An Analysis of the WITT Algorithm

JAN L. TALMON AND HERCO FONTEIJN

TALMON@MI.RULIMBURG.NL

*Dept. of Medical Informatics, University of Limburg, PO Box 616, 6200 MD Maastricht,
The Netherlands*

PETER J. BRASPENNING

BRASPENNING@CS.RULIMBURG.NL

Dept. of Computer Science, University of Limburg, PO Box 616, 6200 MD Maastricht, The Netherlands

Editor: Kenneth DeJong

Abstract. In this article we present an analysis of the WITT algorithm for conceptual clustering as proposed by Hanson and Bauer (1989). We show that the measures proposed for the original WITT algorithm have serious shortcomings. We propose some alternatives for these measures, and, moreover, we make a further analysis of these alternatives such that setting the required thresholds will be less dependent of the characteristics of the cases that are to be clustered.

Keywords: Conceptual clustering, cohesion measures, normalized measures

1. Introduction

The WITT algorithm for conceptual clustering as described by Hanson and Bauer (1989) has some interesting properties. The clusters created are accompanied by the correlational structure of the attribute values of the cases in those clusters. This correlational structure can be used as a means to describe the clusters. Additionally, this structure provides a means to determining the extent to which a *new* case belongs to each of the already defined clusters. Hence, it can be considered to define a classification scheme. This property also allows for *incremental* learning. Once a new case is permanently added to a cluster, the correlational structure of that cluster may be updated.

Since we are interested in developing a tool box with learning algorithms, which should become part of a tool for knowledge acquisition, we have implemented a version of the WITT algorithm for experimentation. Our initial experiments showed that the measures that govern the clustering process have some very undesirable properties. Accordingly, we decided to make an analysis of the measures of the algorithm as proposed by Hanson and Bauer.

In the following, we describe briefly the global structure of the WITT algorithm. Then the measures presented by Hanson and Bauer are discussed extensively and, where necessary, our interpretation of some of the less clear parts of their paper is given. Next, we analyze their measures and demonstrate a number of undesirable properties. Finally, some alternatives are proposed that at least overcome the problems we have identified.

2. Description of the WITT algorithm

The WITT algorithm consists of two components: a **preclustering** algorithm and a **refinement** algorithm. The task of the preclustering algorithm is to generate a few initial clusters,

based on some distance measure between cases. Subsequently, the refinement algorithm adds cases to these clusters, provided those cases are similar enough to one of these clusters. When the refinement algorithm fails, the preclustering algorithm tries to make new clusters using cases not yet assigned to existing clusters. When this fails as well, existing clusters may be merged. In pseudo-code, the complete algorithm appears as follows:

```

PRECLUSTER → initial clusters
STOP := FALSE
REPEAT
    find case that is most similar to one of the existing clusters, using a cohesion measure ( $C_c$ ,
    see below for the definition of the measures)
    IF cohesion  $C_c$  is larger than threshold_1 THEN
        add case to cluster
    ELSE BEGIN
        PRECLUSTER → candidate clusters
        IF there is at least one candidate cluster  $c$  sufficiently different from existing ones (within-
        cluster cohesion  $W_{i \cup c} < \text{threshold\_2}$  for all  $i$ ) THEN
            add those different clusters to the list of existing ones
        ELSE BEGIN
            Try to merge two existing clusters
            IF within-cluster cohesion of the two most similar clusters  $> \text{threshold\_2}$  (the
            clusters  $i$  and  $j$  for which  $W_{i \cup j}$  is maximal,  $i \neq j$ ) THEN
                execute merging of clusters
            ELSE STOP := TRUE
        END
    END
UNTIL no cases left or STOP = TRUE.

```

It is clear that this algorithm does not always cluster all cases. When the remaining cases cannot be clustered such that the cohesion is sufficiently large and when no existing clusters can be merged, the algorithm stops. Leaving cases not clustered is in itself a desirable property.

Following Hanson and Bauer's paper, the refinement algorithm of WITT is based on the cohesion C_c of a cluster c , which is defined as

$$C_c = \frac{W_c}{O_c}, \quad (1)$$

in which W_c represents the within-cluster cohesion and O_c the average cohesion between c and all other clusters.

The within-cluster cohesion is defined as the average variance in the co-occurrences of all possible attribute-value pairs for a given cluster. When we denote the variance in the co-occurrences of the values of an attribute pair $\{i, j\}$ as $D(i, j)$, the within-cluster cohesion is formally given by

$$W_c = \frac{\sum_{i=1}^{K-1} \sum_{j=i+1}^K D(i, j)}{\frac{K * (K - 1)}{2}}, \quad (2)$$

where K is the number of attributes. $D(i, j)$ is computed from the contingency table f_{ij} . The element $f_{ij}(l, m)$ of such a table represents the number of cases in the cluster to which f_{ij} pertains that have the l th value for attribute i and the m th value for attribute j . $D(i, j)$ is defined as

$$D(i, j) = \frac{\sum_{l=1}^L \sum_{m=1}^M f_{ij}(l, m) \ln f_{ij}(l, m)}{\sum_{l=1}^L \sum_{m=1}^M f_{ij}(l, m) * \ln \left[\sum_{l=1}^L \sum_{m=1}^M f_{ij}(l, m) \right]}, \quad (3)$$

with L and M being the number of distinct values of attributes i and j , respectively. L and M are equal to the number of rows and columns of the contingency table f_{ij} .

The term O_c in (1) reflects the average cohesion of a cluster with all other clusters. It is defined as

$$O_c = \frac{\sum_{\substack{k=1 \\ k \neq c}}^P B_{ck}}{P - 1}, \quad (4)$$

with P the number of clusters.

Hanson and Bauer describe and define B_{ck} in the following way (page 349):

We are interested in the relative cohesion between two categories c and k [clusters in our terminology]

$$B_{ck} = \frac{1}{W_c + W_k - 2W_{c \cup k}} \quad (5)$$

which we will assume is related to the intersection of the two categories. The term B_{ck} measures the variance in the attribute-value co-occurrences in the union of the categories, relative to the variance in the isolated categories.

Since we have experienced some difficulties with the algorithm as summarized in this section, we analyze the behavior of $D(i, j)$ in detail in the next section. We provide an analysis of O_c in the section thereafter, including its underlying measure B_{ck} . Suggestions are given as to how to overcome the difficulties we have encountered.

3. Analysis of the D-measure

The D-measure as defined in (3) above, which is the basis for all other measures used in the refinement algorithm, turns out to be problematic when thoroughly analyzed. Hanson and Bauer call $D(i, j)$ the *variance* of the co-occurrence of the values of an attribute pair. This is a misleading term for $D(i, j)$. It is easily seen from (3) that $D(i, j) = 1$ when there is no variance at all in the co-occurrences, i.e., for each attribute all cases in the cluster have the same value. A better interpretation of $D(i, j)$ is that it represents the *predictive power* or *information content* of the co-occurrences of the attribute values.

Hanson and Bauer give four example contingency tables f_{ij} (each example describes a cluster of four cases) with their corresponding $D(i, j)$ value (tables 1-4):

Table 1. $D(i, j) = 1.00$.

0	0
0	4

Table 2. $D(i, j) = 0.59$.

0	1
0	3

Table 3. $D(i, j) = 0.25$.

0	1
1	2

Table 4. $D(i, j) = 0.00$.

1	1
1	1

The calculation of the $D(i, j)$ is straightforward. For example, for table 3,

$$D(i, j) = \frac{0\ln 0 + 1\ln 1 + 1\ln 1 + 2\ln 2}{4\ln 4} = 0.25. \quad (6)$$

Table 4 is the most interesting. If we double the number of cases, we obtain table 5,

Table 5. $D(i, j) = 0.33$.

2	2
2	2

where $D(i, j) = 4 * 2\ln 2 / 8\ln 8 = 0.33$.

In more general terms, for a $2 * 2$ table of the general form of table 6,

Table 6. A contingency table with equal distribution of the cases over the cells.

n	n
n	n

$$D(i, j) = \frac{4 * n \ln n}{4n \ln 4n} = \frac{\ln n}{\ln n + \ln 4} = \frac{1}{1 + \frac{\ln 4}{\ln n}}, \quad (7)$$

when $n \rightarrow \infty$, $D(i, j) \rightarrow 1$. This implies that $D(i, j)$ does *not* reflect the degree of co-occurrence of attribute-value pairs when n becomes large. This behavior is illustrated by comparing the values of $D(i, j)$ for tables 4 and 5. The value of $D(i, j)$ for the latter is larger than for the former, contrary to what intuition tells us. Since all $D(i, j)$'s of a cluster have this tendency, W_c also goes to 1 when the number of included cases increases. Since this holds for all clusters, it becomes more and more difficult to distinguish between them.

Another problem is that the *size* of the contingency tables has no effect on the value of $D(i, j)$. The $D(i, j)$ for table 7 is the same as for table 5, again somewhat counterintuitively. To overcome both of these problems, we propose another approach for the D-measure.

Table 7. An example of a table with a large number of empty cells.

2	2	0	0	0	...
2	2	0	0	0	...
0	0	0	0	0	...
..	...	..	..	..	..

Rather than using the number of occurrences in the computation of $D(i, j)$, we suggest using the *fraction* of cases Fr_{ij} that have a certain co-occurrence of attribute-value pairs. With these fractions, one can compute the entropy for the contingency table:

$$E(i, j) = - \sum_{l=1}^L \sum_{m=1}^M Fr_{ij}(l, m) * \ln Fr_{ij}(l, m), \quad (8)$$

with $F_{ij}(l, m) = f_{ij}(l, m)/N$, in which N is the number of cases in a cluster.

$E(i, j)$ ranges from 0, where all cases are located in one element, to $\ln T$, where all cases are evenly distributed over all elements in the table, with T being the number of entries in a contingency table ($T = L * M$).

By normalizing $E(i, j)$ with respect to $\ln T$ and then subtracting the result from 1, we obtain a *new* formula for the D-measure:

$$\delta(i, j) = 1 - \frac{-\sum_{l=1}^L \sum_{m=1}^M F_{ij}(l, m) * \ln F_{ij}(l, m)}{\ln T}, \quad (9)$$

or, in terms of occurrences,

$$\delta(i, j) = 1 - \frac{\ln N - \frac{\sum_{l=1}^L \sum_{m=1}^M f_{ij}(l, m) * \ln f_{ij}(l, m)}{N}}{\ln T}. \quad (10)$$

Note that we introduce new symbols whenever we propose an alternative for the measures of the original WITT algorithm to make the distinction between the original measures and our proposals clear.

Applying (10) to the various tables given in this article, we observe that $\delta(i, j)$ for table 7 is now larger than $\delta(i, j)$ for table 5. Additionally, $\delta(i, j)$ for table 6 has become 0 for all values of n . This behavior is intuitively more satisfactory than that of Hanson and Bauer's $D(i, j)$ according to (3).

4. Analysis of the within-cluster cohesion

A second point of concern is the computation of O_c . When we apply (3) and (2), taking into account our conclusion from the previous section that $D(i, j) \rightarrow 1$ when $N \rightarrow \infty$, we observe that the denominator of B_{ck} in (5) tends towards zero when $N \rightarrow \infty$. Hence, $B_{ck} \rightarrow \infty$. This implies that *only by virtue of* the fact that the number of cases increases, $C_c \rightarrow 0$. This is obviously an undesirable behavior in the limit.

In addition, B_{ck} takes on a *negative* value in certain circumstances. Suppose, e.g., we have two clusters—A and B—based on two attributes with the contingency tables 8 and 9.

Table 8. The contingency table for cluster A.

2	1
1	0

Table 9. The contingency table for cluster B.

0	1
1	2

According to (3), $D(i, j) = 2\ln 2 / 4\ln 4 = 0.25$ for both tables. When we combine both clusters, the contingency table becomes table 10.

Table 10. The contingency table for the union of clusters A and B.

2	2
2	2

For this table $D(i, j) = 4 * 2 \ln 2 / 8 \ln 8 = 0.33$. Since there are only two attributes, $W_c = D(i, j)$, and thus the denominator of B_{ck} becomes $0.25 + 0.25 - 2 * 0.33 = -0.16$. We have observed that B_{ck} can also become negative when there are more than two attributes.

Although our proposed metric $\delta(i, j)$ partly overcomes this problem, (5) and its interpretation given by Hanson and Bauer for B_{ck} is still problematic. B_{ck} is described by Hanson and Bauer as *the variance in the attribute-value co-occurrences in the union of the categories, relative to the variance in the isolated categories*. Based on this definition, one might expect B_{ck} to equal 1 when forming the union of two identical clusters. However, because the denominator of B_{ck} is equal to 0, B_{ck} actually goes to ∞ . To overcome this problem, we propose a new approach for the calculation of *the within-cluster cohesion* O_c .

Let us first analyze the desired behavior of O_c , and thus of B_{ck} . When we try to assign a case to one of a set of clusters, we prefer to assign that case to the cluster with the highest within-cluster cohesion. However, when this cluster is similar to other clusters, one is likely to prefer a cluster that may have a slightly lower within-cluster cohesion. To achieve this behavior of C_c , O_c should be 1 when a cluster is completely different from all other clusters, while it should be larger than 1 when the cluster is not completely different from the other clusters. Since O_c is the average of a number of B_{ck} 's, B_{ck} should have the same behavior.

B_{ck} , as introduced by Hanson and Bauer, certainly does not have this desired property. We propose an alternative for B_{ck} , denoted as β_{ck} , based on the notion of the within-cluster cohesion of the union of two clusters, which are *assumed* to be disjunct. The union of two such clusters will be denoted as $W_{c \cup k}^*$.

We define β_{ck} as

$$\beta_{ck} = \frac{W_{c \cup k}}{W_{c \cup k}^*}. \quad (11)$$

According to (11), β_{ck} represents the relative within-cluster cohesion of the actual union of the clusters relative to the within-cohesion of the union of two, presumably disjunct, clusters with the same within-cluster cohesion. This approach is based on what *actually* happens when the union is taken relative to what can *maximally be achieved* by taking the union of presumably disjunct clusters. Therefore, (11) describes the *relative* cohesion between clusters c and k much better than B_{ck} as defined by (5).

In the following we will first derive an explicit expression for $W_{c \cup k}^*$ (i.e., enforced disjunctness) in terms of W_c and W_k for a special case. Later we will describe a procedure of how to compute $W_{c \cup k}^*$ in the general case.

Taking our proposed $\delta(i, j)$ according to (10), consider the term

$$\sum_{l=1}^L \sum_{m=1}^M f_{ij}(l, m) \ln f_{ij}(l, m). \quad (12)$$

For a cluster c we use the following simplified notation:

$$\Sigma x_c \ln x_c. \quad (13)$$

It is easily shown that for the union of two clusters c and k ,

$$x_{c+k} \ln x_{c+k} \geq x_c \ln x_c + x_k \ln x_k. \quad (14)$$

The equality holds only when x_c and/or x_k equals 0, which, in fact, is only the case for completely disjunct clusters. Assuming this to be the case for all contingency tables, we can derive an expression for $W_{c \cup k}^*$ in terms of W_c and W_k .

By dropping some indexes in our notation, we write (10) as

$$\delta_c = 1 - \frac{\ln N_c - \frac{\Sigma x_c \ln x_c}{N_c}}{\ln T}, \quad (15)$$

and similarly for δ_k ,

$$\delta_k = 1 - \frac{\ln N_k - \frac{\Sigma x_k \ln x_k}{N_k}}{\ln T}, \quad (16)$$

For the union of the clusters we can write

$$\delta_{c \cup k} = 1 - \frac{\ln(N_c + N_k) - \frac{\Sigma x_{c+k} \ln x_{c+k}}{N_c + N_k}}{\ln T}, \quad (17)$$

From (15) follows

$$\Sigma x_c \ln x_c = [\ln N_c - (1 - \delta_c) \ln T] N_c. \quad (18)$$

A similar expression holds for $\Sigma x_k \ln x_k$.

By combining (17) and (18) with (14), under the assumption that the equality holds, one obtains (see appendix A for derivation):

$$\delta_{c \cup k}^* = \frac{N_c \delta_c + N_k \delta_k}{N_c + N_k} - \frac{(N_c + N_k) \ln(N_c + N_k) - N_c \ln N_c - N_k \ln N_k}{(N_c + N_k) \ln T}. \quad (19)$$

Averaging over the various tables gives

$$W_{c \cup k}^* = \frac{N_c W_c + N_k W_k}{N_c + N_k} - \frac{(N_c + N_k) \ln(N_c + N_k) - N_c \ln N_c - N_k \ln N_k}{(N_c + N_k)} * \frac{\sum \frac{1}{N_T}}{N_T}, \quad (20)$$

in which N_T is the total number of contingency tables.

Equation (20) can only be used when for each attribute pair the *sum* of the number of non-zero elements in the contingency tables of the two clusters is less than the number of elements in each individual table. When this is not the case, $W_{c \cup k}^*$ may well become less than the minimum value of $W_{c \cup k}$, as is easily seen from the example given in the beginning of this section.

Table 11. The contingency table for cluster A.

2	1
1	0

Table 12. The contingency table for cluster B.

0	1
1	2

For both tables 11 and 12, $\delta(i, j) = 0.25$ and $N = 4$. Also $T = 4$; hence,

$$\delta_{c \cup k}^* = 0.25 - \frac{8 \ln 8 - 4 \ln 4 - 4 \ln 4}{8 \ln 4} = 0.25 - 0.5 = -0.25. \quad (21)$$

However, this particular problem can be overcome by taking a *constructive* approach in which contingency-like tables are created for the union of the two clusters. In this approach the contingency-like tables are constructed from the contingency tables of the clusters, but the position of the non-zero elements is lost. The union of two clusters results in a minimal value of $\delta_{c \cup k}^*$ when as many as possible non-zero elements of the contingency table of a cluster can be matched with the zero elements of the contingency table of the other cluster. In case the sum of the number of non-zero elements in both clusters is greater than the number of elements in the table, the values of some elements of the contingency tables of the clusters have to be summed up to find a value for the union of the clusters. Since we assume for the calculation of $\delta_{c \cup k}^*$ that the clusters are maximally disjunct, we only need to compute the sum of the smallest non-zero elements.

Our algorithm is based on the following three requirements.

1. The number of cases should be preserved. This means that the sum of the elements of a contingency table for the union of two clusters has to be equal to the sum of the number of cases in the two clusters.

2. Cases that appear in two different elements of a table in a cluster should also appear in two different elements of the table of the union of the cluster. In addition, cases that appear in the same element of a table should also occur in one single element of the table for the union of the clusters.
3. The algorithm should result in a minimum value for $\delta_{c \cup k}^*$.

Rather than working with tables, the construction algorithm is expressed in terms of lists. Our proposal goes along the following lines (we will illustrate our approach with an example—printed in *italics*—using tables 13 and 14):

Table 13. One contingency table used to demonstrate our constructive approach.

20	7
3	0

Table 14. The second contingency table to demonstrate our constructive approach.

0	3
1	20

1. One constructs two lists of the non-zero elements of the contingency tables of the two clusters.
Taking tables 13 and 14 given above as an example, the two lists become $\{20\ 7\ 3\}$ and $\{3\ 1\ 20\}$.
2. When the length of the concatenation of these lists is less than the number of elements in the table, $\delta_{c \cup k}^*$ is computed according to (10) by taking the elements of the concatenated list as the frequencies $f_{ij}(l, m)$.
3. In case the length of the concatenated list is greater than the number of entries in the tables, a new list has to be constructed that has a length precisely equal to the number of elements in the contingency table. In order to create that new list, the two lists mentioned under point 1 are sorted in descending order.
In our example, the length of the combined list is 6, which is greater than 4, the size of the table. After sorting, the lists become $\{20\ 7\ 3\}$ and $\{20\ 3\ 1\}$ respectively.
4. In order to construct the new list, the two lists are each divided into two parts—a head and a tail, the tails having equal length—such that the sum of the length of the heads plus the length of a tail equals precisely the number of elements in the contingency table.
Applying this step to our example, we get as a result $\{20\ | \ 7\ 3\}$ and $\{20\ | \ 3\ 1\}$, respectively.
5. The heads of the two lists are removed, and part of the new list is made by the concatenation of these heads.
In our example, the initial part of the new list becomes $\{20\ 20\}$, while the old lists reduce to $\{7\ 3\}$ and $\{3\ 1\}$, respectively.
6. One of the remainders of the stored lists is reversed. Then the new list is extended with elements that are the sum of the elements of the remainders of the two lists. This entails the summation of the larger elements of one of the remaining lists with the smaller elements of the other remaining list.

Assuming that the first list is reversed, we get $\{3\ 7\}$ and $\{3\ 1\}$. From these lists we get the elements 6 and 8, respectively, that should be added to the new list. This results in a list $\{20\ 20\ 6\ 8\}$. Note that reversing one list is necessary to minimize $\delta_{c \cup k}^*$. Without reversing, the resulting list would be $\{20\ 20\ 10\ 4\}$, which would result in a higher value of $\delta_{c \cup k}^*$.

7. Finally, $\delta_{c \cup k}^*$ is computed from the new list.

This constructive approach allows us to compute the value of the within-cluster cohesion of the union of the clusters assuming that they are disjunct.

At this point there is only one drawback in the definition of β_{ck} according to (11), namely, the fact that its value is limited. If we maximize β_{ck} by assuming that the clusters are identical and that both clusters contain the same number of objects while the size of the tables is 4, we can derive from (20) that

$$\beta_{ck_{\max}} = \frac{W}{W - \frac{1}{2}}. \quad (22)$$

This approach leads, however, to values for β_{ck} that are too large, since the calculation of the denominator of (11) using (20) assumes a *disjunct* combination of the clusters. For $2 * 2$ tables this will not necessarily be the case, and therefore the denominator will be larger than $W^{-1/2}$. As a matter of fact, the denominator can only become zero when the combination of each pair of contingency tables results in an even distribution of cases over the number of entries of the tables, since only in that case is W zero. In many practical situations, the denominator will be closer to W . The reason is the *uneven* distribution of cases over the cells of the contingency-like tables for the union of the two clusters.

Finally, a different solution to the problem of the calculation of O_c can be obtained by giving a different interpretation for β_{ck} . First, we introduce the within-cluster cohesion that is maximally achievable when two clusters are merged. We can again use a constructive approach by sorting the lists of the entries in the corresponding tables in descending order. A new list is constructed from the sums of corresponding elements of the sorted lists. In the event that the lists are of unequal length, the shortest one is extended with zeros. For tables 13 and 14, the lists become $\{20\ 7\ 3\}$ and $\{20\ 3\ 1\}$. The combined list is $\{40\ 10\ 4\}$ which serves as input for the computation of the within-cluster cohesion.

Now β_{ck} can be defined as the difference in within-cluster cohesion for the maximally similar and the *disjunct* union of the clusters, computed using the constructive approach, divided by the difference in within-cluster cohesion for the maximally similar and the *actual* union of the clusters:

$$\beta_{ck} = \frac{W_{c \cup k_{\max}}^* - W_{c \cup k_{\min}}^*}{W_{c \cup k_{\max}}^* - W_{c \cup k}}. \quad (23)$$

So when the clusters are in actuality maximally disjunct, the numerator and denominator of (23) are equal and β_{ck} becomes 1. When the same elements of the contingency tables

have non-zero values and when, after sorting the lists, these elements occur at the same position in the sorted lists, $\beta_{ck} \rightarrow \infty$. Thus, this approach is also intuitively plausible in this respect: given (4) and using β_{ck} instead of B_{ck} , O_c will vary between 1 and ∞ .

5. Analysis of the evaluation metric C_c

So far we have dealt with the measures from which the cohesion C_c of the cluster is computed. In the refinement algorithm, a threshold value for C_c is used to decide whether a new case belongs to one of the existing clusters. This means, that a new case is tentatively added to each of the clusters and then the cluster with the maximum value of C_c is selected. The new case is added permanently to the cluster *if and only if* this maximum C_c exceeds the threshold. When a case is added that is completely different from a cluster, the value of W_{c+1} follows from (20) by taking $W_k = 1$ and $N_k = 1$:

$$W_{c+1_{\min}} = \frac{N_c W_c + 1}{N_c + 1} - \frac{(N_c + 1) \ln(N_c + 1) - N_c \ln N_c}{(N_c + 1)} * \frac{\sum \frac{1}{\ln T}}{N_T}. \quad (24)$$

The first term of the right-hand side of (24) $\rightarrow W_c$ and the second term $\rightarrow 0$ if $N_c \rightarrow \infty$. This implies that when a sufficiently well-defined cluster of considerable size exists, there is a high probability that all new cases will be assigned to that cluster. Hence the resulting clusters may not properly describe the underlying relations between the attribute values of the analyzed cases. To avoid this situation, we suggest using another numerator in (1) that is sensitive to the extent to which the value of W_c can decrease. Here again, a *normalized* difference in within-cluster cohesion is proposed. Rather than taking W_{c+1} , we suggest using the difference between the within-cluster cohesion of a cluster with *one case added* and the corresponding minimally achievable within-cluster cohesion divided by the difference between the within-cluster cohesion of the cluster *without the case* and the corresponding minimally achievable within-cluster cohesion:

$$\gamma_c = \frac{\frac{W_{c+1} - W_{c+1_{\min}}}{W_c - W_{c+1_{\min}}}}{O_c}. \quad (25)$$

γ_c is independent of the value of W_c . The numerator describes the relative reduction in W_c , while the denominator describes the cohesion of the cluster under study with respect to the other clusters.

Note that when a case is added that is *prototypical* for a cluster, the numerator of γ_c will be greater than 1. Hence, depending upon the denominator, γ_c may become larger than 1. This is also plausible, since adding a prototypical case should increase the cohesion of the cluster.

6. Summary and conclusions

We have made an analysis of the various measures that underlie the WITT clustering algorithm as proposed by Hanson and Bauer. We have shown that most of these measures have very undesirable characteristics. We have presented alternatives for these measures to overcome all identified problems. The clustering procedure becomes *independent of the size* of the clusters by the incorporation of the suggested modifications. The cohesion measure now takes into account the *size of the contingency tables* from which it is derived. Furthermore, the thresholds used in the algorithm as described in the paper of Hanson and Bauer can be set *independent of the cohesion* of the clusters.

Our alternatives are still based on the *conceptual* ideas behind the original clustering algorithm while overcoming the distinct problems identified above.

Appendix A

Below we give the derivation of (19) of this article.

The derivation uses the following formulas:

$$x_{c+k} \ln x_{c+k} \geq x_c \ln x_c + x_k \ln x_k \quad (14)$$

$$\delta_{c \cup k} = 1 - \frac{\ln(N_c + N_k) - \frac{\Sigma x_{c+k} \ln x_{c+k}}{N_c + N_k}}{\ln T} \quad (17)$$

$$\Sigma x_c \ln x_c = [\ln N_c - (1 - \delta_c) \ln T] N_c. \quad (18)$$

Using (14) with the equal sign in (17) yields

$$\delta_{c \cup k} = 1 - \frac{\ln(N_c + N_k) - \frac{\Sigma x_c \ln x_c + \Sigma x_k \ln x_k}{N_c + N_k}}{\ln T}. \quad (A1)$$

Substituting (18) in (A1) gives

$$\delta_{c \cup k} = 1 - \frac{\ln(N_c + N_k) - \frac{[\ln N_c - (1 - \delta_c) \ln T] N_c + [\ln N_k - (1 - \delta_k) \ln T] N_k}{N_c + N_k}}{\ln T}. \quad (A2)$$

Rearranging terms gives

$$\delta_{c \cup k} = 1 - \frac{(N_c + N_k) \ln(N_c + N_k) - N_c \ln N_c - N_k \ln N_k + [(1 - \delta_c) N_c + (1 - \delta_k) N_k] \ln T}{(N_c + N_k) \ln T}, \quad (A3)$$

which further simplifies to

$$\delta_{c \cup k} = 1 - \frac{(N_c + N_k) \ln(N_c + N_k) - N_c \ln N_c - N_k \ln N_k}{(N_c + N_k) \ln T} - \frac{(1 - \delta_c)N_c + (1 - \delta_k)N_k}{(N_c + N_k)}. \quad (\text{A4})$$

From (A4) equation (19) follows directly.

Acknowledgment

The authors would like to thank the editor and the reviewers for their constructive comments. The research reported was performed in the framework of the KAVAS project (Knowledge Acquisition, Visualization and Assessment study), which was partially funded by the Commission of the European Communities under contract A1021 in the Exploratory AIM programme.

References

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

Received June 17, 1991

Accepted September 18, 1992

Final Manuscript September 29, 1992