Applied Soft Computing 9 (2009) 1296–1303

Contents lists available at ScienceDirect

Applied Soft Computing

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

Evolutionary multi-feature construction for data reduction: A case study☆

Leila S. Shafti *, Eduardo Pérez

Universidad Autónoma de Madrid, 28049 Madrid, Spain

ARTICLE INFO

Article history:
Received 6 November 2008
Received in revised form 11 March 2009
Accepted 26 April 2009
Available online 6 May 2009

Keywords:
Genetic algorithm
Feature construction
Non-algebraic feature
Attribute interaction
Machine learning
Data reduction

ABSTRACT

Real-world data are often prepared for purposes other than data mining and machine learning and, therefore, are represented by primitive attributes. When data representation is primitive, preprocessing data before looking for patterns becomes necessary. The low-level primitive representation of real-world problems facilitates the existence of complex interactions among attributes. If lack of domain experts prevents traditional methods to uncover patterns in data due to complex attribute interactions, then the use of soft computing techniques such as genetic algorithms becomes necessary. This article introduces MFE3/GA^{DR}, a data reduction method derived from the learning preprocessing system MFE3/GA. The method restructures the primitive data representation by capturing and compacting hidden information into new features in order to highlight regularities to the learner. We thoroughly analyze the empirical results obtained on the poker hand data set. The results show that this approach successfully compacts the set of low-level primitive attributes into a smaller set of highly informative features which outline patterns to the learner; thus, the new approach provides data reduction and yields learning a smaller and more accurate classifier.

1. Introduction

Since the birth of data mining, its data preprocessing phase has been recognized as crucial for the success of the overall process [7]. One of the main steps in data preprocessing is data reduction. In domains lacking human expertise or knowledge, data reduction becomes harder and requires automatic treatment by proper techniques.

Feature selection is one of the common techniques used for reducing data dimension. Feature selection is crucial for highlighting the relevant attributes to the learner. However, when data is represented only in terms of primitive attributes and no prior knowledge is provided about the concept, feature selection becomes more difficult. Primitive data representation enables the existence of attribute interactions, whose complexity may become a severe hindrance to most feature selection algorithms used in data mining. *Interaction* among attributes exists when the relation between one attribute and the target concept is not constant for all values of the other attributes [24,21,8]. When attribute interactions exist, a feature selection method may fail to improve learning since interacting attributes are easily confounded with irrelevant attributes [8,13]. Distinguishing interacting attributes from irrelevant ones has recently received attention [19,13,33]. However, when primitive attributes are provided for representing data, even if a feature selection method successfully detects all interacting attributes, still the underlying relations among attributes and the target concept are opaque and difficult to learn [3,29]. Thus, identifying relevant attributes by a feature selection method is not sufficient for learning concepts with primitive data representation.

Combination of feature selection and feature construction has been used and proved to have an outstanding impact on data mining results [17]. When feature construction (FC) is applied to concepts with interactions, it aims to transform the primitive data representation of data into a new one where interactions are encapsulated into new features and highlighted to the learner. If FC finds the appropriate features, such a change of representation makes the target concept easy to learn. Many progresses have been achieved by FC methods [31,34,35]; nevertheless, learners still face serious difficulties to succeed when confronted with complex attribute interactions [9,28].

This paper suggests a new form of applying MFE3/GA (multi-feature extraction with a genetic algorithm) as a preprocessing FC method to transform the data representation into a smaller and more regular representation. MFE3/GA originally adds constructed features to the given set of attributes and, therefore, increases the data dimension. It assumes that the combination of new features with original attributes can improve learning accuracy. Previous empirical experiments demonstrated that MFE3/GA successfully

☆ This work was partially supported by the Spanish Ministry of Science and Technology, under Grant numbers TSI2005-08225-C07-06 and TIN2008-02081.

* Corresponding author.
E-mail addresses: leila.shafti@uam.es (L.S. Shafti), eduardo.perez@uam.es (E. Pérez).

1568-4946/$ – see front matter © 2009 Elsevier B.V. All rights reserved.
doi:10.1016/j.asoc.2009.04.003

selects the interacting attributes and constructs highly informative features defined as functions over selected attributes, therefore, significantly improves predictive accuracy [27]. However, it was observed that the standard learner applied to the set of original and constructed features focussed mainly on the constructed features, raising the question of whether the original features could be removed from MFE3/GA's output. Thus, in this work, the method is modified to replace the original attributes with the new features. The modified method, named MFE3/GA^{DR}, benefits from the characteristics inherited from MFE3/GA to improve learning. The use of genetic algorithm (GA) as a global search strategy helps this method to deal with the problem of interactions better than other FC methods. Moreover, GA facilitates the construction and evaluation of several features simultaneously which is necessary when several complex interactions exist among attributes. Also, the use of non-algebraic (operator-free) representation for expressing new features in MFE3/GA^{DR} contributes to reduce the error rate on unseen data. Experiments reported in this article show that the highly informative features constructed by the new method reduce data dimension, increase learning accuracy, and are easily interpretable. The constructed features successfully capture and encapsulate relations among attributes when the only available knowledge about the concept is primitive training data.

Section 2 reviews MFE3/GA and highlights the most important aspects of the method. Section 3 explains how this method can be used for data reduction by abstracting relations among attributes into a set of highly informative features. The benefit of this form of data reduction is shown empirically using the poker hand data set (from UCI benchmarks [2]) whose results are thoroughly analyzed as a case study in Section 4. This data set is represented by primitive attributes and its underlying concept is difficult to discover by current machine learning systems. Experiments show that the preprocessing method, MFE3/GA^{DR}, successfully discovers regularities in the concept and compacts them into few and easily interpretable constructed features in order to improve learning. Conclusions are summarized in Section 5.

2. MFE3/GA: encapsulating information

MFE3/GA is a preprocessing method that highlights the interactions among attributes by constructing new features in order to facilitate learning the target concept [27]. It receives as input the training data samples and the original attribute set. Each sample consists of a vector of attribute values and a binary class label, stating whether or not the sample belongs to the target concept (label '1' for positive and label '0' for negative). MFE3/GA divides the task of constructing new features into two tasks. The first task is search through the space of attribute subsets to find subsets of interacting attributes. The second is to construct the definition of one function for each subset of attributes generated in the former task. The more promising constructed functions are then added as new features to the original attribute set; after that the new representation of data is given to a standard learner such as C4.5 [23] to proceed with the data mining process. The current version of MFE3/GA assumes that the class labels are binary and a discretization preprocessing algorithm such as [16,4,14] has been applied to transform all continuous attributes to nominal ones before running the system.

The search space for finding relevant attributes and constructing functions is large and with high variation when complex interactions exist in the target concept. The importance of applying a global search to such a space for constructing features has been shown previously [6,18,30,32]. MFE3/GA uses GA, a global search technique that has been shown promising in converging to an optimal solution. The following sections explain important details about GA in MFE3/GA.

2.1. Individual's representation

As any other GA-based approach, MFE3/GA maintains a population of individuals. Each individual represents a set of attribute subsets such as $Ind = \{S_1, \ldots, S_k\}$, where $S_i \subseteq S$ and S is the set of N original attributes x_1 to x_N. Each individual has different number of subsets; thus, the length of individuals is variable. If an individual contains a subset S_i, where $S_i = S$ or $|S_i| \leq 1$, then S_i does not participate in feature construction and fitness measurement. However, it is considered for GA operations to produce diversity in the population. Each subset S_i is represented by a bit-string of length N, where each bit shows the presence or absence (in the subset) of one of the N original attributes. Thus, an individual representing k subsets is a bit-string of length $k \cdot N$ ($k > 0$), such as $Ind = \langle b_{11} \cdots b_{1N} \cdots b_{k1} \cdots b_{kN} \rangle$. To avoid unnecessary growth of individuals, the number of subsets in each individual is limited by a parameter K defined by the user.

Each attribute subset in an individual is associated with a function defined over the attributes in the subset and induced from the data. Such functions are expressed by a non-algebraic (operator-free) representation; that is, no algebraic operator is used for representing functions. Non-algebraic representation has been used successfully for expressing constructed functions by other FC methods [21,29,20,35,1]. This form of representation permits extracting part of the function's description from data and inducing the rest. The function f_i for any given subset $S_i = \{x_{i_1}, \ldots, x_{i_m}\}$ is defined by assigning binary labels (as outcomes of the function) to all the tuples in the Cartesian product $V_{i_1} \times \cdots \times V_{i_m}$ (as inputs of the function), where V_{i_p} is the set of values of attribute x_{i_p}. The label assigned to each tuple t_j (that is, label '1' for positive and label '0' for negative) depends on the class labels of the training samples that match the tuple. A training sample matches a tuple t_j if its values for attributes in S_i are equal to the corresponding values in the j th combination of attribute values in the Cartesian product. That is, a sample matches a tuple t_j if for all $1 \leq p \leq m$, $v_{i_p} = t_{j_p}$, where v_{i_p} is the value of attribute x_{i_p} in the sample and t_{j_p} is the p th element in the tuple t_j. To assign labels, tuples are categorized into three groups, as follows. If there are no training samples matching t_j, this tuple is categorized as *Unknown tuple*. If all training samples matching t_j belong to the same class, the tuple is a *Pure tuple*. If there is a mixture of classes in the samples matching t_j, it is a *Mixed tuple*. Then, a label is assigned to each tuple t_j according to its category, as discussed next:

Category 1: *Unknown tuple*: A label is assigned to $f_i(t_j)$ stochastically, according to the class distribution in the training samples.

Category 2: *Pure tuple*: The label of training samples that match t_j is the label assigned to $f_i(t_j)$.

Category 3: *Mixed tuple*: The label assigned to $f_i(t_j)$ depends on the numbers of tuples labeled by Pure tuples as positive (label '1') and negative (label '0'), p_2 and n_2 respectively. If $p_2 > n_2$, the negative label, that is '0', is assigned; and otherwise, the positive label, that is '1', is assigned to $f_i(t_j)$ (i.e., the opposite label of the majority label in function).

Note that the label of mixed tuples depends on the definitive labels in the function under construction, which are the labels of pure tuples. The opposite label to the most frequent label among pure tuples is selected.

The procedure for extracting the definition of f_i from data partitions the subspace defined by S_i into four areas, as illustrated in Fig. 1. Each f_i identifies similar patterns of interaction among attributes in S_i and *compresses* them into the negative or positive area (Category 2). The unseen area (Category 1) is covered by