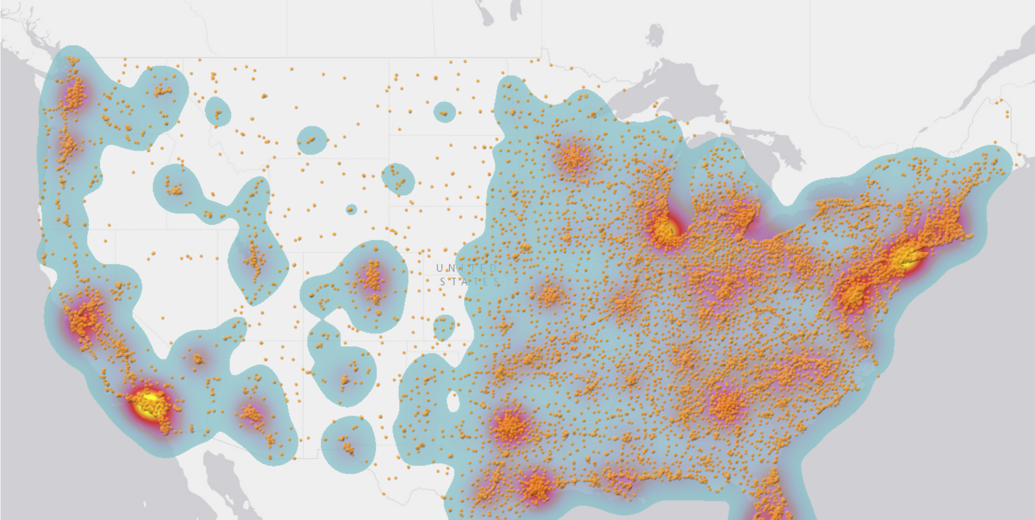
CS 474/574 Machine Learning
7. Clustering

Prof. Dr. Forrest Sheng Bao
Dept. of Computer Science
Iowa State University
Ames, IA, USA

April 5, 2021

It's our nature to see clustering patterns

Distribution of Subway restuarants (Source: https://i.insider.com/58337a73ba6eb6b1018b5a71)



We cluster before we classify

- Breeds of dogs

We cluster before we classify

- Breeds of dogs
- Species of iris flowers

Clustering

1. An **un**supervised approach: no ground truth provided to the computer

Clustering

1. An **un**supervised approach: no ground truth provided to the computer
2. What to do: grouping samples into clusters, such that (usually) samples sharing similar properties belong to the same cluster

Clustering

1. An **un**supervised approach: no ground truth provided to the computer
2. What to do: grouping samples into clusters, such that (usually) samples sharing similar properties belong to the same cluster
3. Clustering usually is not used as an end-to-end solution to a problem (e.g., classification or regression), but a step in a pipeline of data analytics.

Clustering

1. An **un**supervised approach: no ground truth provided to the computer
2. What to do: grouping samples into clusters, such that (usually) samples sharing similar properties belong to the same cluster
3. Clustering usually is not used as an end-to-end solution to a problem (e.g., classification or regression), but a step in a pipeline of data analytics.
4. Applications:

Clustering

1. An **un**supervised approach: no ground truth provided to the computer
2. What to do: grouping samples into clusters, such that (usually) samples sharing similar properties belong to the same cluster
3. Clustering usually is not used as an end-to-end solution to a problem (e.g., classification or regression), but a step in a pipeline of data analytics.
4. Applications:
    - Data partitioning (e.g., separate background with foreground in IP/CV)

Clustering

1. An **un**supervised approach: no ground truth provided to the computer
2. What to do: grouping samples into clusters, such that (usually) samples sharing similar properties belong to the same cluster
3. Clustering usually is not used as an end-to-end solution to a problem (e.g., classification or regression), but a step in a pipeline of data analytics.
4. Applications:
   - Data partitioning (e.g., separate background with foreground in IP/CV)
   - Data compression (e.g., many samples are compressed to one sample the represents them)

Clustering

1. An **un**supervised approach: no ground truth provided to the computer
2. What to do: grouping samples into clusters, such that (usually) samples sharing similar properties belong to the same cluster
3. Clustering usually is not used as an end-to-end solution to a problem (e.g., classification or regression), but a step in a pipeline of data analytics.
4. Applications:
   - Data partitioning (e.g., separate background with foreground in IP/CV)
   - Data compression (e.g., many samples are compressed to one sample the represents them)
   - Data visualization (e.g., showing that the features selected or engineered make sense, sklearn demo on Iris dataset)

Clustering

1. An **un**supervised approach: no ground truth provided to the computer
2. What to do: grouping samples into clusters, such that (usually) samples sharing similar properties belong to the same cluster
3. Clustering usually is not used as an end-to-end solution to a problem (e.g., classification or regression), but a step in a pipeline of data analytics.
4. Applications:
   - Data partitioning (e.g., separate background with foreground in IP/CV)
   - Data compression (e.g., many samples are compressed to one sample the represents them)
   - Data visualization (e.g., showing that the features selected or engineered make sense, sklearn demo on Iris dataset)
   - Simple but usually explanable classification (e.g., in combination with k-NN classifiers)

Clustering

1. An **un**supervised approach: no ground truth provided to the computer
2. What to do: grouping samples into clusters, such that (usually) samples sharing similar properties belong to the same cluster
3. Clustering usually is not used as an end-to-end solution to a problem (e.g., classification or regression), but a step in a pipeline of data analytics.
4. Applications:
   - Data partitioning (e.g., separate background with foreground in IP/CV)
   - Data compression (e.g., many samples are compressed to one sample the represents them)
   - Data visualization (e.g., showing that the features selected or engineered make sense, sklearn demo on Iris dataset)
   - Simple but usually explanable classification (e.g., in combination with k-NN classifiers)
5. Approaches

Clustering

1. An **un**supervised approach: no ground truth provided to the computer
2. What to do: grouping samples into clusters, such that (usually) samples sharing similar properties belong to the same cluster
3. Clustering usually is not used as an end-to-end solution to a problem (e.g., classification or regression), but a step in a pipeline of data analytics.
4. Applications:
   - Data partitioning (e.g., separate background with foreground in IP/CV)
   - Data compression (e.g., many samples are compressed to one sample the represents them)
   - Data visualization (e.g., showing that the features selected or engineered make sense, sklearn demo on Iris dataset)
   - Simple but usually explanable classification (e.g., in combination with k-NN classifiers)
5. Approaches
   - centroid-based (e.g., k-means)

Clustering

1. An **un**supervised approach: no ground truth provided to the computer
2. What to do: grouping samples into clusters, such that (usually) samples sharing similar properties belong to the same cluster
3. Clustering usually is not used as an end-to-end solution to a problem (e.g., classification or regression), but a step in a pipeline of data analytics.
4. Applications:
   - Data partitioning (e.g., separate background with foreground in IP/CV)
   - Data compression (e.g., many samples are compressed to one sample the represents them)
   - Data visualization (e.g., showing that the features selected or engineered make sense, sklearn demo on Iris dataset)
   - Simple but usually explanable classification (e.g., in combination with k-NN classifiers)
5. Approaches
   - centroid-based (e.g., k-means)
   - hierarchical/Agglomerative (e.g., linkage)

Clustering

1. An **un**supervised approach: no ground truth provided to the computer
2. What to do: grouping samples into clusters, such that (usually) samples sharing similar properties belong to the same cluster
3. Clustering usually is not used as an end-to-end solution to a problem (e.g., classification or regression), but a step in a pipeline of data analytics.
4. Applications:
   - Data partitioning (e.g., separate background with foreground in IP/CV)
   - Data compression (e.g., many samples are compressed to one sample the represents them)
   - Data visualization (e.g., showing that the features selected or engineered make sense, sklearn demo on Iris dataset)
   - Simple but usually explanable classification (e.g., in combination with k-NN classifiers)
5. Approaches
   - centroid-based (e.g., k-means)
   - hierarchical/Agglomerative (e.g., linkage)
   - density-based (e.g., DBScan)

Clustering

1. An **un**supervised approach: no ground truth provided to the computer
2. What to do: grouping samples into clusters, such that (usually) samples sharing similar properties belong to the same cluster
3. Clustering usually is not used as an end-to-end solution to a problem (e.g., classification or regression), but a step in a pipeline of data analytics.
4. Applications:
   - Data partitioning (e.g., separate background with foreground in IP/CV)
   - Data compression (e.g., many samples are compressed to one sample the represents them)
   - Data visualization (e.g., showing that the features selected or engineered make sense, sklearn demo on Iris dataset)
   - Simple but usually explanable classification (e.g., in combination with k-NN classifiers)
5. Approaches
   - centroid-based (e.g., k-means)
   - hierarchical/Agglomerative (e.g., linkage)
   - density-based (e.g., DBScan)
   - and others

k-means I

- It was originated from signal processing.

k-means I

- It was originated from signal processing.
- A cluster is a set of samples. All clusters are mutually exclusive, meaning that no more than one cluster can share a common sample simultaneously.

k-means I

- It was originated from signal processing.
- A cluster is a set of samples. All clusters are mutually exclusive, meaning that no more than one cluster can share a common sample simultaneously.
- k-means means k clusters, whose means are called **centroids**.

k-means I

- It was originated from signal processing.
- A cluster is a set of samples. All clusters are mutually exclusive, meaning that no more than one cluster can share a common sample simultaneously.
- k-means means k clusters, whose means are called **centroids**.
- Given a cluster s_i which is a set of samples, its centroid is simply the arithmetic mean of all samples in s_i, i.e.,

$$c_i = \frac{1}{|s_i|} \sum_{\mathbf{x} \in s_i} \mathbf{x}. \tag{1}$$

k-means I

- It was originated from signal processing.
- A cluster is a set of samples. All clusters are mutually exclusive, meaning that no more than one cluster can share a common sample simultaneously.
- k-means means k clusters, whose means are called **centroids**.
- Given a cluster s_i which is a set of samples, its centroid is simply the arithmetic mean of all samples in s_i, i.e.,

$$c_i = \frac{1}{|s_i|} \sum_{\mathbf{x} \in s_i} \mathbf{x}. \tag{1}$$

- The number k is given by the user, empirically or arbitrarily.

k-means I

- It was originated from signal processing.
- A cluster is a set of samples. All clusters are mutually exclusive, meaning that no more than one cluster can share a common sample simultaneously.
- k-means means k clusters, whose means are called **centroids**.
- Given a cluster s_i which is a set of samples, its centroid is simply the arithmetic mean of all samples in s_i, i.e.,

$$c_i = \frac{1}{|s_i|} \sum_{\mathbf{x} \in s_i} \mathbf{x}. \tag{1}$$

- The number k is given by the user, empirically or arbitrarily.
- The algorithm is not stable: it depends on seed centroids.

k-means I

- It was originated from signal processing.
- A cluster is a set of samples. All clusters are mutually exclusive, meaning that no more than one cluster can share a common sample simultaneously.
- k-means means k clusters, whose means are called **centroids**.
- Given a cluster s_i which is a set of samples, its centroid is simply the arithmetic mean of all samples in s_i, i.e.,

$$c_i = \frac{1}{|s_i|} \sum_{\mathbf{x} \in s_i} \mathbf{x}. \tag{1}$$

- The number k is given by the user, empirically or arbitrarily.
- The algorithm is not stable: it depends on seed centroids.
- It uses an iterative process.

k-means I

- It was originated from signal processing.
- A cluster is a set of samples. All clusters are mutually exclusive, meaning that no more than one cluster can share a common sample simultaneously.
- k-means means k clusters, whose means are called **centroids**.
- Given a cluster s_i which is a set of samples, its centroid is simply the arithmetic mean of all samples in s_i, i.e.,

$$c_i = \frac{1}{|s_i|} \sum_{\mathbf{x} \in s_i} \mathbf{x}. \tag{1}$$

- The number k is given by the user, empirically or arbitrarily.
- The algorithm is not stable: it depends on seed centroids.
- It uses an iterative process.
  a. Initially, k samples are randomly picked as the centroids of their perspective clusters.

k-means I

- It was originated from signal processing.
- A cluster is a set of samples. All clusters are mutually exclusive, meaning that no more than one cluster can share a common sample simultaneously.
- k-means means k clusters, whose means are called **centroids**.
- Given a cluster s_i which is a set of samples, its centroid is simply the arithmetic mean of all samples in s_i, i.e.,

$$c_i = \frac{1}{|s_i|} \sum_{\mathbf{x} \in s_i} \mathbf{x}. \tag{1}$$

- The number k is given by the user, empirically or arbitrarily.
- The algorithm is not stable: it depends on seed centroids.
- It uses an iterative process.
    a. Initially, k samples are randomly picked as the centroids of their perspective clusters.
    b. Assignment step: then scan every sample, and assigned it to the closest centroid.

k-means I

- It was originated from signal processing.
- A cluster is a set of samples. All clusters are mutually exclusive, meaning that no more than one cluster can share a common sample simultaneously.
- k-means means k clusters, whose means are called **centroids**.
- Given a cluster s_i which is a set of samples, its centroid is simply the arithmetic mean of all samples in s_i, i.e.,

$$c_i = \frac{1}{|s_i|} \sum_{\mathbf{x} \in s_i} \mathbf{x}. \tag{1}$$

- The number k is given by the user, empirically or arbitrarily.
- The algorithm is not stable: it depends on seed centroids.
- It uses an iterative process.
  a. Initially, k samples are randomly picked as the centroids of their perspective clusters.
  b. Assignment step: then scan every sample, and assigned it to the closest centroid.
  c. Update step: For every cluster, recompute its centroid using Eq. 1.

k-means I

- It was originated from signal processing.
- A cluster is a set of samples. All clusters are mutually exclusive, meaning that no more than one cluster can share a common sample simultaneously.
- k-means means k clusters, whose means are called **centroids**.
- Given a cluster s_i which is a set of samples, its centroid is simply the arithmetic mean of all samples in s_i, i.e.,

$$c_i = \frac{1}{|s_i|} \sum_{\mathbf{x} \in s_i} \mathbf{x}. \tag{1}$$

- The number k is given by the user, empirically or arbitrarily.
- The algorithm is not stable: it depends on seed centroids.
- It uses an iterative process.
    a. Initially, k samples are randomly picked as the centroids of their perspective clusters.
    b. Assignment step: then scan every sample, and assigned it to the closest centroid.
    c. Update step: For every cluster, recompute its centroid using Eq. 1.
- The algorithm stops after the centroids no longer changes or change is small enough.

k-means II

- k-means starts with seed centroids. Then iterate to shift the centroids. This is called a **run** consisting of many **iterations**.

k-means II

- k-means starts with seed centroids. Then iterate to shift the centroids. This is called a **run** consisting of many **iterations**.
- Because the initial centroid choice matters, usually the algorithm runs with different initial centroids and the one that minimizes a loss function is chosen at the end.

k-means II

- k-means starts with seed centroids. Then iterate to shift the centroids. This is called a **run** consisting of many **iterations**.
- Because the initial centroid choice matters, usually the algorithm runs with different initial centroids and the one that minimizes a loss function is chosen at the end.
- Assignment step. Given a sample $\mathbf{x}$, and a list of centroids C, the index of the cluster that $\mathbf{x}$ should be assigned to is

$$i = \arg\min_i d(\mathbf{x}, \mathbf{c}_i), \forall \mathbf{c}_i \in C$$

  where $d(\cdot)$ is the function to measure distance. In simplest form, it could be Euclidean distance where $d(\mathbf{x}, \mathbf{c}_i) = ||\mathbf{x} - \mathbf{c}_i||$.

k-means II

- k-means starts with seed centroids. Then iterate to shift the centroids. This is called a **run** consisting of many **iterations**.
- Because the initial centroid choice matters, usually the algorithm runs with different initial centroids and the one that minimizes a loss function is chosen at the end.
- Assignment step. Given a sample $\mathbf{x}$, and a list of centroids C, the index of the cluster that $\mathbf{x}$ should be assigned to is

$$i = \arg\min_i d(\mathbf{x}, \mathbf{c}_i), \forall \mathbf{c}_i \in C$$

  where $d(\cdot)$ is the function to measure distance. In simplest form, it could be Euclidean distance where $d(\mathbf{x}, \mathbf{c}_i) = ||\mathbf{x} - \mathbf{c}_i||$.
- In other words, a cluster $s_i = \{\mathbf{x} | d(\mathbf{x}, \mathbf{c}_i) \leq d(\mathbf{x}, \mathbf{c}_j), \forall j \in [1..k]\}$.

k-means II

- k-means starts with seed centroids. Then iterate to shift the centroids. This is called a **run** consisting of many **iterations**.
- Because the initial centroid choice matters, usually the algorithm runs with different initial centroids and the one that minimizes a loss function is chosen at the end.
- Assignment step. Given a sample $\mathbf{x}$, and a list of centroids C, the index of the cluster that $\mathbf{x}$ should be assigned to is
$$i = \arg\min_i d(\mathbf{x}, \mathbf{c}_i), \forall \mathbf{c}_i \in C$$
where $d(\cdot)$ is the function to measure distance. In simplest form, it could be Euclidean distance where $d(\mathbf{x}, \mathbf{c}_i) = ||\mathbf{x} - \mathbf{c}_i||$.
- In other words, a cluster $s_i = \{\mathbf{x}|d(\mathbf{x}, \mathbf{c}_i) \leq d(\mathbf{x}, \mathbf{c}_j), \forall j \in [1..k]\}$.
- When to stop the iterations? When the Frobenius norm (L2 norm on matrixes) of the difference in the cluster centers of two consecutive iterations falls below a pre-set tolerance, we declear the **convergence** and stop.

k-means II

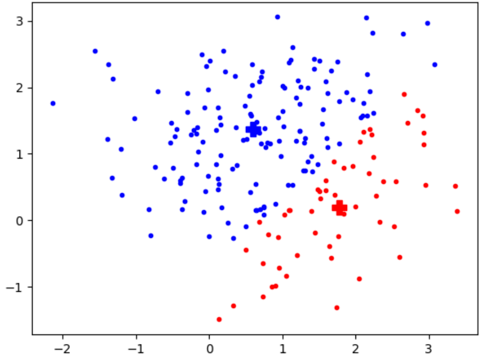
- k-means starts with seed centroids. Then iterate to shift the centroids. This is called a **run** consisting of many **iterations**.
- Because the initial centroid choice matters, usually the algorithm runs with different initial centroids and the one that minimizes a loss function is chosen at the end.
- Assignment step. Given a sample $\mathbf{x}$, and a list of centroids C, the index of the cluster that $\mathbf{x}$ should be assigned to is

$$i = \arg\min_i d(\mathbf{x}, \mathbf{c}_i), \forall \mathbf{c}_i \in C$$

  where $d(\cdot)$ is the function to measure distance. In simplest form, it could be Euclidean distance where $d(\mathbf{x}, \mathbf{c}_i) = ||\mathbf{x} - \mathbf{c}_i||$.
- In other words, a cluster $s_i = \{\mathbf{x} | d(\mathbf{x}, \mathbf{c}_i) \leq d(\mathbf{x}, \mathbf{c}_j), \forall j \in [1..k]\}$.
- When to stop the iterations? When the Frobenius norm (L2 norm on matrixes) of the difference in the cluster centers of two consecutive iterations falls below a pre-set tolerance, we declear the **convergence** and stop.
- The method just described above is called Naive k-means. There are many variants of it.

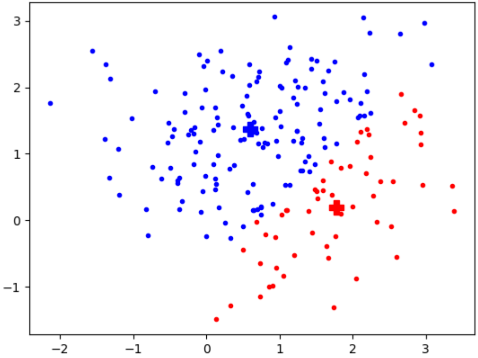
Demo

- Let's see a demo `kmeans.py`



Demo

- Let's see a demo `kmeans.py`
- A centroid (big crosses in the figure below) is NOT necessarily a sample, although it is initialized to a sample.



K-means III: Hyperparameters

- k

K-means III: Hyperparameters

- k
- the tolerance to declare convergence and to stop iteration

K-means III: Hyperparameters

- k
- the tolerance to declare convergence and to stop iteration
- number of runs

Single-linkage clustering I

- k-means has a major drawback: it ignores the gap between samples.

Single-linkage clustering I

- k-means has a major drawback: it ignores the gap between samples.
- If you have a spoke-like distribution of data, you may prefer samples that resemble dot lines to belong to the same cluster.

Single-linkage clustering I

- k-means has a major drawback: it ignores the gap between samples.
- If you have a spoke-like distribution of data, you may prefer samples that resemble dot lines to belong to the same cluster.
- Hierarchical/Agglomerative clustering is to solve this problem. Here we just discuss one example, single-linkage clustering.

Single-linkage clustering I

- k-means has a major drawback: it ignores the gap between samples.
- If you have a spoke-like distribution of data, you may prefer samples that resemble dot lines to belong to the same cluster.
- Hierarchical/Agglomerative clustering is to solve this problem. Here we just discuss one example, single-linkage clustering.
- Instead of working on the distance from a sample to the centroid of a cluster, it focuses on the distance from a sample to the closest sample in a cluster. In this way, it has a great power to "connect the dots".

Single-linkage clustering I

- k-means has a major drawback: it ignores the gap between samples.
- If you have a spoke-like distribution of data, you may prefer samples that resemble dot lines to belong to the same cluster.
- Hierarchical/Agglomerative clustering is to solve this problem. Here we just discuss one example, single-linkage clustering.
- Instead of working on the distance from a sample to the centroid of a cluster, it focuses on the distance from a sample to the closest sample in a cluster. In this way, it has a great power to "connect the dots".
- It works bottom up. Initially, every sample is a cluster.

Single-linkage clustering I

- k-means has a major drawback: it ignores the gap between samples.
- If you have a spoke-like distribution of data, you may prefer samples that resemble dot lines to belong to the same cluster.
- Hierarchical/Agglomerative clustering is to solve this problem. Here we just discuss one example, single-linkage clustering.
- Instead of working on the distance from a sample to the centroid of a cluster, it focuses on the distance from a sample to the closest sample in a cluster. In this way, it has a great power to "connect the dots".
- It works bottom up. Initially, every sample is a cluster.
- Then it enters into an iterative process. It evaluates the distances for all pairs of clusters. It then merges the closest pair.

Single-linkage clustering I

- k-means has a major drawback: it ignores the gap between samples.
- If you have a spoke-like distribution of data, you may prefer samples that resemble dot lines to belong to the same cluster.
- Hierarchical/Agglomerative clustering is to solve this problem. Here we just discuss one example, single-linkage clustering.
- Instead of working on the distance from a sample to the centroid of a cluster, it focuses on the distance from a sample to the closest sample in a cluster. In this way, it has a great power to "connect the dots".
- It works bottom up. Initially, every sample is a cluster.
- Then it enters into an iterative process. It evaluates the distances for all pairs of clusters. It then merges the closest pair.
- The iterative process stops when there is only one cluster left, the entire dataset.

Single-linkage clustering I

- k-means has a major drawback: it ignores the gap between samples.
- If you have a spoke-like distribution of data, you may prefer samples that resemble dot lines to belong to the same cluster.
- Hierarchical/Agglomerative clustering is to solve this problem. Here we just discuss one example, single-linkage clustering.
- Instead of working on the distance from a sample to the centroid of a cluster, it focuses on the distance from a sample to the closest sample in a cluster. In this way, it has a great power to "connect the dots".
- It works bottom up. Initially, every sample is a cluster.
- Then it enters into an iterative process. It evaluates the distances for all pairs of clusters. It then merges the closest pair.
- The iterative process stops when there is only one cluster left, the entire dataset.
- Note that the inter-cluster distance is re-evaluated at each iteration.

Single-linkage clustering II

- Hold on. One cluster in the end? What if I want 2, 3, . . . , clusters?

Single-linkage clustering II

- Hold on. One cluster in the end? What if I want 2, 3, . . . , clusters?
- The iterative process can be represented as a binary tree, called **dendrogram**.

Single-linkage clustering II

- Hold on. One cluster in the end? What if I want 2, 3, . . . , clusters?
- The iterative process can be represented as a binary tree, called **dendrogram**.
- Each tree node is associated with a value. The leaf nodes are the samples. Two nodes have the same parent if they merge into one cluster. The value for the parent node is the minimal distance between the child cluster.

Single-linkage clustering II

- Hold on. One cluster in the end? What if I want 2, 3, . . . , clusters?
- The iterative process can be represented as a binary tree, called **dendrogram**.
- Each tree node is associated with a value. The leaf nodes are the samples. Two nodes have the same parent if they merge into one cluster. The value for the parent node is the minimal distance between the child cluster.
- To get k clusters, just find a threshold such that removing all nodes on the dendrogram whose values are greater than the threshold will result in k trees. Each of the remaining trees is a cluster.

Single-linkage clustering II

- Hold on. One cluster in the end? What if I want 2, 3, . . . , clusters?
- The iterative process can be represented as a binary tree, called **dendrogram**.
- Each tree node is associated with a value. The leaf nodes are the samples. Two nodes have the same parent if they merge into one cluster. The value for the parent node is the minimal distance between the child cluster.
- To get k clusters, just find a threshold such that removing all nodes on the dendrogram whose values are greater than the threshold will result in k trees. Each of the remaining trees is a cluster.
- Single-linkage clustering has a property: "the rich get richer."

Single-linkage clustering II

- Hold on. One cluster in the end? What if I want 2, 3, . . . , clusters?
- The iterative process can be represented as a binary tree, called **dendrogram**.
- Each tree node is associated with a value. The leaf nodes are the samples. Two nodes have the same parent if they merge into one cluster. The value for the parent node is the minimal distance between the child cluster.
- To get k clusters, just find a threshold such that removing all nodes on the dendrogram whose values are greater than the threshold will result in k trees. Each of the remaining trees is a cluster.
- Single-linkage clustering has a property: "the rich get richer."
- Let's go over the example on Wikipedia.
  https://en.wikipedia.org/wiki/Single-linkage_clustering#Working_example

Single-linkage clustering III

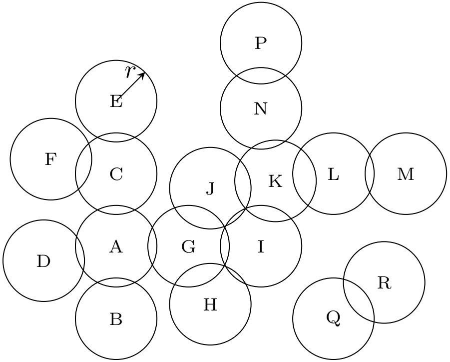
Algorithm 1: Single-linkage clustering

```
Initialize the distance matrix/dictionary D, and the distance dictionary δ ;
while D has more than one row or column do
    i, j ← arg min_{i,j}(D);
    d = min({D(x, y)|x ∈ i, y ∈ j});
    δ(i, j) = d/2;
    Remove columns and rows for i and j from D;
    Create a new column and new row for a new cluster k = i ∪ j;
    Populate the column and row for k in D;
return δ
```

DBSCAN: a density-based clustering algorithm I

- Recall the Subway map earlier. A cluster has a higher density of samples than other parts.

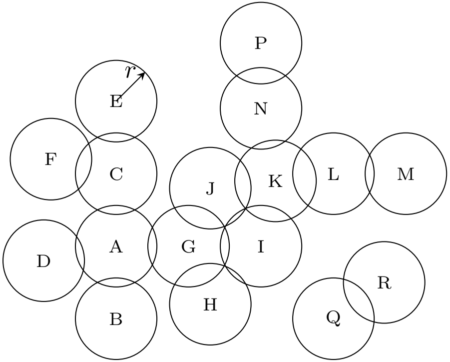
An example. Two samples are neighbors if their distance $< \epsilon$. For each sample, we draw a circle of radius $r = \epsilon/2$.



DBSCAN: a density-based clustering algorithm I

- Recall the Subway map earlier. A cluster has a higher density of samples than other parts.
- Density means enough similar samples over an area.

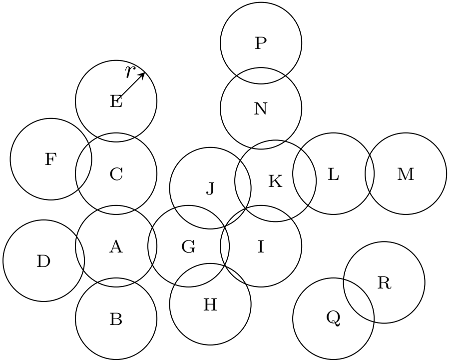
An example. Two samples are neighbors if their distance $< \epsilon$. For each sample, we draw a circle of radius $r = \epsilon/2$.



DBSCAN: a density-based clustering algorithm I

- Recall the Subway map earlier. A cluster has a higher density of samples than other parts.
- Density means enough similar samples over an area.
- How to measure density: Given a sample, count the number of samples similar enough to it.

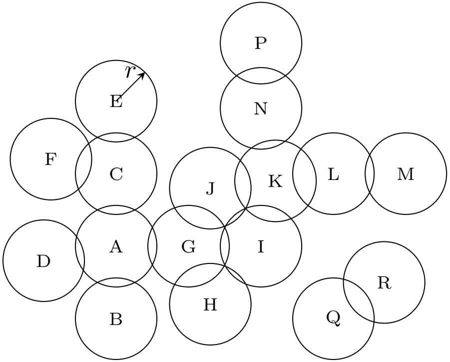
An example. Two samples are neighbors if their distance $< \epsilon$. For each sample, we draw a circle of radius $r = \epsilon/2$.



DBSCAN: a density-based clustering algorithm I

- Recall the Subway map earlier. A cluster has a higher density of samples than other parts.
- Density means enough similar samples over an area.
- How to measure density: Given a sample, count the number of samples similar enough to it.
- How to measure "similar enough": distance (e.g., Euclidean) between samples below a threshold ϵ.

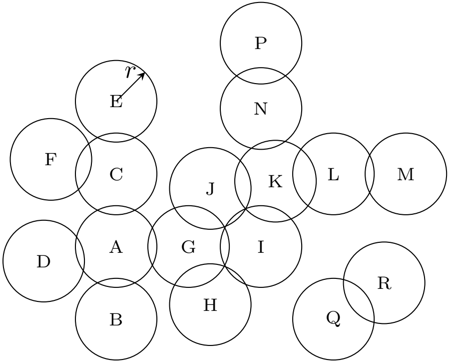
An example. Two samples are neighbors if their distance $< \epsilon$. For each sample, we draw a circle of radius $r = \epsilon/2$.



DBSCAN: a density-based clustering algorithm I

- Recall the Subway map earlier. A cluster has a higher density of samples than other parts.
- Density means enough similar samples over an area.
- How to measure density: Given a sample, count the number of samples similar enough to it.
- How to measure "similar enough": distance (e.g., Euclidean) between samples below a threshold ϵ.
- In DBSCAN, samples similar enough to a sample are called the **neighbors** of the sample.

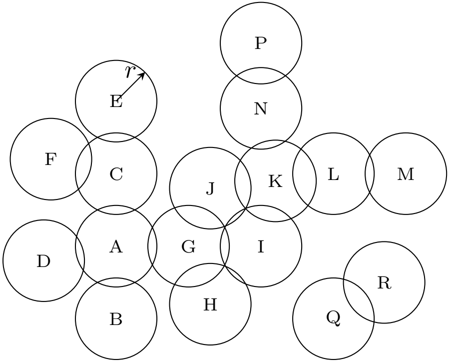
An example. Two samples are neighbors if their distance $< \epsilon$. For each sample, we draw a circle of radius $r = \epsilon/2$.



DBSCAN: a density-based clustering algorithm I

- Recall the Subway map earlier. A cluster has a higher density of samples than other parts.
- Density means enough similar samples over an area.
- How to measure density: Given a sample, count the number of samples similar enough to it.
- How to measure "similar enough": distance (e.g., Euclidean) between samples below a threshold ϵ.
- In DBSCAN, samples similar enough to a sample are called the **neighbors** of the sample.
- Idea 1: Neighbors of samples already in a cluster should be included in the cluster as well.

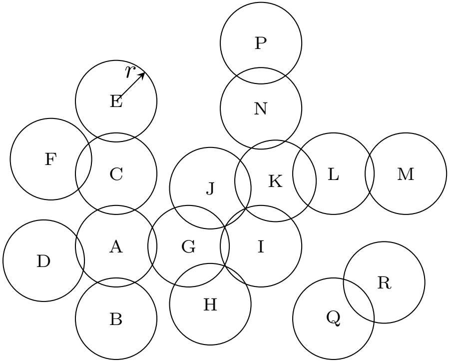
An example. Two samples are neighbors if their distance $< \epsilon$. For each sample, we draw a circle of radius $r = \epsilon/2$.



DBSCAN: a density-based clustering algorithm I

- Recall the Subway map earlier. A cluster has a higher density of samples than other parts.
- Density means enough similar samples over an area.
- How to measure density: Given a sample, count the number of samples similar enough to it.
- How to measure "similar enough": distance (e.g., Euclidean) between samples below a threshold ϵ.
- In DBSCAN, samples similar enough to a sample are called the **neighbors** of the sample.
- Idea 1: Neighbors of samples already in a cluster should be included in the cluster as well.
- Idea 2: But the expansion stops at a neighbor if it doesn't have enough neighbors – density not high enough.

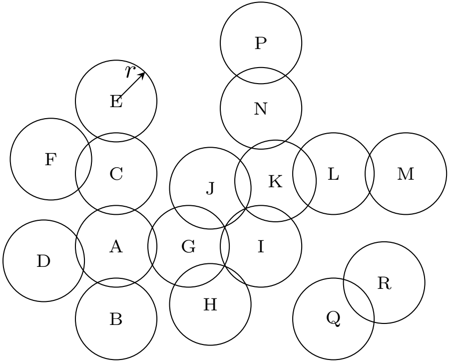
An example. Two samples are neighbors if their distance $< \epsilon$. For each sample, we draw a circle of radius $r = \epsilon/2$.



DBSCAN II: an example

1. Let $T = 2$. Denote a set S to hold cluster members to be examined. Initially $S = \{A\}$.

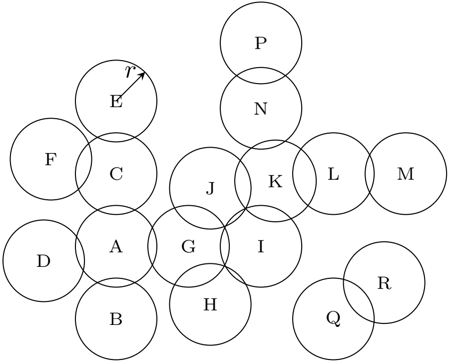
Two samples are neighbors if their distance $< \epsilon$. For each sample, we draw a circle of radius $r = \epsilon/2$.



DBSCAN II: an example

1. Let $T = 2$. Denote a set S to hold cluster members to be examined. Initially $S = \{A\}$.
2. Since A has more than T neighbors, create a cluster with A. And add A's neighbors, B, C, D, G into the cluster. Let $S = \{B, C, D, G\}$.

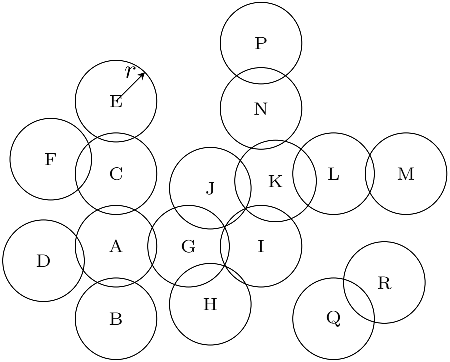
Two samples are neighbors if their distance $< \epsilon$. For each sample, we draw a circle of radius $r = \epsilon/2$.



DBSCAN II: an example

1. Let $T = 2$. Denote a set S to hold cluster members to be examined. Initially $S = \{A\}$.
2. Since A has more than T neighbors, create a cluster with A. And add A's neighbors, B, C, D, G into the cluster. Let $S = \{B, C, D, G\}$.
3. Check whether each element of S has more than T neighbors. C, G pass the test, so add G's neighbor J, I, H (A already in the cluster) and C's neighbor E, F into the cluster. Let $S = \{E, F, J, I, H\}$.

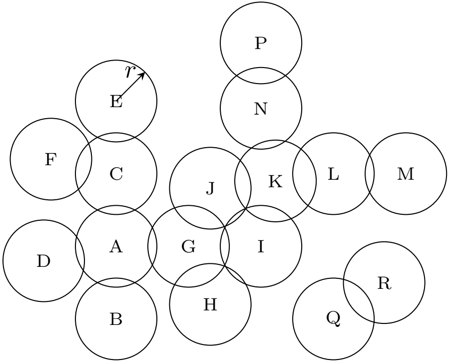
Two samples are neighbors if their distance $< \epsilon$. For each sample, we draw a circle of radius $r = \epsilon/2$.



DBSCAN II: an example

1. Let $T = 2$. Denote a set S to hold cluster members to be examined. Initially $S = \{A\}$.
2. Since A has more than T neighbors, create a cluster with A. And add A's neighbors, B, C, D, G into the cluster. Let $S = \{B, C, D, G\}$.
3. Check whether each element of S has more than T neighbors. C, G pass the test, so add G's neighbor J, I, H (A already in the cluster) and C's neighbor E, F into the cluster. Let $S = \{E, F, J, I, H\}$.
4. Check whether each element of S has more than T neighbors. I, J pass the test, so add J's neighbor K (G, I already in the cluster) and I's neighbor K (G, H, J already in the cluster) into the cluster. Let $S = \{K\}$.

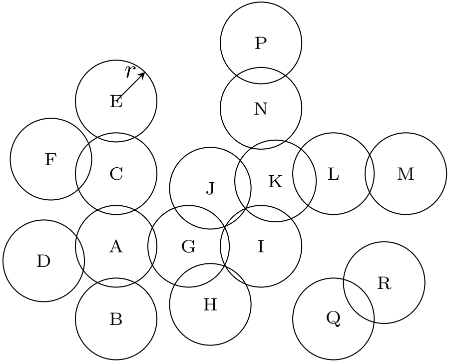
Two samples are neighbors if their distance $< \epsilon$. For each sample, we draw a circle of radius $r = \epsilon/2$.



DBSCAN II: an example

1. Let $T = 2$. Denote a set S to hold cluster members to be examined. Initially $S = \{A\}$.
2. Since A has more than T neighbors, create a cluster with A. And add A's neighbors, B, C, D, G into the cluster. Let $S = \{B, C, D, G\}$.
3. Check whether each element of S has more than T neighbors. C, G pass the test, so add G's neighbor J, I, H (A already in the cluster) and C's neighbor E, F into the cluster. Let $S = \{E, F, J, I, H\}$.
4. Check whether each element of S has more than T neighbors. I, J pass the test, so add J's neighbor K (G, I already in the cluster) and I's neighbor K (G, H, J already in the cluster) into the cluster. Let $S = \{K\}$.
5. Check whether each element of S has more than T neighbors. K passes the test, so add K's neighbor N, L (J, I already in the cluster) into the cluster. Let $S = \{N, L\}$.

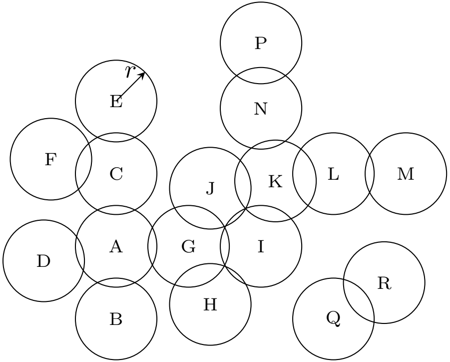
Two samples are neighbors if their distance $< \epsilon$. For each sample, we draw a circle of radius $r = \epsilon/2$.



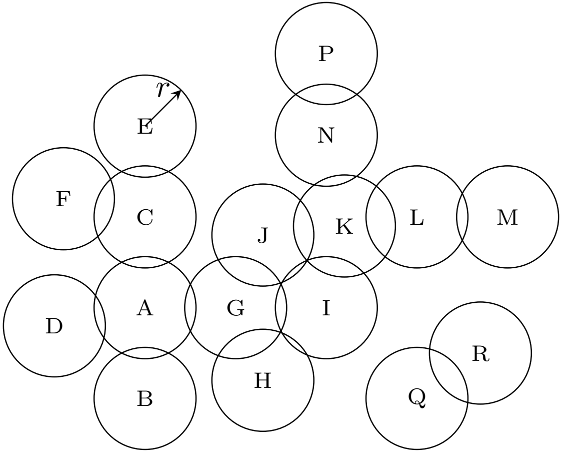
DBSCAN II: an example

1. Let $T = 2$. Denote a set S to hold cluster members to be examined. Initially $S = \{A\}$.
2. Since A has more than T neighbors, create a cluster with A. And add A's neighbors, B, C, D, G into the cluster. Let $S = \{B, C, D, G\}$.
3. Check whether each element of S has more than T neighbors. C, G pass the test, so add G's neighbor J, I, H (A already in the cluster) and C's neighbor E, F into the cluster. Let $S = \{E, F, J, I, H\}$.
4. Check whether each element of S has more than T neighbors. I, J pass the test, so add J's neighbor K (G, I already in the cluster) and I's neighbor K (G, H, J already in the cluster) into the cluster. Let $S = \{K\}$.
5. Check whether each element of S has more than T neighbors. K passes the test, so add K's neighbor N, L (J, I already in the cluster) into the cluster. Let $S = \{N, L\}$.
6. Check whether each element of S has more than T neighbors. N, L both fail the test. No more expansions.

Two samples are neighbors if their distance $< \epsilon$. For each sample, we draw a circle of radius $r = \epsilon/2$.

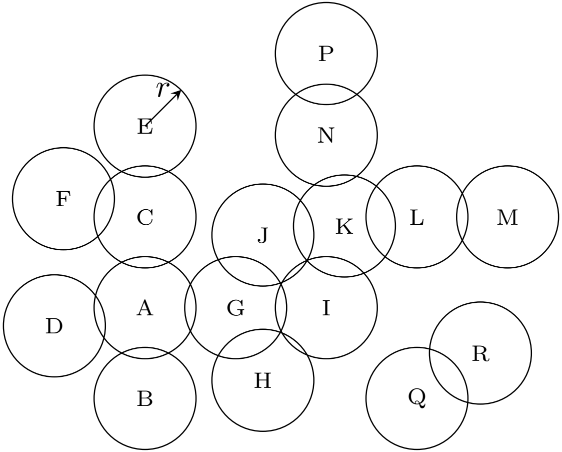


DBSCAN III : an example



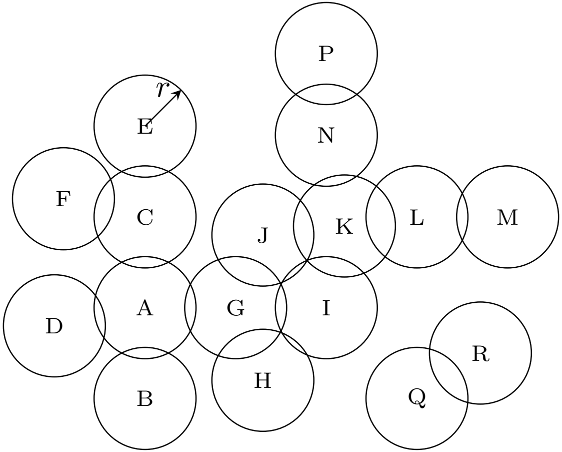
- What if we start from Q?

DBSCAN III : an example



- What if we start from Q?
- Because it has less than $T = 2$ neighbors, no cluster will be created from it.

DBSCAN III : an example



- What if we start from Q?
- Because it has less than $T = 2$ neighbors, no cluster will be created from it.
- Unlike in k-means or hierarchical clustering, not every sample becomes part of a cluster.

DBSCAN IV

Algorithm 2: Slightly varied DBSCAN

Data: X: samples, T: a threshold

Initialize cluster index $i \leftarrow 1$;
foreach *sample* $x \in X$ **do**
    if x *are NOT assigned to a cluster* **then**
        $N(x) \leftarrow$ neighbors of x ;
        if $|N(x)| > T$ **then**
            Assign x to cluster i;
            Seed set of cluster i: $S \leftarrow N(x)$;
            while $S \neq \emptyset$ **do**
                $y \leftarrow$ one element of S;
                Assign y to cluster i;
                Remove y from S;
                if $|N(y)| > T$ **then**
                    $S \leftarrow S \cup N(y)$;
        i++;

- $N(\cdot)$: a function to compute a sample's neighbors

DBSCAN IV

Algorithm 3: Slightly varied DBSCAN

Data: X: samples, T: a threshold

Initialize cluster index $i \leftarrow 1$;

foreach *sample* $x \in X$ **do**

- **if** x *are NOT assigned to a cluster* **then**
  - $N(x) \leftarrow$ neighbors of x ;
  - **if** $|N(x)| > T$ **then**
    - Assign x to cluster i;
    - Seed set of cluster i: $S \leftarrow N(x)$;
    - **while** $S \neq \emptyset$ **do**
      - $y \leftarrow$ one element of S;
      - Assign y to cluster i;
      - Remove y from S;
      - **if** $|N(y)| > T$ **then**
        - $S \leftarrow S \cup N(y)$;
  - i++;

- $N(\cdot)$: a function to compute a sample's neighbors
- Neighbors to a sample are those in close proximity, e.g., in Euclidean distance.

DBSCAN IV

Algorithm 4: Slightly varied DBSCAN

Data: X: samples, T: a threshold

Initialize cluster index $i \leftarrow 1$;
foreach *sample* $x \in X$ **do**
    if x *are NOT assigned to a cluster* **then**
        $N(x) \leftarrow$ neighbors of x ;
        if $|N(x)| > T$ **then**
            Assign x to cluster i;
            Seed set of cluster i: $S \leftarrow N(x)$;
            while $S \neq \emptyset$ **do**
                $y \leftarrow$ one element of S;
                Assign y to cluster i;
                Remove y from S;
                if $|N(y)| > T$ **then**
                    $S \leftarrow S \cup N(y)$;
        i++;

- $N(\cdot)$: a function to compute a sample's neighbors
- Neighbors to a sample are those in close proximity, e.g., in Euclidean distance.
- The While-loop is basically a BFS or a DFS, which exhaustively expands a cluster from a sample x.

DBSCAN IV

Algorithm 5: Slightly varied DBSCAN

Data: X: samples, T: a threshold

Initialize cluster index $i \leftarrow 1$;
foreach *sample* $x \in X$ **do**
  if x *are NOT assigned to a cluster* **then**
    $N(x) \leftarrow$ neighbors of x ;
    if $|N(x)| > T$ **then**
      Assign x to cluster i;
      Seed set of cluster i: $S \leftarrow N(x)$;
      while $S \neq \emptyset$ **do**
        $y \leftarrow$ one element of S;
        Assign y to cluster i;
        Remove y from S;
        if $|N(y)| > T$ **then**
          $S \leftarrow S \cup N(y)$;
    i++;

- $N(\cdot)$: a function to compute a sample's neighbors
- Neighbors to a sample are those in close proximity, e.g., in Euclidean distance.
- The While-loop is basically a BFS or a DFS, which exhaustively expands a cluster from a sample x.
- The expansion stops when all samples in the cluster either have all their neighbors visited, or few enough ($< T$) neighbors.

DBSCAN IV

Algorithm 6: Slightly varied DBSCAN

Data: X: samples, T: a threshold

Initialize cluster index $i \leftarrow 1$;
foreach *sample* $x \in X$ **do**
    if x *are NOT assigned to a cluster* **then**
        $N(x) \leftarrow$ neighbors of x ;
        if $|N(x)| > T$ **then**
            Assign x to cluster i;
            Seed set of cluster i: $S \leftarrow N(x)$;
            while $S \neq \emptyset$ **do**
                $y \leftarrow$ one element of S;
                Assign y to cluster i;
                Remove y from S;
                if $|N(y)| > T$ **then**
                    $S \leftarrow S \cup N(y)$;
        i++;

- $N(\cdot)$: a function to compute a sample's neighbors
- Neighbors to a sample are those in close proximity, e.g., in Euclidean distance.
- The While-loop is basically a BFS or a DFS, which exhaustively expands a cluster from a sample x.
- The expansion stops when all samples in the cluster either have all their neighbors visited, or few enough ($< T$) neighbors.
- Cluster members with enough neighbors are called **core points** (such as A, G, J, I in the example earlier) while those without enough neighbors are called **non-core points** (such as E, F, D, B, H, N, L).

DBSCAN IV

Algorithm 7: Slightly varied DBSCAN

Data: X: samples, T: a threshold

Initialize cluster index $i \leftarrow 1$;
foreach *sample* $x \in X$ **do**
    if x *are NOT assigned to a cluster* **then**
        $N(x) \leftarrow$ neighbors of x ;
        if $|N(x)| > T$ **then**
            Assign x to cluster i;
            Seed set of cluster i: $S \leftarrow N(x)$;
            while $S \neq \emptyset$ **do**
                $y \leftarrow$ one element of S;
                Assign y to cluster i;
                Remove y from S;
                if $|N(y)| > T$ **then**
                    $S \leftarrow S \cup N(y)$;
        i++;

- $N(\cdot)$: a function to compute a sample's neighbors
- Neighbors to a sample are those in close proximity, e.g., in Euclidean distance.
- The While-loop is basically a BFS or a DFS, which exhaustively expands a cluster from a sample x.
- The expansion stops when all samples in the cluster either have all their neighbors visited, or few enough ($< T$) neighbors.
- Cluster members with enough neighbors are called **core points** (such as A, G, J, I in the example earlier) while those without enough neighbors are called **non-core points** (such as E, F, D, B, H, N, L).
- Originally published in 1996. Won 2014 SIGKDD Time of Test Award. A follow-up paper won 2015 SIGMOD Best Paper Award.

DBSCAN IV

Algorithm 8: Slightly varied DBSCAN

Data: X: samples, T: a threshold

Initialize cluster index $i \leftarrow 1$;
foreach *sample* $x \in X$ **do**
    if x *are NOT assigned to a cluster* **then**
        $N(x) \leftarrow$ neighbors of x ;
        if $|N(x)| > T$ **then**
            Assign x to cluster i;
            Seed set of cluster i: $S \leftarrow N(x)$;
            while $S \neq \emptyset$ **do**
                $y \leftarrow$ one element of S;
                Assign y to cluster i;
                Remove y from S;
                if $|N(y)| > T$ **then**
                    $S \leftarrow S \cup N(y)$;
        i++;

- $N(\cdot)$: a function to compute a sample's neighbors
- Neighbors to a sample are those in close proximity, e.g., in Euclidean distance.
- The While-loop is basically a BFS or a DFS, which exhaustively expands a cluster from a sample x.
- The expansion stops when all samples in the cluster either have all their neighbors visited, or few enough ($< T$) neighbors.
- Cluster members with enough neighbors are called **core points** (such as A, G, J, I in the example earlier) while those without enough neighbors are called **non-core points** (such as E, F, D, B, H, N, L).
- Originally published in 1996. Won 2014 SIGKDD Time of Test Award. A follow-up paper won 2015 SIGMOD Best Paper Award.
- Why you should still use DBSCAN, ACM Trans. on Database Systems, 42(3):21, 2017

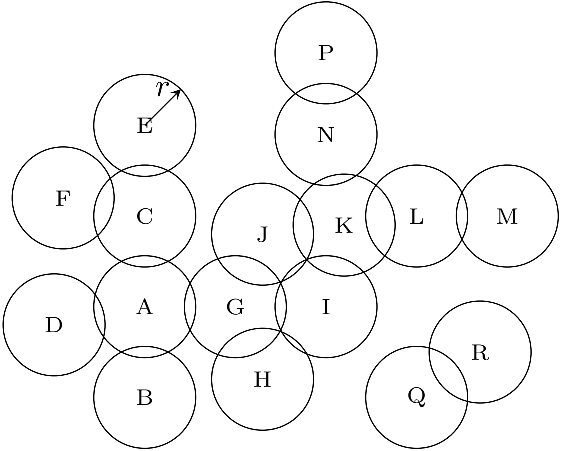
DBSCAN V

Algorithm 9: Slightly varied DBSCAN

Data: X: samples, T: a threshold

Initialize cluster index $i \leftarrow 1$;
foreach *sample* $x \in X$ **do**
    if x *are NOT assigned to a cluster* **then**
        $N(x) \leftarrow$ neighbors of x ;
        if $|N(x)| > T$ **then**
            Assign x to cluster i;
            Seed set of cluster i: $S \leftarrow N(x)$;
            while $S \neq \emptyset$ **do**
                $y \leftarrow$ one element of S;
                Assign y to cluster i;
                Remove y from S;
                if $|N(y)| > T$ **then**
                    $S \leftarrow S \cup N(y)$;
        i++;

An example (radii of circles are half of T)



Other clustering approaches

- Mean-shift

Reading materials

- https://scikit-learn.org/stable/modules/clustering.html