Expert Systems with Applications 42 (2015) 833–843

Contents lists available at ScienceDirect

Expert Systems with Applications

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

Pattern generation for multi-class LAD using iterative genetic algorithm with flexible chromosomes and multiple populations

Hwang Ho Kim, Jin Young Choi *

Department of Industrial Engineering, Ajou University, Suwon, Republic of Korea

ARTICLE INFO

Article history:
Available online 6 September 2014

Keywords:
Multi-class classification
Logical analysis of data
Decomposition
Genetic algorithm
Classification accuracy

ABSTRACT

In this paper, we consider a pattern generation method for multi-class classification using logical analysis of data (LAD). Specifically, we apply two decomposition approaches—one versus all, and one versus one—to multi-class classification problems, and develop an efficient iterative genetic algorithm with flexible chromosomes and multiple populations (IGA-FCMP). The suggested algorithm has two control parameters for improving the classification accuracy of the generated patterns: (i) the number of patterns to select at the termination of the genetic procedure; and (ii) the number of times that an observation is covered by some patterns until it is omitted from further consideration. By using six well-known datasets available from the UCI machine-learning repository, we performed a numerical experiment to show the superiority of the IGA-FCMP over existing multi-class LAD and other supervised learning algorithms, in terms of the classification accuracy.

1. Introduction

Supervised learning is a machine-learning task for gaining information regarding data characteristics such as class or quality by learning from training data. Supervised learning analyzes training data and produces a model or function that can be used to give the correct result when new data (so-called test data) is given as input, without knowing the target *a priori* (Bishop, 2006; Jiawei & Kamber, 2001; Lugosi, 2002; Mitchell, 1997). Many algorithms using supervised learning have been developed and applied to classification problems in novel instances. These can be divided into binary classification and multi-class classification, depending on the number of classes K in a dataset (Berry & Linoff, 1997; Gehler & Nowozin, 2009).

For binary classification problems with $K = 2$, various classification methods have been suggested, such as decision trees (J48) (Safavian & Landgrebe, 1991), support vector machines (SVMs) (Schölkopf & Smola, 2002; Furey et al., 2000), and neural networks (NNs) (Hagan, Demuth, & Beale, 1996). Further, in the literature cited, their performance was evaluated through numerical experiments. However, it is known that they have some limitations regarding their ability to explain the causes of the classification results. To rectify this, an efficient method exhibiting high accuracy and explanatory power, called logical analysis of data (LAD), has recently been proposed and used in the medical, service, and manufacturing fields, among others (Hammer, Kogan, & Lejeune, 2012). LAD is an efficient data analysis methodology based on patterns containing hidden structural information of binary training data (Boros, Hammer, Ibaraki, & Kogan, 1997; Boros et al., 2000; Crama, Hammer, & Ibaraki, 1988; Hammer & Bonates, 2006; Han, Kim, Jeong, & Yum, 2011).

Meanwhile, for multi-class classification problems with $K \geqslant 3$, various algorithms have been provided, and we can categorize them in two ways: (i) the direct approach, and (ii) the decomposition approach. The direct approach deals with K classes directly by finding classifiers for K classes simultaneously, including k-nearest neighbors, naïve Bayes, and decision trees (Aly, 2005). However, there are disadvantages because its performance is influenced by noise data and suffers from high computational complexity in large datasets with many instances and classes. By contrast, the decomposition approach divides the original problem into several binary classification problems. These can be solved efficiently using binary classifiers (Tax & Duin, 2002). The decomposition approach follows two basic strategies: One versus All (OvA) and One versus One (OvO) (Galar, Fernández, Barrenechea, Bustince, & Herrera, 2011; Mayoraz & Alpaydin, 1999; Tax & Duin, 2002).

OvA decomposition splits K into two classes—dividing the class under consideration from the remaining $(K-1)$ classes—eventually building K different binary classification problems. Then, for each problem, we find one classifier. For example, if we find the classifier f_1 from a dataset with four classes, as in Fig. 1, we let all instances in

* Corresponding author.
E-mail address: choijy@ajou.ac.kr (J.Y. Choi).

http://dx.doi.org/10.1016/j.eswa.2014.08.050
0957-4174/

Class I stand as the positive set, and all instances in the remaining classes (II, III, and IV) stand as the negative set, as in Fig. 2. Then, by combining all classifiers $f_i(i = 1, 2, 3, 4)$, we derive the function

$$f(x) = \arg\max_i f_i(x), \tag{1}$$

to classify test data x as class i, with the index giving the largest value f_i, $\forall i$ (Tax & Duin, 2002).

The alternative strategy, OvO, builds $K(K-1)$ binary classification problems by considering all possible class pairs. With OvO, we find one classifier for each problem. For example, if we find the classifier f_{12} to identify Class I over Class II from a dataset with four classes, as in Fig. 1, we let Class I stand as the positive set and Class II as the negative set, as in Fig. 3. Then, by combining all classifiers $f_{ij}(i = 1, 2, 3, 4, j \neq i)$, we derive the function

$$f(x) = \arg\max_i \left(\sum_{j \neq i} f_{ij}(x) \right), \tag{2}$$

to classify test data x as class i, corresponding to the index giving the largest value $\sum_{j \neq i} f_{ij}$, $\forall i$ (Tax & Duin, 2002).

In a similar effort to improve classification accuracy and explanatory power, LAD for multi-class classification has also been proposed (Moreira, 2000; Mortada, Yacout, & Lakis, 2013; Herrera & Subasi, 2013). We can divide multi-class LAD into OvA and OvO as well, depending on the decomposition type. Moreira (2000) and Mortada (2010) studied OvO-type LAD algorithms for multi-class classification problems. Moreira (2000) proposed two methods for breaking down a multi-class classification problem into binary-class problems using OvO. The first method, called a multi-layered 2-class LAD, used the typical OvO approach, which does not require any modification of the structure of the original LAD algorithm. The second method modified the architecture of the pattern generation and theory formation steps in the original LAD algorithm, and introduced the decomposition matrix used for recording multi-class patterns generated through iterations of LAD. In comparing these two methods, the second approach is less accurate, but more intuitive. Moreover, because both are enumeration-based techniques, combining and enumerating random binary attributes (called "literals") as a way to find all patterns in the solution space, they suffer from high computational complexity and lengthy execution times.

To improve the classification accuracy of the second OvO-type method, Mortada (2010) proposed a multi-class LAD algorithm, integrating the second approach by Moreira (2000) with a pattern generation method using the mixed 0–1 integer and linear programming (MILP) model by Ryoo and Jang (2009). Mortada (2010) showed that the classification accuracy of the proposed method is higher than the LAD models proposed by Moreira (2000). Furthermore, Herrera and Subasi (2013) proposed an algorithmic approach using MILP to build OvA-type LAD patterns in a multi-class dataset efficiently. However, its classification accuracy is worse than with OvO (Tax & Duin, 2002). This is because the negative set using OvA is the mixed set of class-specific properties for all the remaining classes, making it difficult to represent the characteristics pertaining exclusively to the positive set.

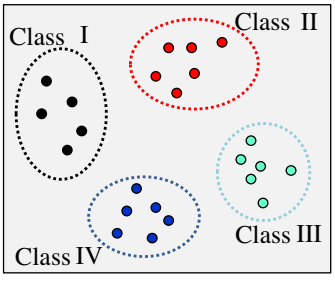

Fig. 1. Example of a dataset containing four classes.

Overall, these MILP-based approaches face three challenges. First, such approaches generate only a single pattern per iteration by finding an optimal solution to the considered MILP, and then delete some data covered by the generated pattern. In generating patterns, the next iteration is applied to the remaining dataset until the data is removed. Using this procedure, we observe that deleting data covered by one pattern can affect the generation of other patterns in later iterations. In other words, were it not removed, the deleted data might have been useful for providing relevant information for identifying subsequent patterns. Second, if most generated patterns cover only a slight amount of data, it will take extensive time to find the multi-class LAD patterns in large-sized data. In the worst case, if each pattern covers only one data item, it would need to be repeated for each observation. Finally, it would take too much time and effort to develop and solve MILP models for large-sized data.

Motivated by these observations, we propose a new pattern generation algorithm for multi-class LAD, namely the "iterative genetic algorithm with flexible chromosomes and multiple populations" (IGA-FCMP). This paper is devoted to explaining our proposal. The algorithm is designed to improve the performance of generated patterns by using the following control parameters: (i) the number of patterns to select at the termination of a genetic procedure; and (ii) the number of times that an observation is covered by some patterns before it is deleted. More specifically, we designed two types of IGA-FCMP using OvA- and OvO-type decompositions for multi-class LAD, and we shall demonstrate the superiority of their performance over existing multi-class LAD, including MILP-based LAD, and other supervised learning algorithms. We shall do so with a numerical experiment showing that our algorithm increases the efficiency of OvO and the accuracy of OvA simultaneously.

The rest of this paper is organized as follows. Section 2 introduces the notations and basic concept behind the LAD framework. Section 3 suggests a new pattern generation method for multi-class LAD using an iterative genetic algorithm with special structures, especially with OvA- and OvO-type decomposition methods. We explain how we tested the performance of the proposed algorithm with a numerical experiment in Section 4. Finally, in Section 5, we discuss the direction of future work in this area.

2. Basic concept of LAD

In this section, we explain the notations and summarize the basic idea behind the LAD framework. There are four steps (see Fig. 4): data binarization, support-set generation, pattern generation, and theory formulation (Boros et al., 2000).

2.1. Data binarization

All the input data for the LAD framework must be binary. Thus, binarization is usually performed as the first step of LAD. Non-binary input variables require a data conversion process. In LAD, we use several cut points (cp) for the binarization of numerical data. In general, cp is defined as the median value of the observation O^+ from the positive set Ω^+ and the observation o^- from the negative set Ω^-. For each cp, we define a binary attribute and assign its value as follows. If the value of an observation is larger than cp, the observation is 1. Otherwise, it is 0. We represent them by using a literal, either a Boolean variable b or its complement. A term t is a conjunction of literals such as $t = b_1 b_2 \bar{b}_3 \bar{b}_4$. The degree G_t of a term t is the number of literals in it.

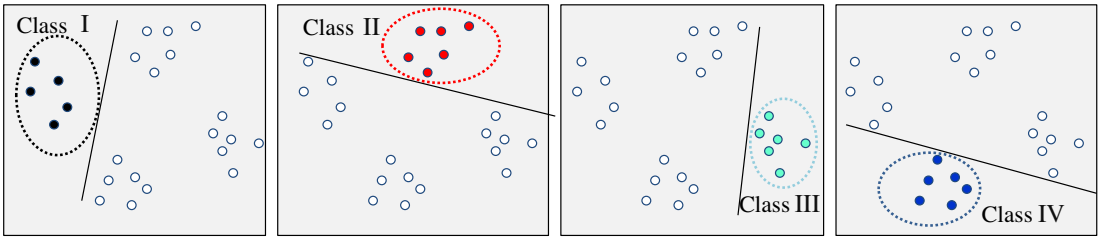

Fig. 2. One versus All (OvA) approach.

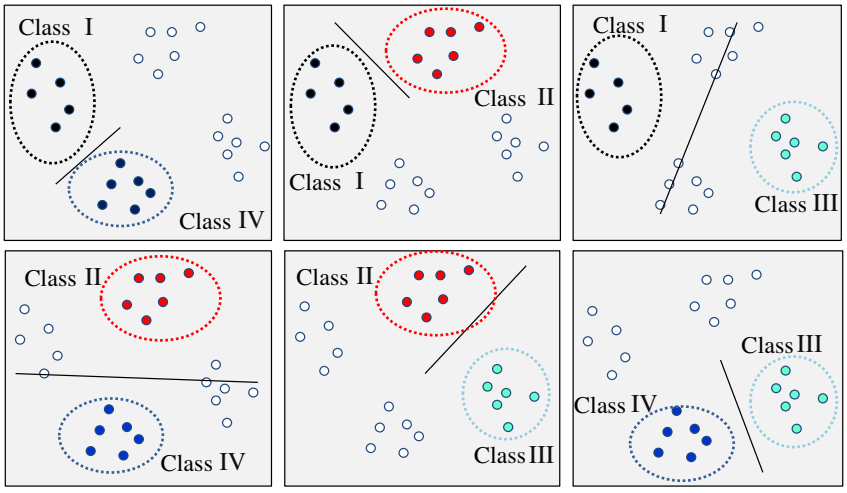

Fig. 3. One versus One (OvO) approach.

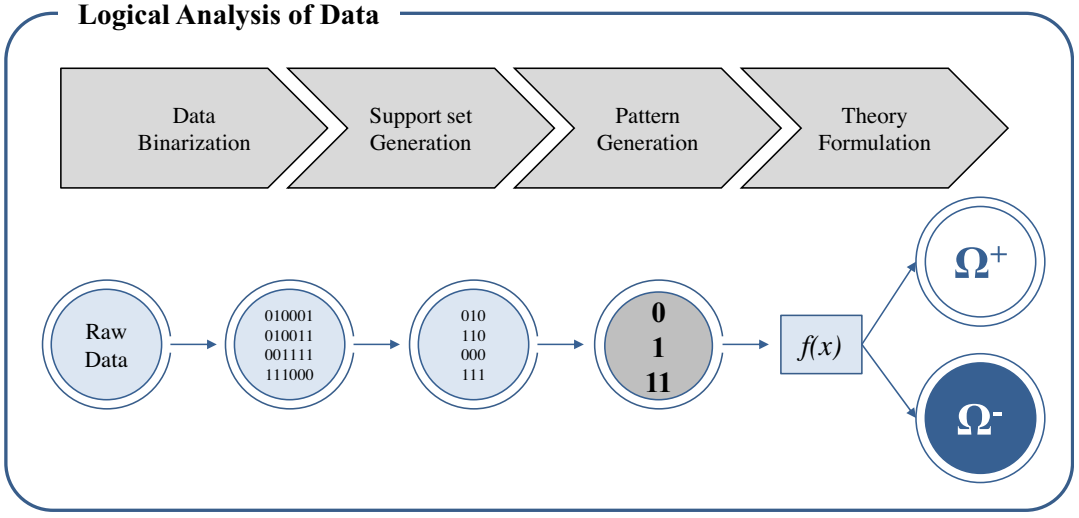

Fig. 4. LAD framework.

2.2. Support-set generation

After binarization, the dimension of the binarized dataset will be increased when one attribute is divided into several binary attributes. In order to reduce the complexity of the LAD pattern generation problem, we generate a support-set after data binarization. The support-set is defined as the minimal subset of the binary attributes necessary for distinguishing the observation O^+ in the positive set Ω^+ from the observation O^- in the negative set Ω^-. In the LAD framework, generating a support-set can be formulated as the minimum set covering problem (SCP), which is NP-complete (Hammer & Bonates, 2006).

2.3. Pattern generation

Pattern generation in the LAD framework is the step required to find the structural characteristics used for distinguishing observations in the positive class from those in the negative class (and vice

versa). This step is the most decisive in determining the classification accuracy of LAD (Alexe, Alexe, Bonates, & Kogan, 2007; Han, Kim, Yum, & Jeong, 2011). Some definitions to characterize a pattern are needed: A term t is said to "cover" an observation if the same shape as term t is included in an observation. The positive (or negative) pattern is a term t that covers at least one positive (or negative) observation and does not cover any negative (or positive) observation.

2.4. Theory formulation

In the final step for the LAD framework, generated patterns are employed to form a classification model, known as a "theory." The theory plays the role of a classifier as the weighted linear combination of positive and negative patterns. In other words, a theory is a binary classification model for test data, and the performance of patterns can be determined by calculating the classification accuracy of the theory. In classifying test data x, the theory is derived by

$$\Delta^*(x) = \sum_{p_i^+ \in p^+} w_i p_i^+(x) - \sum_{p_j^- \in p^-} w_j p_j^-(x) \tag{3}$$

where p_i^+ and p_j^- are patterns in positive pattern set p^+ and negative pattern set p^-, respectively. Furthermore, $p_i^+(x) = 1$ if observation x is covered by positive pattern p_i^+, and $p_i^+(x) = 0$, otherwise. The values w_i and w_j represent the weights for positive and negative patterns, respectively (Boros et al., 2000). When classifying test data x using Eq. (3), x is classified as a positive class if $\Delta^*(x) > 0$ and a negative class if $\Delta^*(x) < 0$. If $\Delta^*(x) = 0$, test data x is left unclassified.

2.5. An example

Suppose that we have a dataset with three numerical attributes as in Table 1. The first step of LAD is to perform binarization by defining cut points. For instance, four values for attribute AA are arranged in non-descending order, differentiating sets as [0.8], [2.1], [2.4], [3.3]. Then, we can define three cut points by computing the median values between sets as $cp_1 = 1.45$, $cp_2 = 2.25$, $cp_3 = 2.85$. Similarly, for attribute BB and CC, we have one and two cut points as $cp_4 = 8.2$ and $cp_5 = 6.2$, $cp_6 = 9.0$, respectively. By defining one binary attribute y_i for each cut point cp_i, we can produce a new binary dataset by evaluating each observation as in Table 2: For $O^{\cdot} \in \Omega^{\cdot}$, ($\cdot \in \{+, -\}$), $y_i(O^{\cdot}) = 1$ if $O^{\cdot} > cp_i$, and $y_i(O^{\cdot}) = 0$, otherwise. For example, $y_1(0.8) = 0$ since $cp_1 = 1.45 > 0.8$, and $y_1(2.4) = 1$ since $cp_1 = 1.45 < 2.4$.

The second step is to generate a support-set, defined as the minimal subset of y_i's necessary for distinguishing the observation O^+ in the positive set Ω^+ from the observation O^- in the negative set Ω^-. We can model it as the minimum SCP as in Eq. (4). Each constraint is generated using each pair of observations in positive and negative sets. More precisely, for the jth pair of observations, we construct a constraint $\sum_{i=1}^{6} a_{ij} y_i \geqslant 1$ by comparing y_i's. We assign $a_{ij} = 1$ if the values of y_i 's are different, and $a_{ij} = 0$, otherwise. By solving the SCP, we can obtain a support-set as in Table 3.

Table 1
An example dataset.

Attributes	AA	BB	CC
Positive set (Ω^+)	0.8	10.3	6.7
	2.4	11.8	8.3
Negative set (Ω^-)	3.3	3.6	5.5
	2.1	6.1	9.7

Table 2
Binarized dataset.

Attributes	y_1	y_2	y_3	y_4	y_5	y_6
Positive set (Ω^+)	0	0	0	1	1	0
	1	1	0	1	1	0
Negative set (Ω^-)	1	1	1	0	0	0
	1	0	0	0	1	1

Table 3
A support-set generated.

Attributes	y_1	y_2	y_3
Positive set (Ω^+)	0	0	0
	1	1	0
Negative set (Ω^-)	1	1	1
	1	0	0

$$\begin{aligned} &\textit{Min} && y_1 + y_2 + y_3 + y_4 + y_5 + y_6 \\ &\textit{subject to} && y_1 + y_2 + y_3 + y_4 + y_5 \geqslant 1 \\ &&& y_1 + y_4 + y_6 \geqslant 1 \\ &&& y_3 + y_4 + y_5 \geqslant 1 \\ &&& y_2 + y_4 + y_6 \geqslant 1 \\ &&& y_1, y_2, y_3, y_4, y_5, y_6 \in \{0, 1\} \end{aligned} \tag{4}$$

The third step is to generate patterns. For example, we can simply identify positive patterns $p_1^+ = 0_{(1)}$, $p_2^+ = 0_{(1)}0_{(2)}$, and negative patterns $p_1^- = 1_{(2)}1_{(3)}$, $p_2^- = 1_{(1)}0_{(2)}0_{(3)}$, where $p_1^+ = 0_{(1)}$ represents a term with 0 in the first position. As the final step, we can establish a theory as $\Delta^*(x) = p_1^+(x) + p_2^+(x) - p_1^-(x) - p_2^-(x)$ and use it for data classification. For example, if $x = 1_{(1)}1_{(2)}1_{(3)}$, it is classified as in the negative class since $p_1^+(x) = 0, p_2^+(x) = 0, p_1^-(x) = 1, p_2^-(x) = 0$, resulting in $\Delta^*(x) = -1$.

3. Design of IGA-FCMP for multi-class LAD

In this section, we propose a new pattern generation method for multi-class LAD. We have named it the iterative genetic algorithm with flexible chromosomes and multiple populations (IGA-FCMP). Specifically, we have designed pattern generation methods using two decomposition types, OvO and OvA, by following the four steps in the LAD framework described in Section 2.

3.1. Data binarization, support-set generation, and decomposition for multi-class LAD

For data binarization, we used the Fayyad–Irani Discretization method, a supervised hierarchical split method. This method uses the class information entropy of candidate partitions to select boundaries for discretization (Kotsiantis & Kanellopoulos, 2006; Boros & Menkov, 2004). Class information entropy is a measure of purity, quantifying the amount of information necessary for specifying which class an instance belongs to. It considers one large interval containing all the known values of a feature and then recursively partitions this interval into smaller subintervals until an optimal number of intervals is achieved (Mitov et al., 2009).

After data binarization, we performed support-set generation using an efficient heuristic algorithm based on the Best First Search (BFS) method. This step reduces the number of increased binary data attributes, and the complexity of the pattern generation problem. For these two initial steps in the LAD framework, we used WEKA 3.6.10, a freely available and popular suite in machine-learning software written in Java (Witten & Frank, 2005).

Next, we decomposed the dataset into either a positive set or a negative set depending on the decomposition type under

consideration. For OvA, we had K pairs of positive and negative sets $(\Omega_i^+, \Omega_i^-),\ \forall i = 1, 2, \ldots, K$, where Ω_i^+ and Ω_i^- correspond to the sets of observations in C_i and $\bigcup_{j \neq i} C_j$, respectively. For OvO, we had $K(K-1)$ pairs of positive and negative sets $(\Omega_{ij}^+, \Omega_{ij}^-),\ \forall i = 1, 2, \ldots, K(K-1), j \neq i$, where Ω_{ij}^+ and Ω_{ij}^- correspond to the sets of observations in C_i and $C_j (j \neq i)$, respectively. However, in our implementation—for efficient memory allocation— we did not assign computer memory space physically for those positive and negative sets, whereas we handled them logically by just choosing the corresponding subsets as necessary.

3.2. Pattern generation for multi-class LAD using IGA-FCMP

The next step is to generate multi-class LAD patterns and it is based on our previous work on binary classification using LAD in Kim and Choi (2013a, 2013b). More precisely, Kim and Choi (2013a) suggested a tentative result showing that a basic GA using a fixed-sized chromosome and general two-point crossover operation works well to binary classification problem. Then, Kim and Choi (2013b) tried to improve the performance of it by introducing the concept of iterative operation, while choosing one pattern per iteration of GA. In this work, we extended these results to develop pattern generation algorithm for multi-class LAD using GA. We summarize the main components of the IGA-FCMP in what follows.

3.2.1. Design of a chromosome for representing a pattern

We suggested a variable-sized chromosome to efficiently represent patterns with different lengths. This allows us to handle sizable data efficiently while avoiding any unnecessary memory allocation. A chromosome is defined as a 2-tuple $C = (C_{pos}, C_{lit})$ with integer-valued strings such that $C_{pos} = (C_{pos}[1], C_{pos}[2], \ldots, C_{pos}[l])$ and $C_{lit} = (C_{lit}[1], C_{lit}[2], \ldots, C_{lit}[l])$, where l is the length of the pattern t considered ($G_t = l$), C_{pos} is the sub-chromosome for representing the positions of selected attributes, C_{lit} is the sub-chromosome for representing literal values selected, and $C_{pos}[i]$ and $C_{lit}[i]$ in C_{pos} and C_{lit} are the attribute's number and its value, selected at the i th order, respectively. Fig. 5 is an example of a variable-sized chromosome with length 5. The values in C_{pos} do not need to be sequential, and their values 0, 0, 1, 1, and 1 represented by $C_{pos}[i]$ s make a pattern as $t = 00111$.

3.2.2. Multiple populations

We generated a chromosome for a pattern (i.e., term t) by deciding the length l of a chromosome and selecting l attributes from the positive set randomly. By using Q chromosomes, we formed one population with size Q for each positive set decomposed. Hence, we define the K-tuple population $POP_{OvA} = \left(POP_{OvA}^1, POP_{OvA}^2, \ldots, POP_{OvA}^K\right)$ for OvA-type decomposition, where POP_{OvA}^i is the population of patterns for the positive set Ω_i^+ by class i over all remaining classes. For OvO-type decomposition, we define $K(K-1)$-tuple population $POP_{OvO} = (POP_{OvO}^{12}, POP_{OvO}^{13}, \ldots, POP_{OvO}^{K,K-1})$, where POP_{OvO}^{ij} is the population of patterns for the positive set Ω_{ij}^+ corresponding to class i over class $j (j \neq i)$.

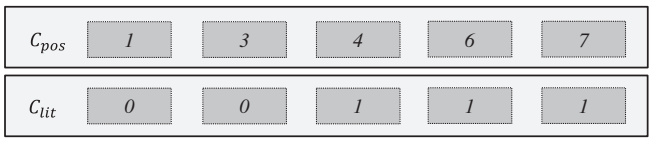

Fig. 5. Example of a variable-sized chromosome.

3.2.3. Fitness function

Experiments have shown that the classification accuracy of a strong prime pattern is higher than in other patterns (Hammer, Kogan, Simeone, & Szedmák, 2004; Guo & Ryoo, 2012). The quality of a pattern t can be evaluated by employing a fitness function in GA. In our work, for generating patterns with high coverage (namely, strong) and lower degree (namely, prime), we stipulate the fitness function fit_t for the chromosome t (i.e., term t) as

$$fit_t = G_t + \frac{1}{R_t^+} + M \times R_t^-, \tag{5}$$

where M is a large number used as a penalty for the case of covering observations in the negative set, and R_t^+ (R_t^-) represent the coverage rate (i.e., the ratio of the number of positive or negative observations covered by the pattern t to the size of the positive or negative set). From the definition of fit_t described above, we declare that a chromosome with the lowest final value fit_t has a tendency toward a short length and high coverage.

However, in order to avoid over-fitting, caused by noise contained in the training data, term t covering any observation in the negative class is considered as a pattern within the tolerance range $\gamma (0 < \gamma < 1)$. In other words, if the ratio of the number of observations in the negative set covered by term t to the number of observations in the positive set covered by term t is less than a pre-specified value γ, then we set R_t^- to zero and consider term t a pattern.

3.2.4. Genetic operators

The proposed algorithm requires setting various genetic operators to generate offspring. Table 4 offers a detailed description of the genetic operators. These include parent selection, crossover, and mutation, with crossover probability $(0 < r_c < 1)$ and mutation probability $(0 < r_m < 1)$ (Mitchell, 1998). We applied these operators to all chromosomes in POP_{OvA} and POP_{OvO}, depending on the decomposition type considered. Specifically, by generating a random number r between 0 and 1, we applied crossover and mutation operations to each parent only if $r < r_c$ and $r < r_m$, respectively.

3.2.5. Reconstruction of populations

By applying the genetic operators to each pair of chromosomes, we can generate Q offspring from each population. After evaluating the fitness of the offspring, we can select the best Q chromosomes, reconstructing a new population with the same size. Therefore, we have a new POP_{OvA} or POP_{OvO}, depending on the decomposition type considered.

3.2.6. Termination conditions

We terminate the algorithm when either the positive set is empty or the number of patterns generated is greater than a threshold. We define multiple sets of generated patterns as L_i-tuple $P_{OvA}^i = \left(p_{OvA}^{i1}, p_{OvA}^{i2}, \ldots, p_{OvA}^{i,L_i}\right),\ \forall i = 1, 2, \ldots, K$ for the positive set Ω_i^+ by OvA-type decomposition, and L_{ij}-tuple $P_{OvO}^{ij} = \left(p_{OvO}^{ij1}, p_{OvO}^{ij2}, \ldots, p_{OvO}^{ij,L_{ij}}\right),\ \forall i = 1, 2, \ldots, K, j \neq i$, for the positive set Ω_{ij}^+ by OvO-type decomposition, where p_{OvA}^{ik} and p_{OvO}^{ijk} are the k th pattern (i.e., term t) in P_{OvA}^i and P_{OvO}^{ij}, respectively, and L_i and L_{ij} are the numbers of patterns included in P_{OvA}^i and P_{OvO}^{ij}, respectively.

3.3. Theory formulation for multi-class LAD

By using the patterns and the coverage rates for the patterns, we can formulate theories for the two decomposition types as follows.

Table 4
Genetic operators.

Genetic operators	Description
Parent selection	Roulette-wheel selection
Crossover	Two-point crossover by selecting two points from two parents randomly and interchanging genes between them
Mutation	If a randomly generated integer η is in C_{pos}, the corresponding gene is deleted. Otherwise, a new gene with value η is added to C_{pos} and one of {0, 1} is randomly inserted into C_{lit} as the value for it.

3.3.1. OvA-type theory

To classify test data x, the OvA-type theory given is:

$$f_{OvA}(x) = \arg\max_i \sum_{k=1}^{L_i} w_{OvA}^{ik} \cdot p_{OvA}^{ik}(x), \tag{6}$$

where w_{OvA}^{ik} represents the coverage rate for p_{OvA}^{ik}. We have $p_{OvA}^{ik}(x) = 1$ if test data x is covered by pattern p_{OvA}^{ik}—otherwise, $p_{OvA}^{ik}(x) = 0$. When classifying test data x using Eq. (6), x is classified under class i if the value of $\sum_{k=1}^{L_i} w_{OvA}^{ik} \cdot p_{OvA}^{ik}(x)$ is the largest in class i. If $f_{OvA}(x) = 0$ or if there are at least two classes having the largest value $\sum_{k=1}^{L_i} w_{OvA}^{ik} \cdot p_{OvA}^{ik}(x)$, then test data x is left unclassified.

3.3.2. OvO type theory

The OvO-type theory is derived by

$$f_{OvO}(x) = \arg\max_i \sum_{j\neq i} \sum_{k=1}^{L_{ij}} w_{OvO}^{ijk} \cdot p_{OvO}^{ijk}(x) \tag{7}$$

where w_{OvO}^{ijk} represents the coverage rate of p_{OvO}^{ijk}. We have $p_{OvO}^{ijk}(x) = 1$ if test data x is covered by pattern p_{OvO}^{ijk}—otherwise, $p_{OvO}^{ijk}(x) = 0$. The class of test data x is decided according to the index giving the largest value $\sum_{j\neq i}\sum_{k=1}^{L_{ij}} w_{OvO}^{ijk} \cdot p_{OvO}^{ijk}(x)$. On the other hand, if $f_{OvO}(x) = 0$, or if there are at least two classes having the largest value $\sum_{j\neq i}\sum_{k=1}^{L_{ij}} w_{OvO}^{ijk} \cdot p_{OvO}^{ijk}(x)$, then test data x is left unclassified.

3.4. Procedure of IGA-FCMP for multi-class LAD

The overall procedure for the IGA-FCMP proposed—using two types of decompositions, OvA and OvO—is summarized in Figs. 6 and 7, respectively.

4. A numerical experiment

We performed a numerical experiment to evaluate the performance of the patterns generated by the IGA-FCMP algorithm using OvA- and OvO-type decompositions for the multi-class LAD as follows.

4.1. Design of a numerical experiment

For the experiment, we adopted six well-known datasets that have been used in the pertinent works for multi-class classification (Moreira, 2000; Mortada et al., 2013; Herrera & Subasi, 2013). They are available from the UCI machine-learning repository (Asuncion & Newman, 2007) and they are summarized in Table 5. These datasets have diverse ranges in terms of the number of classes and instances. For accurate results, we used all observations in the experiment, excluding instances with missing values. For example, we omitted eight instances in the *Escherichia coli (E. coli)* dataset because they contained missing values.

Using Weka 3.6.10, we implemented and performed the data binarization and the support-set generation procedure on the datasets. Next, we divided the pre-processed data into training and test datasets with a 50:50 rate and applied the suggested IGA-FCMP using OvA- and OvO-type decompositions to the training data, as described in Figs. 6 and 7. To do this, we implemented the IGA-FCMP algorithm using MATLAB R2013a by locating a set of suitable values for genetic parameters in pre-testing. More precisely, we considered six binary datasets from the UCI machine-learning repository in Table 6, and performed sensitivity analysis to find a good parameter setting for the IGA-FCMP with $\alpha = \beta = 1$ such as r_c, r_m, γ. Then, we conducted another sensitivity analysis to test the impact of two parameters α and β on the performance of the resulting theory at the three values of α and β (1, 5, and 10), and compared the results of the $3 \times 3 = 9$ resulting theories as in Table 7. Based on the experimental analysis, we could find that $(\alpha, \beta) = (5, 5)$ is the best combination in most case. Specifically, we could set the size of the population $Q = 100$, the probability of crossover $r_c = 0.8$, the probability of mutation $r_m = 0.01$, the maximum number of generations = 1000, and the number of consecutive generations without any improvement on the fitness score = 100. Furthermore, we set $\alpha = \beta = 5$, $\gamma = 0.1$ and $L_i = L_{ij} = 100$ for all sets of patterns.

For each positive set, Ω_i^+ and Ω_{ij}^+, we applied the IGA-FCMP and generated pattern sets P_{OvA}^i and P_{OvO}^{ij}. We then evaluated the classification performance of the generated patterns by implementing

IGA-FCMP procedure for OvA type multi-class LAD
Input: *Training data set,* K, α, β, γ
Set $\Omega_i^+, \Omega_i^-, P_{OvA}^i = \phi (i = 1, 2, \cdots, K)$
for $i = 1$ ***to*** K ***do***
  Initialize POP_{OvA}^i
  while $\Omega_i^+ \neq \phi$ *and* $n(P_{OvA}^i) \leq L_i$ ***do***
    Reconstruct POP_{OvA}^i *by applying genetic operators*
    Select a set, T, *of* α *chromosomes with best* fit_t *in* POP_{OvA}^i
    Check if each instance in Ω_i^+ *is covered by any pattern in* T
    if $fit_t < M$ *for any* $t \in T$
      $P_{OvA}^i \leftarrow P_{OvA}^i \cup T$,
      $\Omega_i^+ \leftarrow \Omega_i^+ \setminus V$, *where* V *is a set of data covered* β *times in* Ω_i^+,
      Set $T = \phi$
    end if
  end while

Fig. 6. IGA-FCMP procedure using OvA type decomposition.

IGA-FCMP procedure for OvO type multi-class LAD
Input: *Training data set,* K, α, β, γ
Set $\Omega_{ij}^{+}, \Omega_{ij}^{-}, P_{OvO}^{ij} = \phi (i = 1, 2, \cdots, K, j \neq i)$
for $i = 1$ ***to*** $K(K-1)$ ***do***
  for $j = 1$ ***to*** $K(K-1)$ ***do***
    Initialize POP_{OvO}^{ij}
    while $j \neq i$ *and* $\Omega_{ij}^{+} \neq \phi$ *and* $n(P_{OvO}^{ij}) \leq L_{ij}$ ***do***
      Reconstruct POP_{OvO}^{ij} *by applying genetic operators*
      Select a set, $\Omega_i^{+} \neq \phi$ *of* α *chromosomes with best* fit_t *in* POP_{OvO}^{ij}
      Check if each instance in fit_t *is covered by any pattern in T*
      if *fit* *for any* *fit*
        $P_{OvO}^{ij} \leftarrow P_{OvO}^{ij} \cup T$,
        $\Omega_{ij}^{+} \leftarrow \Omega_{ij}^{+} \setminus V$, *where* Ω_i^{+} *is a set of data covered* β *times in* Ω_{ij}^{+},
        Set $\Omega_i^{+} \leftarrow \Omega_i^{+} \setminus V$
      end if
    end while
  end for
end for

Fig. 7. IGA-FCMP procedure using OvO type decomposition.

Table 5
Dataset for a numerical experiment.

Datasets	Iris	Wine	Glass	*E. coli*	Dermatology	Segmentation
# of classes	3	3	6	6	7	8
# of instances	150	178	213	258	210	336
# of attributes	Numerical (4)	Numerical (13)	Numerical (10)	Nominal (34)	Numerical (19)	Nominal (1) Numerical (7)

Table 6
Binary dataset to set IGA-FCMP parameters.

Datasets	WBC	CHD	LD	BH	PID	CCS
# of instances (positive, negative)	683 (239, 444)	297 (137, 160)	345 (200, 145)	506 (260, 246)	768 (268, 500)	690 (307, 383)
# of attributes	11	14	7	14	9	15

WBC: Wisconsin breast canner.
CHD: Cleveland heart disease.
LD: BUPA liver disorder.
BH: Boston housing.
PID: Pima Indian diabetes.
CCS: Credit card scoring.

Table 7
Sensitivity analysis of (α, β) on classification accuracy.

(α, β)	WBC		CCS		BH		CHD		LD		PID	
	(a)	(b)	(a)	(b)	(a)	(b)	(a)	(b)	(a)	(b)	(a)	(b)
(1, 1)	92.98	15	84.35	50	82.61	39	76.51	32	64.16	72	69.01	108
(1, 5)	95.03	80	84.06	266	83.00	213	77.18	182	70.52	358	69.50	527
(1, 10)	93.86	173	84.35	543	82.61	438	76.51	398	71.10	752	74.74	999
(5, 1)	92.98	70	84.93	235	81.42	195	75.17	140	66.47	320	71.35	555
(5, 5)	95.61	95	84.64	335	84.90	245	77.40	235	70.00	420	74.00	650
(5, 10)	95.32	205	84.93	630	83.40	490	76.51	525	68.79	840	72.40	1000
(10, 1)	92.69	120	84.35	440	83.00	380	77.85	260	65.90	700	69.01	1000
(10, 5)	94.44	160	84.93	550	85.38	410	75.17	330	63.58	690	69.79	1000
(10, 10)	96.49	330	84.64	780	82.61	640	75.17	910	67.05	890	72.40	1550

(a): Average of the classification accuracy (%).
(b): Average of the number of patterns.

the test data. We calculated the classification accuracy rate (CAR) with the following equation:

$$\mathrm{CAR}(100\%) = \frac{\text{The number of instances with correct classification}}{\text{Size of Test Data set}} \times 100. \tag{8}$$

We repeated this procedure 30 times for each positive set, where the computing platform was an Intel i5 3.40 GHz processor desktop with 16 GB RAM.

4.2. Experimental results

First, we compared the characteristics of the generated patterns using OvA- and OvO-type decompositions from two representative data sets: Iris and Wine. The comparison is shown in Tables 8 and 9. Because these two datasets have three classes, the OvA-type has three positive sets and the OvO-type has six positive sets, as represented in the second column in the tables. The patterns generated using OvO-type decomposition have shorter average and maximum degrees than those using OvA-type decomposition. We think that the patterns generated using OvO decomposition are the strong prime patterns. The coverage rates for the generated patterns using OvO and OvA have some variation among pattern sets: We could not find any unilateral trend or predominance using one type of decomposition on the coverage rate of the generated patterns when we considered all positive sets.

For the convergence analysis of the suggested IGA-FCMP, we investigated the generated patterns in Tables 8 and 9, and found that, except only one case Ω_{21}^{+} for Wine dataset, most patterns with minimum degree are strong prime patterns with the maximum coverage rates in the last columns of Tables. In case of Ω_{21}^{+} for Wine dataset, a pattern with degree 2 was a strong prime pattern with coverage 94.59%. Obviously, these strong prime patterns are the optimal solutions to the corresponding pattern generation problem to find a strong prime pattern on positive set Ω_i^{+} or Ω_{ij}^{+}. Therefore, we claim that the solution generated by the suggested IGA-FCMP can converge to optimal solution, and two control parameters α and β play import roles to improve the quality of generated patterns by increasing the flexibility and efficiency of pattern generation procedure.

We can explain the behavior of the proposed IGA-FCMP for generating patterns as follows. In early iterations, patterns with a low degree and high coverage are generated with the fitness function. These simple patterns (i.e., the strong prime relaxation pattern) play the role of increasing the classification accuracy of the LAD theory. However, patterns with a high degree and low coverage are also generated as the iterations proceed, and the patterns in later iterations cover only some instances that the initial simple patterns cannot. These complex patterns (i.e., spanned patterns) have the role of covering observations that are uncovered in early iterations. They will also be of help to LAD decision theory to reduce misclassified decisions as well as to improve LAD robustness (Alexe, Alexe, Hammer, & Kogan, 2008; Alexe et al., 2007; Hammer et al., 2004).

Table 10 exhibits the classification accuracy for the proposed IGA-FCMP using OvA-type and OvO-type decompositions and compares them to the existing multi-class LAD and other supervised learning algorithms. All results in the table are expressed in 'average ± standard deviation' format. Due to the differences in the training and test sets for each dataset that were used in the research, we compared all the models using the results of the average classification accuracy. The values in bold font in the table represent the highest classification accuracy for each dataset.

Overall, the performance of the suggested IGA-FCMP is superior to others in almost every case. Except with one dataset—viz., Segmentation—the IGA-FCMP had better classification accuracy than all others. In particular, the IGA-FCMP using OvO-type decomposition had the highest degree of accuracy of all other approaches for the Iris, Wine, *E. coli* and Dermatology datasets. Besides the average values, the standard deviations for the IGA-FCMP were smaller than others. This means that the IGA-FCMP has more stable power than other algorithms for classifying various test data. Furthermore, if we compare the two types of decompositions, OvO was more accurate than OvA-type decomposition because it was more accurate in four of the five datasets.

Specifically, we can compare the results of the IGA-FCMP using OvO with the results in (a) and (c), where (a) shows the classification accuracy of two algorithms in Moreira (2000), and (c) is the experimental results of multi-class LAD in Mortada et al. (2013). Except the Segmentation dataset, the IGA-FCMP using OvO had better accuracy in four of the five datasets. For the Glass dataset, the accuracy difference with (a) is more than 30%, whereas the IGA-FCMP was worse than MC-LAD in (a) for the Segmentation dataset with a small difference less than 2%. This is because MC-LAD in Moreira (2000) used the enumeration-based pattern generation method without considering pattern characteristics such as prime, spanned or strong patterns. The central innovation with this method was to combine and enumerate random binary attributes (i.e., literals) as a way to find all patterns in the solution space. Unfortunately, the whole process is inefficient. Moreover, TABATA in (a) used the Tabu search algorithm for generating strong patterns, while focusing on the efficiency of heuristic-based pattern generation process. It resulted in a relatively poor performance compared with the IGA-FCMP. Mortada et al. (2013) proposed a multi-class LAD method for generating strong prime patterns, integrating ideas presented by Moreira (2000) using OvO and the MILP-based pattern generation method by Ryoo and Jang (2009). This method using strong prime patterns could improve the classification accuracy by reducing the number of unclassified observations. However, it still could not decrease classification errors, which can be achieved using strong spanned patterns (Hammer et al., 2004). However, the IGA-FCMP generates two types of patterns iteratively as explained above, while avoiding over-fitting by considering the tolerance range γ. As a result, our algorithm was more accurate than others, referring again to (a) and (c). These results show that the IGA-FCMP algorithm captures the efficiency as well as the effectiveness in the pattern generation procedure using OvO type decomposition.

Moreover, we can compare the results of the IGA-FCMP using OvA with the results in (b) and (d), where (b) refers to two approaches: CN2 (Clark & Niblett, 1989) and C4.5 (Quinlan, 1993), reported in Moreira (2000). CN2 is a machine learning algorithm based on a sequential covering rule induction, and C4.5 is a direct approach using decision tree. Although C4.5 and our algorithm report relatively similar accuracy, the IGA-FCMP was superior to C4.5 when using the Glass dataset, being more than 34% accurate. Furthermore, the IGA-FCMP had a smaller standard deviation than C4.5. Row (d) reports the experimental results using the MILP-based multi-class LAD by Herrera and Subasi (2013), once again inferior to the IGA-FCMP because of the limitation using only strong prime patterns. Thus, the proposed IGA-FCMP using OvA-type decomposition obtained better accuracy overall than other approaches in (b) and (d).

Table 11 shows the average CPU runtime of the proposed IGA-FCMP and other algorithms under consideration. Overall, the CPU runtime of the suggested IGA-FCMP is inferior to that by others in almost case. This is because IGA-FCMP is based on evolutionary search procedure to find a solution, and it takes a long time to converge as a basic GA. However, we notice that the CPU runtime by the IGA-FCMP is less than about 2 min in most case, and therefore, it seems to be favorable to apply the suggested

Table 8
Information on patterns for Iris dataset.

Decomposition type	Positive sets	Patterns by IGA-FCMP						
		Number of patterns	Degree			Pattern coverage rate (%)		
			min	average	max	min	average	max
OvA	Ω_1^+	5	1	1.80	2	100.00	100.00	100.00
	Ω_2^+	5	2	2.80	3	95.83	95.83	95.83
	Ω_3^+	5	1	1.80	2	95.65	95.65	95.65
OvO	Ω_{12}^+	5	1	1.80	2	100.00	100.00	100.00
	Ω_{13}^+	5	1	1.80	2	100.00	100.00	100.00
	Ω_{21}^+	10	1	1.80	2	12.50	56.25	100.00
	Ω_{22}^+	5	1	1.80	2	95.83	95.83	95.83
	Ω_{31}^+	10	1	1.60	2	47.83	76.96	100.00
	Ω_{32}^+	10	1	1.80	2	21.74	56.52	95.65

Table 9
Information on patterns for wine dataset.

Decomposition type	Positive sets	Patterns by IGA-FCMP						
		Number of patterns	Degree			Pattern coverage rate (%)		
			min	average	max	min	average	max
OvA	Ω_1^+	5	2	2.80	3	96.97	96.97	96.97
	Ω_2^+	25	1	2.96	4	2.70	22.16	75.68
	Ω_3^+	15	1	2.80	4	5.26	52.63	89.47
OvO	Ω_{12}^+	10	1	2.80	4	6.06	50.00	93.94
	Ω_{13}^+	10	1	1.80	2	6.06	52.42	100.00
	Ω_{21}^+	20	1	2.05	3	5.41	29.46	94.59
	Ω_{22}^+	15	1	2.27	3	2.70	26.12	94.59
	Ω_{31}^+	15	1	2.26	3	5.26	52.63	100.00
	Ω_{32}^+	5	1	1.8	2	89.47	89.47	89.47

Table 10
Comparison of the results of our algorithm with others.

		Classification accuracy of data sets (%)					
		Iris	Wine	Glass	*E. coli*	Dermatology	Segmentation
IGA-FCMP		94.80 ± 0.40	96.18 ± 1.76	**96.26 ± 1.06**	75.07 ± 0.60	92.18 ± 0.43	91.81 ± 2.18
(OvA)		96.00*	100.00*	98.13*	75.60*	92.74*	93.33*
IGA-FCMP		**95.73 ± 0.53**	**96.86 ± 1.48**	93.46 ± 1.92	**81.88 ± 1.00**	**96.26 ± 1.39**	89.24 ± 2.63
(OvO)		96.00*	98.88*	94.39*	82.74*	97.21*	91.43*
(a)	MC-LAD	n.a	92.70 ± 2.54	62.41 ± 5.88	78.34 ± 3.40	89.07 ± 2.84	91.21 ± 1.34
	TABATA	n.a	70.56 ± 10.41	58.57 ± 5.64	42.74 ± 0.44	80.66 ± 4.57	80.32 ± 6.58
(b)	CN2	n.a	90.23 ± 2.71	65.47 ± 5.25	77.03 ± 3.42	87.10 ± 2.57	85.73 ± 1.05
	C4.5	n.a	89.90 ± 3.11	62.80 ± 4.43	80.59 ± 4.14	94.48 ± 2.69	**95.09 ± 0.61**
(c)	MC-LAD	n.a	93.10 ± 3.20	65.00 ± 5.40	79.20 ± 4.20	n.a	n.a
(d)	MC-LAD	94.00 ± 2.20	91.33 ± 3.54	79.54 ± 5.35	n.a	n.a	n.a

(a): Multi-class LAD (MC-LAD) in Moreira (2000) – OvO type.
(b): Data mining methods in Moreira (2000) – OvA type.
(c): MC-LAD in Mortada et al.(2013) – OvO type.
(d): MC-LAD in Herrera et al. (2013) – OvA type.
n.a: Not available.
* Best classification accuracy.

IGA-FCMP to multi-class classification problems. If we compare the two types of decompositions, OvO had longer CPU runtime than OvA-type decomposition because it has more pairs of positive and negative sets to consider for generating patterns.

Based on the analysis results using a numerical experiment, we claim that the suggested IGA-FCMP has some advantages, making it superior to existing multi-class LAD and other supervised learning algorithms in terms of the classification accuracy. First, the IGA-FCMP uses a variable-sized chromosome to efficiently represent patterns with different lengths. This can allow us to handle sizable data efficiently, while avoiding any unnecessary memory allocation. Second, the fitness function considers the length and coverage of a pattern simultaneously so that the IGA-FCMP focuses on finding strong prime patterns preferentially. The fitness function also considers avoiding over-fitting, caused by noise contained in the training data, by introducing the tolerance range γ, which can extend the search up to strong spanned patterns simultaneously. As a result, the generated patterns can increase the classification accuracy. Finally, the IGA-FCMP uses two more control parameters α and β, managing the number of selected patterns at the end of GA procedure and the information used for pattern generation per iteration. This can improve the quality of generated

Table 11
Comparison of average CPU time.

		Average CPU time (sec.)					
		Iris	Wine	Glass	*E. coli*	Dermatology	Segmentation
IGA-FCMP (OvA)		15.1	16.9	24.9	31.4	21.4	22.5
IGA-FCMP (OvO)		21.1	22.8	73.9	129.4	74.2	107.6
(a)	MC-LAD	n.a	1.2	12.7	17.5	10.1	187.1
	TABATA	n.a	1.0	1.0	2.1	1.1	214.7
(b)	CN2	n.a	1.0	1.0	1.0	1.5	8.8
	C4.5	n.a	1.0	1.0	1.0	1.0	1.4
(c)	MC-LAD	n.a	1.0	39.0	49.0	n.a	n.a
(d)	MC-LAD	n.a	n.a	n.a	n.a	n.a	n.a

(a): Multi-class LAD (MC-LAD) in Moreira (2000) – OvO type.
(b): Data mining methods in Moreira (2000).
(c): MC-LAD in Mortada et al.(2013) – OvO type.
(d): MC-LAD in Herrera et al. (2013) – OvA type.
n.a: Not available.

patterns by enriching the search space for GA procedure. However, the IGA-FCMP has some drawbacks such that it has a relatively long CPU runtime compared to that by others, and it has a lot of control parameters to be set. In general, parameter setting is problem-dependent, and it is not easy to find a good combination of them to generate a good set of patterns with high classification accuracy.

5. Conclusions

In this paper, we proposed an efficient pattern generation method for multi-class LAD, called the IGA-FCMP, to solve multi-class classification problems. In particular, we considered two types of decomposition – namely, One versus All and One versus One – and designed a flexible chromosome and multiple populations. The suggested IGA-FCMP has two parameters for improving the classification accuracy of the generated patterns controlling the number of patterns to select at the termination of a genetic procedure and the number of times that an observation is covered by some patterns before deletion. Using a numerical experiment on six benchmark datasets from the UCI machine-learning repository, we demonstrated that the proposed IGA-FCMP was the most accurate and efficient approach since the IGA-FCMP outperformed other multi-class LAD algorithms and other supervised learning approaches.

The suggested IGA-FCMP using a variable-sized chromosome can be applied to sizable dataset in industry efficiently, while avoiding any unnecessary memory allocation. For instance, the IGA-FCMP for yield management in semiconductor manufacturing processes can classify the quality of wafers into multiple levels, reflecting defects and tolerance errors based on large datasets produced. Furthermore, we believe that two control parameters β and γ employed could improve the performance of the IGA-FCMP by enriching the pattern search space per iteration and helping to avoid over-fitting caused by noise, respectively. Finally, from the numerical experiment, we could have a slight idea for choosing one of decomposition types: The IGA-FCMP using OvO-type decomposition is suitable for a noisy dataset (i.e., Glass or *E. coli* datasets), whereas both methods are appropriate for a well-classified dataset (i.e., Iris or Wine datasets). This is an ongoing issue to be resolved in the literature these days.

In future research, we plan to consider another type of decomposition for multi-class classification problems using LAD, namely hierarchical decomposition. In particular, we plan to design an efficient genetic algorithm, one based on the IGA-FCMP, and compare its performance to the results presented in this paper. To do this, we will conduct another numerical experiment using the same six benchmark datasets. Eventually, we can compare all three decompositions types for multi-class classification using LAD. Furthermore, we intend to apply these methods to industrial multi-class classification problems, especially in the area of semiconductor manufacturing.

Acknowledgements

This research was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education, Science and Technology (2012R1A1A2005504).

References

Alexe, G., Alexe, S., Bonates, T. O., & Kogan, A. (2007). Logical analysis of data – The vision of Peter L. hammer. *Annals of Mathematics and Artificial Intelligence, 49*(1–4), 265–312.
Alexe, G., Alexe, S., Hammer, P. L., & Kogan, A. (2008). Comprehensive vs. comprehensible classifiers in logical analysis of data. *Discrete Applied Mathematics, 156*(6), 870–882.
Aly, M. (2005). *Survey on multiclass classification methods*. Technical Report: California Institute of Technology.
Asuncion, A., & Newman, D. (2007). *UCI machine learning repository*. Retrieved, 2014, Retrieved from http://archive.ics.uci.edu/ml/.
Berry, M. J., & Linoff, G. (1997). *Data mining techniques: For marketing, sales and customer support*. John Wiley & Sons, Inc.
Bishop, C. M. (2006). *Pattern recognition and machine learning*. New York: Springer.
Boros, E., Hammer, P. L., Ibaraki, T., & Kogan, A. (1997). Logical analysis of numerical data. *Mathematical Programming, 79*(1–3), 163–190.
Boros, E., Hammer, P. L., Ibaraki, T., Kogan, A., Mayoraz, E., & Muchnik, I. (2000). An implementation of logical analysis of data. *IEEE Transactions on Knowledge and Data Engineering, 12*(2), 292–306.
Boros, E., & Menkov, V. (2004). Exact and approximate discrete optimization algorithms for finding useful disjunctions of categorical predicates in data analysis. *Discrete Applied Mathematics, 144*(1), 43–58.
Clark, P., & Niblett, T. (1989). The CN2 induction algorithm. *Machine Learning, 3*(4), 261–283.
Crama, Y., Hammer, P. L., & Ibaraki, T. (1988). Cause-effect relationships and partially defined Boolean functions. *Annals of Operations Research, 16*(1), 299–325.
Furey, T. S., Cristianini, N., Duffy, N., Bednarski, D. W., Schummer, M., & Haussler, D. (2000). Support vector machine classification and validation of cancer tissue samples using microarray expression data. *Bioinformatics, 16*(10), 906–914.
Galar, M., Fernández, A., Barrenechea, E., Bustince, H., & Herrera, F. (2011). An overview of ensemble methods for binary classifiers in multi-class problems: Experimental study on one-vs-one and one-vs-all schemes. *Pattern Recognition, 44*(8), 1761–1776.
Gehler, P., & Nowozin, S. (2009). On feature combination for multiclass object classification. In *Proceedings of 2009 IEEE 12th International Conference on Computer Vision* (pp. 221–228).
Guo, C., & Ryoo, H. S. (2012). Compact MILP models for optimal and pareto-optimal LAD patterns. *Discrete Applied Mathematics, 160*(16), 2339–2348.
Hagan, M. T., Demuth, H. B., & Beale, M. H. (1996). *Neural network design*. Boston: PWS Publishing.
Hammer, P. L., & Bonates, T. O. (2006). Logical analysis of data—An overview: From combinatorial optimization to medical applications. *Annals of Operations Research, 148*(1), 203–225.
Hammer, P. L., Kogan, A., & Lejeune, M. A. (2012). A logical analysis of banks' financial strength ratings. *Expert Systems with Applications, 39*(9), 7808–7821.

Hammer, P. L., Kogan, A., Simeone, B., & Szedmák, S. (2004). Pareto-optimal patterns in logical analysis of data. *Discrete Applied Mathematics, 144*(1), 79–102.

Han, J., Kim, N., Jeong, M. K., & Yum, B. (2011). Comparisons of classification methods in the original and pattern spaces. *Expert Systems with Applications, 38*(10), 12432–12438.

Han, J., Kim, N., Yum, B., & Jeong, M. K. (2011). Pattern selection approaches for the logical analysis of data considering the outliers and the coverage of a pattern. *Expert Systems with Applications, 38*(11), 13857–13862.

Herrera, J. F. A., & Subasi, M. M. (March 2013). *Logical analysis of multi class data*. (No. RRR 5–2013). Rutcor research report: Rutgers center for operations research Rutgers University.

Jiawei, H., & Kamber, M. (2001). *Data mining: Concepts and techniques*. San Francisco: Morgan Kaufmann Publishers.

Kim, H. H., & Choi, J. Y. (2013a). An efficient genetic algorithm for pattern generation in logical analysis of data. In *Proceedings of international IIE conference*, Istanbul, Turkey.

Kim, H. H., & Choi, J. Y. (2013b). Iterative genetic algorithm based model to pattern generation for the logical analysis of data. In *Proceedings of 17th international conference on industrial engineering theory, applications and practice*, Busan, Korea.

Kotsiantis, S., & Kanellopoulos, D. (2006). Discretization techniques: A recent survey. *GESTS International Transactions on Computer Science and Engineering, 32*(1), 47–58.

Lugosi, G. (2002). *Pattern classification and learning theory. Principles of nonparametric learning*. Springer (pp. 1–56). Springer.

Mayoraz, E., & Alpaydin, E. (1999). Support vector machines for multi-class classification. In *Proceedings of the international workshop on artificial neural networks (IWANN99)* (pp. 833–842).

Mitchell, M. (1998). *An introduction to genetic algorithms*. Cambridge, Massachusetts: MIT Press.

Mitchell, T. M. (1997). *Machine learning*. New York: McGraw-Hill.

Mitov, I., Ivanova, K., Markov, K., Velychko, V., Stanchev, P., & Vanhoof, K. (2009). *Comparison of discretization methods for preprocessing data for pyramidal growing network classification method*. Sofia: New Trends in Intelligent Technologies (pp. 31–39). Sofia: New Trends in Intelligent Technologies.

Moreira, L. M. (2000). *The use of boolean concepts in general classification contexts*. (PhD thesis). École Polytechnique Fédérale de Lausanne.

Mortada, M. (2010). *Applicability and interpretability of logical analysis of data in condition based maintenance*. (PhD thesis). École Polytechnique de Montréal.

Mortada, M., Yacout, S., & Lakis, A. (2013). Fault diagnosis in power transformers using multi-class logical analysis of data. *Journal of Intelligent Manufacturing*. http://dx.doi.org/10.1007/s10845-013-0750-1.

Quinlan, J. R. (1993). *C4.5: Programs for machine learning*. San Mateo, CA: Morgan Kaufmann.

Ryoo, H. S., & Jang, I. (2009). MILP approach to pattern generation in logical analysis of data. *Discrete Applied Mathematics, 157*(4), 749–761.

Safavian, S. R., & Landgrebe, D. (1991). A survey of decision tree classifier methodology. *IEEE Transactions on Systems, Man, and Cybernetics, 21*(3), 660–674.

Schölkopf, B., & Smola, A. J. (2002). *Learning with kernels: Support vector machines, regularization, optimization, and beyond*. MIT Press.

Tax, D. M., & Duin, R. P. (2002). Using two-class classifiers for multiclass classification. In *Proceedings of 16th international conference on pattern recognition* (pp. 124–127).

Witten, I. H., & Frank, E. (2005). *Data mining: Practical machine learning tools and techniques*. Morgan Kaufmann.