COOLCAT: An entropy-based algorithm for categorical clustering [*]

Daniel Barbará
George Mason University
ISE Department
MSN 4A4
Fairfax, VA 22030
dbarbara@gmu.edu

Julia Couto
James Madison University
ISAT Department
P.O. Box 1212
Harrisonburg, VA 22807
coutoji@jmu.edu

Yi Li
George Mason University
ISE Department
MSN 4A4
Fairfax, VA 22030
yli1@gmu.edu

ABSTRACT

In this paper we explore the connection between clustering categorical data and entropy: clusters of similar poi lower entropy than those of dissimilar ones. We use this connection to design an incremental heuristic algorithm, COOLCAT, which is capable of efficiently clustering large data sets of records with categorical attributes, and data streams. In contrast with other categorical clustering algorithms published in the past, COOLCAT's clustering results are very stable for different sample sizes and parameter settings. Also, the criteria for clustering is a very intuitive one, since it is deeply rooted on the well-known notion of entropy. Most importantly, COOLCAT is well equipped to deal with clustering of *data streams* (continuously arriving streams of data point) since it is an incremental algorithm capable of clustering new points without having to look at every point that has been clustered so far. We demonstrate the efficiency and scalability of COOLCAT by a series of experiments on real and synthetic data sets.

Categories and Subject Descriptors

I.5.3 [**Computing Methodologies**]: Pattern Recognition—*Clustering*

General Terms

Entropy

Keywords

clustering, entropy, data streams

1. INTRODUCTION

Clustering is a widely used technique in which data points are partitioned into groups, in such a way that points in the same group, or cluster, are more similar among themselves than to those in other clusters. Clustering of categorical attributes (i.e., attributes whose domain is not numeric) is a difficult, yet important task: many fields, from statistics to psychology deal with categorical data. In spite of its importance, the task of categorical clustering has received scant attention in the KDD community as of late, with only a handful of publications addressing the problem ([18, 14, 12]).

Much of the published algorithms to cluster categorical data rely on the usage of a distance metric that captures the separation between two vectors of categorical attributes, such as the Jaccard coefficient. In this paper, we present COOLCAT (the name comes from the fact that we reduce the entropy of the clusters, thereby "cooling" them), a novel method which uses the notion of entropy to group records. We argue that a classical notion such as entropy is a more natural and intuitive way of relating records, and more importantly does not rely in arbitrary distance metrics. COOLCAT is an incremental algorithm that aims to minimize the expected entropy of the clusters. Given a set of clusters, COOLCAT will place the next point in the cluster where it minimizes the overall expected entropy. COOLCAT acts incrementally, and it is capable to cluster every new point without having to re-process the entire set. Therefore, COOLCAT is suited to cluster data streams (continuosly incoming data points) [2]. This makes COOLCAT applicable in a large variety of emerging applications such as intrusion detection, and e-commerce data.

This paper is set up as follows. Section 2 offers the background and relationship between entropy and clustering, and formulates the problem. Section 3 reviews the related work. Section 4 describes COOLCAT, our algorithm. Section 5 presents the experimental evidence that demonstrates the advantages of COOLCAT. Finally, Section 6 presents conclusions and future work.

2. BACKGROUND AND PROBLEM FORMULATION

In this section, we present the background of entropy and clustering and formulate the problem.

2.1 Entropy and Clustering

Entropy is the measure of information and uncertainty of a random variable [28]. Formally, if X is a random variable, $S(X)$ the set of values that X can take, and $p(x)$ the prob-

[*]This work has been supported by NSF grant IIS-0208519

ability function of X, the entropy $E(X)$ is defined as shown in Equation 1.

$$E(X) = - \sum_{x \in S(X)} p(x)log(p(x)) \quad (1)$$

The entropy of a multivariate vector $\hat{x} = \{X_1, \cdots, X_n\}$ can be computed as shown in Equation 2, where $p(\hat{x}) = p(x_1, \cdots, x_n)$ is the multivariate probability distribution.

$$E(\hat{x}) - \sum_{x_1 \in S(X_1)} ... \sum_{x_n \in S(X_n)} p(\hat{x})logp(\hat{x}) \quad (2)$$

Entropy is sometimes referred to as a measure of the amount of "disorder" in a system. A room with socks strewn all over the floor has more entropy than a room in which socks are paired up, neatly folded, and placed in one side of your sock and underwear drawer.

2.2 Problem formulation

The problem we are trying to solve can be formulated as follows. Given a data set D of N points $\hat{p_1}, \cdots, \hat{p_N}$, where each point is a multidimensional vector of d categorical attributes, i.e., $\hat{p_j} = (p_j^1, \cdots, p_j^d)$, and given an integer k, we would like to separate the points into k groups $C_1, \cdots, C_k$, or clusters, in such a way that we minimize the entropy of the whole arrangement. Unfortunately, this problem is NP-Complete, and moreover, difficult to approximate [13]. In fact, the problem is NP-Complete for *any* distance function $d(x, y)$, defined over pairs of points x, y, such that the function maps pairs of points to real numbers (and hence, our entropy function qualifies), therefore we need to resort to heuristics to solve it.

We first have to resolve the issue of what we mean by the "whole entropy of the system." In other words, we have to make our objective function clear. We aim to minimize the expected entropy, whose expression is shown in Equation 3, where $E(C_1), \cdots, E(C_k)$, represent the entropies of each cluster, C_i denotes the points assigned to cluster i, $C_i \subset D$, with the property that $C_i \cap C_j = \emptyset$, for all $i, j = 1, .., k \;\; i \neq j$. The symbol $\breve{C} = \{C_1, \cdots, C_k\}$ represents the clustering.

$$\bar{E}(\breve{C}) = \sum_k (\frac{|C_k|}{|D|}(E(C_k))) \quad (3)$$

This function, as we will see later, allows us to implement an incremental algorithm that can effectively deal with large datasets, since we do not need to look at the entire set of points to decide about the entropy of an arrangement. Rather, we will be able to decide for each point, how it would affect the entropy of each of the existing clusters if placed in each one of them.

The solution we propose in this paper (and present in Section 4) is a heuristic based in finding a set of initial clusters (using the entropic criteria), and then incrementally (greedily) add points to the clusters according to a criteria that minimizes Equation 3.

Furthermore, we make a simplification in the computation of entropy of a set of records. We assume independence of the attributes of the record, transforming Equation 2 into Equation 5. In other words, the joint probability of the combined attribute values becomes the product of the probabilities of each attribute, and hence the entropy can be calculated as the sum of entropies of the attributes.

	members	E	Exp. Entropy
Cluster0	{"*red*","*heavy*"} {"*red*","*medium*"}	1.0	0.66
Cluster1	{"*blue*","*light*"}	0	
Cluster0	{"*red*","*heavy*"} {"*blue*","*light*"}	2.0	1.33
Cluster1	{"*red*","*medium*"}	0	
Cluster0	{"*red*",*heavy*"}	0	1.33
Cluster1	{"*red*","*medium*"} {"*blue*","*light*"}	2.0	

Figure 1: Three different clusterings for the set v_1, v_2, v_3. Clustering 1 minimizes the expected entropy of the two clusters.

$$\begin{aligned} E(\hat{x}) &= - \sum_{x_1 \in S(X_1)} \cdots \sum_{x_n \in S(X_n)} \quad (4) \\ &\quad \prod_i (p(x_i))log(\prod_i p(x_i)) \\ &= E(X_1) + E(X_2) + \cdots + E(X_n) \quad (5) \end{aligned}$$

Assume that we have a set of three records, $v_1 = \{$"*red*" ,"*heavy*"$\}$, $v_2 = \{$"*blue*","*light*"$\}$, and $v_3 = \{$"*red*", "*medium*"$\}$, and we want to form two clusters with them. Figure 1 shows all the possible arrangements, with the entropy of each cluster, and the expected entropy in each arrangement. As we can see, the minimum expected entropy is that of arrangement 1, which obviously is the correct way of clustering the records (using two clusters).

Even though the assumption of attribute independence is not true in every data set, it proves to work very well in practice (as shall be shown in the experimental section of this paper). Moreover, in the cases we can demonstrate that there is a correlation between two or more attributes of the data set, we can always change the data points by creating attributes that reflect these correlations and then apply Equation 5 to compute the join entropy. For instance, if the data set is composed of records of attributes A, B, C, D, E, F and we know that (A, B), (A, C) and (E, F) are correlated. we can convert the data set into one having records with attributes AB, AC, D, EF and compute the entropy assuming that these new attributes are independent. Notice that for the grouped attributes, we are in effect computing their joint probabilities. The correlations between attributes can be easily found by techniques such as the Chi-Square and likelihood ratio tests. In our experimental experience, the gains obtained by doing this are small enough to justify the usage of the independence assumption.

2.3 Expected entropy and the Minimum Description Length principle

The Minimum Description Length principle (MDL) [26, 27] recommends choosing the model that minimizes the sum of the model's algorithmic complexity and the description of the data with respect to that model. This principle is widely used to compare classifiers (see [23]) but it has not been used much to deal with clustering.

Formally, the complexity of the model can be stated as shown in Equation 6, where $K()$ indicates the complexity, h is the model, and D denotes the data set. The term $K(h)$ denotes the complexity of the model, or model encoding, while $K(D \text{ using } h)$ is the complexity of the data encoding with respect to the chosen model.

$$K(h, D) \;=\; K(h) + K(D \text{ using } h) \tag{6}$$

Consider first the term $K(h)$. To encode the model, we need to encode for each cluster the probability distribution for the attribute values. This can be done by encoding the number of times each attribute value appears in the cluster, and the number of points in each cluster. Assuming that there are d attributes in the data, and that attribute A_j can assume v_j different values. As usual, k represents the number of clusters in the model. $K(h)$ can be written as shown in Equation 7. In each cluster i, we need to encode $c_i \;=\; \sum_{j=0}^{d-1} v_j$ values. So, the total number of values we need to encode is $\sum_{i=0}^{k-1} c_i \;=\; \beta k$, where β is a constant. We also need to encode the number of points in each cluster, or k values. The number of bits needed to encode the number of times each attribute value occurs in the cluster, or the number of points in a cluster is equal to $log(|D|)$, since the maximum number for these values is the size of the entire data set. Therefore $K(h)$ is a linear function of k, with α a constant that represents all the contributions described above.

$$K(h) \;=\; \alpha k log(|D|) \tag{7}$$

On the other hand, the encoding of the data given the model can be stated as shown in Equation 8. Once the probabilities of occurrence of each attribute value in each cluster are known, an optimal code (Huffman) can be chosen to represent each attribute value in the cluster. Each point is simply represented by the encoding of its attributes' values. The optimal code is achieved by giving to each value a number of bits proportional to $log(P_{ijl})$, where $P(_{ijl})$ is the probability that the l value of attribute j occurs in cluster i. The second term in the equation simply indicates the membership of all the points, needing $log(k)$ for the encoding of the individual memberships.

$$K(D \text{ using } h) \;=\; \sum_{i=0}^{k-1} \frac{|C_i|}{|D|} \sum_{j=0}^{d-1} \sum_{l=0}^{v-1} P_{ijl} log(P_{ijl}) + D log(k) \tag{8}$$

Noticing that the first term of Equation 8 is simply the expected entropy of the clustering, we can write $K(h, D)$ as shown in Equation 9. Notice that for a fixed k, the MDL principle indicates that the best model can be found by minimizing the expected entropy of the clustering, which is precisely our goal.

$$K(h, D) \;=\; \alpha log(|D|) + D log(k) + \bar{E}(\check{C}) \tag{9}$$

2.4 Evaluating clustering results

A frequent problem one encounters when applying clustering algorithms in practice is the difficulty in evaluating the solutions. Different clustering algorithms (and sometimes multiple applications of the same algorithm using slight variations of initial conditions or parameters) result in very different solutions, all of them looking plausible. This stems from the fact that there is no unifying criteria to define clusters, and more often than not, the final clusters found by the algorithm are in fact the ones that correspond to the criteria used to drive the algorithm. Methods to evaluate whether or not the structure found is a property of the data set and not one imposed by the algorithm are needed.

Authors have pondered about good ways to validate clusters found by algorithms (e.g., see [21, 1]). Two widely used methods are the following:

- *Significance Test on External Variables* This technique calls for the usage of significance tests that compare the clusters on variables not used to generate them. One way of doing this is to compute the entropy of the solution using a variable that did not participate in the clustering. (A class attribute.) The entropy of an attribute C in a cluster C_k is computed as shown in Equation 10, where V_j denotes one of the possible values that C can take. The evaluation is performed by computing the expected entropy (taken into consideration the cluster sizes). The smaller the value of $E(C_k)$, the better the clustering fares.

$$E(C_k) \;=\; \sum_j P(C \;=\; V_j) log P(C \;=\; V_j) \tag{10}$$

- *The category utility function* The category utility (CU) function [15] attempts to maximize both the probability that two objects in the same cluster have attribute values in common and the probability that objects from different clusters have different attributes. The expression to calculate the expected value of the CU function is shown in Equation 11, where $P(A_i \;=\; V_{ij}|C_k)$ is the conditional probability that the attribute i has the value V_{ij} given the cluster C_k, and $P(A_i \;=\; V_{ij})$ is the overall probability of the attribute i having the value V_{ij} (in the entire set). The function aims to measure if the clustering improves the likelihood of similar values falling in the same cluster. Obviously, the higher the value of CU, the better the clustering fares.

$$CU \;=\; \sum_k \frac{\|C_k\|}{|D|} \sum_i \sum_j [P(A_i = V_{ij}|C_k)^2 - P(A_i = V_{ij})^2] \tag{11}$$

We have used both techniques in validating our results, as shall be seen in the experimental section.

2.5 Number of clusters

The issue of choosing the number of clusters is one common to all clustering methods, and our technique is no exception. Many methods have been proposed for determining the right number of clusters (e.g.,[4, 9]). Unfortunately many of these methods (e.g., [4]) assume that it is possible to compute a centroid for each cluster, which in categorical data is not easy. We consider this issue out of the scope of this paper since we plan to examine good ways of selecting the optimal number of clusters in the context of our metric.

3. RELATED WORK

Clustering is an extensively researched area not only by data mining and database researchers [31, 11, 17, 18, 3], but also by people in other disciplines [10]. Among the numerical clustering algorithms, ENCLUS [6] uses entropy as a criteria to drive the algorithm. However, ENCLUS follows a completely different algorithm to our approach, dividing the hyperspace recursively. For each subspace, ENCLUS estimates its density and entropy and determines if it satisfies the goodness criteria: its entropy has to be lower than a threshold. However, it is not possible to translate either the algorithm or the relationships to the area of categorical clustering, since the notion of density has no intuitive meaning when the attributes are categorical. In a recent paper [16], the authors use Renyi's definition of entropy [25] to define a clustering evaluation function that measures the distance between clusters as the *information potential* [24] between them. Using this function, they describe an algorithm that, starting with a random placing of points in clusters, perturbs the placement until the improvement on the information potential is not appreciable. This algorithm, however, cannot scale to large data sets since it requires all points to perform the calculation of the distance.

In the area of clustering categorical records, a few recent publications are worth mentioning. In [19], the authors address the problem of clustering transactions in a market basket database by representing frequent item sets as hyperedges in a weighted hypergraph. The weight of the graph is computed as the average of the confidences for all possible association rules that can be generated from the item set. Then, a hypergraph partitioning algorithm is employed to partition the items, minimizing the weight of the cut hyperedges. The algorithm does not produce a clustering of the transactions and it is not obvious how to obtain one from the item clusters. A related paper by Gibson et al [14] also treats categorical clustering as hypergraph partitioning, but uses a less combinatorial approach to solving it, based on non-linear dynamical systems.

CACTUS [12], is an agglomerative algorithm that uses the author's definitions of *support*, *strong connection* and *similarity* to cluster categorical data. Support for an attribute value pair (a_i, a_j), where a_i is in the domain of attribute A_i and a_j in the domain of attribute A_j is defined as the number of tuples that have these two values. The two attributes a_i, a_j are strongly connected if their support exceeds the value expected under the attribute-independence. This concept is then extended to sets of attributes. A cluster is defined as a region of attributes that are pairwise strongly connected, no sub-region has the property, and its support exceeds the expected support under the attribute-independence assumption.

ROCK [18] computes distances between records using the Jaccard coefficient. Using a threshold, it determines, for each record, who are its neighbors. For a given point p, a point q is a neighbor of p if the Jaccard coefficient $J(p,q)$ exceeds the threshold. Then, it computes the values of a matrix $LINK$, in which the entries $link(p,q)$ are the number of common neighbors between p and q. The algorithm then proceeds to cluster the records in an agglomerative way, trying to maximize for the k clusters (k is a predefined integer) the function $\sum_{i=1}^{k} n_i \sum_{p,q \in C_i} \frac{link(p,q)}{n_i^{1+2f(\phi)}}$, where ϕ is the threshold, and $f(\phi)$ is a function selected by the user. The choice of $f(\phi)$ is critical in defining the fitness of the clusters formed the the ROCK algorithm, and, as the authors point out, the function is dependent on the data set as well as on the kind of cluster the user is interested in. We feel that choosing the function is a delicate and difficult task for users that may be a roadblock to using ROCK efficiently.

Snob [29, 30] is an unsupervised learning algorithm based on the notion of Minimum Message Length (MML). MML is an information theoretic criterion for parameter estimation and model selection. Although MML is similar to the MDL criterion of Rissanen, MML is a Bayesian criterion and therefore uses an a-priori distribution of parameter values. Snob is in the category of mixture model algorithms [22]. Snob is iterative in nature and therefore does not scale with large data sets. Moreover, contrary to COOLCAT, it is difficult to envision how Snob can be used to cluster data streams. AUTOCLASS [5] also uses mixture models and Bayesian criteria to cluster data sets. Again, AUTOCLASS does not scale well with large data sets.

4. OUR ALGORITHM

Our entropy-based algorithm, COOLCAT, consists of two steps: initialization and incremental step.

4.1 Initialization

The initialization step "bootstraps" the algorithm, finding a suitable set of clusters out of a sample S, taken from the data set ($|S| << N$), where N is the size of the entire data set. We first find the k most "dissimilar" records from the sample set by maximizing the minimum pairwise entropy of the chosen points. We start by finding the two points ps_1, ps_2 that maximize $E(ps_1, ps_2)$ and placing them in two separate clusters (C_1, C_2), marking the records (this takes $O(|S|^2)$). From there, we proceed incrementally, i.e., to find the record we will put in the j-th cluster, we choose an unmarked point ps_j that maximizes $min_{i=1,..,j-1}(E(ps_i, ps_j))$.

The rest of the sample unmarked points ($|S| - k$), as well as the remaining points (outside the sample), are placed in the clusters using the incremental step.

We are interested in determining the size of the sample that guarantees with high probability the existence in the sample of at least one member of each cluster, given the number of clusters. In [17], the authors address the same problem and use Chernoff bounds[7] to bound the size of the sample given an estimate of the size of the smallest cluster with respect to the average size ($\frac{|D|}{k}$), and the confidence level δ for the probability of finding at least a member of each cluster. The estimate of the size of the smallest cluster with respect to the average size is given in the form of a parameter $\rho = \frac{\frac{|D|}{k}}{m}$, where m is the size of the smallest cluster. The parameter ρ is then a number greater than 1. The bound on the size of the sample is then given by Equation 12.

$$s \;=\; k\rho + k\rho log(\frac{1}{\delta}) + k\rho\sqrt{(log(\frac{1}{\delta}))^2 + 2log(\frac{1}{\delta})} \qquad (12)$$

It is important to remark that Equation 12 **does not** depend on the size of the data set, which makes the bound

```
1.Given an initial set of clusters $\breve{C} = C_1, \cdots, C_k$
    2.Bring points to memory from disk and
    for each point $p$ do
        3. For $i = 1, .., k$
            4. Tentatively place $p$ in $C_i$ and
            compute $\bar{E}(\breve{C}^i)$ where $\breve{C}^i$
            denotes the clustering obtained
            by placing $p$ in cluster $C_i$
            5. Let $j = argmin_i(\bar{E}(\breve{C}^i))$
        6. Place $p$ in $C_j$
    7. Until all points have been placed in
    some cluster
```

Figure 2: Incremental step.

very favorable for larger sets (and unfavorable for small ones, but this is not a problem since for small sets we can simply use the entire set as a sample).

4.2 Incremental Step

After the initialization, we process the remaining records of the data set (the rest of the sample and points outside the sample) incrementally, finding a suitable cluster for each record. This is done by computing the expected entropy that results of placing the point in each of the clusters and selecting the cluster for which that expected entropy is the minimum. We proceed in the incremental step by bringing a buffer of points to main memory and clustering them one by one.

The order of processing points has a definite impact on the quality of the clusters obtained. It is possible that a point that seems a good fit for a cluster at a given point in time, becomes a poor fit as more points are clustered. In order to reduce this effect, we enhanced the heuristic by re-processing a fraction of the points in the batch. After a batch of points is clustered, we select a fraction m of points in the batch that can be considered the worst fit for the clusters they were put in. We proceed to remove these points from their clusters and re-cluster them. The way we figure out how good a fit a point is for the cluster where it landed originally, is by keeping track of the number of occurrences of each of its attributes' values in that cluster. That is, at the end of the batch, we know the values of q_{ij}, for each record i in the batch and each attribute j, where q_{ij} represent the number of times that the value V_{ij} appears in the cluster where i was placed. We convert these numbers into probabilities by dividing q_{ij} by the cluster size (i.e., $\|C_l\|$, where C_l is the cluster where i was placed). Let us call these numbers p_{ij}. For each record, we can compute a fitting probability $p_i = \prod_j (p_{ij})$. Notice that the lower the p_i is, the worst fit the record is in that cluster (we can say that the global combination of attributes is not very common in the cluster). We then sort records according to p_i and select the m records in the batch with lowest p_i as the records to be reprocessed. Each re-processed record is placed in the cluster that minimizes the expected entropy (as done originally in the incremental step).

5. EXPERIMENTAL RESULTS

Our experiments were run in a DELL server equipped with a Pentium III running at 800 MHz, and 1 Gigabyte of main memory, running Red Hat Linux 2.2.14. We used two kinds of data sets: real data sets (for evaluating the quality of our algorithm) and synthetic data sets (for the evaluation of scalability). The experiments were conducted using the following datasets (plus a synthetically generated data set to test the scalability of the algorithm).

- *Archaeological data set*

  Our first data set is a hypothetical collection of human tombs and artifacts from an archaeological site. Although the data set is not "real," it is realistic enough and so we include it in this section. It has also the property of being small, so brute force can be used to find the optimal clustering. The data set is taken from [1] The first attribute (not used for clustering but for verification) indicates the sex (M for male, F for female) of the individuals buried. The other eight attributes are binary (1 present, 0 non-present), and represent artifacts types (e.g., ceramics, bracelets, arrow points) that were found (or not found) in the tomb.

- *Congressional votes* This data set was obtained from the UCI KDD Archive ([20]) and contains the United States Congressional Voting Records for the year 1984. Each record contains a Congressman's votes on 16 issues. All the attributes are boolean ("yes" or "no"), with a few of the votes containing missing values. We decided to treat missing values as another domain value for the attribute. A classification field with the labels "Democrat," or "Republican" is provided for each record, which are not used for clustering, but can be loosely used for quality measuring. (Some congressmen "crossed" parties to vote.) There are 435 records in the set (267 Democrats and 168 Republicans).

- KDD Cup 1999 data This data set can be obtained from the UCI Archive [20], and was used for the the Third International Knowledge Discovery and Data Mining Tools Competition. This database contains a standard set of network audit data, which includes a wide variety of simulated intrusions. Each record, corresponding to a connection, contains 42 features, some of them categorical, and the rest continuous variables. We transformed the continuous variables in categorical by a simple process of discretization: we computed the median of each attribute, and assigned any value below and including the median a label "0," while the rest of the values were assigned a label "1." There are many intrusion data sets in the repository, some of them to be used as training sets and some as test sets. We utilized the set that corresponds to 10% of the training data. In this set, records have an extra attribute (class), labeled with a "1" if the connection is part of an attack, or a "0" if it is not. We use this attribute for the evaluation of external entropy (not in the clustering process).

5.1 Archaeological Data

Figure 3 show the results of using COOLCAT in the archaeological data set. We performed experiments with 2 clusters, since the attribute with which we evaluate the external entropy (not used in the clustering) is Sex (and the

Alg.	m	CU	Ext E. (sex)	Expected entropy
COOLCAT	0%	0.7626	0	4.8599
	10%	0.7626	0	4.8599
	20%	0.7626	0	4.8599
Brute Force	-	0.7626	0	4.8599
ROCK	-	0.3312	0.9622	n/a

Figure 3: Results for COOLCAT, ROCK and brute force in the Archaeological data set.

Alg.	m	CU	Ext.Ent. (pol. affl.)	Expected entropy	Running time (sec.)
COOL CAT	0%	2.9350	0.4975	13.8222	0.16
	10%	2.9350	0.4975	13.8222	0.26
	20%	2.9350	0.4975	13.8222	0.28
ROCK	-	2.6282	0.4993	N/A	0.51

Figure 4: Results for COOLCAT and ROCK in the Congressional Voting data set

data set is small, so we believed that the clustering could effectively separate the two sexes). We conducted experiments with the original data set (which we label "independent"), and a modified data set in which we grouped attributes in the following way: (1), (24), (26), (34), (35), (46), (78), to reflect the correlations found among the attributes of the set (found by using a Likelihood ratio test). However, we only report the results for independent data, since the correlated set results are essentially the same. (The same phenomena was observed in the other experiments.) We also conducted "brute force" experiments, in which we found the optimum clustering, i.e., that for which the expected entropy was the minimum. We did this to compare how well our heuristic (COOLCAT) performed. We also report in the table the best results found by ROCK (which have to be found by varying the parameter ϕ over a range of values). The results shown in Figure 3 show that the expected entropy function does an excellent job in clustering this data. The results obtained by COOLCAT (in terms of CU, and external entropy with respect to the variable sex, which is not used in the clustering), and expected entropy are the same obtained by the brute force (optimal) approach. In all cases, both the CU function and the external entropy of the COOLCAT solutions are better than those found for the best ROCK solution. Particularly encouraging is the fact that the external entropy for the variable SEX (which the authors of the data set indicated as the one being more correlated with the clusters), is 0 in all the COOLCAT solutions, so a perfect separation is achieved. (ROCK's solution does not achieve this, resulting in a high external entropy.) In this data set, the re-processing step does not have any effect, as seen by the fact that the results are the same for all the values of m. This is attributed to the size of the data set (only 20 records). Both COOLCAT and ROCK took 0.01 seconds to find a solution for this data set.

5.2 Congressional Voting results

Figure 4 summarizes the results obtained by COOLCAT in the Congressional Voting records (no grouping of attributes was performed), for three values of m. The results obtained for various sample sizes are extremely stable. The CU values for the clusterings obtained with COOLCAT are, in all the cases superior to the one obtained by ROCK. The values show no fluctuations on our results as m changes, while the value for CU is 11% better than ROCK's value. The external entropy for the COOLCAT solutions is slightly better than the value in ROCK's solution. The buffer size (batch) in this experiment was 100 records, making the number of re-processed points 0,10, and 20 ($m = 0\%, 10\%, 20\%$). (Again, these numbers correspond to the means of 500 runs.) The running time of COOLCAT is significantly better than the one for ROCK (a decrease of 45% in the slowest case, $m = 20\%$, of COOLCAT).

5.3 KDD Cup 1999 data set

Since we did not have explicit knowledge of how many clusters we could find in this data set, we decided to find clusterings for many k values, and report, in each case, the expected entropy, external entropy (with respect to the attribute that denotes whether the record is an attack or not), and CU. The results are shown in the form of a graph in Figure 5. In the figure, the left hand side scale is used for expected entropy and CU, while the right hand side is used for external entropy (the values of external entropy are between 0 and 1, while the other parameters have larger ranges). The figure shows that all the parameters tend to an asymptotic limit as k grows. The saturation starts to occur in the value $k = 10$, which exhibits an external entropy of 0.09, which indicates that most of the clusters are "inhabited" by either attack records or attack-free records. In other words, the clustering achieves a good separation of the points. The experiments were conducted using a sample size of 1,000 points, which guarantees a level of confidence of 95% (for $\rho = 10$).

5.4 Synthetic data set

We used a synthetic data generator ([8]) to generate data sets with different number of records and attributes. We used these data sets to test the scalability of COOLCAT. The results are shown in the graph of Figure 7, where the y-axis shows the execution time of COOLCAT in seconds, and the x-axis the number of records (in multiples of 10^3), for four different number of attributes ($A = 5, 10, 20, 40$). In all the cases, COOLCAT behaves linearly with respect to the number of records, due to the incremental nature of the algorithm (it processes each record in the data set at most twice: those that are selected for re-processing are clustered twice, the rest only once; moreover, points are brought from disk to memory only once). We used for these experiments an m equal to 20%, and a buffer size of 300 records. Notice that in this experiment, we do not report running times for ROCK. The reason for this is that ROCK is designed to be a main memory algorithm. In [18], the authors make it explicit that ROCK deals with large data sets by using random sampling (not by looking at the entire set). Therefore, it would have been unfair to compare COOLCAT's running times with those of ROCK (over samples of the sets).

We performed another experiment with synthetic data

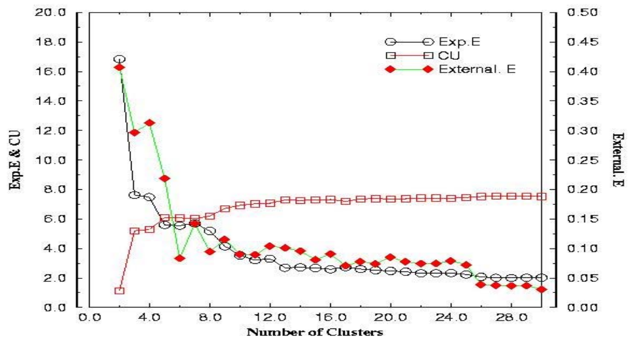

Figure 5: Expected entropy, external entropy and CU vs. Number of Clusters (k) in the KDD Cup 1999 data set. The left scale (y-axis) is used for xpected entropy and CU, while the right one is used for external entropy.

sets generated by [8]. In this experiment, each synthetic set contained 8,124 records of 23 attributes each. Twenty two of the attributes are used for clustering and one ($Indx$) for the evaluation of external entropy. Each data set was generated using 21 different types of rules. A rule involves 12 attributes. An example of the rules used is: $A = c\&C = a\&D = b\&K = c\&N = a\&O = c\&Q = b\&R = c\&S = c\&T = b\&U = b \Rightarrow Indx = r_1$. This rule says that when the 11 attributes on the left hand side take the values shown, the attribute $Indx$ takes the value r_1. Every record obeys one of the rules. Two sets were generated, using different probability distributions for the rules. In the first one (uniform), every rule is used in the same number of records in the data set. (In other words the number of records that obey a particular rule is equal to the size of the data set divided by 21.) In the normal distribution, the populations are distributed following a Gaussian distribution (some rules receive more records than others). The 23rd attribute takes the value of the rule number (rule index). The external entropy is calculated using this attribute (which does not participate in the clustering). Figure 6 shows the evaluation of clusters obtained by COOLCAT over different synthetic data sets. The table shows also the results obtained by using ROCK. As we can see, COOLCAT results are significantly better than those obtained by ROCK for both data sets. Particularly significant is the fact that the external entropy for the COOLCAT solutions in the Uniform case with $m = 10\%, 20\%$ are 0, indicating a perfect separation of rules. The values for other cases are extremely close to 0 as well. As expected, re-processing (increasing m) helps in finding a better clustering. However, the impact is more marked when going from no re-processing ($m = 0$) to re-processing 10% of the points, leveling out from then on. The running times of COOLCAT are more than one order of magnitude smaller than those of ROCK.

6. CONCLUSIONS

In this paper we have introduced a new categorical clustering algorithm, COOLCAT, based in the notion of entropy. The algorithm groups points in the data set trying to minimize the expected entropy of the clusters. The ex-

Dist.	m	CU	Ext.Ent. (rule index)	Expected entropy	Running time (sec.)
COOLCAT					
Uniform	0%	6.9187	0.00816	17.4302	6.73
	10%	6.9268	0.00000	17.2958	11.85
	20%	6.9268	0.00000	17.3958	12.95
Normal	0%	6.8893	0.02933	17.4969	6.88
	10%	6.8996	0.00813	17.4458	11.99
	20%	6.9008	0.00742	17.4328	13.07
ROCK					
Uniform	-	6.6899	0.09861	n/a	207.37
Normal	-	6.2749	0.34871	n/a	223.49

Figure 6: Results for COOLCAT and ROCK in the synthetic data sets

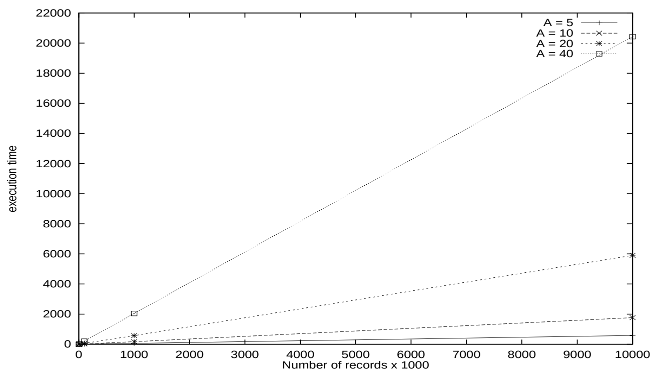

Figure 7: COOLCAT's performance for the synthetic data sets: response time (in seconds) vs. the number of records in the data set (in multiples of 10^3), for different number of attributes ($A = 5, 10, 20, 40$).

perimental evaluation supports our claim that COOLCAT is an efficient algorithm, whose solutions are stable for different samples (and sample sizes) and it is scalable for large data sets (since it incrementally adds points to the initial clusters). We have evaluated our results using category utility function, and the external entropy which determines if the clusters have significance with respect to external variables (i.e., variables not used in the clustering process). In our comparisons with ROCK, COOLCAT always shows a small advantage in terms of the quality measures (CU and external entropy). However, the real advantage of COOLCAT resides in the fact that ROCK is extremely difficult to tune (finding the right ϕ), while COOLCAT's behavior to its only parameter (m) is extremely stable: small values of m are sufficient to obtain a good result. In the largest data set for which we compared both techniques (Mushrooms), COOLCAT had a significantly better running time.

The incremental nature of COOLCAT makes it possible to apply the algorithm to data streams, and as the results in scalability show, the algorithm can cope with large volumes of data. We are currently doing research in tracking evolving clusters using COOLCAT.

7. ACKNOWLEDGMENTS

We like to thank Vipin Kumar and Eui-Hong (Sam) Han for lending us their implementation of ROCK.

8. REFERENCES

[1] M.S. Aldenderfer and R.K. Blashfield. *Cluster Analysis*. Sage Publications, (Sage University Paper series on Quantitative Applications in the Social Sciences, No. 44), 1984.

[2] D. Barbará. Requirements for clustering data streams. *SIGKDD Explorations (Special Issue on Online, Interactive, and Anytime Data Mining)*, 3(2), 2002.

[3] D. Barbará and P. Chen. Using the fractal dimension to cluster datasets. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Boston, MA*, August 2000.

[4] R.B. Calinski and J. Harabasz. A dendrite method for cluster analysis. *Communications in Statistics*, pages 1–27, 1974.

[5] P. Cheeseman and J. Stutz. Bayesian classification (AUTOCLASS): Theory and Results. In U.M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy, editors, *Advances in Knowledge Discovery and Data Mining*. AAAI Press, Menlo Park, 1995.

[6] C. CHen, A.W. Fu, and Y. Zhang. Entropy-based Subspace Clustering for Mining Numerical Data. In *Proceedings of ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Diego, CA*, August 1999.

[7] H. Chernoff. A Measure of Asymptotic Efficiency for Tests of a Hypothesis Based on the Sum of Observations. *Annals of Mathematical Statistics*, pages 493–509, 1952.

[8] DataGen. Data Generator: Perfect data for an imperfect world. http://www.datasetgenerator.com/.

[9] R.C. Dubes and A.K. Jain. Validity studies in clustering methodologies. *Pattern Recognition*, pages 235–254, 1979.

[10] R.O. Duda and P.E. Hart. *Pattern Classification and Scene Analysis*. Wiley-Interscience, New York, 1973.

[11] M. Ester, H.P. Kriegel, and X. Wu. A density-based algorithm for discovering clusters in large spatial database with noise. In *Proceedings of the International Conference on Knowledge Discovery and Data Mining, Portland, Oregon*, August 1996.

[12] V. Ganti, J. Gehrke, and R. Ramakrishnan. CACTUS-Clustering Categorical Data Using Summaries. In *Proceedings of the ACM-SIGKDD International Conference on Knowledge Discovery and Data Mining, San Diego, CA*, 1999.

[13] M. Garey and D. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W.H. Freeman, 1979.

[14] D. Gibson, J. Kleinberg, and P. Raghavan. Clustering Categorical Data: An Approach Based on Dynamical Systems. In *Proceedings of the International Conference on Very Large Databases (VLDB), New York, NY*, September 1998.

[15] A. Gluck and J. Corter. Information, uncertainty, and the utility of categories. In *Proceedings of the Seventh Annual Conference of the Cognitive Science Society*, 1985.

[16] E. Gokcay and J.C. Principe. Information Theoretic Clustering. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(2), February 2002.

[17] S. Guha, R. Rastogi, and K. Shim. CURE: A clustering algorithm for large databases. In *Proceedings of the ACM SIGMOD Conference on Management of Data, Seattle, WA*, May 1998.

[18] S. Guha, R. Rastogi, and K. Shim. ROCK: A Robust Clustering Algorithm for Categorical Attributes. In *Proceedings of the 15th International Conference on Data Engineering, Sydney, Australia*, April 1999.

[19] E.H. Han, G. Karypis, V. Kumar, and B. Mobasher. Clustering based on association rule hypergraphs. In *Proceedings of the SIGMOD Workshop on Research Issues on Data Mining and Knowledge Discovery*, June 1997.

[20] S. Hettich(librarian). UCI KDD Archive. http://kdd.ics.uci.edu/.

[21] A.K. Jain and R.C. Dubes. *Algorithms for clustering data*. Prentice Hall, 1988.

[22] G.J McLachlan and K.E. Basford. *Mixture Models*. Marcel Dekker, New York, 1988.

[23] T.M. Mitchell. *Machine Learning*. McGraw-Hill, 1997.

[24] J.C. Pincipe, D. Xu, and J. Fisher. Information theoretic learning. In S. Haykin, editor, *Unsupervised Adaptive Filtering*. John Wiley & Sons, 2000.

[25] A. Renyi. On Measures of Entropy and Information. In *Proc. of the Fourth Berkeley Symp. Math., Statistics, and Probability*, 1960.

[26] J. Rissanen. A universal prior for integers and estimation by minimum description length. *The Annals of Statistics*, 1983.

[27] J. Rissanen. *Stochastic complexity in statistical inquiry*. World Scientific Pub., 1989.

[28] C.E. Shannon. A mathematical theory of communication. *Bell System Techical Journal*, pages 379–423, 1948.

[29] C.S. Wallace and D.M. Boulton. An information measure for classification. *The Computer Journal*, 11(2), 1968.

[30] C.S. Wallace and D.L. Dowe. Intrinsic classification by MML, the Snob program. In *Proceedings of the 7th Australian Joint Conference on Artificial Intelligence*, 1994.

[31] R. Zhang, R. Ramakrishnan, and M.Livny. Birch: An efficient data clustering method for very large databases. In *Proceedings of the ACM SIGMOD Conference on Data Management, Montreal, Canada*, June 1996.