Soft Comput (2017) 21:2069–2089
DOI 10.1007/s00500-015-1907-y

METHODOLOGIES AND APPLICATION

Multi-objective genetic programming for feature extraction and data visualization

Alberto Cano[1,2] · **Sebastián Ventura**[1,3] · **Krzysztof J. Cios**[2,4]

Published online: 26 October 2015

Abstract Feature extraction transforms high-dimensional data into a new subspace of lower dimensionality while keeping the classification accuracy. Traditional algorithms do not consider the multi-objective nature of this task. Data transformations should improve the classification performance on the new subspace, as well as to facilitate data visualization, which has attracted increasing attention in recent years. Moreover, new challenges arising in data mining, such as the need to deal with imbalanced data sets call for new algorithms capable of handling this type of data. This paper presents a Pareto-based multi-objective genetic programming algorithm for feature extraction and data visualization. The algorithm is designed to obtain data transformations that optimize the classification and visualization performance both on balanced and imbalanced data. Six classification and visualization measures are identified as objectives to be optimized by the multi-objective algorithm. The algorithm is evaluated and compared to 11 well-known feature extraction methods, and to the performance on the original high-dimensional data. Experimental results on 22 balanced and 20 imbalanced data sets show that it performs very well on both types of data, which is its significant advantage over existing feature extraction algorithms.

Keywords Classification · Feature extraction · Visualization · Genetic programming

Communicated by V. Loia.

✉ Sebastián Ventura
sventura@uco.es

Alberto Cano
acano@vcu.edu

Krzysztof J. Cios
kcios@vcu.edu

[1] Department of Computer Science, Virginia Commonwealth University, Richmond, VA 23284, USA

[2] Department of Computer Science and Numerical Analysis, University of Córdoba, Córdoba, Spain

[3] Computer Sciences Department, Faculty of Computing and Information Technology, King Abdulaziz University, Jeddah 21589, Saudi Arabia

[4] IITiS Polish Academy of Sciences, Gliwice, Poland

1 Introduction

High-dimensional data presents many challenges to machine learning algorithms. Their performance is deteriorated due to well-known problems such as the curse of dimensionality and high computational costs. These problems degrade rapidly the accuracy and efficiency of algorithms as the dimensionality increases (Verleysen and Franois 2005).

Dimensionality reduction techniques (van der Maaten et al. 2009), and specifically, feature extraction (Guyon et al. 2006), refers to the mapping of the high-dimensional data into a lower dimensional space. These techniques aim to transform the data allowing for data compression in fewer dimensions, and removal of noisy, irrelevant, or redundant features. Therefore, it is intended to improve the subsequent machine learning process on the lower dimensional space, avoiding the problems from the original high-dimensional space. Moreover, transformation of high-dimensional data onto 2D or 3D allows for human visualization of data. Data visualization (Fayyad et al. 2001) has an enormous potential for extracting previously unknown knowledge, and identifying useful data structures and patterns.

Dimensionality reduction algorithms may pursue different objectives in regards to their scope and purpose (Gisbrecht and Hammer 2015). On the one hand, unsupervised methods (Lee and Verleysen 2010), usually aim to minimize the

information loss and preserve variance (Pearson 1901), preserve data similarity (Borg and Groenen 2005; Sammon 1969), capture the lower dimensional geometric structure underlying the patterns (Fernández et al. 2015), or find self-organizing structures (Kohonen 1982). On the other hand, supervised methods (Liu and Motoda 1998), usually focus on maximizing the class discrimination to improve the classification accuracy in the new subspace (Fisher 1936).

Achieving both good classification and data visualization in the lower dimensional space is not simple (Ferreira de Oliveira and Levkowitz 2003). These objectives are conflicting and frequently it is necessary to achieve a trade-off among them. Multi-objective optimization is concerned with the simultaneous optimization of more than one objective. Evolutionary algorithms have been successfully used to resolve multi-objective optimization problems. Genetic programming (GP) is an evolutionary algorithm-based methodology which has been already applied to multi-objective optimization and feature extraction achieving good results (Fernández-Blanco et al. 2013; Icke and Rosenberg 2011; Zhang and Rockett 2007, 2010). However, these studies are limited to a low number of algorithms and data sets. Specifically, most of the data sets are binary class (two data classes), and comprise low-dimensional data with very low number of instances. Moreover, the size ratio of the classes is balanced, i.e., the number of instances belonging to each data class is similar. In recent years, the problem of learning from imbalanced data has attracted attention of both academia and industry (He and Garcia 2009; López et al. 2013). This problem concerns the performance of algorithms in the presence of classes with many times more examples than other classes. However, classic feature extraction had not heeded this problem which often happens in real-world data. Therefore, it is a necessary to design new algorithms capable of handling both balanced and imbalanced of data.

This paper presents a Pareto-based multi-objective genetic programming algorithm for feature extraction and data visualization, named MOGPFEV. Its aim is to simultaneously optimize the classification performance and visualization of the data using genetic programming-based transformations. The algorithm evolves problem transformations using a multi-objective evolutionary-based NSGA-II (Deb et al. 2002) approach. MOGPFEV takes into account the data class distribution, aiming for obtaining good classification and visualization results both on balanced and imbalanced data. The algorithm is evaluated and compared to 11 other dimensionality reduction techniques, and to the performance over the original high-dimensional data. Experimental results on 22 balanced and 20 imbalanced data sets show the good performance of MOGPFEV using three classification and three visualization measures. Results show the significantly better performance of the algorithm, especially on imbalanced data sets, outperforming the other methods in regards to the classification and visualization measures evaluated. A statistical analysis is carried out to evaluate whether there are statistically significant differences between the algorithms. This analysis supports statistically the better overall performance of our algorithm.

This paper is organized as follows. Section 2 introduces related works. Section 3 presents the MOGPFEV algorithm. Section 4 presents the experimental study, and results are discussed in Sect. 5. Finally, Sect. 6 presents the concluding remarks.

2 Background

This section reviews related works on feature extraction and data visualization, as well as multi-objective genetic programming approaches to solve this problem.

2.1 Feature extraction

Feature extraction algorithms can be grouped into two categories based on whether they focus on maintaining representation capability while reducing dimensionality or whether they seek to enhance discrimination capability.

In the former category, the objective is to maintain the representation fidelity between the original data structure and the projected data. The major concern of these methods is to maintain the discriminatory information of the original input space. van der Maaten et al. (2009) presented a comparative review of these methods. The principal component analysis (PCA) (Pearson 1901) is a well-known variance preservation method that uses orthogonal transformations. Kernel principal component analysis (KPCA) (Schölkopf et al. 1998) is a PCA extension using kernel methods. The originally linear operations of PCA are done in a kernel Hilbert space with a non-linear mapping. There are also similarity preservation methods, known as multi-dimensional scaling (MDS) (Borg and Groenen 2005). MDS methods are focused on improving visualization of the data projected into 2D or 3D, while preserving distances between objects as possible. Sammon projection (Sammon 1969) is a MDS algorithm that preserves the structure of inter-point distances in high-dimensional space in the lower dimension projection. Moreover, self-organization methods are also of high interest and produce discretized representation of the input space, such as Kohonen's self-organizing feature maps (SOMs) (Kohonen 1982). In random projection (RP) (Fradkin and Madigan 2003), the high-dimensional data are projected onto a lower dimensional subspace using a random matrix whose columns have unit lengths. RP was found to be computationally efficient, yet sufficiently accurate.

In the latter category, the objective is to enhance the discriminability between classes in the new features space and

to improve classification performance (Dhir et al. 2012). This category comprises methods such as linear discriminant analysis LDA (Fisher 1936), which finds a linear combination of features which separates two or more classes. The projection vector is obtained by maximizing the between class covariance and simultaneously minimizing the within-class covariance. LDA can be also extended with a kernel method for non-linear mapping (KDA). However, computing the projections in KDA involves eigen-decomposition of the kernel matrix, which is very expensive when there are a large number of instances. To solve this problem, spectral regression kernel discriminant analysis (KSR) simplifies KDA computation using spectral graph analysis. LDA also fails to discover local geometrical structure of the data. Locality sensitive discriminant analysis (LSDA) (Cai et al. 2007b) was proposed to solve this problem and it introduces a local manifold structure to maximize margin between classes at each local area. Liu et al. (2014) proposed a orientation distance-based discriminative feature extraction based on the Fisher discriminant idea to determine a kernel function to map the input data.

2.2 Genetic programming and multi-objective optimization

Genetic programming has been widely used for feature extraction and classification tasks (Guo et al. 2005; Krawiec 2002; Neshatian et al. 2007). Its capability to dynamically build programs and expressions is especially useful in problems characterized by a high dimensionality of the space of the features. Guo et al. (2005) applied genetic programming to generate features for bearing fault classification in machine condition monitoring. Krawiec (2002) presented a genetic programming framework for feature construction to improve symbolic classifiers while maintaining their readability. Neshatian et al. (2007) also employed genetic programming for building new features based on class dispersion and entropy. These studies propose the use of GP for feature construction in classification problems and the experimental results clearly show that this approach is effective for improving the classification accuracy. However, they focus on the optimization of a single objective, which is the classification performance.

On the other hand, multi-objective optimization has been widely employed in data mining (Mukhopadhyay et al. 2014). Specifically, as for feature extraction, several methods have been proposed to optimize both the accuracy of the classifier along with the tree size as the complexity measure (Icke and Rosenberg 2011). Considering the tree size as an objective of the algorithm aims to minimize the complexity of the data transformation, i.e., to produce simpler expressions to be more comprehensible by humans. However, the simplicity of expressions for data transformation is a very conflicting objective, especially when trading off with the accuracy, and it is difficult to optimize both simultaneously (Alcalá et al. 2008).

Icke and Rosenberg (2011) proposed a Pareto-based genetic programming multi-objective approach for feature extraction and data visualization. They identified the classification and visualization as equally important objectives of the data transformation problem for dimensionality reduction. However, their study is very limited to only four data sets and data projection to 2D. Moreover, even though they identify several visualization measures, they are not considered altogether.

Zhang and Rockett (2006, 2007, 2009) proposed a framework to produce optimal feature extractors independent of domain knowledge and class distributions using multi-objective genetic programming named MMGP. Through a multi-objective optimization process, their mappings comprised a series of mathematical transformations projecting input data into a one-dimensional decision space. Their review (Zhang and Rockett 2006) identifies two distinct research strands: either GP is used to evolve the whole classifier or tree expressions that produce a mapping to a real-valued features space which forms the input of a conventional classifier. Specifically, Zhang and Rockett (2009) presented a multi-objective approach that not only evolved the set of mappings to a multi-dimensional decision space, but also simultaneously optimized the dimensionality of that decision space.

These related works show the benefits and flexibility of multi-objective genetic programming for feature extraction. It allows efficient reduction of the data dimensionality while preserving the classification performance. The evolutionary process is capable of learning the combination of best original features to build new ones, and to overcome the presence of noisy features which may spoil the classification performance. However, the experimental studies from these related works were limited to accuracy maximization on a small number of data sets with low number of instances and balanced classes. Specifically, Zhang and Rockett (2007, 2010) experiments were limited to eight data sets up to 30 features and 1000 instances, and seven data sets, up to 21 features, and 10,996 instances, van der Maaten et al. (2009) experimented with five data sets up to ten features, and 5000 instances, Icke and Rosenberg (2011) experimented with four data sets up to 30 features and 786 instances, and Liu et al. (2014) experimented with ten data sets with up to 10,992 instances.

These methods showed good performance on balanced data as they focused on accuracy maximization. However, under the presence of imbalanced data, accuracy is a misleading metric since a default-hypothesis classifier can still achieve a very good accuracy (López et al. 2013). Moreover, we have shown the importance of data visualization when projecting data into lower dimensionality, usually 2D and 3D.

Therefore, our goal is to propose a new algorithm capable of reducing features dimension, improving data visualization and keeping classification performance simultaneously both on balanced and imbalanced data. This is the main motivation to propose a multi-objective genetic programming algorithm.

3 MOGPFEV algorithm

This section presents the multi-objective genetic programming algorithm for feature extraction and data visualization, and details the individual representation, the genetic operators, the fitness function and the evolutionary process.

3.1 Individual representation

An individual represents a set of expressions as a complete solution to the data transformation problem. Given a labeled data set in X^n the transformation problem consists in finding a mapping Z from X^n to X^m such that $m < n$. The genotype of an individual is an array of syntax trees (also known as derivation trees), whose length is the number of dimensions m of the desired subspace. The phenotype is an array of expression trees (functions), generated from the syntax trees, representing the data transformation function Z. Each data transformation function is made up of a number of mathematical operations over the original features and represents a constructed feature on the low-dimensional subspace. This encoding provides high flexibility to the algorithm to generate data transformations with any number of dimensions.

Derivation trees are created by means of a context-free grammar (Mckay et al. 2010), which establishes a formal definition of the syntactical restrictions of the problem to be solved and its possible solutions, so that only grammatically correct trees are generated. The use of a context-free grammar provides high flexibility to generate linear, non-linear, or user-defined data transformations. Figure 1 shows the grammar used to generate the derivation trees (left) and the structure of a sample derivation tree (right), where *feature* is any feature from the data set and *value* is a random continuous value.

$\langle S\rangle \rightarrow + \langle S\rangle \langle S\rangle$
$\langle S\rangle \rightarrow - \langle S\rangle \langle S\rangle$
$\langle S\rangle \rightarrow * \langle S\rangle \langle S\rangle$
$\langle S\rangle \rightarrow / \langle S\rangle \langle S\rangle$
$\langle S\rangle \rightarrow pow \langle S\rangle \langle S\rangle$
$\langle S\rangle \rightarrow \log \langle S\rangle$
$\langle S\rangle \rightarrow feature$
$\langle S\rangle \rightarrow value$

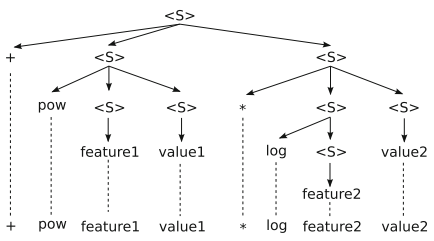

Fig. 1 Context-free grammar to generate derivation trees and a sample

The use of grammars to generate GP individuals is known as grammar-guided genetic programming, and it has been widely applied to data mining (Espejo et al. 2010; Mckay et al. 2010). The grammar generates expressions for feature extraction using the symbols and production rules in Fig. 1. The evolutionary process is responsible of finding the most appropriate data transformation function. Figure 2 shows the mapping of the derivation trees from the genotype into functions to build the new subspace such that $X^m = Z(X^n)$.

3.2 Initialization

The initialization process generates the initial population. Our algorithm employs a simple and commonly used approach to generate the individuals by means of the context-free grammar. This approach employs the production rules of the language defined by the grammar, generating only valid individuals and guaranteeing that the syntax tree is between a minimum and maximum number of derivations.

The creation of a new individual selects a number of derivations and derives the production rules to generate the syntax tree within the selected number of derivations. The derivations of the production rules start with the initial symbol of the grammar, and then choose one of the available productions. The process continues deriving the non-terminal symbols until all the non-terminal symbols have been derived to terminal ones. The process of the creation of individuals was also shown in Fig. 1, following a top-down derivation scheme by means of the grammar's production rules. A parameter controls the maximum derivation depth, both when initializing the individuals and when performing genetic operators, so that bloat is controlled. Details about parameter settings are shown in Sect. 4.3.

3.3 Genetic operators

MOGPFEV uses two genetic operators to generate new individuals. These operators are based on selective crossover and selective mutation, and their basic principles and functioning are described in this section.

Genotype = [derivationTree1, derivationTree2, derivationTree3]
derivationTree1 = <S> ➔ + <S> <S> ➔ + pow <S> <S> <S> ...
derivationTree2 = <S> ➔ / <S> <S> ➔ / feature1 value1
derivationTree3 = <S> ➔ log <S> ➔ log feature2

Phenotype = Z = [transFunction1, transFunction2, transFunction3]
transFunction1 = (+ pow feature1 value * log feature2 value2)
transFunction2 = (/ feature1 value1)
transFunction3 = (log feature2)

Fig. 2 Example of genotype–phenotype mapping of an individual

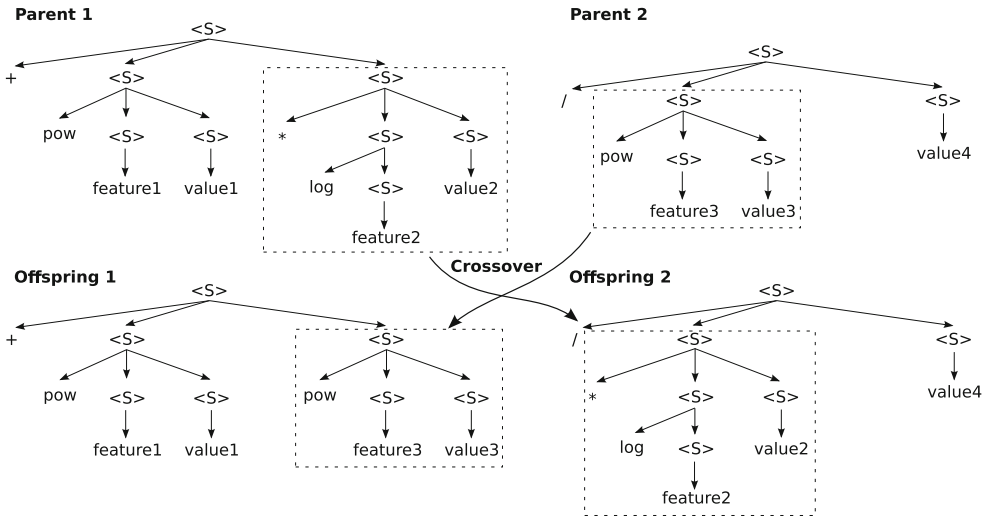

Fig. 3 Crossover operator

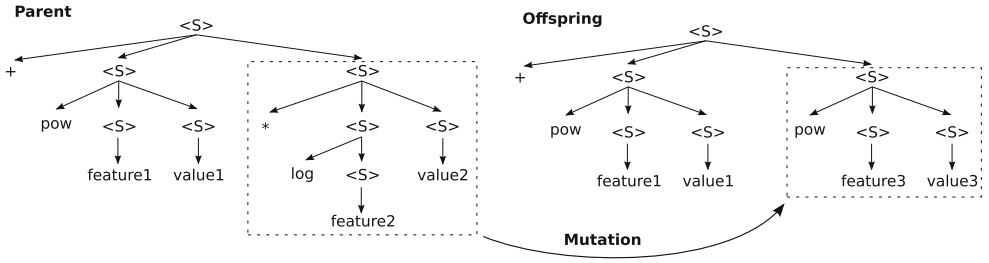

Fig. 4 Mutation operator

3.3.1 Crossover operator

The crossover operator creates new syntax tress by mixing the contents of two parent syntax trees. To do so, a non-terminal symbol is chosen at random with uniform probability. Two sub-trees (one from each parent) are selected, whose roots coincide with the symbol adopted or with a compatible symbol, and are swapped.

All non-terminal symbols (excepting the root symbol) have the same probability of being selected as the symbol from which the sub-trees are swapped. On the other hand, to reduce bloating, if one of the new offspring surpasses the maximum size allowed, one of the two parents is randomly selected to pass to the next generation without modification. If both offspring surpass this size or at least one of them does not contain a compatible symbol, the crossover is aborted and both parents are reproduced. Figure 3 shows a sample crossover of two parents to produce two offspring.

3.3.2 Mutation operator

The mutation operator is responsible for preventing the loss of genetic diversity in the population, which is highly significant in the genetic convergence process. It produces random changes in an individual to engender a new offspring. This operator randomly selects the node in the tree where the mutation is to take place. If the node is a terminal symbol, it will be replaced by another compatible terminal symbol. If the node is a non-terminal symbol, the subtree underneath this node will be substituted with any other derivation subtree. The procedure used to generate this subtree is the same as the one used to create new individuals and guarantees that the individual does not exceed the maximum size allowed. Figure 4 shows a sample mutation of a parent to produce a offspring.

3.4 Fitness function

The fitness function evaluates the quality of data transformation solutions provided by individuals. There exist many classification and visualization measures to evaluate the quality of the subspaces generated (Bertini et al. 2011; Ferri et al. 2009). Different measures allow to observe complementary aspects of the data transformation, which together increase the strength of the fitness evaluation. Two sets of measures are evaluated, namely, classification measures (accuracy, area under the curve, Cohen's kappa) and visualization measures (C-index, Davies–Bouldin index, Dunn's).

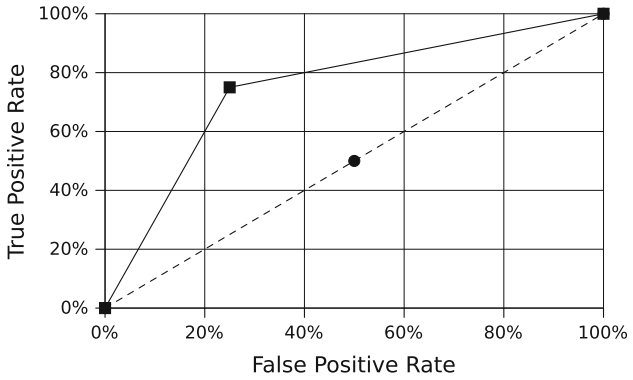

Fig. 5 Example of ROC plot. The *solid line* is a good performing classifier whereas the *dashed line* represents a random classifier

3.4.1 Classification measures

- Accuracy is the number of successful predictions relative to the total number of classifications. Unfortunately, it may be misleading when classes are strongly imbalanced, since a default-hypothesis classifier can achieve a very good accuracy. Therefore, it is not employed for imbalanced data.
- Area under the curve (AUC) (Huang and Ling 2005) shows the trade-off between the true-positive rate (*TPR*) and the false-positive rate (*FPR*) as demonstrated in López et al. (2013). The way to build the ROC space is to plot on a two-dimensional chart the true-positive rate (Y-axis) against the false-positive rate (X-axis) as shown in Fig. 5. The points (0, 0) and (1, 1) are trivial classifiers in which the class is always predicted as negative and positive, respectively, and the point (0, 1) represents perfect classification. AUC is calculated using the graphic's area:

$$\text{AUC} = \frac{1 + \text{TPR} - \text{FPR}}{2} = \frac{1 + \frac{T_P}{T_P + F_N} - \frac{F_P}{F_P + T_N}}{2}. \tag{1}$$

- Cohen's kappa rate (Ben-David 2008) evaluates the merit of the classifier, i.e., the actual hits (coverage of true positives) that can be attributed to the classifier and not to mere chance. Kappa statistic ranges from -1 (total disagreement) through 0 (random classification) to 1 (total agreement). It is calculated as follows:

$$\text{Kappa} = \frac{N \sum_{i=1}^{k} x_{ii} - \sum_{i=1}^{k} x_{i.} x_{.i}}{N^2 - \sum_{i=1}^{k} x_{i.} x_{.i}} \tag{2}$$

where x_{ii} is the count of cases in the main diagonal of the confusion matrix, N is the number of examples, and $x_{.i}$, $x_{i.}$ are the column and row total counts. Kappa penalizes all-positive or all-negative predictions (default hypothesis), which is especially important for imbalanced data. Kappa is very useful for multi-class data, measuring a classifier's accuracy while compensating for random successes.

3.4.2 Visualization measures

Visualization metrics aim to measure the *clustering* and *separability* of the data examples belonging to same/different classes. Therefore, they are based on a distance measure definition. In accordance with previous data visualization studies (Icke and Rosenberg 2011), we employ the Euclidean distance, although any other distance definition is also valid.

- C-index (Hubert and Levin 1976) is a validation index defined as follows:

$$\text{C-index} = \frac{S - S_{\min}}{S_{\max} - S_{\min}} \tag{3}$$

where S is the sum of distances over all pairs of instances from the same class. Let l be the number of those pairs. Then $S_{\min}$ is the sum of the l smallest distances if all pairs of instances are considered (regardless data class). Similarly $S_{\max}$ is the sum of the l largest distances out of all pairs. Hence, a small value of C-index indicates a good clustering of the data classes.
- Davies–Bouldin index (Davies and Bouldin 1979) is a function of the ratio of sum of within-class distance to between-class separation and is given by:

$$\text{DB} = \frac{1}{c} \sum_{i=1}^{c} \max_{i \neq k} \left\{ \frac{S(v_i) + S(v_k)}{d(v_i, v_k)} \right\} \quad \text{for } 1 \leq i, k \leq c. \tag{4}$$

The Davies–Bouldin index minimizes the within-class distance $S(v_i)$ and maximizes the between-class separation $d(v_i, v_k)$, where v_i is the sample mean (centroid) of class i, and c is the numer of classes. Therefore, the higher the similarity values within the class and the between-class separation, the lower would be the Davies–Bouldin index value. A good feature set should have the value of Davies–Bouldin index as low as possible.
- Dunn's index (Bezdek and Pal 1998) is also designed to identify sets of cluster structures that are compact and well separated. Dunn's index maximizes:

$$\text{Dunn} = \min_{i} \left\{ \min_{i \neq k} \left\{ \frac{d(v_i, v_k)}{\max_l S(v_l)} \right\} \right\} \quad \text{for } 1 \leq i, k, l \leq c. \tag{5}$$

A good feature subset should have the value of Dunn index as high as possible.

These six measures are defined as objectives of the multi-objective fitness function. The evolutionary process is responsible for optimizing them altogether. However, it is important to note that even though there are six objectives, there are actually two groups' objectives (classification metrics, visualization metrics) and each of the objectives within a group are not conflicting but complementary, i.e., they evaluate different and complementary aspects of the classification/visualization performance. Therefore, we do not face a multi-objective optimization in which all the six objectives are conflicting with each other but they are grouped into two main conflictive groups.

The outcome of the multi-objective algorithm is a set of non-dominated solutions known as Pareto optimal front. Eventually, to select a single solution from the Pareto front we rank the individuals according to how many times they obtain the best result for the objective functions. This ranking is inspired by the Friedman's M statistic procedure (García et al. 2010). Thus, we obtain a subset of the Pareto front in which all the individuals obtained the best results for a given number of metrics and then we compare pairwise solutions based on the rest of the metrics. Eventually, in case of equality we choose the one with the highest accuracy.

The evolutionary process is led by the multi-objective NSGA-II algorithm (Deb et al. 2002). NSGA-II uses dominance ranking to sort the population and determines the solutions belonging to the Pareto optimal front, achieving good spread, convergence and diversity of solutions with relatively low computational requirements.

3.5 Implementation on GPUs

Run-time of evolutionary-based algorithms, and more specifically genetic programming, is a primary concern for researchers. Over the last few years, increasing attention has focused on graphic processing units (GPUs). GPUs are devices with multi-core architectures and massive parallel processor units, which provide fast parallel hardware for a fraction of the cost of a traditional parallel system. Since the introduction of the Computer Unified Device Architecture (CUDA) in 2007, researchers have harnessed the GPU for general purpose computing, and specifically, genetic programming (Cano et al. 2012, 2015a), and dimensionality reduction (Yeh et al. 2011).

Most of the time of the genetic programming algorithm's execution on data mining problems is devoted to the fitness evaluation (Cano et al. 2012), whose complexity is typically $O(P \times N)$, where P is the population size and N is the number of data instances. Fortunately, the evaluation of the individuals of the population can be parallelized as well as their execution on each of the data instances. These approaches are known as population and data parallel, and they are able to speed up the fitness function significantly and obtain high speedups even in small data sets while also scaling efficiently to big data sets (Cano and Ventura 2014), which justifies the use of GPUs.

The implementation of the fitness function on GPUs consists of three steps. First, for each individual the Genetic Programming interpreter runs the data transformation function in parallel. Second, for each data instance their projection are mapped in parallel into the new feature space. Third, the six classification and visualization measures are computed simultaneously. This process involves thousands or even millions of threads that collaborate for fast and efficient fitness computation, solving the run-time problem of the evolutionary algorithm. More specific details about the parallel implementation are out of the scope of this paper, and the reader is referred to the articles in Cano et al. (2012, 2015a) for GPU implementation details.

4 Experimental study

This section presents the experimental study: hardware setup, data sets, and algorithms. Details about the data sets, experiments, and user-interactive online 3D browsing of data sets projections are provided as additional material at the web http://www.uco.es/grupos/kdis/wiki/MOGPFEV.

4.1 Hardware setup

The experiments were run on a machine equipped with an Intel Core i7-3820 and 12 GB DDR3. The video cards were two dual-GPU NVIDIA GTX 690 with 4 GB GDDR5. Each GTX 690 had two GPUs with 1536 CUDA cores, adding up four GPUs and 6144 CUDA cores. The host was GNU/Linux Ubuntu 12.04 with CUDA 5.0.

4.2 Data sets

Table 1 summarizes the information of the data sets used in the experimental study. These numeric data sets were collected from the KEEL (Alcalá-Fdez et al. 2011) and UCI repositories. Balanced data sets comprise binary and multi-class data, low and high number of dimensions and instances. Imbalanced data sets comprise two classes and they are depicted by means of the imbalance ratio (I. Ratio), which represents the ratio of size of the majority class (negative) to minority class (positive). These data sets outnumber studies in similar works as reviewed in Sect. 2, increasing the number of data sets and their dimensionality as measured by the number of features and instances. Specifically, the largest data set addressed in the literature contains only 11k instances whereas our largest data set contains up to 58k instances. It is very important to note that it is difficult to address big data due to the computational complexity of algorithms, e.g.

Table 1 Data sets information

Balanced data sets	Attributes	Instances	Classes
appendicitis	7	106	2
bands	19	539	2
bupa	6	345	2
cleveland	13	303	5
ecoli	7	336	8
glass	9	214	7
ionosphere	33	351	2
iris	4	150	3
madelon	500	2600	2
magic	10	19,020	
movement	90	360	15
pima	8	768	2
segment	19	2310	7
shuttle	9	58,000	
sonar	60	208	2
spectfheart	44	267	2
vehicle	18	846	4
vowel	13	990	11
wdbc	30	569	2
wine	13	178	3
winequality-white	11	4898	11
yeast	8	1484	10
Unbalanced data sets	**Attributes**	**Instances**	**I. Ratio**
abalone19	8	4174	129.4
abalone9–18	8	731	16.4
ecoli2	7	336	5.5
ecoli4	7	336	15.8
glass0	9	214	2.1
glass2	9	214	11.6
glass4	9	214	15.5
glass5	9	214	22.8
glass6	9	214	6.4
led7	7	443	11.0
page-blocks-1–3_vs_4	10	472	15.9
shuttle-c0-vs-c4	9	1829	13.9
vowel0	13	988	10.0
wisconsin	9	683	1.9
yeast-1_vs_7	7	459	14.3
yeast-2_vs_4	8	514	9.1
yeast3	8	1484	8.6
yeast4	8	1484	28.1
yeast5	8	1484	32.7
yeast-vs_7	8	693	22.1

kernel PCA is $O(n^3)$ time complex and requires $O(n^2)$ memory, which limits their application to high-dimensional data. Therefore, it is not feasible to address even larger data comprising millions of instances because the algorithms in the comparison will not run.

4.3 Algorithms

The algorithms used in the experimental study were collected from the WEKA and RapidMiner software tools, Cai (2012) and van der Maaten et al. (2009) online repositories of algorithms for dimensionality reduction.

Algorithms are applied to the original data to obtain the new subspace of features in $\mathbb{R}^3$, so that it is visualizable using 3D rotations. Classification results are obtained from the KNN classifier in the $\mathbb{R}^3$ subspace using the stratified tenfold cross-validation procedure. The KNN has been commonly employed in data visualization research (Bae et al. 2010; Venna et al. 2010). The motivation is due to its distance-based classification behavior that fits the formulas from visualization measures in a natural way, which are also based on distance-based clustering properties. Specifically, a value of $k = 3$ was employed because it obtained the best results in the majority of data sets with low deviation. The algorithms run for the comparison are the following:

1. *KSR* kernel spectral regression (Cai et al. 2007a).
   Parameters: regulation parameter 0.01, Tikhonov regularization, Gaussian kernel.
2. *LDA* linear discriminant analysis (Fisher 1936).
   Parameters: number of components 3.
3. *LSDA* locality sensitive discriminant analysis (Cai et al. 2007b).
   Parameters: Gaussian kernel, beta 0.1.
4. *MDA* multi-dimensional analysis (Biber 1992).
   Parameters: number of components 3.
5. *GHA* generalized hebbian algorithm (Sanger 1989).
   Parameters: number of components 3, learning rate 0.01.
6. *ICA* independent component analysis (Comon 1994).
   Parameters: number of components 3, alpha 1.0, tolerance 1.0E−4.
7. *PCA* principal component analysis (Pearson 1901).
   Parameters: number of components 3.
8. *KPCA* kernel principal component analysis (Schölkopf et al. 1998).
   Parameters: Radial kernel, kernel gamma 1.0.
9. *RP* random projection (Fradkin and Madigan 2003).
   Parameters: number of attributes 3

Table 2 MOGPFEV parameters

Parameter	Value
Multi-objective algorithm	NSGA-II
Population size	100
Number of generations	250
Maximum tree depth	50
Crossover operator	One-point subtree crossover
Crossover probability	0.9
Mutation operator	Subtree mutation
Mutation probability	0.1
Functions	{+, −, *, /, power, log}
Cross-validation	Ten times tenfold CV (total 100 runs)

10. *SOM* self-organizing map (Kohonen 1982).
    Parameters: number of dimensions 3, net size 30, train rounds 30, learning rate 0.8–0.01, adaption radius 10.0–1.0.
11. MMGP multi-dimensional feature extraction using multi-objective genetic programming (Zhang and Rockett 2009).
    Parameters: number of dimensions 3, population size 200, max depth 50, subtree preservation prob 0.2.

Finally, MOGPFEV has been implemented in the JCLEC software (Cano et al. 2015b) and its main parameters are shown in Table 2. The parameter values were obtained through a sensitivity analysis on a subset of data sets. The maximum tree depth was set to 50 to allow generating complex feature extraction transformations for those more difficult data sets with higher number of features. Moreover, MOGPFEV was run ten times for each fold and the average results for 100 executions are shown.

5 Results

This section presents and discusses the experimental results. First, the classification and visualization results for balanced data sets are compared. Second, the results for imbalanced data sets are compared. Third, an overall comparison to determine the optimal trade-offs for all the measures and algorithms is performed. Fourth, the convergence of the evolutionary algorithm is analyzed. Finally, the visualization of the data transformations in the 3D subspace is discussed and examples are provided.

To evaluate whether there are statistically significant differences in the performance of the algorithms, the Wilcoxon rank-sum non-parametric statistical test (Wilcoxon 1945 is used to validate pairwise comparisons among the algorithms (García et al. 2009, 2010).

5.1 Balanced data sets

Table 3 shows the classification performance in terms of the accuracy, AUC and kappa results for balanced data sets, which are to be maximized. Results reported are the mean values from the 10 runs of the tenfold cross-validation test for the data sets (rows) and methods (columns). The first column represents the results over the original high-dimensional data, whereas the rest of the columns show the results for the different methods. The two bottom rows show the average values and ranks of the algorithms. Ranks are based on the Friedman's M statistic (García et al. 2010). The rank r_{ij} of an algorithm i on a data set j is the sorted index of the results of the algorithm i as compared with the rest of the algorithms for the data set j. Finally, the rank r_i of the algorithm is calculated as the mean of the ranks for all data sets.

Table 3 results indicate that MOGPFEV obtains the best accuracy on 13 data sets, the best AUC on 11, and the best Kappa on 12 data sets. KSR obtains the second best results, followed by MMGP, but their ranks are significantly distanced from MOGPFEV. On the other hand, kernel PCA achieves the worst results among all the algorithms, with average and rank values significantly worse.

Table 4 shows the results for the visualization measures in terms of the C-index, Davies–Bouldin, and Dunn's index for balanced data sets. KSR obtains the best mean and rank results for visualization but they are very close to MOGPFEV. Rank differences between these algorithms on visualization measures are not significant. This is very important because it means that metrics evaluating visualization report similar results, whereas rank differences in classification performance are much more significant. In other words, pairwise differences between MOGPFEV and KSR are small in terms of visualization performance but are significantly larger when referring classification. Therefore, when considering a trade-off among the metrics for the two algorithms, MOGPFEV is preferred, as also later discussed. On the other hand, kernel PCA and Self-Organizing Maps are shown to achieve the worst results.

5.2 Imbalanced data sets

Previous section evaluated classification and visualization performance on balanced data sets. However, the true challenge and difficulty happen on imbalanced data, where data distribution skews significantly affect the performance of algorithms.

Table 5 shows the AUC and kappa results for imbalanced data sets. Accuracy is not considered since it is an unreliable metric for imbalanced data. The number of data sets in which MOGPFEV achieves best results (14 of 20) is increased when compared to balanced data sets. Moreover, the ranks are improved, obtaining larger differences when compared to the

Table 3 Accuracy, AUC, and kappa results for balanced data sets

Algorithm	Original	MOGPFEV	KSR	LDA	LSDA	MDA	GHA	ICA	PCA	KPCA	RP	SOM	MMGP
Accuracy (higher is better)													
appendicitis	0.8500	**0.8609**	0.8509	0.8500	0.8500	0.8218	0.8509	0.8600	0.8509	0.8491	0.8591	0.7836	0.8513
bands	0.7041	**0.7869**	0.7235	0.6358	0.6821	0.6525	0.5731	0.5754	0.5727	0.5231	0.5345	0.7037	0.7597
bupa	0.6343	**0.7187**	0.6318	0.6292	0.6495	0.6381	0.5505	0.5592	0.5563	0.4986	0.5455	0.6349	0.6566
cleveland	0.5587	**0.6162**	0.5825	0.5461	0.5793	0.5625	0.5053	0.5392	0.5359	0.4683	0.4715	0.5220	0.6035
ecoli	0.7919	**0.8102**	0.7559	0.6906	0.6332	0.6846	0.6611	0.6547	0.6901	0.5091	0.7594	0.7354	0.7379
glass	0.6823	**0.7167**	0.4994	0.6483	0.5890	0.6019	0.6019	0.6450	0.6543	0.5690	0.5091	0.5890	0.6140
ionosphere	0.8604	**0.9658**	0.9602	0.8804	0.9310	0.8917	0.7976	0.8602	0.8660	0.7663	0.8318	0.8434	0.9503
iris	0.9533	**0.9867**	0.9467	0.9467	0.9600	0.9667	0.9533	0.9133	0.9400	0.8600	0.9667	0.8533	0.9464
madelon	0.5685	0.6685	**1.0000**	0.5019	0.6638	0.6815	0.5231	0.6704	0.6677	0.5000	0.4854	0.5558	0.6640
magic	0.8272	0.8376	0.8176	0.7852	0.8024	0.7556	0.7590	0.7757	0.7721	**1.0000**	0.6699	0.8087	0.8055
movement	**0.8056**	0.6972	0.7306	0.5722	0.5722	0.5722	0.4778	0.5167	0.5250	0.1528	0.5250	0.6861	0.6879
pima	0.7436	**0.7487**	0.7397	0.7111	0.7228	0.7149	0.7007	0.6797	0.6914	0.6433	0.6849	0.7163	0.7320
segment	0.9576	**0.9762**	0.9294	0.5978	0.9377	0.1429	0.8424	0.8424	0.8364	0.6017	0.8212	0.9372	0.9034
shuttle	**0.9981**	0.9978	0.9958	0.9937	0.9953	0.9901	0.9974	0.9978	0.9974	0.7900	0.9974	0.9921	0.9953
sonar	0.8362	0.8893	**1.0000**	0.8793	0.9524	0.8840	0.6731	0.7633	0.7588	0.5338	0.5874	0.8079	0.9355
spectfheart	0.7006	**0.8650**	0.8278	0.7873	0.8323	0.8205	0.7607	0.7605	0.7234	0.7412	0.7379	0.7226	0.8408
vehicle	0.7210	0.7032	**0.7790**	0.7127	0.7577	0.7647	0.5472	0.5673	0.5815	0.2777	0.4752	0.6678	0.7099
vowel	**0.9788**	0.9040	0.6152	0.2384	0.7030	0.4212	0.5949	0.7768	0.7859	0.3798	0.7869	0.8879	0.7854
wdbc	0.9701	**0.9737**	0.9719	0.9613	0.9718	0.9718	0.9051	0.9402	0.9384	0.6204	0.8506	0.9156	0.9655
wine	0.9660	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	0.9163	0.9556	0.9552	0.5350	0.7876	0.8546	**1.0000**
winequality	**0.5615**	0.5506	0.5394	0.5163	0.5316	0.5163	0.4798	0.4980	0.4929	0.4708	0.4853	0.5500	0.5227
yeast	**0.5357**	0.5034	0.4979	0.4332	0.4670	0.4805	0.4609	0.4454	0.4441	0.3349	0.4272	0.5060	0.5056
Avg. values	0.7821	**0.8081**	0.7907	0.7053	0.7629	0.7062	0.6878	0.7180	0.7198	0.5739	0.6727	0.7397	0.7806
Avg. ranks	4.4773	**2.0909**	4.4091	7.7955	5.6364	6.7500	9.4318	8.0455	8.4318	11.8636	9.7273	7.6591	4.6818
AUC (higher is better)													
appendicitis	0.7514	0.7194	0.7431	0.7215	0.7215	0.6854	**0.7632**	0.7576	0.7410	0.7618	0.7382	0.6611	0.7190
bands	0.6815	**0.7626**	0.6863	0.5828	0.6342	0.6095	0.5166	0.5274	0.5252	0.4696	0.4859	0.6703	0.7188
bupa	0.6299	**0.7074**	0.6214	0.6206	0.6357	0.6224	0.5401	0.5369	0.5401	0.4829	0.5321	0.6237	0.6612
cleveland	0.6842	0.7382	0.7327	0.6728	0.7261	**0.7395**	0.6595	0.7116	0.6987	0.7346	0.7126	0.7265	0.7267
ecoli	0.9612	**0.9657**	0.9422	0.9314	0.9169	0.9341	0.9290	0.9247	0.9325	0.8931	0.9538	0.9450	0.9372
glass	**0.9487**	0.9417	0.8886	0.9290	0.9214	0.9218	0.9218	0.9275	0.9305	0.9174	0.8912	0.9121	0.9385
ionosphere	0.8131	**0.9569**	0.9554	0.8629	0.9144	0.8770	0.7550	0.8347	0.8407	0.7562	0.8032	0.8108	0.9216
iris	0.9767	**0.9933**	0.9733	0.9733	0.9800	0.9833	0.9750	0.9567	0.9700	0.9300	0.9833	0.9194	0.9799
madelon	0.5685	0.6685	**1.0000**	0.5019	0.6638	0.6815	0.5231	0.6704	0.6677	0.5000	0.4854	0.5558	0.6641
magic	0.7904	0.8103	0.7795	0.7515	0.7683	0.7226	0.7207	0.7407	0.7374	**1.0000**	0.6171	0.7748	0.7619
movement	**0.9851**	0.9706	0.9764	0.9542	0.9542	0.9542	0.9441	0.9529	0.9538	0.9389	0.9463	0.9698	0.9352
pima	0.7037	**0.7116**	0.6969	0.6803	0.6903	0.6729	0.6620	0.6414	0.6513	0.5879	0.6387	0.6713	0.7089
segment	0.9925	**0.9959**	0.9874	0.8856	0.9888	0.8571	0.9693	0.9681	0.9669	0.8971	0.9648	0.9883	0.9717
shuttle	0.9742	**0.9836**	0.9562	0.9624	0.9598	0.9487	0.9694	0.9710	0.9663	0.9430	0.9615	0.9506	0.9596
sonar	0.8318	0.8884	**1.0000**	0.8775	0.9508	0.8838	0.6678	0.7616	0.7566	0.5000	0.5872	0.8056	0.8631
spectfheart	0.5405	**0.7549**	0.7333	0.6391	0.6827	0.6921	0.5996	0.6543	0.5780	0.5511	0.5302	0.5259	0.7279
vehicle	0.8804	0.8715	0.9192	0.8855	0.9079	0.9126	0.7763	0.7924	0.7986	0.5424	0.7332	0.8586	**0.9255**
vowel	**0.9978**	0.9892	0.9353	0.7618	0.9556	0.8675	0.9303	0.9705	0.9726	0.8454	0.9736	0.9876	0.9405
wdbc	0.9644	**0.9721**	0.9668	0.9575	0.9669	0.9658	0.8995	0.9303	0.9290	0.5866	0.8318	0.9031	0.9237
wine	0.9852	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	0.9595	0.9792	0.9795	0.6575	0.8777	0.9235	**1.0000**
winequality	**0.9383**	0.9377	0.9362	0.9299	0.9342	0.9299	0.9214	0.9305	0.9278	0.9191	0.9249	0.9381	0.9317

Table 3 continued

Algorithm	Original	MOGPFEV	KSR	LDA	LSDA	MDA	GHA	ICA	PCA	KPCA	RP	SOM	MMGP
yeast	**0.9079**	0.8789	0.8887	0.8807	0.8931	0.8900	0.8731	0.8869	0.8886	0.8223	0.8643	0.8974	0.8711
Avg. values	0.8412	0.8736	**0.8781**	0.8165	0.8530	0.8342	0.7944	0.8194	0.8160	0.7380	0.7744	0.8191	0.8540
Avg. ranks	4.5000	**2.7500**	4.6818	7.7273	5.4545	6.4091	9.4545	7.8182	8.0227	10.9091	9.9773	7.5000	5.7955
Kappa (higher is better)													
appendicitis	0.4957	0.4667	0.4714	0.4522	0.4621	0.3868	0.5148	0.5148	0.4727	**0.5177**	0.5018	0.3107	0.4662
bands	0.3633	**0.5280**	0.3826	0.1738	0.2814	0.2296	0.0337	0.0567	0.0525	−0.0638	−0.0306	0.3513	0.4702
bupa	0.2544	**0.4178**	0.2434	0.2416	0.2758	0.2470	0.0808	0.0749	0.0791	−0.0339	0.0651	0.2480	0.3169
cleveland	0.2523	**0.3230**	0.3143	0.2204	0.2901	0.2699	0.1674	0.2272	0.2259	0.0178	0.0518	0.1784	0.3003
ecoli	0.7082	**0.7354**	0.6547	0.5592	0.4687	0.5510	0.5209	0.5134	0.5614	0.2843	0.6641	0.6218	0.6668
glass	0.5605	**0.6135**	0.3032	0.5124	0.4341	0.4419	0.4419	0.5057	0.5197	0.4039	0.3016	0.4223	0.4470
ionosphere	0.6675	**0.9230**	0.9128	0.7353	0.8466	0.7626	0.5363	0.6869	0.7003	0.5044	0.6220	0.6450	0.9117
iris	0.9300	**0.9800**	0.9200	0.9200	0.9400	0.9500	0.9300	0.8700	0.9100	0.7900	0.9500	0.7800	0.9500
madelon	0.1369	0.3369	**1.0000**	0.0038	0.3277	0.3631	0.0462	0.3408	0.3354	0.0000	−0.0292	0.1115	0.1494
magic	0.6049	0.6351	0.5826	0.5161	0.5530	0.4533	0.4549	0.4943	0.4868	**1.0000**	0.2443	0.5669	0.5072
movement	**0.7912**	0.6749	0.7107	0.5410	0.5410	0.5410	0.4397	0.4810	0.4900	0.1019	0.4905	0.6628	0.6708
pima	0.4192	**0.4332**	0.4083	0.3613	0.3835	0.3539	0.3307	0.2865	0.3082	0.1823	0.2850	0.3543	0.4061
segment	0.9505	**0.9722**	0.9177	0.5308	0.9273	0.0000	0.8162	0.8162	0.8091	0.5354	0.7914	0.9268	0.9618
shuttle	**0.9945**	0.9937	0.9879	0.9819	0.9865	0.9716	0.9925	0.9937	0.9925	0.0081	0.9925	0.9772	0.9863
sonar	0.6673	0.7770	**1.0000**	0.7564	0.9041	0.7668	0.3373	0.5235	0.5138	0.0000	0.1733	0.6129	0.7471
spectfheart	0.0770	**0.5524**	0.4678	0.2986	0.4137	0.4072	0.2098	0.3090	0.1410	0.1178	0.0536	0.0595	0.5061
vehicle	0.6281	0.6042	**0.7052**	0.6170	0.6769	0.6863	0.3972	0.4241	0.4430	0.0405	0.3015	0.5573	0.6299
vowel	**0.9767**	0.8944	0.5767	0.1622	0.6733	0.3633	0.5544	0.7544	0.7644	0.3178	0.7656	0.8767	0.7492
wdbc	0.9351	**0.9435**	0.9393	0.9172	0.9392	0.9389	0.7973	0.8701	0.8667	0.1757	0.6744	0.8165	0.8946
wine	0.9487	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	0.8740	0.9328	0.9321	0.2882	0.6797	0.7806	**1.0000**
winequality	**0.3429**	0.3261	0.3116	0.2764	0.2991	0.2764	0.2185	0.2540	0.2432	0.2126	0.2294	0.3250	0.2854
yeast	**0.3987**	0.3550	0.3492	0.2679	0.3124	0.3277	0.2970	0.2805	0.2793	0.1230	0.2526	0.3627	0.3374
Avg. values	0.5956	**0.6585**	0.6436	0.5021	0.5880	0.5131	0.4542	0.5096	0.5058	0.2511	0.4105	0.5249	0.6073
Avg. ranks	4.2955	**2.3182**	4.4091	7.9318	5.6591	6.8864	9.3182	7.8864	8.2273	11.4545	10.0455	7.7727	4.7955

Bold results indicate better performance

Table 4 C-index, Davies–Bouldin, and Dunn's index results for balanced data sets

Algorithm	Original	MOGPFEV	KSR	LDA	LSDA	MDA	GHA	ICA	PCA	KPCA	RP	SOM	MMGP
C-index (lower is better)													
appendicitis	0.3364	0.4812	0.2561	0.3598	**0.2470**	0.3853	0.3013	0.4131	0.3424	0.5380	0.3230	0.5429	0.3548
bands	0.4908	**0.3672**	0.3939	0.4766	0.4008	0.4811	0.4730	0.4985	0.5002	0.5030	0.4957	0.4898	0.3940
bupa	0.5089	**0.4014**	0.4480	0.4840	0.4787	0.5097	0.5092	0.5062	0.5085	0.5092	0.5201	0.4895	0.4201
cleveland	0.4209	**0.1354**	0.2061	0.2312	0.1823	0.2149	0.2598	0.3847	0.3261	0.4036	0.4798	0.4315	0.1700
ecoli	0.1868	**0.1137**	0.1505	0.1263	0.1707	0.1446	0.1821	0.1651	0.1644	0.4373	0.1857	0.4594	0.1799
glass	0.2689	0.1782	0.1811	0.1591	**0.1439**	0.2515	0.2515	0.2420	0.2507	0.4510	0.3110	0.4719	0.1857
ionosphere	0.3272	0.1567	**0.0417**	0.2877	0.1967	0.1840	0.4159	0.3792	0.4139	0.5668	0.3446	0.4210	0.1976
iris	0.0468	**0.0000**	0.0356	0.0181	0.0410	0.0428	0.1986	0.2316	0.1000	0.1676	0.0224	0.4396	0.0347
madelon	0.4869	0.4837	**0.0000**	0.4995	0.3795	0.4431	0.4988	0.4851	0.4837	0.4997	0.4997	0.4992	0.4899
magic	0.3759	0.2684	0.2289	0.3277	0.2992	0.2955	0.3544	0.3712	0.3848	**0.0000**	0.4685	0.4790	0.2805
movement	0.2866	0.1364	0.1069	0.0937	0.0920	0.0920	0.1980	0.2835	0.2826	**0.0620**	0.2350	0.3691	0.1484
pima	0.4496	0.3761	**0.3359**	0.3667	0.3510	0.3956	0.3892	0.4210	0.4156	0.5183	0.4492	0.4309	0.3840
segment	0.1749	0.0384	0.0604	0.0516	0.0464	**0.0000**	0.1073	0.1188	0.1013	0.3225	0.1313	0.3875	0.1518

Table 4 continued

Algorithm	Original	MOGPFEV	KSR	LDA	LSDA	MDA	GHA	ICA	PCA	KPCA	RP	SOM	MMGP
shuttle	0.1632	**0.0294**	0.0349	0.0580	0.0467	0.0412	0.2464	0.2295	0.1629	0.6032	0.3462	0.4724	0.1329
sonar	0.4633	0.3267	**0.0241**	0.2340	0.2639	0.4274	0.4581	0.4278	0.4484	0.5096	0.4759	0.4881	0.3770
spectfheart	0.6669	0.5750	**0.3107**	0.4893	0.3602	0.5339	0.6229	0.6460	0.6465	0.6309	0.6340	0.5024	0.6461
vehicle	0.3400	0.2527	**0.1179**	0.1821	0.1766	0.1783	0.3783	0.3858	0.3852	0.4158	0.4072	0.4532	0.3818
vowel	0.3636	0.3067	0.1640	0.5449	0.1513	0.5781	0.3802	0.3471	0.3581	**0.1474**	0.3954	0.4533	0.3225
wdbc	0.1745	0.0545	**0.0406**	0.1286	0.0890	0.1544	0.1540	0.2663	0.1891	0.5252	0.2205	0.4948	0.1486
wine	0.1763	**0.0114**	0.0131	0.0198	0.0267	0.0218	0.1653	0.1465	0.0951	0.4210	0.2643	0.4751	0.0219
winequality	0.4293	0.3372	**0.3355**	0.3822	0.3690	0.3822	0.4344	0.4309	0.4277	0.4372	0.4408	0.5050	0.3936
yeast	0.2900	**0.1968**	0.2686	0.2040	0.2230	0.2126	0.2839	0.2703	0.2726	0.4564	0.3142	0.4768	0.2377
Avg. values	0.3376	0.2376	**0.1707**	0.2602	0.2153	0.2714	0.3301	0.3477	0.3300	0.4148	0.3620	0.4651	0.2752
Avg. ranks	9.2273	3.0227	**2.5909**	4.7955	3.3864	5.5227	8.1364	8.6364	8.1136	10.3182	10.0682	11.4545	5.7273
Davies–Bouldin index (lower is better)													
appendicitis	1.5412	2.1611	**0.9365**	1.5527	1.3122	1.9133	1.0506	2.0747	1.5076	4.1853	1.2393	3.6241	2.4718
bands	2.8568	1.8544	**1.5624**	8.3976	2.2768	4.2983	11.5808	8.5783	8.8031	6.9873	10.2282	9.0085	1.7315
bupa	5.8088	2.2112	**1.7238**	7.9703	6.9979	5.9849	24.7047	9.4407	10.1881	19.8766	14.3660	12.0100	3.9159
cleveland	9.7509	**2.9771**	3.2328	4.7953	4.9802	3.7704	4.9112	7.9820	7.1330	6.1188	10.8975	8.9140	3.0070
ecoli	2.6671	**1.7006**	5.2657	2.4068	3.4898	3.1622	2.4513	2.9094	2.9047	3.5694	3.2481	7.1969	4.5003
glass	3.2828	1.8074	2.2780	**1.5294**	3.2530	3.3421	3.3421	5.0037	4.9004	3.8123	6.3051	5.4339	1.9745
ionosphere	4.0864	0.9064	**0.3723**	2.7314	1.1714	1.1283	1.9559	1.9599	2.4564	2.1360	2.3452	4.0683	1.6169
iris	0.7517	**0.1625**	0.6571	0.6506	0.8709	0.8770	1.1986	1.7488	1.0527	2.5993	0.4334	5.1770	0.6420
madelon	15.6150	3.6206	**0.0566**	26.8700	1.5226	3.5264	20.0013	8.3150	7.9072	2.7284	37.4453	35.1325	6.8540
magic	2.9098	1.1238	0.8727	2.1685	1.4213	1.4384	1.7664	2.7692	2.6960	**0.0000**	12.8663	9.8239	1.2006
movement	3.8585	2.5307	**1.6741**	3.3429	3.2823	3.2823	7.9126	7.1679	7.1191	Infinity	4.4297	21.4245	3.2061
pima	4.4275	2.1032	**1.1372**	2.3448	2.2855	2.6306	1.7173	3.0699	2.8567	4.1393	3.5720	4.9474	2.6606
segment	2.3385	1.1565	**1.0371**	2.7426	1.3418	Infinity	2.6068	4.2513	4.6796	3.7534	3.8992	5.7274	2.1096
shuttle	2.3892	**0.6780**	2.6253	1.8132	3.8219	2.3846	2.6153	2.5394	2.5642	12.4835	2.4188	4.6580	2.5524
sonar	5.6858	1.5329	**0.3687**	1.2257	1.4037	3.1727	2.2007	2.6753	3.1245	3.0972	4.3962	13.5245	2.9893
spectfheart	2.5476	1.5943	**0.7465**	2.0192	2.0579	1.7368	1.2464	2.4774	1.8378	4.2397	5.0140	7.1312	3.1157
vehicle	12.6298	2.4260	2.0243	4.0087	1.7651	**1.7595**	69.0200	10.5237	13.8463	19.5255	8.0510	21.7123	4.2044
vowel	11.0796	4.0385	3.8141	414.126	**2.2120**	325.224	30.6485	10.2767	10.6462	13.3098	13.1484	15.2452	14.0366
wdbc	0.7206	0.5007	**0.4036**	0.9174	0.5714	1.0427	0.6289	1.5869	1.0793	8.0061	1.0059	4.0094	0.7164
wine	1.5155	**0.4074**	0.4265	0.5880	0.5519	0.5804	1.6086	1.0213	0.8681	5.8600	1.2457	9.8819	0.4237
winequality	6.5613	**2.6972**	3.1796	3.9485	3.8511	3.9485	5.0556	8.2355	8.9029	15.8717	13.4326	8.5409	3.1003
yeast	2.9282	2.9720	3.0856	3.5222	3.4591	3.1428	4.1633	**2.8400**	2.9075	8.8862	3.7752	7.4070	3.0711
Avg. values	4.8160	1.8710	**1.7037**	22.7124	2.4500	17.1976	9.1994	4.8840	4.9992	6.8721	7.4438	10.2090	3.1864
Avg. ranks	7.5909	**2.7273**	2.9091	6.2500	4.8864	6.2955	7.7045	8.1364	8.0909	9.8636	9.3182	11.9091	5.3182
Dunn's index (higher is better)													
appendicitis	0.2783	0.0522	**0.3293**	0.2308	0.2732	0.1650	0.2741	0.2255	0.2738	0.1022	0.2501	0.1497	0.1999
bands	0.0734	0.0835	**0.1765**	0.0351	0.1325	0.0977	0.0262	0.0572	0.0581	0.0062	0.0395	0.0684	0.1358
bupa	0.0433	0.0896	**0.0947**	0.0371	0.0482	0.0424	0.0109	0.0350	0.0324	0.0189	0.0140	0.0507	0.0803
cleveland	0.0225	0.0304	**0.1041**	0.0684	0.0740	0.0919	0.0437	0.0406	0.0484	0.0014	0.0249	0.0497	0.0646
ecoli	0.0873	**0.1597**	0.0139	0.0292	0.0253	0.0319	0.0400	0.0321	0.0341	0.0667	0.0400	0.0566	0.0870
glass	0.0282	0.0567	0.0400	0.0913	0.0159	0.0312	0.0312	0.0131	0.0155	**Infinity**	0.0164	0.0769	0.0504
ionosphere	0.1430	0.5219	**0.7054**	0.0599	0.2400	0.3057	0.1883	0.2225	0.1925	0.0664	0.1498	0.1404	0.5886
iris	0.4238	**0.8685**	0.4305	0.6123	0.4474	0.4324	0.2340	0.2122	0.2947	0.1240	0.7192	0.0993	0.6316
madelon	0.0542	0.0107	**4.4856**	0.0119	0.1737	0.1038	0.0068	0.0550	0.0555	0.0009	0.0111	0.0171	0.3176
magic	0.1438	0.2276	0.2688	0.0721	0.1506	0.1608	0.1140	0.1336	0.1566	**Infinity**	0.0212	0.0557	0.2159

Table 4 continued

Algorithm	Original	MOGPFEV	KSR	LDA	LSDA	MDA	GHA	ICA	PCA	KPCA	RP	SOM	MMGP
movement	0.0690	0.0777	**0.1115**	0.0284	0.0292	0.0292	0.0153	0.0208	0.0219	0.0000	0.0354	0.0047	0.0530
pima	0.0522	0.1206	**0.2056**	0.1762	0.1175	0.1190	0.1522	0.1103	0.1191	0.0548	0.0934	0.1169	0.1132
segment	0.0301	0.0880	0.1470	0.0059	0.0668	**Infinity**	0.0140	0.0061	0.0068	0.0422	0.0356	0.0776	0.0827
shuttle	0.0672	**0.0719**	0.0593	0.0121	0.0128	0.0139	0.0393	0.0561	0.0718	0.0054	0.0506	0.0672	0.0650
sonar	0.1261	0.1889	**0.9504**	0.3664	0.2132	0.1197	0.1353	0.1840	0.1684	0.0117	0.1158	0.0447	0.1424
spectfheart	0.1630	0.2636	**0.3897**	0.1914	0.1357	0.1412	0.1599	0.0946	0.1427	0.0163	0.0536	0.0835	0.2413
vehicle	0.0208	0.0305	0.1128	0.0636	0.1622	**0.1727**	0.0034	0.0117	0.0142	0.0052	0.0313	0.0155	0.0706
vowel	0.0424	0.1025	0.0510	0.0009	**0.1280**	0.0007	0.0038	0.0161	0.0174	0.0025	0.0357	0.0224	0.0777
wdbc	0.2320	**0.6475**	0.6241	0.2433	0.1959	0.1867	0.3556	0.1554	0.2637	0.0060	0.2600	0.1346	0.6389
wine	0.1106	**1.0577**	0.8033	0.6930	0.5685	0.7963	0.1125	0.2329	0.3285	0.0282	0.2055	0.0494	0.8922
winequality	**Infinity**	**Infinity**	**Infinity**	**Infinity**	**Infinity**	**Infinity**	**Infinity**	**Infinity**	**Infinity**	**Infinity**	**Infinity**	**Infinity**	**Infinity**
yeast	0.0657	**0.0864**	0.0482	0.0138	0.0212	0.0233	0.0231	0.0374	0.0369	0.0260	0.0551	0.0332	0.0496
Avg. values	0.1035	0.2198	**0.4614**	0.1383	0.1469	0.1393	0.0902	0.0887	0.1070	0.0266	0.1026	0.0643	0.2181
Avg. ranks	6.9773	4.0000	**3.2273**	7.7273	6.2955	6.6364	8.7273	8.8636	7.2727	10.3636	8.2500	8.4773	4.1818

Bold results indicate better performance

Table 5 AUC and kappa results for imbalanced data sets

Algorithm	Original	MOGPFEV	KSR	LDA	LSDA	MDA	GHA	ICA	PCA	KPCA	RP	SOM	MMGP
AUC (higher is better)													
abalone19	0.5000	**0.5143**	0.4999	0.4998	0.4996	0.4999	0.5000	0.4999	0.4998	0.4998	0.4998	0.4995	0.4665
abalone9–18	0.5950	**0.8047**	0.7066	0.7435	0.6844	0.7102	0.5325	0.5268	0.5149	0.5822	0.5621	0.6086	0.7190
ecoli2	0.9331	**0.9531**	0.8683	0.8697	0.8632	0.9167	0.8935	0.8092	0.8318	0.5876	0.8347	0.8436	0.8443
ecoli4	0.8984	**0.9500**	0.9484	0.9250	0.8734	0.8968	0.7437	0.6655	0.7155	0.6389	0.6623	0.7702	0.9172
glass0	0.7820	**0.9047**	0.6924	0.6890	0.7069	0.7315	0.6959	0.7671	0.7562	0.7091	0.6252	0.7059	0.7763
glass2	0.5816	**0.7450**	0.6006	0.5824	0.6258	0.6382	0.5533	0.5156	0.5206	0.5822	0.5124	0.6174	0.6224
glass4	0.8185	**0.9975**	0.7326	0.7876	0.7493	0.8401	0.4927	0.6400	0.5992	0.7159	0.5659	0.7617	0.8616
glass5	**0.7902**	0.7427	0.5951	0.6927	0.7476	0.6951	0.6378	0.7378	0.6378	0.4780	0.6951	0.7451	0.6374
glass6	0.8413	**0.9667**	0.9032	0.9086	0.9086	0.9086	0.8805	0.9086	0.9113	0.7042	0.7111	0.8771	0.8673
led7	**0.9092**	0.8788	**0.9092**	0.8795	0.8372	0.8937	0.8446	0.8460	0.8460	0.8259	0.8814	0.8383	0.8327
page-blocks-4	0.9633	0.9777	0.8977	0.9088	**0.9955**	0.9566	0.8965	0.9033	0.9411	0.7732	0.7577	0.7777	0.8638
shuttle-0–4	0.9958	**1.0000**	0.9958	**1.0000**	0.9958	**1.0000**	0.9958	0.9958	**1.0000**	0.9721	0.9958	0.9872	0.9966
vowel0	**1.0000**	0.9833	0.9822	0.8811	**1.0000**	0.9372	0.8655	0.9350	0.9350	0.6855	0.8233	0.9817	0.9566
wisconsin	0.9720	**0.9827**	0.9674	0.9629	0.9708	0.9664	0.9643	0.9594	0.9613	0.9270	0.9600	0.9462	0.9373
yeast-1_vs_7	0.6275	**0.7775**	0.6039	0.6252	0.6608	0.6252	0.5550	0.5620	0.5442	0.4953	0.6620	0.5729	0.6499
yeast-2_vs_4	0.7915	**0.8563**	0.8330	0.8363	0.8273	0.8173	0.7247	0.8129	0.8241	0.5454	0.7161	0.8118	0.8280
yeast3	0.8309	**0.8578**	0.8378	0.8223	0.7992	0.8457	0.6628	0.6762	0.6739	0.5552	0.7435	0.8146	0.8065
yeast4	0.6133	0.6222	0.6040	**0.6615**	0.5633	0.5946	0.5048	0.5644	0.5648	0.5165	0.5165	0.6220	0.5913
yeast5	0.8378	0.8319	0.7052	0.7424	0.6844	**0.8455**	0.7208	0.7128	0.7115	0.4976	0.5938	0.7802	0.7148
yeast-vs_7	0.5273	**0.6644**	0.5121	0.4947	0.4985	0.4985	0.5152	0.5121	0.5106	0.4970	0.5129	0.4955	0.5416
Avg. values	0.7904	**0.8506**	0.7698	0.7756	0.7746	0.7909	0.7090	0.7275	0.7250	0.6394	0.6916	0.7529	0.7716
Avg. ranks	4.4500	**1.7750**	6.1750	5.9000	6.3500	4.6750	8.8750	8.4250	8.3250	11.6000	9.7500	8.0000	6.7000
Kappa (higher is better)													
abalone19	0.0000	**0.0497**	−0.0004	−0.0008	−0.0012	−0.0004	0.0000	−0.0004	−0.0007	−0.0008	−0.0007	−0.0016	0.0246
abalone9–18	0.2661	**0.6915**	0.4798	0.5489	0.4312	0.5168	0.1049	0.0776	0.0427	0.2097	0.1672	0.2746	0.5420
ecoli2	0.8626	**0.8889**	0.6914	0.7455	0.7497	0.8486	0.7704	0.6137	0.6711	0.1983	0.6487	0.6856	0.7706
ecoli4	0.8473	**0.9398**	0.9167	0.9005	0.8172	0.8291	0.5606	0.3674	0.4768	0.3336	0.3662	0.6202	0.9048

Table 5 continued

Algorithm	Original	MOGPFEV	KSR	LDA	LSDA	MDA	GHA	ICA	PCA	KPCA	RP	SOM	MMGP
glass0	0.5377	**0.8092**	0.3899	0.3778	0.4101	0.4621	0.3928	0.5208	0.5117	0.4260	0.2512	0.4249	0.5747
glass2	0.2088	**0.5499**	0.1545	0.1350	0.2352	0.2282	0.1599	0.0150	0.0647	0.1820	0.0452	0.2530	0.3400
glass4	0.5875	**0.9576**	0.4099	0.4962	0.4499	0.6533	−0.0118	0.3634	0.2472	0.4492	0.1212	0.6018	0.4742
glass5	**0.6154**	0.4526	0.2199	0.3353	0.5575	0.4526	0.2113	0.4568	0.2778	−0.0397	0.3871	0.4887	0.3226
glass6	0.7559	**0.9550**	0.8057	0.8377	0.8357	0.8357	0.7540	0.8358	0.8538	0.4229	0.4606	0.7566	0.8164
led7	**0.8097**	0.7371	**0.8097**	0.7534	0.6905	0.7783	0.6407	0.6789	0.6789	0.6829	0.6742	0.6724	0.7076
page-blocks-4	**0.9574**	0.9412	0.7936	0.8278	0.9334	0.8556	0.7453	0.8313	0.9012	0.6210	0.5872	0.6467	0.8269
shuttle-0–4	0.9955	**1.0000**	0.9955	**1.0000**	0.9955	**1.0000**	0.9955	0.9955	**1.0000**	0.9276	0.9955	0.9783	0.9464
vowel0	**1.0000**	0.9812	0.9693	0.7863	**1.0000**	0.8614	0.7758	0.8872	0.8872	0.3795	0.7306	0.9630	0.9396
wisconsin	0.9422	**0.9615**	0.9267	0.9172	0.9360	0.9265	0.9233	0.9194	0.9198	0.8368	0.9138	0.8882	0.9027
yeast-1_vs_7	0.3348	**0.6272**	0.2623	0.3234	0.4110	0.3020	0.1451	0.1751	0.1253	−0.0143	0.4281	0.1768	0.4210
yeast-2_vs_4	0.6984	**0.7526**	0.7011	0.7343	0.7291	0.7145	0.5274	0.6728	0.7019	0.1151	0.5177	0.6624	0.7158
yeast3	0.7096	**0.7312**	0.6916	0.6675	0.6400	0.7124	0.3813	0.3908	0.3888	0.1386	0.5486	0.6839	0.6552
yeast4	0.2978	0.2969	0.2795	**0.3984**	0.1794	0.2531	0.0084	0.1740	0.1761	0.0545	0.0521	0.3070	0.2756
yeast5	**0.7302**	0.6261	0.4136	0.4989	0.4048	0.6379	0.4414	0.4925	0.5004	−0.0081	0.2313	0.5983	0.4875
yeast-vs_7	0.0823	**0.4438**	0.0304	−0.0154	−0.0050	−0.0050	0.0503	0.0432	0.0326	−0.0094	0.0382	−0.0128	0.1333
Avg. values	0.6120	**0.7196**	0.5470	0.5634	0.5700	0.5931	0.4288	0.4755	0.4729	0.2953	0.4082	0.5334	0.5891
Avg. ranks	3.9500	**1.9000**	6.8500	6.4000	6.2000	5.0000	9.5000	8.2750	8.0500	11.3250	10.2000	7.8000	5.5500

Bold results indicate better performance

Table 6 C-index, Davies–Bouldin, and Dunn's results for imbalanced data sets

Algorithm	Original	MOGPFEV	KSR	LDA	LSDA	MDA	GHA	ICA	PCA	KPCA	RP	SOM	MMGP
C-index (lower is better)													
abalone19	0.7053	**0.0000**	0.6521	0.6643	0.6510	0.7062	0.7055	0.7439	0.7148	0.6327	0.6988	0.5735	0.6277
abalone9–18	0.5408	**0.0000**	0.2055	0.2716	0.2719	0.6034	0.5657	0.4846	0.5478	0.6083	0.5452	0.4262	0.3070
ecoli2	0.4511	0.2223	**0.1724**	0.3465	0.2666	0.3806	0.5805	0.4233	0.4522	0.5238	0.6036	0.4208	0.2335
ecoli4	0.3974	**0.0000**	0.0776	0.1958	0.2557	0.3039	0.5161	0.4995	0.5014	0.6809	0.6528	0.3567	0.1404
glass0	0.5625	0.5145	0.4585	0.5046	0.5058	0.5668	0.5628	0.5662	0.5594	**0.4218**	0.5944	0.4610	0.5510
glass2	0.7050	0.5549	0.5522	0.5171	**0.5165**	0.7298	0.7005	0.7168	0.7133	0.5822	0.7134	0.5272	0.5725
glass4	0.3313	**0.1133**	0.1771	0.4215	0.2747	0.3320	0.5333	0.4024	0.4192	0.6611	0.3958	0.5252	0.1459
glass5	0.5453	**0.0000**	0.2893	0.5162	0.4557	0.5881	0.7054	0.5892	0.5962	0.6171	0.5576	0.5078	0.3443
glass6	0.2981	**0.0472**	0.0745	0.1232	0.0726	0.3687	0.2915	0.2924	0.2761	0.5448	0.5828	0.4548	0.0620
led7	0.2528	**0.0000**	0.1502	0.2354	0.2200	0.2149	0.3537	0.3481	0.2916	0.5095	0.3578	0.4497	0.1784
page-blocks-4	0.5239	**0.0000**	0.1280	0.2932	0.2498	0.2895	0.3870	0.3789	0.3701	0.6734	0.4380	0.4506	0.1660
shuttle-0–4	0.1972	**0.0000**	0.0002	**0.0000**	0.0074	0.0162	0.1447	0.0196	0.0115	0.3797	0.3787	0.4427	0.0153
vowel0	0.5180	0.1145	**0.0334**	0.3355	0.1230	0.4260	0.5365	0.2819	0.2966	0.8443	0.5306	0.5269	0.1741
wisconsin	0.0553	0.0363	**0.0261**	0.0648	0.0433	0.0382	0.0473	0.1591	0.0622	0.3510	0.0798	0.3392	0.0371
yeast-1_vs_7	0.5388	**0.3229**	0.4011	0.4634	0.3855	0.4324	0.5476	0.5998	0.5794	0.5305	0.4844	0.4688	0.3891
yeast-2_vs_4	0.3303	**0.1125**	0.1566	0.2968	0.2572	0.2922	0.3931	0.4348	0.3990	0.6220	0.4208	0.4435	0.1414
yeast3	0.5728	0.2214	**0.2063**	0.3424	0.3508	0.4096	0.6128	0.6382	0.6344	0.6272	0.5186	0.3792	0.2386
yeast4	0.5175	**0.0000**	0.3709	0.5603	0.4993	0.4175	0.5972	0.6455	0.6005	0.7149	0.5601	0.4973	0.3966
yeast5	0.4186	**0.0000**	0.1772	0.3314	0.2749	0.2760	0.5025	0.5644	0.4921	0.7303	0.5438	0.4364	0.1907
yeast-vs_7	0.6483	0.5815	0.5723	0.6814	0.5321	0.6120	0.6508	0.6384	0.6228	0.5493	0.6568	**0.4937**	0.6788
Avg. values	0.4555	**0.1421**	0.2441	0.3583	0.3107	0.4002	0.4967	0.4714	0.4570	0.5902	0.5157	0.4591	0.2795
Avg. ranks	8.3500	**1.8750**	2.5500	5.9750	3.9500	7.2500	9.9500	9.8000	9.0000	10.6000	10.1500	7.6000	3.9500

Table 6 continued

Algorithm	Original	MOGPFEV	KSR	LDA	LSDA	MDA	GHA	ICA	PCA	KPCA	RP	SOM	MMGP
Davies–Bouldin index (lower is better)													
abalone19	2.2554	**0.0000**	0.9876	2.0267	2.3029	2.7478	2.0107	3.2488	2.1697	3.1487	1.9298	9.9434	1.5077
abalone9–18	1.9954	**0.0000**	0.7882	1.3556	1.3111	3.2307	1.8963	2.1371	1.9551	2.7316	1.6432	4.2405	1.3941
ecoli2	1.7762	0.7190	**0.6404**	0.9888	1.0652	1.3913	2.3411	1.1973	1.3825	2.7122	2.0222	5.7737	1.6878
ecoli4	1.3693	**0.0000**	0.4163	0.7450	0.8075	0.9148	1.1758	1.1937	1.3014	2.0276	1.5986	3.9057	1.2200
glass0	2.4441	1.5323	**1.0674**	2.0904	4.5022	3.5115	3.1744	2.4819	2.4357	6.1717	5.5887	7.6177	1.2865
glass2	2.8576	1.4813	**0.8783**	2.6832	2.4748	4.1056	4.6206	3.7516	3.6904	6.1128	4.8903	10.5380	1.2256
glass4	1.8035	**0.6585**	0.6790	1.4228	1.8854	1.5128	2.9367	1.9902	2.1864	1.6467	3.1736	2.0009	0.6681
glass5	1.9111	**0.0000**	0.6752	1.2092	1.7171	1.7182	1.8954	1.6765	1.9118	1.8412	1.5212	2.9780	1.3517
glass6	1.1281	**0.2901**	0.3635	0.7018	0.7421	0.9898	1.1807	0.9026	0.8518	2.7800	8.9340	4.3092	0.3294
led7	1.3833	**0.0000**	0.3843	1.0082	0.9563	0.9626	0.8071	1.2643	1.0586	1.6273	1.5327	3.4907	0.8978
page-blocks-4	1.2765	**0.0000**	0.7594	1.2261	1.4965	1.5592	1.4123	1.5953	1.3800	2.4154	2.8841	8.3396	1.3892
shuttle-0–4	1.5583	**0.0000**	0.1101	0.2924	0.2473	0.8714	0.7079	0.8679	0.8185	0.9721	0.9821	3.1724	0.5561
vowel0	3.6861	0.6761	**0.3080**	3.4815	1.6791	3.8440	4.3285	1.9665	2.1306	3.7444	6.7071	4.7433	0.9508
wisconsin	0.7960	0.3950	**0.2301**	0.7211	0.4795	0.4997	0.3854	1.3888	0.6073	0.8744	0.5679	2.2923	0.7219
yeast-1_vs_7	2.8310	1.6015	**1.0153**	1.9958	2.1960	2.0019	6.7876	2.4553	2.2917	26.1674	3.0078	4.6312	1.6225
yeast-2_vs_4	1.4523	1.1284	**0.6194**	1.1483	1.0285	1.1075	1.2856	1.2061	1.1026	1.5927	1.5694	2.2489	1.9154
yeast3	2.4775	0.6474	**0.5862**	1.1163	1.3544	1.5419	2.5990	2.1148	2.2710	4.9763	1.5478	3.3219	1.6375
yeast4	1.8334	**0.0000**	0.8081	1.4722	1.4233	1.4448	1.9297	1.6374	1.4712	2.3581	1.7932	2.6704	1.3960
yeast5	1.0145	**0.0000**	0.4216	0.9488	0.7621	0.6701	0.8622	0.8831	0.7794	5.4913	1.2872	3.0523	0.4146
yeast-vs_7	4.1877	2.6958	**1.5888**	2.8877	3.7910	2.3963	9.1324	3.9959	3.7353	4.5288	3.6757	8.5801	2.7258
Avg. values	2.0019	**0.5913**	0.6664	1.4761	1.6111	1.8511	2.5735	1.8978	1.7765	4.1960	2.8428	4.8925	1.2449
Avg. ranks	8.8500	1.8000	**1.7000**	4.9500	5.3500	6.8500	8.5000	8.1500	7.1000	11.1000	9.4500	12.5000	4.7000
Dunn's index (higher is better)													
abalone19	0.1166	**Infinity**	0.1189	0.0666	0.0739	0.0742	0.1063	0.0159	0.0651	0.1483	0.1201	0.0513	0.0900
abalone9–18	0.1459	**Infinity**	0.2437	0.1783	0.1963	0.0729	0.1467	0.1102	0.1544	0.1368	0.1730	0.1545	0.2038
ecoli2	0.2503	0.2601	**0.4372**	0.1753	0.2788	0.3197	0.1098	0.1121	0.1108	0.1501	0.0833	0.1134	0.2303
ecoli4	0.3202	**Infinity**	0.6802	0.5260	0.4266	0.3761	0.2514	0.1166	0.1205	0.1400	0.1132	0.1822	0.5790
glass0	0.1178	0.2479	**0.2746**	0.1340	0.0508	0.0436	0.0774	0.0788	0.0977	0.0891	0.0356	0.0853	0.1881
glass2	0.0868	**0.2762**	0.1838	0.1029	0.1325	0.0410	0.0491	0.0508	0.0594	0.0744	0.0391	0.0571	0.2403
glass4	0.2294	**0.7554**	0.4584	0.2901	0.2197	0.2765	0.1123	0.1807	0.1726	0.1672	0.1094	0.2391	0.3507
glass5	0.1803	**Infinity**	0.3384	0.2355	0.2105	0.1289	0.1142	0.1646	0.1440	0.1887	0.1567	0.1895	0.2641
glass6	0.2585	**0.9014**	0.6229	0.4541	0.4323	0.1553	0.2399	0.2077	0.2588	0.1605	0.0253	0.1483	0.5030
led7	0.5042	**Infinity**	0.7089	0.4191	0.4238	0.4221	0.4190	0.3795	0.4824	0.2679	0.3161	0.1674	0.5199
page-blocks-4	0.1653	**Infinity**	0.4503	0.2627	0.2486	0.2689	0.2589	0.2545	0.2639	0.1280	0.1462	0.0733	0.3510
shuttle-0–4	0.0180	**Infinity**	0.9107	1.2340	0.2001	0.2037	0.1698	0.1239	0.1138	0.4501	0.2472	0.1795	0.9115
vowel0	0.1706	0.6940	**0.8696**	0.2373	0.2858	0.2138	0.0939	0.2684	0.2541	0.0123	0.0843	0.1196	0.5206
wisconsin	0.5734	0.7553	**0.8761**	0.4395	0.6459	0.7059	0.5192	0.2186	0.4984	0.4074	0.4506	0.2405	0.6235
yeast-1_vs_7	0.1525	0.2618	**0.3192**	0.1385	0.1658	0.1743	0.0320	0.0830	0.0969	0.0165	0.1293	0.1293	0.2259
yeast-2_vs_4	0.2480	0.3105	**0.4262**	0.3055	0.2887	0.2359	0.1678	0.0897	0.1210	0.1019	0.1931	0.2446	0.3090
yeast3	0.1312	0.2687	**0.2720**	0.2066	0.2071	0.1577	0.0710	0.0735	0.0772	0.0659	0.1600	0.1873	0.2297
yeast4	0.1971	**Infinity**	0.2716	0.1578	0.1960	0.2108	0.1042	0.1091	0.1366	0.1237	0.1603	0.1937	0.2994
yeast5	0.3065	**Infinity**	0.5074	0.2818	0.3606	0.3262	0.1919	0.1765	0.2210	0.0436	0.1968	0.1910	0.3744
yeast-vs_7	0.0973	0.0574	**0.1398**	0.0880	0.0854	0.0974	0.0240	0.0605	0.0714	0.0896	0.0842	0.0717	0.0373
Avg. values	0.2135	**Infinity**	0.4555	0.2967	0.2565	0.2252	0.1629	0.1437	0.1760	0.1481	0.1512	0.1509	0.3526
Avg. ranks	6.8500	1.9500	**1.9000**	5.8500	5.7500	7.0500	10.1500	10.4500	8.8000	9.6000	9.7250	9.1250	3.8000

Bold results indicate better performance

other algorithms. Therefore, MOGPFEV also achieves good performance on imbalanced data sets, whereas other methods such as KSR are shown to decrease performance on such type of data. On the other hand, and similarly to balanced data, kernel PCA obtains the worst results.

Table 6 shows the C-index, Davies–Bouldin, and Dunn's results for imbalanced data sets. In this scenario, it is shown that visualization measures are significantly better for the proposal MOGPFEV, outperforming the results of KSR and MMGP. In comparison with the balanced data experiment, where MOGPFEV and KSR obtained similar results but in favor of KSR, results on imbalanced data are clearly in favor for MOGPFEV.

5.3 Statistical analysis and overall comparison

To evaluate whether there are significant differences in the results of the algorithms, the Friedman test is performed. This non-parametric test is applied to rank the algorithms over the data sets according to the χ^2 distribution. The results of the test when applied to the performance results, indicate significant differences in the performance of algorithms with p values lower than 0.05, i.e., having a statistical confidence higher than 95 %. Thereby, we apply the Wilcoxon test to evaluate where there exists such differences. The Wilcoxon test is a pairwise non-parametric test that detects significant differences between the performance of two algorithms. It computes two sums of ranks, $R+$ and $R-$ depending on the difference between two algorithms. If the results of the minimal of both rankings is below a certain critical value for a level of significance α, then the algorithms are significantly different. The use of the Wilcoxon test is recommended for pairwise comparisons in classification and evolutionary computation (Derrac et al. 2014; García et al. 2009, 2010; García and Herrera 2008).

Table 7 shows the results of the Wilcoxon rank-sum test to compute pairwise comparisons between MOGPFEV and the other methods. Results indicate the p values of the statistical test. Bolded results show significant differences in the results of the algorithms with p values lower than 0.05, i.e., significance levels higher than 95 %. MOGPFEV obtains statistically significant better results than the majority of algorithms, both on balanced and imbalanced data. KSR results on balanced data sets are statistically similar than those from MOGPFEV. However, their performance on imbalanced data is significantly different and MOGPFEV achieves best statistical confidences.

These results show the intrinsic multi-objective conflicting behavior of classification and visualization measures. Therefore, it is necessary to compute some overall results which show an overview and trade-off of the results from the six measures evaluated. Table 8 collects the ranks the algorithms for the different measures and data types considered. Therefore, the average rank and the meta-rank of the algorithms (the rank of the ranks) are computed and shown in the two bottom rows. MOGPFEV is shown to achieve the best average rank and the best meta-rank, followed distantly by KSR, which achieves the second best results, and MMGP, third best. This proves the best overall performance of MOGPFEV. On the other hand, kernel PCA and Random Projection obtain the worst results.

5.4 Convergence analysis and Pareto front distribution

MOGPFEV is a multi-objective genetic programming algorithm. Therefore, it is also necessary to analyze the fitness

Table 7 Wilcoxon test results

MOGPFEV vs	Original	KSR	LDA	LSDA	MDA	GHA	ICA	PCA	KPCA	RP	SOM	MMGP
Balanced data sets												
Accuracy	**0.0301**	0.0707	**7.40E−5**	**0.0025**	**5.09E−4**	**3.50E−5**	**7.42E−5**	**3.70E−5**	**1.05E−4**	**3.70E−5**	**4.91E−5**	**0.0011**
AUC	**0.0276**	≥0.2	**2.05E−4**	**0.0091**	**0.0022**	**1.46E−4**	**3.14E−4**	**1.66E−4**	**5.13E−4**	**8.61E−5**	**1.12E−4**	**0.0011**
Kappa	**0.0321**	0.0763	**7.43E−5**	**0.0015**	**4.47E−4**	**4.91E−5**	**8.63E−5**	**5.72E−5**	**1.46-4**	**5.72E−5**	**4.91E−5**	**1.99E−4**
C-index	**4.02E−4**	≥0.2	≥0.2	≥0.2	0.0691	**4.54E−4**	**8.61E−5**	**6.69E−4**	**0.0013**	**3.14E−4**	**6.53E−5**	**2.13E−4**
Davies–Bouldin	**7.50E−5**	≥0.2	**3.14E−4**	**0.0441**	**3.55E−4**	**2.44E−4**	**5.70E−5**	**8.61E−5**	**5.70E−5**	**7.50E−5**	**3.70E−5**	**4.76E−6**
Dunn	**0.0015**	≥0.2	**0.0238**	0.1098	0.1098	**0.0013**	**0.0019**	**0.0024**	**0.0123**	**7.48E−4**	**0.0031**	**1.86E−4**
Imbalanced data sets												
AUC	**0.0036**	**6.29E−5**	**9.53E−5**	**1.67E−4**	**5.34E−5**	**1.90E−6**	**1.90E−6**	**3.81E−6**	**1.90E−6**	**3.81E−6**	**9.53E−6**	**1.90E−6**
Kappa	**0.0192**	**6.29E−5**	**9.53E−5**	**1.33E−4**	**1.50E−4**	**1.90E−6**	**3.81E−6**	**3.81E−6**	**1.90E−6**	**1.90E−6**	**2.67E−5**	**1.90E−6**
C-index	**1.90E−6**	**0.0214**	**2.67E−5**	**4.82E−4**	**1.90E−6**	**1.90E−6**	**1.90E−6**	**1.90E−6**	**1.90E−5**	**1.90E−6**	**2.67E−5**	**1.90E−6**
Davies–Bouldin	**1.90E−4**	≥0.2	**1.90E−6**	**5.72E−6**	**1.33E−5**	**3.81E−6**	**1.90E−6**	**3.81E−6**	**1.90E−6**	**1.90E−6**	**1.90E−6**	**1.04E−4**
Dunn	**5.72E−6**	**0.0136**	**5.72E−6**	**1.33E−5**	**1.33E−5**	**1.90E−6**	**3.81E−6**	**3.81E−6**	**3.81E−6**	**3.81E−6**	**3.81E−6**	**8.05E−5**

Table 8 Meta-rank results (rank of the ranks)

Measure	Original	MOGPFEV	KSR	LDA	LSDA	MDA	GHA	ICA	PCA	KPCA	RP	SOM	MMGP
Balanced data sets													
Accuracy	4.4773	**2.0909**	4.4091	7.7955	5.6364	6.7500	9.4318	8.0455	8.4318	11.8636	9.7273	7.6591	4.6818
AUC	4.5000	**2.7500**	4.6818	7.7273	5.4545	6.4091	9.4545	7.8182	8.0227	10.9091	9.9773	7.5000	5.7955
Kappa	4.2955	**2.3182**	4.4091	7.9318	5.6591	6.8864	9.3182	7.8864	8.2273	11.4545	10.0455	7.7727	4.7955
C-index	9.2273	3.0227	**2.5909**	4.7955	3.3864	5.5227	8.1364	8.6364	8.1136	10.3182	10.0682	11.4545	5.7273
Davies–Bouldin	7.5909	**2.7273**	2.9091	6.2500	4.8864	6.2955	7.7045	8.1364	8.0909	9.8636	9.3182	11.9091	5.3182
Dunn	6.9773	4.0000	**3.2273**	7.7273	6.2955	6.6364	8.7273	8.8636	7.2727	10.3636	8.2500	8.4773	4.1818
Imbalanced data sets													
AUC	4.4500	**1.7750**	6.1750	5.9000	6.3500	4.6750	8.8750	8.4250	8.3250	11.6000	9.7500	8.0000	6.7000
Kappa	3.9500	**1.9000**	6.8500	6.4000	6.2000	5.0000	9.5000	8.2750	8.0500	11.3250	10.2000	7.8000	5.5500
C-index	8.3500	**1.8750**	2.5500	5.9750	3.9500	7.2500	9.9500	9.8000	9.0000	10.6000	10.1500	7.6000	3.9500
Davies–Bouldin	8.8500	1.8000	**1.7000**	4.9500	5.3500	6.8500	8.5000	8.1500	7.1000	11.1000	9.4500	12.5000	4.7000
Dunn	6.8500	1.9500	**1.9000**	5.8500	5.7500	7.0500	10.1500	10.4500	8.8000	9.6000	9.7250	9.1250	3.8000
Avg. ranks	6.3198	**2.3826**	3.7638	6.4820	5.3562	6.3023	9.0680	8.5897	8.1304	10.8180	9.6965	9.0725	5.0182
Meta rank	5.2727	**1.3636**	2.5455	6.0000	4.3182	5.3636	10.3636	9.8182	8.6364	12.4545	11.3636	9.2727	4.4273

Bold results indicate better performance

Table 9 Convergence results for the segment data set

Gen.	Acc.	AUC	Kappa	C-index	D–B.	Dunn	T(s)
0	0.9091	0.9835	0.8939	0.1679	3.3816	0.0266	0.0
25	0.9771	0.9961	0.9732	0.0894	1.1720	0.0078	24.7
50	0.9779	0.9963	0.9742	0.0514	1.4154	0.0717	49.6
75	0.9784	0.9963	0.9747	0.0395	1.1555	0.0769	74.5
100	0.9810	0.9968	0.9778	0.0366	1.0747	0.0734	99.5
125	0.9810	0.9968	0.9778	0.0373	1.1316	0.0768	124.5
150	0.9857	0.9976	0.9833	0.0387	1.1599	0.0871	149.4
175	0.9840	0.9973	0.9813	0.0235	0.9150	0.0443	174.3
200	0.9853	0.9975	0.9828	0.0362	1.0759	0.0782	199.2
225	0.9853	0.9975	0.9828	0.0339	1.0553	0.0661	224.2
250	0.9857	0.9976	0.9833	0.0338	1.0640	0.0664	249.2

evolution of the solutions along generations to observe improvements in the multiple quality measures. Table 9 shows the results of MOGPFEV on the segment data set, as a sample of the 42 data sets, along generations of the experiment run. Results at generation 0 show the classification and visualization measures using the initial population. Results quickly improve in early generations as the algorithm iterates to find better solutions using the genetic programming operators. The process converges to a stable solution which attempts to optimize and achieve a trade-off among all the objectives.

It is also interesting to analyze the distribution of the individuals in the Pareto front to see whether classification and visualization are conflicting objectives. Figure 6 shows the Pareto distribution on the segment data set by comparing the accuracy with each of the three visualization metrics. As observed, most of the individuals achieve high accuracy and they are distributed along the domain of the visualization metrics. There are also few individuals which obtain very good visualization values but their accuracy are significantly worse and unacceptable for classification.

Similarly, Table 10 shows the six best solutions for each objective on the segment data set, to analyze and compare how different are best solutions for individual objectives. It is observed that solutions #2 and #4 score best in three objectives. Comparing these two solutions, it is observed that #4 achieves better C-Index and Dunn values than #2, then this solution is preferred and selected.

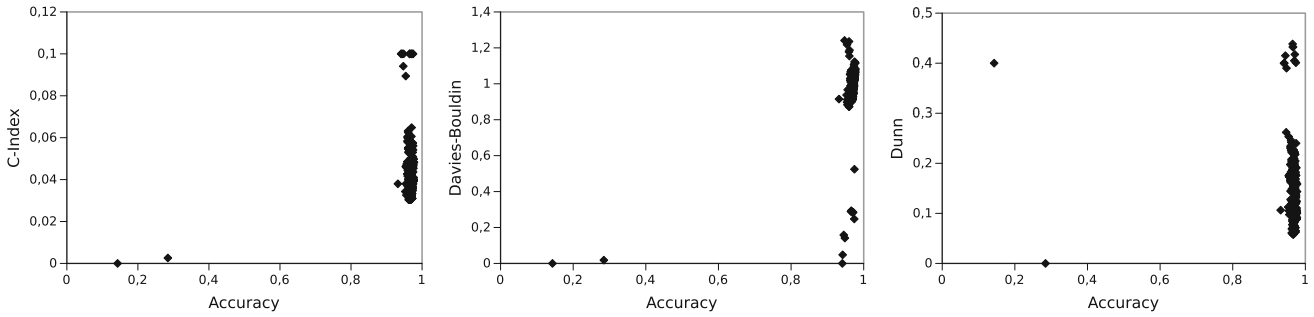

Fig. 6 Distribution of individuals in the Pareto front for the segment data set

Table 10 Comparison of best solutions for the segment data set

Solution	Accuracy	AUC	Kappa	C-index	D–B.	Dunn
1	0.1429	0.8571	0.0000	**0.0000**	**0.0000**	0.4000
2	**0.9771**	**0.9961**	**0.9732**	0.0483	1.0832	0.1435
3	0.9411	0.9896	0.9313	0.1000	**0.0000**	0.4000
4	**0.9771**	**0.9961**	**0.9732**	0.0482	1.1158	0.1586
5	0.9649	0.9941	0.9591	0.1000	0.2892	**0.4384**
6	0.9597	0.9931	0.9530	**0.0000**	0.1064	0.1659

Bold results indicate better performance

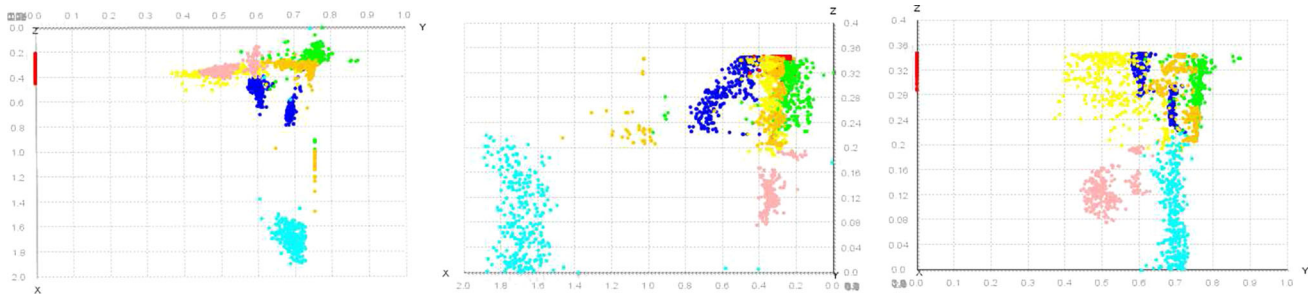

Fig. 7 Sample 2D visualization of segment data set

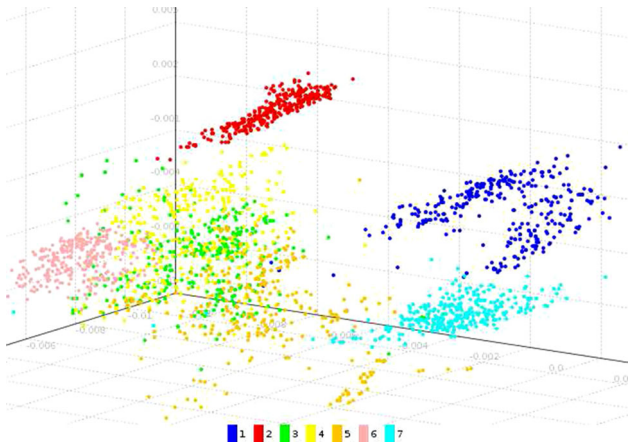

Fig. 8 Sample 3D visualization of segment data set

5.5 Visualization results

3D transformations are difficult to represent in this document. Therefore, user interactive 3D browsing of data sets transformations and projections is available at the website http://www.uco.es/grupos/kdis/wiki/MOGPFEV.

The reader is referred to this link to visualize data transformations for all the algorithms and data sets. Figure 7 shows a sample of the visualization of the segment data set (14 attributes) transformed into three dimensions, using 2D projections partial views. Figure 8 shows the 3D representation as available in the website, the user can interactively rotate the data visualization to see the accurate discrimination on the subspace of features.

MOGPFEV and KSR demonstrated best performance on the experimental study and both show an interesting behavior when visualizing the data sets projections. For some data sets, these methods find more optimum transformations on 1D and 2D instead of the allowed 3D subspace. Specifically, MOGPFEV identifies four data sets in which the 2D projection is preferred rather than the 3D (abalone19, glass4, shuttle-0–4, and wisconsin). On the other hand, KSR makes 3D projections for nine data sets, 2D projections for two data sets, and 1D projections for 29 data sets. According to equations for visualization measures in Sect. 3.4, inter-class distances are to be maximized and intra-class distances are to be minimized. In other words, distances between instances from the same class are desired to minimize in the new subspace, whereas distances between instances from different classes are desired to be maximized. This improves the discriminability between classes in the new features space. However, distances on 2D spaces are a priori shorter than on 3D spaces. Thus, given equally accurate classifications, 2D spaces may gather instances closer, and therefore visualization measures are valued better.

This behavior is feasible for MOGPFEV since the context-free grammar used for individual representation allows that a dimension can be a constant value, or indirectly, the result of the mathematical expression can be also a constant value. In such scenarios, the algorithm selected the solutions with 2D projections rather than the 3D projections because their fitness values (measures of the six objectives to evaluate) were strictly better. Nevertheless, the grammar can be easily edited by the user to change or remove this behavior, depending on the desired projections. This provides high flexibility to the algorithm to generate data transformations with any number of dimensions and any type of mathematical expressions, attending to user requirements and limitations.

6 Conclusion

In this paper we introduced a Pareto-based multi-objective genetic programming algorithm for feature extraction and data visualization. The algorithm considered the intrinsic multi-objective nature of this task to provide good transformations from high-dimensional data to lower dimensional subspaces of features. The multi-objective fitness function was designed to consider six different classification and visualization measures as objectives of the evolutionary algorithm. The NSGA-II evolutionary process conducted the evolution of the population to obtain solutions which achieve good Pareto front and trade-off among the quality of the multiple conflicting objectives. Experiments carried out over 22 balanced and 20 imbalanced data sets demonstrated the good performance of the algorithm in terms of obtaining accurate classification and good visualization measures on the projected subspace of new features. Experimental results show the flexible representation of the data transformation and the good performance achieved. The results were validated using non-parametric statistical tests, which support better performance of the algorithm than other ten algorithms used in comparisons. The convergence and the Pareto front distribution among the objectives were also analyzed.

Acknowledgements This work has been supported by the National Institutes of Health Grant 1R01HD056235-01A1 (KJC), the Spanish Ministry of Economy and Competitiveness Project TIN2014-55252-P and FEDER funds (SV).

Compliance with ethical standards

Conflict of interest The authors declare that there is no conflict of interest regarding the publication of this paper.

References

Alcalá R, Alcalá-Fdez J, Gacto MJ, Herrera F (2008) On the use of multiobjective genetic algorithms to improve the accuracy-interpretability trade-off of fuzzy rule-based systems. In: Multi-objective evolutionary algorithms for knowledge discovery from data bases, vol 98. Springer, New York, pp 91–107

Alcalá-Fdez J, Fernández A, Luengo J, Derrac J, García S, Sánchez L, Herrera F (2011) KEEL data-mining software tool: data set repository, integration of algorithms and experimental analysis framework. Anal Framew J Mult-Valued Log Soft Comput 17:255–287

Bae SH, Choi JY, Qiu J, Fox GC (2010) Dimension reduction and visualization of large high-dimensional data via interpolation. In: Proceedings of the 19th ACM international symposium on high performance distributed computing, pp 203–214

Ben-David A (2008) About the relationship between ROC curves and Cohen's kappa. Eng Appl Artif Intell 21(6):874–882

Bertini E, Tatu A, Keim D (2011) Quality metrics in high-dimensional data visualization: an overview and systematization. IEEE Trans Vis Comput Graph 17(12):2203–2212

Bezdek JC, Pal NR (1998) Some new indexes of cluster validity. IEEE Trans Syst Man Cybern Part B Cybern 28(3):301–315

Biber D (1992) The multi-dimensional approach to linguistic analyses of genre variation: an overview of methodology and findings. Comput Humanit 26(5):331–345

Borg I, Groenen PJF (2005) Modern multidimensional scaling: theory and applications. In: Springer series in statistics. Springer, New York

Cai D (2012) Matlab codes for dimensionality reduction (subspace learning). http://www.cad.zju.edu.cn/home/dengcai/Data/DimensionReduction.html

Cai D, He X, Han J (2007a) Spectral regression for efficient regularized subspace learning. In: Proceedings of the IEEE international conference on computer vision, pp 1–8

Cai D, He X, Zhou K, Han J, Bao H (2007b) Locality sensitive discriminant analysis. In: Proceedings of the international joint conference on artificial intelligence, pp 1713–1726

Cano A, Ventura S (2014) Gpu-parallel subtree interpreter for genetic programming. In: Proceedings of the conference on genetic and evolutionary computation, pp 887–894

Cano A, Zafra A, Ventura S (2012) Speeding up the evaluation phase of GP classification algorithms on GPUs. Soft Comput 16(2):187–202

Cano A, Zafra A, Ventura S (2015a) Speeding up multiple instance learning classification rules on GPUs. Knowl Inf Syst 44(1):127–145

Cano A, Luna JM, Zafra A, Ventura S (2015b) A classification module for genetic programming algorithms in JCLEC. J Mach Learn Res 16:491–494

Comon P (1994) Independent component analysis, a new concept? Signal Process 36(3):287–314

Davies DL, Bouldin DW (1979) A cluster separation measure. IEEE Trans Pattern Anal Mach Intell 1(2):224–227

Deb K, Pratap A, Agarwal S, Meyarivan T (2002) A fast elitist multiobjective genetic algorithm: NSGA-II. IEEE Trans Evol Comput 6(2):182–197

Derrac J, García S, Hui S, Nagaratnam Suganthan P, Herrera F (2014) Analyzing convergence performance of evolutionary algorithms: a statistical approach. Inf Sci 289:41–58

Dhir CS, Lee J, Lee SY (2012) Extraction of independent discriminant features for data with asymmetric distribution. Knowl Inf Syst 30(2):359–375

Espejo PG, Ventura S, Herrera F (2010) A survey on the application of genetic programming to classification. IEEE Trans Syst Man Cybern Part C (Appl Rev) 40(2):121–144

Fayyad U, Grinstein GG, Wierse A (2001) Information visualization in data mining and knowledge discovery. Morgan Kaufmann, San Francisco

Fernández A, González AM, Díaz J, Dorronsoro JR (2015) Diffusion maps for dimensionality reduction and visualization of meteorological data. Neurocomputing 163:25–37

Fernández-Blanco E, Rivero D, Gestal M, Dorado J (2013) Classification of signals by means of genetic programming. Soft Comput 17(10):1929–1937

Ferreira de Oliveira MC, Levkowitz H (2003) From visual data exploration to visual data mining: a survey. IEEE Trans Vis Comput Graph 9(3):378–394

Ferri C, Hernandez-Orallo J, Modroiu R (2009) An experimental comparison of performance measures for classification. Pattern Recognit Lett 30(1):27–38

Fisher RA (1936) The use of multiple measurements in taxonomic problems. Ann Eugen 7:179–188

Fradkin D, Madigan D (2003) Experiments with random projections for machine learning. In: Proceedings of the SIGKDD international conference on knowledge discovery and data mining, pp 517–522

García S, Herrera F (2008) An extension on "statistical comparisons of classifiers over multiple data sets" for all pairwise comparisons. J Mach Learn Res 9:2677–2694

García S, Molina D, Lozano M, Herrera F (2009) Study on the use of non-parametric tests for analyzing the evolutionary algorithms' behaviour: a case study. J Heuristics 15:617–644

García S, Fernández A, Luengo J, Herrera F (2010) Advanced nonparametric tests for multiple comparisons in the design of experiments in computational intelligence and data mining: experimental analysis of power. Inf Sci 180(10):2044–2064

Gisbrecht A, Hammer B (2015) Data visualization by nonlinear dimensionality reduction. Wiley Interdiscip Rev Data Min Knowl Discov 5(2):51–73

Guo H, Jack LB, Nandi AK (2005) Feature generation using genetic programming with application to fault classification. IEEE Trans Syst Man Cybern Part B Cybern 35(1):89–99

Guyon I, Gunn S, Nikravesh M, Zadeh LA (2006) Feature extraction: foundations and applications. In: Studies in fuzziness and soft computing. Springer, New York

He H, Garcia EA (2009) Learning from imbalanced data. IEEE Trans Knowl Data Eng 21(9):1263–1284

Huang J, Ling CX (2005) Using AUC and accuracy in evaluating learning algorithms. IEEE Trans Knowl Data Eng 17(3):299–310

Hubert LJ, Levin JR (1976) A general statistical framework for assessing categorical clustering in free recall. Psychol Bull 78(6):1072–1080

Icke I, Rosenberg A (2011) Multi-objective genetic programming for visual analytics. In: Proceedings of the European conference on genetic programming, pp 322–334

Kohonen T (1982) Self-organized formation of topologically correct feature maps. Biol Cybern 43(1):59–69

Krawiec K (2002) Genetic programming-based construction of features for machine learning and knowledge discovery tasks. Genet Program Evol Mach 3:329–343

Lee JA, Verleysen M (2010) Unsupervised dimensionality reduction: overview and recent advances. In: Proceedings of the IJCNN IEEE world congress on computational intelligence, pp 4163–4170

Liu H, Motoda H (1998) Feature extraction, construction and selection: a data mining perspective. Kluwer Academic Publishers, Norwell

Liu B, Xiao Y, Yu PS, Hao Z, Cao L (2014) An efficient orientation distance-based discriminative feature extraction method for multiclassification. Knowl Inf Syst 39(2):409–433

López V, Fernández A, García S, Palade V, Herrera F (2013) An insight into classification with imbalanced data: empirical results and current trends on using data intrinsic characteristics. Inf Sci 250:113–141

Mckay RI, Hoai NX, Whigham PA, Shan Y, O'Neill M (2010) Grammar-based genetic programming: a survey. Genet Program Evol Mach 11(3–4):365–396

Mukhopadhyay A, Maulik U, Bandyopadhyay S, Coello CA (2014) A survey of multiobjective evolutionary algorithms for data mining: part I. IEEE Trans Evol Comput 18(1):4–19

Neshatian K, Zhang M, Johnston M (2007) Feature construction and dimension reduction using genetic programming. In: Orgun MA, Thornton J (eds) AI 2007: advances in artificial intelligence. Lecture notes in computer science, vol 4830, pp 160–170

Pearson K (1901) On lines and planes of closest fit to systems of points in space. Philos Mag 2(6):559–572

Sammon JW (1969) A nonlinear mapping for data structure analysis. IEEE Trans Comput 18(5):401–409

Sanger TD (1989) Optimal unsupervised learning in a single-layer linear feedforward neural network. Neural Netw 2(6):459–473

Schölkopf B, Smola A, Müller KR (1998) Nonlinear component analysis as a kernel eigenvalue problem. Neural Comput 10(5):1299–1319

van der Maaten L, Postma EO, van den Herik HJ (2009) Dimensionality reduction: a comparative review. Technical report, Tilburg University Technical Report, TiCC-TR 2009–005

Venna J, Peltonen J, Nybo K, Aidos H, Kaski S (2010) Information retrieval perspective to nonlinear dimensionality reduction for data visualization. J Mach Learn Res 11:451–490

Verleysen M, Franois D (2005) The curse of dimensionality in data mining and time series prediction. In: Cabestany J, Prieto A, Sandoval F (eds) Computational intelligence and bioinspired systems. Lecture notes in computer science, vol 3512. Springer, Berlin, pp 758–770

Wilcoxon F (1945) Individual comparisons by ranking methods. Biom Bull 1(6):80–83

Yeh TT, Chen TY, Chen YC, Wei HW (2011) Parallel non-linear dimension reduction algorithm on GPU. Int J GranuL Comput Rough Sets Intell Syst 2(2):149–165

Zhang Y, Rockett PI (2006) Feature extraction using multi-objective genetic programming. In: Jin Y (ed) Multi-objective machine learning. Studies in computational intelligence, vol 16, chapter 4. Springer, New York, pp 79–106

Zhang Y, Rockett PI (2007) Multiobjective genetic programming feature extraction with optimized dimensionality. In: Soft computing

in industrial applications. Advances in soft computing, vol 39. Springer, New York, pp 159–168

Zhang Y, Rockett PI (2009) A generic multi-dimensional feature extraction method using multiobjective genetic programming. Evol Comput 17(1):89–115

Zhang Y, Rockett PI (2010) Domain-independent feature extraction for multi-classification using multi-objective genetic programming. Pattern Anal Appl 13:273–288