STAT 214 Spring 2025 Week 6

Austin Zane

Clustering

https://vdsbook.com/07-cluster

What is clustering?

What is clustering?

A clustering of X made up of k clusters is a collection of sets $C = \{C_i\}_{i=1}^{k}$, where C forms a partition of X ($C_i \subseteq X$, $i \neq j \Rightarrow C_i \cap C_j = \emptyset$, $\bigcup_i C_i = X$).

What is the point of clustering? Why do we do it?

What is the point of clustering? Why do we do it?

Don’t do clustering just because it’s popular, or everyone else in a field is doing it.

Think about the scientific question which clustering will answer, or how it will better help you understand the world.

A “grouping” of things doesn’t always exist in the world.

- Things exist on a continuum
- Humans nonetheless group things to help us organize and synthesize our knowledge.
  - Species
  - Disease type
  - Book genre
  - Astronomical objects (think planets vs. asteroids)

Computers can cluster things for us that we can’t wrap our brains around!

How do we cluster?

1. Put the data in some space where distances between points somehow reflect reality (often a dimension reduction)
2. Run some clustering algorithm

Many fancy “clustering algorithms” do both 1 and 2. But really step 2 is where the grouping into clusters happens.

Let’s start with a brief detour into step 1, expanding on Week 4.

Turning data into a graph

k-nn graph: $G = (V, E) : \ V = X, \ E = \{(x, y) | y \in k\text{-NN of } x\}$

ε-ball graph: $G = (V, E) : \ V = X, \ E = \{(x, y) | d(x, y) < \epsilon\}$

What can we do with a graph?

Get distances: $d(x,y) = \{\text{Path distance from } x \text{ to } y \text{ in } G\}$

A distance-based clustering algorithm (or a dimension-reduction algorithm) can directly use these distances.

Get measure of similarity, e.g.

Adjacency matrix:

$$s(x,y) = \begin{cases} 1; & (x,y) \in E \\ 0; & (x,y) \notin E \end{cases}$$

Heat Kernel:

$$s(x,y) = \begin{cases} \exp\left(\frac{-d(x,y)^2}{2\sigma^2}\right); & (x,y) \in E \\ 0; & (x,y) \notin E \end{cases}$$

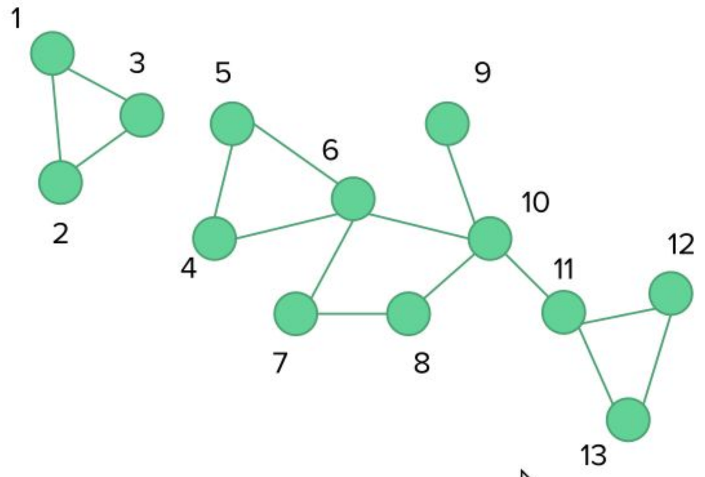
What can we do with a graph? Laplacian Eigenmaps

Let’s let W be a matrix made from one of the similarity measures.

We can define a diagonal matrix D such that $D_{ii} = \sum_j W_{ij}$

Then a graph Laplacian matrix L is simply D - W

$$L = \left(\begin{array}{c|c|c|cc} \sum_j w_{1j} & -w_{12} & -w_{13} & 0 & \cdots \\ \hline -w_{12} & \sum_j w_{2j} & -w_{23} & 0 & \cdots \\ \hline -w_{13} & -w_{23} & \sum_j w_{3j} & 0 & \cdots \\ \hline 0 & 0 & 0 & \ddots & \\ \vdots & \vdots & \vdots & & \end{array}\right)$$



We see that e.g. [1, 1, 1, 0, 0, … 0] is an eigenvector of L

What can we do with a graph? Laplacian Eigenmaps

1. Build an unweighted graph from the data (1.9)
2. Compute a similarity matrix $\mathbf{W}$ from the graph (1.10.2 or 1.10.3)
3. Compute the Laplacian $\mathbf{L}$ from $\mathbf{W}$.
4. Compute the bottom k eigenvectors $e_1, \ldots, e_k$ of $\mathbf{L}$ after discarding the 0-eigenvalue.
5. The new embedded data is the $n \times k$ matrix $X' = [e_1, \ldots, e_k]$

This is a **nonlinear dimension reduction** of our data
(btw, W doesn't necessarily have to come from a graph! It can be any similarity matrix)

Laplacian Eigenmap embedding + *k*-means = "Spectral Clustering"

cdr-cookbook

Distance is important!

Virtually all clustering algorithms rely on some “distance” or “similarity” between points.

A useful clustering relies on a good notion of distance between observations. The **numerical** distance/dissimilarity/similarity should somehow reflect the **real-world** distance/dissimilarity/similarity between observations.

- Manhattan distance (L1 norm)
- Euclidean distance (L2 norm)
- Lp norm for arbitrary p?
- Cosine similarity
- Something domain-specific
  - (e.g. Jukes-Cantor in genetics)
- Path distance (on a graph)
- Adjacency matrix
- ???

Distances be computed *after* a dimension reduction! **Dimension reduction can put your data in a space that more closely reflects the real-world distance between observations**

How do we cluster?

1. Put the data in some space where distances between points somehow reflect reality (often a dimension reduction)
2. Run some clustering algorithm

sometimes referred to as “Manifold Learning”

Many fancy “clustering algorithms” do both 1 and 2. But really step 2 is where the grouping into clusters happens.

Now let’s talk about 2.

Clustering methods

- Centroid methods
- Agglomerative/bottom-up hierarchical clustering
- Generative/distribution-based clustering

Centroid methods

These methods seek to find a set of points $c_1, \ldots, c_k$, and from these a collection of clusters $\{C_i\}_{i=1}^k$ with:

$$C_i = \{x \in X | \forall j \; d(x, c_i) \leq d(x, c_j)\},$$

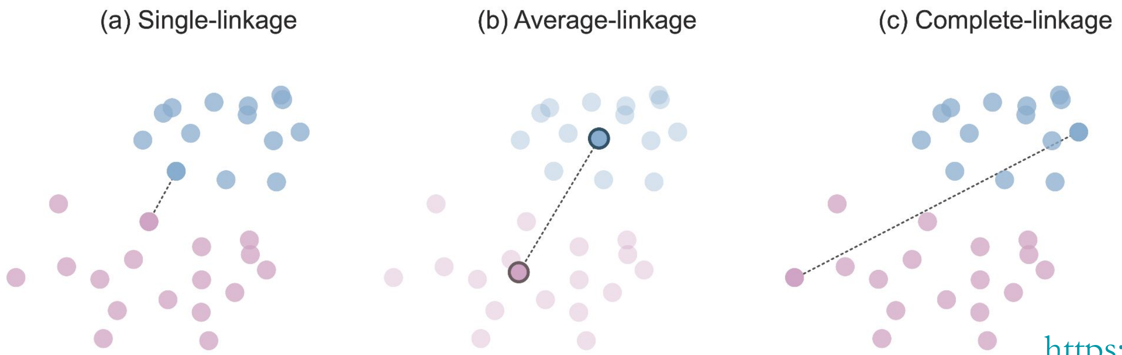
e.g. *k*-means: Euclidean distance is used between the points and the centroids, and run a randomized approximation algorithm to to compute the best cluster centers.

cdr-cookbook

Agglomerative/bottom-up hierarchical clustering

```
1. Start with a collection of clusters $\{C_i\}_i = \{\{x_i\}\}_i$
2. Until the collection of clusters is $\{X\}$:
    (a) Let $A, B$ be the clusters in $\{C_i\}_i$ minimizing $d(A, B)$
    (b) Merge $A$ and $B$. i.e. $\{C_i\}_i \leftarrow (\{C_i\}_i - \{A, B\}) \cup \{A \cup B\}$.
```

Different ways to measure distances between clusters:



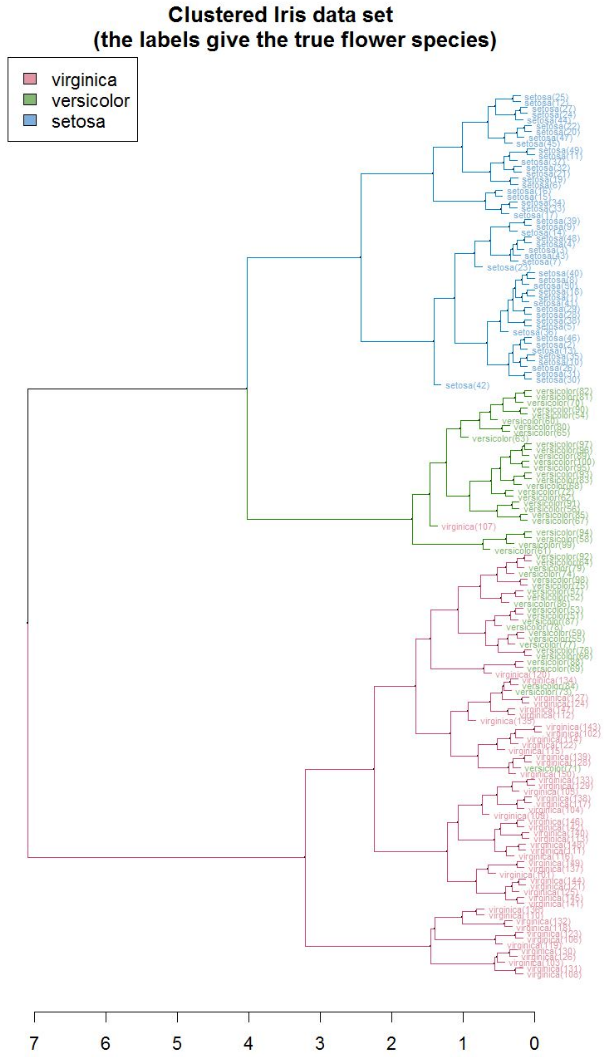
https://vdsbook.com/07-cluster

cdr-cookbook

Hierarchical clustering gives dendrograms!

Dendrograms show the hierarchy of clusters, with the “height” of a split/merge happening at the corresponding distance $d(A, B)$

They can be helpful for choosing *k*! A clustering can be considered “stable” if, for a wide range of “heights”, the clustering is the same.



https://commons.wikimedia.org/wiki/File:Iris_dendrogram.png

Generative/distribution-based clustering

In generative clustering methods, we define some mixture model M (which is a mixture over k different submodels $M_1, \ldots, M_k$) of how our data are generated:

$$x_1, \ldots, x_n \overset{iid}{\sim} M = \begin{cases} M_1 & \text{w.p. } \pi_1 \\ \vdots & \\ M_k & \text{w.p. } \pi_k \end{cases}.$$

After fitting the model to the data (i.e. choosing parameters θ_i for each M_i and priors π_i) to maximize likelihood:

$$\boldsymbol{\theta}, \boldsymbol{\pi} = \arg\max_{\boldsymbol{\theta}, \boldsymbol{\pi}} P(x_1, \ldots, x_n | M, \boldsymbol{\theta}, \boldsymbol{\pi}),$$

we assign each point to the part of the mixture model which assigns it the highest probability, hence getting that the ith cluster C_i is:

$$C_i = \{x \in X | \forall j \ \pi_i P(x|M_i, \boldsymbol{\theta}) \geq \pi_j P(x|M_j, \boldsymbol{\theta})\}$$

cdr-cookbook

e.g. Gaussian mixture: M_i is $N(\mu_i, \Sigma_i)$ and $\boldsymbol{\theta}$ is $\{\mu_i, \Sigma_i\}_i$. Optimize by EM

How to choose number of clusters?

- Stability of clusters to perturbations, random seeds, etc.
- Metrics of cluster quality
    - https://vdsbook.com/07-cluster#sec-cluster-quality
- Cross-validation
    - https://vdsbook.com/07-cluster#sec-cv-intro
- Elbow method
- Silhouette
- Length of dendrogram branch
- Real-world interpretation of clusters

DBSCAN

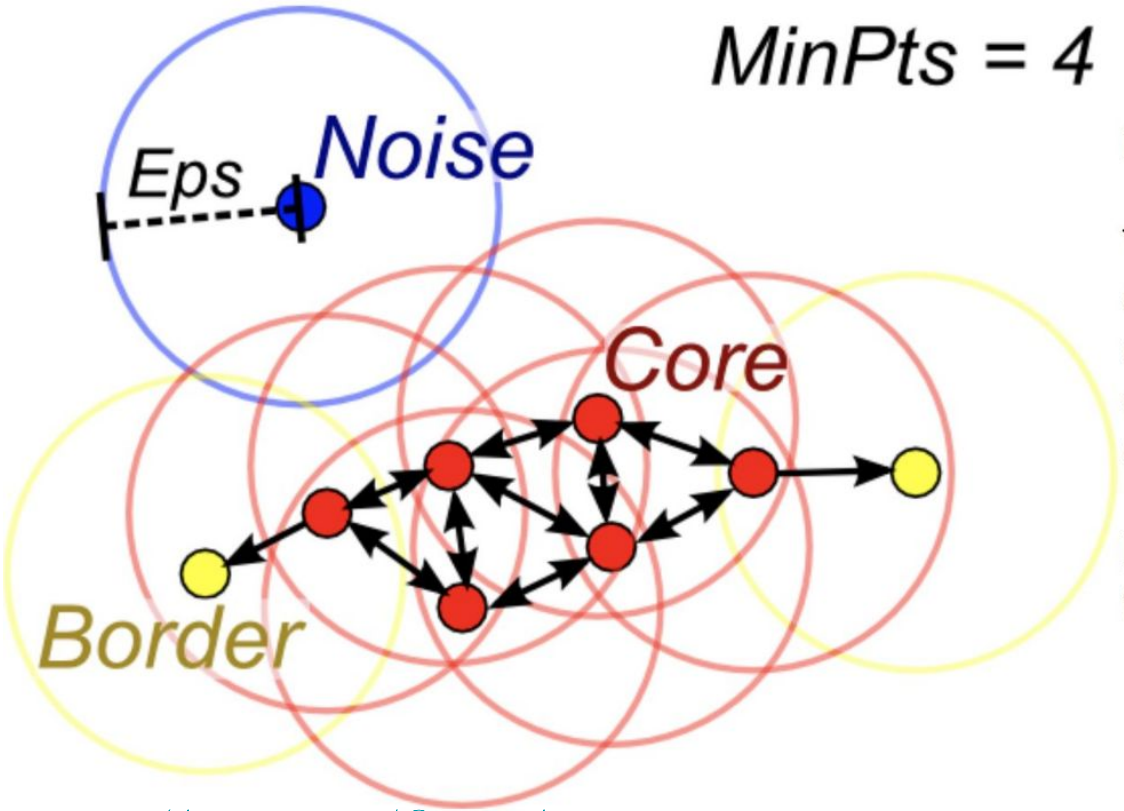
What if we don’t have to choose k?

DBSCAN is a density-based clustering

- Idea: group together points that are closely packed together (points with many nearby neighbors) while marking points that lie in low-density regions as outliers
- Choose (two?) parameters:
  - ε: how close points should be to each other to be in the same cluster (so we need a distance metric)
  - minPts: minimum number of points require to form a “dense region”

Source: https://medium.com/@elutins/dbscan-what-is-it-when-to-use-it-how-to-use-it-8bd506293818
The slides on DBSCAN thanks to Tiffany Tang

DBSCAN



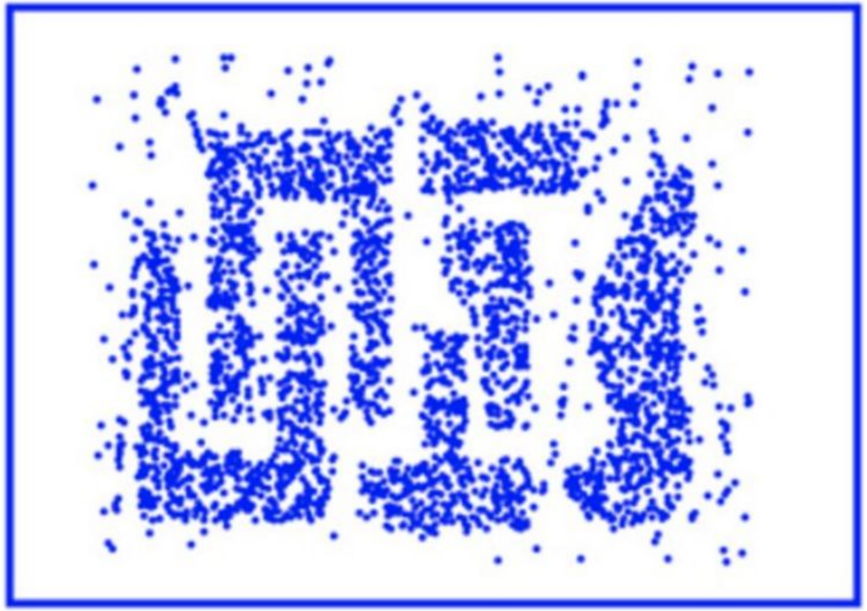
Red: Core Points

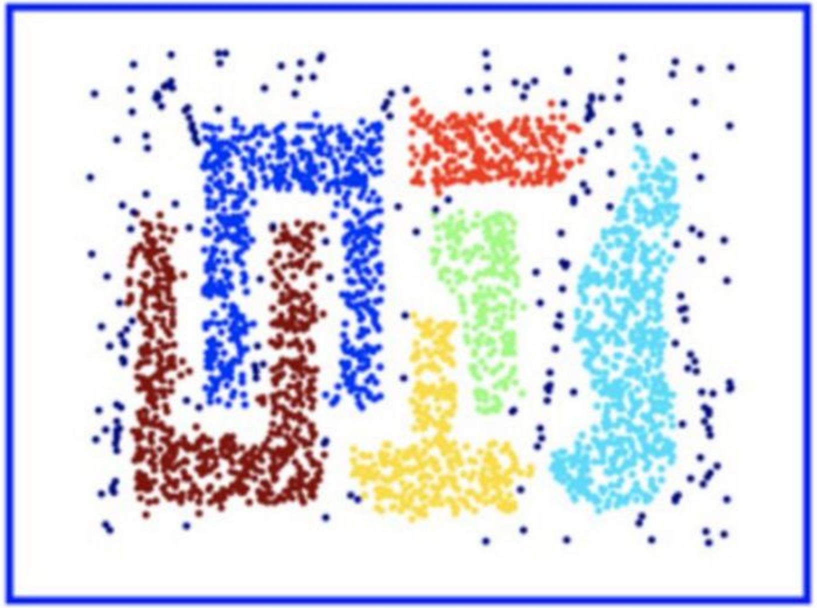
Yellow: Border points. Still part of the cluster because it's within epsilon of a core point, but not does not meet the min_points criteria

Blue: Noise point. Not assigned to a cluster

Source: https://medium.com/@elutins/dbscan-what-is-it-when-to-use-it-how-to-use-it-8bd506293818

DBSCAN





Source: https://medium.com/@elutins/dbscan-what-is-it-when-to-use-it-how-to-use-it-8bd506293818

DBSCAN

Advantages

- Don’t have to choose K (but depends on choice of ε and minPts)
- Great for spatial data
- Great at separating clusters of similar densities that are well separated
- Robust to outliers
- Flexible to arbitrarily-shaped clusters

Disadvantages

- If the data and scale are not well understood, choosing a meaningful distance threshold and minPts can be difficult
- Struggles when clusters are of varying densities since (ε , minPts) cannot be chosen appropriately for all clusters
- Curse of dimensionality when distance metric is Euclidean distance
- Algorithm depends on ordering of points; border points that are reachable from more than one cluster can be part of either cluster, depending on the order the data are processed

Clustering in Python

scikit-learn does pretty much everything

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

https://scikit-learn.org/stable/unsupervised_learning.html

But it won’t do the thinking for you!