Lecture 20: Clustering

Tim LaRock

larock.t@northeastern.edu

bit.ly/cs3000syllabus

Business

Extra Credit Assignments 1 & 2 are open

Midterm grades on track to go out tomorrow night

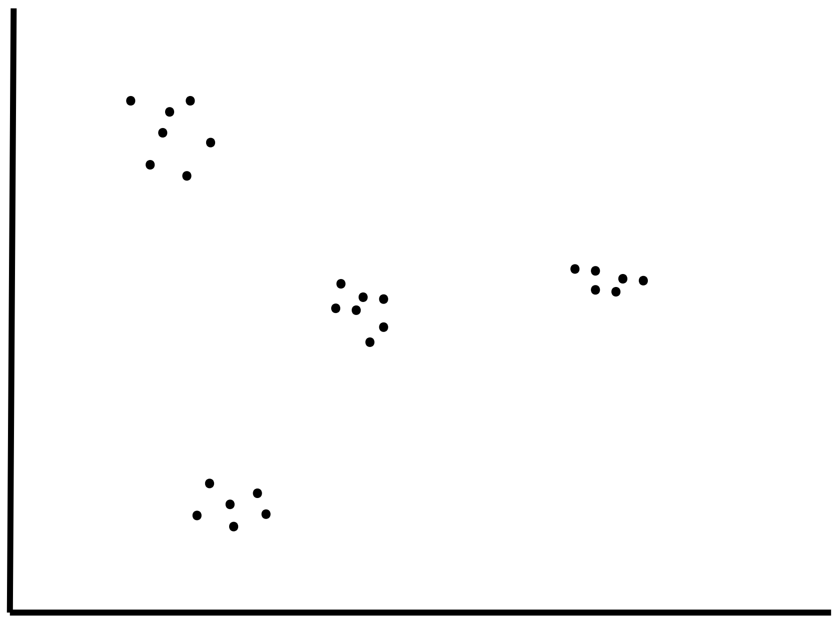
Final exam review questions form sent out last night

This Week

- Today: Greedy algorithm for clustering
- Tomorrow: Advanced topics and course wrap-up
- Thursday: Final Exam Review

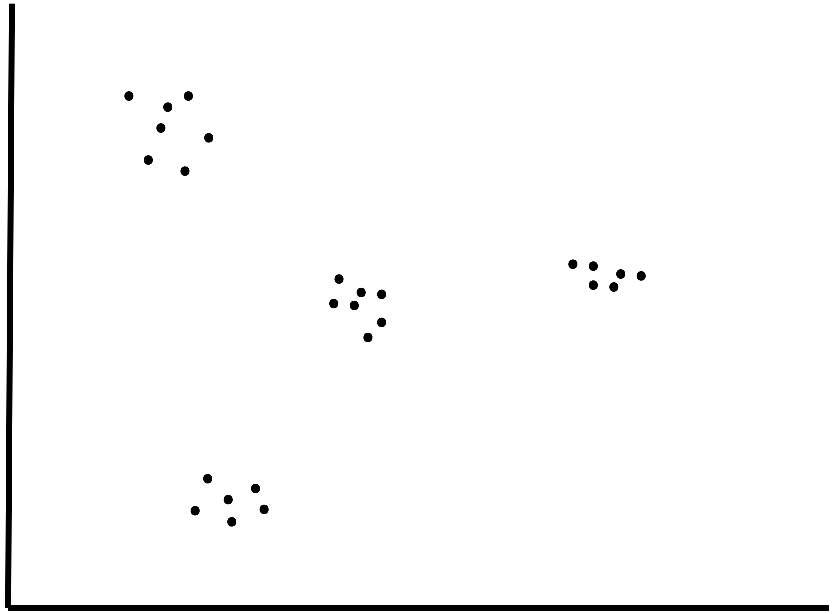
Clustering

Imagine you have a set of objects, represented by points in a space



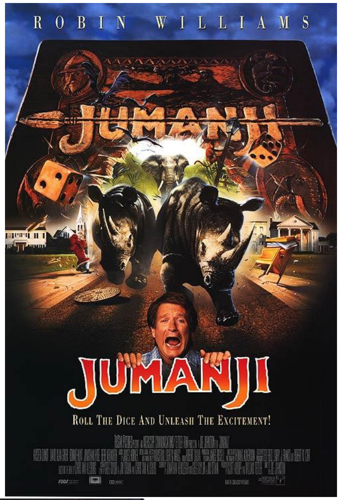
Clustering

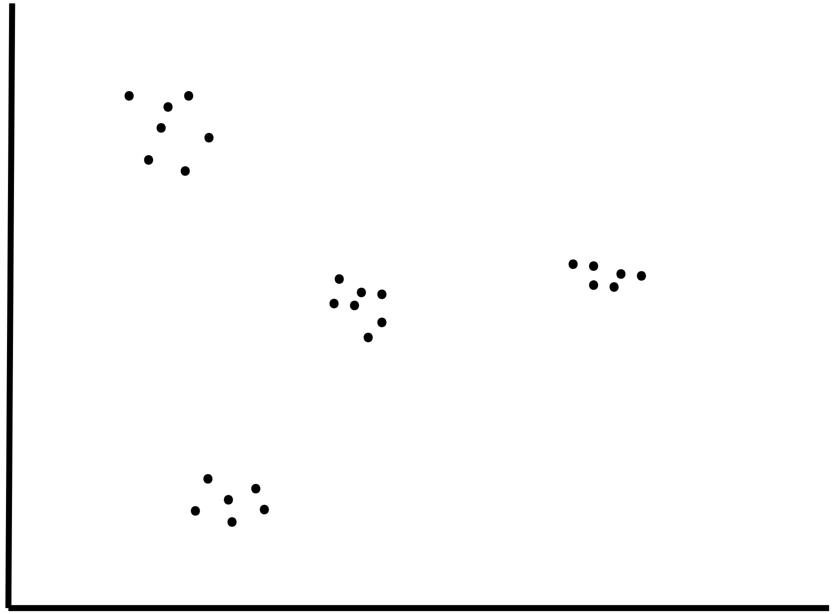
Imagine you have a set of objects, represented by points in a space

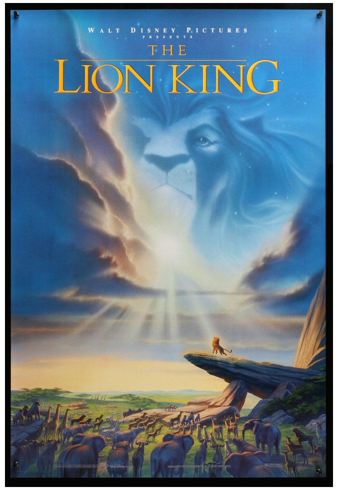


Clustering

Imagine you have a set of objects, represented by points in a space

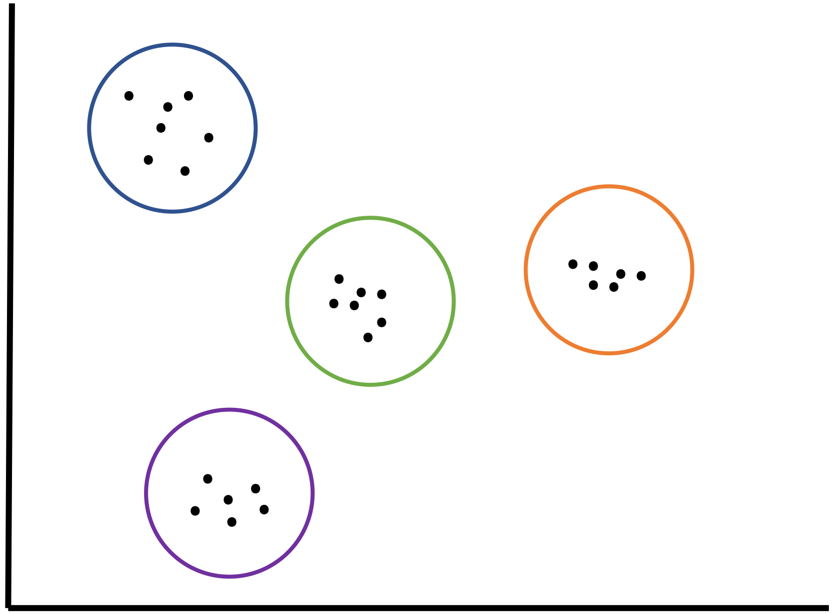






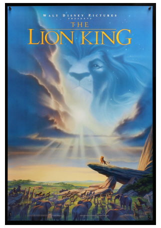
Clustering

The goal is to find *clusters* such that two objects who are in the same cluster are in some sense “similar” to each other



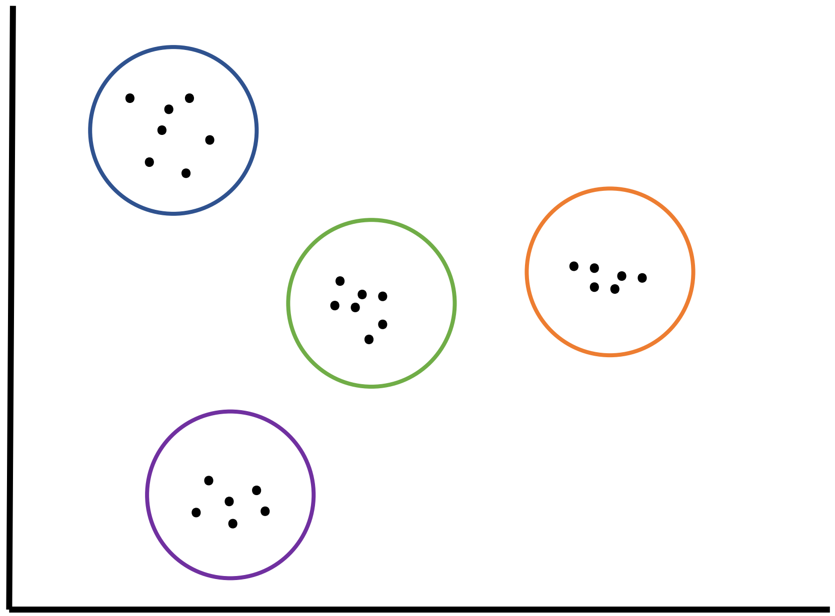
Clusters may represent similarity in how a plant looks (e.g. green vs. not green) or role in the ecosystem

Clusters may represent movie genre (drama, comedy, documentary) or medium (animation, live action)



Clustering

Clustering is *extremely important* in science and industry



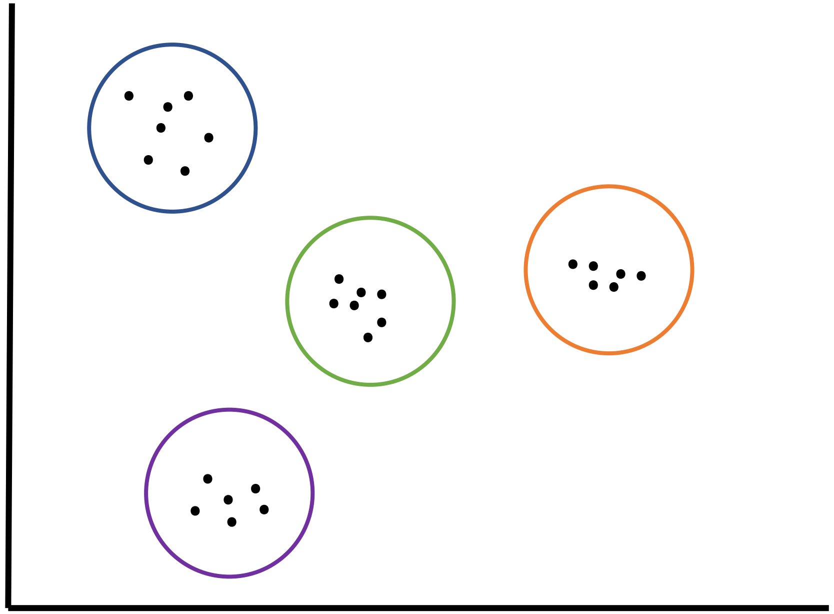
Anything that has to do with "big data" almost certainly involves some kind of clustering

- Scientists use clustering to find similarity in noisy data
  - Clustering *genes* to find functional similarities
  - Clustering *brain scans* (or other health data) to understand differences between people with/without certain conditions
  - Clustering *organisms* to understand evolution
- Netflix recommends what to watch next by clustering with what you have watched previously
  - Similar with Amazon, your local grocery store chain, or any other retailer!

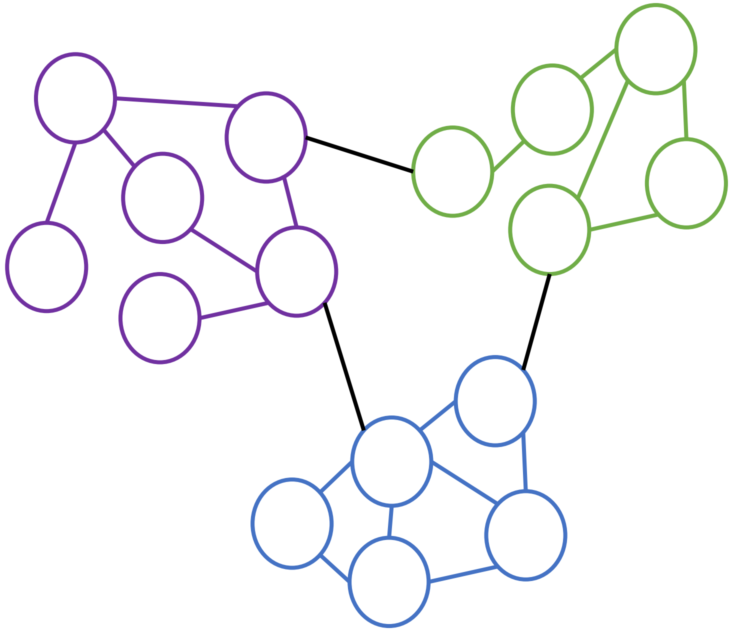
Clustering

We will study two kinds of clustering

General clustering

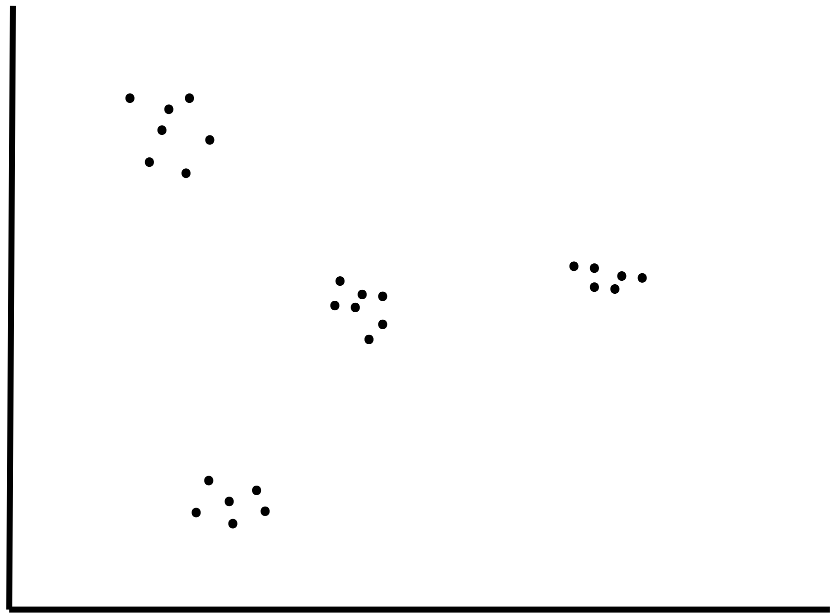


Clustering in graphs
(also known as *community detection*)



Clustering

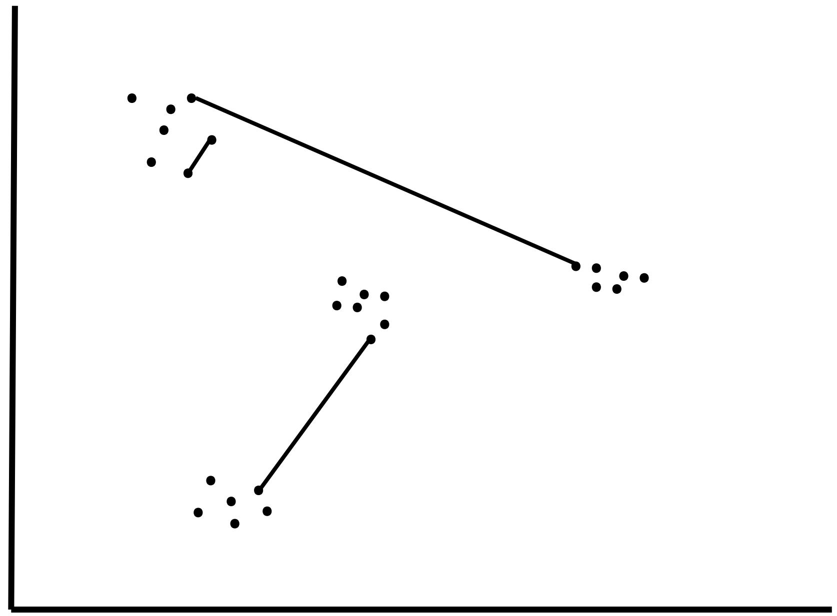
Imagine you have a set of objects, represented by points in a space



You are given a set of n objects $U = \{x_1, x_2, \cdots, x_n\}$ and a point in space $P = \{p_1, p_2, \cdots p_n\}$ for each object and an integer k, representing the number of clusters

Clustering

Imagine you have a set of objects, represented by points in a space



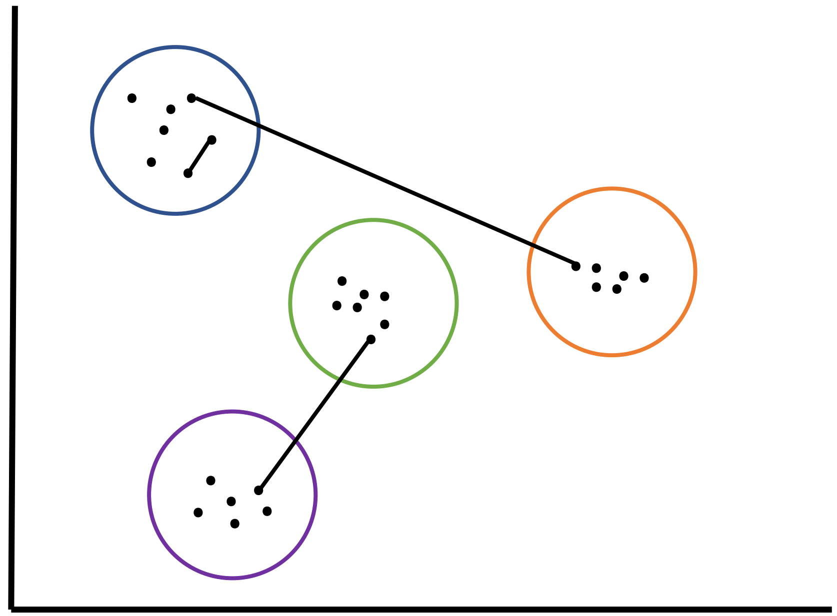
You are given a set of n objects $U = \{x_1, x_2, \cdots, x_n\}$ and a point in space $P = \{p_1, p_2, \cdots p_n\}$ for each object and an integer k, representing the number of clusters

You are also given a *distance* (or *similarity*) *function* $dist(p_1, p_2)$ that takes two points in space and returns a real-valued distance between them

- Distance should be *symmetric*, meaning $dist(p_i, p_j) = dist(p_j, p_i)$ for all i, j
- Distance should be nonzero if $x_i \neq x_j$

Clustering

Imagine you have a set of objects, represented by points in a space



You are given a set of n objects $U = \{x_1, x_2, \cdots, x_n\}$ and a point in space $P = \{p_1, p_2, \cdots p_n\}$ for each object and an integer k, representing the number of clusters

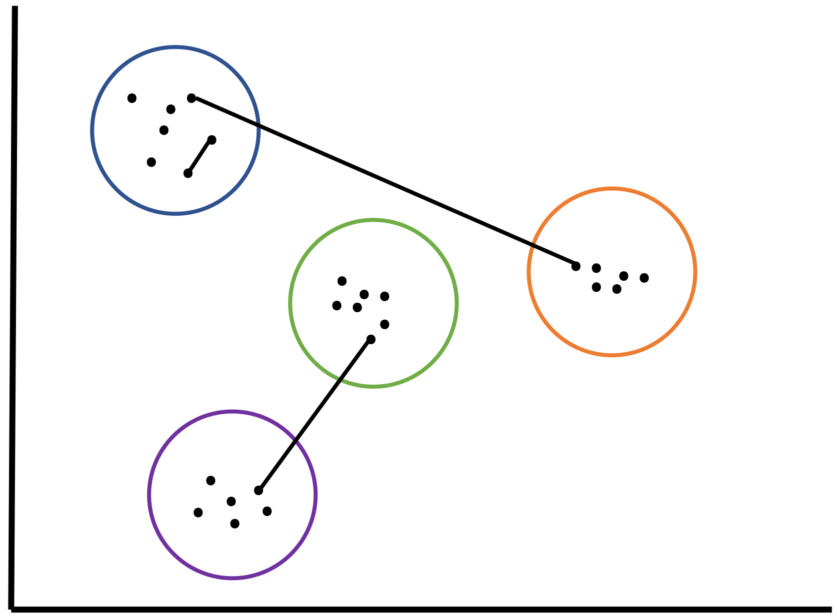
You are also given a *distance* (or *similarity*) *function* $dist(p_1, p_2)$ that takes two points in space and returns a real-valued distance between them

- Distance should be *symmetric*, meaning $dist(p_i, p_j) = dist(p_j, p_i)$ for all i, j
- Distance should be nonzero if $x_i \neq x_j$

Goal: Find k clusters *of maximum spacing*, meaning a clustering where the minimum distance between points in different clusters (spacing) is as large as possible

Clustering

Idea: Construct a disconnected graph by "greedily" connecting the closest points first until all points have a cluster!



You are given a set of n objects $U = \{x_1, x_2, \cdots, x_n\}$ and a point in space $P = \{p_1, p_2, \cdots p_n\}$ for each object and an integer k, representing the number of clusters

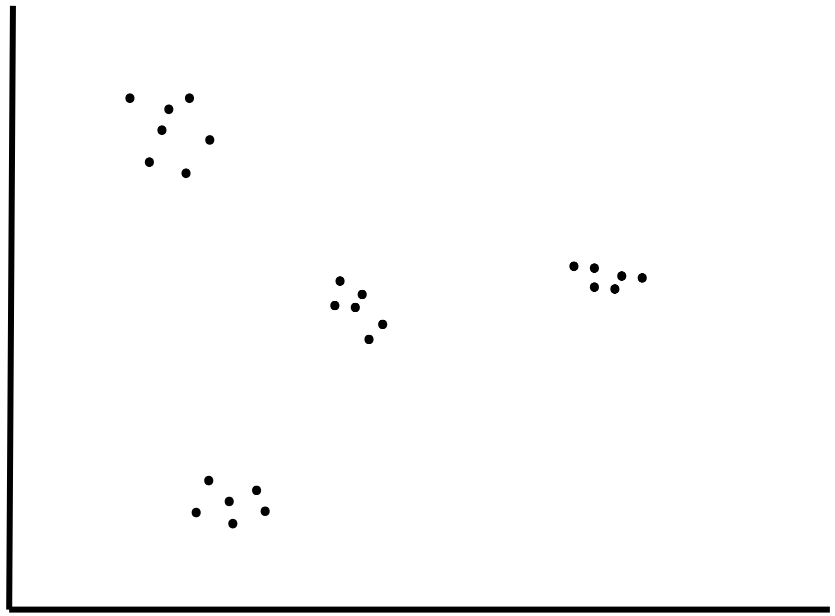
You are also given a *distance* (or *similarity*) *function* $dist(p_1, p_2)$ that takes two points in space and returns a real-valued distance between them

- Distance should be *symmetric*, meaning $dist(p_i, p_j) = dist(p_j, p_i)$ for all i, j
- Distance should be nonzero if $x_i \neq x_j$

Goal: Find k clusters *of maximum spacing*, meaning a clustering where the minimum distance between points in different clusters (spacing) is as large as possible

Clustering

Idea: Construct a disconnected graph by "greedily" connecting the closest points first until all points have a cluster!



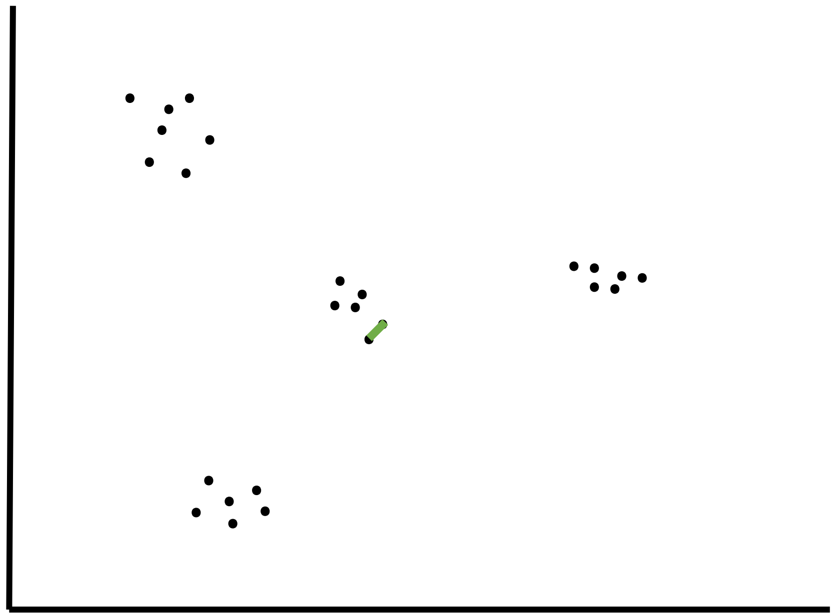
Begin with all points unassigned. Fill a priority queue Q with pairs of points (i, j) with values $dist(p_i, p_j)$

While there are points without any assignment, pull the next smallest distance pair (i, j) and…

1. If neither of (i, j) are assigned and there are fewer than k clusters, connect them and put them in their own cluster
2. If neither of (i, j) are assigned and there are already k clusters, do nothing
3. If one of i or j is assigned, connect them and assign them to the same cluster
4. If both i and j are assigned, merge their clusters
5. If i and j are in the same cluster, do nothing

Clustering

Idea: Construct a disconnected graph by "greedily" connecting the closest points first until all points have a cluster!



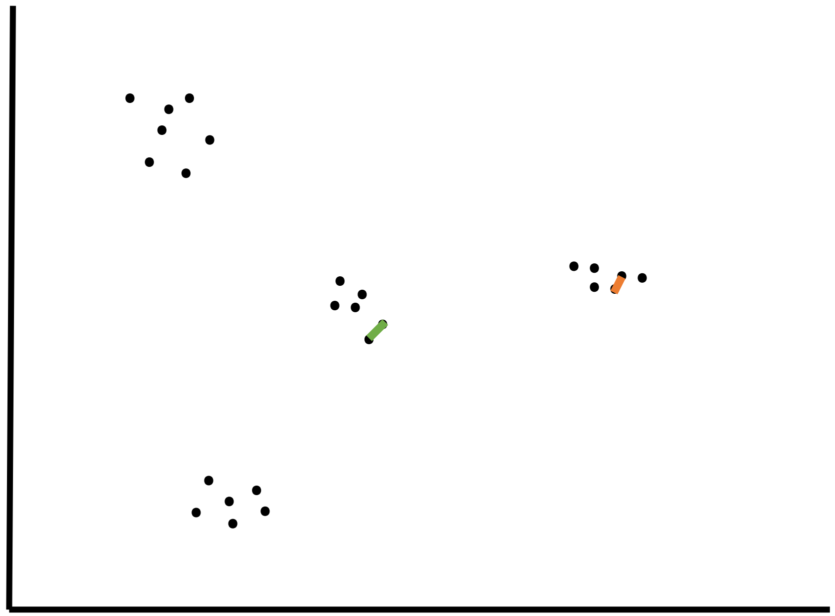
Begin with all points unassigned. Fill a priority queue Q with pairs of points (i, j) with values $dist(p_i, p_j)$

While there are points without any assignment, pull the next smallest distance pair (i, j) and…

1. If neither of (i, j) are assigned and there are fewer than k clusters, connect them and put them in their own cluster
2. If neither of (i, j) are assigned and there are already k clusters, do nothing
3. If one of i or j is assigned, connect them and assign them to the same cluster
4. If both i and j are assigned, merge their clusters
5. If i and j are in the same cluster, do nothing

Clustering

Idea: Construct a disconnected graph by "greedily" connecting the closest points first until all points have a cluster!



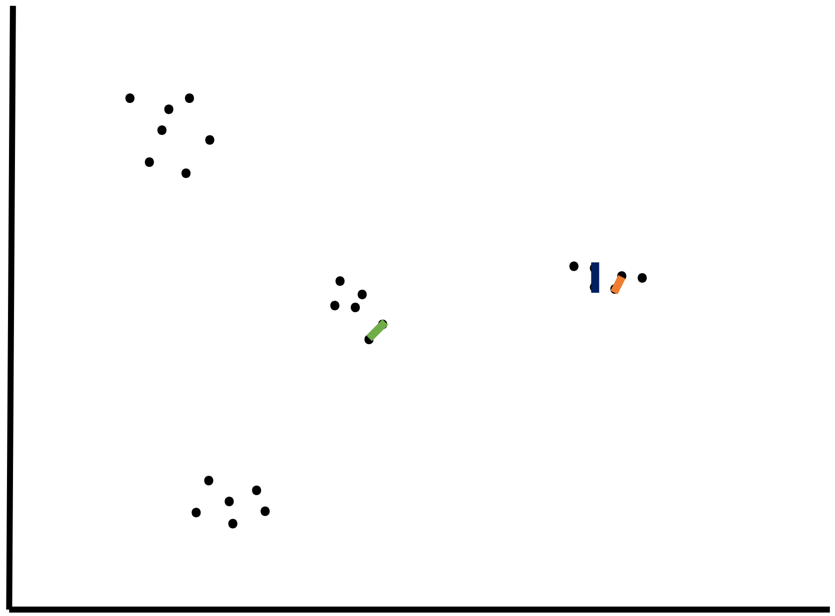
Begin with all points unassigned. Fill a priority queue Q with pairs of points (i, j) with values $dist(p_i, p_j)$

While there are points without any assignment, pull the next smallest distance pair (i, j) and…

1. If neither of (i, j) are assigned and there are fewer than k clusters, connect them and put them in their own cluster
2. If neither of (i, j) are assigned and there are already k clusters, do nothing
3. If one of i or j is assigned, connect them and assign them to the same cluster
4. If both i and j are assigned, merge their clusters
5. If i and j are in the same cluster, do nothing

Clustering

Idea: Construct a disconnected graph by "greedily" connecting the closest points first until all points have a cluster!



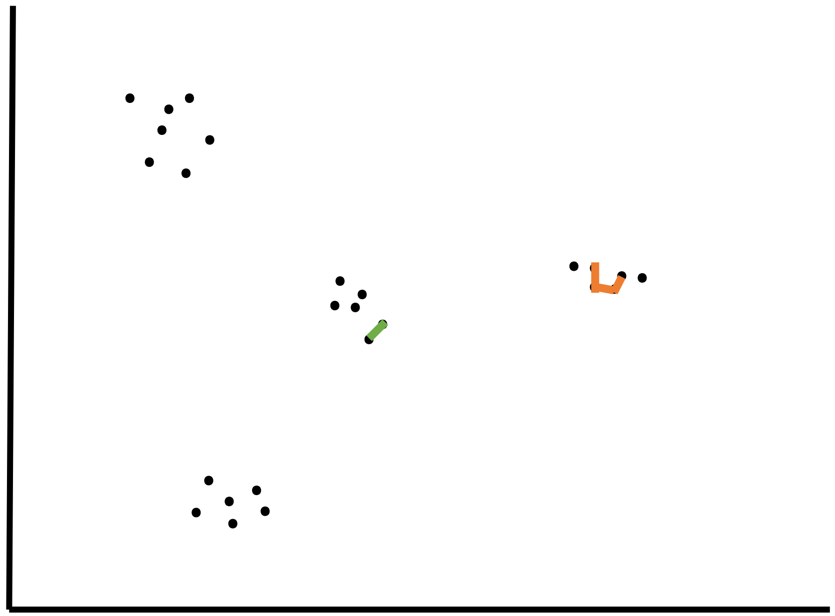
Begin with all points unassigned. Fill a priority queue Q with pairs of points (i, j) with values $dist(p_i, p_j)$

While there are points without any assignment, pull the next smallest distance pair (i, j) and…

1. If neither of (i, j) are assigned and there are fewer than k clusters, connect them and put them in their own cluster
2. If neither of (i, j) are assigned and there are already k clusters, do nothing
3. If one of i or j is assigned, connect them and assign them to the same cluster
4. If both i and j are assigned, merge their clusters
5. If i and j are in the same cluster, do nothing

Clustering

Idea: Construct a disconnected graph by "greedily" connecting the closest points first until all points have a cluster!



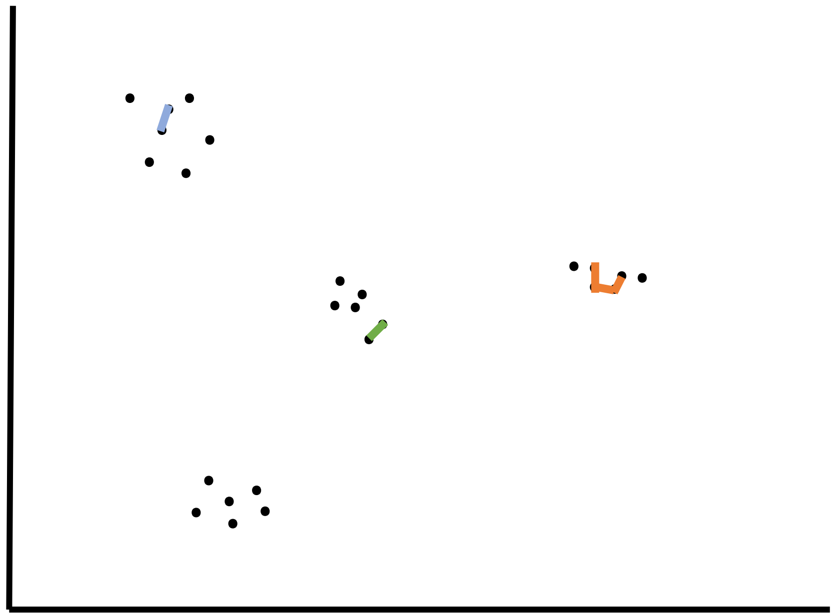
Begin with all points unassigned. Fill a priority queue Q with pairs of points (i, j) with values $dist(p_i, p_j)$

While there are points without any assignment, pull the next smallest distance pair (i, j) and…

1. If neither of (i, j) are assigned and there are fewer than k clusters, connect them and put them in their own cluster
2. If neither of (i, j) are assigned and there are already k clusters, do nothing
3. If one of i or j is assigned, connect them and assign them to the same cluster
4. If both i and j are assigned, merge their clusters
5. If i and j are in the same cluster, do nothing

Clustering

Idea: Construct a disconnected graph by "greedily" connecting the closest points first until all points have a cluster!



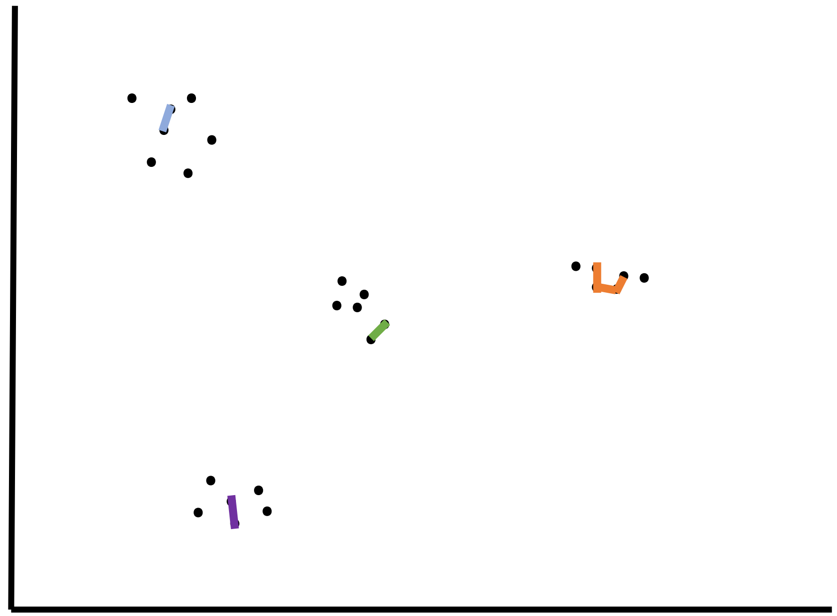
Begin with all points unassigned. Fill a priority queue Q with pairs of points (i, j) with values $dist(p_i, p_j)$

While there are points without any assignment, pull the next smallest distance pair (i, j) and…

1. If neither of (i, j) are assigned and there are fewer than k clusters, connect them and put them in their own cluster
2. If neither of (i, j) are assigned and there are already k clusters, do nothing
3. If one of i or j is assigned, connect them and assign them to the same cluster
4. If both i and j are assigned, merge their clusters
5. If i and j are in the same cluster, do nothing

Clustering

Idea: Construct a disconnected graph by "greedily" connecting the closest points first until all points have a cluster!



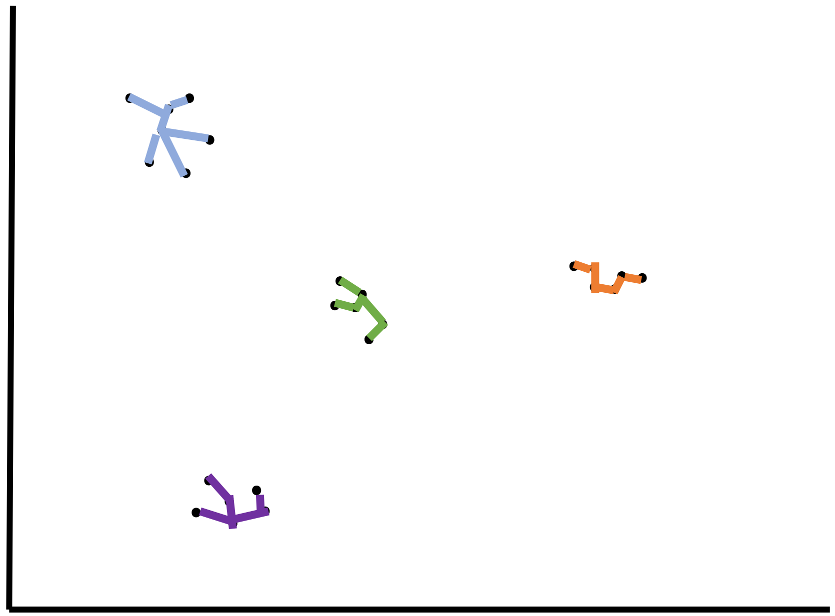
Begin with all points unassigned. Fill a priority queue Q with pairs of points (i, j) with values $dist(p_i, p_j)$

While there are points without any assignment, pull the next smallest distance pair (i, j) and…

1. If neither of (i, j) are assigned and there are fewer than k clusters, connect them and put them in their own cluster
2. If neither of (i, j) are assigned and there are already k clusters, do nothing
3. If one of i or j is assigned, connect them and assign them to the same cluster
4. If both i and j are assigned, merge their clusters
5. If i and j are in the same cluster, do nothing

Clustering

Idea: Construct a disconnected graph by "greedily" connecting the closest points first until all points have a cluster!



Begin with all points unassigned. Fill a priority queue Q with pairs of points (i, j) with values $dist(p_i, p_j)$

While there are points without any assignment, pull the next smallest distance pair (i, j) and…

1. If neither of (i, j) are assigned and there are fewer than k clusters, connect them and put them in their own cluster
2. If neither of (i, j) are assigned and there are already k clusters, do nothing
3. If one of i or j is assigned, connect them and assign them to the same cluster
4. If both i and j are assigned, merge their clusters
5. If i and j are in the same cluster, do nothing

Clustering

Idea: Construct a disconnected graph by "greedily" connecting the closest points first until all points have a cluster!

Begin with all points unassigned. Fill a priority queue Q with pairs of points (i, j) with values $dist(p_i, p_j)$

While there are points without any assignment, pull the next smallest distance pair (i, j) and…

1. If neither of (i, j) are assigned and there are fewer than k clusters, connect them and put them in their own cluster
2. If neither of (i, j) are assigned and there are already k clusters, do nothing
3. If one of i or j is assigned, connect them and assign them to the same cluster
4. If both i and j are assigned, merge their clusters
5. If i and j are in the same cluster, do nothing

Clustering

Idea: Construct a disconnected graph by "greedily" connecting the closest points first until all points have a cluster!

Does this algorithm or its output remind us of anything?

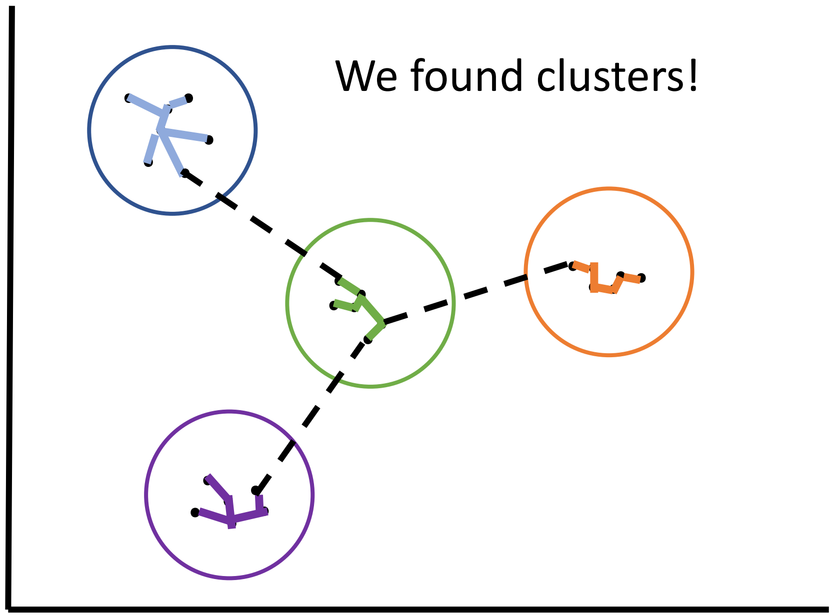
Begin with all points unassigned. Fill a priority queue Q with pairs of points (i, j) with values $dist(p_i, p_j)$

While there are points without any assignment, pull the next smallest distance pair (i, j) and…

1. If neither of (i, j) are assigned and there are fewer than k clusters, connect them and put them in their own cluster
2. If neither of (i, j) are assigned and there are already k clusters, do nothing
3. If one of i or j is assigned, connect them and assign them to the same cluster
4. If both i and j are assigned, merge their clusters
5. If i and j are in the same cluster, do nothing

Clustering

Idea: Construct a disconnected graph by "greedily" connecting the closest points first until all points have a cluster!



Does this algorithm or its output remind us of anything?
We found a subgraph of a minimum spanning tree!

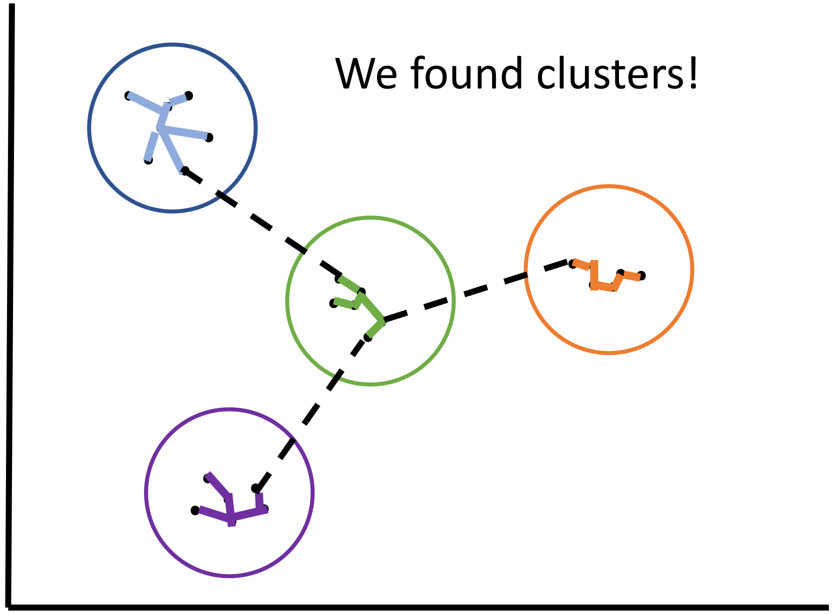
Begin with all points unassigned. Fill a priority queue Q with pairs of points (i, j) with values $dist(p_i, p_j)$

While there are points without any assignment, pull the next smallest distance pair (i, j) and…

1. If neither of (i, j) are assigned and there are fewer than k clusters, connect them and put them in their own cluster
2. If neither of (i, j) are assigned and there are already k clusters, do nothing
3. If one of i or j is assigned, connect them and assign them to the same cluster
4. If both i and j are assigned, merge their clusters
5. If i and j are in the same cluster, do nothing

Clustering

We can modify Kruskal's algorithm for finding a minimum spanning tree



Kruskal's MST Algorithm

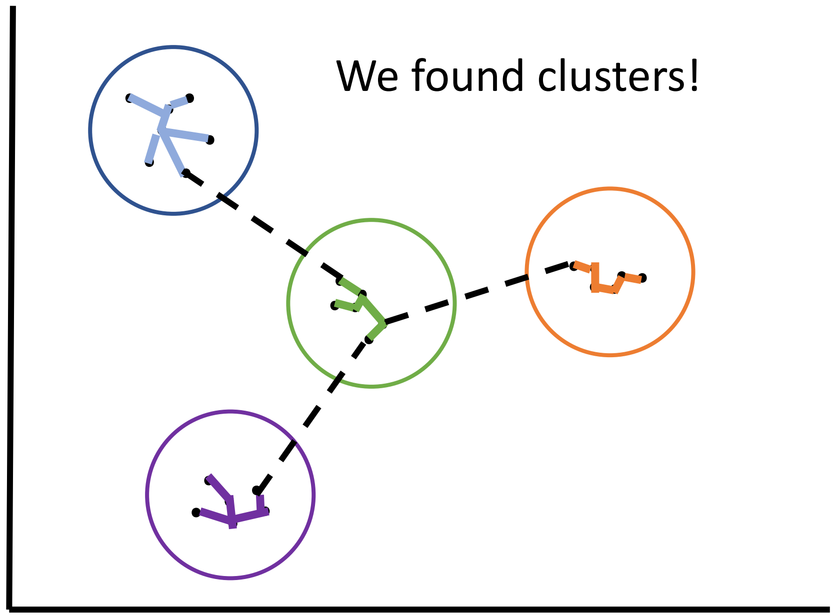
Start with $T = \emptyset$
For each edge (i, j) in ascending order of weight:

- If adding (i, j) would decrease the number of connected components in the graph, add (i, j) to T

We just need to stop the algorithm when we have k connected components, which are our clusters!

Clustering

We can modify Kruskal's algorithm for finding a minimum spanning tree



Kruskal's MST Algorithm

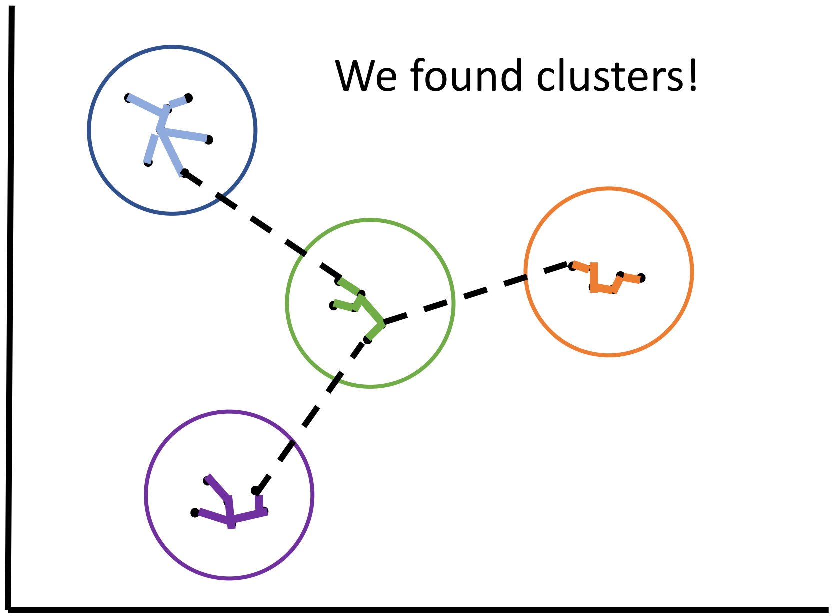
Start with $T = \emptyset$
For each edge (i,j) in ascending order of weight:
- If adding (i,j) would decrease the number of connected components in the graph, add (i,j) to T

We just need to stop the algorithm when we have k connected components, which are our clusters!

Equivalently: we could compute the whole MST using any algorithm, then remove the $k-1$ most expensive edges!

Clustering

We can modify Kruskal's algorithm for finding a minimum spanning tree



Kruskal's MST Algorithm

Start with $T = \emptyset$
For each edge (i, j) in ascending order of weight:

- If adding (i, j) would decrease the number of connected components in the graph, add (i, j) to T

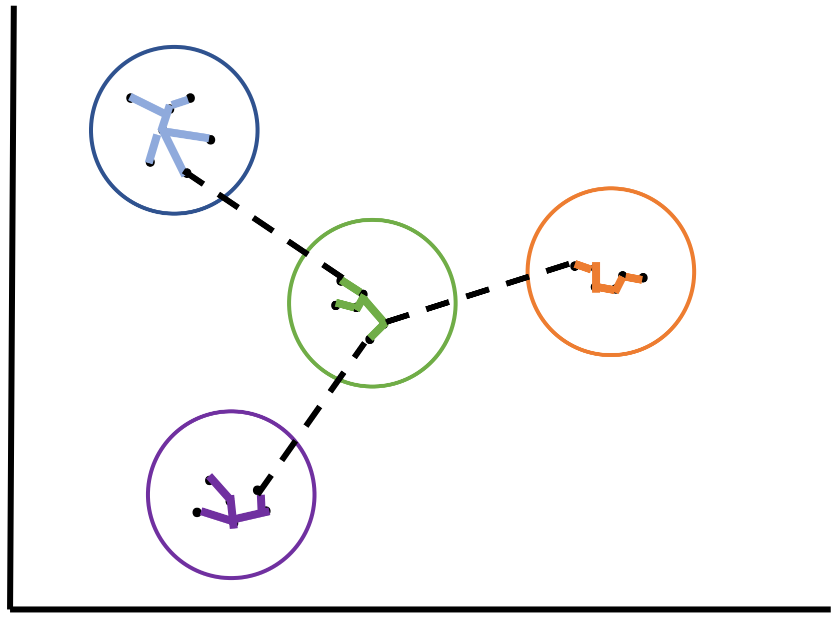
We just need to stop the algorithm when we have k connected components, which are our clusters!

Equivalently: we could compute the whole MST using any algorithm, then remove the $k - 1$ most expensive edges.

So we can reduce the problem of finding a maximum spacing clustering to finding an MST!

Clustering

Claim: The components $C_1, C_2, \cdots C_k$ formed by deleting the $k - 1$ most expensive edges of the minimum spanning tree T constitute a k-clustering of maximum spacing.

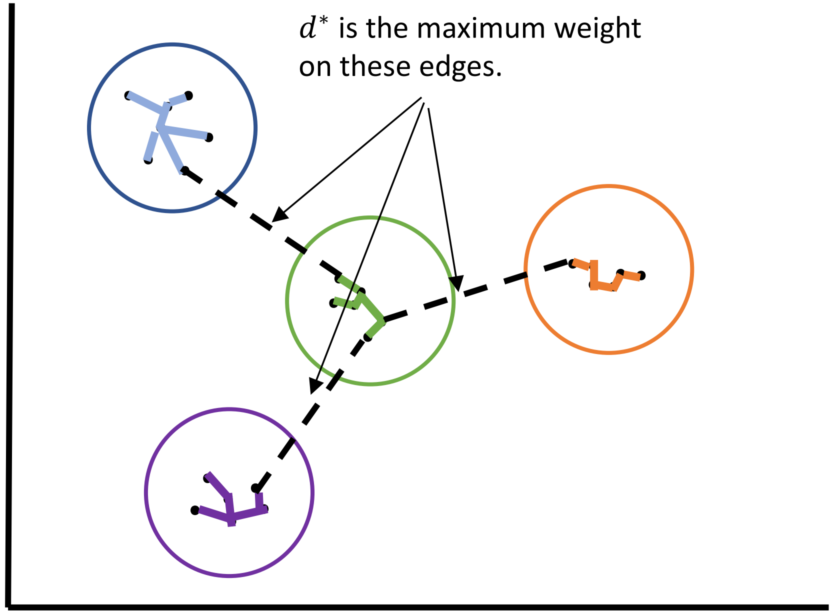


Clustering

Claim: The components $C_1, C_2, \cdots C_k$ formed by deleting the $k-1$ most expensive edges of the minimum spanning tree T constitute a k-clustering of maximum spacing.

Let C denote the clustering found by the procedure above.

The spacing of C is the weight of the $(k-1)^{st}$ most expensive edge in T. Denote this weight d^*.



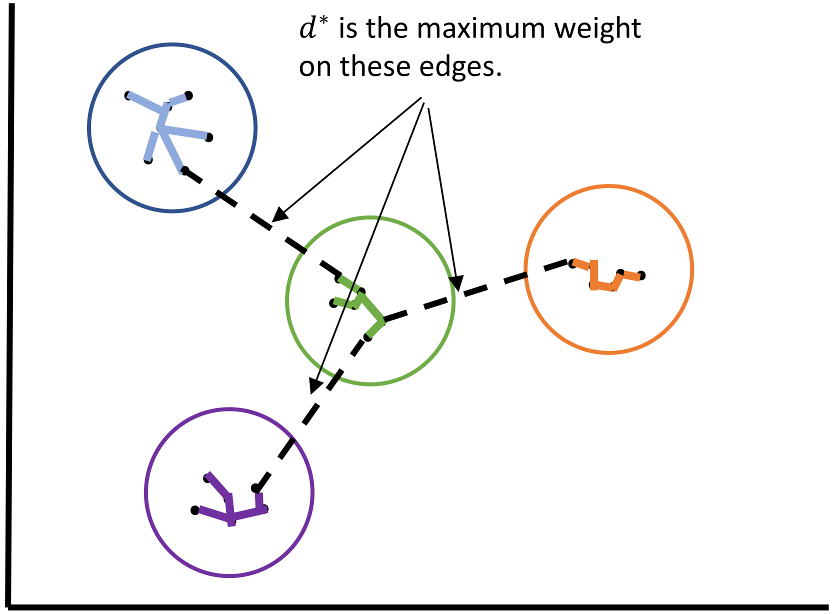
Clustering

Claim: The components $C_1, C_2, \cdots C_k$ formed by deleting the $k-1$ most expensive edges of the minimum spanning tree T constitute a k-clustering of maximum spacing.

Let C denote the clustering found by the procedure above.

The spacing of C is the weight of the $(k-1)^{st}$ most expensive edge in T. Denote this weight d^*.

Consider some other valid k-clustering C'. We need to show that the spacing of C' is at most d^*.



Clustering

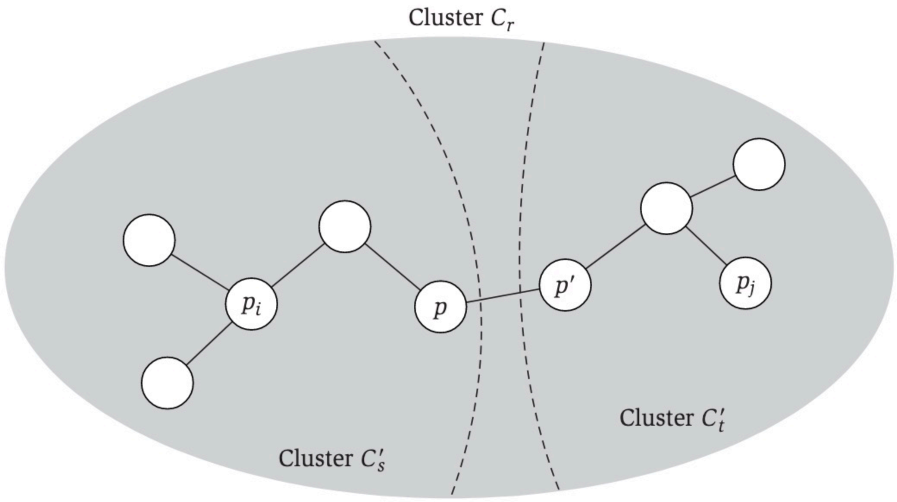
Claim: The components $C_1, C_2, \cdots C_k$ formed by deleting the $k-1$ most expensive edges of the minimum spanning tree T constitute a k-clustering of maximum spacing.

Let C denote the clustering found by the procedure above.

The spacing of C is the weight of the $(k-1)^{st}$ most expensive edge in T. Denote this weight d^*.

Consider some other valid k-clustering C'. We need to show that the spacing of C' is at most d^*.

Since C and C' are not the same, one of our clusters C_r is not a subset of any of the k sets in C'. This means there must be points $p_i, p_j \in C_r$ that belong to difference clusters in C', for example $p_i \in C_s{}'$ and $p_j \in C_t{}'$.



Clustering

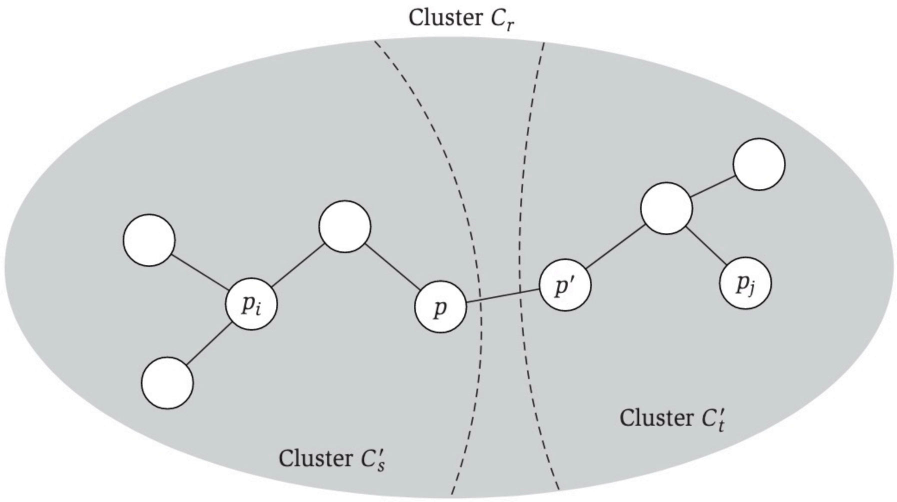
Claim: The components $C_1, C_2, \cdots C_k$ formed by deleting the $k-1$ most expensive edges of the minimum spanning tree T constitute a k-clustering of maximum spacing.

Let C denote the clustering found by the procedure above.

The spacing of C is the weight of the $(k-1)^{st}$ most expensive edge in T. Denote this weight d^*.

Consider some other valid k-clustering C'. We need to show that the spacing of C' is at most d^*.

Since C and C' are not the same, one of our clusters C_r is not a subset of any of the k sets in C'. This means there must be points $p_i, p_j \in C_r$ that belong to difference clusters in C', for example $p_i \in C_s{}'$ and $p_j \in C_t{}'$.



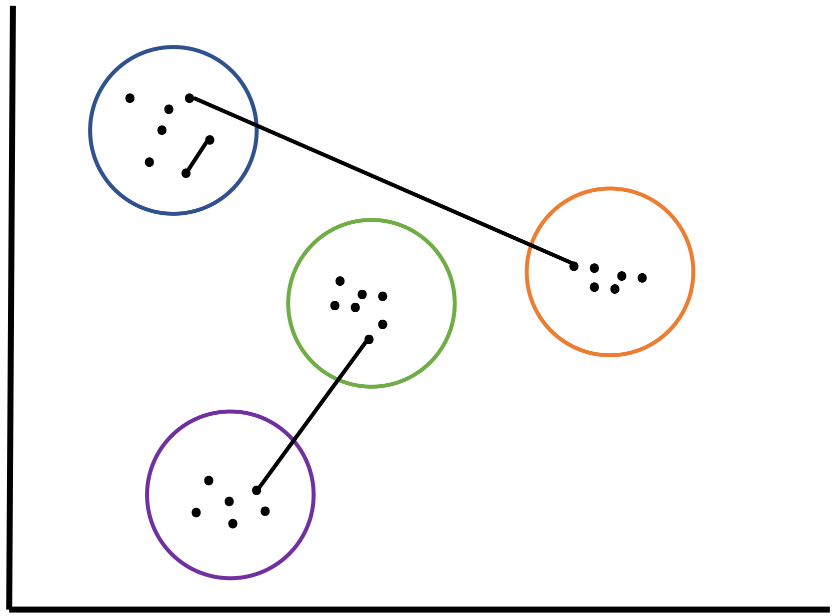
Since our MST algorithm included the edge from p to p', it must have had weight at most d^*.

Since p and p' belong to different clusters in C', its spacing must be at most d^*! Proof complete.

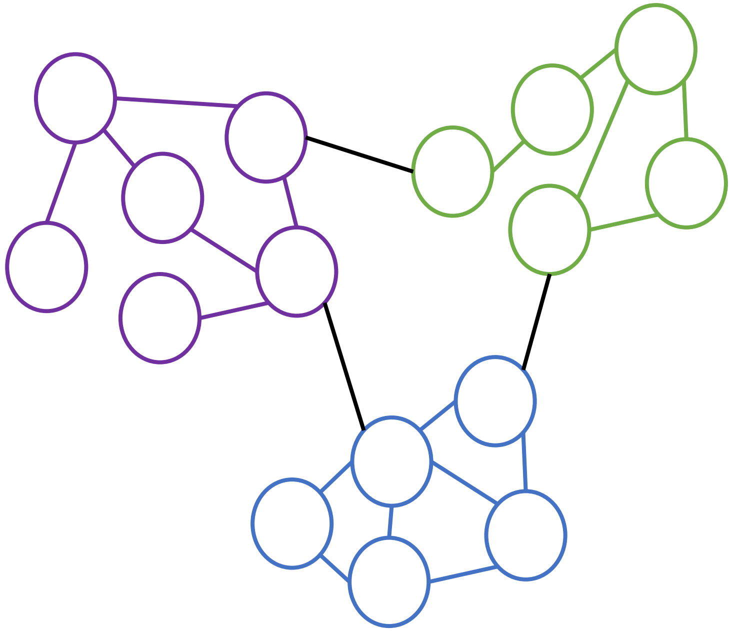
Clustering

We will study two kinds of clustering

General clustering



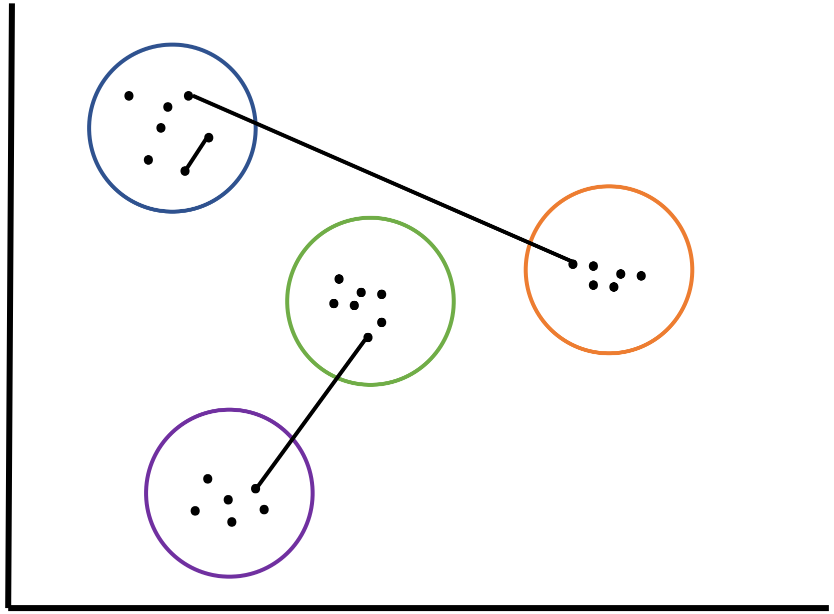
Clustering in graphs
(also known as *community detection*)



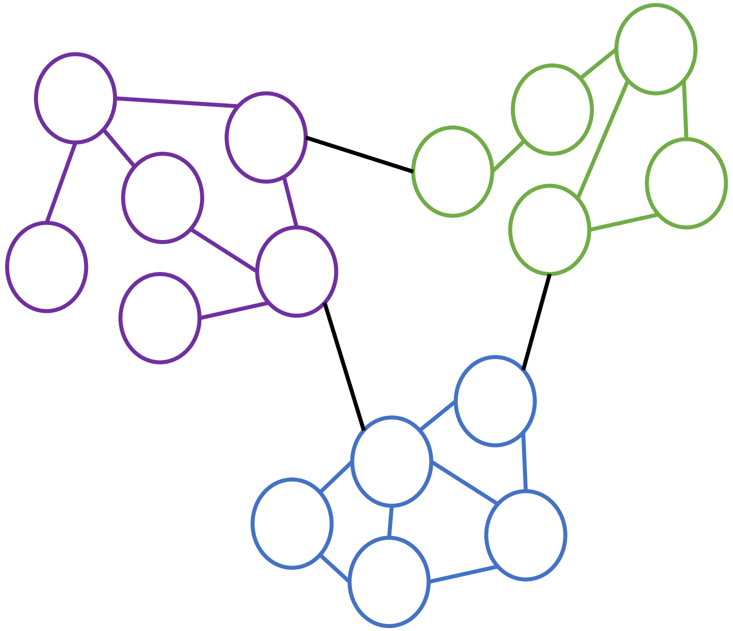
Clustering

We will study two kinds of clustering

General clustering



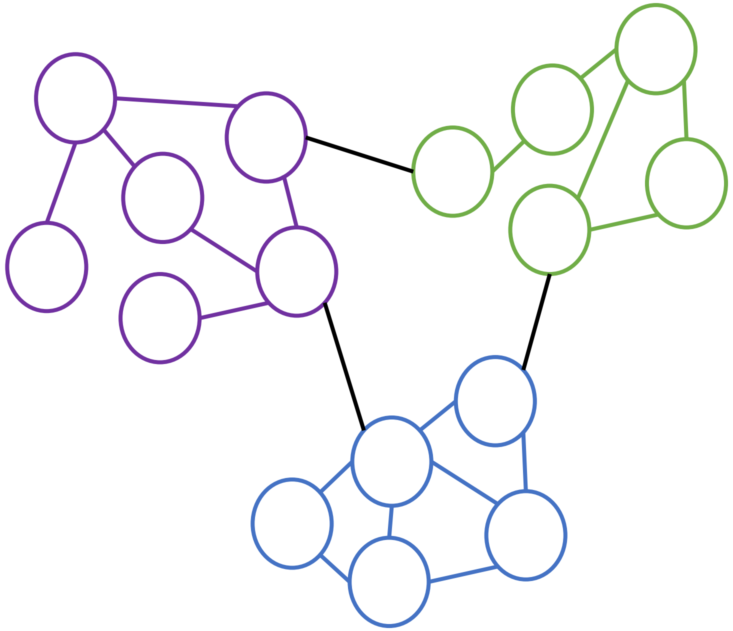
Clustering in graphs
(also known as *community detection*)



Can we apply the same techniques for clustering in graphs?

Next Time: Clustering in Graphs

Clustering in graphs
(also known as *community detection*)



Given a graph, we want to partition the nodes in to *clusters* or *communities* for some purpose

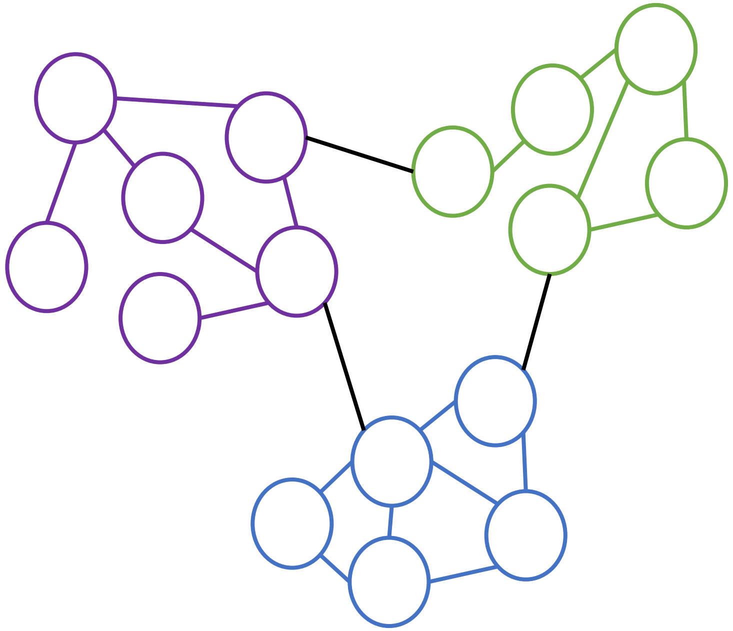
- Detecting literal communities in social networks
- Detecting functionally similar genes in networks of gene interactions
- Detecting functional units in brain networks
- Detecting suspicious actors in financial transaction networks

If we have a weighted graph, we can obviously apply the generic clustering approach where $dist(i, j)$ is the weighted shortest path (or other distance function) between nodes i and j.

However, graphs already encode structure directly, so why not use it?

Next Time: Clustering in Graphs

Clustering in graphs
(also known as *community detection*)



There are approximately one bajillion different ways to partition graphs for this task

- Subfield of data mining/network science called "community detection".

Some techniques rely purely on structural information (e.g. the connections in the graph).

- We will focus on techniques like these tomorrow

Other techniques incorporate *domain-specific* information when partitioning the nodes.

- These are too specific for this course, but generally combine the generic clustering technique with some version of structural partitioning

Wrap-up

Extra Credit Assignments are open

Tomorrow: Clustering in graphs + other advanced topics

Thursday: Final Exam Review (form for questions sent out last night)

Final Exam: Released Thursday night, due Monday night