Towards a Simple Clustering Criterion Based on Minimum Length Encoding

Marcus-Christopher Ludl[1] and Gerhard Widmer[1,2]

[1] Austrian Research Institute for Artificial Intelligence, Vienna
[2] Department of Medical Cybernetics and Artificial Intelligence
University of Vienna, Austria

Abstract. We propose a simple and intuitive clustering evaluation criterion based on the minimum description length principle which yields a particularly simple way of describing and encoding a set of examples. The basic idea is to view a clustering as a restriction of the attribute domains, given an example's cluster membership. As a special operational case we develop the so-called *rectangular uniform message length* measure that can be used to evaluate clusterings described as sets of hyper-rectangles. We theoretically prove that this measure punishes cluster boundaries in regions of uniform instance distribution (i.e., unintuitive clusterings), and we experimentally compare a simple clustering algorithm using this measure with the well-known algorithms KMeans and AutoClass.

1 Introduction and Motivation

Clustering has long been recognized as a major research topic in various scientific fields, such as Artificial Intelligence, Statistics, Psychology, Sociology and others. Also termed *unsupervised learning* in Machine Learning, it represents a fundamental technique for Knowledge Discovery in Databases, useful, e.g., for data compression, data understanding, component recovery or class separation.

Various methods for grouping a set of objects have been devised. The different approaches can roughly be categorized as follows:

- **Linkage-based:** hierarchical methods (top-down/ dividing, bottom-up/ agglomerating), graph partitioning...
- **Density-based:** kernel-density estimation, grid clustering...
- **Model-based:** partitional clustering techniques, k-means, self-organizing maps, MML/MDL, mixture modeling...
- **Spectral-based:** clusterings based on the eigenvectors of the (normalized) affinity matrix

[4] gives an excellent overview of clustering techniques which can be adopted for use with large data sets. For the quite recent spectral approaches to clustering and segmentation, we refer the reader to [7] and [14].

MDL-based approaches are attractive because they provide a non-parametric way of automatically deciding on the optimal number of clusters [8], by trading

T. Elomaa et al. (Eds.): ECML, LNAI 2430, pp. 258–270, 2002.

off model complexity and fit on the data. However, up to now MDL (or MML)[1] criteria have been used in clustering predominantly in *mixture modelling* algorithms (e.g., Snob [13] or AutoClass [3]), where they proved to be very effective. The downside, however, is that mixture models are extremely hard to interpret directly.

Most clustering approaches make no effort of abstracting from the training instances and giving a description of the clusters in terms of the spaces covered (e.g. by their shapes). Overlapping clusters, probability distributions over cluster memberships (e.g. AutoClass) or centroid-based models (e.g., KMeans, which implicitly assumes that each cluster is modeled by a spherical Gaussian distribution [2]) make it difficult to develop a global viewpoint of the dependencies and subspaces created by the clusters.

For the non-technical person, on the other hand, a clustering is simply a partition, possibly described by simple conditions on properties of cluster members. Thus, in this paper we focus on a novel MDL-based clustering criterion which has two main advantages: Firstly, it should produce easily interpretable clusterings based on space descriptions (here, we focus on the special case of hyperrectangles), while achieving the same quality of results as state-of-the-art clustering algorithms. Secondly, as an MDL-based method, it yields a non-parametric way of estimating the number of clusters.

The basic idea is extremely simple. For the moment, we restrict ourselves to a simple attribute-value framework, where each example is a vector of values. We will view a clustering as an implicit restriction on the values allowed for attributes, given information about the cluster membership of an instance. Under this view, it is straightforward to formulate a generic MDL-type measure for the encoding length of a clustering. For the special case of clusters represented by hyper-rectangles and instances distributed uniformly within clusters (*not* within the instance space), we will derive an operational instantiation of this criterion and prove theoretically that this criterion behaves in an 'intuitively correct' way; that is to say, it punishes cluster boundaries in regions of uniform instance distribution (i.e., unintuitive clusterings).

In experiments with synthetic and real-world datasets, the clusterings selected by our measure will be compared to the results of two 'standard' clustering algorithms (AutoClass and KMeans). The results produced by our measure are at least competitive (with respect to the quality measures we will apply in the experiments), even if the data to be clustered do not satisfy the uniform distribution condition, and the 'true' clusters are not generated by hyper-rectangles.

[1] Although related concepts, there are subtle differences between MML and MDL [1]. In this paper, we will only be interested in the basic idea of selecting the hypothesis which yields the smallest two-part encoding, i.e. minimizes the sum of the length of the description of the model and the description of the data, relative to the model. We will not try to approximate an unknown probability distribution. In the following we will be using the term *minimum description length (MDL)* simply to mean this basic idea, not to relate our approach to either MDL or MML.

2 Minimum Length Encoding

To apply the MDL principle to the clustering task, we have to quantify exactly what amount of information a certain grouping of the examples contains, i.e. in what way and how much it reduces the cost for storing and/or transmitting the data in question. This can be answered by regarding a clustering as a restriction as to what values are allowed for certain attributes or as a narrowing of the involved intervals.

An intuitive example should make the underlying idea clear: Consider an "object" o, which we know is a living being (animal or human). Failing more information we would probably assume the number of *legs* to be an integer value from, say, $[0, 8]$, the *height* to be a real value from $]0m, 10m]$, and binary values for attributes like $canfly$ or *sex* and so on. Now, saying that o is a dog, would maybe restrict the attributes as follows: $legs = 4$, $height \in [0.1m, 1m]$ and $canfly = NO$. This classification would not affect the attribute *sex*.

So, *tagging* the object with an additional class attribute, which in this case has the value *dog*, gives information about the object by restricting the possibilities.

2.1 Clustering Message Length

Definition 1. *Let R be the relation scheme of a relational database r, n the number of tuples (i.e. examples) and c the number of clusters (i.e. distinguished classes). The* clustering message length *of the two-part clustering encoding, consisting of theory and data, is defined as follows:*

$$c.m.length(clustering) = \operatorname{ld} n + c \cdot length(clusterdescription) + \sum_{c_i \in Clusters} |elements(c_i)| \cdot length_{c_i}(element) + n \operatorname{ld} c$$

The first line of this definition captures the theory part of the encoding, $\operatorname{ld} n$ is the cost for specifying an integer number in $[0, n]$ (the number of clusters: $c \leq n$). The second line specifies the code length: $|elements(c_i)|$ is the number of examples in c_i, while $length_{c_i}$ encodes the cost for specifying an exact position in c_i, i.e. one example in this cluster. The term $n \operatorname{ld} c$ is important: it accounts for the *tagging*, i.e. the mapping of an example to one of the clusters (note: this is the worst case cost). Note that we distinguish between *message length* (the length of the complete two-part encoding) and *data code length* (the length of the data encoded relative to the theory).

2.2 Uniform Distribution over Rectangles

To illustrate the basic principle, as a special case of the aforementioned *clustering message length*, we now only consider rectangular clusters (parallel to the axes) and a uniform distribution of examples within these clusters. I.e. we assume each position in a cluster to be equally likely and encode the cost for specifying one such position: This measures the cost for encoding one example relative

to the restrictions of the according cluster. Note that we restrict ourselves to the numerical case here (with the instance space "discretized" according to the resolution of measurement – see below).

Definition 2. *With the specifications of definition 1 holding, let m be the number of attributes, v_j the number of values that attribute j ($j \in [1, m]$) can assume and $v_{i,j}$ the number of values that attribute j can assume in cluster i ($i \in [1, c]$). The* rectangular uniform message length *of the two-part clustering encoding is defined as follows:*

$$r.u.m.length(clustering) = \operatorname{ld} n + c \cdot 2 \cdot \sum_{j=1}^{m} \operatorname{ld} v_j + \sum_{i=1}^{c} \left[|elements(c_i)| \cdot \sum_{j=1}^{m} \operatorname{ld} v_{i,j} \right] + n \operatorname{ld} c$$

Again, the first line encodes the theory: Specifying rectangles parallel to the axes necessitates 2 values (`min` and `max`) from the domain of each attribute. The second line encodes the data, i.e. the examples relative to the (restricted) domains of the attributes.

This definition holds only for numerical domains, of course. In such a case, we can define $v_j = \frac{b_j - a_j}{\rho_j}$, where $[a_j, b_j]$ is the domain of attribute j and ρ_j the resolution of measurement; $v_{i,j}$ and $\rho_{i,j}$ can be defined accordingly. Note that this does not mean that we discretize the data – we merely calculate the number of possible positions to be encoded within each cluster by taking into account the resolution of measurement (as given by the data).

Symbolic attributes would necessitate, e.g., a description length of one bit per value (if we allow all possible combinations). Furthermore, we may not need `min` and `max` values for all attributes in the numerical case. We chose this particular encoding for the sake of simplicity and to illustrate the basic principle.

2.3 The *R.U.M.Length* is Well-Behaved

Clearly, the theory length contributes to the overall message length only an additive constant, if we vary the number of examples, but not the number, sizes and positions of the clusters. So, with an increasing sample size from the same source, the exact way of encoding the clusters becomes less and less important. Instead, the *data code length* needs an important property, which we will prove below.

Lemma 1. *The* rectangular uniform data code length *(the data code part of the r.u.m.length) of a sequence of uniformly distributed values, split into two clusters by one inner split point, is equal to or higher than the* rectangular uniform data code length *of the complete cluster.*

Proof: Let n be the number of uniformly distributed examples in an interval $[a, b]$. Let s be an inner split point, i.e. $a < s < b$, which splits the values into two regions. Let $p = \frac{s-a}{b-a}$ and n_1 and n_2 be the numbers of examples in these two regions, respectively.

Because the examples are uniformly distributed, we may assume that $n_1 = pn$ and $n_2 = (1-p)n$. Furthermore, we'll use the abbreviation $h = b - a$. Now, to prove the lemma, we have to show that

$$\begin{aligned} n \operatorname{ld} \frac{h}{\rho} + n \operatorname{ld} 1 &\leq np \operatorname{ld} \frac{hp}{\rho} + n(1-p) \operatorname{ld} \frac{h(1-p)}{\rho} + n \operatorname{ld} 2 \\ \operatorname{ld} h^n &\leq \operatorname{ld}(hp)^{np} + \operatorname{ld}(h(1-p))^{n(1-p)} + \operatorname{ld} 2^n \\ h^n &\leq h^{np} p^{np} h^{n(1-p)} (1-p)^{n(1-p)} 2^n \\ h^n &\leq h^n \left[p^p (1-p)^{1-p} 2\right]^n \\ p^p (1-p)^{1-p} &\geq \frac{1}{2} \end{aligned}$$

To solve this inequation, we'll prove that $f(p) = p^p(1-p)^{1-p}$ has an extreme value at $(\frac{1}{2}|\frac{1}{2})$ and that this is a minimum in $]0;1[$.

First of all, note that $f(\frac{1}{2}) = \frac{1}{2}$. To differentiate $f(p)$, we use the product rule and differentiate the logarithmic forms of the factors.[2] This yields

$$f'(p) = p^p(1-p)^{1-p}(\ln p - \ln(1-p))$$

$$f''(p) = p^p(1-p)^{1-p}\left(\frac{1}{p} + \frac{1}{1-p}\right)$$

Setting $f'(p) = 0$, yields $p = \frac{1}{2}$, thus $(\frac{1}{2}|\frac{1}{2})$ is the only extremum in $]0;1[$. Also, $f''(\frac{1}{2}) > 0$, which means that $(\frac{1}{2}|\frac{1}{2})$ is a minimum.

This means that

$$f(p) = p^p(1-p)^{1-p} \geq \frac{1}{2}$$

for $p \in]0;1[$. Note that this proof works only because we assume a uniform distribution of values in the interval $[a;b]$. **QED**

Theorem 1. *The* rectangular uniform code length *of a sequence of uniformly distributed values, split into two or more clusters, is equal to or higher than the* rectangular uniform code length *of the complete cluster.*

Proof: This follows by induction from lemma 1.

Remark: By this theorem we know that the *rectangular uniform message length* punishes "unnecessary" and "unintuitive" splits within a uniformly distributed region. Even more than that: We may expect the best clustering, i.e. the one with the lowest *r.u.m.length* to be one where regions of different density are separated from each other.

[2] Logarithmic differentiation is a not too well known technique for finding the derivative of a form such as $f(x) = g(x)^{h(x)}$. Taking the logarithm, differentiating both sides and solving for $f'(x)$, yields: $f'(x) = f(x)\left[h'(x)\ln g(x) + h(x)\frac{g'(x)}{g(x)}\right]$.

3 Experimental Evaluation

For evaluating the capabilities of such an MDL-based clustering measure, we tested the *rectangular uniform message length*, together with a brute-force stochastic algorithm, on several artificial datasets and two real-world datasets.

3.1 Evaluation Methodology

Recall: For computing *recall* values we made use of the methodology introduced in [5], yet slightly modified the definitions. In the following we will briefly explain the necessary terms, however, for further details, we refer the reader to the original paper.

A *component* is a set of related entities; in our context it may suffice to regard a *component* as a set of examples, i.e. a single cluster. Let $References$ be the set of *reference components* in a clustering (the "true" clusters) and $Candidates$ the set of *candidate components* (the clustering produced by the algorithm).

The basic tool for comparing candidate and reference components is the degree of overlap between a candidate and a reference:

$$overlap(R, C) = \frac{|elements(R) \cap elements(C)|}{|elements(R) \cup elements(C)|}$$

The *accuracy* of a matching between two sets of components is then based on the degree of *overlap*:

$$acc\left(\{R_1, ..., R_m\}, \{C_1, ..., C_n\}\right) = overlap(\bigcup_{i=1}^{m} elements(R_i), \bigcup_{j=1}^{n} elements(C_j))$$

Furthermore, in order to identify corresponding subcomponents (e.g. candidates that are similar only to a part of a reference component), the following *partial subset relationship* is used ($0.5 < p \leq 1.0$ is a tolerance parameter, so no candidate can be involved in more than one such matching):

$$C \subseteq_p R \Leftrightarrow \frac{|elements(R) \cap elements(C)|}{|elements(C)|} \geq p$$

Often a clustering algorithm might not be able to identify the reference components exactly. In such cases one reference component might be (partly) covered by more than one candidate component. Thus, to account for different "granularities", for each reference component we compute the set of candidate components which at least partly cover it (note however that $p > 0.5$):

$$matchings(R) = \{C | C \subseteq_p R\}$$

Based on these concepts and abstracting from granularity, we can now define *recall*, which provides a summarizing value for how well the learned clusters "reconstruct" the original classes (i.e. coincide with the reference clustering).

$$recall = \frac{\sum_{R \in References} acc(\{R\}, matchings(R))}{|References|}$$

Entropy: Additionally we also computed *entropy* values for comparing candidate clusterings, using the definitions from [12]. Briefly:

For each cluster C within a candidate clustering and each cluster R in the reference clustering we first calculate the probability that a member of candidate cluster C belongs to reference cluster R. Let this be $p_{C,R}$. The entropy of each candidate cluster is then calculated as follows:

$$E_C = -\sum_R p_{C,R} \log p_{C,R}$$

The entropy of a complete candidate clustering is then calculated as the weighted sum over the entropies of the single clusters (n being the number of instances):

$$E = \sum_C \frac{|elements(C)|}{n} E_C$$

3.2 Comparative Evaluation

As [5] note, "for comparable evaluations of automatic clustering techniques, a common reference corpus [...] is needed for which the actual components are known." Thus, for comparative experiments we used several synthetic and two real-world datasets with different properties. The first synthetic dataset, introduced by Ripley [10], represents a two-dimensional two-class problem. We used the 250 examples from the training set. The class information was ignored in the learning phase, but was used as the reference clustering for the calculation of the recall values. Figure 1 shows the Ripley training set.

Furthermore, we generated two two-dimensional datasets, one of which consisted of four rectangular clusters (two of which slightly overlapping), with 500 instances uniformly distributed within these clusters (figure 2), while the other dataset contained 1000 instances, normally distributed around three centers (figure 3). Finally we generated a dataset with 1000 instances and three classes similar to figure 3, but extended it into four dimensions and used mixed distributions: Two of the classes were normally distributed, the third one uniformly within the bounds of a hyperrectangle.

In addition to the aforementioned synthetic datasets we also used two real-world datasets from the UCI machine learning repository (4 and 13 dimensions, respectively), which were originally intended for classification purposes. In this context we simply interpreted the class labels as reference clusters.[3]

[3] For the purpose of these experiments we assumed that the class distribution within the instance space correlates with a possible clustering of the dataset. Of course, there is no theoretical argument supporting the validity of this assumption.

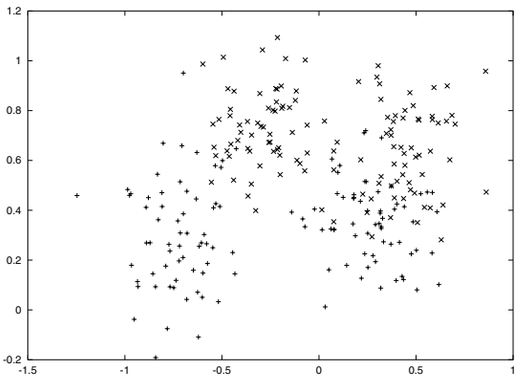

Fig. 1. The Ripley-dataset: two dimensions, two classes

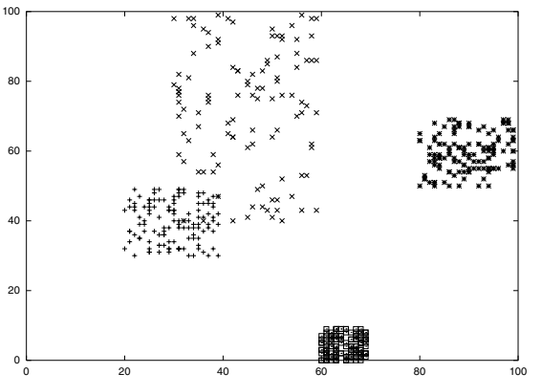

Fig. 2. The Uniform-dataset: two dimensions, four classes

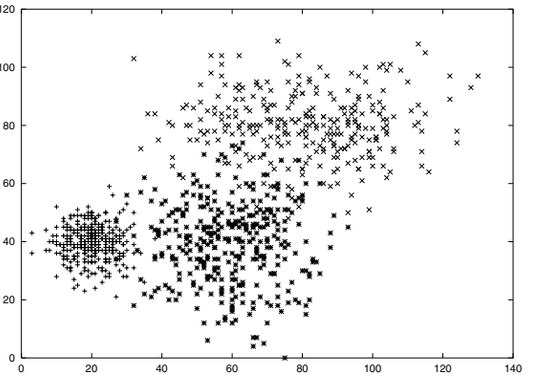

Fig. 3. The Normal-dataset: two dimensions, three classes

Table 1. Comparative results for several synthetic datasets (see text). Listed are *recall* values, *entropies* and number of classes found

dataset	att.	cl.	KMeans	AutoClass	R.U.M.
ripley (250)	2	2	71.07% / 0.32 / 8	76.18% / 0.30 / 6	71.18% / 0.43 / 4
uniform (500)	2	4	97.53% / 0.06 / 7	93.90% / 0.09 / 7	98.73% / 0.03 / 5
normal (1000)	2	3	88.52% / 0.24 / 3	86.89% / 0.22 / 5	90.47% / 0.21 / 3
mixed (1000)	4	3	96.65% / 0.08 / 3	96.67% / 0.08 / 7	94.43% / 0.12 / 4
iris (150)	4	3	80.74% / 0.29 / 3	79.74% / 0.15 / 7	77.14% / 0.27 / 4
wine (178)	13	3	43.33% / 0.65 / 4	94.87% / 0.09 / 4	90.63% / 0.19 / 4

To be able to judge the performance of the MDL evaluation scheme without having to devise a clustering algorithm, we used a stochastic procedure: We generated 100000 random rectangular clusterings by first generating a random number of rectangles (3 to 6), then using these rectangular regions as classifiers[4] and finally evaluating the resulting clusterings by the *r.u.m.length*.

The best so-achieved classification was then compared against the results of AutoClass and KMeans[5]. KMeans is known to converge to a local optimum of its quality measure [11], so it is reasonable to assume that the final result will be a near-optimal clustering according to this measure. We initialized KMeans with the number of initial centroids ranging from 2 to 10 and accepted the best result on the datasets. For AutoClass we used the default parameters. In all of the evaluations we set the tolerance level p to 0.7, as suggested by [5] (see section 3.1). Refer to table 5 for the results.

As can be seen, the granularity (number of clusters found) of the best rectangular clustering according to the *r.u.m.length* is in most cases better (i.e. lower) than the one achieved by AutoClass or KMeans. In addition, the *r.u.m.length* yields clusterings with about the same *recall* and *entropy* levels (no significance tests applied) – in our rather small experimental setting, these are quite promising results.

3.3 Comparing the Measures

Even under the assumption that both KMeans and AutoClass produce clusterings which are (locally) optimal relative to a specific clustering criterion, it might be argued that the effects from the first set of experiments could be due to the massive "search" performed by the stochastic method (versus the deterministic procedures used by KMeans and AutoClass).

Thus, in a second set of experiments, we abstracted from specific algorithms and aimed at a direct comparison of the best clusterings according to

[4] Overlapping regions were treated as separate clusters, so that the number of theoretically possible clusters was much higher than 6.

[5] We used the freely available version by Dan Pelleg [9].

Table 2. Results for comparing the measures *r.u.m.length* against *distortion* (as used by KMeans). Listed are *recall* values, *entropies* and number of classes found

instances	noise level	distortion	R.U.M.
100	05%	86.31% / 0.13 / 4	86.3%1 / 0.13 / 4
100	15%	79.38% / 0.30 / 5	86.25% / 0.24 / 4
100	30%	09.93% / 0.91 / 5	23.00% / 0.92 / 2
1000	05%	70.52% / 0.17 / 5	69.25% / 0.20 / 3
1000	15%	81.43% / 0.28 / 4	80.08% / 0.33 / 4
1000	30%	74.14% / 0.40 / 5	80.36% / 0.38 / 4

the *r.u.m.length* and the measure used by KMeans. After all, what we would like to compare are the clusterings which a suitable algorithm could theoretically produce using the respective measures.

KMeans uses the following quality measure [9]:

$$distortion_{\Phi} = \frac{1}{R} \cdot \sum_{x \in Instances} d^2(x, \Phi(x))$$

with R being the total number of points and Φ representing a clustering, i.e. a mapping which associates a centroid with every instance.

To compare this measure against our MDL-based criterion, we again used our mixed-dataset from section 3.2. In addition we applied background noise (attribute noise) of varying intensity to the data: At a 30% noise level, e.g., 30 percent of the instances were randomly (uniformly) distributed within the instance space. Such examples were not classified into any of the three reference clusters, so that actually there were four different classes now, one of which represented noise.

By the same process as before, we randomly generated a small number of hyperrectangles (3 to 6) within the bounds of the instance space and used these rectangular regions to classify the instances. This time, however, we evaluated the resulting clusterings both by the *r.u.m.length* and by *distortion* as used by the KMeans algorithm. We repeated this process 100000 times and accepted the best clusterings according to each of the two measures as the respective results. Thus, each criterion was tested on exactly the same set of random clusterings.

As can be seen in table 3.3, both measures achieved good *recall* values and *entropies* for low noise settings and gradually deteriorated with increasing noise – unintuitive results (compare the values of 1000/05 against 1000/15) are probably due to the stochastic procedure. In two cases *distortion* achieved a slightly better fit, whereas in most cases *r.u.m.length* was able to select a better candidate clustering. In all of the cases, however, the granularity (number of clusters) produced by *r.u.m.length* was lower than or equal to the one produced by *distortion*, which leads us to the assumption that this measure produces simpler clusterings.

3.4 Scalability

In our opinion, the experiments conducted so far provide evidence that the *clustering message length* does indeed yield good clusterings in terms of simplicity compared to state-of-the-art methods, while hardly losing on the quality in terms of *recall* or *entropy*. We are, however, aware of the fact that in the context of the special case we evaluated in this paper (the *rectangular uniform message length*), scalability to high dimensions is problematic for two reasons:

Firstly, there is no constructive algorithm yet which uses this criterion, and the stochastic procedure we used in the experiments is, of course, not suitable for dealing with very high dimensions, because the number of necessary trials would explode exponentially. Secondly, the restriction to hyperrectangles yields vastly inefficient clusterings, when the reference clusters have more complex shapes. Thus, in higher dimensions, efficiently determining complex cluster shapes (e.g., rotated bounding boxes, halfspaces etc.) becomes more and more important. This is one of our future research directions.

4 Summary and Further Work

Again, we would like to make it clear that what we presented is not a complete clustering algorithm itself, but an MDL-based clustering criterion which could enable a suitable learner to find the theoretically best clustering based on space descriptions. Additionally, the *clustering message length* yields a non-parametric criterion for the appropriate number of clusters in a database.

Experiments in synthetical and real-world datasets show that the best clustering according to a special case of this criterion in most cases outperforms the clusterings found by AutoClass and KMeans in terms of granularity, while achieving about the same *recall* and *entropy* values. In addition, we directly compared our criterion against *distortion* (the measure used by KMeans) and achieved favorable results. Furthermore we provided a theoretical proof that this measure – the *rectangular uniform message length* – is well-behaved.

Thus, in our – admittedly – rather small experimental setting we could produce quite encouraging results. One of our current research topics is the theoretical and practical evaluation of the *clustering message length* in larger settings, the extension to different cluster shapes and the development of a clustering algorithm which optimizes exactly this criterion.[6]

Finally, we would like to mention that – in contrast to measures based on the normal distribution – the *r.u.m.length* assigns lower (i.e. better) ratings to clusterings whose boundaries coincide with "edges" in the instance space, where significant changes in the density distribution occur. This criterion could therefore be used as a non-parametric way for detecting "edges" between areas of different densities. An application in a multivariate discretization algorithm,

[6] The basic idea could be to start from an estimated centroid (density center) of the dataset and linearly extend the scope of the cluster to include more and more instances, while calculating the *clustering message length* for each step.

where such a module is of crucial importance [6] is another one of our current research topics.

Acknowledgements

This research is supported by the Austrian *Fonds zur Förderung der Wissenschaftlichen Forschung (FWF)* under grant no. P12645-INF. We would like to thank Johannes Fürnkranz and Johann Petrak for helpful discussions and Markus Mottl for his help on the programming language OCaml.

References

1. R. A. Baxter and J. Oliver. MDL and MML: Similarities and differences. Technical report, Dept. of Computer Science, Monash University, Clayton, 1994. (TR 207). 259
2. P. S. Bradley and U. M. Fayyad. Refining initial points for k-means clustering. In *Proceedings of the* 15^{th} *Int. Conference on Machine Learning*, 91–99, 1998. 259
3. P. Cheeseman, J. Kelly, M. Self, J. Stutz, W. Taylor, and D. Freeman. Autoclass: A bayesian classification system. In *Proceedings of the* 5^{th} *International Workshop on Machine Learning*, 54–64, 1988. 259
4. D. Keim and A. Hinneburg. Clustering techniques for large data sets: From the past to the future. In *Tutorial Notes for ACM SIGKDD 1999 International Conference on Knowledge Discovery and Data Mining*, San Diego, CA, 1999. 258
5. R. Koschke and T. Eisenbarth. A framework for experimental evaluation of clustering techniques. In *Proceedings of the International Workshop on Program Comprehension (IWPC2000)*, Limerick, Ireland, 2000. IEEE. 263, 264, 266
6. M.-C. Ludl and G. Widmer. Relative unsupervised discretization for association rule mining. In *Proceedings of the* 4^{th} *European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD2000)*, Lyon, 2000. 269
7. A. Y. Ng, M. Jordan, and Y. Weiss. On spectral clustering: Analysis and an algorithm. In *Proceedings of NIPS 14*, 2002. (to appear). 258
8. J. J. Oliver, R. A. Baxter, and C. S. Wallace. Unsupervised learning using MML. In *Proceedings of the* 13^{th} *International Conference on Machine Learning*, 364–372, San Francisco, CA, 1996. Morgan Kaufmann. 258
9. D. Pelleg and A. Moore. Accelerating exact k-means algorithms with geometric reasoning. In *Proceedings of the* 5^{th} *International Conference on Knowledge Discovery and Data Mining (KDD99)*, 277–281, 1999. 266, 267
10. B. D. Ripley. Pattern recognition and neural networks. *Statistics*, 33:1065–1076, 1996. 264
11. S. Z. Selim and M. A. Ismail. K-means-type algorithms: A generalized convergence theorem and characterization of local optimality. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6(1):81–87, 1984. 266
12. M. Steinbach, G. Karypis, and V. Kumar. A comparison of document clustering techniques. In *KDD Workshop on Data Mining*, 2000. 264
13. C. S. Wallace and D. L. Dowe. Intrinsic classification by MML – the SNOB program. In *Proceedings of the* 7^{th} *Australian Joint Conference on Artificial Intelligence*, 37–44, Singapore, 1994. World Scientific. 259

14. Y. Weiss. Segmentation using eigenvectors: A unifying view. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV99)*, 975–982, 1999. 258