Context-Based Distance Learning for Categorical Data Clustering

Dino Ienco, Ruggero G. Pensa, and Rosa Meo

Dept. of Computer Science, University of Torino, Italy
{ienco,pensa,meo}@di.unito.it

Abstract. Clustering data described by categorical attributes is a challenging task in data mining applications. Unlike numerical attributes, it is difficult to define a distance between pairs of values of the same categorical attribute, since they are not ordered. In this paper, we propose a method to learn a context-based distance for categorical attributes. The key intuition of this work is that the distance between two values of a categorical attribute A_i can be determined by the way in which the values of the other attributes A_j are distributed in the dataset objects: if they are similarly distributed in the groups of objects in correspondence of the distinct values of A_i a low value of distance is obtained. We propose also a solution to the critical point of the choice of the attributes A_j. We validate our approach on various real world and synthetic datasets, by embedding our distance learning method in both a partitional and a hierarchical clustering algorithm. Experimental results show that our method is competitive w.r.t. categorical data clustering approaches in the state of the art.

1 Introduction

Clustering is a popular data mining technique that enables to partition data into groups (clusters) in such a way that objects inside a group are similar, and objects belonging to different groups are dissimilar [1]. Clearly, the notion of similarity is central in such a process. When objects are described by numerical (real, integer) features, there is a wide range of possible choices. Objects can be considered as vectors in a n-dimensional space, where n is the number of features. Then, many distance metrics can be used in n-dimensional spaces. Among them, probably the most popular metric is the Euclidean distance (or 2-norm distance), which is a special case of Minkowski distance (also called p-norm distance). Given two objects, these measures only depend on the difference between the values of the feature vectors.

In data mining applications, however, data are often described by categorical attributes that take values in a (usually finite) set of unordered nominal values. This makes impossible even to rank or compute differences between two values of the feature vectors. For categorical data the simplest comparison measure is *overlap* [2]. The proximity between two multivariate categorical entities is proportional to the number of attributes in which they match. Other metrics, such as the Jaccard coefficient, are derived from *overlap* and have been adopted in several (partitional and hierarchical) clustering algorithms [3,4,5].

Clearly, these distance metrics do not distinguish between the different values taken by the attribute, since they only measure the equality between pair of values. This is a strong limitation for a clustering algorithm, since it prevents to capture similarities that are clearly identified by human experts. For instance, given an attribute like *city*, which takes values in the set $\{Paris, Rome, Florence\}$ it is obvious that Florence is more similar to Rome than to Paris, from a geographic point of view. However, in some other contexts, Paris might be more similar to Rome, since both of them are capitals, and they may share similar behaviors.

In literature some measures that take into consideration the context of the features, have also been employed but refer to continuous data, e.g., Mahalanobis distance.

In this paper we present a new methodology to compute a context-based distance between values of a categorical variable and apply this technique to clustering categorical data with both partitional and hierarchical techniques. For the introduction of our technique, consider the dataset described in figure 1(a), with only two categorical attributes: *City(Milan, Turin, Florence)* and *Sex(M,F)*. The contingency table in Figure 1(b) shows how these values are distributed. We observe that *City=Florence* occurs only with *Sex=Female*, and the *City=Turin* occurs only with *Sex=Male*. The value *City=Milan* occurs both with *Sex=Male* and *Sex=Female*. From this distribution of data, we infer that, in this particular context, *Florence* is more similar to *Milan* than to *Turin* because the probability to observe a person of a given sex is closer.

Sex	**City**
Male	Turin
Female	Milan
Male	Turin
Male	Milan
Female	Florence

(a)

	Turin	Milan	Florence
Female	0	1	1
Male	2	1	0

(b)

Fig. 1. A toy dataset (a) and its related contingency table (b)

From this example we can deduce that the distribution of the co-occurrence table may help to define a distance between values of a categorical attribute. To this purpose, we propose a two-step method:

1. for each categorical attribute X, first identify a suitable context constituted by a set of attributes $Y \neq X$, such that each attribute belonging to the context is correlated to the attribute X.
2. compute a distance matrix between any pair of values (x_i, x_j) of X: we take into account the distribution of x_i and x_j in objects having the same values for the context attributes.

The key contribution of our work are the following:

– we introduce a new method to compute the distance between any pair of values of a specific categorical attribute;
– we define a distance-learning approach which is independent of the employed distance-based clustering;
– we show the impact of our approach within two different distance-based clustering algorithms.

We will also show that our approach is scalable w.r.t. to the numbers of instances in the dataset, and can manage thousands of categorical attributes.

2 Related Work

Clustering is an important task in data mining, in information retrieval and in a wide range of analytical and scientific applications [1]. The goal of clustering is to find a partition of the instances according to a predefined distance measure or an objective function to optimize. The problem is particularly difficult when categorical attributes are involved in the clustering process. In literature, many approaches to categorical data clustering have been proposed. Most of them try to optimize a global objective function without using any notion of distance between the values of the same attribute. Furthermore they suffer in terms of efficiency and time complexity with large data sets.

One of the first work in the field of categorical clustering is K-MODES [3]. It tries to extend K-Means algorithm for categorical data. A cluster is represented as a data point which is composed by the most frequent value in each attribute domain. Therefore, in K-MODES the similarity of an unlabeled data point and a cluster representative can be simply calculated by the overlap distance [2].

Another approach to categorical clustering is ROCK [4]. It employs links to measure similarity/proximity between pairs of data points. An instance belongs to the neighborhood of another instance if the Jaccard similarity between them exceeds a user-defined threshold. It heuristically optimizes a cluster quality function with respect to the number of links in an agglomerative hierarchical way. The base algorithm has cubic complexity in the size of the data set, which makes it unsuitable for large datasets.

LIMBO [5], is a scalable hierarchical categorical clustering algorithm built on the Information Bottleneck framework. As a hierarchical algorithm, LIMBO is not as fast as partitional methods. The algorithm builds Distributional Cluster Features (DCF) trees to summarize the data in k clusters, where each node contains statistics on a subset of instances. Starting from DCF and the number of clusters k a scan over the whole data set is performed to assign each instance to the cluster with the closest DCF.

CLICKS [6] is a clustering algorithm based on graph/hypergraph partitioning. In general the cost of clustering with graph structures is acceptable, provided that the underlying data is low dimensional. CLICKS finds clusters in categorical datasets based on a search method for k-partite maximal cliques. The vertices of the graph are the value of the different attributes and there is an edge between two vertexes if the two attribute-values occur in the same instance. All maximal k-partite cliques in the graph are enumerated and the support of the candidate cliques within the original dataset is verified to form the final clusters.

Alternative approaches include combinatorial algorithms [7] and entropy-based methods [8,9]

A first attempt of computing a distance for categorical attributes is [10] where the authors propose a probabilistic framework which considers the distribution of all the attributes in the dataset. However they only compare their approach with K-MODES and on small and low dimensional datasets.

3 The DILCA Method

In this section we present DILCA (DIstance Learning in Categorical Attributes) for computing distances between any pair of values of a categorical attribute.

Let us consider a set $F = \{X_1, X_2, \ldots, X_m\}$ of m categorical attributes. We refer to the cardinality of an attribute (or feature) X as $|X|$. Let $D = \{d_1, d_2, \ldots, d_n\}$ be a dataset of instances defined over F. We denote by x_i a specific value of an attribute X.

From the example in Section 1 it turns out that the distribution of values of an attribute can be informative about the way in which another attribute is distributed in the dataset objects. Thanks to this method we can infer a context-based distance between any pair of values of the same attribute. In real applications there are several attributes: for this reason our approach is based on two steps:

1. selection of a relevant subset of the whole attributes set that we use as the context for a given attribute;
2. computation of the distance measure between pair of values of the same attribute using the context defined in the previous step.

Context selection. We investigate the problem of selecting a good (informative) set of features w.r.t. a given one. This is a classic problem in data mining named feature selection. Feature selection is a preprocessing step of data mining. Its goal is to select a subset of relevant and not redundant features and discard all the other ones w.r.t. a given class attribute (supervised feature selection [11]). In this branch of research many approaches for measuring the correlation/association between two variables have been proposed. An interesting metrics is the *Symmetric Uncertainty*, introduced in [12]. This measure is a correlation-based measure inspired by information theory. Symmetric Uncertainty is derived from entropy: it is a measure of the uncertainty of a random variable. The entropy of a random variable X is defined as:

$$H(X) = -\sum_i P(x_i) \log_2(P(x_i))$$

where $P(x_i)$ is the probability of the value x_i of X. The entropy of X after having observed the values of another variable Y is defined as:

$$H(X|Y) = -\sum_j P(y_j) \sum_i P(x_i|y_i) \log_2(P(x_i|y_i))$$

where $P(x_i|y_i)$ is the probability that $X = x_i$ after we have observed that $Y = y_i$. The information about X provided by Y is given by the *information gain* [13] which is defined as follows:

$$IG(X|Y) = H(X) - H(X|Y)$$

When $IG(X|Y) > IG(Z|Y)$ then the feature X is more correlated to Y than Z. Moreover, the Information gain is symmetrical for two random variables X and Y [12].

The Symmetrical Uncertainty is then defined as follows:

$$SU(X,Y) = 2 \cdot \frac{IG(X|Y)}{H(X) + H(Y)}$$

This measure varies between 0 and 1 (1 indicates that knowledge of the value of either X or Y completely predicts the value of the other variable; 0 indicates that X and Y are independent). The advantage of Symmetrical Uncertainty (SU) w.r.t. Information Gain is that this measure is not biased by the number of values of an attribute.

During the first step, we select a set of context attributes for a given target attribute X. This context, named $context(X)$, is such that the attributes Y belonging to this set have a high value of $SU(X,Y)$. Determining an adequate number of attributes for $context(X)$ is not trivial. We propose to use an heuristic to set this number. The heuristic is based on the mean value of SU for a specific target attribute X. Given a target attribute X we want to compute $SU(X,Y)$, for each attribute $Y \neq X$. We denote this Symmetric Uncertainty $SU_X(Y) = SU(X,Y)$. The mean of this quantity is:

$$E[SU_X] = \frac{\sum_{Y \in F \setminus X} SU_X(Y)}{|F| - 1}$$

To determine the context of an attribute X we use the features that satisfy the following inequality:

$$context(X) = \{Y \neq X \;\; s.t. \;\; SU_X(Y) \geq \sigma E[SU_X]\}$$

where $\sigma \in [0,1]$ is a trade-off parameter that controls the influence of the mean value. According to this heuristic, at least one attribute is assigned to $context(X)$. This is simple to demonstrate: if $\mathrm{SU}_X(Y)$ is the same for all Y then $\mathrm{SU}_X(Y) = E[SU_X]$ for all Y; in this case all Y would be selected in $context(X)$. If there exists at least one attribute Y such that $\mathrm{SU}_X(Y) \geq E[SU_X]$, then those attributes Y would be selected.

Distance computation. The second step of our approach consists in computing the distance between each pair of values of the considered feature. To compute this distance between x_i and x_j where $x_i \in X$, $x_j \in X$ we use the following formula:

$$d(x_i, x_j) = \sqrt{\sum_{Y \in context(X)} \sum_{y_k \in Y} (P(x_i|y_k) - P(x_j|y_k))^2} \tag{1}$$

For each context attribute Y we compute the conditional probability for both values x_i and x_j given the values $y_k \in Y$ and then we apply the Euclidean distance. Our intra-attribute distance measure is an application of the Euclidean distance. As such, our definition of distance is a metric.

We introduce now our algorithm which, for each attribute X, computes the similarity matrices between any pair of values of X.

Algorithm 1 shows the procedure adopted to compute the correlation matrix between each pair of features based on Symmetric Uncertainty. This algorithm takes as parameter the entire data set D. Function $feature(D)$ returns the set of all the features contained in D. Then the algorithm computes the co-occurrence table (CO_{XY}) between each pair of attributes using function $ComputeCoOccurrenceTable(D, X, Y)$. It computes the joint probability of (X, Y). These tables are used to compute the Symmetric Uncertainty between attributes to be stored in $matrixSU$.

Algorithm 2 computes the distance matrix between the values of the target attribute X. At the first line it selects the vector storing the correlation between X and all other features Y. During the second step it computes the mean of the vector and then it selects the features that will be included in the context of X. When the features are chosen, the distance matrix for the values of attribute X is built, using (1).

Algorithm 1. computeCorrelationMatrix(D)

```
1: for all $X, Y \in feature(D) | X \neq Y$ do
2:     $CO_{XY}$ = ComputeCoOccurrenceTable($D$,$X$,$Y$)
3:     $matrixSU[X][Y] = SU(CO_{XY})$
4: end for
5: return $matrixSU$
```

Complexity. Before computing the distance matrix, we must compute $l = m * (m - 1)/2$ matrices $CO_{X,Y}$. These l matrices store the co-occurrence of the values between any pair of attributes. To build these matrices we need to perform a complete scan of the entire data set. We use the co-occurrence matrices to compute $matrixSU$. $matrixSU$ is $m \times m$, where m is the number of attributes in the dataset. Using $matrixSU$ we can compute $E[SU_X]$ and then select the right context making use of σ. To compute the distance matrix for each attribute we can use l co-occurrence matrices without the necessity of further scans of the

Algorithm 2. DILCA($matrixSU$,X,σ)

```
1: $VectorSU_X = MatrixSU[X]$
2: $E =$ computeMean($VectorSU_X$)
3: $context(X) = \emptyset$
4: for all $y \in VectorSU_X$ do
5:    if $VectorSU_X[y] \geq \sigma E$ then
6:       insert($Y$,$context(X)$)
7:    end if
8: end for
9: for all $x_i, x_j \in X | x_i \neq x_j$ do
10:    $DistanceMatrix[x_i][x_j] = \sqrt{\sum_{Y \in context(X)} \sum_{y_k \in Y} (P(x_i|y_k) - P(x_j|y_k))^2}$
11: end for
12: return $DistanceMatrix_X$
```

dataset. From this, we derive that our algorithm only needs to scan the entire dataset once. In conclusion, our approach is $O(nm^2)$, with n the number of instances and m the number of involved attributes.

4 Experiments and Results

In this section we present a comprehensive evaluation of our approach. Since our method enables to use distance based approaches for clustering, we coupled it with two standard methods: a partitional one, and a hierarchical one. We compared both of them with state-of-the-art techniques for categorical data clustering.

Evaluation Measures for Clustering. Determine the clustering quality provided by an algorithm is often a hard and subjective task. Therefore, we use two objective criteria to evaluate the results: Accuracy and Normalized Mutual Information.

The first considers the original class label as a mean to evaluate clustering results. Assume that the instances in D have been already classified in p classes $\{p_1, p_2, ..., p_P\}$. Consider a clustering algorithm that partitions D into c clusters $\{cl_1, cl_2, ..., cl_C\}$. We refer to a one-to-one mapping, f, from classes to clusters, such that each class p_i is mapped to the cluster $cl_j = f(p_i)$. The *classification error* of the mapping is defined as:

$$E = \sum_{i=1}^{P} |p_i \cap \overline{f(p_i)}|$$

where $|p_i \cap \overline{f(p_i)}|$ measures the number of objects in class p_i that received the wrong label. The optimal mapping between clusters and classes is the one that minimizes the classification error. We use E_{min} to denote the classification error of the optimal mapping. Then to obtain the Accuracy we compute the following formula:

$$Acc = 1 - \frac{E_{min}}{|D|}$$

The second metrics provides an information that is independent of the number of clusters [14]. This measure takes its maximum value when the clustering partition matches completely the original partition. We can consider NMI as an indicator of the purity of the clustering results. NMI is computed as the average mutual information between any pair of clusters and classes:

$$\mathbf{NMI} = \frac{\sum_{i=1}^{C} \sum_{j=1}^{P} x_{ij} \log \frac{n * n_{ij}}{n_i n_j}}{\sqrt{\sum_{i=1}^{C} n_i \log \frac{n_i}{n} \sum_{j=1}^{P} n_j \log \frac{n_j}{n}}}$$

where n_{ij} is the cardinality of the set of objects that occur both in cluster i and in class j; n_i is the number of objects in cluster i; n_j is the number of objects in class j; n is the total number of objects. C and P are respectively the number of clusters and the number of classes.

Datasets for Categorical Clustering Evaluation. For the evaluation of our distance learning approach on categorical data, we used two collections of datasets. The first collection consists in real world data sets downloaded from the UCI Machine Learning Repository [15]. The second collection contains synthetic datasets produced by a data generator [16] using Gaussian distributed random attributes. The main characteristics of these datasets are summarized in Table 1. Notice that *Breast-w* and *Sonar* contain numerical variables. Indeed, they have been discretized using the supervised method proposed in [17].

Table 1. Datasets characteristics

Dataset	Type	Instances	Features	Values	Classes
Votes	Real	435	16	32	2
Mushroom	Real	8124	22	117	2
Breast-w	Real	699	9	29	2
Sonar	Real	208	60	81	2
SynA	Synth	1000	50	1000	5
SynB	Synth	3000	20	600	4

4.1 Experimental Settings and Results

Here, we report on the performance results of K-MODES$_{DILCA}$ (DILCA coupled with a simple K-MODES algorithm) and HCL$_{DILCA}$ (DILCA coupled with Ward hierarchical clustering (HCL). We compared them with ROCK [4] and LIMBO [5]. For all the algorithms we set the number of clusters equal to the number of classes. We implemented K-MODES$_{DILCA}$ within the WEKA platform [17], a Java open source library that provides machine learning and data mining algorithms. HCL$_{DILCA}$ was implemented over the Java Murtagh's implementation of HCL[1].

[1] `http://astro.u-strasbg.fr/~{}fmurtagh/mda-sw/`

We run the experiments on a PC with a 1.86GHz Intel Pentium M processor, 1024MB of RAM running Linux. For each particular algorithm we used the following setting:

– For K-MODES$_{DILCA}$ we varied parameter σ between 0 to 1 (with steps of 0.1) and we report the value which gave the best results. Since the initial partition is random, we run many times the algorithm and then we report the average result in terms of accuracy and Normalized Mutual Information.
– For HCL$_{DILCA}$ we varied parameter σ between 0 to 1 with step of 0.1 and we report the value which gave the best results. Since the hierarchical algorithm returns a dendrogram which, at each level, contains a different number of clusters, we considered the level corresponding to the number of clusters equal to the number of classes.
– For ROCK we set the threshold parameter between 0.2 to 1 with steps of 0.05. Also for this algorithm we retained the best obtained result.
– For LIMBO we set ϕ parameter between 0 to 1 with steps of 0.25 and we report the best obtained result. This parameter influences the information loss during the merging phase.

In Table 2 we report the result of the comparative evaluation with other clustering algorithms. For each dataset we report the average Accuracy in percentage and the average Normalized Mutual Information achieved by each algorithm. In almost all the experiments, our approach achieves the best results in at least one category of clustering, and in some cases (Sonar and Votes), the performance parameters are sensibly better than in ROCK and LIMBO. The only exception is SynA, where NMI computed for ROCK is slightly higher than NMI achieved by HCL$_{DILCA}$. However, the Accuracy measured for ROCK is lower than the one achieved by HCL$_{DILCA}$. Moreover, we observed that ROCK is very sensitive to the parameter value, and in many cases this algorithm produces one giant cluster that includes instances from more classes.

Table 2. Experiments on real and synthetic data

	K-MODES$_{DILCA}$		HCL$_{DILCA}$		ROCK		LIMBO	
Dataset	Acc.	NMI	Acc.	NMI	Acc.	NMI	Acc.	NMI
Sonar	**71.63%**	**0.9912**	55.29 %	0.0191	56.25 %	0.0093	66.35%	0.0843
Votes	87.59%	0.4892	**89.89%**	**0.5195**	83.90%	0.3446	87.12%	0.4358
Breast-w	**95.99%**	**0.7435**	94.13%	0.6650	76.68%	0.2570	55.94%	0.0009
Mushroom	**89.02%**	0.5518	**89.02%**	**0.5938**	50.57%	0.05681	88.95%	0.5522
SynA	89.50%	0.7864	**94.30%**	0.8641	80.3%	**0.8965**	87.6%	0.7540
SynB	**100%**	**1.0000**	**100%**	**1.0000**	**100%**	**1.0000**	26.77%	0.0017

Impact of σ. To evaluate the impact of σ parameter we use *Mushroom* and *Votes* datasets. For each dataset we plot the behavior of K-MODES$_{DILCA}$. We let vary the parameter σ from 0 to 1 with steps of 0.1. When the parameter is equal to 0 all the features are included in the context. We observed that the influence of different settings of σ w.r.t. accuracy is small (curves are omitted here). In both datasets, the variation in accuracy is very low (less than 0.50%). Although there is no general law about how to choose this parameter, we estimate that its impact is less important than standard clustering parameters (such as, the number of clusters, or other algorithm-specific parameters).

4.2 Scalability of DILCA

We introduce now a study on the scalability of our distance learning approach, by analyzing the overall computational time of K-MODES$_{DILCA}$ and HCL$_{DILCA}$, and the portion of time needed to perform distance computation (DILCA). We evaluate the scalability varying the two dimensions of the dataset that may have an impact on time performances.

The first dimension is the number of instances. For this purpose, we generated 30,000 synthetic instances described by 100 attributes, then we built 30 datasets containing from 1000 to 30,000 instances. We report the results in Figure 2(a). The second dimension is the number of attributes: we generated a synthetic dataset consisting in 5,000 attributes and 1,000 instances. Then we built 10 datasets containing from 100 to 5000 features. The results are depicted in Figure 2(b). We perform also some experiments to evaluate the impact of parameter σ on running time using the complete dataset (see Figure 2(c)).

As the value of σ enables to select a different number of attributes in the context of the target attribute, this parameter could have an impact on the time performances of our approach. In Figure 2(c) we observe how changes of σ influence the execution time of our algorithms. For the part that computes the distance we can see that the higher the value of σ the lower the time used to build the intra-attribute distance matrix.

We also compared HCL$_{DILCA}$ with LIMBO (which is also hierarchical) on the scalability w.r.t. the number of features. We used another synthetic dataset with 1,000 instances and 1,000 attributes from which we built 10 datasets containing a variable number of features: from 100 to 1,000 features. We report the results in figure 2(e). In Figure 2(f), we show a comparison between the two algorithms on the dataset composed by an increasing number of instances. We observe that HCL$_{DILCA}$ is faster than LIMBO w.r.t. the size and dimensionality of the dataset.

Finally, we also investigated on the time spent by the HCL$_{DILCA}$ to perform clustering. In Figure 2(d) we report the time spent by the three parts of HCL coupled with DILCA. The three curves represent respectively the time spent by DILCA to compute distances, the time spent to compute the point-to-point distance matrix given as input to the hierarchical algorithm, and the effective time spent by Ward algorithm to build the dendrogram. In any case the most consistent portion of the overall computation time is employed to calculate the point-to-point distance measure between each pair of instances.

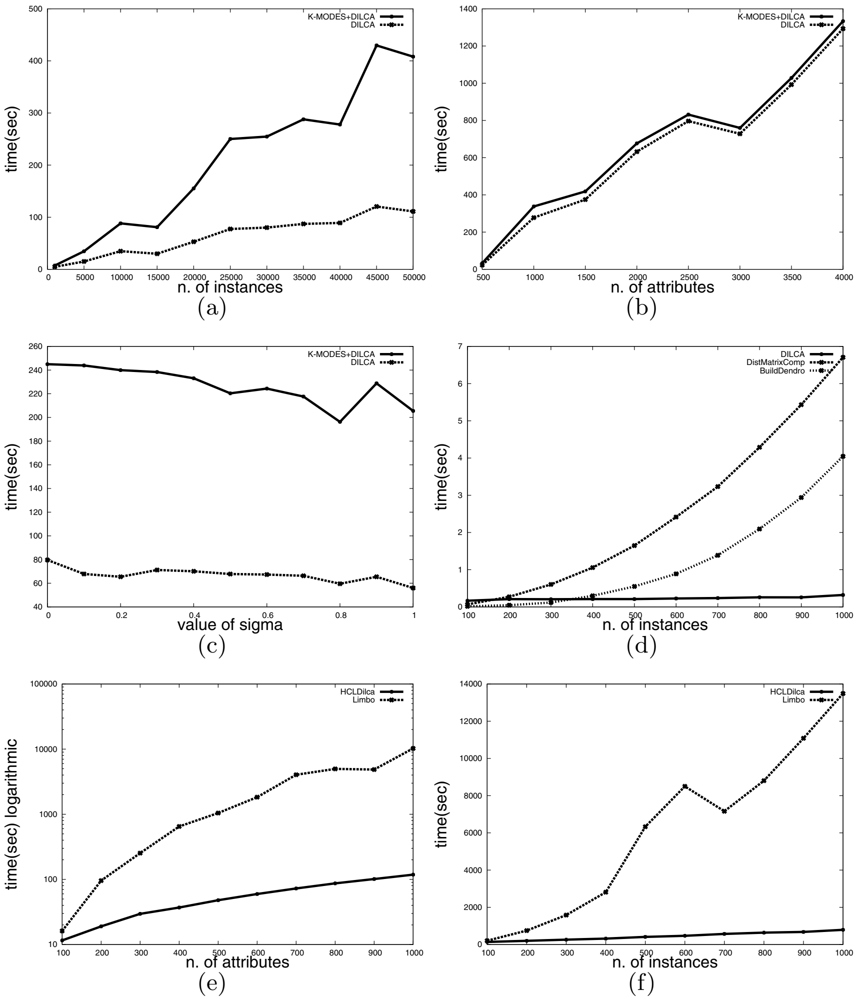

Fig. 2. Time performances of DILCA

5 Conclusion

We introduced a scalable approach to learn a context-based distance between values of categorical attributes. We showed the effective impact of this approach on two distance-based clustering approaches. We believe that the proposed method is general enough and it can be applied to any data mining task that involves categorical data and requires distance computations. As a future work we will investigate the application of our distance learning approach to different distance-based tasks such as: outlier detection and nearest neighbors classification. Moreover, using this distance it will be possible to compute distances between objects described by both numerical and categorical attributes.

Acknowledgments. The authors wish to thank Dr. Eui-Hong Han who provided the source code ROCK, Dr. Periklis Andritsos who provided the implementation of LIMBO, and Elena Roglia for stimulating discussions. Ruggero G. Pensa is co-funded by Regione Piemonte.

References

1. Han, J., Kamber, M.: Data Mining: Concepts and Techniques. The Morgan Kaufmann Series in Data Management Systems. Morgan Kaufmann, San Francisco (2000)
2. Kasif, S., Salzberg, S., Waltz, D., Rachlin, J., Aha, D.: Towards a framework for memory-based reasoning (manuscript, 1995) (in review)
3. Huang, Z.: Extensions to the k-means algorithm for clustering large data sets with categorical values. Data Min. Knowl. Discov. 2(3), 283–304 (1998)
4. Guha, S., Rastogi, R., Shim, K.: Rock: A robust clustering algorithm for categorical attributes. In: Proc. of IEEE ICDE 1999 (1999)
5. Andritsos, P., Tsaparas, P., Miller, R.J., Sevcik, K.C.: Scalable clustering of categorical data. In: Proc. of EDBT 2004, pp. 123–146 (2004)
6. Zaki, M.J., Peters, M.: Clicks: Mining subspace clusters in categorical data via k-partite maximal cliques. In: Proc. of IEEE ICDE 2005, pp. 355–356 (2005)
7. Ganti, V., Gehrke, J., Ramakrishnan, R.: Cactus-clustering categorical data using summaries. In: Proc. of ACM SIGKDD 1999, pp. 73–83 (1999)
8. Barbara, D., Couto, J., Li, Y.: Coolcat: an entropy-based algorithm for categorical clustering. In: Proc. of CIKM 2002, pp. 582–589. ACM Press, New York (2002)
9. Li, T., Ma, S., Ogihara, M.: Entropy-based criterion in categorical clustering. In: Proc. of ICML 2004, pp. 536–543 (2004)
10. Ahmad, A., Dey, L.: A method to compute distance between two categorical values of same attribute in unsupervised learning for categorical data set. Pattern Recogn. Lett. 28(1), 110–118 (2007)
11. Guyon, I., Elisseeff, A.: An introduction to variable and feature selection. J. Mach. Learn. Res. 3, 1157–1182 (2003)
12. Yu, L., Liu, H.: Feature selection for high-dimensional data: A fast correlation-based filter solution. In: Proc. of ICML 2003, Washington, DC (2003)
13. Quinlan, R.J.: C4.5: Programs for Machine Learning. Morgan Kaufmann Series in Machine Learning. Morgan Kaufmann, San Francisco (1993)
14. Strehl, A., Ghosh, J., Cardie, C.: Cluster ensembles - a knowledge reuse framework for combining multiple partitions. Journal of Machine Learning Research 3, 583–617 (2002)
15. Blake, C.L., Merz, C.J.: UCI repository of machine learning databases (1998), `http://www.ics.uci.edu/~mlearn/MLRepository.html`
16. Melli, G.: Dataset generator, perfect data for an imperfect world (2008), `http://www.datasetgenerator.com`
17. Witten, I.H., Frank, E.: Data Mining: Practical Machine Learning Tools and Techniques, 2nd edn. Data Management Systems. Morgan Kaufmann, San Francisco (2005)