ISSN 1392 – 124X INFORMACINĖS TECHNOLOGIJOS IR VALDYMAS, 2004, Nr. 2(31)

DECISION RULES EXTRACTION FROM NEURAL NETWORK: A MODIFIED PEDAGOGICAL APPROACH

Darius PLIKYNAS

Vilnius Management College
Basanavičiaus 29a, Vilnius, Lithuania

Abstract. Neural network (NN) methods are sometimes useless in practical applications, because they are not properly tailored to the particular market's needs. We focus thereinafter specifically on financial market applications. NNs have not gained full acceptance here yet. One of the main reasons is the "Black Box" problem (lack of the NN decisions explanatory power). It remains an open issue for the top and middle managerial level. There are though some NN decisions rule extraction methods like decompositional, pedagogical or eclectic, but they suffer from low portability of the rule extraction technique across various neural net architectures, high level of granularity, algorithmic sophistication of the rule extraction technique etc. The author propose to eliminate some known drawbacks using an innovative extension of the pedagogical approach. The idea is exposed by the use of a widespread MLP neural net (as a common tool for the decisions' space fragmentation in the financial problems' domain) and SOM (for the input data space clusterization). The feedback of both nets' performance is related and targeted through the iteration cycle by achievement of the best matching between the decision space fragments and input data space clusters. Three sets of rules are generated algorithmically or by fuzzy membership functions. Empirical validation of the common financial benchmark problems is conducted with an appropriately prepared software solution.

Keywords: Neural Networks; Decisions Reasonong; Information Extraction.

1. Indroduction

Rule extraction techniques seek to clarify to the user how the network arrived at its decision. Generated rule quality refers to five primary classification criteria [1, 2], *viz* a) the *expressive power* or the rule format of the extracted rules; b) the *quality* of the extracted rules; c) the *translucency* of the view taken within the rule extraction technique of the underlying neural network; d) the *complexity* of the rule extraction algorithm; e) the *portability* of the rule extraction technique across various neural network architectures (i. e. the extent to which the underlying NN incorporates specialized training regimes). In this research, we will mainly focus on three criteria: expressive power, complexity and portability of the proposed rule extraction algorithm.

The essential task of using NNs for inductive inference is to transform the knowledge embodied within the architecture and weights of the trained network into a set of symbolic (for example, propositional if-then) rules. A number of different strategies have been developed for performing this task [3]. Up to date two distinct approaches – decompositional and pedagogical – are distinguished in the mainstream. Andrews et al. [4] also propose the third category, which they labeled as "eclectic" to accommodate elements of both mainstream approaches. Decompositional approach is aimed to search for combinations of input values which, when satisfied, cause a given (hidden or output) unit within the NN to become "active" irrespective of the state of other inputs to the unit [5]. Rules are extracted at the level of individual hidden and output layer units. An alternative pedagogical approach treats the trained NN as a "Black Box" (see [6]). Here extracted rules describe global relationships between inputs and outputs; no analysis of the detailed characteristics of the NN itself is undertaken.

In contrast to the decompositional approaches, the motive in the pedagogical approaches is to view the trained NN at the minimum possible level of granularity. The focus is then on finding rules that map the NN inputs directly into outputs. Validity Interval Analysis (VIA) algorithm [7] and the Rule-Extraction-As-Learning technique (REAL) [8] are two examples of what might be historically considered as the epitome of pedagogical approaches to extracting rules from a trained ANN.

Tickle et al. [2], Baesens et al. [1] and other studies concluded that, at this stage, no compelling evidence has emerged which mandates the use of a particular type of NN architecture and/or a particular type of rule extraction technique in a given class of problem domains. Publications show that considerable scope still exists for synthesizing methodologies and techniques which are applicable across a broad spectrum of NN implementations and architectures.

Pedagogical or learning based rule-extraction techniques are chosen in this research, because they make no assumptions about the underlying NN architecture and are offered as an efficient alternative to decompositional algorithms [5]. Particularly considerable potential appears to exist in exploiting the pedagogical approach to rule extraction from trained NNs to develop a set of techniques for financial problem domain. The author is looking for the extension of the learning based rule-extraction technique by adopting a new approach, which would make it more suitable for financial decisions making. The idea is briefly described in further sections.

This paper is organised as follows. In sections 2 and 3, the basic concepts of the proposed method are briefly explained. Section 4 presents the matching algorithm between the input space clusters and solution space fragments. NN's decision reasoning (extracted rules sets) is presented in section 5. Section 6 is dedicated for the experimentation set up and benchmarking. Conclusions are drawn in section 7.

2. Premises and confines

First of all, we will choose a standard financial problem domain and formalize it. This may be credit risk management, portfolio risk-profit operational management, insurance pay-off management etc. So, let us assume that the financial problem domain $\Omega(\Omega^{in}; \Omega^{out})$ is characterized by (1) sub domain Ω^{in}, which consists of the set of input data space vectors $\{I^i_n\}$ (where n denotes the input space dimensionality and i=[1, k] indicates the input data vector); (2) sub domain Ω^{out}, which consists of the set of output data space vectors $\{O^i_m\}$ with appropriate output dimensionality m.

NN is used for mapping given input space onto the desirable output space (NN makes decisions in the particular financial problem domain). Our goal further consists from investigating the mapping function Φ

$$\Phi(\{I^i_n\})\rightarrow\{O^i_m\} \qquad (1)$$

Multilayer perceptron (MLP) network serves as an universal approximator, which learns how to relate the set of the input space vectors $\{I^i_n\}$ to the set of output (solutions) space vectors $\{O^i_m\}$. We have chosen MLP because it is widely used in the finance sector. Transformation function Φ is then characterized by the MLP structural parameters like weights' matrix W, biases B, number of neurons N, topology structure T, learning parameters L

$$\Phi = \Phi(W, B, N, T, L) \qquad (2)$$

We are not going to investigate structural parameters in the decompositional manner, where rules are generated describing the discretised hidden or output unit activation values in terms of the original inputs [9]. Our method contains a different assumption, i.e. there is no need for the decompositional approach, because following the analogy of biological NNs, we have to model input/output interaction between specialized NNs (see [7, 8]). This means that one NN may be specialized in recognizing and clustering the input space while another net may use the former NN results for mapping the input space clusters onto the output space clusters. The third net may optimize the matching of both NNs' results and so on. The result is an autonomous intelligence capable not only to recognize the problem, but also to make decisions, justify them and improve the performance from the past experience. The autonomy also means very high portability over different problem domains.

This approach of the possible interconnection mechanism between different NNs is further briefly described by using a model of the two (SOM and MLP) different neural nets:

1. SOM – self organizing neural network

It searches input data space Ω^{in} for some distinct features and makes the set of clusters $\{C^j_n\}$ out of the set of input data vectors $\{I^i_n\}$ (where $j<i$)

$$\bigcup \;, \{C^j_n\}\subseteq\{I^i_n\} \qquad (3)$$

2. MLP – multilayer perceptron neural network

This is supervised learning. MLP gains experience by learning how to relate the input space vectors $\{I^i_n\}$ to known output vectors $\{O^i_m\}$. Now we parameterize expression (1)

$$\Phi_{W, B, N, T, L}\,(\{I^i_n\}) \rightarrow \{O^i_m\} \qquad (4)$$

Both NNs contribute to the transformation process of a prior data to the decisions' data

$$\Phi \rightarrow \Phi_{SOM} + \Phi_{MLP} \qquad (5)$$

The whole process of the decision's rules generating could be described as it is shown in Figure 1.

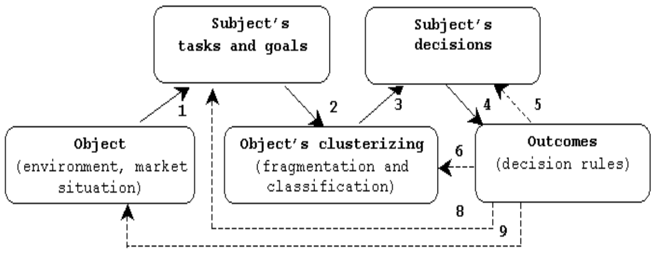
Subject's tasks and goals
Subject's decisions
1
2
3
4
5
Object (environment, market situation)
Object's clusterizing (fragmentation and classification)
6
Outcomes (decision rules)
8
9

Figure 1. Scheme of the decision rules generating process

The main stages and iteration cycles are (see Figure 1)

1) 1→2→3→4: there are no cycles nor rules,
2) 1→2→3→4 + ⟨5→4⟩ cycle: search for the optimal decision,

3) $1 \to 2 \to 3 \to 4$ + $\langle\langle 5 \to 4\rangle$ and $6 \to 3 \to 4\rangle$ cycles: search for the optimal decision and clusterizing,
4) $1 \to 2 \to 3 \to 4$ + $\langle\langle\langle 5 \to 4\rangle$ and $6 \to 3 \to 4\rangle$ and $8 \to 2 \to 3 \to 4\rangle$ cycles: search for the optimal decision, clusterizing and task (verification),
5) $1 \to 2 \to 3 \to 4$ + $\langle\langle\langle\langle 5 \to 4\rangle$ and $6 \to 3 \to 4\rangle$ and $8 \to 2 \to 3 \to 4\rangle$ $9 \to 1 \to 2 \to 3 \to 4\rangle$ cycles: search for the optimal decision, clusterizing, task (verification) and object specification.

Here are some distinct modules, which embrace different parts of the experience and decision rules generating process (see (4)). The stages and cycles form confines for the proposed algorithm. The next section discusses the issue.

3. General scheme

Now we are going to construct the general scheme. As mentioned earlier, the proposed method explores interaction of two different types of NNs: MLP and SOM. The MLP plays a central role, because it is responsible for learning how the input variables are related to output decisions. In the overall research scheme (see Figure 2), MLP is indicated as Module I (backpropagation NN with learning data). The goal of Module I is the projection of the input data space to the output solution space (see Figure 2, steps 1-2). This module is taught to memorize optimal solutions from the given learning and testing data set. In a biological brain analogy: MLP gets experienced. Our objective is to extract decision rules from the MLP.

Meanwhile self-organizing neural net (SOM) is indicated as Module II: Kohonen type self-organizing NN [10]. This net is responsible for clusterizing of the input data space (see Figure 2, steps 3 and 4). There are specialized NN methods, which are capable of doing this task (e. g. competitive Kohonen or bias learning, self-organizing maps, learning vector quantization networks etc). Kohonen type NN method will stay as default for the further reference. Given a biological brain-like analogy, it assorts objects to different groups according to grouping criteria (input variables). However, unlike the human brain, our SOM net has no predefined priorities for the input space best grouping criteria. Therefore, this process is exposed to mismatching.

Before making the final projection of the clustered input data space $\{C^j_n\}$ to the solution space fragments $\{F^s_m\}$ (m - indicates dimensionality, $s = [1, e]$)

$$\bigcup \quad , \{F^s_m\} \subseteq \{O^i_m\} \qquad (6)$$

we have to be sure about best matching of both. Solution space fragmentation is as precise as an investigator wants, but for the input space clustering this is not so. We have not to forget the fact that input space clusterization is done by SOM network, which is not coherently bounded up to the MLP performance. Therefore SOM clustering process (through the feedback relation) has to be directed for the search of the best representation of the factual input clusters. If we define some conformance criteria for the optimal choice, then we will make clustering process iterate until those matching criteria are met.

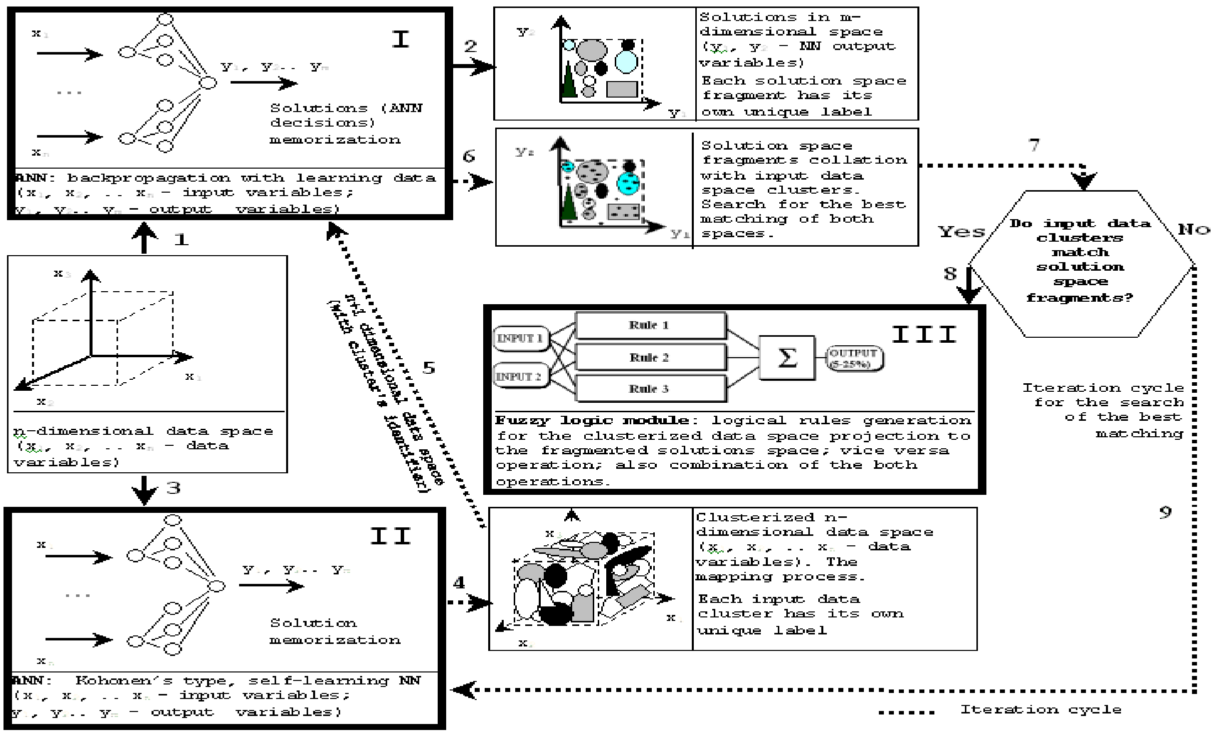
I
x1
xn
y1, y2.. ym
Solutions (ANN decisions) memorization
ANN: backpropagation with learning data (x1, x2, .. xn – input variables; y1, y2.. ym – output variables)
2
y2
y1
Solutions in m-dimensional space (y1, y2 – NN output variables)
Each solution space fragment has its own unique label
6
Solution space fragments collation with input data space clusters. Search for the best matching of both spaces.
7
Yes
No
Do input data clusters match solution space fragments?
8
1
x3
x1
x2
n-dimensional data space (x1, x2, .. xn – data variables)
n+1 dimensional data space (with cluster's identifier)
5
INPUT 1
INPUT 2
Rule 1
Rule 2
Rule 3
Σ
OUTPUT (5-25%)
III
Fuzzy logic module: logical rules generation for the clusterized data space projection to the fragmented solutions space; vice versa operation; also combination of the both operations.
Iteration cycle for the search of the best matching
3
II
Solution memorization
ANN: Kohonen's type, self-learning NN (x1, x2, .. xn – input variables; y1, y2.. ym – output variables)
4
Clusterized n-dimensional data space (x1, x2, .. xn – data variables). The mapping process.
Each input data cluster has its own unique label
9
Iteration cycle

Figure 2. Overall scheme for the input data space clusterizing (SOM) and projection (MLP) to the matching solution space

Suppose $\{C^{opt}{}_{n}\}$ a priori stands for the set of optimally clustered input data space. Then through the iteration cycle

$$\longrightarrow \qquad \text{until}$$

$$\sum \tag{7}$$

where ε is optional, we could arrive at the optimal clustering option. This problem tackled by iteration cycle (steps 4, 5, 6, 7, 9 and 4, see Figure 2) could resolve the issue. Module II makes n-dimensional input data space clustered: SOM finds complex relations between data points and groups them according to the similar features (steps 3 and 4). All data points in the same cluster get the same "cluster label". Initial n-dimensional input data space becomes then n+1 – dimensional data space (step 5). Then MLP (Module I) analyzes enriched data space and produces fragmented solution space $\{F^{s}{}_{m+1}\}$ with m+1 dimensionality (step 6). Additional dimension arises regarding the fact that each input data cluster has its own unique label, which is transmitted to all cluster elements.

If input data clusters match solution space fragments (see step 7), then the rule extraction mechanism starts (see step 8, Module III in Figure 2). Otherwise, the iteration cycle turns round until the best matching between the data clusters and solution space fragments appears (steps 4, 5, 6, 7, 9, 4). But originally we do not know $\{C^{opt}{}_{n}\}$ yet. So, *s*ome additional criteria for the best matching should be formulated. This type of algorithm will be drawn in the further section.

4. Matching algorithm

We have formulated matching problem between the data clusters and solution space fragments in the previous section (see (7)). Now follows the search for the solution, e. g. for $\{C^{opt}{}_{n}\}$.

Again let us refer to the biological neural nets. After the input data space clustering and solution space (decisions) fragmentation biological neural nets memorize both outcomes [11]. For the purpose of simplicity, let us assume that in our model the data matrix represents single memory storage. Following biological neurons' behavior our initial data matrix $D_{k\,\mathrm{x}\,n}$ after the input space clusterizing transforms into $D_{k\,\mathrm{x}\,(n+1)}$. The row index k indicates the total number of the input space vectors and the column index n - dimensionality of the input space or the total number of data attributes, see (1). An additional attribute identifies input space cluster (each input data cluster has its own unique label).

Biological neurons also memorize solution space fragments. So, upon termination of the learning process each data vector gets a mark of the appropriate solution space fragment. This again is reflected in the data matrix $D_{k\,\mathrm{x}\,(n+2)}$ by adding another attribute. The latter data matrix clears the path for employment of any known rules' extracting techniques like visual decision trees [1], symbolic binary or fuzzy rules [9, 12]. But there is a weak chain - additional attribute, which identifies the input space clusters in our model (SOM clustering, see (3)). This is because of uncertainty of clustering results and their best matching to the outcomes (MLP decisions).

Through the iteration cycle (see (7)) we expect to find the best clusterizing result $\{C^{opt}{}_{n}\}$. In each iteration cycle SOM network creates a new representation of the input space clusters. The author proposes (in every iteration) for every input space vector to check out whether an appropriate cluster attribute belongs to the certain solution fragment or not. This is possible, because we have matrix $D_{k\,\mathrm{x}\,(n+2)}$. But checking should be handled for each cluster separately. A suitable algorithm might be described in steps as follows:

1. Start iteration process by choosing one cluster, e.g. $C^{1}{}_{n}$ from the set of clusters $\{C^{j}{}_{n}\}$, which are uniquely defined by SOM for this particular iteration cycle (see (3)).
2. For the set of input space vectors $\{I^{i}{}_{n}\}$ from the given cluster $C^{1}{}_{n}$ check to which solution fragment $\{F^{s}{}_{m}\}$ they belong and memorize it $\{I^{i}{}_{n}\} \rightarrow \{I^{i}{}_{n+2}\}$ (we store data in the data matrix $D_{k\,\mathrm{x}\,(n+2)}$).
3. Find solution fragment from the set $\{F^{s}{}_{m}\}$ (see (6)), which gets maximum input vectors from the given cluster $C^{1}{}_{n}$,

$$\max_{S_P}\left(\sum_{s=1}^{e}\sum_{1}^{count\,C^1}\left(\text{if}\,(\text{Altr.number}(n+2)\,\text{from}\,I^{C^1}_{n+2}) = s\right.\right. \tag{8}$$
$$\left.\left. then\,S_P = i+1\right)\right)$$

   where S_p indicates the number of input vectors targeted to the particular solution fragment (*count* notation means counting the total number of input vectors for the given cluster). Formula (8) indicates the most targeted solution space fragment. We assume that only this solution fragment is related to the given input cluster.
4. Calculate relative measure R_j for the estimation of fitness between the given input cluster and targeted solution space fragment

$$R_j = \frac{\max S_P}{\sum_{s} S_P} \tag{9}$$

5. Continue steps from 1 to 4 for all clusters sequentially. Evaluate the total fitness R_{iter} for the whole set of clusters relative to the solution fragments

$$R_{iter} = \sum_{j} \frac{R_j}{C^{j}_{n}} \tag{10}$$

6. Continue steps from 1 to 5 until maximum number of user-defined iterations is reached. Select C^{opt}_{n} which represents the best match between

clustered input data space (SOM range) and solution space fragments (MLP range)

$$C_n^{opt} \Leftrightarrow R_{iter}^{opt} = \max\left(R_{iter}\right) \qquad (11)$$

The algorithm described above relates two distinct input and output spaces in an optimal way. The author admit a possibility for more accurate definition of matching between two spaces. Mainly it concerns steps 3-5, where Euclidean distances might be calculated for each input vector point in the solution space [10].

5. NN's decisions reasoning

In order to make the discussion more obvious, let us assume that the following such specific sets of rules (G1 and G2) in the hypothetical financial problem domain were *a priori* determined, see Figure 3:

Set G1. If less then 20% of all data points from the given data cluster influence decisions in the particular solution fragment, we'll have feeble matching; if less then 40%, but more then 20% - good matching; if more then 40% - excellent matching.

Set G2. If less then 35% of all data points in the given solution fragment belong to the particular cluster, we'll have feeble population density for this cluster; if more then 35%, but less then 65% - we'll have good population density for this cluster; if more then 65% - we'll have excellent population density for this cluster.

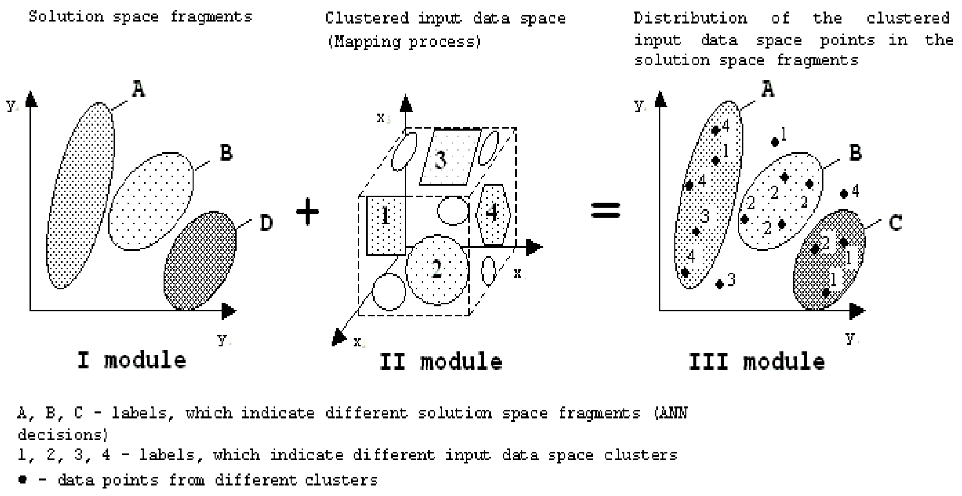
Solution space fragments
Clustered input data space (Mapping process)
Distribution of the clustered input data space points in the solution space fragments
A
B
D
C
1
2
3
4
x
y
+
=
I module
II module
III module
A, B, C - labels, which indicate different solution space fragments (ANN decisions)
1, 2, 3, 4 - labels, which indicate different input data space clusters
• - data points from different clusters

Figure 3. Solution space fragments (MLP decisions) collation with input data space clusters (SOM clustering)

After application of G1 and G2 sets of rules to the case shown in Figure 3, we obtain the following results displayed in Table 1.

Table 1. Results of matching between the input clusters and solution fragments (according to G1 and G2 sets of rules)

Fragment's label	**Cluster's label**							
	1 [%]		2[%]		3[%]		4[%]	
	G1	*G2*	*G1*	*G2*	*G1*	*G2*	*G1*	*G2*
A	25	20	0	0	50	20	75	60
B	0	0	80	100	0	0	0	0
C	50	33	20	33	0	0	0	0

The fuzzy logic module makes an additional set of rules, targeted to define more specific relationships of both matched spaces [13]. MLP neural network decisions are ready for the reasoning now. We have generated three sets of rules algorithmically or by use of fuzzy membership functions:

1. For each data cluster the appropriate solution space fragment or their combinations are assigned (search for the MLP decisions settled by a particular data cluster). For example, the input data from the cluster labeled as 1 (see Table 1: G1 column from the first label) with probability 75% will influence decisions A (25% - good matching) and C (50% - excellent matching).

2. For each solution fragment the appropriate data cluster or their combinations are assigned (search for the data clusters, which influence the particular MLP decision), see Figure 3. For example, fragment A (see Table 1: row A and column G2) is 100% filled by the clusters 1 (20% - feeble population density), 3 (20% - feeble population density) and 4 (60% - good population density).

3. The combination of the first and the second MLP rule sets.

For instance, fragment C is excellently matched (50%) by the data cluster 1 (according to the first rule), but fragment C is poor populated by this cluster (33%, according to the second rule). Therefore, the fuzzy logic module generates an additional set of rules, targeted to define more specific relationships for both matched spaces.

6. Experimental validation

Experimentation has some constraints concerning input and output data sets, i.e. some prewhitening is needed. MLP input data set should be continuous, represented by normalized input variables. Otherwise, it will not fit into the MLP and SOM learning terms [14]. Highly mutually correlated or unrelated attributes are also pruned down.

Solution space specification depends on the nature of the task. It may concern recognition, classification, prognosis, approximation and other goals set for various financial domain problems [15]. Solution space is characterized by output variables such as 1) qualitative (to buy or not to buy; good or bad investment, ratings etc), 2) quantitative (various relative or absolute measures).

This method strives for self-learning flexibility, independence from the data structures and real time execution. Consequently, data gathering, processing and logical rules extraction works softly and quickly on the integrated software solution. Therefore, MATLAB v6.0 software package was chosen, see Figure 4. It has ready to use toolboxes: SIMULINK, Neural network toolbox, Fuzzy logic toolbox.

Because of the integrated nature of MATLAB's environment, specialized tools have been created to customize the method by composing (description with user-written M-files) earlier mentioned toolboxes or by adding others, such as the Control System or Optimization Toolbox, to mention only a few of the possibilities.

The main part – SIMULINK, where the model in a block diagram simulation environment could be easily tested. At each step, SIMULINK computes new values for the system's inputs, states, and outputs and updates the model to reflect the computed values (see Figure 4). At the end of the simulation, the model reflects the final values of the system's inputs, states, and outputs. SIMULINK provides data display and logging blocks. Here intermediate results can be displayed and/or logged by including these blocks in to the model.

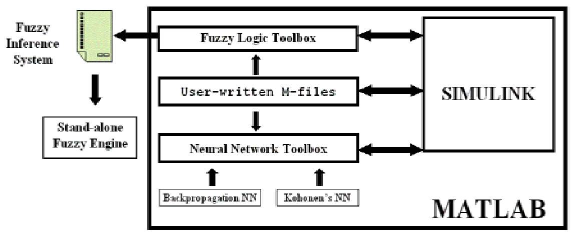

Figure 4. Integrated software development aimed to meet project implementation

The Fuzzy Logic Toolbox allows to do several things, but the most important thing - it enables to create and edit fuzzy inference systems (see Geva [13]). We can create these systems using graphical tools or command-line functions, or we can generate them automatically using either clustering or adaptive neuro-fuzzy techniques. The toolbox also allows to run our own stand-alone C programs directly, without the need for Simulink [16].

Some benchmarking financial problems were explored for practical validation. It concerned credit approval, insurance fees, and bankruptcy cases (all the data is taken from the UCI Repository of machine learning databases [17]). For the current stage of the research, promising results were obtained. Due to the limited space available and the scope of this work there is a clear need to present them for wider discussion in the next paper.

7. Conclusions

An overview of various NN applications in the financial market sector indicates a lack of universally applicable NN decision rules extraction techniques, which puts NN methods in a comparative disadvantage against other methods. Top and middle management requires logically reasoned decision support systems. The method presented here distinguishes to a minimum level of granularity, high portability of the rule extraction technique across various NN architectures and algorithmic simplicity of the rule refinement technique. This method is intended for various financial market domain problems, where user friendly and well-reasoned decision support systems are expected.

The author propose an innovative extension of the well known pedagogical approach. The idea is exposed by the use of a widespread MLP neural net (as a common tool for the decisions' space fragmentation in the financial problems' domain) and SOM (for the input data space clusterization). The feedback of both nets' performance is related and targeted through the iteration cycle by achievement of the best matching between the decision space fragments and input data space clusters. Three sets of rules are generated algorithmically or by fuzzy membership functions.

The result is an autonomous intelligence capable not only to recognize the problem, but also to make decisions, justify them and improve the performance from the past experience. The autonomy also means very high portability over different problem domains. The other result is an integrated software solution, which makes rules extraction technique easily transferable to the different neural nets designed for various tasks like recognition, prognoses, and optimisation. This is especially suitable for the financial capital market needs, where logically reasoned decisions have to be made.

References

[1] **B. Baesens, R. Setiono, C. Mues, J. Vanthienen.** Using Neural Network Rule Extraction and Decision Tables for Credit-Risk Evaluation. *Management Science, Vol.*49, *No.*3, *March*, 2003, 312-329.

[2] **A. Tickle, R. Andrews, M. Golea, J. Diederich.** Directions and Challenges in Extracting the Knowledge Embedded Within Trained Artificial Neural Networks. *IEEE Trans Neural Networks No.*9(6), 1998, 1057-1068.

[3] **L.C. Giles, S. Lawrence, A.C. Tsoi.** Rule Inference for Financial Prediction using Recurrent Neural Networks. *Proceedings of IEEE/IAFE Conference on Computational Intelligence for Financial engineering (CIFEr), IEEE, Piscataway, NJ*, 1997, 253-259.

[4] **R. Andrews, J. Diederich, A.B. Tickle.** A survey and critique of techniques for extracting rules from trained neural networks. *Knowledge Based Systems*, 8(6), 1995, 373-389.

[5] **R. Andrews, S. Geva.** Rule Extraction From Local Cluster Neural Nets. *Neurocomputing, Vol.*3, 2000, 217-233.

[6] **M.W. Craven, J.W. Shavlik.** Using sampling and queries to extract rules from trained neural networks, Machine Learning. *Proceedings of the Eleventh International Conference, W.W. Cohen & H. Hirsh (Eds.), San Francisco, CA: Morgan Kaufmann*, 1995.

[7] **A.B. Tickle, F. Maire, G. Bologna, J. Diederich.** Lessons from Past. *Current Issues and Future Research Directions in Extracting the Knowledge Embedded in Artificial Neural Networks , in Neural Hybrid Systems, S. Wermter and R. Sun (eds.), Springer Verlag*, 1999.

[8] **W. Duch, R. Adamczak, K. Grabczewski.** A new methodology of extraction, optimization and application of crisp and fuzzy logical rules. *IEEE Trans Neural Networks*, 11(2), 2000, 1-31.

[9] **R. Setiono, W.K. Leow.** On mapping decision trees and neural networks. *Knowledge Based Systems*, 13, 1999, 95-99.

[10] **T. Kohonen.** Self-Organizing Maps. *Second Edition, Berlin: Springer-Verlag*, 1997.

[11] **M.T. Hagan, H.B. Demuth, M.H. Beale.** Neural Network Design. *Boston, MA: PWS Publishing*, 1996.

[12] **R. Setiono, W.K. Leow.** Opening the neural network black box: an algorithm for extracting rules from function approximating artificial neural networks. *ICIS proceedings of the* XXI *international conference on IS, Queensland, Australia*, 2000.

[13] **A.B. Geva.** Hierarchical unsupervised fuzzy clustering. *IEEE Transactions on Fuzzy Systems. Vol.*7, *No.*6, Dec 1999, 723-733.

[14] **D. Plikynas, L, Simanauskas, S. Būda.** Research of Neural Network Methods for Compound Stock Exchange Indices Analysis. *Informatica, Vilnius, Institute of Mathematics and Informatics Lithuanian Academy of Sciences,* 2002, 13 (4), 465-484.

[15] **Y. Hiemstra.** Linear Regression Versus Backpropagation Networks to Predict Quartely Excess Returns. *The Second international Workshop on Neural Networks in the Capital Markets, CalTech, Pasadena.* 1999.

[16] Home Page of the Mathworks Web Site. *Last updated in* 2004.02.12 *http://www.mathworks.com/.*

[17] **C.L. Blake, C.J. Merz.** UCI Repository of machine learning databases. *Irvine, CA: University of California, Department of Information and Computer Science*, 1998. *[http://www.ics.uci.edu/~mlearn/MLRepository.html].*