Clustering
Clustering

Goal: given a set of $n$ “points” we should group the points in some sensible manner.

What are some possible sets of points?
• Webpages, images, genome fragments, people, etc.

For anyone interested in machine learning, clustering is a relative of unsupervised learning.
Clustering

Assumptions:
1. We are given a similarity (or dissimilarity) value for all points
2. Similarities are symmetric

\[ d(p, q) \] is the similarity between points \( p \) and \( q \)
And \( d(p, q) = d(q, p) \)

Examples include Euclidean distance and edit distance
Goal: cluster "nearby" points
Goal: cluster "nearby" points
Goal: cluster "nearby" points
Clustering Topics/Algorithms

- Related to data mining, statistical data analysis, machine learning, pattern recognition, image analysis, information retrieval, bioinformatics, data compression, and computer graphics.

- Hierarchical clustering
- Centroid clustering (*k-means*)
- Distribution Clustering
- Density Clustering
Max-Spacing K-Clustering

• We assume that we know a good value for $k$, where $k$ is the number of clusters that we are going to form.
• $k$ is not discovered completely automatically (pick a few values and try them out).

• Two $p$ and $q$ points are separated if they are in different clusters.
• Thus, points that are similar should not be separated.
• Spacing for a set of $k$-clusters is given by:

$$S = \min_{\text{for all separated } p,q} d(p, q)$$

• Given the above definition, do you think it is better to have a small or large $S$?
Max-Spacing K-Clustering

• **Problem statement**: given a distance measure \(d\) and a number of clusters \(k\), compute the \(k\)-clustering with a maximum spacing \(S\).

• Let’s solve this problem with a greedy approach.

• **Greedy algorithm setup**:
  • We will not care about the number of clusters we produce until the end
  • We will start by putting every point into its own cluster
  • What is our greedy choice?
  • How do we make spacing bigger each iteration?
Max-Spacing K-Clustering

Put each point into its own cluster

Repeat until we have only $k$ clusters

\[
\text{let } p, q = \text{closest pair of separated points}
\]

This is the operation that determines spacing

merge the clusters containing $p$ and $q$
Max-Spancing K-Clustering

Put each point into its own cluster

Repeat until we have only k clusters
  p, q = closest pair of separated points
  merge the clusters containing p and q
Max-Spancing K-Clustering

Put each point into its own cluster

Repeat until we have only $k$ clusters
  \[ p, q = \text{closest pair of separated points} \]
  merge the clusters containing $p$ and $q$
Max-Spacing K-Clustering

Put each point into its own cluster

Repeat until we have only k clusters
  p, q = closest pair of separated points
  merge the clusters containing p and q

k = 3
Max-Spacing K-Clustering

Put each point into its own cluster

Repeat until we have only k clusters
   p, q = closest pair of separated points
   merge the clusters containing p and q

$k = 3$
Max-Spacing K-Clustering

Put each point into its own cluster

Repeat until we have only k clusters

\[ p, q = \text{closest pair of separated points} \]

merge the clusters containing \( p \) and \( q \)
Max-Spacing K-Clustering

Put each point into its own cluster

Repeat until we have only k clusters
   p, q = closest pair of separated points
   merge the clusters containing p and q

k = 3
Max-Spacing K-Clustering

Put each point into its own cluster

Repeat until we have only k clusters
  \( p, q = \text{closest pair of separated points} \)
  merge the clusters containing \( p \) and \( q \)

\( k = 3 \)
Max-Spaning K-Clustering

Put each point into its own cluster

Repeat until we have only k clusters
  p, q = closest pair of separated points
  merge the clusters containing p and q

k = 3
Does this algorithm look familiar?

• This procedure is nearly identical to Kruskal’s Algorithm for MST
**Kruskals**

Sort $E$ by edge cost

$T = \text{empty}$

Each vertex into disjoint set

Repeat until only 1 set:

$u, v = \text{next cheapest edge}$

if $\text{Find}(u) = \text{Find}(v)$

merge sets

**Max-Spacing k-Clustering**

Each point into own cluster

Repeat until only $k$ clusters:

$p, q = \text{next closest points}$

if $p$ and $q$ are separated

merge clusters
Does this algorithm look familiar?

• This procedure is nearly identical to Kruskal’s Algorithm for MST

• What are the vertices?
• What are the edge costs?
• How many edges are there?
  • This gives us a “complete” graph.

• Using Kruskal’s algorithm for cluster is called single link clustering.
Proof

• **Theorem**: single-link clustering finds the max-spacing $k$-clustering of a set of points.
• Although we are using Kruskal’s algorithm, the objective has changed.
• So, we cannot use the proof from before.

• Let $C_1, \ldots, C_k$ be the $k$ clusters computed by the greedy algorithm
• Let $S$ be the spacing of these $k$ clusters
• Let $C_1', \ldots, C_k'$ be any other $k$ clusters, with spacing $S'$

• To prove our theorem, we need to show that $S' \leq S$
Proof of Single-Link Clustering

• Note: it would be bad to find a case where $S' > S$

• **Case 1 (edge case):** C1’, ..., Ck’ are just a renaming C1, ..., Ck

  • In which case, $S' = S$ and we are done with this case

• **Case 2:** We can find a pair of points a and b such that:
  • a and b are in the same greedy cluster Ci
  • a and b are in different clusters Ca’, Cb’
Proof of Single-Link Clustering

• Case 2a: points a and b are \textbf{directly} merged at some point
• How does $d(a, b)$ relate to $S$?
Max-Spacing K-Clustering

Put each point into its own cluster

Repeat until we have only $k$ clusters
   $p, q =$ closest pair of separated points
   merge the clusters containing $p$ and $q$

$$S = \min_{\text{for all separated } p, q} d(p, q)$$
Max-Spacing K-Clustering

Put each point into its own cluster

Repeat until we have only \( k \) clusters
  \( p, q = \) closest pair of separated points
  merge the clusters containing \( p \) and \( q \)

\[
S = \min_{\text{for all separated } p,q} d(p,q)
\]
Max-Spacing K-Clustering

Put each point into its own cluster

Repeat until we have only \( k \) clusters

\( p, q = \) closest pair of separated points
merge the clusters containing \( p \) and \( q \)

\[
S = \min_{for \ all \ separated \ p, q} d(p, q)
\]
Max-Spacing K-Clustering

Put each point into its own cluster

Repeat until we have only $k$ clusters

$p, q =$ closest pair of separated points
merge the clusters containing $p$ and $q$

$$S = \min_{\text{for all separated } p, q} d(p, q)$$
Max-Spacing K-Clustering

Put each point into its own cluster

Repeat until we have only k clusters
p, q = closest pair of separated points
merge the clusters containing p and q

$$S = \min_{\text{for all separated } p,q} d(p, q)$$
Proof of Single-Link Clustering

• **Case 2a**: points $a$ and $b$ are **directly** merged at some point during the greedy algorithm

• How does $d(a, b)$ relate to $S$?

• If two points $a$ and $b$ are directly merged, then $S \geq d(a, b)$

• Additionally, the distance between any two merged points only goes up (or stays the same) after each iteration
Proof of Single-Link Clustering

• Case 2a: points $a$ and $b$ are directly merged at some point during the greedy algorithm

• How does $d(a, b)$ relate to $S$?

• If two points $a$ and $b$ are directly merged, then $S \geq d(a, b)$

• Additionally, the distance between any two merged points only goes up (or stays the same) after each iteration

• So we have that $S' \leq d(a, b) \leq S \implies S' \leq S$

Case 2: We can find a pair of points $a$ and $b$ such that:
- $a$ and $b$ are in the same greedy cluster $C_i$
- $a$ and $b$ are in different clusters $C_a'$, $C_b'$

To prove our theorem, we need to show that $S' \leq S$
Proof of Single-Link Clustering

- **Case 2b**: points $a$ and $b$ are *indirectly* merged at some point during the greedy algorithm.
- How does $d(a, b)$ relate to $S$?
- Lines denote direct merges.
- All points are in the same cluster in the end.
Proof of Single-Link Clustering

- **Case 2b**: points a and b are *indirectly* merged at some point during the greedy algorithm.

- Let $\langle a, a_1, \ldots, a_L, b \rangle$ be the path of direct merges connecting a and b.

- Since a is in $C_a'$ and b is in $C_b'$ there must be some consecutive pair where $a_j$ is in $C_a'$ and $a_{j+1}$ is in $C_b'$.

- Thus $S' \leq d(a_j, a_{j+1}) \leq S \Rightarrow S' \leq S$.
Proof of Single-Link Clustering

- So, we have proved that under all circumstances, $S$ is the biggest possible spacing for the points.
- Thus, the greedy (Kruskal’s-based) algorithm is optimal and correct.