Cluster Cores-based Clustering for High Dimensional Data

Yi-Dong Shen, Zhi-Yong Shen and Shi-Ming Zhang
Laboratory of Computer Science
Institute of Software, Chinese Academy of Sciences
Beijing 100080, China
{ydshen, zyshen, zhangsm}@ios.ac.cn

Qiang Yang
Department of Computer Science
Hong Kong University of Science and Technology
Clearwater Bay, Kowloon, Hong Kong
qyang@cs.ust.hk

Abstract

In this paper we propose a new approach to clustering high dimensional data based on a novel notion of cluster cores, instead of nearest neighbors. A cluster core is a fairly dense group with a maximal number of pairwise similar/related objects. It represents the core/center of a cluster, as all objects in a cluster are with a great degree attracted to it. As a result, building clusters from cluster cores achieves high accuracy. Other characteristics of our cluster cores-based approach include: (1) It does not incur the curse of dimensionality and is scalable linearly with the dimensionality of data. (2) Outliers are effectively handled with cluster cores. (3) Clusters are efficiently computed by applying existing algorithms in graph theory. (4) It outperforms both in efficiency and in accuracy the well-known clustering algorithm, ROCK.

1 Introduction

Clustering is a major technique for data mining, together with association mining and classification [4, 11, 21]. It divides a set of objects (data points) into groups (clusters) such that the objects in a cluster are more similar (or related) to one another than to the objects in other clusters. Although clustering has been extensively studied for many years in statistics, pattern recognition, and machine learning (see [14, 15] for a review), as Agrawal et al [3] point out, emerging data mining applications place many special requirements on clustering techniques, such as scalability with high dimensionality of data. A number of clustering algorithms have been developed over the last decade in the data base/data mining community (e.g., DBSCAN [5], CURE [9], CHAMELEON [17], CLARANS [18], STING [20], and BIRCH [22]). Most of these algorithms rely on a distance function (such as the Euclidean distance or the Jaccard distance that measures the similarity between two objects) such that objects are in the same cluster if they are nearest neighbors. However, recent research shows that clustering by distance similarity is not scalable with the dimensionality of data because it suffers from the so-called *curse of dimensionality* [13, 12], which says that for moderate-to-high dimensional spaces (tens or hundreds of dimensions), a full-dimensional distance is often irrelevant since the farthest neighbor of a data point is expected to be almost as close as its nearest neighbor [12, 19]. As a result, the effectiveness/accuracy of a distance-based clustering method would decrease significantly with increase of dimensionality. This suggests that "shortest distances/nearest neighbors" are not a robust criterion in clustering high dimensional data.

1.1 Contributions of the Paper

To resolve the curse of dimensionality, we propose a new definition of clusters that is based on a novel concept of *cluster cores*, instead of nearest neighbors. A cluster core is a fairly dense group with a maximal number of pairwise similar objects (neighbors). It represents the core/center of a cluster so that all objects in a cluster are with a great degree attracted to it. This allows us to define a cluster to be an expansion of a cluster core such that every object in the cluster is similar to most of the objects in the core.

Instead of using Euclidean or Jaccard distances, we define the similarity of objects by taking into account the meaning (semantics) of individual attributes. In particular, two objects are *similar* (or *neighbors*) if a certain number of their attributes take on similar values. Whether two values of an attribute are similar is semantically dependent on applications and is defined by the user. Note that since any two objects are either similar/neighbors or not, the concept of nearest neighbors does not apply in our approach.

Our definition of clusters shows several advantages. Firstly, since the similarity of objects is measured semantically w.r.t. the user's application purposes, the resulting clusters would be more easily understandable by the user. Secondly, a cluster core represents the core/center of a clus-

ter, with the property that a unique cluster is defined given a cluster core and that all objects in a cluster are with a great degree attracted to the core. Due to this, the cluster cores-based method achieves high accuracy. Thirdly, since clusters are not defined in terms of nearest neighbors, our method does not incur the curse of dimensionality and is scalable linearly with the dimensionality of data. Finally, outliers are effectively eliminated by cluster cores, as an outlier would be similar to no or just a very few objects in a core.

1.2 Related Work

In addition to dimensionality reduction (e.g., the principal component analysis (PCA) [14, 8]) and projective/subspace clustering (e.g., CLIQUE [3], ORCLUS [2] and DOC [19]), more closely related work on resolving the dimensionality curse with high dimensional data includes shared nearest neighbor methods, such as the Jarvis-Patrick method [16], SNN [6], and ROCK [10]. Like our cluster cores-based method, they do not rely on the shortest distance/nearest neighbor criterion. Instead, objects are clustered in terms of how many neighbors they share. A major operation of these methods is to merge small clusters into bigger ones. The key idea behind the Jarvis-Patrick method is that if any two objects share more than T (specified by the user) neighbors, then the two objects and any cluster they are part of can be merged. SNN extends the Jarvis-Patrick method by posing one stronger constraint: two clusters C_i and C_j could be merged if there are two *representative* objects, $o_i \in C_i$ and $o_j \in C_j$, which share more than T neighbors. An object A is a representative object if there are more than T_1 objects each of which shares more than T neighbors with A. To apply SNN, the user has to specify several thresholds including T and T_1. ROCK is a sophisticated agglomerative (i.e. bottom-up) hierarchical clustering approach. It tries to maximize links (common neighbors) within a cluster while minimizing links between clusters by applying a criterion function such that any two clusters C_i and C_j could be merged if their goodness value $g(C_i, C_j)$ defined below is the largest:

$$\frac{link[C_i, C_j]}{(|C_i| + |C_j|)^{1+2f(\theta)} - |C_i|^{1+2f(\theta)} - |C_j|^{1+2f(\theta)}}$$

where $link[C_i, C_j]$ is the number of common neighbors between C_i and C_j, θ is a threshold for two objects to be similar under the Jaccard distance measure (see Equation (2)), and $f(\theta)$ is a function with the property that for any cluster C_i, each object in it has approximately $|C_i|^{f(\theta)}$ neighbors. The time complexity of ROCK is $O(|DB|^2 * log|DB| + |DB| * \overline{D}_G * m)$, where DB is a dataset, and $\overline{D}_G$ and m are respectively the maximum number and average number of neighbors for an object. To apply ROCK, the user needs to provide a threshold θ and define a function $f(\theta)$. However, as the authors admit [10], it may not be easy to decide on an appropriate function $f(\theta)$ for different applications.

Unlike the above mentioned shared nearest neighbor approaches, our cluster cores-based method does not perform any merge operations. It requires the user to specify only one threshold: the minimum size of a cluster core. We will show that our method outperforms ROCK both in clustering efficiency (see Corollary 4.3) and in accuracy (see experimental results in Section 5).

2 Semantics-based Similarity

Let $A_1, ..., A_d$ be d attributes (dimensions) and $V_1, ..., V_d$ be their respective domains (i.e. A_i takes on values from V_i). Each V_i can be finite or infinite, and the values in V_i can be either numerical (e.g., 25.5 for price) or categorical (e.g., $blue$ for color). A dataset (or database) DB is a finite set of objects (or data points), each o of which is of the form $(id_o, a_1, ..., a_d)$ where id_o is a natural number representing the unique identity of object o, and $a_i \in V_i$ or $a_i = nil$. nil is a special symbol not appearing in any V_i, and $a_i = nil$ represents that the value of A_i is not present in this object. For convenience of presentation, we assume the name o of an object is the same as its identity number id_o.

To cluster objects in DB, we first define a measure of similarity. The Euclidean distance and the Jaccard distance are two similarity measures which are most widely used by existing clustering methods. Let $o_1 = (o_1, a_1, ..., a_d)$ and $o_2 = (o_2, b_1, ..., b_d)$ be two objects. When the a_is and b_js are all numerical values, the Euclidean distance of o_1 and o_2 is computed using the formula

$$distance(o_1, o_2) = (\sum_{k=1}^{d} (a_k - b_k)^2)^{1/2} \qquad (1)$$

When the a_is and b_js are categorical values, the Jaccard distance of o_1 and o_2 is given by

$$distance(o_1, o_2) = 1 - \frac{\sum_{k=1}^{d} a_k \bullet b_k}{2d - \sum_{k=1}^{d} a_k \bullet b_k} \qquad (2)$$

where $a_k \bullet b_k = 1$ if $a_k = b_k \neq nil$; otherwise $a_k \bullet b_k = 0$. The similarity of objects is then measured such that for any objects o_1, o_2 and o_3, o_1 is *more similar* (or *closer*) to o_2 than to o_3 if $distance(o_1, o_2) < distance(o_1, o_3)$. o_2 is a *nearest neighbor* of o_1 if $distance(o_1, o_2) \leq distance(o_1, o_3)$ for all $o_3 \in DB - \{o_1, o_2\}$.

As we mentioned earlier, clustering by a distance similarity measure like (1) or (2) suffers from the curse of dimensionality. In this section, we introduce a non-distance measure that relies on the semantics of individual attributes.

We begin by defining the similarity of two values of an attribute. For a numerical attribute, two values are considered similar w.r.t. a specific application if their difference is within a scope specified by the user. For a categorical attribute, however, two values are viewed similar if they are in the same class/partition of the domain. The domain is partitioned by the user based on his/her application purposes. Formally, we have

Definition 2.1 (Similarity of two numerical values) Let A be an attribute and V be its domain with numerical values. Let $\omega \geq 0$ be a scope specified by the user. $a_1 \in V$ and $a_2 \in V$ are *similar* if $|a_1 - a_2| \leq \omega$.

Definition 2.2 (Similarity of two categorical values) Let A be an attribute and $V = V_1 \cup V_2 \cup ... \cup V_m$ be its domain of categorical values, with each V_i being a partition w.r.t the user's application purposes. $a_1 \in V$ and $a_2 \in V$ are *similar* if both are in some V_i.

For instance, we may say that two people are similar in *age* if the difference of their ages is below 10 years, and similar in *salary* if the difference is not over \$500. We may also view two people similar in *profession* if both of their jobs belong to the group {soldier, police, guard} or the group {teacher, researcher, doctor}. The similarity of attribute values may vary from application to application. As an example, let us consider an attribute, *city*, with a domain $V_{city} = \{$Beijing, Shanghai, Guangzhou, Chongqing, Lasa, Wulumuqi$\}$. For the user of government officials, V_{city} would be partitioned into {Beijing, Shanghai, Chongqing} $\cup$ {Lasa, Wulumuqi} $\cup$ {Guangzhou}, as Beijing, Shanghai and Chongqing are all municipalities directly under the Chinese Central Government, while Lasa and Wulumuqi are in two autonomous regions of China. However, for the user who is studying the SARS epidemic, V_{city} would be partitioned into {Beijing, Guangzhou} $\cup$ {Shanghai, Chongqing} $\cup$ {Lasa, Wulumuqi}, as in 2003, Beijing and Guangzhou were two most severe regions in China with the SARS epidemic, Shanghai and Chongqing had just a few cases, and Lasa and Wulumuqi were among the safest zones with no infections.

With the similarity measure of attribute values defined above, the user can then measure the similarity of two objects by counting their similar attribute values. If the number of similar attribute values is above a threshold δ, the two objects can be considered similar w.r.t. the user's application purposes. Here is the formal definition.

Definition 2.3 (Similarity of two objects) Let o_1 and o_2 be two d-dimensional objects and $\mathcal{K}$ be a set of key attributes w.r.t. the user's application purposes. Let δ be a *similarity threshold* with $0 < \delta \leq |\mathcal{K}| \leq d$. o_1 and o_2 are *similar* (or *neighbors*) if they have at least δ attributes in $\mathcal{K}$ that take on similar values.

The set of key attributes is specified by the user. Whether an attribute is selected as a key attribute depends on whether it is relevant to the user's application purposes. In case that the user has difficulty specifying key attributes, all attributes are taken as key attributes. The similarity threshold δ is expected to be specified by the user. It can also be elicited from a dataset by trying several possible choices (at most d alternatives) in clustering/learning experiments. The best similarity threshold is one that leads to a satisfying clustering accuracy (see Section 5). Note that since any two objects are either similar/neighbors or not, the concept of nearest neighbors does not apply in our approach.

3 Cluster Cores and Disjoint Clusters

The intuition behind our clustering method is that every cluster is believed to have its own distinct characteristics, which are implicitly present in some objects in a dataset. This suggests that the principal characteristics of a cluster may be represented by a certain number of objects C^r such that (1) any two objects in C^r are similar and (2) no other objects in the dataset can be added to C^r without affecting property (1). Formally, we have

Definition 3.1 (Cluster cores) $C^r \subseteq DB$ is a *cluster core* if it satisfies the following three conditions. (1) $|C^r| \geq \alpha$, where α is a threshold specifying the minimum size of a cluster core. (2) Any two objects in C^r are similar w.r.t. the user's application purposes. (3) There exists no C', with $C^r \subset C' \subseteq DB$, that satisfies condition (2). C^r is a *maximum cluster core* if it is a cluster core with the maximum cardinality.

Note that a cluster core C^r is a fairly dense group with a maximal number of pairwise similar/related objects. Due to this, it may well be treated as the core/center of a cluster, as other objects of the cluster not in C^r must be attracted with a great degree to C^r.

Definition 3.2 (Clusters) Let $C^r \subseteq DB$ be a cluster core and let θ be a *cluster threshold* with $0 \leq \theta \leq 1$. C is a *cluster* if $C = \{v \in DB | v \in C^r$ or C^r contains at least $\theta * |C^r|$ objects that are similar to $v\}$.

The threshold θ is the support (degree) of an object being attracted to a cluster core. Such a parameter can be learned from a dataset by experiments (see Section 5). Note that a unique cluster is defined given a cluster core. However, a cluster may be derived from different cluster cores, as the core/center of a cluster can be described with different sets of objects that satisfy the three conditions of Definition 3.1.

The above definition of cluster cores/clusters allows us to apply existing graph theory to compute them. We first define a similarity graph over a dataset.

Definition 3.3 (Similarity graphs) A *similarity graph*, $SG_{DB} = (V, E)$, of DB is an undirected graph where $V = DB$ is the set of nodes, and E is the set of edges such that $\{o_1, o_2\} \in E$ if objects o_1 and o_2 are similar.

Example 3.1 Let us consider a sample dataset, DB_1, as shown in Table 1, where nil is replaced by a blank. For simplicity, assume all attributes have a domain $\{1\}$ with a single partition $\{1\}$. Thus, two values, v_1 and v_2, of an attribute are similar if $v_1 = v_2 = 1$. Let us assume all attributes are key attributes, and choose the similarity threshold $\delta \geq 2$. The similarity relationship between objects of DB_1 is depicted by a similarity graph SG_{DB_1} shown in Figure 1.

Table 1. A sample dataset DB_1.

id	A_1	A_2	A_3	A_4	A_5	A_6	A_7	A_8	A_9	A_{10}	A_{11}
1	1	1	1								
2		1	1	1							
3	1		1	1							
4	1	1		1	1	1	1	1			
5				1	1	1	1	1		1	1
6								1	1	1	
7									1	1	1
8								1		1	

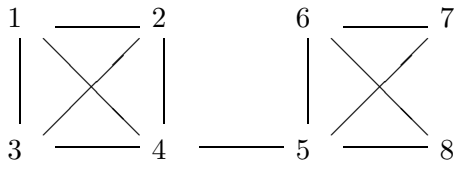

Figure 1. A similarity graph SG_{DB_1}.

Let $G = (V, E)$ be an undirected graph and $V' \subseteq V$. We denote $G(V')$ for the subgraph of G induced by V'; namely, $G(V') = (V', E')$ such that for any $v_1, v_2 \in V'$, $\{v_1, v_2\} \in E'$ if and only if $\{v_1, v_2\} \in E$. G is *complete* if its nodes are pairwise adjacent, i.e. for any two different nodes v_1, $v_2 \in V$, $\{v_1, v_2\} \in E$. A *clique* C is a subset of V such that $G(C)$ is complete. A *maximal clique* is a clique that is not a proper subset of any other clique. A *maximum clique* is a maximal clique that has the maximum cardinality. It turns out that a cluster core in DB corresponds to a maximal clique in SG_{DB}.

Theorem 3.1 *Let $C^r \subseteq DB$ with $|C^r| \geq \alpha$. C^r is a cluster core if and only if it is a maximal clique in SG_{DB}. C^r is a maximum cluster core if and only if it is a maximum clique in SG_{DB}.*

Similarly, we have the following result.

Theorem 3.2 *Let $C^r \subseteq DB$ with $|C^r| \geq \alpha$ and let $V = \{v \notin C^r | v$ is a node in SG_{DB} and there are at least $\theta * |C^r|$ nodes in C^r that are adjacent to $v\}$. C is a cluster with a cluster core C^r if and only if C^r is a maximal clique in SG_{DB} and $C = C^r \cup V$.*

Example 3.2 In Figure 1, we have four maximal cliques: $C_1 = \{1, 2, 3, 4\}$, $C_2 = \{4, 5\}$, $C_3 = \{5, 6, 7\}$ and $C_4 = \{5, 6, 8\}$. If we choose the least cluster size $\alpha = 2$, all of the four maximal cliques are cluster cores. Let us choose $\alpha = 3$ and the cluster threshold $\theta = 0.6$. C_1, C_3 and C_4 are cluster cores. C_1 is also a cluster and $\{5, 6, 7, 8\}$ is a cluster which can be obtained by expanding either C_3 or C_4.

From Example 3.2, we see that there may be overlaps among cluster cores and/or clusters. In this paper we are devoted to building disjoint clusters.

Definition 3.4 (Disjoint clusters) $DC(DB) = \{C_1, ..., C_m\}$ is a collection of *disjoint clusters* of DB if it satisfies the following condition: C_1 is a cluster in DB, and for any $i > 1$, C_i is a cluster in $DB - \bigcup_{j=1}^{j=i-1} C_j$.

Consider Figure 1 again. Let $\alpha = 2$ and $\theta = 0.6$. We have several collections of disjoint clusters including: $DC(DB_1)_1 = \{\{1, 2, 3, 4\}, \{5, 6, 7, 8\}\}$ and $DC(DB_1)_2 = \{\{4, 5\}, \{1, 2, 3\}, \{6, 7\}\}$. Apparently, we would rather choose $DC(DB_1)_1$ than $DC(DB_1)_2$. To obtain such optimal ones, we introduce a notion of maximum disjoint clusters.

Definition 3.5 (Maximum disjoint clusters) $DC(DB) = \{C_1, ..., C_m\}$ is a collection of *maximum disjoint clusters* if it is a collection of disjoint clusters with the property that each C_i is a cluster with a maximum cluster core in $DB - \bigcup_{j=1}^{j=i-1} C_j$.

Back to the above example, $DC(DB_1)_1$ is a collection of maximum disjoint clusters, but $DC(DB_1)_2$ is not, as $\{4, 5\}$ is not a maximum cluster core in DB_1.

4 Approximating Maximum Clusters

Ideally, we would like to have maximum disjoint clusters. However, this appears to be infeasible for a massive dataset DB, as computing a maximum cluster core over DB amounts to computing a maximum clique over its corresponding similarity graph SG_{DB} (Theorem 3.1). The

maximum clique problem is one of the first problems shown to be NP-complete, which means that, unless P=NP, exact algorithms are guaranteed to return a maximum clique in a time which increases exponentially with the number of nodes in SG_{DB}.

In this section, we describe our algorithm for approximating maximum disjoint clusters. We begin with an algorithm for computing a maximal clique. For a node v in a graph G, we use $AD_G(v)$ to denote the set of nodes adjacent to v, and refer to $|AD_G(v)|$ as the *degree* of v in G.

Algorithm 1: Computing a maximal clique.
Input: An undirected graph $G = (V, E)$.
Output: A maximal clique C in G.
function $maximal\_clique(V, E)$ **returns** C
1) $C = \emptyset$;
2) $V' = V$;
3) **while** ($V' \neq \emptyset$) **do begin**
4) Select v at random from V';
5) $C = C \cup \{v\}$;
6) $V' = AD_{G(V')}(v)$
7) **end**
8) **return** C
end

Since there may be numerous maximal cliques in a graph, Algorithm 1 builds one in a randomized way. Initially, C is an empty clique (line 1). In each cycle of lines 3 - 7, a new node is added to C. After $k \geq 0$ cycles, C grows into a clique $\{v_1, ..., v_k\}$. In the beginning of cycle k+1, V' consists of all nodes adjacent to every node in C. Then, v_{k+1} is randomly selected from V' and added to C (lines 4-5). The iteration continues with $V' = AD_{G(V')}(v_{k+1})$ (line 6) until V' becomes empty (line 3).

Let $\overline{D}_G$ be the maximum degree of nodes in G and let S be the (sorted) set of nodes in G that are adjacent to a node v. The time complexity of line 6 is bounded by $O(\overline{D}_G)$, as computing $AD_{G(V')}(v)$ is to compute the intersection of the two sorted sets S and V'. Thus we have the following immediate result.

Theorem 4.1 *Algorithm 1 constructs a maximal clique with the time complexity* $O(\overline{D}_G^2)$.

Algorithm 1 randomly builds a maximal clique. This suggests that a maximum clique could be approximated if we apply Algorithm 1 iteratively for numerous times. Here is such an algorithm (borrowed from Abello et al. [1, 7] with slight modification).

Algorithm 2: Approximating a maximum clique.
Input: A graph $G = (V, E)$ and an integer $maxitr$.
Output: A maximal clique C^r.
function $maximum\_clique(V, E, maxitr)$ **returns** C^r
1) $C^r = \emptyset$;
2) $i = 0$;
3) **while** ($i < maxitr$) **do begin**
4) $C = maximal\_clique(V, E)$;
5) **if** $|C| > |C^r|$ **then** $C^r = C$;
6) $i = i + 1$
7) **end**
8) **return** C^r
end

The parameter $maxitr$ specifies the times of iteration. In general, the bigger $maxitr$ is, the closer the final output C^r would be to a maximum clique. In practical applications, it would be enough to do $maxitr \leq \overline{D}_G$ iterations to obtain a maximal clique quite close to a maximum one. For instance, in Figure 1 choosing $maxitr = 2$ guarantees to find a maximum clique, where $\overline{D}_{SG_{DB_1}} = 4$.

By Theorem 3.1, Algorithm 2 approximates a maximum cluster core C^r when G is a similarity graph SG_{DB}. Therefore, by Theorem 3.2 an approximated maximum cluster can be built from C^r by adding those nodes v such that C^r contains at least $\theta * |C^r|$ nodes adjacent to v. So we are ready to introduce the algorithm for approximating maximum disjoint clusters over a similarity graph.

Algorithm 3: Approximating maximum disjoint clusters.
Input: $SG_{DB} = (V, E)$, $maxitr$, α and θ.
Output: $Q = \{C_1, ..., C_m\}$, a collection of approximated maximum disjoint clusters.
function $disjoint\_clusters(V, E, \alpha, \theta, maxitr)$ **returns** Q
1) $Q = \emptyset$;
2) $(V', E') = peel((V, E), \alpha)$;
3) **while** ($|V'| \geq \alpha$) **do begin**
4) $C^r = maximum\_clique(V', E', maxitr)$;
5) **if** $|C^r| < \alpha$ **then break**;
6) $C = C^r \cup \{v \in V' - C^r | C^r$ has at least $\theta * |C^r|$ nodes adjacent to $v\}$;
7) Add C to the end of Q;
8) $V_1 = V' - C$;
9) $(V', E') = peel(SG_{DB}(V_1), \alpha)$
10) **end**
11) **return** Q
end

Given a graph G, the function $peel(G, \alpha)$ updates G into a graph $G' = (V', E')$ by recursively removing (labeling with a flag) all nodes whose degrees are below the least size of a cluster core α (lines 2 and 9), as such nodes will not appear in any disjoint clusters. By calling the function $maximum\_clique(V', E', maxitr)$ (line 4), Algorithm 3 builds a collection of disjoint clusters $Q = \{C_1, ..., C_m\}$, where each C_i is an approximation of some maximum cluster in $DB - \bigcup_{j=1}^{j=i-1} C_j$.

Theorem 4.2 *When Algorithm 3 produces a collection of m disjoint clusters, its time complexity is $O(m * maxitr * \overline{D}_G^2)$.*

Since it costs $O(|DB|^2)$ in time to construct a similarity graph SG_{DB} from a dataset DB,[1] the following result is immediate to Theorem 4.2.

Corollary 4.3 *The overall time complexity of the cluster cores-based clustering is $O(|DB|^2 + m * maxitr * \overline{D}_G^2)$.*

In a massive high dimensional dataset DB, data points would be rather sparse so that $\overline{D}_G \ll |DB|$. Our experiments show that most of the time is spent in constructing SG_{DB} because $|DB|^2 \gg m * maxitr * \overline{D}_G^2$. Note that the complexity of our method is below that of ROCK.

5 Experimental Evaluation

We have implemented our cluster cores-based clustering method (Algorithms 1-3 together with a procedure for building a similarity graph SG_{DB}) and made extensive experiments over real-life datasets. Due to the limit of paper pages, in this section we only report experimental results on one dataset: the widely-used Mushroom data from the UCI Machine Learning Repository (we downloaded it from http://www.sgi.com/tech/mlc/db/). The Mushroom dataset consists of 8124 objects with 22 attributes each with a categorical domain. Each object has a class of either edible or poisonous. One major reason for us to choose this dataset is that ROCK [10] has achieved very high accuracy over it, so we can make a comparison with ROCK in clustering accuracy. Our experiments go through the following two steps.

Step 1: Learn a similarity threshold δ. In most cases, the user cannot provide a precise similarity threshold used for measuring the similarity of objects. This requires us to be able to learn it directly from a dataset.

For a dataset with d dimensions, δ has at most d possible choices. Then, with each $\delta_i \leq d$ we construct a similarity graph $SG_{DB}^{\delta_i}$ and apply Algorithm 3 with $\theta = 1$ to derive a collection of disjoint clusters. Note that since $\theta = 1$, each cluster is itself a cluster core. Next, we compute the clustering precision for each selected δ_i using the formula $precision(\delta_i, 1, |DB|) = \frac{N}{|DB|}$, where N is the number of objects sitting in right clusters. Finally, we choose $\delta = max_{\delta_i}\{precision(\delta_i, 1, |DB|)\}$ as the best threshold since under it the accuracy of a collection of disjoint cluster cores would be maximized.

Step 2: Learn a cluster threshold θ given δ. With the similarity threshold δ determined via Step 1, we further elicit a cluster threshold θ from the dataset, aiming to construct a collection of disjoint clusters with high accuracy.

With each $\theta_i \in \{0.8, 0.81, ..., 0.99, 1\}$ we apply Algorithm 3 to generate a collection of disjoint clusters and compute its precision using the formula $precision(\delta, \theta_i, |DB|)$. The best θ comes from $max_{\theta_i}\{precision(\delta, \theta_i, |DB|)\}$.

Note that θ_i begins with 0.8. This value was observed in our experiments. It may vary from application to application. In addition, to obtain clusters with largest possible sizes, one may apply different cluster thresholds: a relatively small one for large cluster cores and a relatively big one for small cluster cores. Again, such cluster thresholds can be observed during the learning process.

We performed our experiments on a PC computer with a Pentium 4 2.80GHz CPU and 1GB of RAM. To approximate maximum cluster cores (Algorithm 2), we set $maxitr = 10$ (i.e. we do 10 times of iteration). In Step 1, different similarity thresholds were tried, with different accuracy results produced as shown in Figure 2. Clearly, $\delta = 15$ is the best for the Mushroom dataset. With this similarity threshold, in Step 2 we applied different cluster thresholds and obtained different accuracy results as shown in Figure 3. We see that the highest accuracy is obtained at $\theta = 0.88$. The sizes of clusters that were produced with $\delta = 15$ and $\theta = 0.88$ are shown in Table 2 (C# stands for the cluster number and the results of ROCK were copied from [10]). Note that our method achieves higher accuracy than ROCK.

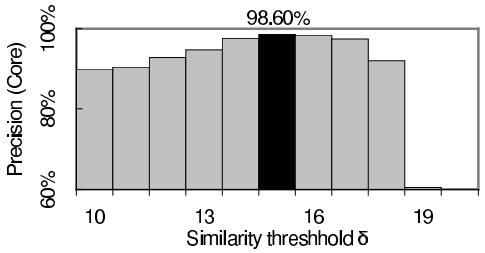

Figure 2. Accuracy with varying similarity thresholds.

To show that the two thresholds learned via Steps 1 and 2 are the best, we made one more experiment by learning the two parameters together. That is, for each $\delta_i \leq d$ and $\theta_j \in \{0.8, 0.81, ..., 0.99, 1\}$, we compute $precision(\delta_i, \theta_j, |DB|)$. If δ and θ selected in Steps 1 and 2 are the best, we expect $precision(\delta, \theta, |DB|) \approx max_{\delta_i, \theta_j}\{precision(\delta_j, \theta_j, |DB|)\}$. Apparently, Figure 4 confirms our expectation.

In addition to having high clustering accuracy, our cluster cores-based method can achieve high accuracy when being applied to make classifications. The basic idea is as

[1] It costs $O(d)$ to compute the similarity of two objects with d dimensions. Since $d \ll DB$, it is often ignored in most existing approaches.

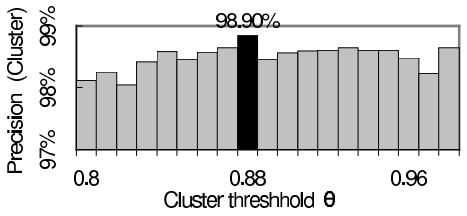

Figure 3. Accuracy with varying cluster thresholds given $\delta = 15$.

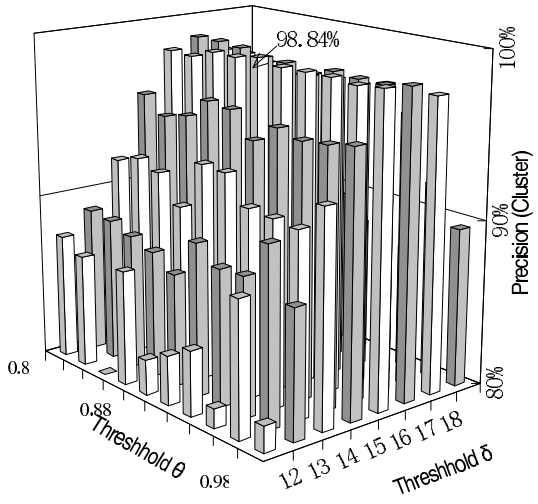

Figure 4. Accuracy with varying similarity and cluster thresholds.

Table 2. Clustering results (Mushroom data).

Cluster Cores-based Clustering					
C#	Edible	Poisonous	C#	Edible	Poisonous
1	1704	0	12	192	0
2	0	1280	13	0	172
3	0	1556	14	32	72
4	762	0	15	96	0
5	0	288	16	96	0
6	288	0	17	48	0
7	509	0	18	48	0
8	192	0	19	0	36
9	0	256	20	0	32
10	0	192	21	24	0
11	192	0			
$Precision = 8035/8124 = 98.9\%$					
ROCK					
C#	Edible	Poisonous	C#	Edible	Poisonous
1	96	0	12	48	0
2	0	256	13	0	288
3	704	0	14	192	0
4	96	0	15	32	72
5	768	0	16	0	1728
6	0	192	17	288	0
7	1728	0	18	0	8
8	0	32	19	192	0
9	0	1296	20	16	0
10	0	8	21	0	36
11	48	0			
$Precision = 7804/8124 = 96.1\%$					

follows. We first apply Algorithm 3 over a training dataset to learn a similarity threshold δ and generate a collection of disjoint cluster cores with this threshold and $\theta = 1$. Then, we label each cluster core with a class. Next, for each new object o to be classified we count the number N_i of similar objects of o in each cluster core C_i^r. Finally, we compute $i = max_i\{N_i/|C_i^r|\}$ and label o with the class of C_i^r.

We also made experiments for classification on the Mushroom data. The 8,124 objects of the Mushroom dataset have already been divided into a training set (5416 objects) and test set (2,708 objects) on the SGI website (http://www.sgi.com/tech/mlc/db/). The experimental results are shown in Figure 5, where we obtain over 98% of classification accuracy at $\delta = 15$.

6 Conclusions and Future Work

We have presented a new clustering approach based on the notion of cluster cores, instead of nearest neighbors. A cluster core represents the core/center of a cluster and thus using cluster cores to derive clusters achieves high accuracy. Since clusters are not defined in terms of nearest neighbors, our method does not incur the curse of dimensionality and is scalable linearly with the dimensionality of data. Outliers are effectively eliminated by cluster cores, as an outlier would be similar to no or just a very few objects in a cluster core. Although our approach needs three thresholds, α (the minimum size of a cluster core) and δ (the similarity threshold) and θ (the cluster threshold), the last two can well be learned from data. We have shown that our approach outperforms both in efficiency and in accuracy the well-known ROCK algorithm.

Like all similarity matrix/graph-based approaches, most of the time of our approach is spent in building a similarity graph. Therefore, as part of future work we are seeking more efficient ways to handle similarity graphs of massive datasets. Moreover, we are going to make detailed comparisons with other closely related approaches such as SNN [6].

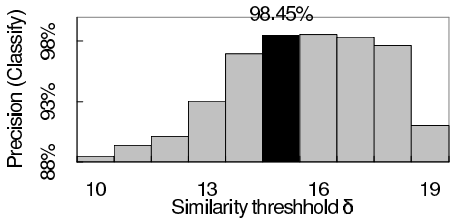

Figure 5. Accuracy of clustering-based classification (Mushroom data).

References

[1] J. Abello, P.M. Pardalos and M.G.C. Resende, On maximum cliques problems in very large graphs, in: (J. Abello and J. Vitter, Eds.) *DIMACS Series on Discrete Mathematics and Theoretical Computer Science*, American Mathematical Society (50): 119-130 (1999).

[2] C. C. Aggarwal and P. S. Yu, Finding generalized projected clusters in high dimensional spaces, in: *Proc. of ACM SIGMOD Intl. Conf. Management of Data*, pp. 70-81, 2000.

[3] R. Agrawal, J. Gehrke, D. Gunopolos and P. Raghavan, Automatic subspace clustering of high dimentional data for data mining applications, *ACM SIGMOD Intl. Conf. Management of Data*, pp. 94-105, 1998.

[4] M. R. Anderberg, *Cluster Analysis for Applications*, Academic Press, New York and London, 1973.

[5] M. Ester, H. P. Kriegel, J. Sander and X. Xu, A density-based algorithm for discovering clusters in large spatial databases with noise, in: *Proc. 2nd Intl. Conf. Knowledge Discovery and Data Mining*, pp. 226-231, 1996.

[6] L. Ertoz, M. Steinbach and V. Kumar, Finding clusters of different sizes, shapes, and densities in noisy, high dimensional data, in: *Proc. 3rd SIAM International Conference on Data Mining* San Francisco, CA, USA, 2003

[7] T.A. Feo and M.G.C. Resende, Greedy randomized adaptive search procedures, *J. of Global Optimization* (6): 109-133 (1995).

[8] K. Fukunaga, *Introduction to Statistical Pattern Recognition*, Academic Press, 1990.

[9] S. Guha, R. Rastogi and K. Shim, CURE: An efficient clustering algorithm for large databases, in: *Proc. ACM SIGMOD Intl. Conf. Management of Data*, pp. 73-84, 1998.

[10] S. Guha, R. Rastogi and K. Shim, ROCK: A robust clustering algorithm for categorical attributes, in: *Proc. ICDM Intl. Conf. on Data Engineering*, pp. 512-521, 1999.

[11] J. Han and M. Kamber, *Data Mining: Concepts and Techniques*, Morgan Kaufmann Publishers, 2000.

[12] A. Hinneburg, C. C. Aggarwal and D. A. Keim, What is the nearest neighbor in high dimentional spaces? *VLDB*, pp. 506-515, 2000.

[13] A. Hinneburg and D. A. Keim, Optimal grid-clustering: towards breaking the curse of dimentionality in high-dimentional clustering? *VLDB*, pp. 506-517, 1999.

[14] A. K. Jain and R. C. Dubes, *Algorithms for Clustering Data*, Prentice-Hall, 1988.

[15] A. K. Jain, M. N. Murty and P. J. Flynn, Data clustering: a review, *ACM Computing Surveys* 31(3): 264-323 (1999).

[16] R. A. Jarvis and E. A. Patrick, Clustering using a similarity measure based on shared nearest neighbors, *IEEE Transactions on Computers* C-22(11): 264-323 (1973).

[17] G. Karypis, E.-H. Han and V. Kumar, Chameleon: Hierarchical clustering using dynamic modeling, *IEEE Computer* 32(8): 68-75 (1999).

[18] R. T. Ng and J. Han, Efficient and effective clustering methods for spatial data mining, in: *Proc. 20th Intl. Conf. Very Large Data Bases*, pp. 144-155, 1994.

[19] C. M. Procopiuc, M. Jones, P. K. Agarwal and T. M. Murali, A monte carlo algorithm for fast projective clustering, *ACM SIGMOD Intl. Conf. Management of Data*, pp.418-427, 2002.

[20] W. Wang, J. Yang and R. Muntz, STING: a statistical information grid approach to spatial aata mining, in: *Proc. of the 23rd VLDB Conference*, pp. 186-195, 1997.

[21] I. H. Witten and E Frank, *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations*, Morgan Kaufmann Publishers, 1999.

[22] T. Zhang, R. Ramakrishnan and M. Livny, Birch: an efficient data clustering method for very large databases, in: *Proc. ACM SIGMOD Intl. Conf. Management of Data*, pp.103-114, 1996.