A Survey of Clustering Methods

John H. Gennari

Department of Information and Computer Science
University of California, Irvine, CA 92717

Oct. 30, 1989

Technical Report 89-38

I would like to thank Pat Langley, Doug Fisher, Wayne Iba and Kevin Thompson for their help with this paper. This research was supported by Contract MDA 903-85-C-0324 from the Army Research Institute.

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER: Technical Report No. 6	2. GOVT ACCESSION NO.
3. RECIPIENT'S CATALOG NUMBER	
4. TITLE *(and Subtitle)*: A Survey of Clustering Methods	5. TYPE OF REPORT & PERIOD COVERED: Interim Report 7/88–6/89
	6. PERFORMING ORG. REPORT NUMBER: UCI–ICS Technical Report 89–38
7. AUTHOR*(s)*: John Gennari	8. CONTRACT OR GRANT NUMBER*(s)*: MDA 903-85-C-0324
9. PERFORMING ORGANIZATION NAME AND ADDRESS: Department of Information & Computer Science, University of California, Irvine, CA 92717	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS: Army Research Institute, 5001 Eisenhower Avenue, Alexandria, Virginia 22333	12. REPORT DATE: October 30, 1989
	13. NUMBER OF PAGES: 24
14. MONITORING AGENCY NAME & ADDRESS *(if different from Controlling Office)*	15. SECURITY CLASS. *(of this report)*: Unclassified
	15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT *(of this Report)*

Approved for public release; distribution unlimited

17. DISTRIBUTION STATEMENT *(of the abstract entered in Block 20, if different from Report)*

18. SUPPLEMENTARY NOTES

19. KEY WORDS *(Continue on reverse side if necessary and identify by block number)*

machine learning
concept formation
numerical taxonomy
cluster analysis

20. ABSTRACT *(Continue on reverse side if necessary and identify by block number)*

In this paper, I describe a large variety of clustering methods within a single framework. This paper unifies work across different fields, from biology (*numerical taxonomy*) to machine learning (*concept formation*). An important objective for this paper is to show that one can benefit by a knowledge of research across different disciplines. After describing the task from a set of different viewpoints or paradigms, I begin by describing the similarity measures or evaluation functions that form the basis of any clustering technique. Next, I describe a number of different algorithms that use these measures, and I close with a brief discussion of ways to evaluate different approaches to clustering.

1. Introduction

Researchers in a variety of fields have studied the basic task of *clustering* instances into classes. Although specific instantiations of this task differ from field to field, a general statement of the problem is as follows:

- Given: a set of instances, each described by some number of attribute-value pairs;
- Find: a set of classes that group those instances.

For example, suppose one were suddenly placed in a jungle on an unknown planet. As a learning agent, one would immediately begin creating concepts to classify and organize the instances observed (plants, animals, or any perceived objects). This example emphasizes the *unsupervised* nature of the problem – the learner is trying to impose structure on the environment without any feedback.

A fundamental difficulty for the clustering task is that it requires some means of evaluating a set of potential classes. For example, an *evaluation function* measures the quality of a set of classes with respect to the data. Creating an evaluation function is closely related to defining some sense of *similarity* among instances. In turn, the attributes (and attribute types) that describe instances affect the similarity measure. In addition, there are a number of different *algorithms* that create classes from instances. Although some algorithms require particular evaluation functions, often the researcher can try a set of different functions with a single algorithm, and vice versa.

In this paper, I will organize a large number of clustering techniques under one framework as a space of possible methods. My hope is that this will offer more insights than simply listing different methods from different fields, or trying to define a 'best ' or 'optimal' technique. Although the framework I develop in this survey may not cover every clustering method, I believe it brings to light some interesting insights, and that it describes a large range of possible methods.

To some degree, researchers in machine learning, or artificial intelligence in general, have been unaware of related work outside of their own field. There is a large body of research in statistics and biology, usually known as *cluster analysis*, that is applicable to work in machine learning, if one allows for the different biases of these disciplines. Although a few AI researchers have acknowledged this area of work (Michalski & Stepp, 1983a; Stepp, 1987; Fisher & Langley, 1986), there has been no comprehensive survey of cluster analysis for AI. In particular, cluster analysis is very similar to the study of *concept formation* in machine learning. One goal of this paper is to emphasize this similarity and to show how researchers in machine learning can benefit from a knowledge of cluster analysis.

I begin this survey by presenting overviews of the clustering problem as seen from several different perspectives, beginning with a machine learning view. In the third section, I describe the difficulty and importance of choosing a similarity measure or an evaluation function; this section also includes some of the most common and useful measures. I follow this with a description of

algorithms that use these measures, and I conclude with a discussion of the difficulty of validating or evaluating a clustering technique.

2. Paradigms and Biases

Rather than attempting a more precise definition of the clustering task, I will instead describe the problem from four different paradigms: machine learning, biology, statistics, and decision theory. In this way, the goals and biases of researchers from different fields will be made explicit.

My own biases for clustering come from the machine learning paradigm and the terminology used in this survey is from that body of literature. Both to give an idea of the diversity of the terminology, and for readers from other paradigms, the following is a brief 'translation' list of terms (the terms used in this paper are presented last and italicized):

- an object, organizational taxonomic unit (OTU), event, or case is an *instance*,
- the characters, features, or variables that describe an instance are *attributes*,
- the distance metrics, association measures, or similarity coefficients that compare instances are *similarity measures*, and
- the closely related optimization criterion is an *evaluation function*.

Because researchers have approached the clustering problem from such different directions, not only is the problem described with different terminology, but the task itself varies slightly. This is hardly surprising – biology has very different goals than machine learning. Since I wish to compare other efforts to those in machine learning, I begin with this paradigm.

2.1 Clustering in Machine Learning

From the perspective of machine learning, clustering is viewed as a problem of concept formation. As with most of artificial intelligence, this field is biased by a computational analogy to human processes. Therefore, the process of clustering and the concepts produced by that process have implications for (human) learning, and for (human) knowledge organization and representation.[1] Although machine learning does not always make explicit this connection, there is usually at least a weak analogy to the human clustering system.

As an example of an application for concept formation, consider an exploring robot that perceives a sequence of different balls. Even if the robot is equipped with a perceptual system that reduces each instance to a set of attribute-value pairs, it still must create and organize a useful set of concepts about these balls. For example, after observing a few baseballs, it should create a

[1] These biases are from the closely related field of cognitive psychology, where the explicit goal is to study human cognitive processes. Concept formation has been studied by cognitive psychologists such as Smith and Medin (1981), Mervis and Rosch (1981), Barsalou (1987), Corter, Gluck and Bower (1988), and Anderson (1988).

concept for these and be able to recognize a new baseball as a member of that class, and not as an instance of some other class (volleyballs, tennis balls, etc.).

One distinguishing feature of concept formation is that the classes learned should be *intensional*, rather than extensional. For example, the baseball class should be a "conceptual description" of the baseballs seen, rather than simply a list of all member instances (Michalski & Stepp, 1983b). This emphasis on intensional concept definitions means that evaluation functions that compare classes are more appropriate for concept formation than similarity measures that compare instances.

A second aspect of concept formation is that the classes learned are usually arranged in a concept hierarchy. That is, the learned concepts are organized into a hierarchy with more general, inclusive concepts toward the top, and more specific, exclusive concepts toward the bottom. This reflects the hierarchical nature of knowledge in typical machine learning domains. For example, soccer balls and volleyballs are more similar to each other than to baseballs or to lacrosse balls. A natural hierarchy for these four types of balls would be to put soccer balls and volleyballs together into a more general "soft, large" class, and lacrosse balls and baseballs into a "hard, small" class.

A third characterizing feature of concept formation is that learning occurs *incrementally*. As the robot observes each successive ball, it should add to its knowledge immediately; the concepts learned are updated by each new experience without reprocessing previous instances. In contrast, a nonincremental system must receive the entire set of instances before producing a set of classes. Such a system is incompatible with the goals of concept formation because one may not know the complete 'set' of instances, and one may need to use the learned concepts at any point in time. For example, the robot should be able to use its knowledge at any point during learning, and it should continue learning, no matter how many balls it encounters. These problems are perhaps more obvious for human learners, who observe a never-ending sequence of instances.

Researchers in machine learning are usually interested in robust algorithms, rather than special-purpose clustering methods. A researcher will therefore apply his method to a wide variety of domains, often including large, noisy data sets. Finding a single clustering method that works over a large number of varied domains is motivated by the psychological evidence that there is at least one such algorithm: the human clustering system.

Finally, if a system learns, one should be able to measure its improvement on some *performance task*. This is a task used to test (and quantify) the ability of the system before and after learning. With this type of numeric measure, the success of a concept formation system can be evaluated over a number of different domains, or a set of different systems can be compared on given data set.

As I have described the problem, clustering is *unsupervised*. However, there is also a large amount of work in machine learning on supervised concept formation, usually known as "learning from examples". Although this is a related task, the differences between these two problems are very important. Supervised concept formation learns to determine which of a known set of classes

an instance belongs to, while unsupervised learning imposes a structure of concepts (ones that are not known *a priori*) on the set of instances.

2.2 Clustering in Biology

Historically, the first computational approaches to clustering arose in the field of biology. In particular, 'numerical taxonomy' grew out of the following problem: given a set of species or organisms, find a taxonomic hierarchy that organizes them into species, genera, phyla, and classes. The principal purpose of this hierarchy is to suggest evolutionary relationships among individuals.

As an example, suppose a biologist discovers a new set of worms. After describing each worm by some set of attributes (or "characters"), the biologist inputs this set of instances to a clustering system. The resulting hierarchy should define classes of similar worms, as well as showing how these classes are related to each other. This information can lead to predictions about how related worms may behave, and also to theories about how worm attributes have evolved over time.

The emphasis in this discipline is to find a practical method, instead of worrying about the theoretical implications of a particular technique. For instance, biologists are not very concerned with validating or measuring the performance of a given technique. Instead, their measure of success is subjective: if a method produces a useful taxonomy (one that leads to new insights, or has interesting evolutionary implications), then it is a good one. This bias has led other researchers to state that "the [biologist's] view of clustering is considered a radically empirical approach" (Aldenderfer & Blashfield, 1984, p. 21).

Since these methods are explicitly looking for a taxonomy, the classes found should be disjoint and organized into a hierarchy. A flat list of classes or a set of overlapping classes is not as useful. However, in contrast to machine learning methods, the classes created by numerical taxonomy techniques are usually extensional, and similarity measures, rather than evaluation functions, are used as the basis for clustering.

Although a robust, general-purpose algorithm is appreciated in any field, these are not crucial features for numerical taxonomy. In this paradigm, it is reasonable to use different algorithms for different data sets. Also, for any given clustering problem, the researcher is only interested in a finite (usually small) set of instances. Therefore, the ability of a clustering method to incrementally process new instances is not very important.

This survey emphasizes biological clustering methods; however, many of these same biases can be seen when clustering techniques are used in other sciences. In particular, the same emphasis on practical methods, similarity measures, and subjective validation appears in clustering for ecology and psychology.

2.3 Clustering in Statistics

The statistician has a much more formal view of the clustering problem. In this approach, researchers are interested in a careful definition of clustering and in exploring theoretical implications of clustering methods. Although this paradigm has had some success, the heuristic nature of clustering can be an obstacle to the type of rigorous analysis preferred by statisticians. Likewise, objectively evaluating the result of a clustering technique (validation) has proven difficult.

For the statistician, one place to begin is a comparison of cluster analysis with other, well-established statistical methods such as factor analysis, analysis of variance, and discriminant analysis. For example, statisticians point out that choosing a set of attributes that describe instances is the general problem addressed by factor analysis. However, it appears that performing factor analysis as a pre-processing step has a detrimental effect on clustering.[2] Similarly, the standard multivariant practice of normalizing variables can cause problems: normalization can obscure differences that may be crucial for clustering (Everitt, 1980).

Statisticians have also analyzed and compared the algorithms and evaluation functions of clustering methods themselves. Although this effort has shown that some methods and similarity measures are redundant (Anderberg, 1973), it has not been able to establish any single clustering method as best. The difficulty is that, unlike most statistical methods, clustering is *heuristic*. Since the algorithms use 'rules of thumb' that are not guaranteed to produce correct solutions, they are difficult to analyze and compare.

Although one cannot measure the subjective goal of finding an 'interesting' set of classes, statisticians are interested in quantitatively evaluating aspects of a solution. Unlike the biologist's perspective, the statistician's 'solution' need not be a hierarchy of classes: for some domains, a flat list of classes is more appropriate; for others, overlapping or probabilistic classes may be preferable.

2.4 Clustering as Decision Theory

An abstraction of the clustering problem has been studied by a few researchers in the field of decision theory (Jaynes, 1986; Cheeseman, Kelly, Self, Stutz, Taylor, & Freeman, 1988). In this view, the goal is to correctly predict the probabilities that a new instance x is a member of a class ω_i: $P(\omega_i|x)$. This expression can be re-written using Bayes' theorem:

$$P(\omega_i|x) = \frac{P(x|\omega_i)P(\omega_i)}{P(x)} \quad .$$

The probability of each class, $P(\omega_i)$, is usually known – it can be computed as the number of members of ω_i divided by the total number of instances. Additionally, since $P(x)$ is the probability of x independent of class, it can be ignored – when comparing two different classes, ω_1 and ω_2, the

[2] There is considerable debate on this issue. See Everitt (1979) or Aldenderfer and Blashfield (1984) for more discussion.

denominator is the same and can be dropped. In fact, the only unknown term on the righthand side is the class conditional probability density function, $P(x|\omega_i)$. This is the probability of x given ω_i, or the value of x that would be predicted by class ω_i. These functions must be estimated; for example, one can assume a normal distribution, and search for a good estimate of the parameters μ and σ that characterize this distribution.

Using this foundation for a clustering method guarantees that the method will maximize the probability of correctly characterizing the data. Hence, if a system is faithful to the theory, it is "provably" optimal, and there is little need for empirical validation. For this reason, researchers from this paradigm place less emphasis on the algorithms used to implement a Bayesian classifier. However, it is usually difficult to create a system that is faithful to the theory, and a number of assumptions must be made along the way. Most solutions based on decision theory encounter the usual problems of heuristic search, but at this lower, algorithmic level. The principal advantage of the Bayesian framework is that by providing an underlying theory of decisions, the researcher can easily make explicit the theoretical implications and necessary assumptions for any solution to the clustering task.

3. Similarity Measures and Evaluation Functions

Although it is not always made explicit, any clustering technique produces a set of classes in which the members of a given class are similar to each other in some way. Using the 'search' terminology, one is looking for classes in which the similarity between instances within a given class is greater than that between instances from different classes. From this perspective, a clustering technique can be characterized by how it defines 'similarity'.

For some techniques this basis is made explicit: there is a *similarity measure* or a *distance metric* that quantitatively measures the distance between two instances. For others, the goal is to maximize some *evaluation function* or criterion. Such a function measures the 'goodness' of a set of classes; usually this is based on the similarity among classes, rather than between two instances.

Clustering depends upon a similarity measure; in turn, the scores of a similarity measure depend upon the attributes used to describe an instance. I begin this section by describing some issues and problems pertaining to attributes. Next, I present a set of similarity measures followed by evaluation functions. These are described independently from the clustering algorithms that use them, so that they can be compared directly. This also makes clear that there is usually more than one choice of measure or evaluation function available to the researcher.

3.1 Attributes: Choices and Representations

For the biologist or social scientist who is approaching clustering techniques as a tool, there are a potentially infinite number of attributes available to describe an instance. Rather than blindly using as much information as can be found, the scientist can choose only those attributes that are 'relevant' to the task. As mentioned earlier, factor analysis is a well-defined statistical method

that is sometimes used to create a smaller number of more 'appropriate' attributes from the pool of available attributes.

Unfortunately, this is somewhat circular logic. In spite of its widespread use, Everitt (1979) argues against using factor analysis or any method which eliminates attributes before clustering. The purpose of clustering is to discover an unknown set of classes. As it searches for these, it will establish which attributes are 'relevant', but to decide this beforehand would slant the clustering process. Factor analysis can have the detrimental effect of hiding those attributes that may be crucial to finding a hierarchy of classes. Factor analysis assumes a single, known class; hence, Everitt suggests that it may be used after clustering, but never beforehand. Researchers also sometimes place weights on the attributes prior to clustering. This has the same effect as using factor analysis, and is vulnerable to the same criticism.

Any method for measuring similarity depends to some degree on the representation used for the attributes that describe an instance. Anderberg (1973) points out that there are two ways of characterizing attributes: the measurement scale used for the attribute, and the number of possible values an attribute may take on. For this paper, I will describe four main types of attributes: continuous, ordinal, symbolic, and binary.[3]

Continuous attributes have an infinite range and are measured along a continuous scale. Examples of this type of attribute are real-valued measurements of height, weight, and temperature. An *ordinal* attribute has a finite range with an ordering on the possible attribute values. Examples might be number-of-fins, or any continuous attribute that has been rounded, such as age to the nearest year. A *symbolic* attribute also has a finite range, but there is no order to its values. Examples of this may be shape, place of birth, or type of sailboat. Finally, a *binary* attribute has only two possible values. Often, these are presence-or-absence attributes such as has-backbone, or is-hungry.

The similarity measure or evaluation function employed will be dependent upon the attribute types used to describe instances. In fact, instances may be described by a combination of different types of attributes. Unfortunately, one of the many unsolved problems in cluster analysis (from a statistician's point of view) is that there is no 'good' way to combine different attribute types. That is, despite some attempts, there are no theoretically sound similarity measures that can be applied to different attribute types, especially if binary and continuous attributes are combined. For this reason, the measures described below are organized according to whether they are appropriate for continuous, ordinal, symbolic, or binary attributes.

3.2 Measures for Continuous or Ordinal Attributes

I shall begin by considering similarity measures for continuous attributes: measures comparing two instances that are described by a set of continuous or ordinal attributes. Let i and j be the instances, each described by K attributes, e.g., $i = \{x_1, x_2, .., x_K\}$. One of the most obvious

[3] Note that my terminology is different from Anderberg's, reflecting my machine learning bias.

similarity measures between these two points is to use a *distance metric*. Each attribute defines a dimension, and each instance can be plotted in this K-dimensional space. The Euclidean distance between two points is

$$D_{ij} = \left[\sum_{k=1}^{K}(x_{ik} - x_{jk})^2\right]^{1/2} .$$

A simpler, and related metric, is the city-block distance:

$$D_{ij} = \sum_{k=1}^{K}|x_{ik} - x_{jk}| .$$

Euclidean distance is probably the most well-used measure for similarity; it is also used as the basis for some evaluation functions. Although a city-block distance may seem less intuitive, it is computationally much cheaper, and may be appropriate when ordinal attributes are used.

Both of these measures are sensitive to linear transformations of the input data. For example, if some attributes are transformed from, say, inches to miles, then these re-scaled data will have completely different similarity measures.[4] As Duda and Hart (1973) point out, this may or may not be a problem, depending on whether such transformations are natural to one's domain.

The use of Pearson's correlation measure is one way to achieve invariance with respect to linear transformations. The original intent of this measure is to correlate pairs of attributes for factor analysis. In order to correlate instances, researchers simply reversed the equation syntactically. That is, imagine the data set as a $K \times N$ matrix of values (if there are K attributes and N instances); the usual use of correlation is to measure similarity among the columns (the attributes), so the reversed equation should measure similarity among the rows (the instances). This reversed correlation is referred to as *Q-mode factor analysis* (Anderberg, 1973, p. 113). It states that the correlation (distance) between two instances is

$$D_{ij} = \frac{\sum_{k}^{K}(x_{ik} - \bar{\bar{x}}_i)(x_{jk} - \bar{\bar{x}}_j)}{\left[\sum_{k}^{K}(x_{ik} - \bar{\bar{x}}_i)^2 \sum_{k}^{K}(x_{jk} - \bar{\bar{x}}_j)^2\right]^{1/2}} ,$$

where $\bar{\bar{x}}_i = 1/K \sum_k^K x_{ik}$, the average attribute value for a given instance, i.

This approach has been used with some success among researchers in psychology (See Aldenderfer & Blashfield, 1973, pp. 22-23). However, this measure is largely discredited (especially in

[4] Note that this 'attribute scaling' is related to attribute weighting, described earlier. Attribute weighting is a controversial practice only if the similarity measure used is sensitive to linear transformations of the data.

other fields) because there is no justification for the syntactic reversal. The meaning of the equation is lost: for example, since $\bar{\bar{x}}$ is an average across different attributes, it may be averaging 'apples and oranges', and may not have the semantics expected for that term.

3.3 Measures for Binary or Symbolic Attributes

Neither correlation nor Euclidean distance can be applied to an attribute with binary or symbolic values. One characteristic of such an attribute is that, given two values, the expression $x_i - x_j$ does not have any meaning. A similarity measure for symbolic attributes is faced with a simple comparison: either two values are the same, or they are different. Over a set of attributes, the simplest way of comparing two instances is to find the percentage of matching attributes:

$$S_{ij} = \frac{number\ of\ matching\ attributes}{total\ number\ of\ attributes} .$$

In artificial intelligence, this is a 'partial match': a score of one indicates that all attributes match, while a zero says that no attributes match.

TABLE 1. A 2 x 2 association table

j \ i	1	0
1	a	b
0	c	d

Since binary attributes are very common, researchers have usually treated this case separately. If one looks at two instances, i and j, there are four possible relationships for each binary attribute; these are shown in a 2×2 association table, as Table 1. If these are totaled over all attributes, a and d represent the number of matched attributes, while c and b are mismatches. Therefore, the simple matching measure described above can be expressed as

$$S_{ij} = \frac{a+d}{K} ,$$

where $K = a + b + c + d$, or the number of attributes. A distance measure can be defined as $b + c$: the more mis-matches, the greater the distance between the instances. This is known as the *Hamming distance* (Hamming, 1980). The distinction between a, the number of positive matches, and d, the number of negative matches, is made because binary attributes can express the presence or absence of some observable feature. This is often the case in biology; in such a domain, it may be more appropriate to use a measure that does not count 'missing' matches:

$$S_{ij} = \frac{a}{a+b+c} \quad .$$

This similarity measure is known as *Jaccard's matching coefficient* (Romesburg, 1984, p. 143).

Jaccard's coefficient and the simple matching measure are the most commonly used similarity measures for binary attributes. However, there are a large number of related similarity measures that can be defined in terms of Table 1. Most of these other measures are closely related to Jaccard's coefficient or the simple matching measure; see Romesburg (1984) for a description of 12 different measures for binary attributes. As one example, similarity can be described as the probability that instances i and j match on a variable minus the probability that they mismatch:

$$S_{ij} = \frac{(a+d)-(c+b)}{K} \quad .$$

Although these measures are defined expressly for binary attributes, they can be easily adapted for symbolic attributes. This can be accomplished either by converting a symbolic attribute into a set of binary attributes, or by converting the measure itself. Converting the attribute can be accomplished by using the n possible values to create n presence-or-absence binary attributes. (Of course, the use of this type of attribute suggests Jaccard's coefficient as an appropriate measure.) Alternatively, a measure defined for binary attributes can be used with symbolic attributes, if one lets $a+d$ be the number of matches, $b+c$ be the number of mismatches, and d be the number of times the corresponding attribute is missing, or not applicable to the given pair of instances.[5]

Although it is not difficult to convert from symbolic to binary attributes, the only measure that can compare instances with both symbolic and continuous attributes is a simple combination of existing measures. Gower's coefficient is a sum over all attributes of one of three measures: if the attribute is binary, Jaccard's coefficient is used; if the attribute is symbolic, the simple matching measure is used; and if the attribute is continuous (or ordinal) a normalized city-block distance metric is used. Despite its apparent generality, Romesburg (1984) notes that the measure has rarely been used in practice. This may be because the mathematical properties of Gower's coefficient are unknown; of course, this criticism can be made against other, well-used measures.

[5] Anderberg (1973) presents this conversion, as well as a large number of other ways to convert measures and attributes from one type to another.

3.4 Evaluation Functions

Evaluation functions are distinct from similarity measures in that they compare sets of classes, rather than a pair of instances. This difference can be trivial; some evaluation functions are simple extensions of similarity measures. However, the emphasis on classes rather than instances is important to machine learning. From this perspective, as the system learns, the evaluation function controls the search for useful classes by evaluating the quality of a set of concepts with respect to the data. In contrast to similarity measures, evaluation functions often imply a particular type of concept definition. As I present different evaluation functions, I will point out their relations to various similarity measures, as well as their implications for concept representations.

3.4.1 Average Distance

The most straightforward way to evaluate classes is to evaluate all of the members by using a similarity function. For example, one common method is to sum the distance from each object to the class mean; the lower this average distance, the better the class. Although this could be defined with any metric, the most common function is based on Euclidean distance; over all classes, the evaluation function is:

$$trace(\mathbf{W}) \;=\; \sum_{j}^{J}\sum_{k}^{K}\frac{1}{N_j}\sum_{i}^{N_j}(x_i - \bar{x}_{jk})^2 \quad ,$$

where N_j is the number of members in class j, and $\bar{x}_{jk}$ is the average over all members of class j for attribute k. This expression is known as $trace(\mathbf{W})$, because it is the sum along the diagonal of the within-group covariance matrix. (Also known as a *scatter matrix*, this is the matrix of all possible covariances among K attributes.) Note that for a single class and attribute, this expression corresponds to the variance for that attribute. In fact, this function suggests a concept representation consisting of a list of means and variances for each attribute, since this information is needed to compute $trace(\mathbf{W})$.

This evaluation function is one of a set based on the matrix identity, $\mathbf{T} = \mathbf{W} + \mathbf{B}$, where $\mathbf{T}$, $\mathbf{B}$ and $\mathbf{W}$ are the total, between-group, and within-group scatter matrices, respectively.[6] In general, these functions attempt to either minimize $\mathbf{W}$ (a measure of within-group differences) or maximize $\mathbf{B}$ (between-group differences). In order to compare matrixes, they must be converted to a scalar: one can use either the determinant of the matrix, or (more cheaply) the 'trace' of a matrix. The three most common functions are minimizing $trace(\mathbf{W})$ (defined above), maximizing $trace(\mathbf{W}^{-1}\mathbf{B})$, and minimizing the determinant of $\mathbf{W}$.

It is important to note that $trace(\mathbf{W})$ has the same problem as Euclidean distance; it is sensitive to normalization of the data, and to linear transformations of attributes. It also prefers

[6] See Hand (1981) for a more detailed discussion of this identity, as well as further references to the use of these functions.

clusters that form hyper-spheres in the attribute space. However, the other proposed functions are computationally much more expensive, especially since they need to compute the inverse of the matrix $\mathbf{W}$. Anderberg (1973, p. 175) cites evidence that there are no good reasons for selecting one function over another, and therefore he suggests considering only $trace(\mathbf{W})$, the simplest function.

These average distances functions are defined to work with instances described by continuous attributes. Although one can define a 'distance' between two instances with symbolic attributes, there certainly cannot be a mean for each symbolic attribute. If the data includes symbolic attributes, the researcher must use some other evaluation function.

3.4.2 Attribute Correlation

Hanson and Bauer (1989) describe an evaluation function for symbolic attributes based on correlation between attributes. Each class is defined by all possible pairwise contingency tables (as in Table 1). To evaluate the class, the distribution over each contingency table, D_{ij}, is averaged for all pairs of attributes; the greater this average correlation, the better the class. The 'cooccurance distribution' (to use the authors' term) is defined as:

$$D_{ij} = \frac{\sum_m \sum_n (x_{mn} log\ x_{mn})}{(\sum_m \sum_n x_{mn})(log\ \sum_m \sum_n x_{mn})} ,$$

where x_{mn} is a count in the contingency table for attributes i and j. Recall that each position in a contingency table counts the number of times attribute value m for attribute i co-occurs with value n for attribute j. The average value of D_{ij} is a measure of the within-class cohesion. The authors label this as W_c, and state that the complete evaluation function is to maximize W_c/O_c, where O_c is defined in a similar way, except that the cohesion is measured across classes. Once again, the general goal is to maximize within-group similarities, and between-group differences.

This 'correlation' measure, D_{ij}, is more closely related to the chi-square test of independence than Pearson's correlation coefficient for continuous attributes. For this reason at least, their method is distinct from Q-mode factor analysis. However, Anderberg (1973) shows that there are dangers in interpreting results with any measure based on the correlation of attributes. In particular, the distribution measure may not be consistent across different attributes; if $D_{ij} = .52$, and $D_{kl} = .41$, one *cannot* conclude that i and j are more closely related than k and l.

Another problem with this particular method is that it is only well suited to domains with relatively few binary attributes. Although there are no theoretical problems with other types of data, the storage costs and, to a lesser degree, the computational costs become very severe. With K attributes each with M values, each concept needs $(K^2 - K)/2$ tables, each of which has M^2 entries.

The advantage of a method based on correlation is that it can easily find concepts that depend on some relationship between two attributes.[7] Because this relationship is explicitly represented in

[7] Of course, as used here, 'correlation' only refers to pairs of attributes. Three-way, or N-way correlations are not explored.

the correlation tables, the concept can be easily discovered and represented by Hanson and Bauer's system. Systems that do not use some form of correlation often have difficulty with this type of class.

3.4.3 Functions Based on Information Theory

Gluck and Corter (1985) present an 'information theoretic' evaluation function, *category utility*, for symbolic attributes. This function is based on the probability of an attribute value, $P(x_{kv})$. This probability can be expressed as the number of times attribute k has had value v, divided by the total number of instances. (Note that this probability is closely related to the simple matching measure.) Category utility measures the information that is gained by partitioning instances into classes. For a given attribute k,

$$Category\ Utility(k) \quad = \quad \frac{\sum_{j=1}^{J}\left[P(C_j)\sum_{v=1}^{V}P(x_{kv}|C_j)^2\right] \quad - \quad \sum_{v=1}^{V}P(x_{kv})^2}{J} \quad ,$$

where V is the number of attribute values for attribute k and J is the number of classes. $P(x_{kv}|C_j)$ is the probability of the attribute value conditioned by the class C_j, meaning that only those instances in class C_j are considered. In contrast, $P(x_{kv})$ is that probability without any class information; it is the information at the parent class.[8] Although category utility is based on the simple matching measure, the subtraction of the final term allows the function to measure information gain from parent to children. This gain is then divided by the number of children, so that different size partitions can be compared.

Both category utility and Hanson and Bauer's evaluation function work only for symbolic attributes; because they iterate through all possible attribute values, they cannot be applied to continuous attributes. Classes are defined as a set of probabilities for every possible attribute-value pair. Gennari, Langley, and Fisher (1989) use category utility as the basis for a related measure for normally distributed continuous attributes. Because a normal distribution is assumed, this measure is based on the standard deviation, σ_k, for a given attribute, k. The evaluation function used by the CLASSIT system of Gennari et al. is:

$$\frac{\sum_{j=1}^{J}P(C_j)/\sigma_{jk} \quad - \quad 1/\sigma_{pk}}{J} \quad ,$$

where σ_{jk} is the standard deviation within a given class j, and σ_{pk} is the standard deviation without any class information.

[8] Gluck and Corter (1985) defined category utility for two classes; here, I have shown Fisher's (1987a) generalization to J classes. The information theoretic model also uses logs instead of squared terms ($P(x)log(P(x))$ instead of $P(x)^2$). However, the authors claim that this difference will not affect the behavior of the clustering system.

As one might guess, this measure is similar to minimizing $trace(\mathbf{W})$. For instance, classes are defined in the same way – a set of means and standard deviations for each attribute. In fact, the only important difference is the subtraction of $1/\sigma_{pk}$. However, this is exactly what distinguished Gluck and Corter's category utility from other evaluation functions. Additionally, there is some rough evidence that subtracting this context information reduces the measure's sensitivity to re-scaling or linear transformations of attributes, a definite problem for the $trace(\mathbf{W})$ function.[9]

3.4.4 Bayesian Class Evaluation

Although a Bayesian clustering system explicitly compares classes, it does not usually have the same type of evaluation function as those described so far. Instead of evaluating classes with respect to all instances, the basic decision theory equation (presented in section 2.4) compares a single instance against a set of classes. The difficulty with this equation is that in order to compute the class conditional probabilities, $P(x|\omega_i)$, one needs an estimation of the class parameters that define each ω_i. Duda and Hart (1973) conclude that, in general, there is no analytically simple way to find this estimation, and that computational costs for an exact solution rise exponentially with the number of instances.

However, there are a number of estimation techniques that have been empirically successful. Fried & Holyoke, 1984, use a simpler algorithm based on the Euclidean distance similarity measure to find initial class parameter estimates. Using these estimates, they can then determine $P(x|\omega_i)$. Anderson (in press) uses a *coupling probability* (a user-defined parameter) to define the prior probability of each class and the probability that an instance is a member of a new class. This allows him to get an initial partitioning of instances into classes. Then, for a new instance, he can compute $P(x|\omega_i)$ based on the membership of each ω_i.[10]

Unfortunately, it is difficult to compare these systems' "evaluation functions" to others. Part of the difficulty is that the task is described as estimating (or updating) class parameters, rather than evaluating the quality of class definitions. For example, Cheeseman et al. (1988) describe the update algorithm for their Bayesian system, but do not give an equation for this evaluation process, nor describe how it compares to other clustering evaluation functions.

4. Algorithms for Clustering

Any clustering technique can be described as an algorithm that uses some kind of a similarity measure or evaluation function to search for a set of classes. Thus far, I have described these measures out of context; this section will show how they are used by algorithms. My goal in presenting these components separately is to show that any combination of algorithm and measure defines a clustering technique. To some degree, one may choose a part from bin 'a' and a part from

[9] This effect was observed by the author while experimenting with the Classit system.

[10] Because Anderson works with symbolic attributes, his evaluation function is related to the simple matching measure, except that the new instance is compared to the set of all member instances. Anderson's work is also interesting because his algorithm is *incremental* (see section 4.3).

bin 'b' to create a clustering method. Of course, very few researchers have actually tried this, and not every nut will fit onto every bolt. For each algorithm, I will point out the original similarity measure proposed, as well as others that could be used.

Although there are several ways to organize algorithms, the most useful is based on their approach to the clustering problem. I have chosen to divide algorithms into three groups: *agglomerative* algorithms, *iterative optimization* algorithms and *incremental* algorithms. The agglomerative approach is the oldest, having been proposed by workers in biology and ecology. With the advent of the computer, iterative optimization methods became popular as a more efficient heuristic approach to clustering. Finally, incremental algorithms were inspired by human concept formation, and were created by researchers in machine learning.

In addition to describing some of the most important algorithms, I will consider two characteristics of each method. First, although any algorithm must produce some set of classes as output, the form and organization of the classes is dictated by the researcher's goals. For example, the researcher may prefer a simple list of classes or he may need a specific-to-general hierarchy of classes. Likewise, the researcher may prefer each instance to be assigned to a single class, or to more than one class, or even to all classes probabilistically. Second, different algorithms have very different computational and memory costs. The computer cannot be treated as an infinitely powerful machine. Especially from a machine learning point of view, it is important that the algorithm and the similarity measure be as inexpensive as possible.

4.1 Agglomerative Methods

Historically, the first algorithms for clustering were agglomerative methods. Since they were developed by biologists, they produced a hierarchy (a taxonomy) of classes, from the most general class (including all instances) to the most specific classes (covering only one instance). Although these are still the most widely used algorithms, they are expensive both in space and time requirements.

An agglomerative method begins with each instance as a separate class, and repeatedly combines these smaller, specific classes to form larger, more general classes. This process builds up a hierarchy of classes, finishing when all instances have been agglomerated into one top-level class.[11] In order to determine which instances to 'agglomerate', these algorithms require a *similarity matrix* that shows how close, according to some similarity measure, every instances is to every other instance. Given this matrix, a general agglomerative algorithm can be described as follows:

1. `Compute and store the similarity matrix.`
2. `Find the smallest (best) value in the matrix and its associated pair of instances.`
3. `Merge these two instances (or classes) into a larger class.`

[11] The reverse of this approach is known as a *divisive* algorithm. This begins by assuming every instance is in the same highest-level class, then repeatedly divides this class into some number of children, until each (very specific) class has only one instance. Although a few such algorithms have been proposed (MacNaughton-Smith et al. 1964, Fisher 1984), they have been rarely used.

4. `Create a new similarity matrix that includes the new class. (This may require re-computing some values.)`
5. `If there is only one class, return the hierarchy produced and stop. Else, go to step 2.`

This description does not make clear exactly how to accomplish step four. In order to enter a new class into the similarity matrix, one must decide how to use the similarity measure to compare a class to an instance, or to other classes. The simplest (or at least the cheapest) solution is to define the similarity between two classes, A and B, as the similarity of the closest two instances a and b, where $a \in A$ and $b \in B$. This is called the "nearest neighbor" or "single linkage" algorithm, because two classes are combined by the shortest single link between them. This algorithm tends to produce long chaining clusters; depending on the domain, this can be a disadvantage.

Other agglomerative algorithms use different ways to measure the 'similarity' between two classes. For example, instead of a single link, one can find the "average link" distance between classes. This requires computing the class mean for each attribute every time a class is created or expanded, and then measuring the distance to other groups or instances from this mean. This method avoids creating long chains, and instead prefers clusters that form hyper-spheres in the instance space.

Unfortunately, this method is rather expensive: because it requires recomputing part of the similarity matrix, it has a computational cost of $O(n^3)$, where n is the number of instances. The nearest neighbor algorithm can be implemented so that no recalculation occurs, but the computational cost is still $O(n^2)$, and it requires a sorted similarity matrix. Depending on the similarity measure and the number of attributes per instance, computing and storing the similarity matrix alone may be prohibitively expensive. In fact, for domains where there are a large number of attributes and instances, agglomerative methods are largely unsuitable.

Another significant problem with agglomerative algorithms is that they produce only *binary* hierarchies. Instead of this structure, the researcher is often interested in finding some 'optimal' number of classes. Although there are some methods for cutting or flattening the binary hierarchy (see Aldenderfer & Blashfield, 1984), these are not guaranteed to find the best or most appropriate set of classes. In fact, Everitt (1979) points out that such techniques may be misleading; he concludes that finding the optimal number of classes from a binary hierarchy is an open problem.

4.2 Iterative Optimization

Iterative optimization algorithms were created in response to the expense of agglomerative methods and to the difficulty of finding the correct number of classes. Instead of trying to find this ideal number, one can assume that the number of classes, k, is given to the clustering system. Iterative optimization searches for these k classes by repeatedly re-assigning instances to different classes in order to improve the score of some evaluation function. Although these algorithms do

not produce hierarchies of classes, they are more efficient than agglomerative methods, and by transferring instances from class to class they can recover from an initial 'bad' decision.

In general, one can view the clustering problem as a search over the huge space of possible partitionings of the instances into classes. A simple method would examine every possible partition, and find the one with the best score according to an evaluation function. Unfortunately, this is computationally impossible even with a relatively small number of instances; for example, there are 5.28×10^{28} ways to partition 50 instances into four classes.[12] Therefore, instead of a complete search through this space, iterative optimization methods use hill-climbing techniques to iteratively improve the evaluation score until an optimum is reached. As with any hill-climbing method, the starting point for the search may be critical, and the algorithm can converge on a local optimum instead of the global optimum.

Theoretically, one can use any evaluation function as the criterion to optimize at each iteration. However, in order to keep the overall clustering system efficient, the researcher should choose a relatively simple evaluation function, one that that system can compute cheaply as it considers each re-assignment. For example, one of the simplest and most popular algorithms is the k-means algorithm:

1. Use the first k instances as seed points.
2. Assign each of the remaining instances to the class represented by the nearest (Euclidean distance) seed point.
3. Recompute new seed points as the centroids (the average attribute values) of each class.
4. Iterate between steps 2 and 3 until no re-assignments are made.

Although the number of iterations required before halting is unknown, Anderberg (1973) gives a proof that such algorithms will eventually converge, and in practice this is usually a reasonably small number (less than ten). When Euclidean distance is used, Hand (1981) shows that this algorithm is equivalent to optimizing the $trace(\mathbf{W})$ evaluation function.

One can make a number of modifications to this algorithm. First, since the starting point can be critical to a hill-climbing searcher, different methods can be used to choose it. For example, the k seed instances can be chosen randomly, or they can be chosen so that all seeds are at least some minimum distance apart. Duda and Hart (1973) point out that the entire algorithm can be repeated with different seed selections so that the researcher can compare possibilities. In fact, they even suggest using an agglomerative method to find the initial partition, although this seems expensive. Anderberg (1973) also describes a number of seed selection techniques.

A second modification can be made by computing new class centroids whenever an instance is re-assigned to a class. In this case, the algorithm may converge much earlier; for example, MacQueen's (1967) k-means algorithm uses only two passes through the instances. In the first

[12] Duda and Hart (1973) give the exact expression for the number of ways to partition n instances into c classes; an approximation is $c^n/c!$.

pass, centroids are modified as each re-assignment occurs; during the second pass the centroids remain fixed.

Finally, one can expand the algorithm so that it addresses two higher-level problems. First, because one may not always be able to specify the number of classes, k, *a priori*, one can try the k-means algorithm with different values of k, allowing an estimate of the 'best' k. Second, the researcher may need a hierarchy of classes, rather than the flat list that iterative optimization algorithms usually produce. To this end, one can simply execute the algorithm recursively on each of the k groups identified on its first execution.

Although these extensions appear to be expensive, 'brute-force' solutions, Michalski and Stepp (1983b) have incorporated them into the k-means algorithm in their CLUSTER/2 program.[13] This system also includes a step that helps avoid local optima: when the k-means algorithm converges, CLUSTER/2 chooses new seed points at the edge of each class, instead of at the centroids. It then restarts the k-means algorithm with these new seeds, and if the resulting partition remains unchanged, the system returns those classes as the solution. Finally, CLUSTER/2 differs from most iterative optimization techniques in that it uses domains with symbolic, rather than continuous attributes. As the $trace(\mathbf{W})$ evaluation function only works with continuous domains, this system uses a special purpose evaluation function defined by a set of (user-specified) parameters. Unfortunately, iterative optimization has not been tried with any more principled evaluation function for symbolic attributes.

Cheeseman et al. (1988) also use an iterative optimization approach, but their system, AUTOCLASS, does not strictly partition instances into classes. This system carries out a hill-climbing search for the best set of classes, where each class is defined by a mean and a standard deviation. However, class membership is completely probabilistic, specifying the probability that $x \in c$, for any object x and for every class c. These 'fuzzy' classes are especially suitable when an instance is described reasonably well by two (or more) competing classes.

Although iterative optimization algorithms have met with some clear successes, their domain of application is somewhat constrained. If an approximation of k is available, if the instances are described by continuous attributes, and if a flat list of classes is sufficient, then iterative optimization is a good approach to the clustering problem. Although the CLUSTER/2 system relaxes these constraints, it does so only at substantial computational cost.

4.3 Incremental Methods

In contrast to other approaches, incremental algorithms view the set of instances as a potentially infinite sequence. As each instance is processed, the algorithm makes an incremental modification to its current set of concepts. At any point in time, the concepts reflect information gained from all the instances encountered up to that point. This approach was designed with the

[13] Cluster/2 is not usually identified as using a k-means iterative optimization clustering algorithm. This characterization became clear only after surveying these older methods.

human clustering system in mind; it seems unlikely that a human learner would need to first receive some number of instances, and then stop receiving and perform the computation of the clustering task.

Even without this bias, these incremental systems are useful for a number of more pragmatic reasons. They require less computational time than other algorithms, and can therefore process larger databases. These algorithms also avoid the problem of selecting the number of classes. Finally, this type of algorithm is almost essential for any application where the class definitions are dynamic. Schlimmer and Granger (1986) refer to this as *concept drift*: if new instances reflect new or different concepts, an incremental algorithm can adjust its concept definitions over time.

A general incremental algorithm for adding each new instance x to a hierarchy of classes can be described as:

1. `Incorporate` x `to the root node.`
2. `Either: a) Incorporate` x `to an existing child class, or`
   `b) create a new child class based on` x`.`
3. `Recurse (if desired) on each child class.`

Usually these algorithms produce a hierarchy of classes, but step three can be omitted if one prefers a simple list of classes. Unlike agglomerative methods, incremental algorithms do not produce binary hierarchies: the branching factor is variable and determined by how often new classes are created (in step 2b). Determining when to make a new class is critical to these algorithms – this choice allows incremental algorithms to automatically find an appropriate number of classes from the data.

EPAM (Feigenbaum, 1963) is one of the first systems in artificial intelligence to approach the clustering problem. This system applies *monothetic* decision making to the basic incremental algorithm. The system associates each level in the hierarchy with a single attribute. In order to determine which action to take in step two (above), EPAM inspects the value of that attribute, and creates a new disjunct if that value does not match any of the existing children. Otherwise, the system sorts the instance to the child class with a matching attribute value.

In contrast, UNIMEM (Lebowitz, 1985, 1986) uses a *polythetic* strategy. At each level, the system inspects some subset of the attributes, and then uses these values in conjunction with an evaluation function to choose a class. This algorithm also goes beyond a strict partitioning to allow *clumping*, or sorting instances to more than one class. However, UNIMEM does not allow the completely probabilistic classification suggested by Cheeseman et al. (1988).

As defined above, the incremental algorithm is a pure hillclimber — it can get trapped in the same kind of local optima as iterative optimization methods. Fisher's COBWEB system (1987a, 1987b) added some operators to the algorithm that were designed to alleviate this problem. In addition to options a) and b) at step two, the system considers *merging* two existing classes, or *splitting* a class into its children. These operators permit the system to move away from local optima, since they allow a form of backtracking through the space of possible concept hierarchies.

As described earlier, machine learning approaches to clustering prefer creating intensional class definitions. Incremental algorithms modify these definitions with each new instance, rather than adding the instance to a membership list. This suggests that an incremental system need not store every instance. This ability leads to reduced memory costs, especially if the number of instances is very large. Developing ways to 'forget' instances, or avoid storing them at all, is a current research topic (Gennari et al., 1989).

5. Evaluating a Clustering Method

With this many clustering methods to choose from, one would hope for some principled way of comparing different methods. In general, we would like to have a quantitative measure for how well a given method has succeeded. Unfortunately, because the goals of clustering vary and are often very poorly defined, a single definitive measure is impossible. To some degree, any quantitative comparison of results is impossible because the goal of clustering is subjective. For example, some researchers prefer one clustering method to another simply because it produces a 'better', 'more intuitive' or 'more pleasing' set of classes. However, researchers with a more formal bias have found a number of ways to measure different aspects of a solution.

An intuitive measure used in discriminant analysis and in machine learning is to see how often a system classifies instances correctly. In these domains, the correct class is known beforehand, so the hierarchy produced by some number of "training" instances can be evaluated by how well it classifies a number of "test" instances. For clustering methods, the problem is that the 'correct' class is not known. At best, this measure can be used to compare an 'intuitive' set of classes with the new set produced from the data; it is usually *not* a validation of the process that created the new classes.

One measure that can be used to compare hierarchies is "cophenetic correlation". This method compares the output of agglomerative algorithms, and is therefore used mostly by biologists. This measure uses correlation to compare the original similarity matrix with a new matrix that is derived from the final hierarchy. An entry (i, j) in the new matrix has a value equal to the similarity measure when instances i and j were agglomerated together in the hierarchy. These two matrices can then be compared by computing their correlation – a high correlation between entries suggests that the clustering method did a good job of capturing the information in the original similarity matrix. Unfortunately, Aldenderfer and Blashfield (1984) point out that this measure is not statistically sound. The use of correlation assumes that the values in the matrices are normally distributed. Because the derived matrix is dependent on the original matrix and contains much less information, this is usually not the case.

As an alternative, Aldenderfer and Blashfield suggest that a better way to compare concept hierarchies is to use what they call a "Monte Carlo" validation method. There are three steps in this procedure. First, a set of random data is generated that is normally distributed and based on the averages of the original data. In effect, this represents the original instances, but grouped as a single class. Next, this random data set is clustered, resulting in a "base-line" hierarchy.

Finally, the original hierarchy is compared to this base-line hierarchy (for example, by comparing an analysis of variance within each class). A large difference between hierarchies means that the algorithm has done a good job of finding classes from the data. Although the score resulting from this procedure has little absolute meaning, it can be used to compare a set of different methods – the method with the biggest difference is the 'best' for a given data set.

Rather than comparing hierarchies, there is a more general way of assessing the performance of a clustering method. Instead of evaluating whether a classification is 'correct', the idea is to judge how 'useful' that classification may be. Although this may seem difficult to do, one measurement of 'useful' is to see how well the classes can predict attribute values of a new instance. This prediction ability can be related to the 'recall' task in cognitive psychology, and has been used in machine learning (Fisher, 1987; Gennari et al., 1989). The recall task says that given a set of cues (attributes) from a new instance, the agent should be able to use its memory of past instances to recall the unspecified attributes of the instance.

To use prediction as a performance task for evaluating a clustering system, a random attribute is omitted from a test instance. After the system classifies this instance, the chosen class is used to predict the value of the omitted attribute. In addition to depending on the goodness of the classes, the accuracy of this prediction depends on the test set instance (how typical that instance is) and the omitted attribute (how consistent that attribute is). Therefore, once this score is averaged over instances and attributes, this average predictive accuracy can be used as a general measure of the utility of the classes created by the clustering system. This performance measure can then be used to compare different methods with the same data, or the same method over different data sets.

6. Conclusions

The framework delineated in this paper allows for a comparison of clustering methods across a wide spectrum of research fields. By describing a technique in relation to others proposed by different fields, one is able to concentrate on distinguishing features. Additionally, an awareness of these related efforts allows the researcher to avoid duplication of work.

Although 'concept formation' in machine learning offers some new insights to the clustering problem, there is certainly information to be gained from other clustering methods. Machine learning can gain simply by realizing the scope and variety of existing clustering methods. Too often, the researcher in concept formation proceeds without any awareness of other potential solutions. This can mean that the researcher may not apply an existing solution to a particular clustering problem, or worse yet, he may present 'new research' that is identical or very close to an older solution to the same problem.

At the same time, machine learning certainly offers a new perspective for clustering methods. The biases it brings from cognitive psychology can be useful if applied to traditional cluster analysis

methods. For example, the preference for low-cost, incremental methods brings at least computational benefits without a loss of performance. Also, using predictive accuracy as a performance task seems to provide a useful method of validation in an otherwise unexplored area.

This survey of clustering methods is intended as the beginnings of a bridge between statistical methods and efforts in machine learning. A recognition of these older efforts has been overdue in the machine learning literature, while an understanding of machine learning's "concept formation" work is missing in other fields. By combining resources, the search for good solutions to the general clustering problem can advance as a unified endeavor.

References

Aldenderfer, M. and Blashfield, R. (1984). *Cluster analysis.* Beverly Hills: Sage Publications.

Anderberg, M. (1973). *Cluster analysis for applications.* New York: Academic Press.

Anderson, J. R. (1988). The place of cognitive architectures in a rational analysis. *Proceedings of the Tenth Annual Cognitive Science Conference* (pp 1–10). Montreal, CA:Lawrence Erlbaum.

Anderson, J. R. (in press). The place of rational analysis in a cognitive architecture. In K.Van Lehn (Ed.), *Architectures for intelligence.* Hillsdale, NJ: Lawrence Erlbaum.

Barsalou, L. (1987). The instability of graded structure: implications for the nature of concepts. In U. Neisser (Ed.), *Concepts and conceptual development: Ecological and intellectual factors in categorization.* New York: Cambridge University Press.

Cheeseman, P., Kelly, J., Self, M., Stutz, J., Taylor, W., & Freeman, D. (1988). Autoclass: A Bayesian classification system. *Proceedings of the Fifth International Conference on Machine Learning* (pp 54–64). Ann Arbor, MI: Morgan Kaufmann.

Corter, J., Gluck, M., & Bower, G. (1988). Basic levels in hierarchically structured categories. *Proceedings of the Tenth Annual Cognitive Science Conference* (pp 118–124). Montreal, CA:Lawrence Erlbaum.

Duda, R. & Hart, P. (1973). *Pattern classification and scene analysis.* New York: John Wiley and Sons.

Everitt, B. (1979). Unresolved problems in cluster analysis. *Biometrics, 35*, 169–181.

Everitt, B. (1980). *Cluster analysis.* London: Heinemann Educational.

Feigenbaum, E. A. (1963). The simulation of verbal learning behavior. In E. A. Feigenbaum & J. Feldman (Eds.), *Computers and thought.* New York: McGraw–Hill.

Fisher, D. (1984). *A hierarchical conceptual clustering algorithm* (Technical Report 85-21). Irvine: University of California, Department of Information and Computer Science.

Fisher, D. (1987a). Knowledge acquisition via incremental conceptual clustering. *Machine Learning, 2*, 139–172.

Fisher, D. (1987b). *Knowledge acquisition via incremental conceptual clustering.* Doctoral dissertation, Department of Information & Computer Science, University of California, Irvine.

Fisher, D., & Langley, P., (1986). Methods of conceptual clustering and their relation to numerical taxonomy. In W. Gale (Ed.), *Artificial intelligence and statistics.* Reading MA: Addison Wesley.

Fried, L. S., & Holyoak, K. J. (1984). Induction of category distributions: A framework for classification learning. *Journal of Experimental Psychology: Learning, Memory, and Cognition, 10*, 234–257.

Gennari, J. H., Langley, P., & Fisher, D. (1989). Models of incremental concept formation. *Artificial Intelligence, 40*, 11–61.

Gluck, M., & Corter, J. (1985). Information, uncertainty and the utility of categories. *Proceedings of the Seventh Annual Conference of the Cognitive Science Society* (pp. 283–287). Irvine, CA: Lawrence Erlbaum.

Hamming, R. W. (1980). *Coding and information theory.* Englewood Clifs, NJ: Prentice Hall.

Hand, D. J. (1981). *Discrimination and classification.* New York: John Wiley & Sons.

Hanson, S. J., & Bauer, M. (1989). Conceptual clustering, categorization, and polymorphy. *Machine Learning, 3,* 343–372.

Jaynes, E. T. (1986). Bayesian methods: General background. In J. H. Justice (Ed.), *Maximum entropy and Bayesian methods in applied statistics* (pp. 1–25). Cambridge, MA: Cambridge University Press.

Langley, P. (1988). Machine learning as an experimental science. *Machine Learning, 3,* 5–8.

Lebowitz, M. (1985). Categorizing numeric information for generalization. *Cognitive Science, 9,* 285–309.

Lebowitz, M. (1986). Concept learning in a rich input domain: Generalization based memory. In R. S. Michalski, J. G. Carbonell, & T. M. Mitchell (Eds.), *Machine learning: An artificial intelligence approach* (Vol. 2). Los Altos, CA: Morgan Kaufmann.

MacNaughton-Smith, P., Williams, W. T., Dale, M. B., & Mockett L. G. (1964). Dissimilarity analysis. *Nature, 202,* 1034–1035.

Mervis, C. & Rosch, E. (1981). Categorization of natural objects. *Annual Review of Psychology, 32,* 89–115.

Michalski, R. S., & Stepp, R. (1983a). Learning from observation: Conceptual clustering. In R. S. Michalski, J. G. Carbonell, & T. M. Mitchell (Eds.), *Machine learning: An artificial intelligence approach.* Los Altos, CA: Morgan Kaufmann.

Michalski, R. S., & Stepp, R. (1983b). Automated construction of classifications: Conceptual clustering versus numerical taxonomy. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 5,* 396–409.

Romesburg, H. C. (1984). *Cluster analysis for researchers.* Belmont, CA: Lifetime Learning Publications.

Schlimmer, J. C., & Granger, J. H. (1986). Beyond incremental processing: Tracking concept drift. *Proceedings of the Fifth National Conference on Artificial Intelligence* (pp. 502–507). Philadelphia, PA: Morgan Kaufmann.

Smith, E. E., & Medin, D. L. (1981). *Categories and concepts.* Cambridge, MA: Harvard University Press.

Stepp, R. (1987). Concepts in conceptual clustering. *Proceedings of the Tenth International Joint Conference on Artificial Intelligence* (pp. 211–213). Milan, Italy: Morgan Kaufmann.