STATISTICAL LEARNING NETWORKS IN SIMULATIONS FOR BUSINESS TRAINING AND EDUCATION

Mihail Motzev
Walla Walla University
Mihail.Motzev@WallaWalla.edu

ABSTRACT

Statistical Learning Networks can address the common problems of Artificial Neural Networks (ANNs) such as: difficulties in interpretation of the results, the problem of overfitting, designing ANNs topology is in general a trial-and-error process and there are no rules for using the theoretical a priori knowledge in ANNs design. This paper discusses a highly automated procedure for developing Statistical Learning Networks in the form of Multi-Layered Networks of Active Neurons (MLNAN) for business simulations using the Group Method of Data Handling. MLNAN helps researchers by making business simulations development more cost-effective. All results so far show that MLNAN is able to develop reliable complex models with better overall error rates than state-of-the-art methods. This paper presents some of the results from international research done in Europe, Australia, and the United States.

Keywords: Business Simulations, Artificial Neural Networks (ANNs), Deep Neural Networks (DNNs), Statistical Learning Networks (SLNs), Multi-Layered Networks of Active Neurons (MLNAN), Group Method of Data Handling (GMDH).

INTRODUCTION

Business simulation, according to Greenlaw et al. (1962), is described as a sequential decision-making exercise structure, which is build around a model of a business operation, in which participants assume the role of managing the simulated operation. A properly designed simulation for business training would closely follow the assumptions and rules of the theoretical models within this discipline. If the model does not accurately represent the real system, then the knowledge that the students receive about real-life business is questionable. If the simulation model is not accurate and the predictions made by students are not close enough to the real -life business case, then learning will be minimal.

Deep learning attempts to model high-level abstractions in data by using model architectures composed of multiple non-linear transformations. Many of the most successful deep learning methods involve *Artificial Neural Networks (ANNs),* where *A Deep Neural Network (DNN)* is defined to be an artificial neural network with multiple hidden layers of units between the input and output layers.

DNNs make it possible to develop faster and more accurate simulation models, but they are difficult to develop and hard to understand. The difficulties with ease of development and 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 DNN is a complex nonlinear model that does not produce rules.

"One of the great things about deep learning is that users can essentially just feed data to a neural network, or some other type of learning model, and the model eventually delivers an answer or recommendation. The user doesn't have to understand how or why the model delivers its results; it just does. But some enterprises are finding that the black box nature of some deep learning models -- where their functionality isn't seen or understood by the user -- isn't quite good enough when it comes to their most important business decisions. A lack of transparency into how deep learning models work is keeping some businesses from embracing them fully…" (Burns, 2017).

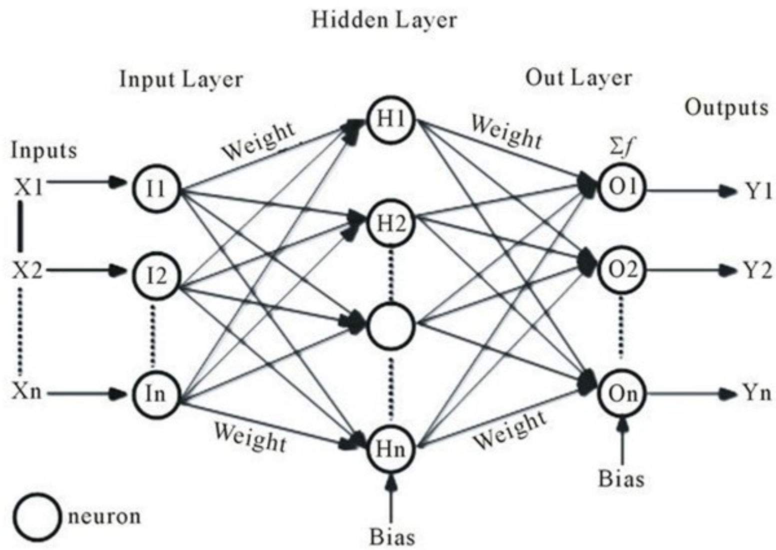
Different techniques were developed to help with this important point. *Statistical Learning Networks (SLNs)* can address the common problems of *DNNs* such as: difficulties in interpretation of the results (*DNNs* are implicit models with no explanation component by default), the problem of overfitting, designing *DNN* topology is in general a trial-and-error process and there are no rules how to use the theoretical a priori knowledge in *DNN* design (Müller and Lemke, 2003).

This paper discusses a highly automated procedure for developing *Statistical Learning Networks* in the form of *Multi-Layered Networks of Active Neurons* for business simulations using the *Group Method of Data Handling* (Madala and Ivakhnenko, 1994). It presents some of the results from international research done in Europe, Australia, and the United States.

DEEP LEARNING ARTIFICIAL NEURAL NETWORKS

According to Berry and Linoff (2000) and many other authors (Zhang et al. 1998) the basic objective of *Artificial Neural Networks (ANNs)* is to construct a model for mimicking the intelligence of the human brain in machines. Similar to the work of a human brain, ANNs try to recognize regularities and patterns in the input data, "*learn*" from experience, and then provide generalized results based on their known previous knowledge. Hence, an ANN is presented as a network of "*neurons*" organized in *layers* (see exhibit 1).

EXHIBIT 1
GENERAL EXAMPLE OF MULTILAYER FEED-FORWARD ANN



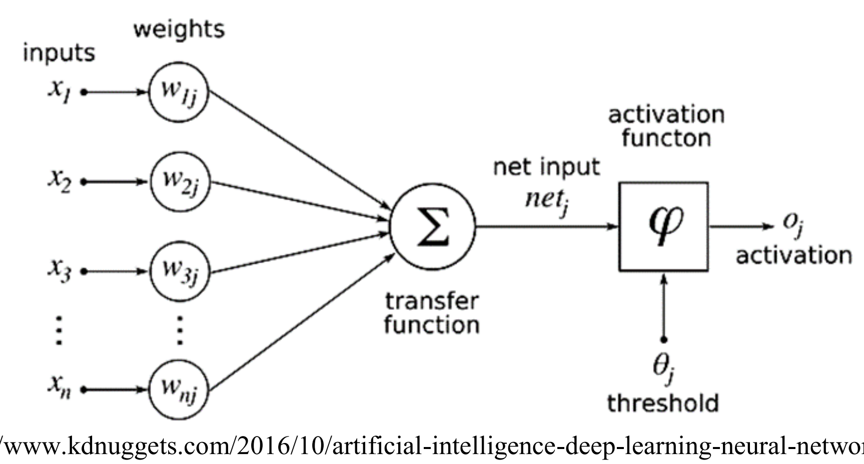
(source: http://python3.codes/neural-network-python-part-1-sigmoid-function-gradient-descent-backpropagation/)

The first layer is referred to as the *input layer* and contains the original factors or some transformation of their values. The projections form the last layer (*output layer*). Each of the inputs has its own unit, called a *network node* (a *neuron*). In general, it is not the actual values of the input variables that are *fed* into the input layer, but some transformation of those values. In simple ANNs each input unit is connected directly to the output unit with a *weight*. Inside the output unit, the input weights are combined using a *combination function* 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* (Exhibit 2). The value produced by the activation function of the output node can be the prediction or a transformation of the actual desired output.

Most ANNs have one or more additional layers of units between the input and output layers. These intermediate layers are called *hidden layers* and the units in them are *hidden units* (*hidden neurons*). 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.

With intermediate layers containing hidden neurons, the ANN becomes non-linear. This is referred to as a "*multilayer feed-forward network*" (*FNN*) where each layer of nodes receives inputs from the previous layers. By *feed-forward* we mean that data

EXHIBIT 2
EXAMPLE OF ANN COMPUTATIONAL PROCESS



(Source: https://www.kdnuggets.com/2016/10/artificial-intelligence-deep-learning-neural-networks-explained.html) enters at the input nodes and exits at the output nodes without ever looping back on itself. ANNs like this are also known as *multilayer perceptrons*. There are many ANN architectures that may include loops or ones where the inputs arrive in waves, not all at the same time.

The outputs of nodes in one layer are inputs to the next layer. The inputs to each node are combined using a weighted linear combination. The result is then modified by a nonlinear function before being output. The parameters of the functions and the weights are "*learned*" from the data.

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 ANN best approximates the underlying function. This is an *optimization problem* and most software packages for building ANN models use some variation of the technique known as *backpropagation*. The term *backpropagation* refers to any method of training an ANN that involves comparing the expected result for a given set of inputs to the output of the network (the *activation threshold*) during a training run, and feeding that difference back through the network to adjust the weights. The weights are selected in the neural network framework using a "*learning algorithm*" that minimizes a "*cost function*" such as *Mean Squared Error (MSE)*.

As mentioned above, a *Deep Neural Network* (*DNN*) is an artificial neural network with multiple hidden layers of units between the input and output layers. The neural network-based deep learning models, by default, contain a great number of layers, since they rely more on optimal model selection and optimization through model tuning. ANNs and the deep learning techniques are considered as some of the most capable tools for solving very complex problems. They are data-driven and self-adaptive in nature, i.e. there is no need to specify a particular model form or to make any a priori assumption about the statistical distribution of the data. Perhaps their greatest advantage is the ability to be used as an arbitrary function approximation mechanism that "learns" from observed data. According to Hornik et al. (1989) ANNs are universal functional approximators and can deal with situations where the input data are erroneous, incomplete, or fuzzy.

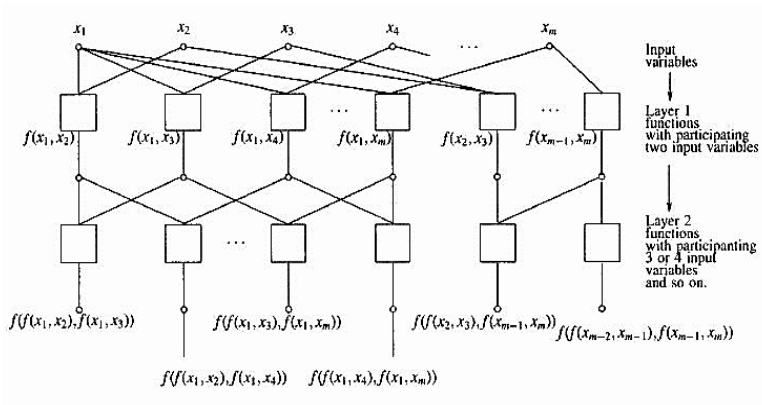
It is worth pointing out, however, that to have an extremely robust ANN (DNN or other type ANN) the model, cost function, and learning algorithm must be appropriately selected. Like typical ANNs, many issues can arise with DNNs if they are naively trained. Two common issues are overfitting and computation time, due to increased model and algorithmic complexity, which results in very significant computational resource and time requirements. Other important questions should be considered as well, such as "How can secondary data series be inferred for the network generating process", or "How many input nodes for the ANN are appropriate (i.e. what is the order of the model)". In addition, concerning the ANNs architecture, other questions should be addressed such as "How many hidden nodes are appropriate" or "Which is the best activation function in any given instance?"

STATISTICAL LEARNING NETWORKS AND SELF-ORGANIZING DATA MINING

Statistical learning theory deals with the problem of finding a predictive function based on data. In general, it is a framework for machine learning drawing from the fields of statistics and functional analysis (Hastie et al. 2017, Mohri et al. 2012). One of the most important group of techniques in *Statistical Learning Networks* (*SLNs*) is *Self-Organizing Data mining* (Müller and Lemke, 2003).

Proven to be one of the most successful methods in SLNs is the *Group Method of Data Handling* (*GMDH*). GMDH (Madala and Ivakhnenko, 1994) is an inductive approach to model building based on self-organization principles. It is also referred to as "Polynomial Neural Networks" or "Statistical Learning Networks" (see www.gmdh.net). In GMDH algorithms, models are generated adaptively from data in the form of an ANN 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 (Exhibit 3).

EXHIBIT 3
GENERAL SCHEME OF GMDH SELF-ORGANIZING MODELING ALGORITHM



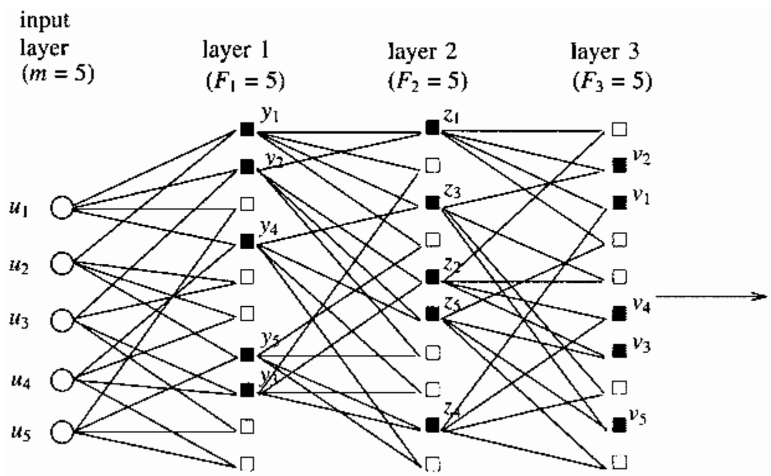
(Source: Madala & Ivakhnenko, 1994, p. 8)

GMDH algorithms address successfully the ANN problems when the user estimates ANN structure by choosing the number of layers and the number and transfer functions of nodes, which requires not only knowledge about the theory of ANNs, but also knowledge of the modeled object's nature. GMDH algorithms pick out knowledge about the object directly from data sampling. GMDH is an inductive sorting-out method, which has advantages in case of rather complex objects and when there is no definite theory.

The biggest drawback of a typical ANN 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, general ANNs are not a good choice. Müller and Lemke (1995) make comparisons and point out that in distinction to ANNs, the results of GMDH algorithms are explicit mathematical models generated in a relatively short time on the basis of even small samples.

The well-known problems of an optimal (subjective) choice of the neural network architecture are addressed in the GMDH algorithms by means of an adaptive synthesis (objective choice) of the architecture. GMDH algorithms could be used to estimate networks of the right size with a structure evolved during the estimation process to provide a parsimonious model for the particular desired function. There are many different ways to select the right order of a GMDH-type ANN. The most popular one is called multilayered inductive procedure (Exhibit 4). It is equivalent to an ANN with polynomial activation function of neurons (Polynomial NN).

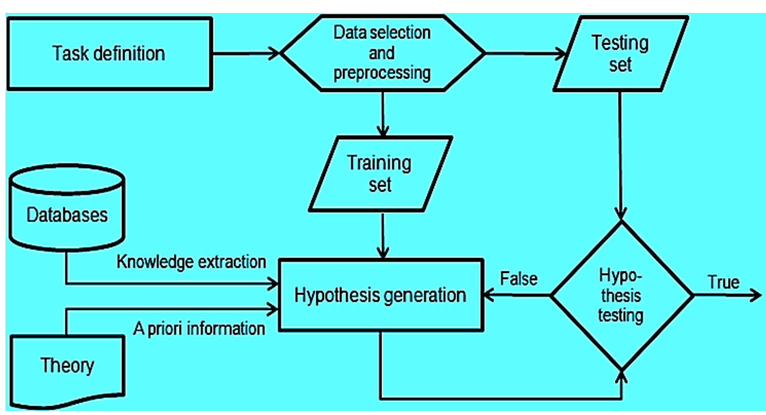
EXHIBIT 4
GMDH ITERATIVE PROCEDURE –
A MULTILAYERED ACTIVE NEURON NEURAL NETWORK



(Source: Madala & Ivakhnenko, 1994, p. 32)

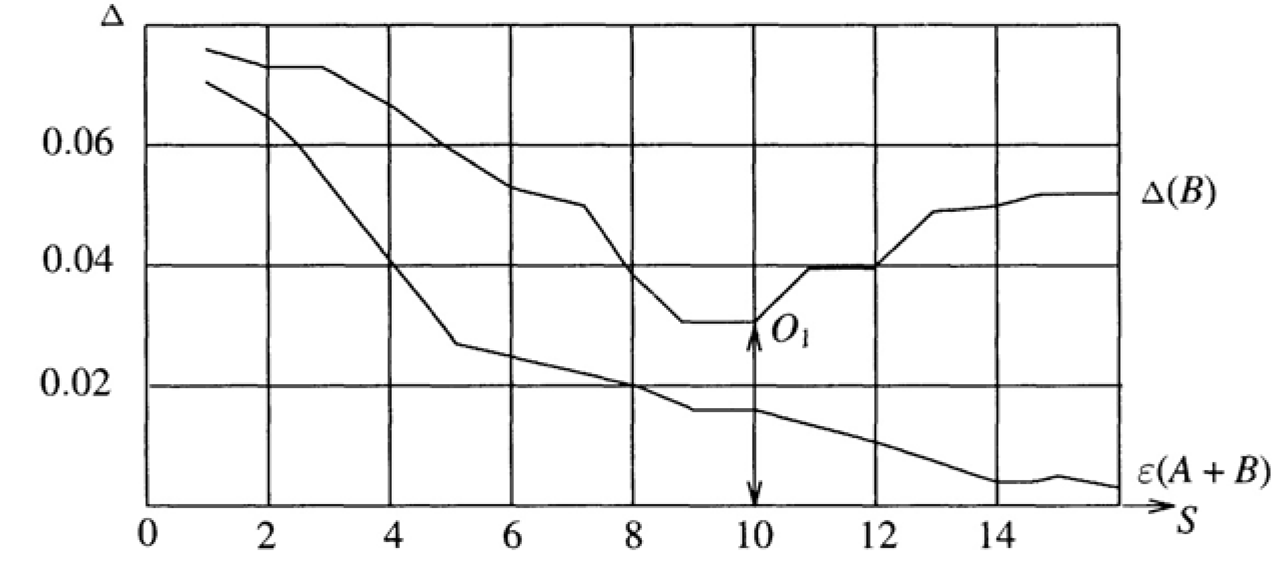
As mentioned above, deep learning and neural network algorithms can be prone to overfitting. GMDH algorithms address the problem of overfitting with the cross-validation technique (Exhibit 5). Following Beer (1959), only the external criteria (Exhibit

EXHIBIT 5
SELF-ORGANIZING MODELING
USING CROSS VALIDATION WITH A PRIORI INFORMATION



(Source: Motzev, 2016, p. 410)

EXHIBIT 6
MODEL ERROR Δ(B) CALCULATED ON NEW INDEPENDENT INFORMATION (B) VS. OVER-FITTING I.E. TRADITIONAL ERROR Ε(A+B) CALCULATED ON WHOLE DATA SET (A+B)



1 Variation in least square error $\varepsilon(A+B)$ and error measure of an "external complement" $\Delta(B)$ for a regression equation of increasing complexity S; O_1 is the model of optimal complexity

(Source: Madala & Ivakhnenko, 1994, p.11)

6), calculated on new, independent information can produce the minimum of the sorting-out characteristic (model error). Such algorithms, combining in a powerful way the best features of ANNs and statistical techniques, identify the entire model structure in the form of a network of polynomial functions, differential equations and others. Models are selected automatically based on their ability to solve a task such as approximation, identification, prediction, and classification.

MULTY-LAYERED NETWORKS OF ACTIVE NEURONS FOR BUSINESS SIMULATIONS

Self-organizing GMDH algorithms are similar in many respects to an ANN of active neurons. The goal is to enhance the accuracy in achieving the assigned task through a better use of input data by combining many neurons into a network. In the beginning, an exhaustive search is applied to determine the number of neuron layers and the sets of input and output variables for each neuron. The activation threshold is the minimum of the cross-validation criterion (*Mean Squared Error*), which identifies the variables for which it is advantageous to build a neural network. Active neurons are able, during the self-organizing process, to estimate which inputs are necessary to minimize the given objective function of the neuron. The number of hidden neuron layers is also self-organized, i.e. it is defined through the process of multi-stage selection procedure.

The *Multi-Layer Net of Active Neurons (MLNAN)* algorithm proposed in this paper works as a net of complex active neurons that choose the effective inputs and their corresponding coefficients in a process of highly automated self-organization (Motzev & Marchev, 1988). The MLNAN model with multiple inputs (x_j) and one output (Y) is a subset of components of the base function (Exhibit 7):

EXHIBIT 7
BASE FUNCTION IN MLNAN

$$Y(x_1, x_2 \dots x_n) = a_0 + \sum_{i=1}^{m} (a_i f_i) \qquad (1)$$

- where f_i are functions dependent on different sets of inputs (i=1, 2 … m);
- x_j (j=1, 2 …n) are the inputs at the first layer (predictors)
- a_0 is the constant term;
- a_i are the unknown coefficients and
- m is the number of the base function components.

To find the best solution MLNAN analyzes various component subsets (*partial models*) of the base function (1) where the unknown coefficients a_i are estimated by the *Least Squares* (*LS*) method. MLNAN algorithms gradually increase the number of partial model components as shown on Exhibit 4 until finding a model structure with optimal complexity selected by the minimum value of the external (cross-validation) criterion. In this algorithm, first, the partial models on a lower layer are estimated and the corresponding intermediate outputs are computed. These outputs become inputs of the next layer where the new partial models are estimated, and so on. Finally, if additional layers provide no further improvements to the accuracy of the model, the network self-organization stops.

At the end, a given number of partial models with similar accuracy from the last layer are selected and the base function (1) for each of them is restored using an automated backward tracking algorithm, providing "*freedom of choice*" to the decision maker, who can make a selection within a set of alternative good models.

Many researchers have applied successfully similar MLNAN algorithms in different areas, including business simulations and model-based business games for training and education. For example, a team of researchers applied a working prototype of the MLNAN in developing a series of increasingly complex simulation models (linear systems of simultaneous equations) of the national economy with a very high level of accuracy (Marchev et al. 1985). Other successful applications in predictive modeling with similar GMDH algorithms were done in Germany (Müller and Lemke 2003), the United States (Klein et al. 1980) and other countries. Comparisons of the results from different business simulations and predictions, made with these models, show almost insignificant differences in their errors and a very high level of accuracy (Motzev, 2016).

Another area of MLNAN applications is in model-based simulation games for business training and education. An existing business game "National Economy", which contains a model developed with the general multiple regression analysis, was improved significantly after applying MLNAN. The same data and set of variables were used to build a new model using the MLNAN algorithm. The new model has much better accuracy (more than five times smaller MSE%, see Exhibit 8) and thus provides a more reliable base for simulations and what-if analysis.

EXHIBIT 8
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: Motzev & Lemke, 2015)

More recently, the MLNAN was used in the process of developing the "NEW PRODUCT" game series, which is an integrated, role-playing, model-based simulation game designed for the purposes of business training and education (Motzev, 2012). The latest versions of the game cover all major stages in the process of new product planning and development, production and operations management, sales and marketing. It could be used not only as an educational tool for teaching business, but it may also be carried out for business training in general management, production management, inventory and stock control, and small business management.

CONCLUSIONS

The benefits of utilizing an MLNAN algorithm in business simulations are extensive. MLNAN provides opportunities to shorten the design time and reduce the cost and the efforts in model building. At the same time, MLNAN makes it possible to develop even complex models reliably with low overall error rates. Ever increasing model accuracy helps researchers analyze problems more precisely, which leads to deeper and better understanding. Models with higher accuracy simply produce better predictions and support managers in making better decisions that are much closer to the real-life business case.

It is easy to prove the above statement. For example (Motzev, 2016, pp. 317-318) increasing the model accuracy in forecasting equation, i.e. reducing its relative MSE from 7.5% to 2.44% means that the prediction interval at 99% level of significance will be reduced from 45% (MSE relative to the mean) to 14.64% (MSE relative to the mean), that is 3.07 times.

Increasing model accuracy provides many other benefits. The MLNAN makes it possible to improve the decisions that students make during model-based business games. As mentioned above, if the game model does not accurately represent the real system, then the knowledge that the students receive about real-life business is questionable. It is easy to prove that the learning is minimal if the simulation model is not accurate and the predictions made by students are not close enough to the real-life business.

In summary, MLNAN helps researchers by making business simulations and model-based business games development more cost-effective. All results so far show that it is able to develop even complex models reliably with better overall error rates than most of the current methods.

Of course, data fitting does limit the value of such algorithms somewhat and decision makers have to decide the overall importance of the model outputs. For example, the future cannot always be predicted based on history. However, the MLNAN does provides processed data that are needed in the business context. The extracted information is useful to a business in making decisions that create value or decisions that predict market behavior in a way that provide a competitive advantage.

List of acronyms used:

ANNs - Artificial Neural Networks
FNN - Feed-Forward Network
LS - Least Squares
MSE - Mean Squared Error
DNNs - Deep Neural Networks
GMDH - Group Method of Data Handling
MLNAN - Multi-Layered Networks of Active Neurons
SLNs - Statistical Learning Networks

REFERENCES

Beer, S. (1959) *Cybernetics and Management*, London: English University Press, p. 280.

Berry, M. & Linoff, G. (2000) *Mastering Data Mining*, Wiley.

Burns, Ed. (2017). Deep learning models hampered by black box functionality. *TechTarget, Advanced Analytics.* Retrieved from http://searchbusinessanalytics.techtarget.com/feature/Deep-learning-models-hampered-by-black-box-functionality?utm_content=control&utm_medium=EM&asrc=EM_ERU_82565194&utm_campaign=20170914_ERU%20Transmission%20for%2009/14/2017%20(User Universe:%202429004)&utm_source=ERU&src=5669449

Greenlaw, P., Herron, L., Rawdon, R. (1962). *Business Simulation in Industrial and University Education* Prentice-Hall.

Hastie, T., Tibshirani, R., Friedman J. (2017) *The Elements of Statistical Learning: data mining, inference, and prediction* , New York: Springer.

Hornik, K., Stinchcombe, M., White, H. (1989) Multilayer feed-forward networks are universal approximators. *Neural Networks* 2, 359–366.

Klein, L., Müller, J-A., Ivakhnenko, A.G. (1980) Modeling of the Economics of the USA by Self-organization of the System of Equations, *Soviet Automatic Control* 13, 1, 1-8.

Madala, H., Ivakhnenko, A.G. (1994) *Inductive Learning Algorithms for Complex Systems Modelling,* CRC Press Inc.

Marchev A., Motzev M. and Müller, J-A. (1985) Applications of Self-Organization Procedures for Business System Models Building", *Automatics*, Vol. 1, 37-44.

Mohri, M., Rostamizadeh, A., Talwalkar, A. (2012) *Foundations of Machine Learning,* The MIT Press.

Motzev, M. (2012) New Product – An Integrated Simulation Game in Business Education. In: *Bonds & Bridges. Proceedings of the World Conference of the ISAGA*, 63-75.

Motzev, M. (2016) *Business Forecasting: A Contemporary Decision Making Approach,* Eudaimonia Productions.

Motzev, M. & Lemke, F. (2015) Self-Organizing Data Mining Techniques in Model Based Simulation Games for Business Training and Education. *Vanguard Scientific Instruments in Management*, Vol. 11.

Motzev, M. & Marchev, A. (1988) Multi-Stage Selection Algorithms in Simulation. In: *Proceedings of XII IMACS World Congress*, Paris: vol. 4, 533-535.

Müller, J-A. & Lemke, F. (2003) *Self-Organizing Data Mining: An Intelligent Approach To Extract Knowledge From Data*, Trafford Publishing.

Müller, J-A. & Lemke, F. (1995) Self-Organizing modelling and decision support in economics. In: *Proceedings of the IMACS Symposium on Systems Analysis and Simulation*, Gordon and Breach Publ., 135-138.

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