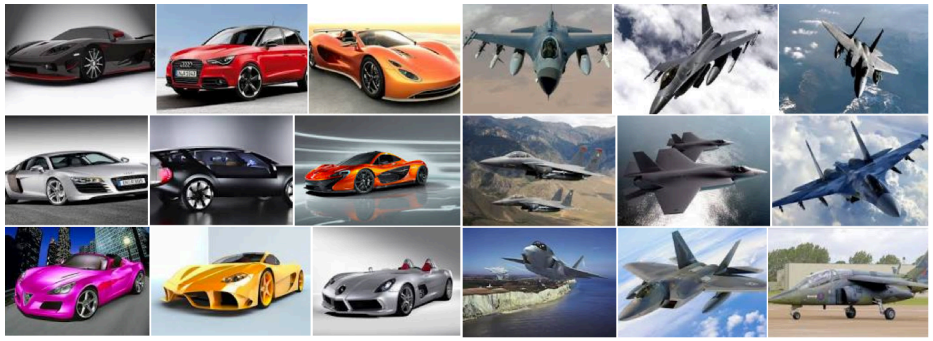
Clustering

Ishwar K Sethi

What is Clustering?



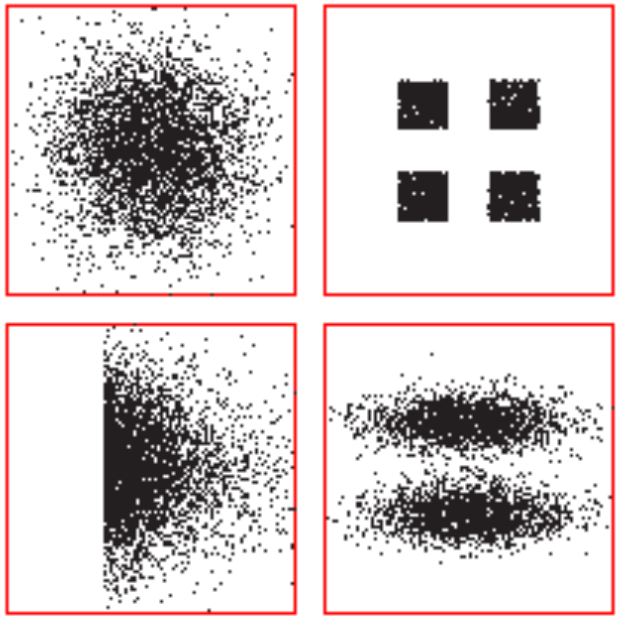
Clustering is the process of organizing objects into groups whose members are similar in some way.

Clustering is also a way of learning, for example being able to decide whether an object should be placed in group one or group two. Since there is no teacher to illustrate examples from different groups (no class labels), the learning in clustering is often called unsupervised learning.

Unsupervised Learning

- Unsupervised learning refers to situations where we have a collection of data, but each data instance is unlabeled/unmarked. In such situations, the best we can do is to organize data into groups for further analysis
- The process of grouping data is known as *Clustering*. Clustering has wide applications and is known through a variety of names - *unsupervised classification, Q analysis, typology, numerical taxonomy, and market segmentation*

Why Clustering?



These four data sets have identical first-order and second-order statistics. We need to find other ways of modeling their structure. Clustering is an alternative way of describing the data in terms of groups of patterns.

Clustering Example

Market segmentation to group customers into different groups

Clustering Example

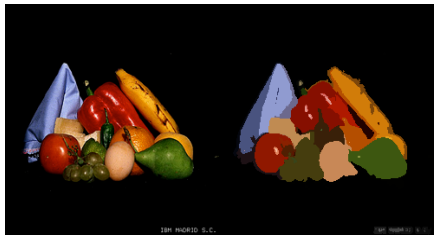
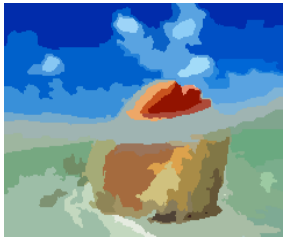
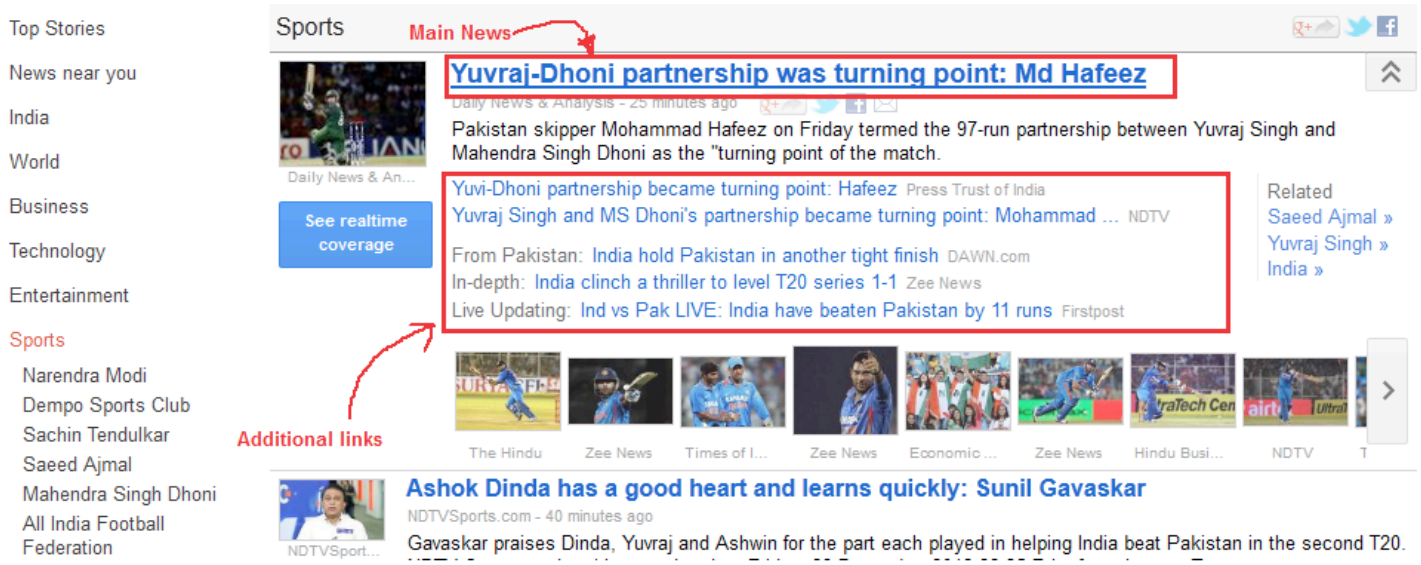


Image segmentation to group pixels into meaningful regions

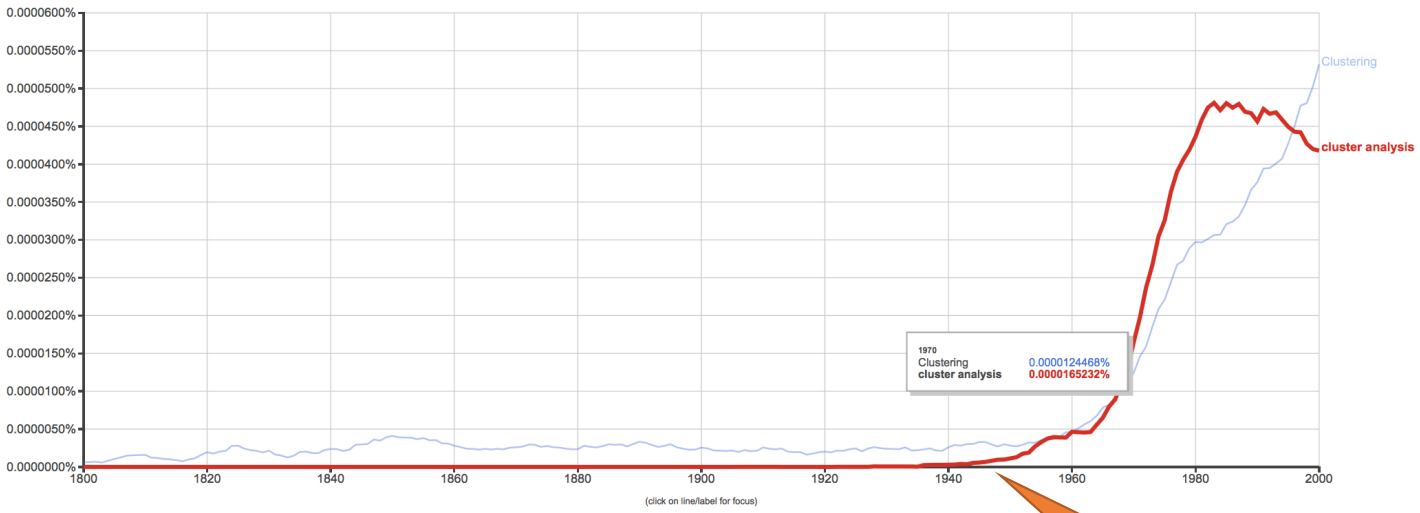


Clustering Example



Clustering news stories

History of Clustering



Google Books Ngram Viewer

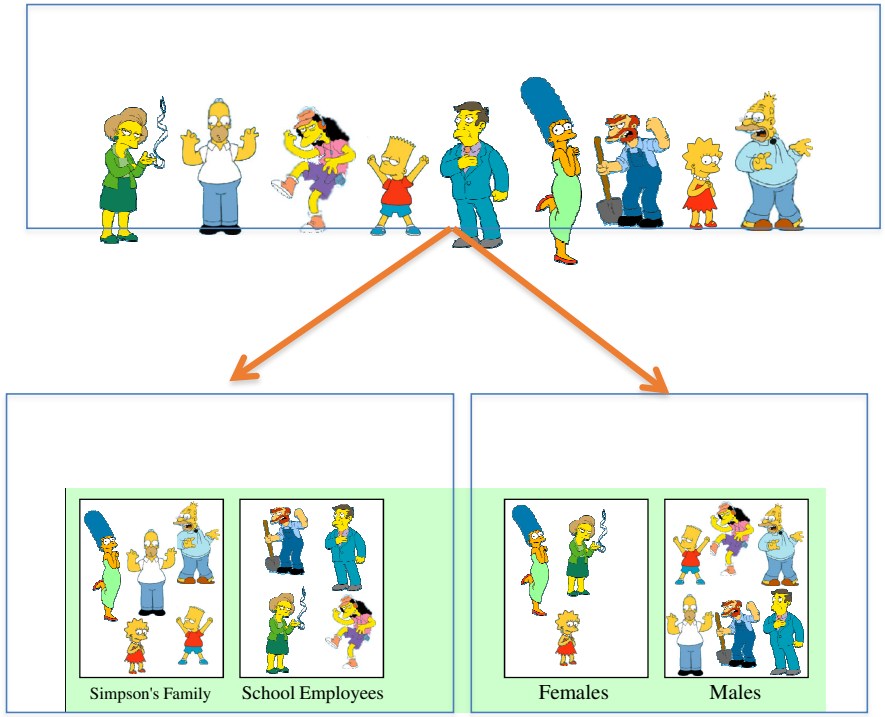
Beginning of Clustering

Books.google.com/ngrams

How many clusters?

How many clusters should be formed is not an easy number to determine. Domain knowledge is always helpful in this regard.

Multiple Clustering Results are Possible



Results must be validated against knowledge about the application domain.

Similarity Measures

The basis of clustering lies in measuring similarity between a pair of objects

- A general class of metrics for d-dimensional patterns is the *Minkowski metric*

$$L_p(\mathbf{x}, \mathbf{y}) = \left(\sum_{i=1}^{d} |\mathbf{x}_i - \mathbf{y}_i|^p \right)^{1/p}$$

  also referred to as the L_p *norm*.

- The *Euclidean distance* is the L_2 norm

$$L_2(\mathbf{x}, \mathbf{y}) = \left(\sum_{i=1}^{d} |\mathbf{x}_i - \mathbf{y}_i|^2 \right)^{1/2}$$

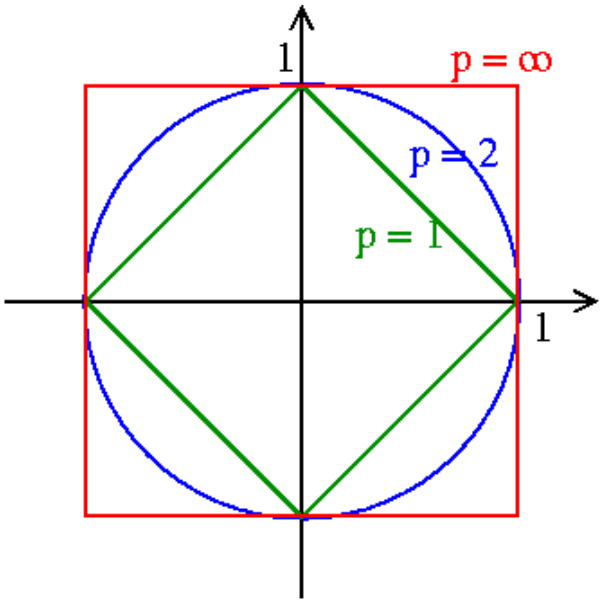
- The *Manhattan* or *city block distance* is the L_1 norm

$$L_1(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^{d} |\mathbf{x}_i - \mathbf{y}_i|$$

- The L_∞ norm is the maximum of the distances along individual coordinate axes

$$L_\infty(\mathbf{x}, \mathbf{y}) = \max_{i=1}^{d} |\mathbf{x}_i - \mathbf{y}_i|$$

Minkowski Metric: Contours of Constant Distance



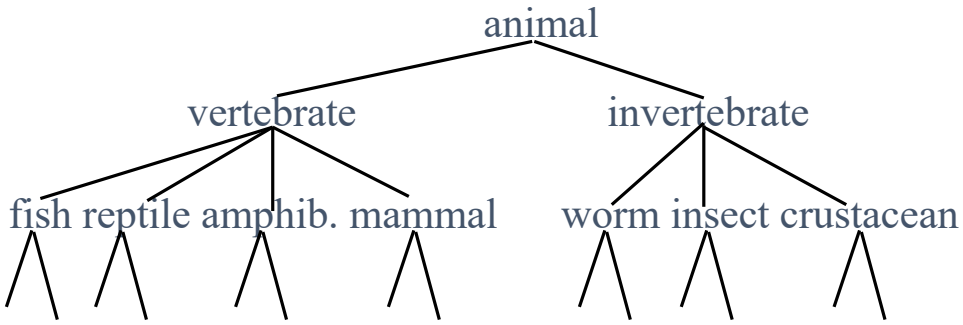
Loci of points at unit distance in 2-D are shown. The value of p indicates the metric being used.

Taxonomy of Clustering Methods

- Hierarchical
  - Agglomerative
  - Divisive
- Partitional
  - Sequential or simultaneous procedures
  - Direct or indirect methods

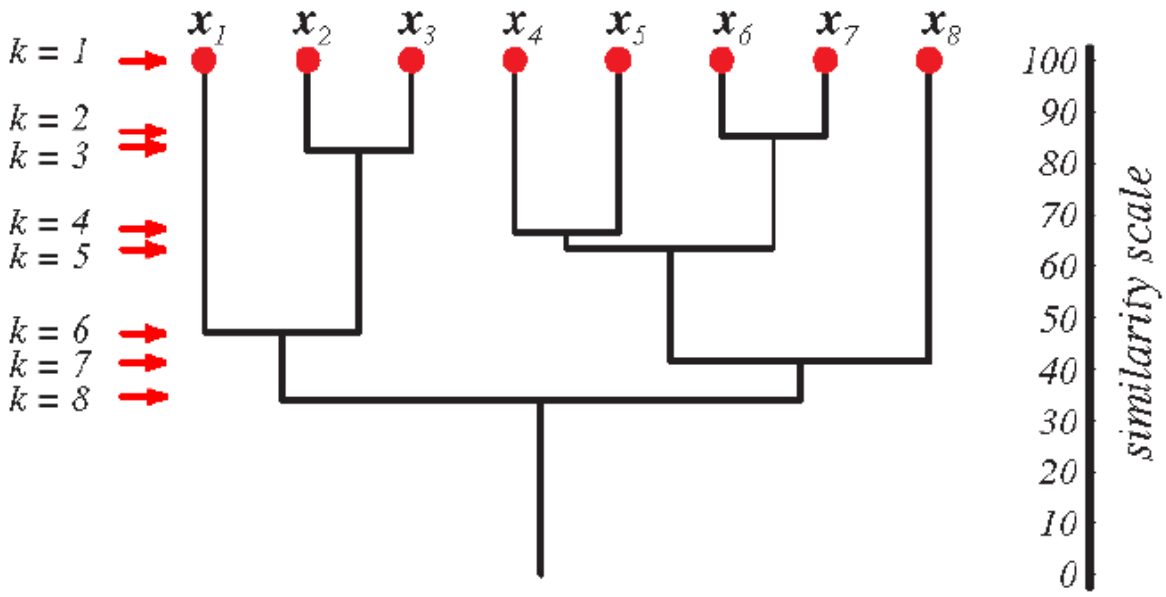
Hierarchical Clustering

- Builds a tree-based hierarchical taxonomy (*dendrogram*) from a set of examples.



- Recursive application of a standard clustering algorithm can produce hierarchical clustering.

Dendrogram


x1
x2
x3
x4
x5
x6
x7
x8
k = 1
k = 2
k = 3
k = 4
k = 5
k = 6
k = 7
k = 8
100
90
80
70
60
50
40
30
20
10
0
similarity scale

Agglomerative Vs Divisive Clustering

- *Agglomerative* (*bottom-up*) methods start with each example in its own cluster and iteratively combine them to form larger and larger clusters.
- *Divisive* (*partitional, top-down*) separates all examples immediately into clusters.

Direct Clustering Methods

- *Direct clustering* methods require a specification of the number of clusters, *k*, desired.
  - A *clustering evaluation function* assigns a real-value quality measure to a clustering.
  - The number of clusters can be determined automatically by explicitly generating clustering for multiple values of *k* and choosing the best result according to a clustering evaluation function.

How Many Clusters?

- Statistical significance of differences between clusters
- Cluster sizes
- Meaningful cluster profiles
- Aggregation or decomposition patterns of clusters at different stages of clustering

Hierarchical Agglomerative Clustering

- Assumes a *similarity function* for determining the similarity of two instances.
- Starts with all instances in separate clusters and then repeatedly joins the two clusters that are most similar until there is only one cluster.
- The history of merging forms a binary tree or hierarchy.

Cluster Similarity

- How to compute similarity of two clusters each possibly containing multiple instances?
  - Single Link: Similarity of two most similar members.
  - Complete Link: Similarity of two least similar members.
  - Group Average: Average similarity between members.
  - Mean : Similarity between the two cluster means

Cluster Similarity

Popular distance measures (for two clusters $\mathcal{D}_i$ and $\mathcal{D}_j$):

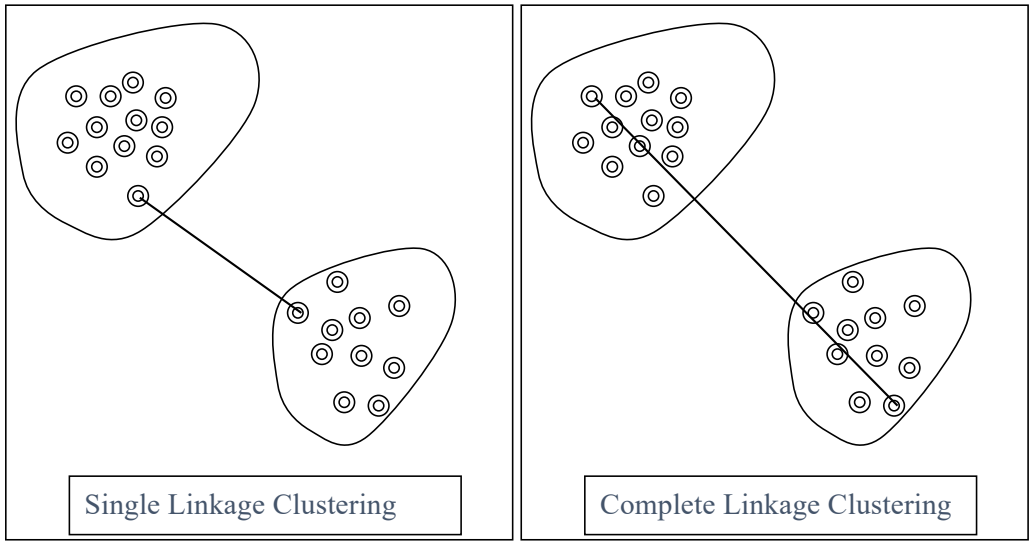
$$d_{\text{min}}(\mathcal{D}_i, \mathcal{D}_j) = \min_{\substack{\mathbf{x} \in \mathcal{D}_i \\ \mathbf{x}' \in \mathcal{D}_j}} \|\mathbf{x} - \mathbf{x}'\|$$

$$d_{\text{max}}(\mathcal{D}_i, \mathcal{D}_j) = \max_{\substack{\mathbf{x} \in \mathcal{D}_i \\ \mathbf{x}' \in \mathcal{D}_j}} \|\mathbf{x} - \mathbf{x}'\|$$

$$d_{\text{avg}}(\mathcal{D}_i, \mathcal{D}_j) = \frac{1}{\#\mathcal{D}_i \, \#\mathcal{D}_j} \sum_{\mathbf{x} \in \mathcal{D}_i} \sum_{\mathbf{x}' \in \mathcal{D}_j} \|\mathbf{x} - \mathbf{x}'\|$$

$$d_{\text{mean}}(\mathcal{D}_i, \mathcal{D}_j) = \|\mathbf{m}_i - \mathbf{m}_j\|$$

Popular Agglomerative Clustering Procedures



a.k.a. nearest neighbor clustering

a.k.a. furthest neighbor clustering

Hierarchical Clustering Example

Let us consider five examples: A, B, C, D, and E. Let the interpoint distances between these examples be given by the following distance matrix.

$$\mathbf{DM}(1) = \begin{array}{c|ccccc} & A & B & C & D & E \\ A & 0 & & & & \\ B & 1 & 0 & & & \\ C & 5 & 3 & 0 & & \\ D & 6 & 8 & 4 & 0 & \\ E & 8 & 7 & 6 & 2 & 0 \end{array}$$

Hierarchical Clustering Example

Using the *nearest neighbor* measure, also known as the *single linkage* measure, we merge A and B to form a cluster, since they are closest. Next we compute the distances between this cluster and the remaining examples. We can get these distances from the above distance matrix. The values for these are as follows:

$d_{(AB)C} = min\{d_{AC}, d_{BC}\} = d_{BC} = 3$

$d_{(AB)D} = min\{d_{AD}, d_{BD}\} = d_{AD} = 6$

$d_{(AB)E} = min\{d_{AE}, d_{BE}\} = d_{BE} = 7$

At this point we can form an updated distance matrix. This is given as:

$$\mathbf{DM}(2) = \begin{array}{c|cccc} & AB & C & D & E \\ AB & 0 & & & \\ C & 3 & 0 & & \\ D & 6 & 4 & 0 & \\ E & 7 & 6 & 2 & 0 \end{array}$$

Hierarchical Clustering Example

Since the smallest entry in above distance matrix is 2, examples D and E are merged to form another cluster. At this point, we repeat the distance calculations to obtain the following set of values:

$$d_{(AB)C} = 3$$

$$d_{(AB)(DE)} = min\{d_{AD}, d_{AE}, d_{BD}, d_{BE}\} = d_{AD} = 6$$

$$d_{(DE)C} = min\{d_{CD}, d_{CE}\} = d_{CD} = 4$$

The new distance matrix thus becomes

$$\mathbf{DM}(3) = \begin{array}{c|ccc} & AB & C & DE \\ AB & 0 & & \\ C & 3 & 0 & \\ DE & 6 & 4 & 0 \end{array}$$

This matrix indicates that C should be merged with A and B. At this stage we have only two clusters left that are joined to form a single cluster of five examples.

Linkage Function in Scipy

- The calculations shown in previous slides for agglomerative clustering are performed by the *linkage* function of the Scipy library. The linkage function accepts distance matrix as input, and you can specify the clustering similarity criterion.
- A simple example of using the linkage function and creating a dendrogram is shown in the next slide.

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn import datasets
```

```
X,y = datasets.make_blobs(n_samples=10,n_features=2, random_state=5)
```

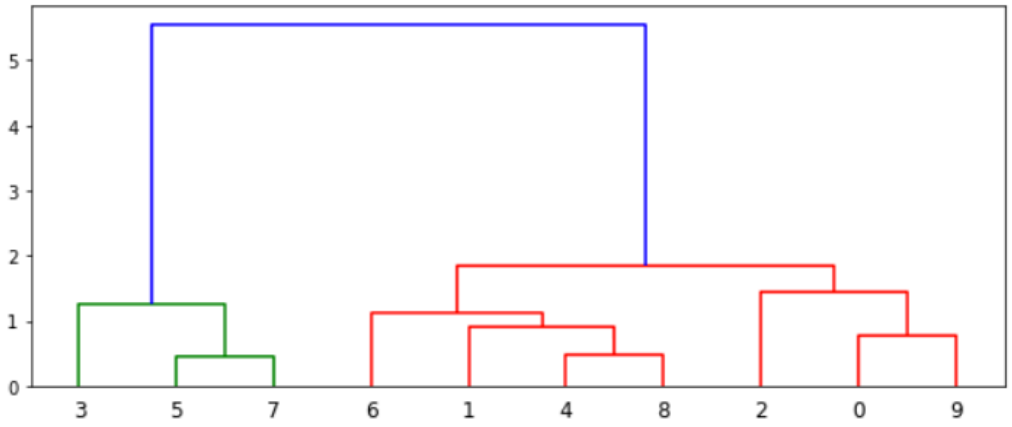
```
from scipy.spatial import distance
```

```
Y = distance.pdist(X, 'euclidean')
```

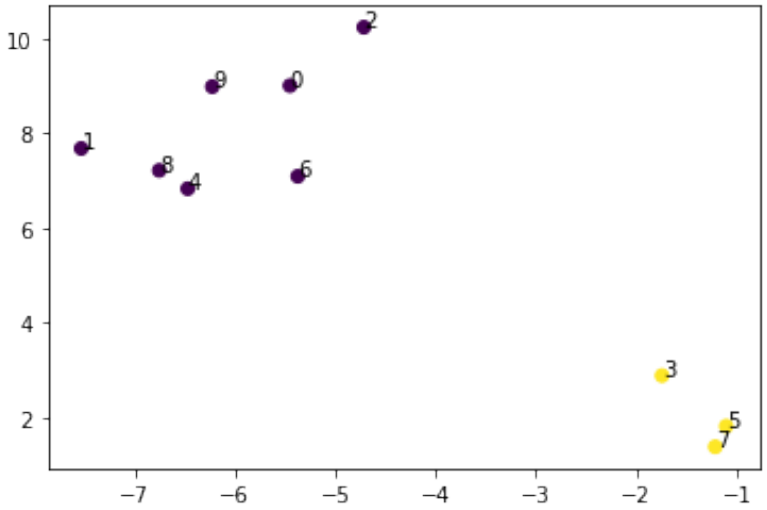
Form the distance matrix

```
from scipy.cluster.hierarchy import dendrogram, linkage
```

```
z = linkage(Y,'single')
fig = plt.figure(figsize=(10, 4))
dn = dendrogram(z)
plt.show()
```



Scatter Plot of the Data. The points #5 and #7 are closest. So they are merged first as shown by the dendrogram.



Agglomerative Clustering of Text Documents Example

- This simple example shows the clustering of a small set of documents.

```
mycorpus = ['The sky is blue and beautiful.',
        'Love this blue and beautiful sky!',
        'The quick brown fox jumps over the lazy dog.',
        "A king's breakfast has sausages, ham, bacon, eggs, toast and beans",
        'I love green eggs, ham, sausages and bacon!',
        'The brown fox is quick and the blue dog is lazy!',
        'The sky is very blue and the sky is very beautiful today',
        'The dog is lazy but the brown fox is quick!']
```

We had earlier obtained the td-idf representation of these documents while discussing information retrieval. We will use that representation to do agglomerative clustering of these 8 documents.

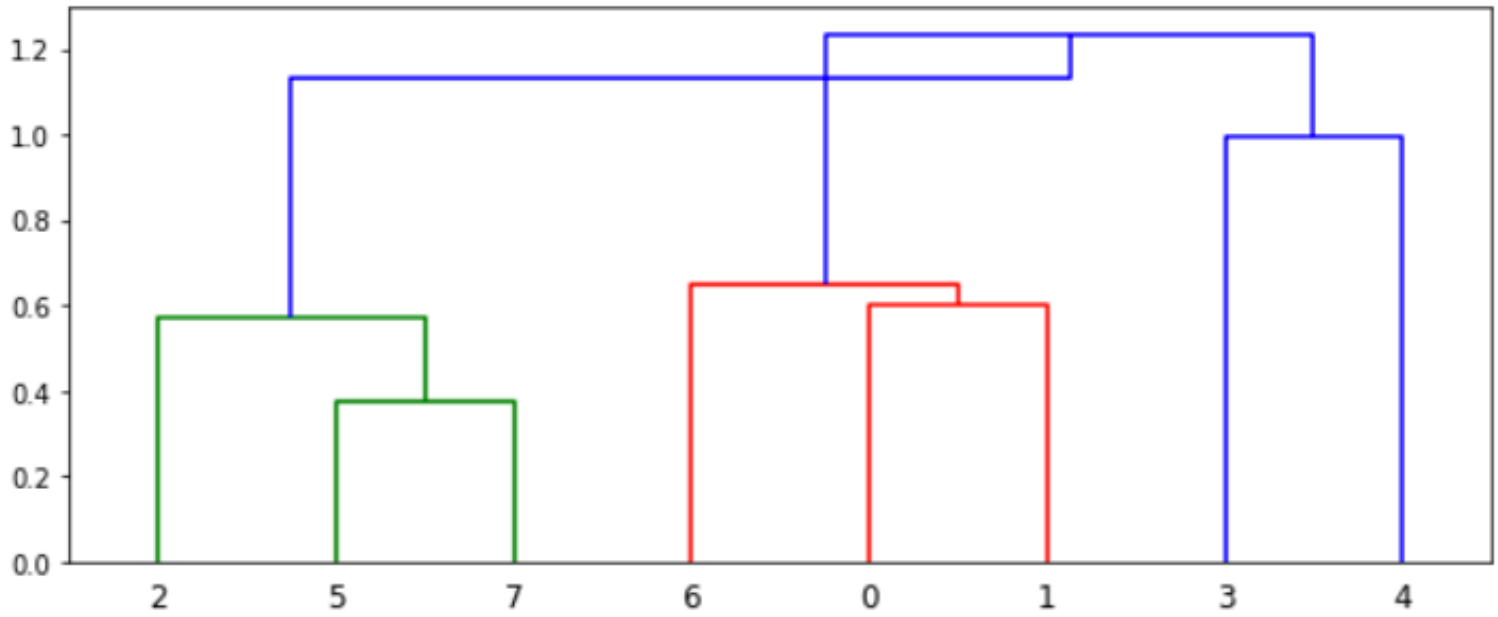
TD-IDF Representation of the Example Corpus

	bacon	beans	beautiful	blue	breakfast	brown	dog	eggs	fox	green	ham	jumps	king	lazy	love	quick	sausages	sky	toast	today
0	0.000000	0.000000	0.600978	0.526925	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.600978	0.000000	0.000000
1	0.000000	0.000000	0.493162	0.432394	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.571505	0.000000	0.000000	0.493162	0.000000	0.000000
2	0.000000	0.000000	0.000000	0.000000	0.000000	0.380362	0.380362	0.000000	0.380362	0.000000	0.000000	0.525949	0.000000	0.380362	0.000000	0.380362	0.000000	0.000000	0.000000	0.000000
3	0.321164	0.383215	0.000000	0.000000	0.383215	0.000000	0.000000	0.321164	0.000000	0.000000	0.321164	0.000000	0.383215	0.000000	0.000000	0.000000	0.321164	0.000000	0.383215	0.000000
4	0.394554	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.394554	0.000000	0.470784	0.394554	0.000000	0.000000	0.000000	0.394554	0.000000	0.394554	0.000000	0.000000	0.000000
5	0.000000	0.000000	0.000000	0.365048	0.000000	0.416351	0.416351	0.000000	0.416351	0.000000	0.000000	0.000000	0.000000	0.416351	0.000000	0.416351	0.000000	0.000000	0.000000	0.000000
6	0.000000	0.000000	0.360826	0.316365	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.721652	0.000000	0.498935
7	0.000000	0.000000	0.000000	0.000000	0.000000	0.447214	0.447214	0.000000	0.447214	0.000000	0.000000	0.000000	0.000000	0.447214	0.000000	0.447214	0.000000	0.000000	0.000000	0.000000

```
Y = distance.pdist(Doc_tfidf_Matrix, 'euclidean')
```

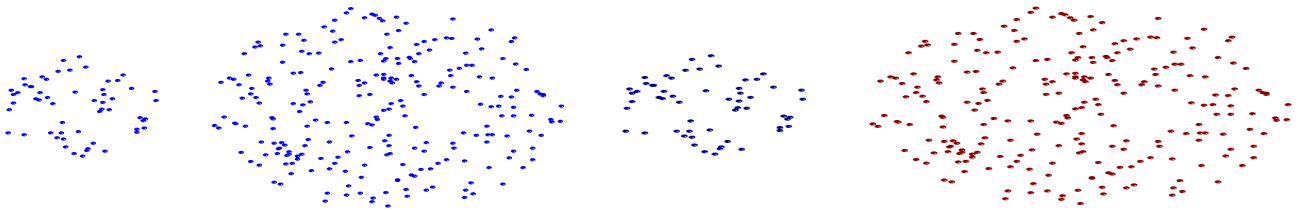
```
from scipy.cluster.hierarchy import dendrogram, linkage
```

```
z = linkage(Y,'centroid')
fig = plt.figure(figsize=(10, 4))
dn = dendrogram(z)
plt.show()
```



Single Linkage Behavior

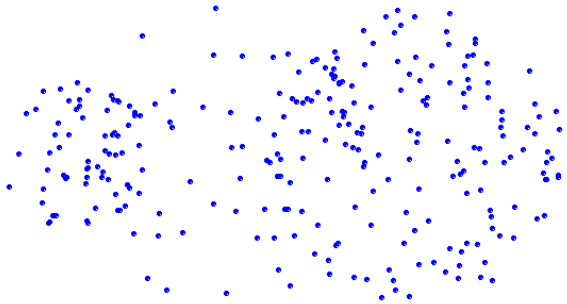
Can handle non-elliptical shapes



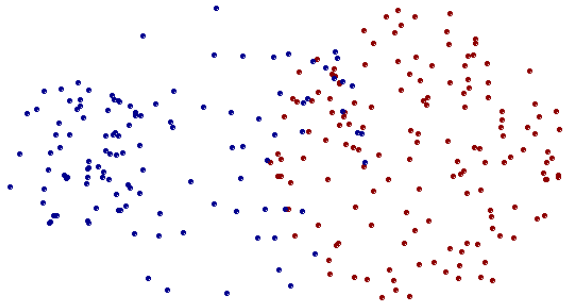
Original Points

Result (Two Clusters)

Single Linkage Behavior



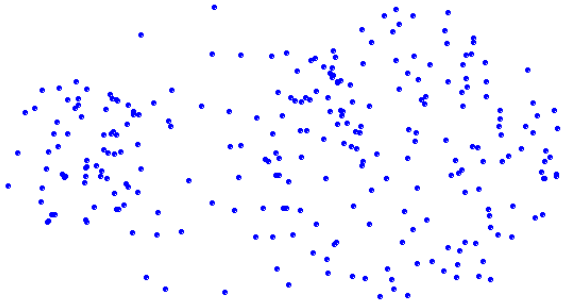
Original Points



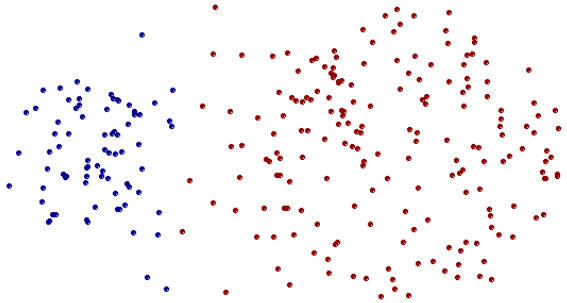
Resulting Two Clusters

Sensitive to noise and outliers

Complete Linkage Behavior



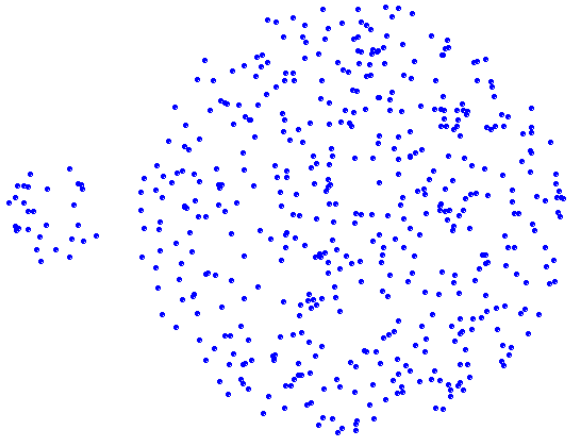
Original Points



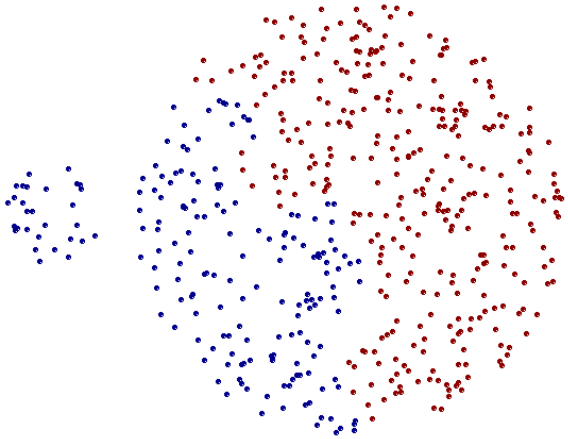
Resulting Two Clusters

Less sensitive to noise and outliers

Complete Linkage Behavior



Original Points



Two Clusters

Tends to break large clusters

Ward's Stepwise Optimal Hierarchical Clustering

- In this procedure, a clustering criterion, e.g. the sum of squares (SSE), is used to specify the quality of the clustering. The SSE is defined as:

$$SSE = \sum_{i=1}^{K} \sum_{\mathbf{x} \in C_i} ||\mathbf{m}_i - \mathbf{x}||^2,$$

where $\mathbf{m}_i$, the center of the *i-th* cluster with n_i data items, is defined by

$$\mathbf{m}_i = \frac{1}{n_i} \sum_{\mathbf{x} \in C_i} \mathbf{x}$$

The SSE criteria means that grouping of data items should be as compact as possible. In other words, data items in a cluster should reside close to the cluster center.

- While determining the pair-wise similarity of clusters for merging in the agglomerative process, the pair whose merger would increase the criterion function as little as possible is selected. [An iterative optimization algorithm described later uses the same idea for refining clustering result]

How to Choose the Number of Clusters in Hierarchical Clustering

- Lifetime Method
  - The *lifetime* of a cluster is defined as the absolute value of the difference between the dendrogram level at which it is created and the level at which it is absorbed into a larger cluster. Using lifetime as a criterion, a user can search for cluster that have a large lifetime.
- Self-similarity Measure Method
  - This method uses a function *h(C)* that measures the dissimilarity between the vectors of the same cluster *C*. A cutoff value for the selected measure can be used to control the number of clusters.
  - Examples of possible functions are

    $h(C) = max\{d(\mathbf{x}, \mathbf{y}),\ For\ all\ \mathbf{x}, \mathbf{y}\ from\ cluster\ C\}$

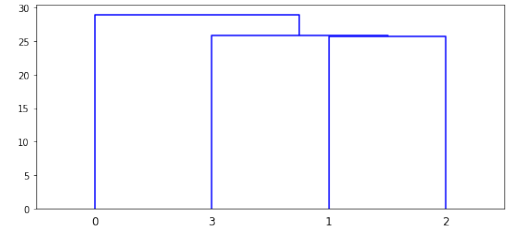
    $h(C) = med\{d(\mathbf{x}, \mathbf{y}),\ For\ all\ \mathbf{x}, \mathbf{y}\ from\ cluster\ C\}$

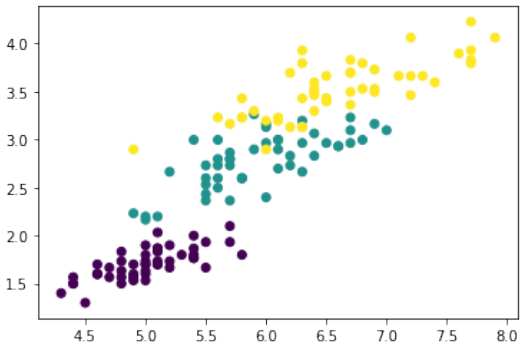
Feature Aggregation using Agglomerative Clustering

- The idea of agglomerative clustering can be used for *dimensionality reduction* as well. In place of constructing an *N-by-N* distance matrix of examples for aggregation, we construct a *d-by-d* correlation matrix and use this to perform agglomerative clustering. The result in this case is a set of new features fewer in number.

Feature Aggregation using Agglomerative Clustering Example

- Let's perform feature aggregation on Iris data. Recall that it has four features and 150 examples from three classes. Using 150 examples, we can construct a 4-by-4 matrix which captures correlations between these features. We pass this onto the *linkage* function and get the dendrogram as shown below.





The dendrogram shows merger of features 1,2, and 3. Thus we can create a new feature as an average of the features 1,2, and 3. This leaves us with two features, the original feature 0 and the new aggregated feature. Let's do a scatter plot of our examples with these two features. As you can see, the plot clearly maintains separation between the three Iris classes.

Direct Clustering : K-Means Clustering

- Most popular clustering algorithm. It assumes instances are real-valued vectors.
- Clustering is performed to minimize the SSE:

$$SSE = \sum_{i=1}^{K} \sum_{\mathbf{x} \in C_i} ||\mathbf{m}_i - \mathbf{x}||^2,$$

where $\mathbf{m}_i$, the center of the *i-th* cluster with n_i data items, is defined by

$$\mathbf{m}_i = \frac{1}{n_i} \sum_{\mathbf{x} \in C_i} \mathbf{x}$$

The SSE criteria means that grouping of data items should be as compact as possible. In other words, data items in a cluster should reside close to the cluster center.

What modification will be needed if attributes are not real valued?

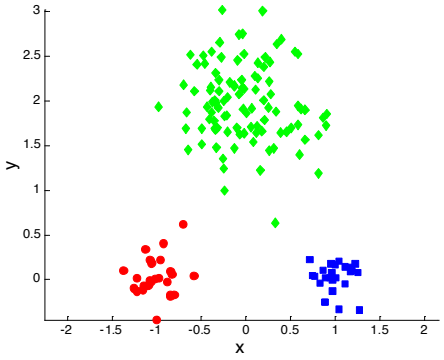
K-Means Clustering Steps

- Assumes the number of clusters, *K*, is given. Randomly initialize all cluster centers.
- Select an example and assign it to the cluster whose center is closest. Repeat this for all examples.
- Update the cluster centers based on the assignments of the previous step.
- Again assign and perform update. This is continued till cluster centers are stabilized.

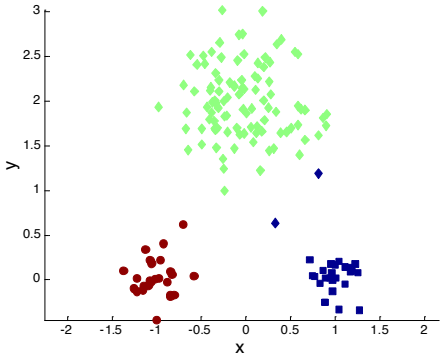
K-Means Clustering : Initialization of Centers

- Results can vary based on random initialization or seed selection.
- Some seeds can result in poor convergence rate, or convergence to sub-optimal clustering.
- It is a good practice to select good seeds using a heuristic or the results of another method.

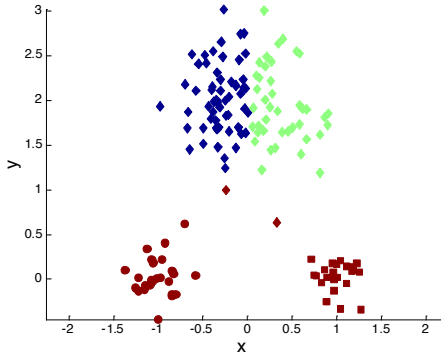
K-Means Clustering Can Yield Different Results



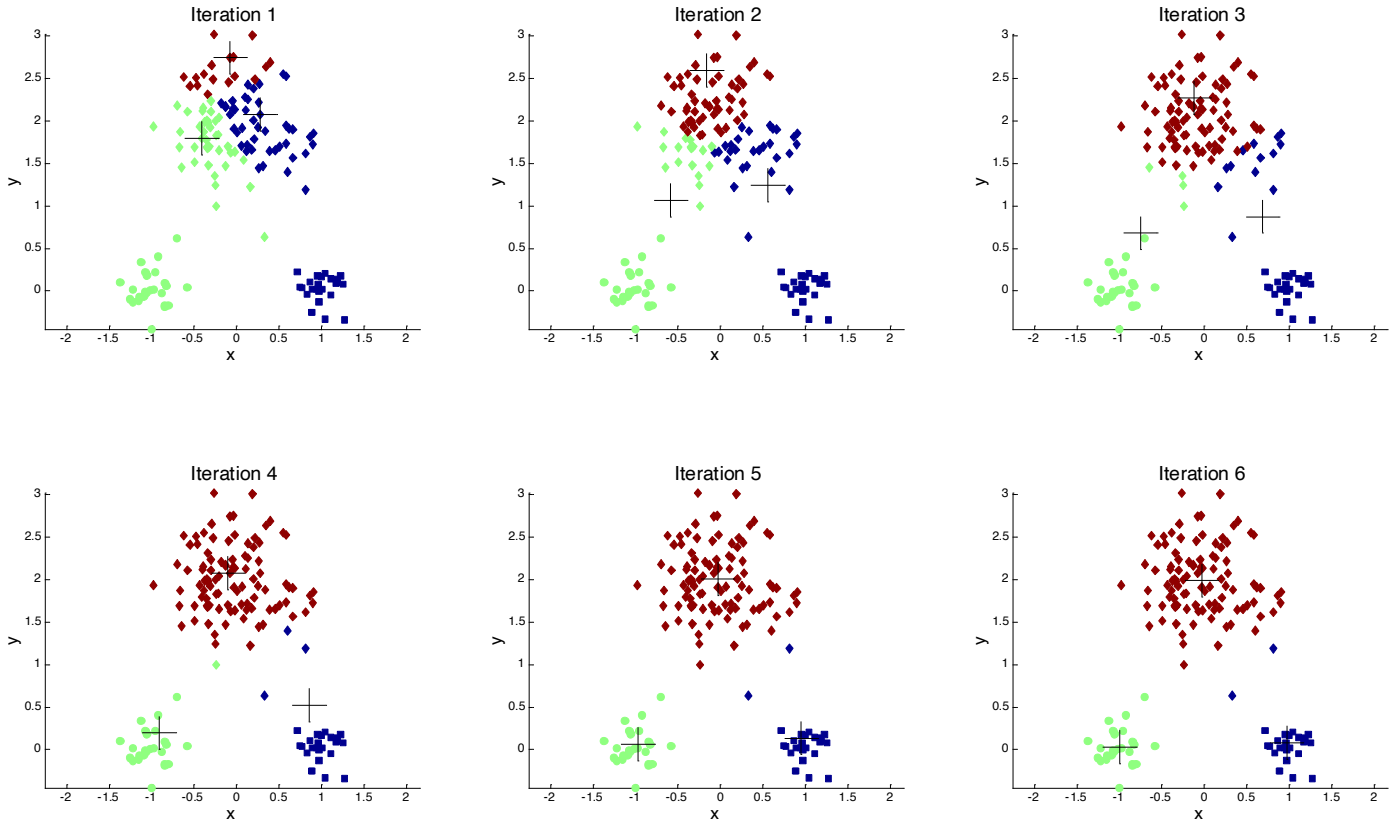
Original Points



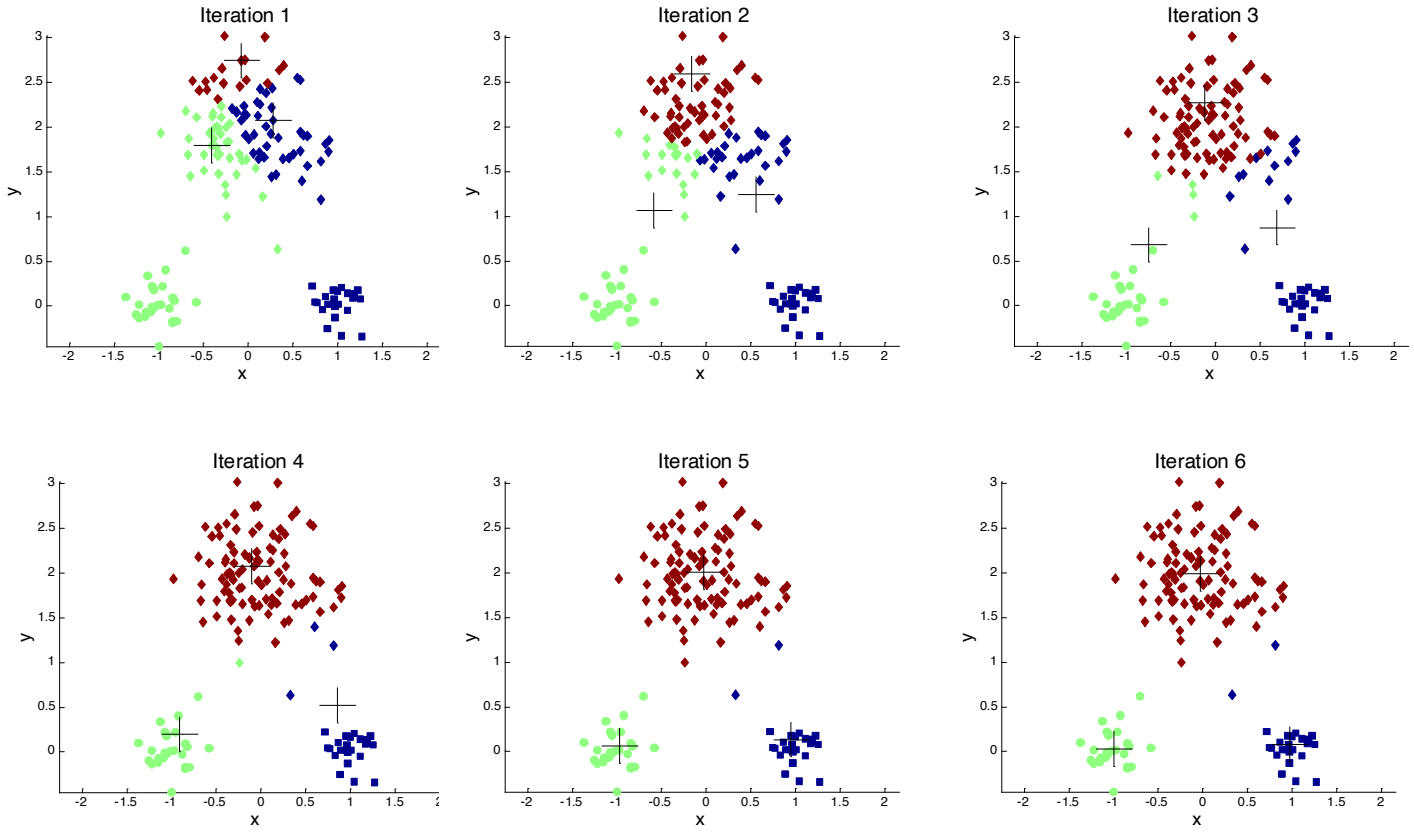
Optimal Clustering



Sub-optimal Clustering


Iteration 1
Iteration 2
Iteration 3
Iteration 4
Iteration 5
Iteration 6
x
y

Results with Different Seed Points



For an Excel implementation of K-means and how the centers get updated, see the following post:

https://iksinc.online/2017/10/05/k-means-demonstration-using-excel/

Evaluating K-Means Clusters

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

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

  - Given two clusters, we can choose the one with the smallest error
  - One easy way to reduce SSE is to increase K, the number of clusters. A good clustering with smaller K can have a lower SSE than a poor clustering with higher K

K-means Illustration

- USArrests data containing arrest information for 50 states. The information is arrest rates for three crimes per 100,000 residents. The other attribute is urban population percentage
- After scaling/normalization the data looks like as shown below

	Murder	Assault	UrbanPop	Rape
Alabama	1.25517927	0.7907871	-0.5261951	-0.0034511
Alaska	0.51301858	1.1180595	-1.2240666	2.5094239
Arizona	0.07236067	1.4938168	1.0091222	1.0534662
Arkansas	0.23470832	0.2332119	-1.0844923	-0.1867939
California	0.28109336	1.2756352	1.7767809	2.0888139

Q. Why are some rates negative?

```
import pandas as pd
#Read data
df = pd.read_csv(USArrests.csv)#This loads the file
```

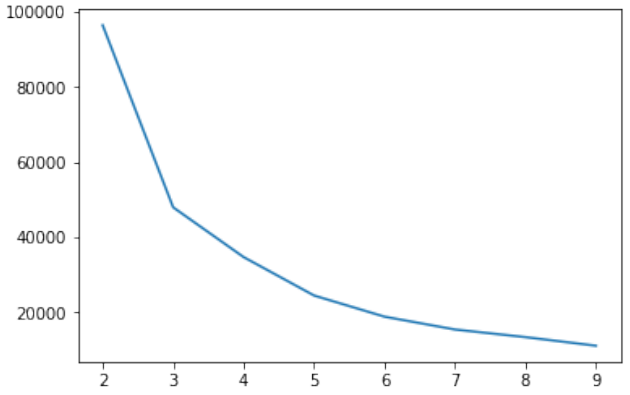
```
print(df.describe())
```

	Murder	Assault	UrbanPop	Rape
count	50.00000	50.000000	50.000000	50.000000
mean	7.78800	170.760000	65.540000	21.232000
std	4.35551	83.337661	14.474763	9.366385
min	0.80000	45.000000	32.000000	7.300000
25%	4.07500	109.000000	54.500000	15.075000
50%	7.25000	159.000000	66.000000	20.100000
75%	11.25000	249.000000	77.750000	26.175000
max	17.40000	337.000000	91.000000	46.000000

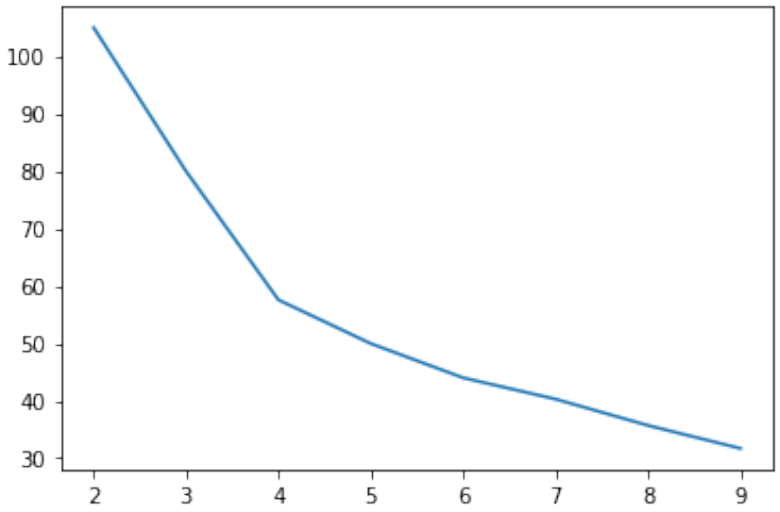
Data summary

```
# Without normalization
from sklearn import cluster
SSEU =[]# Unnormalized
for k in range (8):
    k_means = cluster.KMeans(n_clusters=k+2)
    k_means.fit(X)
    SSEU.append(k_means.inertia_)
```

SSE plot against K. A knee point is visible at K = 3.



```
# With normalization
from sklearn.preprocessing import scale
Xs = scale(X)
```



A knee point now appears at K = 4

```
labels = k_means.labels_
print(labels[0:5])
```

```
[1 3 3 1 3]
```

```
Centers = k_means.cluster_centers_
print(Centers)
```

```
[[-0.98483178 -1.14153431 -0.99725843 -1.01543161]
 [ 1.42622412  0.88321132 -0.82279055  0.01946669]
 [-0.51290944 -0.41277681  0.53292755 -0.27878259]
 [ 0.70212683  1.04999438  0.72997363  1.28990383]]
```

Cluster labels for the first five states: Alabama, Alaska, Arizona, Arkansas, and California. Alabama and Arkansas are in the same cluster (#1). Alaska, Arizona, and California are together in cluster #3.

A Hybrid Algorithm to Tackle Seed Points Selection

- Combines HAC (Hierarchical Agglomerative Clustering) and K-Means clustering.
- First randomly take a sample of instances of size $\sqrt{n}$
- Run group-average HAC on this sample, which takes only O(n) time.
- Use the results of HAC as initial seeds for K-means.
- Overall algorithm is O(n) and avoids problems of bad seed selection.

Clustering for Large Data Sets

- Desirable attributes of a clustering procedure for very large data sets
  - No more than one pass (scan) of the database
  - Should allow for incremental updates of results as more data becomes available
  - Work with limited main memory

K-Means Clustering for Large Data Sets

- Choose a random sample of data and perform k-means. Return the k-means and the quality of cluster measure
- Choose another random sample and repeat
- Compare the new result with the old and save the better of the two results
- Repeat above predetermined number of times

Indirect Clustering Through Iterative Optimization

- This approach works best when an initial clustering solution can be obtained through other means
- We use the Sum of Squared (SSE) criterion; other measures are possible with suitable changes in the algorithm.
- SSE Measure

$$J_e = \sum_{i=1}^{k} \sum_{\mathbf{x} \in C_i} \|\mathbf{x} - \mathbf{m}_i\|^2 = \sum_{i=1}^{k} J_i \,.$$

Similar to Ward's method mentioned earlier

Suppose we take an example $\hat{\mathbf{x}}$ from cluster i and contemplate on moving it cluster j. This move implies that means of clusters i and j would change. The new means would be given as

$$\mathbf{m}_i^{new} = \mathbf{m}_i^{old} - \frac{\hat{\mathbf{x}} - \mathbf{m}_i^{old}}{n_{iold} - 1}, \text{ and}$$

As a result of adding an extra example to cluster j, its error measure would increase. Similarly, the error measure for cluster i would decrease because it would now have one less example. It is not hard to show that the new and old values of error measures for the two clusters are related by the following equations:

$$J_{inew} = J_{iold} - \frac{n_{iold}}{n_{iold} - 1}\left\|\hat{\mathbf{x}} - \mathbf{m}_i\right\|^2$$

$$J_{jnew} = J_{jold} + \frac{n_{jold}}{n_{jold} + 1}\left\|\hat{\mathbf{x}} - \mathbf{m}_j\right\|^2.$$

We should move $\hat{\mathbf{x}}$ from cluster i to cluster j only when there is a decrease in the value of the SSE criterion function. Thus, we should move $\hat{\mathbf{x}}$ from cluster i to cluster j only when

$$\frac{n_{iold}}{n_{iold} - 1}\left\|\hat{\mathbf{x}} - \mathbf{m}_i\right\|^2 > \frac{n_{jold}}{n_{jold} + 1}\left\|\hat{\mathbf{x}} - \mathbf{m}_j\right\|^2.$$

Iterative Optimization Steps

Based on above, we can perform indirect clustering through the following steps:

1. Obtain an initial partition of the n examples into k clusters and compute each cluster mean.
2. Select a candidate example for move from one of the clusters and check if it is profitable to move it into another cluster. Move and update cluster means.
3. Repeat Step 2 with another example until no more moves are profitable.

Cluster Validity

- How do we validate our clustering result?
- One possible approach is to run the algorithm with different parameter settings and observe how the performance metric changes
- Another possibility is to see how compact the clusters are using some suitable metric.
- Often, the results are validated against a hypothesis based on the domain knowledge
- Use ground truth, if available, to validate clustering results

Silhouette Coefficient

- The Silhouette Coefficient is a measure of how compact a cluster is and how far is it from other clusters. We can use this measure to validate our clustering results when the ground truth is not known.
- The Silhouette Coefficient is defined for each sample and is composed of two scores:

  a: The mean distance between a sample and all other points in the same class.

  b: The mean distance between a sample and all other points in the *next nearest cluster*.
- The Silhouette Coefficient *s* for a single sample is then given as:

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

- We can sum up the coefficient values for every example in a cluster to get the cluster score for a given cluster

Rand Index

- Suppose you know the ground truth and want to compare your clustering result against the ground truth. In that case, Rand Index (RI) is one measure that you can use.
- In place of the ground truth, you can also use RI to compare your clustering results with those obtained by your friend to see how similar are the results.
- RI works by looking at all possible unordered pairs of examples. Suppose there are *N* examples for clustering. Then, we have *N*(N-1)/2* example pairs.
- Some of the pairs are together in the clustering result as well as in the ground truth. Some of the pairs are never together while the remaining pairs are sometimes together and sometimes not together. Based on this, the RI is defined as:
- RI = count of pairs always in agreement or disagreement/Total number of pairs
- We can notice from the formula that RI can never exceed 1 and its possible lowest value is 0.

RI Example

- Examples are A, B, C, D, and E
- 10 pairs are possible: These are: {*A*, *B*}, {*A*, *C*}, {*A*, *D*}, {*A*, *E*}, {*B*, *C*}, {*B*, *D*}, {*B*, *E*}, {*C*, *D*}, {*C*, *E*}, and {*D*, *E*}.
- Clustering method #1 groups A, B, and C in one cluster, and D and E into another
- Clustering method #2 groups A and B in one group and C, D, and E in another
- Looking at the clustering results, we see that the A and B are always grouped together. The same goes for D and E. We also note that A and D never occur together. A and D, A and E, B and D, and B and E are never grouped together.
- Thus, RI = 6/10 -> 0.6
- A drawback of RI is that even for a random grouping, it gives a high value for the index. This is observed more frequently when the number of clusters is large. Thus in practice, Adjusted Rand Index (ARI) is used as a metric. You can read about ARI at the following link:

https://iksinc.online/2019/05/06/how-similar-are-two-clustering-results/

Entropy Based Cluster Validity Measure

- *Entropy* is an information theoretic criterion that measures the homogeneity of the distribution of the clusters with respect to different classes.
- Given K as the number of clusters resulting from the clustering algorithm and C as the number of classes in the ground truth, let
  - h_{ck} denote the number of patterns assigned to cluster k with a ground truth class label c.
  - $h_{c.} = \sum_{k=1}^{K} h_{ck}$ denote the number of patterns with a ground truth class label c.
  - $h_{.k} = \sum_{c=1}^{C} h_{ck}$ denote the number of patterns assigned to cluster k.

Cluster Validity

- The quality of individual clusters is measured in terms of the homogeneity of the class labels within each cluster.
- For each cluster k, the cluster entropy E_k is given by

$$E_k = -\sum_{c=1}^{C} \frac{h_{ck}}{h_{.k}} \log \frac{h_{ck}}{h_{.k}}.$$

- Then, the overall cluster entropy E_{cluster} is given by a weighted sum of individual cluster entropies as

$$E_{\text{cluster}} = \frac{1}{\sum_{k=1}^{K} h_{.k}} \sum_{k=1}^{K} h_{.k} E_k.$$

Cluster Validity

- A smaller cluster entropy value indicates a higher homogeneity.
- However, the cluster entropy continues to decrease as the number of clusters increases.
- To overcome this problem, another entropy criterion that measures how patterns of the same class are distributed among the clusters can be defined.

Cluster Validity

- For each class c, the class entropy E_c is given by

$$E_c = -\sum_{k=1}^{K} \frac{h_{ck}}{h_{c.}} \log \frac{h_{ck}}{h_{c.}}.$$

- Then, the overall class entropy E_{class} is given by a weighted sum of individual class entropies as

$$E_{\text{class}} = \frac{1}{\sum_{c=1}^{C} h_{c.}} \sum_{c=1}^{C} h_{c.} E_c.$$

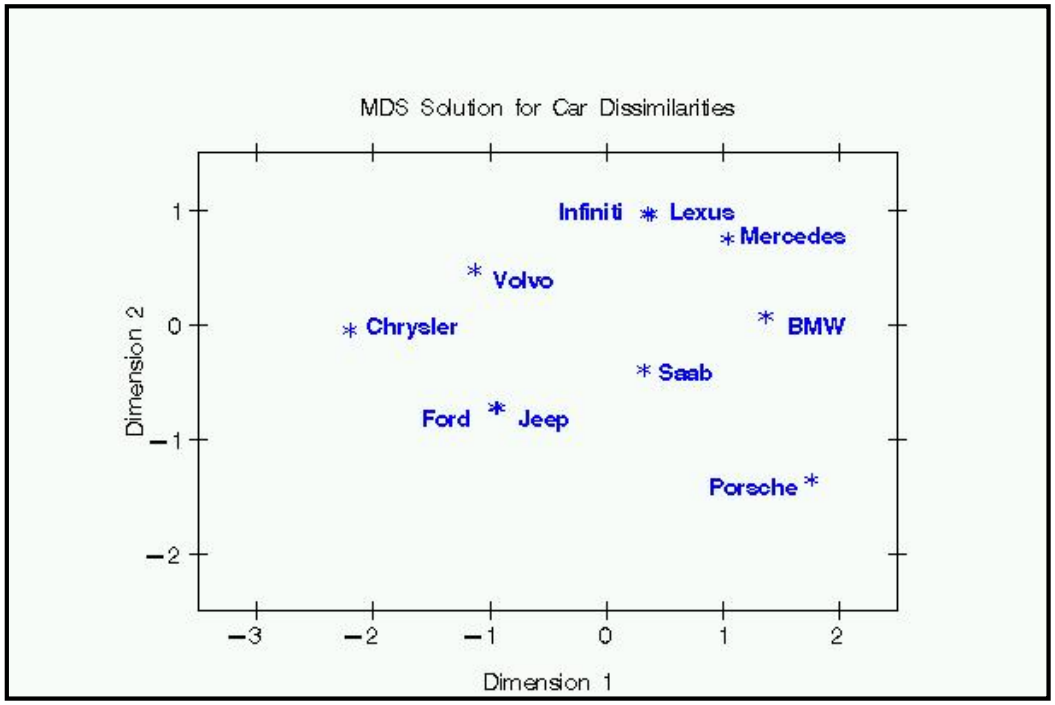
Cluster Validity

- Unlike the cluster entropy, the class entropy increases when the number of clusters increases.
- Therefore, the two measures can be combined for an overall entropy measure as

$$E = \beta E_{\text{cluster}} + (1 - \beta) E_{\text{class}}$$

where $\beta \in [0, 1]$ is a weight that balances the two measures.

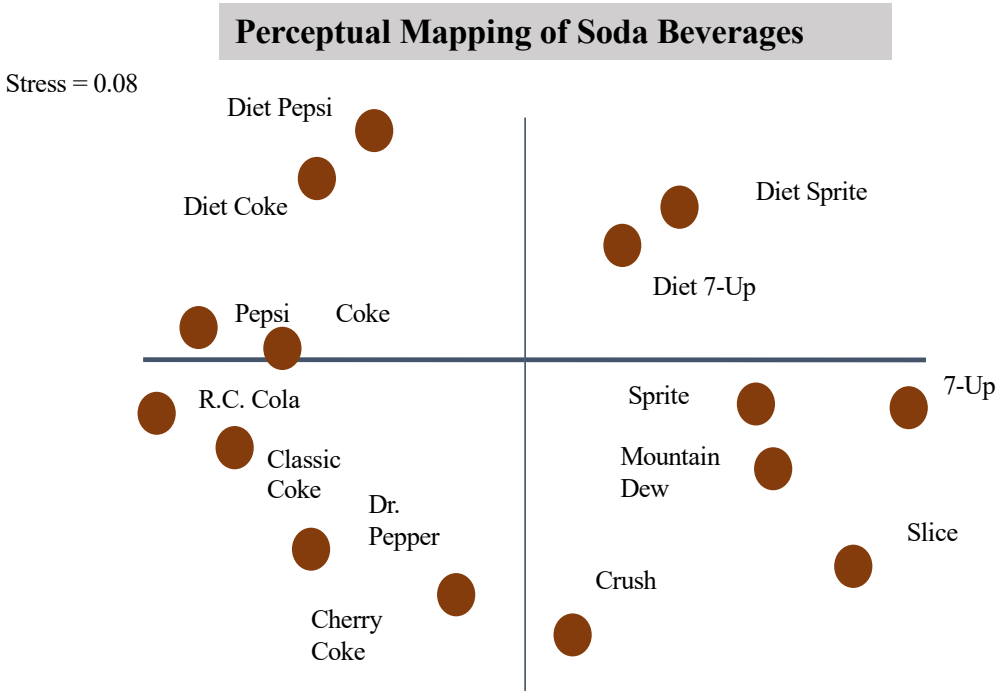
Multidimensional Scaling (MDS)



What is MDS?

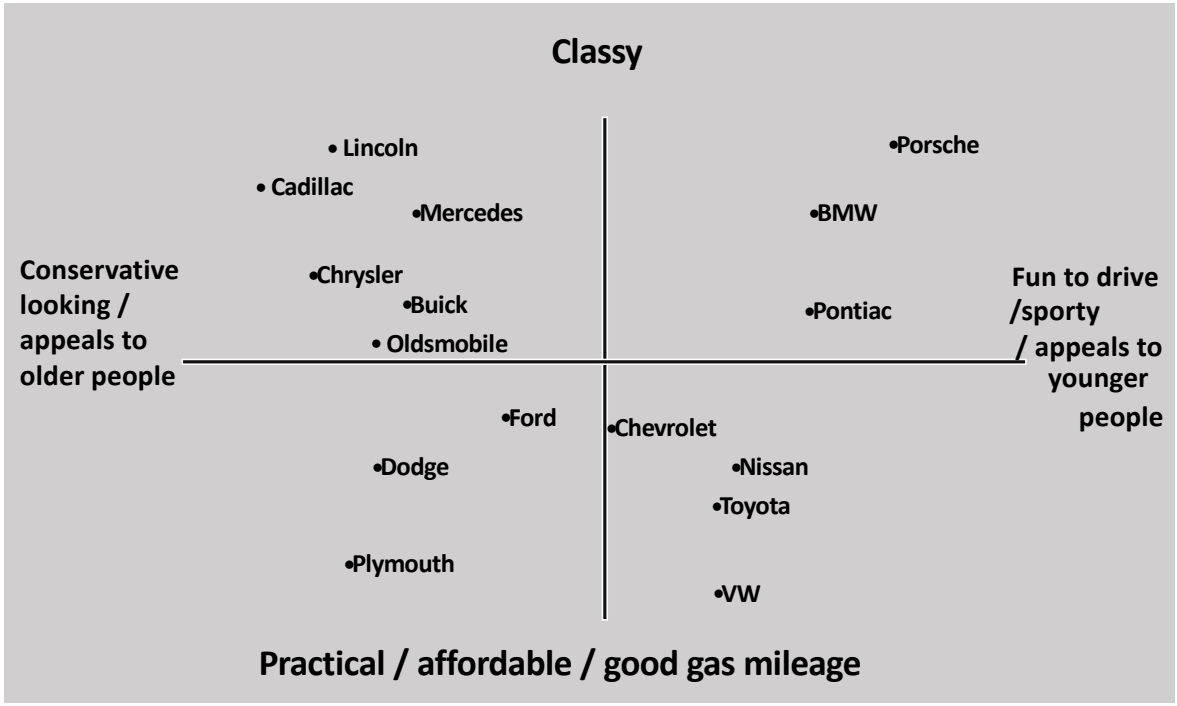
- Multidimensional Scaling (MDS) is a data analysis method for mapping a group of points in a low (2-3) dimensional space while preserving as well as possible inter-point distances between the points in the original high dimensional space.
- MDS can aid data understanding/clustering results via visualization
- Originally developed as a tool to uncover hidden structure to account for perceived similarities among a group of objects or items.
- MDS is also known as perceptual mapping
- Widely used in marketing and product placement (Analyzing surveys)

Perceptual Mapping of Soda Beverages



Another Perceptual Map Example

Axes labels are external and kind of provide an explanations for observed groupings



Mathematical Representation of MDS

- Let δ_{ij} = *Dist*($\boldsymbol{X}_i$, $\boldsymbol{X}_j$) in the original high-dimensional space for a pair of examples
- d_{ij} = *Dist*($\boldsymbol{Y}_i$, $\boldsymbol{Y}_j$) – distance in the mapped space
- A measure for mapping, called stress, is defined as

$$J_1 = \frac{1}{\sum_{i<j} \delta_{ij}^2} \sum_{i<j} (d_{ij} - \delta_{ij})^2$$

This measure emphasizes large errors. An optimization procedure like the gradient search is used to obtain the result

Mathematical Representation of MDS

- Another possible measure is the following. It emphasizes large fractional errors

$$J_2 = \sum_{i<j} \left(\frac{d_{ij} - \delta_{ij}}{\delta_{ij}}\right)^2$$

Mathematical Representation of MDS

- Yet another is as follows which is a compromise between the previous two measures

$$J_3 = \frac{1}{\sum_{i<j} \delta_{ij}} \sum_{i<j} \left(\frac{d_{ij} - \delta_{ij}}{\delta_{ij}}\right)^2$$

- The mapping performed by any of these three measures is known as metric MDS

Non-Metric MDS

- Non-metric MDS is more useful. It tries to preserve the order or rank of similarities. Let the ranked similarities be

$$\delta_{i_1 j_1} \le \Delta \le \delta_{i_m j_m}; m = n(n-1)/2$$

- Then the mapping is obtained by finding *m* numbers that satisfy the following constraint

$$d_{i_1 j_1} \le D \le d_{i_m j_m}$$

and minimize the function:

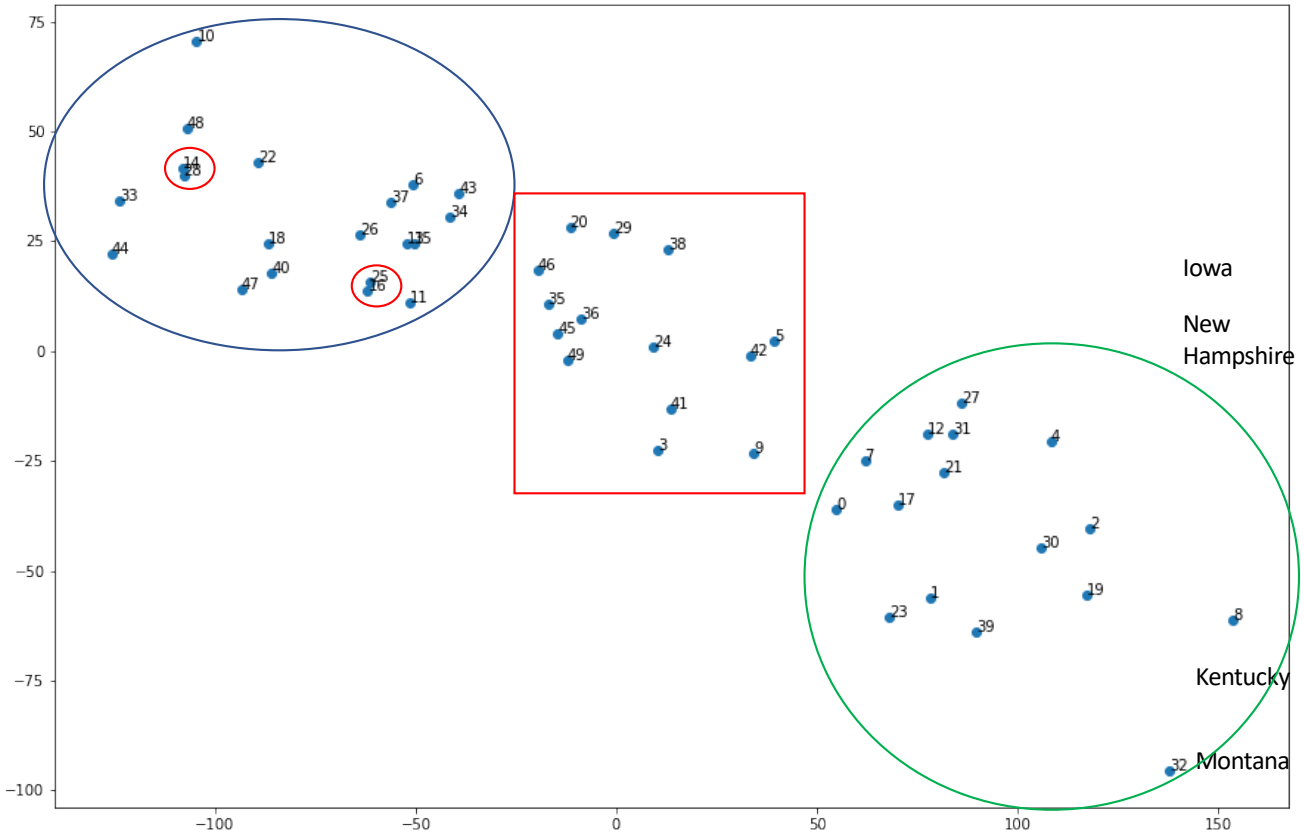
$$J = \min \frac{1}{\sum_{i<j} d_{ij}^2} \sum_{i<j} (d_{ij} - d_{ij})^2$$

The above measure is invariant to translation, rotation, and dilation of the point configuration in the original space.

MDS Example

- We will work with the USArrest data, used earlier for k-means.
- We will map the 50 states of the data using MDS
- The mapped cities in 2-d are shown in the next slide

```
from sklearn.manifold import MDS
mds = MDS(n_components=2)
proj = mds.fit_transform(X)
```



States # 14(Iowa) and 20(New Hampshire) are closest in the map. Their numbers are as follows:

Iowa	2.2	56	57	11.3
New Hampshire	2.1	57	56	9.5

States # 16(Kentucky) and 25(Montana) are also closest in the map. Their numbers are as follows:

Kentucky	9.7	109	52	16.3
Montana	6	109	53	16.4

Visual inspection suggests 3 clusters corresponding to the knee point at k=3 for the unnormalized data shown earlier.

Summary

- Clustering has numerous applications across many disciplines
- There are many clustering algorithms other than those discussed here
- Getting the right number of clusters and validating clusters in absence of any ground truth is hard without domain knowledge
- MDS is a good visualization tool that is used for explaining how similarities are perceived