Clustering

24 March 2026

Alex Lyman

Supervised vs Unsupervised Learning

Types of Machine Learning - Supervised

- The machine learns from **labeled data** (input-output pairs). The algorithm is provided with the "correct answers" during training.
- Goal - to map input variables to an output variable effectively enough to predict the output for new, unseen data.
- **Examples -** Spam filtering, house price prediction, image classification. (Everything we've done this semester)

Types of Machine Learning - Unsupervised

- The machine learns from **unlabeled data**. The algorithm must explore the data to find structure or patterns on its own without guidance.
- Goal -To model the underlying structure or distribution in the data to learn more about it (e.g., grouping similar items).
- Examples – Clustering, segmentation, dimensionality reduction, anomaly detection.

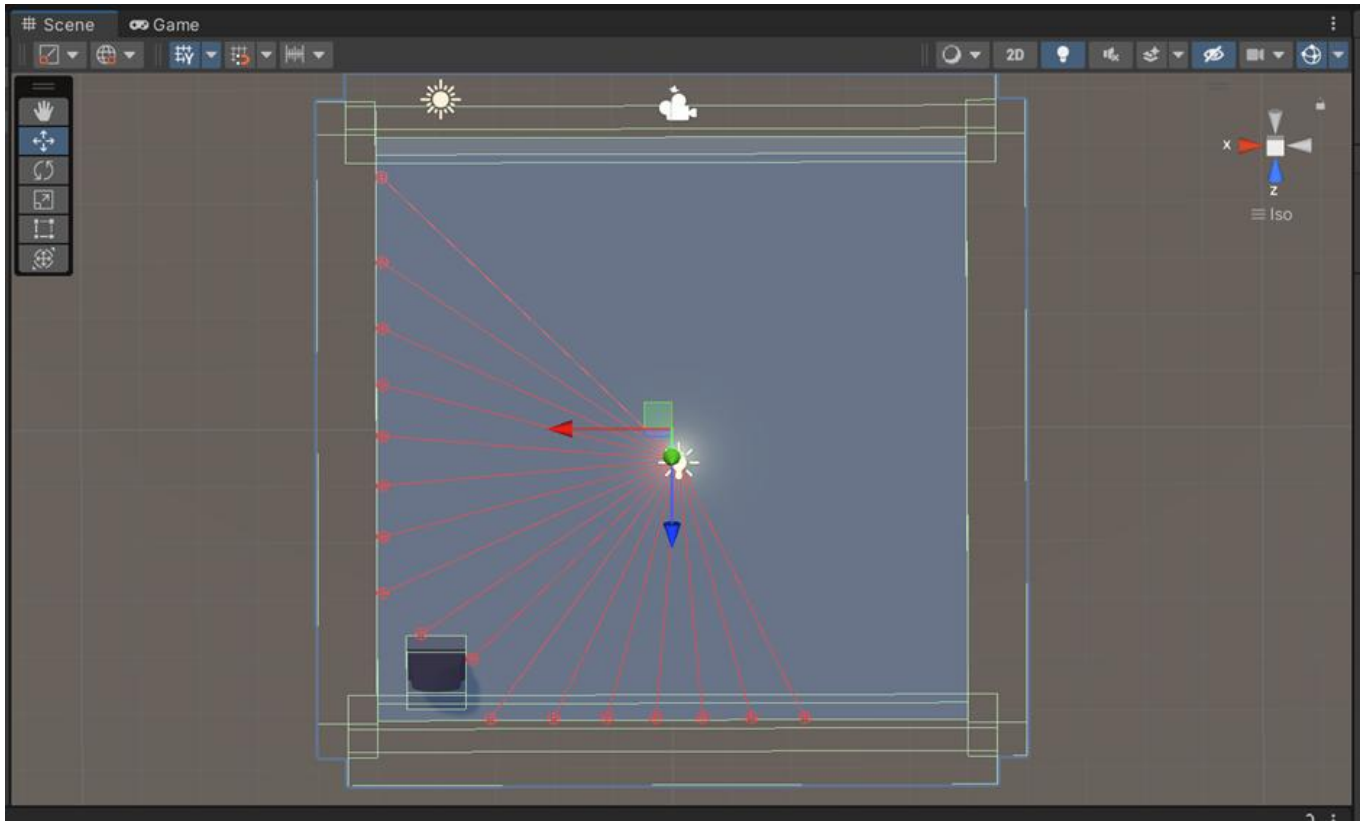
Types of Machine Learning - Reinforcement

- An **agent** learns to make decisions by interacting with an environment. It learns through a trial-and-error process based on feedback (rewards or penalties).
- Goal - To maximize the cumulative reward over time to achieve a specific goal.
- Examples - Game playing (e.g., Chess, Go), robotics, self-driving cars.

Reinforcement Learning 2

In this task the **agent** is a little ghost I modeled in blender.
Actions:

- Rotate Right
- Rotate Left
- Don't Rotate
- Move Forward
- Move Backwards
- Don't Move



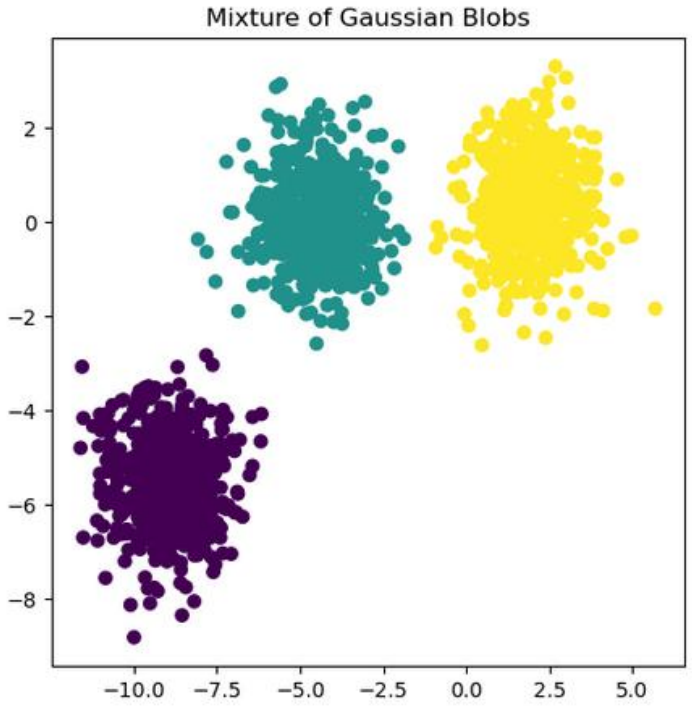
His **observations** are the red rays in the image. They are how the agent 'sees' the world around him. He can differentiate between the goal and the boundaries where he might fall off the edge.

His rewards are simple. At each timestep, he gets a very small existential penalty (a teeny negative reward) to encourage him to take an action. If he falls off the platform, he gets a large penalty and gets reset. If he hits the goal, he gets a large reward and the goal resets somewhere else for him to find.

Clustering Overview

Clustering

- In unsupervised learning you are given a data set with no output classifications (labels)
- Clustering is an important type of unsupervised learning



- The goal in clustering is to find "natural" clusters (classes) into which the data can be divided – a particular breakdown into clusters is a clustering (aka grouping, partition)
- How many clusters should there be (k)? – Either user-defined, discovered by trial and error, or automatically derived
- Example: Taxonomy of the species – one correct answer?
- Generalization – After clustering, when given a novel instance, we just assign it to the most similar cluster

Clustering

- How do we decide which instances should be in which cluster?
- Typically put data which is "similar" into the same cluster
  - Similarity is measured with some distance metric
- Also try to maximize between-class dissimilarity
- Seek balance of within-class similarity and between-class dissimilarity
- Similarity Metrics
  - Euclidean Distance most common for real valued instances
  - Can use (1,0) distance for nominal and unknowns like with *k*-NN
  - Can create arbitrary distance metrics based on the task
  - Important to normalize the features

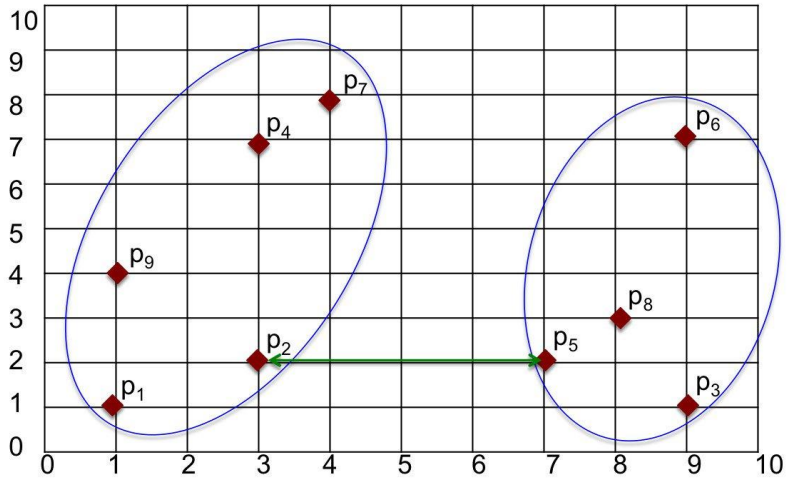
Outlier Handling

- Outliers
  - noise, or
  - correct, but unusual data
- Approaches to handle them
  - become their own cluster
    - Problematic, e.g. when *k* is pre-defined (How about *k* = 2 above)
    - If *k* = 3 above then it could be its own cluster, rarely used, but at least it doesn't mess up the other clusters
    - Could remove clusters with 1 or few elements as a post-process step
  - Absorb into the closest cluster
    - Can significantly adjust cluster radius, and cause it to absorb other close clusters, etc
  - Remove with pre-processing step
    - Detection non-trivial – when is it really an outlier?
    - Unsupervised Outlier detection algorithms available

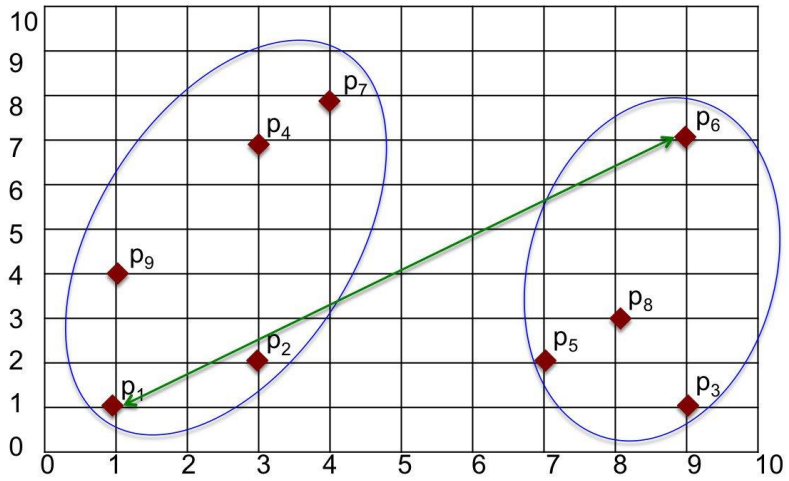
Distances Between Clusters

- Easy to measure distance between instances (elements, points), but how about the distance of an instance to another cluster or the distance between 2 clusters
- Can represent a cluster with
  - Centroid – cluster mean
    - Then just measure distance to the centroid
  - Medoid – an actual instance which is most typical of the cluster (e.g. Medoid is point which would make the average distance from it to the other points the smallest)
- Other common distances between two Clusters *A* and *B*
  - Single link – Smallest distance between any 2 points in *A* and *B*
  - Complete link – Largest distance between any 2 points in *A* and *B*
  - Average link – Average distance between points in *A* and points in *B*

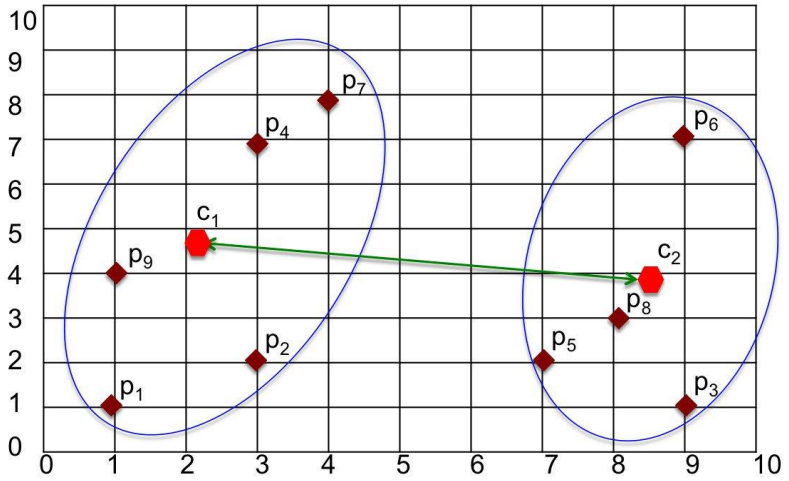
Distances Between Clusters



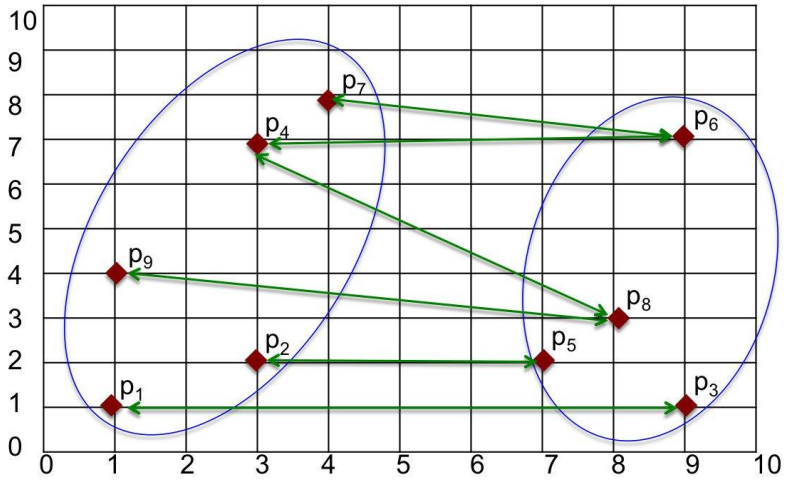
Single Link



Complete(max) Link



Centroid Link



Average Link

Hierarchical and Partitional Clustering

- Two most common high-level approaches
- Hierarchical clustering is broken into two approaches
  - Agglomerative: Each instance is initially its own cluster. Most similar instance/clusters are then progressively combined until all instances are in one cluster. Each level of the hierarchy is a different clustering/grouping of clusters. (e.g. HAC)
  - Divisive: Start with all instances as one cluster and progressively divide until all instances are their own cluster.
  - You then decide which clustering you want to output.

- With partitional clustering the algorithm creates one clustering of the data (with multiple clusters), typically by minimizing some objective function (e.g. K-means)
  - Note that you could run the partitional algorithm again in a recursive fashion on all the new clusters if you want to build a hierarchy

K-Means Clustering

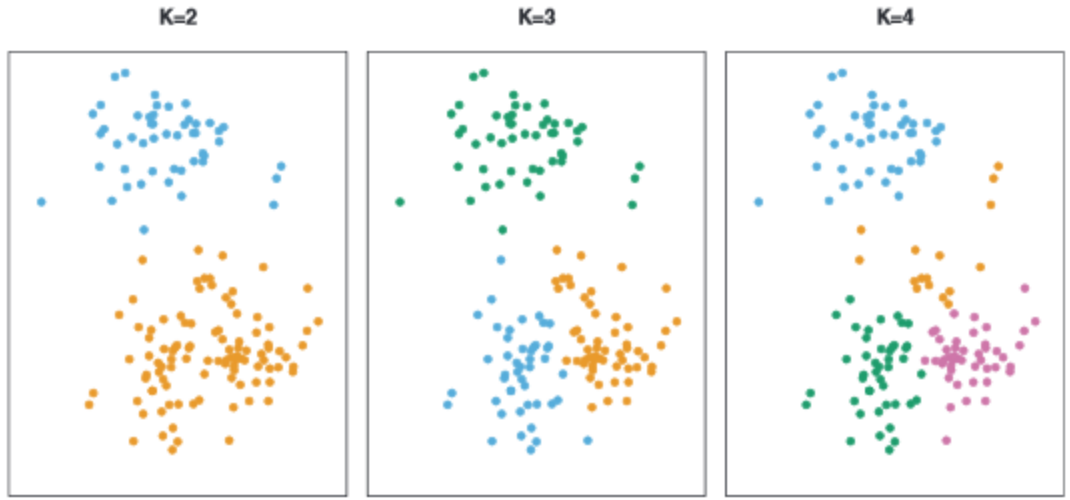
K-Means

- Perhaps the most well-known clustering algorithm
  - Partitioning algorithm
  - **Must choose a *k* beforehand**

1. Randomly choose *k* instances from the data set to be the initial *k* centroids
2. Repeat until no (or negligible) more changes occur
   a) Group each instance with its closest centroid
   b) Recalculate the centroid based on its new cluster

K-Means

- **Must choose a *k* beforehand**
- Thus, typically try a spread of different *k*'s (e.g. 2-10) and then compare results to see which made the best clustering
- Could use cluster validity metrics (e.g. Silhouette) to help in the decision

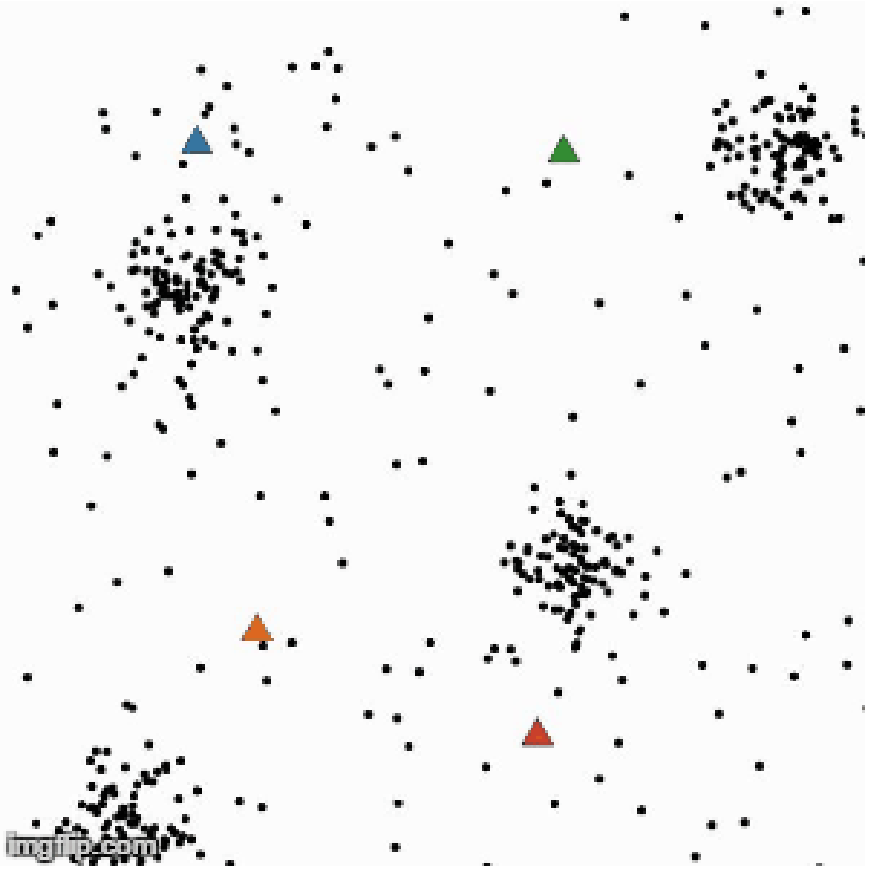


- Time complexity is O(*mkn*) where *m* is # of iterations and space is O(*n*), both much better than HAC time and space (n^3 and n^2)

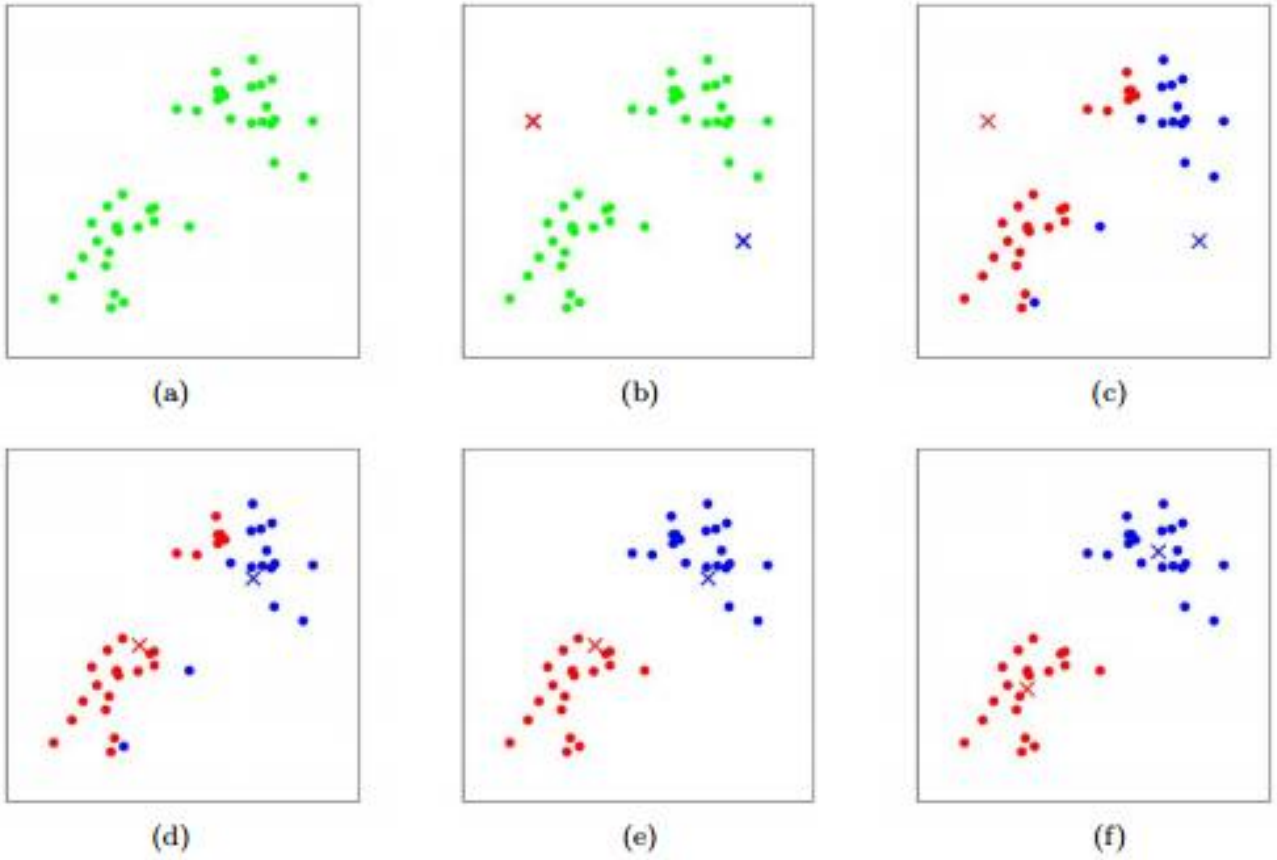
K-Means Visualization

1. Randomly choose *k* instances from the data set to be the initial *k* centroids

2. Repeat until no (or negligible) changes occur
    a) Group each instance with its closest centroid
    b) Recalculate the centroid based on its new cluster



K-means Example

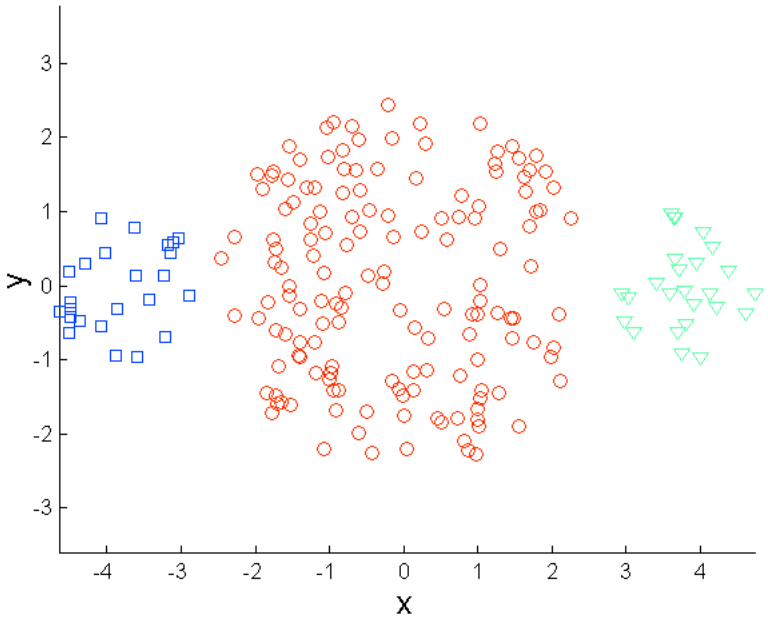


Limitations of K-means

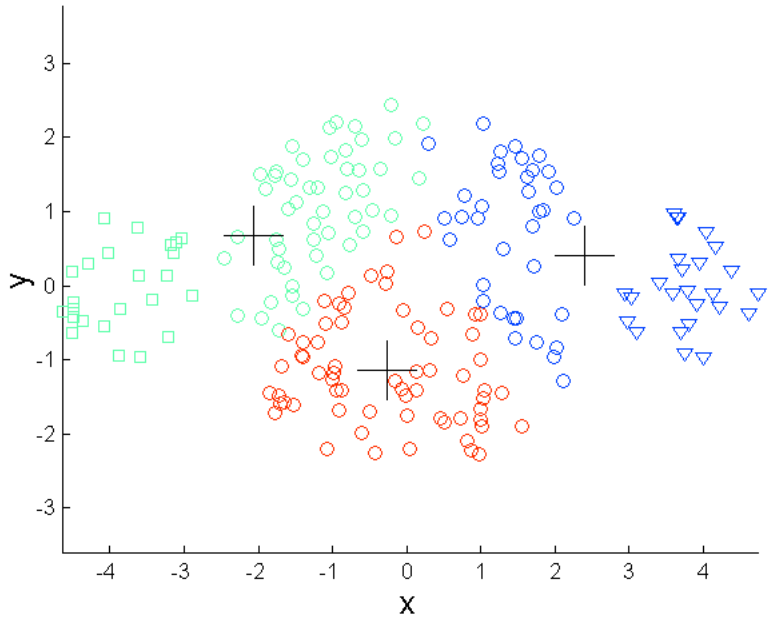
- K-means has problems when clusters are of differing
  - Sizes
  - Densities
  - Non-globular shapes

- K-means has problems when the data contains outliers.

Limitations of K-means: Differing Sizes

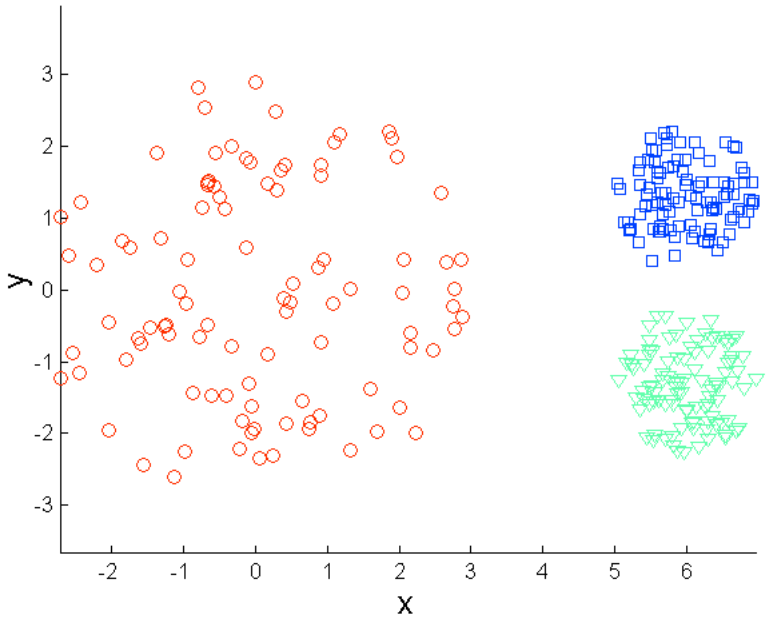


Original Points

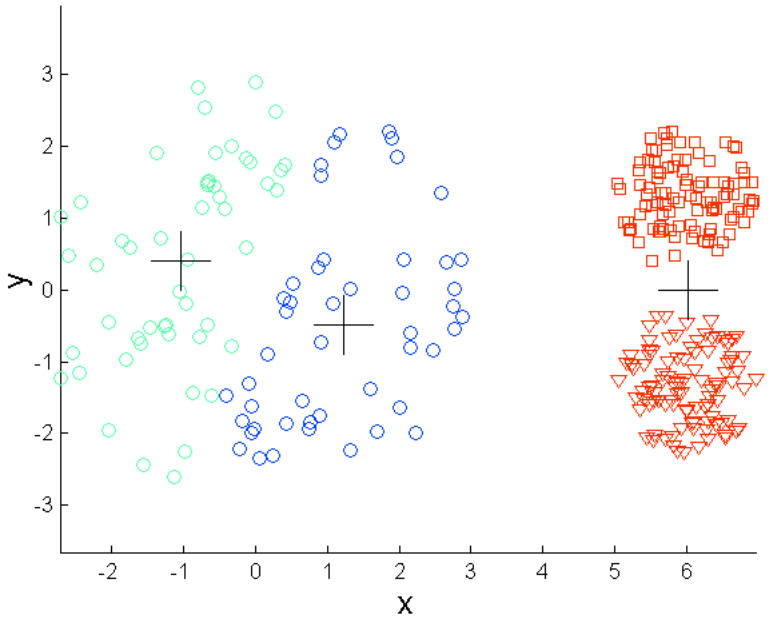


K-means (3 Clusters)

Limitations of K-means: Differing Density

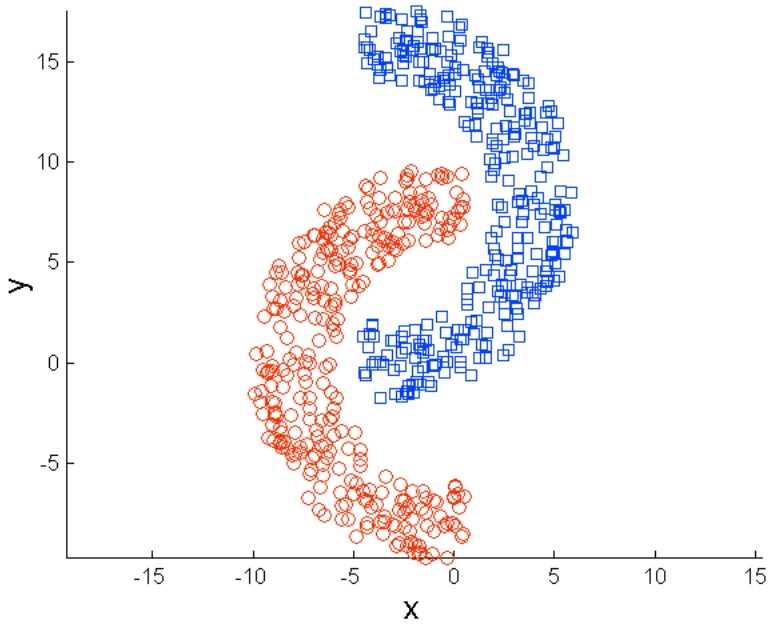


Original Points

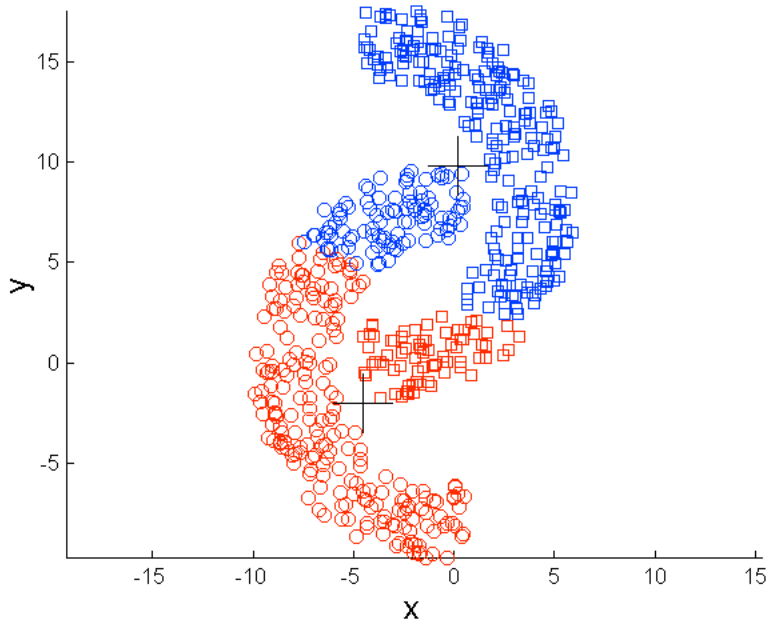


K-means (3 Clusters)

Limitations of K-means: Non-globular Shapes

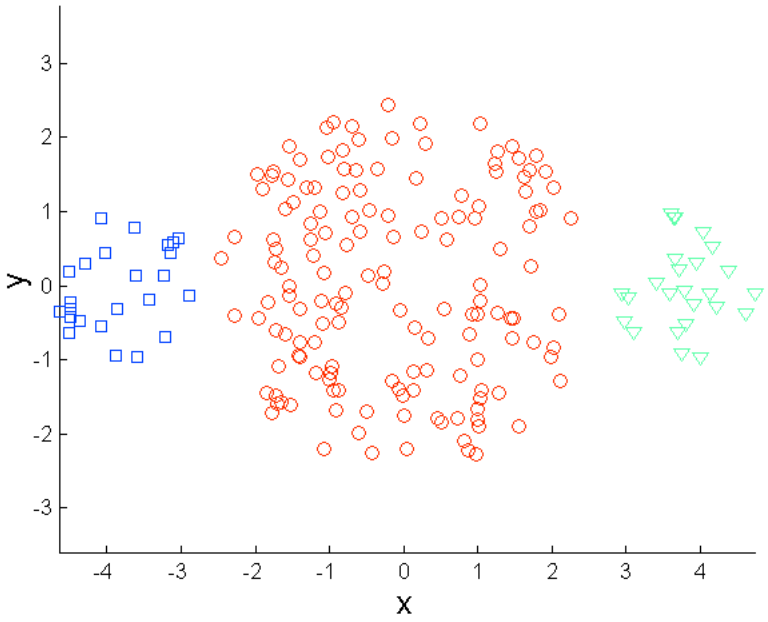


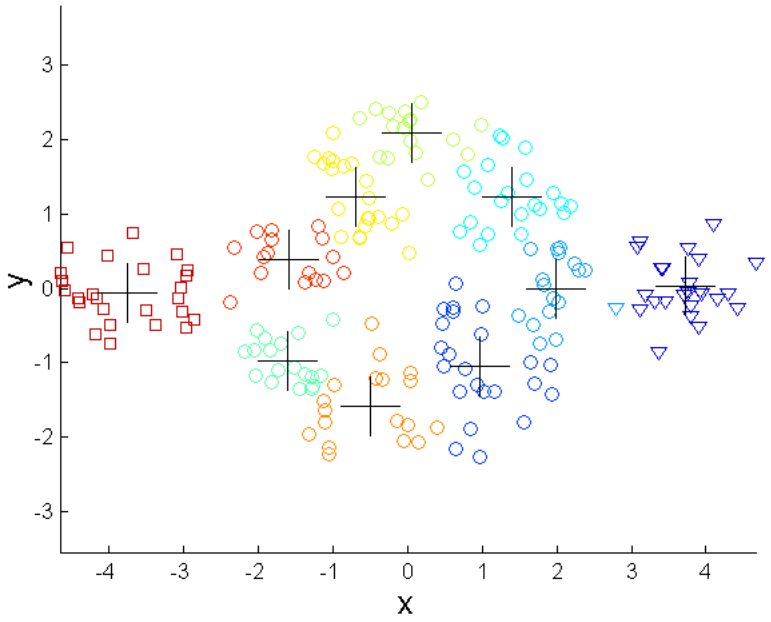
Original Points



K-means (2 Clusters)

Overcoming K-means Limitations





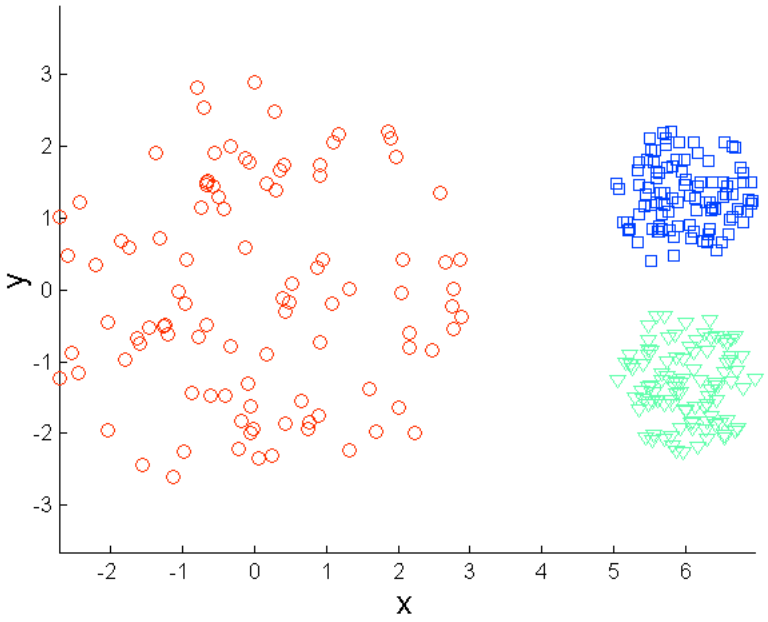
Original Points

K-means Clusters

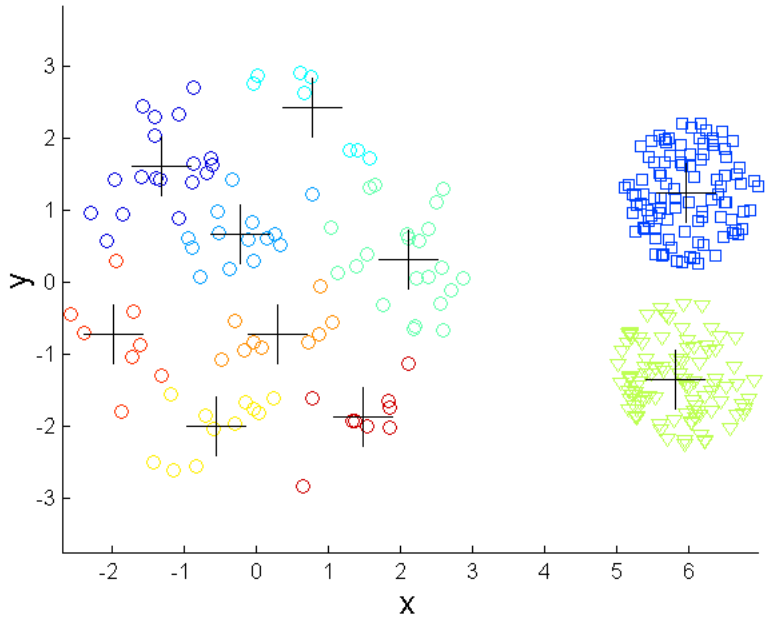
One solution is to use many clusters.

Find parts of clusters, but need to put together.

Overcoming K-means Limitations

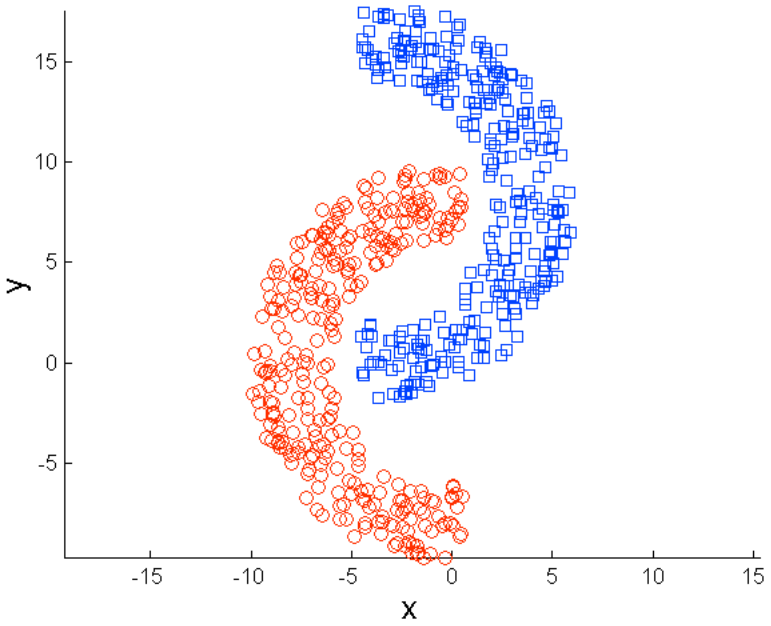


Original Points

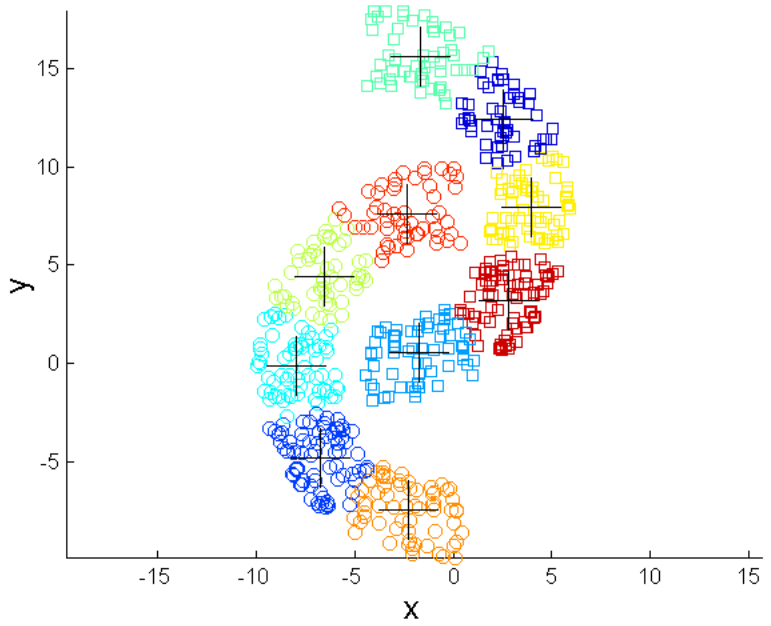


K-means Clusters

Overcoming K-means Limitations



Original Points



K-means Clusters

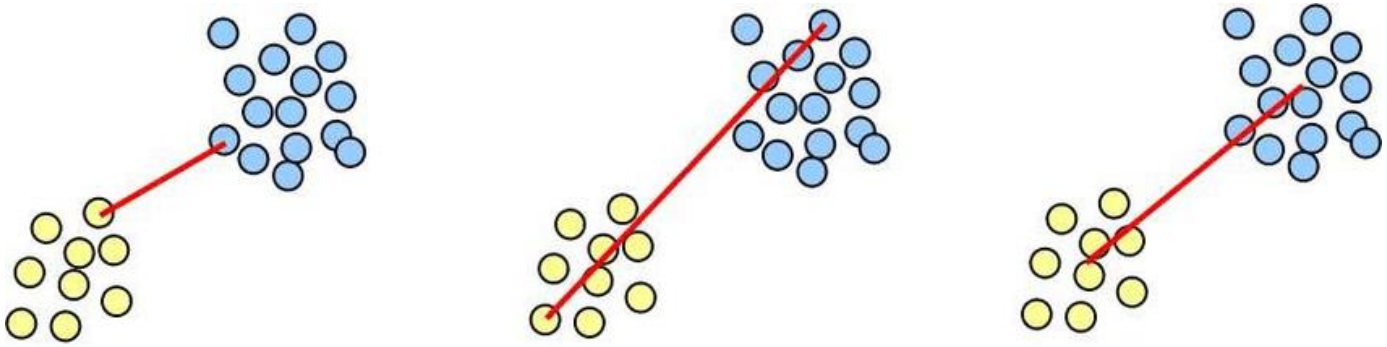
Hierarchical Clustering

Hierarchical Clustering

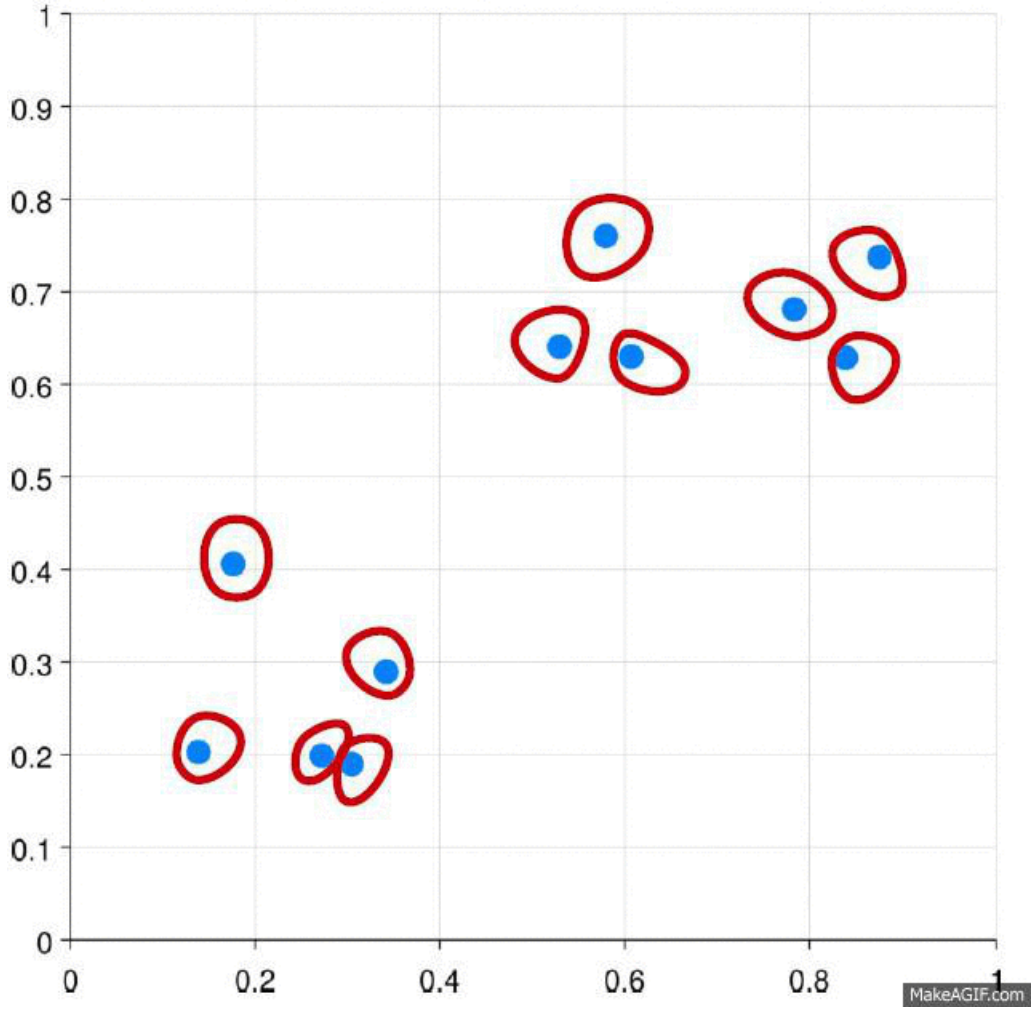
- The Limitation of K-Means:
  - K-Means forces us to guess the number of clusters (k) before we even look at the data.
  - It assumes all clusters are "flat" and exist on the exact same level.
- The Hierarchical Approach:
  - What if our data has nested relationships? (e.g., Clusters inside of other clusters).
  - Instead of forcing data into flat groups, we build a tree (called a Dendrogram) that records the relationships between all data points at every level of distance.
- The Ultimate Advantage:
  - No need to guess k! We let the algorithm build the entire tree first, and then we decide where to "cut" the branches later depending on how much detail we want.

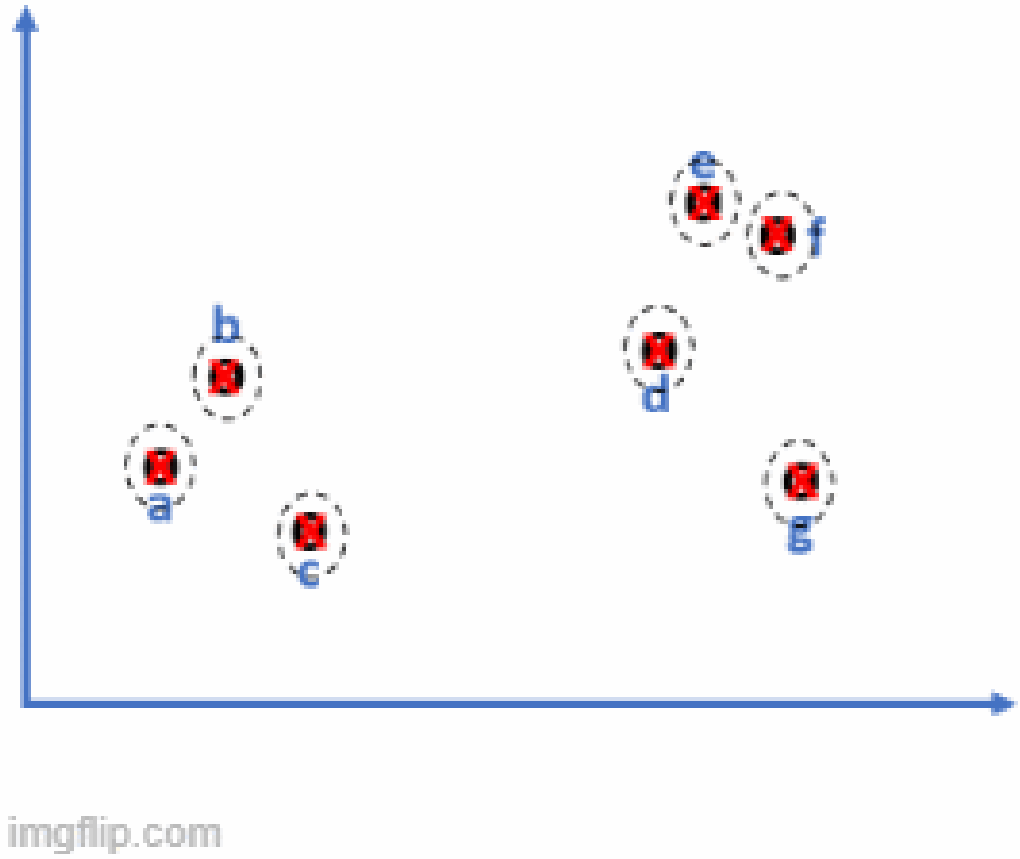
Hierarchical Agglomerative Clustering (HAC)

- Input is an $n \times n$ adjacency matrix giving the distance between each pair of instances
- Initialize each instance to be its own cluster
- Repeat until there is just one cluster containing all instances
  - Merge the two "closest" remaining clusters into one cluster
- HAC varies based on "Closeness definition"
  - single, complete, and average link distances common

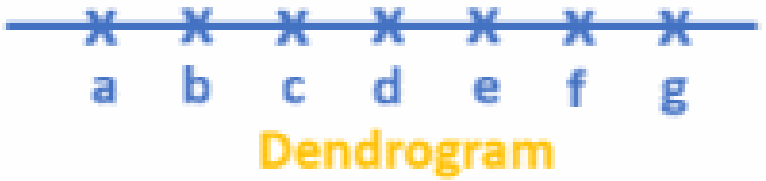


Hierarchical Clustering Visualization



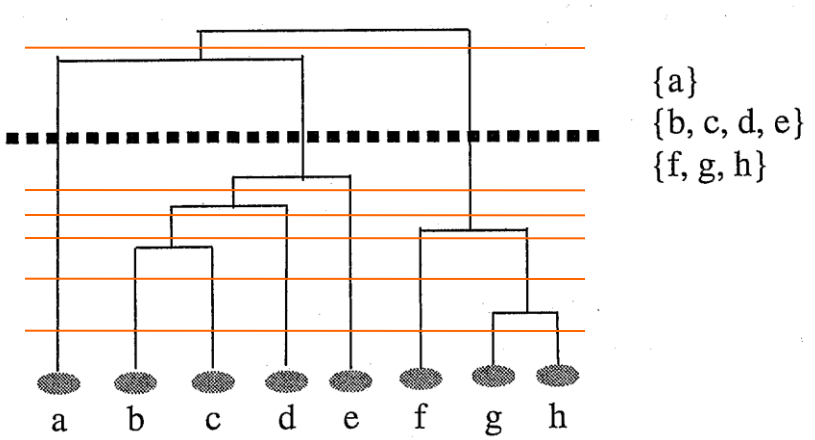


imgflip.com



Dendrogram

Which cluster level to choose?

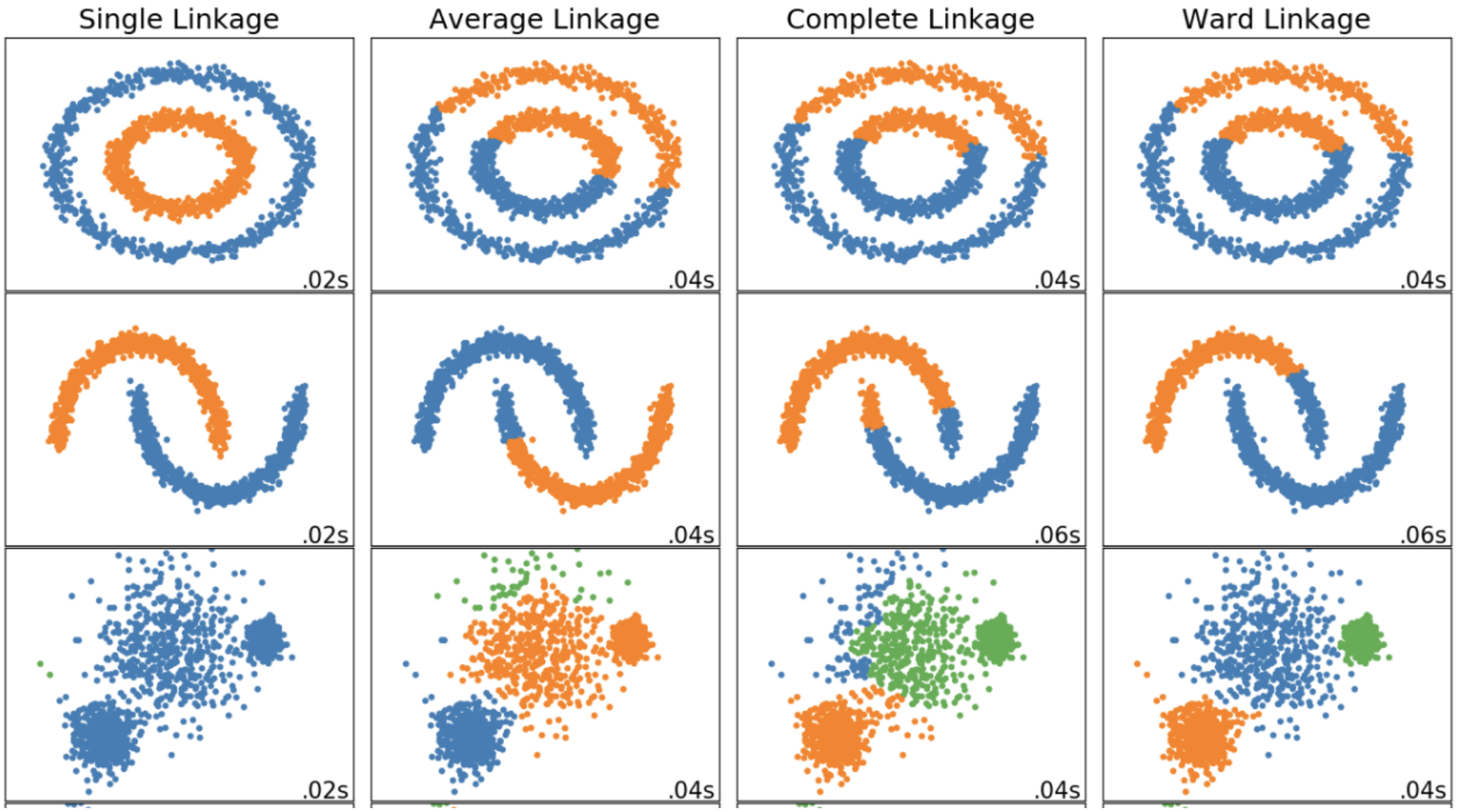


- Depends on goals
  - May know beforehand how many clusters you want - or at least a range (e.g. 2-10)
  - Could analyze the dendrogram and data after the full clustering to decide which subclustering level is most appropriate for the task at hand
  - Could use automated *cluster validity* metrics to help
- Could do stopping criteria during clustering

HAC Linkage Comparisons

- Single link – (nearest neighbor) can lead to long chained clusters where some points are quite far from each other
- Complete link – (farthest neighbor) finds more compact clusters
- Average link – Used less because have to re-compute the average each time
- Ward linkage – Can often be the most suitable method for quantitative features
- Once you have distances between clusters, always merge the closest clusters in terms of the distance

Linkage Methods



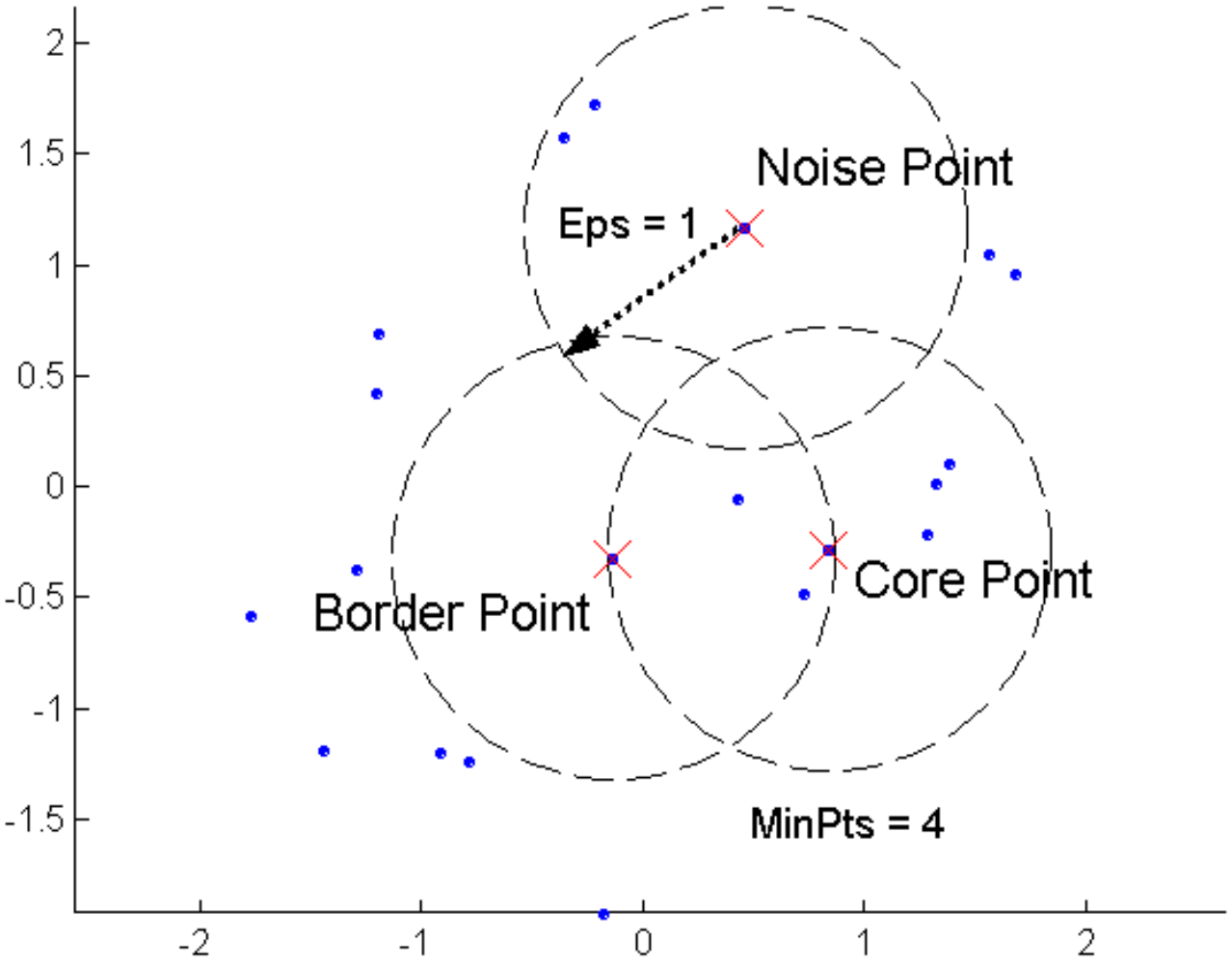
HAC Summary

- Complexity – Relatively expensive algorithm
  - n^2 space for the adjacency matrix
  - mn^2 time for the execution where m is the number of algorithm iterations, since we have to compute new distances at each iteration. m is usually $\approx n$ making the total time n^3 (can be $n^2 \log n$ with priority queue for distance matrix, etc.)
  - All k ($\approx n$) clusterings returned in one run. No restart for different k values. Must then decide which clustering you want.

DBSCAN

DBSCAN

- DBSCAN is a density-based algorithm.
  - Density = number of points within a specified radius (Eps)
  - A point is a core point if it has more than a specified number of points (MinPts) within Eps
    - These are points that are at the interior of a cluster
  - A border point has fewer than MinPts within Eps, but is in the neighborhood of a core point
  - A noise point is any point that is not a core point or a border point.



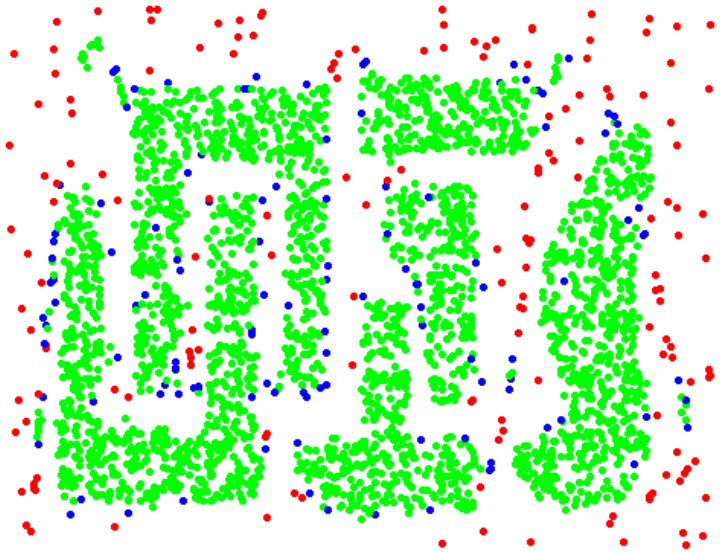
DBSCAN Algorithm

- Eliminate noise points
- Perform clustering on the remaining points

```
For every unvisited point P in the dataset:
    Mark P as visited.
    Find all neighbors within Epsilon (ε).

    If neighbors < Min_Points:
        Label P as NOISE.

    Else:
        Label P as a CORE point and create a NEW CLUSTER.
        Add P to the cluster.

        // The Expansion Queue
        For every neighbor N of P:
            If N was previously labeled NOISE:
                Relabel N as a BORDER point.
                Add N to the cluster.

            If N is unvisited:
                Mark N as visited.
                Find all neighbors of N within Epsilon (ε).

                If N's neighbors >= Min_Points:
                    Label N as a CORE point.
                    Add N's neighbors to our expansion queue!

                Add N to the cluster.
```

DBSCAN: Core, Border and Noise Points

Original Points

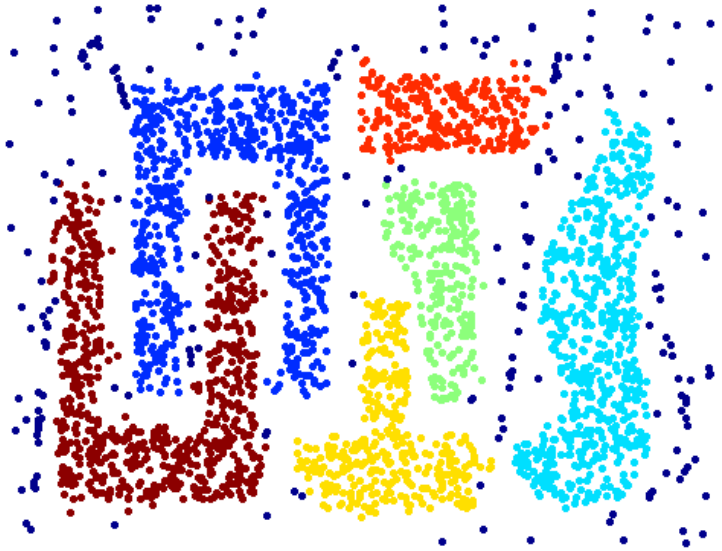


Point types: core, border and noise

Eps = 10, MinPts = 4

When DBSCAN Works Well

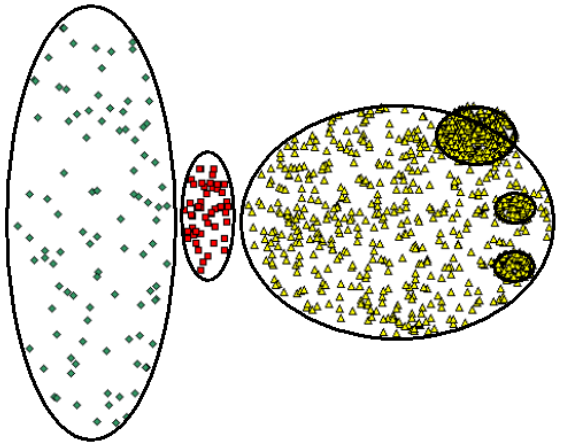
Original Points



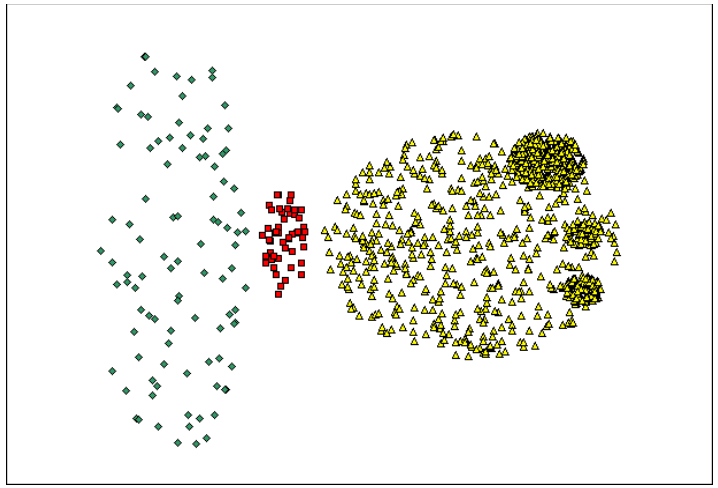
Clusters

- Resistant to Noise
- Can handle clusters of different shapes and sizes

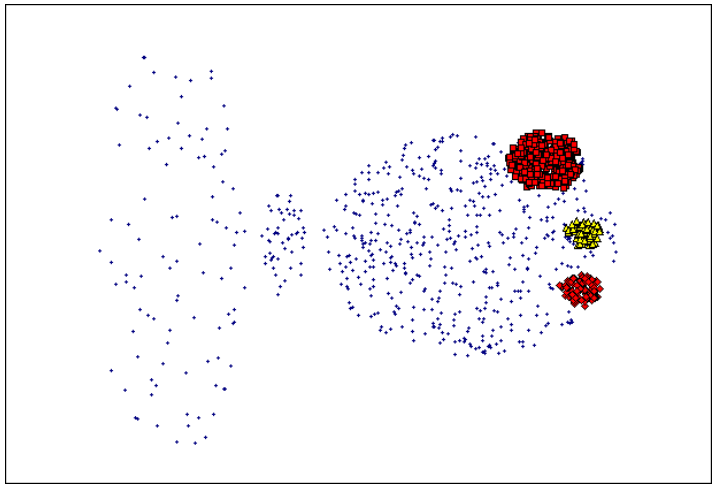
When DBSCAN Does NOT Work Well



Original Points



(MinPts=4, Eps=9.75).



(MinPts=4, Eps=9.92)

- Varying densities
- High-dimensional data

Break

Clustering Metrics

Silhouette

Cluster Validity Metrics - Compactness

- One good goal is *compactness* – members of a cluster are all similar and close together
  - One measure of compactness of a cluster is the sum of squares distance from each point to its cluster centroid (aka SSE)

$$Comp(C) = \sum_{i=1}^{|X_c|} (\mathbf{c} - \mathbf{x}_i)^2$$

  - where **c** is the centroid of a cluster *C*, made up of instances X_c. Lower is better.
  - The overall compactness of a particular clustering is the sum of the compactness of the individual clusters
  - Gives us a numeric way to compare different clusterings by seeking clusterings which minimize the compactness metric
  - Sometimes called *Inertia*

Cluster Validity Metrics - Separability

- Another good goal is *separability* – members of one cluster are sufficiently different from members of another cluster (cluster dissimilarity)
  - One measure of the separability of two clusters is their squared distance. The bigger the distance the better.
  - $dist_{ij} = (\mathbf{c}_i - \mathbf{c}_j)^2$ where $\mathbf{c}_i$ and $\mathbf{c}_j$ are two cluster centroids
  - For a clustering which cluster distances should we compare?
  - For each cluster we add in the distance to its closest neighbor cluster

$$Separability = \sum_{i=1}^{|C|} \min_j dist_{ij}(\mathbf{c}_i, \mathbf{c}_j)$$

  - We would like to find clusterings where separability is maximized
- However, separability is usually maximized when there are are very few clusters
  - squared distance amplifies larger distances

Silhouette

- We want techniques that find a balance between inter-cluster similarity and intra-cluster dissimilarity
- Silhouette is one good popular approach
- Scores any clustering with an arbitrary number of unique clusters. Clustering can come from any clustering algorithm.
- $a(i)$ = average dissimilarity of instance i to all other instances in the cluster to which i is assigned – Want it small
  - Dissimilarity could be Euclidian distance, etc.
- $b(i)$ = the smallest average dissimilarity of instance i to instances in other clusters – Want it large
- $b(i)$ is smallest for the best different cluster that i could be assigned to – the best cluster that you would move i to if needed

Silhouette

***a*(*i*)** = average dissimilarity of instance *i* to all other instances in the cluster to which *i* is assigned
– Want it small

***b*(*i*)** = the smallest average dissimilarity of instance *i* to instances in other clusters
– Want it large

$$s(i) = \frac{b(i) - a(i)}{\max\{a(i), b(i)\}}$$

$$s(i) = \begin{cases} 1 - a(i)/b(i) & \text{if } a(i) < b(i) \\ 0 & \text{if } a(i) = b(i) \\ b(i)/a(i) - 1 & \text{if } a(i) > b(i) \end{cases}$$

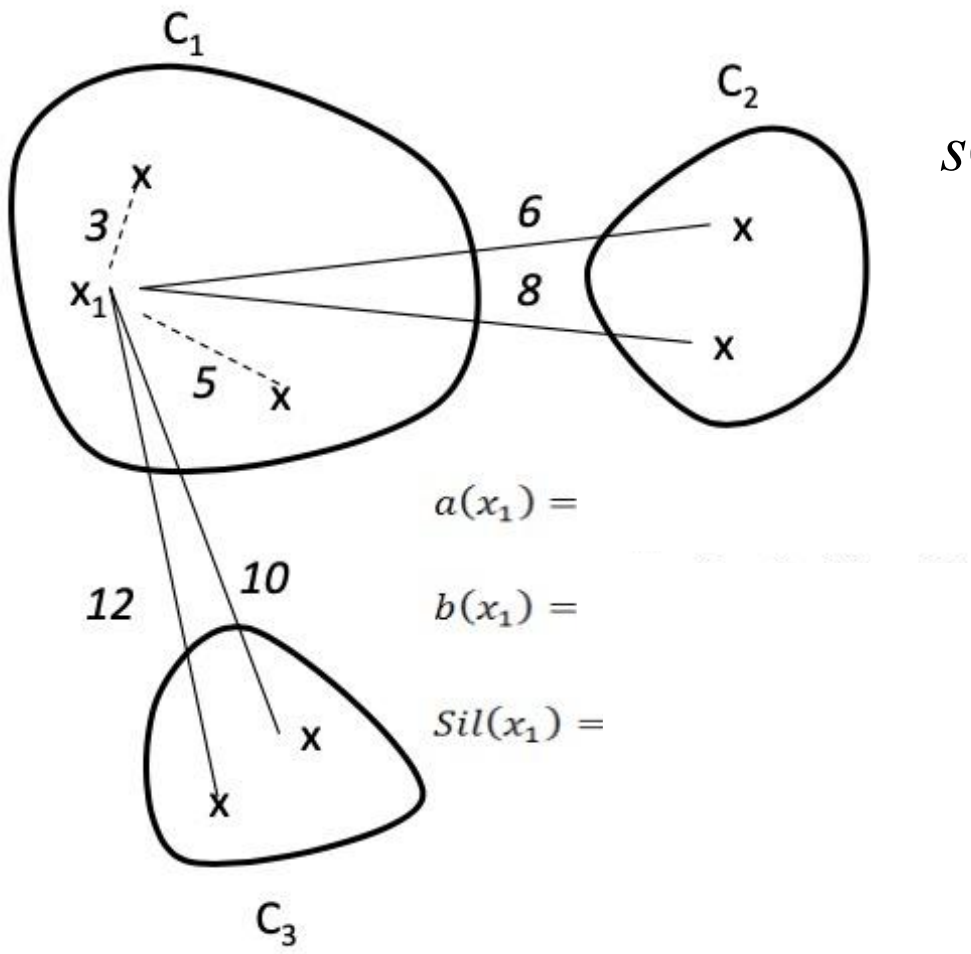
$$-1 \le s(i) \le 1$$

Silhouette

$$s(i) = \frac{b(i) - a(i)}{\max\{a(i), b(i)\}}$$

***a*(*i*)** = average dissimilarity of instance *i* to all other instances in the cluster to which *i* is assigned
– Want it small

***b*(*i*)** = the smallest average dissimilarity of instance *i* to instances in other clusters
– Want it large



$$s(i) = \begin{cases} 1 - a(i)/b(i) & \text{if } a(i) < b(i) \\ 0 & \text{if } a(i) = b(i) \\ b(i)/a(i) - 1 & \text{if } a(i) > b(i) \end{cases}$$

Silhouette

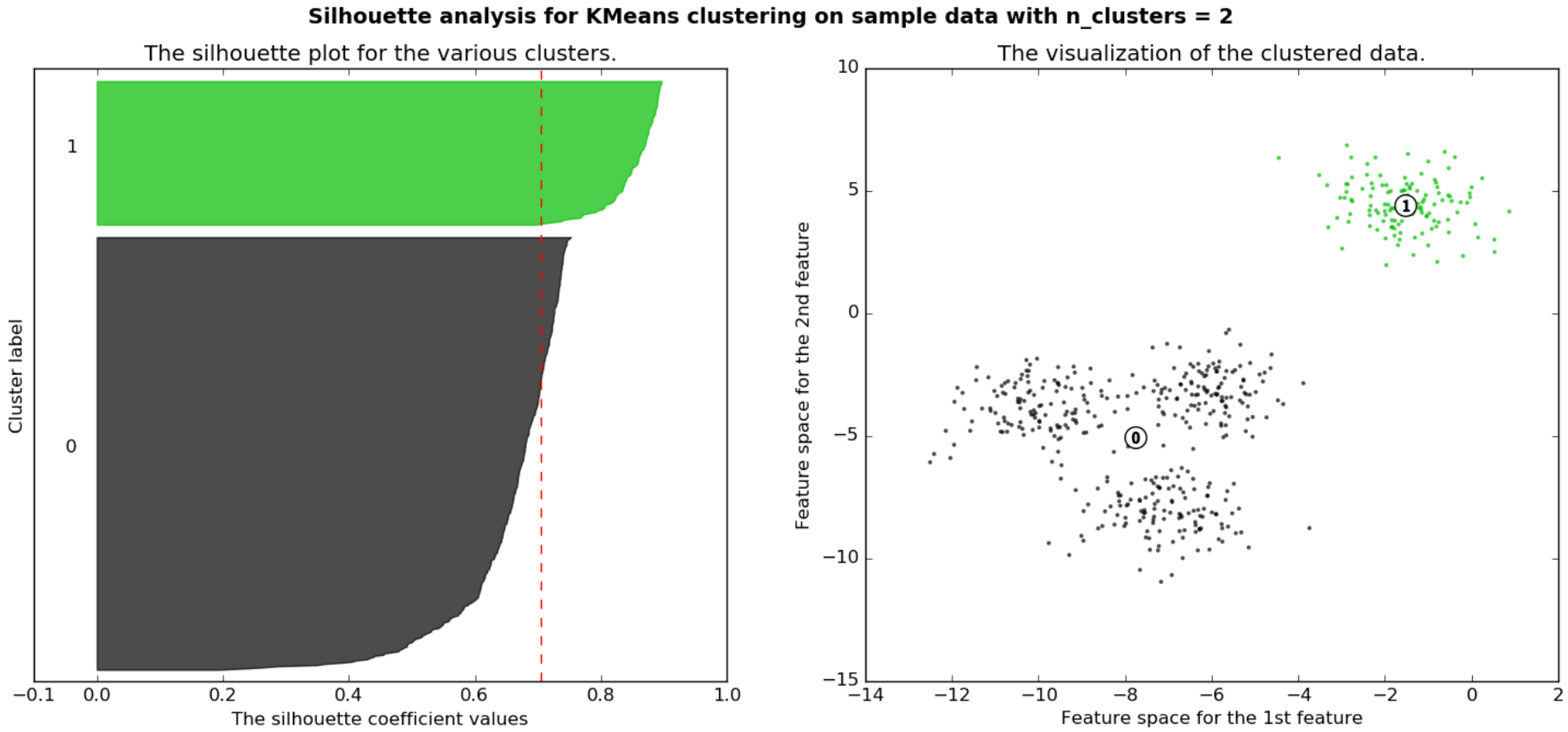
$$s(i) = \begin{cases} 1 - a(i)/b(i) & \text{if } a(i) < b(i) \\ 0 & \text{if } a(i) = b(i) \\ b(i)/a(i) - 1 & \text{if } a(i) > b(i) \end{cases} \qquad s(i) = \frac{b(i) - a(i)}{\max\{a(i), b(i)\}}$$

$$-1 \le s(i) \le 1$$

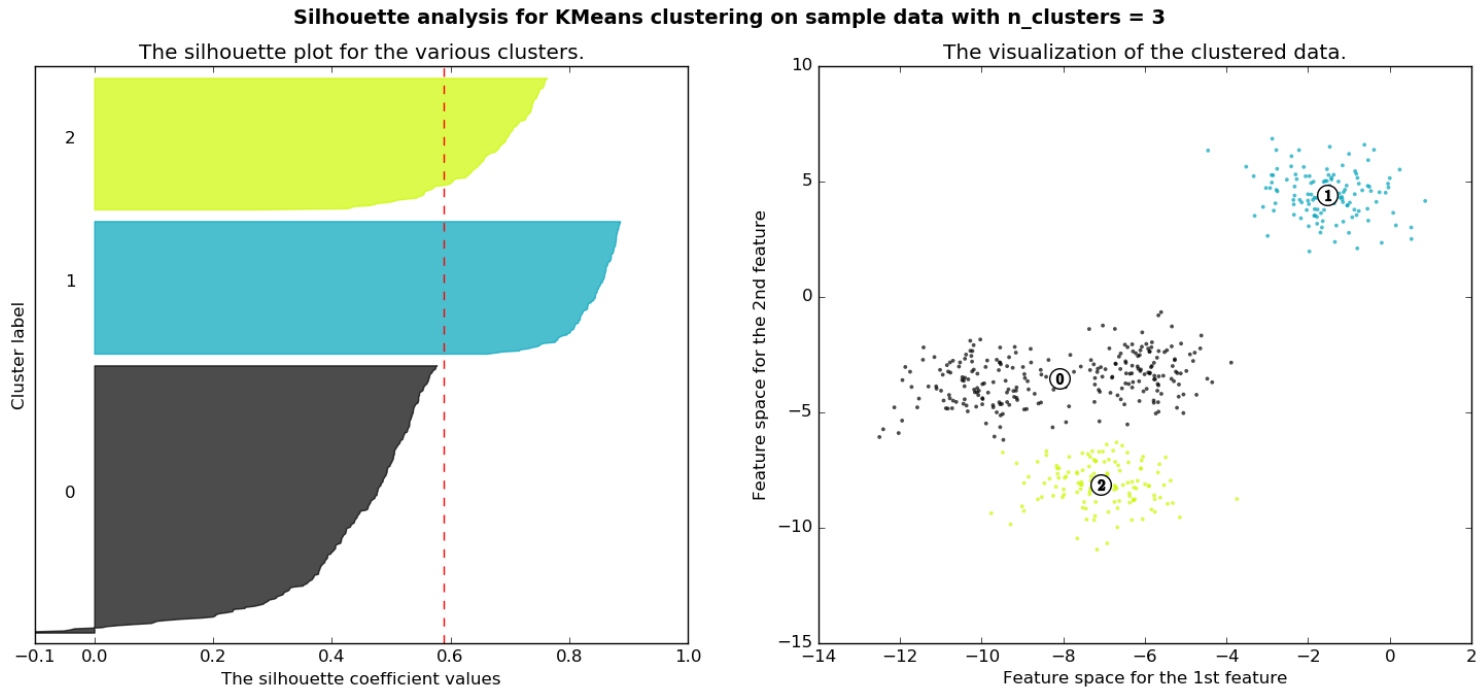
- $s(i)$ is close to one when "within" similarity is much smaller than smallest "between" similarity
- $s(i)$ is 0 when i is right on the border between two clusters
- $s(i)$ is negative when i probably belongs in another cluster
- By definition, $s(i) = 0$ if it is the only node in the cluster

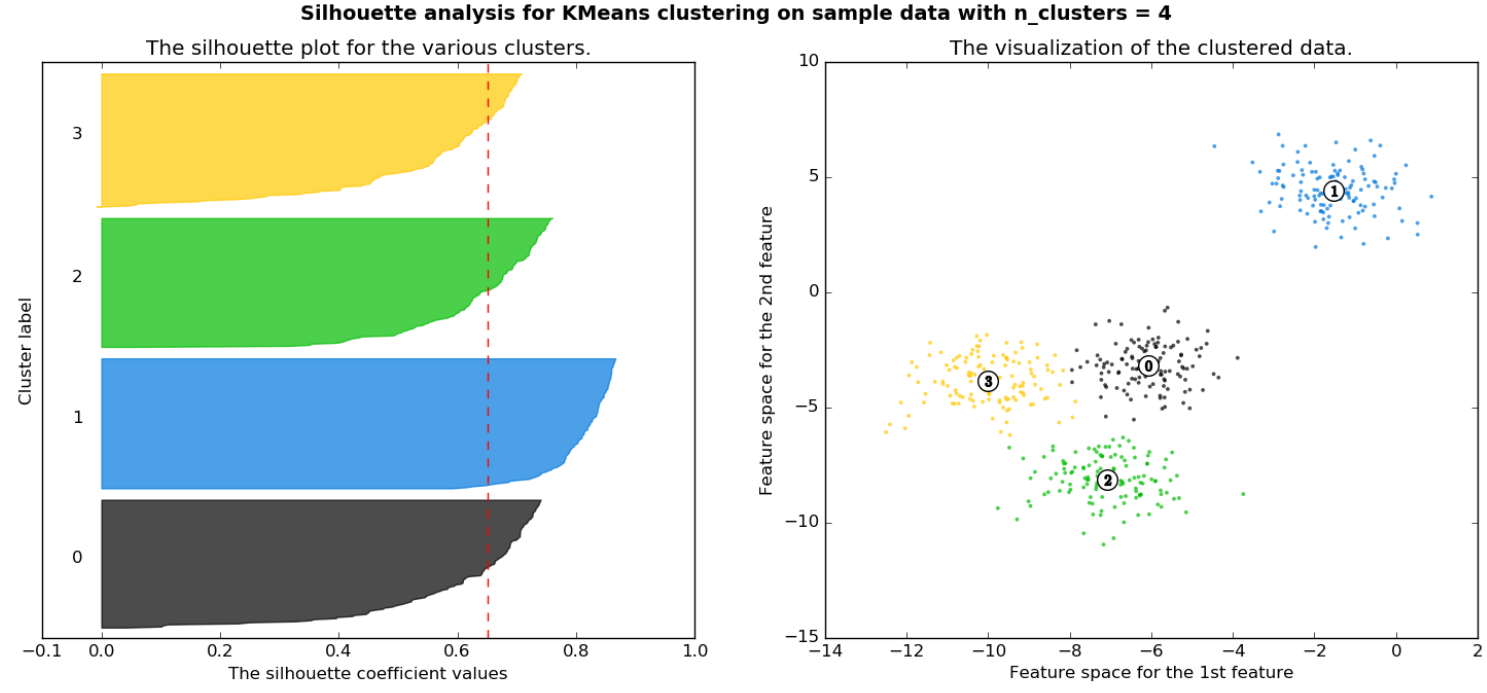
- The quality of a single cluster can be measured by the average silhouette score of its members, (close to 1 is best)
- The quality of a total clustering can be measured by the **average silhouette score of all the instances**
- To find best clustering, compare total silhouette scores across clusterings and choose the highest

Visualizing Silhouette

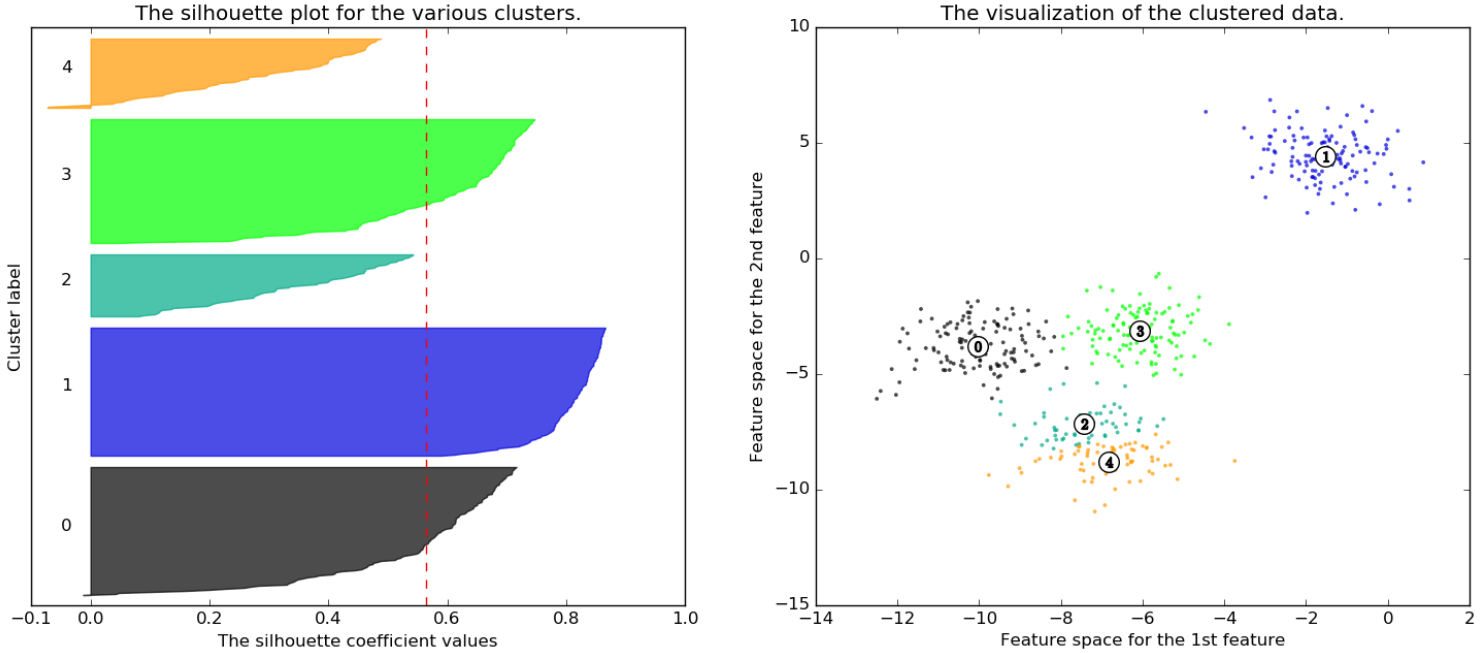


Width is quality of cluster, height is size of cluster
Dashed line is average silhouette score for clustering
Why does cluster 1 have a higher score?

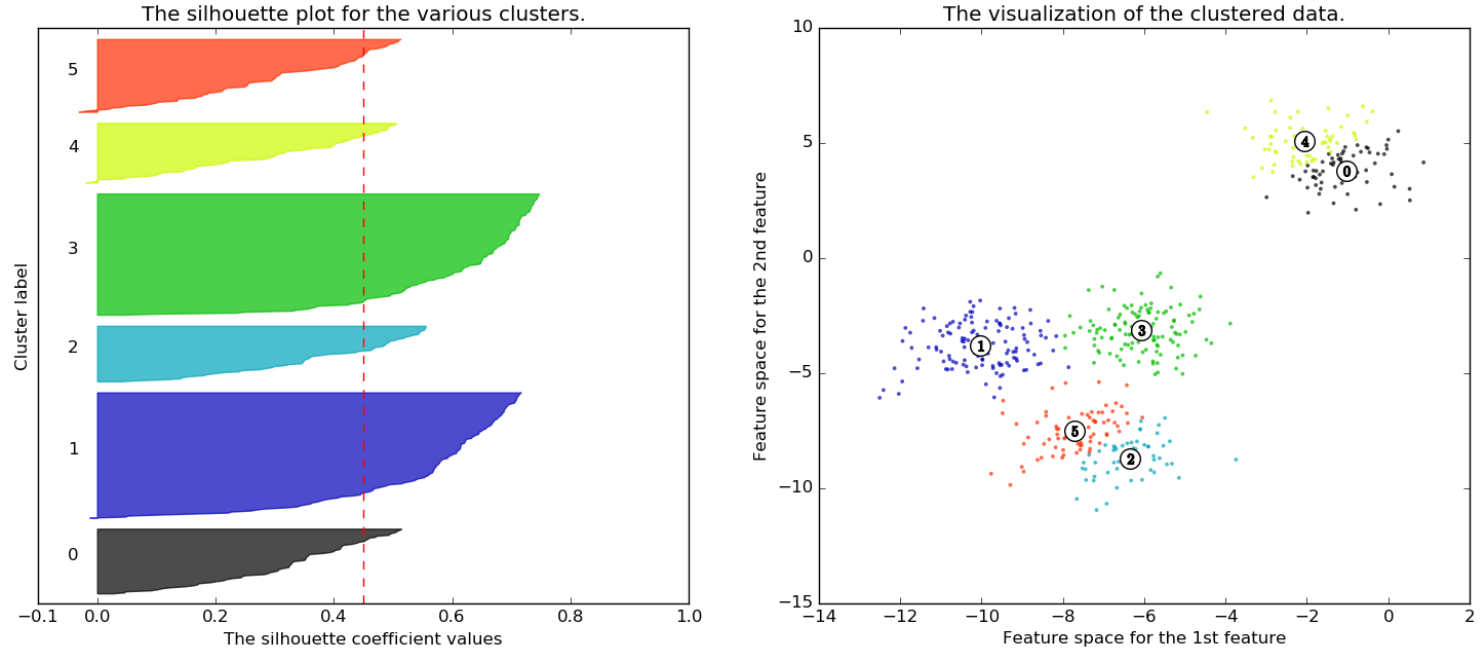

Silhouette analysis for KMeans clustering on sample data with n_clusters = 3
The silhouette plot for the various clusters.
The visualization of the clustered data.
Cluster label
The silhouette coefficient values
Feature space for the 2nd feature
Feature space for the 1st feature


Silhouette analysis for KMeans clustering on sample data with n_clusters = 4
The silhouette plot for the various clusters.
The visualization of the clustered data.
Cluster label
The silhouette coefficient values
Feature space for the 2nd feature
Feature space for the 1st feature

Silhouette analysis for KMeans clustering on sample data with n_clusters = 5



Silhouette analysis for KMeans clustering on sample data with n_clusters = 6



Silhouette

- Best case graph for silhouette?
- Clusters are wide – Scores close to 1
- Not many small silhouette instances
- Depending on your goals:
  - Clusters are similar in size
  - Cluster size and/or number are close to what you want

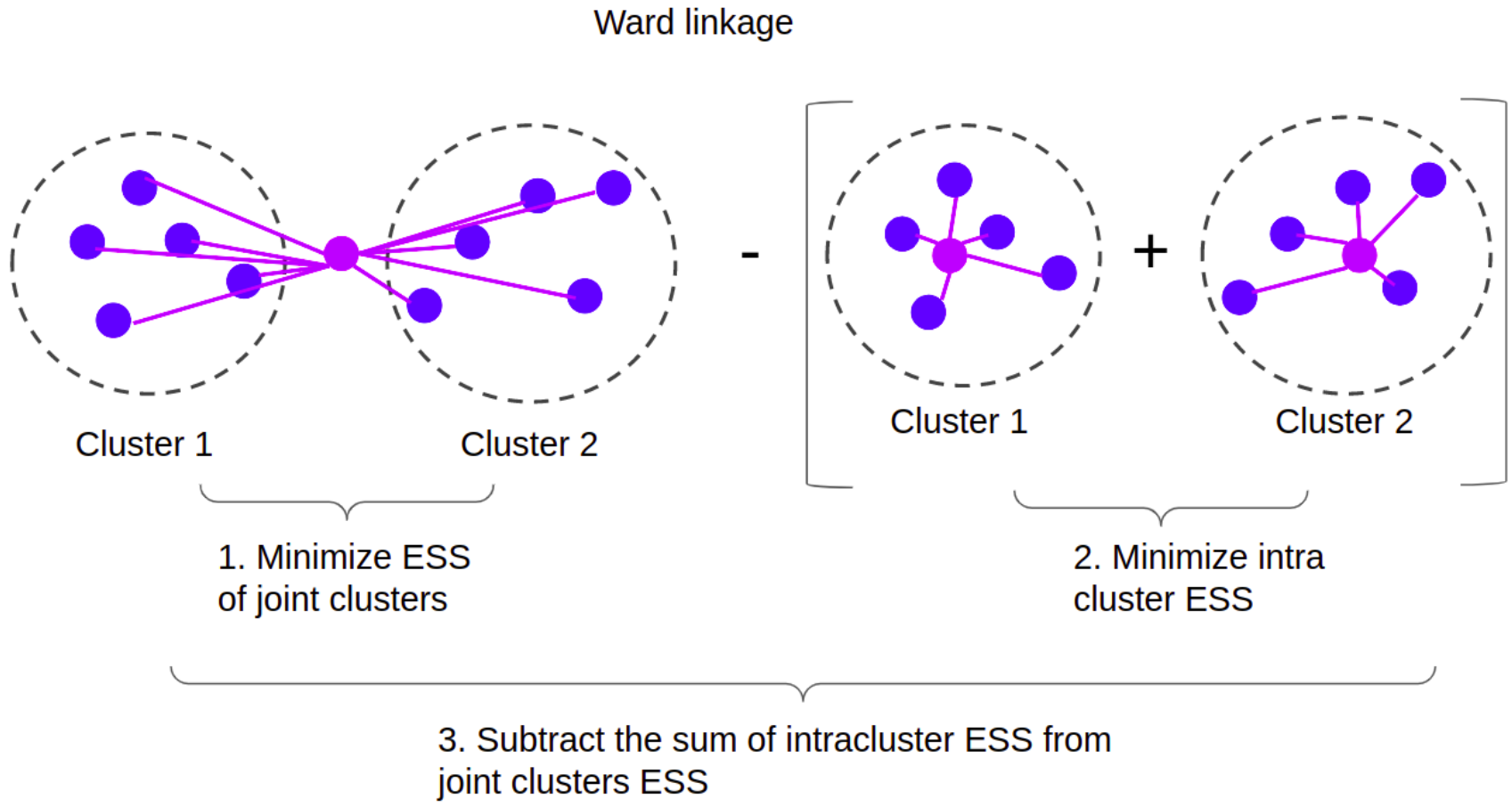
Silhouette

- Could just use total silhouette average to decide best clustering but best to do silhouette analysis with a visualization tool and use score along with other aspects of the clustering
  - Cluster sizes
  - Number of clusters
  - Shape of clusters
  - Etc.
- Note when task dimensions are > 3 (typical and no longer visualizable for us), silhouette graph still easy to visualize
- $O(n^2)$ complexity due to $b(i)$ computation
- There are other cluster metrics out there
- These metrics are rough guidelines and should be "taken with a grain of salt"

Ward Linkage

Ward Linkage

Ward linkage measures variance of clusters. The distance between two clusters, A and B, is how much the sum of squares distance from each point to its centroid would increase if we merged them. Merge the two clusters with minimum increase.



Summary

- Standard clustering highly used
- Can also use clustering as a discretization technique on continuous data for many other models which favor nominal or discretized data
  - Including supervised learning models (Decision trees, Naïve Bayes, etc.)
- With so much (unlabeled) data out there, opportunities to do unsupervised learning are growing
  - Semi-Supervised learning is becoming very important
  - Use unlabeled data to augment the more limited labeled data to improve accuracy of a supervised learner
- Deep Learning – Unsupervised training of early layers is an important approach in some deep learning models

Clustering Example

PNAS RESEARCH ARTICLE | POLITICAL SCIENCES COMPUTER SCIENCES OPEN ACCESS

Leveraging AI for democratic discourse: Chat interventions can improve online political conversations at scale

Lisa P. Argyle[a,1,2], Christopher A. Bail[b,1,2], Ethan C. Busby[a,1], Joshua R. Gubler[a,1], Thomas Howe[c,1], Christopher Rytting[c,1], Taylor Sorensen[d,1], and David Wingate[c,1]

Edited by Kathleen Jamieson, University of Pennsylvania, Philadelphia, PA; received July 9, 2023; accepted August 18, 2023

Political discourse is the soul of democracy, but misunderstanding and conflict can fester in divisive conversations. The widespread shift to online discourse exacerbates many of these problems and corrodes the capacity of diverse societies to cooperate in solving social problems. Scholars and civil society groups promote interventions that make conversations less divisive or more productive, but scaling these efforts to online discourse is challenging. We conduct a large-scale experiment that demonstrates how online conversations about divisive topics can be improved with AI tools. Specifically, we employ a large language model to make real-time, evidence-based recommendations intended to improve participants' perception of feeling understood. These interventions improve reported conversation quality, promote democratic reciprocity, and improve the tone, without systematically changing the content of the conversation or moving people's policy attitudes.

democratic deliberation | computational social science | generative AI | political science

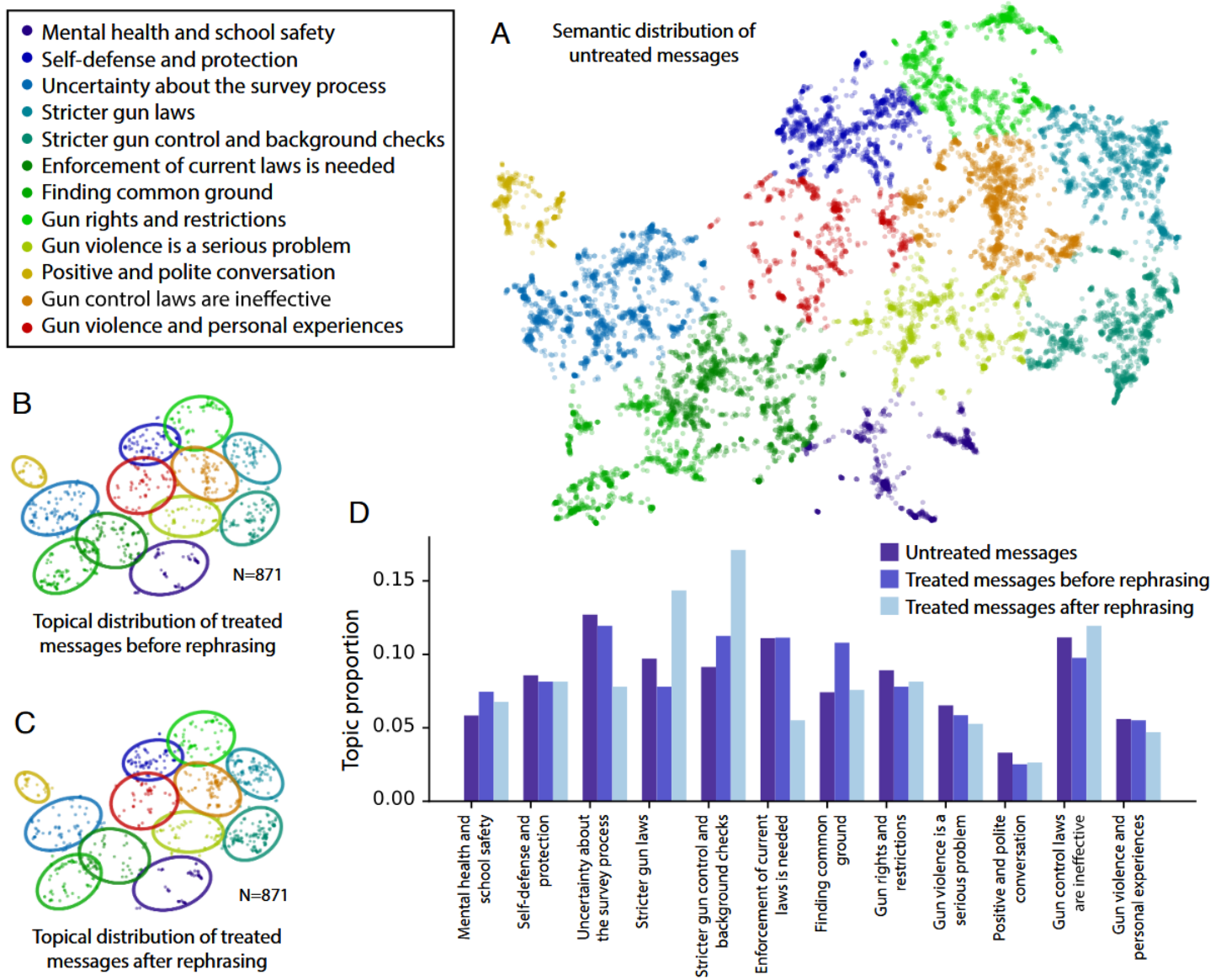


Fig. 3. Analysis of semantic content of messages. Panel (*A*) presents a visualization of the topical distribution of messages sent on the platform. Each point is the semantic embedding of a message; points that are close to each other represent messages that are semantically similar. Messages are clustered with k-means, and clusters are automatically labeled by GPT-4; see *SI Appendix* for technical details. As demonstrated by the figure, the conversations spanned a wide range of subtopics about gun control, including background checks, school safety, the role of guns in school, mental health, and enforcement issues surrounding gun ownership. Additional topic clusters show general conversational dynamics, such as introductory or closing material. Panels (*B* and *C*) graphically show the distribution of messages selected for treatment and the distribution of the corresponding rephrased messages. Both sets of messages are similarly distributed, both to each other and to the untreated messages shown in Panel (*A*). Panel (*D*) quantitatively shows the topic proportions; statistical analysis shows that the distributions are not significantly different.