example in the training set slowly nudge the weights toward their optimal values. The goal is to generalize and identify patterns in the input, not to match the training set exactly. Adjusting the weights is “like a leisurely walk instead of a mad-dash sprint”. After being shown enough training examples, the weights on the network no longer change significantly and the error no longer decreases. This is the point where training stops – the network has learned the input.

One of the concerns with any neural network training technique is the risk of falling into something called a local optimum. This happens when the adjustments to the network weights suggested by whatever optimization method is in use no longer improve the performance of the network even though there is some other combination of weights, significantly different from those in the network, that yields a much better solution. This is analogous to trying to climb to the top of a mountain and finding that you have only climbed to the top of a nearby hill. There is a tension between finding the local best solution and the global best solution. Adjusting parameters such as the learning rate and momentum helps to find the best solution.

Neural networks can produce very good predictions, but they are neither easy to use nor easy to understand. The difficulties with ease of use stem mainly from the extensive data preparation required to get good results from a neural network model. The results are difficult to understand because a neural network is a complex nonlinear model that does not produce rules.

The biggest drawback of typical neural networks in a business decision support context is that they *cannot explain results*. For many users, understanding what is going on is often as important, if not more important, than getting the best prediction. In situations where explaining rules may be critical, such as denying loan applications, neural networks are not a good choice. There are many situations, however, when the prediction itself matters far more than the explanation. The neural network models that can spot a

potentially fraudulent credit card transaction before it has been completed are a good example. An analyst or data miner can study the historical data at leisure in order to come up with a good explanation of why the transaction was suspicious, but in the moments after the card is swiped, the most important thing is to make a quick and accurate prediction.

When to use Neural Networks – Neural networks are a good choice for most classification and prediction tasks when the results of the model are more important than understanding how the model works. Neural networks actually represent complex mathematical equations, with lots of summations, exponential functions, and many parameters. These equations describe the neural network, but are quite opaque to human eyes. The equation is the rule of the network, and it is useless for our understanding.

Typical neural networks do not work well when there are many hundreds or thousands of input features. Large numbers of features make it more difficult for the network to find patterns and can result in long training phases that never converge to a good solution. Here, neural networks can work well with decision tree methods. Decision trees are good at choosing the most important variables and these can then be used for training a network.

The primary lesson that one should take away from this discussion is that no one data mining technique is right for all situations and a possible solution is in unification of different techniques. One possible solution could be the so called *Statistical Learning Networks*. In next sections we'll see how this method can help researchers analyzing the massive amounts of data and turning information located in the data into successful decisions.

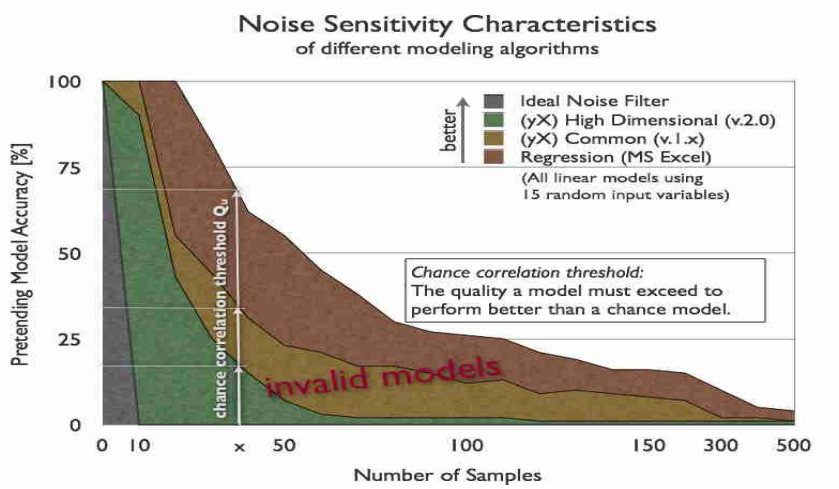
5. SELF-ORGANIZING DATA MINING

5.1. Predictive Techniques and Models

Predictive analytics is an area of *data mining* that deals with extracting information from data and using it to predict trends and behavior patterns. Often the unknown event of interest is in the future, but predictive analytics can be applied to any type of unknown, whether it is in the past, present or future. The core of predictive analytics relies on capturing relationships between explanatory variables and predicted variables from past occurrences, and exploiting them to predict the unknown outcome.

In the past, before the advent of modern forecasting techniques and the power of the electronic computers, the manager's judgment, based on experience and very often just intuition, was the only tool available in decision making. This situation totally changed in the second part of the last century. Today it includes a number of advanced statistical methods for regression and classification. In certain applications it is sufficient to directly predict the dependent variable without focusing on the underlying relationships. In other cases, the underlying relationships can be very complex and the mathematical form of the dependencies unknown. For such cases, machine learning techniques emulate human cognition and learn from training examples to predict future events.

Fig. 10 Example of *KnowledgeMiner* software user-friendly output



The real-life experience shows that a simple forecast method, which is well understood, will be better implemented than one with all inclusive features but that is unclear in certain facets. Historically, using predictive analytics tools, as well as understanding the results they delivered, required advanced skills. However, modern predictive analytics tools are no longer restricted to specialists. As more organizations adopt predictive analytics into decision-making processes and integrate it into their operations, they are creating a shift in the market toward business users as the primary consumers of the information.

Business users want tools they can use on their own. Vendors are responding by creating software that removes the mathematical complexity, provides user-friendly graphic interfaces, and/or builds in short cuts that can, for example, recognize the kind of data available and suggest an appropriate predictive model. Predictive analytics tools have become sophisticated enough to adequately present and dissect data problems, so that any data-savvy information worker can utilize them to analyze data and retrieve meaningful, useful results. For example, modern tools like *KnowledgeMiner* software (see *Mueller and Lemke, 2003)* present findings using simple charts, graphs, and scores that indicate the likelihood and/or the level of possible outcomes (Fig. 10).

Nearly any regression model can be used for prediction purposes. Broadly speaking, there are two classes of predictive models: parametric and non-parametric. In parametric, the modeler makes “specific assumptions with regard to one or more of the population parameters that characterize the underlying distribution(s)” (see *Sheskin 2011*), while non-parametric regressions require fewer assumptions than their parametric counterparts. A third class of semi-parametric models also exists, which includes features of both.

The approaches and techniques to conduct predictive analytics can be generally grouped into *regression techniques* and *machine learning techniques*. It is important to note that the accuracy and usability of results will depend greatly on the level of data analysis and the quality of assumptions. Unfortunately in economy, ecology, sociology, etc., many objects are ill-defined systems that can be characterized by inadequate a priori

information about the system, big number of immeasurable variables, fuzzy objects with attributive variables, noisy and/or small data samples.

Regression analysis focus lies on establishing a mathematical equation as a model to represent the interactions between the different variables in consideration. Depending on the situation, there is a wide variety of models that can be applied while performing predictive analytics – multiple regression (linear or non-linear), logistic and probit regression, time series models, robust regression, multivariate adaptive regression splines, and others with much narrower application.

The performance of regression analysis methods in practice depends on the form of the data generating process and how it relates to the regression approach being used. Since the true form of the data-generating process is generally not known, regression analysis often depends to some extent on making assumptions about this process. Most significant problems here are related to complex objects systems identification, pattern recognition, approximation, and extrapolation. Technically, that means massive arrays of potential explanatory variables and at the same time short time-series data (i.e. overfitting), strong relationships between explanatory variables (multicollinearity), autocorrelation of the errors, unknown time lags and other problems.

It should be noted that regression techniques continue to be an area of active research. In recent decades, new methods have been developed for robust regression, regression involving correlated responses such as time series and growth curves, regression in which the predictor or response variables are curves, images, graphs, or other complex data objects, regression methods accommodating various types of missing data, nonparametric regression, Bayesian methods for regression, regression in which the predictor variables are measured with error, regression with more predictor variables than observations, and causal inference with regression.

Machine learning, a branch of artificial intelligence, was originally employed to develop techniques to enable computers to learn. Today, since it includes a number of advanced statistical methods for regression and classification, it finds application in a wide variety of fields including medical diagnostics, credit card fraud detection, face and speech recognition, analysis of the stock market and more. Some of the methods used commonly for predictive analytics are geospatial predictive modeling, k-nearest neighbors, support vector machines, radial basis functions, and artificial neural networks.

One of the main problems is that the mathematical relationship that assigns an input to an output and imitates the behavior of a real-world system using these relationships usually has nothing to do with the real processes running in the system. Machine learning models are implicit models with no explanation component by default, and the systems details and relationships are not at all described. The analyzed system is treated as a black box, and this approach cannot be used to analyze cause-and-effect relationships. Another important problem is that the knowledge of observed input values does not uniquely specify the output. In most cases designing topology is a trial-and-error process; there are no rules concerning how to use the theoretical (a priori) knowledge in design process and other less significant issues.

5.2. Group Method of Data Handling

The *Group Method of Data Handling (GMDH)* is a heuristic, self-organizing modeling method *(Ivakhnenko, 1966)*. It contains a family of inductive algorithms for computer-based mathematical modeling of multi- parametric datasets that features fully automatic structural and parametric identification of models. It could be used in such fields as data mining, knowledge discovery, prediction, complex systems modeling, optimization and pattern recognition. In GMDH-type self-organizing algorithms, models are generated adaptively from data in the form of networks of active neurons in a repetitive generation of populations of competing models of growing complexity, corresponding cross-validation, and model selection until an optimal complex model is finalized (Fig. 11).

The most popular base function used in GMDH is the gradually complicated Kolmogorov-Gabor polynomial (1):

$$Y(x_1,\ldots,x_n) = a_0 + \sum_{i=1}^{n} a_i x_i + \sum_{i=1}^{n}\sum_{j=i}^{n} a_{ij} x_i x_j + \sum_{i=1}^{n}\sum_{j=i}^{n}\sum_{k=j}^{n} a_{ijk} x_i x_j x_k + \cdots \quad (1)$$

In order to find the best solution, GMDH algorithms consider various component subsets of the base function (1) called partial models. Coefficients of these models are estimated by the least squares method. GMDH algorithms gradually increase the number of partial model components and find a model structure with optimal complexity indicated by the minimum value of an external criterion.

External criterion, also known as *cross-validation technique (Stone, 1977)* is one of the key features of *GMDH*. It is always calculated from a separate part of the data sample (testing set) that has not been used for estimation of coefficients (Fig. 12). *GMDH* is also known as *polynomial neural networks* and *statistical learning networks*, thanks to implementation of the corresponding algorithms in several commercial software products *(Madala and Ivakhnenko, 1994)*.

Fig. 11 *GMDH* iterative algorithm – a multilayered active neuron neural network

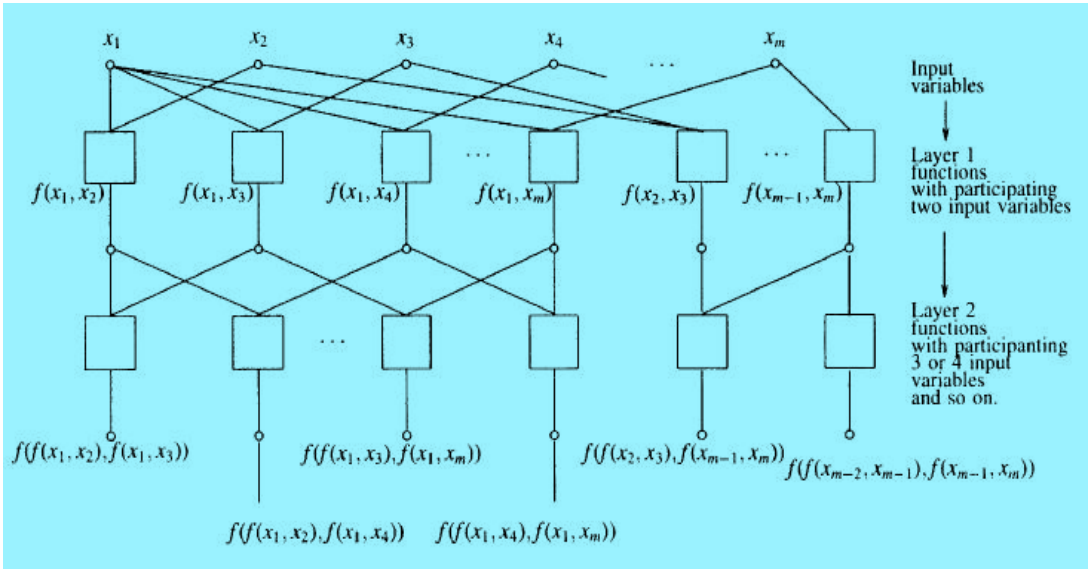
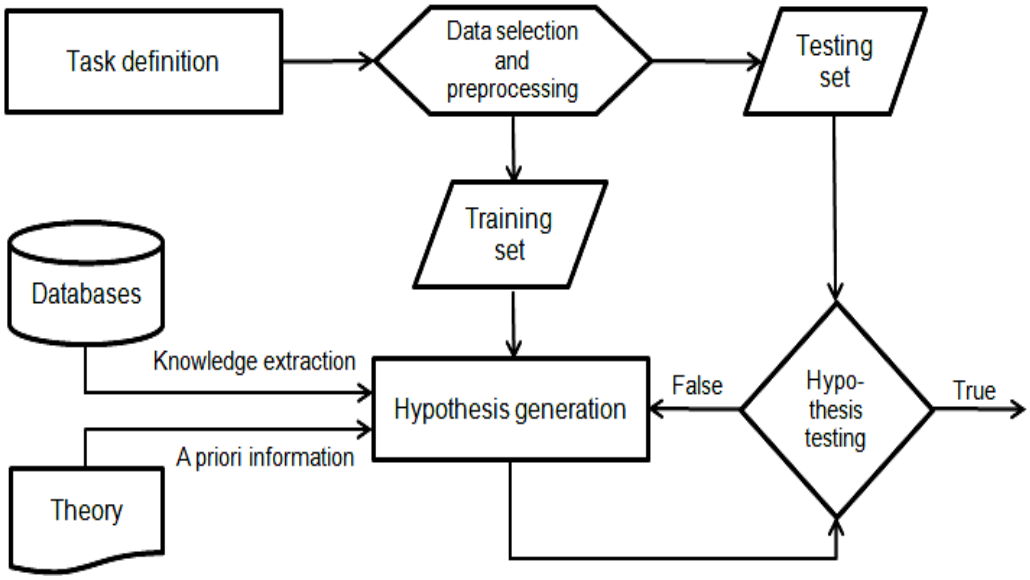


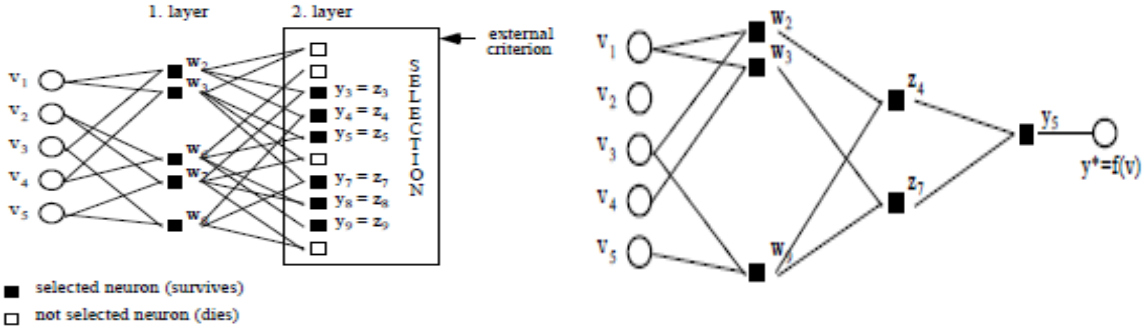
Fig. 12 *Basic scheme of self-organizing modeling with a priori information*



This modeling approach grows a tree-like network out of data of input and output variables (seed information) in a pair-wise combination and competitive selection (like the genetic algorithms) from a simple single individual (a neuron) to a desired final solution (the model – Fig. 13), which does not have a predefined behavior. Neither the number of neurons, the number of layers in the network, nor the actual behavior of each created neuron is predefined. In this way, the modeling process is self-organizing because the number of neurons, the number of layers, and the actual behavior of each created neuron are adjusting during the model-building process.

The algorithm presented in this paper was initially developed by *Marchev and Motzev (1985)* in the 1980s[1], during the first steps of GMDH and machine learning techniques. It was designed as a ***multilayer net of active neurons (MLNAN)***, which works both for multi-input to single-output models' identification (for example different type of regression models) and for building complex models of simultaneous equations (i.e. multi-input to multi-output).

Fig. 13 Examples of *ANNs* after selection of best models at the second layer and optimal model **y*** selected by a neural network with three layers



[1] For this research in 1980 the author was nominated the First Prize Award in Individual Competition at the National Scientific Session for Students in Economics, Sponsored by Bulgarian Ministry of Education.

The basic idea *(Motzev, 1985)* in this MLNAN is that first the elements on a lower level are estimated and the corresponding intermediate outputs are computed and then the parameters of the elements of the next level are estimated. At the first layer, all possible pairs of the inputs are considered as potential factors, and only some of the best (in the sense of the selection criteria – here coefficient of correlation) intermediate models are used as inputs for the next layer(s). In the succeeding layers all possible pairs of the intermediate models from the preceding layer(s) are connected as inputs to the components of the next layer(s). This means that the output of a component at a processed level is or may become an input, depending on a local threshold value (the selection criterion here is the coefficient of determination of the partial model), to several other components at the next level. Finally, when additional layers provide no further improvement, the network synthesis stops.

It should be noted that self-organization does not replace a good domain theory. Inclusion of some well-known, a priori information widens the basic scheme (Fig. 12) of self-organizing modeling by knowledge extraction from data and scientific theory. However, as mentioned in *Mueller and Lemke (2003)*, very often self-organization provides the only way to get any knowledge from a complex system or to add some new aspects to existing theoretical fragments.

The procedure described above is a multilayer GMDH algorithm for multi-input to single-output models identification. In case of synthesizing complex models in the form of *Simultaneous Equations (SE)*, i.e. multi-input to multi-output models, an additional part was developed as an iterative procedure (see *Motzev and Marchev, 1988*). Here, the equations from the previous part are used to synthesize the *SE*:

- The intermediate models are generated combining already chosen, good equations according to the combinatorial algorithm.
- Each of the competing hypotheses is a hypothesis about the significance of entering a given version of a single equation into the system of SE.
- Each generated SE is considered as a potential model for the system of interest, which competes with others “fighting for survival”.
- The evaluation of these competing models is done, using a complex set of criteria – MSE, coefficient of determination, MAPE, and others.
- If the results are unsatisfactory after solving the structural form of the system (biased values of the coefficients, low accuracy of the equations, or other), the procedure returns to the first part.
- The decision maker can then apply some new, a priori knowledge and/or add fresh data, or change the selection criteria etc. Then a new synthesis of the structural equations is completed and with the so-obtained new set of equations the second part begins again. It ends when satisfactory results have been achieved in the sense of the selection criteria.
- The final choice of the “best” model is made by the decision maker, who has one final option to apply additional, qualitative information/knowledge, but after having the

guarantee that a large number of possible models have been evaluated and the final choice is based on a small number of good ones.

All these characteristics make the proposed *MLNAN* very useful for addressing most of the model-building problems discussed above. For example, overfitting is eliminated by the use of external criterion (cross-validation) for validating the model. The small number of independent measurements (or short time-series) is also not an issue, because the inverted matrix size is always 2x2 (pair-wise combinations). This helps in dealing with the problem of multicollinearity as well. The autocorrelation is eliminated by adding lagged time series values as predictors. Despite the totally automated procedure the decision maker has options at the crucial points to apply additional insights, knowledge or hypotheses.

5.3. Applications and benefits from the *MLNAN*

The first working prototype of the *MLNAN* described above was used to improve an existing business game (see *Motzev and Marchev, 1984*). The "*National Economy*" game had been used for many years at the Economic University in Sofia, Bulgaria. The original version contains a model developed with the general multiple regression analysis. The same data and set of variables were used to build a new model, using the MLNAN algorithm.

The brief comparison shows (Table 1) that the new version has much better accuracy (more than five times smaller MSE%) and thus provides a more reliable base for simulations and what-if analysis. Increasing model accuracy provides many other benefits. For example, it makes it possible to analyze more precisely the problem in consideration, which leads to a deeper and clearer understanding of the problem itself. Also, a model with higher accuracy will generate better predictions and help players making better and more cost effective decisions.

Another area of application of the MLNAN was the macroeconomic modeling. A series of increasingly complex simulation models of the Bulgarian economy was developed (*Marchev and Motzev, 1985, 1989, 1991*) as follows:

- SIMUR I (1980): One-product macroeconomic model in the form of 5 SE. Contains 5 endogenous, 5 lag and 1 exogenous variables. Average MSE%=2.7%

Tab.1 Business game *National Economy* - Model characteristics and comparisons

Characteristics	**Old Version**	**Improved Version**
Model description	A one-product macro-economic model developed as a system of five SE. Contains five endogenous, one exogenous, and five lag variables.	A one-product macroeconomic model with the same structure. Contains same set of variables.
Model-building technique	Indirect OLS used to estimate unknown coefficients in equations.	Model synthesized using the GMDH procedure.
Model accuracy	Mean squared error relative to the mean (MSE%) = 14%	MSE% = 2.7%

Source: Own data

- SIMUR II (1985): Aggregated macroeconomic model in the form of 12 SE. Contains 12 endogenous, 5 exogenous and 26 lag variables with lag of up to 3 years. Average MSE% = 2.0%
- SIMUR III (1987) – Complex macro-economic model of 39 SE, with 39 endogenous, 7 exogenous and 82 lag variables (time lag up to 5 years). Average MSE% <1%.

The results confirmed that the proposed approach provides opportunities to shorten the time and reduce the cost and the efforts in model building. Also, the accuracy of the models show that this technique is able to develop even complex models reliably with low overall error rates.

It is important to note that the MLNAN was also used in time-series analysis to build autoregressive models (ARM) for 24 macroeconomic variables with a time lag of up to 5 years *(Motzev et al., 1986)*. The average MSE% for all models was 4.74% and the coefficient of determination is > 0.9 for most of them (average R2 = 0.9339). Recently, another detailed study *(Onwubolu, 2009)* of the predictive performance of two time series forecasting techniques (Elman neural network and GMDH algorithms) against the autoregressive integrated moving average (ARIMA) also confirmed that GMDH based techniques are able to develop even complex models reliably and achieve lower overall error rates than state-of-the-art methods.

Predictive analytics and in particular GMDH based techniques require both powerful hardware and fast software, designed and elaborated for this specific aim. One element in the first working prototype that needed improvement was the software, which had been designed for mainframes and mini computers (*Marchev and Motzev 1989*). The original version was too large and too complex — its total volume was about 30 thousand program lines in PL/1. It was designed for an IBM 4331 computer under the VM-370 operating system. At present, this system has a multitude of abilities and parameters that are assigned in interactive mode, which in fact often hampers more than aids the unprepared user. Moreover, it ties him to an outdated operating system and an expensive computer.

Developing your own computer program in this area is a big project, which takes extensive resources, time and highly qualified professionals. The numerous tools, available on the market, that help with the execution of predictive analytics range from those that need very little user sophistication to those that are designed for the expert practitioner. One possible solution was the "*KnowledgeMiner*" data mining software *(Mueller and Lemke, 2003)*. It is a self-organizing tool for modeling and predictions that implements GMDH, Analog Complexion, and Fuzzy Rule Induction techniques.

"KnowledgeMiner" and its new improved version *"Insights"* can be used to create linear & nonlinear, static and dynamic time series models, multi-input/single-output and multi-input/multi-output models as systems of equations even from small and noisy data samples. The model outputs are presented both analytically (as equations with estimated coefficients) and graphically, by a system graph reflecting the interdependent structure of the system.

To evaluate the proposed tool, a comparison was done using a model like SIMUR II, created with similar data for the German national economy in the form of 13 SE by the developers of “KnowledgeMiner” software (*Mueller and Lemke, 2003*). It wasn’t possible to use the same set of data, and the results could be used only for general comparisons, however, both models show similar levels of high reliability and accuracy.

It is important to note that one software module in the original prototype (program *SIMUL*) provides some options not covered by “KnowledgeMiner”. “KnowledgeMiner” has an excellent module for complex evaluation of the synthesized model, its adequacy and reliability, but *SIMUL* provides more options for conducting different simulation experiments and what-if analysis. Updating this program and making it compatible with “KnowledgeMiner (yX) for Excel” software would be a useful new project in the future.

With the new software, the proposed MLNAN was used in the model-based management game “New Product” *(Motzev, 2012)*, designed for students in “Production and Operations Management” class at the Walla Walla University. In this game, after accumulating good knowledge and experience at the initial steps of the game students work on a case-scenario which is an example of stochastic business process. Here, like in the real-life business, there is similarity between observations, which means that the time series data are autocorrelated and could be presented with an ARM. It specifies that the output variable depends on its own previous values and is one of the prediction functions used to forecast an output of a system Y(t) based on the previous outputs Y(t-i). However, in most cases the order of the general ARM is unknown, which makes very difficult both its structural and parametrical identification. Leading researchers suggest applying an iterative approach to model building for forecasting and decision making *(Box and Jenkins, 2008)*. This makes the MLNAN presented above very appropriate tool to develop the unknown ARM, which students can use to make better and more cost/effective decisions at the next stages of the game.

The game was accepted with great interest and students reported in their feedback that the game was a valuable learning tool that helped to increase their knowledge and competencies.

The benefits of utilizing MLNAN in business simulations are obvious. The proposed approach provides opportunities to shorten the time and reduce the cost and the efforts in model building and at the same time to develop even complex models reliably with low overall error rates. Increasing model accuracy helps researcher to analyze more precisely the problem, which leads to its deeper and better understanding. Also, a model with higher accuracy will generate better predictions and support managers making better decisions, much closer related to the real-life business problems.

If a business games (especially model-based games) does not represent with high accuracy the real system, than the knowledge that the students will receive about the real-life business is questionable. For example, it does matter to know more precisely how much will increase the marginal profit by reducing the total cost of production, or by increasing the cost of advertising. If the game model is not accurate and the predictions made by the players are not close enough to the real-life business case there is not too much learning.

6. CONCLUSIONS

The proposed MLNAN provides opportunities to shorten the time and reduce the cost and the efforts in business simulations and model-based business games. The results so far show that it is able to develop even complex models reliably and achieves lower overall error rates than state-of-the-art methods. Of course, there are some limitations of predictive models based on data fitting. For example, history cannot always predict the future; using relations derived from historical data to predict the future implicitly assumes certain steady-state conditions or constants in the complex system. This is almost always inaccurate when the system involves people.

Another issue is the “unknown unknowns”. In all data collection, the researcher first defines the set of variables for which data is collected. However, no matter how extensive the researcher considers his selection of the variables, there is always the possibility of new variables that have not been considered or even defined, yet that are critical to the outcome.

It is imperative to conclude, however, that the model outputs must always be evaluated by the researcher to figure out whether new and useful knowledge of the domain has been discovered. Predictive analytics create and provide data, but the real-life business needs information, i.e. data in the business context. The extracted information is valuable to a business only when it leads to actions that create value or market behavior that gives a competitive advantage. The researcher has to determine the ultimate importance of the information generated by algorithms like the MLNAN described in this paper.

References

1. Elgood, Ch., 2005. Using Management Games. Burlington, USA. Ashgate Publishing, p.10.
2. Luhn, H.P., 1958. A Business Intelligence System. IBM Technical Journals, Volume 2, Number 4, p. 314.
3. Power D.J. (ed.). 2007. A Brief History of Decision Support Systems (ver. 4.0). Decision Support Systems Resources, p. 6. Available at: < http://dssresources.com/history/dsshistory.html.
4. Davenport, T. and Harris, J., 2007. Competing on Analytics: The New Science of Winning. Boston: Harvard Business School Press, March 2007, pp. 7-8.
5. Beller, M. and Barnett, A., 2009. Next Generation Business Analytics Technology Trends. Lightship Partners LLC, p. 5. Available at: < (http://www.docstoc.com/docs/7486045/Next-Generation-Business-Analytics-Technology-Trends).
6. Shmueli, G. and others, 2007. Predictive vs. Explanatory Modeling. IS Research, Conference on Information Systems and Technology, Seattle, WA, November 3-4, Available at: < (http://www.citi.uconn.edu/cist07/5c.pdf).
7. Nyce, Ch., 2007. Predictive Analytics. White Paper, American Institute for Chartered Property Casualty Underwriters/Insurance Institute of America, p. 1.
8. Turban, E. and Aronson, J., 2001. Decision Support Systems and Intelligent Systems. Prentice-Hall, p. 148.
9. Berry, M. and Linoff, G., 2000. Mastering Data Mining. Wiley, p. 8.
10. Lucey, T., 1991, Management Information Systems. DP Publications Lim., 6th ed., p. 16.

11. Fayyad, U., Piatetsky-Shapiro, G., and Smyth P., 1996. From Data Mining to Knowledge Discovery in Databases. American Association for Artificial Intelligence Magazine, Fall 1996, pp. 37-54. Available at: (http://www.kdnuggets.com/gpspubs/aimag-kdd-overview-1996-Fayyad.pdf).
12. Marakas, G., 2003. Decision Support Systems in the 21st Century. Prentice-Hall, p.328.
13. White paper, 2005. An Architecture for Enterprise Business Intelligence. MicroStrategy, Available at: http://www.microstrategy.com/Publications/Whitepapers.
14. Mueller, J-A. and Lemke, F., 2003. Self-Organizing Data Mining: An Intelligent Approach To Extract Knowledge From Data. Trafford Publishing, Canada.
15. Ivakhnenko, A. G., 1966. Group Method of Data Handling - A Rival of the Method of Stochastic Approximation. *Soviet Automatic Control*, No.13, pp. 43-71.
16. Stone, M., 1977. Asymptotics for and against cross-validation. Biometrika 64 (1) pp. 29–35.
17. Madala, H. and Ivakhnenko, A.G., 1994. Inductive Learning Algorithms for Complex Systems Modeling. CRC Press, Boca Raton, FL, USA.
18. Marchev, A. and Motzev, M., 1985. Computer Macroeconomic Models for Simulation Experiments. Systems Analysis and Simulation, Berlin, Band 28, 1985 II, pp.145-150.
19. Marchev, A., Motzev M., and Muller, J-A., 1985. Applications of The Self-Organization Procedures for Business System Models Building. Automatics, Kiev, Vol. 1, 1985, pp. 37-44.
20. Marchev, A. and Motzev, M., 1989. Principles of Multi-Stage Selection in Software Development in Decision Support Systems. Methodology and Software for Interactive Decision Support (Lecture Notes in Economics and Mathematical Systems), Springer-Verlag, 337 IIASA, pp. 181-189.
21. Motzev, M., and Marchev, A., 1984. Applications of Management Simulation Games for Student Training in Business Education, X Internationales Seminar Uber Rechnergestutzte Planspiele, Berlin.
22. Motzev, M., and Marchev, A., 1988. Multi-Stage Selection Algorithms in Simulation, Proceedings of XII IMACS World Congress, Paris, France: vol. 4, July, pp. 533-535.
23. Motzev, M., and Marchev, A., 1991. Macroeconomic Models for Simulation of the Bulgarian Economy. Systems Analysis, Models and Simulation, vol. 8.
24. Motzev, M., Marchev, A., and Muller, J-A., 1986. Modeling and Forecasting on Macro-Economic Systems Using Auto-Regressive Models., Social Management, Sofia, vol 6, pp. 77-93.
25. Motzev M., 1985. A New Approach for Simulation Models Building. XVI IFAC/ISSAGA Workshop, Alma-Ata, June.
26. Motzev, M., 2012. New Product – An Integrated Simulation Game in Business Education. Bonds & Bridges, Proceedings of the World Conference of the ISAGA, pp. 63-75.
27. Onwubolu, G. (ed.), 2009. Hybrid Self-Organizing Modeling Systems. Springer-Verlag Berlin Heidelberg.
28. Sheskin, D., 2011. Handbook of Parametric and Nonparametric Statistical Procedures. Boca Raton, FL: CRC Press, p. 109.