Expert Systems with Applications 39 (2012) 2203–2210

Contents lists available at SciVerse ScienceDirect

Expert Systems with Applications

journal homepage: www.elsevier.com/locate/eswa

Hardware–software platform for computing irreducible testors

Alejandro Rojas, René Cumplido *, J. Ariel Carrasco-Ochoa, Claudia Feregrino, J. Francisco Martínez-Trinidad

Computer Science Department, National Institute for Astrophysics, Optics and Electronics, Sta. Ma. Tonanzintla, Puebla 72840, Mexico

ARTICLE INFO

Keywords:
Feature selection
Testor theory
Custom architectures
FPGAs

ABSTRACT

In pattern recognition, feature selection is a very important task for supervised classification. The problem consists in, given a dataset where each object is described by a set of features, finding a subset of the original features such that a classifier that runs on data containing only these features would reach high classification accuracy. A useful way to find this subset of the original features is through testor theory. A testor is defined as a subset of the original features that allows differentiating objects from different classes. Testors are very useful particularly when object descriptions contain both numeric and non-numeric features. Computing testors for feature selection is a very complex problem due to exponential complexity, with respect to the number of features, of algorithms based on testor theory. Hardware implementation of testor computing algorithms helps to improve their performance taking advantage of parallel processing for verifying if a feature subset is a testor in a single clock cycle. This paper introduces an efficient hardware–software platform for computing irreducible testors for feature selection in pattern recognition. Results of implementing the proposed platform using a FPGA-based prototyping board are presented and discussed.

1. Introduction

Reconfigurable computing based on the combination of conventional microprocessors and field programmable gate arrays (FPGAs), has become increasingly popular for implementing special-purpose hardware to accelerate complex tasks. Usually an FPGA-based implementation is embedded in a PC or workstation, which drives its activity and manages the results. Following this trend, we developed an efficient hardware–software platform for computing irreducible testors (Lazo-Cortés, Ruiz-shulcloper, & Alba-cabrera, 2001) for feature selection in pattern recognition (Al-Ani, 2009; Chen, Tseng, & Hong, 2008; Jain & Zongker, 1997; Kwan & Choi, 2002; Liu & Setiono, 1998).

The feature selection problem in pattern recognition consists in, given a dataset where each object is described by a set of features, finding a subset of the original features such that a classifier that runs on data containing only these features would reach higher classification accuracy. This procedure can reduce not only the cost of recognition by reducing the number of features to be collected, but in some cases it can also provide better classification accuracy. For this task, a higher performance with lower computational effort is expected (Kwan & Choi, 2002). Several algorithms have been proposed for feature selection, however, most of them were developed for numeric features (Guyon & Elisseeff, 2003; Jain & Zongker, 1997). We chose BT, an algorithm based on testor theory, which can be applied on datasets described with both numeric and non-numeric features, even when there are missing data. Although the theoretical aspect of computing irreducible testors is advanced (Asaithambi & Valev, 2004; Djukova, 2005; Kudryavtsev, 2006; Martínez-Trinidad & Guzmán-Arenas, 2001; Valev & Sankur, 2004), there are not practical hardware implementations reported previously, excepting our previous works.

In our first work, an architectural design based on a brute force approach for computing testors was proposed (Cumplido, Carrasco, & Feregrino, 2006). In this first approach, each candidate was generated by a counter that incremented its value by 1 on each iteration. The architecture is able to evaluate if a candidate is a testor in a single clock cycle, however, the architecture did not exploit the characteristics of a particular data set that could allow to significantly reduce the number of candidates tested. The next step in our architectural design (Rojas, Cumplido, Carrasco-Ochoa, Feregrino, & Martnez-Trinidad, 2007) was the implementation of BT algorithm for computing testors where a candidate generator that jumps over unnecessary candidates allows reducing the number of comparisons needed in the brute force approach. These two previous works compute the whole set of testors, however for pattern recognition applications where testor theory can be applied, it is important to obtain only testos that are irreducible. Thus, as the next step in our design, this work proposes a hardware–software platform for computing only irreducible testors. This platform consists of the combination of a specialized hardware architecture that is implemented on a commercial FPGA-based prototyping board and a host application running on a PC. The architecture

* Corresponding author.
E-mail address: rcumplido@inaoep.mx (R. Cumplido).

implements the BT algorithm, as in Rojas et al. (2007), but it also includes a new module that eliminates most of the testors that are not irreducible before transferring them to the host application for final processing.

The intensive computational requirements due to the exponential complexity, with respect to the number of features, of the testor theory based algorithms can be met by a combination of technological improvements and efficient hardware architectures based on parallel computational models. Specific parallel architectures can be designed to exploit the parallelism found in the irreducible testor computing algorithms. Further optimizations such as incremental processing and the use of multiple processing elements are also possible.

2. Computing irreducible testors

In pattern recognition, feature selection is a very important task for supervised classification. A useful way to do this selection is through testor theory. The concept of testor for pattern recognition was introduced by Dmitriev, Zhuravlev, and Krendeliev (1966). They defined a testor as a subset of features that allows differentiating objects from different classes. Testors are quite useful, especially when object descriptions contain both numeric and nonnumeric features, and maybe they are incomplete (mixed incomplete data) (Martínez-Trinidad & Guzmán-Arenas, 2001).

Let *TM* be a training matrix with *K* objects described through *N* features of any type $(x_1,\ldots,x_N)$ and grouped in *r* classes. Let *DM* be a dissimilarity Boolean matrix (0 = similar, 1 = dissimilar), obtained from feature by feature comparisons of every pair of objects from *TM* belonging to different classes. *DM* has *N* columns and *M* rows, where $M \gg K$.

Testors and irreducible testors are defined as follows:

Definition 1. A subset of features *T* is a testor if and only if when all features are eliminated from *DM*, except those from *T*, there is not any row of *DM* with only 0s.

Definition 2. A subset of features *T* is an irreducible testor if and only if *T* is a testor and there is not any other testor *T′* such that $T' \subset T$.

In Definition 1, if there is not any row of *DM* with only 0's it means that there is not a pair of objects from different classes that are similar on all the features of *T*, that is, a testor *T* allows differentiating between objects from different classes.

The number of rows in *DM* could be too large, therefore a strategy to reduce this matrix without losing relevant information for computing irreducible testors was introduced by Lazo-Cortés et al. (2001).

Definition 3. If *t* and *p* are two rows of *DM*, then *p* is a sub-row of *t* if and only if:

(a) *t* has 1 everywhere *p* has 1
(b) there is at least one column such that *t* has 1 and *p* has 0.

Definition 4. A row *t* of *DM* is a basic row of *DM* if and only if *DM* does not have any other row *t′* such that *t′* is a sub-row of *t*.

Definition 5. The matrix that contains only the basic rows of *DM* is called basic matrix and is denoted by *BM*.

Let *TT*(*M*) be the set of all irreducible testors of the Boolean matrix *M*, then

Table 1
N-tuples omitted by step 3.

α	Feature set	
011001000	$\{x_2,x_3,x_6\}$	α
011001001	$\{x_2,x_3,x_6,x_9\}$	
011001010	$\{x_2,x_3,x_6,x_8\}$	
...	...	
011001111	$\{x_2,x_3,x_6,x_7,x_8,x_9\}$	7 non-irreducible testors
011010000	$\{x_2,x_3,x_5\}$	$\alpha' = \alpha + 2^{N-k}$

Proposition 1. *TT*(*DM*) = *TT*(*BM*).

This proposition indicates that the set of all irreducible testors calculated using *DM* or *BM* is the same (Lazo-Cortés et al., 2001). However, *BM* is smaller than *DM* and the construction of *BM* from *DM* is a very fast process, for example, the time for obtaining a *BM* matrix with 48 columns and 32 rows from a *DM* matrix with 48 columns and 193,753 rows, is about 0.21 s on a PC with an Intel Centrino Duo processor running at 1.6 GHz, with 1024 MB of RAM.

There are two kinds of algorithms for computing irreducible testors: the *internal scale algorithms* and the *external scale algorithms*. The former analyzes the matrix to find out some conditions that guarantee that a subset of features is an irreducible testor. The latter looks for irreducible testors over the whole power set of features; algorithms that search from the empty set to the whole feature set are called *Bottom–Top* algorithms and algorithms that search from the whole feature set to the empty set are called *Top–Bottom* algorithms. The selected algorithm is a *Bottom–Top external scale algorithm*, called BT (Sánchez-Díaz & Lazo-Cortés, 2002). This algorithm was selected because of its simplicity and inherent parallelism which can be easily exploited in a hardware architecture.

In order to review all the search space, BT codifies the feature subsets as binary *N*-tuples where 0 indicates that the associated feature is not included and 1 indicates that the associated feature is included (some examples can be seen in Table 1). For computing testors, BT follows the order induced by the binary natural numbers, this is, from the empty set to the whole feature set. The BT algorithm is as follows:

1. Generate first no null *N*-tuple
$$\alpha = (\alpha_1, \ldots, \alpha_N) = (0, \ldots, 0, 1).$$
2. Determine if the generated N-tuple α is a testor of *BM*.
3. If α is a testor of *BM*, store it and take
$$\alpha' = [(\alpha)_b + 2^{N-k}]_{tuple},$$

where *k* is the index of the last 1 in α and $(\alpha)_b$ represents the natural number corresponding to α and $[(\alpha)_b + 2^{N-k}]_{tuple}$ represents the *tuple* associated with the natural number $(\alpha)_b + 2^{N-k}$.

4. If αis not a testor of *BM*, determine the first row *v* of *BM* with only 0's in the columns where α has 1's and generate α′ as:
$$\alpha'_j = \begin{cases} \alpha_j & j < k, \\ 1 & j = k, \\ 0 & j > k, \end{cases}$$

where *k* is the index of the last 1 in *v*.

5. Take α = α′.
6. If αis not after (1,1,. . .,1,1) then, go to 3.
7. Eliminate from the stored testors those which are not irreducible testors.

Step 3 jumps over all the supersets that can be constructed from α by adding 1's (features) after the last 1 in α. For example if *N* = 9 and α = (0,1,1,0,0,1,0,0,0) then *k* = 6 and the following $2^{N-k} - 1 = 2^{9-6} - 1 = 7$ *N*-tuples represent supersets of the feature