UDC 004.421.2

Improving Categorical Data Clustering Algorithm by Weighting Uncommon Attribute Value Matches

Zengyou He, Xiaofei Xu, Shenchun Deng

Department of Computer Science and Engineering,
Harbin Institute of Technology,
92 West Dazhi Street, P.O Box 315, China, 150001
zengyouhe@yahoo.com, {xiaofei,dsc} @hit.edu.cn

Abstract. This paper presents an improved *Squeezer* algorithm for categorical data clustering by giving greater weight to uncommon attribute value matches in similarity computations. Experimental results on real life datasets show that, the modified algorithm is superior to the original *Squeezer* algorithm and other clustering algorithm with respect to clustering accuracy.

1. Introduction

Clustering typically groups data into sets in such a way that the intra-cluster similarity is maximized while the inter-cluster similarity is minimized. The clustering technique has been extensively studied in many fields such as data mining, pattern recognition, customer segmentation, similarity search and trend analysis.

Most previous clustering algorithms focus on numerical data whose inherent geometric properties can be exploited naturally to define distance functions between data points. However, much of the data existed in the databases is categorical, where attribute values can't be naturally ordered as numerical values. An example of categorical attribute is *shape* whose values include *circle*, *rectangle*, *ellipse*, etc. Due to the special properties of categorical attributes, the clustering of categorical data seems more complicated than that of numerical data.

A few algorithms have been proposed in recent years for clustering categorical data (e.g., [1-8]). For detailed descriptions and discussions about these algorithms, the readers may refer to [2]. It is sufficient to point out that all the attribute values are treated equally in the similarity computations of those existing algorithms. That is, uncommon attribute value matches have the same contribution to similarity computation like those of frequent ones.

In this paper, we make an attempt to improve the clustering accuracies of existing categorical data clustering algorithms by giving greater weight to uncommon attribute value matches in similarity computations. More precisely, *Squeezer* algorithm [1] is selected as a representative of categorical data clustering algorithms for this purpose. Experimental results show that, the modified *Squeezer* algorithm is superior to the original *Squeezer* algorithm with respect to clustering accuracy. Therefore, we believe that other categorical data clustering algorithms [2-8] can also be improved in a similar way by weighting uncommon attribute value matches.

2. Weighting Uncommon Attribute Value Matches

In this paper, we employ the approach proposed by Goodall [9] for weighting uncommon attribute value matches. The Goodall measure was first proposed for biological and genetic taxonomy problems, where unusual characteristics shared by biological entities is often attributed to closely related genetic information resulting in these entities being classified into the same species [9]. Li and Biswas [10] have extended it to clustering problems in more general domains. Therefore, we can adopt the method given by Li and Biswas [10].

A pair of objects (i, j) is considered more similar than a second pair of objects (l, m), if and only if the objects i and j exhibit a greater match in attribute values that are less common in the population. In other words, similarity among objects is decided by the un-commonality of their attribute value matches. Similarity computed using the heuristic of weighting uncommon attribute value matches helps to define more cohesive, tight clusters where objects grouped into the same cluster are likely to share special and characteristic attribute values. One should note that common attributes values also play an important role in the similarity computation and in the clustering process. The similarity computation is realized by weighting attribute value matches between a pair of objects by the frequency of occurrence of the attribute value in the dataset [10].

Let $A_1, \ldots, A_m$ be a set of categorical attributes with domains $D_1, \ldots, D_m$ respectively. Let the dataset D be a set of objects where each object t: $t \in D_1 \times \ldots \times D_m$. We use $VAL_1, \ldots, VAL_m$ to denote the sets of distinct attribute values for $A_1, \ldots, A_m$ on the dataset D. For each $v_{ij} \in VAL_i$, $f(v_{ij})$ is used to denote its frequency of occurrence in D. Then, the *More Similar Attribute Value Set* of v_{ij} is defined as:

$$MSFVS(v_{ij}) = \{v_{ik} \mid f(v_{ik}) \le f(v_{ij}) \text{ and } v_{ik} \in VAL_i\} \tag{1}$$

This is the set of attribute values with lower or equal frequencies of occurrence than that of v_{ij}. Note that a value pair is more similar if it has lower frequency of occurrence. The *weight* of attribute value v_{ij} is defined as:

$$W(v_{ij}) = 1 - \sum_{v_{ik} \in MSFVS(v_{ij})} \frac{f(v_{ik})(f(v_{ik})-1)}{n(n-1)} \tag{2}$$

where *n* is the total number of objects in the dataset. By the giving weight to each attribute value, the uncommon value matches in similarity computation will make more contribution to similarity values.

Taking an example used in [10] to illustrate the computation. Table 1 shows a table with 10 objects, each described by a categorical attribute. In the given dataset, *f* (*a*) = 3, *f* (*b*) = 3 and *f* (*c*) = 4, i.e., values *a* and *b* are less frequent than value *c*. The *MSFVS* for attribute value *c* is:

MSFVS (*c*) = {*a*, *b*, *c*}.

Given the *MSFVS*, the *weight* of attribute value *c* is calculated by (2):

$W(c) = 1 - (\frac{3(3-1)}{10(10-1)} + \frac{3(3-1)}{10(10-1)} + \frac{4(4-1)}{10(10-1)}) = 1\text{-}0.276 = 0.733.$

Similarly, the *MSFVS* for attribute value *a* is:

MSFVS (*a*) = {*a*, *b*}.

Hence, $W(a) = 1 - (\frac{3(3-1)}{10(10-1)} + \frac{3(3-1)}{10(10-1)}) = 0.867.$

Therefore, attribute 1 contributes a value of 0.733 to the similarity between objects when attribute value *c* is matched. Similarly, attribute 1 contributes a value of 0.867 to the similarity between objects if attribute value *a* is matched. From this viewpoint, objects 1 and 4 are more similar than objects 5 and 6 with respect to attribute 1.

Table 1. Sample Data Set

Object ID	Attribute 1
1	A
2	B
3	B
4	A
5	C
6	C
7	C
8	C
9	A
10	B

3. Uncommon Attribute Value Biased *Squeezer* Algorithm: *NabSqueezer*

In this section, we describe the improved *NabSqueezer* algorithm, which provides greater weight to uncommon attribute value matches in similarity computations of *Squeezer* algorithm.

Let $A_1, \ldots, A_m$ be a set of categorical attributes with domains $D_1, \ldots, D_m$ respectively. Let the dataset *D* be a set of objects where each object *t*: $t \in D_1 \times \ldots \times D_m$. Let *TID* be the set of unique identifier of every object. For each *tid* $\in$ *TID*, the attribute value for A_i of corresponding object is represented as $tid.A_i$.

Definition 1: (*Cluster*) *Cluster* $\subseteq$ *TID* is a subset of *TID*.

Definition 2: Given a *Cluster C*, the set of different attribute values on A_i with respect to *C* is defined as: $VAL_i(C) = \{ tid.A_i \mid tid \in C \}$.

Definition 3: Given a *Cluster C*, let $a_i \in D_i$, the frequency of a_i in *C* with respect to A_i is defined as: $f(a_i,C) = |\{tid \mid tid.A_i = a_i, tid \in C\}|$.

Definition 4: (*Summary*) Given a *Cluster C*, the *Summary* for *C* is defined as:

Summary = $\{VS_i \mid 1 \le i \le m\}$ where $VS_i = \{(a_j, f(a_j,C)) \mid a_j \in VAL_i(C)\}$.

Intuitively, the *Summary* of a *Cluster* contains summary information about this *Cluster*. In general, each *Summary* will consists of *m* elements, where *m* is number of attributes. The element in *Summary* is the set of pairs of attribute values and their corresponding frequency values. We will show later that information contained in *Summary* is sufficient enough to compute the similarity between an object and a *Cluster*.

Definition 5: (*Cluster Structure, CS*) Given a *Cluster C*, the *Cluster Structure (CS)* for *C* is defined as: *CS* = {*Cluster, Summary*}.

Definition 6: Given a *Cluster C* and an object *t* with *tid* $\in$ *TID*, the *similarity* between *C* and *tid* is defined as:

$Sim(C, tid) = \sum_{i=1}^{m} \left(\frac{W(a_i) * f(a_i, C)}{|C|} \right)$ where $a_i = tid.A_i$ and $W(a_i)$ is the *weight* of attribute value a_i.

From the definition 6, it is clear that the similarity used is capable of handling clustering problem with different weights on different attribute values. The weight of each attribute value is pre-computed using the approach discussed in Section 2.

The *NabSqueezer* algorithm is presented in Fig.1. It accepts as input the dataset *D* and the value of the desired similarity threshold *s*. The algorithm can be divided into two stages according to its two scans over the dataset.

In the first pass over the dataset, the frequency of occurrence of each attribute value is counted (Step 1-2). Then, the *MSFVS* for each attribute value is calculated (Step 3) and the weight of each attribute value is also derived (Step 4).

In the second pass over the dataset, initially, the first object is read in, and the sub-function *addNewClusterStructure*() is used to establish a new *Clustering Structure*, which includes *Summary* and *Cluster* (Step 7-8). For the consequent objects, the similarity between each existing *Cluster C* and the object is computed using sub-function *simComputation*()(Step 10-11). Consequently, we get the maximal value of similarity (denoted by *sim_max*) and the corresponding index of *Cluster* (denoted by *index*) (Step 12-13). Then, if the *sim_max* is larger than the input threshold *s*, the sub-function *addObjectToCluster*() will be called to assign the object to the *Cluster* that achieved best similarity value(Step 14-15). If it is not the case, the sub-function *addNewClusterStructure()* will be called to construct a new *CS* (Step 16-17). Finally, the clustering results will be labeled on the disk (Step 19).

```
Algorithm NabSqueezer (D, s)
Begin
// The first scan over the dataset
1.  while (D has unread object){
2.      for each attribute value, update its frequency value}
3.  Computing the MSFVS of each attribute value
4.  Computing the weight of each attribute value
// The second scan over the dataset
5.  while (D has unread object){
6.      object = getCurrentObject (D)
7.      if (object.tid = = 1){
8.          addNewClusterStructure (object.tid) }
9.      else{
10.         for each existed cluster C
11.             simComputation (C, object)
12.         get the max value of similarity: sim_max
13.         get the corresponding Cluster Index: index
14.         if sim_max >= s
15.             addObjectToCluster (object, index)
16.         else
17.             addNewClusterStructure (object.tid) }
18. }
19. outputClusteringResults ()
End
```

Fig. 1. *NabSqueezer* Algorithm

4. Experimental Results

A comprehensive performance study has been conducted to evaluate the *NabSqueezer* algorithm. In this section, we describe those experiments and their results. We ran our algorithm on real-life datasets obtained from the UCI

Machine Learning Repository [11] to test its clustering performance against the *Squeezer* algorithm and *k*-modes algorithm.

4.1. Real Life Datasets and Evaluation Method

We experimented with three real-life datasets: Soybean Disease dataset, Mushroom dataset and Wisconsin Breast Cancer dataset, which were obtained from the UCI Repository [11]. Now we will give a brief introduction about these three datasets.

Soybean Disease Data: The first data set was the soybean disease data set, which has frequently been used to test categorical clustering algorithms. The soybean data set has 47 instances, each being described by 35 attributes. Each instance is labeled as one of the four diseases: Diaporthe Stem Canker, Charcoal Rot, Rhizoctonia Root Rot, and Phytophthora Rot. Except for Phytophthora Rot that has 17 instances, all other diseases have 10 instances each.

Mushroom Data: The mushroom dataset has 22 attributes with 8124 instances. Each instance represents physical characteristics of a single mushroom. A classification label of poisonous or edible is provided with each instance. The numbers of edible and poisonous mushrooms in the dataset are 4208 and 3916, respectively.

Wisconsin Breast Cancer Data[1]: It has 699 instances with 9 attributes. Each instance is labeled as *benign* (458 or 65.5%) or *malignant* (241 or 34.5%). In this paper, all attributes are considered to be categorical.

Validating clustering results is a non-trivial task. In the presence of true labels, as in the case of the data sets we used, the clustering accuracy for measuring the clustering results was computed as follows. Given the final number of clusters, *k*, clustering accuracy was defined as $r = \frac{\sum_{i=1}^{k} a_i}{n}$, where n is the number of objects in the dataset and a_i is the number of objects with the class label that dominates cluster *i*. Consequently, the clustering error is defined as $1-r$.

For instance, suppose that the mushroom dataset is clustered into 3 clusters, where cluster 1 has 2807 objects (poisonous: 580, edible: 2227), cluster 2 has 2807 objects (poisonous: 435, edible: 1965) and cluster 3 has 2807 objects (poisonous: 2901, edible: 16). In this example, class label “edible” dominates both cluster 1 and 2, while class label “poisonous” dominates cluster 3. Hence, $r = \frac{2227+1965+2901}{8124} = 0.873$ and e=0.127.

[1] We use a dataset that is slightly different from its original format in UCI Machine Learning Repository, which has 683 instances with 444 benign records and 239 malignant records. It is public available at: http://research.cmis.csiro.au/rohanb/outliers/breast-cancer/brcancerall.dat.

The intuition behind clustering error defined above is that clusterings with "pure" clusters, i.e., clusters in which all objects have the same class label, are preferable. That is, if a partition has clustering error equal to 0 it means that it contains only pure clusters. These kinds of clusters are also interesting from a practical perspective. Hence, we can conclude that smaller clustering error means better clustering results in real world applications.

4.2. Clustering Results

We studied the clustering results found by three algorithms, standard *k*-modes algorithm [7], *Squeezer* algorithm [21] and *NabSqueezer* algorithm. We let these algorithms to cluster the each dataset into different number of clusters, varying from 2 to 9. For each fixed number of clusters, the clustering errors of different algorithms were compared.

In *k*-modes clustering algorithm, initial *k* modes are constructed by selecting the first *k* objects from the dataset. That is, we use one run to get the clustering outputs for *k*-modes.

Since *Squeezer* algorithm and *NabSqueezer* algorithm can not specify the final number of clusters directly, we set the similarity threshold parameter to a proper value to get the desired number of clusters in both algorithm (In *Squeezer* algorithm, we observed that if the number of output clusters is same, the clustering accuracy is almost identical. Hence, we can use *any* similarity threshold value that can make the algorithm get the desired number of clusters. In *NabSqueezer* algorithm, we have the similar observation).

Fig. 2 shows the results of different clustering algorithms on soybean dataset. From Fig. 2, we can summarize the relative performance of these algorithms as Table 2. In Table 2, the numbers in column labelled by *k* (*k*=1, 2 or 3) are the times that an algorithm has rank *k* among these algorithms. For instance, in the 8 experiments, *k*-modes algorithm performed third best in 6 cases, that is, it is ranked 3 for 6 times.

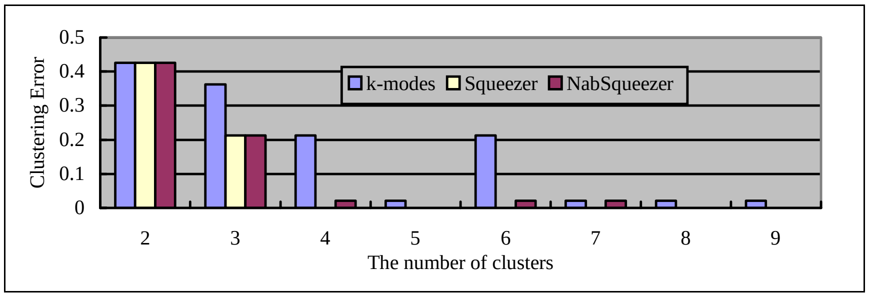

Fig. 2. Clustering error vs. Different number of clusters (soybean dataset)

One important observation from Fig.2 and Table 2 is that, among all of these algorithms, *k*-modes performed the worst in most cases. Although the performance of *Squeezer* algorithm on this dataset is better than that of the *NabSqueezer* algorithm, their clustering errors are almost identical. In other words, *Squeezer* algorithm and *NabSqueezer* algorithm achieved the same level performance on soybean data.

Table 2. Relative performance of different clustering algorithms (soybean dataset)

Ranking	1	2	3	Average Clustering Error
***k*-modes**	1	1	6	0.162
Squeezer	8	0	0	**0.080**
NabSqueezer	5	3	0	0.088

The experimental results on mushroom dataset are described in Fig. 3 and the relative performance is summarized in Table 3. As shown in Fig. 3 and Table 3, *NabSqueezer* algorithm beats all the other algorithms in average clustering error. Furthermore, although the *NabSqueezer* algorithm didn't always perform best on this dataset, it performed best and second best in most cases.

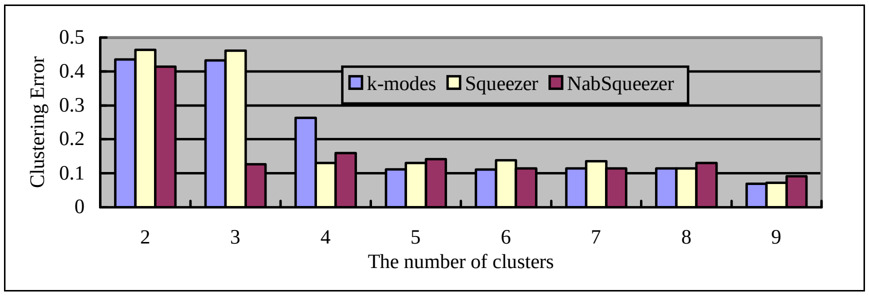

Fig. 3. Clustering error vs. Different number of clusters (mushroom dataset)

Table 3. Relative performance of different clustering algorithms (mushroom dataset)

Ranking	1	2	3	Average Clustering Error
***k*-modes**	5	2	1	0.206
Squeezer	2	3	3	0.206
NabSqueezer	3	3	2	**0.162**

The experimental results on the breast cancer dataset are described in Fig.4 and the summarization on the relative performance of three algorithms is given in Table 4. It is easy to see that the *NabSqueezer* algorithm performed the best in all cases and never performed the worst. At the same time, *NabSqueezer* algorithm is also superior to the *Squeezer* algorithm and *k*-modes algorithm with respect to average clustering error.

The above experimental results demonstrate the effectiveness of the modified *Squeezer* algorithm by giving greater weight to uncommon attribute value matches in similarity computations. Therefore, it is promising to improve the clustering accuracies of other existing categorical data clustering algorithms in a similar way.

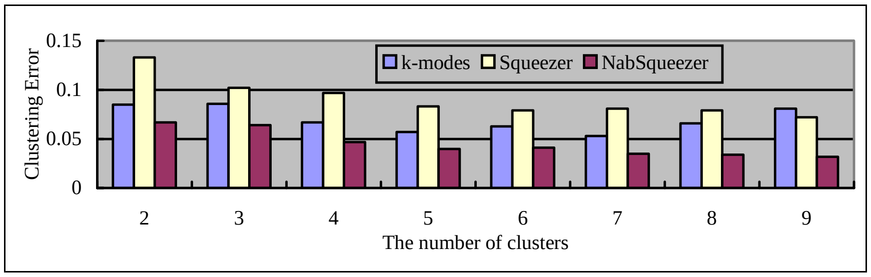

Fig. 4. Clustering error vs. Different number of clusters (cancer dataset)

Table 4. Relative performance of different clustering algorithms (cancer dataset)

Ranking	1	2	3	Average Clustering Error
***k*-modes**	0	7	1	0.070
Squeezer	0	1	7	0.091
NabSqueezer	8	0	0	**0.045**

5. Conclusions

Taking *Squeezer* algorithm as a representative, we empirically demonstrate that clustering accuracies of existing categorical data clustering algorithms can be further improved by giving greater weight to uncommon attribute value matches in similarity computations.

In the future work, we will investigate the feasibility of applying the heuristic of weighting uncommon attribute value matches to other similarity based categorical data clustering algorithm, such as *k*-modes [7] and ROCK [8]

Acknowledgements

The comments and suggestions from the anonymous reviewers greatly improve the paper. This work was supported by the High Technology Research and Development Program of China ((No. 2004AA413010, No. 2004AA413030), the National Nature Science Foundation of China (No. 40301038) and the IBM SUR Research Fund.

References

1. He, Z., Xu, X., Deng, S.: Squeezer*:* An Efficient Algorithm for Clustering Categorical Data. Journal of Computer Science and Technology, Vol. 17, No.5, pp. 611-624. (2002)
2. He, Z., Xu, X., Deng, S.: A Cluster Ensemble Method for Clustering Categorical Data. Information Fusion, Vol. 6, No. 2, 143-151. (2005)
3. Chang, C., Ding, Z.: Categorical Data Visualization and Clustering Using Subjective Factors. Data & Knowledge Engineering, Vol. 53, No. 3, 243-263. (2005)
4. Ng, M. K., Wong, J. C.: Clustering Categorical Data Sets Using Tabu Search Techniques. Pattern Recognition, Vol. 35, No.12, 2783-2790. (2002)
5. He, Z., Deng, S., Xu, X.: Improving K-modes Algorithm Considering Frequencies of Attribute Values in Mode. Lecture Notes in Artificial Intelligence, Vol. 3801, 157-162. (2005)
6. He, Z., Xu, X., Deng, S.: TCSOM: Clustering Transactions Using Self-Organizing Map. Neural Processing Letters, Vol. 22, No. 3, 249-262. (2005)
7. Huang, Z.: Extensions to the k-Means Algorithm for Clustering Large Data Sets with Categorical Values. Data Mining and Knowledge Discovery, Vol. 2, 283-304. (1998)
8. Guha, S., Rastogi, R., Shim, K.: ROCK: A Robust Clustering Algorithm for Categorical Attributes. Information Systems, Vol. 25, No. 5, 345-366. (2000)
9. Goodall, D. W.: A New Similarity Index Based on Probability. Biometrics, Vol. 22, 882-907. (1966).
10. Li, C., Biswas, G.: Unsupervised Learning with Mixed Numeric and Nominal Data. IEEE Transactions on Knowledge and Data Engineering, Vol. 14, No. 4, 673-690. (2002)
11. Merz, C. J., Merphy, P.: UCI Repository of Machine Learning Databases (1996). [Online]. Available: Http://www.ics.uci.edu/~mlearn/MLRRepository.html

Zengyou He received his PhD degree in computer science from Harbin Institute of Technology (HIT) in China in 2006. His main research interests include data mining and machine learning.

Xiaofei Xu received both his M.S degree and Ph.D. degree in computer science from HIT in 1985 and 1988. He is currently a professor and dean of School of Computer Science and Technology, dean of National Pilot School of Software in Harbin Institute of Technology in China. His research fields include computer integrated manufacturing systems, databases, supply chain management, e-business, knowledge engineering, etc.

Shengchun Deng received his Ph.D. degree in computer science from HIT in 2002. He is currently an Associate Professor in the Department of Computer Science and Engineering, HIT. His main research interests include data mining and supply chain management.