Neurocomputing 116 (2013) 311–316

Contents lists available at SciVerse ScienceDirect

Neurocomputing

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

Two-stage learning for multi-class classification using genetic programming

Hajira Jabeen [a,*], Abdul Rauf Baig [b]

[a] Abasyn University, Islamabad, Pakistan
[b] National University of Computer and Emerging Sciences, Islamabad, Pakistan

ARTICLE INFO

Available online 16 January 2013

Keywords:
Classification
Genetic programming
Classifier
Expression
Rule
Algorithm

ABSTRACT

This paper introduces a two-stage strategy for multi-class classification problems. The proposed technique is an advancement of tradition binary decomposition method. In the first stage, the classifiers are trained for each class versus the remaining classes. A modified fitness value is used to select good discriminators for the imbalanced data. In the second stage, the classifiers are integrated and treated as a single chromosome that can classify any of the classes from the dataset. A population of such classifier-chromosomes is created from good classifiers (for individual classes) of the first phase. This population is evolved further, with a fitness that combines accuracy and conflicts. The proposed method encourages the classifier combination with good discrimination among all classes and less conflicts. The two-stage learning has been tested on several benchmark datasets and results are found encouraging.

1. Introduction

Data classification finds its application in many real world problems, like fraud detection, face recognition, speech recognition and knowledge extraction from databases. The field of data classification is receiving increased importance due to unpredictability and complexity of real-world data. Evolutionary algorithms have shown evident performance for classification tasks. Genetic Programming (GP) is one of the evolutionary algorithms introduced by Koza [1] for automatic evolution of computer programs (including classifiers). GP has been successfully used for evolution of classifier-programs like decision trees [2]. Other GP based classification approaches include evolution of neural networks [3–5], autonomous classification systems [6], rule induction algorithms [7], fuzzy rule based systems and fuzzy petri nets [5,8]. Most of these methods involve defining a grammar that is used to create and evolve classification algorithms using GP.

Various researchers [9–13] have used GP for evolution of classification rules. The rule based systems include, atomic representations proposed by Eggermont [14,15] and SQL based representations proposed by Freitas et al. [12]. Tunsel and Jamshidi [16], Berlanga et al. [17] and Mendes et al. [18] introduced evolution of fuzzy rules using GP. Chien et al. [19] used fuzzy discrimination function for classification. Falco et al. [20] discovered comprehensive classification rules that use continuous value attributes. Bozarczuk et al. [21,22] used different set of functions applicable to different type of attributes that represent rules as disjunctive normal form. This type of GP is also referred as constrained syntax GP. Tsakonas et al. [23] introduced two GP based systems for medical domain and achieved noticeable performance. Lin et al. [24] proposed a layered GP, where different layers correspond to different populations that perform feature extraction and classification. Another method is evolution of arithmetic expressions for classification. The arithmetic expressions can be used for numerical data and they output a real value. This real value is translated into the classification decision using different thresholds. This includes static thresholds [25,26], dynamic thresholds [26,27] and slotted thresholds [28].

Multi-class classification problems are common in the real world applications for the tasks like object recognition, character recognition, person recognition, disease diagnosis and several others. Many classification algorithms are binary in nature and must be extended for multi-class classification. These include neural networks, decision trees, k-nearest neighbor, naive Baye's classifiers, and support vector machines [29]. GP also needs to be extended for multiclass classification problems. Several methods have been presented to use GP for multi-class classification problems. Most noticeable among them is the one-versus-all method also known as binary decomposition method. This method has been used widely in GP based multi-class classification. In this method, one classifier is evolved for each class, discriminating a particular class from other classes present in the data. The final decision is made by presenting the input vector to classifiers of all classes. The classifier with positive or highest output is declared the winner. This method has been explored by many researchers [30–34]. Another relatively different method proposed by Muni et al. [35], uses a multi-tree representation,

* Corresponding author. Tel.: +92 321 5708908.
E-mail addresses: hajirajabeen@hotmail.com (H. Jabeen), rauf.baig@nu.edu.pk (A.R. Baig).

0925-2312/$ - see front matter © 2012 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.neucom.2012.01.048

where a single classifier is an integrated version of individual classifiers for all classes. This amalgamated classifier is evolved in search of best classifier that has the ability to classify any of the class in one evolution.

Several other methods like 'all versus all' [36], error correcting output codes [37], and generalized error correcting output codes [38] have also been used to tackle multi-class classification problems by binary classification algorithms. However, none of them has been used in GP due to the large number of computations.

The drawback of binary decomposition method is the conflicting situations, where more than one classifier outputs a positive signal or none of the classifier outputs a belong-to signal. This situation degrades the classification accuracy. Several conflict resolution methods have been devised for this problem but they require extra processing during training and classification step. Another problem is the presence of skewed data. The data appears unbalanced for classification of a single class versus remaining classes. This problem is solved by increasing the number of training instances to make them appear balanced for each class [30,36]. This is named 'interleaved data format' where the samples belonging to class under consideration are repeated and alternately placed between samples belonging to other classes. This strategy increases the training data as well as the training time.

The proposed staged approach overcomes these two problems. It evolves the classifiers in two different stages that perform discrimination and integration, and incorporates a discriminative fitness function which takes care of skewed data without increasing the computation. The integrated evolution eliminates the conflicting situations decreasing the evaluation time required for conflict resolution. The proposed algorithm is detailed in the next section.

2. Proposed methodology

Many attempts have been made to develop general approaches to multi-class classification. One of the well known methods, in machine learning community, is one vs. all method. It involves learning a discriminator for each pair of class labels. The proposed classification mechanism uses the same principle but divides the training process into two phases. The first stage resembles the traditional binary decomposition method. The output, given by this phase, is a set of classifier populations for each class in the data. The second phase uses this population to populate the individual chromosomes using some selection criteria. Once the chromosome population is created, it is evolved in search of better accuracy. The output of second phase is a single chromosome that can classify any class present in the data.

2.1. Stage 1

In this stage, populations of arithmetic classifiers are trained to discriminate between absence and presence of a particular class amongst many classes present in the training data. Therefore we will have as many populations, as there are classes in the training data.

2.1.1. Classifier representation

The arithmetic classifier expression (ACE) is represented as a binary tree created using the function set $=\{+,-,/,*\}$ and terminal set $=$ {real attributes of the data, ephemeral constant}. The classifier represents an arithmetic relationship between attributes of the data. For each instance, it outputs a real value. For binary classification, the positive real value indicates the presence of a class and a negative value represents the absence of that class.

For example, consider a dataset with four attributes $[A_1,A_2,A_3,A_4]$ and two classes C_1 and C_2. two classes.

An arbitrary classifier for class C_1 can be

$$(A3 \times A4)/(A1+A2) \tag{2}$$

For an instance with values

$$[1,2,3,4] \text{ belonging to the class } C_1 \tag{3}$$

The classifier will output a real value '4'.

This indicates that the instance belongs to the class C_1.

2.1.2. Initialization

Initialization plays an important role in success of any evolutionary algorithm. A diverse and efficient initialization technique can lead to effective search during the evolutionary process. We have used the well known ramped half and half method [1] for initialization of the population. The ramped half and half method utilizes advantages of both full and grow initialization schemes with equal probability for each depth level till the maximum allowed depth. This method has been widely used for initialization in classification problems [30,35].

2.1.3. Fitness

For a particular class, the classifier should be trained to output a positive response (accept) for the class under consideration and negative response (reject) for the instances belonging to the other classes. This measure leads to the problem of skewed data i.e. the number of positive, and a negative instance is not same. The researchers suggest that, in case of skewed data high accuracy may not represent good discrimination ability of a classifier [44]. On the other hand, a classifier with high area under the convex hull AUC has better discrimination ability. Therefore, we have used the discriminative power of the classifier as its fitness function.

The AUC function is considered better than accuracy in case of skewed data, a classifier with 80% accuracy may represent a case where a classifier always output a negative value and 80% of the data does not belong to the class C_i. On the other hand, 80% of AUC means the classifier can successfully discriminate between 80% of the samples, as belonging or not belonging to the class C_i. Several classifiers like rotation forests also deal with the skewed data [42,43].

Let 'n' be the number of classes present in the data. Let C_i be the class under consideration, P_i be the number of samples belonging to the class C_i (positive samples) and N_i be the number of samples belonging to other classes C_i' (negative samples).

The fitness function for evolving classifiers for class C_i is

$$F_i = 1/2[\,(\text{true}-\text{positives}/P_i)+(\text{true}-\text{negatives}/N_i)] \times 100 \tag{4}$$

A classifier is evolved to discriminate between one class and rest of the classes such that output O of classifier F for class C_i is positive for instances belonging to class C_i and negative for instances not belonging to class C_i.

$$O[F(C_i)] \in +Z\ \forall \text{ instances of class } C_i \tag{5}$$

and

$$O[F(C_i)] \in -Z\ \forall \text{ instances of class } C_i' \tag{6}$$

where

$$C_i' = [C_1,C_2 \ldots C_n]-C_i \tag{7}$$

These positive and negative real values are converted into boolean values by considering positive values as 1 and negative values as 0.