Obtaining Classification Rules Using lvqPSO

Laura Lanzarini, Augusto Villa Monte(✉), Germán Aquino, and Armando De Giusti

Institute of Research in Computer Science LIDI (III-LIDI), Faculty of Computer Science, National University of La Plata (UNLP), La Plata, Buenos Aires, Argentina
{laural,avillamonte,gaquino,degiusti}@lidi.info.unlp.edu.ar

Abstract. Technological advances nowadays have made it possible for processes to handle large volumes of historic information whose manual processing would be a complex task. Data mining, one of the most significant stages in the knowledge discovery and data mining (KDD) process, has a set of techniques capable of modeling and summarizing these historical data, making it easier to understand them and helping the decision making process in future situations. This article presents a new data mining adaptive technique called lvqPSO that can build, from the available information, a reduced set of simple classification rules from which the most significant relations between the features recorded can be derived. These rules operate both on numeric and nominal attributes, and they are built by combining a variation of a population metaheuristic and a competitive neural network. The method proposed was compared with several methods proposed by other authors and measured over 15 databases, and satisfactory results were obtained.

Keywords: Classification rules · Data mining · Adaptive strategies · Particle swarm optimization · Learning vector quantization

1 Introduction

Data mining is a research field that in recent years has gained attention from various sectors. Government employees, business people and academics alike, for very different reasons, have contributed to the development of various techniques that can summarize the information that is available. This is one of the most important stages in the knowledge discovery and data mining process, and it is characterized for producing useful and novel information without any prior hypotheses. It encompasses a set of techniques capable of modeling available information and, even though there are different types of models, decision makers usually choose those that are self-explanatory. For this reason, rules, i.e., statements of the *IF condition1 THEN condition2* type, are preferred when characterizing that huge

A. Villa Monte—Post-Graduate Fellow, UNLP
G. Aquino—Post-Graduate Fellow, CONICET

volume of historical data that were automatically saved. In particular, we were interested in obtaining classification rules, i.e., rules whose consequence is formed by a single condition with the same attribute being involved: the class. However, most of the existing methods produce sets of rules that are so large and complex that, despite having the *IF-THEN* structure, rules become almost unreadable. For this reason, a new method to obtain classification rules is proposed in this article, with two essential features: the cardinality of the set of rules obtained is low, and the antecedent of the rules that are generated is reduced. To this end, the method proposed combines a competitive neural network with an optimization technique. The former is responsible for the supervised grouping of the examples with the purpose of identifying the most relevant attributes for building the rule. Then, by means of an optimization technique, the search process is guided towards the appropriate set of rules.

This paper is organized as follows: Section 2 lists some related articles, Sections 3 and 4 briefly describe the neural network and metaheuristic used, respectively, Section 5 details the method proposed, Section 6 presents the results obtained, and Section 7 presents a summary of the conclusions.

2 Related Work

The literature describes several methods for building classification rules that can operate with numerical and nominal attributes. The most popular one is the method known as C4.5, defined by Quinlan in [16], which can be used to generate a pruned classification tree whose branches allow obtaining the desired set of rules.

It should be noted that the lvqPSO method proposed in this article, unlike the C4.5 method, generates a list of classification rules. That is, when using the rules to classify new examples, they are not analyzed separately but rather in their order of appearance. Therefore, the rules are inspected one by one until the one that is applicable to the case at hand is found. This is related to how the set of rules was built, and has been used in other methods, such as PART, defined Witten in [2]; PSO/ACO, defined by Holden in [3]; and cAnt-MinerPB, proposed by Medland in [14].

In the case of PART, a pruned partial tree is built to determine each rule, following the ideas proposed by Quinlan in C4.5. The difference lies in that the tree is not built in full, but rather an error quota is applied to stop tree generation and select the best brach obtained so far. On the other hand, the PSO/ACO and cAnt-MinerPB method share with lvqPSO the idea of using a population-based metaheuristic to search for the best rules.

In particular, lvqPSO presents an approach that is based on Particle Swarm Optimization (PSO). This technique has already been used in previous works [1,5,8,19]. Unquestionably, one of the main issues when operating with nominal attributes is the impossibility of adequately covering all areas of the search space with the examples that are available. This results in a poor start for the population and a premature convergence to a local optimum. As a way to solve this problem, and at the same time reducing rule generation time, the initial state is obtained from an LVQ (Learning Vector Quantization) competitive neural network.

The literature describes methods that optimize a competitive neural network with PSO and significantly reduce the calculation time for the training phase [15], or methods that use PSO to determine the optimal number of competitive neurons to be used in the network, such as [4]. Unlike these papers, our proposal is using PSO to obtain the set of rules, and an LVQ network to avoid the premature convergence of the population.

3 Learning Vector Quantization (LVQ)

Learning Vector Quantization (LVQ) is a supervised classification algorithm that is based on centroids or prototypes [9]. It can be interpreted as a competitive neural network formed by three layers. The first layer is just an input layer. The second layer is where competence takes place. The output layer is responsible for the classification process. Each neuron in the competitive layer is associated to a number vector whose dimension is the same as that of the input examples, and a label that indicates the class that it is going to represent. Once the adaptive process finishes, these vectors will contain the information related to the classification centroids or prototypes. There are several versions of the training algorithm. The one used in this article is described below.

When the algorithm is started, the number K of centroids to be used must be indicated. This allows defining the architecture for the network, since the number of input entries and output results are given by the problem.

Centroids are initialized taking K random examples. Examples are then entered one by one, and centroid position is then adapted. To do so, the centroid that is closest to the example being analyzed is selected using a preset distance measurement. Since this is a supervised process, it is possible to determine if the example and the centroid belong or not to the same class. If the centroid and the example do belong to the same class, the centroid is "moved closer" to the example in order to strengthen representation. If, on the contrary, they belong to different classes, the centroid is "moved away". These movements are done by means of a factor or adaptation speed that allows weighing the distance for the move.

This process is repeated until modifications are below a preset threshold, or until the examples are identified with the centroids themselves in two consecutive iterations, whichever happens first.

For the implementation used in this article, the second nearest centroid is also analyzed and, should it belong to a different class than that of the example, and should it be at a distance that is less than 1.2 times the distance to the first centroid, the "moving away" step is applied.

Several variations of LVQ are described in [10].

4 Obtaining Classification Rules with PSO

Particle Swarm optimization or PSO is a population-based metaheuristic proposed by Kennedy and Eberhart [6] where each individual in the population, called particle, represents a possible solution to the problem and adapts by following three factors: its knowledge of the environment (its fitness value), its historical knowledge or previous experiences (its memory), and the historical knowledge or previous experiences of the individuals in its neighborhood (its social knowledge).

PSO was originally defined to work on continuous spaces, so a few considerations should be taken into account when working on discrete spaces. For this reason, Kennedy and Eberhart defined in [7] a new binary version of the PSO method. On of the key problems of this last method is its difficulty to change from 0 to 1 and from 1 to 0 once it has stabilized. This has resulted in different versions of binary PSO that seek to improve its exploratory capacity. In particular, the variation defined by Lanzarini et al. [13] will be used in this article.

Using PSO to generate classification rules that can operate on nominal and numerical attributes requires a combination of the methods mentioned above, since the attributes that will be part of the antecedent (discrete) have to be selected and the value or range of values they can take (continuous) has to be determined.

Since this is a population-based technique, the required information has to be analyzed for each individual in the population. A decision has to be made between representing a single rule or the entire set for each individual, and the representation scheme has to be selected for each rule. Given the objectives proposed for this work, the Iterative Rule Learning (IRL) [18] approach was followed, where each individual represents a single rule and the solution to the problem is built from the best individuals obtained after a sequence of runs. Using this approach implies that the population-based technique will be applied iteratively until achieving the desired coverage and obtaining a single rule in each iteration: the best individual in the population. Additionally, a fixed-length representation was chosen, where only the antecedent of the rule will be coded and, given the approach adopted, an iterative process will be carried out to associate all individuals in the population to a preset class, which does not require consequent codification. The code used for each particle is described in detail in [11].

The efficacy of population-based optimization techniques is closely related to the size of the population. For this reason, the method proposed here uses the variable population strategy defined in [12]. Thus, a minimum-size population can be used to initiate the process and then adjust the number of particles during the adaptive process.

The fitness value for each particle is calculated as follows:

$$Fitness = \alpha * balance * support * confidence - \beta * lengthAntecedent \tag{1}$$

where

- *support*: it is the support value for the rule. That is, the quotient between the number of examples that fulfill the rule and the total number of examples being analyzed.
- *confidence*: it is the confidence value for the rule. That is, the quotient of the number of examples that satisfy the rule and the number of those that satisfy the antecedent.
- *lengthAntecedent*: it is the quotient between the number of conditions used in the antecedent and the total number of attributes. It should be noted that each attribute can only be used once within the rule antecedent.
- α, β: these are two constants that represent the significance assigned to each term.
- *balance*: it takes values between (0,1], and it is used to compensate the effect of the imbalance between classes when calculating the support value. It is applied only when working with classes that have a number of examples that is above the mean. Let $C_1, C_2, ..., C_i, ..., C_N$ be the classes into which the examples are divided. N is the total number of classes. Let E_i be the number of examples in the n^{th} class. Let T be the total number of examples being used. That is,

$$T = \sum_{i=1}^{N} E_i \tag{2}$$

  Let j be the class to which the rule corresponding to the phenotype of the particle belongs. Let S_i be the number of examples in class C_i covered by the rule. Note that S_j corresponds to the support of the rule and

$$\sum_{i=1, i \neq j}^{N} S_i \tag{3}$$

  is the total number of examples incorrectly covered by that rule. Then, the value of this factor is calculated as follows

$$balance = \sum_{i=1, i \neq j}^{N} \frac{E_i - S_i}{T - E_j} \tag{4}$$

  That is, *balance* will have a value of 1 if the rule is perfect, i.e., its confidence is 1. On the other hand, Balance will be 0 if the rule covers all of the examples being used regardless of their class.

5 Proposed Method for Obtaining Rules: lvqPSO

Rules are obtained through an iterative process that analyzes non-covered examples in each class, starting with the largest classes. Each time a rule is obtained, the examples that are correctly covered by the rule are removed from the input data set. The process continues until all examples are covered or until the number of non-covered examples in each class is below the corresponding established minimum support or until a maximum number of tries has been done to obtain a rule, whichever happens first. It should be noted that, since the examples are removed from the input data set as they are covered by the rules, the rules operate as a classification list. That is, in order to classify a new example, the rules must be applied in the order in which they were obtained, and the example will be classified with the class that corresponds to the consequent of the first rule whose antecedent is verified for the example at hand.

Before starting the iterative process for obtaining the rules, the method starts with the supervised training of an LVQ neural network using the entire set of examples. The purpose of this training is identifying the most promising areas of the search space.

Since neural networks only operate with numerical data, nominal attributes are represented by means of dummy code that uses both binary digits and the different options that may be present in such nominal attribute. Also, before starting the training process, each dimension that corresponds to a numerical attribute is linearly scaled in [0,1]. The similarity measure used is the Euclidean distance. Once training is finished, each centroid will contain approximately the average of the examples it represents.

To obtain each of the rules, the class to which the consequent belongs is first determined. Seeking high-support rules, the method proposed will start by analyzing those classes with higher numbers of non-covered examples. The minimum support that any given rule has to meet is proportional to the number of non-covered examples in the class upon rule generation. That is, the minimum required support for each class decreases as iterations are run, as the examples in the corresponding class are gradually covered. Thus, it is to be expected that the first rules will have a greater support than the final ones.

After selecting the class, the consequent for the rule is determined. To obtain the antecedent, a swarm population will be optimized using the process described in Section 4. This swarm will be initialized with the information from the centroids. In Algorithm 1, the pseudo-code of the method proposed is shown.

6 Results Obtained

In this section, the performance obtained with the method proposed is compared against that of the cAnt-MinerPB, J48 (implementation of C4.5) and PART methods mentioned in Section 2 for generating classification rules for a known set of 15 databases of the UCI repository [17].

Thirty separate runs of ten-fold cross-validation were performed for each method, and an LVQ network with 9 neurons was used. In the case of the PART and C4.5 methods, a confidence factor of 0.3 and 0.25, respectively, was used for pruning the tree. The cAnt-MinerPB method was used with a colony of 9 ants. For the rest of the parameters, their default values were used.

Tables 1, 2 and 3 summarize the results obtained with each method for each of the databases, calculating mean and deviation. In each case, not only the

Algorithm 1. Pseudocode of the proposed method

```
Train LVQ network using all training examples.
Calculate the minimum support for each class.
while (termination criterion is not reached) do
    Choose the class with the highest number of non-covered examples.
    Build a reduced population of individuals from centroids.
    Evolve the population using variable population PSO.
    Obtain the best rule for the population.
    if (the rule meets support and confidence requirements) then
        Add the rule to the set of rules.
        Consider the examples classified by this rule as correctly covered.
        Recalculate the minimum support for this class.
    end if
end while
```

coverage accuracy of the set of rules was considered (Table 1), but also the clarity of the model obtained, which is reflected in the average number of rules obtained (Table 2) and the average number of terms used to form the antecedent (Table 3). In each case, a two-tailed mean difference test with a significance level of 0.05 was carried out, where the null hypothesis establishes that the means are equal. Based on the results obtained, when the difference is signficant according to the specified level, the best option was shaded and highlighted in bold in the table and when the difference is not significant equivalent solutions were only highlighted in bold.

As shown in Table 2, in most of the cases, the number of rules used by the method proposed is lower than with the other methods. This is due to the emphasis placed on the simplification of the model. The purpose is not only solving a classification problem, but also building a specific help tool for decision making. The expectation is generating a model that can identify the most relevant attributes and exposes how they are related among themselves when classifying available information.

In Table 1, in 5 of the cases, the accuracy achieved with lvqPSO is equivalent to or better than that obtained with other methods. In the case of the "Breast cancer", "Credit-a", "Heart disease" and "Zoo" databases, accuracy is better or approximately the same, and the number of rules is still lower. In the case of the "Iris" database, the number of rules is similar to that used by the deterministic methods C4.5 and PART, and lower than with cAnt-MinerPB.

However, when the problem to solve requires a large number of rules, the method proposed is less accurate. This happens in other databases that were tested. If these cases are analyzed in average, it can be stated that for a 2% improvement, a five-fold increase in rule set cardinality was required. For example, for the "Diabetes" database, the best accuracy is obtained with the cAnt-MinerPB method, with 26 rules, followed by C4.5, with 20 rules. There is a 2% difference in accuracy when compared to lvqPSO, but the latter uses less than 4 rules to solve the problem, i.e., only one sixth of the number of rules used by cAnt-MinerPB. The same happens with the "Breast-w" database, with lvqPSO

Table 1. Accuracy of the rule set obtained when applying the lvqPSO, PART, cAnt-MinerPB and J48 methods

Database	lvqPSO	cAnt-MinerPB	C4.5 (J48)	PART
Balance scale	$0,7471 \pm 0,0231$	$0,7696 \pm 0,0105$	$0,7732 \pm 0,0073$	$\mathbf{0,8219 \pm 0,0129}$
Breast cancer	$\mathbf{0,7203 \pm 0,0121}$	$0,7088 \pm 0,0208$	$0,5641 \pm 0,0466$	$0,6631 \pm 0,0207$
Breast-w	$0,9490 \pm 0,0079$	$0,9485 \pm 0,0049$	$\mathbf{0,9549 \pm 0,0050}$	$\mathbf{0,9566 \pm 0,0065}$
Credit-a	$\mathbf{0,8569 \pm 0,0055}$	$0,8493 \pm 0,0087$	$0,8515 \pm 0,0047$	$0,7454 \pm 0,0444$
Credit-g	$0,7060 \pm 0,0128$	$\mathbf{0,7374 \pm 0,0091}$	$0,7095 \pm 0,0072$	$0,6998 \pm 0,0111$
Diabetes	$0,7242 \pm 0,0152$	$\mathbf{0,7409 \pm 0,0050}$	$\mathbf{0,7438 \pm 0,0116}$	$\mathbf{0,7377 \pm 0,0156}$
Heart disease	$\mathbf{0,7650 \pm 0,0151}$	$\mathbf{0,7598 \pm 0,0164}$	$0,7433 \pm 0,0169$	$\mathbf{0,7647 \pm 0,0264}$
Heart statlog	$0,7626 \pm 0,0180$	$0,7648 \pm 0,0141$	$\mathbf{0,7811 \pm 0,0083}$	$0,7667 \pm 0,0150$
Iris	$\mathbf{0,9427 \pm 0,0218}$	$0,9380 \pm 0,0122$	$\mathbf{0,9453 \pm 0,0103}$	$\mathbf{0,9440 \pm 0,0110}$
Kr-vs-kp	$0,9365 \pm 0,0037$	$0,9812 \pm 0,0010$	$\mathbf{0,9929 \pm 0,0007}$	$0,9917 \pm 0,0013$
Mushroom	$0,9676 \pm 0,0046$	$\mathbf{0,9969 \pm 0,0001}$	$0,9843 \pm 0,0252$	$\mathbf{0,9962 \pm 0,0106}$
Promoters	$0,6977 \pm 0,0261$	$\mathbf{0,7917 \pm 0,0250}$	$0,7082 \pm 0,0389$	$0,6782 \pm 0,0600$
Soybean	$0,8735 \pm 0,0156$	$\mathbf{0,9066 \pm 0,0056}$	$\mathbf{0,9082 \pm 0,0053}$	$0,8936 \pm 0,0090$
Wine	$0,8727 \pm 0,0146$	$\mathbf{0,9192 \pm 0,0124}$	$0,8800 \pm 0,0144$	$0,8867 \pm 0,0091$
Zoo	$\mathbf{0,9417 \pm 0,0204}$	$\mathbf{0,9408 \pm 0,0153}$	$0,3290 \pm 0,0277$	$0,3190 \pm 0,0307$

Table 2. Number of rules obtained when applying the lvqPSO, PART, cAnt-MinerPB and J48 methods

Database	lvqPSO	cAnt-MinerPB	C4.5 (J48)	PART
Balance scale	$\mathbf{9,6700 \pm 0,5945}$	$14,6900 \pm 0,4483$	$41,3300 \pm 1,3300$	$38,5100 \pm 1,2215$
Breast cancer	$\mathbf{5,8600 \pm 0,4742}$	$25,4100 \pm 1,0744$	$11,4500 \pm 1,1128$	$18,8900 \pm 1,4098$
Breast-w	$\mathbf{2,8900 \pm 0,3281}$	$12,9000 \pm 0,5477$	$10,8500 \pm 0,7091$	$10,3500 \pm 0,4950$
Credit-a	$\mathbf{3,4100 \pm 0,1595}$	$25,0000 \pm 0,8654$	$18,0400 \pm 1,8404$	$33,3200 \pm 1,3028$
Credit-g	$\mathbf{8,3600 \pm 0,8113}$	$41,4100 \pm 2,0776$	$85,4500 \pm 3,4574$	$70,5700 \pm 1,5868$
Diabetes	$\mathbf{3,7900 \pm 0,3178}$	$26,6400 \pm 0,8996$	$22,1900 \pm 2,7517$	$7,5200 \pm 0,4264$
Heart disease	$\mathbf{4,9200 \pm 0,2821}$	$16,0800 \pm 0,7406$	$23,9000 \pm 0,9043$	$19,6400 \pm 0,4248$
Heart statlog	$\mathbf{4,6400 \pm 0,3777}$	$15,1200 \pm 0,6663$	$18,1900 \pm 1,2556$	$17,8700 \pm 0,4191$
Iris	$\mathbf{3,0600 \pm 0,0843}$	$8,3500 \pm 0,3808$	$4,6600 \pm 0,0966$	$3,7800 \pm 0,2658$
Kr-vs-kp	$\mathbf{3,6500 \pm 0,3274}$	$18,0900 \pm 0,9049$	$29,0700 \pm 0,6717$	$22,1100 \pm 0,6385$
Mushroom	$\mathbf{3,2300 \pm 0,1059}$	$22,6900 \pm 2,0328$	$18,6100 \pm 0,2378$	$11,2400 \pm 0,2591$
Promoters	$\mathbf{7,4250 \pm 0,3775}$	$11,5300 \pm 0,6430$	$16,7500 \pm 0,5874$	$\mathbf{7,2800 \pm 0,3765}$
Soybean	$\mathbf{24,6857 \pm 0,7151}$	$62,9700 \pm 2,1334$	$43,5400 \pm 0,2366$	$31,7300 \pm 0,4322$
Wine	$\mathbf{3,7818 \pm 0,0874}$	$6,6900 \pm 0,3900$	$7,6600 \pm 0,4766$	$5,6200 \pm 0,1317$
Zoo	$\mathbf{6,9500 \pm 0,0548}$	$10,9700 \pm 0,2214$	$8,3500 \pm 0,0707$	$7,6300 \pm 0,0483$

Table 3. Antecedent average length for each rule obtained when applying the lvqPSO, PART, cAnt-MinerPB and J48 methods

Database	lvqPSO	cAnt-MinerPB	C4.5 (J48)	PART
Balance scale	$1,8035 \pm 0,0910$	$\mathbf{1,5900 \pm 0,0527}$	$6,3494 \pm 0,0668$	$3,0807 \pm 0,0694$
Breast cancer	$\mathbf{1,7373 \pm 0,0497}$	$1,8464 \pm 0,0578$	$2,1210 \pm 0,0939$	$2,0048 \pm 0,0570$
Breast-w	$3,3173 \pm 0,3546$	$\mathbf{1,1622 \pm 0,0329}$	$3,8899 \pm 0,1581$	$2,1249 \pm 0,0800$
Credit-a	$1,4049 \pm 0,1186$	$\mathbf{1,2247 \pm 0,0504}$	$4,8299 \pm 0,2014$	$2,4844 \pm 0,0746$
Credit-g	$2,0349 \pm 0,1276$	$\mathbf{1,9258 \pm 0,1227}$	$5,6443 \pm 0,1209$	$2,9695 \pm 0,0890$
Diabetes	$2,4647 \pm 0,2027$	$\mathbf{1,1627 \pm 0,0238}$	$5,7461 \pm 0,3253$	$1,9255 \pm 0,1043$
Heart disease	$1,8766 \pm 0,1039$	$\mathbf{1,7455 \pm 0,0682}$	$3,9391 \pm 0,0952$	$2,5415 \pm 0,0692$
Heart statlog	$1,8491 \pm 0,0715$	$\mathbf{1,2927 \pm 0,0346}$	$4,6700 \pm 0,1629$	$2,8783 \pm 0,1099$
Iris	$1,2083 \pm 0,0518$	$1,1793 \pm 0,0329$	$2,6118 \pm 0,0540$	$\mathbf{0,9949 \pm 0,0135}$
Kr-vs-kp	$2,4750 \pm 0,1236$	$\mathbf{1,4986 \pm 0,1125}$	$7,7738 \pm 0,0318$	$3,1275 \pm 0,0689$
Mushroom	$1,6400 \pm 0,0733$	$\mathbf{1,0899 \pm 0,0294}$	$2,6302 \pm 0,0384$	$1,2609 \pm 0,0153$
Promoters	$1,1131 \pm 0,0302$	$1,0409 \pm 0,0527$	$2,2847 \pm 0,0364$	$\mathbf{0,9976 \pm 0,0416}$
Soybean	$3,1299 \pm 0,2543$	$3,3858 \pm 0,0537$	$6,0213 \pm 0,0349$	$\mathbf{2,7127 \pm 0,0709}$
Wine	$2,7811 \pm 0,2248$	$\mathbf{1,0626 \pm 0,0340}$	$3,1027 \pm 0,1232$	$1,5529 \pm 0,0759$
Zoo	$1,6675 \pm 0,0462$	$1,6888 \pm 0,0601$	$4,0057 \pm 0,0242$	$\mathbf{1,4693 \pm 0,0145}$

being less accurate than C4.5 and PART by approximately 1% but using only one third of the number of rules with less queries in each antecedent.

7 Conclusions

A novel method for obtaining classification rules has been presented. This method is based on PSO and can operate with numerical and nominal attributes.

An LVQ neural network was used to adequately initialize the population of rules. The centroids obtained when grouping available data allow identifying the relevance of each attribute for the examples. In any case, this metric is not enough to select the attributes that will form the rule, and it is at this point where PSO takes control to carry out the final selection.

A representation for the rules was used, combining a binary representation that allows selecting the attributes that are used in the rule with a continuous representation used only to determine the boundaries of the numerical attributes that are part of the antecedent. A variation of binary PSO was used whose population is adequately initialized with the information from the centroids in the previously trained LVQ network and which has the ability of adjusting population size.

The results obtained when applying the method proposed on a set of test databases show that the lvqPSO method obtains a simpler model. In average, it uses approximately 40% of the number of rules generated by the other methods, with antecedents formed by just a few conditions and an acceptable accuracy given the simplicity of the model obtained.

Although not included in this article, the measurements performed using the method proposed but using fixed-size population PSO resulted in a less accurate set of rules. This is because the architecture of the LVQ network must be indicated beforehand, which affects grouping quality.

References

1. Chen, M., Ludwig, S.: Discrete particle swarm optimization with local search strategy for rule classification. In: 2012 Fourth World Congress on Nature and Biologically Inspired Computing (NaBIC), pp. 162–167 (2012)
2. Frank, E., Witten, I.H.: Generating accurate rule sets without global optimization. In: Proceedings of the Fifteenth International Conference on Machine Learning, ICML 1998, pp. 144–151. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA (1998)
3. Holden, N., Freitas, A.A.: A hybrid pso/aco algorithm for discovering classification rules in data mining. Journal of Artificial Evolution and Applications **2008**, 2:1–2:11 (2008)
4. Hung, C., Huang, L.: Extracting rules from optimal clusters of self-organizing maps. In: Second International Conference on Computer Modeling and Simulation, ICCMS 2010, vol. 1, pp. 382–386 (2010)
5. Jiang, Y., Wang, L., Chen, L.: A hybrid dynamical evolutionary algorithm for classification rule discovery. In: Second International Symposium on Intelligent Information Technology Application, vol. 3, pp. 76–79 (2008)
6. Kennedy, J., Eberhart, R.C.: Particle swarm optimization. In: Proceedings of the IEEE International Conference on Neural Networks, pp. 1942–1948 (1995)
7. Kennedy, J., Eberhart, R.C.: A discrete binary version of the particle swarm algorithm. In: Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics, vol. 5, pp. 4104–4108. IEEE Computer Society, Washington, DC, USA (1997)
8. Khan, N., Iqbal, M., Baig, A.: Data mining by discrete pso using natural encoding. In: 2010 5th International Conference on Future Information Technology (FutureTech), pp. 1–6 (2010)
9. Kohonen, T.: The self-organizing map. Proceedings of the IEEE **78**(9), 1464–1480 (1990)
10. Kohonen, T., Schroeder, M.R., Huang, T.S. (eds.): Self-Organizing Maps, 3rd edn. Springer, New York (2001)
11. Lanzarini, L., Monte, A.V., Ronchetti, F.: Som+pso. a novel method to obtain classification rules. Journal of Computer Science & Technology **15**(1), 15–22 (2015)
12. Lanzarini, L., Leza, V., De Giusti, A.: Particle swarm optimization with variable population size. In: Rutkowski, L., Tadeusiewicz, R., Zadeh, L.A., Zurada, J.M. (eds.) ICAISC 2008. LNCS (LNAI), vol. 5097, pp. 438–449. Springer, Heidelberg (2008)
13. Lanzarini, L., López, J., Maulini, J.A., De Giusti, A.: A new binary PSO with velocity control. In: Tan, Y., Shi, Y., Chai, Y., Wang, G. (eds.) ICSI 2011, Part I. LNCS, vol. 6728, pp. 111–119. Springer, Heidelberg (2011)

14. Medland, M., Otero, F.E.B., Freitas, A.A.: Improving the *c*Ant-Miner$_{\mathrm{PB}}$ classification algorithm. In: Dorigo, M., Birattari, M., Blum, C., Christensen, A.L., Engelbrecht, A.P., Groß, R., Stützle, T. (eds.) ANTS 2012. LNCS, vol. 7461, pp. 73–84. Springer, Heidelberg (2012)
15. Özçift, A., Kaya, M., Gülten, A., Karabulut, M.: Swarm optimized organizing map (swom): A swarm intelligence based optimization of self-organizing map. Expert Systems with Applications **36**(7), 10640–10648 (2009)
16. Quinlan, J.R.: C4.5: programs for machine learning. Morgan Kaufmann Publishers Inc., San Francisco (1993)
17. UCI: Machine learning repository. http://archive.ics.uci.edu/ml
18. Venturini, G.: Sia: a supervised inductive algorithm with genetic search for learning attributes based concepts. In: Brazdil, P.B. (ed.) ECML-93. LNCS, vol. 667, pp. 280–296. Springer, Berlin Heidelberg (1993)
19. Wang, H., Zhang, Y.: Improvement of discrete particle swarm classification system. In: 2011 Eighth International Conference on Fuzzy Systems and Knowledge Discovery (FSKD), vol. 2, pp. 1027–1031 (2011)