Generality-Based Conceptual Clustering with Probabilistic Concepts

Luis Talavera and Javier Béjar

Abstract—Statistical research in clustering has almost universally focused on data sets described by continuous features and its methods are difficult to apply to tasks involving symbolic features. In addition, these methods are seldom concerned with helping the user in interpreting the results obtained. Machine learning researchers have developed *conceptual clustering* methods aimed at solving these problems. Following a long term tradition in AI, early conceptual clustering implementations employed logic as the mechanism of concept representation. However, logical representations have been criticized for constraining the resulting cluster structures to be described by necessary and sufficient conditions. An alternative are *probabilistic concepts* which associate a probability or weight with each property of the concept definition. In this paper, we propose a symbolic hierarchical clustering model that makes use of probabilistic representations and extends the traditional ideas of specificity-generality typically found in machine learning. We propose a parameterized measure that allows users to specify both the number of levels and the degree of generality of each level. By providing some feedback to the user about the balance of the generality of the concepts created at each level and given the intuitive behavior of the user parameter, the system improves user interaction in the clustering process.

Index Terms— Conceptual clustering, hierarchical clustering, probabilistic concepts, user interaction.

1 Introduction

Hierarchical clustering methods are unsupervised learning techniques that construct tree structures reflecting the underlying patterns in a given data set. Commonly, an agglomerative strategy is used together with a similarity (or distance) measure to merge at each step of the process the most similar pair of objects. Several criteria may be selected to determine the similarity between newly formed clusters and some other clusters such as single-link or complete-link methods. In addition, there are a number of similarity measures to choose from. A number of clustering algorithms may be obtained by combining some criterion with some similarity metric [8].

The trees obtained by the clustering methods are called *dendrograms* and are typically binary. A dendrogram consists of several layers of nodes representing different clusters and are usually drawn to show the similarity between clusters. Data analysts can extract different partitions by cutting the dendrogram horizontally. The similarity values are often used to determine the validity of the groupings. The examination of the reduction of similarity required to go from one level to another may help to decide if the groupings appear to be natural in the studied domain.

Although these methods have been successfully applied to a number of problems, they may pose some difficulties to nonexperienced users, as reported by some machine learning researchers [5], [10]. The application of a clustering method is an iterative process with much of the search control left to the users, so they should be familiar with the statistical concepts involved in the clustering process. Particularly, to extract a partition from a dendrogram, users have to understand the behavior of the similarity metric used. This problem is aggravated if, instead of a flat partition, a nonbinary hierarchy has to be extracted from the dendrogram. In such a case, users must specify suitable similarity thresholds to obtain the desired partition, a rather difficult process, especially if the number of objects to be clustered is not trivially small.

Moreover, statistical research in clustering has often focused on data sets described by continuous features and most similarity measures are best suited to numerical data. Existing measures to deal with nominal data are mainly addressed to binary-valued features. Nominal data is often found in symbolic Artificial Intelligence (AI) domains, so it is not surprising that researchers in this area developed *symbolic clustering* methods to deal with these kinds of problems. Particularly, machine learning researchers have developed methods for *conceptual clustering* [3], [6], [9], [10] aiming to provide a better integration between the clustering and interpretation stages of the data analysis process. Conceptual clustering systems do not only evaluate clusters based on some metric, but also evaluate the "goodness" of the concepts represented by those clusters. In order to do that, these systems explicitly deal with concept descriptions and not only with extensional summaries of the clusters.

Following a long term tradition in AI, early conceptual clustering implementations used logic as the mechanism of concept representation. CLUSTER/2 is an example of this approach [10] that uses an iterative algorithm that modifies logical cluster descriptions until no further quality improvements can be made. Logic provides a well-known framework to represent concepts and, at the same time,

• *The authors are with the Departament de Llenguatges i Sistemes Informàtics, Universitat Politècnica de Catalunya, Campus Nord, Jordi Girona 1-3, 08034 Barcelona, Spain. E-mail: {talavera, bejar}@lsi.upc.es.*

Manuscript received 26 Dec. 1998; revised 10 Nov. 1999; accepted 17 July 2000.
Recommended for acceptance by T. Ishida.
For information on obtaining reprints of this article, please send e-mail to: tpami@computer.org, and reference IEEECS Log Number 108508.

allows quality measures to be defined easily, such as the simplicity or the fit of concept descriptions.

However, logical representations have been criticized by both machine learning and cognitive psychology researchers [3], [15] because they can only represent clusters defined by sufficient and necessary conditions. An interesting alternative are *probabilistic concepts* which associate a weight with each property of the concept definition. This representation permits us to describe domains that are not defined by all-or-none conditions and should be more robust in the face of uncertain data. A representative example is the COBWEB system [3] that builds concept hierarchies using a probabilistic objective function. This approach shows an advantage of probabilistic concepts: They lend naturally to use probabilistic defined measures. In this manner, we can use both a description formalism and a metric that capture more detailed information.

We have raised two issues, the use of probabilistic concepts in symbolic clustering and the need to ease user interaction in order to improve the interpretation task. In this paper, we address both issues by proposing a symbolic clustering model that provides the users with an intuitive method of tuning the results obtained in the clustering process under the framework of a probabilistic concept representation. We describe a hierarchical clustering model using probabilistic concepts that extends the ideas of specificity-generality typically found in machine learning. We define a probabilistic measure that makes it possible to characterize the generality of concepts and use this new measure to define heuristics that guide the clustering process. Moreover, the measure is parameterized in such a way that users may specify both the number of levels and the degree of generality of each level. By providing some feedback to the user about the balance of the generality of the concepts created at each level and, given the intuitive behavior of the user parameter, the system improves user interaction.

2 Preliminary Concepts

We provide here some preliminary definitions and concepts that will be used throughout the paper and that should help to follow subsequent discussion.

2.1 Symbolic Objects

In machine learning, symbolic objects are typically described using a feature-value representation. A *feature*, denoted X_i is a variable of discrete type, taking one of several *values*. The set of allowable values for a given feature is termed the *domain* of the feature. Formally, for a feature X_i, we represent its domain as $D_i = \{v_{i1}, v_{i2}, , v_{im_i}\}$.

An object is represented by a list of feature-value pairs defined over the global set of features. More formally, for a set of features $\{X_1, X_2, \ldots, X_n\}$ and their domains, an object is defined as $O = \{[X_1 = v_1], [X_2 = v_2], \ldots, [X_n = v_n]\}$, where $v_1, v_2, \ldots, v_n$ are taken from the domains $D_1, D_2, , D_n$, respectively.

For instance, if we consider the following set of features {*size, shape, color*} having the domains {*small, medium*}, {*square, pyramid, sphere*}, and {*black, blue, red*}, respectively, the following are object descriptions:

$$O_1 = \{[size = small], [shape = square], [color = blue]\}$$
$$O_2 = \{[size = medium], [shape = square], [color = blue]\}.$$

To simplify notation, we can see an object as a *feature vector* of the form $O = \{v_1, v_2, , v_n\}$, omitting feature names.

2.2 Logic-Based Representation of Concepts

A natural form of representing concepts in conceptual clustering is as conjunctive expressions of feature-value pairs. This is a simple extension of the object representation allowing *internal disjunction,* that is, a disjunction of values within a feature. Note that this addition is necessary in order to cover more than a single object with a concept description. Therefore, a symbolic concept C_k is a conjunction denoted as:

$$C_k = \{[X_1 = v_1], [X_2 = v_2], \ldots, [X_n = v_n]\},$$

where $V_i = \{v_{i1}, v_{i2}, \ldots, v_{ij}\}$ is either a value or a set of values denoting a disjunction.

For instance, the following description covers objects O_1 and O_2 in the example:

$$\{[size = \{small, medium\}],$$
$$[shape = \{square\}], [color = \{blue\}]\}.$$

Under this representation, the object description language becomes a subset of the concept description language. This is sometimes referred to as the "single representation trick." Notation can be further simplified by omitting references to features that can take any of the values in its domain. For instance, a description for objects O_1 and O_2 may omit the feature $size$.

2.3 A Partial Ordering of Concepts

The space of concept descriptions may be viewed as partially ordered by a *more-general-than* relation [11]. If a concept A covers all the objects in a concept B, along with other objects, then A is said to be more general than B. This space is bounded by the most general concept that includes all the objects and the set of most specific concepts in which each object represents a concept. The ordering facilitates the search through the hypothesis space, which depends on the object and concept description languages. The former must provide some method to compare generalizations by examining descriptions and without explicitly considering sets of objects.

For example, Fig. 1 shows a part of a space of concept descriptions. Concepts C_1 and C_4 cover object O_1, but C_4 turns out to be more general than C_1 because it covers a greater number of possible objects. C_1 and C_2 are not comparable because, although the sets of objects described intersect, neither set contains the other.

One can take advantage of this ordering when looking for concept descriptions by applying generalization and/or specialization operators which make it possible to move through the hypothesis space acting over concept descriptions. For instance, the technique of *dropping conditions* always produces more general concepts as less constrained descriptions cover a wider number of possible objects. On the contrary, adding more terms generates more constrained and, thus, more specific descriptions. In the

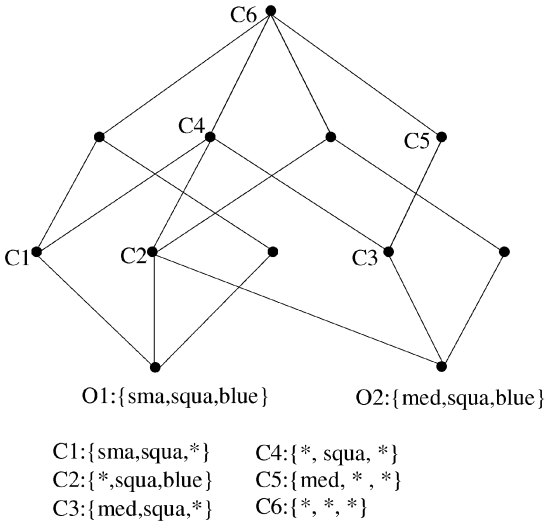

Fig. 1. A partial ordering of concepts.

example of Fig. 1, C_4 can be obtained by dropping the condition $[size = small]$ from C_1. Using these operators, we can traverse the concept space until finding good concepts according to some quality criterion.

3 Probabilistic Concepts

A key difference between conceptual and statistical clustering is that the former explicitly deals with concept descriptions. Following a widespread tradition in AI, early conceptual clustering systems used a logic-based representation language that provided clear semantics and understandable results. However, later research has raised problems with this sort of representation, mainly that they restrict the sort of concepts to be learned to those that can be described by sufficient and necessary conditions and that they can be more sensitive to noise and uncertainty [3], [15] .

Cognitive psychologists defined *probabilistic concepts* [15] which capture the distributional information of the feature values in a concept by associating a weight with these values. In this manner, it is possible for a concept to be characterized by neither necessary nor sufficient properties. Probabilistic concepts have been successfully adapted by machine learning researchers [3], [2].

Let us first introduce some notation for the rest of the discussion. Let n_{C_k} be the number of objects from concept C_k, n_{ij} the number of objects with the *j*th value of the *i*th feature, and I_{ijk} the number of objects from concept C_k with the *j*th value in the *i*th feature. Further, let $p_{ij|k}$ denote the conditional probability of an object having value j of a feature i in concept C_k and $p_{k|ij}$ denote the conditional probability of an object being in concept C_k if it has the *j*th value in the *i*th feature. A simple estimation of these terms is the observed frequencies in the data:

$$p_{ij|k} = \frac{I_{ijk}}{n_{C_k}} \quad p_{k|ij} = \frac{I_{ijk}}{n_{ij}}. \tag{1}$$

We define a probabilistic concept C_k as a list of conditional probabilities $p_{ij|k}$ for each possible feature-value pair $[X_i = v_{ij}]$. For instance, a probabilistic description of the concept C_2 covering objects O_1 and O_2 in Fig. 1 would be:

$$p(size = small \mid C) = 0.5$$
$$p(size = medium \mid C) = 0.5$$
$$p(shape = square \mid C) = 1.0$$
$$p(color = blue \mid C) = 1.0$$

Using this approach, concept descriptions can reveal subtle differences between concepts that would have the same description in the logic-based representation. For example, if we add a new object to the concept having small size, the logical description remains the same. However, the probabilistic description reflects the change by modifying the conditional probabilities corresponding to the feature $size$.

The probabilistic representation of concepts blurs the idea of a partially ordered space of concept descriptions. It is not immediately clear from any pair of concept descriptions which is more general. This occurs because the more-general-than relation is defined extensionally and relies on some matching procedure that determines the set of objects covered by a given description. For logical conjunctive descriptions, the matching procedure simply counts the number of objects satisfying the conditions in the description. This definition of the matching procedure makes it possible to compare the generality of two descriptions by just looking at their terms since it is known that a smaller number of terms will match a larger number of possible objects. However, for probabilistic concepts, object recognition does not rely on a matching procedure, but in a *partial matching* one. A simple procedure is to sum the weights of all concept feature values present in an object. If the sum passes a given threshold, the object is assumed to be recognized as a member of the concept. This procedure is typically augmented by using a competitive strategy that assumes that an object is recognized as member of the concept that maximizes the sum. Given two probabilistic descriptions, we are able to decide which one provides a "better" match for a given object, but there is not a clear boundary of membership.

In practice, the generality of a probabilistic concept is estimated extensionally by looking at the objects included in the concept. Probabilistic concepts can be arranged into a hierarchy [7] as depicted in Fig. 2 to reflect different levels of generality. Although the generality of each concept cannot be readily inferred from the description itself, it stems from the objects covered under each node.

Adopting a probabilistic representation of concepts still makes it possible to learn more "classical" concepts since conditioned probabilities subsume logical representations that only represent necessary and sufficient conditions. Probabilistic concepts should approximate all-or-none logic-based descriptions under noisy environments typically found in real-world problems. In addition, they can represent concepts that do not have fully sufficient and necessary conditions. Additionally, Smith and Medin [15] discuss some advantages of probabilistic representations

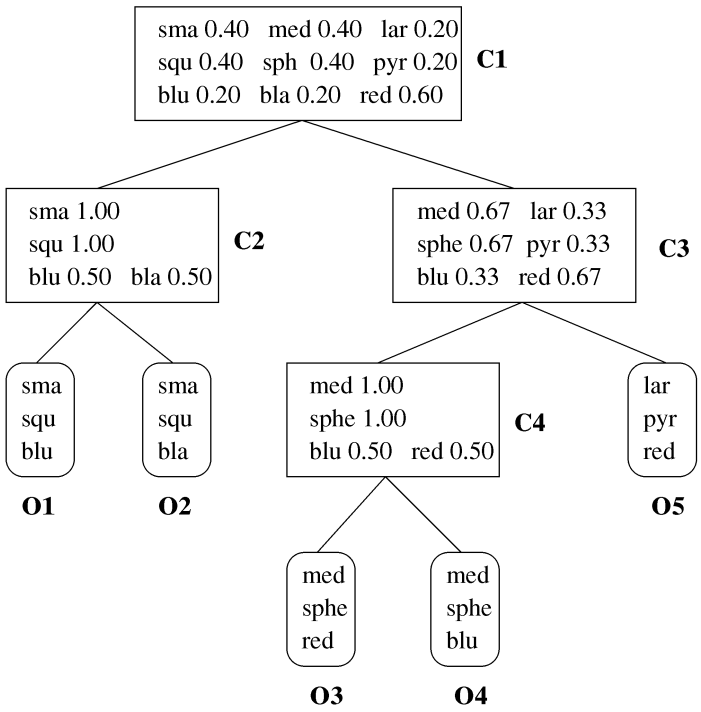

Fig. 2. An example of a hierarchical probabilistic representation.

over what they call the *classical view* of concepts in accounting for psychological phenomena.

4 Probabilistic Concepts and Concept Generality

Probabilistic concepts provide a more powerful representation than logic, but they lack the explicit specialization/generalization mechanisms suggested by the partial ordering of conjunctive descriptions. However, the idea of an ordering is still important when building concept hierarchies since most clustering systems are looking for a "good" level, neither too general nor too specific. In practice, while the notion of ordering is not explicitly used, it remains implicit in the learning process.

Basically, every clustering system aims to form clusters that maximize intracluster similarity and minimizes intercluster similarity. In the context of probabilistic representations, these two properties are usually measured for a given cluster k and a feature i and a value j by using the two conditional probabilities $p_{ij|k}$ and $p_{k|ij}$, respectively. It is well-known that the behavior of these measures is such that they tend to monotonically increase in an inverse manner to each other along the concept hierarchy. Very specific concepts are difficult to differentiate from other concepts and they tend to exhibit more values with low $p_{k|ij}$ scores. On the other hand, exemplars of specific concepts share a large number of properties and, therefore, these concepts have a large number of values scoring high $p_{ij|k}$. Conversely, more general concepts are easier to differentiate from other concepts, scoring high $p_{k|ij}$, but their members share fewer properties, thus scoring low $p_{ij|k}$. It is clear that the notion of generality is implicitly present in these considerations.

Motivated by this behavior, conceptual clustering systems using probabilistic concepts typically use metrics including some sort of trade-off between both probabilities. The category utility metric used in the COBWEB system is an example. The problem with this approach is that they rely on a fixed trade-off and do not allow users to intervene in the clustering process. Therefore, if the user is not satisfied with the results, it has to postprocess the resulting hierarchy.

Although traditionally it is considered that nonparameterized systems are better than parameterized ones, for some tasks, flexibility may be a more important concern than autonomy. It is worth stating at this point that some sort of parameters may not be adequate since they may not have a clear meaning for the user. For example, the typical approach in statistical clustering of specifying distance thresholds to decide the final levels of a hierarchy is not readily understandable for average users. Criteria may vary between different metrics and some statistical knowledge is needed in order to decide when differences between levels are significant. In order to achieve a reasonably easy interaction, parameters have to reflect some meaningful property of the concepts to be created. This should be easier to model in conceptual clustering systems since they include explicit representations of concepts.

4.1 Level of Generality of Probabilistic Concepts

Following these guidelines, we propose an alternative approach based on the notion of the *level of generality* of concepts. The idea is to take advantage of the previously discussed behavior of the probability distributions along the levels of a concept hierarchy in order to explicitly take into account the notion of generality. Although it is difficult to define a clear partial ordering for probabilistic concepts,

TABLE 1
Level of Generality (γ) for the Objects and Concepts in Fig. 2

O_1	O_2	O_3	O_4	O_5
0.20	0.33	0.20	0.20	0.43

C_1	C_2	C_3	C_4
0.89	0.57	0.76	0.50

we will show how the generality of a concept can be estimated from its probabilistic description and how this estimate can be used to heuristically guide the clustering process.

Following Fisher [3], we define the *predictability* of the set of features describing a given concept C_k as:

$$PBL(C_k) = \sum_i \sum_j p_{ij|k}^2, \tag{2}$$

where i indexes the different features in the data and j indexes the values for the domain of feature X_i. This expression estimates the expected number of feature values that one can correctly guess for an arbitrary member of cluster C_k, assuming a probability matching strategy. It assumes that one guesses a value with probability $p_{ij|k}$ and that this guess is correct with the same probability. As mentioned earlier, the expression can also be interpreted as an intracluster similarity measure.

The *predictiveness* of the features describing a given concept C_k is defined as:

$$PVN(C_k) = \sum_i \sum_j p_{k|ij}^2 \tag{3}$$

that, similarly to the previous measure, can either be viewed from a predictive viewpoint or as a measure of intercluster similarity. Taking these definitions and their expected behavior along a concept hierarchy as a basis, we define the *level of generality* γ of a probabilistic concept C_k as

$$\gamma(C_k) = \frac{PVN(C_k)}{PVN(C_k) + PBL(C_k)}. \tag{4}$$

We can expect the γ value to be roughly between 0 and 1. If PVN is much larger than PBL, then γ will approximate to 1. If PBL is much larger than PVN, then γ will approximate to 0. Therefore, as γ increases, we have more general concepts.

Table 1 shows the γ scores for the objects and concepts in Fig. 2 and the trends we can expect from the proposed measure. For example, all the objects are in a lower level of generality because of their high PBL scores, with one exception, object O_5. This is because this object possesses two properties, $[size = large]$ and $[shape = pyramid]$, that are sufficient to differentiate the object from other objects/concepts, thus scoring a high PVN compared with the rest of objects. In fact, the level of generality of this object description is close to the level of concept C_2, thus suggesting that the object might constitute a cluster by itself when considering certain level of abstraction.

4.2 A Measure of Generality

The notion of generality can be exploited to allow users to flexibly interact with the clustering algorithm. The user can provide a list of γ values so that the algorithm builds a level for each value representing the degree of generality desired. We have seen that the generality of a probabilistic concept will approximate 1 for more general concepts and 0 for more specific ones. Therefore, tuning the γ values should be reasonably easy for users in order to end up with a suitable partition for a given task since the behavior of the system with different parameter values is quite intuitive.

First, we can generalize the definitions of PVN and PBL for a partition $P = \{C_1, C_2, \ldots, C_K\}$ by averaging the individual scores of each cluster:

$$PVN(P) = \sum_k^K \frac{PVN(C_k)}{K} \tag{5}$$

$$PBL(P) = \sum_k^K \frac{PBL(C_k)}{K}. \tag{6}$$

Now, we can derive a generality measure indicating the relative generality of a given partition P as a function of a desired level γ by rewriting (4) as:

$$Gen(P, \gamma) = (1 - \gamma) * PVN(P) - \gamma * PBL(P). \tag{7}$$

If the partition is below the level indicated by γ, the measure yields a negative value; if it is above this level, it yields a positive value. The measure scores close to 0 when the level of generality of the concept is γ. For example, applying the measure to the example in Fig 2, we obtain the following scores for two given γ values:

$$Gen(\{C_2, C_4, O_5\}, 0.5) = 0.00$$
$$Gen(\{C_2, C_4, O_5\}, 0.67) = -2.72$$
$$Gen(\{C_2, C_3\}, 0.5) = 2.16$$
$$Gen(\{C_2, C_3\}, 0.67) = 0.01$$

It can be observed that the partition into three clusters is on the level of generality corresponding to a γ value of 0.5. The negative generality score for $\gamma = 0.67$ indicates that the partition is too specific for that level. Similarly, the partition into two clusters scores a high positive generality when using $\gamma = 0.5$, indicating that it is too general for this level. The right level is $\gamma = 0.67$ as suggested by the score close to 0.

We can exploit this measure either in agglomerative or divisive clustering algorithms. In agglomerative methods, initial partitions will score a negative value because of their specificity. Thus, merging—generalization—operations will make the measure increase and can be applied until obtaining a score close to 0. Analogously, for divisive algorithms, the initial score will be positive and successive divisions should decrease the score until reaching, again, a value close to 0.

As opposed to measures based in a trade-off between predictability and predictiveness, the generality measure is not specifically intended to be an objective function to guide the clustering process. Although using the same conditional probabilities, metrics such as the category utility commonly

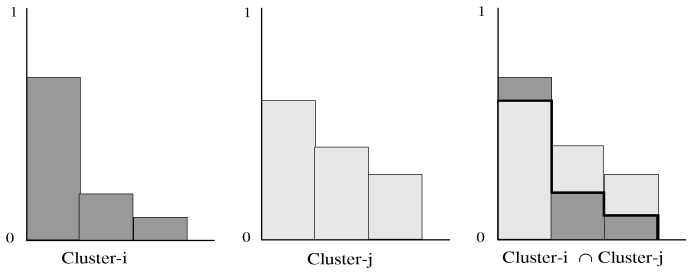

Fig. 3. Similarity between clusters as the intersection of probability distributions.

measure the information gain of forming clusters by subtracting base rates and adding weighting terms, so they should be more robust as objective functions (see, for instance, [3] for a detailed account of how category utility is derived). We cannot expect that a balanced trade-off between PVN and PBL using a $\gamma = 0.5$ should provide similar results to these objective functions for every domain. However, the generality measure allows different levels of abstraction to be *characterized* in a manner that should be understandable by many users. In this manner, they can obtain some feedback about the structure of different potential levels to be included in the final hierarchy, as we will show in the next sections.

5 Generality-Based Clustering

The generality measure fits naturally with a *specific-to-general* learning scheme. Given a set of γ values specified by the user, we start with the set of most specific concepts—singleton concepts—and generalize this partition until reaching the desired degree of abstraction. Intermediate mergings need not to be retained because the desired levels in the resulting hierarchy are specified in advance. The resulting level is then stored and again a generalization process is started to obtain the next level. Repeating this process for each γ value, we obtain the number of levels specified by the user.

From an extensional point of view, generalization is a matter of merging clusters into larger clusters, so we can take advantage of existing agglomerative schemes. This is what we propose in our *Generality-based Concept Formation (GCF)* model. Typically, agglomerative methods merge at each step of the process the most similar pair of clusters, removing both clusters and creating a new one. These operations are performed by looking at and updating a distance—or similarity—matrix, without explicitly considering cluster descriptions. From the conceptual clustering point of view, we need to deal with cluster descriptions, so we need to define some specific similarity metric between probabilistic representations used for both objects and clusters.

5.1 A Similarity Measure

Intuitively, we can consider that two probabilistic descriptions are similar when the probability distributions that they represent are also similar. If we represent probabilistic descriptions as histograms, a natural way of represent the degree of coincidence is to draw a new histogram containing the intersection of the two given histograms, as shown in Fig. 3. To get a numerical measure of this coincidence, we can take the area of the resulting histogram. More formally, we can define the similarity between two clusters C_m and C_n for a given feature X_i with a domain with J values as:

$$Sim_a(C_m, C_n, X_i) = \sum_{j}^{J} min\{p_{ij|m}, p_{ij|n}\}. \tag{8}$$

We can easily generalize this measure for a set of I features by averaging the individual similarity for each feature as follows:

$$Sim(C_m, C_n) = \frac{\sum_i^I Sim_a(C_m, C_n, X_i)}{I}. \tag{9}$$

An object can be considered as a particular case of a probabilistic description, where one of the values has probability 1 and all the other values 0. When applied to individual objects, this measure reduces to one minus the Hamming distance, giving 1 if the value for a given feature is the same for both objects, and 0 otherwise. Thus, we can see the measure as the inverse of a *generalized Hamming distance*.

The main reason for adopting this similarity measure is that it is computationally cheap as compared to alternative measures that involve more complex calculations (e.g., mutual information). This is an important concern in agglomerative algorithms since most of the computational complexity of these algorithms stems from the computation of similarities. However, note that the framework proposed does not impose any constrain on the choice of the similarity measure.

Taking some objects of Fig 2 and computing their similarities, we obtain:

$$Sim(O_1, O_2) = 0.66$$
$$Sim(O_1, O_3) = 0,$$

which is just the number of common feature values between each pair of objects divided by the total number of features. Similarly, we can give some examples of similarities between clusters and between clusters and objects:

$$Sim(C_2, C_3) = 0.33$$
$$Sim(C_4, O_5) = 0.50.$$

5.2 The Algorithm

We have proposed a modification of a basic agglomerative procedure to deal with probabilistic cluster descriptions,

removing the binary tree constraint associated to dendrograms and adding a generality-based stopping criterion. However, the generality measure provides additional knowledge that could be profitably used during the clustering process. Since the measure is averaged over all clusters, it is possible to obtain partitions in which some clusters are very general and others very specific, still giving a score close to 0. A simple heuristic can be applied to avoid this effect consisting of selecting as the first candidate to merge at each step of the process, the cluster scoring the lowest generality. The second cluster chosen is the cluster that is more similar to the first candidate. In this way, we should bias generalization toward the generation of balanced levels formed by concepts of approximately the same level of abstraction. It is important to see that, when we speak of the same level of abstraction, we refer to the generality of the concept descriptions, not necessarily to the number of objects included in a cluster. For instance, a partition dividing animals into birds, all the mammals but dogs, and only dogs is an example of a set of unbalanced descriptions, because dogs are very similar to the rest of mammals and a cluster containing only dogs should be placed at a lower level. However, if the cluster of dogs is a cluster of dolphins instead, the partition could make sense since dolphins are a very special case of mammals. Although based in a traditional statistical algorithm, the GCF algorithm differs from these approaches in that it bias the learning process by explicitly "reasoning" over probabilistic descriptions. That is, it makes decisions based on the similarity and generality of the characterization of the clusters rather that only focusing on extensional concerns.

The final algorithm for the described procedure is the following:

1. Let $P = \{C_1, C_2, \ldots, C_K\}$ be the initial partition, where each C_i can be either a cluster from a previously created level or an object. Let γ, the level of generality specified by the user.
2. Find the cluster C_{min} with the minimum generality.
3. Compute the similarities between C_{min} and the rest of the clusters in P using (9). Let C' be the cluster scoring the highest similarity with C_{min}.
4. Reduce the number of clusters in P by 1, merging the objects in C_{min} and C' into a new cluster. Compute a new probabilistic description for this cluster.
5. If $Gen(P, \gamma) < 0$, go to Step 2; otherwise, return P.

The algorithm presents only the generalization procedure applied to one γ value, existing as an outer loop that iterates over the parameter values specified by the user.

An additional benefit of the proposed algorithm is that, by applying the heuristic discussed above, the time complexity is quadratic with respect to the number of objects n without any particular optimization. The generality of clusters can be computed in $O(n)$ because there are at most n (singleton) clusters. Similarity of the least general cluster with the other cluster is also bounded by $O(n)$ because the similarity function will be called at most $n-1$ times. Since the maximum number of iterations is $n-1$, the overall time complexity of the algorithm is bounded by $O(n^2)$.

Finally, in order to help users decide which are the best levels in the hierarchy, we provide a measure of how balanced the generality scores of the concepts of a given partition are. The rationale is that partitions with concepts with approximately the same level of abstraction should be more comprehensible. This helps users in deciding which γ values are the best. Of course, in a real setting, users may have some particular criterion to select the useful levels and they may experiment with several γ values. The degree of balance of the generality of a set of clusters can be measured by computing the dispersion of the list of generality scores for each individual cluster. Specifically, we use a statistical measure called *variation coefficient (VC)* that is defined for a set of values with mean $\bar{x}$ and standard deviation s as $s/\bar{x}$. Unlike the standard deviation, which is sensitive to the absolute scale of the measurements, the variation coefficient removes the influence of the magnitude of the data. In this manner, we avoid artificially promoting partitions scoring lower generality scores.

5.3 A Simple Example

To illustrate the operation of the algorithm, we will use the example in Fig 2. Let us assume that we are looking for a two-level hierarchy specified by the set of parameters $(0.5, 0.65)$. First, the object with the lower generality score is selected. In this case, there is a tie between O_1, O_2, and O_4, so one of them is arbitrarily chosen. Let's assume the algorithm chooses O_1. Similarities with all the other objects are computed, giving:

$$Sim(O_1, O_2) = 0.66$$
$$Sim(O_1, O_3) = 0$$
$$Sim(O_1, O_4) = 0.33$$
$$Sim(O_1, O_5) = 0.$$

Therefore, object O_2 is chosen as the candidate to merge with O_1. As a result of the merging, the resulting partition $\{C_2, O_3, O_4, O_5\}$ scores a generality of –2.25, thus indicating that it is still below the specified level. Again, the least general object/cluster is selected, in this case, either O_3 or O_4. Let us assume that the algorithm chooses O_3. The similarities with the rest of objects/clusters are:

$$Sim(O_3, O_4) = 0.66$$
$$Sim(O_3, O_5) = 0.33$$
$$Sim(O_3, C_2) = 0.$$

Now, object O_4 is chosen as the candidate to merge with O_3, obtaining cluster C_4. The generality of this new partition for $\gamma = 0.5$ is 0, stopping the process and returning $\{C_2, C_4, O_5\}$ as the first level of the two specified by the user. Next, the algorithm looks for the following γ value, in our case, 0.65. Generality of the three-cluster partition for this level scores –2.40, so the algorithm selects the least general cluster which is O_5. The computed similarities with the rest of clusters are:

$$Sim(O_5, C_2) = 0.00$$
$$Sim(O_5, C_4) = 0.50.$$

TABLE 2
Artificial Data Set Used in Experiment 1

Level 1	Level 2	Level 3	X1	X2	X3	X4	X5
1****	*11**	1111*	1	1	1	1	1
			1	1	1	1	2
		1112*	1	1	1	2	1
			1	1	1	2	2
	**23*	1223*	1	2	2	3	1
			1	2	2	3	2
		1323*	1	3	2	3	1
			1	3	2	3	2
2****	*43**	2431*	2	4	3	1	1
			2	4	3	1	2
		2434*	2	4	3	4	1
			2	4	3	4	2
	*5*5*	2545*	2	5	4	5	1
			2	5	4	5	2
		2555*	2	5	5	5	1
			2	5	5	5	2

Object O_5 and cluster C_4 are merged into cluster C_3. The generality of the partition $\{C_2, C_3\}$ is 0.27. The process stops because the generality is positive. Note that the score suggests that the result corresponds to an upper level of $\gamma = 0.65$, but it is still closer to the desired level of generality than the previous level.

6 Experimental Results

We conducted several experiments to test the GCF model, two of them using artificial data and the last one using data sets from the UCI Repository of ML Datasets [1]. The first experiments show how the generality measure effectively helps in discovering meaningful levels in a hierarchy by tuning the γ parameter. The last experiment compares the GCF model with the well-known COBWEB clustering algorithm.

6.1 Artificial Data

In our first experiment, we used an artificial data set consisting of 16 objects described by five symbolic features taking two or five values each and used by Murphy and Smith in psychological experiments [12]. The interest of this data set stems from the fact that it exhibits an underlying hierarchical structure of three levels with two, four, and eight classes, respectively. Table 2 shows the objects in the data set and the three-level hierarchical structure. The table shows, for levels 1 and 2, the features that are necessary and sufficient for the cluster descriptions. In level 3, no single feature holds these properties, so the conjunction of features characterizing the level is shown instead.

We generated levels starting from $\gamma = 0.025$ and using 0.025 increments and recorded the VC score for each level. Fig. 4 depicts the different VC values for levels with different number of clusters. Clearly, partitions into two, four, and eight clusters are the ones showing a more balanced generality among its components, thus suggesting the set of γ parameters $(0.25, 0.50, 0.75)$ as a reasonable choice. In fact, using this set of parameters, the system recovered a

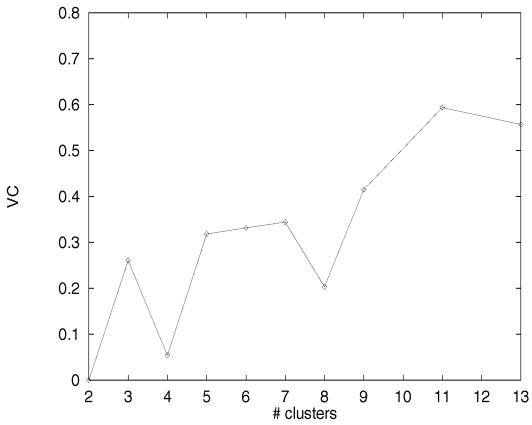

Fig. 4. Evolution of the variation coefficient for partitions with different number of clusters in the Murphy and Smith artificial data set.

hierarchical structure in full agreement with the one shown in Table 2. This demonstrates that our approach can effectively help users in deciding the final structure of the hierarchy. In case users are not satisfied, they can easily change some of these parameters because they can understand the effect that these changes will have in the resulting hierarchy. Moreover, by analyzing both the degree of balance of the concepts at each level and their global generality, we can conclude that the middle level is probably the best since its VC score is very low and it is obtained with $\gamma = 0.5$, thus providing a balanced trade-off between predictive and predictable features. Note that this is a particularly well-structured domain and that, in other domains, we might have to choose between well-balanced levels without such a perfect trade-off between PVN and PBL.

In our second experiment with artificial data, we generated 1,000 instances of a data set containing 10 symbolic features with six values each. The data set was structured into two classes described for each five different rules. In order to be able to represent this sort of disjunctive descriptions, hierarchical clusterers typically have to construct one internal cluster for each particular rule, independently, that more general upper levels might exist. Furthermore, we should expect a level with 10 clusters to be one of the more comprehensible levels—if not the best—since it is the one that reflects the individual rules. To make the problem more complex, each rule included four conjunctions containing one internal disjunction each, for instance:

```
if (D=1 or D=4) and (E=1 or E=2) and
  (F=2 or F=4) and
(G=2 or G=5) and (J=2 or J=5) then C1
```

We ran the GCF algorithm for several γ values and using several random orderings, resulting in the set of parameters (0.2, 0.9) scoring significantly lower VC. Using these values, the system tended to generate a hierarchy with two clusters at the top level and 10 clusters at the second. Clusters at the top level did not correspond exactly to the two-class division, classifying correctly only a 73 percent of the objects. However, the 10 clusters at the second represented all the disjunctive rules in the domain, classifying correctly almost 90 percent of objects on average. For comparison

purposes, we ran COBWEB over the same data set using several orderings. On average, COBWEB obtained a top level with four clusters and correctly classified only a 68 percent of the objects, thus demonstrating that the two classes of this domain were too complex to be discovered at very high levels. However, as opposed to the GCF algorithm, COBWEB was not able to create intermediate nodes corresponding to the disjunctions in the rules. We repeated the experiment with simpler domains by reducing the number of disjunctive rules. In each case, the GCF algorithm was able to find the level representing each individual disjunction as one of the better balanced levels as regards generality.

The conclusions of this experiment are twofold. First, the GCF scheme appears to be guided by different biases than COBWEB and, probably, as it is well-known in supervised settings, none of the algorithms is superior for every domain. Second, this experiment demonstrates the limitation of fixed trade-off approaches. Probably, users might come up with a reasonably good hierarchy by reorganizing the levels of the tree constructed by COBWEB, but our approach provides an easier method to experiment with different levels and choose the more suitable one.

6.2 UCI Data Sets

To assess the performance of the GCF algorithm in nonartificial data, we ran experiments on data sets obtained from the UCI Machine Learning Repository. Since our aim is exploring how well the induced hierarchies reflect the structure of each data set, examining only one level appears to be inadequate. An alternative is to use the descriptions stored at each node and an objective function or similarity measure to predict the class of the objects in a separate testing set. In this manner, we can see if an algorithm that does not provide a good partition at the top level can take advantage of the specialization provided by the hierarchy. In a real setting, the labels would be provided by an external user after the clustering is created. In the experiments, it appears reasonable to label nodes with the modal value of the class assuming that this labeling is the one that a user would provide. This provides a better estimate of performance in domains with classes in which the clustering algorithm needs to create internal disjuncts at inner levels in order to recover the structure of the data.

However, this procedure can provide a wrong picture of the quality of the resulting hierarchy. If we make predictions at the leaves of the hierarchy since they contain individual observations, a sort of instance-based prediction is performed. To be able of make such use of a clustering in a real problem, a user should label every leaf in the tree, which appears to be unfeasible. It is likely that users would select only part of the hierarchy to describe the target domain so that only a limited number of levels would be labeled. A more informative test is to make predictions by traversing the hierarchy to a limited depth. Additionally, this procedure can provide some insight into the comprehensibility of the recovered structure since hierarchies that need fewer nodes to make good predictions should be more understandable for users.

We ran the GCF algorithm on each data set and made predictions by sorting objects to a limited number of levels. We generated trees of different sizes by selecting, first, only the best γ, then, the two best γ, and so on. The values for γ were selected according to their VC scores. It is worth to noting that each additional level has not necessarily to significantly increased the size of the tree. For instance, adding a new γ value may either result in a single merging and, therefore, add just an additional node to the hierarchy, or select, a full set of new nodes. The result depends on the proximity of the γ values selected. We predicted the class of each unseen instance returning the modal class value found in the deepest node reached. Obviously, labels were used only for testing, but hidden during training.

Again, we ran COBWEB for comparison purposes. The system is intended to build a complete hierarchy automatically and does not provide a mean for selecting levels. Some extensions to COBWEB that are able to make predictions at different levels of the hierarchy have been proposed, but they are not useful for the label prediction task used in our experiments. Fisher [4] proposed a method for "pruning" the hierarchy by detecting different prediction frontiers using a separate validation set. The method requires the predicted features to be known by the system, which is the case for the flexible prediction task used by Fisher, which predicts every feature present in the data. This method cannot be applied in our experiments since the target feature (the label) is hidden during training. Analogously, Reich and Fenves [14] developed an extension of COBWEB called BRIDGER that makes predictions at different depths. The system describes the nodes in the hierarchy using only their characteristic properties, which are property values whose conditional probabilities exceed a predetermined threshold. When sorting an instance to make a prediction, the process stops when all the features describing the instance are matched at some point of the path with some characteristic property. Reich and Fenves report a successful application of this scheme to design tasks. However, we conducted some preliminary experiments with our data sets using the threshold suggested in [14] and did not obtain similar results. Predictions were made at roughly the same depth as using the plain COBWEB algorithm. Probably this behavior is due to the fact that, for the design task in which the method was tested, the number of properties to match was significantly lower than in our data sets. Therefore, we resorted to a much simpler method in which we made predictions at different depths by simply constraining the maximum number of levels traversed.

Results were obtained by performing 10 repetitions of a 10-fold cross validation and the same folds were used for both algorithms. Additionally, we conducted a *paired t-test* to assess the significance of the differences between the algorithms. However, this test is just an approximation and its results should be taken carefully since the testing procedure violates some independence assumptions [13].

Table 3 shows the results for both systems, including the average accuracy and number of nodes examined to make the predictions with the standard deviations. The average number of nodes examined can be viewed as a measure of the complexity of the hierarchy. A $+$ or a $-$ sign in the last column indicates that the accuracy of GCF is better or worst

TABLE 3
Predictive Accuracies and Number of Nodes Examined in Prediction for GCF and COBWEB for Different Tree Depths

	GCF		COBWEB		
Data set	Accuracy	#Nodes	Accuracy	#Nodes	test
bcw	90.47 (0.62)	2.74 (0.19)	94.78 (0.17)	2.00 (0.00)	−
	92.36 (0.34)	5.20 (0.46)	94.76 (0.17)	5.17 (0.31)	−
	92.36 (0.42)	7.19 (0.41)	95.18 (0.74)	8.64 (0.52)	−
bupa	56.74 (1.18)	2.62 (0.23)	58.78 (2.07)	2.99 (0.51)	−
	61.92 (1.82)	6.84 (0.75)	63.67 (2.31)	6.04 (0.49)	−
	62.01 (1.96)	7.44 (0.66)	62.21 (3.04)	9.22 (0.46)	
cleve	82.46 (0.67)	2.05 (0.05)	81.70 (1.51)	2.02 (0.06)	+
	82.09 (1.41)	4.75 (0.78)	81.21 (2.37)	5.09 (0.30)	
	81.80 (1.12)	7.79 (1.30)	80.69 (1.78)	8.71 (0.33)	+
crx	81.30 (1.20)	2.89 (0.69)	78.77 (3.05)	2.97 (0.09)	+
	82.16 (0.94)	5.66 (0.92)	80.84 (1.91)	5.81 (0.26)	+
	82.42 (0.92)	6.39 (0.57)	82.48 (1.92)	9.11 (0.29)	
glass	44.89 (1.59)	2.64 (0.22)	42.68 (3.92)	2.22 (0.34)	+
	58.01 (1.40)	5.82 (0.19)	56.19 (3.22)	5.07 (0.30)	+
	59.08 (1.04)	6.10 (0.24)	57.73 (2.70)	8.29 (0.43)	+
horse	66.68 (3.01)	2.57 (0.24)	63.56 (2.21)	2.05 (0.07)	+
	71.61 (3.11)	4.97 (0.86)	75.88 (2.16)	5.14 (0.26)	−
	72.56 (2.09)	6.82 (1.01)	75.99 (2.62)	8.72 (0.41)	−
pima	65.90 (0.86)	2.62 (0.21)	64.72 (1.39)	2.66 (0.37)	+
	67.63 (1.06)	5.16 (0.77)	66.05 (1.81)	5.70 (0.54)	+
	68.58 (1.29)	7.80 (0.90)	67.64 (1.86)	8.87 (0.57)	+

respectively than that of COBWEB at the 90 percent level of significance.

Results show that none of the systems is the best for every domain, thus suggesting that each one may be better suited for a particular type of problem. GCF performs better than COBWEB in three domains (cleveland, glass, and pima), while performing worse in other three domains (breast cancer wisconsin, bupa, and horse). On the crx data set, performance is roughly the same for both algorithms, but COBWEB needs to build a deeper hierarchy to attain good results. For this data set, COBWEB needs to examine more than nine nodes on average to attain the same accuracy that GCF provides by traversing only about six nodes. Considering the best accuracies for both systems on each data set, GCF needs to examine an average of six nodes. COBWEB traverses 6.6 nodes in average. This suggests a very desirable bias in the level construction and selection procedure of GCF towards compact and simple trees.

In sum, results confirm that the GCF model can build clusterings with similar quality to major conceptual clustering systems such as COBWEB. These results are remarkable since COBWEB uses an objective function that implicitly weights features and it is intended to produce predictive clusterings. As opposed, similarity metrics are sensitive to the choice of features and may need some information about feature weights to be more robust. We think that this can be a possible reason for COBWEB performing much better than GCF in some domains. Moreover, the generality-based selection of levels appears to be a good heuristic to obtain simple but accurate clusterings. This suggests an interactive procedure in which users may start with a set of levels with low VC scores and then tune the different γ values to suit their needs.

7 Related Work

Although user interaction has been a largely neglected topic in conceptual clustering research, it is worth mentioning again Reich and Fenves work on BRIDGER [14]. They developed an extension to COBWEB that provided users with a mechanism called "hierarchy correction" to modify the clusterings built by the system. The process uses critical properties provided by the user to guide a reorganization of the clusters in the hierarchy. This sort of user interaction differs from our work in that it requires *background knowledge* to be available from the user, in this case, a set of critical properties. Our approach can be deemed a knowledge-weak approach to user interaction since it does not require any background knowledge to operate. Of course, users might employ implicitly some knowledge when tuning the parameter values of the system, but they are not required to explicitly provide any information.

8 Conclusions

We described a conceptual clustering algorithm that builds probabilistic concept hierarchies as an alternative to the logic-based representations. Probabilistic concepts subsume logical ones and facilitate representing clusters that are not described by fully necessary and sufficient conditions. This work differs from previous approaches in that it explicitly uses the notion of generality applied to probabilistic

concepts by means of a measure that indicates the level of generality of a concept description. We have shown how this measure can be easily adapted in order to obtain a parameterized generality measure that allows partitions to be characterized at different levels of abstraction. By incorporating this latter measure to an agglomerative procedure and giving some feedback, we provide the user with an intuitive method to tune the results of the clustering process. Modifying the system parameter, the user can specify both the number of desired levels and their degree of generality in the final hierarchy. In addition, the generality measure has proven to be useful in defining internal heuristics to guide the clustering process.

Although results are very encouraging, we think that there are a number of issues that deserve further research. The focus of this paper was in presenting the general framework for the generality-based clustering so that we have committed to the choice of a particular similarity measure without considering alternatives. Although the proposed measure is very attractive due to its low computational cost and it appears to work fairly well, the framework allows other measures to be used and a more exhaustive comparison with some of these measures should be done.

Another issue raised by this work is that some conceptual clustering algorithms may be better suited for some problems than others. In supervised learning, there are several works comparing different approaches and characterizing in which domains and under which conditions a method may perform better. A similar comparison for conceptual clustering methods would be of great interest in order to see, for instance, if bottom-up approaches may be guided by different biases than top-down methods.

Finally, it is well-known that similarity-guided learning methods are very sensitive to the choice of features. We think that this is also true for the probabilistic characterization of generality proposed here. Although the generality measure resembles other metrics used in conceptual clustering, the latter usually measure the information gain of forming clusters by subtracting base rates and adding weighting terms and are less sensitive to imperfect data. Future work should explore how feature selection and/or weighting mechanisms impact in the performance of the proposed framework.

ACKNOWLEDGMENTS

The authors would like to thank the anonymous referees for many valuable comments on earlier drafts that greatly improved the quality of the paper. This work is partially supported by UPC grant PR99-09.

REFERENCES

[1] C. Blake, E. Keogh, and C.J. Merz, "UCI Repository of Machine Learning Databases," Univ. of California, Irvine, Dept. of Information and Computer Sciences, 1998. http://www.ics.uci.edu/~mlearn/MLRepository.html.

[2] D. Fisher, L. Xu, J. Carnes, Y. Reich, S. Fenves, J. Chen, R. Shiavi, G. Biswas, and J. Weinberg, "Applying AI Clustering to Engineering Tasks," *IEEE Expert*, vol. 8, pp. 51-60, 1993.

[3] D.H. Fisher, "Knowledge Acquisition via Incremental Conceptual Clustering," *Machine Learning,* no. 2, pp. 139-172, 1987.

[4] D.H. Fisher, "Iterative Optimization and Simplification of Hierarchical Clusterings," *J. Artificial Intelligence Research,* vol. 4, pp. 147-179, 1996.

[5] D.H. Fisher and M.J. Pazzani, "Computational Models of Concept Learning," *Concept Formation: Knowledge and Experience in Unsupervised Learning*, D.H. Fisher, M.J. Pazzani, and P. Langley, eds. pp. 3-43, San Mateo, Calif.: Morgan Kauffmann, 1991.

[6] S.J. Hanson and M. Bauer, "Conceptual Clustering, Categorization, and Polymorphy," *Machine Learning,* no. 3, pp. 343-372, 1989.

[7] W. Iba and P. Langley, "Unsupervised Learning of Probabilistic Concept Hierarchies," technical report, Inst. for Study of Learning and Expertise, Palo Alto, Calif., 1999.

[8] A.K. Jain and R.C. Dubes, *Algorithms for Clustering Data.* Englewood Cliffs, N.J.: Prentice Hall, 1988.

[9] M. Lebowitz, "Experiments with Incremental Concept Formation: UNIMEM," *Machine Learning,* vol. 2, pp. 103-138, 1987.

[10] R.S. Michalski and R.E. Stepp, "Learning from Observation: Conceptual Clustering," *Machine Learning: An Artificial Intelligence Approach,* R.S. Michalski, J.G. Carbonell, and T.M. Mitchell, eds, pp. 331-363, San Mateo, Calif.: Morgan Kauffmann, 1983.

[11] T.M. Mitchell, "Generalization as Search," *Artificial Intelligence,* vol. 18, pp. 203-226, 1982.

[12] G.L. Murphy and E.E. Smith, "Basic Level Superiority in Picture Categorization," *J. Verbal Learning and Verbal Behavior,* vol. 21, pp. 1-20, 1982.

[13] Y. Reich and S.V. Barai, "Evaluating Machine Learning Models for Engineering Problems," *Artificial Intelligence in Eng.,* vol. 13, no. 3, pp. 257-272, 1999.

[14] Y. Reich and S. Fenves, "The Formation and Use of Abstract Concepts in Design," *Concept Formation: Knowledge and Experience in Unsupervised Learning,* D.H. Fisher, M.J. Pazzani, and P. Langley, eds., pp. 323-353, San Mateo, Calif.: Morgan Kauffmann, 1991.

[15] E.E. Smith and D.L. Medin, *Categories and Concepts.* Cambridge, Mass.: Harvard Univ. Press, 1981.

Luis Talavera received the BS and MS degrees in computer science from the Technical University of Catalonia in 1994 and 1996, where he has been a part-time lecturer since 1995. He also works in the industry on web mining and online personalization issues. His research interests are on machine learning and data mining, focusing in unsupervised learning topics including clustering, feature selection, use of background knowledge, and incremental learning.

Javier Béjar received the BSc degree in computer science from the Universitat Politecnica de Catalunya in 1990 and the PhD degree in computer science from the Universitat Politecnica de Catalunya in 1995. He has been an assistant professor in the Software Department at the Universitat Politecnica de Catalunya since 1990. His research interests are in the machine learning and data mining areas. He developed his PhD thesis on algorithms for unsupervised machine learning (conceptual clustering). His research interests and publications are within the areas of automatic data analysis, data mining, and machine learning techniques applied to the construction of knowledge bases in real domains. He also works in the area of autonomous agents for electronic commerce.