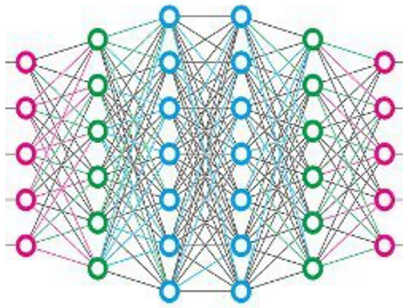
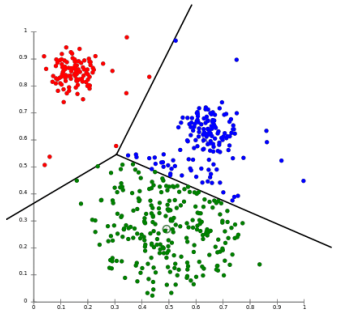
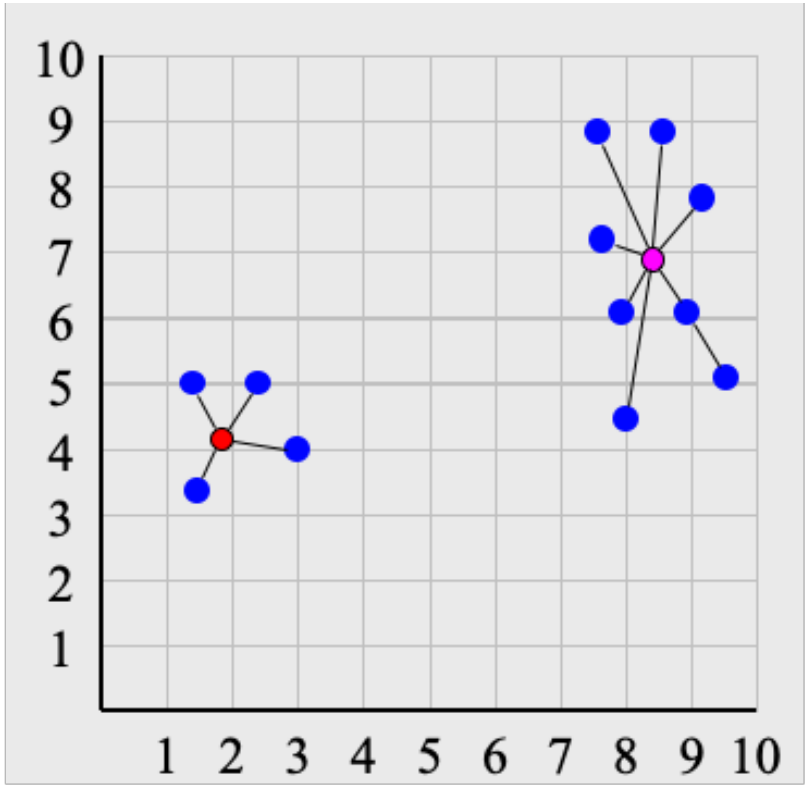
CS 445
Introduction to Machine Learning

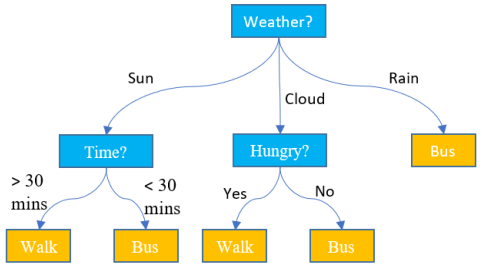
Unsupervised Learning
Clustering

Instructor: Dr. Kevin Molloy









Learning Objectives From Last Class

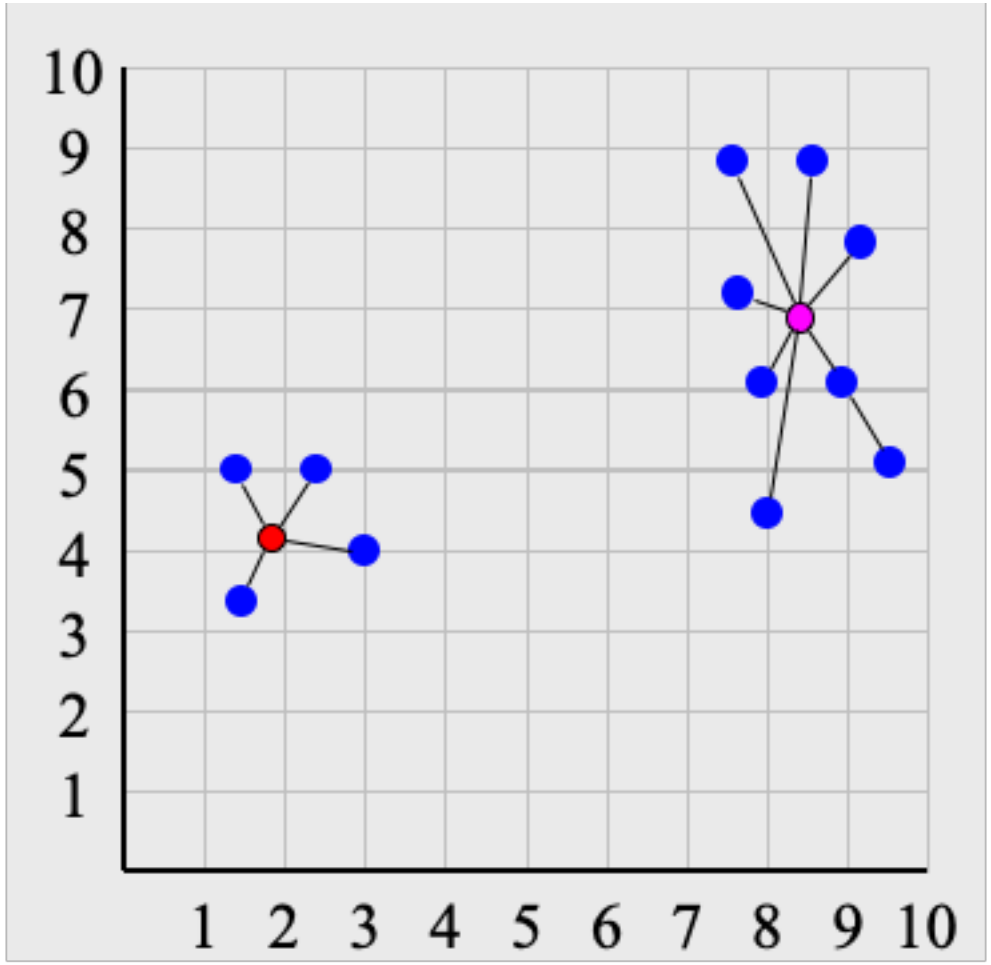
- Define unsupervised learning
- Hierarchical Clustering
  - Optimal clustering is intractable (minimize intracluster and maximize intercluster distances)
  - Heuristic algorithm uses bottom-up approach (or top-down)
  - Techniques for computing distances between individual members and clusters (which is a hyperparameter)
- Partitional Clustering (K-Means)
  - Approximation is used (Lloyd's method)
  - Sensitive to initial centroid location
  - Hyperparameter includes distance metric and the number of clusters ***k***

Learning Objectives

- Revisit K-Means as a multi-objective optimization problem
- Define and sketch a density based clustering method
- Cluster evaluation

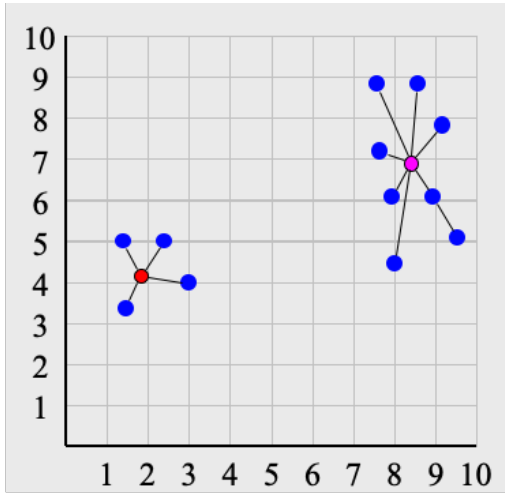
K-Means Clustering

- Partitional clustering
- Number of clusters defined in advanced (it is a hyperparameter)
- Each cluster has a centroid (which is computed and not part of the dataset)
- All data points belong to exactly 1 cluster
- Optimal K-Means clustering is NP-complete, use Lloyd's approximation method known
- Lloyd's method initially selects the centroids randomly (although there are other techniques for doing this)



Variance when running K-Means

- Final cluster assignments are highly dependent on the initial centroid location.
- Solution? Repeat the method several times and pick the "best" clustering



$$SSE = \sum_{k=1}^{K} \sum_{x \in C_k} \|x - c_k\|^2$$

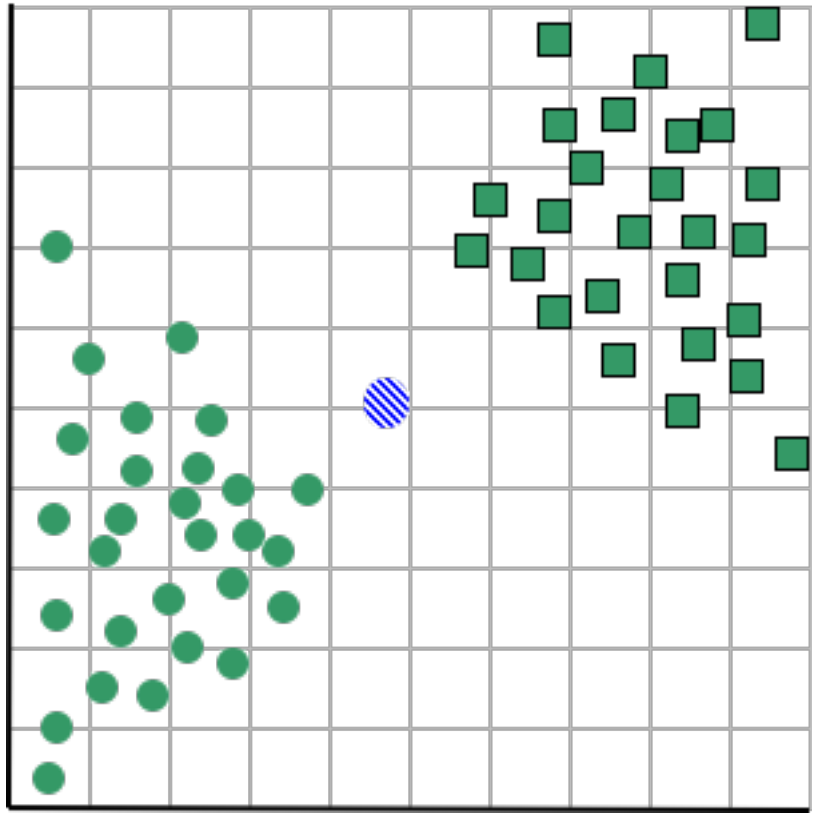
Sum of square errors (SSE) is used to judge the quality of the solution. This is also known as the **objective** function (what we are trying to minimize)

Where c_k is the centroid and C_k is the set of points assigned to cluster k, *x* are the individual points assigned to cluster *k*.

SSE is available from the scikit-learn *kmeans* object (**kmeans.inertia_**).

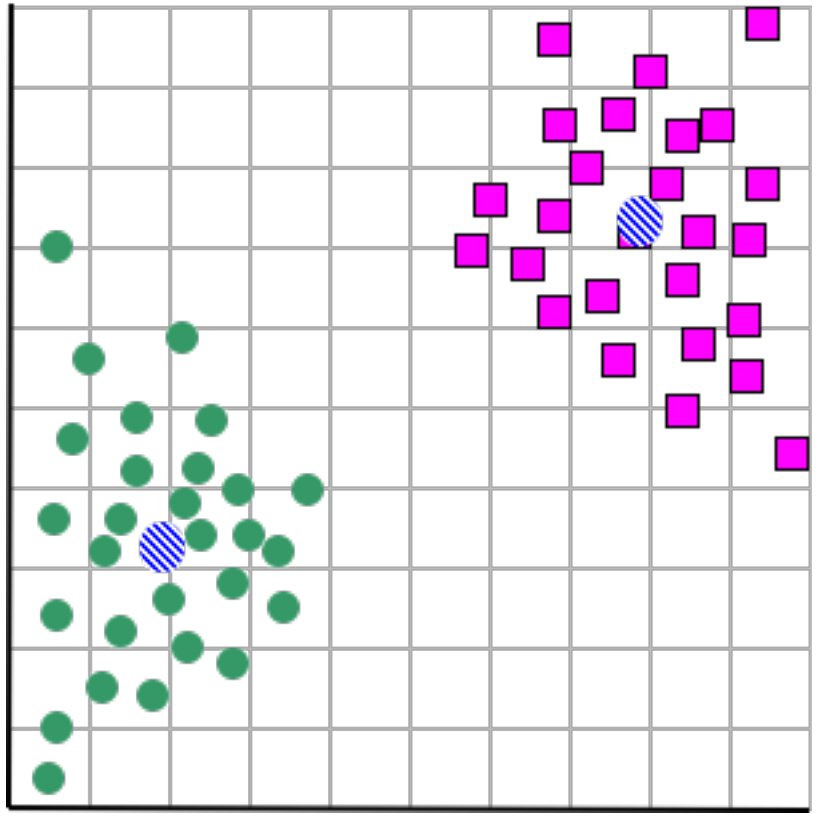
Selecting *k*

- When *k* = 1, the SSE/objective function is 873.



Selecting *k*

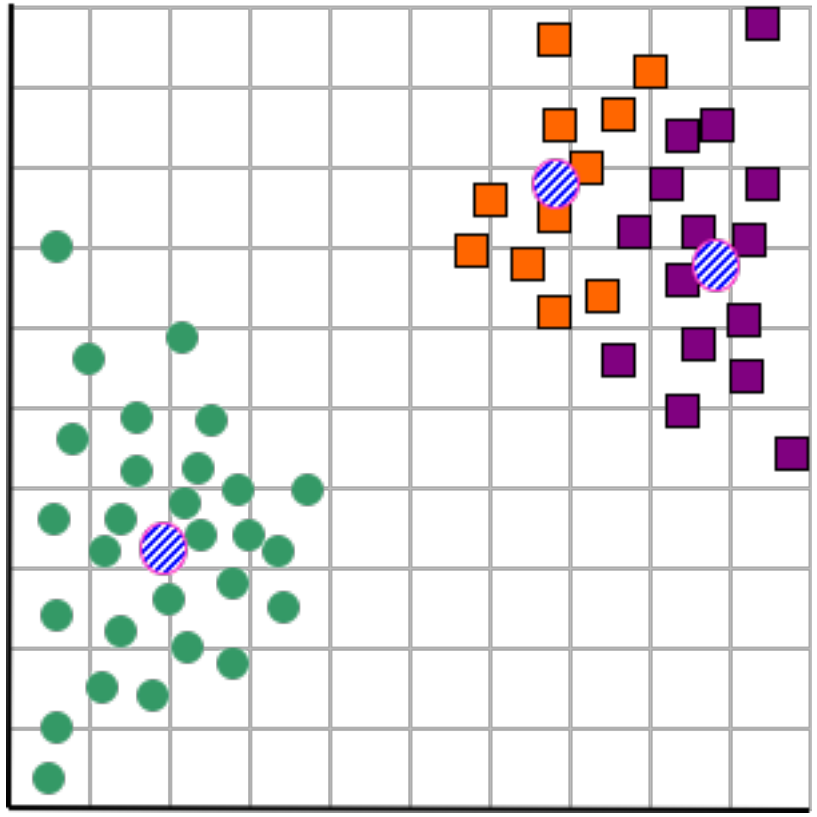
- When *k* = 1, SSE is 873.
- When *k* = 2, SSE is 173.1.



Selecting *k*

- When *k* = 1, SSE is 873.
- When *k* = 2, SSE is 173.1.
- When k = 3, SSE s 133.6.

What happens when we pick *k* = *n*?

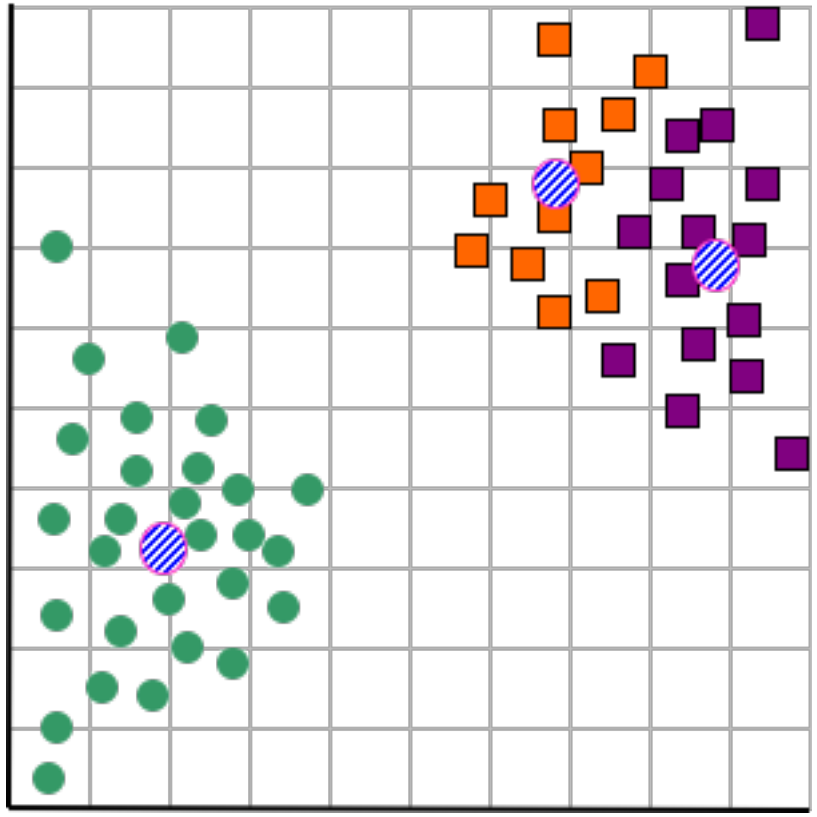


Selecting *k*

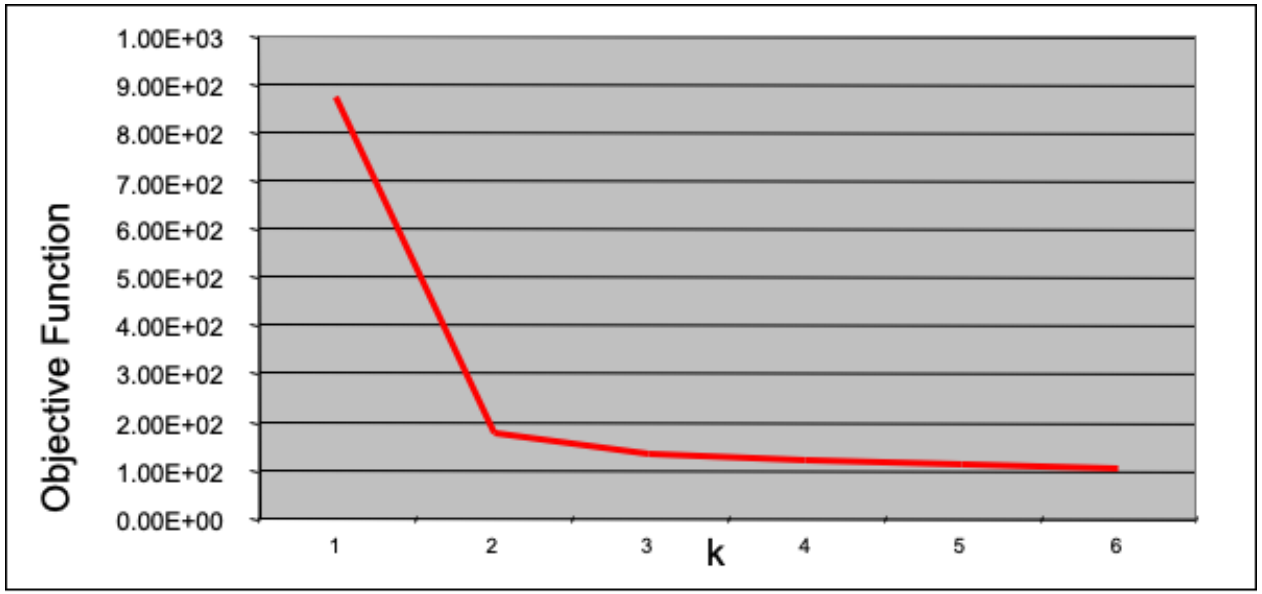
- When *k* = 1, SSE is 873.
- When *k* = 2, SSE is 173.1.
- When k = 3, SSE s 133.6.

What happens when we pick *k* = *n*?

SSE goes to zero, but what value is it to have *n* clusters of size 1.



Looking for the "knee" in the SSE

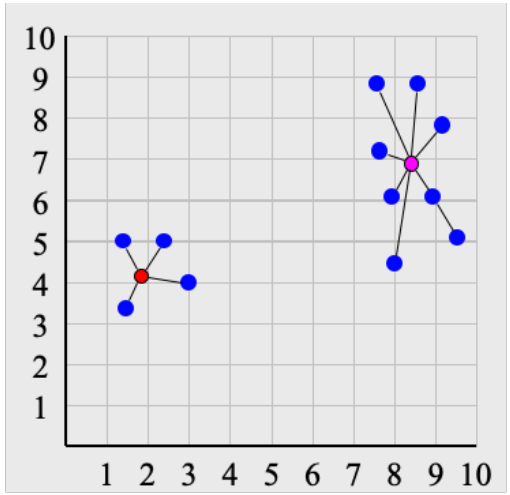


- The abrupt change at k = 2 is highly suggestive of two clusters in this data. This technique is known as "elbow" or "knee" finding.

- Can be accomplished quantitatively by computing the ratio of the SSE. When this ratio falls below some threshold, you accept the value of *k* (now this ratio is the hyperparameter).
- We will see another technique for measuring cluster quality momentarily.

$$\frac{SSE_{k-1}}{SSE_k} > \varepsilon$$

K-Means Issues and Alternatives



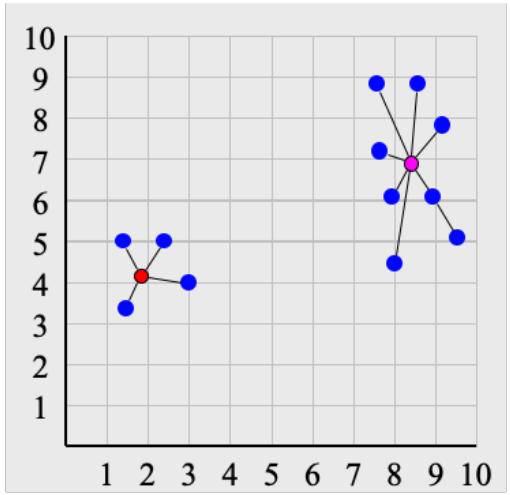
Handling Empty clusters: Random centroid creation could result in a cluster with no members. Use alternate approach to select centroid fixes this issue.

Outliers: SSE is inflated and cluster become non-represenative as a result. Common technique is to try and eliminate outliers before clustering.

Postprocessing to Escape Local Minimum: Lloyd's method will almost certainly converge to a local minimum, thus, we can *escape* this assignment and explore others by modifying the cluster assignments through a cycle of:

- Add a cluster by:
  - Split the cluster with the largest SSE OR
  - introduce a new centroid
- Disperse/remove the cluster that
  - contributes the SSE the least OR
  - merge two clusters (one idea is to choose the clusters with the two closest centroids).

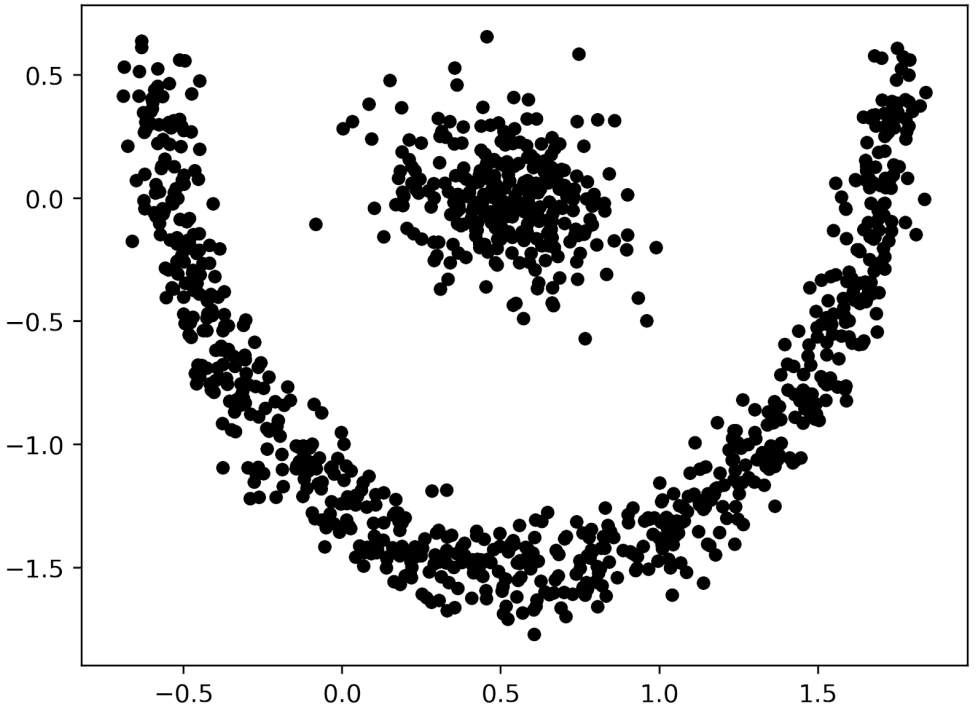
K-Means++ and Other Issues



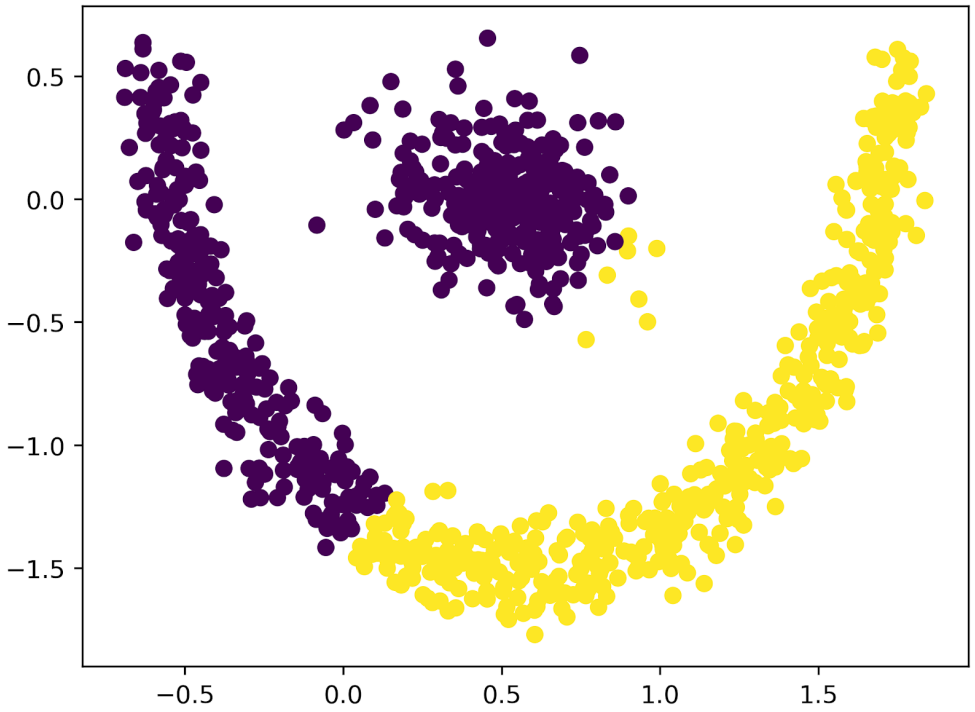
"Smart" initial centroid selection method proposed by David Arthur and Sergei Vallilvitsmii in 2007 (paper here).

1. Pick one of the data points as a centroid
2. Repeat k -1 times:
    1. Weight points by their distance to their nearest centroid
    2. Randomly select a point with respect to these weights

K-Means Challenges

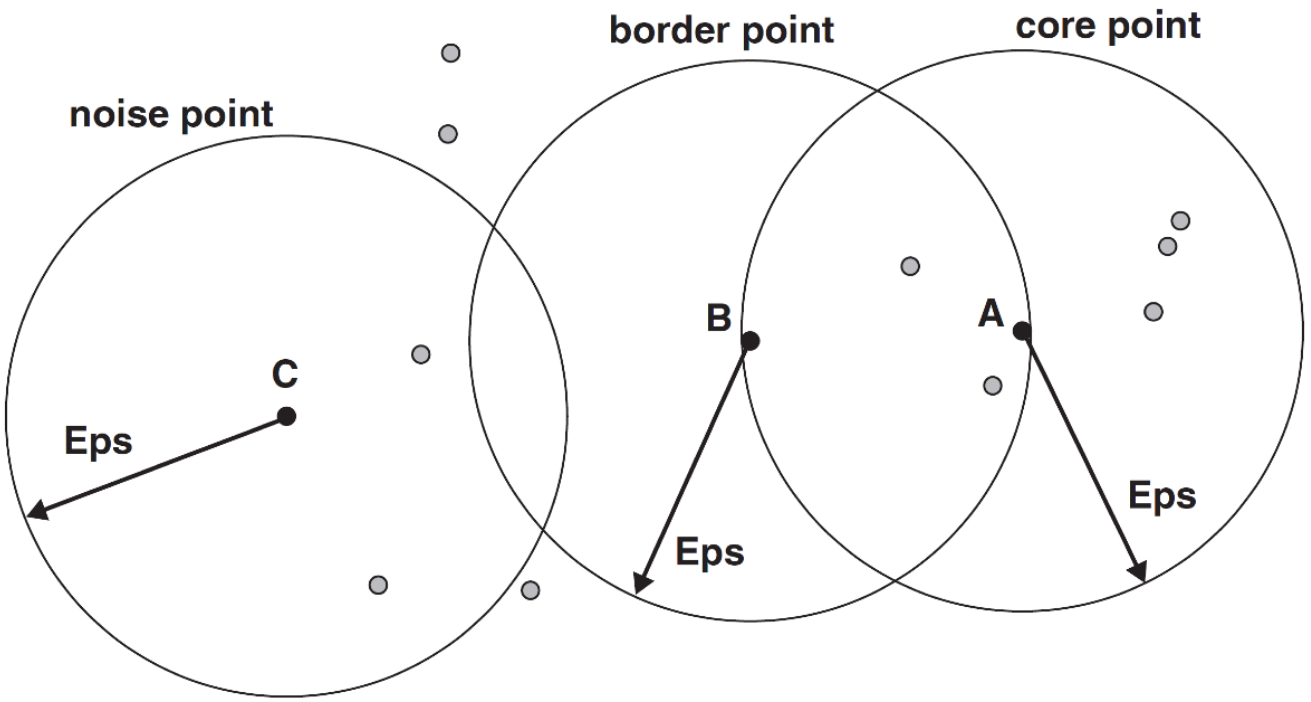


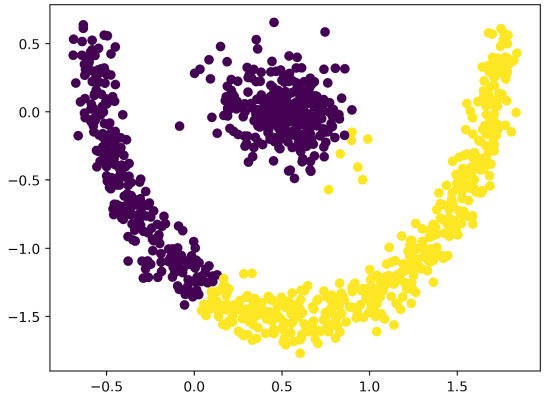
What clusters will k-means find when k=2?



Not so ideal.

Using Densities – Meet DBScan





Density – number of points within a radius **(eps)**

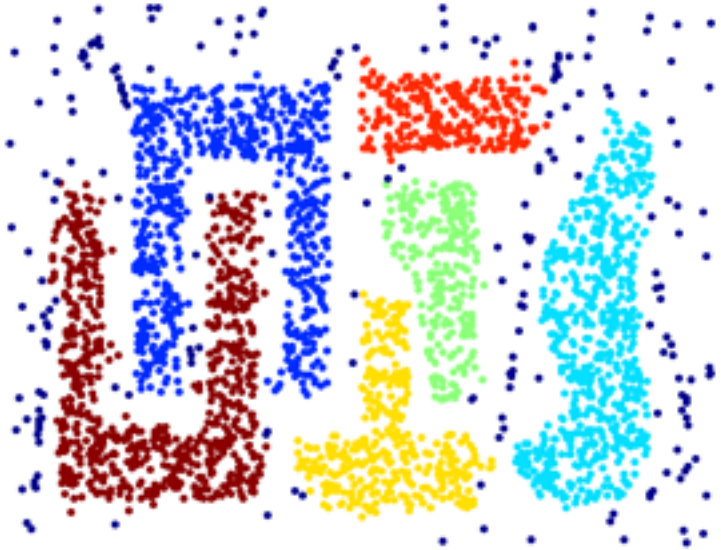
A point is a **core point** if it has at least a specified number of points (MinPts) within eps(including itself)

A **border point** is not a core point, but is the neighborhood of a core point.

A **noise point** is any point that is not a core or border point.

DBScan Works Well

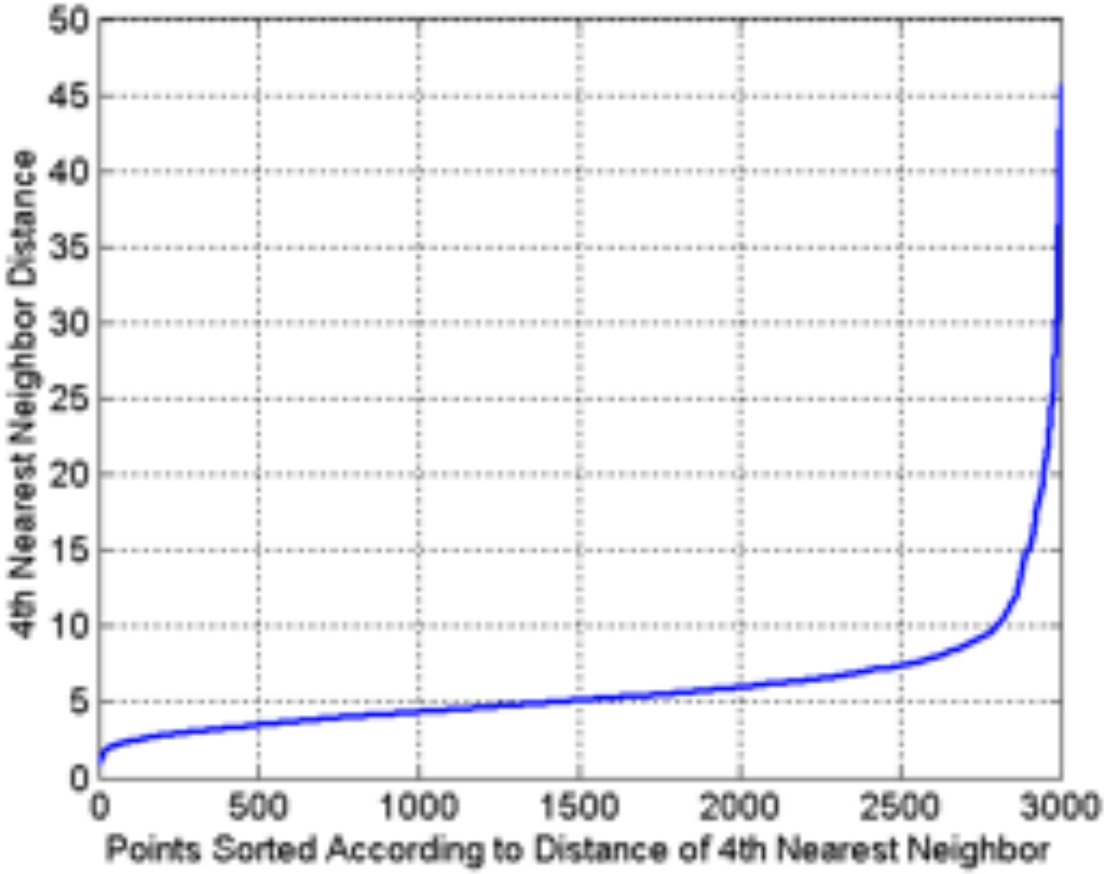
Original Points



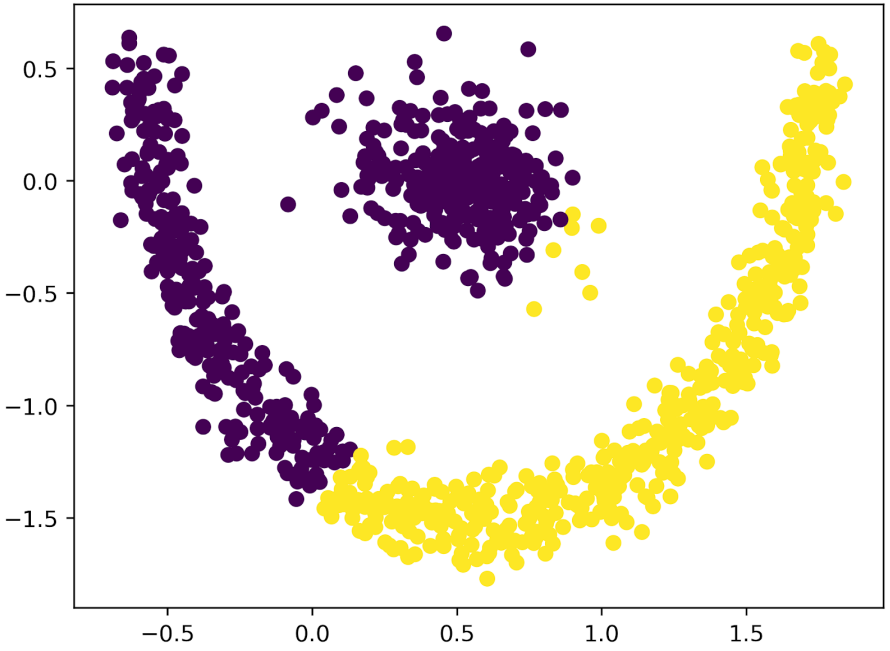
Clustered Points

DBScan: Determining EPS and MinPts

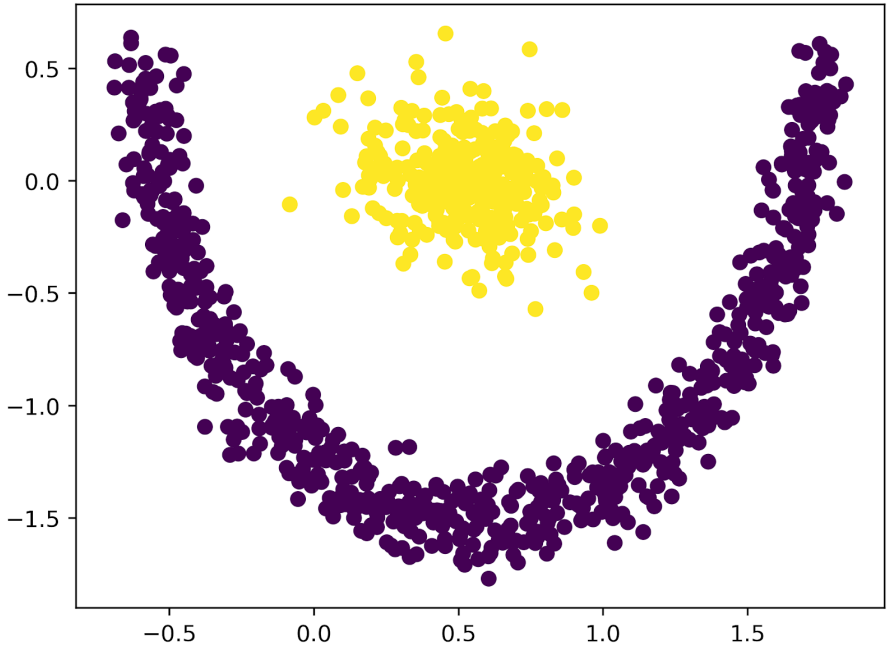
- Idea is that for points in a cluster, their k^{th} nearest neighbor are roughly the same distance
- Noise points have the k^{th} nearest neighbor further away
- Plot sorted by distance of every point to its k^{th} nearest neighbor show on right. Notice that for many points, the distances are consistent.



Recall K-Mean's Performance



K-Means (k=2)



DBScan (eps = 0.25, minPts = 5)

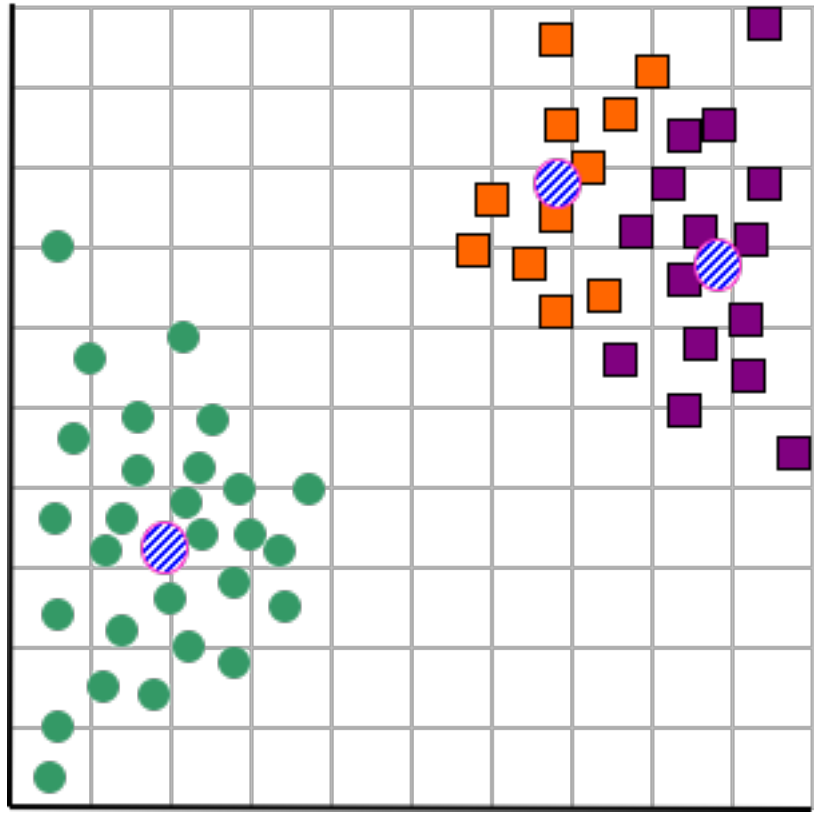
Determining Cluster Validity

What happen to accuracy, precision, recall?

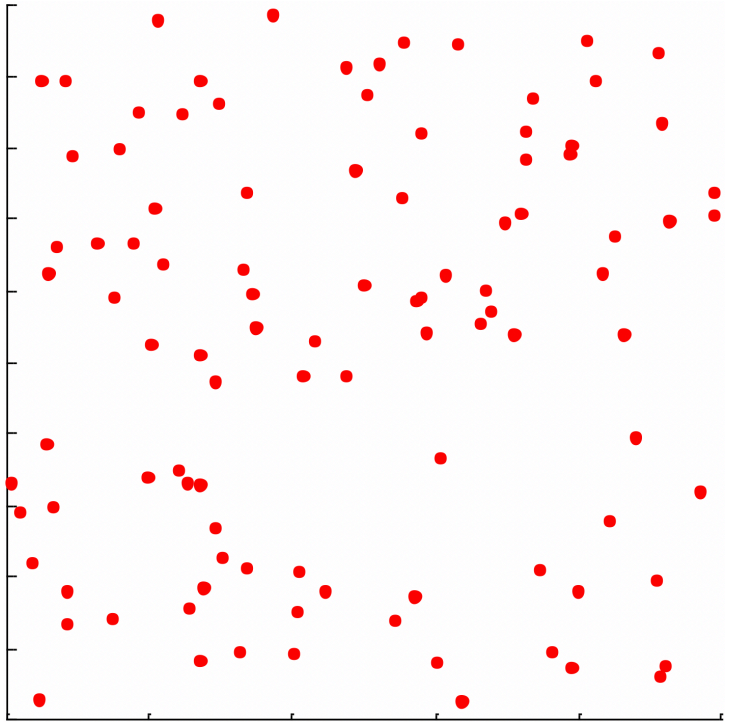
Cluster is subjective, so, hard to quantify what "good" means

Evaluate cluster to:

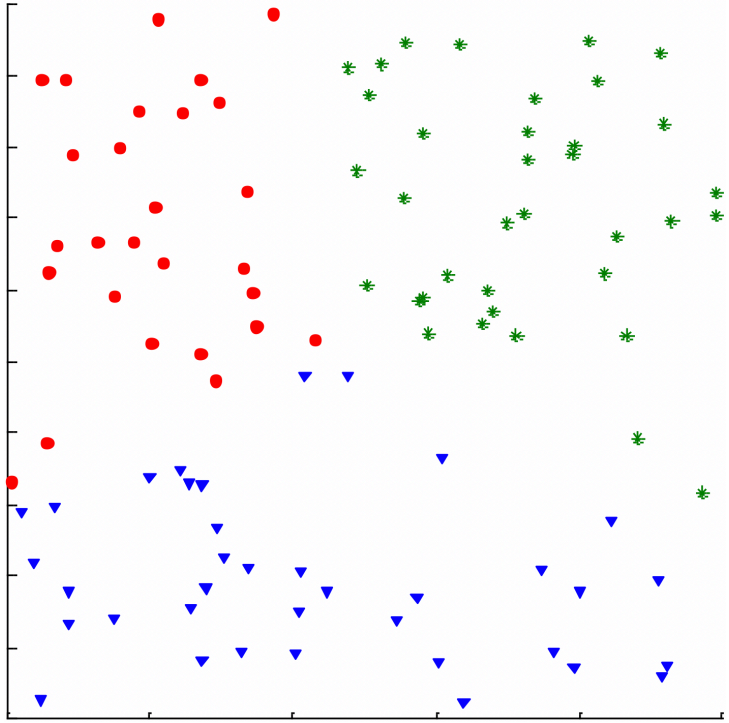
- Avoid finding patterns in the noise
- Compare clustering algorithms
- Compare to sets of clusters



Clustering Challenges



Random Noise



K-Means (k = 3)

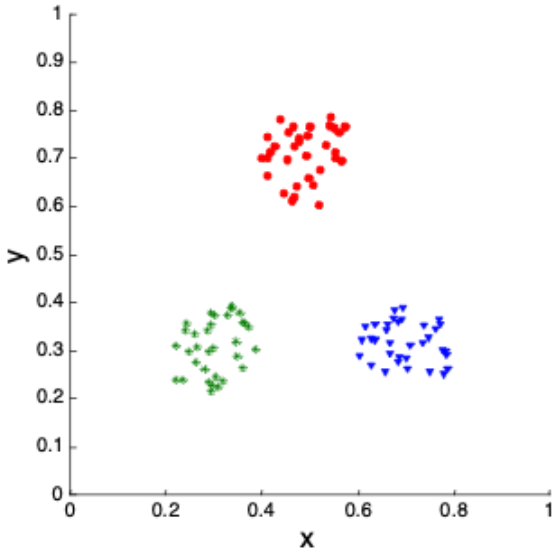
If you ask for a clustering, the methods will oblige, no questions asked.

Visually, you can see this not a good clustering. But how do we do this in high dimensions?

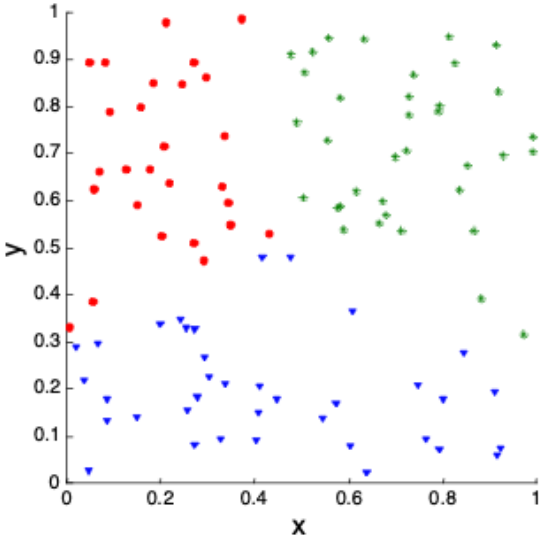
Clustering Validity Via Correlation

Two matrices: **proximity/similarity** matrix and **ideal similarity** matrix

- One row/column for each data point (*n* x *n)*
- An entry is a 1 IF the associated pair belongs to the same cluster, other 0
- Compute the correlation between the 2 matrices



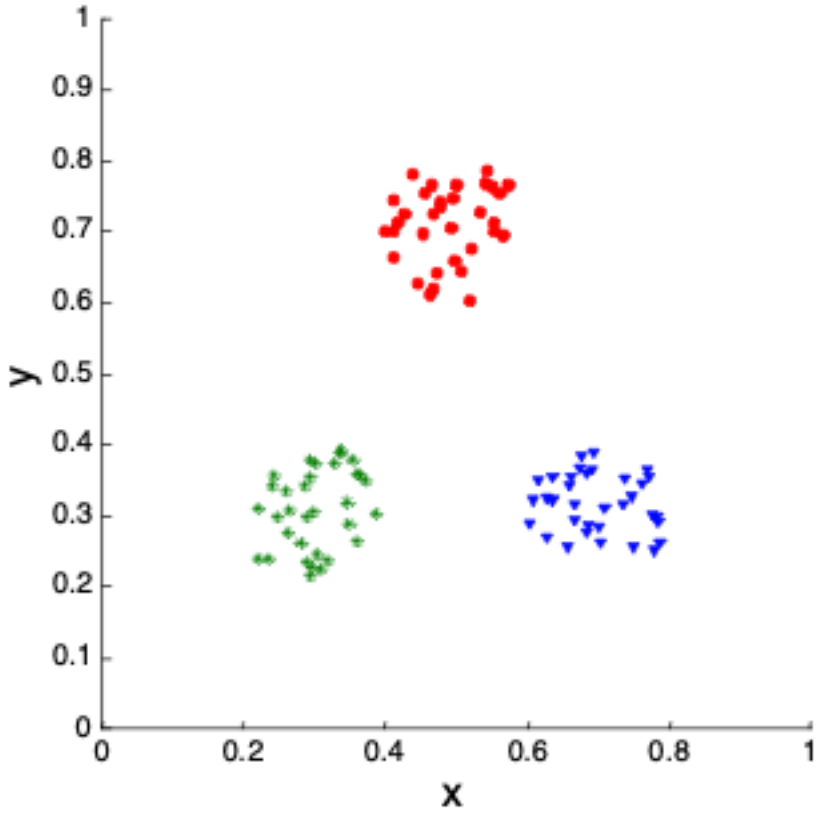
Correlation = 0.9235

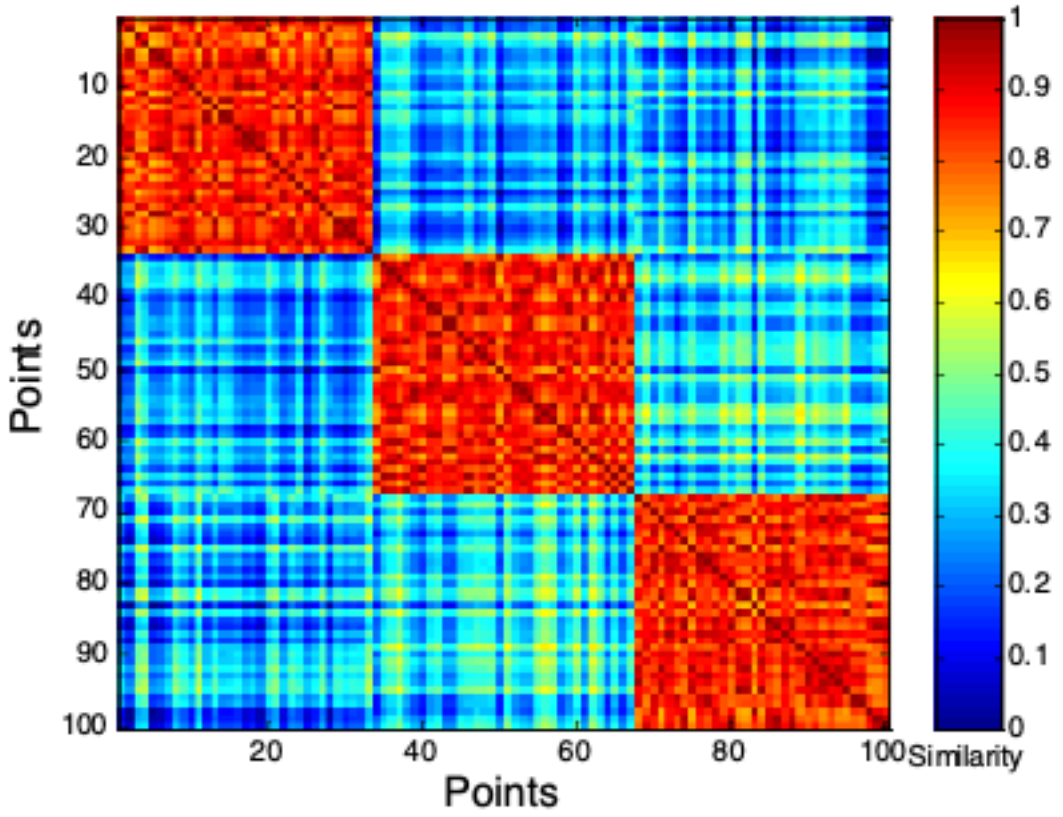


Correlation = 0.581

Similarity Matrix for Validation

Two matrices: **proximity** matrix and **ideal similarity** matrix

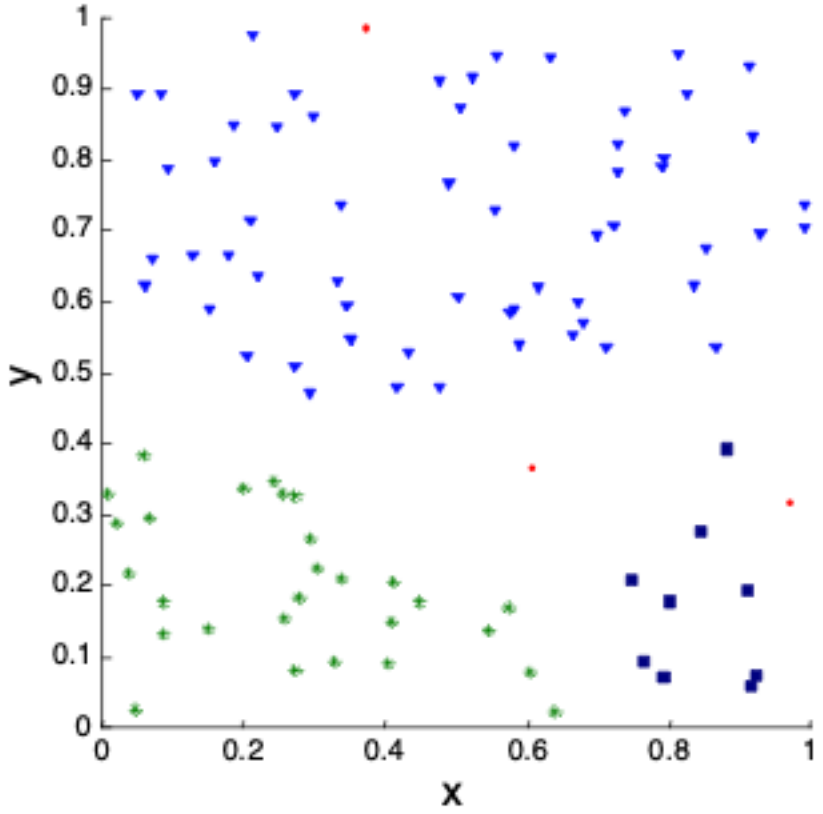


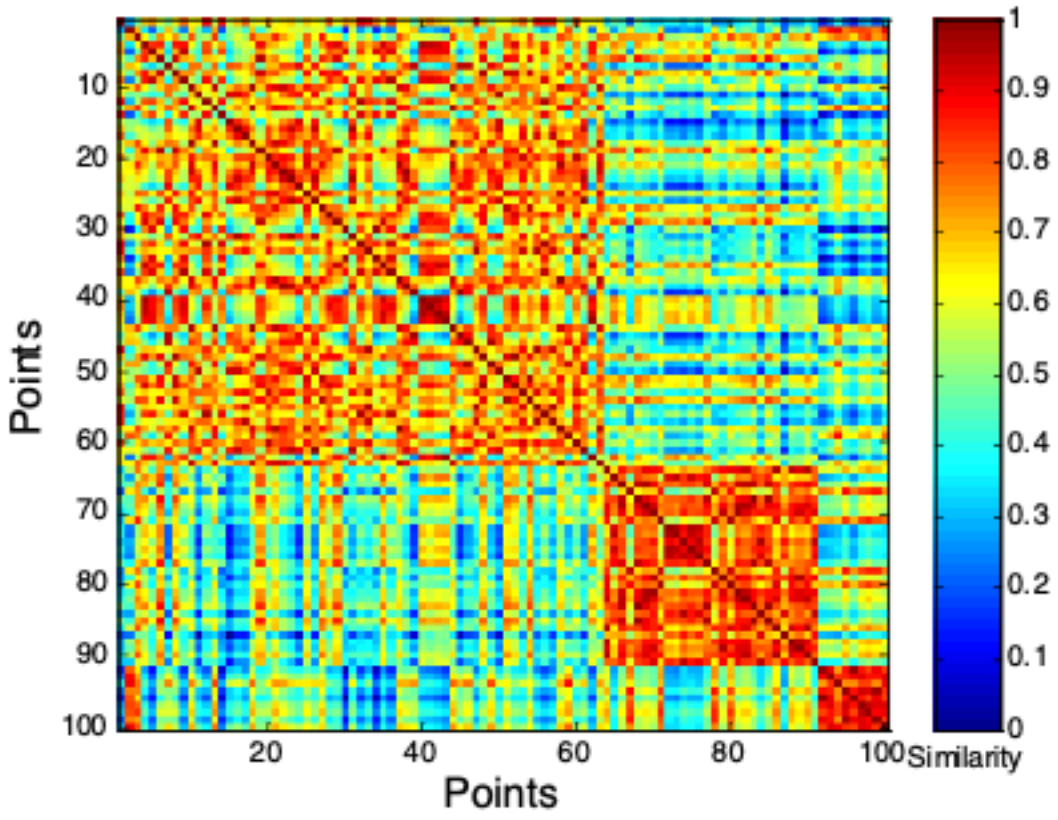


Data sorted/ordered by cluster label

Similarity Matrix for Validation

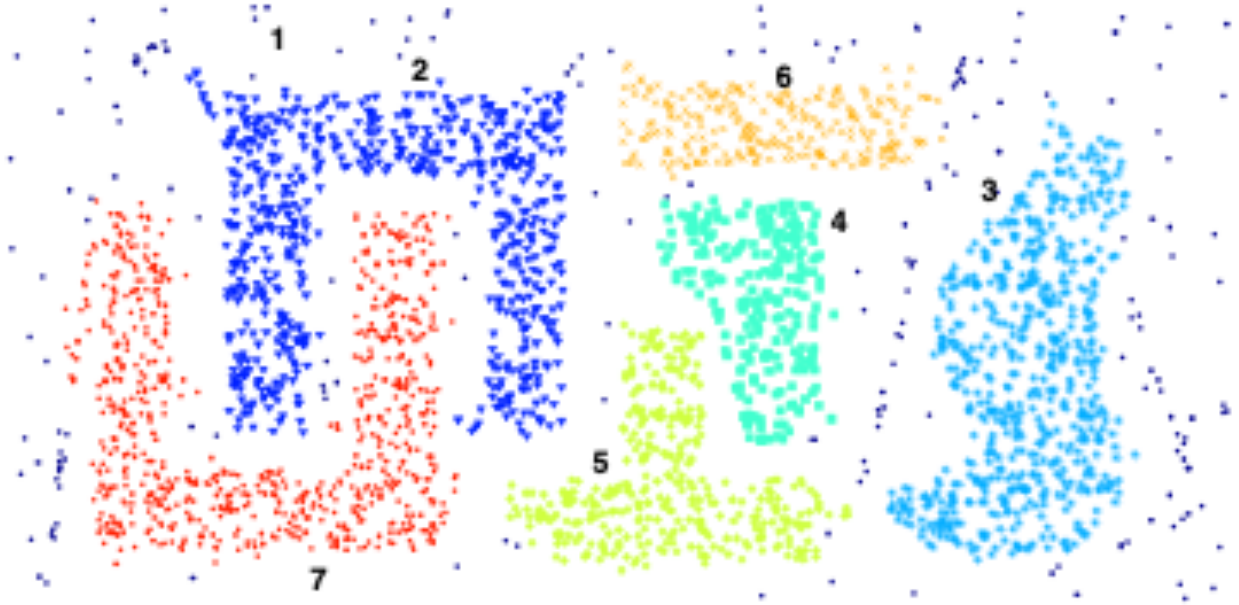
Two matrices: **proximity** matrix and **ideal similarity** matrix

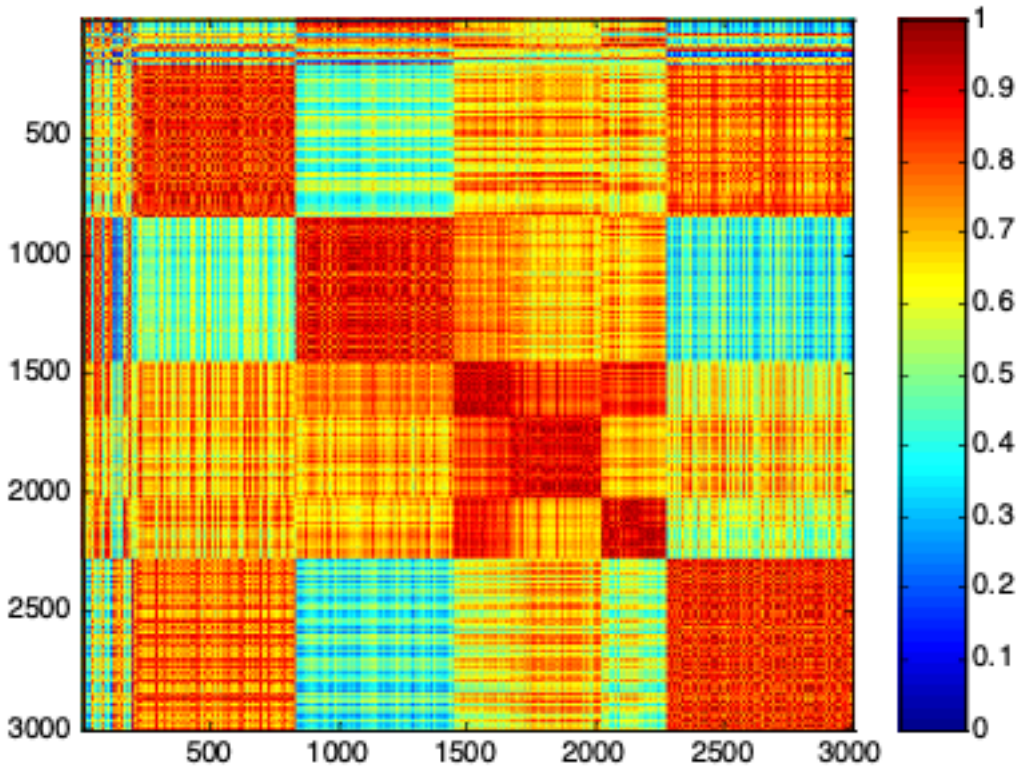




Similarity Matrix for Validation

DBScan cluster quality visualized via a similarity matrix.



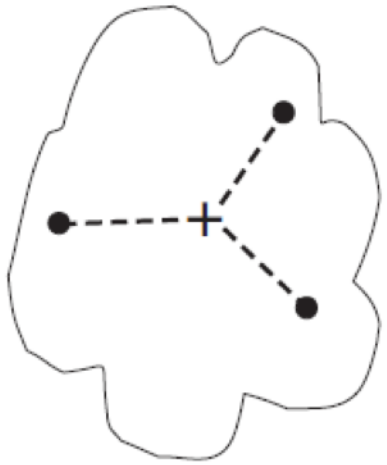


Internal Measures

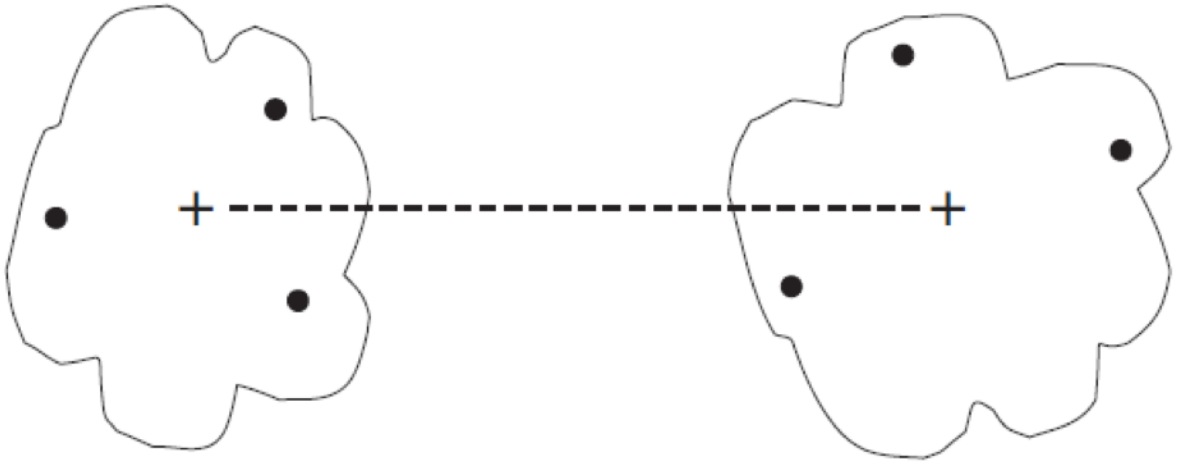
- **Cluster cohesion** is how closely related objects are in a cluster (within cluster sum of squares, or WSS). **Cluster separation** measures how distinct of well separated a cluster is from other clusters, the between cluster sum of squares (or BSS).

$$WSS = \sum_{k=1}^{K} \sum_{x \epsilon C_k} (x - c_k)^2 \qquad BSS = \sum_{k=1}^{K} |C_k|(m - m_k)^2$$

$|C_k|$is size of cluster *k*, *m* is the mean of all the data and $m_k$is the mean of cluster *k*.
NOTE: that the sum of SSE and BSS is a constant for a given set of data points *n*.



Cohesion



Cluster Separation

Figures from Intro to Data Mining (Tan et al).

Silhouette Coefficient

- Relatively efficient (especially if you can store the distance matrix in memory)

Method:

For each datapoint $i \in N$:

1. For each data point, compute a_i, which is its average distance to all other objects that are in the same cluster as a_i(except point *i* itself) $a(i) = \frac{1}{|C_i|}\sum_{j \in C_i, i \neq j} d(i,j)$
2. For each data point, calculate the mean distance to all other clusters and take the minimum of these values. Think of this as the mean distance to its closest neighboring cluster. Call this value b_i. $b(i) = \min_{k \neq i} \frac{1}{|C_k|}\sum_{j \in C_k} d(i.j)$
3. Data point i's silhouette coefficient is $s(i) = \frac{b_i - a_i}{\max(a_i, b_i)}$ if $|C_i| > 1$ otherwise s(i) = 0 (if $|C_i| = 1$.

When judging the qualify of the overall clustering, the average of silhouette coefficients over all data points is used.