Hierarchical Clustering of Massive, High Dimensional Data Sets by Exploiting Ultrametric Embedding

Fionn Murtagh (1), Geoff Downs (2), and Pedro Contreras (3)

(1) Department of Computer Science, Royal Holloway, University of London
Egham TW20 0EX, UK f.murtagh@acm.org

(2) Digital Chemistry Ltd., The Iron Shed, Harewood House Estate
Harewood, Leeds LS17 9LF, UK geoff.downs@digitalchemistry.co.uk

(3) Department of Computer Science, Royal Holloway, University of London
Egham TW20 0EX, UK pedro@cs.rhul.ac.uk

January 22, 2007

Running title: Massive, High Dimensional Data Clustering

Abstract

Coding of data, usually upstream of data analysis, has crucial implications for the data analysis results. By modifying the data coding – through use of less than full precision in data values – we can aid appreciably the effectiveness and efficiency of the hierarchical clustering. In our first application, this is used to lessen the quantity of data to be hierarchically clustered. The approach is a hybrid one, based on hashing and on the Ward minimum variance agglomerative criterion. In our second application, we derive a hierarchical clustering from relationships between sets of observations, rather than the traditional use of relationships between the observations themselves. This second application uses embedding in a Baire space, or longest common prefix ultrametric space. We compare this second approach, which is of $O(n \log n)$ complexity, to k-means.

Key words. Hierarchical clustering, ultrametric, tree distance, partitioning, hashing

AMS subject classifiers. 98.52.Cf, 89.75.Hc, 89.75.Fb
1 Introduction

1.1 Chemical Clustering and Matching

Clustering is a requirement for dataset matching, and to support fast proximity searching. For the processing of chemical structure databases, see [3, 6, 7, 30, 9, 14]. Both locally (e.g., for neighborhood search) and globally (for data summarization), hierarchical clustering is beneficial.

In the 1990s, the Ward minimum variance hierarchical clustering method became the method of choice in the chemoinformatics community due to its hierarchical nature and the quality of the clusters produced. Unfortunately the method reached its limits once the pharmaceutical companies tried processing datasets of more than 500,000 compounds due to: the $O(n^2)$ processing requirements of the reciprocal nearest neighbor algorithm; the requirement to hold all chemical structure “fingerprints” in memory to enable random access; and the requirement that parallel implementation use a shared-memory architecture. In this article we develop and study alternative hierarchical agglomerative clustering algorithms that bypass these difficulties. Our first innovative algorithm is a hybrid one, incorporating the Ward agglomerative criterion; and our second algorithm adopts a different target in regard to the ultrametric (or tree distance based) output.

The structure of this article is as follows. After having noted how crucially important data coding is, for facilitating the finding of local ultrametric relationships in one’s data, we then explore data coding that is based on (real-valued, or floating point) data precision. We find that this can make our data ultrametric to a greater or lesser extent, thereby bringing us closer or more remote from our objective.

Levin [21] points to “subtle” differences between the metric viewpoint and the topological viewpoint, respectively, based on closest point versus all points which are not closest. In this article, we will try to elucidate Levin’s remark, by focusing on ultrametric topology. We explore two different directions for this topologically inspired perspective on the data analysis. In section 4 we use it to decrease the quantity of data that we will handle using a traditional hierarchical clustering approach. In section 6, we explore the benefits of inducing an ultrametric through embedding our data in a Baire space.

1.2 Background to Algorithms

Massive and high dimensional data spaces often have hidden regularity. An important, practical form of regularity is hierarchical regularity. If we can exploit regularity in massive, high dimensional data sets, then we have one approach to addressing the computational and other performance problems raised by best match (or nearest neighbor) search, and related matching problems. In this work we discuss such a solution for data sets of large numbers of objects (tens of thousands), in high dimensional spaces.

The next two subsections consider, firstly, the mapping of metric (or other)
data into an ultrametric, or embedding a metric in an ultrametric, which leads to
the study of the distortion involved in this. Secondly, we consider the traditional
multivariate data analysis approach of fitting a hierarchy to data, i.e., fitting a
model, which leads to questions of optimization. As opposed to such work, we
focus on recoding the data, principally through modifying the data precision.
By a focus, in this way, on the data measurement process, we can find a new,
general approach to the discovery of (hierarchical) structure in data.

1.3 Ultrametric Embedding: Mapping a Metric into Ultrametric

In [1], it is discussed how “large systems necessarily contain large, highly struc-
tured subsystems”. Among solutions investigated in such work (see [2, 12]),
there is the mapping of a point set, in a high dimensional space, into a hier-
archy or tree, with known bound on the distortion of the output (ultrametric)
relative to input (metric) data, and with a known bound also on the computa-
tional requirement. Our work, presented in this article, does not use these
solutions because (i) the bounds are often not tight and they hold globally but
not locally, (ii) classes of metric data are studied, which are not sufficiently rel-
levant to our specific inputs, and (iii) algorithms are not well enough elucidated
for our purposes. Our biggest objection to these distortion-characterizing ap-
proaches is that data coding or recoding is not addressed, yet such data recoding
is crucial in practice in data analysis and handling.

We keep the same objective as that which is studied in the foregoing ref-
erences, i.e., we seek the “highly structured subsystems” that are contained in
larger, complex data collections. We take “hierarchical substructures” as defined
by being endowed with an ultrametric. We look at the precise characterization
of the data in terms of ultrametricity. Then we study efficient and effective
algorithms for finding an ultrametric embedding of the data.

1.4 Fitting a Hierarchical Structure

Our objective is not the same as fitting a hierarchical structure, as is tradition-
ally used in multivariate data analysis.

A mainstream approach over at least 4 decades of data analysis has been
to fit a tree structure well to a data set, with quality of fit presupposing a
clustering (mostly agglomerative, but possibly divisive). Instead we seek in-
herent ultrametricity in a data set, and so the more ultrametricity we find in
our data the better: we open the door to very much easier (and possibly more
computationally efficient) processing.

Both traditional agglomerative (and some divisive) algorithms and our new
approach can be considered as mapping our data into an ultrametric space.
In the traditional approach, for example, we can consider an agglomerative al-
gorithm as a mapping of all pairwise distances (or indeed dissimilarities) into
ultrametric distances [29]. As an example, the single link hierarchical agglom-
erative clustering criterion furnishes the subdominant ultrametric: for any pair
of points, the ultrametric distance will always be less than or equal to an input distance. We too, in our new approach, seek an ultrametric embedding, or ultrametrization [31], in that we look for subsets of the data that ab initio are ultrametric. There are two major practical implications of this. Firstly, we bypass the need for a clustering criterion because natural or inherent ultrametricity leads to an identical result with most commonly used agglomerative criteria (for example, as specified by the Lance-Williams formula: see [26]). Secondly, how we code our data becomes very central: coding our data in the most propitious way can help greatly with how inherently ultrametric our data is.

2 Data

2.1 Notation Used, and Data Normalization

We will use the notation $x$ for the data matrix to be analyzed, and $x_i$ denotes any particular row. A chemical structure (or chemical) $i$ is represented by a row, and the set of chemical structures, or rows, is denoted $I$. We work with just over 1.2 million chemicals, $i \in I$. Similarly the column codes or attributes, 1052 in number, are denoted by set $J$. Needless to say, our chemicals $\times$ codes view of the data, used here for convenience of exposition, is fully compatible with a more appropriate form of storage.

We will take the notation a little further (as in [29]) by writing $x_{IJ}$ for the given data, and a row of this matrix is denoted $x_{iJ}$ (so we are indicating row $i$ and the column set, $J$). The sum of the columns gives the vector (marginal) $x_J$. We normalize the data by dividing each matrix value by its column sum, and the resulting normalized matrix is denoted $x_{IJ}^{\star}$. Here we are saying: the presence of a code $j$ in chemical $i$ must take into account whether that code is rare, implying importance of the presence property; or common, implying a lower value of presence. Given our notation, a tensor product allows us to reconstruct our original data: $x_{iJ}^{\star} \otimes x_J = x_{IJ}$. Normalization can be very important, to homogenize the effects of the coding identifiers (set $J$) that are used: see Figure 1.

2.2 Distributional Properties of the Data Used

We use a set of 1,219,553 chemical structures coded through 1052 presence/absence values, using the Digital Chemistry bci1052 dictionary of fragments [36]. Our experimental work is based on a matrix of binary-valued vectors: in some instances it would be more efficient to work directly on the small set of code offsets rather than a 1052-vector. The binary-valued matrix is sparse: occupancy is 8.6347%.

A power law (see [25]) is a distribution (e.g. of frequency of occurrence) containing the general form $x^{-\alpha}$ where constant $\alpha > 0$; and an exponential law is of the form $e^{-x}$. For a power law, $P(x > x_0) \sim cx^{-\alpha}$, $c, \alpha > 0$. A power law has heavier tails than an exponential distribution. In practice $0 \leq \alpha \leq 2$. For
Figure 1: Histogram of column sums, denoted $x_J$ in the text.
such values, $x$ has infinite (i.e. arbitrarily large) variance; and if $\alpha \leq 1$ then the mean of $x$ is infinite. The density function of a power law is $f(x) = ace^{-\alpha x}$, and so $\ln f(x) = -\alpha \ln x + C$, where $C$ is a constant offset. Hence a log-log plot shows a power law as linear. Power laws have been of great importance for modeling networks and other complex data sets.

Figure 2 shows a log-log plot based on the 1052 presence/absence attributes, using all 1.2 million chemicals. In a very similar way to the power law properties of large networks (or file sizes, etc.) we find an approximately linear regime, ending (at the lower right) in a large fan-out region. The slope of the linear region characterizes the power law. For this data, we find that the probability of having more than $n$ chemicals per attribute to be approximately $c/n^{1.25}$ for large $n$.

The histogram of attributes per chemical, on the other hand, does not show a pronounced power law; see Figure 3. In fact, it it close to a Gaussian.

In summary, we find the attributes per chemical (analysis of $x_I$ to use our terminology) to be approximately Gaussian distributed; whereas we find the chemicals per attribute (analysis of $x_J$) to be approximately power law distributed.

3 Measuring Hierarchical Substructure

3.1 Quantifying Degree of Ultrametricity

Among various properties of an ultrametric topology (see e.g. [20]) we focus on the triangular inequality, a relationship holding for all triplets of points, in order to quantify extent of ultrametricity, or inherent hierarchical structure. The triangular inequality holds for a metric space: $d(x, z) \leq d(x, y) + d(y, z)$ for any triplet of points $x, y, z$. In addition the properties of symmetry and positive definiteness are respected. The “strong triangular inequality” or ultrametric inequality is: $d(x, z) \leq \max \{d(x, y), d(y, z)\}$ for any triplet $x, y, z$. An ultrametric space implies respect for a range of stringent properties. For example, the triangle formed by any triplet is necessarily isosceles, with the two large sides equal; or is equilateral.

Our measure of extent of ultrametricity, introduced in [27], can be described algorithmically. We assume a Euclidean metric. We examine triplets of points (exhaustively if possible, but in practice with large data sets through sampling), and determine the three angles formed by the associated triangle. We select the smallest angle formed by the triplet points. Then we check if the other two remaining angles are approximately equal. If they are equal then our triangle is isosceles with small base, or equilateral (when all angles are equal). The approximation to equality is given by 2 degrees (0.0349 radians). Our motivation for the approximate (“fuzzy”) equality is that it makes our approach robust and independent of measurement precision.

Studies are discussed in Murtagh [27] showing how numbers of points in our clouds of data points are irrelevant; but what counts is the ambient spa-
Figure 2: Log-log plot of numbers of chemicals per attribute, based on the whole data set of 1.2M chemicals.
Figure 3: Histogram of presence/absence attributes for 3 different subsets of the chemicals.
tial dimensionality. Among cases looked at are statistically uniformly (hence “unclustered”, or without structure in a certain sense) distributed points, and statistically uniformly distributed hypercube points (so the latter are random 0/1 valued vectors). Using our ultrametricity measure, there is a clear tendency to ultrametricity as the spatial dimensionality (hence spatial sparseness) increases.

A note on the use of Euclidean distances follows. A complete, normed vector space, which additionally has a scalar product associated with the norm, defines a Hilbert space. Scalar product gives us vector projection, and subject to normalization of the vectors it furnishes the angle between the two vectors. Hence our assumption that, by working in a Hilbert space, we have a convenient and practical environment. With finiteness, we have the Euclidean environment. In other non-Hilbert spaces, such as the $L_1$ Minkowski space, also referred to as the space endowed with the cityblock, Manhattan, or taxicab metric, then the notion of triangle and angle must be defined. For the $L_1$ space, a discussion and proposal can be found in [33]. In our experience there is little, if anything, to be gained in departing from the Euclidean or (allowing infinite dimensionality) Hilbert space context.

3.2 Increasing Ultrametricity Through Data Recoding

Data recoding plays an important role in the correspondence analysis tradition [29]. Data may be doubled meaning that each data value, and its difference from a maximum value (for an attribute) are used. The row and column marginal distributions, which furnish row and column, respectively, mass distributions, are of course altered if we recode an input data array in this way.

More generally, booleanizing, or making qualitative, data in this way, for a varying (value-dependent) number of target value categories (or modalities) leads to the form of coding known as complete disjunctive form.

Such coding increases the embedding dimension of the data to be analyzed, and data sparseness, and thus may encourage extent of ultrametricity. That it can do more we will now show.

The iris botanical data has been very widely used as a toy data set since Fisher used it in 1936 ([13], taking the data from a 1935 article by Anderson) to exemplify discriminant analysis. It consists of 150 iris flowers, each characterized by 4 petal and sepal, width and breadth, measurements. On the one hand, therefore, we have the 150 irises in $\mathbb{R}^4$. Next, each variable value was recoded to be a rank (all ranks of a given variable considered) and the rank was boolean-coded (viz., for the top rank variable value, 1000, for the second rank variable value, 0100, etc.). As a result of equal ranks, the second data set embedded the 150 irises in $\mathbb{R}^{123}$ (rather than $\mathbb{R}^{150}$). Actually, this definition of the 150 irises is more accurately in the space $\{0,1\}^{123}$ rather than in $\mathbb{R}^{123}$.

Our triangle-based measure of the degree of ultrametricity in a data set (here the set of irises), with $0 =$ no ultrametricity, and $1 =$ every triangle an ultrametric-respecting one, gave the following: for irises in $\mathbb{R}^4$, 0.017; and for irises in $\{0,1\}^{123}$: 0.948.
This provides a nice illustration of how recoding can dramatically change the picture provided by one’s data. Furthermore it provides justification for data recoding if the ultrametricity can be instrumentalized by us in some way.

### 3.3 Recoding Distances between Chemicals Increases Ultrametricity

Given the $m$-dimensional encoding of chemical structures, it is known that ties in proximities necessarily result [24]. We have therefore that $d(i, i') = 0$ does not imply that $x_i = x_{i'}$. (This runs counter to most axiomatizations of distance. We can simply say that within the confines of the encoding or characterization scheme used, chemical structures $i$ and $i'$ are identical.) Hence a preliminary stage of finding clusters furnished by 0-valued distances is sensible. Next, we proceed somewhat further in this direction, by forcing small-valued distances to be 0 in a particular way.

Our procedure is as follows: (i) normalize the chemicals data by dividing each 0/1 chemical/attribute value by the column (or attribute) total; (ii) determine the Euclidean distance (remark: not the distance squared) between the vectors of any two chemicals, normalized in this way; and (iii) recode this distance to a limited precision. The last step is implemented by multiplying by a fixed constant, e.g. 1000, and then taking the integer part of the value. The latter, recoded, distances were assessed for their ultrametricity.

Typical sets of results are shown in Table 1. The first set of results is based on 1000 chemical structures at the start of the dataset, and this is followed by 20,000 chemical structures also from the start of the dataset used. The third set of results are on a set of 20,000 chemical structures from the 10,000th chemical structure onwards. In these results, we used 2000 samplings of triplets, but we looked at other numbers of samplings also (e.g., 1000, 4000). These results are also based on one random number seed used for the random samplings but various other seeds were used, and gave very similar results.

For the first batch of (1000 chemical structures) assessments, clearly the ultrametricity degrades with precision. For the second (20,000) chemical structures assessments, we find that the precision = 1 case is very trivial. The values in the input matrix (viz., $x_{ij}$) are smaller because the column sums are so much smaller compared to the 1000 chemical structures case. What we have in the precision = 1 case is that the digit 0, only, is nearly always used for the distances. By taking a bigger precision, this triviality disappears.

Our conclusions are as follows. For the chemical data, we find that ultrametricity is small, – at full precision. But if we allow for distances between chemical structures (Euclidean distances between “profiles”, i.e. where value of the $i$th chemical structure, on the $j$th code identifier, is $x_{ij} / x_j$, $x_j$ being the column sum) to be of less than full precision, then we can find far higher ultrametricity. We have confirmed the consistency of this result for the various system parameters used (such as the sampling of triplets, or the sampling of chemical structures, on which this conclusion was based).
<table>
<thead>
<tr>
<th>No. points</th>
<th>Dim. (Eff.)</th>
<th>Precision</th>
<th>Non-degen.</th>
<th>Equil.</th>
<th>Ultrametricity</th>
</tr>
</thead>
<tbody>
<tr>
<td>1000</td>
<td>1052 (886)</td>
<td>1</td>
<td>1776</td>
<td>176</td>
<td>0.653716</td>
</tr>
<tr>
<td>1000</td>
<td>1052 (886)</td>
<td>2</td>
<td>1990</td>
<td>2</td>
<td>0.162312</td>
</tr>
<tr>
<td>1000</td>
<td>1052 (886)</td>
<td>3</td>
<td>1991</td>
<td>1</td>
<td>0.102963</td>
</tr>
<tr>
<td>1000</td>
<td>1052 (886)</td>
<td>4</td>
<td>1991</td>
<td>1</td>
<td>0.097438</td>
</tr>
<tr>
<td>1000</td>
<td>1052 (886)</td>
<td>5</td>
<td>1991</td>
<td>1</td>
<td>0.097941</td>
</tr>
<tr>
<td>1000</td>
<td>1052 (886)</td>
<td>6</td>
<td>1991</td>
<td>1</td>
<td>0.097941</td>
</tr>
<tr>
<td>20000</td>
<td>1052 (1000)</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1.0</td>
</tr>
<tr>
<td>20000</td>
<td>1052 (1000)</td>
<td>2</td>
<td>1109</td>
<td>235</td>
<td>0.710550</td>
</tr>
<tr>
<td>20000</td>
<td>1052 (1000)</td>
<td>3</td>
<td>2000</td>
<td>14</td>
<td>0.270000</td>
</tr>
<tr>
<td>20000</td>
<td>1052 (1000)</td>
<td>4</td>
<td>2000</td>
<td>1</td>
<td>0.097500</td>
</tr>
<tr>
<td>20000</td>
<td>1052 (1000)</td>
<td>5</td>
<td>2000</td>
<td>1</td>
<td>0.099000</td>
</tr>
<tr>
<td>20000</td>
<td>1052 (1000)</td>
<td>6</td>
<td>2000</td>
<td>1</td>
<td>0.099500</td>
</tr>
<tr>
<td>20000</td>
<td>1052 (1007)</td>
<td>1</td>
<td>2</td>
<td>0</td>
<td>1.0</td>
</tr>
<tr>
<td>20000</td>
<td>1052 (1007)</td>
<td>2</td>
<td>1062</td>
<td>238</td>
<td>0.734463</td>
</tr>
<tr>
<td>20000</td>
<td>1052 (1007)</td>
<td>3</td>
<td>1999</td>
<td>6</td>
<td>0.277639</td>
</tr>
<tr>
<td>20000</td>
<td>1052 (1007)</td>
<td>4</td>
<td>2000</td>
<td>2</td>
<td>0.087000</td>
</tr>
</tbody>
</table>

Table 1: “No. of points” are the numbers of chemical structures taken on each occasion. “Dim.” is the dimensionality, or number of descriptive codes (as used in all of this work). Since a limited number of chemical structures was used on each occasion, the “Eff.” or effective dimensionality was smaller, i.e., some columns were all 0-valued. “Precision” is the number of significant digits retained. “Non-degen.” is the number of non-degenerate (i.e., degeneracy is when two or three points in the triplet overlap), out of the 2000 samplings of triplets used in each case; “Equil.” is the number of equilateral triangles found; “Ultrametricity” is 1 for complete ultrametricity.
For searching, both isosceles with small base, or (the very rare) equilateral,
cases of ultrametricity imply that data points are pulled apart. This can be of
help in expediting search.

This finding can be interpreted as a hashing of the distances into fixed bin
sizes. But hashing usually is applied to the original data, not the distances. So,
this finding justifies the interpretation of this procedure as a new data coding
procedure. Replacing each original distance, we map a pair of points onto a
precision-limited dissimilarity.

3.4 Remarks on Our Recoding Procedure

Firstly, a note on metric properties of this recoding scheme follow. Whether
or not the precision-based recoding of distances remains Euclidean is important
for the clustering application below. For, if the recoded distances remain dis-
tances, then by virtue of the triangular inequality, having recoded distance
d′(x, y) = 0 and d′(x, z) = 0 implies that d′(y, z) = 0. Unfortunately we
cannot guarantee this. Consider a situation where we have input distances
d(x, y) = 1.0, d(x, z) = 0.9, d(y, z) = 0.9. Then we have the triangular inequal-
ity for any triplet, e.g., d(x, y) ≤ d(x, z) + d(y, z) or 1.0 ≤ 0.9 + 0.9. With our
recoding we now ask if d′(x, y) ≤ d(x, z) + d(y, z)? For one digit precision re-
coding we have, respectively, 1.0 and 0.9, and the triangular inequality no longer
holds. In practice the situation may be very favorable for us.

Clearly, by construction, the distances remain Euclidean if the precision is
high enough. In the final batch of results in Table 1, this is the case for the
precision with 4 decimal digits (i.e., last row of the table). With 3 decimal digits
precision, we find 3998 triplets used in a 4000-samples set to be Euclidean, viz.,
practically all. With 2 decimal digits precision, we find 2102 triplets used in a
4000-samples set to be Euclidean, viz., about 50%. Finally, with 1 decimal
digit precision, which from Table 1 gives very degenerate results (most distances
become equal to 0), we find just 3 triplets out of a 4000-samples set to be
Euclidean. What this indicates is that in the 2 decimal digits case, about 50% of
the triplets change (actually, our tests indicate that at least one of the distances
in the triplet of distances is so affected) to dissimilarities and no longer respect
the triangular inequality. This is not a problem for us: among the distances,
and among triplets of distances, we find a sufficiently many, nonetheless, to be
ultrametric.

Zero distances are of greater interest to us, and we again checked the cases
considered in Table 1. For the 2000-sampling cases, and for the precision 1
cases, we found 98% and 99.95% of triplets respecting the triangle inequality.
We can claim that we can approximate the triangular inequality for recoded
distances very well (without guaranteeing this in all cases).

A note on possible use of this recoding follows. We checked the 2 decimal
digits result towards the end of Table 1 (3rd last row) using 4000 triplets, and
hence 12000 distances, and found just 31 unique dissimilarities or distances.
For the 4 decimal digits result towards the end of Table 1 (last row) using 4000
triplets, and hence 12000 distances, we found 4666 unique distances. The far
more discrete set of dissimilarities or distances, in the former case, may allow
greater ease when establishing bounds on nearest neighbor proximities. What
we have just sketched out will now be further studied for data clustering.

4 Data Condensation through Recoding

4.1 Implications of Increasing Ultrametricity

Data clustering is facilitated greatly through recoding, and this will be discussed
in this section.

Non-uniqueness of distances such as the Euclidean distance, or other dissim-
ilarities, in the sense of chemicals $i$ and $i'$ having identical dissimilarity as do
chemicals $i$ and $i''$, is a known issue with the data being studied [8]. We will
now go further and use an “aggressive” distance recoding as described in the
last section (section 3.4) to further map chemicals into clusters.

A few principles, some demonstrable and some heuristic, underly this pro-
cedure.

If we have a near-ultrametric set of distances (or, equally, dissimilarities)
then minimal and maximal distances from a cluster member to an outsider will
be identical. This is necessitated by respect for ultrametricity. (We can similarly
state that the single linkage and complete linkage agglomerative hierarchical
clustering algorithms will produce identical output.) To the extent that the
data is embedded completely in an ultrametric space, then most commonly
used hierarchical clustering agglomerative algorithms (see e.g. [26]) will give
an identical result. This lessens any need to pay attention to the agglomerative
criterion. Even with high ultrametricity, we approximate this “all agglomerative
algorithms the same” situation.

The results of Table 1 in regard to ultrametricity indicate that for crude
precision-based recoding, we are approximating global ultrametricity, and the
approximation that we have the situation of “all agglomerative algorithms the
same” is approximately tenable. For the 0-distance case, respect for the trian-
gular inequality would suffice. As discussed in section 3.4, we have an excellent
basis for assuming that this is generally the case in practice, although it cannot
be guaranteed.

By retaining distance values with limited, small precision, we are essentially
mapping distances onto clusters. Furthermore it may be of interest to pursue
a phase of data condensation, by finding clusters of chemicals that are mapped
onto the same point through our distance recoding. It is clear therefore that
many 0-valued distances leads to an exploitable situation, for data condensation,
and not a trivial one.

While there is pronounced ultrametricity in the data shown in Table 2, Fig-
ure 4 shows the implications. Perfect ultrametricity implies identity in agglom-
erative clustering based on the three agglomerative criteria used in Figure 4.
What we find is closeness but not identity of results, due to lack of full ultra-
metricity. So, chemicals 20 and 28 are agglomerated late; 30 and 19, and 21 and
Table 2: Sample of 30 chemicals: recoded distances shown. Recoding used: integer part of 100 * distance. A blank is used in the principal diagonal.
29, slightly earlier; and for all remaining chemicals, there is just the subclustering at a very small criterion value that is evident in the Ward (or minimum variance) and complete link agglomerative criteria.

4.2 Fast Processing of Near Neighbors

For each of \( n \) chemicals to seek a nearest neighbor among all other chemicals implies \( O(n^2) \) computational cost, and we seek a way to avoid this, even if we have to accept an approximate solution. We have already noted that the data we are handling has many tied distances, and we have also seen how recoding our data – more specifically, recoding our distances – leads to many additional 0 distances. Therefore we will give particular attention to the fast finding of 0 distances. Because of the recoding, this particular interest in 0 distances is quite general, and widely applicable. Note too that 0 distances are trivially ultrametric; and all commonly used agglomerative clustering criteria will behave the same: pairwise agglomerations will be carried out at the same criterion value.

A spanning path [19] has been used to facilitate finding near neighbors (and also locally dense regions of space). It incorporates all points (hence is “spanning”). A minimal total distance spanning path is a Hamiltonian path, and a heuristic solution to the traveling salesperson problem, which is well-known to be NP-complete. We experimented with many ways to define a spanning path which help subsequently with finding near neighbor, and in particular 0-valued near neighbor, relationships, including random projection onto one or more axes [35], and random attribute selection. We retained from this study the use of row sums, i.e., \( x_I \) in the notation introduced earlier, or the row marginal density distribution, in order to define the spanning path. The row density itself can be found in \( O(n) \) time for \( n \) rows. Finding the order of \( n \) row density values requires a sort, and therefore \( O(n \log n) \) time.

Consider the set of all 0-distance objects, \( S_0 \); and the set of 0-distance neighbors in the spanning path, \( S_p \). Then \( S_p \subseteq S_0 \). Clearly the reverse subset relationship does not often hold, and so there is no guarantee of having each 0-distance as a neighbor, nor necessarily close, in the spanning path. Let us see how this works in practice, since we have already noted that 0-distances are very robust relative to agglomerative criterion.

Figure 5 shows the spanning path, and pairwise recoded distances. Note that “wrap-around”, or recoded distance from last to first, is not considered here. Seeking 0-distance neighbors in the spanning path, and agglomerating them, leads to a first pass over the spanning path. The spanning path itself provides candidates. By repeating (i) the defining of a new spanning path, and (ii) use of this new spanning path for locating 0-distance neighbors, we carry out a new pass over the data. Typically as seen in Figure 5 we quickly exhaust the possibilities of this approach to find further agglomerable 0-distance neighbors. However this exhausting of possibilities is not guaranteed. Looking at the pairwise recoded distances between remaining clusters (or singletons) – see at the bottom of Figure 5 – we see that one 0-distance has eluded us.
Figure 4: Hierarchical clustering, Ward’s minimum variance method, with clustering cardinality weighting.
However a traditional hierarchical clustering, using, say, the Ward minimum variance method, is applicable.

4.3 Scaling Up

Table 3 shows results for four data sets. Most 0-distances are found on the first pass. Subsequent passes add to this, and since the processing is carried out on much shorter spanning paths, in each successive path, the computational time required is shorter. Figure 6 illustrates the cluster properties, with truncation of very frequent cluster sizes. Processing 20,000 chemicals, with 1052 attributes, takes a few minutes on a 1.8 GHz PowerPC G5 Apple processor, with 256 MB memory, using the high level, interpreted (and hence inefficient) R environment.

As noted previously: (i) the clusters that are based on the 0-valued recoded distances are exact and not approximated in any way; (ii) however we may well miss out on 0-valued recoded distances that should be included; (iii) furthermore we cannot exclude the possibility that clusters found ought to be merged, through having 0-valued recoded distances between at least some of their members.

From the discussion in subsection 4.1 our algorithm is a connectedness (or single linkage) one. By virtue of the discussion in subsection 3.4, the assumption that the results hold for any common cluster membership criterion (e.g., complete linkage, average linkage, etc.) is well-founded and approximates well the practical situation relating to the data used in this work.

Our preprocessing provides a good basis for subsequently hierarchically clustering the “condensed” data. From Table 3, between 30% and 40%, roughly, of the data remains to be processed. Typically from the 20,000 chemical set, about 8000 chemicals, available as an approximate 8000 \times 1052 data array, are finally hierarchically clustered using, e.g. the Ward minimum variance agglomerative criterion.

A typical timing test for the \(O(n^2)\) Ward minimum variance agglomerative criterion, on a 5-year old 2.4MHz Pentium 4 with 1 GB RAM running Windows XP SP2 using non-optimised compilations of the clustering programs, was as follows. For 15,465 structures, represented by the 1052-bit fingerprints, the times taken was 42.5 minutes.

4.4 Conclusions on Data Condensation through Recoding

The approach described in this article represents a new perspective relative to mainstream approaches using indexing (e.g., [4], using datasets of \(n = 1\) million, and \(m = 64\)) or hashing (e.g. [10]). A simple statement of our approach is: use partial single linkage agglomerative clustering as a first phase, followed by the superior [30, 6, 7, 3] minimal variance (or Ward) agglomerative clustering algorithm. Furthermore, this first single linkage phase is for one level only, where by design the agglomerate criterion value is 0. What we show empirically in this article, and theoretically using the ultrametric topology viewpoint, is that the single linkage criterion used in the first phase is not important: essentially
Spanning path, followed by pairwise recoded distances

|   | 18 | 15 | 4 | 5 | 1 | 9 | 10 | 7 | 8 | 23 | 2 | 3 | 16 | 22 | 13 | 14 | 6 | 11 | 12 | 26 | 27 | 17 | 30 | 24 | 25 | 29 | 20 | 19 | 21 | 28 |
|---|----|----|---|---|---|---|---|---|---|----|---|---|----|----|---|---|---|---|----|----|---|---|----|----|---|---|----|----|
| 0 | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 1  | 1  | 0  | 1  | 3  | 3  | 2  | 2  |

First pass clustering
New 15 ---> 18
Rep 4 ---> 18
Rep 5 ---> 18
Rep 1 ---> 18
Rep 9 ---> 18
Rep 10 ---> 18
Rep 7 ---> 18
Rep 8 ---> 18
Rep 23 ---> 18
Rep 2 ---> 18
Rep 3 ---> 18
Rep 16 ---> 18
New 13 ---> 22
Rep 14 ---> 22
Rep 6 ---> 22
Rep 11 ---> 22
Rep 12 ---> 22
Rep 26 ---> 22
Rep 27 ---> 22
Rep 17 ---> 22

Second pass clustering
New 25 ---> 24
New 22 ---> 18

Third pass clustering
No change

Recoded distance matrix resulting from this

<table>
<thead>
<tr>
<th></th>
<th>18</th>
<th>30</th>
<th>24</th>
<th>29</th>
<th>20</th>
<th>19</th>
<th>21</th>
<th>28</th>
</tr>
</thead>
<tbody>
<tr>
<td>18</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>3</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>30</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>3</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>24</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>3</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>29</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>19</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>21</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>28</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td></td>
</tr>
</tbody>
</table>

Figure 5: Results of our procedure for finding many 0-distances, based on recoded distances and a spanning path, using the data in Table 2.
<table>
<thead>
<tr>
<th>Data set</th>
<th>Pass</th>
<th>No. rows</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>–</td>
<td>20000</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>8487</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>8393</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
<td>8372</td>
</tr>
<tr>
<td>1</td>
<td>4</td>
<td>8364</td>
</tr>
<tr>
<td>1</td>
<td>5</td>
<td>8360</td>
</tr>
<tr>
<td>2</td>
<td>–</td>
<td>20000</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>6969</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>6825</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>6776</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>6757</td>
</tr>
<tr>
<td>2</td>
<td>5</td>
<td>6747</td>
</tr>
<tr>
<td>3</td>
<td>–</td>
<td>20000</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>7266</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>7142</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>7111</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td>7102</td>
</tr>
<tr>
<td>3</td>
<td>5</td>
<td>7097</td>
</tr>
<tr>
<td>4</td>
<td>–</td>
<td>20000</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>5365</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>5269</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>5248</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>5238</td>
</tr>
<tr>
<td>4</td>
<td>5</td>
<td>5235</td>
</tr>
</tbody>
</table>

Table 3: Results of 5 passes of 0-finding on four data sets, each of 20,000 chemicals (in sequence from, respectively, the 20 thousandth, 30 thousandth, 90 thousandth and 100 thousandth).
Figure 6: Histogram of cardinalities resulting from the 0-distance processing of two different data sets.
any other commonly used criterion, including the minimum variance one, would do equally well.

Our approach is practical and generally applicable. The data characteristics do play a role, and it was for this reason that our work began with a study of chemical/attribute frequency of occurrence density, and marginal densities. The general principles of our procedure are clear. Firstly, we discretize the precision with which we measure distance, in order to force a good deal of the data into clusters, where each cluster has its members (i.e., observation vectors) superimposed. This principle is a filtering one, widely used for denoising in signal and image processing, or regression in statistical data analysis. This condenses the data considerably. Then, secondly, on the condensed data we apply an agglomerative hierarchical clustering criterion of choice. For the data under consideration, and for the software environments used, we recommend preprocessing of about 20,000 observations at a time, and the exact hierarchical clustering on up to about 8500 observations, which can be carried out in reasonable compute time, and which lends itself to the processing in parallel of very large data sets on a computing cluster.

5 Ultrametric from Longest Common Prefixes

5.1 Hierarchical Clustering and Formal Concept Analysis

Typically hierarchical clustering is based on a distance (which can be relaxed often to a dissimilarity, not respecting the triangular inequality, and mutatis mutandis to a similarity), defined on all pairs of the object set: \( d : I \times I \rightarrow \mathbb{R}^+ \).

I.e., a distance is a positive real value. Usually we require that a distance cannot be 0-valued unless the objects are identical (but we will ignore this here, since our data encoding may well have non-identical objects being identically represented). That is the traditional approach. Now we consider a different definition of distance, such that it maps pairs of objects onto elements of a join semilattice. The latter can represent all subsets of the attribute set, \( J \). That is to say, it can represent the power set, commonly denoted \( 2^J \), of \( J \).

Now, consider, say, \( n = 5 \) objects characterized by 3 boolean (presence/absence) attributes, shown in Table 4.

Define dissimilarity between a pair of objects in Table 4 as a set of 3 com-
ponents, corresponding to the 3 attributes, such that if both components are 0, we have 1; if either component is 1 and the other 0, we have 1; and if both components are 1 we get 0. This is the simple matching coefficient [17]. We could use, e.g., Euclidean distance for each of the values sought; but we prefer to treat 0 values in both components as signaling a 0 contribution. We get then:

\[
\begin{align*}
  d(a, b) &= 1, 1, 0 \\
  d(a, c) &= 0, 1, 0 \\
  d(a, e) &= 0, 1, 1 \\
  d(a, f) &= 1, 1, 0 \\
  d(b, c) &= 1, 1, 0 \\
  d(b, e) &= 1, 1, 1 \\
  d(b, f) &= 1, 1, 0 \\
  d(c, e) &= 0, 1, 1 \\
  d(c, f) &= 1, 1, 0 \\
  d(e, f) &= 1, 1, 1
\end{align*}
\]

If we take the three components in this distance as \(d_1, d_2, d_3\), and considering a lattice representation with linkages between all ordered subsets where the subsets are to be found in our results above (e.g., \(d(c, f) = 1, 1, 0\) implies that we have subset \(d_1, d_2\)), and finally such that the order is defined on subset cardinality, then we see that the scheme shown in Figure 7 suffices to illustrate all salient relationships.

In Formal Concept Analysis, it is the lattice itself which is of primary interest. In [17] there is discussion of the close relationship between the traditional hierarchical cluster analysis based on \(d : I \times I \rightarrow \mathbb{R}^+\), and hierarchical cluster analysis “based on abstract posets” (a poset is a partially ordered set), based on \(d : I \times I \rightarrow 2^J\).

5.2 From Boolean Data to Normalized, Real-Valued Data

Consider now our need to normalize the data. The data in Table 4 will be normalized, as throughout this work, by dividing each value by its corresponding column sum. We can consider the hierarchical cluster analysis from abstract posets as based on \(d : I \times I \rightarrow \mathbb{R}^{|J|}\) as is done in the previous section, section 5.1. (In [17], the median of the \(|J|\) distance values is used, as input to a traditional hierarchical clustering, with alternative schemes discussed).

Our approach is as follows. As discussed, we consider the hierarchical cluster analysis from abstract posets based on \(d : I \times I \rightarrow \mathbb{R}^{|J|}\). Next we use the longest common prefix distance (which is an ultrametric) on each of the \(|J|\) distance components. Finally, we will cluster objects based on the same longest common prefix distance on all \(|J|\) distance components.

Let us consider direct analysis of our data, but taken as unnormalized. That is to say, we have as input data an array denoted \(x_{iJ}\), or again that we have any two objects that we can denote by the boolean vectors \(x_{iJ}, x_{i'J}\). Let us use the simple matching coefficient as in the worked example in the previous section, section 5.1. Then all aspects of our simple illustrative example above are
Potential lattice vertices | Lattice vertices found | Level
--- | --- | ---
d1, d2, d3 | d1, d2, d3 | 3
/\  
/\  
d2, d3  d1, d3 | d1, d2  d2, d3 | 2
\/  
\/  
d1  d2  d3 | d2 | 1

The set d1, d2, d3 corresponds to: d(b, e) and d(e, f)
The subset d1, d2 corresponds to: d(a, b), d(a, f), d(b, c), d(b, f), and d(c, f)
The subset d2, d3 corresponds to: d(a, e) and d(c, e)
The subset d2 corresponds to: d(a, c)

Clusters defined by all pairwise linkage at level <= 2:
a, b, c, f
a, e
c, e

Clusters defined by all pairwise linkage at level <= 3:
a, b, c, e, f

Figure 7: Lattice and its interpretation, corresponding to the data shown in Table 4 with the simple matching coefficient used. (See text for discussion.)
relevant. We come to a stage where we must decide to find a parsimonious join semilattice representation, or to more directly build a hierarchical clustering, by (i) forming a composite of the distance components (the 3 values in the illustrative example), and (ii) using this single valued pairwise distance, adopting a criterion like the single or complete linkage ones (see [17] for examples).

We prefer to pursue another direction. Reasons include the following: (i) there is value in normalizing our data, and we want to support this operation on our data; and (ii) we seek to avoid a choice of agglomerative criterion.

Consider the following simple ("toy") example of our new procedure. Let us take two illustrative objects,

1, 1, 0, 0, and
0, 1, 0, 1.

The simple matching coefficient, elementwise, gives the distance vector

1, 0, 1, 1.

Now our procedure will be described. Through normalization (i.e., column sum division of each value), let us assume that the two illustrative objects become the following:

0.0070, 0.0121, 0.0000, 0.0000, and
0.0000, 0.0121, 0.0000, 0.0013.

For convenience we use 4 decimal places of precision, and we assume, again for illustrative convenience, that the full precision in all cases is to 4 decimal places of precision.

Our hierarchy is precision-based. We begin with equality of all values to precision 1, i.e., 0.0 above.

Next we look for precision 2 equality above. For successive values we have: 0.00, 0.01, 0.00, and 0.00. Thus we have equality of the above pair of vectors to precision 2.

We see that in the first and fourth components we no longer have precision 3 equality. (The first components are: 0.0070 and 0.0000; the fourth components are: 0.0000 and 0.0013.)

We say that the above two vectors are equal up to precision 2.

We will pursue the elaboration and assessment of this procedure in the remainder of this article.

The rationale for cluster analysis based on posets is to classify, not through summarizing the attributes used (via distances or dissimilarities), but instead through direct processing of the individual attributes. That is to say the attributes are not mapped into a single value, but retain their separate roles, at least for some way into the procedure. This rationale is also our motivation for efficiently handling massive datasets.
6 Ultrametrization through Baire Space Embedding

6.1 Notation, Longest Common Prefix Ultrametric

A Baire space [22] consists of countably infinite sequences with a metric defined in terms of the longest common prefix: the longer the common prefix, the closer a pair of sequences. What is of interest to us here is this longest common prefix metric, which additionally is easily seen to be an ultrametric. The longest common prefixes at issue here are those of precision of any value (i.e., \( x_{ij} \), for chemical compound \( i \), and chemical structure code \( j \)). Consider two such values, \( x_{ij} \) and \( y_{ij} \), which, when the context easily allows it, we will call \( x \) and \( y \). Each are of some precision, and we take the integer \(|K|\) to be the maximum precision. We pad a value with 0s if necessary, so that all values are of the same precision. Finally, we will assume for convenience that each value \( x \in [0, 1) \) and this can be arranged by normalization.

6.2 The Case of One Attribute

Thus we consider ordered sets \( x_k \) and \( y_k \) for \( k \in K \). In line with our notation, we can write \( x_K \) and \( y_K \) for these numbers, with the set \( K \) now ordered. (So, \( k = 1 \) is the first decimal place of precision; \( k = 2 \) is the second decimal place; \( \ldots ; k = |K| \) is the \(|K|\)th decimal place.) The cardinality of the set \( K \) is the precision with which a number, \( x_K \), is measured. Without loss of generality, through normalization, we will take all \( x_K, y_K \leq 1 \). We will also consider decimal numbers, only, in this article (hence \( x_k \in \{0, 1, 2, \ldots, 9\} \) for all numbers \( x \), and for all digits \( k \)), again with no loss of generality to non-decimal number representations.

Consider as examples \( x_K = 0.478; \) and \( y_K = 0.472 \). In these cases, \(|K| = 3 \). For \( k = 1 \), we find \( x_k = y_k = 4 \). For \( k = 2 \), \( x_k = y_k \). But for \( k = 3 \), \( x_k \neq y_k \).

We now introduce the following distance:

\[
d_B(x_K, y_K) = \begin{cases} 
1 & \text{if } x_1 \neq y_1 \\
\inf 2^{-n} & x_n = y_n \quad 1 \leq n \leq |K|
\end{cases}
\]  

(1)

The Baire distance is used in denotational semantics where one considers \( x_K \) and \( y_K \) as words (of equal length, in the finite case), and then this distance is defined from a common \( n \)-length prefix, or left substring, in the two words. For a set of words, a prefix tree can be built to expedite word matching, and the Baire distance derived from this tree.

We have \( 1 \geq d_B(x_K, y_K) \geq 2^{-|K|} \). Identical \( x_K \) and \( y_K \) have Baire distance equal to \( 2^{-|K|} \). The Baire distance is a 1-bounded ultrametric.

The Baire ultrametric defines a hierarchy, which can be expressed as a multiway tree, on a set of numbers, \( x_{1K} \). So the number \( x_{iK} \), indexed by \( i \), \( i \in I \), is of precision \(|K| \). It is actually simple to determine this hierarchy. The partition at level \( k = 1 \) has clusters defined as all those numbers indexed by \( i \) that share
the same kth or 1st digit. The partition at level \( k = 2 \) has clusters defined as all those numbers indexed by \( i \) that share the same kth or 2nd digit; and so on, until we reach \( k = |K| \). A strictly finer, or identical, partition is to be found at each successive level (since once a pair of numbers becomes dissimilar, \( d_B > 0 \), this non-zero distance cannot be reversed). Identical numbers at level \( k = 1 \) have distance \( \leq 2^0 = 1 \). Identical numbers at level \( k = 2 \) have distance \( \leq 2^{-1} = 0.5 \). Identical numbers at level \( k = 3 \) have distance \( \leq 2^{-2} = 0.25 \); and so on, to level \( k = |K| \), when distance = \( 2^{-|K|} \).

### 6.3 The Case of Multiple Attributes

In our case, we have \(|J|\) values for each chemical structure. So the \( i \)th chemical structure, for each \( j \in J \) value with precision \(|K|\), is \( x_{i,j,K} \). Collectively, all our data are expressed by \( x_{i,j,K} \). As before, we normalize by column sums to work therefore on \( x_{i,j,K} \). To find the Baire distance properties we work simultaneously on all \( J \) values, corresponding to a given chemical structure. Therefore the partition at level \( k = 1 \) has clusters defined as all those numbers indexed by \( i \) that share the same kth or 1st digit in all \( J \) values.

Consider identity between rows \( i \) and \( i' \) to precision \( k_0 \). So for all digits \( 1 \leq k \leq k_0 \), we have \( x_{ij,k} = x_{i'j,k} \). The maximum Manhattan or \( L_1 \) distance between chemical compounds \( i \) and \( i' \) is then given by a discrepancy of 0.999999... in the \( k_0 + 1 \) significant digit of each attribute, \( j \). For all attributes, then, the maximum \( L_1 \) distance between \( i \) and \( i' \) is \( 1 \cdot |J| = |J| \). If we use instead the Euclidean or \( L_2 \) distance, then the maximum distance between \( i \) and \( i' \) is seen to be the same, \( 1^2 \cdot |J| = |J| \).

### 6.4 Analysis: Baire Ultrametrization from Numerical Precision

In this section we use (i) a random projection of vectors into a 1-dimensional space (so each chemical structure is mapped onto a scalar value, by design \( \geq 0 \) and \( \leq 1 \)) followed by (ii) implicit use of a prefix tree constructed on the digits of the set of scalar values. First we will look at this procedure. Then we will return to discuss its properties.

We seek all \( i, i' \) such that

1. for all \( j \in J \),
2. \( x_{ij,K} = x_{i'j,K} \)
3. to fixed precision \( K \)

Recall that \( K \) is an ordered set. We impose a user specified upper limit on precision, \(|K|\).

Now rather than \(|J|\) separate tests for equality (point 1 above), a **sufficient condition** is that \( \sum_j w_j x_{ij,K} = \sum_j w_j x_{i'j,K} \) for a set of weights \( w_j \). What helps in making this sufficient condition for equality work well in practice is that many of
Table 5: Results for the three different data sets, each consisting of 7500 chemicals, are shown in immediate succession. The number of significant decimal digits is 4 (more precise, and hence more different clusters found), 3, 2, and 1 (lowest precision in terms of significant digits).

<table>
<thead>
<tr>
<th>Sig. dig. c</th>
<th>No. clusters</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>6591</td>
</tr>
<tr>
<td>4</td>
<td>6507</td>
</tr>
<tr>
<td>4</td>
<td>5735</td>
</tr>
<tr>
<td>3</td>
<td>6481</td>
</tr>
<tr>
<td>3</td>
<td>6402</td>
</tr>
<tr>
<td>3</td>
<td>5360</td>
</tr>
<tr>
<td>2</td>
<td>2519</td>
</tr>
<tr>
<td>2</td>
<td>2576</td>
</tr>
<tr>
<td>2</td>
<td>2135</td>
</tr>
<tr>
<td>1</td>
<td>138</td>
</tr>
<tr>
<td>1</td>
<td>148</td>
</tr>
<tr>
<td>1</td>
<td>167</td>
</tr>
</tbody>
</table>

The $x_{iJK}$ values are 0: cf. the approximate 8% matrix occupancy rate that holds here. We experimented with such possibilities as $w_j = j$ (i.e., \{1, 2, ..., |J|\}) and $w_j = |J|+1-j$ (i.e., \{|J|, |J|-1, ..., 3, 2, 1\}). A first principal component would allow for the definition of the least squares optimal linear fit of the projections. The best choice of $w_j$ values we found for uniformly distributed values in (0, 1): for each $j$, $w_j \sim U(0, 1)$.

Table 5 shows, in immediate succession, results for the same set of three data sets used previously. The normalizing column sums were calculated and applied independently to each of the three data sets. Insofar as $x_{iJ}$ is directly proportional, whether calculated on 7500 chemical structures or 1.2 million, leads to a constant of proportionality, only, between the two cases. A random projection was used. Finally, identical projected values were read off, to determine clusters.

Let us look closer at one outcome here, the 4-digit precision set of 6591 clusters found for the first of the three data sets used. We may ask whether these clusters are “balanced” or if, in fact, one massive cluster accounts for most of the chemical structures. Figure 8 shows a histogram, indicating clearly the “balance” in cluster cardinalities.

For a smaller precision, however, such as 1-digit, we find that one very large cluster dominates in terms of cardinality (cf. discussion of k-means results in section 6.8 below).
Figure 8: Histogram of cluster sizes.
6.5 Discussion: Random Projection and Hashing

Random projection is the finding of a low dimensional embedding of a point set—dimension equals 1, or a line or axis, in this work—such that the distortion of any pair of points is bounded by a function of the lower dimensionality [35]. There is a burgeoning literature in this area, e.g. [10]. While random projection per se will not guarantee a bijection of best match in original and in lower dimensional spaces, our use of projection here is effectively a hashing method ([23] uses MD5 for nearest neighbor search), in order to deliberately find hash collisions—thereby providing a sufficient condition for the mapped vectors to be identical.

Collision of identically valued vectors is guaranteed, but what of collision of non-identically valued vectors, which we want to avoid?

To prove such a result may require an assumption of what distribution our original data follow. A general class is referred to as a stable distribution [18]: this is a distribution such that a limited number of weighted sums of the variables is also itself of the same distribution. Examples include both Gaussian and long-tailed or power law distributions.

Interestingly, however, very high dimensional (or equivalently, very low sample size or low n) data sets, by virtue of high relative dimensionality alone, have points mostly lying at the vertices of a regular simplex or polygon [27, 16]. This intriguing aspect is one reason, perhaps, why we have found random projection to work well. Another reason is the following: if we work on normalized data, then the values on any two attributes j will be small. Hence \( x_j \) and \( x'_j \) are small. Now if the random weight for this attribute is \( w_j \), then the random projections are, respectively, \( \sum_j w_j x_j \) and \( \sum_j w_j x'_j \). But these terms are dominated by the random weights. We can expect near equal \( x_j \) and \( x'_j \) terms, for all \( j \), to be mapped onto fairly close resultant scalar values.

Further work is required to confirm these hypotheses, viz., that high dimensional data may be highly “regular” or “structured” in such a way; and that, as a consequence, hashing is particularly well-behaved in the sense of non-identical vectors being nearly always collision-free.

6.6 Discussion: Prefix Trees or Tries

A prefix tree, or trie, is well-known in the searching and sorting literature [15], and is used to expedite the finding of longest common prefixes. At level one, nodes are associated with the first digit. At level two, nodes are associated with the second digit, and so on through deeper levels of the tree.

Berkeley DB (Berkeley Database, www.oracle.com/database/berkeley-db.html) provides for great scalability in dataset size, and furthermore supports trie storage. In future work we will investigate its use for efficiently and effectively supporting ultrametrization through Baire space embedding.
6.7 Simple Clustering Hierarchy from the Baire Space Embedding

The Baire ultrametrization induces a (fairly flat) multiway tree on the given data set.

Consider a partition yielded by identity (over all the attribute set) at a given precision level. Then for precision levels \(k_1, k_2, k_3, \ldots\) we have, at each, a partition, such that all member clusters are ordered by reverse embedding (or set inclusion): \(q(1) \supseteq q(2) \supseteq q(3) \supseteq \ldots\). Call each such sequence of embeddings a chain. The entire data set is covered by a set of such chains. This sequence of partitions is ordered by set inclusion.

The computational time complexity is as follows. As usual, let the number of chemicals be denoted \(n = |I|\); the number of attributes is \(|J|\); and the total number of digits precision is \(|K|\). Consider a particular number of digits precision, \(k_0\), where \(1 \leq k_0 \leq |K|\). Then the random projection takes \(n \cdot k_0 \cdot |J|\) operations. A sort follows, requiring \(O(n \log n)\) operations. Then clusters are read off with \(O(n)\) operations. Overall, the computational effort is bounded by \(c_1 \cdot |I| \cdot |J| \cdot |K| + c_2 \cdot |I| \cdot \log |I| + c_3 |I|\) (where \(c_1, c_2, c_3\) are constants), which is equal to \(O(|I| \log |I|)\) or \(O(n \log n)\).

6.8 Comparison with Other Clustering Algorithms

Consider the choice of a given ("significant") digit, \(|K|\), and consider distance 0 between two values. Then for digit \(|K| + 1\), a maximum value of 0 and a minimum value of 1 is possible. So any two values will be strictly less than digit \(|K|\) in value: the tolerance on the 0-value of the pair is \(10^{-|K|}\).

Example: 0.9124 and 0.9127. Let \(|K| = 3\), implying that we focus on the same 3 significant digits, here, viz. 0.912. The greatest discrepancy (e.g., using \(L_1\) or \(L_2\) distance) between two 0 values is the case, for example, were we to have 0.9120 and 0.9129. The tolerance on such a 0-value is therefore \(10^{-3}\).

Now we are considering, in all, the set \(J\) of such values. For a Euclidean distance, the overall tolerance on a 0-distance is therefore \(|J| \cdot 10^{-|K|}\). For a city-block distance, the tolerance on a 0-distance is \(|J| \cdot \frac{1}{2} \cdot 10^{-|K|}\); and for a Chebyshev \(L_\infty\) distance, the tolerance on a 0-distance is \(10^{-|K|}\). Whatever distance we choose we have a tolerance.

It results that our clusters of approximately, mutually, all 0-distance pairs, can also be characterized as being within a diameter given by one of the foregoing tolerances (which one depends on the distance chosen). Our clustering therefore is a fixed diameter one, or – using more usual terminology – a fixed radius clustering method. The distinction between radius and diameter holds for non-ultrametric frameworks, which is likely to be our point of departure. In an ultrametric space, the radius is equal to the diameter (just as every point in a sphere is its center).

A fixed radius clustering can be used as input to hierarchical clustering in the following way. Using Euclidean or some other distance, determine all neighbors of a given point that are within distance \(\rho\). Then from Bruynooghe’s reducibil-
Table 6: Results of k-means for the same three data sets used heretofore, each relating to 7500 chemical structures, with 1052 descriptors. “Sig. dig.”: number of significant digits used. “No. clusters”: number of clusters in the data set of 7500 chemical structures, associated with the number of significant digits used in the Baire scheme. “Largest cluster”: cardinality. “No. discrep.”: number of discrepancies found in k-means clustering outcome. “No. discrep. cl.”: number of clusters containing these discrepant assignments.

7 Conclusions

We have developed and carried out assessments on (i) an approach to “condensing” a data set by finding very close neighbors, followed by a traditional hierarchical clustering of the condensed data; and (ii) an approach to “threading” all values that are identical up to successively finer digits of precision. We linked the latter to Formal Concept Analysis. A hierarchical clustering, in the second case a flat, multiway tree, is the overall objective of both approaches.

In both cases, data precision plays a central role. We linked data precision to data coding. In turn we strongly motivated our interest in such data coding by the benefits it brings vis-à-vis the hierarchical structure property, or ultrametricity, that is our ultimate goal. We have shown both approaches to be eminently practical, for our purposes of a hierarchical summarization of data.

A range of avenues for further work have been touched on in this article. Principal among them are the following. Firstly, both our algorithms implicitly use hashing. Hence further comparisons with hashing approaches pursued by Indyk (cf. [18]) could be of importance. So too would further study of the partitioning that is implicit in the flat hierarchies that we have obtained. In [37] and related work by Karypis, the benefits of partitioning over hierarchical structure are pointed to, and [32] proposes a “bisecting” or divisive (hence hybrid partitioning/hierarchical) k-means algorithm. In subsection 6.8 we have started a comparative evaluation with k-means. Our preliminary results show
that a similar outcome is obtained. Further work is needed, in particular given the $O(n \log n)$ computational requirement of our approach and of the “bisecting” k-means algorithm.

References


