Joint use of neural network technologies and decision trees for logical patterns exploration in data

V.N. Gridin[1], V.I. Solodovnikov[1]

[1]*Design information technologies Center RAS, Str. Marshal Biryuzov 7a, 143000, Odintsovo, Moscow Region, Russia*

Abstract

The issues of joint use of neural network technologies with methods of logical deduction and decision making support in data mining tasks are considered. The analysis of searching for logical patterns algorithms, their advantages and disadvantages is carried out. The description of combined algorithms for rules extraction from the trained neural networks and presentation the result in the form of a hierarchical, sequential structure of "if-then" rules is given. The representation of decision trees in the form of the semantic network facts is considered.

Keywords: neural network; decision trees; logical conclusion; rules extraction; date mining

1. Introduction

Data is a valuable resource, which contains a great potential opportunities for the extraction of useful analytical information. Therefore, the tasks of revealing hidden regularities, developing decision making strategies, forecasting, which requires more detailed consideration of the logical patterns exploration in classification problems, are becoming increasingly important. The peculiarity of algorithms and methods applicable for solving the data mining problems is the absence of a priori assumptions about the sampling structure and the distributions type of the analyzed indicators values. One of the closest correspondences to this condition could be the usage of an approach based on neural network technologies. This is due to the ability of neural networks for nonlinear processes modeling, working with the extremely complex dependencies, adaptability to the functioning conditions, and most importantly, the ability to extract and generalize essential features from incoming information. Thus, the network constructs rules, but these rules are contained in weighting coefficients, activation functions, and neuronal connections, but usually their structure is too complex to perceive and determine the effect of a particular characteristic on the output value. The neural network, in fact, acts as a "black box", the input of which is supplied with the initial data and the certain output result is obtained, however, it is not provided any rationale why this decision was made. To solve this problem, it is proposed the joint use of the neural network technologies with logical deduction methods, in particular decision trees, as a means of decision-making support, logical patterns exploration and the result presentation in the form of a hierarchical structure of classifying rules. And the use of semantic networks provides additional opportunities in construction of the deduction mechanisms and presentation the decision-making process.

2. Searching for logical patterns in the data

We will understand by logical regularity an easily interpreted rule that allocates a lot of objects of one class from the training sample and practically does not allocate objects of other classes. Logical patterns are elementary "building blocks" for a wide class of classification algorithms. Rules, that express regularities, are formulated in the language of first-order logic predicates and have the following form:

IF (condition_1) AND (condition_2) AND … AND (condition_N) THEN (conclusion),

where condition i could be $x_i = c_1$, $x_i < c_2$, $x_i > c_3$, $c_4 < x_i < c_5$ etc., x_i - variable, c_1, c_2, c_3, c_4, c_5 – some constants. For nominative data, the following predicates are used: «=» and «<>».

2.1. The limited search algorithms

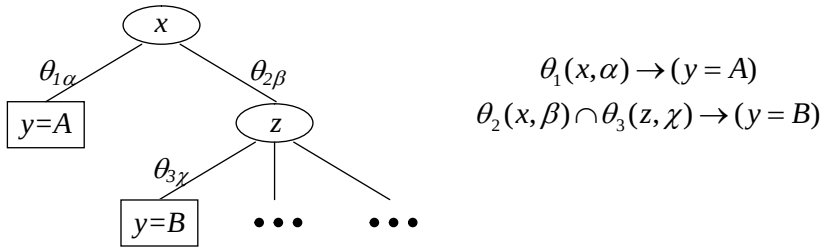
The limited search algorithms are used for logical regularities search in data, for solving classification and forecasting problems [1]. The main idea of this method is to analyze the frequency of occurrence of various combinations of simple logical events. At the initial stages, short associative chains are searched for, which are complicated in the process of the system's functioning, by adding new elements to them. Based on the analysis, the system makes a conclusion about the usefulness of this or that combination and, thus, establishes the logical patterns in the data. Its main disadvantage is the fact that this algorithm is capable in an acceptable time to find a solution for only a small dimension data.

2.2. Decision trees

Decision trees relate to the methods of logical regularities searching in data, and are the main approach applicable in decision making theory. They represent the hierarchical structure of "if-then" classifying rules, which have the form of a tree. Their main advantage is the simplicity and clarity of the decision-making process description. The disadvantage of their use in the problem of logical patterns search is the fact, that they are not able to find the most complete and accurate rules in the data and only

implement the simplest principle of sequential viewing of attributes and form fragments of regularities. Also, for large volumes of multidimensional data, these algorithms can produce a very complex tree structure that has many nodes and branches. Such trees could be very difficult for analyzing and understanding. Accordingly, the rules and patterns discovered by such a tree would be difficult for comprehension. In addition, a branchy tree with many nodes divides the training set into a large number of subsets consisting of a small number of objects. While it is much more preferable to have a tree with a small number of nodes, for each of which correspond a large number of objects from the training sample. To solve this problem, branch cutoff algorithms are often used [2], but they can not always lead to the desired result. However, methods of searching regularities with the help of decision trees allow us to find such connections that are concluded not only in certain features, but also in a combination of features, which in many cases gives these methods a significant advantage over classical methods of multivariate analysis.

Figure 1 shows an example of such a decision tree, and the corresponding logical deduction, where $\theta 1, \theta 2, \theta 3$ - predicates, x, y, z – variables, α, β, χ - constants.



$$\theta_1(x,\alpha) \rightarrow (y = A)$$
$$\theta_2(x,\beta) \cap \theta_3(z,\chi) \rightarrow (y = B)$$

Fig. 1. An example of a decision tree.

Rules that express regularities are formulated in the form of expressions: «IF A THEN B» or in the case of a set of conditions: «IF (condition 1) $\wedge$ (condition 2) $\wedge$ … $\wedge$ (condition N) THEN (the output node value)».

The decision trees construction is usually carried out by following ways:

- on the expert assessments basis;
- using sample processing algorithms (CLS, ID3 (Interactive Dichotomizer), C4.5, CART (classification and regression trees) etc.);
- using genetic algorithms and evolutionary programming.

Each of these approaches has its advantages and disadvantages and can be used to solve its specific tasks.

2.3. Genetic algorithms

The most difficult problem in search for logical regularities in data sets is to find the elementary events, representing the terms of the conditional part "IF". At present, genetic algorithms (GA) are increasingly used to solve this problem, which include algorithms: Bucket-Brigade, REGAL, G-NET, HIDER, SIAO1 and some others. However, they are not without a number of drawbacks: a fixed set of rules and their length, as well as accuracy and completeness in most of them are not taken into account.

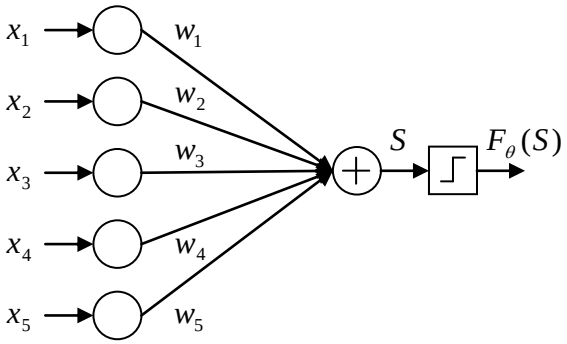
2.4. Neural network methods

The usage of the approach based on neural network data processing technologies is caused by the ability of neural networks to model non-linear processes, act with extremely complex dependencies, adaptate to operating conditions, work with noisy data and with the lack of a priori information. And most importantly, they are able to learn from experience, generalize previous precedents into new cases and extract significant features from incoming information. Thus, the network constructs rules, but these rules are contained in weighting coefficients, activation functions, and neuronal connections, but usually their structure is too complex to perceive. Moreover, these parameters can represent non-linear, non-monotonic relationship between the input and target values in a multilayer network. Thus, as a rule, it is not possible to separate the effect of a certain attribute to the target value, cause of this effect can be mediated by the values of other parameters. The neural network, in fact, acts as a "black box", the input of which is supplied with the initial data and the certain output result is obtained, however, it is not provided any rationale, why this decision was made.

3. Getting logical patterns from a trained neural network

Let the problem consists in the classification of a certain set of data with the help of a perceptron and the subsequent analysis of the obtained network in order to find the classifying rules that characterizes each of the classes.

First, let's consider this problem with the example of a single-layer perceptron, which consists of five Boolean inputs and one output neuron. This network can be accurately interpreted by a finite number of "if-then" rules, since a finite number of possible input vectors are defined for it.



$$S = \sum_i w_i \cdot x_i$$

$$y = F_\theta(S) = \begin{cases} 1 & if\ S > \theta \\ 0 & else \end{cases}$$

Fig. 2. Single-layer perceptron with five Boolean inputs and one output.

Let the weights take on the following values: $w_1 = 6$, $w_2 = 4$, $w_3 = 4$, $w_4 = 0$, $w_5 = -4$, and the bias $\theta = 9$. In this case, the following set of rules can be extracted from the network:

$x_1 \wedge x_2 \wedge x_3 \rightarrow y$

$x_1 \wedge x_2 \wedge \neg x_5 \rightarrow y$

$x_1 \wedge x_3 \wedge \neg x_5 \rightarrow y$

Thus, the decision-making procedure is to predict the value $y = true$, if the activation of the output neuron is 1, and $y = false$, if the activation is 0.

Generally speaking, it is possible to distinguish two approaches for extracting rules from the multilayer neural networks [3]. The first approach is to extract a set of global rules that characterize the output classes directly through the values of the input parameters. An alternative is to extract local rules by separating a multi-layer network into a collection of single-layer networks. Each extracted local rule characterizes a separate hidden or output neuron, taking into account elements that have weighted connections with it. Then all got rules are combined into a set that determines the behavior of the entire network as a whole. The local approach is illustrated at Figure 3.

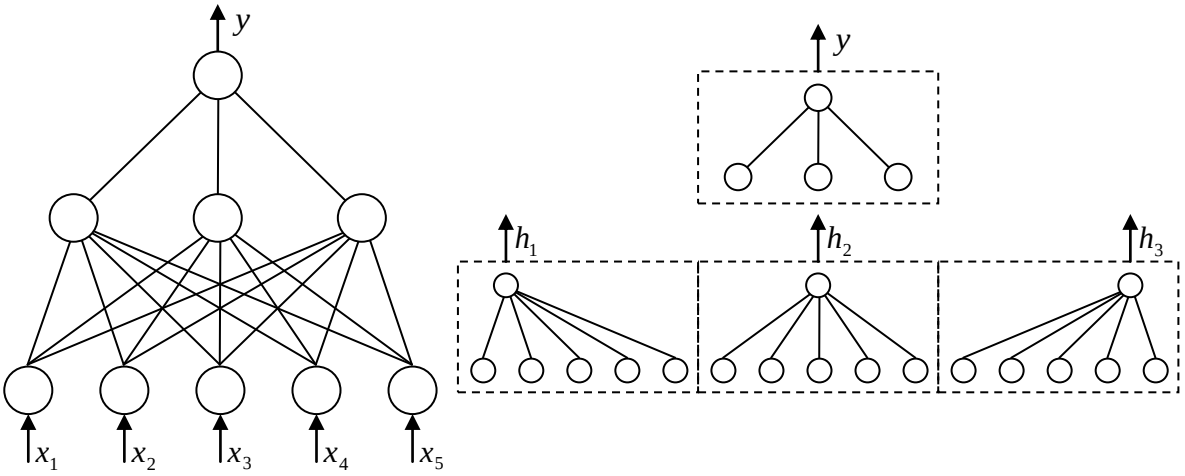


Fig. 3. A local approach for extracting rules. Multilayer neural network is divided into a set of single-layered. Rules for describtion of each component are extracted, which are combined into a set that characterizes the multi-layer network.

Let's consider the problem of extracting rules in a more general form.

Let X denote a set of n properties $X_1, X_2, ..., X_n$, and $\{x_i\}$ is the set of possible values that a property X_i can take. And C denotes the set of classes $c_1, c_2, ..., c_m$. The associated pairs of input and output vector values are known for the training sample $(x_1, ..., x_n, c_j)$, where $c_j \in C$.

3.1. Local approach for rules extraction

NeuroRule is one of the algorithms for extracting rules from neural networks, which were trained to solve the classification problem [4]. This algorithm includes three main steps:

Step 1. Neural network training.

At the first stage, a two-layer perceptron is trained until sufficient classification accuracy would be obtained. At the initial time, a large number of the hidden layer neurons are selected. Unnecessary neurons and connections would be discarded after training.

Step 2. Thinning of a neural network.

The trained neural network contains all possible connections between input neurons and hidden layer neurons, as well as between hidden and output neurons. Usually the total number of these links is so large that it is impossible to extract observable for user classifying rules from their values analysis. Thinning consists of removing unnecessary connections and neurons, which absence does not increase the network classification error. The resulting network usually contains much less neurons and connections between them, and the operation of such a network is able to be investigated.

Step 3. Rules extraction.

At this stage, the rules that take form of «IF ($x_1 \Theta q_1$) AND ($x_2 \Theta q_2$) AND ... AND ($x_n \Theta q_n$) THEN c_j » are extracted from the thinned neural network. Here $q_1,\ldots, q_n$ are constants and Θ is the relational operator ($=, \geq, \leq, ><$). First the preparation for rules extraction is taking place, which includes coding of all continuous quantities for input and interior network values. Also the coding process is performed for features of the classified objects if they have continuous values. To represent them, it is possible to use binary neurons and a coding principle such as a thermometer. The resulting values of the hidden layer neurons are clustered and replaced with values that determine the centers of this clusters. It is important to select a small number of such clusters. The objects classification accuracy by the network is checked, after such discretization of the hidden neurons functionality. If it remains acceptable, than the preparation for rules extraction comes to the end. Further, the rules extraction is taking place, during which the movement through the network occurs from the classifying output neurons to the network inputs. It is assumed that these rules are fairly obvious while verified and are easily could be applied to the large databases.

However, this algorithm establishes rather strict limitations on the architecture of the neural network, the number of elements, connections and the type of activation functions. So for the hidden neurons the hyperbolic tangent is used and their states change in [-1,1] interval, and for the output neurons the Fermi function with the state interval [0,1] is applied.

3.2. Global rules extraction

The lack of universality and scalability could be mantioned as the main drawbacks of most algorithms of rules extraction. In this regard, TREPAN algorithm [5] gets the most interest. It lacks these shortcomings and does not impose any requirements to the network architecture, input and output values, learning algorithm, etc. This approach builds a decision tree on the base of knowledge embedded in the trained neural network, and it is enough that the network is a kind of "black box", "expert" or "oracle", to whom it is possible to ask questions and get answers. Moreover, this algorithm is sufficiently universal and can be applied to a wide range of other trained classifiers. It also scales well and has no sensitive to the input attributes dimension and the size of the network.

This algorithm builds the decision tree, that approximates the functionality of the trained neural network, and consists of two following stages.

Preliminary stage:

1. Construct and train a neural network that will later act as an "Expert" or "Oracle".
2. Initialize the root of the tree *R* as a leaf.
3. Use the entire training set of examples *S* to construct a distribution model M_R of the input vectors that reach the node *R*. Compute value *q = max(0, minSamples - |S|),* where minSamples is the minimum number of training examples used in each node of the tree, *S* is the current training sample (|*S*| is the training sample volume). Thus, *q* is the number of additional examples that need to be generated.
4. *q* new learning examples are generated randomly on the base of the attributes distribution evaluation from *S*. $query_R$ is the set of *q* examples generated by the model M_R.
5. Use neural network as an "Oracle" to classify both new $query_R$ and old examples from the set *S* to a particular class. For each vector of attributes $x \in (S \cup query_R)$, put a class label $x = Oracle(x)$.
6. Initialize the $Queue$, by placing the set $\langle R, S, query_R, \{empty\_constr\} \rangle$.

Main stage:

7. Take the next set $\langle N, S_N, query_N, constr_N \rangle$ from beginning of the $Queue$, where N is the node of the tree, S_N is the training sample in the node N, $constr_N$ is a set of restrictions on the certain attributes of the training examples for reaching the node N.
8. Use $F, S_N, query_N$ for construction branching T in a node N.

   Here *F* is a function for estimating the node *N*. It has the following form $F(N) = R(N) \cdot (1 - f(N))$, where $R(N)$ is the probability of reaching node *N* by an example, and $f(N)$ is the correctness evaluation of these examples processing by a tree. Thus, the best node is chosen, which branching has the greatest impact to the classification accuracy of the generated tree. The separation of the examples, which reach this internal node of the tree, is carried out depending on the $m-of-n$ test [5,6]. Such a test is considered to be passed when it is satisfied, at least m from n conditions. On the other hand, it is possible to split the set *S* as in the usual algorithm for decision tree construction.
9. Create next-generation nodes for each branch *t* of branching *T*:
   a. Create C as a new child node in a relation to N.

b. Add a restriction from the branch *t* to $constr_C = constr_N \cup \{T = t\}$.

c. Generate set S_C, which contains examples from the set S_N that satisfy the condition on the branch *t*.

d. Construct a model M_C for examples distribution that reach the node C.

Calculate the number of examples to generate *q = max(0, minSamples -* $|S_C|$*).*

e. *q* new learning examples are randomly generated on the base of the characteristics distribution evaluation from S_C and constraint values $constr_C$. $query_C$ is a set of q examples, which were generated by the model M_C and constraint $constr_C$.

f. Use neural network as an "Oracle" to classify new examples $x \in query_C$ and expose the class label $x = Oracle(x)$.

g. Initially, it is assumed that the node *C* is a leaf. Use S_C and $query_C$ to determine the class label for *C*.

h. Check the necessity of the further branching of the node *C*. Put the set $< C, S_C, query_C, constr_C >$ into the $Queue$ if the local stop criterion is not satisfied. A local criterion in this case is the probability that in a given node there are instances of one class.

10. If the $Queue$ is not empty and the global stop criterion is not fulfilled, then go to step 7, otherwise return the tree with the root R.

The maximum tree size and the overall classification quality evaluation of examples by a tree are used as the global criterion for completing the algorithm.

The generalization ability of artificial neural networks, which allows to obtain more simple decision trees is the main advantage of this approach. In addition, the applying of such an "Oracle" allows to compensate the lack of data, which is usually could be observed at lower levels during the decision trees construction by the sample processing algorithms. Thus, it is possible to extract structured knowledge not only from extremely simplified neural networks, but also from arbitrary classifiers that makes possible the appliance of this algorithm in a wide range of practical problems.

4. Rules representation in a form of a semantic network

A simple semantic network, sometimes called a computational semantic network, is actually a bipartite graph.

Lets there are a finite set *A = {Al,. . ., Ar}*, which is called attributes, and the finite set *R = {R1,. . ., Rn}* of relations. The scheme or intensional of the ratio *Ri (*$i = 1,...,n$*)* is the set of pairs:

INT(Ri) = { . . ., [Aj, DOM(Aj)], . . .},

where *Ri* is the name of the relation, *DOM(Aj)* is the domain of *Aj (* $j = 1,...,r$*)*,i.e. the set of attribute *Aj* values of the relation *Ri*. The union of all domains is called the base set of the model or the set of objects, on which the relations *R* are specified.

An extensional of *Ri* relation is the set:

EXT(Ri) = {F1, . . ., Fp},

where *Fk (* $k = 1,..., p$*)* is the fact of the relationship *Ri*. The fact is set by an aggregate of attribute-value pairs, called attribute pairs. Fact is a concretization of a certain relationship between the specified objects. In a graphical interpretation, fact is a subgraph of a semantic network that has a star-shaped structure. The root of a subgraph is a vertex of a predicate type, labeled with a unique label that includes the name of the corresponding relationship. From the vertex of the fact connections are coming out, which are marked with the names of attributes of this fact. They are leading to the vertices of the base set and are the values of these attributes.

It is worth noting that the semantic network can be represented as a storage of facts that were derived from the decision trees processing, i.e. the decision tree is transformed into a semantic network. In this case, each fact is presented in the form of a ready-made deduction output. This provides additional opportunities for analysis. For example, even the usual means of databases can find the facts that relate to different relationships but have the same attributes and values that characterize them. To store semantic networks in the database, or rather their extensional, it is possible to use a table of the form:

Table 1. Table of extensional.

Field name	Data type	Field Properties
CodeValue	Numeric	Key field
FactMark	Numeric	The fact mark
FactAttributes	Text	Attribute of fact
AttributeValue	Text	Attribute Value

Thus, the decision tree view will be obtained in the form of a fact table, where a record with the fact attributes values exists for each value of the output deduction variable. Since there is a mapping of one representation to another, so from the formal point of view these representations are identical.

The conclusions could be drawn in a semantic network, which are far from obvious for a decision tree. Let's take this simple example. The decision tree determines the conclusion about the establishment of the parental relations in the first generation. It is necessary to define the parent relationship in the second generation. The solution of this problem for the semantic network is

obvious. It is necessary to highlight the facts of parental relations, then to delete objects from the intensional that do not match the parents and repeat the process of highlighting the facts of the parent relationship.

This means that it is possible in the semantic network introduce means of constructing functional dependencies similar to how it is done in functional programming languages, for example, such as LISP [7]. You can consider the fact as a list of atoms, each of which is assigned a value either directly or in the process of output. If operations for such lists manipulation are entered then it is possible to create and modify decision trees with formal methods.

Thus, it seems advisable to combine in a single system a representation in the form of decision trees and a class of semantic networks, which makes it possible to visually display the decision-making process and gives additional possibilities in constructing the deduction mechanisms.

5. Conclusion

In this paper, the problems of logical regularities search in the classification tasks were considered. A joint use of neural network technologies with logical deduction methods, in particular decision trees, as a means of logical patterns exploration and the result presentation in a hierarchical structure form of classifying rules, is proposed. Two main approaches are identified. The first is to extract local rules, where the multilayer network is divided into a set of single-layered. Each local rule characterizes a separate hidden or output neuron, taking into account elements that have weighted connections with it. Then the rules are combined into a set that determines the behavior of the entire network as a whole. However, this approach often establishes fairly strict limitations on the network architecture, the number of elements, links and the type of activation functions, which negatively affects the universality and scalability.

An alternative is to extract a set of global rules that characterize classes at the output directly through the values of the input parameters. In the framework of this approach a modified algorithm for decision trees construction on the base of the trained neural networks is developed. It does not impose any requirements on architecture, learning algorithm, input and output values and other network parameters. The construction of the tree is base on knowledge that embedded into the trained neural network, and it is enough that the network is a kind of "Black Box" or "Expert", for which it is possible to ask questions and get answers. The generalization ability of artificial neural networks, which allows to obtain more simple decision trees and, if it is necessary, to compensate the lack of initial data are the main advantages of this approach. Moreover, this algorithm is sufficiently universal, well scalable and not sensitive to the input attributes dimension and the size of the network. This circumstance acquires special significance in the light of the rapid development of deep learning technology. Thus, it is possible to extract structured knowledge not only from extremely simplified neural networks, but also from arbitrary classifiers that makes possible the appliance of this algorithm in a wide range of practical problems.

In addition, an algorithm for converting already formed decision trees into semantic networks in the form of a bipartite graph has been developed. This provides a solution tree view in the form of a fact table and allows a quick search by known attributes.

The automation tools introduction into the data mining systems could shorter the time, improve the quality and effectiveness of the decisions making.

Acknowledgements

Work is carried out with the financial support of the RFBR, the project 15-07-01117a.

References

[1] Dyuk V, Samojlenko A. Data Mining. SPb: Piter, 2001; 368 p.
[2] BaseGroup Labs company WebSite, Trees of decisions - general principles of work. URL: https://basegroup.ru/community/articles/description (10.01.2017).
[3] Gridin VN, Solodovnikov VI, Evdokimov IA, Filippkov SV. Building decision trees and extracting rules from trained neural networks. Iskusstvennyj intellekt i prinyatie reshenij 2013; 4: 26–33.
[4] Ezhov AA, Shumskij SA. Neurocomputing and its application in economics and business. M.: MIFI, 1998; 224 p.
[5] Craven MW, Shavlik JW. Extracting tree-structured representations of trained networks. Advances in Neural Information Processing Systems. MIT Press, Cambridge, MA 1996; 8: 24–30.
[6] Murphy PM, Pazzani MJ. ID2-of-3: Constructive induction of M-of-N concepts for discriminators in decision trees. Proceedings of the Eighth International Machine Learning Workshop, Evanston, IL 1991; 183–187.
[7] Hyuvenen EH, Seppyanen I. Mir LISPA. Methods and systems of programming. M.: Mir, 1990; 320 p.