Pattern Recognition Letters 95 (2017) 22–28

Contents lists available at ScienceDirect

Pattern Recognition Letters

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

A CUDA-based hill-climbing algorithm to find irreducible testors from a training matrix

Ivan Piza-Davila[a], Guillermo Sanchez-Diaz[b,*], Manuel S. Lazo-Cortes[c], Luis Rizo-Dominguez[a]

[a] Instituto Tecnologico y de Estudios Superiores de Occidente, Periferico Sur Manuel Gomez Morin 8585, Tlaquepaque, Jalisco, 45604, Mexico
[b] Universidad Autonoma del Estado de San Luis Potosi, Facultad de Ingenieria, Dr. Manuel Nava 8, San Luis Potosi, SLP, 78290, Mexico
[c] Instituto Nacional de Astrofisica, Optica y Electronica, Tonantzintla, Puebla, 72840, Mexico

ARTICLE INFO

Article history:
Received 9 November 2016
Available online 26 May 2017

Keywords:
Pattern recognition
Feature selection
Irreducible testors
CUDA
Hill climbing

ABSTRACT

Irreducible testors have been used to solve feature selection problems. All the exhaustive algorithms reported for the generation of irreducible testors have exponential complexity. However, several problems only require a portion of irreducible testors (only a subset of all). The hill-climbing algorithm is the latest approach that finds a subset of irreducible testors. So this paper introduces a parallel version of the hill-climbing algorithm which takes advantage of all the cores available in the graphics card because it has been developed on a CUDA platform. The proposed algorithm incorporates a novel mechanism that improves the exploration capability without adding any extra computation at the mutation step, thus increasing the rate of irreducible testors found. In addition, a Bloom filter is incorporated for efficient handling of duplicate irreducible testors. Several experiments with synthetic and real data, and a comparison with other state-of-the-art algorithms are presented in this work.

1. Introduction

Feature selection is a significant task in supervised classification and other pattern recognition areas. It identifies those features that provide relevant information for the classification process.

Feature selection is addressed in the Logical Combinatorial Pattern Recognition approach [16] using Testor Theory [8]. The concept of testor, applied to pattern recognition problems, was introduced by Zhuravlev [6]. This concept has been extended in several ways [8,23]. Ruiz-Shulcloper [17,18] introduced the characterization of irreducible testors -known as typical testors- and several exhaustive algorithms for computing the whole set of irreducible testors from a training matrix; these algorithms are very useful particularly when object descriptions are defined in terms of numeric and non-numeric features. The concepts of testor and irreducible testor have also been used by V. Valev under the terms descriptor and non-reducible descriptor, respectively [24].

The computation of the whole set of irreducible testors using exhaustive algorithms requires exponential time [22]. However, there are real-world problems which do not require the whole set of irreducible testors, but only a subset. An example is the determination of factors associated with Transfusion Related Acute Lung Injury [21]. More of these real problems are explained in detail in [19].

1.1. Related work

There are some studies related to the development of hardware-software platforms based on the Field Programmable Gate-Array (FPGA) that generate irreducible testors from a training matrix [4,14,15]. Each of these platforms consists of the implementation on a FPGA of a known algorithm that finds irreducible testors, taking advantage of the high degree of parallelism.

The CT-Ext algorithm described in [14] tends to be the fastest. However, a fair comparison on CPU shows that this algorithm has been surpassed by others [19,20]. Also, for a new matrix, the outcome will be delayed until a new FPGA synthesis process is completed.

On the other hand, various hill-climbing algorithms have been implemented using CUDA; these algorithms have been applied primarily to address combinatorial optimization problems, in particular, the Traveling Salesman Problem [9,13]. Although these proposed approaches try to find the near global optimum solution using an iterative random restart hill-climbing algorithm, this algorithm focuses on finding many local optimum solutions, e.g., the greatest possible subset of irreducible testors.

* Corresponding author.
E-mail address: guillermo.sanchez@uaslp.mx (G. Sanchez-Diaz).

http://dx.doi.org/10.1016/j.patrec.2017.05.026
0167-8655/

This work is based on a hill-climbing algorithm (denoted HC), that was successfully adapted to find a subset of irreducible testors [19], as well as on its parallel version [11] (denoted PHC), which works on multicore processors. The goal of both algorithms is to iteratively find irreducible testors across the search space, without building all the possible combinations of features. However, when processing training matrices made up of hundreds of rows and features, HC may take hours to finish; PHC reduces the execution time proportionally to the number of threads employed, which should correspond to the number of cores available. Unfortunately, the number of cores available on current multiprocessors is still very limited: no higher than eight. In many of these processors, half of the cores are not actually physical, which results in a loss of performance. Workstations provide a higher number of cores per processor, typically sixteen cores and two processors, but their prices are still high.

1.2. General-purpose computing on graphics processing units

General-purpose GPU computing [10], or GPGPU, is the use of a GPU to do general-purpose scientific and engineering computing. One recent application for GPGPU is the evaluation of solutions for combinatorial optimization problems using metaheuristics [3,25]. Given a set of candidate solutions, the GPU can be used to evaluate each one in parallel.

Following this approach, we present a CUDA-based version of HC, hereafter denoted CHC, that takes advantage of all the cores available on the graphics card.

1.3. Contributions

The contributions of CHC with respect to PHC are the following:

1. The adaptation of the hill-climbing algorithm for running on a Graphics Processing Unit with support for hundreds of cores, taking into consideration some limitations of GPU programming, namely, limited resources and time spent in data transfer between host and device memories.
2. A more efficient mechanism for picking random features to mutate, which increases the exploration capability and reduces the number of calculations on the GPU.
3. A more efficient mechanism to handle repetitive irreducible testors. In many data sets, this single process might take longer than the hill-climbing algorithm itself.

2. Problem statement

For a better understanding of this paper, this section provides some definitions and notations related to irreducible testors. They were introduced in [18] and [2].

Let U be a set of objects described in terms of n features $R = \{x_1, x_2, \cdots, x_n\}$. Let $\{O_1, O_2, \cdots, O_m\} \subset U$, a set containing m objects, each one belonging to a class $K_i \in \{K_1, K_2, \cdots, K_c\}$. We call *training matrix* (denoted *TM*) to the matrix representation of this set, where the ith row is a description of O_i in terms of the features in R.

Let *DM* be a *dissimilarity matrix* where the rows are obtained from a feature-by-feature comparison between every pair of objects belonging to different classes, setting 0 if the values of the compared features are similar, and setting 1, otherwise.

Let $T \subseteq R$ be a subset of features. DM^T denotes the sub-matrix obtained from *DM* considering only the features belonging to T.

Definition 1. Let p be a row of DM^T; we say that p is *a zero row* if it contains only zeros.

Definition 2. T is a testor of *TM*, if there is no zero row in DM^T

Definition 3. Let $T \subseteq R$ and $x_i \in T$. x_i is called a *non-removable feature* of T if the number of zero rows in $DM^{T-\{x_i\}}$ is greater than in DM^T. Otherwise x_i is called a *removable feature* of T.

Definition 4. A testor T is denominated an *irreducible testor* of *TM* if every feature $x_i \in T$ is a non-removable feature of T. It means that T is an irreducible testor if no subset of T is a testor.

Definition 5. Let p and q be two rows of *DM*. We say that p is a sub-row of q if: $\forall j\ [q_j = 0 \Rightarrow p_j = 0]$ and $\exists_i\ [p_i = 0\ AND\ q_i = 1]$; $i, j \in \{1, \cdots, n\}$.

Definition 6. A row p of *DM* is called basic if no row in *DM* is a sub-row of p. The sub-matrix of *DM* containing all its basic rows (without repetitions), is called a basic matrix (denoted *BM*).

Irreducible testors may be computed using either *DM* or *BM* as stated in the following proposition.

Proposition 1. *Let DM be a dissimilarity matrix and BM its corresponding basic matrix, and let* $\psi^*(DM)$ *and* $\psi^*(BM)$ *be the set of irreducible testors of DM and BM respectively; then* $\psi^*(DM) = \psi^*(BM)$.

Proof. Let $\mathcal{R}_{DM}$ and $\mathcal{R}_{BM}$ be the set of rows in *DM* and *BM* respectively. Let $\psi(DM)$ and $\psi(BM)$ be the set of all testors of *DM* and *BM* respectively. Let $T \in \psi(DM)$; then DM^T has no zero rows, and since $\mathcal{R}_{BM} \subseteq \mathcal{R}_{DM}$ then BM^T has no zero rows either, and $T \in \psi(BM)$. Now let $T \notin \psi(DM)$, meaning that DM^T has at least one zero row $(a_i)^T$. If a_i is a basic row, then $a_i \in \mathcal{R}_{BM}$ and $T \notin \psi(BM)$. Alternatively, if a_i is not a basic row, then it has at least one sub-row among the rows in $\mathcal{R}_{BM}$, and by definition all sub-rows of a_i (even those in $\mathcal{R}_{BM}$) have zeros in all columns in which a_i has a zero. Therefore BM^T has at least one zero row and $T \notin \psi(BM)$. Consequently $\psi(DM) = \psi(BM)$, and from Definition 4 it follows that $\psi^*(DM) = \psi^*(BM)$. □

Since *BM* has fewer rows but the same irreducible testors as *DM*, this result explains why most algorithms for computing irreducible testors are run on basic matrices instead of dissimilarity matrices.

In this work, we use concepts and definitions that have been developed in the Logical Combinatorial Pattern Recognition approach [16]. However, there are other equivalent definitions and concepts of testor and irreducible testor [6,24].

3. CUDA-based hill-climbing algorithm (CHC)

3.1. Hill-climbing algorithm

The hill-climbing algorithm is a local-search stochastic method, where a candidate solution is mutated across the search space [7]. The mutation is performed over a combination of features, called individual, according to certain characteristics of the combination which enables it to climb up the hill until reaching a local optimum: an irreducible testor [5].

In [11], a parallel version of the hill-climbing algorithm is introduced; this version is useful for finding irreducible testors from a training matrix. Each CUDA thread runs an instance of this algorithm, during which, an individual is mutated over G generations accordingly to its current fitness value [See Fig. 2]. The seed used for each thread to randomly create the first individual, is calculated as $S + T$, where S is the seed passed by the kernel function and T is the identifier of the current thread.

The mutation methods randomly select a feature to add or remove. Instead of calculating random values all the time, as in [11], a circular list is built a priori, containing a random permutation of indices in the range $[1, \ldots, F]$, one per CUDA thread. Thus, the mutation method only selects the next element from this permutation. In addition to considerably reducing the number of com-