Efficient Shared Near Neighbours Clustering of Large Metric Data Sets

Stefano Lodi, Luisella Reami, and Claudio Sartori

University of Bologna, Department of Electronics, Computer Science and Systems,
CSITE-CNR, viale Risorgimento 2, 40136 Bologna, ITALY
{slodi,csartori}@deis.unibo.it

Abstract. Very few clustering methods are capable of clustering data without assuming the availability of operations which are defined only in strongly structured spaces, such as vector spaces. We propose an efficient data clustering method based on the shared near neighbours approach, which requires only a distance definition and is capable of discovering clusters of any shape. Using efficient data structures for querying metric data and a scheme for partitioning and sampling the data, the method can cluster effectively and efficiently data sets whose size exceeds the internal memory size.

1 Introduction

An important component of a data mining system is the *data clustering* method, whose purpose is to solve the following problem: Given a similarity measure between pairs of points of a multidimensional space and a data set on that space, find a partition of the data set such that similar points belong to the same member of the partition and dissimilar points belong to distinct members, without utilizing a priori knowledge about the data, and with the following additional constraints: (a) The use of computing resources must be minimized and (b) the data set is large, i.e. the size of the data set exceeds the internal memory size. Note that (a) and (b) imply that I/O cost has to be minimized.

In the literature, a collection of desirable additional features is usually considered, which can be summarized as follows. The method should have the ability to discover clusters of arbitrary density, shape and relative position, or weakly separated, and to detect outliers. The method should need a minimal number of input parameters and be moderately sensitive to their value.

Many existing methods satisfy the above requirements. However, assumptions are often made about the mathematical structure of the space, e.g., the space is a vector space over the reals. Such assumptions usually allow for improvements in the effectiveness and efficiency of the method but result in a loss of generality. In particular, methods requiring the ability to compute operations on numbers, or a total order relation, cannot be applied easily to categorical data.

We present here a revised shared near neighbours clustering method which is applicable to large metric data sets. The only requirement is that the dissimilarity measure is a distance function. The method requires little additional external memory and uses a metric access method, the multi-vantage point tree, to organize data points in the available internal memory and efficiently retrieve neighbourhood information. Large data sets are dealt with by performing partial clustering steps, sampling the results and finally clustering the union of all samples.

2 Clustering Large Metric Data Sets

The shared near neighbours (SNN) cluster analysis method was introduced in [1]. First we formally define clusters according to the SNN approach, then we explain our new modified algorithm based on the approach.

Let X be a data set of N multidimensional points, $(Y, <)$ a totally ordered set, and $d: X \rightarrow Y$ a dissimilarity function. For any point $x \in X$, let x_i be an enumeration of $X - \{x\}$ that orders points by their dissimilarity from x, namely such that $d(x, x_p) < d(x, x_q)$ only if $p < q$. Let $\mathrm{NN}(x, k)$ be the set of the k first neighbours of x in the enumeration, i.e. $\{x_1, \ldots, x_k\}$. For k, $t \in \mathbb{N}$, $k \geq t$, define a binary relation $R_{k,t}$ on X as follows: $R_{k,t}(x, y)$ iff $x \in \mathrm{NN}(y, k)$, $y \in \mathrm{NN}(x, k)$, $|\mathrm{NN}(x, k) \cap \mathrm{NN}(y, k)| \geq t$. Recall that a set A is closed w.r.t. a binary relation R (or R-closed) if $a \in A$, $R(a, b)$ imply $b \in A$. A *cluster* in X w.r.t. k, t is a least $R_{k,t}$-closed subset of X.

The SNN method fulfils many traditional requirements well. Clusters of different shapes, densities and relative positions are recognized correctly, and the method has the ability to separate regions when there is a sharp gradient between them. Different orderings of the data do not affect the result in practice. The globularity of the clusters increases with k, whereas increasing t tends to split large clusters into subclusters. Both parameters influence the method's behaviour smoothly and predictably.

Nevertheless, the method has two main drawbacks when employed in database environment. Firstly, computing the neighbourhood table by brute force costs $O(N^2)$ time, where N is the number of points in the data set. Secondly, the table is sized $O(N)$. Thus, if the data set is substantially larger than the size of internal memory, then we should expect that it cannot be clustered.

The first problem is addressed in [2], where points are stored into a multidimensional access structure, the KD-tree. The cost of creating a KD-tree is $O(N \log N)$ and a single k nearest neighbour query costs $O(\log N)$, resulting in dramatic improvements in the cost of filling the neighbourhood table. However, the KD-tree performs well only when dimensionality is low [3]. Moreover, it can index only numerical data, thus restricting the generality of the approach. In contrast, high dimensionality and categorical properties are not uncommon in clustering applications.

To address the first problem, we propose to use the multi-vantage point tree (mvp-tree) [4] as an access method for the SNN approach. The mvp-tree is designed to minimize distance computations, therefore it is expected to perform well in high dimensional spaces. Moreover, it only requires a distance, thus it does not rule out the possibility of clustering categorical data.

The problem of data sets exceeding internal memory can be addressed by adopting a two-phase clustering strategy, as follows. In phase one, the data set is partitioned and every partition is loaded and clustered in an internal memory area of fixed capacity. For every partition, a phase one labeling vector is stored in external memory and, from every cluster in the partition, a number of points proportional to its size is randomly sampled and stored with its label. In phase two, the union of all samples is clustered again, and a map of phase one labels onto new labels is generated. Finally, every point in the data set is given the new label its phase one label maps to. The following algorithm realizes in more detail the above approach.

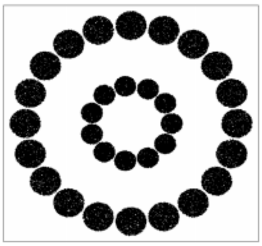
- Input: A data set B of points in a fixed number of dimensions. Integers k, t, *SampleSize*, *MinClSize*. Real δ.
- Output: A clustering vector V.
- Data Structures:
  - A list D of multidimensional points of capacity *NMax*.
  - A set of neighbour lists L_i each of size k, for $i = 1, \dots, NMax$.
  - An array C to store cluster information for the current partition. The array is organized as a set of circular linked lists: The i-th element of the array contains the cluster label of the i-th point in D and the index of the next point in the cluster.
  - An array *CS* to store cluster information for the samples, organized as a set of circular linked lists. The i-th element of the array contains the cluster label of the i-th randomly sampled point and the index of the next point in the cluster.
  - A list S of data points to store sampled data.
  - A sequential structure P in external memory of partially labeled data points.
  - A list M of pairs of cluster labels, representing a mapping on partial cluster labels onto final cluster labels.
- Algorithm:
  1. Repeat until the end of the data set is reached:
     a) Load points of the data set into D until either *NMax* points have been loaded or the end of the data set is reached.
     b) Create a mvp-tree on D.
     c) For every point i in D, compute the neighbour list $L_i = i_1, \dots, i_{k_i}$, $k_i \leq k$, obtained as the result of the query $\mathrm{NN}(i,k) \cap \mathrm{RANGE}(i,\delta)$, where $\mathrm{RANGE}(i,\delta)$ denotes the result of a range query of radius δ centered in i.
     d) Initialize the array C. The i-th element has cluster label i and next point i.
     e) For all pairs i, j, $i < j$:
        i. Test whether i occurs in L_j, j occurs in L_i, and L_i, L_j have at least t points in common.
        ii. If the test succeeds, update C by setting the label of every point in $c[j]$ (the cluster containing j) to i and merging in C the circular lists containing i and j.

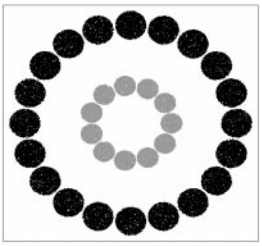
```
    f) For every cluster $c$ in $C$:
        i. if the size of the cluster $|c|$ is less than $MinClSize$, then update
           $C$ by setting the label of every point in $c$ to a unique "outlier"
           label. Continue with next cluster in $C$.
       ii. Compute the number of points to extract from the cluster as
           $s_c = \lceil SampleSize|c|/NMax \rceil$.
      iii. Extract a sample $s$ of size $s_c$ and store it in $S$.
       iv. Generate a new unique label $\ell_c$ for the cluster $c$ and update $C$
           by setting the label of every point in $c$ to $\ell_c$. Create a new entry
           in $M$ for $\ell_c$.
        v. Create a new circular linked list in $CS$. Insert every sampled
           point $j$ of $s$ in the new list, setting the point's label to $\ell_c$.
    g) For every point $i$ in $D$, read the partial label of $i$ from $C$ and append
       it to $P$.
2.  a) Create a mvp-tree on $S$.
    b) For every point $i$ in $S$, compute the neighbour list $L_i = i_1, \ldots, i_k$,
       obtained as the result of the query $\mathrm{NN}(i, k)$.
    c) For all pairs $i, j$, $i < j$:
        i. Let $\ell_i$, $\ell_j$ be the labels of $i$, $j$ in $CS$.
       ii. Test whether $i$ occurs in $L_j$, $j$ occurs in $L_i$, and $L_i$, $L_j$ have at
           least $t$ points in common.
      iii. If the test succeeds, update $CS$ by setting the label of every point
           in $c[j]$ to $\ell_i$, merge in $CS$ the circular lists containing $i$ and $j$,
           and update $M$ by setting the final label to $\ell_i$ where the partial
           label equals $\ell_j$.
    d) Scan $P$. For every point in $P$, read its partial current label from $P$,
       map it to the final cluster label through $M$ and append to $V$ the
       final label.
```

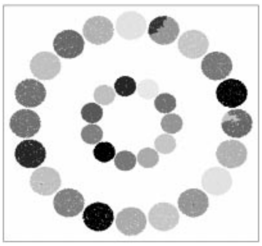
Three new parameters *SampleSize*, *MinClSize*, and δ have been added (with respect to the original SNN algorithm) with the following meaning. *SampleSize* represents the number of randomly sampled points for every partition. *MinClSize* is the minimum size of clusters that will be considered. δ is the maximum distance of nearest neighbours that will be in the neighbours list for a point. *MinClSize* and δ may be used to detect outliers, when the notion of outlier is distance based, i.e. it implies an estimate of the maximum distance between points in a cluster. Setting δ to a value smaller than this distance will cause outliers to be grouped into clusters of small size, which can be eliminated by setting *MinClSize* to a suitably large cardinal.

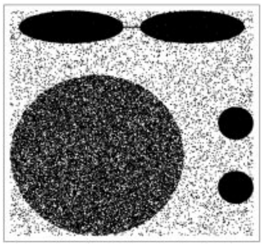
3 Experimental Results and Conclusions

We have evaluated the performance of our algorithm by conducting experiments on synthetic two dimensional data sets. In all tests, the distance function is Euclidean, the partition size is $NMax = 10000$, the architecture is Intel i586


(a) set 1


(b) results ($t = 5$)


(c) results ($t = 10$)


(d) set 2

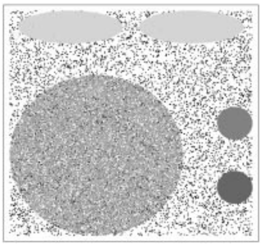

(e) results

Fig. 1. Test data sets

at 133 MHz with 128 MByte of internal memory, and the operating system is Windows NT 4.0.

Data set 1 in figure 1 is based on a data set introduced to test the WaveCluster algorithm [5]. It is composed of 64188 points in 31 spherical clusters. The algorithm recognizes two ring shaped clusters for $t = 5$, and 37 mostly spherical clusters for $t = 10$. k is set to 16.

Data set 2 has been introduced as a test for the CURE algorithm [6]. It has 100000 points distributed as follows: Three spherical clusters of different density, a dense region made by two ellipsoids connected by a dense chain of points, and randomly placed outliers. The clustering result was obtained by setting $k = 10$, $t = 3$, $\delta = 0.27$, *MinClSize* = 20, *SampleSize* = 50. The algorithm recognizes four clusters and groups outliers correctly. The connected ellipsoids are merged since the distance between points in the chain does not differ substantially from the distance between points in the regions.

We compared the speed performance of our algorithm with one of the fastest and widely referenced available algorithms, DBSCAN [7]. Three data sets with 20000, 60000 and 100000 points have been used for the comparison. In all data sets, 10% of the points are outliers and the remaining points are equally distributed among three spheres of different radius. Since the performance of

DBSCAN is affected by the values of the two input parameters, we chose the values that resulted in the best performance, among the values giving the correct clustering. On the three data sets, DBSCAN terminated the tree creation phase in 59.0, 181.0, and 310.0 seconds, respectively, and the clustering phase in 49.4, 151.3, and 282.8 seconds, whereas our algorithm finished in 39.3, 118.0, and 212.6 seconds.

These results show that the performance of the algorithm is not inferior to the performance of DBSCAN in the clustering phase. Therefore, the algorithm is suitable for employment when an access structure is not available. When compared to CURE, the algorithm is not immune to the well-known chaining problem (see figure 1). However, the robustness of CURE is obtained within explicit limitations in the authors' approach, namely that data are numeric. Furthermore, in a metric approach, the very notion of chaining seems problematic. Future work will include a validation with high-dimensional data sets, experiments with real data benchmarks and the extension of the algorithm for incremental clustering.

Acknowledgments The authors would like to thank Andrea Lodi, Rossella Miglio, Angela Montanari, Marco Patella, and Gabriele Soffritti for helpful discussions on the subject.

References

1. R. Jarvis and E. Patrick. Clustering using a similarity measure based on shared near neighbours. *IEEE Transactions on Computers*, 22(11):1025–1034, November 1973.
2. R. Jarvis and I. Hofman. Robust and efficient cluster analysis using a shared near neighbours approach. In *ICPR'98, Proc. of the 14th Int'l Conference on Pattern Recognition*, pages 243–247. IEEE Computer Society, 1998.
3. Jeffrey K. Uhlmann. Satisfying general proximity/similarity queries with metric trees. *Information Processing Letters*, 40(4):175–179, 25 November 1991.
4. T. Bozkaya and Z. M. Özoyoglu. Distance-based indexing for high-dimensional metric spaces. In *Proceedings of the 1997 ACM SIGMOD International Conference on Management of Data*, pages 357–368. ACM Press, 1997.
5. G. Sheikholeslami, S. Chatterjee, and A. Zhang. WaveCluster: A multi-resolution clustering approach for very large spatial databases. In *VLDB'98, Proceedings of 24th International Conference on Very Large Data Bases*, pages 428–439. Morgan Kaufmann, 1998.
6. S. Guha, R. Rastogi, and K. Shim. CURE: An efficient clustering algorithm for large databases. In *Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMOD-98)*, pages 73–84. ACM Press, 1998.
7. M. Ester, H.-P. Kriegel, J. Sander, and X. Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining (KDD-96)*, page 226. AAAI Press, 1996.