An Equivalence Class Based Clustering Algorithm for Categorical Data

Liu Qingbao, Wang Wanjun, Deng Su
Sci. and Technol. on Inf. Syst. Eng. Laboratory
National University of Defense Technology
Changsha, China, 410073
e-mail: liuqingbao@nudt.edu.cn,
wwjunbelieve@gmail.com, Sudeng@sohu.com

Guozhu Dong
Department of Computer Science & Engineering
Wright State University
Dayton, Ohio, USA 45435
e-mail: guozhu.dong@wright.edu

***Abstract*—Categorical data clustering plays an important role in data mining for many applications, including popular applications involving text mining and blog mining. While most traditional clustering methods rely on a distance function. However, the distance between categorical data is hard to define, especially for exploratory situations where the data is not well understood. As a result, many clustering methods do not perform well on categorical datasets. In this paper we propose a novel Equivalence Class based Clustering Algorithm for Categorical data (ECCC). ECCC takes the support transaction sets of selected frequent closed patterns as the candidate clusters. We define a novel quality measure to evaluate the suitability of frequent closed patterns to form the clusters; the measure is based on two factors: cluster coherence expressed in terms of closed patterns, and cluster discriminativeness expressed in terms of quality and diversity of minimal generator patterns. ECCC uses that measure to select the high quality frequent closed patterns to form the final clusters.**

Keywords-clustering analysis; categorical data; equivalence class

I. INTRODUCTION

Clustering is unsupervised and highly explorative. It is an important approach, widely used in life science, medicine, social science, engineering and many other fields [1]. Most traditional clustering methods rely on a distance function. Since the distance between categorical data is hard to define, especially for exploratory situations where the data is not well understood, many clustering methods do not work well on categorical datasets.

Several clustering methods have been recently developed to handle categorical data, including k-ANMI introduced in [2], Squeezer proposed in [3], GAClust in [4], ccdByEnsemble in [5], the Entropy-based algorithm in [6], and ECCLAT in [7]. However, these methods have various shortcomings; for example, the Entropy-based algorithm emphasizes intra-cluster purity while ignoring inter-cluster separation.

Here we detail the ECCLAT algorithm, which we will compare our method with. Firstly, it is necessary to explain the term "frequent closed itemset" in a general manner: a closed itemset is a maximal set of items shared by a set of transactions; when the frequency of a closed itemset is larger than the frequency threshold (denoted as: *minfr*), it is called a frequent closed itemset. ECCLAT extracts a subset of concepts from the lattice of frequent closed itemsets, using the evaluation measure *interestingness(X) = (homogeneity(X) + concentration(X))/2* [7] (*X* is an itemset). More specifically, ECCLAT first mines the set of frequent closed itemsets; it views the support transaction set of each frequent closed itemset as a candidate cluster. Then, it computes the interestingness of the candidate clusters, and iteratively selects the next candidate cluster having the highest interestingness as the next final cluster. ECCLAT is an approximate clustering algorithm and allows the clusters to have transaction overlap. The *concentration(X)* measure is defined to limit the overlap of transactions between clusters by taking into account the number of candidate clusters where each transaction of *X* appears. Checking the definition of *concentration(X)* in [7], we can see that it treats all candidate clusters as equally important. Such weaknesses of ECCLAT lead to poor clustering results.

In this paper, we propose a novel method called ECCC (an Equivalence Class based Clustering Algorithm for Categorical data). The main contribution of the paper is to provide a better quality measure to replace the concentration quality measure of ECCLAT. Our quality measure, called inter-cluster discriminativeness index, will consider the quality and diversity/richness of the minimal generator patterns. Specifically, our algorithm first mines the equivalence classes [9] of patterns, including the closed patterns and their associated sets of generator patterns. Similarly to ECCLAT, we also regard the support transaction set of each closed pattern as a candidate cluster. We combine the intra-cluster homogeneity index of ECCLAT and our inter-cluster discriminativeness index into a general and objective quality index on clusters. Our algorithm then selects the high quality clusters from the candidate clusters using that quality index.

Compared against ECCLAT, our ECCC uses both the closed pattern and the generator patterns, instead of just the closed patterns, of equivalence class to define the *discriminativeness index.* ECCC prefers the equivalence classes that have a long closed pattern and many short generator patterns. The first advantage of ECCC is that it avoids the drawback of ECCLAT mentioned above. The second advantage is that there is no transaction overlap among the final clusters. The third advantage is that ECCC needs only one parameter (*minfr)* while ECCLAT needs two parameters (*minfr* and *M* [7]*).* As a result, ECCC is more accurate in recovering expert defined classes than ECCLAT.

In Section II, we describe our discriminativeness index and ECCC algorithm, after giving relevant preliminaries. In Section III, we report experimental results. Our conclusions are presented in Section IV.

Supported by the National Natural Science Fund of China under Grant No.70771110.

 ISBN: 978-1-61208-162-5

II. ECCC

In this section we will present our method, namely ECCC, after firstly giving some definitions.

A. Preliminaries

We assume that we are given a dataset D (a set of transactions) in the following definitions. For each itemset z, let $f_D(z)=\{t \in D | z \subseteq t\}$ denote the support set of transactions for z.

Definition 1 (*EC: Equivalence Class***)** [9]. An equivalence class *EC* is a (maximal) set of frequent itemsets (also called frequent patterns) that have a common support set of transactions.

So, if *EC* is an equivalence class and x and $y \in EC$, then $f_D(x)=f_D(y)$.

Here we give the definition of "frequent closed itemset" which is called "closed pattern" in our ECCC.

Definition 2 (*cp: closed pattern***)** [9]. Given an equivalence class *EC*, the closed pattern *cp* of *EC* is

$$cp = \bigcup_{p \in EC} p \,. \tag{1}$$

Definition 3 (*gp: generator pattern***)** [9]. Given an equivalence class *EC*, a pattern $gp \in EC$ is a generator pattern of *EC* if, for $\forall\ z \in EC$ *s.t.* $z \neq gp$, it is the case that $z \not\subset gp$.

It is well known that an equivalence class has only one closed pattern and it has one or more generator patterns. We will represent an equivalence class *EC* as *EC*=[*G(cp),cp*] [9], where $G(cp)=\{gp_i | 1 \le i \le k\}$ which is the set of generator patterns and cp is the closed pattern of EC.

Definition 4 (*candidate cluster***)**. A set of transactions $CC \subseteq D$ satisfying $CC=f_D(cp)$ for some closed pattern cp is called a candidate cluster associated with *cp*; this CC will be denoted by *CC(cp)*.

B. Homogenity and Discriminativeness Measures

There are often many candidate clusters, and only a few candidate clusters can become the final clusters. For example, with *minfr* = 5%, there are 9738 candidate clusters in the mushroom dataset. So we need quality measures to select the candidate clusters as final clusters. Our algorithm will use one factor's formula used by ECCLAT, and replaces the other factor's formula using a new one.

Definition 5 (*HI: Homogeneity Index***)**. The Homogeneity Index [7] of a candidate cluster *CC(cp)* is defined by:

$$HI_{CC}(cp) = \frac{|CC(cp)| \times |cp|}{divergence(cp) + |CC(cp)| \times |cp|} \tag{2}$$

where $divergence(cp) = \sum_{t \in f_D(cp)} |t - cp|$, and |S| denotes the cardinality of a set S.

Homogeneity Index is used to measure the intra-cluster similarity. Larger values are better. Using this index, we prefer those candidate clusters whose closed patterns are very long. If a candidate cluster *CC(cp)* has a very long closed pattern *cp*, then all the transactions in *CC(cp)* share all items in *cp*, implying that *CC(cp)* is highly coherent; we note that *divergence(cp)* is small and $HI_{CC}(cp)$ is large in this situation.

For the inter-cluster diversity, we propose a novel measure called *Discriminativeness Index* which is defined below.

Definition 6 (*DI: Discriminativeness Index***)**. The *discriminativeness index* of a candidate cluster *CC(cp)* is defined as:

$$DI_{CC}(cp) = \prod_{gp_i \in G(cp)} (1 + \frac{|cp - gp_i|}{|cp|}) \tag{3}$$

where $|cp|$ and $|cp\text{-}gp_i|$ are the number of items in *cp* and $cp\text{-}gp_i$, respectively.

Larger $DI_{CC}(cp)$ values are better. Using this *Discriminativeness Index*, we prefer the candidate cluster which has a very long closed pattern and many short generator patterns. Our rationale for *Discriminativeness Index* is similar to that in [10]. If a candidate cluster *CC(cp)* has *many short* generator patterns, then each such short generator pattern *gp* is a strong discriminator that can be used to easily separate and distinguish *CC(cp)* from the other candidate clusters. The shorter gp is the easier it is to do the separation. The more such short gp patterns we have, the more different ways we have to describe the cluster and discriminate it from other clusters. So we think the high $DI_{CC}(cp)$ value implies that this candidate cluster *CC(cp)* is significantly different from other candidate clusters, is identified very easily, and has better quality.

Definition 7 (*QI: Quality Index***)**. The EC based Quality Index of the candidate cluster *CC(cp)* is defined as follows:

$$QI_{CC}(cp) = HI_{CC}(cp) \times DI_{CC}(cp) \,. \tag{4}$$

Our idea is to select these candidate clusters with high *Quality Index* as the final clusters. The next section presents an algorithm for this task.

C. The Process of ECCC

On dataset *D*, we first use DPMiner algorithm [9] to mine the closed patterns and their generators simultaneously, using a minimal frequency threshold *minfr*. Then, we determine the candidate clusters of the frequent closed patterns, calculate the quality of each candidate cluster, and select the candidate cluster *CC(cp*)* with highest quality as a final cluster *C(cp*)*. When there are two and more highest quality candidate clusters, we prefer the candidate cluster with larger number of transactions. For any remaining candidate cluster *CC(cp)* such that $cp \neq cp^*$ and $CC(cp) \bigcap C(cp^*) \neq \varnothing$, we modify the candidate cluster *CC(cp)* as *CC(cp)=CC(cp)-C(cp*)*. If $|CC(cp)| < minfr$, we delete the candidate cluster *CC(cp)*. Then we recalculate $HI_{CC}(cp)$, $DI_{CC}(cp)$ and $QI_{CC}(cp)$ of the

 ISBN: 978-1-61208-162-5

candidate clusters, and select the candidate cluster *CC(cp*)* with highest quality as the next final cluster. We repeat the process above, until there is no candidate cluster. At the end, we classify all remaining transactions into the trash set.

The *pseudo-codes* of ECCC are given below.

Input:
D is a dataset to be clustered;
minfr is the frequency threshold;
Output:
CL is the set of Clusters;
Trash is the set of the trash transactions;
Description:

1. mine $CP=\{cp_k|1\le k\le N\}$, $G(cp_k)$, $CC(cp_k)$;
2. **for** each $cp\in CP$ **do**
3. calculate $HI_{CC}(cp)$ *and* $DI_{CC}(cp)$;
4. $QI_{CC}(cp)= HI_{CC}(cp)\times DI_{CC}(cp)$;
5. **end for**
6. select $CC(cp^*)$, *s.t.* $QI_{CC}(cp^*)= \max_{cp_k\in CP}\{QI_{CC}(cp_k)\}$;
7. $C(cp^*)= CC(cp^*)$;
8. delete $CC(cp^*)$;
9. insert $C(cp^*)$ into *CL*;
10. **for** each $CC(cp) \wedge (CC(cp)\cap C(cp^*)\neq \varnothing)$ **do**
11. $CC(cp)=CC(cp)-C(cp^*)$;
12. **if** $|CC(cp)|<minfr$ **then**
13. delete $CC(cp)$;
14. **else**
15. recalculate $HI_{CC}(cp)$, $DI_{CC}(cp)$ and $QI_{CC}(cp)$;
16. **end if**
17. **end for**
18. repeat *steps 6--17* until there is no candidate cluster;
19. classify the remaining transactions of *D* into the *Trash*;
20. return *CL* and *Trash*;

III. EXPERIMENT RESULTS

We now use experiments to demonstrate that (1) the ECCC algorithm is accurate and (2) the ECCC algorithm is scalable.

Experiments were conducted on a desktop computer with a 2.33 GHz Intel CPU and 3 GB memory running the Windows XP.

A. Accuracy test on Mushroom Dataset and Zoo Dataset

We evaluate the accuracy of our algorithm ECCC on two real datasets available at the UCI Repository [8]. One is the well known mushroom dataset which has 22 attributes, 8124 transactions, and two class labels provided by domain experts. The other is the zoo database (101 transactions with 7 class labels provided by domain experts) which has 15 boolean attributes and a numerical one ("legs"), we used the six values of "legs" as categorical values.

1) Error rates test on mushroom dataset

In this section, we present experiment results of ECCC on the mushroom dataset.

a) Compare against ECCLAT algorithm

For *minfr* = 5% and *M* = *minfr*, ECCLAT obtains 16 clusters and a trash cluster with slight overlapping between clusters 14 and 16 [7]. Also ECCC can obtain 16 clusters and a trash cluster without overlapping when *minfr* = 4%.

The comparison between the above two clusterings is shown in Table I. It is obvious that the ECCC clustering errors are lower than that of the ECCLAT clustering.

TABLE I. COMPARISON BETWEEN ECCC AND ECCLAT

Cluster No.	ECCC (*minfr* = 4%)		ECCLAT (M=*minfr* = 5%)	
	#Poisonous	***#Edible***	***# Poisonous***	***# Edible***
1	0	576	0	432
2	432	0	0	432
3	0	384	0	432
4	0	384	0	432
5	864	0	648	0
6	576	0	648	0
7	576	0	432	0
8	576	0	432	0
9	0	400	432	0
10	0	400	432	0
11	272	***96***	0	768
12	***72***	528	0	512
13	***128***	384	352	***96***
14	0	384	***288***	896
15	***144***	384	0	416
16	240	***96***	***72***	560
Trash	***36***	192	180	***160***
Error	***572***		***616***	

b) Average clustering error rates comparison

To further test the clustering errors of ECCC, we repeated ECCC with 10 different *minfr*s from 1% to 10% on mushroom dataset. The 10 results are shown in Table II.

TABLE II. RESULTS IN TERM OF THE DIFFERENT *MINFRS*

***minfr*(%)**	**#Clusters (including the trash cluster)**	**# Errors**
1	28+1	252
2	24+1	252
3	20+1	172
4	16+1	572
5	12+1	890
6	11+1	890
7	11+1	890
8	6+1	890
9	6+1	890
10	5+1	890

 ISBN: 978-1-61208-162-5

Table II shows that the clustering errors change with different *minfr*s. For *minfr* =3%, the result is the best. And for *minfr* =4%, the result is the middle. But when *minfr* changes from 5% to 10%, the errors do not change. So we can think the ECCC is a stable clustering algorithm.

In addition, we used the EM algorithm [11] which is implemented in WEKA [12] to generate clustering for comparison against the ECCC. Results are given in Table III.

TABLE III. AVERAGE CLUSTERING ERROR RATES COMPARISON

Algorithm	Average Clustering Error Rates
ECCC	0.081
EM	0.133
k-ANMI	0.165
ccdByEnsemble	0.315
GAClust	0.393
Squeezer	0.206

Table III indicates that the average clustering error rate in Table II is lower than that of EM and the algorithm of [2].

2) Purity test on zoo dataset

We now report experiments on the zoo database, to demonstrate that ECCC is accurate with high purity, and to compare it against Entropy-based algorithm and K-means algorithm. Table IV indicates that our ECCC is better than Entropy-based algorithm and K-means algorithm on purity. (The purity of a clustering $(C_1,\ldots,C_m)$ against an expert given clustering $(C'_1,\ldots,C'_k)$ is defined as follows: For each cluster C_i, let C'_{i*} denote the expert cluster with the largest overlap with C_i. Purity of C_i is defined as $|C_i \bigcap C'_{i*}| / |C_i|$. The purity of the clustering $(C_1,\ldots,C_m)$ is defined as the weighted average of the purity of the clusters.)

TABLE IV. RESULTS COMPARISON ON ZOO DATASET

	ECCC	Entropy-based	K-means
Purity	0.9208	0.9000	0.8400

In summary, these experimental results on both mushroom dataset and zoo dataset demonstrate the accuracy and stability of the ECCC algorithm.

B. Scalability Test

The purpose of this experiment is to test the scalability of the ECCC algorithm when the sizes of the datasets increase. We picked the first 1K, 2K, 3K, 4K, 5K, 6K, 7K and 8K records respectively from the mushroom dataset to form 8 testing datasets. Figure 1 shows the run time of ECCC testing on the 8 datasets.

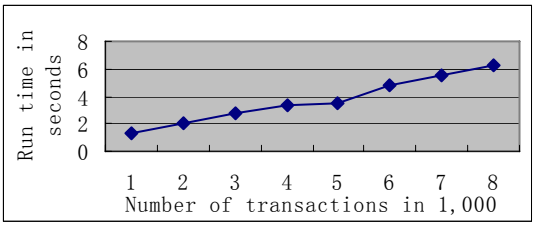

Figure 1. time vs. number of transactions

From Figure 1, it is easy to see that the computation cost increases linearly in terms of the number of transactions, which is highly desired in real data mining applications.

IV. CONCLUSION

In this paper, we gave a better quality index (especially the *Discriminativeness Index)* based on equivalence classes of frequent patterns, and proposed an equivalence class based clustering algorithm (ECCC) for categorical data. ECCC mines clusters with high intra-cluster similarity, strong inter-cluster diversity and no cross-cluster overlap. The experiment results showed that our method is accurate, stable and scalable on the real datasets.

For future work, we will find a good way to merge some clusters for reducing the number of clusters according to user's requirement.

ACKNOWLEDGMENT

Thanks go to the authors of [9] for the C++ code of DPMiner.

REFERENCES

[1] Weining Qian and Aoying Zhou. Analyzing Popular Clustering Algorithms from Different Viewpoints. Journal of Software, 13(8): 1382-1394, 2002.

[2] Z. He, X. Xu and S. Deng. K-ANMI: A mutual information based clustering algorithm for categorical data. Information Fusion, 9:223-233, 2008.

[3] Z. He, X. Xu and S. Deng. Squeezer: an efficient algorithm for clustering categorical data. Journal of Computer Science & Technology, 17(5): 611-624, 2002.

[4] D. Cristofor and D. Simovici. Finding median partitions using information-theoretical-based genetic algorithms. Journal of Universal Computer Science, 8(2): 153-172, 2002.

[5] Z. He, X. Xu and S. Deng. A cluster ensemble method for clustering categorical data. Information Fusion, 6(2): 143-151, 2005.

[6] T. Li, S. Ma and M. Ogihara. Entropy-based criterion in categorical clustering. ICML, 2004.

[7] Nicolas Durand and Bruno Cr´emilleux. ECCLAT: a New Approach of Clusters Discovery in Categorical Data. 22nd SGAI International Conference on Knowledge Based Systems, December 2002.

[8] http://archive.ics.uci.edu/ml/datasets.html, July 2011.

[9] J. Li, G. Liu and L. Wong. Mining Statistically Important Equivalence Classes and Delta-Discriminative Emerging Patterns. KDD, 2007.

[10] Qingbao Liu and Guozhu Dong. A Contrast Pattern based Clustering Quality Index for Categorical Data. IEEE ICDM, 2009.

[11] G. McLachlan and T.Krishnan. The EM algorithm and extensions. Journal of Classification, 15(1): 154-156, 1997.

[12] www.cs.waikato.ac.nz/ml/weka/, August 2010.

 ISBN: 978-1-61208-162-5