Neurocomputing 123 (2014) 40–48

Contents lists available at ScienceDirect

Neurocomputing

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

Towards an efficient genetic algorithm optimizer for sequential projection pursuit

Soledad Espezua, Edwin Villanueva, Carlos D. Maciel *

Department of Electrical Engineering, Sao Carlos School of Engineering, University of Sao Paulo, Brazil

ARTICLE INFO

Article history:
Received 19 April 2012
Received in revised form
14 August 2012
Accepted 5 September 2012
Available online 8 April 2013

Keywords:
Projection pursuit
Sequential projection pursuit
Genetic algorithms
Crossover operators

ABSTRACT

Sequential projection pursuit (SPP) is a useful tool for revealing interesting structures hidden in high-dimensional data. SPP constructs sequentially the bases of a low-dimensional space where the projected data evidence such structures. Genetic algorithms (GAs) are promising finders of these bases, but their performance is determined by the choice of the crossover operator. Until now it is not clear which operator is more suitable for SPP. In this paper we compare the performance of eight crossover operators: three available in literature (arithmetic, single-point and multi-point) and five newly proposed here (two hyperconic, two fitness-biased and one extension of arithmetic crossover). The results on five benchmark datasets showed that the proposed hyperconic operators have the best performance in finding high-fitness projections. The performance of a canonical GA with one of these hyperconic operators was compared against two representative SPP optimizers, the PSO and the RSSA algorithms. We found that our GA with the hyperconic operator tends to find better solutions than the other methods at different numbers of fitness computations. These results suggest that the optimization of SPP can be improved with GAs by taking advantage of the exploratory capabilities of the proposed hyperconic operators.

1. Introduction

The collection of data sets with large amounts of measured features is becoming increasingly common in many industrial and scientific areas. This makes dimension reduction an active research topic in data mining, machine learning and statistics. Projection pursuit (PP) [1,2] is a framework of methods proposed to deal with such high-dimensional data sets that has become very popular in the statistical literature. PP faces the curse of dimensionality by searching for "interesting" low-dimensional projections of the data, where the interestingness of the projections is assessed by a pre-defined function, known as *projection pursuit index* (PP index). Thus, the most interesting projection spaces are found by optimizing this function.

A wide range of data-mining problems can be tackled with PP [3], depending on the PP index used. For example, PP can take the form of the popular PCA method (principal component analysis) when the variance of the projected data is used as the PP index [3]. Several PP indices have been proposed for different applications (for reviews see [4–7]). Indeed, there are PP indices suitable for cluster analysis [8,9], classification [4,10], feature selection [11] and regression analysis [12].

One of the major difficulties when working with PP is the optimization of the index function [7]. Most of the seminal work in this area were carried out in the context of exploratory projection pursuit (EPP), where the target space was limited to, at most, three dimensions [2,13–16,9]. Gradient-based methods were initially preferred [17,18], however, such methods were very susceptible to being trapped in local optima, thereby capturing projections of low relevance [2]. Further developments were carried out to alleviate such problems (e.g. [2,14,8]), but the difficulty to scale beyond 3D projections remained (mainly due to computational constraints in computing PP indices in such spaces). It was only after the emergence of the sequential projection pursuit (SPP) method [19] that the dimensionality constraint was effectively circumvented. In SPP, the bases of the projection space (called PP factors) are sought sequentially instead of all simultaneously. Each PP factor is obtained by optimizing a one-dimensional PP index over residual data (data resulting from removing the structure found in the previous PP factors). In this way, a m-dimensional optimization problem is converted into m problems of one dimension. SPP opened up new possibilities for PP, such as feature selection [11] and feature extraction for machine learning [6].

The original optimizer proposed for SPP was based in a canonical genetic algorithm (GA), where the candidate PP factors are represented as binary strings and typical crossover and mutation operators are used to evolve. An alternative optimizer was proposed by Webb-Robertson et al. [20], the random scan sampling algorithm (RSSA), arguing that the GA optimizer for SPP was slow in getting the PP factors. Other global methods such as simulated annealing (SA) [21] and particle swarm optimization (PSO) [22,23] have been proposed for optimizing SPP. However, based on the success that GA methods have shown in many difficult real-world problems [24], we believe that their potential was under explored as optimizers for SPP.

* Corresponding author. Tel.: +55 16 33739350.
E-mail addresses: carlos.maciel@usp.br, maciel@sc.usp.br (C.D. Maciel).

0925-2312/$ - see front matter © 2013 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.neucom.2012.09.045

Genetic algorithms have some desirable features attractive for SPP, such as [25]: (1) GAs can search the solution space (that is known to be multi-modal) in a parallel and multi-directional way, giving more chance to find highly informative projections; (2) at any time we can take a solution, which gets better with time; (3) one can control the diversity of the population, which can be useful for EPP since many alternative solutions are required for inspection; and (4) GAs can be straightforwardly implemented in parallel and distributed platforms, expanding the applicability of PP to problems with huge data sets. Despite these desirable features, the performance of a GA is strongly influenced by the choice of the crossover operator, since this constitutes the primary search mechanism of the GA, responsible for the rapid exchange of useful information among solutions to locate better solutions [25]. The crossover operator used in the original SPP was a generic operator, which seems to give reasonable results in mid-range problems [19,11]. However, it is unknown whether other operators could perform better in the optimization of PP factors for SPP.

Moving in this direction, we present in this paper a comparative experimental study of eight crossover operators: three currently used in SPP and PPEA (arithmetic crossover and single and multi-point crossover) and five new operators (one single extension of the arithmetic crossover, two hyperconic crossovers and two fitness-biased crossovers). First, we assess the performance of the crossover operators in a prototypical GA by measuring the mean fitness of the population at different stages of evolution and the converged fitness and number of generations needed to achieve convergence. In addition, the influence of the evolutionary pressure in the performance of the different operators is analyzed. Second, we compare the performance of the best GA (the GA with the best crossover operator) against two alternative state-of-the-art optimizers for SPP: the RSSA [20] and the PSO used in the SPP context by [7]. This evaluation determines the efficiency of the optimizers under similar amounts of computation units (number of PP index calls). The study was carried out on five benchmark datasets of increasing dimensionality ranging from 13 to 743 variables. The results suggest that we can improve the performance of the GA optimizers for SPP with the proposed operators, even at levels above those of the other methods discussed.

The paper is organized as follows. Section 2 introduces some important concepts of PP, SPP, and GAs. Section 3 describes the crossover operators studied. Section 4 presents the experimental setup. The results and discussions are presented in Section 5. Finally, our conclusions are presented in Section 6.

2. Background

2.1. Projection pursuit

The projection pursuit concept was formally introduced in the paper of Friedman and Tukey [1], although the seminal ideas were originally posed by Kruskal [26]. To describe the concept of PP assume that the data set is arranged in a $n \times p$ matrix $\mathbf{X}$ with n instances and p attributes or variables. PP seeks a m-dimensional projection space ($m < p$), defined by the orthonormal bases $\mathbf{A} \in \mathbb{R}^{p \times m}$, where the projected data $\mathbf{X} \cdot \mathbf{A}$ exposes information of interest. The degree of interestingness of the projection is measured by the function $\mathfrak{I}$, called the *projection pursuit index* (PP index). Thus, PP can be formulated as the optimization problem

$$\mathbf{A}^* = \arg\max_{\mathbf{A}} \{\mathfrak{I}(\mathbf{X} \cdot \mathbf{A})\} \quad \text{s.t.} \quad \mathbf{A}^T \cdot \mathbf{A} = \mathbf{I}. \tag{1}$$

The choice of the PP index is a key consideration. A great deal of research has been centered on the construction of a globally useful and robust index, but the effectiveness of an index is often dependent on the application and characteristics of the given dataset [6]. One dominant consideration in developing PP indices has been the so-called *affine invariance* [5]. A PP index $\mathfrak{I}$ is said to be affine invariant if $\mathfrak{I}(\mathbf{X}) = \mathfrak{I}(s\mathbf{X} + \mathbf{v})$ for a nonsingular linear transformation s and a constant vector $\mathbf{v}$. Thus, affine invariance ensures that changes in scale and location of the projected data do not affect the index value.

A common approach to insure affine invariance is *sphering* the original data matrix $\mathbf{X}$ to have zero mean and identity covariance matrix. This can be done by the following transformation [14]:

$$\mathbf{Z} = \Lambda^{-1/2}\mathbf{Q}(\mathbf{X} - E[\mathbf{X}]) \tag{2}$$

where $\mathbf{Q}$ and Λ are, respectively, the eigenvector and eigenvalue matrices resulting from the eigen-decomposition of the covariance matrix $\Sigma = \mathbf{Q}\Lambda\mathbf{Q}^T$. For simplicity, in the rest of the paper we will refer to the original data matrix $\mathbf{X}$ as the sphered version of it.

In clustering applications, *entropy* is commonly used as the PP index of the projected data [18].

However, the entropy calculation is computationally intensive, requiring high-order integrals and a density estimator. A simpler and robust alternative to entropy is the *Holes* PP index [27], which returns comparable (and often better) results than entropy [20]. The Holes index is defined for a one-dimensional projection as

$$\mathfrak{I}_{Holes} = 1 - \frac{1}{n}\sum_{i=1}^{n} e^{-(1/2)y_i^2} \tag{3}$$

where y_i is the projection of the ith data instance $\mathbf{x}_i$ onto the direction of the basis vector $\mathbf{a}$, $y_i = \mathbf{x}_i \cdot \mathbf{a}$. We adopt Holes as the PP index to be optimized in all experiments presented in this paper.

Sequential projection pursuit (SPP) [19] tackles the m-dimensional constrained optimization problem in Eq. (1) by converting it into a sequence of m one-dimensional optimization problems. The first basis (PP factor) $\mathbf{a}_1$ in $\mathbf{A}$ is obtained by searching (with a GA) a p-dimensional vector of unit length that maximizes the PP index. Once the first PP factor $\mathbf{a}_1$ is found, the data set is projected onto it, obtaining the score vector $\mathbf{y}_1 = \mathbf{X} \cdot \mathbf{a}_1$. The residual data is then computed as $\mathbf{X} = \mathbf{X} - \mathbf{y}_1 \cdot \mathbf{a}_1^T$. The process is then repeated on this residual data to obtain $\mathbf{a}_2$, $\mathbf{y}_2$ a new residual data, subject to the constraint that $\mathbf{a}_2$ is orthogonal to $\mathbf{a}_1$. In this way, the predefined m PP factors are obtained in SPP.

2.2. Genetic algorithms

Genetic algorithms (GAs) [28] belong to a class of algorithms inspired in Darwinian evolutionary theory. A typical GA begins with a random population of solutions (individuals), which are evaluated by a fitness function (that encodes the problem objective function). The algorithm is then subject to an evolutionary loop, where each iteration (generation) produces a new population as follows: a subset of individuals is stochastically chosen from the current population (based on their fitness) and recombined by a crossover operator (and possibly slightly altered by a mutation operator) to produce an offspring population; the new population is created by selecting individuals from the original and the offspring populations by means of a replacement operation.

Here we use a steady-state continuous GA to evaluate the crossover operators described in the next section. This GA is thus named PPGA, or projection pursuit genetic algorithm. The individuals in PPGA are encoded as real-value unit-length vectors. This representation is used instead of the binary representation of SPP to avoid loss of precision. Thus, an individual i is a candidate basis vector $\mathbf{a}_i = [a_{i1}, a_{i2}, \ldots, a_{ip}]^T$ to project the data $\mathbf{X} \cdot \mathbf{a}_i$. The fitness function is the Holes PP index (Eq. (3)). The selection operator is implemented as a tournament selection: we randomly choose ts individuals (the tournament size) from the current population (of size w) and return the best individual. With this selection method, $w/2$ pairs of different