Technologies used: Eclipse Java, Weka libraries, Java awt, swing.

Number of lines of code = 1049 lines

Abstract

With the rapid increase in the use of databases, missing data make up an important and unavoidable problem in data management and analysis. Because the mining of association rules can effectively establish the relationship among items in databases, therefore, discovered rules can be applied to predict the missing data. We have used a new method that uses association rules based on weighted voting to impute missing data. Three databases were used to demonstrate the performance of the proposed method.

Association rule mining is an important type of data mining. Its target is to find interesting associations or correlation relationships among a large set of data items. So if we use association rules to the application domain of missing data the research on association rules can effectively establish the relationship of attribute in databases containing missing values.

1. Introduction

Missing Data imputation is one of the most difficult problems in data analysis processing which has been extensively studied by the statistics community. The five basic ways of dealing with missing data are ignore the tuple, fill the missing value manually, use global constant, use attribute mean for missing or use probable value to fill the missing value.

Association rule mining is an important type of data mining. Its target is to find interesting association or correlation relationships among a large set of data items. So if we use association rules to the application domain of missing data, the research on association rules can effectively establish the relationship of attribute in databases containing missing values. Because the association rules describe the where all data, including the missing ones, should hold the similar relationships.

Weka is a popular suite of Machine Learning software which contains collection of visualization tools and algorithms for data analysis and predictive modeling. Weka supports several data mining tasks such as data preprocessing, clustering, classification, regression, visualization and feature selection. The experimental result demonstrates that our accuracy is better than Weka.

We have extended our algorithm to replace noisy data. To detect noisy data we have made use of HyperClique Patterns and replaced the noisy data by Weighted Voting Method.

A. Generating Association Rule

Association rule mining is a basis of the missing data imputation procedure, so we first give a brief review on the definition of association rules. Let I = {$i_{1\,2}$,...,t.} be a set of

attribute values, called items. Let D be a set of database transactions where each transaction T is a set of items such that T ci . An association rule is an implication of the form A B, where A c I ,B c 1, and A n B = 0 . We refer to A as the antecedent of the rule, and B as the consequent of the rule. The rule AB holds in the transaction set D can be described with support and confidence. That is,

1] Support (A => B) = P (A u B)

2] Confidence (A => B) = P(B/A) = Support (A,B) I Support (A)

Support(A) is the percentage of transactions in D that contain A. Support and confidence are two measures of rule interestingness which reflect the usefulness and certainty of discovered rules respectively. Rules that satisfy both a minimum support threshold and a minimum confidence threshold are called strong ones.

Association rule mining is a two-step process :

Find all frequent item sets that means each of these item sets will occur at least as frequently as a pre-determined minimum support count.

Generate strong association rules from the frequent item sets that means these rules must satisfy minimum support and minimum confidence.

B. **General Method**

The object of the study is to discover data item associations from giving databases and using them to impute missing data. For this purpose, first, we preprocessed training sets and test sets. The training set is complete data set and the test set will be removed all data in an attribute or some data in more than one attribute to form incomplete data set. So we test if our method can successfully recover the missing ones. After that, we use the association rule mining method to discovery data items association from the training data set to generate strong association rules. For example, supposed that a data set has three variables and a, b, c are data items, a rule a Abc may be obtained from the training set. If a data case contains missing data item such as a, b, ?, the rule a Abc indicates that the data item c is likely to be missing data item. Next, for the imputing missing data, we classify strong association rules to generate a special rules subset named constrained association rules set whose consequents are restricted to the missing data items. Finally, we use missing data imputation procedure to select data items to impute missing data.

The measure used to evaluate the proposed method is imputation correcting rate. For n missing data, it is defined as 1-mrn, if m data are incorrectly guessed.

C. Weighted voting method

Although we have discussed using association rule mining based method to select one rule for missing data imputation, we also consider that it is possible that for one available rules set, using some rules to imputation missing data will be superior to one rule. So we explore using association rule mining based on weighted voting method to impute missing data. Weighted voting procedure can be described as follows:

Count weight value of each rule in rules set. Suppose total number of rules in set is t, if antecedent length of a rule is al, then the weight value of the rule is al/t.

Group by the same rules' consequent that means if two rules have the same rules' consequent, they will be in the same group.

Total voting results by respectively counting the sum of weight values, sum of confidence values, and sum of support values in each group.

If only one group is the highest score according to the sum of weight values, this group should be selected.

Else in the groups which have the same sum of weight values, if only one group is the highest score according to the sum of confidence values, this group should be selected.

Else in the groups which have the same sum of confidence values, if only one group is the highest score according to the sum of support values, this group should be selected.

Else this means more than one group have the same sum of weight values, sum of confidence values, and sum of support values, then the first rule in set will be selected.

We first consider the rules length in above weighted voting method because the length first strategy enables us to obtain more accurate rules, thus improving the missing data imputation accuracy .

D. Missing Data Imputation Procedure

Once we have obtained the constrained association rules, we can use association rule based on weighted voting to predict missing data. For each case of the test data set, we scan the rules one by one and identify which rules' antecedent is implicated in the case. These identified rules form matching rules set called R_m. If R_{in} is not null, we call the weighted voting procedure which return values are the missing data. Figure 2 is the algorithmic description of the missing data imputation procedure.

Algorithm:

WEIGHTED VOTING METHOD

```
1.  w=c=s=1;
2.  t=total(Rules set); //total number of rules
3.  for each rule ri e Rules set do
4.   wr[i]=1ength_of _antecedent(ri)/t;
5.  endfor  //group by the same consequent of rules
6.  for each rule ri e Rules set do
7.   flag[i]=0;
8.  endfor
9.  k=1;
10. for each rule ri e Rules set do
11.  if (!flag[i]) then
12.     group[k]=group[k] uri; flag[i]=1; 1
13.  endif
14.  for each rule rj>i e Rules set do
15.  if (consequent(ri)- consequent(r)
        && !flag[j]) then
16.     group[k]=group[k] urj; flag[j]=1; 1
17.  endif
18.  endfor
19.  k++;
20. endfor
21. for i=1 to k do
22.  wv[i]=con[i]=sup[i]=0;
23.  for each rule rj e group [i] do
24.   wv[i]=wv[i]+wr[j];
25.   con[i]=con[i]+conf[j];
26.    sup [i]=sup[i]+supp [j];
27.  endfor
28. endfor
29. // w=number_same_weight();
30. for i=0 to k do
31.  for j=i+1 to k do
32.  if (wv[i]==wv[j]) then w++; endif
```

```
33.  endfor
34. endfor
35. if (w==1) then
36.  return consequent(Maxweight group);
37. else
38. { c=number_same_confidence();
39.  if (c=1) then
40.  return consequent(Maxconf group);
41.  else
42. { s=number_same_support();
43.    if (s==1) then return
44.     consequent(Maxsupp_group);
45.    else
46.     return consequent(fristrule);
47.    endif  1
48.  endif  1
49.  endif
```

RULES SET GENERATION

1: for each case T_ie the test data set do
2: R_m=null;
// finding matching rules set R_m
3: for each rule e Rules set do
4: if antecedent(rule)c T_i then
5: R_m= R_m urule;
6: endif
7: endfor
// select imputation data
8: if R_n4null then
9: Imputation data=Weighted voting(R_m);
10: endif
11: endfor

EXPERIMENTS

In this section, we present the experimental results of the missing data imputation using association rule mining based on weighted voting method in some given data sets. Experimental data sets are from UC Irvine Machine Learning Repository. Description of the data sets sees table 1 in details.

Data Set S	Cases	Variable	Missing Data Rate	Source
Monk 1	432	7	None	UCI
Monk 2	601	7	None	UCI
Monk 3	554	7	None	UCI

In the univariate missingness pattern, only one attribute contains missing data. We have used association rule mining based on weighted voting method to do experiments in monkl, monk2, and monk3 data sets. We removed all values of attribute #3 in test data sets. We test the validity of association rules mining by comparing it with WEKA's method of replacing missing values . From the experimental results below we can find that the missing data imputation rates in our method are better than that of WEKA's in monkl and monk3 data sets. Using this method for each attribute with missing data, all missing data can be predicted with a certain confidence level. This seems to show that our method can be used as a method to impute missing data.

WEKA's method of Replacing missing values

1] For Nominal Data WEKA replaces the missing values by the mode ie., the highest occurring attribute value

2] For Numeric Data WEKA replaces the missing values by the mean ie., the average of all the values taken by that attribute.

3] If there are multiple missing values in one attribute they are all replaced by the same value ie., mode for nominal and mean for numeric data. WEKA does not consider the association of the missing attribute with the other attributes of that tuple.

Correct missing values	confidence	Imputation correcting rate	remark
169/432	0.1	39.12%	All RMs generated
175/432	0.3	40.5%	All RMs generated
191/432	0.5	44.2%	All RMs generated
216/432	0.6	50%	416 RMs generated
177/432	0.7	40.9%	226 RMs generated
WEKA	**144/432**	**33.33%**	

Correct missing values	Confidence	Imputation correcting rate	remark
144/432	0.1	33.33%	All RMs generated
144/432	0.3	33.33%	All RMs generated
148/432	0.5	34.25%	430 RMs generated
146/432	0.6	33.8%	173 RMs generated

WEKA	**144/432**	**33.33%**

HANDLING NOISY DATA

Noise is “irrelevant or meaningless data”. For most existing data cleaning methods, the focus is on the detection and removal of noise (low-level data errors) that is the result of an imperfect data collection process. The need to address this type of noise is clear as it is detrimental to almost any kind of data analysis. However, ordinary data objects that are irrelevant or only weakly relevant to a particular data analysis can also significantly hinder the data analysis, and thus these objects should also be considered as noise, at least in the context of a specific analysis. For instance, in document data sets that consist of news stories, there are many stories that are only weakly related to the other news stories. If the goal is to use clustering to find the strong topics in a set of documents, then the analysis will suffer unless irrelevant and weakly relevant documents can be eliminated. Consequently, there is a need for data cleaning techniques that remove both types of noise.
We have made use of a hyperclique patterns as a filter to eliminate data objects that are not tightly connected to other data objects in the data set.

HYPERCLIQUE PATTERNS

A hyperclique pattern is a new type of association pattern that contains objects that are highly affiliated with each other. Unlike frequent patterns, a hyperclique pattern contains items that are strongly correlated with each other. Indeed, the presence of an item in one transaction strongly implies the presence of every other item that belongs to the same hyperclique pattern. The h-confidence measure is specifically designed to capture the strength of this association.

The h-confidence of a pattern $X = \{i1, i2, \cdots, im\}$, denoted as $hconf(X)$, is a measure that reflects the overall affinity among items within the pattern. This measure is defined as $\min(conf(\{i1\} \rightarrow \{i2, \ldots, im\}), conf(\{i2\} \rightarrow \{i1, i3, \ldots, im\}), \ldots, conf(\{im\} \rightarrow \{i1, \ldots, im-1\}))$,
where conf is the confidence of association rule as given above.

Example 1: For the sample transaction data set shown in Table I, let us consider a pattern $X = \{i2, i3, i4\}$. We have $supp(\{i2\}) = 80\%$, $supp(\{i3\}) = 80\%$, $supp(\{i4\}) = 60\%$, and $supp(\{i2, i3, i4\}) = 40\%$. Then,
$conf(\{i2\} \rightarrow \{i3, i4\}) = supp(\{i2, i3, i4\})/supp(\{i2\}) = 50\%$
$conf(\{i3\} \rightarrow \{i2, i4\}) = supp(\{i2, i3, i4\})/supp(\{i3\}) = 50\%$
$conf(\{i4\} \rightarrow \{i2, i3\}) = supp(\{i2, i3, i4\})/supp(\{i4\}) = 66.7\%$
So, $hconf(X) = \min(conf(\{i2\} \rightarrow \{i3, i4\}), conf(\{i3\} \rightarrow \{i2, i4\}), conf(\{i4\} \rightarrow \{i2, i3\})) = 50\%$.

ANTIMONOTONE PROPERTY OF H-CONFIDENCE

If the h-confidence of an itemset *P* is greater than a user-specified threshold, then the same applies for every subset of *P*. h_conf is monotonically non-increasing as the size of the hyperclique pattern increases. Such a property allows us to push the h-confidence constraint directly into the mining algorithm. Specifically, when searching for hyperclique patterns, the

algorithm eliminates every candidate pattern of size *m* having at least one subset of size *m ¡* 1 that is not a hyperclique pattern.

CROSS-SUPPORT PROPERTY

Cross Support Patterns: Given a threshold t, a pattern P is a cross-support pattern with respect to t if P contains two items x and y such that supp(x) / supp(y) < t, where 0 < t < 1.

This property can be used to efficiently eliminate cross- support patterns. Also, we show that the cross-support property is not limited to h-confidence and can be generalized to some other association measures. Finally, a sufficient condition is provided for verifying whether a measure satisfies the cross-support property or not.

THE STRONG AFFINITY PROPERTY

The strong affinity property guarantees that if a hyperclique pattern has an h-confidence value above the minimum h-confidence threshold, hc, then every pair of items within the hyperclique pattern must have a cosine similarity greater than or equal to hc. As a result, the overall affinity of hyperclique patterns can be controlled by properly setting an h-confidence threshold.

ALGORITHM

We first derive all size-3 hyperclique patterns at a given h-confidence threshold hc from the training set. The noise objects are simply those which are not a member of any of these hyperclique patterns. In other words, for any identified noise object, we cannot find two other objects which have pairwise cosine similarity with this object above the h-confidence threshold, hc. Indeed, the h-confidence threshold specifies the fraction of noise data objects. If we fix the support threshold, a higher h-confidence threshold means that more objects will be labeled as noise. Therefore, the noise percentage increases as the h-confidence threshold increases.

1] Generate size 3 hyperclique patterns from training set.
2] For a given tuple, mark all attributes as false
3] If hyperclique pattern is a subset of tuple
4] Mark the subset in the tuple as true
5] Repeat steps 3 to 4 for all hyperclique patterns
6] If tuple contains false attribute
    Attribute = noise
7] Call Weighted Voting Method to predict correct value of noise
8] Repeat steps 2 to 7 for all tuples.