DSC 40B

Lecture 28 :

MSTs and Clustering

MSTs and Clustering

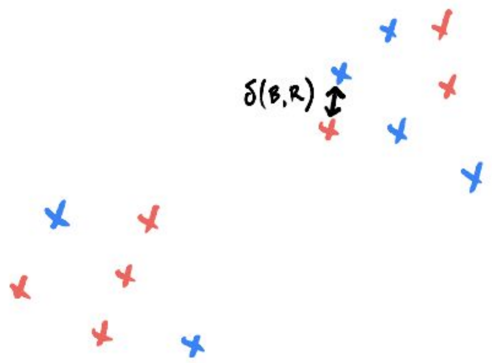
Clustering: Find Groups

Goal: identify the groups in data.

Example:

Clustering, Formalized

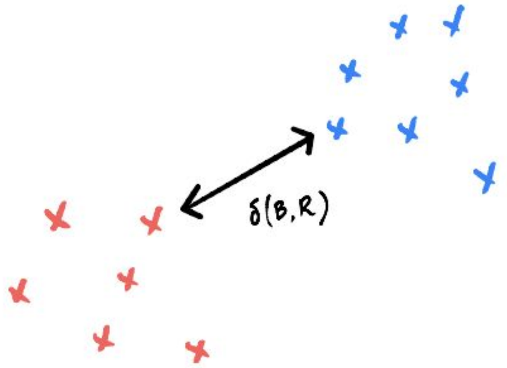
- We frame as an **optimization** problem.
- **Given**: n data points.
- **Goal**: assign color to each point (red or blue) to *maximize* the distance between the *closest* pair of red and blue points.



Bad Clustering

Clustering, Formalized

- We frame as an **optimization** problem.
- **Given**: n data points.
- **Goal**: assign color to each point (red or blue) to *maximize* the distance between the *closest* pair of red and blue points.



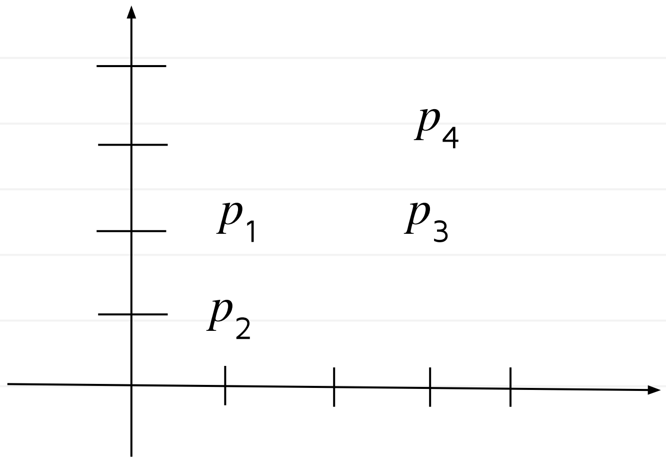
Good Clustering

Brute Force Solution

- Try all possible assignments; return best.
- If there are n data points, there are **$\Theta(2^n)$** assignments.
- **Exponential time**; very slow.
  - Practical only for ~ 50 data points.
- Instead, we will turn it into a **graph problem**.

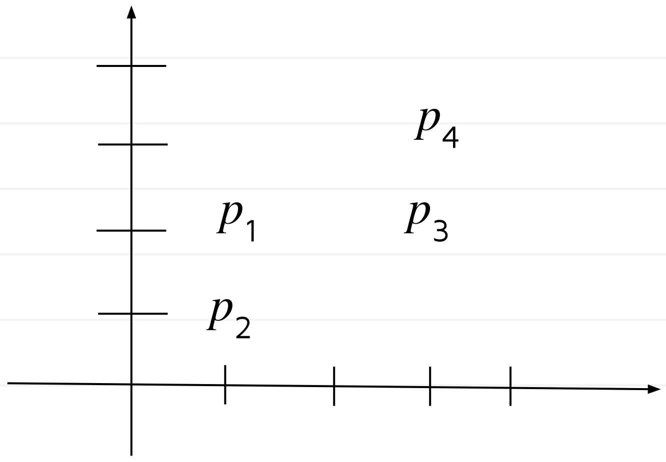
Distance Graphs

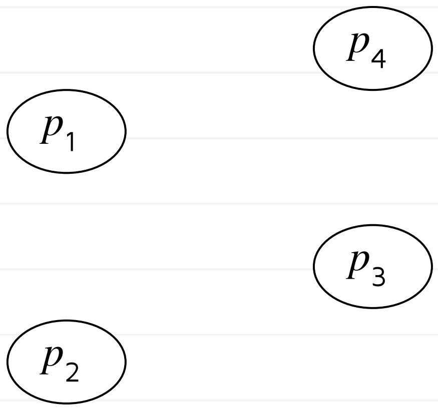
- Given n data points, $p_1, p_2, \ldots, p_n$, create *complete* graph with $V = \{p_1, \ldots, p_n\}$.
- Set weight of edge (p_i, p_j) = $\mathrm{dist}(p_i, p_j)$.
- The result is a weighted, undirected **distance graph.**



Distance Graphs

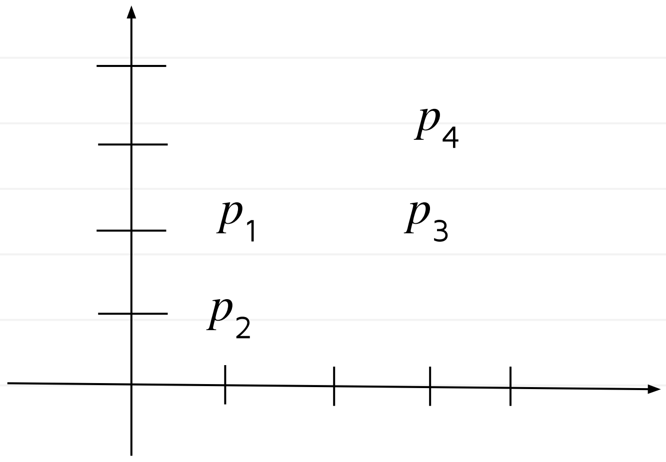
- Given n data points, $p_1, p_2, \dots, p_n$, create *complete* graph with $V = \{p_1, \dots, p_n\}$.
- Set weight of edge (p_i, p_j) = dist(p_i, p_j).
- The result is a weighted, undirected **distance graph.**

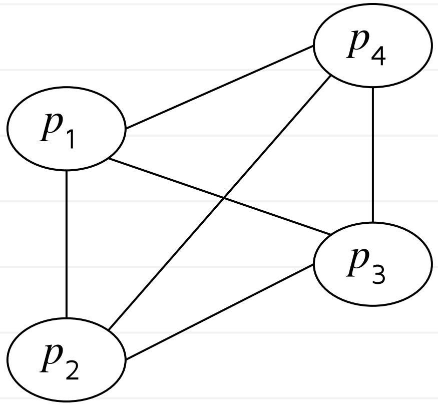




Distance Graphs

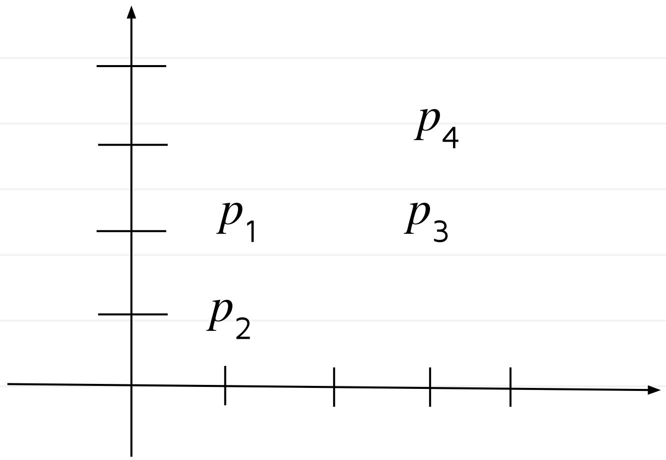
- Given n data points, $p_1, p_2, \dots, p_n$, create *complete* graph with $V = \{p_1, \dots, p_n\}$.
- Set weight of edge (p_i, p_j) = dist(p_i, p_j).
- The result is a weighted, undirected **distance graph.**

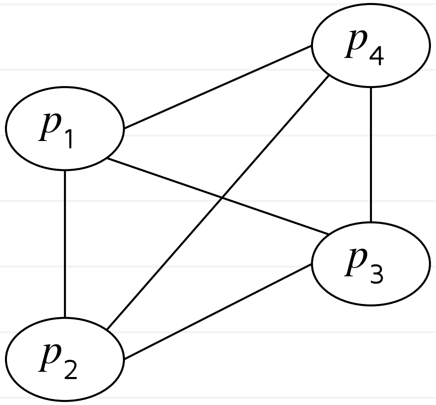




Distance Graphs

- Given n data points, $p_1, p_2, \dots, p_n$, create *complete* graph with $V = \{p_1, \dots, p_n\}$.
- Set weight of edge (p_i, p_j) = $\mathrm{dist}(p_i, p_j)$.
- The result is a weighted, undirected **distance graph.**

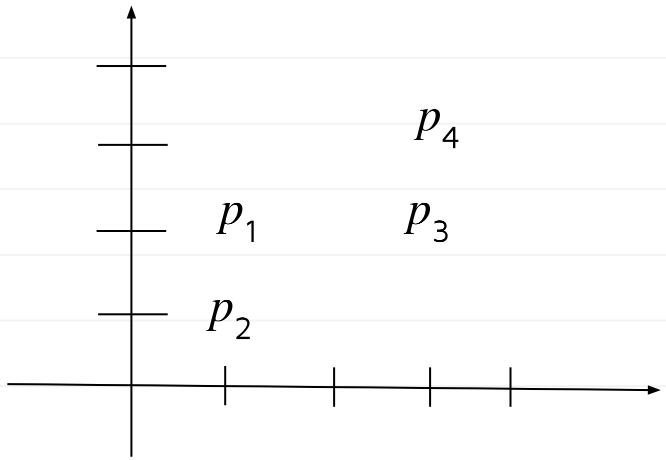


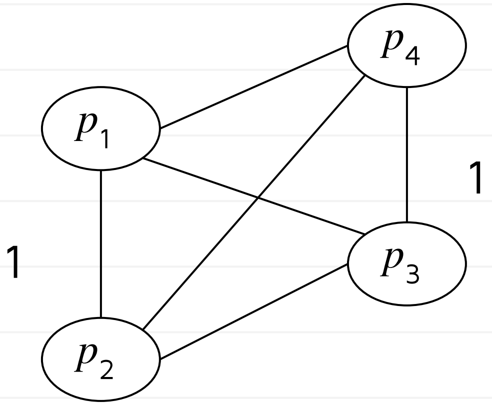


Weight is the distance

Distance Graphs

- Given n data points, $p_1, p_2, \dots, p_n$, create *complete* graph with $V = \{p_1, \dots, p_n\}$.
- Set weight of edge (p_i, p_j) = $\mathrm{dist}(p_i, p_j)$.
- The result is a weighted, undirected **distance graph.**

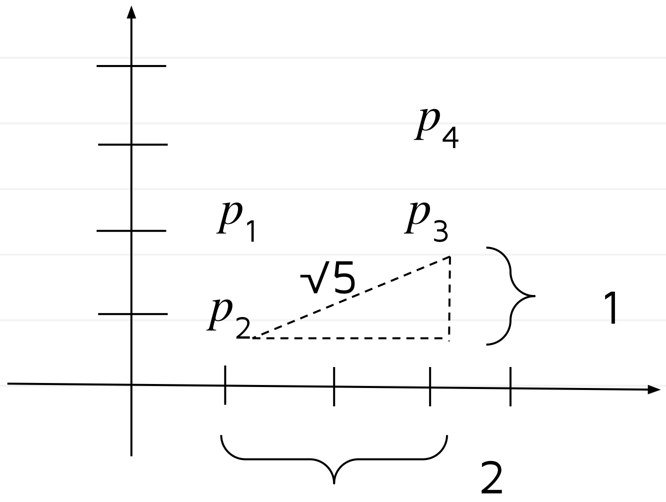


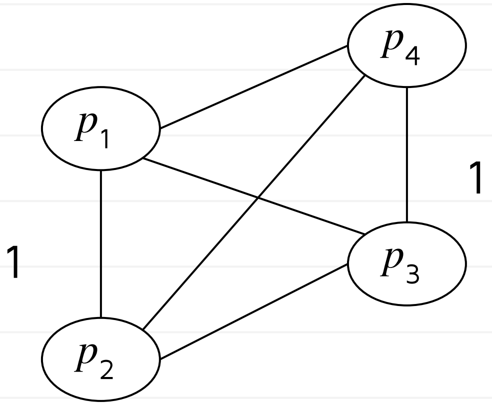


Weight is the distance

Distance Graphs

- Given n data points, $p_1, p_2, \dots, p_n$, create *complete* graph with $V = \{p_1, \dots, p_n\}$.
- Set weight of edge $(p_i, p_j) = \mathrm{dist}(p_i, p_j)$.
- The result is a weighted, undirected **distance graph.**

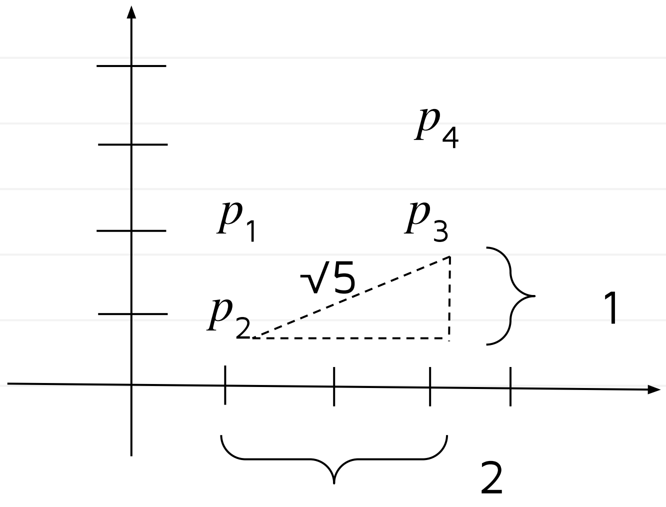


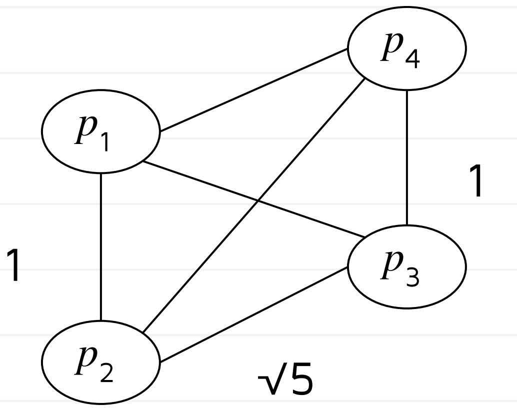


Weight is the distance

Distance Graphs

- Given n data points, $p_1, p_2, \ldots, p_n$, create *complete* graph with $V = \{p_1, \ldots, p_n\}$.
- Set weight of edge (p_i, p_j) = dist(p_i, p_j).
- The result is a weighted, undirected **distance graph.**

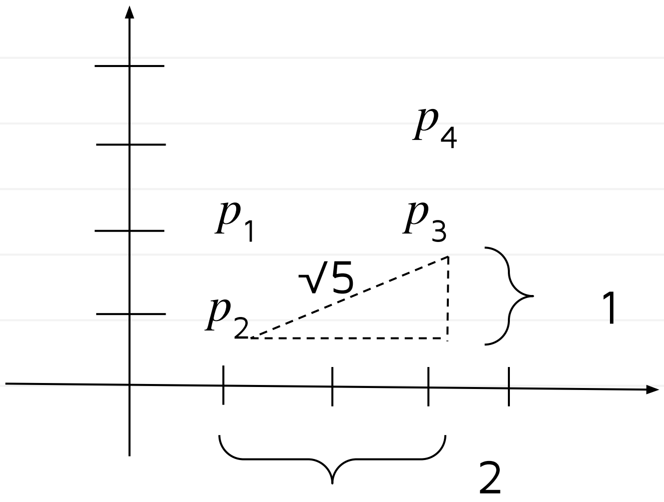


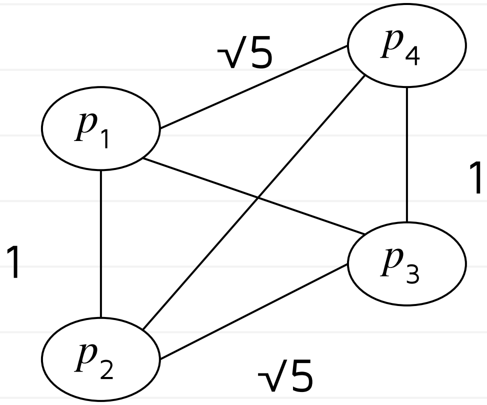


Weight is the distance

Distance Graphs

- Given n data points, $p_1, p_2, \dots, p_n$, create *complete* graph with $V = \{p_1, \dots, p_n\}$.
- Set weight of edge $(p_i, p_j) = \text{dist}(p_i, p_j)$.
- The result is a weighted, undirected **distance graph.**

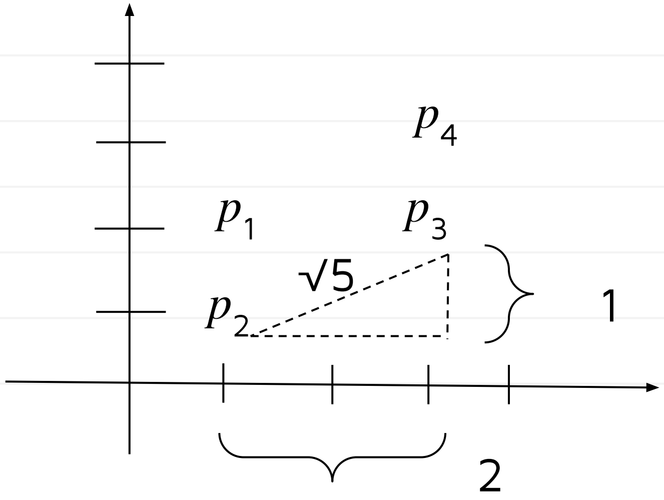


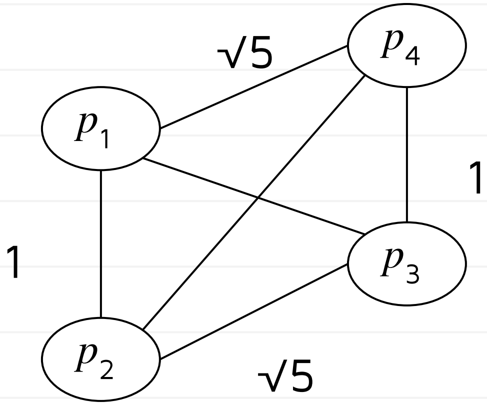


Weight is the distance

Distance Graphs

- Given n data points, $p_1, p_2, \dots, p_n$, create *complete* graph with $V = \{p_1, \dots, p_n\}$.
- Set weight of edge (p_i, p_j) = $\mathrm{dist}(p_i, p_j)$.
- The result is a weighted, undirected **distance graph.**





You can finish the rest

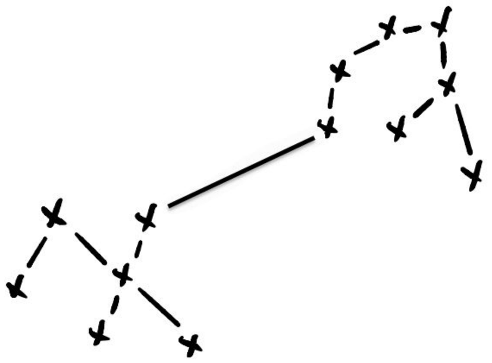
Main Idea

We can always think of a set of points in a (metric) space as a weighted distance graph.

This is a ***very important idea,*** because it allows us to use our graph algorithms!

Clustering with MSTs

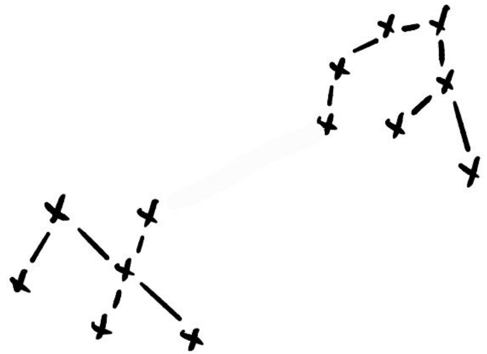
- Given n data points and a number of clusters, k:
  - Create distance graph G.
  - Run Kruskal's Algorithm on G until there are only k components.



- The resulting connected components are the clusters.
- This is known as single-linkage clustering.

Clustering with MSTs

- Given n data points and a number of clusters, k:
  - Create distance graph G.
  - Run Kruskal's Algorithm on G until there are only k components.



- The resulting connected components are the clusters.
- This is known as single-linkage clustering.

Single-Linkage Clustering

- Time complexity of single-linkage is determined by Kruskal's Algorithm: $\Theta(E \log E)$.
- Since distance graph is complete, $E = \Theta(V^2)$, and so $\Theta(E \log E) = \Theta(V^2 \log V) = \Theta(n^2 \log n)$
- Practically, can cluster ~ 10, 000 points.

Summary

- We started the quarter with a **brute force solution**.
  - Took $\Theta(2^n)$ time, only feasible for a few dozen points.
- We've now reframed the problem using graph theory.
  - Now only $\Theta(n^2 \log n)$ time!
  - Feasible for tens of thousands of points.

Why Algorithms?

- Data scientists use computers as tools.
- But solving a problem isn't just about coding it up.
- You need to know how to analyze your code and use the right algorithms and data structures to make your solution efficient.

Thank you!

Do you have any questions?

CampusWire!