CT-EXT: An Algorithm for Computing Typical Testor Set

Guillermo Sanchez-Díaz[1] and Manuel Lazo-Cortés[2]

[1] Center of Technologies Research on Information and Systems, UAEH,
Carr. Pachuca-Tulancingo Km. 4.5, C.P. 42084, Pachuca, Hgo., Mexico
sanchezg@uaeh.reduaeh.mx
[2] Institute of Cybernetics, Mathematics and Physics,
15 No. 551 Vedado, C.P. 10400, Havana, Cuba
mlazo@get.tur.cu

Abstract. Typical testors are a useful tool for feature selection and for determining feature relevance in supervised classification problems, especially when quantitative and qualitative features are mixed. Nowadays, computing all typical testors is a highly costly procedure; all described algorithms have exponential complexity. Existing algorithms are not acceptable methods owing to several problems (particularly run time) which are dependent on matrix size. Because of this, different approaches, such as sequential algorithms, parallel processing, genetic algorithms, heuristics and others have been developed. This paper describes a novel external type algorithm that improves the run time of all other reported algorithms. We analyze the behaviour of the algorithm in some experiments, whose results are presented here.

Keywords: feature selection, pattern recognition, typical testors.

1 Introduction

Feature selection is a significant task in supervised classification and other pattern recognition areas. This task consists of identifying those features that provide relevant information for the classification process. In Logical Combinatorial Pattern Recognition [6,8], feature selection is solved by using Testor Theory [5]. Yu. I. Zhuravlev introduced the use of the testor concept in pattern recognition problems [4]. He defined a testor as a set of features that does not confuse objects descriptions belonging to different classes. This concept has since been extended and generalized in several ways [5]. This concept is especially well suited to problems which involve qualitative and quantitative features (mixed data) and even incomplete descriptions.

Computing all typical testors is very expensive procedure; all described algorithms have exponential complexity. In addition to sequential algorithms, different methods, such as parallel computing [14], genetic algorithms used for calculating a subset of typical testors [13], etc. But even through the application of these techniques, the run time of existing algorithms continues to be unacceptable owing to several problems which are dependent on matrix size.

L. Rueda, D. Mery, and J. Kittler (Eds.): CIARP 2007, LNCS 4756, pp. 506–514, 2007.

The present paper proposes a novel external type algorithm, named CT-EXT, for calculating the typical testor set of a learning matrix. The classic concept of a testor, where classes are assumed to be both hard and disjointed, is used. The comparison criteria used for all features are Boolean, regardless of the feature type (qualitative or quantitative). The similarity function used for comparing objects is also Boolean. These concepts are formalized in the following section.

2 Preliminary Concepts

Let $U = \{O_1, O_2, \cdots, O_s, \cdots\}$ be a universe of objects, $\Omega = \{O_1, O_2, \cdots, O_m\}$ is a subset of U, $I(O_1), I(O_2), \cdots, I(O_m)$, are the object descriptions in terms of the set of features $R = \{X_1, X_2, \cdots, X_n\}$, where each feature X_i, has a set of admissible values M_i (M_i is the domain of X_i) associated to it.

Each object description can be represented as $I(O_i) = (X_1(O_i), X_2(O_i), \cdots, X_n(O_i)$. Let us consider a function $C_i : M_i \times M_i \rightarrow L_i$, where L_i is a totally ordered set, such as if C_i is a dissimilarity criterion, then $C_i(X_i(O_s), X_i(O_s)) = min\{y|y \in L_i\}$, and if C_i is a similarity criterion, then $C_i(X_i(O_s), X_i(O_s)) = max\{y|y \in L_i\}$ [10] . This function can be regarded as the comparison criterion for the feature X_i.

Set U is a union of a finite number of c disjoint subsets $K_1, \cdots, K_c$, which are called classes. Each object O has an associated c-tuple of membership degrees, which describes the belonging of the object O to the classes $K_1, \cdots, K_c$. This c-tuple of membership degrees is denoted by $\alpha(O)$. Then, $\alpha(O) = (\alpha_1(O), \cdots, \alpha_c(O))$, where $\alpha_i(O) = 1$ means that $O \in K_i$ and $\alpha_i(O) = 0$ means that $O \notin K_i$ [10]. The information containing both descriptions and c-tuple of membership degrees of objects in Ω is used as learning data for classification, and we call it a Training Sample (TS).

Other denominations like Learning Sample or Learning Matrix (LM) are also used. Table 1 shows the general scheme of a learning matrix.

Table 1. General Learning Matrix scheme

Objects	Features			l-uple of belonging
O_1	$X_1(O_1)$	$\cdots$	$X_n(O_1)$	$\alpha_1(O_1), \cdots, \alpha_c(O_1)$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
O_p	$X_1(O_p)$	$\cdots$	$X_n(O_p)$	$\alpha_1(O_p), \cdots, \alpha_c(O_p)$
O_q	$X_1(O_q)$	$\cdots$	$X_n(O_q)$	$\alpha_1(O_q), \cdots, \alpha_c(O_q)$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
O_m	$X_1(O_m)$	$\cdots$	$X_n(O_m)$	$\alpha_1(O_m), \cdots, \alpha_c(O_m)$

The supervised pattern recognition problem for an object $O \in U - \Omega$ consists in determining $\alpha(O)$ using the learning matrix and the description of object O.

Given LM and n bi-valued comparison criteria $C_1, \cdots, C_n$, a comparison matrix denoted by DM is created. We consider all comparison criteria as dissimilarity ones. This comparison matrix is called a dissimilarity matrix. Rows of DM are obtained from feature by feature comparison of every pair of objects from LM belonging to different classes. As all are dissimilarity criteria, if the objects under comparison are not similar in terms of a feature, a value of 1 is assigned in the corresponding column. If this is not the case, a value 0 is assigned.

Each row of DM may be represented by the set of column indexes, in which this row has a value of 1. This set indicates the features in which the compared objects are not similar.

It is important to realize that each row of DM may contain redundant information.

Let f and g be two rows of DM and let F and G be their associated set of column indexes, respectively. We say that f is a sub-row of g if $F \subset G$. We say that f is a basic row of DM if f has no sub-rows in DM.

A sub-matrix containing all basic rows in DM (without repetitions) is called a basic matrix and we identify it with BM.

Commonly, algorithms used for computing typical testors make use of BM instead of DM or LM by two reasons: 1) unnecessary comparisons among objects are avoided, BM contains the comparisons among the "most similar" objects, and 2) typical testors calculated over BM are the same as those calculated over DM or LM [10]. This is because BM contains all information needed to calculate typical testors of LM.

Definition 1. *A feature subset $T \subseteq R$ which does not confuse any two objects belonging to different classes is called testor. A typical testor is an irreducible testor (if any feature of T (column of LM) is eliminated, then the new set is not a testor [10]).*

3 Previous Algorithms

Algorithms for computing typical testor set can be divided into external type and internal type algorithms. External type algorithms consider the whole power set of features of BM in order to determine whether a feature subset is a typical testor or not, by using a previously defined total order. These algorithms take advantage of several properties for skipping over several sets and thus avoid the analysis of every combination of features. Examples of this kind of algorithms are: BT and TB [9], REC [7], CER [2] and LEX [15].

On the other hand, internal type algorithms analyze BM based on several conditions that guarantee that a feature subset is a typical testor. Instead of verifying whether a set of features is a typical testor, as external type algorithms do, internal type algorithms build typical testors through different strategies. Some of these algorithms are: CC [1] and CT [3].

Generally speaking, external type algorithms work in the following fashion: each combination (feature subset) is represented by a characteristic n-tuple, a total order is defined in the power set of features (natural order over the

characteristic n-tuples, lexicographic order, etc.), a property that characterizes a typical testor is defined. Starting from an initial combination, the algorithm moves about the power set verifying the fulfillment of the property. If the feature combination under analysis is a typical testor, then it is saved. In all cases, the algorithm skips those consecutive feature combinations which may be said, with certainty, not to be typical testors, either because they are not testors or because they are testors but not typical. The characterizing property is verified row by row in BM, in consideration only of columns belonging to the current combination.

The blind generation of feature combinations, which does not take into account how much each feature contributes to the combination under construction, is one of the main limitations of this kind of algorithm.

The most significant difference between LEX and all other external type algorithms is that LEX incorporates into the combination under construction those features that maintain the fulfilment of the typical testor characterizing property. However, the verification of this property for each new added feature is a costly process.

Internal type algorithms build feature combinations considering values in BM. These algorithms take into account elements with a value of 1 in BM in order to generate maximal feature combinations in such a way that the involved columns represent a typical testor. There are two strategies: i) building incremental combinations knowing that they are testors and trying to successfully complete a typical testor and ii) building incremental combinations that have the typicity property and trying to successfully complete a testor. These algorithms have the disadvantage that they generate many repetitions of feature combinations.

4 The Proposed Algorithm CT-EXT

CT-EXT is an external type algorithm which uses the same lexicographic total order that the LEX algorithm uses. The CT-EXT algorithm generates incremental feature combinations reducing, step by step, the number of objects belonging to different classes that are confused, until a combination which is a testor is obtained. Subsequently, CT-EXT verifies whether the generated combination is a typical testor. As well as LEX, CT-EXT rules out those feature combinations that can generate a testor which is not a typical testor, preserving those candidates capable of generating a typical testor only. If a testor is generated, all its consecutive supersets (in the lexicographic order previously introduced into the power set of features) are not analyzed. They are skipped because these feature combinations are testors, but not typical testors.

The Algorithm's name (CT-EXT) is obtained by combining CT (because of the similarities in behaviour it presents with respect to the CT algorithm) and EXT (because it carries out its search over the power set of features). Meaning, it is an EXTernal type algorithm.

In addition to this, to improve the performance of the algorithm, a convenient order is introduced into BM, in a similar way as the case studied in [12]. The algorithm has the following theoretical bases.

Definition 2. *Let f_i be a row of BM. We say that f_i is a zero row of $S \subseteq R$, and we denote it by $f_i^0(S)$, if $\forall X_p \in S$, $f_i[p] = BM[i,p] = 0$.*

Definition 3. *In terms of BM, a testor $T \subseteq R$ is a feature set such that there are no zero rows of T in BM.*

From this definition, if T is a testor then, any superset of T is a testor too.

Definition 4. *Let f_i be a row of BM. We say that f_i is a typical row of $S \subseteq R$ with regard to X_q, and we denote it by $f_i^1(S,q)$ if $\exists X_q \in S$, such that $f_i[q] = BM[i,q] = 1$, and $\forall X_p \in S$, $X_p \neq X_q$, $f_i[p] = BM[i,p] = 0$*

Definition 5. *In terms of BM, $T \subseteq R$ is a typical testor if T is a testor and $\forall X_j \in T, \exists f_i^1(T,j)$*

This means that in a typical testor, for all features, there exists a row in BM in the sub matrix associated to T, having a value of 1 in the position corresponding to that feature, and values of 0 in all remaining positions (there are no zero rows, and if any column of T is eliminated, at least one zero row appears, and the testor property is not fulfilled). Although we have defined a typical testor here in terms of BM, normally it is defined as an irreducible testor (as in definition1).

Definition 6. *Let $T \subseteq R$ and $X_j \in R$, $X_j \neq T$. We denote by $\sum_T f^0$ the number of zero rows of T. We say that X_j contributes to T if, and only if, $\sum_{T \cup \{X_j\}} f^0 < \sum_T f^0$.*

This definition indicates that one feature contributes to a feature combination if for some zero rows in BM, considering only T, the new feature has a value of 1 in at least one of these zero rows. So, adding this feature to T, there are less zero rows of the incremented feature combination than of T.

Being $T \subset T \cup \{X_j\}$, it is not possible that $\sum_T f^0 < \sum_{T \cup \{X_j\}} f^0$. If we increase the feature combination either the zero rows are maintained and in this case the column added does not contribute to the combination, or the number of zero rows decreases.

Proposition 1. *Let $T \subseteq R$ and $X_j \in R$, $X_j \neq T$. If X_j does not contribute to T, then $T \cup \{X_j\}$ can not generate any typical testor.*

Proof. Let $T \subseteq R$ and $X_j \in R$, such that X_j does not contribute to T. Suppose that $T' = T \cup \{X_j \cup Z\}$ is a typical testor. Then, according to definition 5, there exists for X_j at least a typical row in BM. Then, $f_i[j] = BM[i,j] = 1$, and $\forall X_p \in T', X_p \neq X_j, f_i[p] = BM[i,p] = 0$. Thus, we have that f_i is a zero row of $T \cup Z$ and therefore, of T too. So, $\sum_{T \cup \{X_j\}} f^0 < \sum_T f^0$ and we obtain that X_j contributes to T, which contradicts the formulated hypothesis and then, we have that there are no typical testors that include $T \cup \{X_j\}$.

Proposition 2. *Let $T \subseteq R$, $Z \subseteq R$, $Z \neq \emptyset$. If T is a testor, then $T \cup Z$ is a testor too, but it is not a typical testor.*

Proof. Being T a testor, we have that $\sum_T f^0 = 0$, therefore, any feature $X_p \in Z$ contributes to T. Since $T \cup Z$ is a superset of T, then $T \cup Z$ is a testor, but following proposition 1, it can not generate any typical testor.

Description of the algorithm CT_EXT

Input: BM (Basic Matrix)

Output: TT (set of all typical testors)

1. **Ordering rows and columns in BM**.- The row with less quantity of values 1, is set as the first row of BM. Columns of BM are ordered, from left to right, each having a value of 1 in the first row and each subsequent column having a value 0 in the first row of BM. The order of the columns into each group (with the same value of 1 or with the same value of 0) is irrelevant.
2. **Initializing**.- $TT = \emptyset$ (typical testors set); $T = \emptyset$ (current feature combination); $j = 1$ (first feature of BM to be analyzed)
3. **Adding a new feature generator of feature combinations**.- If X_j has a value of 1 in the first row of BM then X_j is added to T ($T = T \cup \{X_j\}$), go to step 5. In another case, the algorithm finishes (any new feature combination will not generate a typical testor, because all these combinations have a zero row).
4. **Evaluating the new feature**.- The new feature is added to the current combination ($T = T \cup \{X_j\}$), and it is verified whether this new feature contributes to the current combination. If the answer is negative, go to step 6.
5. **Verifying testor property**.- Verify whether T is a testor, if yes then verify whether it is a typical testor. If T is a typical testor, the combination is saved in TT ($TT = TT \cup T$). If this is not the case, go to step 7.
6. **Eliminating the last feature analyzed**.-The last feature analyzed X_j is eliminated from T ($T = T \backslash \{Xj\}$). If X_j does not contribute to T, then no combination containing T is verified (proposition 1). Go to step 7. If the combination was a testor, then no consecutive superset of the current combination is analyzed (proposition 2). If $T = \emptyset$ then $j = j + 1$, go to step 3.
7. **Selecting a new feature to analyze**.- The next non-analyzed feature in the current combination is selected. If $j < n$ then $j = j + 1$, and go to step 4. If this is not the case, go to step 6.

5 Experiments

In order to evaluate the performance of the proposed algorithm, a comparison with three algorithms reported in the literature (BT, CT and LEX) was made. The first algorithm selected is a classical external type algorithm, which uses the

last reported algorithm which incorporates several improvements in performance [12]. The second algorithm is an internal type algorithm which has the best run times compared with other internal type algorithms [11]. The final algorithm, LEX, reported the best run time execution of all algorithms.

In order to compare run times, we use several BM with different dimensions. Some of them were randomly generated; one was taken from real medical diagnosis problems. In table 2, the experimental results obtained with the algorithms are shown.

The matrices used are denoted by $M_{rows \times columns}$, and TT denotes the number of typical testors found by algorithms. The experiments were conducted in a Pentium IV, with 2Ghz, and 1 Mbyte of RAM. The execution times are presented in seconds.

Table 2. Run time execution in seconds of several algorithms

Algorithm	$M_{10\times10}$	$M_{10\times34}$	$M_{15\times15}$	$M_{209\times32}$	$M_{20\times38}$	$M_{10\times42}$	$M_{269\times42}$	$M_{209\times47}$
BT	0	14	0	25	105	14	> 43200	> 43200
CT	0	0	0	39	0	0	38691	8026
LEX	0	0	0	14	0	0	2530	1799
CT_EXT	**0**	**0**	**0**	**3**	**0**	**0**	**928**	**483**
TT	30	935	178	6405	2436	935	302066	184920

Experiments show that the LEX algorithm has the best performance of all algorithms (BT and CT). The BT algorithm was unable to find the typical testors whole set for the two largest matrices for more than 12 hours, and the execution was aborted. In all experiments, the proposed algorithm provides faster run times than the other algorithms. As Table 2 shows, CT-EXT run time execution achieves reductions between 63% and 78% with respect to LEX algorithm.

6 Conclusions

This paper proposes a novel external type algorithm which affords the best performance of all algorithms reported in the literature. With this algorithm, thus, we provide researchers and applied scientists who might be interested in calculating typical testors for high dimensional matrices with a useful tool.

Like LEX, the proposed algorithm does not generate a great quantity of comparisons, because it does not verify many feature combinations which, as one may determine a priori, do not generate typical testors. The main difference with respect to LEX is that LEX verifies, as a first condition, whether the combination under analysis has typical rows. In contrast, CT-EXT verifies, first of all, if the combination generated has less zero rows than previous combinations. From this point of view, CT-EXT resembles the REC algorithm, although REC directly works with the learning matrix [7].

In addition to this, an analogy between algorithms LEX and CC can be noted, because both algorithms generate combinations focusing on typical rows. On the other hand, an analogy between algorithms CT-EXT and CT is that these algorithms generate combinations guarantying that they are testors, and, after that, both algorithms verify whether the testor is typical or not. The order introduced into rows and columns of BM, allows us to avoid the analysis of a great quantity of feature combinations.

The CT-EXT and LEX algorithms represent a new kind of algorithm for computing all typical testors. Their chief characteristic is that both algorithms analyze a typical testor candidate before inserting new feature in current combination. They are less blind than their predecessors. Based on experimental results, we can conclude that the proposed algorithm has the highest performance score.

Acknowledgements. Many thanks to Dr. Francisco Martinez Trinidad from INAOE, Mexico, for providing a LEX version for fulfill comparisons between algorithms.

References

1. Aguila Feroz, L., Ruiz Shulcloper, J.: Algorithm CC for the elaboration of k-valued information on pattern recognition problems. Cuba 5(3), 89–101 (1984) (In Spanish)
2. Ayaquica Martinez, I.Y., Jimenez Jacinto, V.: A new external type algorithm for the calculation of tipical testors. In: Proc. TIARP 1997, pp. 141–148 (1997) (In Spanish)
3. Bravo, A.: Algorithm CT for calculating the typical testors of k-valued matrix. Revista Ciencias Matematicas, Cuba 4(2), 123–144 (1983)
4. Dmitriev, A.N., Zhuravlev, Y.I., Krendeliev, F.: About mathematical principles and phenomena classification. Diskretni Analiz 7, 3–15 (1966)
5. Lazo-Cortes, M., Ruiz-Shulcloper, J., Alba Cabrera, E.: An Overview of the evolution of the concept of testor. Pattern Recognition 34(4), 753–762 (2001)
6. Martinez-Trinidad, J.F., Guzman Arenas, A.: The Logical Combinatorial approach for pattern recognition an overview through selected Works. Pattern Recognition 34(4), 741–751 (2001)
7. Morales, R.: A classification system and pattern recognition. B. Sh. Thesis, Mexico, UNAM (1988) (In Spanish)
8. Ruiz-Shulcloper, J., Abidi, M.: Logical Combinatorial Pattern Recognition: A Review. Recent Research Developments in Pattern Recognition. In: Pandalai, S.G. (ed.) Recent Research Developments in Pattern Recognition, Transworld Research Networks, Kerala, India, pp. 133–176 (2002)
9. Shulcloper, J.R., Bravo, M.A.Y., Aguila, F.L.: Algorithms BT and TB for calculating all typical tests. Revista Ciencias Matematicas, Cuba 6(2), 11–18 (1982) (In Spanish)
10. Ruiz Shulcloper, J., Guzman Arenas, A., y Martinez Trinidad J.: Logical combinatorial pattern recognition approach". Advances on pattern recognition series, Edit. Instituto Politecnico Nacional, Mexico (1999) (In Spanish)

11. Sanchez Diaz, G,: Developing and implementing efficient algorithms (batch and parallel) for calculating typical testors of a basic matrix. Master Thesis, BUAP, Puebla, Mexico (1997) (In Spanish)
12. Sanchez Diaz, G., y Lazo Cortes, M.: Modifications to BT algorithm for improving its run time execution. Revista Ciencias Matematicas, Cuba 20(2), 129–136 (2002) (In Spanish)
13. Sanchez Diaz, G., Lazo Cortes, M., y Fuentes Chavez, O.: Genetic algorithm for calculating typical testors of minimal cost. In: Proc. SIARP 1999, pp. 207–213 (1999) (In Spanish)
14. Sanchez Diaz, G., Lazo Cortes, M., Garcia Fernandez, J.: Parallel and distributed models for calculating typical testors. In: Proc. TIARP 1997, pp. 135–140 (1997) (In Spanish)
15. Santiesteban Alganza, Y., Pons Porrata, A.: LEX: A new algorithm for calculating typical testors. Revista Ciencias Matematicas, Cuba 21(1), 85–95 (2003) (In Spanish)