MDL-Based Hierarchical Clustering

Zdravko Markov
Computer Science Department
Central Connecticut State University
New Britain, CT, USA
markovz@ccsu.edu

***Abstract*—This paper presents a new hierarchical clustering algorithm based on the use of the Minimum Description Length (MDL) principle. The clusters are created by recursively splitting the data using the values of an attribute (similarly to decision tree learning), so that each cluster contains the instances that have the same value for this attribute. Attributes are chosen to minimize the MDL evaluation measure of the clustering they create. The algorithm's computational complexity is linear in the number of data instances and quadratic in the total number of different attribute-values in the data and can be substantially reduced by an efficient implementation using bit-level parallelism. We empirically evaluate the algorithm on 20 datasets from the UCI ML repository and show that it compares favorably to *k*-means and EM.**

Keywords—clustering; minimum description length principle;

I. INTRODUCTION

To assess clustering various criteria are used. Criterion functions are also used in the implementation of clustering algorithms for guiding the search for possible clusterings. In this paper we explore this approach. Our algorithm creates possible clusterings by choosing an attribute and splitting the data using the values of this attribute, i.e. each cluster contains the instances that have the same value for that attribute. This approach is similar to the divide-and-conquer approach taken by the supervised decision tree learning algorithms. To choose the attribute we use the unsupervised attribute ranking scheme based on the Minimum Description Length (MDL) principle described in [12]. Hereafter we describe an efficient algorithm for computing the MDL evaluation measure, empirically evaluate the clustering algorithm on numeric, nominal and mixed data, and compare its performance with that of two traditional clustering algorithms, *k*-means and EM.

II. RELATED WORK

Our approach to clustering is related to three areas – attribute selection, MDL principle, and decision tree learning. The attribute selection method we use falls in the framework of wrapper approaches, which evaluate the attributes by the quality of clustering obtained by using these attributes. Reference [4] uses the category utility function and the Cobweb clustering algorithm for this purpose. Reference [5] explores the EM framework and uses the scatter separability and maximum likelihood evaluation functions to find proper subsets of attributes. Another group of methods use information-based measures. A method called entropy reduction is proposed in [3]. It measures the entropy in data based on a normalized distance between pairs of instances and evaluates attributes by the reduction of the entropy when the attribute is removed from data. This and other methods are empirically evaluated for the purposes of text clustering in [10]. Our approach is similar in spirit to the one described in [13], which uses an information compression measure to create clusters with highly similar features and selects a feature from each cluster.

As we use MDL, our approach may be traced back to the classical work [15]. It uses MDL to select the attributes that partition the dataset during the process of creating a decision tree, however in the presence of class labels. There are clustering approaches that use MDL for clustering model evaluation. These approaches differ in the specific MDL encoding scheme and in the way clusters are generated. In [8] MDL is used to evaluate grouping of data items that can be compressed well together, so that the total code length over all data groups is optimized. Thus an efficient compression indicates underlying regularities that are common to all members of a group, which in turn may be used as a similarity metric for the purposes of clustering. In [9] a distance based clustering technique using MDL for evaluating clusters is proposed. Our approach also uses MDL for model evaluation; however it creates clusters by using the divide-and-conquer technique of decision tree learning.

The third related area we discuss is decision tree learning. Our algorithm employs the TDIDT framework used by Quinlan in his classical ID3 algorithm [14] and by others to create classification trees. Classification trees may be also regarded as concept hierarchies. Langley [11] discusses the similarity between classification trees and clustering trees created by hierarchical clustering algorithms such as COBWEB. This similarity is also noted in [6]. A more recent work [2] further explores these ideas and proposes an algorithm for top-down induction of clustering trees. The algorithm called TIC uses a first-order logical representation of clusters, where the nodes (tests) are conjunctions of literals and the leaves contain sets of examples that belong together. The criterion for splitting nodes is based on a distance measure that is used to select the test that maximizes the distance between the resulting clusters. Our algorithm is similar to the algorithms for building clustering trees in that it uses a measure of clustering quality to choose the best split.

III. MDL-Based Clustering Model Evaluation

Consider an information theory setting for encoding (or communicating) a dataset D with the shortest message using the attribute-value description language. Assume that there are m attributes and k different attribute-value pairs that occur in D, i.e. $m = |x|$ and $k = \left|\bigcup_{x \in D} x\right|$, where x is a data instance. Encoding D means encoding each instance $x \in D$, which in turn means choosing a subset of m attribute-value pairs out of k possible ones. Assuming that data are uniformly distributed, the probability of this choice is

$$p(x) = \frac{1}{\binom{k}{m}}$$

According to information theory the minimal code length (in bits) of the message about the occurrence of instance x in data is

$$L(x) = -\log_2 p(x) = \log_2 \binom{k}{m}.$$

To get the minimal code length of the data we multiply the minimal code length of x by the number of instances, i.e.

$$L(D) = |D| \times \log_2 \binom{k}{m}.$$

Let us consider a clustering of the data into n clusters $D = C_1 \cup C_2 \cup \ldots \cup C_n$. This clustering may be viewed as an alternative description of the data, a hypothesis H. According to the *Minimum Description Length* (MDL) principle [16] the best hypothesis should minimize the code length of the data described by the hypothesis. Formally, this means minimizing the sum of the code length of the hypothesis itself and the code length of the data given the hypothesis, i.e. $L(H)+L(D|H)$, or maximizing the difference $L(D)–L(H)–L(D|H)$, called *Information Compression.*

Assuming that there are k_i attribute-value pairs in each cluster C_i, we represent the minimal code length of H as

$$L(H) = \sum_{i=1}^{n} \left(\log_2 \binom{k}{k_i} + \log_2 n\right).$$

The term in the sum represents the number of bits needed to encode the description language for each cluster (choosing k_i out of k attribute-value pairs) and the cluster label (choosing one out of n cluster labels). Once we communicated the hypothesis, encoding each cluster comes to encoding each instance in C_i. Using the same reasoning as for computing $L(D)$ and summing up over all clusters we represent the code length of the data given the hypothesis as

$$L(D \mid H) = \sum_{i=1}^{n} |C_i| \log_2 \binom{k_i}{m}.$$

Thus MDL of H (clustering $C_1 \cup C_2 \cup \ldots \cup C_n$) is

$$MDL(C_1 \cup C_2 \cup \ldots \cup C_n) = \sum_{i=1}^{n} MDL(C_i) \text{ , where}$$

$$\sum_{i=1}^{n} MDL(C_i) = \sum_{i=1}^{n} \left(\log_2 \binom{k}{k_i} + \log_2 n + |C_i| \log_2 \binom{k_i}{m} \right). \quad (1)$$

IV. MDL-Cluster Algorithm

The algorithm (shown in Fig. 1) chooses an attribute A, which produces a data split that minimizes MDL and then recursively applies the same procedure to the resulting splits. The MDL of the split (denoted $MDL(A)$) is computed using (1), where each cluster C_i contains the instances that have the same value for that attribute. For nominal attributes the number of splits is equal to the number of attribute values. The numeric attributes are discretized into two intervals by choosing a breakpoint value. All their values are first ordered (excluding repetitions) and each one is considered as a possible breakpoint, then the breakpoint that minimizes the MDL score of the resulting data split is chosen. For attributes with too many values an equal-interval binning is applied first. The width of the interval h is computed by using the Scott's method as $h = \frac{3.5\sigma}{n^{1/3}}$, where n is the number of values and σ – the standard deviation. The process of growing the clustering tree is controlled by a parameter evaluating the information compression at each node, computed as the difference between the code length of the data at that node and the MDL of the data split produced by the selected attribute. If the compression drops bellow a specified cutoff value (parameter c) the process of growing the tree stops and a leaf node is created.

Figure 1. MDL-Cluster algorithm (c is the compression cutoff).

Function *MDL-Cluster(D)*

1. Choose attribute $A = \arg\min_i MDL(A_i)$
2. Let A take values $x_1, x_2, \ldots, x_n$
3. Split data $D = \bigcup_{i=1}^{n} C_i$, where $C_i = \{X \mid x_i \in X\}$
4. If $L(D) - MDL(A) < c$ then stop. Return D
5. For each $i = 1, \ldots, n$ Call *MDL-Cluster(C_i)*

The MDL-Cluster algorithm is implemented as an extension of the Weka package [7] and is available at http://www.cs.ccsu.edu/~markov/MDLclustering/. Below is its output produced from the soybean-small dataset (UCI ML repository [1]). This dataset is also included in the empirical evaluation (Section VI). The clustering perfectly matches the known classes (the numbers in square brackets show the class distribution followed by the majority class label).

```
> java MDLcluster data/soybean-small.arff 150
stem-cankers=0 (51.15) [0,10,0,0] D2
stem-cankers=1 (153.77)
  canker-lesion=1 (78.09) [0,0,10,0] D3
  canker-lesion=2 (42.95) [0,0,0,8] D4
stem-cankers=2 (53.56) [0,0,0,9] D4
stem-cankers=3 (46.84) [10,0,0,0] D1
```

This example also illustrates how the cutoff value may be chosen. The cluster stem-cankers=1 has substantially higher MDL (153.77) than the other three clusters at the same level, which indicates that it may be further split to produce a better compression at its subclusters. Thus a cutoff value of 150 (specified in the command line) splits that cluster, but keeps the other two because their compression is lower.

V. Efficient Computation of MDL

The key part of computing MDL is finding the parameters k_i, as the other parameters involved, k, m and n, can be determined before processing the actual data instances. The approach we take is based on the so-called *attribute-value matrix* A_{ij}. It's a square binary matrix $k \times k$ (k is the total number of attribute-value pairs in the data), initialized with 0's and processed as follows. Each data instance is represented as a binary vector $(x_1, x_2, ..., x_k)$, where each of its components x_i corresponds to an attribute value. If that value is present in the instance x_i=1, otherwise x_i=0. For each instance vector $(x_1, x_2, ..., x_k) \in D$ and each of its non-zero components (x_i=1) the i^{th} row of the matrix is OR-ed with the instance vector, i.e. $(A_{i1}, A_{i2}, ..., A_{ik}) = (A_{i1}, A_{i2}, ..., A_{ik}) \vee (x_1, x_2, ..., x_k)$. In this way when all instances are processed the number of 1's in the i^{th} row of the matrix represents the number of attribute-values k_i in cluster C_i, i.e. $k_i = \sum_{j=1}^{k} A_{ij}$.

The computational complexity of the algorithm is $O(nk^2)$, where n is the number of instances and k – the number of attribute-value pairs in data. Since the basic operation performed k^2 times is a one-bit OR, bit-level parallel implementations may substantially reduce its computational complexity. A simple use of a bitwise OR operation reduces this term by a factor of 32 or 64 (depending on the processor architecture). Another advantage is that the algorithm does not require storing the instances in the memory as they are processed one at a time. The Java implementation of the algorithm that we ran on a 3GHz Intel-based PC processed the largest dataset from Table I (reuters-3class) in 9 seconds.

VI. Empirical Evaluation

We evaluated our clustering algorithm on 20 datasets from the UCI ML repository [1], listed in Table I. We selected pure nominal, pure numeric and mixed datasets with various number of attributes, instances and classes (given in parentheses after the name of the dataset). The pure nominal and mixed datasets were binarized (indicated with the suffix "–bin") and also used for evaluation. The original reuters dataset was used to create reuters-2class and reuters-3class datasets by including the instances from the two and three largest classes correspondingly. The MDL-Cluster algorithm was compared with the *k*-means and EM algorithms as implemented in the Weka system [7]. The latter were run with their default parameters and number of clusters set to the known number of classes. To minimize their dependence on the initial clustering we ran them 10 times with different random number seeds and the clustering, which maximized the classification accuracy, was used for comparison. The MDL-Cluster compression cutoff parameter was adjusted in order to obtain a clustering tree with number of leaves equal (or close to) the known number of classes. Then the leaf clusters were used for comparison. Three evaluation metrics were used – *classification accuracy*, *log-likelihood*, and *holdout log-likelihood*. Table I shows the results.

The *classification accuracy* is computed as the proportion of instances from the majority class in the clusters (classes-to-clusters evaluation in Weka). This measure depends on whether the class labels are assigned consistently with the underlying patterns in the data and is an indication of the quality of the dataset (for example, the 1.00 accuracy of the soybean-small dataset means that its labeling perfectly matches the underlying patterns in the data). We used this measure to see whether MDL-Cluster is consistent with the best known and widely used classical clustering algorithms, which was confirmed by the experimental results.

The *log-likelihood* function estimates the overall likelihood that the data come from a distribution corresponding to the given clustering and is defined as

$$L = \sum_{i=1}^{m} \log \sum_{j=1}^{n} P(x_i \mid C_j) P(C_j),$$

where x_i are instances and C_j – clusters. The experiments showed that according to this criterion MDL-Cluster performed comparably on all datasets and on 12 datasets performed better or equally well than *k*-means and EM.

The holdout log-likelihood is a criterion for the entire model of clustering as it reflects its ability to predict new data. It is computed by randomly splitting the original data into a training set (66%) and a test set (34%) and then calculating the log-likelihood function on the test set clustered with the clustering model produced from the training set. According to this criterion MDL-Cluster performed comparably or better (on 11 datasets) than *k*-means and EM.

Our algorithm provides better classification accuracy than *k*-means and EM on all binary datasets and higher log-likelihood scores on almost all of them. Also, with binary data the algorithm creates more balanced clustering trees thus avoiding the problem with highly branching attributes, which are often selected by the MDL evaluation metric.

An important advantage of MDL-Cluster is that the models it creates are explicitly described by attribute values and in case of deeper trees have internal structure, which makes them easy to understand and use. For example, the tree produced from the reuters-3class dataset has three clusters and identifies the two most important attributes in the data as shown in the output below.

```
> java MDLcluster eval/reuters-3class.arff 280000
trade=0 (584469.05)
  rate=0 (277543.73) [339,18,30] money
  rate=1 (259602.51) [168,0,177] interest
trade=1 (206999.39) [101,301,12] trade
```

If we are interested in the structure of these clusters we may lower the compression threshold to 250000 and obtain the following tree.

```
> java MDLcluster eval/reuters-3class.arff 250000
trade=0 (584469.05)
  rate=0 (277543.73)
    mln=0 (160192.77) [178,10,28] money
    mln=1 (129134.46) [161,8,2] money
  rate=1 (259602.51)
    market=0 (117196.81) [66,0,117] interest
    market=1 (107679.77) [102,0,60] money
trade=1 (206999.39) [101,301,12] trade
```

This clustering indicates another important attribute (market) that splits the "interest" cluster in two. It also provides higher classification accuracy (0.74). The tree is very similar to the one produced by the Weka's J48 algorithm, which shows that the MDL evaluation measure can identify the important attributes in the data without using class information.

TABLE I. DATA SETS AND EXPERIMENTAL RESULTS.

Dataset (nominal+numeric, instances/classes)	Classification Accuracy			Log-likelihood			Holdout Log-likelihood		
	k-means	EM	MDL	*k*-means	EM	MDL	*k*-means	EM	MDL
anneal (32+6, 898/5)	0.54	0.64	0.77	-21.82	-21.36	-23.25	-20.57	-20.95	-22.69
anneal-bin (64+6, 898/5)	0.57	0.60	0.84	-25.21	-25.57	-25.06	-24.71	-25.14	-24.25
australian (9+6, 690/2)	0.83	0.74	0.66	-20.83	-20.89	-22.42	-19.20	-18.57	-21.70
australian-bin (36+6, 690/2)	0.58	0.56	0.63	-17.27	-19.17	-18.66	-16.05	-17.55	-18.58
breast (0+9, 699/2)	0.96	0.94	0.92	-13.50	-12.68	-13.22	-13.22	-12.23	-12.81
ecoli (0+7, 336/8)	0.74	0.83	0.73	-18.48	-18.40	-18.49	-18.15	-18.25	-18.24
german (13+7, 1000/2)	0.65	0.64	0.70	-30.56	-29.78	-29.89	-29.55	-29.11	-29.18
german-bin (52+7, 1000/2)	0.64	0.59	0.70	-39.50	-39.69	-39.28	-39.20	-39.38	-39.02
heart (0+13, 270/2)	0.80	0.81	0.62	-26.05	-26.16	-26.55	-25.60	-25.72	-26.01
iris (0+4, 150/3)	0.89	0.91	0.82	-10.71	-10.70	-10.81	-9.97	-9.95	-10.17
lymph (15+3, 148/4)	0.53	0.74	0.62	-13.67	-13.47	-14.28	-13.44	-13.22	-14.07
lymph-bin (35/3, 148/4)	0.57	0.67	0.71	-18.48	-19.37	-18.35	-21.26	-19.88	-19.64
reuters-2class (2886+0, 927/2)	0.76	0.72	0.87	-205.81	-203.35	-205.95	-210.97	-208.17	-203.45
reuters-3class (2886+0, 1146/3)	0.60	0.61	0.71	-206.10	-191.84	-197.05	-201.55	-194.19	-200.10
segment (0+19, 2310/7)	0.67	0.62	0.57	-79.34	-75.02	-79.08	-76.09	-71.30	-76.00
soybean-small (35+0, 47/4)	1.00	1.00	1.00	-10.79	-10.79	-10.71	-11.81	-11.81	12.00
soybean-small-bin (45+0, 47/4)	1.00	1.00	1.00	-14.96	-14.96	-14.96	-16.26	-16.26	-16.26
soybean (35+0, 683/19)	0.55	0.69	0.58	-14.84	-13.84	-17.08	-15.82	-14.29	-17.37
soybean-bin (83+0, 683/19)	0.55	0.64	0.70	-21.39	-21.12	-20.80	-23.30	-21.70	-21.53
vehicle (0+18, 846/4)	0.38	0.37	0.43	-55.55	-56.09	-55.55	-55.29	-54.96	-54.48

VII. CONCLUSION

In this paper, we described an MDL-based measure that evaluates clustering quality and presented an algorithm that uses this measure for clustering. We evaluated our algorithm on benchmark datasets and showed that it compares favorably to two traditional clustering algorithms, k-means and EM. The empirical study also showed that the algorithm performs well with binary and sparse data, which are typical in text data mining. Furthermore, it is a hierarchical model-based clustering algorithm that provides structural information about the clusters in human understandable form. Our future work will be aimed at improving its efficiency by parallel implementations and applying it to large text and web datasets.

REFERENCES

[1] Bache, K. and Lichman, M. UCI Machine Learning Repository [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Science, 2013.

[2] Blockeel, H., De Raedt, L., Ramon, J. Top-down induction of clustering trees, Shavlik, Jude (ed.), Proceedings of the 15th International Conference on Machine Learning (ICML-98), Madison, Wisconsin, USA, July 24-27, 1998, pp. 55-63, Morgan Kaufmann.

[3] Dash, M. and Liu, H. Feature selection for clustering. Proceedings of Pacific-Asia Conference on Knowledge Discovery and Data Mining, 2000, pp. 110–121.

[4] Devaney, M. and Ram, A. Efficient Feature Selection in Conceptual Clustering, Proceedings of the 14th International Conference on Machine Learning (ICML-97), pp. 92-97, Nashville, 1997, Morgan Kaufmann.

[5] Dy, J. G. and Brodley, C. E. Feature Selection for Unsupervised Learning. Journal of Machine Learning Research 5: 845-889, 2004.

[6] Fisher, D., and Hapanyengwi, G. Database Management and Analysis Tools of Machine Learning, Journal of Intelligent Information Systems, *2*, 1993, pp. 5-38.

[7] Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., and Witten, I. H. The WEKA Data Mining Software: An Update, SIGKDD Explorations, Volume 11, Issue 1, 2009.

[8] Kontkanen, P., Myllymaki, P., Buntine, W., Rissanen, J, and Tirri, H. An MDL framework for Data Clustering, in Grünwald, P., Myung, J. and Pitt, M. (Eds.), Advances in Minimum Description Length: Theory and Applications, MIT Press, 2005.

[9] Lai, P. H., O'Sullivan, J. A. and Pless, R. Proceedings of the 2009 IEEE International Symposium on Information Theory (ISIT), IEEE Press, Piscataway, NJ, USA, pp. 1318-1322, 2009.

[10] Liu, T., Liu, S., Chen, Z., and Ma, W. An Evaluation on Feature Selection for Text Clustering. Proceedings of the 12th International Conference on Machine Learning (ICML-2003), 488-495, Washington DC, 2003.

[11] Langley, P. Elements of Machine Learning. Morgan Kaufmann, 1996.

[12] Markov, Z. MDL-based Unsupervised Attribute Ranking, Proceedings of the 26th International Florida Artificial Intelligence Research Society Conference (FLAIRS-26), St. Pete Beach, Florida, USA, May 22-24, 2013, AAAI Press 2013, pp. 444-449.

[13] Mitra, P., C. Murthy, C., and Pal S. Unsupervised feature selection using feature similarity, IEEE Transactions on Pattern Analysis and Machine Intelligence, 24(3):301-312, 2002.

[14] Quinlan, J. R. Induction of Decision Trees. Machine Learning 1(1): 81-106, 1986.

[15] Quinlan, J. R. and Rivest, R. Inferring Decision Trees Using the Minimum Description Length Principle, Information and Computation, Vol. 80, 1989, 227 - 248.

[16] Rissanen, J. Modeling by shortest data description. Automatica, 14:465–471, 1978.