Baltic J. Modern Computing, Vol. 11 (2023), No. 3, 398-410
https://doi.org/10.22364/bjmc.2023.11.3.03

Target Class Classification Recursion Preliminaries

Levon ASLANYAN[1], Karen GISHYAN[1,2], Hasmik SAHAKYAN[1]

[1] Institute for Informatics and Automation Problems of the National Academy of Sciences, Armenia, Yerevan 0014, P.Sevak 1
[2] International Scientific Educational Center of the National Academy of Sciences, Armenia, Yerevan 0019, Baghramyan 24d

lasl@sci.am, karengishyan.res@outlook.com, hsahakyan@sci.am

ORCID 0000-0002-5354-2730, ORCID 0000-0002-6239-9294, ORCID 0000-0002-8449-6845

Abstract. In traditional machine learning, geometrically "compact" sets of classes are given by representatives with the aim of correctly classifying new objects into classes. We consider an application where one class is singled out, and the goal is to classify objects into this target class by a stepwise procedure. This problem appears in the precision medicine area under the name of “dynamic treatment regime”. A predefined class-action is applied that transforms the object to the same or some other class. The mapping of class-actions to the classes forms the policy, and the chain of policy-based transformations (under some restrictions) must converge to the target class. With graph theoretical tools, we present and evaluate the policy form, and show that the graph is partitioned into a couple of structural components, which helps to find out the possible policy defects to be corrected by subject domain specialists for better results.

Keywords: Dynamic treatment regime, One class classification, Class transition policy, BFS and DFS.

1. Introduction

The subject of this article arose when considering the medical problem of dynamic treatment regimes (Murphy, 2005) in terms of problems of classifications and transitions of objects between classes. These considerations are intended to complement the traditional statistical analysis of medical sequential treatment problems in this area. The viewpoint began to take shape in works (Aslanyan et al., 2020), (Aslanyan et al., 2020), and the initial results were reported at the DAMSS conference (Aslanyan et al., 2022).

The problem under consideration has similarities with classical supervised machine learning classification, which implies several geometrically "compact" sets of elements (clusters, classes) defined by sets of representatives of these classes (elements of the so-called training set), for which there are procedures capable of correctly classifying new objects into desired classes. However, after the learning/training phase, the recognition/classification of a trial object is a static, one-step process. Unlike the traditional classification approach, our application allocates objects to a single special class by a step-by-step procedure. In a separate recursive step, when the temporal class of an object has been defined, a predefined action on the elements of the class is applied, which transfers that object into the same class or some other class. The objective of the task is to make sure a chain of such transformations will lead the objects to a marked target class. The fact that the paper deals with a classification problem using graph theory is a contribution.

For the designated special class we equivalently use the names "normal" and "target". The novelty lies in the recurrent classifications over a set of objects, in the matching of classes and class actions, and in the convergence of the classifications to the target class in as few steps as possible. We distinguish between "white box" and "black box" cases. The latter is closer to the model of backward reinforcement learning, especially to the concept of learning with apprenticeship. Since the general idea is presented in terms of classification algorithms and is a new research postulate, we think it necessary to point out the specifics of this idea, as well as to recall existing associations with current research tasks in the field of classification algorithms.

Although there is an association with the classification of unbalanced classes (Koço et al., 2013), where the need for general accuracy requires the classification of small classes to be emphasized (attention to the unseen), our classification has additional algorithmic constraints coming from the subject domain (otherwise we would apply the action that transfers the object to the target class in one step).

The second association is with sequential learning algorithms (Even-Zohar et al., 2001). Sequential learning aims to limit the classes step by step until it finds the right class for the object. In the case of our problem, sequential learning/classification aims to find a stepwise approximation of the object being classified to a predefined target class. The classification structure is again constrained based on the set of available data and rules of the subject domain.

A good example of the application of the idea of target class classification (TCC) is the medical problem of stepwise treatment of chronic diseases (Murphy, 2005). Examples of this kind are many in publications (Lavori and Dawson, 2008). The tasks of this group of medical approaches belong to the field of personalized/precision medicine and are well known in terms of dynamic treatment regime (or adaptive treatment strategies) problems (Murphy, 2005), where it is required to construct an optimal treatment policy for patients that consists of their gradual approach to a class of normal health condition.

Clarification of the subject and initial findings requires a general consideration of the field of classification algorithms, raising the need for the approach being presented as a complement to existing concepts (1.1. Notes on Classification Framework). Next, elements of the theory of adaptive treatment strategies (based on (Murphy, 2005)) are mentioned in 1.2. An Example Application Scenario. The data structure for the problem is described in 1.3. Data, and finally, one simple case of the problem with its graph-theoretic analysis is considered in 2. Target-Class Classification, the Simplest Case Analysis). The main result of this proof of concept type paper is formulated as Theorem 1. An extended version of this paper will add experimental results with medical data, aiming to provide further research results on Markov Chain and Hidden Markov Models, classification stability, the limiting behaviour of classification recursions, and reinforcement learning.

1.1. Notes on classification framework

The task of classification in the field of modern machine learning refers specifically to the field of pattern recognition (an earlier name for machine learning) (Bongard, 1967), (Chervonenkis et al., 1974), (Mohri et al., 2018), (Zhuravlev, 1998). The classification problem objective is to identify the class (property, category) to which an observed object belongs, based on the set of associated feature values. Typically, a set of training dataset observations with known class membership is available to train the algorithm, and a validation dataset is foreseen for the algorithm evaluation. The classification problem can be further categorized as:

(a) binary classification problem, in the case when the class label can take only two values,

(b) multiclass classification problem, in the case when the class label can take more than two different values, and

(c) multi-label classification problem, in the case when each observation is associated with more than one class.

Modern classification is a mixture of discrete mathematical procedures, heuristics, approximations, and estimations, with random sampling, average and worst-case analyses, and statistical estimations that are as a rule an overestimation of the reality. A wide variety of algorithms have been proposed to solve the classification problems. Example algorithms for binary classification include linear regression, logistic regression, naive Bayes, voting, support vector machines, decision tree, random forest, k-nearest neighbours, artificial neural networks, etc. It appears that depending on the specific problem type, one or the other algorithm may be more appropriate. For this reason, the performance of different algorithms should be evaluated before selecting the one that best fits the problem characteristics.

An additional idea is the sequential learning model, which utilizes classifiers to sequentially restrict the number of competing classes while maintaining, with high probability, the presence of the true outcome in the candidate set. Sequential classification is central to many practical machine learning problems. In a variety of real-world applications (ranging from protein/DNA classification, speech recognition, intrusion detection, and text classification), there is a need to classify sequence data (we distinguish sequential classification from the classification in a sequence).

Sequence data are different from the more-typical vector representation. The distances between arbitrary pairs of sequences are difficult to determine because the sequences can be of different lengths and the information contained in the order of the sequence elements is lost when applying standard metrics such as Euclidean distance. The concepts of Sequence Profile, Hidden Markov Models (HMM) (Rabiner, 1989), (Murphy, 2005), (Warnow, 2017), and HMM Profile have a rich history in sequence data modeling for classification, segmentation, and clustering. The HMMs' success is based on the convenience of their simplifying assumptions. The space of probable sequences is constrained by assuming only pairwise dependencies over hidden states. Pairwise dependencies, emission, and transition matrices also allow for a class of efficient inference algorithms, where steps are built based on the Forward-Backward algorithm. After training, the per-sequence transition matrices of the HMM are used as fixed-length vector representations for each associated sequence.

The problem considered in this paper is a new, specific classification paradigm, but it can still be interpreted in association with the properties of unbalanced classes and sequential learning. Normal imbalance is meant to protect small classes from misclassification while target class classification focuses on a single, main class which we call the target class. The task is to build an algorithm that assigns all objects to the target class. The classification and transition means are limited, and in order to achieve the goal, it is necessary to apply them consecutively, several times. Classification, or finding an action to apply, in $TCC,$ is a traditional machine learning algorithm, while the target class classification is a chain of classifications and process analysis. It is necessary to answer the question about the consistency of the framework for the final assignment of objects to the target class, as well as to build an iterative classification algorithm with the required accuracy of the final assignment of objects to the target class.

1.2. An Example Application Scenario

Let us turn to the original medical problem, from which the algorithmic problem of classification to the target class originated. Adaptive treatment strategies (also known as dynamic treatment regimes) are emerging as a tool for personalized and precision medicine as a new paradigm for the treatment and long-term management of chronic disorders such as alcoholism, smoking cessation, cocaine abuse, depression, and hypertension (Murphy, 2005). In adaptive treatment strategies, the level and type of treatment are repeatedly adjusted according to the needs of the individual (classification based on feature values).

An adaptive treatment strategy is characterized by a sequence of decision rules, one for each treatment (sequential learning). Researchers now use a combination of clinical experience, trial and error, behavioural, psychosocial, and biological theories, observational studies, and randomized experimental studies conducted for other purposes to formulate the decision rules that constitute an adaptive treatment strategy. The goal of the strategy is to bring patients to a normal state, which we interpret as the final assignment/classification to the target/normal class. In this paper, we analyze the simplest case of treatment where the process is strictly static.

The whole spectrum of cases in question is as follows:

a) A class (state) of an object (patient) is defined, and a standard (single) action (procedure) is assigned to this class; as a result of the action the class objects move to one single class.
b) As a result of the action, the class objects move to a number of other classes by certain probabilities.
c) A number of actions are possible in each step, and each action can be either deterministic, with one output, or probabilistic, with many outputs.

Our long-term goal is to optimize adaptive treatment strategies, that is, to create a treatment strategy that produces the best (or required) classification result. A number of trials have been and are being conducted in this concern.

In order to name the main goals of the TCC, let us define the objectives:

- ***Validation*** - Considering any of the cases a) - c) and the corresponding TCC solution, we need to determine whether it solves the problem of effectively assigning objects to the target class. To do this, we first need to construct such an algorithm, and then we can apply cross validation scheme or a standard control procedure from pattern recognition with a control set to obtain an estimate of success.

- ***Superiority*** - The comparison of two or more procedures is also carried out based on a consistent control set of objects or cross validation type procedures.
- ***Optimization*** is the main procedure of the TCC. It looks at all available data on the problem and analyses it by all available means. It is usually discrete mathematical analysis, it is matrix analysis, Markov chains, HMM, etc.

In most cases, the goal of these trials is to develop adaptive treatment strategies (sequences of treatments for a particular individual), and they are not confirmatory. That is, the goal does not include confirming that one adaptive treatment strategy is better than control or standard treatment. Confirmatory trials comparing an optimized adaptive treatment strategy with a corresponding control or standard treatment should follow such trials.

1.3. Data

TCC problem is a data-driven procedure.

Suppose we need to make at most k decisions, $a_1;\ a_2;\ \dots;\ a_k$ per individual. S_1 denotes pre-treatment information whereas $S_j, 1 < j \leq k$ denotes the intermediate outcome information available after decision a_{j-1} and prior to decision a_j. Thus, the time order is $S_1; a_1;\ S_2; a_2; \dots;\ S_k; a_k; S_{k+1}$. This is an individual track, and we call the set of tracks over the population of individuals a *trellis*. Denote past and present information as $\bar{S}_j = \{S_1;\ \dots;\ S_j\}$. The primary outcome is $Y = u(\bar{S}_{k+1}; \bar{a}_k)$ where u is a known summary function. In the addiction management study, Y is the main percent of days abstinent so u counts the number of days abstinent and divides by study length in days. An adaptive treatment strategy is a sequence of decision rules, one per decision. Thus, we denote an adaptive treatment strategy by the decision rules $\{d_1;\ d_2;\ \dots;\ d_k\}$ where the decision rule d_j takes the information available at time j. $S_j = \{S_1;\ \dots; S_j\}$ and past treatment $\bar{a}_{j-1} = \{a_2;\ \dots; a_{j-1}\}$ outputs a treatment type/level, a_j. For example, in the addiction management study, $k = 2$ and the possible treatments, values for a_1 at the first decision time point are med (with a low level of counseling) and cbt (cognitive behavioural therapy) (Murphy, 2005). The information available for making the second decision includes pre-treatment information denoted by S_1, the first treatment, and the intermediate outcomes denoted by S_2. The structure of the trellis can be very different. The initial states (classes) may be different, but in that case, it is possible to group tracks with the same initial state, which will simplify the preliminary analysis. The problem is also in tracks having different lengths, they may end in different states.

Sometimes only the initial and final states of a track and its length are known. Sometimes, only transitions, the previous state, and the next state are taken from the data. The data used is also characterized by the algorithm applied to classify the states. Sequence profile-type algorithms use state frequencies, while HMM profile class algorithms can use transition frequencies, both first and higher orders, which is akin to the reinforcement learning approach. The algorithm introduced in Section 2.2 is the simplest one for the simplest case of the problem. It considers first-order transitions, and the result of a transition is only one fixed value. The issue is to understand the constraints that provide assigning objects to the target class. If the experimental data do not satisfy the constraints identified by this research framework, this indicates the need to verify, update, or revise the experimental data, or requires the adoption of a new hypothesis about the processes and data.

2. Target-Class Classification, the Simplest Case Analysis

Consider case a) of the problem given in Section 1.2. Once the classes $K_0, K_1, K_2, \dots, K_l$ are given, each class K_i initiates a certain action a_i, as a result of which the object reallocates into the class $a_i(K_i) = K_j$. The class K_0 is the target (normal) class to which an action is not attached and the elements of this class are not moved anywhere. Having a system of these simply defined transitions between classes and policy, it is necessary to understand whether there is a possibility and tendency to move objects to the normal class as a result of successive classifications. Consider the following graph $G = (V, E)$ with vertices that correspond to classes, and edges with marked actions of classes, and the orientation to classes determined by the actions. The normal class K_0 is assigned to the vertex $v_0 \in V$.
In general, a directed graph (or digraph) is called *weakly connected* if the underlying undirected graph is connected. A directed graph is *unilaterally connected* or unilateral (also called semi-connected) if it contains a directed path from u to v or a directed path from v to u for every pair of vertices u, v. It is *strongly connected*, or simply *strong*, if it contains a directed path from u to v and a directed path from v to u for every pair of vertices, v. An *oriented graph* is a digraph with no cycle of length two. Undefined terms can be found in (Bang-Jensen, Gutin, eds., 2018).
The graph $G = (V, E)$ of our problem is very simple: from each of its vertices exactly one oriented edge is going out, except the vertex v_0, where no edge starts from it. Then, - what is the structure of this graph? Usually connected component is given as sets of vertices and edges. We are interested to obtain more information, in terms of cycles, trees, and orientations calling these configurations structural components of G. Some of these components represent cactus graphs, defined as connected graphs in which any two graph cycles have no edge in common. We will describe the structure in terms of connected components of the underlying undirected graph of G, and additionally, mention the orientations of edges. Some fragments of the structure satisfy the properties of unilateral or strongly connected components that we will mention accordingly.

2.1. Structure of TCC graph G

Theorem 1. The underlying graph of TCC graph G consists of one tree G_0, rooted at v_0 and may have several other structural components structured as one-cycle cactus graphs. Orientation of edges in G_0 are towards the vertex v_0. In cacti, the orientation of the edges of cycles is one way and other edges are oriented towards the cycles.

Proof. First, let us prove the existence of the tree G_0, rooted at vertex v_0 (we call it a normal-tree) by constructing it. Consider the vertex v_0 and the set of oriented edges that enter into v_0. Suppose that these edges originated from the vertices $v_{1_1}, v_{1_2}, \dots, v_{1_{k_1}}$. Geometrically, we place vertices layer by layer: v_0 at the lower layer, then $v_{1_1}, v_{1_2}, \dots, v_{1_{k_1}}$ at the next layer, and so on.

Since every vertex in G has only one outgoing edge, then $v_{1_1}, v_{1_2}, \dots, v_{1_{k_1}}$ exhaust their outgoing edges and therefore will not appear further in the upper part construction of the tree. In the next step, we sort out the vertices $v_{1_1}, v_{1_2}, \dots, v_{1_{k_1}}$ of the first layer and collect all the edges entering into these vertices from some collection $v_{2_1}, v_{2_2}, \dots, v_{2_{k_2}}$. We place these vertices on layer two. This procedure, repeated recursively, ends at some step m with the construction of a tree rooted at v_0, with one edge going down from all internal (non-root) vertices to the vertices of the adjacent layer below. But, it is clear that all vertices except the terminal ones (leaves) can contain multiple ingoing oriented edges. Now, if the graph is connected, then its structure of G is already defined (see component G_0 in Figure 1).

We mention that the underlying undirected graph G_0 represents a connected component of the underlying graph of G, because no edges are emanating this subgraph and no new edges may enter into it. The description is given in terms of underlying graphs because G_0 does not have properties neither unilateral nor strong connectivity.

The case of connected graph G is completed.

Now, suppose that G is not connected (there are still vertices outside G_0), and let $G_1, G_2, \dots, G_l$ be the other connected components of its underlying undirected graph. Consider an arbitrary component G_i. First, we claim that there must be a cycle in G_i. This is explained by the finite number of vertices in the components, where each vertex can give exactly one outgoing oriented edge to one other vertex. A cycle can be constructed as follows: we start from an arbitrary vertex, go by the outgoing edge of this vertex to the edge endpoint vertex, and repeat this step sequentially. The resulting path is forced to be closed due to the limited expansion space. We prove that this is a unique cycle. Suppose that there is a second cycle in G_i. Then, these cycles must have a common vertex, since G_i is connected. But in this case, from this common vertex, there must emanate two oriented edges for two cycles, which is a contradiction.

In this manner, each of the connected components has a single cycle. There are no outgoing links from the cycle. Trees similar to the normal-tree can converge to the vertices of this cycle, and thus the structure of such component itself is provided by the one-cycle cactus graph, which completes the proof of the theorem. ■

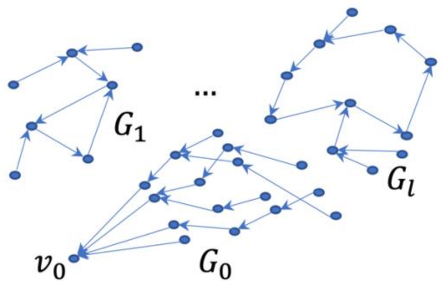

Figure 1. G_0 is the basic component, rooted and oriented to the vertex v_0. All other components consist of one oriented cycle, plus several trees, entering and oriented into a vertex of the component's cycle. The cycle itself is strongly connected but the entire component is in terms of underlying graph connectivity.

Considering Theorem 1. from the point of view of the concerned practical problem, we should note the following. If analysis of the experimental data of the problem shows that the transition graph constructed by available data does not correspond to the above-considered restrictions (supposed in subproblem a)), then this indicates non-compliance with the envisioned policy, and the question should be transferred to the decision makers of the applied problem. If these conditions are met but there is more than one component of G, then the question arises again in terms of accessibility, and it is necessary to make changes in the form of applying updated actions to the classes so that the additional components $G_1, G_2, \dots, G_l$ properly merged with the component G_0. To answer these questions, we have to compute the $G_0, G_1, G_2, \dots, G_l$ representation of the graph G.

2.2. Algorithm of construction of structural components of simple TCC graph G

Input information of TCC problem is the graph G, given by its sets: V, of vertices, and E, of edges. In this section, we consider algorithms that construct and compute the structural components of G. The complexity of algorithms is also an issue of our discussion.

According to Theorem 1. the structural parts $G_0, G_1, G_2, \dots, G_l$ of G correspond to the connected components of the underlying undirected graph G' of G by adding a proper orientation to their edges. G' is a simple graph, and algorithms for constructing connected components of these graphs are well-investigated and well-known. The time complexity of finding all connected components in ordinary graphs is O(|V| + |E|). Basic algorithms use the breadth or depth-first search by the set of vertices (Tarjan, 1972). In fact, data sizes in TCC are not big and these algorithms can be used to obtain the structure. But a couple of factors justify this new reference to these algorithms. First, we may need to justify the complexity formula having that in our particular case the number of graph edges is limited by $|E| = |V| - 1$. Further, the algorithm we construct is a mixture of BFS and DFS and it is to check that rounds of these procedures might be combined into one-pass data. Finally, it is not enough to have connected components as sets of vertices, but the structure of these components is necessary for our needs. On this basis, let us consider the following data structure:

Table 1. The data structure of TCC policy graph analysis.

Classes, vertices	Transition to	Oriented path to v_0	Incoming lists to vertices
v_0	$\emptyset$	$\emptyset$	$L_0 = \{l_{01}, \dots, l_{0k_0}\}$
v_1	v_{j_1}	δ_1	$L_1 = \{l_{11}, \dots, l_{1k_1}\}$
v_2	v_{j_2}	δ_2	$L_2 = \{l_{21}, \dots, l_{2k_2}\}$
…	…	…	…
v_i	v_{j_i}	δ_i	$L_i = \{l_{i1}, \dots, l_{ik_i}\}$
…	…	…	…
v_{n-1}	$v_{j_{n-1}}$	δ_{n-1}	$L_{n-1} = \{l_{(n-1)1}, \dots, l_{(n-1)k_{n-1}}\}$

The first column of this table starts with vertex v_0 and continues with other vertices of G in an arbitrarily fixed order. The second column vertex v_{j_i}, together with the first column vertex v_i indicates to the edge (v_i, v_{j_i}), that comes out from the vertex v_i and enters into the v_{j_i}. The third column will code the existence of an oriented path from vertex v_i to vertex v_0 and this field will be filled a bit later. The fourth column "incoming lists to vertices" will contain lists L_i collecting all vertices with directed edges to the vertex v_i. Some of the lists may be empty.

The algorithm works as follows:

- Making a single pass through the first two columns of the table, the fourth column is filled in the following way: being in the i-th row of the table, v_i is added to the "incoming lists" of the j_i-th row. In the lists, the vertices are arranged according to the order of vertices in the first column. The total amount of information in the fourth column does not exceed n, because every vertex cannot be inserted in two or more lists.
- We may do a limited second pass through the table, when the lists/rows are rearranged, forming the layers of the tree G_0: list L_0 follows vertex v_0, forming the first (after vertex v_0) tree-layer of G_0, then the lists corresponding to vertices from L_0 are placed, thus forming the second tree layer, and so on.

Rearrangement of the lists as tree-layers is possible during a single (second) pass through the table because, for a vertex from these lists, its corresponding list is located directly in the row given by the number of this vertex in the first column, and by the "incoming lists" cell of this row. During this pass, the height (the number of layers) h of the constructed tree is also determined. Thus, at most $2n$ operations of reading and readdressing are performed during two passes, which is the complexity characterization of this algorithmic fragment. The memory occupies so far $3n$ cells but this is reducible when necessary using bit-based coding.

A special note has to be made about the tree leaves. After the second pass, rearranging the lists, a set of vertices with empty lists is formed at the end of the rearranged part of the table. At this level, all vertices have empty lists although in lower layers not all vertices are leaves, and not all lists are empty. All vertices that participated in the second pass including leaves are marked in the third column with a "+" sign indicating the existence of an oriented path from these vertices to the vertex v_0. The subsequent vertices of the first column are not visited by the second pass since they are not connected by an edge going from them to the constructed tree with the root v_0. These vertices are marked in the third column with a "-" sign. We may also introduce labels indicating the number of the layer for lists, and labels indicating leaves.

We conclude that building a tree is a low linear procedure in terms of the complexity of maintained data and memory used. Similar consideration is valid for the rest of the table when the algorithm constructs the cactus-type components. In this case, the process starts from 0-indegree vertices that form the cactus branches. And one pass is sufficient to form all cycles and cacti bunches.

After this introduction to the topic let us turn to the existing classic theory of graph theoretical algorithms.

In graph theory, Depth First Search (DFS) and Breadth First Search (BFS) are multipurpose graph exploration strategies. The power of these simple procedures to understand graph structure is demonstrated in obtaining linear time algorithms for finding cut-edges and cut-vertices of *undirected graphs*, finding strongly connected components of *directed graphs*, for testing whether a graph is planar. Simple DFS or BFS *recursion* constructs the connectivity forest of a graph. Tarjan's strongly connected components algorithm (Tarjan, 1972) is one of the algorithms that constructs the strongly connected components (SCCs) of directed graphs $G(V,E)$ represented by adjacency lists for each $v \in V$. These algorithms run in low linear time $O(|V| + |E|)$, similarly to the alternative methods including Kosaraju's algorithm and the path-based strong component algorithm. DFS or BFS begins a recursion from an arbitrary start node, and consequently, from any node that has not yet been visited. This search visits every node of the graph exactly once. Thus, the collection of search trees is a spanning forest of the graph. The procedure requires limited supplementary data per vertex. But if the graph is represented by the adjacency matrix, the algorithm may require $O(|V|^2)$ time. The order in which the strongly connected components are identified constitutes a reverse topological sort of the directed acyclic graph formed by the strongly connected components.

The final decision on the algorithm to be used for TCC decides in favor of Tarjan's algorithm with some modifications. This algorithm is well-known and will not be discussed here in detail. We just have to start with an important comment:

The recursive step in DFS or BFS starts with an arbitrary unvisited vertex and ends with forming a connectivity fragment of the graph. Both searches do the same work but with different priorities of vertices. And while so, the DFS and BFS can be applied in recursions interchangeably. This comment is applied in the algorithm below.

Algorithm TCC:

A. Start from vertex v_0 and treat the graph as an ordinary one, i.e. we ignore the orientation of edges. As a result, BFS recursion (as it was discussed earlier in this point) constructs the component G_0 of G. The particular use of BFS gives us the layers of the tree as well.

B. Choose arbitrary $v \in G \backslash G_0$ and apply Tarjan's procedure (DFS) to it. A supplementary array $num(v_i)$ keeps the consecutive number of visited vertices. Let a node repetition mentioned in Theorem 1 appears at the vertex u by the current edge (w,u). The vertices with consecutive numbers from $num(u)$ until the $num(w)$, compose the cactus cycle. The path $v, \dots, u$, in reverse order, will be used again in Tarjan's procedure in the construction of cactus bunches when necessary.

C. Uses one more supplementary array $rev(v_i)$ (this can be a bit level array) that is indicating vertices of the cactus cycle with “+” when above the incoming and outgoing edges of the cycle at v_i there exists at least one more edge incoming to the v_i. This will be a bunch edge and this bunch starts at the vertex v_i.

D. Apply point A. on each vertex with $rev(v_i) = "+"$ of the construction and proceed with B., if the vertices of G are not expired. We construct bunches of current cactus layer-by-layer and all Theorem 1 cacti, one after the other.

2.3. TCC defects and their correction

Having Theorem 1. and algorithm TCC, and the fact that TCC solution is correct only when its transition graph structure is a tree with root v_0 and with an orientation of edges to v_0 it is necessary to estimate the degree of displacement of graph structure G from this desired structure. There are two types of defects: critical defects and defects related to optimality. The characteristic of the latter is determined by the height of the tree G_0 where the minimum value is preferable. Critical defects include problems with the existence of many components, as well as the existence of cacti with their cycles. In this case, we are dealing with a medically egregious defect, when the classes of states (patients) can in no way pass to the normal class based on the treatment policy applied. The presence of a cycle in the cactus bulla aggravates the situation because the cycle indicates the possibility of infinite transition between classes without entering the normal class.

Therefore transformations of structural components of graph G are needed to restore the connectedness of the graph and to break cycles. However, if one can transform the graph by introducing and reconstructing the transitions in such a way that the bouquet vertices go to G_0, then this does not correct the defect since all cycle vertices remain directed-isolated from v_0. To restore the cycle's connection to G_0 it is necessary to break the cycle so that the only outgoing edge of one of its vertices goes to the vertices of the G_0 component. Then all vertices of cactus bouquets will also get directed paths to the vertex v_0. This also indicates the minimum required information about the structural composition of the graph, which provides the search and correction of defects in the TCC. It is enough to have a graph G_0 as a set of vertices, and it is necessary to have all cycles of cacti, possibly, as sets of edges.

The specified transformation is a recommendation to the medical institution based on a logical analysis of the data, and the right to decide what is possible and what to change of course belongs to the medical side.

3. Conclusion

Among the unbalanced and recursive classification algorithms, there is a practical need to develop new algorithms that assign objects to one predetermined class through sequential classifications and transitions. On a practical level, this research is related to the needs of precision medicine, where transition rules are defined and specified in the form of the policies they apply. The main research in this direction is conducted in terms of stochastic modeling and reinforcement learning. In this paper, a logical analysis of the transition graph (policy) of the problem is carried out, where the existing defects of transitions, and possibilities of their elimination are determined.

Acknowledgments

This paper is partially supported by grant №21SC-BRFFR-1B029 of the Science Committee of MESCS RA.

References

Allwein, E. L., Schapire, R. E., Singer Y. (2000). Reducing multiclass to binary: A unifying approach for margin classifiers. *Journal of machine learning research*, 1(Dec), pp. 113–141.

Aslanyan, L., Krasnoproshin, V., Ryazanov, V., Sahakyan H. (2020). Logical-combinatorial approaches in dynamic recognition problems. *Mathematical Problems of Computer Science* 54, IIAP NAS RA, Yerevan, pp. 96-107.

Aslanyan, L., Ryazanov, V., Sahakyan, H. (2020). On logical-combinatorial supervised reinforcement learning. *Information Theories and Applications*, ITHEA, Vol. 27, Number 1, pp. 40-51.

Aslanyan, L., Sahakyan, H., Gronau H. D., Wagner P. (2015). Constraint satisfaction problems on specific subsets of the n-dimensional unit cube. In 2015 *Computer Science and Information Technologies (CSIT)*, pp. 47–52. IEEE.

Aslanyan, L., Sahakyan H. (2017). The splitting technique in monotone recognition. *Discrete Applied Mathematics*, 216, pp. 502–512.

Aslanyan, L., Gishyan, K., Sahakyan, H. (2022). Deterministic Recursion in Target Class Classification. Proceedings of the 13th Conference on Data analysis methods for software systems, *Vilnius University Proceedings* 31 (2022), DOI 10.15388/DAMSS.13.2022

Bang-Jensen, J., Gutin, G. eds.: *Classes of directed graphs*, Cham: Springer (2018)

Bongard, M. (1967). *Problem of Cognition* (in Russian). Fizmatgiz, Moscow.

Chakraborty, B. (2013). *Statistical methods for dynamic treatment regimes*. Springer-Verlag, DOI 10:978–1 (2013)

Chervonenkis, A. Ya. Vapnik, V. N. (1974). *Pattern recognition Theory* (in Russian), Nauka, Moscow.

Cortes, C., Vapnik, Vl. (1995). Support-vector networks. *Machine learning*, 20(3), pp. 273–297.

Dietterich, T. G., Bakiri G. (2018). Error-correcting output codes: A general method for improving multiclass inductive learning programs. In *The Mathematics of Generalization*, pp. 395–407. CRC Press.

Even-Zohar, Y., Roth, D. (2001). A sequential model for multi-class classification. *EMNLP* (2001), pp.10-19.

Guo, H., Li, J., Liu, H., He, J. (2022). Learning dynamic treatment strategies for coronary heart diseases by artificial intelligence: real-world data-driven study. *BMC Medical Informatics and Decision Making* 22, no. 1 (2022): 1-16.

Knerr, S., Personnaz, L., Dreyfus G. (1990). Single-layer learning revisited: a stepwise procedure for building and training a neural network. In *Neurocomputing*, pp. 41–50, Springer.

Lavori, P. W., Dawson, R. (2008). Adaptive treatment strategies in chronic disease. *Annu. Rev. Med.* 59 (2008): 443-453.

Liu, N., Liu, Y., Logan, B., Xu, Z., Tang, J., Wang, Y. (2019). Learning the dynamic treatment regimes from medical registry data through deep Q-network. *Scientific reports* 9, no. 1, 1-10.

Liu, Y., Logan, B., Liu, N., Xu, Z, Tang, J., Wang, Y. (2017). Deep reinforcement learning for dynamic treatment regimes on medical registry data. In 2017 *IEEE international conference on healthcare informatics (ICHI)*, pp. 380-385. IEEE, 2017.

Mohri, M., Rostamizadeh, A., Talwalkar, A. (2018). *Foundations of machine learning*. MIT press.

Murphy, S. A. (2005). An experimental design for the development of adaptive treatment strategies. *Statistics in medicine*, 24(10), pp.1455–1481.

Koço, S., Capponi, C. (2013). On multi-class classification through the minimization of the confusion matrix norm. In *Asian Conference on Machine Learning, PMLR*, pp. 277–292.

Rabiner, L. R. (1989). A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2), pp.257–286.

Rocha, A., Goldenstein S. K. (2013). Multiclass from binary: Expanding one-versus-all, one-versus-one and ECOC-based approaches. *IEEE Transactions on Neural Networks and Learning Systems*, 25(2), pp. 289–302.

Sahakyan, H. (2009). Numerical characterization of n-cube subset partitioning. *Discrete Applied Mathematics*, 157(9), pp. 2191–2197.

Tarjan, R. E. (1972), Depth-first search and linear graph algorithms, *SIAM Journal on Computing*, 1(2): 146–160, DOI 10.1137/0201010

Zhuravlev, Yu. I. (1998). *Selected research work*. Magister, Moscow.

Zhuravlev, Yu. I., Aslanyan,L. A., Ryazanov V. V. (2014). Analysis of a training sample and classification in one recognition model. *Pattern recognition and image analysis*, 24(3), pp. 347–352.

Zhuravlev, Yu. I., Aslanyan, L. A., Ryazanov, V. V., Sahakyan H. A. (2017). Application driven inverse type constraint satisfaction problems. *Pattern Recognition and Image Analysis*, 27(3), pp. 418–425.

Zhuravlev, Yu. I., Ryazanov, V. V., Aslanyan, L. H., Sahakyan H. A. (2019). On a classification method for a large number of classes. *Pattern Recognition and Image Analysis*, 29(3), pp. 366–376.

Zhuravlev, Yu. I., Ryazanov, V. Vl., Ryazanov, Vl., V., Aslanyan, L. H., Sahakyan H. A. (2020). Comparison of different dichotomous classification algorithms. *Pattern Recognition and Image Analysis*, 30(3), pp. 303–314.

Warnow T. (2017). *Computational phylogenetics: an introduction to designing methods for phylogeny estimation*. Cambridge University Press.

Received February 21, 2023, revised June 5, 2023, accepted August 7, 2023