http://doi.org/10.35784/iapgos.927

LOGICAL CLASSIFICATION TREES IN RECOGNITION PROBLEMS

Igor Povhan
Uzhgorod National University, Department of Software Systems, Uzhgorod, Ukraine

Abstract. *The paper is dedicated to algorithms for constructing a logical tree of classification. Nowadays, there exist many algorithms for constructing logical classification trees. However, all of them, as a rule, are reduced to the construction of a single classification tree based on the data of a fixed training sample. There are very few algorithms for constructing recognition trees that are designed for large data sets. It is obvious that such sets have objective factors associated with the peculiarities of the generation of such complex structures, methods of working with them and storage. In this paper, we focus on the description of the algorithm for constructing classification trees for a large training set and show the way to the possibility of a uniform description of a fixed class of recognition trees. A simple, effective, economical method of constructing a logical classification tree of the training sample allows you to provide the necessary speed, the level of complexity of the recognition scheme, which guarantees a simple and complete recognition of discrete objects.*

Keywords: pattern recognition problems, logical tree, graph-scheme models, recognition system

LOGICZNE DRZEWA KLASYFIKACJI W ZADANIACH ROZPOZNAWANIA

Streszczenie. *Artykuł poświęcono algorytmom konstruowania logicznych drzew klasyfikacji. Większość tych algorytmów z reguły sprowadzają się do zbudowania jednego drzewa klasyfikacyjnego na podstawie stałej próby uczącej. Należy zauważyć, że niewiele algorytmów budowania drzew klasyfikacyjnych dla prób treningowych o dużej objętości. Oczywiste jest, że mają one obiektywne czynniki związane ze specyfiką generowania takich struktur, metodami pracy z nimi i ich przechowywania. W niniejszym artykule autorzy skupiają się na opisie algorytmu konstruowania drzew klasyfikacyjnych dla dużego zbioru uczącego i wskazują możliwość jednolitego opisu stałej klasy drzew rozpoznawczych. Prosta, skuteczna i ekonomiczna metoda budowy logicznego drzewa klasyfikacyjnego dla danej próby uczącej pozwala na zapewnienie niezbędnej szybkości i stopnia złożoności schematu rozpoznawania, co gwarantuje proste i kompletne rozpoznawanie obiektów dyskretnych.*

Słowa kluczowe: zadania rozpoznawania obrazów, logiczne drzewo, schematy modeli rozpoznawania, system rozpoznawania

Introduction

As of today, various algorithms for constructing logical classification trees are known [7]. However, all of them, as a rule, are reduced to the construction of a single classification tree from the data of a fixed training sample. Note that in the literature there are very few algorithms for constructing logical trees for training samples of large volume. It is clear that this is based on objective factors associated with the features of the generation of such complex structures, methods of working with them and storage [5]. Even using the tools of Java or C#, it is necessary to provide the implementation of special data structures for working with logical trees, and ready-made libraries (LightGBM, XGBoost), although close ideologically (logical tree scheme), do not allow to implement the concept of an algorithmic classification tree, which consists of a set of vertices – different types of Autonomous classification algorithms. However, the main drawback in the construction of logical trees is the lack of algorithms and methods that would allow uniformly describe different algorithms for pattern recognition in the form of tree structures.

The ability to represent the recognition function as a logical tree has great advantages over other representations of classification schemes [4]. It should be noted that the algorithms for generating classification trees according to the training sample complement the methodology of the branched feature selection approach and allow to build simple and effective rules for the classification of discrete objects [1].

In this paper, we will focus on the description of the algorithm for constructing logical training samples for a large volume and show the way to the possibility of a uniform description of a fixed class of logical trees.

1. Statement of the recognition problem

In fact, the central task of pattern recognition is to build such a system that for each object that will be presented to it, would give the number of the class to which the object belongs [3, 4, 5]. The general problem of pattern recognition (classification) can be formulated in the following simplified form. Let on some set *M* of objects *w* a given partition *R* into a finite number *m* of subsets (classes, images) H_i (*i = 0, ..., m*):

$$M = \bigcup_{i=1}^{m} H_i \tag{1}$$

The corresponding sets $H_0, H_1, ..., H_m$ will be called images, and the elements of the set – *M* images or representatives of images $H_0, H_1, ..., H_m$. If the condition is met (1), then this split will be called complete. Objects (images) *w* are defined by sets of values of some features $x_j, j = 1,...,n$. If $w \in H_i$ we assume that this object belongs to the image H_i. In general, images $H_0, H_1, ..., H_m$ can be specified by probability distributions $p(H_0 / w), p(H_1 / w),..., p(H_m / w)$, where $p(H_i / w)$ – the probability (or in the continuous case, the probability density) $w(w \in M)$ of belonging to the image H_i.

As a rule, in the problems of pattern recognition at the beginning is given some a priori information about *R* the nature of the partition and, depending on the nature of this information, it is customary to consider the following recognition tasks with training, without training, with self-learning. In this paper, the main attention will be paid to the problems of recognition with the previous training.

It should be noted that the fixed set of features that are characterized *w*, is always the same for all objects that are considered in solving this problem. Each feature can take values from different sets of valid feature values. For example, very often signs take values from a set $\{0,1\}$ or a sign takes a finite number of values – $\{a_1, a_2,..., a_d\}$ the value of a sign can be a distribution function of some random variable. Objects that belong to one class are characterized by a certain commonality of their features, and objects from different classes do not have such a commonality, therefore the solution to the problem of recognition is to somehow highlight and describe this commonality or its absence.

When recognizing images, the most important, and sometimes the only given information about the partition is the training sample:

$$(A_1, H_{s,1}), (A_2, H_{s,2}),...,(A_z, H_{s,z}) \tag{2}$$

where A_k – certain objects (vectors), $H_{z,k}$ are the numbers of classes that contain objects A_k. Belonging of the object to a particular class is determined, as a rule, as a result of experimental studies, during which the training sample is formed.

It is on the basis of it in the problems with the previous training that the classification rule is built, and the solution is to determine the class to which the object that is being studied belongs.

The main requirement that is imposed on the training sample is the most complete and adequate description of the nature of the breakdown H_0, H_1, ..., H_m. In real tasks to fulfill this requirement is not only difficult, and sometimes impossible, because the nature of the partition may not even know the experimenter, and its full description will lead to a significant increase in the amount of training sample, and therefore – and the time spent on the learning process of the system. Therefore, the process of "error accumulation" begins at this stage, and even the most perfect recognition system in this case will be ineffective and powerless.

Each partition H_0, H_1, ..., H_m can be implemented using some finite-valued function $f_r(w)$ whose argument takes values from the set M. For example, H_0, H_1, ..., H_m it is possible to use a function $f_r(w) = i$ to split, if w necessary H_i, $i = 0, ..., m$.

It should be noted that the function $f_r(w)$, where $w \in M$, determines, in turn, some partition of the set M. Furthermore $f_r(w)$, then is only then everywhere defined in set M the plural when the corresponding partition R is complete. Thus, each partition H_0, H_1, ..., H_m can be set using a finite-valued function $f_r(w)$, and – each finite significant function $f_r(w)$ sets some partition.

For example, M – let the set of real numbers and $H_0 = \{w \mid 0 < w \le 2\}$, $H_1 = \{w \mid -2 \le w \le -1\}$, $H_3 = \{w \mid 3 \le w \le 5\}$. Moreover, $\{w \mid P(w)\}$ there is a set w for which the predicate $P(w)$ will be true. This partition can be set using an arbitrary function of the form:

$$f_r(w) = \begin{cases} \alpha_0, \ if & 0 < w \le 1 \\ \alpha_1, \ if & -2 \le w \le -1 \\ \alpha_2, \ if & 3 \le w \le 5 \end{cases}$$

Note that here $\alpha_0, \alpha_1, \alpha_2$ – arbitrary pairwise different real numbers. Thus, the same partition H_0, H_1, ..., H_m can be specified using many functions $f_r(w)$.

In general , the task of pattern recognition (classification) is to construct a function $f_r(w)$ that implements the splitting R of the set M, or to calculate the values of some predicates $P_i(w)$ (note that the decision about the object belonging to a particular class can be encoded as follows: 0 (w not allowed H_i) and 1 (w belongs H_i).

Let's define a feature of some object w as a function $f(w)$ whose arguments take values from an arbitrary set G, and $f(w)$ – from a countable set $\{0,1,..,k\}$. The function $f(w)$ can be both deterministic and probabilistic. An arbitrary function $f(w)$ defined on G, and taking a finite number of values, we called a sign. Let L be a class of signs on the lot M, and K – class diagrams. Under the scheme $S(f_1, f_2, ..., f_n)$ of the class K we understand the operator, which signs $f_1, f_2, ..., f_n$ with L puts in line some (over significant) function $\phi(x) = S(\alpha_1, ..., \alpha_n)$, defined on the set M.

As rudimentary signs there may be predicates, that is, deterministic characteristics of the host either 0 or 1. From them we will demand that they, in a sense, were the simplest (were a description of a fixed image). A sign that can be obtained in some way from the elementary features $\phi_1, \phi_2, ..., \phi_n$ we call a generalized feature.

2. The scheme of construction of a logical recognition tree

The main purpose of the algorithms of recognition methods based on the logical tree, which will be presented below, is to maximize the value of $W_M(f)$ [2, 5, 7]. The latter means that the algorithms of the logical tree should be found for the training sample (2) such a generalized sign f for $W_M(f)$ which the value is the largest possible.

Note the following, the sample (2) may have a probabilistic character, that is $(x_i, f_R(x_i))$, $(i = 1, 2, ..., M)$ pairs may appear in it according to some probability distributions $p(x / H_0)$, ..., $p(x / H_{k-1})$, but the generalized characteristic is deterministic. Thus, we pose the problem of optimal approximation of the probability sample (2) with the help of some deterministic function, which is generally represented by a generalized feature f. It is obvious that the problem makes sense when the character of images (classes) H_0, H_1, ..., H_{k-1} is close enough to deterministic. The latter will mean that the main share is occupied by those points (objects) x for which the value $\max(p(x / H_0), ..., p(x / H_{k-1}))$ is close to one. This value can vary significantly only at points (objects) that lie on the boundary of several classes H_0, H_1, ..., H_{k-1}.

Note that in practice, they mainly work with tasks where images (classes) H_0, H_1, ..., H_{k-1} have a character close to the deterministic case (for example, recognition of handwritten characters).

All algorithms that will be presented below have the following feature. Each algorithm is a process that consists of certain steps $d_0, d_1, ..., d_i$. Each step d_j here consists in turn of two stages (modes) – training and test.

In the training mode, a generalized feature f_i is formed on the step d_i. In the test mode, the effectiveness of $W_M(f_i)$ [5] relative to the training sample (2) is calculated for this generalized trait. If $W_M(f_i) \ge \delta$, then the learning process ends, if $W_M(f_i) < \delta$, then the transition to the step d_{i+1}; δ – a number that characterizes the evaluation of the effectiveness of training that is required by the task.

Now it is necessary to note the points regarding the nature of the sample submission (2) during training. In practice, two cases are possible:

a) The sample (2) is fixed, that is, all of It is supplied at each step of training d_i.
b) Selection (2) depends on the step d_i, which means that each step d_i of the training is fed its own sample.

The case (a) occurs when the sample (2) is the data of some experiment (for example, computer measurements), which are recorded in permanent memory. The learning algorithm in this case is a multiple sample processing (2). Note that the sample (2) can be very large. Therefore, the algorithms of processing of the sample should be such that it would be in their work sample (2) are not recorded in memory.

If there is no case (a) and stored data in permanent memory, then we have case (b). In this case, all the pairs that are processed in the step d_i are not remembered, and therefore some other series of training pairs of the form (2) are supplied in the step d_{i+1}.

Most of the methods presented below are structured so that they can be applied in both case (a) and case (b). For certainty, we further assume that there is a case (a), that is, at each step d_i the same sample (2) is supplied.

At the heart of all recognition methods in the form of a logical tree is one schematic diagram, which is called the scheme of the tree method. In this scheme, initially selected some elementary feature ϕ_1^1. This characteristic requires that the magnitude $W_M(\phi_1^1)$ is relatively (2) were possibly the greatest. Let us note once $W_M(\phi_1^1)$ that it is calculated according to the method [5, 6]. The following steps of the method of the logical tree, it is convenient to interpret with the help of a tree (Fig. 1).

In each vertex of the tree (Fig. 1) there is either some sign ϕ_i^j or a number m_i^j that belongs to the set $\{0, 1, ..., k-1\}$. Note that the number m_i^j here defines the image (class) $H_{m_j^i}$. The vertex in which it stands m_i^j is called the final vertex of the tree. From each vertex, in which there is a sign ϕ_i^j, depart two guides (arrows) which are marked 0 and 1. Guide, labeled 0, corresponds to the value $\phi_i^j = 0$ and represents the 1 value $\phi_i^j = 1$. The tree is broken conditionally behind tiers. There are signs ϕ_1^j, ϕ_2^j, ..., in j – tier.

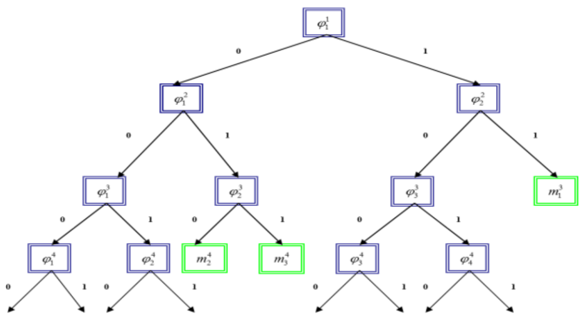

Fig. 1. Initial recognition tree

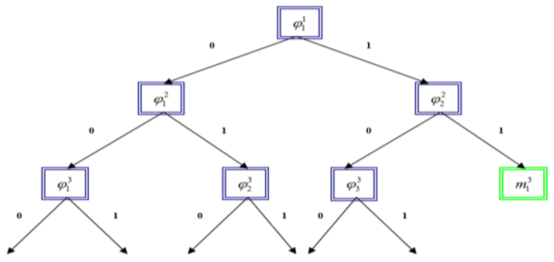

Fig. 2. Recognizer tree after three steps of the synthesis

All the features that are in all tiers, starting from the first and ending with the n – tier are those features that are obtained after the n steps (stages) of the process of building the classification tree. Moreover, the signs standing on the n – tier are those signs that are obtained at the n step (stage) of the process of building a logical tree.

Suppose that only three steps of the tree method are carried out and ϕ_1^1, ϕ_1^2, ϕ_2^2, ϕ_1^3, ϕ_2^3, ϕ_3^3 – all the signs obtained as a result of these steps. The logical tree that we get for these three steps will have the form shown in (Fig. 2).

Each related pair $(x_i, f_R(x_i))$, $(1 \le i \le M)$ of sampling (2) is the corresponding defined path of the tree (Fig. 2). This path is implemented as follows. First calculated $\phi_1^1(x_i) = r_i$. Further from the top ϕ_1^1 down the arrow, which is indicated by r_i. Let, for example $\phi_1^1(x_i) = r_1 = 0$. Then go down to the top, which is a sign ϕ_1^2. Then calculate $\phi_1^2(x_i) = r$ and go down the arrow that comes out of the vertex ϕ_1^2 and marked with the value r_2 and so on.

The path that corresponds to the pair $(x_i, f_R(x_i))$ (it is completely determined by the value x_i) is denoted by T_i. There are two possible cases:

a) Let the path T_i end with some directional arrow for example, if $\phi_1^1(x_i) = 0$, $\phi_1^2(x_i) = 0$, $\phi_1^3(x_i) = 0$ the path T_i ends with an arrow that comes out of the vertex ϕ_1^3 and is denoted by a symbol 0 (Fig. 2).

b) Let the path T_i end with some vertex where the value is m_i^j. For example, when $\phi_1^1(x_i) = 1$, $\phi_2^2(x_i) = 1$ and path T_i ends with a vertex containing a value m_1^3 (Fig. 2).

The paths in case (a) are called unfinished, and the paths in case (b) are finished.

If the path T_i that corresponds to the pair $(x_i, f_R(x_i))$ is complete and there is a value m_i^j, $(m_i^j \in \{0,1,...,k-1\})$ at the end, it means that $f_R(x_i) = m_i^j$. For example (Fig. 2) for all pairs $(x_i, f_R(x_i))$ that satisfy the condition $\phi_1^1(x_i) = \phi_2^2(x_i) = 1$, the following condition is fulfilled: $f_R(x_i) = m_1^3$. It can be said that for a value x_i that corresponds to a complete path T_i, full tree recognition is implemented (Fig. 2). That is, we can say that the pair $(x_i, f_R(x_i))$ belongs to the corresponding path T_i.

When carrying out the following steps in the construction of a classification tree are considered only unfinished journey.

Next, each path on the tree under construction will be denoted by a binary set $r_1, r_2, r_3 ....(r_i \in \{0,1\})$. For example, a binary set 010 in a tree (Fig. 2) denotes a path that ends with a final arrow, with a vertex ϕ_2^3 and a symbol 0. It is obvious that the set 000, 001, 010, 011, 100, 101 is a set of all unfinished paths on the tree (Fig. 2).

Let M_{r_1,r_2,r_3} – the number of all sample pairs $(x_i, f_R(x_i))$ (2) that belong to the unfinished tree path r_1, r_2, r_3 (Fig. 2). Let's M_{r_1,r_2,r_3}^j, $(0 \le j \le k-1)$ – say the number of all pairs that belong to a path r_1, r_2, r_3 and also for them the following relation is executed: $f_R(x_i) = j$. For each unfinished tree path (Fig. 2) calculate the values:

$$t_{r_1,r_2,r_3}^j = \frac{M_{r_1,r_2,r_3}^j}{M_{r_1,r_2,r_3}}, \; (j = 0, 1, ..., k-1) \tag{3}$$

Next, we find such a value $l(r_1, r_2, r_3)$ that:

$$l(r_1, r_2, r_3) \in \{0,1,...,k-1\}, \; t_{r_1,r_2,r_3}^{l(r_1,r_2,r_3)} = \max_j t_{r_1,r_2,r_3}^j \tag{4}$$

Substituting at the end of each path r_1, r_2, r_3 on the tree (Fig. 2) the value $l(r_1, r_2, r_3)$, we obtain the following tree (Fig. 3).

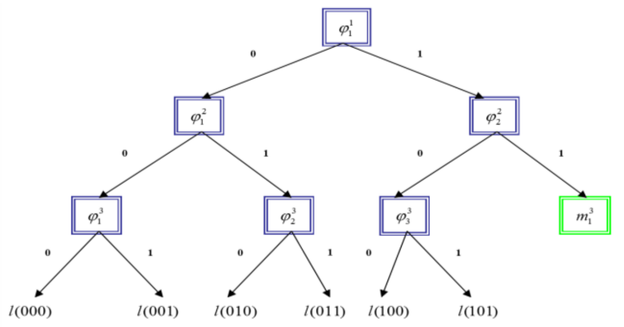

Fig. 3. Modified classification tree

3. Verification test stage

The tree (Fig. 3) implements some generalized trait $f_3(x)$ which is defined on the set G and takes values from the set $\{0,1,...,k-1\}$. The sign $f_3(x)$ is calculated as follows. First find for x the entire path T_x, which corresponds to this element. For example, if $\phi_1^1(x)=0$, $\phi_1^2(x)=1$, $\phi_2^3(x)=1$, then $T_x=011$. As values $f_3(x)$, take the element $\{0,1,...,k-1\}$ that ends the path T_x. For example, if, $T_x=011$ then $f_3(x)=l(011)$. If the object x corresponds to the completed path T_x, at the end of which there is a number m_x, $(0\le m_x\le l)$, then we assign that $f_3(x)=m_x$.

After the feature $f_3(x)$ is built, the verification test stage begins. In the test mode, the number S of all the pairs $(x_i, f_R(x_i))$ from the sample (2) for which the ratio is performed is calculated $f_R(x_i)=f_3(x)$. Next, check the condition $\frac{S}{M}\ge\delta$ here (δ – a number that characterizes the recognition efficiency which is given at the beginning). If this condition is met, the process of constructing the classification tree ends, then the generalized feature $f_3(x)$, which is represented by the tree (Fig. 3) is such that provides an approximation of the sample of the form (2). If $\frac{S}{M}<\delta$ so, the tree building process continues. When building a tree, first select recognition on the tree (Fig. 3) all those values $l(r_1,r_2,r_3)$ for which the ratio is performed $t_{r_1,r_2,r_3}^{l(r_1,r_2,r_3)}=1$. Paths r_1, r_2, r_3 that perform only the specified ratio can be considered complete.

For example let:

$$t_{010}^{l(010)}=t_{011}^{l(011)}=1;\ t_{000}^{l(000)}<1;\ t_{001}^{l(001)}<1;\ t_{100}^{l(100)}<1;\ t_{101}^{l(101)}<1.$$

In this case, you will get the tree view (Fig. 4) where $m_2^4=l(010)$, $m_3^4=l(011)$.

All the ways, 100 and 101, but on the tree (Fig. 4) are unfinished. For each of these paths r_1, r_2, r_3 we consider sets H_{r_1,r_2,r_3}, where H_{r_1,r_2,r_3} is the set of all those pairs $(x_i, f_R(x_i))$ of the training sample that belong to the path r_1, r_2, r_3. Sets H_{r_1,r_2,r_3} can be considered as some samples. In the case of a tree (Fig. 4) we will have the following samples $H_{000}, H_{001}, H_{100}, H_{101}$.

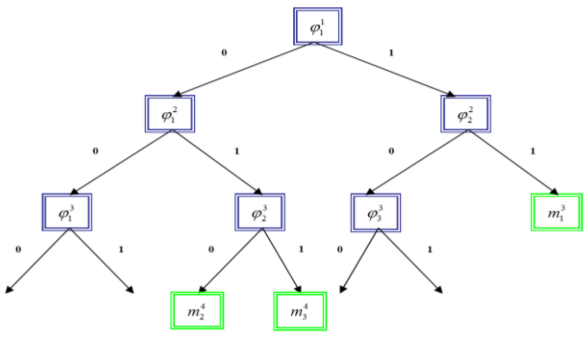

Fig. 4. General view of the final logical classification tree

4. Main results

For each sample H_{r_1,r_2,r_3}, an elementary feature ϕ_{r_1,r_2,r_3} is selected for which the value $W_{H_{r_1,r_2,r_3}}(\phi_{r_1,r_2,r_3})$ is the largest possible. The value $W_{H_{r_1,r_2,r_3}}(\phi_{r_1,r_2,r_3})$ represents the efficiency of recognition of the sample H_{r_1,r_2,r_3} using the characteristic ϕ_{r_1,r_2,r_3} [1, 6]. After selecting the features ϕ_{r_1,r_2,r_3}, we get a new tree, which is shown in (Fig. 1). Note that here $\phi_1^4=\phi_{000}$, $\phi_2^4=\phi_{001}$, $\phi_3^4=\phi_{100}$, $\phi_4^4=\phi_{101}$. Next to the tree (Fig . 1) apply the same process as to the tree (Fig. 2). It is important to note that you do not need to create a separate set of training pairs to implement each sample H_{r_1,r_2,r_3}. All these samples are implemented as follows: sample pairs (2) are submitted sequentially and only those pairs that belong to the path r_1, r_2, r_3 are taken into account. As a result of this process, sampling H_{r_1,r_2,r_3} will be implemented.

So, summarizing all the above, we can draw the following conclusions regarding the construction of logical trees:
A logical tree provides coverage of an array of educational information by fixing the sample objects in its structure. Moreover, such an approach of information storage provides-as a mechanism for further training (expansion) and error correction.
To date, in the specialized literature there is no description of algorithms and methods that would allow you to build logical trees based on large data arrays.
The use of logical trees in pattern recognition problems allows to describe recognition functions simply and compactly, which positively affects the complexity and speed of the resulting classification system.

Thus, the paper presents a step-by-step scheme for constructing a system of recognition of discrete objects based on a logical tree, and the focus of the study was a graph-scheme model of the recognition system in the form of a logical tree. Once again, we note that the results of this work are relevant for all problems of pattern recognition, in which the resulting recognition scheme can be represented as a logical tree.

References

[1] Povhan I.: General scheme for constructing the most complex logical tree of classification in pattern recognition discrete objects. Electronics and information technologies 11/2019, 112–117.
[2] Quinlan J.R.: Induction of Decision Trees. Machine Learning 1/2008, 181–222.
[3] Vasilenko Y.A, Kuhayivsky A.I, Papp S.A.: Construction and optimization of recongnizing systems. Information Technologies and Systems 1(T2)/1999, 122–125.
[4] Vasilenko Y.A., Vashchuk F.G., Povhan I.F.: Overall assessment of minimization of logical tree. European Journal of Enterprise Technologies 1(55)/2012, 29–33.
[5] Vasilenko Y.A., Vashchuk F.G., Povhan I.F.: The problem of estimating the complexity of the logic trees, recognition and general method optimization. European Journal of Enterprise Technologies 6/4(54)/2011, 24–28.
[6] Vtogoff P.E.: Incremental Induction of Decision Trees. Machine Learning 4/2009, 161−186.
[7] Zheng Z., Kohavi R., Mason L.: Real world performance of association rule algorithms. Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining 2008, 401–406.

Ph.D. Povhan Igor
e-mail: igor.povkhan@uzhnu.edu.ua

Igor Povhan, worked as the head of the laboratory of software and hardware of the faculty of information technology in the period from 2003 to 2008. Since 2009, associate Professor of the Department of Software Systems of Uzhgorod National University. The direction of scientific work-artificial intelligence, pattern recognition theory, low-level programming.

http://orcid.org/0000-0002-7034-8702

otrzymano/received: 22.12.2019 *przyjęto do druku/accepted: 26.06.2020*