Efficient Construction of Comprehensible Hierarchical Clusterings

Luis Talavera and Javier Béjar

Departament de Llenguatges i Sistemes Informàtics
Universitat Politècnica de Catalunya
Campus Nord, Mòdul C6, Jordi Girona 1-3
08034 Barcelona, Catalonia, Spain
{talavera,bejar}@lsi.upc.es

Abstract. Clustering is an important data mining task which helps in finding useful patterns to summarize the data. In the KDD context, data mining is often used for description purposes rather than for prediction. However, it turns out difficult to find clustering systems that help to ease the interpretation task to the user in both, statistics and Machine Learning fields. In this paper we present ISAAC, a hierarchical clustering system which employs traditional clustering ideas combined with a feature selection mechanism and heuristics in order to provide comprehensible results. At the same time, it allows to efficiently deal with large datasets by means of a preprocessing step. Results suggest that these aims are achieved and encourage further research.

1 Introduction

Clustering is one of the primary data mining tasks aiming to the goal of finding a useful set of categories or clusters to summarize the data. As in other inductive tasks, clustering results may serve for two different purposes, namely, *prediction* and *description*. As pointed out in [2], in the context of Knowledge Discovery on Databases (KDD), description tends to be a more important task, since the main focus in this discipline is on finding interpretable patterns. Traditionally, when applying clustering algorithms, the interpretation step is usually left to users. So, they have to evaluate the results and change the appropriate settings of the clustering system if the results does not suit their needs. However, the settings of a clustering system may be difficult to interpret for an average user who has not deep knowledge about metrics or control strategies.

Therefore, it may be desirable to have clustering methods which not only perform a partition of the data, but also facilitate the interpretation task. The weaknesses of traditional statistical clustering methods to cope with this requirement, gave raise to the development of *conceptual clustering* methods in the Machine Learning (ML) community [3, 5, 6, 8]. These methods are intended to combine the clustering and interpretation tasks thus making easy the later to the external user.

Furthermore, data mining poses additional problems for the inductive tasks such as *database size*, *high dimensionality* or the need for *user interaction*, which make the process even harder. We present the ISAAC system, an approach that combines traditional clustering concepts with some heuristic procedures in order to provide comprehensible results and, at the same time, efficiently deal with large amounts of data. It also allows the user to decide the structure of the cluster hierarchy with regard to the number of levels and their generality.

2 Isaac

ISAAC is a conceptual clustering system that accepts vectors of nominal attribute-value pairs and summarizes these objects in probabilistic concept hierarchies. A probabilistic concept is represented by a summary description that lists attribute values and its associate probabilities. For each concept C_k, a *prototype* stores $P(A_i = V_{ij} \mid C_k)$, the conditional probability that a value V_{ij} for a feature A_i will occur in an object of the cluster C_K. Probabilistic descriptions are not traditionally used in statistical clustering approaches, but they are more common in conceptual clustering systems [3, 5]. This sort of representation allows gradual updating of clusters descriptions and should be more robust than logic-based representations in the face of noise or graded concepts.

ISAAC is intended to allow users to model the construction of the cluster hierarchy which better suits their needs. Typically, hierarchical clusterings are arranged in a binary tree or dendrogram. From this tree, the user has to extract a useful partition, or apply some automatic procedure to select the best level or levels. In our approach, the system allows the user to specify the number of levels of the hierarchy and their generality. This is done via the NG parameter which is in the [0,1] range. As the NG value increases, the system creates more general partitions with few concepts. Lower NG values instruct the system to build more specific partitions. A complete hierarchy is built by specifying a set of increasing NG values to indicate the desired levels. The user can interact with the system experimenting with different sets of values for this parameter. Since the effect of modifying the NG values is semantically clear to the user, it should be relatively easy to deal with this parameter.

The ISAAC clustering process, which is depicted in Fig. 1, consists of three stages: Preprocessing, Reflection and Refinement which will be detailed in the following sections.

2.1 Preprocessing

Typically, complexity of hierarchical clustering algorithms is $O(n^2)$ where n is the number of objects in the dataset. This complexity may be acceptable if one desires just a 'one-shot' clustering. However, for large datasets, it may result in a relatively slow processing, particularly if the user needs to interact with the clustering system by changing some setting in order to obtain interesting results.

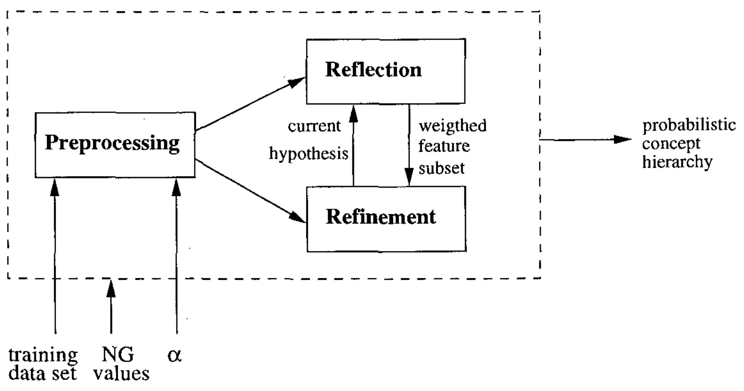

Fig. 1. Stages of the ISAAC system

The Preprocessing stage is intended to cope with this problem. We can see this stage as similar to the typically used feature selection steps which reduce the set of features used in the learning process. The difference here lies in that, instead of selecting objects, the Preprocessing step transforms the initial data vectors into new vectors summarizing compact groups of objects. Particularly, we exploit *incremental* clustering algorithms, which can efficiently deal with large datasets. An incremental clustering algorithm can construct a flat partition in a $O(nk)$ time, where k is the number of clusters created. Note that this complexity is equivalent to $O(n^2)$ only if $k \approx n$. The problem with incremental algorithms is that they are sensitive to ordering effects. To cope with this problem while maintaining the efficiency of incremental processing, we are working on a buffering strategy which has shown to be fairly robust under bad object orderings and it is detailed in [10].

We chose a nearest neighbor algorithm with uses a similarity measure and an α threshold to form clusters. For each instance the similarity with every existing cluster is computed. If the maximum similarity found is greater than α then the instance is incorporated into the cluster. Otherwise, a new cluster is created. With this preliminary step, the size of the database can be compressed in a variable amount dependent on the value of α. Intuitively, similarity is obtained by computing the intersection between the cluster prototypes for each feature. Specifically, similarity for two clusters C_m and C_n with respect to a feature i is computed by the expression:

$$Sim(C_m, C_n, i) = \sum_j min\{P(A_i = V_{ij} \mid C_m), P(A_i = V_{ij} \mid C_n)\} \qquad (1)$$

The total similarity between two clusters is the average similarity computed for each feature, normalized by dividing for the total number of features considered.

This procedure helps to detect very similar -or even identical- objects in a fast manner. It is worth to notice that although very low α values will result in a greater compression of the dataset, it also will produce very general clusters, letting little room for improvements to the next stages. The user may explore a suitable trade-off between these two aspects by using different α values.

2.2 Reflection

This stage is intended to extract useful information about the clustering process in order to guide the following steps. It actuates at every step of the agglomerative clustering procedure which will be explained below. It takes the current set of clusters as the basis for computing the relevance of each feature. For this purpose the *distance measure* [7], a relevance measure used for attribute selection in decision tree induction, is used. Note that by using this sort of measure we are assuming that the more relevant features are those which better discriminate among the clusters of a given partition. As a result of the process, an ordered set of features is obtained and those which do not score high enough are discarded. A heuristic procedure is used to determine this subset of useful features. The rationale behind this procedure is the assumption that the set of attributes which can discriminate among the clusters of a given partition is smaller for more general levels than for specific ones. The level of generality of the level which is being built is indicated by the NG value specified by the user, so it can be used to heuristically define the set of useful features $\mathcal{U}$ as follows:

$$\mathcal{U} = \{a \in \mathcal{A} \mid Rel(a) \geq m \cdot NG\} \tag{2}$$

where $\mathcal{A}$ is the initial set of features, $Rel(a)$ is the relevance of feature a and m is the maximum computed relevance for features in $\mathcal{A}$. This procedure does not guarantee in any case that the selected subset is neither the best nor the minimal for any particular task. We are just using a relatively naive approach to make a conservative selection in an unknown domain allowing the system to dynamically discard features which are very likely to be irrelevant.

2.3 Refinement

This stage consists of an agglomerative procedure which iteratively selects the best two candidates to merge and creates a new cluster. However, a number of differences with traditional clustering procedures exist. First, the starting point for this procedure is not the original dataset, but the clusters obtained in the Preprocessing stage or in the previous generalization if a set of NG values is given. Secondly, the procedure ends when the level of generality indicated by current NG value is achieved and the resulting level does not store all the intermediate pairs of mergings. Therefore, the system does not necessarily produces binary trees. And finally, at each merging step, the current set of clusters is passed to the Reflection step to obtain a weighted subset of useful features which is used in the following computations.

To be able of determining when a given level of generality is reached, we need a measure to characterize cluster generality. ISAAC measures the generality of a cluster using probabilistic analogs of logical sufficiency and necessity. These measures are interpreted as degrees of sufficiency and necessity of the probabilistic descriptions used by the system. From this point of view, they represent a continuous valuation over the sufficiency and necessity properties analogous to the binary one of classical logic, but allowing a greater flexibility. These measures are called *continuous sufficiency* (CS), and *continuous necessity* (CN), and are defined, for a cluster C_k as follows:

$$CS(C_k) = \sum_i \sum_j P(C_k \mid A_i = V_{ij})^2 \tag{3}$$

$$CN(C_k) = \sum_i \sum_j P(A_i = V_{ij} \mid C_k)^2 \tag{4}$$

i indexes the features of the objects, and j indexes the values of each feature. Both measures can be easily generalized to partitions by simply averaging the results for the set of clusters. These two measures evaluate the average degree of sufficiency or necessity for the features contained in the prototypes of the clusters of a given partition.

Each of the measures is biasing the selection of partitions in a different direction as regards the generality of a partition. Favoring CS over CN tends to reward partitions with more general concepts and favoring CN over CS tends to reward partitions with more specific concepts. This property allows the user to bias the process towards the type of partition required, linking the bias to the NG parameter by means of the formula:

$$Gen(P, NG) = (1 - NG) \times CS(P) - NG \times CN(P) \tag{5}$$

This measure indicates whether a given level of generality defined by a given NG value has been achieved and measured is used to allow merging until a certain level of both CN and CS measures is reached. The initial partition usually will have a negative Gen value due to the high score of CN and, as the generalization progresses, the generality of the partition will tend to zero according to the evolution of the CS and CN measures.

The control structure for the Refinement stage is shown in Table 1. As mentioned before, it follows a typical agglomerative schema which merges pairs of clusters until the desired NG level is reached. However, unlike most of its statistical counterparts, our algorithm does not construct a similarity matrix to decide which pair of clusters should merge. Instead, the algorithm takes advantage of the generality measure defined above and always chooses as the first cluster to merge the one who has the lower generality score. The similarity of this cluster with the rest of the clusters in the partition is then computed, by means of the similarity measure previously defined, to find the most similar one and perform the merging operation. All of this computations (similarity and generality) are done in the context provided by $\mathcal{U}$, the subset of useful features obtained in the Reflection stage. This means that only features included in $\mathcal{U}$ are considered when computing both measures, which also use the available weights.

```
Let $P$ be a partition
Let $NG$ be the level of generality desired
Let $\mathcal{U}$ be the weighted subset of useful features

Function Refinement($P$, $NG$)
   U=Reflection($P$, $NG$)
   while Generality($P$, $NG$, $\mathcal{U}$) $< 0$ do
      Let $C$ be the least general cluster in P
      Compute the similarity between $C$ and the rest of clusters in $P$ using $\mathcal{U}$
      Merge $C$ with the most similar concept in $P$
      $\mathcal{U}$ = Reflection($P$, $NG$)
   endwhile
```

Table 1. Algorithm for the Refinement stage of ISAAC

The reason for using the generality measure as a heuristic to decide one candidate to merge is twofold. If we use a similarity matrix, since we want to dynamically adjust similarity computations with feature weights, we are forced to update the whole matrix at each step of the process. These computations will result in a cubic complexity for the algorithm with respect to the number of clusters initially considered. The heuristic used, allows to maintain a quadratic complexity for the algorithm. On the other hand, the heuristic should bias the algorithm to reach levels in which each cluster approximately corresponds to the same level of abstraction, and hence improve the understandability of the obtained clusters.

3 Empirical evaluation

The framework outlined in this paper may suggest several lines for evaluation, but in our experiments we focused on two aspects, the effect of the Preprocessing stage, and the effect of the feature selection mechanism. Particularly, we want to check the degree of compression that can be achieved in the first stage of the process and if this compression can decrease the quality of the final results. Also, we are interested in analyzing the results from the point of view of comprehensibility of the results to confirm the utility of feature selection.

The experiments were carried out using the mushroom dataset from the UCI repository. The mushroom dataset consists of 8124 mushroom descriptions represented by 22 nominal features belonging to two classes, edible and poisonous. ISAAC was run with a set of NG consecutive values with an 0.1 increment until achieving a two-class top level partition. Depending of the initial partition considered, the number of levels ranged between 7 and 9 in order to get the desired number of clusters.

To evaluate the quality of the discovered hierarchies, we measured the degree of fit of the resulting clusters in the top level with the original mushroom division into edible and poisonous. We measured cluster purity by using the measure suggested in [4].

α	# clusters	time	feat./node	Purity	Pur$\geq$0.85
0.75	29.18 $\pm$ 1.10	5.66 $\pm$ 0.17	4.94 $\pm$ 0.43	0.78 $\pm$ 0.10	40 %
0.77	33.82 $\pm$ 1.14	6.81 $\pm$ 0.15	4.50 $\pm$ 0.39	0.79 $\pm$ 0.10	46 %
0.80	54.10 $\pm$ 1.96	10.42 $\pm$ 0.24	4.08 $\pm$ 0.26	0.77 $\pm$ 0.09	26 %
0.82	82.50 $\pm$ 2.18	15.30 $\pm$ 0.29	3.85 $\pm$ 0.23	0.81 $\pm$ 0.10	58 %
0.85	157.54 $\pm$ 3.98	27.25 $\pm$ 0.51	3.64 $\pm$ 0.22	0.81 $\pm$ 0.09	58 %
0.87	263.04 $\pm$ 4.70	43.09 $\pm$ 0.80	3.45 $\pm$ 0.16	0.76 $\pm$ 0.11	36 %
0.90	610.28 $\pm$ 7.92	99.58 $\pm$ 1.80	3.10 $\pm$ 0.09	0.75 $\pm$ 0.11	26 %
0.92	1072.18 $\pm$ 12.89	202.60 $\pm$ 2.57	2.83 $\pm$ 0.05	0.74 $\pm$ 0.13	36 %
0.95	2327.36 $\pm$ 14.81	512.84 $\pm$ 4.65	2.15 $\pm$ 0.03	0.84 $\pm$ 0.09	66 %

Table 2. ISAAC results for different α values.

Table 2 shows the results for 50 ISAAC runs on the mushroom dataset using different α values. Some data from the table is graphically depicted in figure 2, showing that a great amount of compression may be achieved by just using α values around 0.85, which are relatively high. As expected, decreasing the α value allows for greater data compression. Obviously, the amount of data compression is directly correlated with the running times of the system. Figure 2 also shows the average purity scores and stardard deviations achieved by the system for each different α value. Clearly, there is no relationship between these two factors since results appear to be somewhat variable. In fact, we cannot expect to find such a relationship because different initial partitions may bias the subsequent feature selection steps in a different and not easily predictable manner. Actually, the interest here was in demonstrating that the system is able to reach high scores with some of the compressed datasets. The impact of the compression in the final results may vary between different datasets and users should experiment with different α values until obtaining a suitable partition.

For comparison purposes, we also run the well-known AUTOCLASS program [1], obtaining a purity score of 0.90. Since ISAAC results do not appear to follow a very homogeneous distribution, table 2 shows an additional column with the percentage of 'good clusterings' obtained with each α value, considering as good clusterings those with a purity score over 0.85. It is not possible to establish a direct comparison between the two systems, since ISAAC generates hierarchical clusterings as opposed to the flat clusterings of AUTOCLASS. However, the AUTOCLASS score indicates that, despite the variability of its results, our approach may achieve good quality clusterings with a reasonable amount of compression.

Table 2 also give us a picture of the capabilities of the feature selection mechanism. In average, the system only uses between a 10-20 % of features from the initial feature set in order to discriminate between the different hierarchy nodes. This demonstrates the ability of ISAAC to bias the clusterings towards simplicity and, hence, provides results which should be easier to interpret.

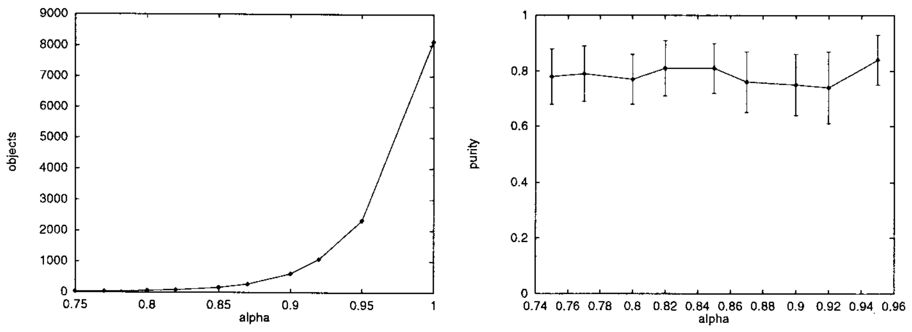

Fig. 2. Data compression (left) and purity of top level partitions (right) as a function of α.

4 Concluding remarks

We have presented ISAAC, a conceptual clustering system which combines statistical procedures with symbolic learning oriented heuristics. The system partitions the clustering process into three stages. The first one, accomplishes two goals, namely, to achieve a compression of the dataset, and to provide an initial set of hypotheses for the rest of the process. The following stages are performed in a collaborative manner. One of them is responsible of obtaining a weighted subset of features from the actual cluster structure. The second, uses this information to generalize the current partition and generates new cluster structures, from which new weights may be computed.

One novel feature in our approach is the definition of a cluster generality measure. This measure allows to parameterize the system in order to allow the user to bias the resulting structures towards the desired level of generality. We think that this sort of parameters are more meaningful to users than those in traditional statistical clustering. Users should be more comfortable interacting with systems which have parameters with a semantically clear interpretation [9].

Our preliminary experiments suggest that this approach achieves two interesting goals from the standpoint of data mining tasks. First, the system should efficiently deal with large datasets by means of the initial Preprocessing step. Second, the resulting hierarchical clusterings should be easy to understand since the system provides descriptions with an important reduction in the number of features used.

On the other hand, results seem to show a large variability with respect to the original labeling in the used dataset. However, we cannot assume the original labeling to be the only interesting underlying structure present in the dataset. Also, variability may be due to the global interaction between the different biases provided by the preprocessing and the feature selection mechanisms. This suggests a future research to explore the isolate and joint influences of all these biases.

Some aspects of the system deserve further research such as the similarity metrics used, the feature weighting measure or the feature selection strategy. Currently, some of these components are chosen largely based on intuitions and empirical results. Probably, a more accurate study of the properties of the different measures used can lead to a better understanding of the behavior of the system.

The modular design of ISAAC easily allows to extend the system. We plan to explore different feature selection measures and their impact in performance tasks like prediction. Also, a set of discretization procedures will be added to the Preprocessing stage to provide the system with the ability to deal with numerical valued features. Finally, we are working on extending the Reflection stage in order to constraint the Refinement process using declarative knowledge provided by the user.

Acknowledgments. This work has been supported by the Spanish Research Council (CICyT) project TIC96-0878.

References

1. P. Cheeseman and J. Stutz. Bayesian classification (autoclass): theory and results. In U. M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy, editors, *Advances in knowledge discovery and data mining*, pages 153–180. AAAI Press, Menlo Park, CA, 1996.
2. U. M. Fayyad, G. Piatetsky-Shapiro, and P. Smyth. From data mining to knowledge discovery: An overview. In U. M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy, editors, *Advances in Knowledge Discovery and Data Mining*, pages 1–34. AAAI Press, Cambridge, Massachusetts, 1996.
3. D. H. Fisher. Knowledge acquisition via incremental conceptual clustering. *Machine Learning*, (2):139–172, 1987.
4. D. F. Gordon, P. M. Tag, and R. L. Bankert. Unsupervised classification procedures applied to satellite cloud data. Technical Report AIC95-005, Navy Center for Applied Research in Artificial Intelligence, 1995.
5. S. J. Hanson and M. Bauer. Conceptual clustering, categorization and polymorphy. *Machine Learning*, (3):343–372, 1989.
6. M. Lebowitz. Experiments with incremental concept formation: UNIMEM. *Machine Learning*, (2):103–138, 1987.
7. R. López de Mántaras. A distance based attribute selection measure for decision tree induction. *Machine Learning*, (6):81–92, 1991.
8. R. S. Michalski and R. E. Stepp. Learning from observation: Conceptual clustering. In R. S. Michalski, J. G. Carbonell, and T. M. Mitchell, editors, *Machine Learning: An Artificial intelligence approach*, pages 331–363. Morgan Kauffmann, 1983.
9. L. Talavera and U. Cortés. Exploiting bias shift in knowledge acquisition. In *10th European Workshop on Knowledge Acquisition, Modeling, and Management*, Lecture Notes in Artificial Intelligence, Sant Feliu de Guixols, Barcelona, Spain, 1997. Springer.
10. L. Talavera and J. Roure. A buffering strategy to avoid ordering effects in clustering. In *Proceedings of the Tenth European Conference on Machine Learning*, volume 1398 of *Lecture Notes in Artificial Intelligence*, Chemnitz, Germany, 1998. Springer.