Clustering Based on Compressed Data for Categorical and Mixed Attributes

Erendira Rendón[1] and José Salvador Sánchez[2]

[1] Lab. Reconocimiento de Patrones, Instituto Tecnológico de Toluca
Av. Tecnológico s/n, 52140 Metepec, Mexico
erendon@ittoluca.edu.mx

[2] Dept. Llenguatges i Sistemes Informàtics, Universitat Jaume I
Av. Sos Baynat s/n, E-12071 Castelló de la Plana, Spain
sanchez@uji.es

Abstract. Clustering in data mining is a discovery process that groups a set of data so as to maximize the intra-cluster similarity and to minimize the inter-cluster similarity. Clustering becomes more challenging when data are categorical and the amount of available memory is less than the size of the data set. In this paper, we introduce CBC (*Clustering Based on Compressed Data*), an extension of the Birch algorithm whose main characteristics refer to the fact that it can be especially suitable for very large databases and it can work both with categorical attributes and mixed features. Effectiveness and performance of the CBC procedure were compared with those of the well-known K-modes clustering algorithm, demonstrating that the CBC summary process does not affect the final clustering, while execution times can be drastically lessened.

1 Introduction

Clustering constitutes a very effective technique for exploratory data analysis and has been widely studied for several years. It has found applications in a broad variety of areas such as pattern recognition, statistical data analysis and modelling, data and web mining, image analysis, marketing and many other business applications. The basic clustering problem consists of grouping a data set into subsets (typically called clusters) such that items in the same subset are similar to each other, whereas items in different subsets are as dissimilar as possible. The general idea is to discover a structure that is already present in the data. Most of the existing clustering algorithms can be classified into two main categories, namely hierarchical (agglomerative or divisive) and partitioning algorithms [8].

Pattern recognition and data mining practical applications frequently require dealing with high volumes of data (thousands or millions of records with tens or hundreds of attributes). This characteristic excludes the possibility of using many of the traditional clustering algorithms. Furthermore, this type of application is often done with data containing categorical attributes, thus becoming more difficult.

It has to be noted that much of the data in the databases are categorical, that is, fields in tables whose attributes cannot naturally be ordered as numerical values. The problem of clustering categorical data involves complexity not encountered in the corresponding problem for numerical data. While considerable research has been done on clustering numerical data, there has been much less work on the important problem of clustering categorical data.

The present paper focuses on clustering databases with categorical and/or mixed (both categorical and numerical) attributes. To this end, a new clustering algorithm is here introduced, which is based on summarizing the data by using a balanced tree structure. The resulting tree is then utilized to group the data into clusters, which will be finally labelled by means of a nearest neighbor rule. Moreover, it is worth noting that this algorithm allows its application to very large databases (data sets of size greater than the size of available memory).

2 Related Algorithms

Clustering numerical data has been the focus of substantial research in various domains for decades, but there has been much less work on clustering categorical data. Recently, the important problem of clustering categorical data started receiving interest. In this section, we briefly review a number of algorithms belonging to the area of clustering categorical records.

The Rock algorithm [5] is an adaptation of an agglomerative hierarchical algorithm. The procedure attempts to maximize a goodness measure that favors merging pairs with a large number of ”links”. Two objects are called neighbors if their similarity exceeds a certain threshold given by the user. The number of links between two objects is the number of common neighbors. The Rock algorithm selects a random sample from the databases after a clustering algorithm that employs links is applied to the sample. Finally, the obtained clusters are used to assign the remaining objects on the disk to the appropriate clusters. Huang proposes the K-modes algorithm [6], which is an extension of the K-means procedure for categorical data. The way to compute the centroid is substituted by a vector of modes. It also proposes two measures of similarity for categorical data. The final clustering of the K-modes algorithm depends on the initial selection of the vector of modes, very much the same as with its predecessor K-means.

The Coolcat algorithm proposed by Barbará et al. [2] is an incremental algorithm that aims to minimize the expected entropy of the clusters. Given a set of clusters, the algorithm will place the next point in the cluster where it minimizes the overall expected entropy. Similar to this proposal is the Limbo technique [1], which constitutes a scalable hierarchical categorical clustering algorithm that builds on the Information Bottleneck (IB) framework for quantifying the relevant information preserved when clustering. As a hierarchical algorithm, Limbo has the advantage that it can produce clusters of different sizes in a single execution. Moreover, Limbo handles large data sets by producing a memory bounded summary model for the data.

Cactus [3] represents an agglomerative hierarchical algorithm that employs data summarization to achieve linear scaling in the number of rows. It requires only two scans of the data. This scheme is based on the idea of co-occurrence for pairs of attributes-values. The Birch algorithm [10] constructs a balanced tree structure (the CF-tree), which is designed for a multi-phase clustering method. First, the database is scanned to build an initial in-memory CF-tree which can be seen as a multi-level compression of the data that tries to preserve the inherent clustering structure of the data.

Second, an arbitrary clustering algorithm can be used to cluster the leaf nodes of the CF-tree.

3 The CBC Algorithm

The new CBC clustering algorithm here introduced consists of three main stages: (1) summary, (2) clustering, and (3) labelling. In the first stage, the data are summarized in a balanced tree structure. The summary obtained in the first step is then grouped by means of a clustering procedure. Finally, in the the third stage we perform a scan over the database and assign each object to the representative (that is, the composite object) closest to the clusters obtained in the previous phase.

Next, we provide the definition of several concepts that will be importantly used by the CBC algorithm.

Definition 1. *An* **event** *is a pair relating features and values. It can be denoted by* $[X_i = E_i]$*, indicating that the feature* X_i *takes the values of* E_i *and* $E_i \subset U_i$*.* E_i *is the subset of values that the feature* X_i *takes,* U_i *is the representation domain of* X_i*. An example of event is* $e_1 = [color = green, blue, red]$.

Definition 2. *A* **categorical object** *is a logical join of events, relating values and features, where features may take one or more values [4]. It is denoted by* $X = [X_1 = E_1] \wedge \ldots \wedge [X_d = E_d]$*. A categorical object can be represented by the Cartesian product set* $E = E_1 \times \ldots \times E_d$*. The domain of categorical object* X *is represented by* $U^{(d)} = U_1 \times \ldots \times U_d$*, that is, the d-dimensional feature space. For example, the categorical object represented by* $X = [HairColor = black, brown] \wedge [BloodType = B+, A+]$ *has the following features: HairColor is black or brown; BloodType is B+ or A+.*

The intersection of two categorical objects $E_i = E_{i1} \times \ldots \times E_{id}$ *and* $E_j = E_{j1} \times \ldots \times E_{jd}$ *is defined as* $E_i \bigotimes E_j = (E_{i1} \bigotimes E_{j1}) \times \ldots \times (E_{id} \bigotimes E_{jd})$*. Analogously, the union of* E_i *and* E_j *is* $E_i \bigoplus E_j = (E_{i1} \bigoplus E_{j1}) \times \ldots \times (E_{id} \bigoplus E_{jd})$*.*

Definition 3. *Let* $E_i = E_{i1} \times \ldots \times E_{id}$ *and* $E_j = E_{j1} \times \ldots \times E_{jd}$ *be two objects defined in* $U^{(d)}$*, then a* **composite object** *is the result of combining* E_i *and* E_j *as* $E_i \bigoplus E_j = (E_{i1} \bigoplus E_{j1}) \times \ldots \times (E_{id} \bigoplus E_{jd})$*. For example, consider two objects* $A = \{green, B+, high\}$ *and* $B = \{black, O+, high\}$*, then the composite object formed from these is* $A \bigoplus B = \{\{green, black\}, \{B+, O+\}, high\}$*.*

3.1 A Similarity Metric for Categorical Objects

In the present work, we have used a distance metric similar to that proposed by Ichino and Yaguchi [7]. The distance between objects $X_i = [X_{i1} = E_{i1}] \wedge \ldots \wedge [X_{id} = E_{id}]$ and $X_j = [X_{j1} = E_{j1}] \wedge \ldots \wedge [X_{jd} = E_{jd}]$ in $U^{(d)}$ is computed by:

$$d_p(X_i, X_j) = [\sum_{k=1}^{d} C_k \psi(E_{ik}, E_{jk})^p]^{1/p} \qquad p \geq 1 \tag{1}$$

where $C_k > 0$ ($k = 1, \ldots, d$) is a weighting factor to control the relative importance of the event E_k or $C_k = 1/d$ when all the events have the same relevance, and

$$\psi(E_{ik}, E_{jk}) = \frac{\phi(E_{ik}, E_{jk})}{|U_k|} \tag{2}$$

being $|U_k|$ the number of possible values in the domain U_k, and $\phi(E_{ik}, E_{jk}) = |E_{ik} \cup E_{jk}| - |E_{ik} \cap E_{jk}|$

Now we can transform the distance measure in Eq. 1 into a similarity metric [8], such as $S(X_i, X_j) = 1 - d_p(X_i, X_j)$.

3.2 Summary of the Database

The central task of the first stage of CBC is the construction of a tree structure to compress the data. It uses a weight-balanced tree, called *CO-tree*, which needs three parameters: the number of entries for non-leaf nodes B, the number of entries for leaf nodes L, and an absorbtion threshold T. Each non-leaf node contains at most B entries, each one with a pointer to its child node, and the composite object represented by this child. A leaf node contains at most L entries with composite objects. Moreover, all entries in a leaf node must satisfy a condition with respect to the absorbtion threshold T. Fig. 1 illustrates an example of the CO-tree structure.

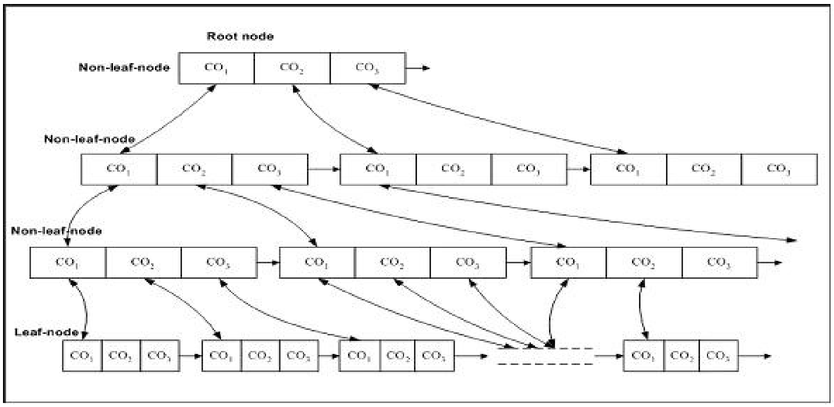

Fig. 1. An example of the CO-tree structure ($B = 3$, $L = 3$)

Insertion into a CO-Tree. Here we present the algorithm for inserting a new entry, say ent, into a given CO-tree.

1. Identifying the appropriate leaf: starting from the root, it recursively descends the CO-tree by choosing the closest child node according to the similarity metric given in Sect. 3.1.
2. Updating the leaf: when reached a leaf node, it finds the closest leaf entry, say L_i, and then tests whether L_i can absorb ent without violating the threshold condition. If so, the new entry is absorbed by L_i. If not, it tests whether there is space for this new entry on the leaf. If so, a new entry for ent is added to the leaf. Otherwise, we must split the leaf node. Splitting is done by choosing the farthest pair of entries as seeds, and redistributing the remaining entries based on the closest criteria.

3. Updating the path to the leaf: after inserting ent into a leaf, we must update the information for each non-leaf entry on the path from the root to the leaf. If no splitting was done, this simply involves the leaf entry that absorbed ent. Conversely, a leaf split means that a new non-leaf entry has to be inserted into the parent. If the parent has space for this new entry, at all higher levels, we only need to update the corresponding entries to reflect the addition of ent. However, we may have to split the parent as well, and so on up to the root. If the soot is split, the tree height increases by one.

The first stage of CBC starts with an initial threshold value $T = 0$, scans the data, and inserts entries into the CO-tree. In this phase, it is possible to carry out an additional step of the summary by using a reconstruction algorithm. This reconstruction algorithm is applied when a more compact CO-tree is required or when the memory is depleted and some objects in the database have not been inserted yet. When rebuilding the CO-tree (smaller than the initial one), the same insertion algorithm is used, but increasing the threshold value $T = T + 0.2$. It reinserts the leaf entries of the old tree and then, the scanning of the data is resumed from the point at which it was interrupted. This process continues until all objects in the database have been scanned. The leafs of the resulting CO-tree contain a summary of the database in the manner of composite objects.

3.3 Clustering the Leaf Nodes

The second stage of the algorithm basically consists of going through the leaf nodes of the resulting CO-tree in order to obtain composite objects that will be representatives of the clusters.

After the construction of the CO-tree, that is, when the entire database has been summarized, we cluster the composite objects present in the leaf nodes, thus producing groups of composite objects. In this phase, any clustering algorithm could be applied. However, we propose a new clustering algorithm called EA. The EA clustering algorithm is described in detail below.

Definition 4. *Let $B = \{X_1, X_2, \ldots, X_n\}$ be a set of composite objects in $U^{(d)}$ obtained from the final CO-tree. Then $C \subseteq B$, $C \neq \emptyset$ will be a* **cluster** *if and only if the following conditions are satisfied:*

a) $\forall X_j \in B \;\; [X_i \in C \wedge \max_{X_t \in B, X_t \neq X_i} \{S(X_i, X_t)\} = S(X_i, X_j) \geq \beta] \Rightarrow X_j \in C$
b) $[X_t \in C \wedge \max_{X_p \in B, X_p \neq X_i} \{S(X_i, X_p)\} = S(X_p, X_t) \cdot \beta] \Rightarrow X_p \in C$
c) $|C|$ *is minimum*

where $0 \leq \beta \leq 1$ is a similarity threshold.

EA Clustering

1. Scan all leaf entries (composite objects) present in the final CO-tree.
2. Compute the similarity matrix using Eq. 1.
3. Compute β as an average of the values in the similarity matrix. It can also be given by a human expert.
4. Compute the clusters C.
5. Compute the composite objects of each cluster formed in Step 4.

4 Experimental Analysis

In this section, we evaluate the performance of CBC and compare its effectiveness for clustering with that of the classical K-modes algorithm. We performed two groups of experiments: in the first one, we ran the CBC algorithm with respect to parameters L, B and memory size. In the second one, we compared CBC with the K-modes algorithm in terms of misclassified objects and running times. We consider an object as misclassified when the original tag was different from that assigned by CBC.

4.1 Description of the Data Sets

The present experimental study was carried out by using three well-known benchmark databases taken from the UCI Machine Learning Database Repository (`http://www.ics.uci.edu/ Mlear/MLRepository.html`).

- Mushroom: each data record contains information that describes the physical characteristics (e.g., color, odor, size, shape) of a single mushroom. The mushroom database has a total of 8124 records belonging to two classes: 4208 edible mushrooms and 3916 poisonous mushrooms.
- Connect-4: it consists of 67557 records, each one described by 42 characteristics. The three classes are *win* with 44473 records, *loss* with 16635, and *draw* with 6449 records.
- Kr-vs-kp: Chess and -Game-King+Rook versus King+Pawn contains 3196 records, each one described by 36 attributes. The two classes are *white-can-win* with 1669 and *white-cannot-win* with 1527 records.

4.2 Results

The first experiment pursues to analyze the effect of memory size on running times and percentage of misclassified objects. To this end, we have tested the CBC algorithm when varying the memory size and keeping constant the values of parameters L and B. For the Mushroom database, $L = 5$ and $B = 3$. For the Kr-vs-kp data set, $L = 6$ and $B = 3$. For the Connect-4 database, $L = 4$ and $B = 3$. Table 1 reports the results corresponding to these experiments. The third column provides the running times for the summary stage, that is, the construction of the CO-tree to summarize the data. Columns 4, 5, and 6 correspond to the percentage of misclassified objects for different values of the parameter β in the second stage of the algorithm. Finally, the seventh column shows the average running times (the time required for the summary along with the time for the clustering stage).

From the results in Table 1, one can see that the time for the summary stage is close to linear with respect to the memory size. On the other hand, the parameter β used in the clustering stage of CBC significantly affects to the quality of the clusters obtained. In general, it seems that the lower the values of β, the lower the percentage of misclassified objects. Finally, the size of the memory does not affect to the result of the CBC clustering algorithm in terms of percentage of misclassified objects.

The second experiment tries to study the effect of the parameter L. Correspondingly, Table 2 provides the results when varying the value of parameter L, while keeping

Table 1. Effect of memory size (Kbytes) on running times (seconds) and percentage of misclassified objects

	Memory size	Summary time	$\beta = 0.2$	$\beta = 0.1$	$\beta = 0.05$	Average time
Mushroom	5	1.34	12.81	7.03	6.08	2.48
	10	1.33	12.81	7.03	6.08	
	20	2.50	12.81	7.03	6.08	
	25	3.00	12.81	7.03	6.08	
Kr-vs-kp	2	1.69	40.47	14.29	10.43	1.42
	3	1.88	46.38	16.05	10.66	
	4	2.19	47.88	20.19	12.58	
	5	2.18	47.88	19.19	12.31	
Connect-4	40	51.21	34.16	35.82	34.16	45.00
	60	50.11	34.16	34.16	34.16	
	80	68.62	34.16	36.14	34.38	
	100	69.17	34.16	36.14	29.58	

Table 2. Effect of parameter L

	L	Summary time	$\beta = 0.2$	$\beta = 0.1$	$\beta = 0.05$	Average time
Mushroom	3	2.50	15.12	6.94	4.55	2.50
	5	2.50	12.81	7.03	6.08	
	7	2.80	18.52	9.56	11.38	
	9	3.38	16.37	13.90	12.83	
	11	3.54	18.63	10.56	10.56	
Kr-vs-kp	4	1.49	38.27	42.25	26.48	1.64
	5	1.83	26.55	16.59	11.53	
	6	1.69	40.57	14.29	10.43	
	7	2.00	47.20	14.03	11.33	
Connect-4	2	50.29	33.56	31.53	31.53	48.09
	3	57.74	30.98	30.98	30.98	
	4	64.94	34.22	34.22	34.22	
	5	60.86	34.16	29.99	30.98	

constant the memory size and the value of B. The memory size is 20, 2, and 20 for Mushroom, Kr-vs-kp, and Connect-4 databases, respectively. For the three data sets we have used $B = 3$. Most comments drawn for the first experiments become valid for the present analysis. In this sense, one can observe that the time for the summary stage is close to linear with respect to the value of L. Also, it seems that the percentage of misclassified objects does not depend on the parameter L.

Analogously, the third group of experiments are devoted to study the effect of the parameter B. Table 3 reports the running times and percentage of misclassified objects when varying the value of the parameter B. The size of available memory is 20, 2, and 20 for Mushroom, Kr-vs-kp, and Connect-4 databases, respectively. For the three data sets we have used $L = 3$. Like in the previous experiments, the value of the parameter B does not affect to the quality of the clusters given by the CBC algorithm.

Table 3. Effect of parameter B

	B	Summary time	$\beta = 0.2$	$\beta = 0.1$	$\beta = 0.05$	Average time
Mushroom	3	2.50	15.12	6.94	4.55	2.52
	5	2.10	18.17	7.88	5.93	
	7	2.56	22.44	12.51	8.34	
	9	2.51	18.66	10.06	6.35	
	11	2.58	16.84	7.33	7.11	
Kr-vs-kp	4	1.53	38.27	35.42	30.03	1.63
	5	1.52	38.27	28.16	22.51	
	6	1.62	40.06	19.90	11.85	
	7	1.51	39.55	17.29	12.26	
Connect-4	3	57.74	30.98	30.98	30.98	49.98
	4	61.21	32.36	32.36	32.36	
	5	58.07	34.16	40.23	40.23	
	6	63.61	34.15	34.15	34.15	

Table 4. Comparison of CBC and K-modes in percentage of misclassified objects

	CBC		K-modes
	Worst	Best	
Mushroom	22.44	4.55	7.42
Kr-vs-kp	47.88	10.43	34.01
Connect-4	40.23	29.99	45.03

Finally, Table 4 allows to compare the percentage of misclassified objects by means of the CBC procedure with that of the well-known K-modes clustering algorithm. As can be seen, in all domains the CBC approach has shown a better behavior than the K-modes algorithm. The most important differences are with the Kr-vs-kp database, in which the CBC algorithm obtains a 10.43% of error rate, while that of the K-modes is 34.01%. On the other hand, it has to be noted that in the case of Connect-4, even the worst cluster given by CBC (40.23%) outperforms the result of K-modes (45.03%).

5 Concluding Remarks

This paper introduces the CBC algorithm, *Clustering Algorithm Based on Compressed Data*, which builds a summary of the database in the main memory using a balanced tree structure called CO-tree. The CBC algorithm works with categorical features, and also with mixed data. Another important characteristic of the new algorithm refers to the fact that it can handle large data sets.

The CBC clustering algorithm consists of tree main stages: (1) summary, (2) clustering, and (3) labelling. Although the databases have to be summarized, the results of our experimental study with three benchmark databases are very encouraging because it clearly outperforms the K-modes in terms of percentage of misclassified objects.

Possible extensions to this work are in the direction of testing the CBC algorithm with other similarity measures. Also, the possibility of using a different clustering approach in the second stage becomes especially important in order for improving the quality of the resulting clusters. Finally, a more exhaustive empirical analysis is necessary to corroborate the conclusions given in the present paper.

Acknowledgments

This work has been partially supported by research grants TIC2003-08496 from the Spanish CICYT (Ministry of Science and Technology).

References

1. Andritsos, P., Tsaparas, P., Miller, R.J., Sevcik, K.C.: LIMBO: scalable clustering of categorical data, In: Proc. 9th Intl. Conf. on Extending Database Technology (2004) 123–146.
2. Barbará, D., Li, Y., Couto, J.: COOLCAT: an entropy-based algorithm for categorical clustering, In: Proc. 11th Intl. Conf. on Information and Knowledge Management (2002) 582–589.
3. Ganti, V., Gehrkeand, J., Ramakrishanan, R.: CACTUS — Clustering categorical data using summaries, In: Proc. 5th ACM Sigmod Intl. Conf. on Knowledge Discovery in Databases (1999) 73–83.
4. Gowda, K., Diday, E.: Symbolic clustering using a new dissimilarity measure, Pattern Recognition **24** (1991) 567–578.
5. Guha, S., Rastogi, R., Shim, K.: ROCK: A robust clustering algorithm for categorical attributes, In: Proc. of the IEEE Intl. Conf. on Data Engineering (1999) 512–521.
6. Huang, Z.: A fast clustering algorithm to cluster very large categorical data sets in data mining, In: Proc. SIGMOD Workshop on Research Issues on Data Mining and Knowledge Discovery, Tech. Report 97–07, UBC, Dept. of Computer Science (1997).
7. Ichino, M., Yaguchi, H.: Generalized Minkowski metrics for mixed feature-type data analysis, IEEE Trans. on Systems, Man and Cybernetics **24** (1994) 698–708.
8. Kaufman, L., Rousseeuw, P.J.: *Finding Groups in Data: An Introduction to Cluster Analysis*. John Wiley & Sons, New York (1990).
9. Milenova, B.L., Campos, M.M.: Clustering large databases with numeric and nominal values using orthogonal projection, In: Proc. 29th Intl. Conf. on Very Large Databases (2003).
10. Zhang, T., Ramakrishnan, R., Livny, M.: BIRCH: an efficient data clustering method for very large databases, In: Proc. ACM-SIGMOD Intl. Conf. on Management of Data (1996) 103–114.