# Mining a database of Fungi for Pharmacological Use via Minimum Message Length Encoding

Robert Zimmer[1] and Andrew Barraclough[2]

[1] Department of Computer Science, The Open University, Milton Keynes, England
R.M.Zimmer@open.ac.uk

[2] Information Systems Group, UNISYS UK, The Octagon, Slough, UK.

**Abstract.** This paper concerns the use of fungi in pharmaceutical design. More specifically, this research involves mining a database of fungi to determine which ones have waste products that are unusual in their spectral fingerprints, and therefore worth being tested for medicinal properties. The technique described in this paper involves Minimum Message Length encoding. Minimum Message Length (sometimes called Minimum Description Length) encoding is a method for choosing a binary coding for a set of data. The method's goal is to use the frequency of occurrence of each data point to ensure that frequently occurring data are given short codes. Minimum Message Length encoding provides a solution that is optimal in the sense that if the entire data set is employed in the encoding, then the code generated will have the property that no other unambiguous prefix code will provide a shorter encoded version of the entire set. In this paper, the process is turned on its head. The problem that is addressed is: given a large database, how can we pick out the elements that are quite different from the others. The first step in our solution involves using the Minimum Message Length algorithm to generate a compact code for all, or a representative learning section, of the data. The data that require long descriptions in this code are likely to be the ones that possess unusual features. In this paper, we describe this process in some detail, and explain the application of it to a database of fungi.

## Introduction to the Problem and Technique

The pharmaceutical industry uses fungus excretion to produce drugs. As this process has been used for quite a long time, a large number of chemicals have already been produced and tested. The problem addressed by the techniques described in this paper is the following: given a large database of unexplored fungi how can one decide which fungi should be chosen to explore next. The first step in the answer to this question is to choose a fungus the spectral sequence of which seems to be as different from the rest as possible. One way to do this is by using clustering techniques to group the data and then to choose a fungus that falls out of any of the groups. In this paper we take a different tack. We use a coding technique that is designed to make typical elements have short encodings. Then the fungi with long encodings are likely to be the least typical and, therefore, ripe for further study. Given a large database, the encoding can be generated from a small sample and then all the fungi (and new ones that come along) can be encoded to see how typical they are.

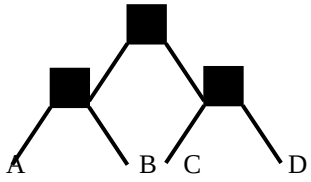
## Minimum Message Length (MML) Encoding

The Minimum Message Length encoding scheme begins by applying the Huffman coding method to a set of characters. Consider, for example, the question of encoding as a string of 0's and 1's the following message:

AABABCAADABAABC

Without taking account of the frequency of the characters, the best thing that could be done is to treat the distribution as uniform, and code each of the four letters as a two-bit string, For example: A → 00, B → 01, C →10, D → 11.

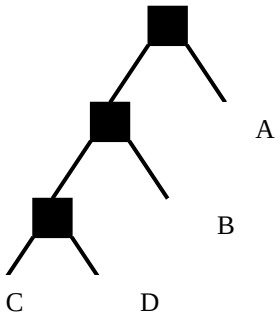
And the deccoding tree for the coding is the following balanced tree:




With the convention that reading "0" means go left. And the this encoded message is the 30-bit long word:

000001000110000011000100000110

This is one of the extrema that could be employed in the coding. The other extremum is to make a near linear tree as follows:




The coding then is A → 1, B → 01, C → 001, and D → 0001. And the message is encoded as:

11011010011100011011101001

which has only 26 bits. This example was designed to make the linear code effective. Indeed, with four characters there is nothing in between the balanced and linear trees. In general the optimal solution is between the two extremes. In the next section the technique, Huffman Encoding, of generating the trees is described.

**Huffman Encoding**

The principal of Huffman encoding can be summarized as follows:

(i) Work out the frequency of each data value, Pi

$$Pi = Ni / N$$

Where:

Pi is the probability of a particular data value
Ni is the number if instances of that particular data value in the data
N is the total number of items in the data

(2) Take the two least frequently occurring letters, treat them as one term whose frequency is the sum of the two individual frequencies. The two letters should be thought of as a node of a binary tree with two subtrees, each of which is a leaf.

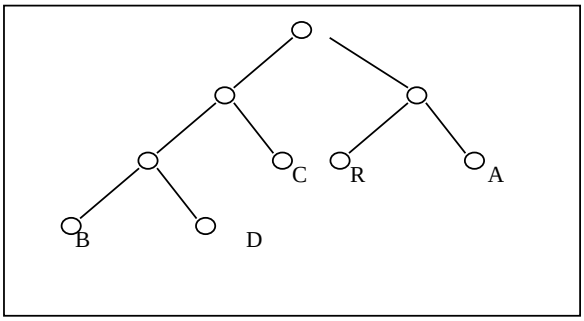
(3) Perform (2) iteratively until you reach a single tree.

Consider the following example. We wish to code the string ABRACADABRARCRC, how can the shortest possible binary string be constructed to represent this?

The first step is to count the frequency of each character:

A occurs 5 times.
B occurs 2 times.
R occurs 4 times.
C occurs 3 time.
D occurs 1 time.

The two least frequent letters are B and D. They are joined together and jointly appear 3 times which now ties the pair with C as the least frequent set. The three are then treated as one data point and are the binary tree with one subtree that is a leaf (labelled with C) and another that has both B and D below it. This tree has cumulative frequency of 6, which is now the highest,, R and A are the two lowest,so get joined, etc. When the probess finishes the following treehas been generated:




This process clearly has the effect of making theleast frequently occuring letters have the longest paths. This is optimal in the sense that:

> "The length of the encoded message is equal to the weighted external path length of the Huffman frequency tree…. "No tree with the same frequencies in external nodes has lower weighted external path length than the Huffman tree"
> (SEDGEWICK. (1988) page 326-7).

That this is the minimal message length of all possible message lengths can be proven by induction. The length in bits of each codeword (i.e. the binary encoding for a specific data value) is:

$$L_i = - \log_2 (P_i)$$

Where $L_i$ is the codeword length required for datatype i and $P_i$ represents the probability of that datatype occurring.

## Mining Fungi Data

The problem under consideration is that of analysing data representing the chemical structure of different types of fungi. A number of such fungi have been collected and each one broken down into 5000 attributes representing its chemical structure. Each one of these attributes is represented as a numerical value.

The goal of the data mining exercise is to identify those fungi that contain an unusual chemical structure. This entails identifying patterns among the 5000 attributes common amongst the fungi and those fungi that stand out as having unusual patterns. This presents an unusual knowledge discovery task in so far as the goal is clearly defined and the data is presented in a consistent numerical format allowing the data mining stage of the process to be immediately embarked upon. However, as will be shown this does not mean that data processing or modelling will not be required.

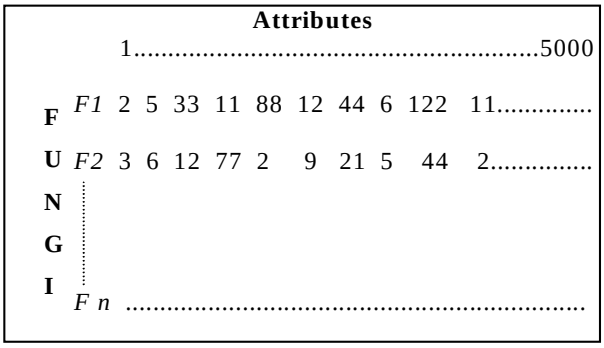
## Using MML to Find Outliers

The overview of the technique is:

1. Take a Sample set of Fungi, and use them to generate the Huffman Code
2. Encode the rest of the database using the Huffman Code
3. Choose the fungi with long descriptions as possible outliers, and test these

### Implementation

The fungi data is represented as 5000 numerical attributes for each individual fungus.




Each attribute from 1 to 5000 has its own set of values consisting of one value per fungus. Using MML encoding a Huffman tree for each of the 5000 attributes based upon the relative frequencies of values of that particular attribute was constructed.

Having created an encoding tree for each of the attributes, these trees can then be used to encode each individual fungus as a binary string. This means that each binary value encoding will be shortest where that particular value is frequent. The fungi with the overall shortest binary encoding will therefore have the overall most ‘common’ attribute values. Those fungi with the longest overall binary encoding will represent the fungi with the ‘least typical’ values. These are the unusual fungi that are required to be identified.

### Possible Pitfalls

In order for this method to be effective the range of values for each attribute was considered individually. It turns out that for some attributes, no particular value occurs terribly frequently. For these attributes the values were binned: so that values within a certain region were treated as identical. Care must be taken not to over emphasise any particular attributes relative to others. Statistical techniques have been used to analyse the attributes and their range and distribution of values.

Given the fact that any processing performed upon the attributes must retain the relative frequency distribution amongst attributes it is probable that some attributes will be significant in the classification process whilst others may well show little or no grouping of values. Such attributes will be identifiable as their encoding trees will be much deeper than those where a high frequency of values leads to fewer values needing to be encoded.

Once identified these attributes need to be excluded from the classification process as they may distort the final results should they be included. This means that once the attribute encoding trees have been constructed their depths can be compared and a maximum depth determined above which attributes are not included in the fungi encoding process.

### *Encoding the data*

Once the encoding trees have been built for each of the attributes the original data can be encoded using these trees.
A simple encoding can be performed where all of the attributes are included in the encoding process. However, some of these attributes do not show any particular grouping of values and as such will not be relevant to the encoding process. Furthermore, if these attributes were included in the encoding process the final results would be distorted. Therefore the relevance of attributes to the classification of cases is necessary to provide a refined encoding where irrelevant attributes are excluded. We've done this by calculating the standard deviation of frequency values for each of the attributes. Those attributes with a high standard deviation will have a higher grouping together of values than those with a lower standard deviation. A threshold is used to determine which attributes are not included in the encoding process. Two alternative thresholds were tried—one based on the average and the other on the median of the standard deviations of the attributes. The efficacy of the two methods was then compared.

## Evaluation

Evaluation showed that the binning function was very susceptible to producing misleading results dependent upon the shape of the data. Where the range and shape of data fit the number of bins set in the program attributes are grouped together correctly. However, where this is not the case attributes can be separated where they logically belong to the same grouping and vice versa. The grouping can be particularly affected by extreme values as shown in Test 9 (e.g. one very high value leading to a very high bin size) which will cause values to be grouped falsely together. This suggests some method of determining optimal bin sizes for each attribute is necessary for this part of the program to function efficiently. The tests showed the limitations of a program based upon a fixed number of bins.

Classification depends upon the encoding length given to values by method traversal. This method was checked to show that the encoding lengths it assigned to values were correctly based upon the values position in the relevant coding tree. This position in turn being based upon the frequency assigned the attribute value. This was verified to be the case. The actual encoding values assigned however can be the same for values of different frequencies where they occur at similar depths within the tree. This means those values of similar but not exactly the same frequencies carry the same encoding weight. Whilst a correct implementation of Huffman encoding it must be noted that the encoding process will differentiate most between large variations in frequency.

### *System Testing*

The first data set consisted of 15 cases and 5 attributes. Each of the attributes had clearly defined values. The purpose of this test was to check the program could identify those cases (rows) with the most common values and assign identical encoding to identical rows. To this end one row was replicated five times, one row replicated four times, one row replicated three times, one row replicated two times and one row left unduplicated. Thus, five groups were created, no two groups shared common values and the values were such that values were allocated separate bins. This enabled expected classification of the rows to be easily predicted and the system performed as expected giving identical rows identical encoding values. The most frequent rows (five occurrences) received the lowest encoding and the single row together with the row with two occurrences received the highest encoding. It was noted that no differentiation occurred between the single occurrence and the double occurrence. This was expected as their values shared the bottom level of the encoding tree.

Also as expected refined encoding produced the same results as all the attributes were equally significant in the encoding process.

**Test Sys2**

The second data set used was a refined version of the data used in Sys1. The same format was applied but in this case attribute values in the identical rows were changed slightly. These changes were made so that each value would still fall within the same bin.

For example, the first row in the table was changed in the following way.

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1<br>.0 | 1.<br>0 | 1.<br>0 | 1.<br>0 | 1<br>5.0 | 1<br>5.0 | 2<br>5.0 | 2<br>5.0 | 2<br>5.0 | 3<br>5.0 | 4<br>0.0 | 4<br>0.0 | 4<br>0.0 | 4<br>0.0 | 4<br>0.0 | 4<br>0.0 |
| 1<br>.1 | 1.<br>2 | 1.<br>3 | 1.<br>4 | 1<br>5.1 | 1<br>5.2 | 2<br>5.1 | 2<br>5.2 | 2<br>5.3 | 3<br>0.5 | 4<br>0.1 | 4<br>0.2 | 4<br>0.3 | 4<br>0.4 | 4<br>0.5 | 4<br>0.6 |

Changed values are still grouped to the same bins.

In this way the program was run again testing that the values were grouped together correctly and the same encoding trees built as in the previous test. The frequency counting and tree building in both examples should produce the same encoding results and this was proven to be the case.

Again, as expected refined encoding produced the same results as all the attributes were equally significant in the encoding process This test does not assess the validity of the actual binning process in grouping data values but shows that the tree building and encoding process functions correctly with the given binned values.

**Test Sys3**

In this test the table used in Sys2 was further modified by incrementally changing the attribute values of case 9, which originally had no common attribute values. It was given first one common value and then progressively more common values until it became a member of the most common class. The addition of one common value immediately raised it above cases 4 and 5 the two rows with least common attributes. As more common values were added the encoding value was reduced accordingly. The test showed the more common attributes a case has the lower its encoding value and that this is affected by changes in individual attributes.

**Test Sys4**

In this test the effect of bin size combined with range of data was examined to see the effect on classification. The two attributes from the fungi data having maximum and minimum range were taken and the binning process applied to them. Where the range was small groupings were best detected with a smaller number of bins – grouping almost disappearing when 100 bins were used. However where the range was large 10 bins grouped together many attributes of widely differing values, the grouping becoming better defined with 50 bins. When encoding was applied to 39 cases using these two attributes the encoding results were different but a general trend of classification remained. In the case of only 10 bins the number of values in the encoding trees was smaller reducing the difference in encoding values of frequent and infrequent values.

## RESULTS OF CLASSIFICATION OF FUNGI DATA

Before classification was attempted some exploratory analysis of the data was performed to try to ascertain the ranges of the 9960 different attributes. It was found that these varied immensely. For example, one attribute has a range of 99948.36, while another has a range of 15.16. The average range of the attributes was 1348.8712921686774

The number of cases used in the demonstration was 39. Tests were first run with different numbers of bins (10, 50, 100) on these two extreme attributes and the results of binning compared. The tests with 10 bins showed that although the attribute with the minimum range (attribute 36) was grouped appropriately, with attribute 1832 large groupings of values occurred often-encompassing widely differing values. As the number of bins was increased less values occurred in each bin but by 100 bins grouping had almost disappeared for attribute 36 and was severely reduced for attribute 1832. A choice of 50 bins seems to provide the best compromise. In general, the less grouping is detected in those attributes with a smaller range emphasising the significance of attributes with a larger range in the classification process. However, a small number of bins leads to false grouping of attributes of greater range.

The tests were then run on the full evaluation set again using 10, 50 and 100 bins. The overall classification results proved to be consistent whatever the bin size chosen the eight most unusual fungi identified being the same in each case indicating that the classification of cases is similar whichever attributes are emphasised.

## CONCLUSIONS

This data mining tool has been constructed with the goal of classifying numerical data and identifying unusual cases within the data. An early version of the tool without any binning of values showed that where comparisons can be made using exact values, the classification process proves quite effective, testing showing that cases with many 'common' values were clearly identified with outliers showing up as having an unusually high encoding value. This requires that all attributes that group do so around exact values. Where this occurs cases can be identified as having similar encoding values. This similarity represents the fact that they have overall the same 'score' of common values. Specific patterns of values within the attributes will not necessarily be detected by this method nor will like cases necessarily share the same common attributes but outliers can clearly be identified as having the largest encoding value indicating the least number of common values.

However, the fungi data the tool has been constructed to analyse does not allow comparison of exact values and this is true of much data requiring such classification. This necessitates the binning together of values and the calculation of group frequencies using these groups to construct the encoding trees. Currently the tool uses a fixed number of bins consistent over all the attributes in an attempt not to emphasise one attribute over another. This approach seems to be flawed, often resulting in false grouping making the final encoding results unreliable, and will be replaced in later versions. In fact the combination of attribute range and number of bins results in some attributes becoming more significant in the encoding process where the range of attribute values varies.

What is needed to make this tool function correctly is a way of making the binning stage of the process more reliable. The approach of a uniform binning of all attributes would need to be replaced by some way of identifying optimal bins for each of the attributes. This could be done by performing some form of clustering on each of the attributes in turn, effectively identifying classes within a particular attribute and use those classes to bin an attributes values together. This would result in an optimal binning for each of the attributes based upon each attributes particular range and distribution of values.

The Huffman trees could then be built upon the frequencies identified for these optimal bins resulting in a more reliable classification. Minimum Message Length encoding is the underlying theory used to produce this classification tool. The idea being that the more frequently a value occurs the more 'common' it is and the minimum encoding length of a set of values can be calculated using this property. This is widely used in data compression applications where Huffman encoding is used to encode data in the shortest way possible giving the most common values the shortest encoding and resulting in a message of minimal length.

Using this technique to identify cases in data sets with few common values proves successful where data consists of specific discreet values. Where this is not the case, the problems associated with binning the data prevent the easy application of this method to many data sets. However, refinements are being implemented that will produce a more practical version of this tool applicable to a wider range of data.

## BIBLIOGRAPHY


Adriaans, P. Zantinge, D. (1996). Data Mining. London: Addison Wesley Longman Ltd.

Agrawal, R. Faloutsos, C. Swami, A. (1993) Efficient Similarity Search In Sequence Databases: In Proc Of The 4th Intl Conf. On Foundations Of Data Organisation And Algorithms (Fodo'93) 1993.

Bollobas, B. Das, G. Gunopulos, D. Manilla, H.(1997). Time-Series Similarity Problems And Well-Separated Geometric Sets: Proc. 13th Acm Symp. Computational Geometry, Scg, Pp. 454-456, Acm Press, 4-6 June 1997.

Boulton,D,M. Wallace,C,S.(1968). A Program For Numerical Classification. The Computer Journal Vol 13. No 1. February 1970 (Pages 63 – 69)

Breiman, L. Freidman, J, H. Olshen, R, A. Stone, C, J. (1984) Classification And Regression Trees: Belmont California. Wadsworth.

Chan, P, K. Matheus, C, J. Piatetsky-Shapiro, G. (1993). Systems For Knowledge Discovery In Databases: Ieee Transactions On Knowledge And Data Engineering. Vol 5. No 6. December 1993.

Fayyad,U.M.(Ed.) Piatetsky-Shapiro,G.(Ed) Smyth,P.(Ed) And Uthurusamy,R(Ed). (1996a). Advances In Knowledge Discovery And Data Mining. Menlo Park, California: The Aaai Press/Mit Press.

Hou,W,C.(1996). Extraction And Application Of Statistical Relationships In Relational Databases. IEEE Transactions On Knowledge And Data Engineering, , Vol 8, No 6, December 1966

Jain, A, K. Dubes, R,C. (1988). Algorithms For Clustering Data: New Jersey: Prentice-Hall, Inc.

Keim, D, A. Kriegel, H, P.(1993). Using Visualization To Support Data Mining Of Large Existing Databases: In Proc. IEEE

Oliver, J. Hand, D.(1994). Introduction To Minimum Encoding Inference. Tech Report 4-94, Dept Of Statistocs, The Open Universtiy, Walton Hall, Milton Keynes Mk7 6aa

Piatetsky-Shapiro, G. (Ed.) Frawley, W.J.(Ed). (1991). Knowledge Discovery In Databases. Menlo Park, California: The Aaai Press/Mit Press.

Quinlan ,J.R. (1982).Semi-Autonomous Acquisition Of Pattern-Based Knowledge: New York, Gordon And Breach.
Quinlan ,J.R. (1990). Induction Of Decision Trees: San Francisco: Morgan Kaufman Publishers Inc.
Quinlan ,J.R. (1993). C4.5: Programs For Machine Learning: San Francisco: Morgan Kaufman Publishers Inc.
Quinlan, J, R.(1993). C4.5: Programs For Machine Learning: San Francisco: Morgan Kaufman Publishers Inc
Quinlan, J. R. And. Rivest, R. L (1987) Inferring Decision Trees Using The Minimum Description Length Principle: Technical Memo, Massachusetts Institute Of Technology, Laboratory For Computer Science, Number Mit/Lcs/Tm-339, P. 22, September 1987.
Reese, G. (1997). Database Programming With Jdbc And Java. Sebastopol Ca: O'reilly & Assc
Risasanen, J. (1987). Stochastic Complexity. Journal of the Royal Statistical Society (Series B), 49:223-239
Sedgewick, R. (1988). Algorithms. Menlo Park California. Addison-Wesley Publishing Inc
Sedgewick, S.(1988). Algorithms: Menlo Park, California. Addison-Wesley Publishing Company Inc.
Shavlic, J,W. Mooney, R, J. Towell, G.G. (1991). Symbolic And Neural Learning Algorithms: An Experimental Comparison. Machine Learning, Vol 6, No 2, Pages 111-143, 1991.
Siddalingaiah,M. Lockwood, S,D.(1997). Java Api For Dummies-A Quick Reference. Foster City Ca: Idg Books Worldwide.
Silberschatz,A. Tuzhilin,A(1996).What Makes Patterns Interesting In Knowledge Discovery Systems. IEEE Transactions On Knowledge And Data Engineering, Vol 8, No 6, December 1966
Wallace, C. S. and Freeman P.R. (1987) Estimation and Inference by Compact Coding, Journal of the Royal Statistical Society (Series B), 49:240-252.
Wallace,C.(1990). Classification By Minimum-Message-Length Inference. Advances In Computing And Information - Icci '90 Springer-Verlag S Lncs V 468p 72-81m May D 1990
Weiss, S,M. Indurkhya,A,N.(1998). Predictive Data Mining. San Francisco: Morgan Kaufman Publishers Inc.
Westphal, C. And Blaxton, T. (1998) Data Mining Solutions. New York: John Wiley And Sons Inc.
Zupan, J.(1982). Clustering Of Large Data Sets: London: John Wiley & Sons Ltd.
Zytgow, J,M. Baker, J.(1991). Interactive Mining Of Regularities In Databases. Knowledge Discovery In Databases, (Pages 31-53), Menlo Park, California: The Aaai Press/Mit Press.