CONCEPTUAL CLUSTERING USING RELATIONAL INFORMATION

Bernd Nordhausen

Department of Information and Computer Science
University of California, Irvine
Irvine, CA 92717
ArpaNet: bernd@ics.uci.edu

Abstract

Work in conceptual clustering has focused on creating classes from objects with a fixed set of features, such as color or size. In this paper we describe a system which uses relations between the objects being clustered as well as features of the objects to form a hierarchy tree of classes. Unlike previous conceptual clustering systems, this algorithm can define new attributes. Using relational information the system is able to find object classifications not possible with conventional conceptual clustering methods.

1. Introduction

Conceptual clustering involves grouping objects into conceptually similar classes and producing a characterization of those classes. In recent years there has been active research in the area of conceptual clustering. For a survey of several conceptual clustering systems, see [2]. All of these systems have focused on feature descriptions of the objects, such as color or size, to form a coherent classification. Only Stepp & Michalski [7] have left this narrow domain and used structural description of objects, i.e., attributes of object components and the relationship among these components to form classes.

However, no system thus far has used *relational* information to classify the set of objects. This paper describes a system called OPUS implemented in Prolog, which addresses this issue by using relations over the set of objects (and not simply object components as in structural description), as well as features of objects, to form classes. We thus extend the definition of conceptual clustering [6] to include relational information.

Given:

- A set of objects
- A set of features describing the objects
- A set of relations between the objects
- Criteria to evaluate the quality of a classification

Find:

- A hierarchy of classes and a characterization of the classes

Using relational information, the OPUS system eliminates a deficiency of previous conventional clustering systems; unlike the other systems, this system is able to distinguish between objects which have the same features but different relations. For example, in the domain of genetics, OPUS is able to classify peas not only in terms of their color but also in terms of their offspring, effectively defining the class of hybrids and purebreds. Another deficiency of other conceptual clustering systems is the inability to create new attributes; all attributes used to characterize objects have to be given to such systems. In contrast, OPUS is able to generate attributes if it determines that the current description of the objects is not sufficient. New attributes are defined as *chunks* formed from relations and features.

In the next section we describe the OPUS system, detailing the use of relations to form a classification and the generation of new attributes. In the third section we give two applications to illustrate the system. We conclude with two proposals for extending this work.

2. The OPUS System

The input to the OPUS system consists of the objects to be classified, a set of features describing the objects, and a set of relations over the object set, such as *eat* or *parent*. The system generates a hierarchical tree of classes, each class having a unique conceptual description. The system divides the object set into mutually exclusive classes, and recursively divides the classes until a final partitioning is found. At first, features such as *color* or *size* are used as attributes to form classes. After the list of current attributes is exhausted (i.e., all members of a given class have the same value for the given features), new attributes are generated. Using these new attributes, the clustering algorithm refines the previously formed classes until all members of the classes have the same value for all current attributes. OPUS continues the cycle of generating attributes and refining classes until new attributes cannot be used to further divide classes. OPUS consists of two distinct parts, the *clustering algorithm* and the *attribute generator*; these are described in detail in the following sections.

2.1 The Clustering Algorithm

The OPUS clustering scheme is based on the RUMMAGE clustering algorithm [1]. The goal of the algorithm is to build a hierarchical tree of mutually exclusive classes (clusters) for a given object set. Each object of the set has associated attribute/value pairs for a list of attributes. The hierarchy tree is built in a top down fashion. At each node in the tree, the algorithm selects an attribute which best partitions the object set according to some clustering criteria.

After an attribute has been selected, the object set is divided into mutually exclusive classes whose members have the same value for the chosen attribute. An arc in the hierarchy tree is labeled with the value for the chosen attribute at that node, and any other value for attributes which are common to all members of that class. The procedure is called recursively until the classes cannot be further divided using the given attributes. At this point OPUS once again defines new attributes and applies the clustering algorithm to refine the classes. If the new attributes cannot further divide the classes, OPUS decides that it has determined the final classes and terminates.

2.1.1 The selection of an attribute

Given an object set and a list of attributes, we want to select that attribute which best partitions the set over the remaining attributes. In order to measure the quality of a proposed clustering, OPUS forms a *complex* for each value of an attribute. A complex is the logical implication for the value of an attribute over the remaining attributes [6]. Suppose that we have the object set {K, L, M, N, O} with associated attribute/value pairs for attributes A, B, and, C as follows*:

$$
\begin{array}{lllllllll}
K & = & \{ & A & = & |a|, & B & = & |x|, \quad C = |m,n|\} \\
L & = & \{ & A & = & |a|, & B & = & |y|, \quad C = |m|\} \\
M & = & \{ & A & = & |b|, & B & = & |x|, \quad C = |m|\} \\
N & = & \{ & A & = & |b|, & B & = & |x|, \quad C = |n|\} \\
O & = & \{ & A & = & |a|, & B & = & |y|, \quad C = |n|\}
\end{array}
$$

Given this data, the complexes for attribute A for values $|a|$ and $|b|$ over attributes B and C are:

$$
\begin{array}{ll}
(1)\ |a| \rightarrow & \{(B = |y| \vee |x|) \wedge (C = |m,n| \vee |m| \vee |n|)\} \\
(2)\ |b| \rightarrow & \{(B = |x|) \wedge (C = |m| \vee |n|)\}
\end{array}
$$

That is, if an object has a value of $|b|$ for attribute A, it implies that it has a value of $|x|$ for attribute B, and a value of $|n|$ or $|m|$ for attribute C. OPUS forms these complexes for all values of all attributes. The complexes are used to determine the quality of an attribute. OPUS uses two clustering criteria, the simplicity of the cluster description and the inter cluster difference, which we now discuss.

*In the OPUS system, values of attributes are sets. (See section 2.2)

2.1.2 The clustering criteria

The *simplicity* criterion is used to choose a partitioning attribute which forms a simple description, so that it is easy to characterize and differentiate classes. A second criterion is used to avoid the trivial and arbitrary classification which might occur if the above criterion were used alone [6]; the *inter cluster difference* measures the disjointness of two complexes. The less values overlap among the remaining attributes, the higher this degree of disjointness will be. A good classification has simple class descriptions and a high degree of inter cluster difference to maximize the distance between classes.

The simplicity measure is a normalized value of the number of terms in the complexes of an attribute. A complex consists of a logical product of *selectors*. Each selector is a list of elements from the possible values of an attribute linked by internal disjunction. The *complexity of a selector* is the number of terms of the selector divided by the number of terms the selector could have, i.e., the number of domain elements for the attribute of the selector. The *complexity of an attribute* is the average of the complexity values of all of the selectors of that attribute. The *simplicity of an attribute* is defined to be the negative of the complexity [6]. The complexity of the second selector of complex (2) in our example is $\frac{2}{3}$, because that selector has two elements, ($|n|$ and $|m|$), and there are three possible values ($|n|, |m|$, and $|m,n|$) that attribute C can have. In complex (1), the second selector has a complexity value of $\frac{3}{3} = 1$. The value of complexity for attribute A is the average of 1, 1, $\frac{1}{2}$, and $\frac{2}{3}$ which is $\frac{19}{24}$. Thus the simplicity for attribute A is $-\frac{19}{24}$.

The computation of the inter cluster difference of two complexes is more involved. We define a *selector element* to be an element of a selector — that is, an element of the domain of an attribute. (Values of attributes in the OPUS system are sets.) The *similarity* between two selector elements, e_1 and e_2, is defined to be $sim(e_1, e_2) = \frac{|e_1 \cap e_2|}{|e_1 \cup e_2|}$.** The *maximum similarity* of a reference element e of a selector S_i to selector S_j is $\max\{\text{sim}(e, e_k)\}$, for all $e_k \in S_j$. The value P_{ij} is the average of the maximum similarities of all selector elements of selector S_i to selector S_j. Now, the *degree of similarity* of complex C_k to C_l, denoted Sim_{kl}, is the average over all P_{ij}, where i and j are all the selectors of identical attribute parts. The *degree of difference* of complex C_k to complex C_l, denoted Dif_{kl}, is just $1 - Sim_{kl}$. Finally, the *inter cluster difference degree* of an attribute X is the average of all Dif_{kl} values, $k \neq l$, where k and l are complexes of all values of the attribute X.

Referring again to the example, we calculate the following values for the various computations to calculate the

**$|e_1 \cap e_2|$ denotes the cardinality of the intersection of set e_1 and set e_2, while $|e_1 \cup e_2|$ denotes the cardinality of the union of the two sets.

inter cluster difference degree for attribute A:

For the selectors of attribute B, we compute values

$$P_{12} = \frac{\max\{sim(|y|,|x|)\} + \max\{sim(|x|,|x|)\}}{2} = \frac{\max\{0\} + \max\{1\}}{2} = \frac{1}{2}$$

$$P_{21} = max\{sim(|x|,|y|), sim(|x|,|x|)\} = max\{0,1\} = 1$$

For the selectors of attribute C, we compute in a similar manner

$$P_{12} = \frac{\max\{\frac{1}{2},\frac{1}{2}\} + \max\{0,1\} + \max\{1,0\}}{3} = \frac{5}{6}$$

$$P_{21} = \frac{\max\{\frac{1}{2},0,1\} + \max\{\frac{1}{2},1,0\}}{2} = 1$$

Thus we have a degree of *degree of similarity* of complex (1) to complex (2) of attribute A of $(\frac{1}{2} + \frac{5}{6})/2 = \frac{2}{3}$ and a *degree of similarity* of complex (2) to complex (1) of $\frac{1+1}{2} = 1$. Therefore the *degree of differences* are $\frac{1}{3}$ and 0 respectively. The *inter cluster difference degree* for attribute A is $(\frac{1}{3} + 0)/2 = \frac{1}{6}$.

This computation of the inter cluster difference for an attribute makes use of the fact that, in the OPUS system, values of attributes are partially ordered. That is, value $|a,b|$ is further from value $|b,c,d|$ than it is from value $|a,b,c|$, and therefore $sim(|a,b|,|b,c,d|)$ is less than $sim(|a,b|,|a,b,c|)$. Class descriptions should be as distinct as possible to ensure classes with different properties. Maximizing the inter cluster difference will promote such classes.

The idea of an asymmetric similarity measure may seem counterintuitive at first. However, Tversky [8] supports an asymmetric similarity measure, and he provides evidence that humans "tend to select the more salient stimulus ... as a referent, and the less salient stimulus ... as a subject." Referring once again to the complexes in the example, any object satisfying the conditions of complex (2) also satisfies the conditions of complex (1), but not vice versa. Therefore Sim_{21} has a higher value than Sim_{12}.

OPUS maximizes a trade off between the inter cluster difference and the simplicity of a class description. At each level in the expanding hierarchy tree, a quality value for each attribute is computed. This value is the sum of

$$u * \text{simplicity} + v * \text{inter cluster difference}$$

for some user specified coefficients u and v. The user can thus weigh the importance of these two criteria. OPUS maximizes the quality value of the attributes selected at each node in the expanding tree.

2.2 Generating Attributes

New attributes have to be defined when current attributes are not sufficient to distinguish between members of the same class. New attributes are chunks composed of relations and features. For this purpose, we define a *complex relation* `r f(X,Y,Z)` to be the composition of a relation `r(X,Y)` and a feature `f(Y,Z)`. For example in the *food chain* domain animals could be described by the feature *size* and the relationship *eat*. Thus the relation `eat(X,Y)` and the feature `size(Y,Z)` are composed to form the complex relation `eat size(X,Y,Z)`, describing that X eats Y and Y is of size Z. Note that the first and second argument of a complex relation are members of the object set, while the third is a value of the feature. Complex relations will be used as attributes.

The value of an attribute is defined as follows. Given a complex relation `r_f(X,Y,Z)`, the value of the attribute `r f` for the object X is the set $\{Z_i \mid \exists Y \ni$ `r_f(X,Y,`Z_i`)`$\}$. That is, the set of all Z's, such that `r_f(X,Y,Z)` is satisfied for some Y. For example, the value of `eat_size` for snakes in the food chain domain is `[small, medium]`, because `eat_size(snakes, Y, small)` is satisfied for Y bound to mice and insects, and `eat size(snakes, Y, medium)` is satisfied for Y bound to snakes. Thus, the attribute `eat size` has a value of `[small, medium]` for snakes, because snakes eat small and medium sized animals.

The system is supplied with a small set of binary relations such as *eat* or *parent*. These primitive relations involve only two objects, and there is a direct "link" between the two objects. In order to define more involved attributes, relations consisting of several primitive relations are formed. We define a *level n relation* as a relation using n primitive relations between two objects. A primitive relation is a relation supplied to the system or the inverse of that relation. The relation `eaten(X,Y)` describes the level one relation *eaten*, meaning X is being eaten by Y, while `eat eat(X,Z)` describes the level two relationship of X eats some Y and Y eats Z. Relations are defined in increasing levels of order, starting at level one. Now, a *level n attribute* is defined from a complex relation composed of a level n relation and an existing feature. Each time new attributes have to be defined the current level k is increased and level k + 1 relations are defined. These level k + 1 relations are composed with features to define complex relations and thus level k + 1 attributes. Relations are not directly used in the clustering process, but rather used to define attributes. Only attributes are used to cluster objects. Thus, objects are first classified based upon their features, then based upon attributes with increasing complexity. If at any time new relations cannot define attributes which refine classes, the system terminates having reached a final classification.

At each level k, new level k relations are defined. A level k − 1 relation is composed with a level one relation to form a level k relation. All inverses of relations are defined. To limit the combinatorial explosion of the number of possible relations which can be defined at each level, only a limited number of the k − 1 relations are considered to define new relations. Only the relations which defined attributes used to refine classes at level k − 1 are used at the next level to define new relations.

3. Two Examples

OPUS has applications in any domain where objects are described by a set of features and a set of binary relations. Two examples of such domains are presented in the following sections: the food chain domain and the genetics domain.

3.1 The Food Chain Domain

In the food chain domain, we characterize animals using two features, *size* and *locomotion*, and relation, *eat*. For example, we describe songbirds using the following facts: `size( songbirds, medium)`, `locomotion(songbirds, fly)`, `eat(songbirds, worms)`, `eat(songbirds, insects)`, and `eat(hawk, songbirds)`. All fourteen objects are characterized by the same two features. Fifty-one relational facts are asserted to describe the relationship *eat* over the objects set.

At first, OPUS uses features as attributes to classify the objects. `size` has the same simplicity value as `locomotion`, but a higher inter-cluster difference value. Therefore `size` is chosen as the first attribute to divide the object set in the hierarchy tree. For example, a class of medium sized objects is created with the following members: hawks, owls, songbirds, and snakes. After the system has used `locomotion` to refine classes, there are no attributes left and new attributes have to be defined.

In response to that OPUS defines all possible level one relations. The following complex relations and attributes are formed: `eat size`, `eat locomotion`, `eaten size`, `eaten locomotion`. The first two describe the size and locomotion of animals eaten by an object, the latter two describe the size and locomotion of the animals that eat that object. These four attributes are used to divide the existing classes. For example, the class of medium-sized flying objects is refined using the attribute `eat size`. Hawks and owls eat medium and small animals, while songbirds only eat small animals.

After the current attributes have been used to refine the classes, there are only two classes with more than one object left, the class of frogs and toads, and the class of hawks and owls. The level two relations `eat eat`, `eat eaten`, `eaten eat` and their inverses are formed, and concatenated with the features to define level two attributes. Frogs and toads have the same values for these new attributes, therefore that class is not refined. However hawks and owls have different values for the attribute `eat_eaten size`, namely `[large, medium]` and `[large, medium, small]`. That is, hawks eat animals which are eaten by large and medium sized animals, while owls eat animals which are eaten by large, medium, and small animals. Thus, the attribute `eat eaten size` is used to divide that class. The next level relations cannot define attributes which refine the class of frogs and toads, so the system terminates. The resulting hierarchy tree is shown in Figure 1.

3.2 The Genetics Domain

Let us now consider an example from the field of genetics. The clustering problem in genetics consists of classifying objects based not only on their observable features, but also on features of their descendants and their ancestors. Gregor Mendel, the founding father of genetics, observed that when a yellow garden pea was crossed with a green garden pea the resulting offspring pea was yellow [4]. When he self-fertilized that pea, it produced both yellow and green offspring. After he continued to self-fertilize peas, he discovered that some of the yellow peas had yellow and green offspring while other yellow peas only produced yellow offspring. Green peas consistently had green offspring. Mendel thus hypothesized the class of *purebreds*, peas which produce offspring with exactly the same features as the parent, and the class of *hybrids*, peas which produce some offspring with the same features and other offspring with features different from their parent.

When OPUS is provided with information about the *color* of each pea and the *offspring* each pea produces, it defines the classes of hybrids and purebreds. At first, the feature *color* is used as an attribute to distinguish yellow and green peas. Next, the attributes `offspring_color` and `parent color` are defined. For the class of yellow peas, the inter-cluster difference and the simplicity value for these attributes are equal. In the running system `parent color` was picked to refine the class of yellow peas. At this point all peas are correctly identified as either a yellow or green purebred or a (yellow) hybrid. Furthermore, the characterization of these classes corresponds with Mendel's characterization. For example, the class of green purebreds only has green offspring, while the class of hybrids contains only yellow peas which have both yellow and green offspring. OPUS continues to refine the classes -- distinguishing, for example, between purebreds with hybrids as parents and purebreds with purebreds as parents.

Mendel continued his experiments, crossing peas with two different traits, *color* and *shape*. He observed nine different classes, all having different dominant and recessive traits. We supplied the OPUS system with the color and shape of each pea and asserted the relations over the object set. Again OPUS correctly defined and characterized as intermediate classes all nine classes which Mendel identified as the various hybrids and purebreds. For example, OPUS defines two different classes of round green peas; one class has members which only have round green peas as offspring, while the other class has members which produce round green and wrinkled green offspring.

4. Summary and Further Research

In this paper, we presented a conceptual clustering system which uses relations over the object set to define a hierarchy of classes. Using the relational information, this system is able to find classifications not possible with conventional methods of conceptual clustering. We presented an example from the domain of genetics where the system

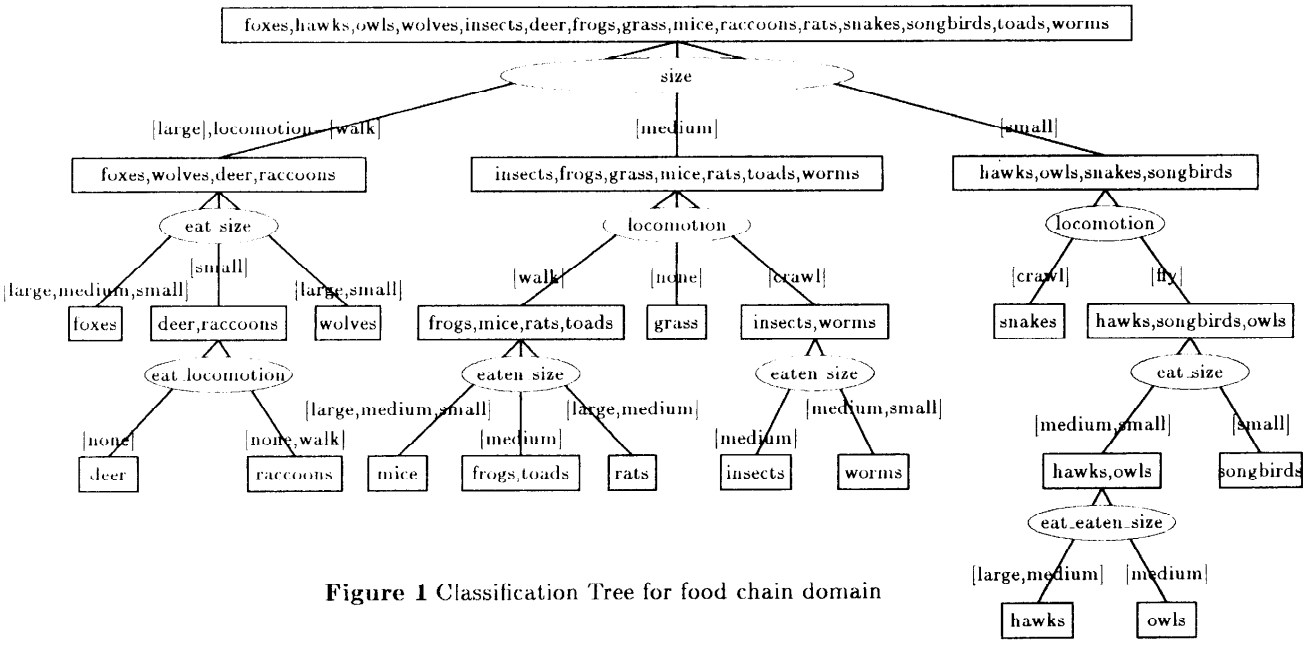

Figure 1 Classification Tree for food chain domain

is able to form the classes of hybrids and purebreds. Furthermore, we introduced a method to define new attributes used in the classification process.

This work can be extended in two ways. It is unrealistic to assume that all the information describing objects is available initially. An incremental version of OPUS would build the hierarchy tree using partial information, predicting missing properties of objects as well as missing objects. As more data becomes available, predictions can either be confirmed, in which case the belief in other similar predictions is reinforced, or they can be disconfirmed, in which case a revision of classes occurs.

The present version of OPUS can handle only binary relations. An extension of the system working with n-ary relations would greatly enhance its power. For example, in the domain of chemistry, some compounds are classified as acids, alkalis and salts depending on (among other properties) their reactive behavior. For example, alkalis react with acids to form salts. Using ternary relations, these classes could be formed in a way similar to GLAUBER [3], yet in a more efficient manner. At the moment, we are actively engaged in working in these directions.

Acknowledgements

I would like to thank Pat Langley, Don Rose and Randy Jones for their help on this work, as well as the numerous people from the machine learning group at UCI who gave me valuable comments on drafts of this paper. This work was supported in part by Contract N00014 84 K 0345 from the Information Sciences Division, Office of Naval Research.

References

[1] Fisher, D. *A hierarchical conceptual clustering algorithm.* Tech. Report 85 21, Dept. of Information and Computer Science, University of California, Irvine, CA, 1984.

[2] Fisher, D. and Langley, P. Approaches to conceptual clustering. Proceedings of the Ninth International Joint Conference on Artificial Intelligence, 691–697, 1985.

[3] Langley, P., Zytkow, J. M., Simon, H. A., and Bradshaw, G. L. The search for regularity: Four aspects of scientific discovery. In *Machine Learning: An Artificial Intelligence Approach, Vol. II*, R. S. Michalski, J. G. Carbonell, and T. M. Mitchell, Eds., Morgan Kaufman Publishers, Los Altos, CA, 1986, 425 469.

[4] Iltis, H. *The Life of Mendel*, Hafner Pub., New York, 1966.

[5] Michalski, R. S. Knowledge acquisition through conceptual clustering: A theoretical framework and algorithm for partitioning data into conjunctive concepts. *International Journal of Policy Analysis and Information Systems, 4* (1980), 219 243.

[6] Michalski, R. S. and Stepp, R. E. Learning from observation: Conceptual clustering. In *Machine Learning: An Artificial Intelligence Approach*, R. S. Michalski, J. G. Carbonell, and T. M. Mitchell, Eds., Tioga Press, Palo Alto, CA, 1983, 331 363.

[7] Stepp, R. E. and Michalski, R. S. Conceptual clustering: Inventing goal oriented classification of structured objects. In *Machine Learning: An Artificial Intelligence Approach, Vol. II*, R. S. Michalski, J. G. Carbonell, and T. M. Mitchell, Eds., Morgan Kaufman Publishers, Los Altos, CA, 1986, 471 498.

[8] Tversky, A. Features of Similarity. *Psychological Review 8*(4), 1977, 327 352.