Expert Systems With Applications 63 (2016) 74–85

Contents lists available at ScienceDirect

Expert Systems With Applications

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

Constrained dynamic rule induction learning

Fadi Thabtah[a], Issa Qabajeh[b,*], Francisco Chiclana[b]

[a] Applied Business and Computing, NMIT, Auckland
[b] Center of Computational Intelligence, School of Computer Science and Informatics, De Montfort University, Leicester, UK

ARTICLE INFO

Article history:
Received 24 February 2015
Revised 29 March 2016
Accepted 19 June 2016
Available online 24 June 2016

Keywords:
Classification
Data mfining
Prediction
PRISM
Rule induction
Online security

ABSTRACT

One of the known classification approaches in data mining is rule induction (RI). RI algorithms such as PRISM usually produce If-Then classifiers, which have a comparable predictive performance to other traditional classification approaches such as decision trees and associative classification. Hence, these classifiers are favourable for carrying out decisions by users and therefore they can be utilised as decision making tools. Nevertheless, RI methods, including PRISM and its successors, suffer from a number of drawbacks primarily the large number of rules derived. This can be a burden especially when the input data is largely dimensional. Therefore, pruning unnecessary rules becomes essential for the success of this type of classifiers. This article proposes a new RI algorithm that reduces the search space for candidate rules by early pruning any irrelevant items during the process of building the classifier. Whenever a rule is generated, our algorithm updates the candidate items frequency to reflect the discarded data examples associated with the rules derived. This makes items frequency dynamic rather static and ensures that irrelevant rules are deleted in preliminary stages when they don't hold enough data representation. The major benefit will be a concise set of decision making rules that are easy to understand and controlled by the decision maker. The proposed algorithm has been implemented in WEKA (Waikato Environment for Knowledge Analysis) environment and hence it can now be utilised by different types of users such as managers, researchers, students and others. Experimental results using real data from the security domain as well as sixteen classification datasets from University of California Irvine (UCI) repository reveal that the proposed algorithm is competitive in regards to classification accuracy when compared to known RI algorithms. Moreover, the classifiers produced by our algorithm are smaller in size which increase their possible use in practical applications.

1. Introduction

Data mining, which is based on computing and mathematical sciences, is a common intelligent tool currently used by managers to perform key business decisions. Traditionally, data analysts used to spend a long time gathering data from multiple sources and little time was spent on analysis due to the limited computing resources. Though since the rapid development of computer networks and the hardware industry, analysts nowadays are spending more time on examining data, seeking useful concealed information. In fact, after the recent development of cloud computing, data collection, noise removal, data size and data location are no longer obstacles facing analysts. Data analysis or data mining is concerned about finding patterns from datasets that are useful for users, particularly managers, to perform planning (Thabtah & Hammoud, 2015).

One of the known data mining tasks that involve forecasting class labels in previously unseen data based on classifiers learnt from training dataset is classification. Normally, classification is performed in two steps: Constructing a model often named the classifier from a training dataset, and then utilising the classifier to guess the value of the class of test data accurately. This type of learning is called supervised learning since while building the classifier the learning is guided toward the class label. Common applications for classification are medical diagnoses (Rameshkumar, Sambath, & Ravi, 2013), phishing detection (Abdelhamid, Ayesh, & Thabtah, 2014), etc. There have been many different classification approaches including decision trees (Witten, Frank, & Hall, 2011), Neural Network (NN) (Mohammad, Thabtah, & McCluskey, 2013), Support Vector Machine (SVM) (Cortes & Vapnik, 1995), Associative Classification (AC) (Thabtah, Cowling, & Peng, 2004), rule induction (RI) (Holte, 1993) and others. The latter two approaches, i.e. AC and RI, extract classifiers which contain "If-Then" rules so this explains their wide spread applicability. However, there are differences

* Corresponding author.
E-mail addresses: Fadi.fayez@nmit.ac.nz (F. Thabtah), p12047781@myemail.dmu.ac.uk (I. Qabajeh), chiclana@dmu.ac.uk (F. Chiclana).

http://dx.doi.org/10.1016/j.eswa.2016.06.041
0957-4174/

between AC and RI especially in the way rules are induced as well as pruned. This article falls under the umbrella of RI research.

PRISM is one of the RI techniques which was developed in Cendrowska (1987) and slightly enhanced by others, i.e. Elgibreen and Aksoy (2013), Stahl and Bramer (2008), and Stahl and Bramer (2014). This algorithm employs separate-and-conquer strategy in knowledge discovery in which PRISM generates rules according to the class labels in the training dataset. Normally for a class, PRISM starts with an empty rule and keeps appending items to the rule's body until this rule reaches zero error (Definition 8- Section 2). When this occurs, the rule gets induced and the training data samples connected with the rule are discarded. The algorithm continues building other rules in the same way until no more data connected with the current class can be found. At this point, the same steps are repeated for the next-in-line class until the training dataset becomes empty.

One of the obvious problems associated with PRISM is the massive numbers of rules induced, which normally results in large size classifiers. This problem is attributed to the way PRISM induces the rules where it keeps adding items to the rule's body until the rule becomes 100% accurate despite the low data coverage. In other words, PRISM does not mind inducing many specific rules, each covering a single data sample, rather producing a rule, say, with 90% accuracy covering 10 data samples. This excessive learning limits the applicability of PRISM as a decision making tool for managers in application domains and definitely overfits the training dataset. This is since managers normally prefer a summarised set of rules that they are able to control and comprehend rather a larger high maintenance set of rules. In fact, there should be a trade-off between the number of rules offered and the predictive accuracy performance of these rules.

This paper investigates shortcomings associated with PRISM algorithm. Specifically, we look into three main issues:

(1) The search space reduction: When constructing a rule for a particular class, PRISM has to evaluate the accuracy of all available items linked with that class in order to select the best one that can be added to the rule's body. This necessitates large computations when the training data has many attribute values and can be a burden especially when several unnecessary computations are made for items that have low data representation (weak items). A frequency threshold that we call (*freq*) can be employed as a pre-pruning of items with low frequency. It prevents these items from being part of rules, and therefore the search space gets minimised.
(2) PRISM only generates a rule when its error is zero, which may cause overfitting the training dataset. We want to derive good quality rules, not necessarily with 100% accuracy, to reduce overfitting and increase data coverage. We utilise rule's strength parameter (Rule_Strength) to separate between acceptable and non-acceptable rules in our classifier.
(3) When removing training data linked with a rule, we ensure that other items which have appeared in the removed data are updated. In particular, we amend the frequency of the impacted items. This indeed maintains the true weight of the items rather the computed frequency from the initial input dataset.

In response to the above raised issues, we are developing in this article, a new dynamic learning method based on RI that we name enhanced Dynamic Rule Induction (eDRI). Our algorithm discovers the rule one by one per class and primarily uses a freq threshold to limit the search space for rules by discarding any items with insufficient data representation. For each rule, eDRI updates items frequency that appeared within the deleted training instances of the generated rule. This indeed gives a more realistic classifier with lower numbers of rules leading to a natural pruning of items during the rule discovery phase. Lastly, the proposed algorithm limits the use of the default class rule by generating rules with accuracy < 100%. Often these rules are ignored by PRISM algorithm since they don't hold zero error. These rules are only used during class prediction phase instead of the default class rule and when no 100% accuracy rule is able to classify a test data.

This paper is structured as follows: Section 2 illustrates the classification problem and its main related definitions. Section 3 critically analyses PRISM and its successors, and Section 4 discusses the proposed algorithm and its related phases besides a comprehensive example that reveals eDRI's insight. Section 5 is devoted to the data and the experimental results analysis, and finally, conclusions are provided in Section 6.

2. The classification problem and definitions

Given a training dataset *T*, which has *x* distinct columns (attributes) Att_1, Att_2, …,Att_n, one of which is the class, i.e. *cl*. The cardinality of *T* is |*T*|. An attribute may be nominal which means it takes a value from a predefined set of values or continuous. Nominal attributes values are mapped to a set of positive integers whereas continuous attributes are preprocessed by discretising their values using any discretisation method. The aim is to make a classifier from *T,* e.g. *Classfiier*: $Att \rightarrow cl$, which forecasts the class of previously unseen dataset.

Our classification method employs a user threshold called ***freq***. This threshold serves as a fine line to distinguish strong items ruleitems<item, class> from weak ones based on their computed occurrences in *T*. Any ruleitem that its frequency passes the *freq* is called as a strong ruleitem, otherwise it is called weak ruleitem. Below are the main terms used and their definitions.

Definition 1. An *item* is an attribute plus its values name denoted (A_i, a_i).

Definition 2. A *training example* in *T* is a row consisting of attribute values (A_{j1}, a_{j1}), …, (A_{jv}, a_{jv}), plus a class denoted by c_j.

Definition 3. A *ruleitem r* has the format<*body, c*>, where *body* is a set of disjoint items and *c*is a class value.

Definition 4. The frequency threshold (*freq*) is a predefined threshold given by the end user.

Definition 5. The body frequency (*body_Freq*) of a *ruleitem r in T* is the number of data examples in *T* that match *r's body*.

Definition 6. The frequency of a *ruleitem r in T* (*ruleitem_freq*) is the number of data examples in *T* that match *r*.

Definition 7. A *ruleitem r* passes the *freq* threshold if, *r*'s $|body\_Freq|/\ |D| \geq freq$. Such a *ruleitem* is said to be a strong *ruleitem*.

Definition 8. A rule *r* expected accuracy is defined as $|ruleitem\_freq|/\ |body\_Freq|$.

Definition 9. A rule in our classifier is represented as: $body \rightarrow cl$, where body is a set of disjoint attribute values and the consequent is a class value. The format of the rule is: $a_1 \wedge a_2 \wedge ... \wedge a_n \rightarrow cl_1$.

3. Literature review

PRISM is a key algorithm for building classification models that contain simple yet effective easy to understand rules. This algorithm was developed in 1987 based on the concept of separate and conquer where data examples are separated using the available class labels. For each class (w_i) data samples, an empty rule (If nothing then w_1) is formed. The algorithm computes the frequency