CLUSTERING LARGE DATA SETS WITH MIXED NUMERIC AND CATEGORICAL VALUES*

ZHEXUE HUANG
CSIRO Mathematical and Information Sciences
GPO Box 664 Canberra ACT 2601, AUSTRALIA
huang@cmis.csiro.au

Efficient partitioning of large data sets into homogenous clusters is a fundamental problem in data mining. The standard hierarchical clustering methods provide no solution for this problem due to their computational inefficiency. The *k-means* based methods are promising for their efficiency in processing large data sets. However, their use is often limited to numeric data. In this paper we present a *k-prototypes* algorithm which is based on the *k-means* paradigm but removes the numeric data limitation whilst preserving its efficiency. In the algorithm, objects are clustered against *k* prototypes. A method is developed to dynamically update the *k* prototypes in order to maximise the intra cluster similarity of objects. When applied to numeric data the algorithm is identical to the *k-means*. To assist interpretation of clusters we use decision tree induction algorithms to create rules for clusters. These rules, together with other statistics about clusters, can assist data miners to understand and identify interesting clusters.

1 Introduction

Many data mining applications require partitioning of data into homogeneous clusters from which interesting groups may be discovered, such as a group of motor insurance policy holders with a high average claim cost, or a group of clients in a banking database showing a heavy investment in real estate. To perform such analyses at least the following two problems have to be solved; (1) efficient partitioning of a large data set into homogeneous groups or clusters, and (2) effective interpretation of clusters. This paper proposes a solution to the first problem and suggests a solution to the second.

A number of data partitioning methods can be employed for the first problem. When little is known about the distribution of data, clustering methods are often used. However, data mining, distinct from other traditional applications of cluster analysis[1,5], deals with large high dimensional data (thousands or millions of records with tens or hundreds of attributes). This characteristic prohibits many existing clustering algorithms from being used in data mining applications. Another characteristic is that data in data mining often contains both numeric and categorical values. The traditional way to treat categorical attributes as numeric does not always produce meaningful results because many categorical domains are not ordered.

Although the standard hierarchical clustering methods can handle data with numeric and categorical values[5,8], the quadratic computational cost makes them

*This Work was supported by the Cooperative Research Centre for Advanced Computational Systems (ACSys) established under the Australian Government's Cooperative Research Centres Program.

unacceptable for clustering large data sets. The *k-means* based methods[12] are efficient for processing large data sets, thus very attractive for data mining. The major handicap for them is that they are often limited to numeric data. The reason is that these algorithms optimise a cost function defined on the Euclidean distance measure between data points and means of clusters[5]. Minimising the cost function by calculating means limits their use to numeric data.

Conceptual clustering algorithms developed in machine learning cluster data with categorical values[6,11,13] and also produce conceptual descriptions of clusters. The latter feature is important to data mining because the conceptual descriptions provide assistance in interpreting clustering results. Unlike statistical clustering methods, these algorithms are based on a search for objects which carry the same or similar concepts. Therefore, their efficiency relies on good search strategies. For problems in data mining, which often involve many concepts and very large object spaces, the concepts based search methods can become a potential handicap for these algorithms to deal with extremely large data sets.

In this paper we present a clustering algorithm to solve data partition problems in data mining. The algorithm is based on the *k-means* paradigm but removes the numeric data only limitation whilst preserving its efficiency. The algorithm clusters objects with numeric and categorical attributes in a way similar to *k-means*. Because objects are clustered against *k prototypes* instead of *k means* of clusters, we call it the *k-prototypes* algorithm. We have developed a method to dynamically update the *k* prototypes in order to maximise the intra cluster similarity of objects. The object similarity measure is derived from both numeric and categorical attributes. When applied to numeric data the algorithm is identical to *k-means*. In testing with real data, this algorithm has demonstrated a capability of partitioning data sets in the range of a hundred thousand records, described by some 20 numeric and categorical attributes, into a hundred clusters in a couple of hours on a SUN Sparc10 workstation.

We also consider conceptual descriptions of clusters. However, in dealing with large data sets we take a different approach. After clustering has been carried out, we fit the classified data into a decision tree induction algorithm[3,15] to create rules for cluster descriptions. These rules, together with some other statistics about clusters, can assist data miners to understand and identify interesting clusters.

The rest of the paper is organised as follows. The next section gives some mathematical preliminaries of the algorithm. In Section 3 we discuss the *k-prototypes* algorithm. In Section 4 we present some simulation results to show how numeric and categorical attributes interact in the process of clustering by the algorithm. We also present some initial performance test results on a large real world

data set. In Section 5 we summarise the discussions and point out some directions for our future work.

2 Mathematical Preliminaries

Let $\boldsymbol{X} = \{X_1, X_2, \ldots, X_n\}$ denote a set of n objects and $X_i = [x_{i1}, x_{i2}, \ldots, x_{im}]$ be an object represented by m attribute values. Let k be a positive integer. The objective of clustering $\boldsymbol{X}$ is to find a partition which divides objects in $\boldsymbol{X}$ into k disjoint clusters.

For a given n, the number of possible partitions is definite but extremely large[1]. It is impractical to investigate every partition in order to find a better one for a classification problem. A common solution is to choose a clustering criterion[1,5] to guide the search for a partition. A clustering criterion is called a *cost function* below.

2.1 Cost Function

The widely used cost function is the trace of the within cluster dispersion matrix[5]. One way to define this cost function is

$$E = \sum_{l=1}^{k} \sum_{i=1}^{n} y_{il} d(X_i, Q_l) \tag{2.1}$$

Here, $Q_l = [q_{l1}, q_{l2}, \ldots, q_{lm}]$ is the *representative vector* or *prototype* for cluster l, and y_{il} is an element of a *partition* matrix Y_{nxl} [9]. d is a similarity measure often defined as the square Euclidean distance.

Y has the following two properties, (1) $0 \le y_{il} \le 1$ and (2) $\sum_{l=1}^{k} y_{il} = 1$. Y is called a *hard partition* if $y_{il} \in \{0,1\}$. Otherwise, it is a *fuzzy partition*[2]. In a hard partition, $y_{il} = 1$ indicates that object X_i is assigned to cluster l by Y. We only consider hard partitions in this paper.

The inner term $E_l = \sum_{i=1}^{n} y_{il} d(X_i, Q_l)$ in Eq. (2.1) is the total cost of assigning $\boldsymbol{X}$ to cluster l, i.e., the total dispersion of objects in cluster l from its prototype Q_l. E_l is minimised if

$$q_{lj} = \frac{1}{n_l} \sum_{i=1}^{n} y_{il} x_{ij} \qquad \text{for } j=1,\ldots,m \tag{2.2}$$

where $n_l = \sum_{i=1}^{n} y_{il}$ is the number of objects in cluster l.

When $\boldsymbol{X}$ has categorical attributes, we can introduce a similarity measure as

$$d(X_i, Q_l) = \sum_{j=1}^{m_r} (x_{ij}^r - q_{lj}^r)^2 + \gamma_l \sum_{j=1}^{m_c} \delta(x_{ij}^c, q_{lj}^c) \tag{2.3}$$

where $\delta(p,q)=0$ for $p=q$ and $\delta(p,q)=1$ for $p \neq q$. x_{ij}^r and q_{lj}^r are values of numeric attributes, whereas x_{ij}^c and q_{lj}^c are values of categorical attributes for object i and the prototype of cluster l. m_r and m_c are the numbers of numeric and categorical attributes. γ_l is a weight for categorical attributes for cluster l.

We can rewrite E_l as

$$\begin{aligned} E_l &= \sum_{i=1}^{n} y_{il} \sum_{j=1}^{m_r} (x_{ij}^r - q_{lj}^r)^2 + \gamma_l \sum_{i=1}^{n} y_{il} \sum_{j=1}^{m_c} \delta(x_{ij}^c, q_{lj}^c) \\ &= E_l^r + E_l^c \end{aligned} \tag{2.4}$$

where E_l^r is the total cost on all numeric attributes of objects in cluster l. E_l^r is minimised if q_{lj}^r is calculated by Eq. (2.2).

Let C_j be the set containing all unique values in the categorical attribute j and $p(c_j \in C_j | l)$ the probability of value c_j occurring in cluster l. E_l^c in Eq. (2.4) can be rewritten as

$$E_l^c = \gamma_l \sum_{j=1}^{m_c} n_l (1 - p(q_{lj}^c \in C_j | l)) \tag{2.5}$$

where n_l is the number of objects in cluster l. The solution to minimise E_l^c is given by the lemma below.

Lemma 1: For a specific cluster l, E_l^c is minimised if and only if $p(q_{lj}^c \in C_j | l) \geq p(c_j \in C_j | l)$ for $q_{lj}^c \neq c_j$ for all categorical attributes.

Finally, we can rewrite E as

$$E = \sum_{l=1}^{k} (E_l^r + E_l^c) = \sum_{l=1}^{k} E_l^r + \sum_{l=1}^{k} E_l^c = E^r + E^c \tag{2.6}$$

Eq. (2.6) is the cost function for clustering a data set with numeric and categorical values. Since both E^r and E^c are non-negative, minimisation of E can be achieved

by minimising E^r and E^c, the total costs on numeric and categorical attributes over all clusters. E^r can be minimised by calculating the numeric elements of the k cluster prototypes by Eq. (2.2), whereas E^c can be minimised by selecting the categorical elements of the k cluster prototypes according to Lemma 1. Therefore, Eq. (2.2) and Lemma 1 define a way to choose cluster prototypes to minimise the cost function Eq. (2.6). This is the basis for the *k-prototypes* algorithm to be discussed in Section 3.

2.2 Similarity Measure

The cost function Eq. (2.6) is defined on Eq. (2.3), which is a combined similarity measure on both numeric and categorical attributes between objects and cluster prototypes. The similarity measure on numeric attributes is the square Euclidean distance whereas the similarity measure on categorical attributes is the number of mismatches between objects and cluster prototypes. Weight γ_l is introduced to avoid favouring either type of attribute.

The influence of weight γ_l in clustering can be illustrated by Figure 1. Assume the triangles and diamonds represent a set of objects described by a categorical and two numeric attributes. Triangle and diamond represent two values of the categorical attribute whereas numeric attribute values are reflected by locations of the objects. These objects are partitioned into two clusters.

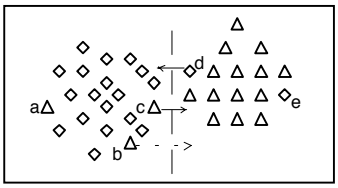


Figure 1. Influence of γ_l in clustering.

If γ_l=0, clustering only depends on numeric attributes, i.e., locations of the objects. The result will be the two clusters separated by the vertical dashed line. If $\gamma_l > 0$, then object *c* may change to the right cluster because it is close to that cluster and its categorical value is the same as that of the majority of objects in that cluster. Similarly, object *d* may change to the left cluster. However, object *a* may still stay in the left cluster because it is too far to the right, even though it has a categorical value as same as that of the majority objects in that cluster. Similarly, object *e* may still be in the right cluster. Object *b* becomes uncertain, depending on whether γ_l is biased towards numeric or categorical attributes. If γ_l is biased to the

categorical attribute, object b may change to the right cluster. Otherwise, it can stay in the left one.

The choice of γ_l is dependent on distributions of numeric attributes. Generally speaking, γ_l is related to σ_l, the average standard deviation of numeric attributes in cluster l. In practice, σ_l can be used as a guidance to determine γ_l. However, since σ_l is unknown before clustering, the overall average standard deviation σ of numeric attributes can be used for all σ_l. In an iterative algorithm, σ_l can be calculated from a preceding clustering result. In Section 4.2 we will present some simulations on γ_l.

3 *k-prototypes* Algorithm

The *k-prototypes* algorithm can be described as follows. (1) Select k initial prototypes from a data set $\boldsymbol{X}$, one for each cluster. (2) Allocate each object in $\boldsymbol{X}$ to a cluster whose prototype is the nearest to it according to Eq. (2.3). Update the prototype of the cluster after each allocation. (3) After all objects have been allocated to a cluster, retest the similarity of objects against the current prototypes. If an object is found such that its nearest prototype belongs to another cluster rather than its current one, reallocate the object to that cluster and update the prototypes of both clusters. (4) Repeat (3) until no object has changed clusters after a full cycle test of $\boldsymbol{X}$.

The algorithm is built upon three processes, *initial prototypes selection*, *initial allocation,* and *re-allocation*. The first process simply randomly selects k objects as the initial prototypes for clusters. The second process is presented in Figure 2. Starting from a set of initial cluster prototypes, this process assigns each object to a cluster and updates the cluster prototype accordingly after each assignment.

In Figure 2, X[i] represents object i and X[i,j] the value of attribute j for object i. O_prototypes[] and C_prototypes[] store the numeric and categorical attribute parts of cluster prototypes respectively. O_prototypes[i,j] and C_prototypes[i,j] are two corresponding numeric and categorical elements of the prototype for cluster i. **Distance**() is a square Euclidean distance function and **Sigma**() is an implementation of function $\delta()$ in Eq. (2.3). Clustership[] and ClusterCount[] record cluster membership of objects and numbers of objects in clusters. SumInCluster[] sums up numeric values of objects in clusters and is used to update numeric attributes of cluster prototypes. FrequencyInCluster[] records frequencies of different values of categorical attributes in clusters. The function **HighestFreq**() is an implementation of Lemma 1 to update categorical attributes of prototypes.

The reallocation process given in Figure 3 is similar to the initial allocation process except that after reallocation of an object, prototypes for both the previous

and current clusters of the object are updated. Variable moves records the number of objects which have changed clusters in the process.

```
FOR i = 1 TO NumberOfObjects
    Mindistance= Distance(X[i],O_prototypes[1])+ gamma* Sigma(X[i],C_prototypes[1])
    FOR j = 1 TO NumberOfClusters
        distance= Distance(X[i],O_prototypes[j])+ gamma * Sigma(X[i],C_prototypes[j])
        IF (distance < Mindistance)
            Mindistance=distance
            cluster=j
        ENDIF
    ENDFOR
    Clustership[i]=cluster
    ClusterCount[cluster] + 1
    FOR j=1 TO NumberOfNumericAttributes
        SumInCluster[cluster,j] + X[i,j]
        O_prototypes[cluster,j]=SumInCluster[cluster,j]/ClusterCount[cluster]
    ENDFOR
    FOR j=1 TO NumberOfCategoricAttributes
        FrequencyInCluster[cluster,j,X[i,j]] + 1
        C_prototypes[cluster,j]=HighestFreq(FrequencyInCluster,cluster,j)
    ENDFOR
ENDFOR
```

Figure 2. Initial allocation process.

```
moves=0
FOR i = 1 TO NumberOfObjects
    …
    (To find the cluster whose prototype is the nearest to object i.  Same as Figure 2)
    …
    IF (Clustership[i]<>cluster)
        moves+1
        oldcluster=Clustership[i]
        ClusterCount[cluster] + 1
        ClusterCount[oldcluster] - 1
        FOR j=1 TO NumberOfNumericAttributes
            SumInCluster[cluster,j] + X[i,j]
            SumInCluster[oldcluster,j] - X[i,j]
            O_prototypes[cluster,j]=SumInCluster[cluster,j]/ClusterCount[cluster]
            O_prototypes[oldcluster,j]= SumInCluster[oldcluster,j]/ClusterCount[oldcluster]
        ENDFOR
        FOR j=1 TO NumberOfCategoricAttributes
            FrequencyInCluster[cluster,j,X[i,j]] + 1
            FrequencyInCluster[oldcluster,j,X[i,j]] - 1
            C_prototypes[cluster,j]=HighestFreq(cluster,j)
            C_prototypes[oldcluster,j]=HighestFreq(oldcluster,j)
        ENDFOR
    ENDIF
ENDFOR
```

Figure 3. Reallocation process.

The algorithm iterates the reallocation process until moves=0, which indicates that the algorithm has stabilised in a local optimum of Eq.(2.6). Figure 4 shows a typical convergence curve of the algorithm, which was obtained from a data set with 75808 records and 20 attributes. The number of clusters is 64.

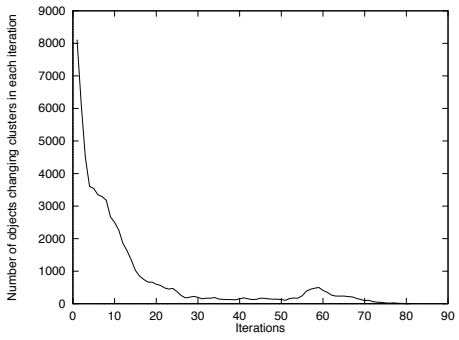


Figure 4. Convergence behaviour of the *k-prototype* algorithm.

We can see that the number of objects changing clusters drops very rapidly in the beginning. Then the dropping speed gradually slows down and becomes nearly unchanged in some stage. It also goes up a little before the number of changes becomes zero. This indicates that iteration of the reallocation process can be terminated at some point when the dropping speed becomes small. This compromise can reduce running time significantly for large data sets.

The computational cost of this algorithm is $O((t+1)kn)$, where n is the number of objects, k the number of clusters and t is the number of iterations of the reallocation process. Usually, $k << n$ and t does not exceed 100 according to our experiments on a large real world data set. Therefore, this algorithm is efficient in clustering large data sets.

Like the *k-means* algorithm, this algorithm also produces local optimal solutions, which are affected by the initial cluster prototypes. To avoid being trapped at a local optimum one can introduce some perturbations in the process. However, it still cannot guarantee a global optimal solution. There are a number of techniques for solving this global optimisation problem, such as simulated annealing[4,10,16] and genetic algorithms[7].

4 Experiments

In this section we present some simulation results of clustering data with mixed numeric and categorical values by the *k-prototypes* algorithm. We also briefly describe an exercise on clustering a real world data set.

4.1 Data Set Construction

In order to simplify illustration we use data records having only three attributes, two numeric and one categorical. These records were generated as follows. We first created two sets of two dimensional points by using methods in [5]. The first set contains 400 points and has one normal distribution (Figure 5a), whereas the second has four normal distributions and contains 300 points (Figure 6a).

We then expanded these points to three dimensions by adding a categorical value to each point (see figures 5b and 6b). For the first data set we deliberately divided it into four parts and assigned to the majority points in each part an identical categorical value and to the rest other categorical values. For instance the majority of points in the left high part in Figure 5b are assigned categorical value C and the rest in this part are assigned A, B or D. All assignments were randomly done. Similar assignments were given to the second point set as well (Figure 6b).

Note that the categorical value of a point does not indicate its class membership. In fact, these points have no classification at all. The categorical values simply represent objects in a third dimension which is not continuous and has no order. We can view this dimension as a set of planes overlaid where the distance between any two planes is 1. Each plane is identified by a unique categorical value. All points are projected to the corresponding planes according to their categorical values.

4.2 Simulations

The main purpose for these simulations was to demonstrate how numeric and categorical attributes interact each other in the process of clustering by the *k-prototypes* algorithm.

Given a set of k cluster prototypes for a data set described in Section 4.1, the rule for assigning an object x to a cluster l is

$$\{x \in l : d(x,q_l) \le d(x,q_v) \wedge l \ne v\} \qquad \text{for } v = 1,\ldots,k$$

Without the third dimension, a point is assigned to a cluster if the prototype of the cluster is nearest to the point. When the third dimension is involved, three situations can happen.

1. A point is assigned to a cluster if the prototype of the cluster is the nearest to the point in the two dimensional numeric space AND the prototype's categorical value of the cluster is same as that of the point.
2. A point may be assigned to a cluster, whose prototype is not the nearest to it in the two dimensional continuous subspace, but the categorical value of majority points in the cluster is same as that of the point.

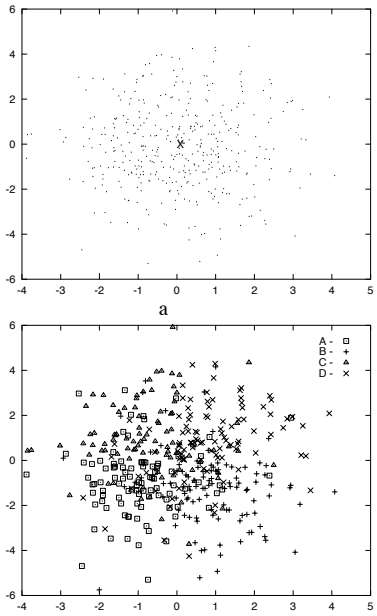


Figure 5.

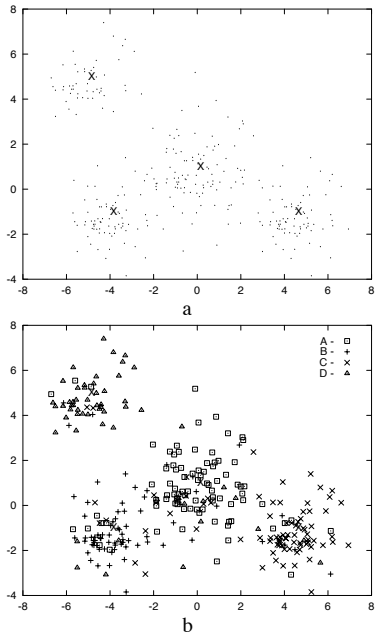


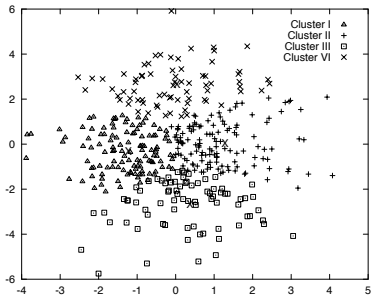
Figure 6.

3. A point may be assigned to a cluster, even though the categorical value of its prototype is different because the distance between the point and the cluster prototype is small enough.

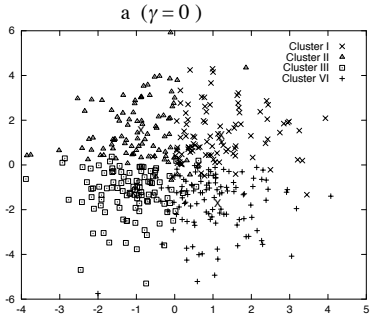
The second situation indicates that the categorical attribute has a strong impact in clustering, whereas the third situation shows a strong impact of numeric attributes.

We have implemented the *k-prototypes* algorithm using a global γ, i.e., the same γ for all clusters. We tested different γ values in clustering the data sets shown in Figure 5b and 6b. A small γ indicates that the clustering process favours the numeric attributes, whereas a large γ means the result is biased towards the categorical attribute.

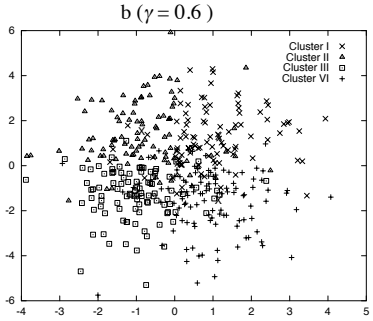
Figure 7 shows the results of clustering the data set in Figure 5b with three different γ values. When $\gamma = 0$ (Figure 7a), the data is clustered completely on the numeric attributes, equivalent to clustering the data in Figure 5a. It is known that the two dimensional points have only one normal distribution which represents one natural cluster. The clustering process divides the inherent one into four which does not make much sense from classification point of view.



a ($\gamma = 0$)

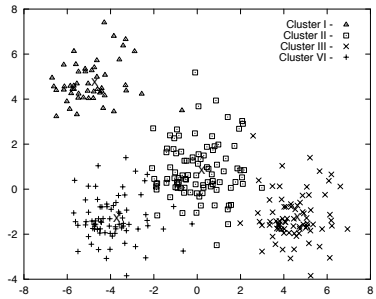


b ($\gamma = 0.6$)

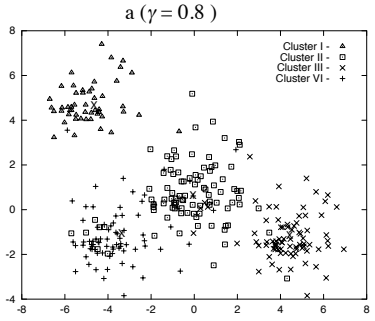


c ($\gamma = 1$)

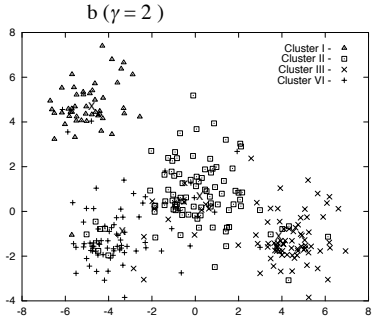
Figure 7.



a ($\gamma = 0.8$)



b ($\gamma = 2$)



c ($\gamma = 3$)

Figure 8.

However, when the third categorical dimension is involved, the inherent distribution is no longer valid in the three dimensional space (see Figure 5b). With some contribution ($\gamma = 0.6$) from the categorical dimension, four new clusters (Figure 7b) were produced which are different from those in Figure 7a. In comparison with Figure 5b, we can find the new clusters represent the four regions, each being dominated by one categorical value. Clustering of points is dependent on both numeric and categorical values. The points, which are spatially close and have the same categorical values, are clustered together. If a point has a categorical value A

but is close the majority of points having categorical value B and far away from the majority points having categorical value A, the point is clustered together with the points, which have categorical value B in majority. In this case, the two numeric values determine the cluster of the point rather than its categorical value. However, if a point having categorical value A and surrounded by points having categorical value B is still not far away from the points with categorical value A, it is clustered with the points with categorical value A. In this case, its cluster membership is determined by its categorical value rather than its spatial location. Therefore, numeric and categorical attributes play an equivalent role in determining cluster memberships of the points.

The result in Figure 7c ($\gamma = 1$) is biased on categorical attribute. Compared to Figure 5b the numeric attributes have played very little role in clustering except for few points.

Figure 8 shows the results of clustering data in Figure 6b. We know the data set has spatially four normal distributions. Figure 8a produced from a small γ clearly recovers the distributions because the clustering is biased towards numeric attributes. Figure 8b shows a medium γ result. In this clustering the categorical attribute has played a significant role. One can see a number of points which are spatially located in cluster VI were clustered in cluster II. However, points in cluster I are very homogenous because they are clustered according to their numeric attributes. This is also evident in Figure 8c which was produced from a large γ.

Selection of a γ value was guided by the average standard deviation σ of numeric attributes. The average standard deviation of data in Figure 5b is 1.658. A suitable γ to balance the similarity measure is between 0.5 and 0.7. The average standard deviation of data in Figure 6b is 3.185. A suitable γ lies between 1 and 2. Therefore, a suitable γ lies between $1/3\,\sigma$ and $2/3\,\sigma$ for these two data sets. However, more work needs to be done to understand the behaviour of γ.

4.3 Clustering a Real Data Set

The algorithm was tested against a real world data set consisting of more than 70,000 records, 9 numeric and 11 categorical attributes. Each record represents a motor insurance policy. The *k-prototypes* algorithm was used to partition the data set into homogenous clusters in order to identify the groups of policy holders which lodged more claims than other groups.

We ran the algorithm on a SUN Sparc10 workstation. Table 1 gives the time performance for partitioning different numbers of policies into 8 clusters. Table 2 shows the time performance for partitioning 30000 policies into different numbers of clusters. The termination condition for these experiments was that no policy changed clusters after a reallocation process. If iteration of the reallocation process was

terminated at some point when the number of policies changing clusters in a reallocation process drops very slowly (see Figure 4), running time would be reduced significantly.

Table 1

Records	Time (sec.)
10000	327
30000	824
50000	1267
70000	2051

Table 2

Clusters	Time(sec.)
16	2083
32	5108
64	10740
128	22380

To interpret the clusters, we use a decision tree induction algorithm to derive descriptions of clusters. For example, the description of a cluster in a partition may look like

Age of Policy Holders ≤ 20 **AND**
Sex = Male **AND**
5000 ≤ Insured Amount ≤ 10000

These kinds of descriptions, together with some statistics for clusters such as the number of policies and average claim cost, can assist data miners to understand and identify interesting policy groups.

5 Summary and Future Work

We have presented the *k-prototypes* algorithm to cluster large real world data sets. This algorithm preserves the efficiency of the *k-means* algorithm but removes its numeric data only limitation. We have demonstrated that it is efficient for clustering large data sets with mixed numeric and categorical values. Such data sets often occur in data mining applications.

We have suggested using decision tree induction algorithms to obtain descriptions of clusters. Such descriptions can assist interpretation of clusters and thus aid data miners to understand and identify interesting object groups represented by clusters. We believe that integration of clustering and decision tree induction algorithms can become a useful tool for knowledge discovery in databases.

This work was motivated by the problem of dealing with large mixed data sets, which are common in data mining applications. When the size of a data set exceeds a certain limit, many existing algorithms will be no longer practical. Extremely large data sets in data mining challenge many existing algorithms which are effective for small data sets. We plan to test our algorithm on larger data sets and to pursue a parallel implementation if necessary. This initial work has also opened several research topics we can pursue, such as global optimisation, effective cluster interpretation, and tree-structured clustering[14] for mixed data.

Acknowledgments

The author wants to thank Mr Peter Milne and Drs Glenn Stone, Graham Williams and Phil Kirby for their valuable comments.

References

1. M. R. Anderberg, *Cluster Analysis for Applications*. Academic Press, New York (1973).
2. L. Bobrowski and J. C. Bezdek, *c-Means Clustering with the l_1 and l_∞ Norms*. IEEE Transactions on Systems, Man and Cybernetics, **21**(3), pp.545-554 (1991).
3. L. Brieman, J. H. Frieman, R. A. Olshen and C. J. Stone, *Classification and Regression Trees*. Wadsworth, Inc. (1984).
4. J. Buhmann and H. Kühnel, *Vector Quantisation with Complexity Costs*. IEEE Transactions on Information Theory, **39**, pp. 1133-1145 (1993).
5. B. Everitt, *Cluster Analysis*. Heinemann Educational Books Ltd. (1974).
6. D. H. Fisher, *Knowledge Acquisition Via Incremental Conceptual Clustering*. Machine Learning, **2**(2), pp.139-172 (1987)
7. D. E. Goldberg, *Genetic Algorithms in Search, Optimisation, and Machine Learning*. Addison-Wesley (1989).
8. J. C. Gower, *A General Coefficient of Similarity and Some of Its Properties*. BioMetrics, 27, pp.857-874 (1971).
9. D. J. Hand, *Discrimination and Classification*. John Wiley & Sons (1981).
10. S. Kirkpatrick, C. D. Gelatt and M. P. Vecchi, *Optimisation by Simulated Annealing*. Science, **220**(4598), pp.671-680 (1983).
11. M. Lebowitz, *Experiments with Incremental Concept Formation*. Machine Learning, **2**(2), pp.103-138 (1987).
12. J. B. MacQueen, *Some Methods for Classification and Analysis of Multivariate Observations*. Proceedings of the 5th Berkeley Symposium on Mathematical Statistics and Probability, pp.281-297 (1967).
13. R. S. Michalski and R. E. Stepp, *Automated Construction of Classifications: Clustering versus Numerical Taxonomy*. IEEE Transactions on Pattern Analysis and Machine Intelligence, **5**(5), pp.396-410 (1983).
14. D. Miller and K. Rose, *A Non-greedy Approach to Tree-Structured Clustering*. Pattern Recognition Letters, **15**, pp. 683-690 (1994).
15. J. R. Quinlan, *C4.5: Programs for Machine Learning*. Morgan Kaufmann Publishers (1993).
16. K., Rose, E. Gurewitz and G. Fox, A *Deterministic Annealing Approach to Clustering*. Pattern Recognition Letters, **11**, pp.589-594 (1990).