Expert Systems With Applications 45 (2016) 410–423

Contents lists available at ScienceDirect

Expert Systems With Applications

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

Discretization of continuous attributes through low frequency numerical values and attribute interdependency

Md. Geaur Rahman, Md Zahidul Islam*

School of Computing and Mathematics, Charles Sturt University, Bathurst, NSW 2795, Australia

ARTICLE INFO

Keywords:
Data discretization
Data pre-processing
Data cleansing
Missing value imputation
Corrupt data detection
Data mining

ABSTRACT

Discretization is the process of converting numerical values into categorical values. There are many existing techniques for discretization. However, the existing techniques have various limitations such as the requirement of a user input on the number of categories and number of records in each category. Therefore, we propose a new discretization technique called low frequency discretizer (*LFD*) that does not require any user input. There are some existing techniques that do not require user input, but they rely on various assumptions such as the number of records in each interval is same, and the number of intervals is equal to the number of records in each interval. These assumptions are often difficult to justify. LFD does not require any assumptions. In LFD the number of categories and frequency of each category are not pre-defined, rather data driven. Other contributions of LFD are as follows. LFD uses low frequency values as cut points and thus reduces the information loss due to discretization. It uses all other categorical attributes and any numerical attribute that has already been categorized. It considers that the influence of an attribute in discretization of another attribute depends on the strength of their relationship. We evaluate LFD by comparing it with six (6) existing techniques on eight (8) datasets for three different types of evaluation, namely the classification accuracy, imputation accuracy and noise detection accuracy. Our experimental results indicate a significant improvement based on the sign test analysis.

© 2015 Elsevier Ltd. All rights reserved.

1. Introduction

1.1. Importance of discretization

Organizations use Data Mining and Knowledge Discovery algorithms for making better decisions. A data mining algorithm extracts interesting patterns (such as logic rules and clusters) that could otherwise be extremely difficult for us to extract (Pyle, 1999; Van Hulse, Khoshgoftaar, & Huang, 2007).

Many data mining algorithms can only deal with categorical attributes and are unable to handle numerical attributes (Agrawal & Srikant, 1994; Dougherty, Kohavi, & Sahami, 1995; Liu, Hussain, Tan, & Dash, 2002). However, natural datasets often contain both numerical and categorical attributes. The data mining algorithms generally discretize/categorize numerical attributes and thereby consider them as categorical attributes in order to perform various data mining tasks. Therefore, discretization techniques (Kurgan & Cios, 2004; Yang & Webb, 2009) are considered to be crucial in various fields of data mining. For example, the Naive-Bayes (Yang & Webb, 2009) and Apriori (Agrawal & Srikant, 1994) algorithms can only handle categorical attributes and therefore they need to discretize numerical attributes. Each of them belongs to the top 10 data mining algorithms (Wu et al., 2008).

Some other data mining algorithms can handle numerical attributes. However, often the efficiency and effectiveness of a data mining algorithm increases when it makes use of a discretization algorithm (Garcia, Luengo, Saez, Lopez, & Herrera, 2013; Yang & Webb, 2009). Discretization is also considered to be an important part of data preprocessing and cleansing that is likely to improve the quality of the results obtained by various data mining algorithms (Han & Kamber, 2006).

Discretization can also be useful for data cleansing tasks including missing value imputation and corrupt data detection (CDD). For example, missing value imputation techniques such as DMI (Rahman & Islam, 2011) and FIMUS (Rahman & Islam, 2014), and CDD techniques such as CAIRAD (Rahman, Islam, Bossomaier, & Gao, 2012) rely on discretization algorithms.

1.2. The problem statement and notations

A discretization algorithm converts a numerical attribute into a categorical attribute with a set of mutually exclusive intervals

* Corresponding author. Tel.: +61263384214.
E-mail addresses: grahman@csu.edu.au (Md. Geaur Rahman), zislam@csu.edu.au (M. Zahidul Islam).
URL: http://csusap.csu.edu.au/~zislam (M. Zahidul Islam)

http://dx.doi.org/10.1016/j.eswa.2015.10.005
0957-4174/© 2015 Elsevier Ltd. All rights reserved.

(Kurgan & Cios, 2004; Yang & Webb, 2009). The intervals are separated by cut-points and each interval has a width w. For example, a numerical attribute called Age can be converted into a categorical attribute with a number of intervals such as [10–14], (14–19], and (19–24]. The width w of an interval here is 5 and the cut points are 10, 14, 19 and 24. Finding the accurate number of intervals t, and the right width w of each interval is a challenging task (Garcia et al., 2013).

We consider a dataset D_F as a two dimensional table where rows represent records $R = \{R_1, R_2, \ldots R_N\}$ and columns represent attributes $A = \{A_1, A_2, \ldots A_M\}$. The size of the dataset is the number of the records $|D_F| = |R| = N$. The attributes can be either numerical or categorical. A numerical attribute such as *Age* contains numerical/continuous values such as 33 and 34. A categorical attribute such as *City* contains categorical values such as Sydney and Bathurst. Categorical values do not have any natural ordering. Let, $A^n \subset A$ be the set of numerical attributes and $A^c \subset A$ be the set of categorical attributes. We represent the number of numerical attributes in D_F by n (that is $|A^n| = n$) and the number of categorical attributes by c (that is $|A^c| = c$). If the jth attribute A_j is numerical then the domain of the attribute can be represented as $A_j = [low, up]$, where low is the lowest limit and up is the highest limit of the domain. If the attribute is categorical then the domain can be represented as $A_j = \{a_{j1}, a_{j2}, \ldots a_{jk}\}$, where the domain has k different values i.e. $|A_j| = k$. The domain size of a numerical attribute (such as "Number of Children") is $|A_i| = (up - low)$ and the domain size of a categorical attribute $|A_j|$ is the number of distinct values of the attribute.

Often an attribute A_{class} out of the set of the categorical attributes A^c (i.e. $A_{\text{class}} \in A^c$) is considered to be the class attribute (Quinlan, 1986; 1993; 1996) of a dataset. A class attribute (also known as labels of the records) can be used to classify the records. For example, the attribute "Diagnosis" can be a class attribute of a patient dataset having information on a number of patients.

We consider that R_{ij} is the jth attribute value of the ith record. By a "missing value" we mean that R_{ij} is missing/absent for some reasons. We denote a missing value by a "?" mark, i.e. $R_{ij} =$?. By a "noisy value" (or "corrupt value") we mean that the R_{ij} value is incorrectly recorded in D_F. Incorrect values can be recorded in a dataset due to various reasons including machine malfunctioning and human error.

A discretization process converts a numerical attribute A_j into a categorical attribute by introducing a set of categories through a set of cut-points $P_j = \{P_{j1}, P_{j2}, \ldots P_{jq}\}$, where P_{j1} and P_{jq} are the lowest and the highest values of A_j, respectively. Based on the cut points the categories/intervals for A_j can be defined as $I = \{[P_{j1}, P_{j2}], (P_{j2}, P_{j3}], \ldots, (P_{jq-1}, P_{jq}]\}$.

1.3. Some existing algorithms and their limitations

A number of discretization techniques have been proposed in the literature (Ching, Wong, & Chan, 1995; Fayyad & Irani, 1993; Kim & Han, 2000; Kurgan & Cios, 2004; Liu & Setiono, 1997; Liu, Wong, & Wang, 2004; Mehta, Parthasarathy, & Yang, 2005; Wong & Chiu, 1987; Yang & Webb, 2009). There are various limitations of the existing techniques (Garcia et al., 2013) and therefore room for further improvement.

An early discretization algorithm called equal width discretizer (EWD; Wong and Chiu, 1987) divides a numerical attribute into t intervals/categories, where the intervals have equal width w. The number of intervals t is user defined. The width of each interval is calculated as $w = \frac{up-low}{t}$, where low is the lowest value and up is the highest value of the domain of the attribute. The intervals for A_j in EWD are calculated as $I = \{[low, low + w], (low + w, low + 2w], \ldots, (low + (t-1)w, up]\}$. While each interval has the equal width, the number of actual values in an interval can of course be different to the number of values in another interval.

Another existing algorithm called equal frequency discretizer (EFD; Wong and Chiu, 1987) divides the values of a numerical attribute into t intervals in such a way so that each interval contains an equal number of values. The number of values (or frequency), f of each interval is calculated as $f = \frac{N}{t}$, where N is the number of records in D_F. In this case the widths of the intervals can be different. Like EWD, EFD also requires a user defined number of intervals t. It can be a difficult task for a user to estimate the accurate number of intervals in advance (Garcia et al., 2013).

Instead of a user defined number of intervals t, FFD (Yang & Webb, 2009) requires a user defined number of records (i.e. frequency) f for each interval. The number of interval t is then calculated as $t = \frac{N}{f}$, where each interval contains a fixed number of records. It is also difficult for a user to guess a suitable/appropriate frequency f of each interval.

A recent algorithm called proportional discretizer (PD; Yang and Webb, 2009) does not require a user input on either the number of intervals t or the frequency f of each interval. PD suggests/assumes that the number of intervals t and the number of records (in each interval) f should be equal (i.e. $t = f$). Based on the assumption, the desired t is then obtained from the relations $t \times f = N$ and $t = f$, where N is the total number of records of a dataset. However, the equality assumption (i.e. $t = f$) is questionable. It is difficult to justify the reason why a sensible discretization can be carried out considering $t = f$ for all datasets.

FIMUS (Rahman & Islam, 2014) therefore drops the equality assumption. It automatically finds the value of t considering $t = \sqrt{|A_j|}$, where $|A_j|$ is the domain size of A_j. The technique then divides the values of A_j into t intervals, where the width of each interval is equal. The width w is calculated as $w = \frac{up-low}{t}$. However, the assumptions of FIMUS also suffer from lack of justifications.

None of the above algorithms consider the influence of a class attribute (A_{class}) while discretizing a numerical attribute. Since a class attribute classifies the records it has an influence/relationship with the other attributes. The influence of the class attribute is considered by a recent discretization algorithm called class attribute interdependence maximization (CAIM; Kurgan and Cios, 2004) while identifying the cut-points of a numerical attribute. The technique aims to discretize numerical values in such a way so that the classification and prediction accuracy of a classifier built from the discretized dataset improves.

The technique initially considers all distinct values of a numerical attribute as the candidate cut-points. It then considers the lowest and highest candidates/values as the first two cut points. For each of the remaining candidates it then calculates the CAIM value (Kurgan & Cios, 2004) which indicates how well the discretization of the attribute using the candidate cut point agrees with the categories of the class attribute. The candidate having the maximum CAIM value is considered to be the third cut-point. The technique then computes the CAIM values for each of the remaining candidates and selects the one having the maximum CAIM value as the fourth cut point. The process of computing the CAIM values and selecting the candidate with the highest CAIM value as a cut point continues as long as the maximum CAIM value of the current iteration is higher than the maximum CAIM value of the previous iteration. It was shown (Kurgan & Cios, 2004) that the technique outperforms six other discretization algorithms.

1.4. The proposed algorithm: LFD

We argue that existing discretization techniques including EWD, EFD, FFD, PD, FIMUS and CAIM have a fundamental flaw in the basic concept. These techniques allow a numerical value (of an attribute) with high frequency to be selected as a cut point. By "high frequency" of a numerical value we mean that the value has a high number of appearances in the dataset for the attribute. That is, many records

Download English Version:

https://daneshyari.com/en/article/382471

Download Persian Version:

https://daneshyari.com/article/382471

Daneshyari.com