Training sequence decomposition

Dekompozycja ciągu uczącego

Swietłana Lebiediewa[1]

Treść: Sformułowano problem dekompozycji ciągu uczącego (CU). Zdefiniowano dwa rodzaje dekompozycji CU dla scentralizowanej bazy danych (SBD). Udowodniono twierdzenia dotyczące zajętości pamięci przez CU po dekompozycji. Oszacowano złożoność obliczeniową algorytmów dekompozycji. Przedstawiono wyniki eksperymentu obliczeniowego ilustrującego zajętość pamięci w zależności od rodzaju dekompozycji, redundancji cech w drzewie i na ścieżce oraz wysokości drzewa.

Słowa kluczowe: baza danych, rozpoznawanie wieloetapowe, ciąg uczący, dekompozycja

Abstract: The problem of decomposition of a training sequence (TS) is formulated. Two types of TS decomposition for a centralized database are formulated. Theorems concerning memory occupancy by the TS after decomposition are proved. The calculation complexity of decomposition algorithms is estimated. The results of a calculation experiment are presented that illustrate memory occupancy depending on the decomposition type, the redundancy of features in the tree, the path, and the tree height.

Keywords: database, multistage recognition, training sequence, decomposition

1. The problem formulated

Recognition algorithms with learning use the training sequence (TS) [1, 3, 4]. The training sequence is a sequence of properly classified patterns, the elements of the TS being (x_k, k) pairs, where x is the vector of the values of the features of the pattern, and k is the class number [1, 2, 4, 5]. An example is the decision tree (DT) in Figure 1 and the corresponding TS in Figure 2.

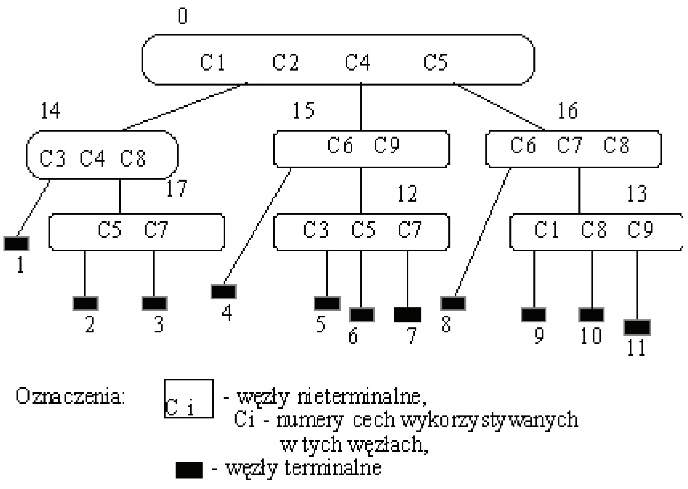

Figure 1. Decision tree No. 1a.

The training sequence presented in Figure 2 has the symbol * at the place of some features. This symbol means that a given feature is irrelevant to the pattern being recognized and not taken into account by the recognition algorithm. The multistage recognition process uses only some elements of the pattern feature vector at various recognition stages. For instance, recognition algorithms only use features C1, C2, C4, and C5 at node 0, and features C3, C4, and C8 at node 14. Additionally, certain features do not occur on each path at all.

C1	C2	C3	C4	C5	C6	C7	C8	C9	CLASS
16	3.44	3	6.55	40.0	48	116	*	14.9	7
15	3.45	2	8.45	40.0	*	*	120	*	1
14	2.4	1	8.33	38.1	*	110	*	*	2
18	3.0	*	7.5	35.0	40	*	*	15.0	4
15	2.7	*	6.0	20.0	35	110	112	*	8

Figure 2. A fragment of the training sequence for DT No. 1a.

Let us use the following designations: CN – the number of elements of feature vector C_n (n=1, 2, ..., CN); K – the number of classes; EN_k – the number of TS elements for class k; UCN_k – the number of features irrelevant to class k (features not used in the process of recognizing a pattern belonging to class k). The number of unused memory units is expressed by the following formula:

$$\sum_{k=1}^{K} LCN_k * LE_k \qquad (1)$$

For the decision tree in Figure 1, the number of all features $CN = 9$, the number of all classes $K = 11$. Assume that the TS element number is the same for every class, e.g. 100; then, under formula (1), the memory waste is 1900 units, i.e. more than 19%. The memory waste is the greater, the fewer features are used by the recognition algorithm on the path from the root of the decision tree to the non-terminal node that is the direct predecessor of the class. E.g., for tree No. 1b (Fig. 3), the memory waste under the same assumptions is over 54.11%.

1. Faculty of Computer Science, Wroclaw School of Information Technology, ul. Wejherowska 28,54-239 Wrocław, Poland, swietlana@lebiediewa.com

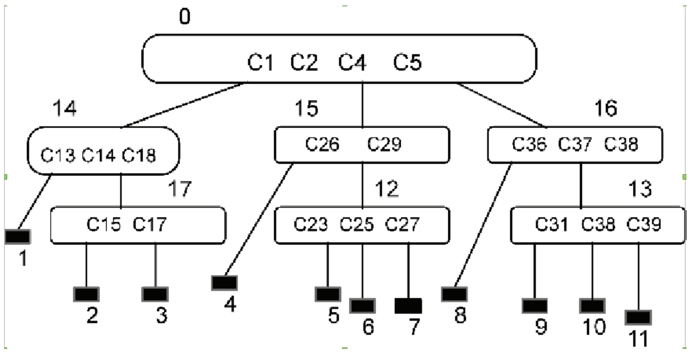

Figure 3. Decision tree No. 1b.

To conserve memory, it is proposed that the TS be decomposed. For a centralized database (CDB), I propose two types of decomposition : decomposition of the first type and decomposition of the second type. In decomposition of the first type, for every node that is the direct predecessor of a terminal node, a TS subsequence is formed that includes the values of only those features that are used along the entire path from the root of the decision tree (DT) to the node concerned until the decision is made to classify the pattern as a member of the class available directly from the node. In decomposition of the second type, for every non-terminal node, a training sequence subsequence is formed that includes the values of only those features that are used in the node.
It can be seen from the recognition algorithms presented in [5, 6] that the time of RA work at each node consists of the time needed to create the data segment and the RA work time. An appropriate form of the TS may shorten the time needed to create a data segment. The purpose of decomposition is to split the TS into subsequences that

1. minimize memory occupancy by the TS;
2. minimize the time needed for pattern recognition.

2. TS decomposition for a centralized database

Decomposition algorithms use information about the TS and the structure of the DT included in the SEQUENCE, CLASS, NODE, and FEATURE relations of the DB conceptual model. The SEQUENCE relation contains information about the TS. The CLASS relation describes the dependency between the class number and the number of the node that is the direct predecessor of the class. For every class, its predecessor is indicated. The NODE relation contains information about the number of every node, its direct successors and predecessors. The FEATURE relation contains information about which nodes use each feature [5, 6].

ALGORITHM 1. (decomposition algorithm 1 of the TS type)

Data:	REC DB conceptual model – SEQUENCE and CLASS relations
To be found:	For every node that is the direct predecessor of a terminal node, form a TS subse quence that includes the values of only those features that are used along the path from the root of the decision tree to the node concerned until the decision is made.
STEP 1.	Take the numbers of nodes that are the di rect predecessors of the terminal nodes from the CLASS relation.
STEP 2.	Create a training subsequence for every such node: take rows from the TS for which CLASS NO = “class reachable directly from the node”,placing theresult in the *R* relation;
STEP 3.	Map the *R* relation to attributes, the numbers of features used on the path from the root to the node and the CLASS NO). **STOP**

Memory occupancy by a subsequence connected with node *i* is expressed by the following formula:

$$ZPD1_i = \sum_{k=1}^{K_i} (LC_i + 1) * LE_k \qquad (2)$$

We will designate memory occupancy by all subsequences of the training sequence obtained as a result of ALGORITHM 1 as *MOD1*. Memory occupancy by these subsequences as expressed by formula (3): where where J - the number of nodes that are direct predecessors classes:

$$ZPD1 = \sum_{i=1}^{J} \sum_{k=1}^{K_i} (LC_i + 1) * LE_k \qquad (3)$$

Decomposition of the first type leads to the most optimal form of the TS in terms of memory occupancy; every TS reduction obtained as a result of decomposition of the first type would lead to a loss of information. What may happen, however, is that the time needed to form a TS fragment used in a certain segment (FRAGSEQ*) from the TS obtained as a result of the decomposition of the first type can be longer than the time needed to form a FRAGSEQ* relation from an undecomposed TS. The time is longer as a result of the need to perform a larger number of operations.

In decomposition of the second type, for every non-terminal node, a training sequence subsequence is formed that includes the values of only those features that are used in the node.

ALGORITHM 2. (decomposition algorithm 2 of the TS type for the CDB)

Data:	DB conceptual model – SEQUENCE, NODE, and FEATURE relations
To be found:	For every non-terminal node, form a training sequence subsequence that

```
            includes the values of only those features
            that are used in the node.

STEP 1.     Add a column containing TS element
            identifiers to the SEQUENCE relation
            storing the TS.
STEP 2.     Take the numbers of all non-terminal
            nodes from the NODE relation.
STEP 3.     For every non-terminal node, form a
            training subsequence (take those row
            from the TS for which CLASS NO = "
            call reachable from a given node";
            remove those attributes from the obtained
            relation that contain feature values
            unused in the node).
            STOP
```

Let n be the number of rows of the TS. The TS memory complexity is given by the formula $O(n^2)$.
Denote the number of all non-terminal nodes of the DT as N; the number of features used by the recognition algorithm at node j as C_j; the number of classes reachable from node j as K_j; the number of TS elements for class k as EN_k; and memory occupancy by the training sequence obtained as a result of decomposition of the second type as *MOD2*. Then, the formula for FRAGSEQ* TS memory occupancy at node j (on the assumption that each sequence contains an additional column for the value of the training sequence element identifier) has the following form:

$$ZDD2_j = \sum_{k=1}^{K_j} (C_j + 2) * LE_k \qquad (4)$$

The formula for memory occupancy by all TSs obtained as a result of decomposition of the second type is as follows:

$$ZPD2 = \sum_{j=1}^{N} \sum_{k=1}^{K_j} \left(C_j + 2\right) * LE_k \qquad (5)$$

The following conclusions follow from the memory occupancy formulas:
Conclusion 1. Memory occupancy by TSs arising from decomposition of the first type is unaffected by the redundancy of features on the path from the tree root to the class or by the length of the path. It only depends on the number of features on the path.
Conclusion 2. *MOD2* is unaffected by the length of the DT feature vector. It only depends on the number of features used at each node and on the number of classes reachable from that node.
Conclusion 3. *MOD2* depends both on the redundancy of features on the path from the root to the class or on the length of the path.
We denote by TMO total memory contents for the TS, and by UCN - number of characteristics unused (irrelevant) in the process of recognition.
Theorem 1. Memory occupancy by TSs obtained as a result of decomposition of the first type is no greater than memory occupancy by the TSs before the decomposition, so the following dependency occurs:

$$MOD1 \leq TMO$$

(the inequality is sharp if $UCN \neq 0$)
Theorem 2. Memory occupancy by TSs obtained as a result of decomposition of the first type is always smaller than memory occupancy by the TSs obtained as a result of decomposition of the second type, so the following dependency occurs:

$$\mathrm{MOD1} < \mathrm{MOD2}$$

3. Calculation experiment

A calculation experiment was carried out examining memory occupancy of TSs obtained as a result of decomposition of the first and of the second type for various trees, depending on the number of features in the tree and on the path, the height of the tree, and earlier-stage recognition. The following trees were considered: No. 4a (Fig. 4a), No. 4b (Fig. 4b), and No. 4c (Fig. 4c). The number of features: 20; the number of classes: 10; tree height: 3; feature use on the path: 47%; TS size before decomposition: 21000. Tree No. 4a has no redundancy of features on the path. Tree No. 4b has the redundancy of two features at nodes on different paths. Tree No. 4c has the redundancy of three features at nodes on different paths. Memory occupancy by TSs obtained as a result of decomposition of the first type and the memory savings after decomposition are presented in Table 4.4a. Memory occupancy by TSs obtained as a result of decomposition of the second type and the memory savings after decomposition are presented in Table 4.4b. A chart showing the TS memory savings after decomposition of first and the second type for trees No. 4a–4c is presented in Figure 5. Memory occupancy by TSs obtained as a result of decomposition of the first type and the memory savings after decomposition are presented in Table 4.5 a. Memory occupancy by TSs obtained as a result of decomposition of the second type and the memory savings after decomposition are presented in Table 4.5b. A chart showing the TS memory savings after decomposition of first and the second type for trees No. 5a–5c is presented in Figure 4.16.

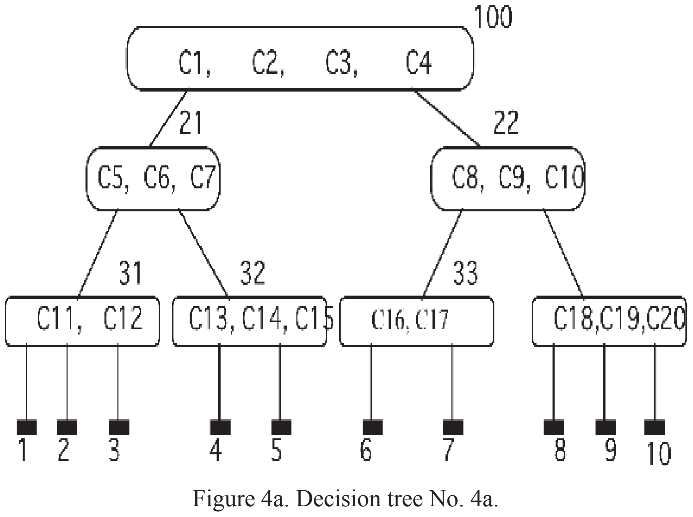

Figure 4a. Decision tree No. 4a.

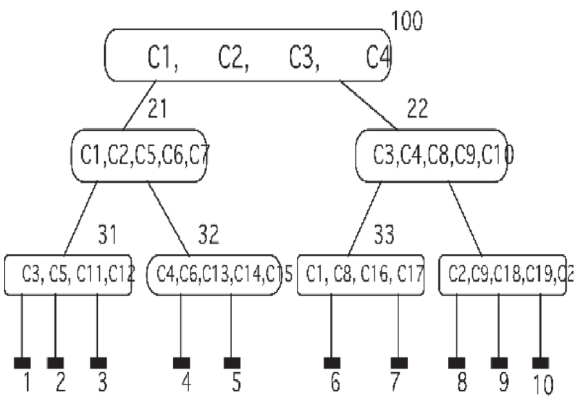

Figure 4b. Decision tree No. 4b.

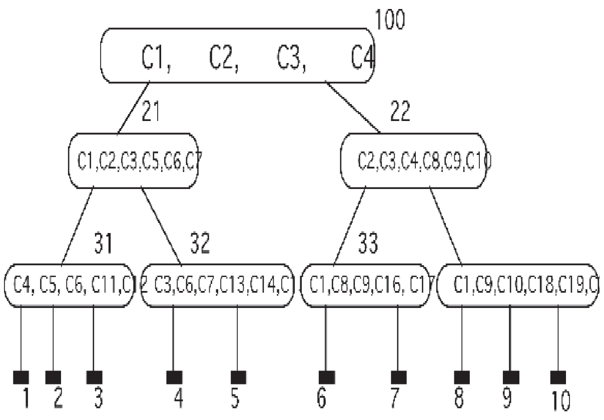

Figure 4c. Decision tree No. 4c.

Table 4.4a. Memory occupancy by TSs obtained as a result of decomposition of the first type (CDB).

TREE NO.	Tree No. 4a	Tree No. 4b	Tree No. 4c
MEMORY SAVINGS	50%	50%	50%

Table 4.4b. Memory occupancy by TSs obtained as a result of decomposition of the second type (CDB).

TREE NO.	Tree No. 4a	Tree No. 4b	Tree No. 4c
MEMORY SAVINGS	26.19%	7.14%	(2.38%)

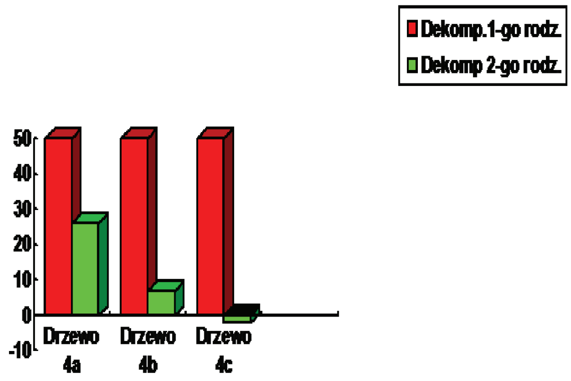

Figure 4.5. A chart showing the TS memory savings after decomposition of first and the second type for trees No. 4a–4c.

Decomposition of the first type always results in memory savings, especially where the number of all features in a DT is much greater than the number of features used on any path (cf. tree No. 1c in Figure 3). Decomposition of the second type is not always favourable in terms of memory occupancy. Decomposition of the second type gives the best results if there is no redundancy of features on any path. The smaller the percentage of feature use on a path, the greater the memory savings upon decomposition both of the first and of the second type.
A smaller percentage of feature use on a path improves the results of decomposition both of the first and of the second type. Even in the case of the redundancy of four features on a path, there are no memory wastes. The result is even better where some of the patterns are recognized in the first stage.
We observe clear memory savings for decomposition of first type. For decomposition of the second type, memory savings occur if there is no redundancy. In the case of the redundancy of four features on a path, TSs obtained as a result of decomposition of the second type require more memory than undecomposed TSs. If some patterns are recognized at the first level, there are clear memory savings for decomposition of both the first type and the second type even in the case of the redundancy of features on a path. A binary tree is a good example indicating a tendency for memory occupancy to be reduced in the case of a decomposed TS. Memory savings for two-, four-, six-, and eight-level binary trees are shown in Figure 4.

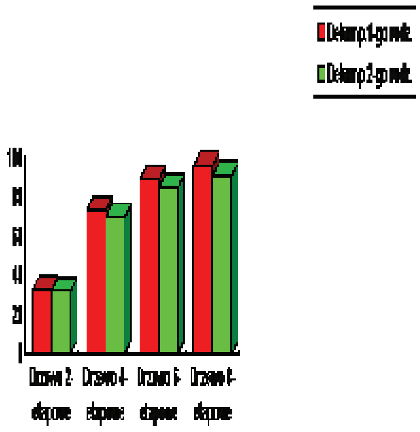

Figure 4. A chart showing the TS memory savings (percentage) after decomposition of the first type and the second type for two-, four-, six-, and eight-level binary trees.

4. Conclusions

The examples cited show that the greatest memory saving effect results from the use of decomposition of the first type in the case of a CDB because decomposition of the first type is not sensitive to feature redundancies on a path or to path lengths. Particularly good results are obtained in the case of a smaller number of features on a path relative to the number of features in the DT. Decomposition of the second type does not always result in memory savings. Decomposition of the second type requires more memory than decomposition of the first type. The best results are given by decomposition of the second type in the case of no feature redundancy and 'longer' trees. For both decompositions, the recognition of a certain number of classes at earlier stages has a positive effect on memory savings. With large numbers of features and 'long' trees, both decompositions have a very large effect on memory savings. Training sequences obtained as a result of decomposition of the second type have the additional advantage that they are identical with fragments of TSs used by recognition algorithms at the node, so if decomposition of the second type is used, it is unnecessary to create special FRAGSEQ*i* relations (*i* being the node number) when creating an external model, which means a considerably shorter processing time.

References

[1] Bubnicki, Z. 'Knowledge-Based Approach as a Generalization of Pattern Recognition Problems and Methods'. *Systems Science* Vol. 19, No 2 (1993), pp. 5–21.
[2] Flasiński, M. *Wstęp do sztucznej inteligencji*. Warsaw: PWN, 2011.
[3] Kurzyński, M. *Algorytmy rozpoznawania wieloetapowego oraz ich zastosowania medyczne i techniczne*. Wrocław: Wyd. PWr., 1987.
[4] Józefczyk, J. 'Rozpoznawanie i zastosowania biomedyczne' [in:] *Problemy automatyki i informatyki*, Wrocław: Wyd. Ossolineum, (1998), pp. 45–58.
[5] Lebiediewa, S, 'System informatyczny dla wieloetapowego rozpoznawania obiektów'. *Biuletyn Naukowy WWIS. Informatyka,* 2014.
[6] Lebiediewa, S. *Metodologia projektowania problemowo zorientowanych baz danych do systemów wielostopniowego podejmowania decyzji*. Wrocław: Wyd. Pwr., 1998.
[7] Lebiediewa, S., Zarzycki, H., and Dobrosielski, W. T. 'A new approach to the equivalence of relational and object-oriented databases', [in] Novel Developments in Uncertainty Representation and Processing, Springer International Publishing (2016), pp 85-93.