Introduction to Machine Learning

Hierarchical Clustering and Large Data Set Clustering

Andres Mendez-Vazquez

August 4, 2020

Outline

Cinvestav

Outline

Cinvestav

Concepts

Hierarchical Clustering Algorithms

They are quite different from the previous clustering algorithms.

Actually

They produce a hierarchy of clusterings.

Concepts

Hierarchical Clustering Algorithms

They are quite different from the previous clustering algorithms.

Actually

They produce a hierarchy of clusterings.

Dendrogram

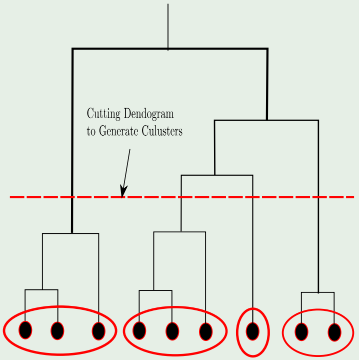
Hierarchical Clustering

The clustering is obtained by cutting the **dendrogram** at a desired level:

- Each connected component forms a cluster.

Example

Dendrogram



Outline

Cinvestav

Basic Ideas

At each step t

A new clustering is obtained based on the clustering produced at the previous step $t-1$

Basic Ideas

At each step t

A new clustering is obtained based on the clustering produced at the previous step $t-1$

Two Main Types

1. Agglomerative Algorithms.
    1. Start with each item being a single cluster.
    2. Eventually all items belong to the same cluster.
2. Divisive Algorithms
    1. Start with all items belong to the same cluster.
    2. Eventually each item forms a cluster on its own.

Cinvestav

Basic Ideas

At each step t

A new clustering is obtained based on the clustering produced at the previous step $t-1$

Two Main Types

1. Agglomerative Algorithms.
   1. Start with each item being a single cluster.
   2. Eventually all items belong to the same cluster.
2. Divisive Algorithms
   1. Start with all items belong to the same cluster.
   2. Eventually each item forms a cluster on its own.

Cinvestav

Basic Ideas

At each step t

A new clustering is obtained based on the clustering produced at the previous step $t-1$

Two Main Types

1. Agglomerative Algorithms.
   1. Start with each item being a single cluster.
   2. Eventually all items belong to the same cluster.
2. Divisive Algorithms
   1. Start with all items belong to the same cluster.
   2. Eventually each item forms a cluster on its own.

Basic Ideas

At each step t

A new clustering is obtained based on the clustering produced at the previous step $t-1$

Two Main Types

1. Agglomerative Algorithms.
   1. Start with each item being a single cluster.
   2. Eventually all items belong to the same cluster.
2. Divisive Algorithms
   1. Start with all items belong to the same cluster.
   2. Eventually each item forms a cluster on its own.

Cinvestav

Basic Ideas

At each step t

A new clustering is obtained based on the clustering produced at the previous step $t-1$

Two Main Types

1. Agglomerative Algorithms.
    1. Start with each item being a single cluster.
    2. Eventually all items belong to the same cluster.
2. Divisive Algorithms
    1. Start with all items belong to the same cluster.
    2. Eventually each item forms a cluster on its own.

Basic Ideas

At each step t

A new clustering is obtained based on the clustering produced at the previous step $t-1$

Two Main Types

1. Agglomerative Algorithms.
    1. Start with each item being a single cluster.
    2. Eventually all items belong to the same cluster.
2. Divisive Algorithms
    1. Start with all items belong to the same cluster.
    2. Eventually each item forms a cluster on its own.

Remark

With hierarchical methods, divisions or fusions, once made

- They are irrevocable
  - Agglomerative algorithm has joined two individuals they cannot subsequently be separated.
  - A divisive algorithm has made a split it cannot be undone.

Remark

With hierarchical methods, divisions or fusions, once made

- They are irrevocable
  - Agglomerative algorithm has joined two individuals they cannot subsequently be separated.
  - A divisive algorithm has made a split it cannot be undone.

As Kaufman and Rousseeuw (1990) colourfully comment (Similar to Forward Feature Selection)

- "A hierarchical method suffers from the defect that it can never repair what was done in previous steps."

Cinvestav

Remark

With hierarchical methods, divisions or fusions, once made

- They are irrevocable
  - Agglomerative algorithm has joined two individuals they cannot subsequently be separated.
  - A divisive algorithm has made a split it cannot be undone.

As Kaufman and Rousseeuw (1990) colourfully comment (Similar to Forward Feature Selection)

- "A hierarchical method suffers from the defect that it can never repair what was done in previous steps."

Cinvestav

Therefore

Given the previous ideas

It is necessary to define the concept of nesting!!!

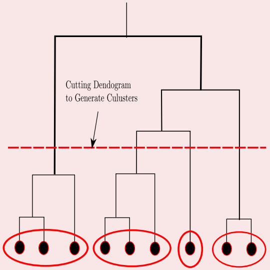
After all given a divisive and agglomerative procedure

Therefore

Given the previous ideas

It is necessary to define the concept of nesting!!!

After all given a divisive and agglomerative procedure



Nested Clustering

Definition

1. A clustering $\Re_i$ containing k clusters is said to be nested in the clustering $\Re_{i+1}$, which contains $r < k$ clusters, if each cluster in $\Re_i$, it is a subset of a set in $\Re_{i+1}$.
2. At least one cluster at $\Re_i$ is a proper subset of a set in $\Re_{i+1}$.

Nested Clustering

Definition

1. A clustering $\Re_i$ containing k clusters is said to be nested in the clustering $\Re_{i+1}$, which contains $r < k$ clusters, if each cluster in $\Re_i$, it is a subset of a set in $\Re_{i+1}$.
2. At least one cluster at $\Re_i$ is a proper subset of a set in $\Re_{i+1}$.

This is written as

$$\Re_i \sqsubset \Re_{i+1} \tag{1}$$

Cinvestav

Nested Clustering

Definition

1. A clustering $\Re_i$ containing k clusters is said to be nested in the clustering $\Re_{i+1}$, which contains $r < k$ clusters, if each cluster in $\Re_i$, it is a subset of a set in $\Re_{i+1}$.
2. At least one cluster at $\Re_i$ is a proper subset of a set in $\Re_{i+1}$.

This is written as

$$\Re_i \sqsubset \Re_{i+1} \tag{1}$$

Cinvestav

Example

We have

The following set$\{x_1, x_2, x_3, x_4, x_5\}$.

Example

We have

The following set$\{x_1, x_2, x_3, x_4, x_5\}$.

With the following structures

- $\Re_1 = \{\{x_1, x_3\}, \{x_4\}, \{x_2, x_5\}\}$
- $\Re_2 = \{\{x_1, x_3, x_4\}, \{x_2, x_5\}\}$

Example

We have

The following set$\{x_1, x_2, x_3, x_4, x_5\}$.

With the following structures

- $\Re_1 = \{\{x_1, x_3\}, \{x_4\}, \{x_2, x_5\}\}$
- $\Re_2 = \{\{x_1, x_3, x_4\}, \{x_2, x_5\}\}$

Again

Hierarchical Clustering produces a hierarchy of clusterings!!!

Cinvestav

Example

We have

The following set$\{x_1, x_2, x_3, x_4, x_5\}$.

With the following structures

- $\Re_1 = \{\{x_1, x_3\}, \{x_4\}, \{x_2, x_5\}\}$
- $\Re_2 = \{\{x_1, x_3, x_4\}, \{x_2, x_5\}\}$

Again

Hierarchical Clustering produces a hierarchy of clusterings!!!

Outline

Cinvestav

Agglomerative Algorithms.

Initial State

- You have N clusters each containing an element of the data X.
  - At each step i, you have an $\Re_i$ structure with $N-i$.
  - Then, a new clustering structure $\Re_{i+1}$ is generated.

Cinvestav

Agglomerative Algorithms.

Initial State

- You have N clusters each containing an element of the data X.
  - At each step i, you have an $\Re_i$ structure with $N - i$.
  - Then, a new clustering structure $\Re_{i+1}$ is generated.

Thus

Agglomerative Algorithms.

Initial State

- You have N clusters each containing an element of the data X.
  - At each step i, you have an $\Re_i$ structure with $N - i$.
  - Then, a new clustering structure $\Re_{i+1}$ is generated.

Thus

Agglomerative Algorithms.

Initial State

- You have N clusters each containing an element of the data X.
    - At each step i, you have an $\Re_i$ structure with $N-i$.
    - Then, a new clustering structure $\Re_{i+1}$ is generated.

Thus

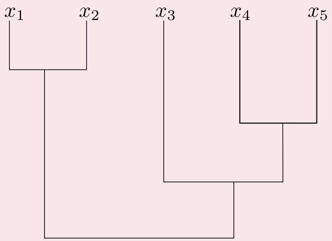
$\{\{x_1\},\{x_2\},\{x_3\},\{x_4\},\{x_5\}\}$

$\{\{x_1,x_2\},\{x_3\},\{x_4\},\{x_5\}\}$

$\{\{x_1,x_2\},\{x_3\},\{x_4,x_5\}\}$

$\{\{x_1,x_2\},\{x_3,x_4,x_5\}\}$

$\{\{x_1,x_2,x_3,x_4,x_5\}\}$



In that way...

We have

At each step, we have that each cluster $\Re_i$ is a proper subset of a cluste in $\Re_i$ or

$$\Re_i \sqsubset \Re_{i+1} \tag{2}$$

The Basic Algorithm for Agglomerative

For this

- We have a function $d(C_i, C_j)$ defined in all pair of cluster to measure similarity or dissimilarity.
- t denotes the current level of the hierarchy.

The Basic Algorithm for Agglomerative

For this

- We have a function $d(C_i, C_j)$ defined in all pair of cluster to measure similarity or dissimilarity.
- t denotes the current level of the hierarchy.

The Basic Algorithm

We have

1. Initialization
2. Choose $\Re_0 = \{Ci = \{x_i\} | i = 1, \dots, N\}$
3. $t = 0$
4. Repeat:
5. $t = t + 1$
6. Find one pair of clusters (C_r, C_s) in $\Re_{t-1}$ such that $d(C_i, C_j) = \max, \min$ of a similarity or dissimilarity function over all pairs
7. Define $C_q = C_i \cup C_j$, $\Re_t = \Re_{t-1} - \{C_i, C_j\} \cup C_q$
8. Until all vectors lay in a single cluster

The Basic Algorithm

We have

1. Initialization
2. Choose $\Re_0 = \{Ci = \{x_i\} \,|i = 1, ..., N\}$
3. $t = 0$
4. Repeat:
5. $t = t + 1$
6. Find one pair of clusters
   (C_r, C_s) in $\Re_{t-1}$ such that
   $d(C_i, C_j) = \max, \min$ of a similarity
   or dissimilarity function
   over all pairs
7. Define $C_q = C_i \cup C_j, \Re_t = \Re_{t-1} - \{C_i, C_j\} \cup C_q$
8. Until all vectors lay in a single cluster

The Basic Algorithm

We have

1. Initialization
2. Choose $\Re_0 = \{Ci = \{x_i\} \,|i = 1, ..., N\}$
3. $t = 0$
4. Repeat:
5. $t = t + 1$
6. Find one pair of clusters (C_r, C_s) in $\Re_{t-1}$ such that $d(C_i, C_j) = \max, \min$ of a similarity or dissimilarity function over all pairs
7. Define $C_q = C_i \cup C_j, \Re_t = \Re_{t-1} - \{C_i, C_j\} \cup C_q$
8. Until all vectors lay in a single cluster

The Basic Algorithm

We have

1. Initialization
2. Choose $\Re_0 = \{Ci = \{x_i\} \, | i = 1, ..., N\}$
3. $t = 0$
4. Repeat:
5. $t = t + 1$
6. Find one pair of clusters
   (C_r, C_s) in $\Re_{t-1}$ such that
   $d(C_i, C_j) = \max, \min$ of a similarity
   or dissimilarity function
   over all pairs
7. Define $C_q = C_i \cup C_j, \Re_t = \Re_{t-1} - \{C_i, C_j\} \cup C_q$
8. Until all vectors lay in a single cluster

The Basic Algorithm

We have

1. Initialization
2. Choose $\Re_0 = \{Ci = \{x_i\} \, | i = 1, ..., N\}$
3. $t = 0$
4. Repeat:
5. $t = t + 1$
6. Find one pair of clusters (C_r, C_s) in $\Re_{t-1}$ such that $d(C_i, C_j) = \max, \min$ of a similarity or dissimilarity function over all pairs
7. Define $C_q = C_i \cup C_j, \Re_t = \Re_{t-1} - \{C_i, C_j\} \cup C_q$
8. Until all vectors lay in a single cluster

The Basic Algorithm

We have

1. Initialization
2. Choose $\Re_0 = \{Ci = \{x_i\} \,|i = 1, ..., N\}$
3. $t = 0$
4. Repeat:
5. $t = t + 1$
6. Find one pair of clusters (C_r, C_s) in $\Re_{t-1}$ such that $d(C_i, C_j) = \max, \min$ of a similarity or dissimilarity function over all pairs
7. Define $C_q = C_i \cup C_j$, $\Re_t = \Re_{t-1} - \{C_i, C_j\} \cup C_q$
8. Until all vectors lay in a single cluster

The Basic Algorithm

We have

```
1 Initialization
2         Choose $\Re_0 = \{Ci = \{x_i\} \,|i = 1, ..., N\}$
3         $t = 0$
4 Repeat:
5         $t = t + 1$
6         Find one pair of clusters
                  $(C_r, C_s)$ in $\Re_{t-1}$ such that
                  $d(C_i, C_j) = \max, \min$ of a similarity
                  or dissimilarity function
                  over all pairs
7         Define $C_q = C_i \cup C_j$, $\Re_t = \Re_{t-1} - \{C_i, C_j\} \cup C_q$
8 Until all vectors lay in a single cluster
```

The Basic Algorithm

We have

```
1 Initialization
2     Choose $\Re_0 = \{Ci = \{x_i\} \,|i = 1, ..., N\}$
3     $t = 0$
4 Repeat:
5     $t = t + 1$
6     Find one pair of clusters
          $(C_r, C_s)$ in $\Re_{t-1}$ such that
          $d(C_i, C_j) = \max, \min$ of a similarity
          or dissimilarity function
          over all pairs
7     Define $C_q = C_i \cup C_j, \Re_t = \Re_{t-1} - \{C_i, C_j\} \cup C_q$
8 Until all vectors lay in a single cluster
```

The Basic Algorithm

We have

```
1  Initialization
2          Choose $\Re_0 = \{Ci = \{x_i\} \,|i = 1, ..., N\}$
3          $t = 0$
4  Repeat:
5          $t = t + 1$
6          Find one pair of clusters
                   $(C_r, C_s)$ in $\Re_{t-1}$ such that
                   $d(C_i, C_j) = \max, \min$ of a similarity
                   or dissimilarity function
                   over all pairs
7          Define $C_q = C_i \cup C_j, \Re_t = \Re_{t-1} - \{C_i, C_j\} \cup C_q$
8  Until all vectors lay in a single cluster
```

The Basic Algorithm

We have

```
1 Initialization
2         Choose $\Re_0 = \{Ci = \{x_i\} | i = 1, ..., N\}$
3         $t = 0$
4 Repeat:
5         $t = t + 1$
6         Find one pair of clusters
                  $(C_r, C_s)$ in $\Re_{t-1}$ such that
                  $d(C_i, C_j) = \max, \min$ of a similarity
                  or dissimilarity function
                  over all pairs
7         Define $C_q = C_i \cup C_j, \Re_t = \Re_{t-1} - \{C_i, C_j\} \cup C_q$
8 Until all vectors lay in a single cluster
```

Additionally

Note the following

“We can say that if two vectors come together into a single cluster at level t of the hierarchy, they will remain in the same cluster for all subsequent clusterings.”

Thus

$$\mathfrak{R}_0 \sqsubset \mathfrak{R}_1 \sqsubset \mathfrak{R}_2 \sqsubset \ldots \mathfrak{R}_{N-1} \sqsubset \mathfrak{R}_N \tag{3}$$

Which Enforces

- The nesting property!!!

Additionally

Note the following

“We can say that if two vectors come together into a single cluster at level t of the hierarchy, they will remain in the same cluster for all subsequent clusterings.”

Thus

$$\Re_0 \sqsubset \Re_1 \sqsubset \Re_2 \sqsubset ...\Re_{N-1} \sqsubset \Re_N \tag{3}$$

Which Enforces

- The nesting property!!!

Cinvestav

Additionally

Note the following

"We can say that if two vectors come together into a single cluster at level t of the hierarchy, they will remain in the same cluster for all subsequent clusterings."

Thus

$$\Re_0 \sqsubset \Re_1 \sqsubset \Re_2 \sqsubset ...\Re_{N-1} \sqsubset \Re_N \tag{3}$$

Which Enforces

- The nesting property!!!

Cinvestav

Outline

Cinvestav

Two Categories of Agglomerative Algorithms

There are two

1. Matrix Theory Based.
2. Graph Theory Based.

Two Categories of Agglomerative Algorithms

There are two

1. Matrix Theory Based.
2. Graph Theory Based.

Outline

Cinvestav

In Matrix Theory Based

Dissimilarity Matrix

As the name says, they are based in dissimilarity matrix $P_0 = P(X)$ of $N \times N$.

In Matrix Theory Based

Dissimilarity Matrix

As the name says, they are based in dissimilarity matrix $P_0 = P(X)$ of $N \times N$.

Merging Process

At each merging the matrix is reduced by one level $\Rightarrow P_t$ becomes a $N - t \times N - t$ matrix.

Matrix Based Algorithm

Matrix Updating Algorithmic Scheme (MUAS)

1. Initialization
2. Choose $\Re_0 = \{Ci = \{x_i\} | i = 1, ..., N\}$
3. $P_0 = P(X)$
4. $t = 0$
5. Repeat
6. $t = t + 1$
7. Find one pair of clusters

   (C_r, C_s) in $\Re_{t-1}$ such that

   $d(C_i, C_j) = \min_{r,s=1,...,N, r\neq s} d(C_r, C_s)$
8. Define $C_q = C_i \cup C_j, \Re_t = \Re_{t-1} - \{C_i, C_j\} \cup C_q$
9. Define P_t by **STRATEGY**
10. Until all vectors lay in a single cluster

Matrix Based Algorithm

Matrix Updating Algorithmic Scheme (MUAS)

```
1 Initialization
2     Choose $\Re_0 = \{Ci = \{x_i\} \,|i = 1, ..., N\}$
3     $P_0 = P(X)$
4     $t = 0$
5 Repeat
6     $t = t + 1$
7     Find one pair of clusters
              $(C_r, C_s)$ in $\Re_{t-1}$ such that
              $d(C_i, C_j) = \min_{r,s=1,...,N, r \neq s} d(C_r, C_s)$
8     Define $C_q = C_i \cup C_j, \Re_t = \Re_{t-1} - \{C_i, C_j\} \cup C_q$
9     Define $P_t$ by STRATEGY
10 Until all vectors lay in a single cluster
```

Matrix Based Algorithm

Matrix Updating Algorithmic Scheme (MUAS)

```
1  Initialization
2          Choose $\Re_0 = \{Ci = \{x_i\} \,|i = 1, ..., N\}$
3          $P_0 = P(X)$
4          $t = 0$
5  Repeat
6          $t = t + 1$
7          Find one pair of clusters
                   $(C_r, C_s)$ in $\Re_{t-1}$ such that
                   $d(C_i, C_j) = \min_{r,s=1,...,N, r\neq s} d(C_r, C_s)$
8          Define $C_q = C_i \cup C_j, \Re_t = \Re_{t-1} - \{C_i, C_j\} \cup C_q$
9          Define $P_t$ by **STRATEGY**
10 Until all vectors lay in a single cluster
```

Matrix Based Algorithm

Matrix Updating Algorithmic Scheme (MUAS)

```
1 Initialization
2     Choose $\Re_0 = \{Ci = \{x_i\} \,|i = 1, ..., N\}$
3     $P_0 = P(X)$
4     $t = 0$
5 Repeat
6     $t = t + 1$
7     Find one pair of clusters
          $(C_r, C_s)$ in $\Re_{t-1}$ such that
          $d(C_i, C_j) = \min_{r,s=1,...,N, r \neq s} d(C_r, C_s)$
8     Define $C_q = C_i \cup C_j, \Re_t = \Re_{t-1} - \{C_i, C_j\} \cup C_q$
9     Define $P_t$ by STRATEGY
10 Until all vectors lay in a single cluster
```

Matrix Based Algorithm

Matrix Updating Algorithmic Scheme (MUAS)

1. Initialization
2. Choose $\Re_0 = \{Ci = \{x_i\} \,|i = 1, ..., N\}$
3. $P_0 = P(X)$
4. $t = 0$
5. Repeat
6. $t = t + 1$
7. Find one pair of clusters

   (C_r, C_s) in $\Re_{t-1}$ such that

   $d(C_i, C_j) = \min_{r,s=1,...,N, r\neq s} d(C_r, C_s)$
8. Define $C_q = C_i \cup C_j, \Re_t = \Re_{t-1} - \{C_i, C_j\} \cup C_q$
9. Define P_t by **STRATEGY**
10. Until all vectors lay in a single cluster

Matrix Based Algorithm

Matrix Updating Algorithmic Scheme (MUAS)

```
1  Initialization
2      Choose $\Re_0 = \{Ci = \{x_i\} \,|i = 1, ..., N\}$
3      $P_0 = P(X)$
4      $t = 0$
5  Repeat
6      $t = t + 1$
7      Find one pair of clusters
               $(C_r, C_s)$ in $\Re_{t-1}$ such that
               $d(C_i, C_j) = \min_{r,s=1,...,N, r \neq s} d(C_r, C_s)$
8      Define $C_q = C_i \cup C_j, \Re_t = \Re_{t-1} - \{C_i, C_j\} \cup C_q$
9      Define $P_t$ by **STRATEGY**
10 Until all vectors lay in a single cluster
```

Matrix Based Algorithm

Matrix Updating Algorithmic Scheme (MUAS)

```
1  Initialization
2      Choose $\Re_0 = \{Ci = \{x_i\} | i = 1, ..., N\}$
3      $P_0 = P(X)$
4      $t = 0$
5  Repeat
6      $t = t + 1$
7      Find one pair of clusters
               $(C_r, C_s)$ in $\Re_{t-1}$ such that
               $d(C_i, C_j) = \min_{r,s=1,...,N, r \neq s} d(C_r, C_s)$
8      Define $C_q = C_i \cup C_j$, $\Re_t = \Re_{t-1} - \{C_i, C_j\} \cup C_q$
9      Define $P_t$ by STRATEGY
10 Until all vectors lay in a single cluster
```

Matrix Based Algorithm

Matrix Updating Algorithmic Scheme (MUAS)

```
1  Initialization
2      Choose $\Re_0 = \{Ci = \{x_i\} \mid i = 1, ..., N\}$
3      $P_0 = P(X)$
4      $t = 0$
5  Repeat
6      $t = t + 1$
7      Find one pair of clusters
               $(C_r, C_s)$ in $\Re_{t-1}$ such that
               $d(C_i, C_j) = \min_{r,s=1,..,N, r \neq s} d(C_r, C_s)$
8      Define $C_q = C_i \cup C_j, \Re_t = \Re_{t-1} - \{C_i, C_j\} \cup C_q$
9      Define $P_t$ by STRATEGY
10 Until all vectors lay in a single cluster
```

Matrix Based Algorithm

Matrix Updating Algorithmic Scheme (MUAS)

```
1  Initialization
2      Choose $\Re_0 = \{Ci = \{x_i\} \, | i = 1, ..., N\}$
3      $P_0 = P(X)$
4      $t = 0$
5  Repeat
6      $t = t + 1$
7      Find one pair of clusters
               $(C_r, C_s)$ in $\Re_{t-1}$ such that
               $d(C_i, C_j) = \min_{r,s=1,..,N, r \neq s} d(C_r, C_s)$
8      Define $C_q = C_i \cup C_j, \Re_t = \Re_{t-1} - \{C_i, C_j\} \cup C_q$
9      Define $P_t$ by STRATEGY
10 Until all vectors lay in a single cluster
```

Matrix Based Algorithm

Matrix Updating Algorithmic Scheme (MUAS)

```
1  Initialization
2      Choose $\Re_0 = \{Ci = \{x_i\} \,|i = 1, ..., N\}$
3      $P_0 = P(X)$
4      $t = 0$
5  Repeat
6      $t = t + 1$
7      Find one pair of clusters
               $(C_r, C_s)$ in $\Re_{t-1}$ such that
               $d(C_i, C_j) = \min_{r,s=1,..,N,r\neq s} d(C_r, C_s)$
8      Define $C_q = C_i \cup C_j, \Re_t = \Re_{t-1} - \{C_i, C_j\} \cup C_q$
9      Define $P_t$ by **STRATEGY**
10 Until all vectors lay in a single cluster
```

Matrix Based Algorithm

Matrix Updating Algorithmic Scheme (MUAS)

```
1  Initialization
2      Choose $\Re_0 = \{Ci = \{x_i\} \,|i = 1, ..., N\}$
3      $P_0 = P(X)$
4      $t = 0$
5  Repeat
6      $t = t + 1$
7      Find one pair of clusters
               $(C_r, C_s)$ in $\Re_{t-1}$ such that
               $d(C_i, C_j) = \min_{r,s=1,..,N,r\neq s} d(C_r, C_s)$
8      Define $C_q = C_i \cup C_j, \Re_t = \Re_{t-1} - \{C_i, C_j\} \cup C_q$
9      Define $P_t$ by **STRATEGY**
10 Until all vectors lay in a single cluster
```

Matrix Based Algorithm

STRATEGY

1. Delete the two rows and columns that correspond to the merged clusters.
2. Add new row and a new column that contain the distances between the newly formed cluster and the old (unaffected at this level) clusters.

Matrix Based Algorithm

STRATEGY

1. Delete the two rows and columns that correspond to the merged clusters.
2. Add new row and a new column that contain the distances between the newly formed cluster and the old (unaffected at this level) clusters.

Distance Used in These Schemes

It has been pointed out that there is only one general distance for these algorithms

$$\begin{aligned} d\left(C_q, C_s\right) =& a_i d\left(C_i, C_s\right) + a_j d\left(C_j, C_s\right) + ... \\ & bd\left(C_i, C_j\right) + c\left|d\left(C_i, C_s\right) - d\left(C_j, C_s\right)\right| \end{aligned}$$

Where different values of a_i, a_j, b and c correspond to different choices of the dissimilarity measures.

Distance Used in These Schemes

It has been pointed out that there is only one general distance for these algorithms

$$
\begin{aligned}
d\left(C_q, C_s\right) = & a_i d\left(C_i, C_s\right) + a_j d\left(C_j, C_s\right) + \ldots \\
& b d\left(C_i, C_j\right) + c\left|d\left(C_i, C_s\right) - d\left(C_j, C_s\right)\right|
\end{aligned}
$$

Where different values of a_i, a_j, b and c correspond to different choices of the dissimilarity measures.

Using this distance is possible to generate several algorithms

1. The single link algorithm.
2. The complete link algorithm.
3. The weighted pair group method average.
4. The unweighted pair group method centroid.
5. Etc...

Distance Used in These Schemes

It has been pointed out that there is only one general distance for these algorithms

$$\begin{aligned} d\left(C_q, C_s\right) =& a_i d\left(C_i, C_s\right) + a_j d\left(C_j, C_s\right) + ... \\ & bd\left(C_i, C_j\right) + c\left|d\left(C_i, C_s\right) - d\left(C_j, C_s\right)\right| \end{aligned}$$

Where different values of a_i, a_j, b and c correspond to different choices of the dissimilarity measures.

Using this distance is possible to generate several algorithms

1. The single link algorithm.
2. The complete link algorithm.
3. The weighted pair group method average.
4. The unweighted pair group method centroid.
5. Etc...

Distance Used in These Schemes

It has been pointed out that there is only one general distance for these algorithms

$$\begin{aligned} d\left(C_q, C_s\right) =& a_i d\left(C_i, C_s\right) + a_j d\left(C_j, C_s\right) + ... \\ & bd\left(C_i, C_j\right) + c\left|d\left(C_i, C_s\right) - d\left(C_j, C_s\right)\right| \end{aligned}$$

Where different values of a_i, a_j, b and c correspond to different choices of the dissimilarity measures.

Using this distance is possible to generate several algorithms

1. The single link algorithm.
2. The complete link algorithm.
3. The weighted pair group method average.
4. The unweighted pair group method centroid.
5. Etc...

Distance Used in These Schemes

It has been pointed out that there is only one general distance for these algorithms

$$\begin{aligned} d\left(C_q, C_s\right) = & a_i d\left(C_i, C_s\right) + a_j d\left(C_j, C_s\right) + \ldots \\ & b d\left(C_i, C_j\right) + c\left|d\left(C_i, C_s\right) - d\left(C_j, C_s\right)\right| \end{aligned}$$

Where different values of a_i, a_j, b and c correspond to different choices of the dissimilarity measures.

Using this distance is possible to generate several algorithms

1. The single link algorithm.
2. The complete link algorithm.
3. The weighted pair group method average.
4. The unweighted pair group method centroid.
5. Etc...

Distance Used in These Schemes

It has been pointed out that there is only one general distance for these algorithms

$$
\begin{aligned}
d\left(C_q, C_s\right) = & a_i d\left(C_i, C_s\right) + a_j d\left(C_j, C_s\right) + \ldots \\
& b d\left(C_i, C_j\right) + c\left|d\left(C_i, C_s\right) - d\left(C_j, C_s\right)\right|
\end{aligned}
$$

Where different values of a_i, a_j, b and c correspond to different choices of the dissimilarity measures.

Using this distance is possible to generate several algorithms

1. The single link algorithm.
2. The complete link algorithm.
3. The weighted pair group method average.
4. The unweighted pair group method centroid.
5. Etc...

Outline

Cinvestav

Single Linkage

Let G and H represent two such group sets

We have that

- Single linkage (SL) agglomerative clustering takes the intergroup dissimilarity to be that of the closest (**Least Dissimilar**) pair:

$$d_{SL}(G, H) = \min_{x_i \in G, x_j \in H} d(x_i, x_j)$$

This is also known as

- This is also often called the nearest-neighbor technique.

Single Linkage

Let G and H represent two such group sets

We have that

- Single linkage (SL) agglomerative clustering takes the intergroup dissimilarity to be that of the closest (**Least Dissimilar**) pair:

$$d_{SL}(G, H) = \min_{\boldsymbol{x}_i \in G, \boldsymbol{x}_j \in H} d(\boldsymbol{x}_i, \boldsymbol{x}_j)$$

This is also known as

- This is also often called the nearest-neighbor technique.

Single Linkage

Let G and H represent two such group sets

We have that

- Single linkage (SL) agglomerative clustering takes the intergroup dissimilarity to be that of the closest (**Least Dissimilar**) pair:

$$d_{SL}(G,H) = \min_{\boldsymbol{x}_i \in G, \boldsymbol{x}_j \in H} d(\boldsymbol{x}_i, \boldsymbol{x}_j)$$

This is also known as

- This is also often called the nearest-neighbor technique.

For example

The single linkage clustering algorithm

This is obtained if we set $a_i = 1/2$, $a_j = 1/2$, $b = 0$, $c = -1/2$

Thus, we have

$$d(C_q, C_s) = \min\{d(C_i, C_s), d(C_j, C_s)\} \quad (4)$$

For example

The single linkage clustering algorithm

This is obtained if we set $a_i = 1/2$, $a_j = 1/2$, $b = 0$, $c = -1/2$

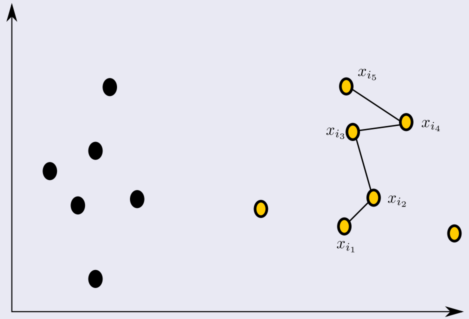
Thus, we have

$$d\left(C_q, C_s\right) = \min\left\{d\left(C_i, C_s\right), d\left(C_j, C_s\right)\right\} \tag{4}$$

What clusters are produced?

First

- Distance Between closest elements in clusters
- It produces long chains $x_{i_1} \rightarrow x_{i_2} \rightarrow x_{i_3} \rightarrow x_{i_4} \rightarrow x_{i_5}$



Another Example of a Single Link Dissimilarity

This can be created using the following cluster distance

$$d_{\min}\left(C_i, C_j\right) = \min_{\boldsymbol{x}\in C_i, \boldsymbol{y}\in C_j} \left\|\boldsymbol{x}-\boldsymbol{y}\right\|^2$$

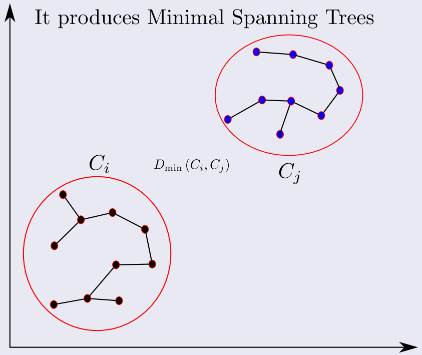
Nearest Neighborhood (Single Linkage)

Another Example of a Single Link Dissimilarity

This can be created using the following cluster distance

$$d_{\min}\left(C_i, C_j\right) = \min_{\boldsymbol{x}\in C_i, \boldsymbol{y}\in C_j} \left\|\boldsymbol{x}-\boldsymbol{y}\right\|^2$$

Nearest Neighborhood (Single Linkage)


It produces Minimal Spanning Trees
C_i
$D_{\min}\left(C_i, C_j\right)$
C_j

Outline

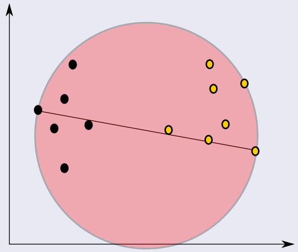
Cinvestav

Complete linkage (CL)

Complete Linkage agglomerative clustering (furthest-neighbor technique)

- It takes the intergroup dissimilarity to be that of the furthest (most dissimilar) pair

$$d_{CL}\left(G,H\right)=\max_{\boldsymbol{x}_i\in G,\boldsymbol{x}_j\in H} d\left(\boldsymbol{x}_i,\boldsymbol{x}_{\boldsymbol{j}}\right)$$



  - Distance between farthest elements in the clusters.
  - Forces, Spherical clusters with consistent diameter.

Example

This can be created using the following cluster distance

$$d_{\max}\left(C_i, C_j\right) = \max_{\boldsymbol{x} \in C_i, \boldsymbol{y} \in C_j} \left\|\boldsymbol{x} - \boldsymbol{y}\right\|^2$$

Outline

Cinvestav

Finally, a compromise

Group average (GA)

- Group average (GA) clustering uses the average dissimilarity between the groups

$$d_{CL}(G, H) = \frac{1}{N_G N_H} \sum_{\boldsymbol{x}_i \in G} \sum_{\boldsymbol{x}_j \in H} d(\boldsymbol{x}_i, \boldsymbol{x}_j)$$

Something Notable

- Average of all the pairwise distances
- Less affected by outliers

Finally, a compromise

Group average (GA)

- Group average (GA) clustering uses the average dissimilarity between the groups

$$d_{CL}(G, H) = \frac{1}{N_G N_H} \sum_{\boldsymbol{x}_i \in G} \sum_{\boldsymbol{x}_j \in H} d(\boldsymbol{x}_i, \boldsymbol{x}_j)$$

Something Notable

- Average of all the pairwise distances
- Less affected by outliers

Cinvestav

Outline

Cinvestav

Agglomerative Algorithms Based on Graph Theory

Consider the following

1. Each node in the graph G correspond to a vector.
2. Cluster are formed by connecting nodes.
3. Certain property, $h(k)$, needs to be respected.

Agglomerative Algorithms Based on Graph Theory

Consider the following

1. Each node in the graph G correspond to a vector.
2. Cluster are formed by connecting nodes.
3. Certain property, $h(k)$, needs to be respected.

Common Properties: Node Connectivity

- The **node connectivity** of a connected subgraph is the largest integer k
  - All pairs of nodes are joined by at least k paths having no nodes in common.

Agglomerative Algorithms Based on Graph Theory

Consider the following

1. Each node in the graph G correspond to a vector.
2. Cluster are formed by connecting nodes.
3. Certain property, $h(k)$, needs to be respected.

Common Properties: Node Connectivity

- The **node connectivity** of a connected subgraph is the largest integer k
  - All pairs of nodes are joined by at least k paths having no nodes in common.

Agglomerative Algorithms Based on Graph Theory

Consider the following

1. Each node in the graph G correspond to a vector.
2. Cluster are formed by connecting nodes.
3. Certain property, $h(k)$, needs to be respected.

Common Properties: Node Connectivity

- The **node connectivity** of a connected subgraph is the largest integer k
  - All pairs of nodes are joined by at least k paths having no nodes in common.

Cinvestav

Agglomerative Algorithms Based on Graph Theory

Common Properties: Edge Connectivity

The **edge connectivity** of a connected subgraph is the largest integer k such that all pairs of nodes are joined by at least k paths having no edges in common.

Common Properties: Node Degree

The **degree** of a connected subgraph is the largest integer k such that each node has at least k incident edges.

Cinvestav

Agglomerative Algorithms Based on Graph Theory

Common Properties: Edge Connectivity

The **edge connectivity** of a connected subgraph is the largest integer k such that all pairs of nodes are joined by at least k paths having no edges in common.

Common Properties: Node Degree

The **degree** of a connected subgraph is the largest integer k such that each node has at least k incident edges.

Cinvestav

Basically, We use the Same Scheme, But...

The function

$$d_{h(k)}\left(C_r, C_s\right) = \min_{x \in C_r, y \in C_s} \left\{ d\left(x, y\right) \middle| Property \right\} \tag{5}$$

Basically, We use the Same Scheme, But...

The function

$$d_{h(k)}\left(C_r, C_s\right) = \min_{x \in C_r, y \in C_s} \left\{d\left(x, y\right) | Property\right\} \tag{5}$$

Property

The G subgraph defined by $C_r \cup C_s$ is

1. It is connected and either
    1. It has the property $h(k)$ or
    2. It is complete

Cinvestav

Basically, We use the Same Scheme, But...

The function

$$d_{h(k)}\left(C_r, C_s\right) = \min_{x \in C_r, y \in C_s} \left\{ d\left(x, y\right) \middle| Property \right\} \quad (5)$$

Property

The G subgraph defined by $C_r \cup C_s$ is

1. It is connected and either
   1. It has the property $h(k)$ or
   2. It is complete

Basically, We use the Same Scheme, But...

The function

$$d_{h(k)}\left(C_r, C_s\right) = \min_{x \in C_r, y \in C_s} \left\{ d\left(x, y\right) | Property \right\} \quad (5)$$

Property

The G subgraph defined by $C_r \cup C_s$ is

1. It is connected and either
   1. It has the property $h(k)$ or
   2. It is complete

Cinvestav

Basically, We use the Same Scheme, But...

The function

$$d_{h(k)}\left(C_r, C_s\right) = \min_{x \in C_r, y \in C_s} \left\{ d\left(x, y\right) \middle| Property \right\} \tag{5}$$

Property

The G subgraph defined by $C_r \cup C_s$ is

1. It is connected and either
    1. It has the property $h(k)$ or
    2. It is complete

Examples

Again

1. Single Link Algorithm
2. Complete Link Algorithm

Cinvestav

Examples

Again

1. Single Link Algorithm
2. Complete Link Algorithm

There is other style of clustering

- Clustering Algorithms Based on the Minimum Spanning Tree

Examples

Again

1. Single Link Algorithm
2. Complete Link Algorithm

There is other style of clustering

- Clustering Algorithms Based on the Minimum Spanning Tree

Outline

Cinvestav

Problems with Schema of Agglomerative Algorithms

First - Related to Nesting Property

No way to recover from a "poor" clustering that may have occurred in an earlier level of the hierarchy.

Problems with Schema of Agglomerative Algorithms

First - Related to Nesting Property

No way to recover from a “poor” clustering that may have occurred in an earlier level of the hierarchy.

Second, if we use a Naive Scheme - Full comparison at each level

- At each level t, there are $N - t$ clusters.
- Thus at level $t+1$ the total number of pairs compared :

$$\binom{N-t}{2} = \frac{(N-t)(N-t-1)}{2} \tag{6}$$

Problems with Schema of Agglomerative Algorithms

First - Related to Nesting Property

No way to recover from a “poor” clustering that may have occurred in an earlier level of the hierarchy.

Second, if we use a Naive Scheme - Full comparison at each level

- At each level t, there are $N - t$ clusters.
- Thus at level $t + 1$ the total number of pairs compared .

$$\binom{N-t}{2} = \frac{(N-t)(N-t-1)}{2} \tag{6}$$

Total Number of pairs compared are

$$\sum_{t=0}^{N-1} \binom{N-t}{2} \tag{7}$$

Problems with Schema of Agglomerative Algorithms

First - Related to Nesting Property

No way to recover from a “poor” clustering that may have occurred in an earlier level of the hierarchy.

Second, if we use a Naive Scheme - Full comparison at each level

- At each level t, there are $N-t$ clusters.
- Thus at level $t+1$ the total number of pairs compared .

$$\begin{pmatrix} N-t \\ 2 \end{pmatrix} = \frac{(N-t)(N-t-1)}{2} \quad (6)$$

Total Number of pairs compared are

$$\sum_{t=0}^{N-1} \begin{pmatrix} N-t \\ 2 \end{pmatrix} \quad (7)$$

Problems with Schema of Agglomerative Algorithms

First - Related to Nesting Property

No way to recover from a “poor” clustering that may have occurred in an earlier level of the hierarchy.

Second, if we use a Naive Scheme - **Full comparison at each level**

- At each level t, there are $N-t$ clusters.
- Thus at level $t+1$ the total number of pairs compared .

$$\binom{N-t}{2} = \frac{(N-t)(N-t-1)}{2} \tag{6}$$

Total Number of pairs compared are

$$\sum_{t=0}^{N-1} \binom{N-t}{2} \tag{7}$$

Thus

We have that

$$\sum_{t=0}^{N-1} \binom{N-t}{2} = \sum_{k=1}^{N} \binom{k}{2} = \frac{(N-1)\,N\,(N+1)}{6} \tag{8}$$

Thus

The complexity of this schema is $O\left(N^3\right)$

However

You still depend on the nature of d.

Thus

We have that

$$\sum_{t=0}^{N-1} \binom{N-t}{2} = \sum_{k=1}^{N} \binom{k}{2} = \frac{(N-1)\,N\,(N+1)}{6} \tag{8}$$

Thus

The complexity of this schema is $O\left(N^3\right)$

However

You still depend on the nature of d.

Thus

We have that

$$\sum_{t=0}^{N-1} \binom{N-t}{2} = \sum_{k=1}^{N} \binom{k}{2} = \frac{(N-1)\,N\,(N+1)}{6} \tag{8}$$

Thus

The complexity of this schema is $O\left(N^3\right)$

However

You still depend on the nature of d.

Then

We need to be able to improve the complexity of Aggregation

- From the Metric Algorithms and Data structures, there are possible solutions...

Outline

Cinvestav

The idea of using a middle point

In order to establish a better performance

- Every time, we join two clusters:
    - We can then use a representative for such join in the agglomeration

Therefore, we need a data structure to be able to support these updates

- We may use a Kd-tree...

The idea of using a middle point

In order to establish a better performance

- Every time, we join two clusters:
  - We can then use a representative for such join in the agglomeration

Therefore, we need a data structure to be able to support these updates

- We may use a Kd-tree...

Cinvestav

In this case, we assume a group average

We need a Kd-tree supporting insertions

- By Logarithmic Rebuilding...

This was born from the fact that

- It is necessary to modify the Kd-tree dynamically to maintain certain performance.

In this case, we assume a group average

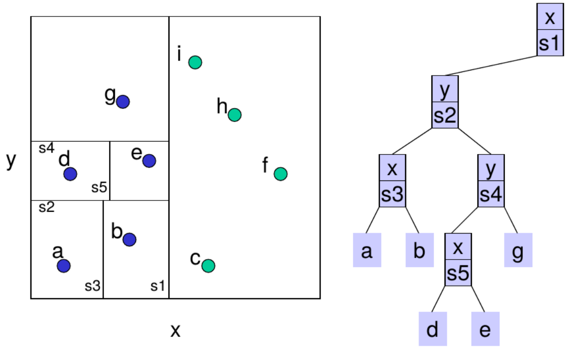
We need a Kd-tree supporting insertions

- By Logarithmic Rebuilding...

This was born from the fact that

- It is necessary to modify the Kd-tree dynamically to maintain certain performance.

Example



In order to keep performance, Logarithmic Rebuilding

We maintain at most $h = O(\log N)$ Kd-trees

- $T_0, T_1, ..., T_{h-1}$ such that the i^{th} $(i \in [1, h])$ tree stores precisely 2^i points.
- **Each point is stored in only one Kd-tree.**

Procedure

We have the following procedure

- To insert a new point $\boldsymbol{p}$, we
  1. Identify the smallest $i \geq 0$ such that T_i is empty
  2. Destroy all of $T_0, T_1, ..., T_{i-1}$. Collect all the points there into a set S.
  3. Construct T_i in $S \cup \{p\}$
  - Note $|T_i| = 2^i$

Amortized Analysis

Construction of T_i

- It takes $O\left(2^i \log 2^i\right)$ time

Charge the cost on the 2^i points in T_i

- Each of which is amortized $O\left(\log 2^i\right) = O\left(\log N\right)$ time.

Each point can be charged only $O\left(\log n\right)$ when moving to a bigger tree

- Amortized insertion time per point $O\left(\log^2 N\right)$

Cinvestav

Amortized Analysis

Construction of T_i

- It takes $O\left(2^i \log 2^i\right)$ time

Charge the cost on the 2^i points in T_i

- Each of which is amortized $O\left(\log 2^i\right) = O\left(\log N\right)$ time.

Each point can be charged only $O\left(\log n\right)$ when moving to a bigger tree

- Amortized insertion time per point $O\left(\log^2 N\right)$

Cinvestav

Amortized Analysis

Construction of T_i

- It takes $O\left(2^i \log 2^i\right)$ time

Charge the cost on the 2^i points in T_i

- Each of which is amortized $O\left(\log 2^i\right) = O\left(\log N\right)$ time.

Each point can be charged only $O\left(\log n\right)$ when moving to a bigger tree

- Amortized insertion time per point $O\left(\log^2 N\right)$

Cinvestav

Querying the Structure

Simply

- Search all of the h trees $T_0, T_1, ..., T_{h-1}$

Query Time

$$O\left(\sqrt{2^{h-1}}+\sqrt{2^{h-2}}+\cdots+\sqrt{2^{0}}+k\right)=O\left(\sqrt{N}+k\right)$$

- Similar to the search on the original Kd-tree.

Querying the Structure

Simply

- Search all of the h trees $T_0, T_1, ..., T_{h-1}$

Query Time

$$O\left(\sqrt{2^{h-1}} + \sqrt{2^{h-2}} + \cdots + \sqrt{2^0} + k\right) = O\left(\sqrt{N} + k\right)$$

- Similar to the search on the original Kd-tree.

Cinvestav

What if we avoid comparing all the elements using a Kd-Tree

Generation of the Structure

- $O\left(N \log^2 N\right)$ to get the data structure with space $O(N)$

Query

- We get to query in $O\left(\sqrt{N}+k\right)$ in the worst case scenario.
  - Here k is the number of elements being reported.

Cinvestav

What if we avoid comparing all the elements using a Kd-Tree

Generation of the Structure

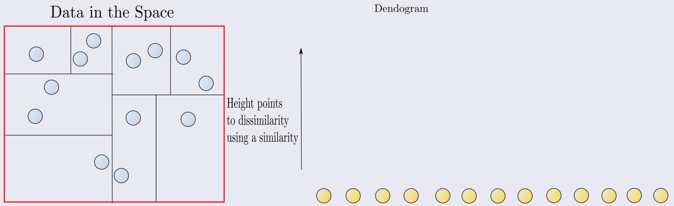
- $O\left(N \log^2 N\right)$ to get the data structure with space $O(N)$

Query

- We get to query in $O\left(\sqrt{N} + k\right)$ in the worst case scenario.
  - Here k is the number of elements being reported.

Therefore

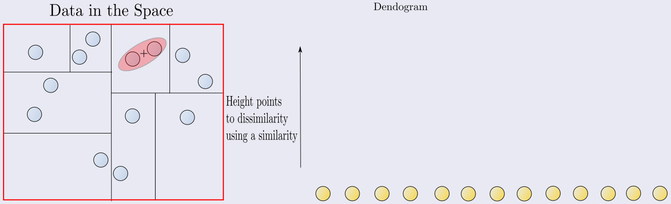
We have



At Each Level

At each level calculate the new centroid

- Insert it
  - Inserting takes $O\left(\log^2 N\right)$, but How many insertions?



We have...

Therefore

- We have total number of insertions assuming pair of them:

$$\frac{N}{2}+\frac{N}{2^2}+....+\frac{N}{2^{\log n}}=N\left(\frac{1-\frac{1}{N}}{\frac{1}{2}}\right)-N=*$$

Therefore

$$*=2\left(N-1\right)-N=N-2$$

Cinvestav

We have...

Therefore

- We have total number of insertions assuming pair of them:

$$\frac{N}{2}+\frac{N}{2^2}+\ldots.+\frac{N}{2^{\log n}}=N\left(\frac{1-\frac{1}{N}}{\frac{1}{2}}\right)-N=*$$

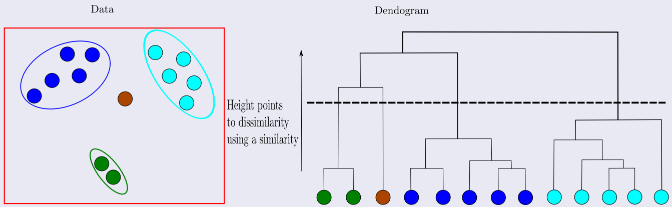
Therefore

$$*=2\left(N-1\right)-N=N-2$$

Cinvestav

Final Complexity

We have after building the data structure



Then, we have

The Clustering takes

$$O\left(N\log^2 N\right) + O\left(N^{3/2} + Nk\right) = O\left(N^{3/2}\right)$$

Given that you need to build a tree for each centroid structure

$$N\log^2 N + \frac{1}{2}N\log^2\frac{N}{2} + \ldots + \frac{1}{2^{\log N}}\log^2\frac{N}{2^{\log^2}} = O\left(N\log^2 N\right)$$

Then, we have

The Clustering takes

$$O\left(N\log^2 N\right) + O\left(N^{3/2} + Nk\right) = O\left(N^{3/2}\right)$$

Given that you need to build a tree for each centroid structure

$$N\log^2 N + \frac{1}{2}N\log^2\frac{N}{2} + ... + \frac{1}{2^{\log N}}\log^2\frac{N}{2^{\log^2}} = O\left(N\log^2 N\right)$$

Cinvestav

Outline

Cinvestav

Divisive Algorithms

Reverse Strategy

- **Start with a single cluster split it iteratively.**

They are lees common than agglomerative methods

- However, Kaufman and Rousseeuw (1990) pointed out:
  - This is revealed when a divisive method is applied

Cinvestav

Divisive Algorithms

Reverse Strategy

- **Start with a single cluster split it iteratively.**

They are lees common than agglomerative methods

- However, Kaufman and Rousseeuw (1990) pointed out:
  - This is revealed when a divisive method is applied

Generalized Divisive Scheme

Algorithm PROBLEM what is wrong!!!

```
1  Initialization
2          Choose $\Re_0 = \{X\}$
3          $P_0 = P(X)$
4          $t = 0$
5  Repeat
6          $t = t + 1$
7          For $i = 1$ to $t$
8                  Given a partition $C_{t-1,i}$
9                  Generate all possible clusters
10         next $i$
11         Find the pair $C^1_{t-1,j}, C^2_{t-1,j}$ that
                 maximize $g$
12         Create
                 $\Re_t = \Re_{t-1} - \{C_{t-1,j}\} \cup \left\{C^1_{t-1,j}, C^2_{t-1,j}\right\}$
13 Until all vectors lie in a single cluster
```

Generalized Divisive Scheme

Algorithm PROBLEM what is wrong!!!

```
1  Initialization
2          Choose $\Re_0 = \{X\}$
3          $P_0 = P(X)$
4          $t = 0$
5  Repeat
6          $t = t + 1$
7          For $i = 1$ to $t$
8                  Given a partition $C_{t-1}, i$
9                  Generate all possible clusters
10         next $i$
11         Find the pair $C^1_{t-1,j}, C^2_{t-1,j}$ that
                 maximize $g$
12         Create
                 $\Re_t = \Re_{t-1} - \{C_{t-1,j}\} \cup \left\{C^1_{t-1,j}, C^2_{t-1,j}\right\}$
13 Until all vectors lie in a single cluster
```

Generalized Divisive Scheme

Algorithm PROBLEM what is wrong!!!

```
1  Initialization
2          Choose $\Re_0 = \{X\}$
3          $P_0 = P(X)$
4          $t = 0$
5  Repeat
6          $t = t + 1$
7          For $i = 1$ to $t$
8                  Given a partition $C_{t-1}, i$
9                  Generate all possible clusters
10         next $i$
11         Find the pair $C^1_{t-1,j}, C^2_{t-1,j}$ that
                maximize $g$
12         Create
                $\Re_t = \Re_{t-1} - \{C_{t-1,j}\} \cup \left\{C^1_{t-1,j}, C^2_{t-1,j}\right\}$
13 Until all vectors lie in a single cluster
```

Generalized Divisive Scheme

Algorithm PROBLEM what is wrong!!!

```
1  Initialization
2      Choose $\Re_0 = \{X\}$
3      $P_0 = P(X)$
4      $t = 0$
5  Repeat
6      $t = t + 1$
7      For $i = 1$ to $t$
8          Given a partition $C_{t-1}, i$
9          Generate all possible clusters
10     next $i$
11     Find the pair $C^1_{t-1,j}, C^2_{t-1,j}$ that
           maximize $g$
12     Create
           $\Re_t = \Re_{t-1} - \{C_{t-1,j}\} \cup \left\{C^1_{t-1,j}, C^2_{t-1,j}\right\}$
13 Until all vectors lie in a single cluster
```

Generalized Divisive Scheme

Algorithm PROBLEM what is wrong!!!

```
1  Initialization
2      Choose $\Re_0 = \{X\}$
3      $P_0 = P(X)$
4      $t = 0$
5  Repeat
6      $t = t + 1$
7      For $i = 1$ to $t$
8          Given a partition $C_{t-1}, i$
9          Generate all possible clusters
10     next $i$
11     Find the pair $C^1_{t-1,j}, C^2_{t-1,j}$ that
           maximize $g$
12     Create
           $\Re_t = \Re_{t-1} - \{C_{t-1,j}\} \cup \left\{C^1_{t-1,j}, C^2_{t-1,j}\right\}$
13 Until all vectors lie in a single cluster
```

Generalized Divisive Scheme

Again, we need to be smart

```
1  Initialization
2        Choose $\Re_0 = \{X\}$
3        $P_0 = P(X)$
4        $t = 0$
5  Repeat
6        $t = t + 1$
7        For $i = 1$ to $t$
8              Given a partition $C_{t-1,i}$
9              Generate all possible clusters
10       next $i$
11       Find the pair $C^1_{t-1,j}, C^2_{t-1,j}$ that
              maximize $g$
12       Create
              $\Re_t = \Re_{t-1} - \{C_{t-1,j}\} \cup \left\{C^1_{t-1,j}, C^2_{t-1,j}\right\}$
13 Until all vectors lie in a single cluster
```

Generalized Divisive Scheme

Again, we need to be smart

```
1  Initialization
2        Choose $\Re_0 = \{X\}$
3        $P_0 = P(X)$
4        $t = 0$
5  Repeat
6        $t = t + 1$
7        For $i = 1$ to $t$
8              **Given a partition $C_{t-1}, i$**
9              **Generate all possible clusters**
10       next $i$
11       Find the pair $C^1_{t-1,j}, C^2_{t-1,j}$ that
              maximize $g$
12       Create
              $\Re_t = \Re_{t-1} - \{C_{t-1,j}\} \cup \left\{C^1_{t-1,j}, C^2_{t-1,j}\right\}$
13 Until all vectors lie in a single cluster
```

Generalized Divisive Scheme

Again, we need to be smart

```
1  Initialization
2      Choose $\Re_0 = \{X\}$
3      $P_0 = P(X)$
4      $t = 0$
5  Repeat
6      $t = t + 1$
7      For $i = 1$ to $t$
8          **Given a partition $C_{t-1,i}$**
9          **Generate all possible clusters**
10     next $i$
11     Find the pair $C^1_{t-1,j}, C^2_{t-1,j}$ that
           maximize $g$
12     Create
           $\Re_t = \Re_{t-1} - \{C_{t-1,j}\} \cup \left\{C^1_{t-1,j}, C^2_{t-1,j}\right\}$
13 Until all vectors lie in a single cluster
```

Generalized Divisive Scheme

Again, we need to be smart

```
1  Initialization
2      Choose $\Re_0 = \{X\}$
3      $P_0 = P(X)$
4      $t = 0$
5  Repeat
6      $t = t + 1$
7      For $i = 1$ to $t$
8          **Given a partition $C_{t-1,i}$**
9          **Generate all possible clusters**
10     next $i$
11     Find the pair $C^1_{t-1,j}, C^2_{t-1,j}$ that
           maximize $g$
12     Create
           $\Re_t = \Re_{t-1} - \{C_{t-1,j}\} \cup \left\{C^1_{t-1,j}, C^2_{t-1,j}\right\}$
13 Until all vectors lie in a single cluster
```

Generalized Divisive Scheme

Again, we need to be smart

```
1  Initialization
2      Choose $\Re_0 = \{X\}$
3      $P_0 = P(X)$
4      $t = 0$
5  Repeat
6      $t = t + 1$
7      For $i = 1$ to $t$
8          **Given a partition $C_{t-1}, i$**
9          **Generate all possible clusters**
10     next $i$
11     Find the pair $C^1_{t-1,j}, C^2_{t-1,j}$ that
           maximize $g$
12     Create
           $\Re_t = \Re_{t-1} - \{C_{t-1,j}\} \cup \left\{C^1_{t-1,j}, C^2_{t-1,j}\right\}$
13 Until all vectors lie in a single cluster
```

Outline

Cinvestav

Possible Complexity

This is computationally demanding

- If all $2^{N_t^i} - 1$ possible division are considered:
  - With N_t^i is the number of elements in the cluster.

However, for data consisting of d binary variables

- Relatively simple and computationally efficient methods exists
  - **Monothetic divisive methods**

Cinvestav

Possible Complexity

This is computationally demanding

- If all $2^{N_t^i} - 1$ possible division are considered:
  - With N_t^i is the number of elements in the cluster.

However, for data consisting of d binary variables

- Relatively simple and computationally efficient methods exists
  - **Monothetic divisive methods**

Cinvestav

Outline

Cinvestav

Monothetic Divisive Methods

They are based on

- These generally divide clusters according to the presence or absence of each of the d variables.
  - At each stage cluster members contain or not certain attributes.

Format of the data

- The data is in the form of a two-mode (binary) matrix.

$$\text{members}\left\{\overbrace{\begin{pmatrix}1&0&1&\cdots&0\\0&1&0&\cdots&0\\0&0&0&\cdots&1\\\vdots&\vdots&\vdots&\ddots&\vdots\\0&0&0&\cdots&1\end{pmatrix}}^{\text{Attributes}}\right.$$

Monothetic Divisive Methods

They are based on

- These generally divide clusters according to the presence or absence of each of the d variables.
  - At each stage cluster members contain or not certain attributes.

Format of the data

- The data is in the form of a two-mode (binary) matrix.

$$\text{members}\left\{\overbrace{\begin{pmatrix} 1 & 0 & 1 & \cdots & 0 \\ 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 0 & \cdots & 1 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 1 \end{pmatrix}}^{\text{Attributes}}\right.$$

Then, if we define

f_k= It is the number of individuals having k^{th} attribute

- We can define the following homogeneity criterion (Information Content):

$$C = dN\log N - \sum_{k=1}^{d}\left\{f_k \log f_k - (n - f_k)\log(n - f_k)\right\}$$

Therefore, if we split the original cluster into two groups A and B

- The reduction in C is $C_X - C_A - C_B$

Therefore

- The ideal set of clusters would have members with identical attributes and C equal to zero.

Then, if we define

f_k= It is the number of individuals having k^{th} attribute

- We can define the following homogeneity criterion (Information Content):

$$C = dN \log N - \sum_{k=1}^{d} \left\{ f_k \log f_k - (n - f_k) \log (n - f_k) \right\}$$

Therefore, if we split the original cluster into two groups A and B

- The reduction in C is $C_X - C_A - C_B$

Therefore

- The ideal set of clusters would have members with identical attributes and C equal to zero.

Then, if we define

f_k= It is the number of individuals having k^{th} attribute

- We can define the following homogeneity criterion (Information Content):

$$C = dN \log N - \sum_{k=1}^{d} \left\{ f_k \log f_k - (n - f_k) \log (n - f_k) \right\}$$

Therefore, if we split the original cluster into two groups A and B

- The reduction in C is $C_X - C_A - C_B$

Therefore

- The ideal set of clusters would have members with identical attributes and C equal to zero.

Therefore

Clusters are split at each stage

- According to possession of the attribute which leads to the greatest reduction in C.

Other possible splitting can be done using

- Association Analysis (Ecology Term)

Therefore

Clusters are split at each stage

- According to possession of the attribute which leads to the greatest reduction in C.

Other possible splitting can be done using

- Association Analysis (Ecology Term)

For Example

For one pair of variables, v_i and $v_j \in [0,1]$

	v_i	
v_j	1	0
1	f_{11}	f_{10}
0	f_{01}	f_{00}

- f_{ij} = the number of times v_i and v_j coincide or not.

Some common measures of association

$$m_1(f) = |f_{11}f_{00} - f_{10}f_{01}|$$

$$m_2(f) = [f_{11}f_{00} - f_{10}f_{01}]^2$$

For Example

For one pair of variables, v_i and $v_j \in [0,1]$

	v_i	
v_j	1	0
1	f_{11}	f_{10}
0	f_{01}	f_{00}

- f_{ij} = the number of times v_i and v_j coincide or not.

Some common measures of association

$$m_1(\boldsymbol{f}) = |f_{11}f_{00} - f_{10}f_{01}|$$
$$m_2(\boldsymbol{f}) = [f_{11}f_{00} - f_{10}f_{01}]^2$$

Therefore

The split at each stage

- It is made according to the presence or absence of the attribute:
    - Thus, its association with the others is a maximum!!!

Outline

Cinvestav

Algorithms for Large Data Sets

There are several

1. The CURE Algorithm
2. The DBSCAN Algorithm
3. The ROCK Algorithm
4. The Chameleon Algorithm
5. The BIRCH Algorithm

Cinvestav

Algorithms for Large Data Sets

There are several

1. The CURE Algorithm
2. The DBSCAN Algorithm
3. The ROCK Algorithm
4. The Chameleon Algorithm
5. The BIRCH Algorithm

Cinvestav

Algorithms for Large Data Sets

There are several

1. The CURE Algorithm
2. The DBSCAN Algorithm
3. The ROCK Algorithm
4. The Chameleon Algorithm
5. The BIRCH Algorithm

Cinvestav

Algorithms for Large Data Sets

There are several

1. The CURE Algorithm
2. The DBSCAN Algorithm
3. The ROCK Algorithm
4. The Chameleon Algorithm
5. The BIRCH Algorithm

Cinvestav

Algorithms for Large Data Sets

There are several

1. The CURE Algorithm
2. The DBSCAN Algorithm
3. The ROCK Algorithm
4. The Chameleon Algorithm
5. The BIRCH Algorithm

Outline

Cinvestav

Clustering Using REpresentatives (CURE)

Basic Idea

- Each cluster C_i has a set of representatives $R_{C_i} = \left\{ \boldsymbol{x}_1^{(i)}, \boldsymbol{x}_2^{(i)}, ..., \boldsymbol{x}_K^{(i)} \right\}$ with $K > 1$.

Clustering Using REpresentatives (CURE)

Basic Idea

- Each cluster C_i has a set of representatives $R_{C_i} = \left\{ \boldsymbol{x}_1^{(i)}, \boldsymbol{x}_2^{(i)}, ..., \boldsymbol{x}_K^{(i)} \right\}$ with $K > 1$.

What is happening

- By using multiple representatives for each cluster, the CURE algorithm tries to "capture" the shape of each one.

- The initially chosen representatives are "pushed" toward the mean of the cluster.

Clustering Using REpresentatives (CURE)

Basic Idea

- Each cluster C_i has a set of representatives $R_{C_i} = \left\{ \boldsymbol{x}_1^{(i)}, \boldsymbol{x}_2^{(i)}, ..., \boldsymbol{x}_K^{(i)} \right\}$ with $K > 1$.

What is happening

- By using multiple representatives for each cluster, the CURE algorithm tries to "capture" the shape of each one.

However

- In order to avoid taking into account irregularities (For example, outliers) in the border of the cluster.
  - The initially chosen representatives are "pushed" toward the mean of the cluster.

Clustering Using REpresentatives (CURE)

Basic Idea

- Each cluster C_i has a set of representatives $R_{C_i} = \left\{ \boldsymbol{x}_1^{(i)}, \boldsymbol{x}_2^{(i)}, ..., \boldsymbol{x}_K^{(i)} \right\}$ with $K > 1$.

What is happening

- By using multiple representatives for each cluster, the CURE algorithm tries to "capture" the shape of each one.

However

- In order to avoid taking into account irregularities (For example, outliers) in the border of the cluster.
  - The initially chosen representatives are "pushed" toward the mean of the cluster.

Cinvestav

Therfore

This action is known

- As “Shrinking” in the sense that the volume of space “defined” by the representatives is shrunk toward the mean of the cluster.

Outline

Cinvestav

Shrinking Process

Given a cluster C

- Select the point $\boldsymbol{x} \in C$ with the maximum distance from the mean of C and set $R_C = \{\boldsymbol{x}\}$ (the set of representatives).

Shrinking Process

Given a cluster C

- Select the point $\boldsymbol{x} \in C$ with the maximum distance from the mean of C and set $R_C = \{\boldsymbol{x}\}$ (the set of representatives).

Then

1. For $i = 2$ to $\min\{K, n_C\}$
2. Determine $y \in C - R_C$ that lies farthest from the points in R_C
3. $R_C = R_C \cup \{y\}$

Shrinking Process

Given a cluster C

- Select the point $\boldsymbol{x} \in C$ with the maximum distance from the mean of C and set $R_C = \{\boldsymbol{x}\}$ (the set of representatives).

Then

1. For $i = 2$ to $\min\{K, n_C\}$
2. Determine $\boldsymbol{y} \in C - R_C$ that lies farthest from the points in R_C
3. $R_C = R_C \cup \{y\}$

Shrinking Process

Given a cluster C

- Select the point $\boldsymbol{x} \in C$ with the maximum distance from the mean of C and set $R_C = \{\boldsymbol{x}\}$ (the set of representatives).

Then

```
1  For $i = 2$ to $\min\{K, n_C\}$
2      Determine $\boldsymbol{y} \in C - R_C$ that lies farthest from the points in $R_C$
3      $R_C = R_C \cup \{\boldsymbol{y}\}$
```

Shrinking Process

Given a cluster C

- Select the point $\boldsymbol{x} \in C$ with the maximum distance from the mean of C and set $R_C = \{\boldsymbol{x}\}$ (the set of representatives).

Then

```
1 For $i = 2$ to $\min\{K, n_C\}$
2         Determine $\boldsymbol{y} \in C - R_C$ that lies farthest from the points in $R_C$
3         $R_C = R_C \cup \{\boldsymbol{y}\}$
```

Shrinking Process

Do the Shrinking

- Shrink the points $\boldsymbol{x} \in R_C$ toward the mean $\boldsymbol{m}_C$ in C by a factor α.

Actually

$$\boldsymbol{x} = (1-\alpha)\,\boldsymbol{x} + \alpha \boldsymbol{m}_C \;\; \forall \boldsymbol{x} \in R_C \tag{9}$$

Cinvestav

Shrinking Process

Do the Shrinking

- Shrink the points $\boldsymbol{x} \in R_C$ toward the mean $\boldsymbol{m}_C$ in C by a factor α.

Actually

$$\boldsymbol{x} = (1-\alpha)\,\boldsymbol{x} + \alpha \boldsymbol{m}_C \;\forall \boldsymbol{x} \in R_C \tag{9}$$

Resulting set R_C

Thus

- The resulting set R_C is the set of representatives of C.

Thus the distance between two cluster is defined as

$$d(C_i, C_j) = \min_{x \in R_{C_i}, y \in R_{C_j}} d(x, y) \tag{10}$$

Resulting set R_C

Thus

- The resulting set R_C is the set of representatives of C.

Thus the distance between two cluster is defined as

$$d(C_i, C_j) = \min_{\boldsymbol{x} \in R_{C_i}, \boldsymbol{y} \in R_{C_j}} d(\boldsymbol{x}, \boldsymbol{y}) \tag{10}$$

Outline

Cinvestav

Clustering Using REpresentatives (CURE)

Basic Algorithm

Input : A set of points $X = \{x_1, x_2, ..., x_N\}$

Output : C clusters

```
1  For every cluster $C_i = \{x_i\}$ store $C_i.m_C = \{x_i\}$ and $C_i.R_C = \{x_i\}$
2  $C_i.closest$ stores the cluster closest to $C_i$.
3  All the input points are inserted into a $K-d$ tree $T$.
4  Insert each cluster into the heap $Q$. (Clusters are arranged in increasing order of distances
   between $C_i$ and $C_i.closest$).
5  While $size(Q) > C$
6      Remove the top element of Q, $C_i$ and merge it with $C_j == C_i.closest$.
7      Then compute the new representative points for the merged cluster $C_k = C_i \cup C_j$.
8      Also remove $C_i$ and $C_j$ from $T$ and $Q$.
9      Also for all the clusters $C_h \in Q$, update $C_h.closest$ and relocate $C_h$.
10     insert $C_k$ into $Q$
```

Clustering Using REpresentatives (CURE)

Basic Algorithm

Input : A set of points $X = \{\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_N\}$

Output : $\mathcal{C}$ *clusters*

1. *For every cluster* $C_i = \{\boldsymbol{x}_i\}$ *store* $C_i.m_C = \{\boldsymbol{x}_i\}$ *and* $C_i.R_C = \{\boldsymbol{x}_i\}$
2. $C_i.closest$ *stores the cluster closest to* C_i.
3. *All the input points are inserted into a* $K-d$ *tree* T.
4. *Insert each cluster into the heap* Q. *(Clusters are arranged in increasing order of distances between* C_i *and* $C_i.closest$*).*
5. *While* $size(Q) > \mathcal{C}$
6. *Remove the top element of Q,* C_i *and merge it with* $C_j == C_i.closest$.
7. *Then compute the new representative points for the merged cluster* $C_k = C_i \cup C_j$.
8. *Also remove* C_i *and* C_j *from* T *and* Q.
9. *Also for all the clusters* $C_h \in Q$, *update* $C_h.closest$ *and relocate* C_h.
10. *insert* C_k *into* Q

Clustering Using REpresentatives (CURE)

Basic Algorithm

Input : *A set of points* $X = \{\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_N\}$

Output : $\mathcal{C}$ *clusters*

1. *For every cluster* $C_i = \{\boldsymbol{x}_i\}$ *store* $C_i.\boldsymbol{m}_C = \{\boldsymbol{x}_i\}$ *and* $C_i.R_C = \{\boldsymbol{x}_i\}$
2. $C_i.closest$ *stores the cluster closest to* C_i.
3. *All the input points are inserted into a* $K-d$ *tree* T.
4. *Insert each cluster into the heap* Q. *(Clusters are arranged in increasing order of distances between* C_i *and* $C_i.closest$*).*
5. *While* $size(Q) > \mathcal{C}$
6.   *Remove the top element of Q,* C_i *and merge it with* $C_j == C_i.closest$.
7.   *Then compute the new representative points for the merged cluster* $C_k = C_i \cup C_j$.
8.   *Also remove* C_i *and* C_j *from* T *and* Q.
9.   *Also for all the clusters* $C_h \in Q$, *update* $C_h.closest$ *and relocate* C_h.
10.   *insert* C_k *into* Q

Clustering Using REpresentatives (CURE)

Basic Algorithm

Input : A set of points $X = \{\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_N\}$

Output : $\mathcal{C}$ *clusters*

```
1  For every cluster $C_i = \{\boldsymbol{x}_i\}$ store $C_i.\boldsymbol{m}_C = \{\boldsymbol{x}_i\}$ and $C_i.R_C = \{\boldsymbol{x}_i\}$
2  $C_i.closest$ stores the cluster closest to $C_i$.
3  All the input points are inserted into a $K-d$ tree $T$.
4  Insert each cluster into the heap $Q$. (Clusters are arranged in increasing order of distances
   between $C_i$ and $C_i.closest$).
5  While $size(Q) > \mathcal{C}$
6          Remove the top element of $Q$, $C_i$ and merge it with $C_j == C_i.closest$.
7          Then compute the new representative points for the merged cluster $C_k = C_i \cup C_j$.
8          Also remove $C_i$ and $C_j$ from $T$ and $Q$.
9          Also for all the clusters $C_h \in Q$, update $C_h.closest$ and relocate $C_h$.
10         insert $C_k$ into $Q$
```

Clustering Using REpresentatives (CURE)

Basic Algorithm

```
Input : A set of points $X = \{\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_N\}$
Output : $\mathcal{C}$ clusters
1  For every cluster $C_i = \{\boldsymbol{x}_i\}$ store $C_i.\boldsymbol{m}_C = \{\boldsymbol{x}_i\}$ and $C_i.R_C = \{\boldsymbol{x}_i\}$
2  $C_i.closest$ stores the cluster closest to $C_i$.
3  All the input points are inserted into a $K - d$ tree $T$.
4  Insert each cluster into the heap $Q$. (Clusters are arranged in increasing order of distances
   between $C_i$ and $C_i.closest$).
5  While $size(Q) > \mathcal{C}$
6      Remove the top element of $Q$, $C_i$ and merge it with $C_j == C_i.closest$.
7      Then compute the new representative points for the merged cluster $C_k = C_i \cup C_j$.
8      Also remove $C_i$ and $C_j$ from $T$ and $Q$.
9      Also for all the clusters $C_h \in Q$, update $C_h.closest$ and relocate $C_h$.
10     insert $C_k$ into $Q$
```

Clustering Using REpresentatives (CURE)

Basic Algorithm

Input : A set of points $X = \{\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_N\}$

Output : $\mathcal{C}$ *clusters*

```
1  For every cluster $C_i = \{\boldsymbol{x}_i\}$ store $C_i.\boldsymbol{m}_C = \{\boldsymbol{x}_i\}$ and $C_i.R_C = \{\boldsymbol{x}_i\}$
2  $C_i.closest$ stores the cluster closest to $C_i$.
3  All the input points are inserted into a $K-d$ tree $T$.
4  Insert each cluster into the heap $Q$. (Clusters are arranged in increasing order of distances
   between $C_i$ and $C_i.closest$).
5  While $size(Q) > C$
6      Remove the top element of $Q$, $C_i$ and merge it with $C_j == C_i.closest$.
7      Then compute the new representative points for the merged cluster $C_k = C_i \cup C_j$.
8      Also remove $C_i$ and $C_j$ from $T$ and $Q$.
9      Also for all the clusters $C_h \in Q$, update $C_h.closest$ and relocate $C_h$.
10     insert $C_k$ into $Q$
```

Clustering Using REpresentatives (CURE)

Basic Algorithm

Input : A set of points $X = \{\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_N\}$

Output : $\mathcal{C}$ *clusters*

1. *For every cluster* $C_i = \{\boldsymbol{x}_i\}$ *store* $C_i.\boldsymbol{m}_C = \{\boldsymbol{x}_i\}$ *and* $C_i.R_C = \{\boldsymbol{x}_i\}$
2. $C_i.closest$ *stores the cluster closest to* C_i.
3. *All the input points are inserted into a* $K - d$ *tree* T.
4. *Insert each cluster into the heap* Q. *(Clusters are arranged in increasing order of distances between* C_i *and* $C_i.closest$*).*
5. *While* $size(Q) > \mathcal{C}$
6. *Remove the top element of Q,* C_i *and merge it with* $C_j == C_i.closest$.
7. *Then compute the new representative points for the merged cluster* $C_k = C_i \cup C_j$.
8. *Also remove* C_i *and* C_j *from* T *and* Q.
9. *Also for all the clusters* $C_h \in Q$, *update* $C_h.closest$ *and relocate* C_h.
10. *insert* C_k *into* Q

Clustering Using REpresentatives (CURE)

Basic Algorithm

Input : *A set of points* $X = \{\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_N\}$

Output : $\mathcal{C}$ *clusters*

1. *For every cluster* $C_i = \{\boldsymbol{x}_i\}$ *store* $C_i.\boldsymbol{m}_C = \{\boldsymbol{x}_i\}$ *and* $C_i.R_C = \{\boldsymbol{x}_i\}$
2. $C_i.closest$ *stores the cluster closest to* C_i.
3. *All the input points are inserted into a* $K - d$ *tree* T.
4. *Insert each cluster into the heap* Q. *(Clusters are arranged in increasing order of distances between* C_i *and* $C_i.closest$*).*
5. *While* $size(Q) > \mathcal{C}$
6.   *Remove the top element of Q,* C_i *and merge it with* $C_j == C_i.closest$.
7.   *Then compute the new representative points for the merged cluster* $C_k = C_i \cup C_j$.
8.   *Also remove* C_i *and* C_j *from* T *and* Q.
9.   *Also for all the clusters* $C_h \in Q$, *update* $C_h.closest$ *and relocate* C_h.
10.   *insert* C_k *into* Q

Clustering Using REpresentatives (CURE)

Basic Algorithm

Input : A set of points $X = \{\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_N\}$

Output : $\mathcal{C}$ *clusters*

1. *For every cluster* $C_i = \{\boldsymbol{x}_i\}$ *store* $C_i.\boldsymbol{m}_C = \{\boldsymbol{x}_i\}$ *and* $C_i.R_C = \{\boldsymbol{x}_i\}$
2. $C_i.closest$ *stores the cluster closest to* C_i.
3. *All the input points are inserted into a* $K-d$ *tree* T.
4. *Insert each cluster into the heap* Q. *(Clusters are arranged in increasing order of distances between* C_i *and* $C_i.closest$).
5. *While* $size(Q) > \mathcal{C}$
6.   *Remove the top element of Q,* C_i *and merge it with* $C_j == C_i.closest$.
7.   *Then compute the new representative points for the merged cluster* $C_k = C_i \cup C_j$.
8.   *Also remove* C_i *and* C_j *from* T *and* Q.
9.   *Also for all the clusters* $C_h \in Q$, *update* $C_h.closest$ *and relocate* C_h.
10.   *insert* C_k *into* Q

Clustering Using REpresentatives (CURE)

Basic Algorithm

Input : *A set of points* $X = \{\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_N\}$

Output : $\mathcal{C}$ *clusters*

```
1  For every cluster $C_i = \{\boldsymbol{x}_i\}$ store $C_i.\boldsymbol{m}_C = \{\boldsymbol{x}_i\}$ and $C_i.R_C = \{\boldsymbol{x}_i\}$
2  $C_i.closest$ stores the cluster closest to $C_i$.
3  All the input points are inserted into a $K - d$ tree $T$.
4  Insert each cluster into the heap $Q$. (Clusters are arranged in increasing order of distances
   between $C_i$ and $C_i.closest$).
5  While $size(Q) > \mathcal{C}$
6          Remove the top element of Q, $C_i$ and merge it with $C_j == C_i.closest$.
7          Then compute the new representative points for the merged cluster $C_k = C_i \cup C_j$.
8          Also remove $C_i$ and $C_j$ from $T$ and $Q$.
9          Also for all the clusters $C_h \in Q$, update $C_h.closest$ and relocate $C_h$.
10         insert $C_k$ into $Q$
```

Clustering Using REpresentatives (CURE)

Basic Algorithm

```
Input : A set of points $X = \{\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_N\}$
Output : $\mathcal{C}$ clusters
1 For every cluster $C_i = \{\boldsymbol{x}_i\}$ store $C_i.\boldsymbol{m}_C = \{\boldsymbol{x}_i\}$ and $C_i.R_C = \{\boldsymbol{x}_i\}$
2 $C_i.closest$ stores the cluster closest to $C_i$.
3 All the input points are inserted into a $K-d$ tree $T$.
4 Insert each cluster into the heap $Q$. (Clusters are arranged in increasing order of distances
  between $C_i$ and $C_i.closest$).
5 While $size(Q) > \mathcal{C}$
6     Remove the top element of Q, $C_i$ and merge it with $C_j == C_i.closest$.
7     Then compute the new representative points for the merged cluster $C_k = C_i \cup C_j$.
8     Also remove $C_i$ and $C_j$ from $T$ and $Q$.
9     Also for all the clusters $C_h \in Q$, update $C_h.closest$ and relocate $C_h$.
10    insert $C_k$ into $Q$
```

Clustering Using REpresentatives (CURE)

Basic Algorithm

```
Input : A set of points $X = \{\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_N\}$
Output : $\mathcal{C}$ clusters
1  For every cluster $C_i = \{\boldsymbol{x}_i\}$ store $C_i.\boldsymbol{m}_C = \{\boldsymbol{x}_i\}$ and $C_i.R_C = \{\boldsymbol{x}_i\}$
2  $C_i.closest$ stores the cluster closest to $C_i$.
3  All the input points are inserted into a $K-d$ tree $T$.
4  Insert each cluster into the heap $Q$. (Clusters are arranged in increasing order of distances
   between $C_i$ and $C_i.closest$).
5  While $size(Q) > \mathcal{C}$
6        Remove the top element of Q, $C_i$ and merge it with $C_j == C_i.closest$.
7        Then compute the new representative points for the merged cluster $C_k = C_i \cup C_j$.
8        Also remove $C_i$ and $C_j$ from $T$ and $Q$.
9        Also for all the clusters $C_h \in Q$, update $C_h.closest$ and relocate $C_h$.
10       insert $C_k$ into $Q$
```

Outline

Cinvestav

Complexity of Cure

Too Prohibitive

$$O\left(N^2 \log_2 N\right) \tag{11}$$

Possible Solution

CURE does the following

- The technique adopted by the CURE algorithm, in order to reduce the computational complexity, is that of ***random sampling***.

Actually

That is, a sample set X' is created from X, by choosing randomly N' out of the N points of X.

However, one has to ensure that the probability of missing a cluster of X, due to this sampling

This can be guaranteed if the number of points N' is sufficiently large.

Possible Solution

CURE does the following

- The technique adopted by the CURE algorithm, in order to reduce the computational complexity, is that of ***random sampling***.

Actually

That is, a sample set X' is created from X, by choosing randomly N' out of the N points of X.

However, one has to ensure that the probability of missing a cluster of X, due to this sampling

This can be guaranteed if the number of points N' is sufficiently large.

Possible Solution

CURE does the following

- The technique adopted by the CURE algorithm, in order to reduce the computational complexity, is that of ***random sampling***.

Actually

That is, a sample set X' is created from X, by choosing randomly N' out of the N points of X.

However, one has to ensure that the probability of missing a cluster of X, due to this sampling

This can be guaranteed if the number of points N' is sufficiently large.

Cinvestav

Then

Having estimated N'

CURE forms a number of $p = \frac{N}{N'}$ sample data sets by successive random samples.

In other words

- X is partitioned randomly in p subsets.

For this a parameter q is selected

- Then, the points in each partition p are clustered until $\frac{N'}{q}$ clusters are formed.
- The distance between the closest pair of clusters to be merged in the next iteration step exceeds a user-defined threshold.

Cinvestav

Then

Having estimated N'

CURE forms a number of $p = \frac{N}{N'}$ sample data sets by successive random samples.

In other words

- X is partitioned randomly in p subsets.

For this a parameter q is selected

- Then, the points in each partition p are clustered until $\frac{N'}{q}$ clusters are formed.
- The distance between the closest pair of clusters to be merged in the next iteration step exceeds a user-defined threshold.

Then

Having estimated N'

CURE forms a number of $p = \frac{N}{N'}$ sample data sets by successive random samples.

In other words

- X is partitioned randomly in p subsets.

For this a parameter q is selected

- Then, the points in each partition p are clustered until $\frac{N'}{q}$ clusters are formed.
- The distance between the closest pair of clusters to be merged in the next iteration step exceeds a user-defined threshold.

Once this has been finished

A second clustering pass is done

One the at most $p\frac{N'}{q} = \frac{N}{q}$ clusters from all the subsets.

The Goal to apply the merging procedure described previously to all (at most) $\frac{N}{q}$

- Then, we end up with the required final number, C, of clusters.

Finally

- We have the following strategy to assign to $x \in X$ to a cluster.

Once this has been finished

A second clustering pass is done

One the at most $p\frac{N'}{q} = \frac{N}{q}$ clusters from all the subsets.

The Goal to apply the merging procedure described previously to all (at most) $\frac{N}{q}$

- Then, we end up with the required final number, C, of clusters.

Finally

- We have the following strategy to assign to $x \in X$ to a cluster.

Once this has been finished

A second clustering pass is done

One the at most $p\frac{N'}{q} = \frac{N}{q}$ clusters from all the subsets.

The Goal to apply the merging procedure described previously to all (at most) $\frac{N}{q}$

- Then, we end up with the required final number, C, of clusters.

Finally

- We have the following strategy to assign to $x \in X$ to a cluster.

Then

First

A random sample of representative points from each of the C clusters is chosen.

Then

First

A random sample of representative points from each of the C clusters is chosen.

Then

Then, based on the previous representatives the point x is assigned to the cluster that contains the representative closest to it.

- Specifically K must be large enough to capture the geometry of each cluster.
- In addition, K must be higher than a certain percentage [illegible]

Then

First

A random sample of representative points from each of the C clusters is chosen.

Then

Then, based on the previous representatives the point x is assigned to the cluster that contains the representative closest to it.

Experiments reported by Guha et al. show that CURE

- It is sensitive to parameter selection.
  - Specifically K must be large enough to capture the geometry of each cluster.
  - In addition, N' must be higher than a certain percentage $\approx 2.5\%$ of N.

Then

First

A random sample of representative points from each of the C clusters is chosen.

Then

Then, based on the previous representatives the point x is assigned to the cluster that contains the representative closest to it.

Experiments reported by Guha et al. show that CURE

- It is sensitive to parameter selection.
  - Specifically K must be large enough to capture the geometry of each cluster.
  - In addition, N' must be higher than a certain percentage $\approx 2.5\%$ of N.

Then

First

A random sample of representative points from each of the C clusters is chosen.

Then

Then, based on the previous representatives the point x is assigned to the cluster that contains the representative closest to it.

Experiments reported by Guha et al. show that CURE

- It is sensitive to parameter selection.
  - Specifically K must be large enough to capture the geometry of each cluster.
  - In addition, N' must be higher than a certain percentage $\approx 2.5\%$ of N.

Not only that

The value of α (Shrinking Factors) affects also CURE

- Small values, CURE looks similar than a Minimum Spanning Tree clustering.
- Large values, CURE resembles an algorithm with a single representative.

Cinvestav

Not only that

The value of α (Shrinking Factors) affects also CURE

- Small values, CURE looks similar than a Minimum Spanning Tree clustering.
- Large values, CURE resembles an algorithm with a single representative.

The worst-case execution time for CURE increases with the sample size N'

$$O\left(N'^2 \log_2 N'\right) \tag{12}$$

Not only that

The value of α (Shrinking Factors) affects also CURE

- Small values, CURE looks similar than a Minimum Spanning Tree clustering.
- Large values, CURE resembles an algorithm with a single representative.

The worst-case execution time for CURE increases with the sample size N'

$$O\left(N'^2 \log_2 N'\right) \tag{12}$$

Outline

Cinvestav

A Large Name

Density-based spatial clustering of applications with noise (DBSCAN)

- It is a data clustering algorithm proposed by Martin Ester, Hans-Peter Kriegel, Jörg Sander and Xiaowei Xu in 1996.

- Given a set of points in some space, it groups together points that are closely packed together.
- Marking as outliers points that lie alone in low-density regions.

A Large Name

Density-based spatial clustering of applications with noise (DBSCAN)

- It is a data clustering algorithm proposed by Martin Ester, Hans-Peter Kriegel, Jörg Sander and Xiaowei Xu in 1996.

Something Notable

- It is a density-based clustering algorithm:
  - Given a set of points in some space, it groups together points that are closely packed together.
  - Marking as outliers points that lie alone in low-density regions.

Cinvestav

A Large Name

Density-based spatial clustering of applications with noise (DBSCAN)

- It is a data clustering algorithm proposed by Martin Ester, Hans-Peter Kriegel, Jörg Sander and Xiaowei Xu in 1996.

Something Notable

- It is a density-based clustering algorithm:
  - Given a set of points in some space, it groups together points that are closely packed together.
  - Marking as outliers points that lie alone in low-density regions.

Cinvestav

A Large Name

Density-based spatial clustering of applications with noise (DBSCAN)

- It is a data clustering algorithm proposed by Martin Ester, Hans-Peter Kriegel, Jörg Sander and Xiaowei Xu in 1996.

Something Notable

- It is a density-based clustering algorithm:
  - Given a set of points in some space, it groups together points that are closely packed together.
  - Marking as outliers points that lie alone in low-density regions.

Cinvestav

Furthermore

Something Notable

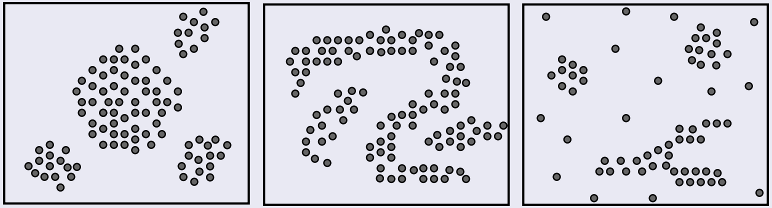
- In 2014, the algorithm was awarded the test of time award at the leading data mining conference, KDD.

Outline

Cinvestav

Looking at clusters

We notice easily those clusters of points and noise points



We are doing something quite human

The main reason why we recognize the clusters

- We use the higher densities to recognize the clusters

Definition (ϵ-neighborhood of a point)

- Given a distance $\text{dist} : \mathbb{R}^d \times \mathbb{R}^d \rightarrow \mathbb{R}^+$, the ϵ-Neighborhood of a point $\boldsymbol{x}$, denoted $N_\epsilon(\boldsymbol{x})$, is defined as
$$N_\epsilon(\boldsymbol{x}) = \left\{ \boldsymbol{y} \in \mathbb{R}^d | \text{dist}(\boldsymbol{x}, \boldsymbol{y}) \leq \epsilon \right\}$$

Cinvestav

We are doing something quite human

The main reason why we recognize the clusters

- We use the higher densities to recognize the clusters

Definition (ϵ-neighborhood of a point)

- Given a distance dist : $\mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}^+$, the ϵ-Neighborhood of a point $\boldsymbol{x}$, denoted $N_\epsilon(\boldsymbol{x})$, is defined as

$$N_\epsilon(\boldsymbol{x}) = \left\{\boldsymbol{y} \in \mathbb{R}^d | \text{dist}(\boldsymbol{x}, \boldsymbol{y}) \leq \epsilon\right\}$$

Outline

Cinvestav

Why not to use the idea of K-NN?

We could use our well know K-NN method

- Thus, we naively could require for each point in a cluster there at least a minimum number $(MinPts)$ of points in the neighborhood of such point

However, you have something more complex

- Points inside of the cluster (Core points)
- Points on the border of the cluster (Border points)

Why not to use the idea of K-NN?

We could use our well know K-NN method

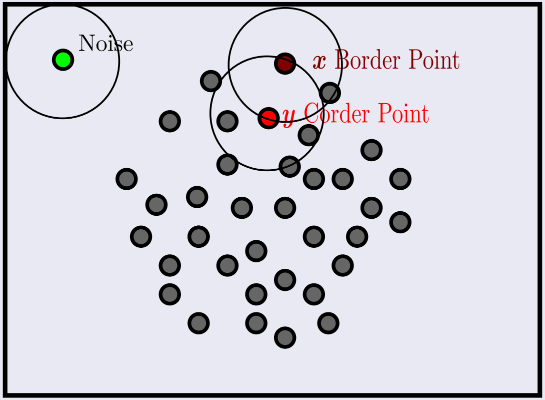
- Thus, we naively could require for each point in a cluster there at least a minimum number ($MinPts$) of points in the neighborhood of such point

However, you have something more complex

- Points inside of the cluster (Core points)
- Points on the border of the cluster (Border points)

Example

Core points and Border points



Therefore

In General

- An ϵ-neighborhood of a border point contains significantly less points than an ϵ-neighborhood of a core point.

Therefore

- $MinPts$ varies in the presence of noise

Cinvestav

Therefore

In General

- An ϵ-neighborhood of a border point contains significantly less points than an ϵ-neighborhood of a core point.

Therefore

- $MinPts$ varies in the presence of noise

Thus, we can ask for a new restriction

Definition (Directly Density-Reachable)

- A point $\boldsymbol{x}$ is directly density reachable from a point $\boldsymbol{y}$ w.r.t. ϵ, $MinPts$ if
  1. $\boldsymbol{x} \in N_{\epsilon}(\boldsymbol{y})$
  2. $|N_{\epsilon}(\boldsymbol{y})| \geq MinPts$ (Core point condition)

- It is not symmetric if one core point and one border point are involved

Thus, we can ask for a new restriction

Definition (Directly Density-Reachable)

- A point $\boldsymbol{x}$ is directly density reachable from a point $\boldsymbol{y}$ w.r.t. ϵ, $MinPts$ if
  1. $\boldsymbol{x} \in N_{\epsilon}(\boldsymbol{y})$
  2. $|N_{\epsilon}(\boldsymbol{y})| \geq MinPts$ (Core point condition)

Remarks

1. Directly density-reachable is symmetric for pairs of core points.
2. It is not symmetric if one core point and one border point are involved.

Cinvestav

Thus, we can ask for a new restriction

Definition (Directly Density-Reachable)

- A point $\boldsymbol{x}$ is directly density reachable from a point $\boldsymbol{y}$ w.r.t. ϵ, $MinPts$ if
  1. $\boldsymbol{x} \in N_{\epsilon}(\boldsymbol{y})$
  2. $|N_{\epsilon}(\boldsymbol{y})| \geq MinPts$ (Core point condition)

Remarks

1. Directly density-reachable is symmetric for pairs of core points.
2. It is not symmetric if one core point and one border point are involved.

Cinvestav

Thus, we can ask for a new restriction

Definition (Directly Density-Reachable)

- A point $\boldsymbol{x}$ is directly density reachable from a point $\boldsymbol{y}$ w.r.t. ϵ, $MinPts$ if
  1. $\boldsymbol{x} \in N_{\epsilon}(\boldsymbol{y})$
  2. $|N_{\epsilon}(\boldsymbol{y})| \geq MinPts$ (Core point condition)

Remarks

1. Directly density-reachable is symmetric for pairs of core points.
2. It is not symmetric if one core point and one border point are involved.

Thus, we can ask for a new restriction

Definition (Directly Density-Reachable)

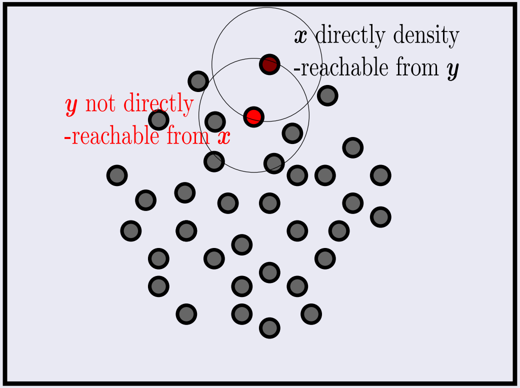
- A point $\boldsymbol{x}$ is directly density reachable from a point $\boldsymbol{y}$ w.r.t. ϵ, $MinPts$ if
  1. $\boldsymbol{x} \in N_\epsilon(\boldsymbol{y})$
  2. $|N_\epsilon(\boldsymbol{y})| \geq MinPts$ (Core point condition)

Remarks

1. Directly density-reachable is symmetric for pairs of core points.
2. It is not symmetric if one core point and one border point are involved.

Example

Density-reachable



Now, Density Reachable

Definition (Density-Reachable)

- A point $\boldsymbol{x}$ is density-reachable from a point $\boldsymbol{y}$ wrt. ϵ and MinPts if there is a chain of points:

$$\boldsymbol{p}_1, \boldsymbol{p}_2, ..., \boldsymbol{p}_k \text{ with } \boldsymbol{p}_1 = \boldsymbol{x}, \boldsymbol{p}_k = \boldsymbol{y}$$

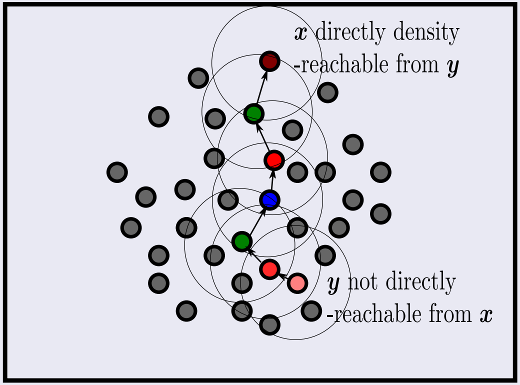
such that $\boldsymbol{p}_{i+1}$ is directly density-reachable from $\boldsymbol{p}_i$.

Cinvestav

Therefore

Density-Reachability is a canonical extension of Direct Density-Reachability

- This relation is transitive, but it is not symmetric.



Then

Remark

- Two border points of the same cluster C are possibly not density reachable from each other:
  - The core point condition might not hold for both of them.

However

- There must be a core point in C from which both border points of C are density-reachable.

Then

Remark

- Two border points of the same cluster C are possibly not density reachable from each other:
  - The core point condition might not hold for both of them.

However

- There must be a core point in C from which both border points of C are density-reachable.

Density-Connected

Definition (Density-Connected)

- A point $\boldsymbol{x}$ is density-connected to a point $\boldsymbol{y}$ w.r.t. ϵ and $MinPts$:
    - if there is a point $\boldsymbol{o}$ such that both, $\boldsymbol{x}$ and $\boldsymbol{y}$ are density-reachable from $\boldsymbol{o}$ w.r.t. ϵ and $MinPts$.

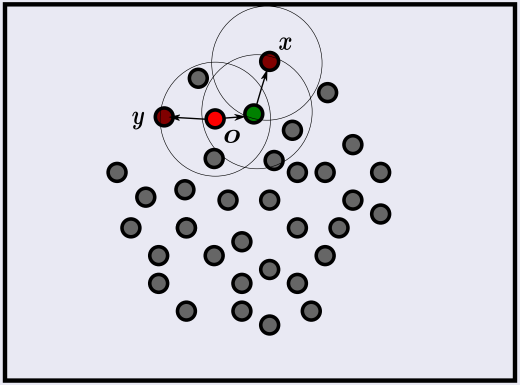
Example

Density-Connected

Definition (Density-Connected)

- A point $\boldsymbol{x}$ is density-connected to a point $\boldsymbol{y}$ w.r.t. ϵ and $MinPts$:
  - if there is a point $\boldsymbol{o}$ such that both, $\boldsymbol{x}$ and $\boldsymbol{y}$ are density-reachable from $\boldsymbol{o}$ w.r.t. ϵ and $MinPts$.

Example



Symmetry in Density-Connectivity

Density-connectivity is a symmetric relation

- Also for density reachable points, the relation of density-connectivity is also reflexive.

We are ready to define the concept of Cluster

- From the point of view density-based

Remark

- Intuitively, a cluster is defined to be a set of density-connected points which is maximal w.r.t. density-reachability.
- Noise is simply the set of points in $\mathbb{R}^d$ not belonging to any of its clusters.

Symmetry in Density-Connectivity

Density-connectivity is a symmetric relation

- Also for density reachable points, the relation of density-connectivity is also reflexive.

We are ready to define the concept of Cluster

- From the point of view density-based

Remark

- Intuitively, a cluster is defined to be a set of density-connected points which is maximal w.r.t. density-reachability.
- Noise is simply the set of points in $\mathbb{R}^d$ not belonging to any of its clusters.

Cinvestav

Symmetry in Density-Connectivity

Density-connectivity is a symmetric relation

- Also for density reachable points, the relation of density-connectivity is also reflexive.

We are ready to define the concept of Cluster

- From the point of view density-based

Remark

- Intuitively, a cluster is defined to be a set of density-connected points which is maximal w.r.t. density-reachability.
- Noise is simply the set of points in $\mathbb{R}^d$ not belonging to any of its clusters.

Outline

Cluster Definition

Definition

- A cluster C w.r.t. ϵ and $MinPts$ is a non-empty subset of $\mathbb{R}^d$ satisfying the following conditions:
  1. $\forall \boldsymbol{x}, \boldsymbol{y}$: if $\boldsymbol{y} \in C$ and $\boldsymbol{x}$ is density-reachable from $\boldsymbol{y}$ w.r.t. ϵ and $MinPts$ then $\boldsymbol{x} \in C$ (Maximality).
  2. $\forall \boldsymbol{x}, \boldsymbol{y} \in C$, $\boldsymbol{x}$ is density-connected to $\boldsymbol{y}$ w.r.t. ϵ and $MinPts$ (Connectivity).

Noise Definition

Definition

- Let $C_1, ..., C_k$ be the clusters in $\mathbb{R}^d$ w.r.t. parameters ϵ_i and $MinPts_i$, $i = 1, ..., k$.
  - Then we define the noise as the set of points in $\mathbb{R}^d$ not belonging to any cluster C_i:

$$Noise = \left\{ \boldsymbol{x} \in \mathbb{R}^d | \forall i : \boldsymbol{x} \notin C_i \right\}$$

Cinvestav

Remarks

Something Notable

- Since C contains at least one point x.
- x must be density-connected to itself via some point o.
  - which may be equal to x.
- Thus, at least o has to satisfy the core point condition
  - Consequently, ϵ-Neighborhood of o contains at least $MinPts$

Remarks

Something Notable

- Since C contains at least one point $\boldsymbol{x}$.
- $\boldsymbol{x}$ must be density-connected to itself via some point $\boldsymbol{o}$.
  - which may be equal to $\boldsymbol{x}$.
- Thus, at least o has to satisfy the core point condition
  - Consequently, ϵ-Neighborhood of o contains at least $MinPts$

Remarks

Something Notable

- Since C contains at least one point $\boldsymbol{x}$.
- $\boldsymbol{x}$ must be density-connected to itself via some point $\boldsymbol{o}$.
  - which may be equal to $\boldsymbol{x}$.
- Thus, at least $\boldsymbol{o}$ has to satisfy the core point condition
  - Consequently, ϵ-Neighborhood of o contains at least $MinPts$

Outline

Cinvestav

Supporting the idea of cluster

Lemma - Reachability

- Let $\boldsymbol{y}$ be a point in $\mathbb{R}^d$ and $|N_\epsilon(\boldsymbol{y})| \geq MinPts$. Then

$$O = \left\{\boldsymbol{o} | \boldsymbol{o} \in \mathbb{R}^d \text{ and } \boldsymbol{o} \text{ is density-reachable from } \boldsymbol{y} \text{ w.r.t. } \epsilon \text{ and } MinPts\right\}$$

is a cluster w.r.t. ϵ and $MinPts$.

Cinvestav

Proof

Given the definition of O

- We have the first part of the definition of Cluster w.r.t. ϵ and $MinPts$.

Given the that two points $o_1, o_2 \in O$ are density-reachable from y

- o_1, o_2 are density connected.

Then

- O is a cluster w.r.t. ϵ and $MinPts$.

Proof

Given the definition of O

- We have the first part of the definition of Cluster w.r.t. ϵ and $MinPts$.

Given the that two points $\boldsymbol{o}_1, \boldsymbol{o}_2 \in O$ are density-reachable from $\boldsymbol{y}$

- $\boldsymbol{o}_1, \boldsymbol{o}_2$ are density connected.

Then

- O is a cluster w.r.t. ϵ and $MinPts$.

Proof

Given the definition of O

- We have the first part of the definition of Cluster w.r.t. ϵ and $MinPts$.

Given the that two points $\boldsymbol{o}_1, \boldsymbol{o}_2 \in O$ are density-reachable from $\boldsymbol{y}$

- $\boldsymbol{o}_1, \boldsymbol{o}_2$ are density connected.

Then

- O is a cluster w.r.t. ϵ and $MinPts$.

Cinvestav

Intuition

Given the parameters ϵ and $MinPts$, we can discover a cluster

- First, choose an arbitrary point from $\mathbb{R}^d$ satisfying the core point condition as a seed.

Then

- Retrieve all points that are density-reachable from the seed **obtaining the cluster** containing the seed.

Cinvestav

Intuition

Given the parameters ϵ and $MinPts$, we can discover a cluster

- First, choose an arbitrary point from $\mathbb{R}^d$ satisfying the core point condition as a seed.

Then

- Retrieve all points that are density-reachable from the seed **obtaining the cluster** containing the seed.

However, it is not enough

We need something else

- Given that it is not obvious that a cluster C w.r.t. ϵ and $MinPts$ is uniquely determined by any of its core points.

- A cluster C contains exactly the points which are density-reachable from an arbitrary core point of C.

However, it is not enough

We need something else

- Given that it is not obvious that a cluster C w.r.t. ϵ and $MinPts$ is uniquely determined by any of its core points.

However

- Each point in C is density-reachable from any of the core points of C.
- A cluster C contains exactly the points which are density-reachable from an arbitrary core point of C.

Cinvestav

However, it is not enough

We need something else

- Given that it is not obvious that a cluster C w.r.t. ϵ and $MinPts$ is uniquely determined by any of its core points.

However

- Each point in C is density-reachable from any of the core points of C.
- A cluster C contains exactly the points which are density-reachable from an arbitrary core point of C.

Cinvestav

Then

Lemma - Cluster Equality to O

- Let C be a cluster w.r.t. ϵ and $MinPts$. and let $\boldsymbol{y}$ be any point in C with $|N_\epsilon(\boldsymbol{y})| \geq MinPts$
  - Then C equals to the set

$$O = \left\{ \boldsymbol{o} | \boldsymbol{o} \in \mathbb{R}^d \text{ and } \boldsymbol{o} \text{ is density-reachable from } \boldsymbol{y} \text{ w.r.t. } \epsilon \text{ and } MinPts \right\}$$

Proof

Given $x \in C$

- We have two cases

Case 1

- x is a Border point that is density reachable from y with $|N_{\epsilon}(y)| \geq MinPts$

Then

- $x \in O$

Proof

Given $x \in C$

- We have two cases

Case 1

- x is a Border point that is density reachable from y with $|N_\epsilon(y)| \geq MinPts$

Then

- $x \in O$

Proof

Given $\boldsymbol{x} \in C$

- We have two cases

Case 1

- $\boldsymbol{x}$ is a Border point that is density reachable from $\boldsymbol{y}$ with $|N_\epsilon(\boldsymbol{y})| \geq MinPts$

Then

- $\boldsymbol{x} \in O$

Now

Case 2

- $\boldsymbol{x}$ is a Core point then $|N_\epsilon(\boldsymbol{x})| \geq MinPts$

Therefore

- By Definition x is density reachable from any y w.r.t. ϵ and $MinPts$.

Therefore

- $C \subset O$ the other contention is similar

Now

Case 2

- $\boldsymbol{x}$ is a Core point then $|N_{\epsilon}(\boldsymbol{x})| \geq MinPts$

Therefore

- By Definition $\boldsymbol{x}$ is density reachable from any $\boldsymbol{y}$ w.r.t. ϵ and $MinPts$.

Therefore

- $C \subseteq O$ the other contention is similar

Now

Case 2

- $\boldsymbol{x}$ is a Core point then $|N_{\epsilon}(\boldsymbol{x})| \geq MinPts$

Therefore

- By Definition $\boldsymbol{x}$ is density reachable from any $\boldsymbol{y}$ w.r.t. ϵ and $MinPts$.

Therefore

- $C \subset O$ the other contention is similar

Outline

Cinvestav

DBSCAN Algorithm

DBSCAN$(X, \epsilon, MinPts)$

```
1 $ClusterId = nextId(NOISE)$
2 for $i = 1$ to $X.size$
3         $\boldsymbol{x} = SetOfPoints.get(i)$
4         if $\boldsymbol{x}.ClId$ is UNCLASSIFIED:
5                 If **ExpandCluster**$(SetPoints, \boldsymbol{x}, ClusterId, \epsilon, MinPts)$
6                         $ClusterId = nextId(ClusterId)$
```

ExpandCluster$(SetPoints, Point, ClId, \epsilon, MinPts)$

```
1  $seed = SetOfPoints.regionQuery\left(Point, \epsilon\right)$
2  If $seeds.size < MinPts$ Then
3          $SetPoints.changeClId\left(Point, NOISE\right)$
4          return FALSE
5  else
6          $SetPoints.changeClId\left(seeds, ClId\right)$
7          $seeds.delete\left(Point\right)$
8          while $seeds \neq NULL$:
9              $currentP = seeds.first\left(\right)$
10             $result = SetOfPoints.regionQuery\left(currentP, \epsilon\right)$
11             if $result.size\left(\right) > MinPts$ then
12                 for $i = 1$ to $result.size$:
13                     $resultP = result.get\left(i\right)$
14                     if $resultP.ClId \in \{NOISE, UNCLASSSIFIED\}$ and
                                            $resultP.ClId = UNCLASSSIFIED$
15                         $seeds.append\left(resultP\right)$
16                     $SetPoints.changeClId\left(resultP, ClId\right)$
17                 $seeds.delete\left(currentP\right)$
18         return TRUE
```

Cinvestav

ExpandCluster$(SetPoints, Point, ClId, \epsilon, MinPts)$

```
1  $seed = SetOfPoints.regionQuery\,(Point, \epsilon)$
2  If $seeds.size < MinPts$ Then
3        $SetPoints.changeClId\,(Point, NOISE)$
4        return FALSE
5  else
6        $SetPoints.changeClId\,(seeds, ClId)$
7        $seeds.delete\,(Point)$
8        while $seeds \neq NULL$:
9            $currentP = seeds.first\,()$
10           $result = SetOfPoints.regionQuery\,(currentP, \epsilon)$
11           if $result.size\,() > MinPts$ then
12               for $i = 1$ to $result.size$:
13                   $resultP = result.get\,(i)$
14                   if $resultP.ClId \in \{NOISE, UNCLASSSIFIED\}$ and
                          $resultP.ClId = UNCLASSSIFIED$
15                       $seeds.append\,(resultP)$
16                   $SetPoints.changeClId\,(resultP, ClId)$
17               $seeds.delete\,(currentP)$
18       return TRUE
```

ExpandCluster$(SetPoints, Point, ClId, \epsilon, MinPts)$

```
1  $seed = SetOfPoints.regionQuery\left(Point, \epsilon\right)$
2  If $seeds.size < MinPts$ Then
3          $SetPoints.changeClId\left(Point, NOISE\right)$
4          return FALSE
5  else
6          $SetPoints.changeClId\left(seeds, ClId\right)$
7          $seeds.delete\left(Point\right)$
8          while $seeds \neq NULL$:
9                  $currentP = seeds.first\left(\right)$
10                 $result = SetOfPoints.regionQuery\left(currentP, \epsilon\right)$
11                 if $result.size\left(\right) > MinPts$ then
12                         for $i = 1$ to $result.size$:
13                                 $resultP = result.get\left(i\right)$
14                                 if $resultP.ClId \in \{NOISE, UNCLASSSIFIED\}$ and
                                         $resultP.ClId = UNCLASSSIFIED$
15                                         $seeds.append\left(resultP\right)$
16                                 $SetPoints.changeClId\left(resultP, ClId\right)$
17                         $seeds.delete\left(currentP\right)$
18                 return TRUE
```

ExpandCluster$(SetPoints, Point, ClId, \epsilon, MinPts)$

```
1  $seed = SetOfPoints.regionQuery\left(Point, \epsilon\right)$
2  If $seeds.size < MinPts$ Then
3      $SetPoints.changeClId\left(Point, NOISE\right)$
4      return FALSE
5  else
6      $SetPoints.changeClId\left(seeds, ClId\right)$
7      $seeds.delete\left(Point\right)$
8      while $seeds \neq NULL$:
9          $currentP = seeds.first\left(\right)$
10         $result = SetOfPoints.regionQuery\left(currentP, \epsilon\right)$
11         if $result.size\left(\right) > MinPts$ then
12             for $i = 1$ to $result.size$:
13                 $resultP = result.get\left(i\right)$
14                 if $resultP.ClId \in \{NOISE, UNCLASSSIFIED\}$ and
                       $resultP.ClId = UNCLASSSIFIED$
15                     $seeds.append\left(resultP\right)$
16                 $SetPoints.changeClId\left(resultP, ClId\right)$
17         $seeds.delete\left(currentP\right)$
18     return TRUE
```

Cinvestav

ExpandCluster$(SetPoints, Point, ClId, \epsilon, MinPts)$

```
1  $seed = SetOfPoints.regionQuery\,(Point, \epsilon)$
2  If $seeds.size < MinPts$ Then
3          $SetPoints.changeClId\,(Point, NOISE)$
4          return FALSE
5  else
6          $SetPoints.changeClId\,(seeds, ClId)$
7          $seeds.delete\,(Point)$
8          while $seeds \neq NULL$:
9              $currentP = seeds.first\,()$
10             $result = SetOfPoints.regionQuery\,(currentP, \epsilon)$
11             if $result.size\,() > MinPts$ then
12                 for $i = 1$ to $result.size$:
13                     $resultP = result.get\,(i)$
14                     if $resultP.ClId \in \{NOISE, UNCLASSSIFIED\}$ and
                                           $resultP.ClId = UNCLASSSIFIED$
15                         $seeds.append\,(resultP)$
16                         $SetPoints.changeClId\,(resultP, ClId)$
17                 $seeds.delete\,(currentP)$
18             return TRUE
```

Cinvestav

ExpandCluster$(SetPoints, Point, ClId, \epsilon, MinPts)$

```
1  $seed = SetOfPoints.regionQuery\,(Point, \epsilon)$
2  If $seeds.size < MinPts$ Then
3          $SetPoints.changeClId\,(Point, NOISE)$
4          return FALSE
5  else
6          $SetPoints.changeClId\,(seeds, ClId)$
7          $seeds.delete\,(Point)$
8          while $seeds \neq NULL$:
9              $currentP = seeds.first\,()$
10             $result = SetOfPoints.regionQuery\,(currentP, \epsilon)$
11             if $result.size\,() > MinPts$ then
12                 for $i = 1$ to $result.size$:
13                     $resultP = result.get\,(i)$
14                     if $resultP.ClId \in \{NOISE, UNCLASSSIFIED\}$ and
                           $resultP.ClId = UNCLASSSIFIED$
15                         $seeds.append\,(resultP)$
16                     $SetPoints.changeClId\,(resultP, ClId)$
17                 $seeds.delete\,(currentP)$
18             return TRUE
```

ExpandCluster$(SetPoints, Point, ClId, \epsilon, MinPts)$

```
1  $seed = SetOfPoints.regionQuery(Point, \epsilon)$
2  If $seeds.size < MinPts$ Then
3          $SetPoints.changeClId(Point, NOISE)$
4          return FALSE
5  else
6          $SetPoints.changeClId(seeds, ClId)$
7          $seeds.delete(Point)$
8          while $seeds \neq NULL$:
9                $currentP = seeds.first()$
10              $result = SetOfPoints.regionQuery(currentP, \epsilon)$
11               if $result.size() > MinPts$ then
12                     for $i = 1$ to $result.size$:
13                           $resultP = result.get(i)$
14                           if $resultP.ClId \in \{NOISE, UNCLASSSIFIED\}$ and
                                               $resultP.ClId = UNCLASSSIFIED$
15                                 $seeds.append(resultP)$
16                           $SetPoints.changeClId(resultP, ClId)$
17                     $seeds.delete(currentP)$
18              return TRUE
```

Outline

Cinvestav

Complexity

$x$is a core point

- It can be implemented using kd-trees

Thus, given the complexities of Kd-trees

	Average	Worst case
Space	$O(n)$	$O(n)$
Search	$O(\log n)$	$O(n)$
Insert	$O(\log n)$	$O(n)$
Delete	$O(\log n)$	$O(n)$

Complexity

$x$is a core point

- It can be implemented using kd-trees

Thus, given the complexities of Kd-trees

	Average	Worst case
Space	$O(n)$	$O(n)$
Search	$O(\log n)$	$O(n)$
Insert	$O(\log n)$	$O(n)$
Delete	$O(\log n)$	$O(n)$

Cinvestav

Therefore

The average Complexity of DBSCAN

- $O\left(dn \log n\right)$ to build the structure for query using a heapsort or mergesort
- $O\left(\left\{n^{1-\frac{1}{d}} + m\right\}\right)$ when m is the number of reported elements and d is the dimensionality of the points.

Outline

Cinvestav

There is a problem

How do we estimate?

- ϵ and $MinPts$.

In the original paper

- They develop a heuristic to determine the parameters ϵ and $MinPts$ of the "thinnest"

There is a problem

How do we estimate?

- ϵ and $MinPts$.

In the original paper

- They develop a heuristic to determine the parameters ϵ and $MinPts$ of the "thinnest"

Heuristic

Let d be the distance of a point x

- to its k^{th} nearest neighbor.

Then, the d-neighborhood of x contains exactly

- $k+1$ points for almost all points x.

The d-neighborhood of x contains more than $k+1$ points

- Only if several points have exactly the same distance d from x which is quite unlikely.

Heuristic

Let d be the distance of a point x

- to its k^{th} nearest neighbor.

Then, the d-neighborhood of x contains exactly

- $k+1$ points for almost all points x.

The d-neighborhood of x contains more than $k+1$ points

- Only if several points have exactly the same distance d from x which is quite unlikely.

Heuristic

Let d be the distance of a point $\boldsymbol{x}$

- to its k^{th} nearest neighbor.

Then, the d-neighborhood of $\boldsymbol{x}$ contains exactly

- $k+1$ points for almost all points $\boldsymbol{x}$.

The d-neighborhood of $\boldsymbol{x}$ contains more than $k+1$ points

- Only if several points have exactly the same distance d from $\boldsymbol{x}$ which is quite unlikely.

Then

Furthermore

- Changing k for a point in a cluster does not result in large changes of d.

This only happens if the k^{th} nearest neighbors of x

- for $k = 1, 2, 3, \ldots$ are located approximately
  - on a straight line which is in general not true for a point in a cluster.

Then

Furthermore

- Changing k for a point in a cluster does not result in large changes of d.

This only happens if the k^{th} nearest neighbors of $\boldsymbol{x}$

- for $k = 1, 2, 3, \ldots$ are located approximately
  - on a straight line which is in general not true for a point in a cluster.

Then, we have

For a given k we define a function k-dist from $\mathbb{R}^d$ to $\mathbb{R}$

- Mapping each point to the distance from its k^{th} nearest neighbor.

When sorting the points of the database in descending order of their k-dist values

- The graph of this function gives some hints concerning the density distribution in the database.

Then, we have

For a given k we define a function k-dist from $\mathbb{R}^d$ to $\mathbb{R}$

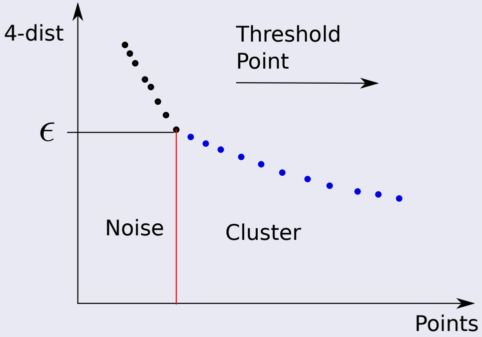
- Mapping each point to the distance from its k^{th} nearest neighbor.

When sorting the points of the database in descending order of their k-dist values

- The graph of this function gives some hints concerning the density distribution in the database.

Example of 4-dist

We set $MinPts = 4$ then we can calculate ϵ by looking



Therefore

For more in the heuristic look at the paper

- "A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise" by Martin Ester, Hans-Peter Kriegel, Jorg Sander, Xiaowei Xu

However, the problem is the same

- Finding the correct number of hyperparameters for getting the correct number of clusters

More advanced methods of clustering exist

- Spectral Clustering - Using the Graph Structure
- Dirichlet Processes - Based in the Generation of a Distribution
- etc

Therefore

For more in the heuristic look at the paper

- "A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise" by Martin Ester, Hans-Peter Kriegel, Jorg Sander, Xiaowei Xu

However, the problem is the same

- Finding the correct number of hyperparameters for getting the correct number of clusters

More advanced methods of clustering exist

- Spectral Clustering - Using the Graph Structure
- Dirichlet Processes - Based in the Generation of a Distribution
- etc

Therefore

For more in the heuristic look at the paper

- “A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise” by Martin Ester, Hans-Peter Kriegel, Jorg Sander, Xiaowei Xu

However, the problem is the same

- Finding the correct number of hyperparameters for getting the correct number of clusters

More advanced methods of clustering exist

- Spectral Clustering - Using the Graph Structure
- Dirichlet Processes - Based in the Generation of a Distribution
- etc