Robust Single Linkage Algorithm and Extract Flat Clustering

Taoran Xue

George Washington University

April 26, 2017

Introduction

- There are many sources of almost unlimited data:
  1. Images from the web.
  2. Speech recorded by a microphone.
  3. Records of credit card or other transactions.
- Noise points occasionally draw between two cluster

Minimum spanning tree

For each i, set $r(x_i)$ to the distance from x_i to its kth nearest neighbor.

As r grows from 0 to ∞:

1. Construct a graph G_r with nodes $\{x_i : r(x_i) \leq r\}$. Include edge (x_i, x_j) if $\|x_i - x_j\| \leq \alpha r$.
2. Let $\mathbb{C}_n(r)$ be the connected components of G_r.

Definition

Set $r_k(x_i)$ to the distance to kth nearest neighbor. For any $r = \max\{r_k(x_i)\}$, connect points x_i and x_j, if $||x_i - x_j|| \leq \alpha r$.

New distance function

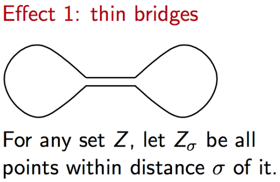


Figure 1: “Thin bridge” effect.

Definition

Set $core_k(x_i)$ to the distance to kth nearest neighbor.

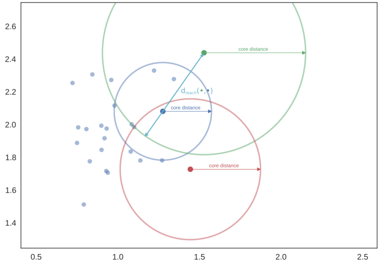
$$d_{\mathrm{mrd}_k}(x_i, x_j) = \max\{core_k(x_i), core_k(x_j), ||x_i - x_j||\}$$

Mutual reachability distance

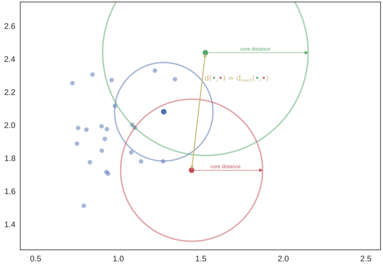
Definition

Set $core_k(x_i)$ to the distance to kth nearest neighbor.

$$d_{\mathrm{mrd}_k}(x_i, x_j) = \max\{core_k(x_i), core_k(x_j), ||x_i - x_j||\}$$



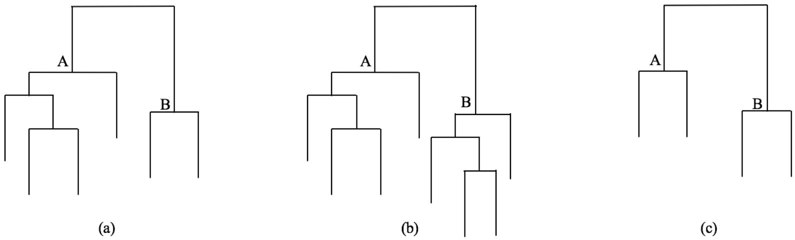
(a)



(b)

Condense tree

- If left child cluster point number is greater than minimum cluster size, but right side is not, Figure(a), keep the left branch and ignore right cluster;
- If left and right child clusters are both greater than minimum cluster size, Figure(b), we consider that a cluster split and let the split persist the whole tree;
- If left and right child clusters are both fewer than minimum cluster size, Figure(c), we ignore the two cluster.



Condense tree algorithm

```
Input: H[m] ← hierarchy tree
Output: T[n] ← condense tree
nodeList ← BFS(H);
for r ← 0 to m do
    if nodeList[r] is ignored then
        Pass;
    end
    left ← nodeList[r].child1;
    leftCount ← H[left].childrenSize;
    right ← nodeList[r].child2;
    rightCount ← H[right].childrenSize;
    if leftCount ≥ minClusterSize and rightCount ≥ minClusterSize then
        T[p++] ← (nextLabel, ++nextLabel, 1/distance, leftCount); T[p++] ← (nextLabel, ++nextLabel,
          1/distance, rightCount);
    end
    if leftCount < minClusterSize then
        for tmpNode in BFS(left) do
            if tmpNode is leaf then
                T[p++] ← (nextLabel, tmpNode, 1/distance, 1);
            end
            Ignore tmpNode;
        end
    end
    if rightCount < minClusterSize then
        for tmpNode in BFS(right) do
            if tmpNode is leaf then
                T[p++] ← (nextLabel, tmpNode, 1/distance, 1);
            end
            Ignore tmpNode;
        end
    end
end
```

Cluster stability

Definition

Let $\lambda = \frac{1}{d_{\mathrm{mrd}_k}}$. For each cluster we give λ_{birth} and λ_p to be the lambda value when the cluster split off then became it's own cluster, and the lambda value (if any) when the cluster split into smaller clusters respectively.

$$S(C) = \sum_{p \in C} (\lambda_p - \lambda_{\mathrm{birth}})$$

Cluster stability algorithm

Input: T[n] $\leftarrow$ condense tree in reverse topological order contains a tuple of (parent,child,λ,childrenSize)

Output: S[n] $\leftarrow$ stability of every node in condense tree

for $r \leftarrow 0$ **to** n **do**
    currChild $\leftarrow$ T[r].child;
    curr$\lambda \leftarrow$ T[r].λ;
    if currChild = prevChild **then**
        min$\lambda \leftarrow$ `Min`(minλ, currλ);
    else
        birthλ[currChild] $\leftarrow$ minλ;
        prevChild $\leftarrow$ currChild;
        min$\lambda \leftarrow$ currλ;
    end
end
for $r \leftarrow 0$ **to** n **do**
    $\mathsf{S}[r] \leftarrow \mathsf{S}[r] + \mathsf{T}[r].\lambda - \mathsf{birth}\lambda[\mathsf{T}[r].\mathrm{parent}] \times \mathsf{T}[r].\mathrm{childrenSize}$;
end

Flat clustering

Definition

Set $SC(C)$ is the sum of the stabilities of the child cluster of cluster C.

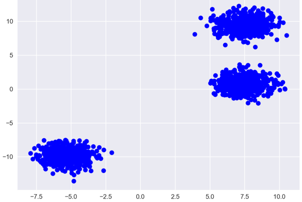
$$SC(C) = \sum_{q \in C} S(q)$$

Flat clustering algorithm

Input: $\mathrm{S}[n] \leftarrow$ stability condense tree sorted in reverse topological order
Output: $\mathrm{L}[n] \leftarrow$ **True** cluster is selected; **False** otherwise

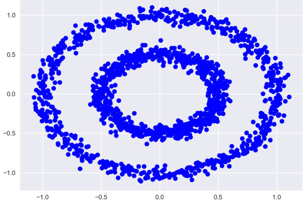
```
L ← {True};
for r ← 0 to n do
    childList ← {list of node whose parent is r};
    subtreeStabilities ← Σ_{c∈childList} S[c];
    if subtreeStabilities > S[r] then
        L[r] ← False;
        S[r] ← subtreeStabilities;
    else
        for tmpNode in BFS(r) do
            if tmpNode ≠ r then
                L[tmpNode] ← False
            end
        end
    end
end
```

Experiment

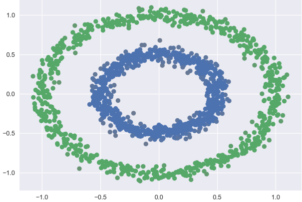


10
5
0
−5
−10
−7.5 −5.0 −2.5 0.0 2.5 5.0 7.5 10.0

10
5
0
−5
−10
−7.5 −5.0 −2.5 0.0 2.5 5.0 7.5 10.0

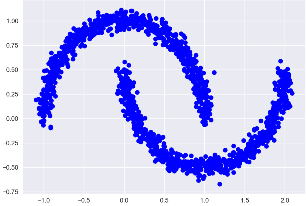


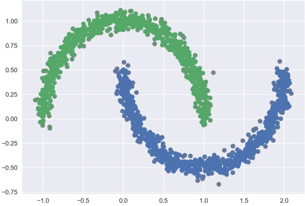
1.0
0.5
0.0
−0.5
−1.0
−1.0 −0.5 0.0 0.5 1.0

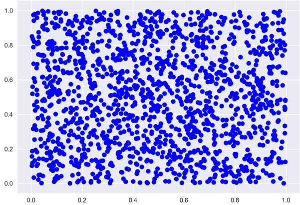


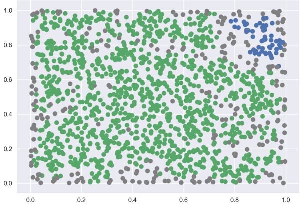
1.0
0.5
0.0
−0.5
−1.0
−1.0 −0.5 0.0 0.5 1.0

Experiment (cont.)

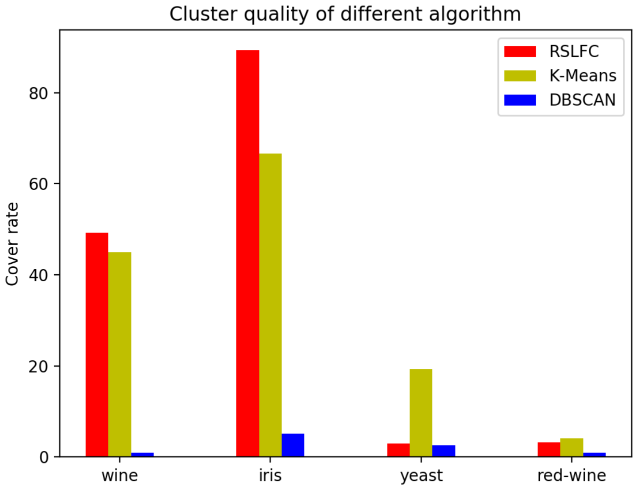








Experiment (cont.)



References

Campello, R. J., Moulavi, D., & Sander, J. (2013). Density-based clustering based on hierarchical density estimates. In *Pacific-asia conference on knowledge discovery and data mining* (pp. 160–172).

Chaudhuri, K., & Dasgupta, S. (2010). Rates of convergence for the cluster tree. In *Advances in neural information processing systems* (pp. 343–351).

McInnes, L., Healy, J., & Astels, S. (2017, mar). hdbscan: Hierarchical density based clustering. *The Journal of Open Source Software*, *2*(11).