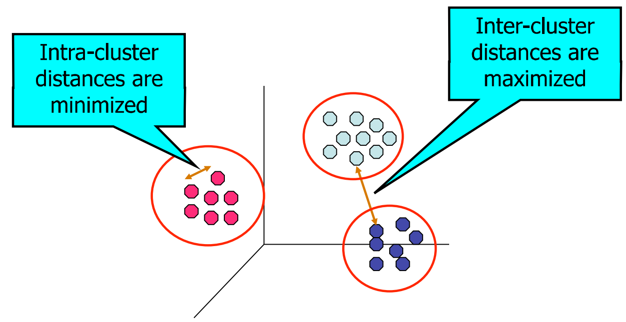
# Cluster Analysis I

# What is Cluster Analysis?

- Finding groups of objects
  - such that the objects in a group will be **similar** to one another and
  - **dissimilar** from the objects in other groups




# Applications

- Numerous!!
  - https://en.wikipedia.org/wiki/Cluster_analysis#Applications
- Any problem where you would like to find groups of items/ objects
- Cluster movies to create genres
- Cluster news articles to find trends/themes
- Cluster credit card users to find anomalies
- etc…

# Solution Format

- Clustering is NP-hard, but we can use heuristics
- A solution would be an assignment for each data point to a cluster. Say we have 10 data points and 3 clusters:

  0 0 0 1 1 1 2 2 2 0

- How can we measure the goodness of a solution?
  - To keep it simple, let's just consider intra-cluster distance
  - Each cluster i has a mean point, called the centroid $\mu_i$

# Tools

- In order to cluster, we need to define intra- and inter-cluster distances
  - We will need either similarity or dissimilarity measures
    - -1*similarity can be used as dissimilarity
- We talked about a few such measures previously
  - Euclidean distance, cosine distance, correlation
    - (which are similarity and which are dissimilarity?)

# Similarity and Dissimilarity

- **Similarity**
  - Numerical measure of how alike two data objects are.
  - Higher when objects are more alike.
- **Dissimilarity (Distance)**
  - Numerical measure of how different are two data objects
  - Lower when objects are more alike

## Similarity/Dissimilarity for Simple Attributes

$p$ and $q$ are the attribute values for two data objects.

- **Nominal**
  - E.g. province attribute of an address with values:
    {BC, AB, ON, QC, …}
    Order not important.
  - **Dissimilarity**
    $d$=0 if $p=q$
    $d$=1 if $p \neq q$

## Similarity/Dissimilarity for Simple Attributes

$p$ and $q$ are the attribute values for two data objects.

- **Ordinal**
  - E.g. quality attribute of a product with values:
    {poor, fair, OK, good, wonderful}
    Order **is important**, but exact difference between values is undefined or not important.
  - Map the values of the attribute to successive integers
    {poor=0, fair=1, OK=2, good=3, wonderful=4}
  - **Dissimilarity**
    $d(p,q) = |p - q| / (\max - \min)$
    **e.g.** $d$(wonderful, fair) = |4-1| / (4-0) = .75

## Similarity/Dissimilarity for Simple Attributes

$p$ and $q$ are the attribute values for two data objects.

- **Continuous (or Interval)**
  - E.g. weight attribute of a product
  - **Dissimilarity**
    $d(p,q) = |p - q| / (\max\text{-}\min)$

## Combining Dissimilarities

- Sometimes attributes are of many different types, but an overall similarity/dissimilarity is needed.
- For the $k$-th attribute, compute a dissimilarity $d_k$ in the range [0,1]. Then,

$$\text{dissimilarity}(P,Q) = \frac{\sum_{k=1}^{n} d_k}{m}$$

## Euclidean Distance

- When all the attributes are continuous we can use the Euclidean Distance

$$dist = \sqrt{\sum_{k=1}^{n} (p_k - q_k)^2}$$

Where $n$ is the number of dimensions (attributes) and $p_k$ and $q_k$ are, respectively, the $k^{th}$ attributes (components) or data objects $p$ and $q$.

- Attribute scaling is necessary, if scales differ
  - E.g. weight, salary have different scales

## Minkowski Distance

- Minkowski Distance is a generalization of Euclidean Distance

$$dist = \sqrt[r]{\sum_{k=1}^{n} | p_k - q_k |^r}$$

Where $r$ is a parameter, $n$ is the number of dimensions (attributes) and $p_k$ and $q_k$ are, respectively, the kth attributes (components) or data objects **p** and **q**.

**Examples**

- $r = 1$. City block (Manhattan, taxicab, $L_1$ norm) distance.
- $r = 2$. Euclidean distance
- $r \rightarrow \infty$. "supremum" ($L_{max}$ norm, $L_\infty$ norm) distance.
  - This is the maximum difference between any component of the vectors

# Similarity Between Binary Vectors

- Common situation is that objects, **p** and **q**, have only binary attributes
- Compute similarities using the following quantities

  $M_{01}$ = the number of attributes where **p** was 0 and **q** was 1
  $M_{10}$ = the number of attributes where **p** was 1 and **q** was 0
  $M_{00}$ = the number of attributes where **p** was 0 and **q** was 0
  $M_{11}$ = the number of attributes where **p** was 1 and **q** was 1

- Simple Matching and Jaccard Coefficients

  SMC = number of matches / number of attributes
  $= (M_{11} + M_{00}) / (M_{01} + M_{10} + M_{11} + M_{00})$

  J = number of $M_{11}$ matches / number of not-both-zero attributes values
  $= (M_{11}) / (M_{01} + M_{10} + M_{11})$

# SMC versus Jaccard: Example

**p** = 1 0 0 0 0 0 0 0 0 0
**q** = 0 0 0 0 0 0 1 0 0 1

$M_{01} = 2$ (the number of attributes where p was 0 and q was 1)
$M_{10} = 1$ (the number of attributes where p was 1 and q was 0)
$M_{00} = 7$ (the number of attributes where p was 0 and q was 0)
$M_{11} = 0$ (the number of attributes where p was 1 and q was 1)

SMC $= (M_{11} + M_{00})/(M_{01} + M_{10} + M_{11} + M_{00}) = (0+7) / (2+1+0+7) = 0.7$

J $= (M_{11}) / (M_{01} + M_{10} + M_{11}) = 0 / (2 + 1 + 0) = 0$

# Cosine Similarity

| TID | W1 | W2 | W3 | W4 | W5 |
|---|---|---|---|---|---|
| D1 | 4 | 3 | 0 | 0 | 0 |
| D2 | 3 | 5 | 0 | 1 | 0 |
| D3 | 0 | 2 | 0 | 0 | 0 |
| D4 | 0 | 0 | 3 | 0 | 0 |
| D5 | 0 | 0 | 0 | 0 | 0 |

If $D_1$ and $D_2$ are two document vectors, then

$$\cos( D_1, D_2 ) = (D_1 \bullet D_2) / ||D_1||.||D_2|| ,$$

where • indicates vector dot product and || D || is the length of vector D.

Example:

$\cos(D_1, D_2)$ =
(4*3 + 3*5) / ( sqrt(4^2 + 3^2) * sqrt(3^2 + 5^2 + 1^2) )
=.91

If cosine similarity is 1, the angle between $D_1$ and $D_2$ is 0°, and $D_1$ and $D_2$ are the same except for the magnitude.

If the cosine similarity is 0, then the angle between $D_1$ and $D_2$ is 90°, and they don't share any terms (words).

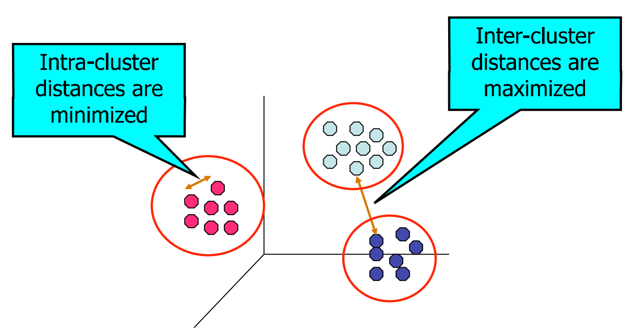
# TYPES OF CLUSTERS IN PRACTICE

# Administrivia

- No class for the rest of the week
  - Reading break
- Assignment 3 is out tomorrow
  - TWO implementation questions
- Mid-term project reports due today
- Project presentations start Nov 23
  - two weeks from tomorrow!!

# What is Cluster Analysis?

- Finding groups of objects
  - such that the objects in a group will be **similar** to one another and
  - **dissimilar** from the objects in other groups



# Solution Format

- Clustering is NP-hard, but we can use heuristics
- A solution would be an assignment for each data point to a cluster. Say we have 10 data points and 3 clusters:

  0 0 0 1 1 1 2 2 2 0
- How can we measure the goodness of a solution?
  - To keep it simple, let's just consider intra-cluster distance
  - Each cluster i has a mean point, called the centroid $\mu_i$

# Types of Clusters: Well-Separated

- Well-Separated Clusters:
  - Any point in a cluster is closer (or more similar) **to every other point** in the cluster than to any point not in the cluster.

**3 well-separated clusters**

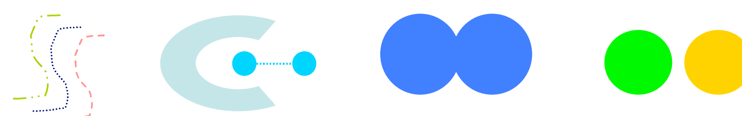
# Types of Clusters: Center-Based

- Center-based
  - An object in a cluster is **closer** (more similar) **to the "center"** of a cluster, **than to the center of any other cluster**
    - The center of a cluster is often a centroid, the average of all the points in the cluster, or a medoid, the most "representative" point of a cluster

**4 center-based clusters**

# Types of Clusters: Contiguity-Based

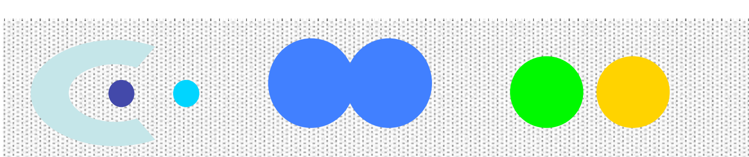
- Contiguous Cluster (Nearest neighbor or Transitive)
  - A point in a cluster is **closer** (or more similar) **to one** or more other points in the cluster **than to any point not in the cluster**.



**8 contiguous clusters**

# Types of Clusters: Density-Based

- Density-based
  - A cluster is a **dense** region of points, which is **separated by low-density** regions, from other regions of high density.



**6 density-based clusters**

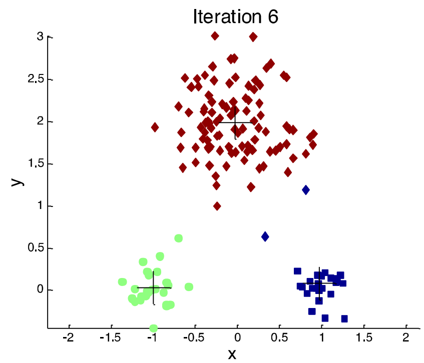
# ALGORITHMS

# K-means Clustering

- Each cluster is associated with a centroid (center point)
- Each point is assigned to the cluster with the closest centroid
- Number of clusters, *K*, must be specified
- Basic algorithm is very simple

1: Select $K$ points as the initial centroids.
2: **repeat**
3: Form $K$ clusters by assigning all points to the closest centroid.
4: Recompute the centroid of each cluster.
5: **until** The centroids don't change

# Example




# K-means Clustering – Details

- Initial centroids may be chosen randomly.
  - Clusters produced vary from one run to another.
  - Rerun several times and pick the clustering with the smallest SSE (see next slide).
- The centroid is (typically) the mean of the points in the cluster.
- 'Closeness' is measured by Euclidean distance, cosine similarity, etc.
- Most of the convergence happens in the first few iterations.
  - Often the stopping condition is changed to 'Until relatively few points change clusters'

# Evaluating K-means Clusters

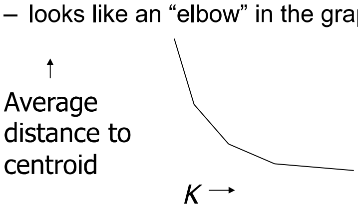
- Most common measure is Sum of Squared Error (SSE)
  - For each point, the error is the distance to the nearest centroid
  - To get SSE, we square these errors and sum them up.

$$SSE = \sum_{i=1}^{K} \sum_{x \in C_i} [dist(\mu_i, x)]^2$$

$x$ is a data point in cluster $C_i$ and
$\mu_i$ is the centroid for cluster $C_i$

# How to choose K?

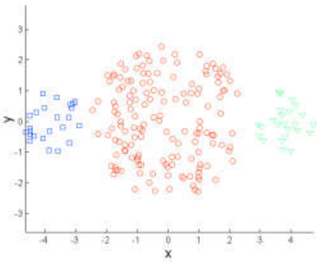
- As K grows, SSE will fall.
  - How low can SSE be?
  - For what setting of K will you have that SSE?
- So how to choose K?
- Try different *K*, looking at the change in the average distance to centroid, as *K* increases.
- Average falls rapidly until "right" *K*, then changes little.
  - looks like an "elbow" in the graph



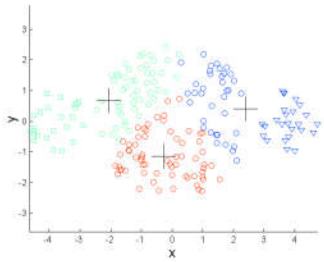

# Limitations of K-means

- K-means has problems when (the real) clusters are of
  - Differing Sizes
  - Differing Densities
  - Non-globular shapes

## Limitations of K-means: Differing Sizes

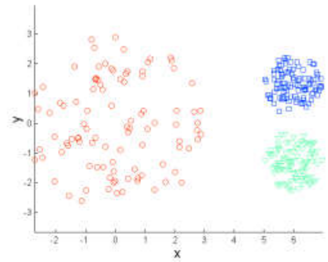


**Original Points**

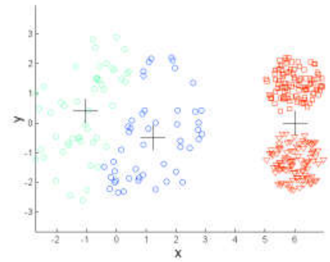


**K-means (3 Clusters)**

## Limitations of K-means: Differing Density

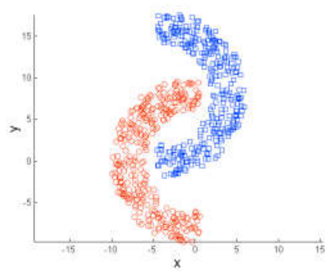


**Original Points**

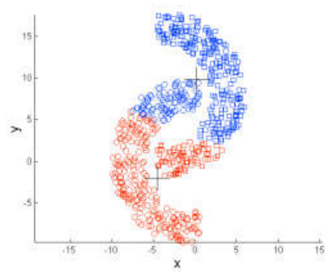


**K-means (3 Clusters)**

## Limitations of K-means: Non-globular Shapes

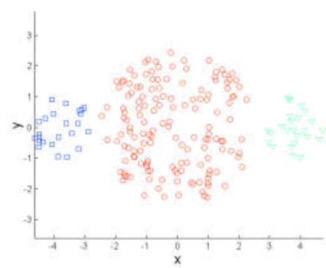


**Original Points**

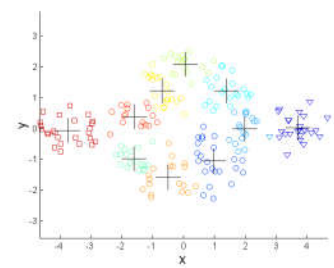


**K-means (2 Clusters)**

## Overcoming K-means Limitations



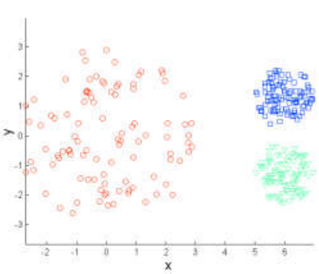
**Original Points**



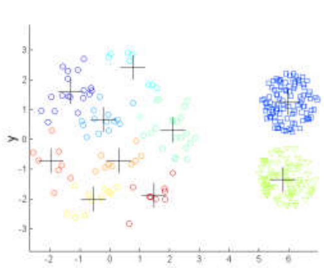
**K-means Clusters**

One solution is to use many clusters.
- Find parts of clusters.
- Apply **merge** strategy (merge clusters that would cause the least increase in SSE)

## Overcoming K-means Limitations

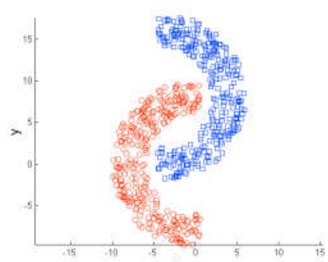


**Original Points**

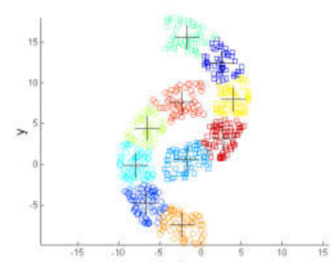


**K-means Clusters**

## Overcoming K-means Limitations

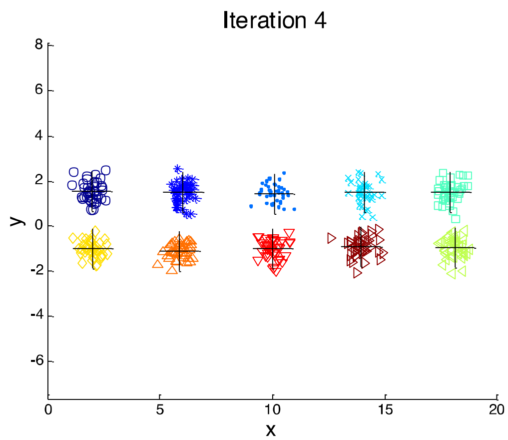


**Original Points**



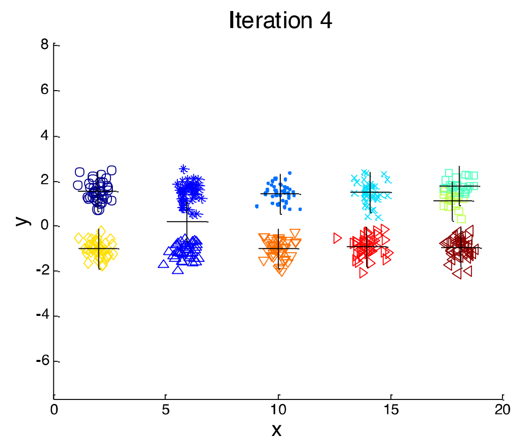
**K-means Clusters**

# Importance of Choosing Initial Centroids




**Starting with two initial centroids in one cluster of each pair of clusters**

# Importance of Choosing Initial Centroids




**Starting with some pairs of clusters having three initial centroids, while other have only one.**

# Problems with Selecting Initial Points

- The ideal would be to choose initial centroids, one from each true cluster. However, this is very difficult.
- If there are $K$ 'real' clusters, then the chance of selecting one centroid from each cluster is small.
  - Chance is relatively small when $K$ is large
    - E.g. If clusters are the same size, $n$, then

$$P = \frac{\text{number of ways to select one centroid from each cluster}}{\text{number of ways to select } K \text{ centroids}} = \frac{K!n^K}{(Kn)^K} = \frac{K!}{K^K}$$

- For example, if $K = 10$, then *probability* = $10!/10^{10} = 0.00036$

# Solutions to Initial Centroids Problem

- Multiple runs
  - Helps, but probability is not on your side
- Bisecting K-means
  - Not as susceptible to initialization issues

# Bisecting K-means

- Straightforward extension of the basic K-means algorithm. Simple idea:

  To obtain K clusters, split the set of points into two clusters, select one of these clusters to split, and so on, until K clusters

**Algorithm (Parameters: K, num_trials)**

Initialize -> one cluster consisting of all points. k = 1

**while k < K**

Choose and remove a cluster from the list of clusters.

(biggest cluster or the cluster with the worst quality)

//Perform several "trial" bisections of the chosen cluster.

**for** $i$ = 1 **to** num_trials **do**

Bisect the selected cluster using basic $K$-means (i.e. 2-means).

Record SSE for this bisection

**end for**

Select the two-clustering trial with the lowest total SSE.

Add these two clusters to the list of clusters.

k = k+1

**end for**

# Bisecting K-means Example