CURE

An Efficient Clustering Algorithm for Large Databases

presenter:Xiang Wan

The Introduction of Background

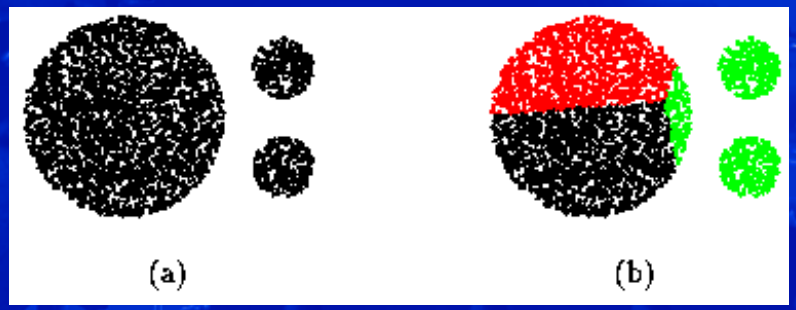
$$d_{mean}(C_i, C_j) = \|m_i - m_j\|$$

$$d_{ave}(C_i, C_j) = 1/(n_i n_j) \sum_{p \in C_i} \sum_{p' \in C_j} \|p - p'\|$$

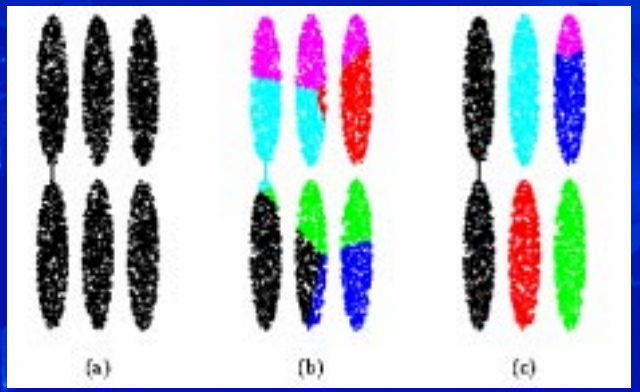
$$d_{max}(C_i, C_j) = \max_{p \in C_i, p' \in C_j} \|p - p'\|$$

$$d_{min}(C_i, C_j) = \min_{p \in C_i, p' \in C_j} \|p - p'\|$$

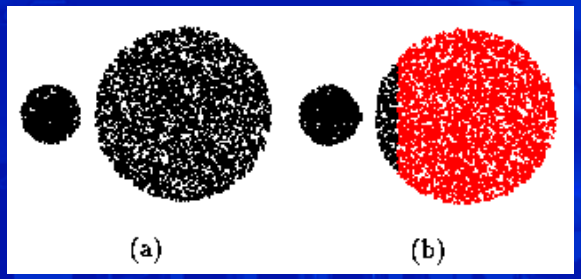
Example One



Example Two



Example Three



The Drawbacks of Traditional Clustering Algorithms

1.These algorithms can't identify the clusters having non-spherical shapes and wide variances in size.

2.These algorithms are sensitive to noises

3.These algorithms can't correctly label the data point when clusters don't have uniform sizes and shapes.

How to handle these problems in CURE

1.Choosing a constant number *C* of well scattered points in a cluster and using them to compute the distance between a pair of clusters.

2.Shrinking these points toward the centroid by a fraction *a* to mitigate the adverse effect of the noise.

Clustering Algorithms

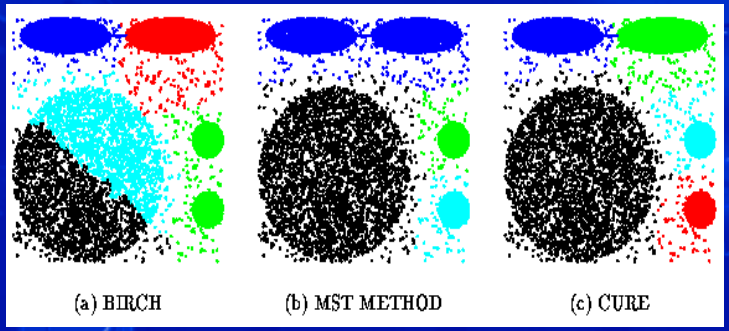
```
procedure cluster(S,k)
begin
1.  T := build_kd_tree(S)
2.  Q := build_heap(S)
3.  while size(Q) > 0 do {
4.      u := extract_min(Q)
5.      v := u.closest
6.      delete(Q, v)
7.      w := merge(u, v)
8.      delete_rep(T, u); delete_rep(T, v); insert_rep(T, w)
9.      w.closest := x /* x is an arbitrary cluster in Q */
10.     for each x ∈ Q do {
11.         if dist(w, x) < dist(w, w.closest)
12.             w.closest := x
13.         if x.closest is either u or v {
14.             if dist(x, x.closest) < dist(x, w)
15.                 x.closest := closest_cluster(T, x, dist(x, w))
16.             else
17.                 x.closest := w
18.             relocate(Q, x)
19.         }
20.         else if dist(x, x.closest) > dist(x, w) {
21.             x.closest := w
22.             relocate(Q, x)
23.         }
24.     }
25.     insert(Q, w)
26. }
end
```

Merging Procedure

```
procedure merge(u, v)
begin
1.  w := u ∪ v
2.  w.mean := (|u|·u.mean + |v|·v.mean) / (|u|+|v|)
3.  tmpSet := ∅
4.  for i := 1 to c do {
5.     maxDist := 0
6.     foreach point p in cluster w do {
7.        if i = 1
8.           minDist := dist(p, w.mean)
9.        else
10.          minDist := min{dist(p, q) : q ∈ tmpSet}
11.       if (minDist ≥ maxDist){
12.          maxDist := minDist
13.          maxPoint := p
14.       }
15.    }
16.    tmpSet := tmpSet ∪ {maxPoint}
17. }
18. foreach point p in tmpSet do
19.    w.rep := w.rep ∪ {p + α*(w.mean - p)}
20. return w
end
```

Xiang Wan

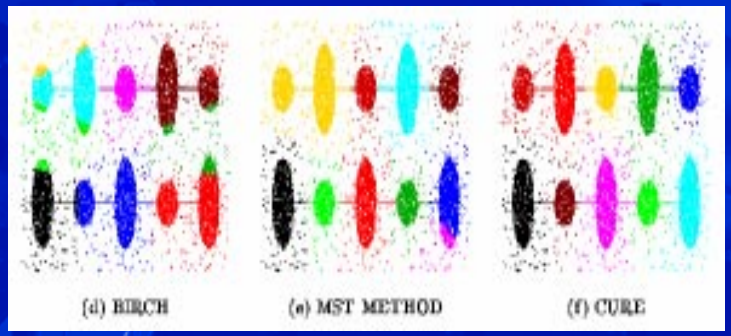
Example Four



(a) BIRCH (b) MST METHOD (c) CURE

Xiang Wan

Example Five



(d) BIRCH (e) MST METHOD (f) CURE

Xiang Wan