Introduction to Clustering

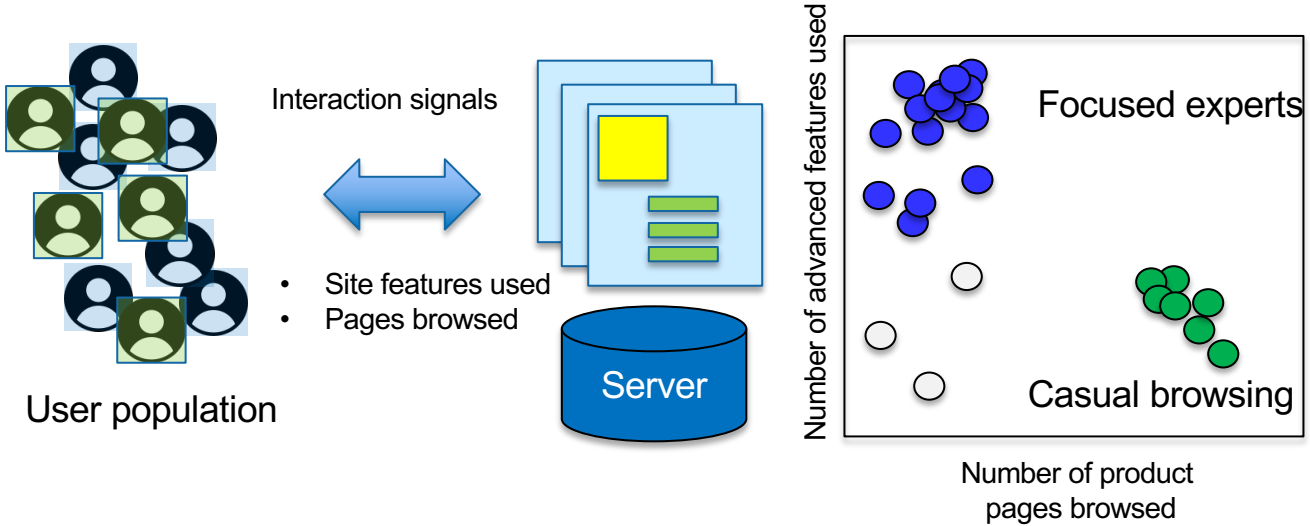
Kevyn Collins-Thompson

Associate Professor of Information and Computer Science
School of Information, University of Michigan

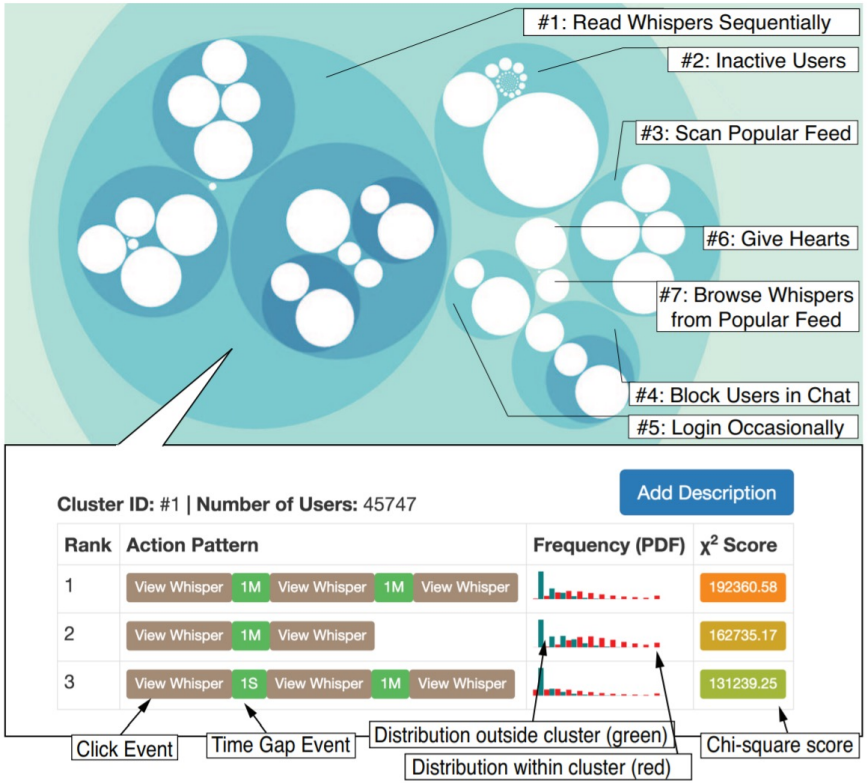
Unsupervised learning

- Transform or summarize data (e.g. dimensionality reduction)
- Find interesting clusters and other structure

Web clustering example



Behavioral clusters on Whisper
[Wang et al., CHI 2016]



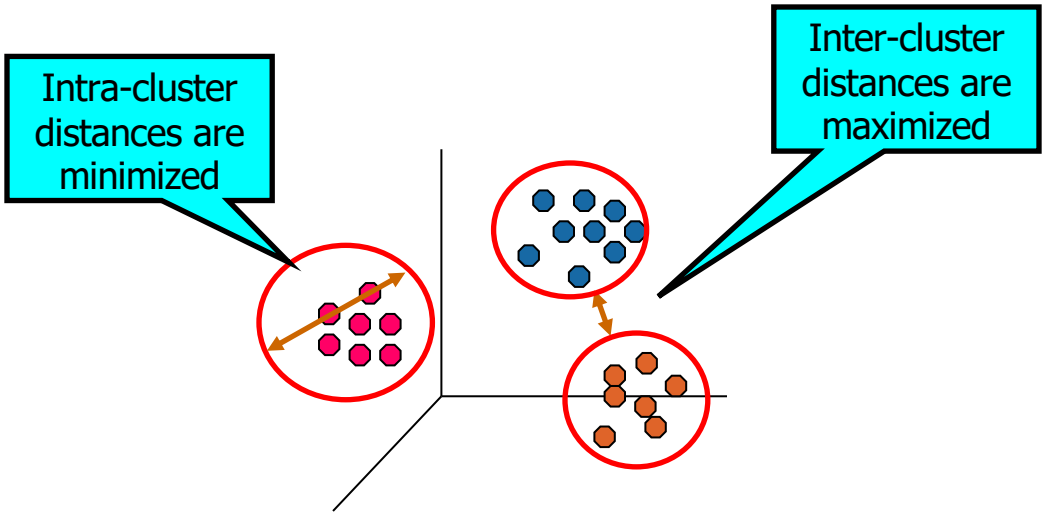
Source: https://sites.cs.ucsb.edu/~ravenben/publications/pdf/clickstream-chi16.pdf

Learning objectives

- Clustering methods in scikit-learn, including k-means, agglomerative clustering, hierarchical clustering, and DBSCAN.
- How to evaluate clustering results using a variety of metrics.
- Understand the tradeoffs and assumptions inherent in different clustering techniques.
- Be aware of advanced clustering methods like spectral clustering.

Cluster analysis finds 'interesting' groups of objects based on similarity

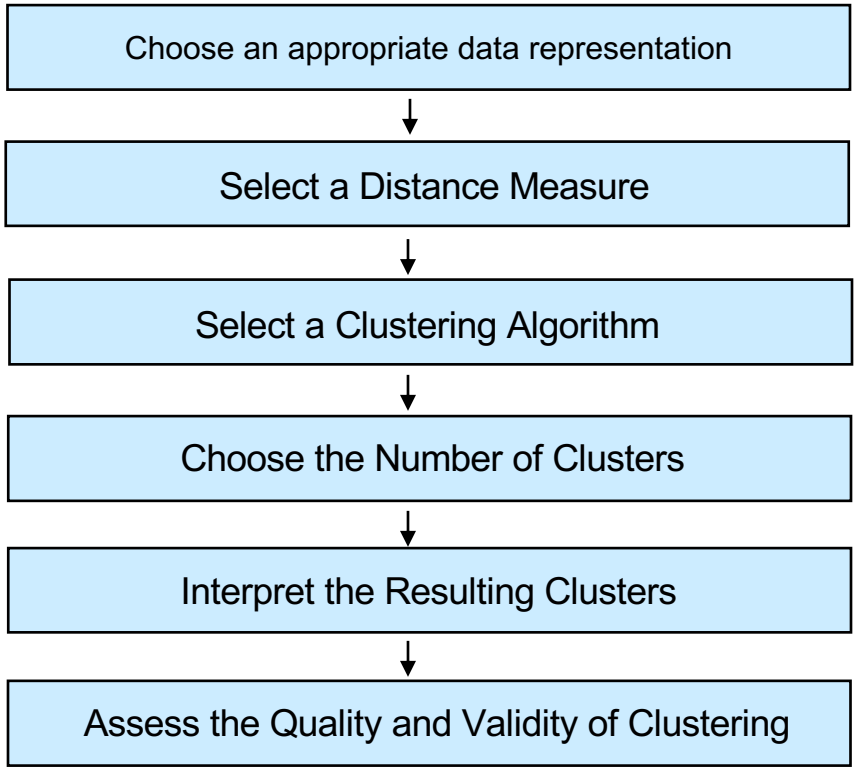
- What typically makes a 'good' clustering?
  - Members are highly similar to each other
    - Minimize within-cluster distances
  - Well-separated from other clusters
    - Maximize between-cluster distances



Clustering is often used as an exploratory data analysis tool

- Data reduction
  - Clustering creates a new nominal level variable that can be used in any further analysis.
  - In one-dimension, a good way to quantize real-valued variables into k non-uniform buckets
  - More generally, can be used to compress or summarize large datasets (e.g. by replacing data points with their cluster centroids)
- Data smoothing
  - Infer missing attributes from cluster neighbors
- Data understanding
  - Finding underlying factors, groups, structure
- Data navigation
  - Web search and browsing

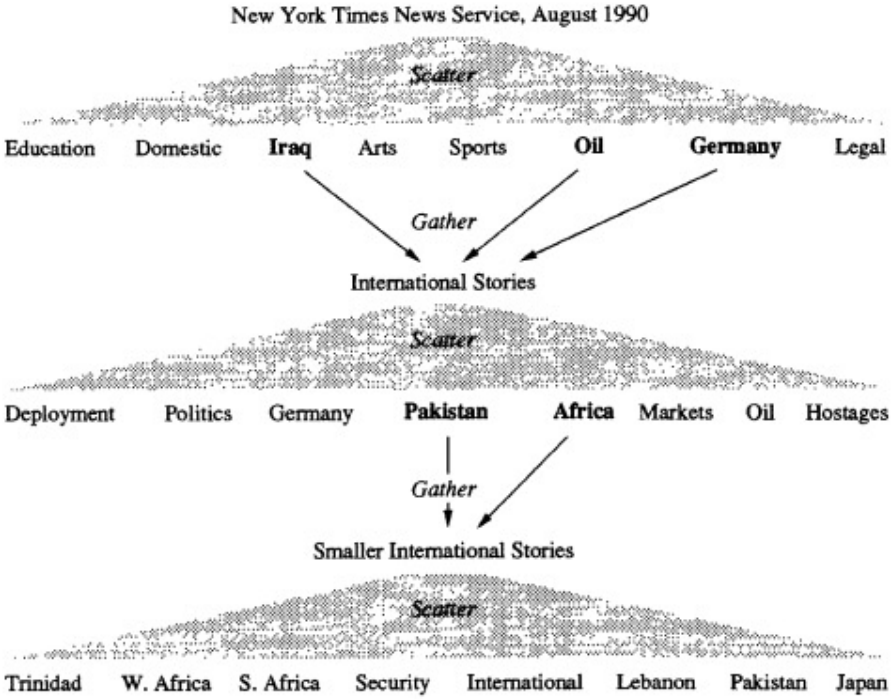
Typical Cluster Analysis Stages



Example: Scatter/Gather. A clustering-based approach to browsing large document collections

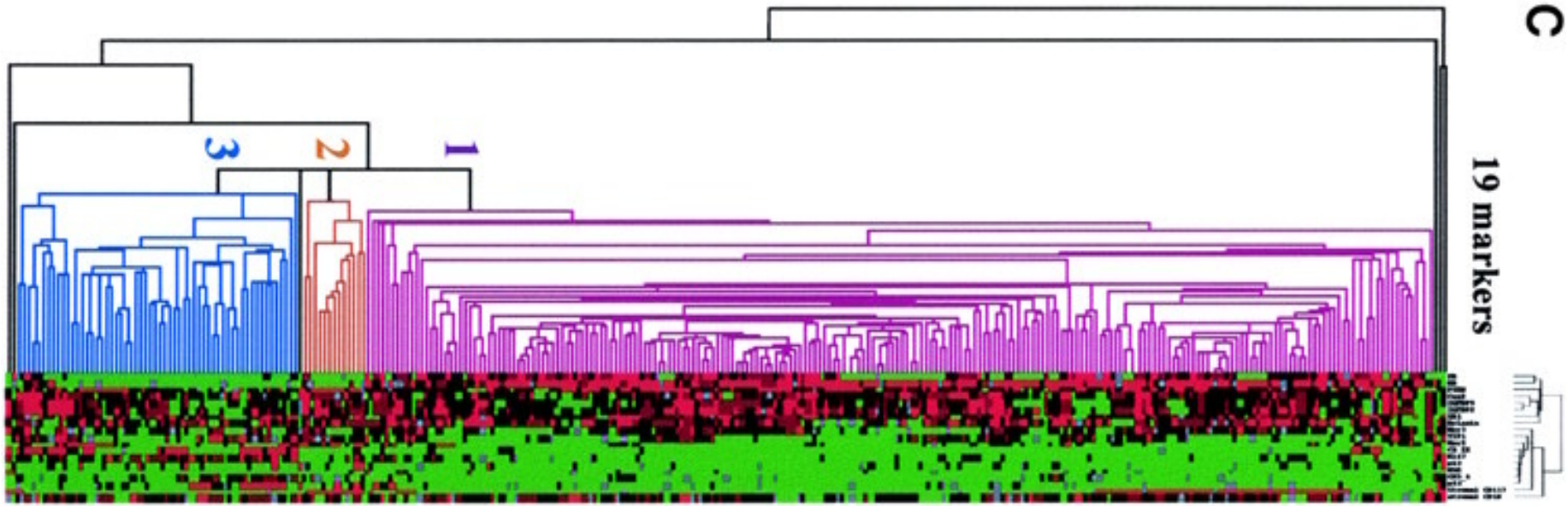
- What if you have a vague information need that spans topics
- And you're not sure which search terms to use?

- Scatter:
  - Present the user with a set of clusters
- Gather:
  - User selects subset of clusters that seem relevant
- Combine clusters:
  - Repeat from step 1 until done.



Source: http://courses.washington.edu/info320/au11/readings/Week4.Cutting.et.al.1992.Scatter-Gather.pdf

Hierarchical clustering analysis groups breast cancer cases (columns) into clinically relevant classes with similar immunomarkers (rows)



Makretsov N A et al. Clin Cancer Res 2004;10:6143-6151

Clinical Cancer Research

Clustering arises naturally in many fields

- Health
  - DNA gene expression
    - Cluster cancer variants into treatment groups, based on immunomarkers of cell samples
  - Medical imaging
    - Find likely tumors in images
- Business
  - Market segments
  - Web site visitors
- Social network analysis
  - Find communities, influencers
- Information Retrieval:
  - Search results clustered by similarity, event or topic
  - Personalization for groups of similar users
- Speech understanding
  - Convert waveforms into one of k categories (known as Vector Quantization)

Clustering can be ambiguous: What is the ‘best’ clustering here?

How many clusters?

Six Clusters

Two Clusters

Four Clusters

Any clustering algorithm should try to satisfy three desirable properties:

- Scale-invariance
  - Changing the units should not change the clustering results
- Richness
  - For any desired partition of the data points there should be a distance function *d* that can achieve that.
- Consistency
  - Suppose we use a distance function *d* to get a clustering. If we create a new distance function *d** whose effect is to shrink points closer inside clusters and enlarge distance between clusters, we should get the same clustering from *d**

Clustering impossibility theorem
[Kleinburg 2002]

- There is no clustering function that can satisfy Scale-invariance, Richness, and Consistency at the same time.
- There exist natural clustering functions satisfying 2 out of 3 of these properties.
- e.g. k-means: doesn’t satisfy Consistency property.

Source: https://www.cs.cornell.edu/home/kleinber/nips15.pdf

What algorithms are used to find clusters? Answer: huge range of approaches

- Assigning objects to clusters
  - ‘Hard’ (partitional) each object belongs to exactly 1 cluster
  - ‘Soft’ : each object can belong to multiple clusters
- Hierarchical vs non-hierarchical
  - A set of nested clusters organized as a tree
- By far most widely-used fall into two types:
  - **Hierarchical**: agglomerative, single-link, etc.
  - **Partitional**: k-means, k-median, etc.

Kevyn Collins-Thompson

kevynct@umich.edu

School of Information

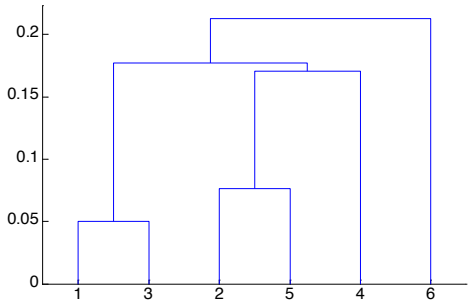
Hierarchical Clustering

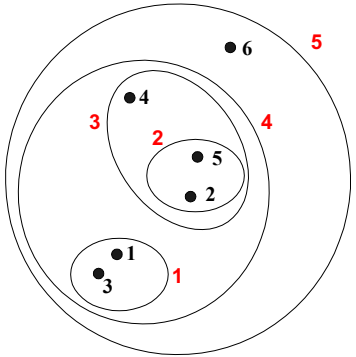
Kevyn Collins-Thompson

Associate Professor of Information and Computer Science
School of Information, University of Michigan

Hierarchical Clustering

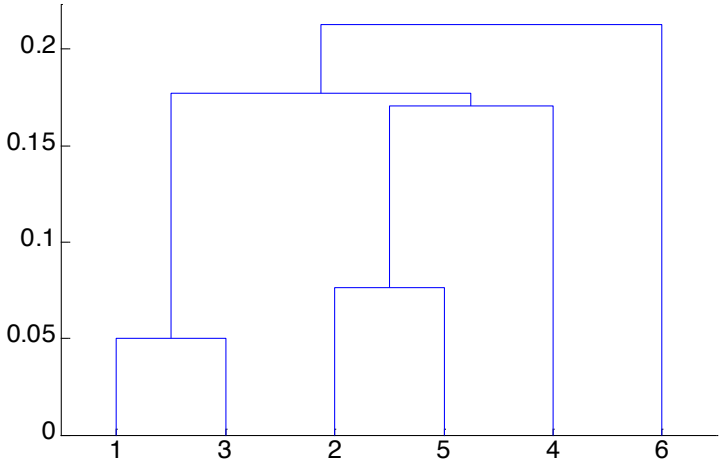
- Produces a set of nested clusters organized as a hierarchical tree
- Can be visualized as a dendrogram
  - A tree like diagram that records the sequences of merges or splits





Strengths of Hierarchical Clustering

- Do not have to assume any particular number of clusters
  - Any desired number of clusters can be obtained by ‘cutting’ the dendrogram at the proper level
- They may correspond to meaningful taxonomies
  - Example in biological sciences (e.g., animal kingdom, phylogeny reconstruction, …)

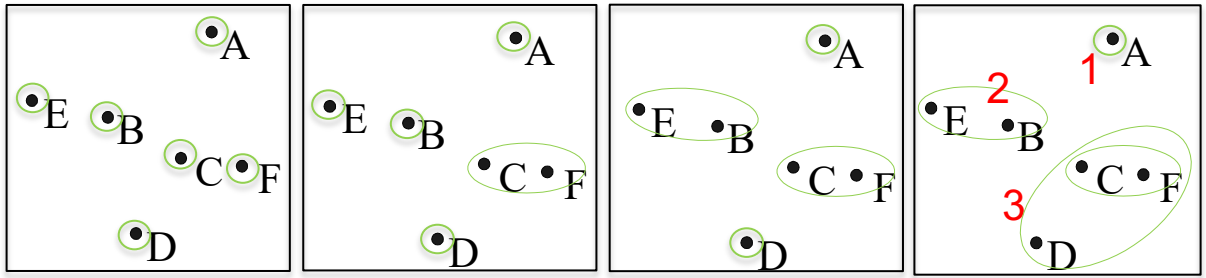


Hierarchical clustering algorithms

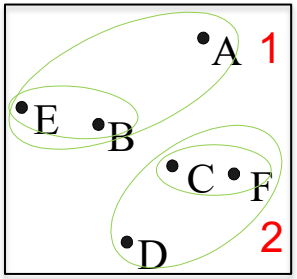
- Bottom-up ('Agglomerative')
  - Start with each point being in its own cluster
  - At each step
    - Merge the most similar pair of clusters based on a cost function
    - Continue until you have k clusters, or everything is in one big cluster
- Top-down ('Divisive')
  - Start with all points in a single big cluster
  - At each step:
    - Split the cluster into two smaller clusters based on a cost function
    - Continue until you have k clusters, or each point is in its own cluster

http://wiki.stat.ucla.edu/socr/index.php/SOCR_EduMaterials_AnalysisActivities_HierarchicalClustering

Agglomerative Clustering Example



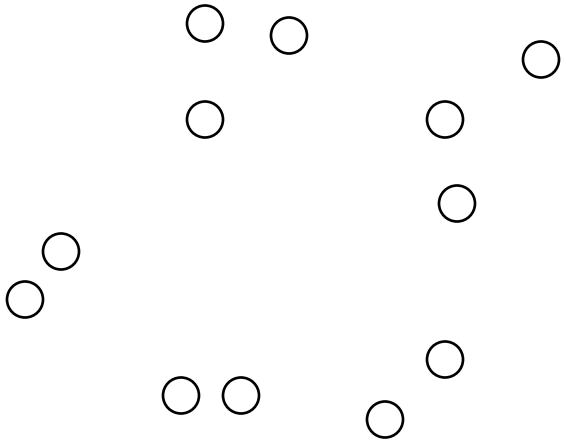
Stage 1 | Stage 2 | Stage 3 | **Stage 4 (three top-level clusters)**



Stage 5 (two top-level clusters)

Agglomerative (bottom-up) Clustering: Starting Phase

- Start with clusters of individual points and a proximity matrix of object-to-object distances



	p1	p2	p3	p4	p5	.
p1						.
p2						
p3						
p4						
p5						
.						
.						
.						

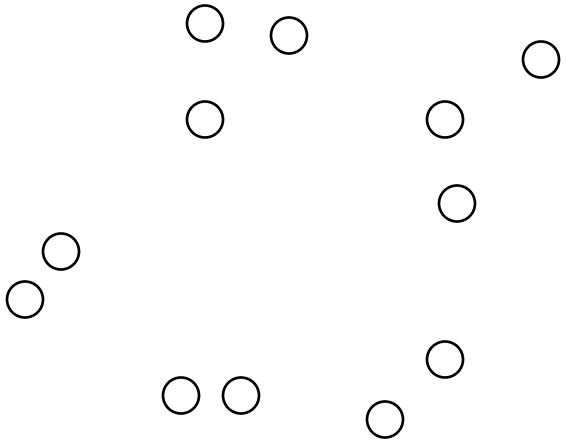
Proximity Matrix

Agglomerative Clustering Algorithm

- More popular hierarchical clustering technique
- Basic algorithm is straightforward:
  1. Compute the proximity matrix
  2. Let each data point be a single-point cluster
  3. Repeat:
     - Merge the two closest clusters
     - Update the proximity matrix
  4. Until only a single cluster remains
- Key operation: computation of the proximity of two clusters. The cost function.
  - Different approaches to defining the distance between clusters distinguish the different algorithms

Agglomerative (bottom-up) Clustering: Starting Phase

- Start with clusters of individual points and a proximity matrix of object-to-object distances

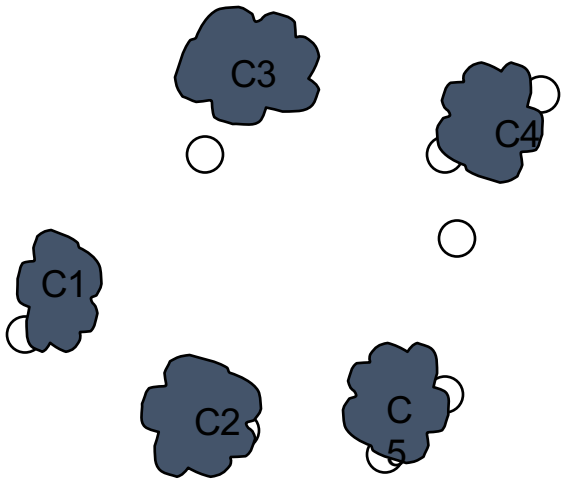


	p1	p2	p3	p4	p5	. .
p1						
p2						
p3						
p4						
p5						
.						
.						
.						

Proximity Matrix

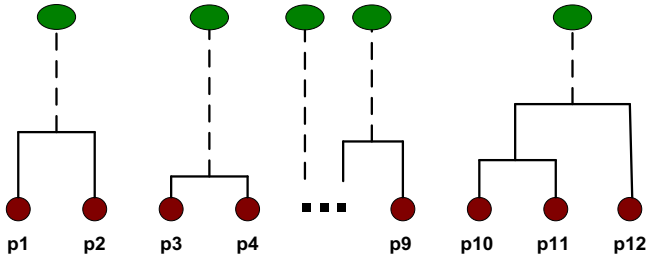
Intermediate Phase

- After some merging steps, we have some clusters



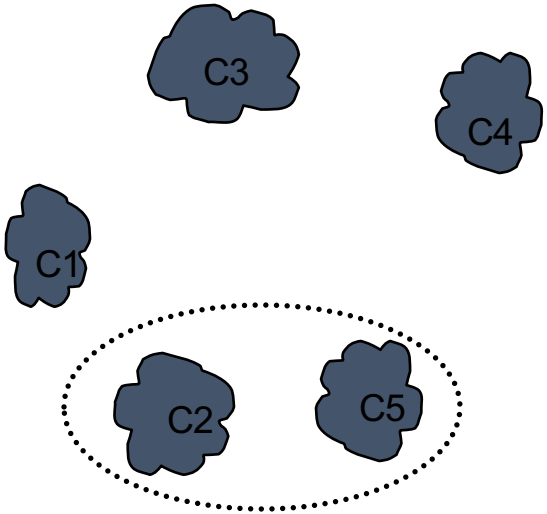
	C1	C2	C3	C4	C5
C1					
C2					
C3					
C4					
C5					

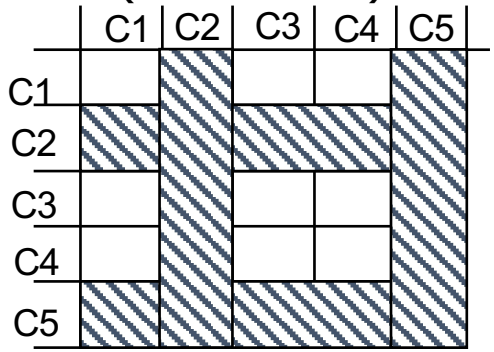
Proximity Matrix



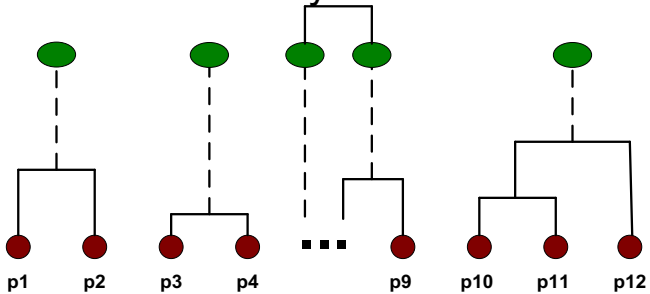
Intermediate Phase

- We want to merge the two closest clusters (C2 and C5) and update the proximity matrix.



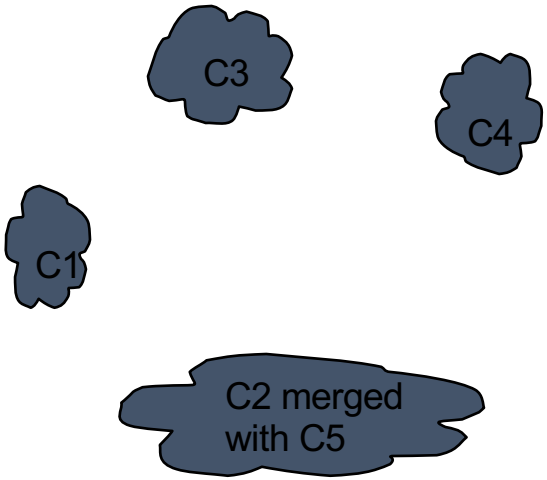


Proximity Matrix



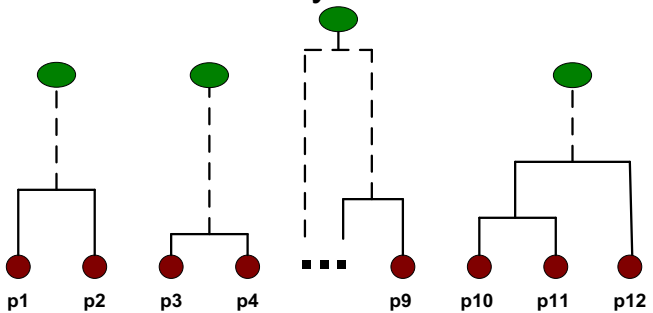
After Merging Clusters

- The question is "How do we update the proximity matrix?"

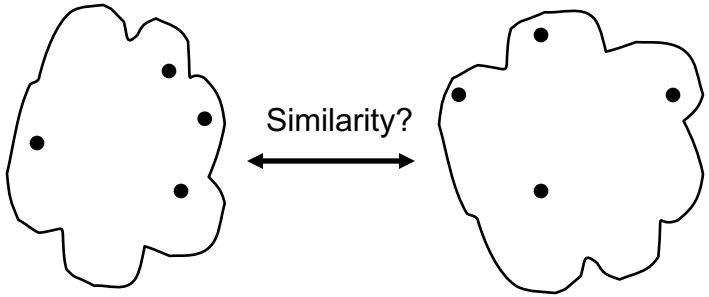


	C1	C2 + C5	C3	C4
C1		?		
C2 + C5	?	?	?	?
C3		?		
C4		?		

Proximity Matrix



How to Define Inter-Cluster Similarity

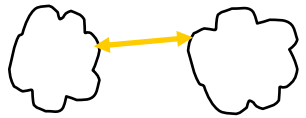


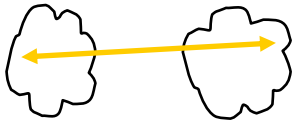
	p1	p2	p3	p4	p5	. . .
p1						
p2						
p3						
p4						
p5						
.						
.						
.						

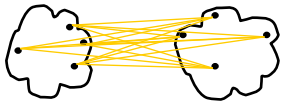
Proximity Matrix

Cost functions for bottom-up (agglomerative) clustering

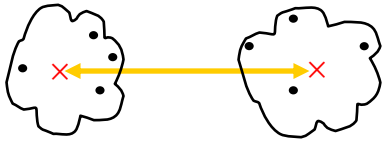
- Single linkage
  - Minimum distance between clusters
- Complete linkage
  - Max distance between clusters
- Average linkage
  - Average distance between clusters



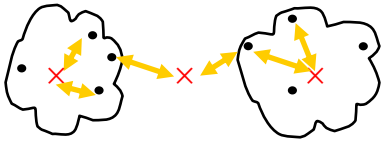




- Distance between centroids



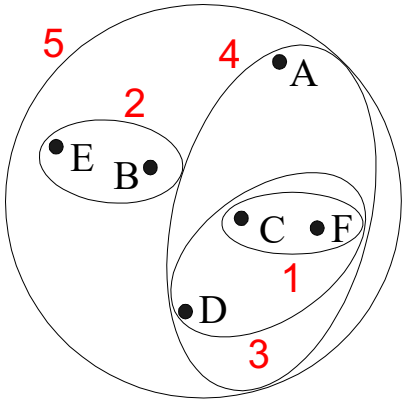
- Ward's method: squared error

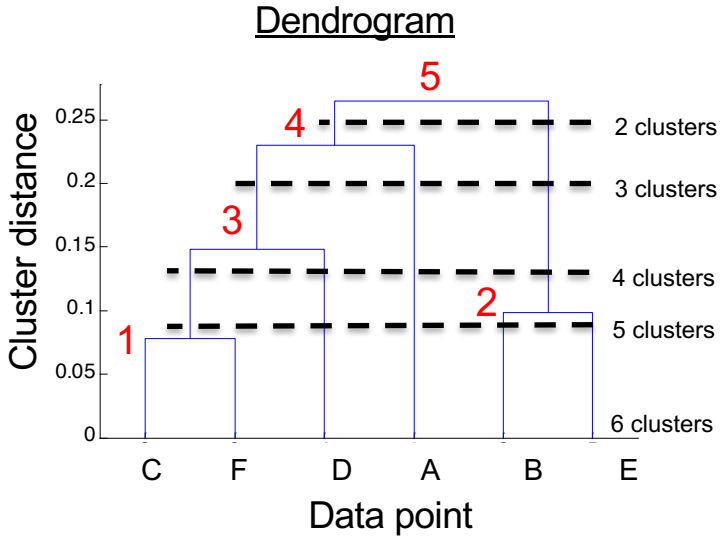


Dendrograms

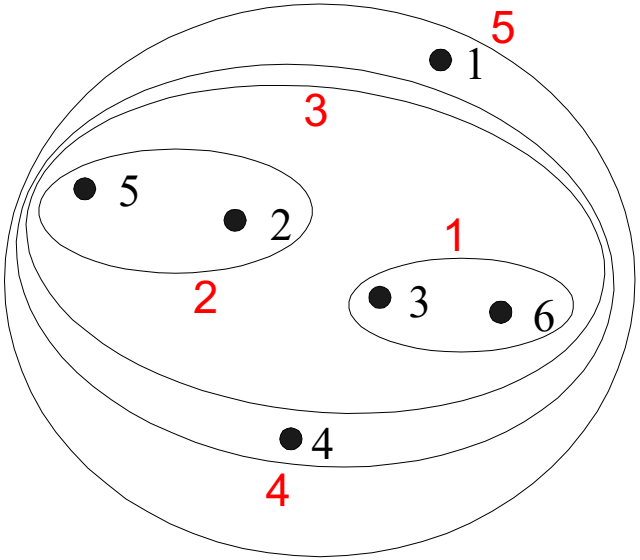
- Agglomerative clustering automatically arranges the data into a hierarchy as an effect of the algorithm
- The hierarchy reflects the order and cluster distance at which each data point is assigned to successive clusters.
- This hierarchy can be useful to visualize, using what's called a ***dendrogram***, which can be used even with high-dimensional data.

Hierarchical Clustering

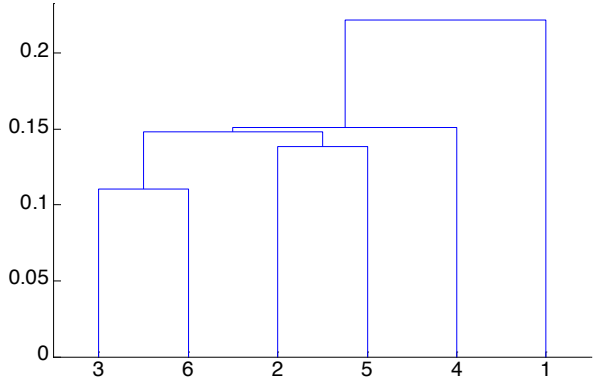




Hierarchical Clustering: MIN (single linkage)

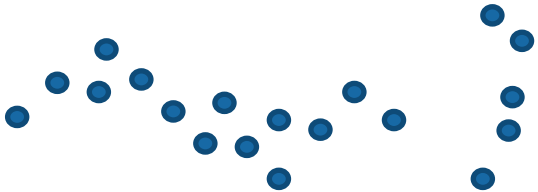


Nested Clusters

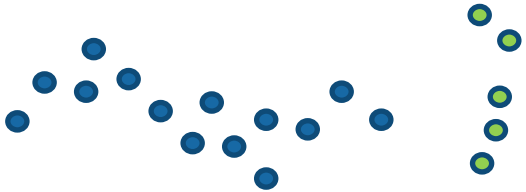


Dendrogram

Strength of MIN similarity



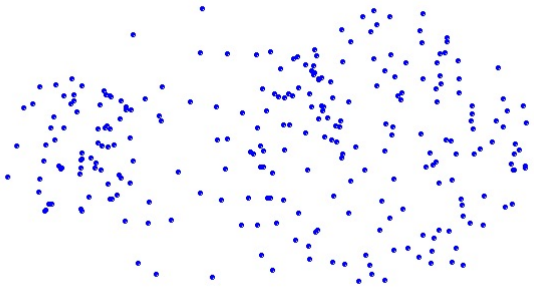
Original Points



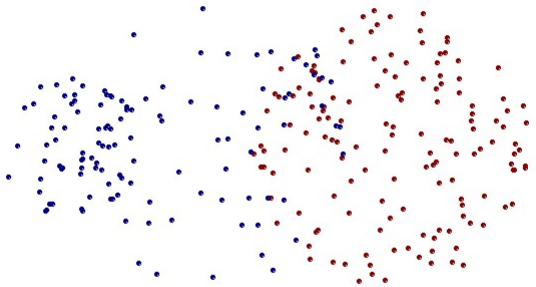
Two Clusters

- Can handle non-elliptical shapes

Limitations of MIN similarity



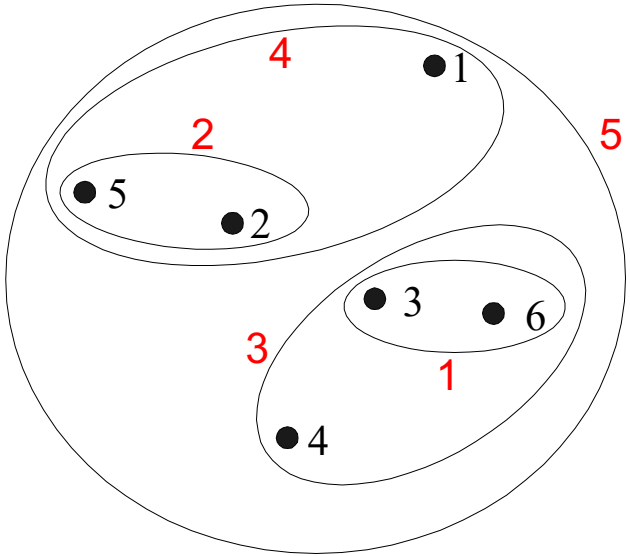
Original Points



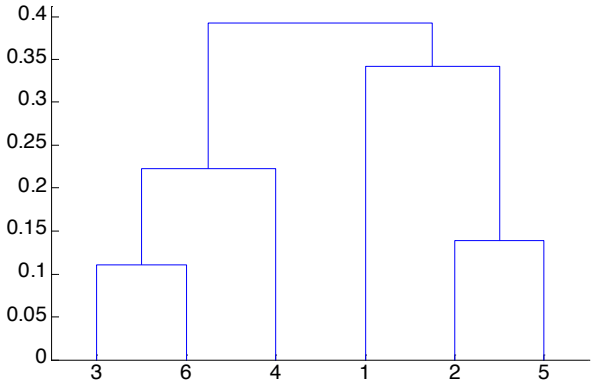
Two Clusters

- Single-link's local merge criterion may cause chaining and create elongated shapes that don't represent ideal clusters.

Hierarchical Clustering: MAX (complete linkage)

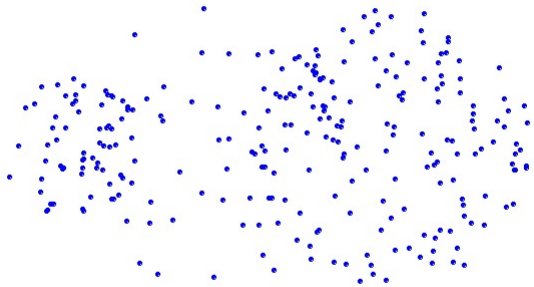


Nested Clusters

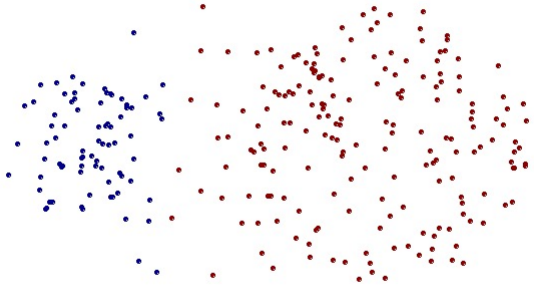


Dendrogram

Strength of MAX similarity



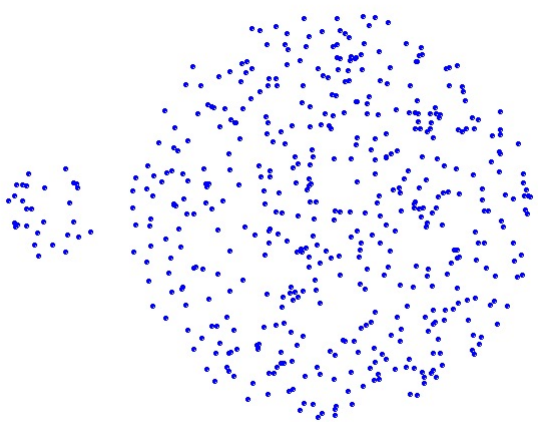
Original Points



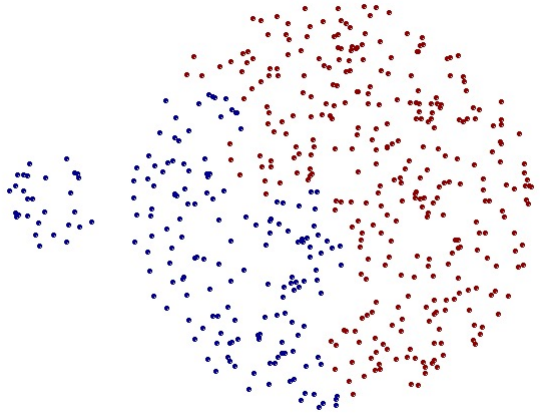
Two Clusters

- Good at finding compact, globular clusters.

Limitations of MAX similarity



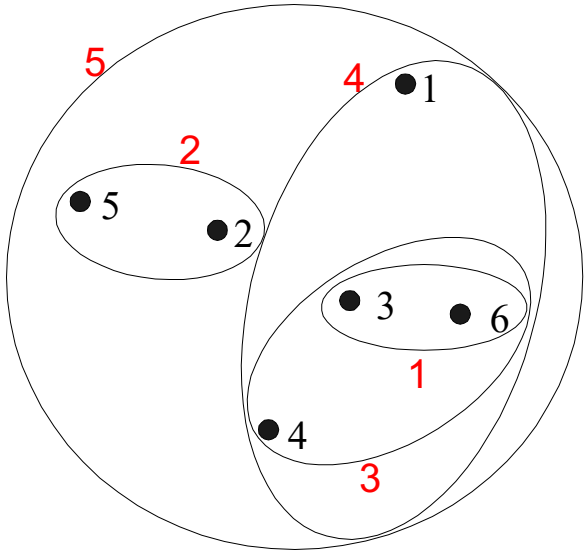
Original Points



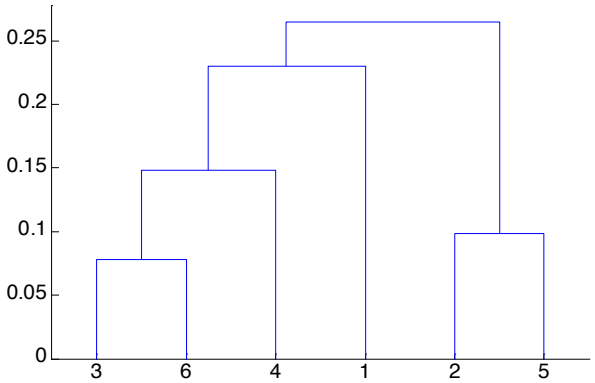
Two Clusters

- May break up large clusters
- Sensitive to outliers

Hierarchical Clustering: Group Average



Nested Clusters



Dendrogram

Hierarchical Clustering: Group Average strengths and limitations

- Compromise between Single and Complete Link
- Strengths
  - Less susceptible to noise and outliers
- Limitations
  - Biased towards globular clusters

Ward's method (1963)

- **Ward's distance** between clusters C_i and C_j is
  - The ***difference*** between:
    - The ***total within-cluster sum of squares for the two clusters separately***
    - The ***within-cluster sum of squares resulting from merging the two clusters*** into cluster C_{ij}

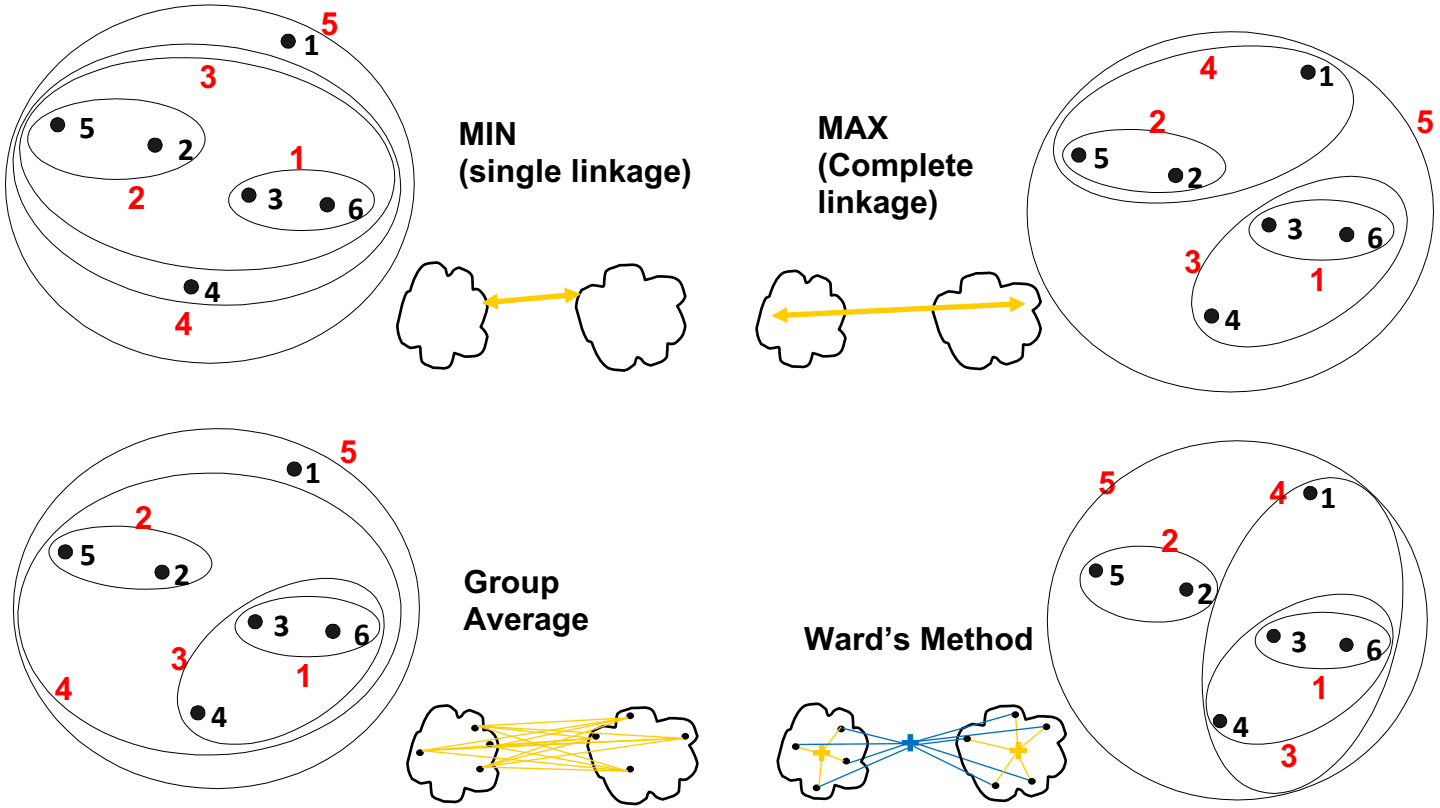
$$D_w(C_i, C_j) = \sum_{x \in C_i} (x - r_i)^2 + \sum_{x \in C_j} (x - r_j)^2 - \sum_{x \in C_{ij}} (x - r_{ij})^2$$

- r_i: centroid of C_i
- r_j: centroid of C_j
- r_{ij}: centroid of C_{ij}

Ward’s distance for clusters

- Similar to group average and centroid distance
- Less susceptible to noise and outliers
- Biased towards globular clusters
- Hierarchical analogue of k-means
  - Can be used to initialize k-means

Hierarchical Clustering: Comparison



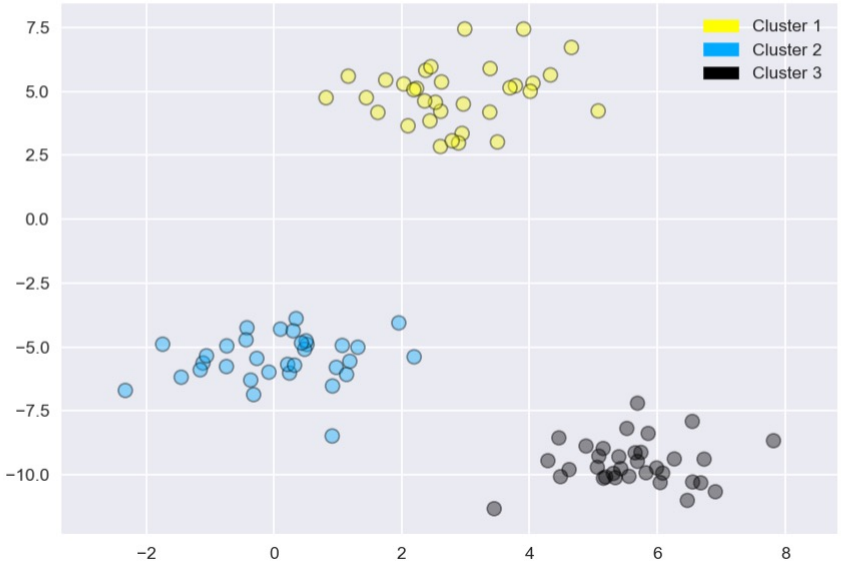
Agglomerative Clustering in Scikit-Learn

```
from sklearn.datasets import make_blobs
from sklearn.cluster import AgglomerativeClustering
from adspy_shared_utilities import plot_labelled_scatter

X, y = make_blobs(random_state = 10)

cls = AgglomerativeClustering(n_clusters = 3)
cls_assignment = cls.fit_predict(X)

X, y = make_blobs(random_state = 10)
plot_labelled_scatter(X, cls_assignment,
        ['Cluster 1', 'Cluster 2', 'Cluster 3'])
```



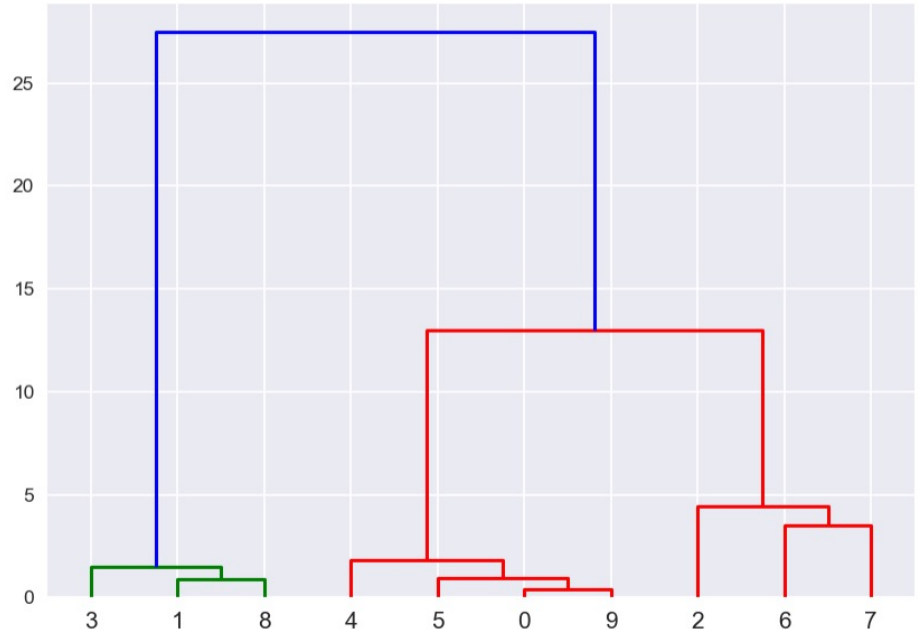
Default method: Ward's with Euclidean distance

Dendrogram Example

```
from scipy.cluster.hierarchy import ward, dendrogram
from sklearn.datasets import make_blobs
from sklearn.cluster import AgglomerativeClustering

X, y = make_blobs(random_state = 10, n_samples = 10)

plt.figure()
dendrogram(ward(X))
plt.show()
```



Interpreting the output from scipy.cluster.hierarchy

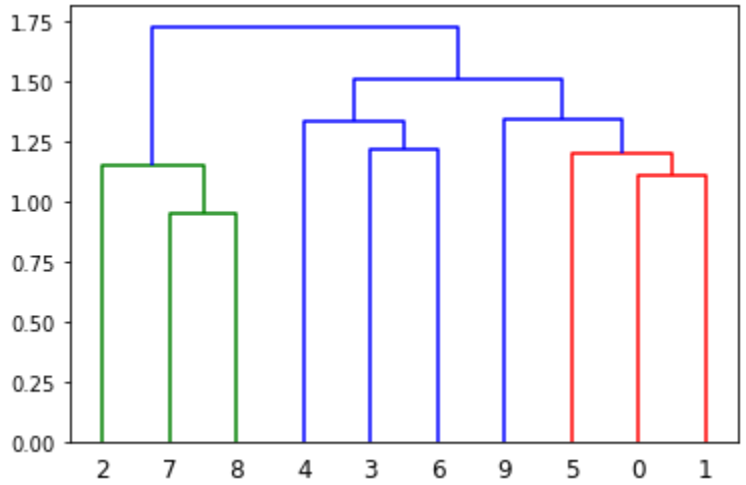
```
cls = ward(X)
```

Child nodes

Distance

leaf nodes merged

```
[[ 7.     8.     0.947  2.   ]
 [ 0.     1.     1.111  2.   ]
 [ 2.    10.     1.15   3.   ]
 [ 5.    11.     1.204  3.   ]
 [ 3.     6.     1.219  2.   ]
 [ 4.    14.     1.33   3.   ]
 [ 9.    13.     1.34   4.   ]
 [15.    16.     1.509  7.   ]
 [12.    17.     1.728 10.   ]]
```



Nodes 0 – 9 are leaf nodes.
Nodes 10 – 18 are internal nodes.

Which type of hierarchical clustering to use?

- Ward's method tends to give equal sized clusters.
- Ward's method is the most effective method for noisy data.

- Single linkage (nearest neighbor) tends to make long strings into a cluster.
- Single linkage is fast and does well on non-globular data but may not be robust in the presence of noise.

- Average and complete linkage perform well on cleanly separated globular clusters, but have mixed results otherwise.
- Top-down is sensitive to early errors: a bad first split choice can wreck the entire process.
- Bottom-up can't see the whole dataset and so may miss information that would lead to better clustering decisions overall.

Where are we so far?

- Two major clustering algorithms
  - Heirarchical
  - K-means
- General clustering questions:
  - How many clusters is best?
  - How can we assess and visualize cluster quality?
  - How can we visualize clusters?

Kevyn Collins-Thompson

kevynct@umich.edu

School of Information

K-Means Clustering

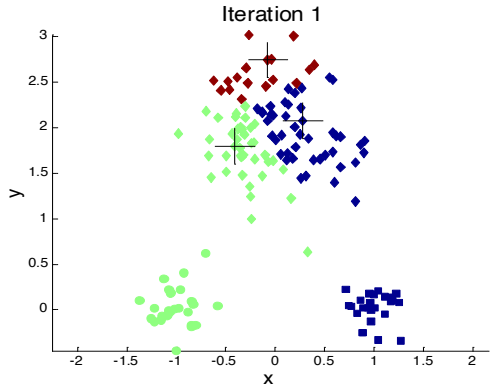
Kevyn Collins-Thompson

Associate Professor of Information and Computer Science
School of Information, University of Michigan

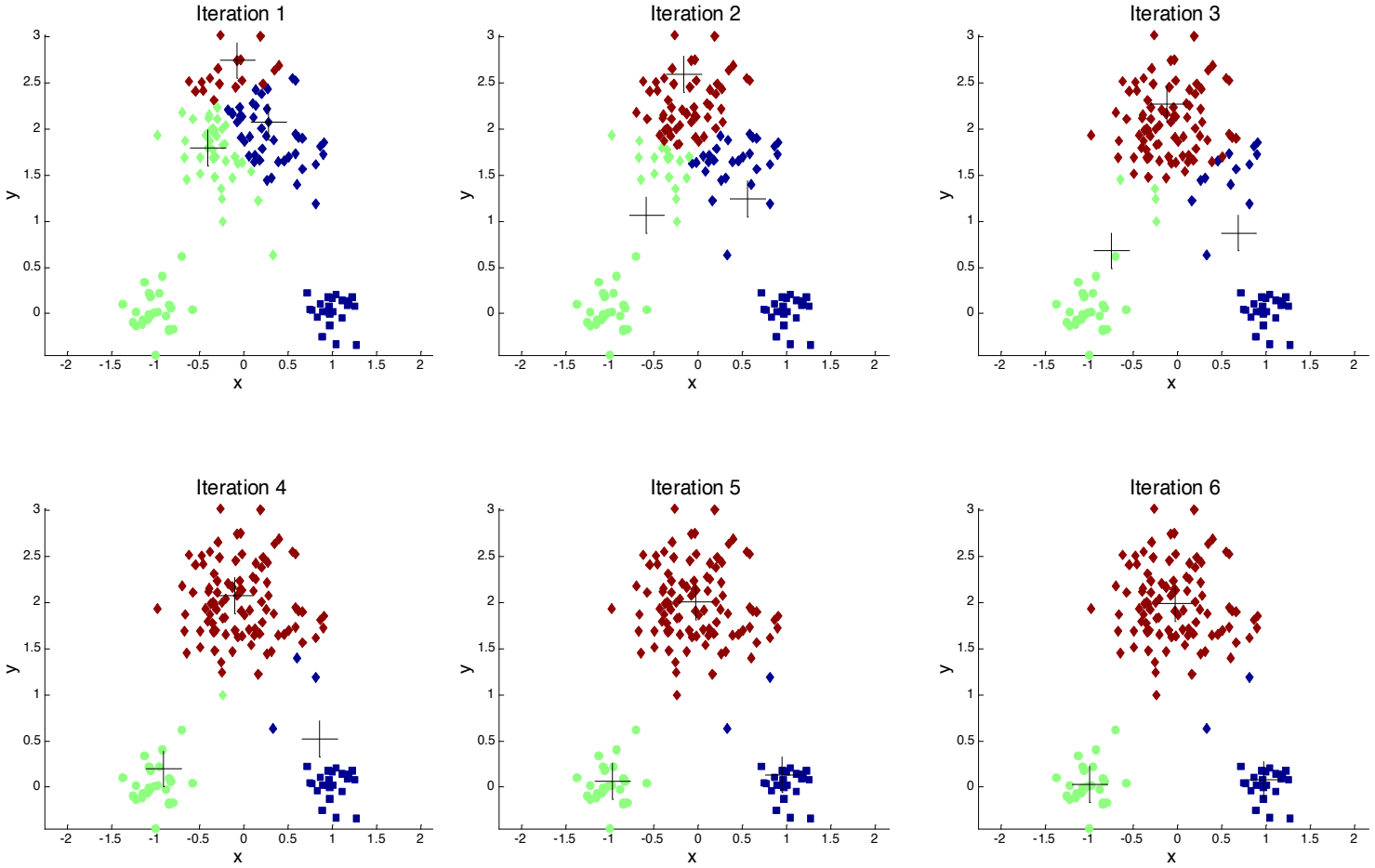
K-means: a popular clustering method.. and very different in nature from hierarchical clustering

- Partitional clustering approach
- Each cluster associated with a centroid (centerpoint)
- Each point is assigned to the cluster with the closest centroid
- Number of clusters, K, must be specified in advance
- The 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



The k-means algorithm (k = 3)



K-means Clustering

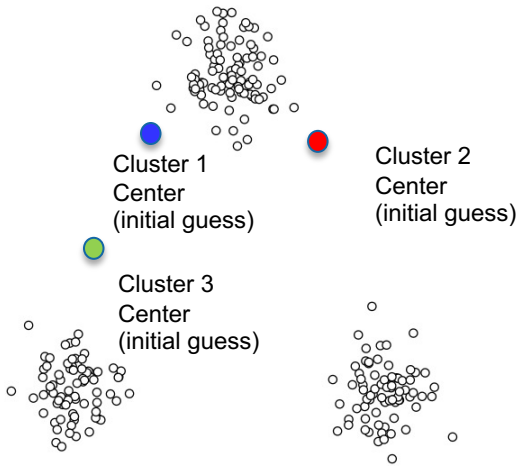
The k-means algorithm

Initialization Pick the number of clusters *k* you want to find. Then pick *k random* points to serve as an initial guess for the cluster centers.

Step A Assign each data point to the nearest cluster center.

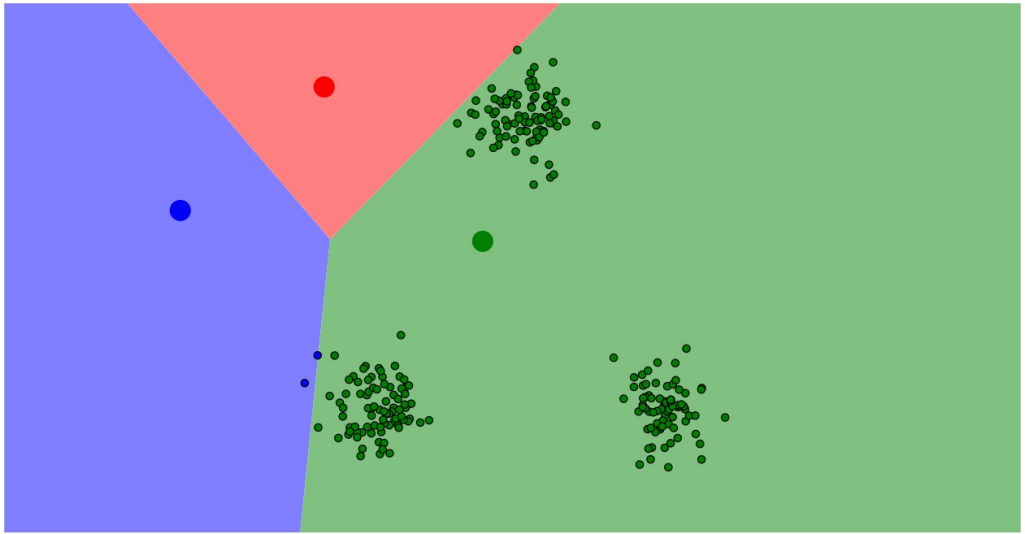
Step B Update each cluster center by replacing it with the mean of all points assigned to that cluster (in step A).

Repeat steps A and B until the centers converge to a stable solution.



Demo: https://www.naftaliharris.com/blog/visualizing-k-means-clustering/

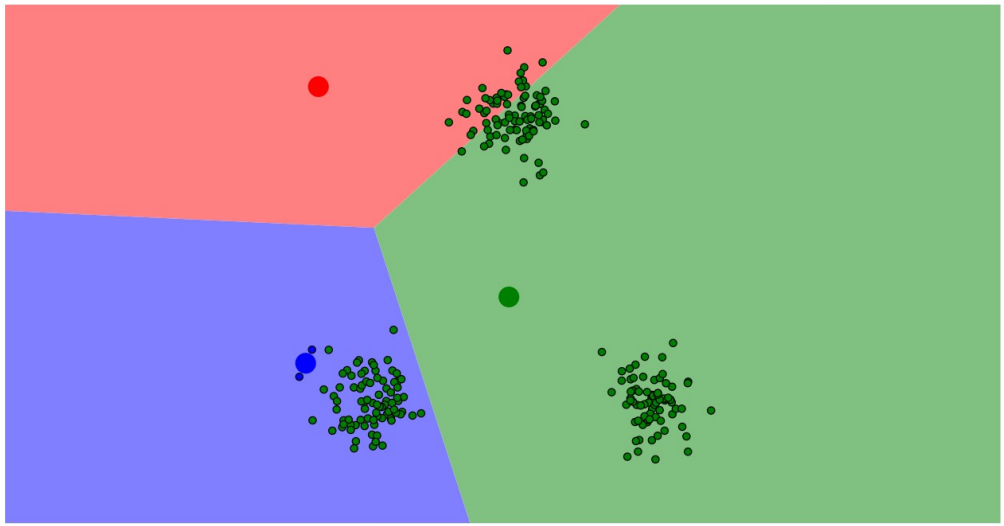
K-means Example: Step 1A



We want three clusters, so three centers are chosen randomly.

Data points are colored according to the closest center.

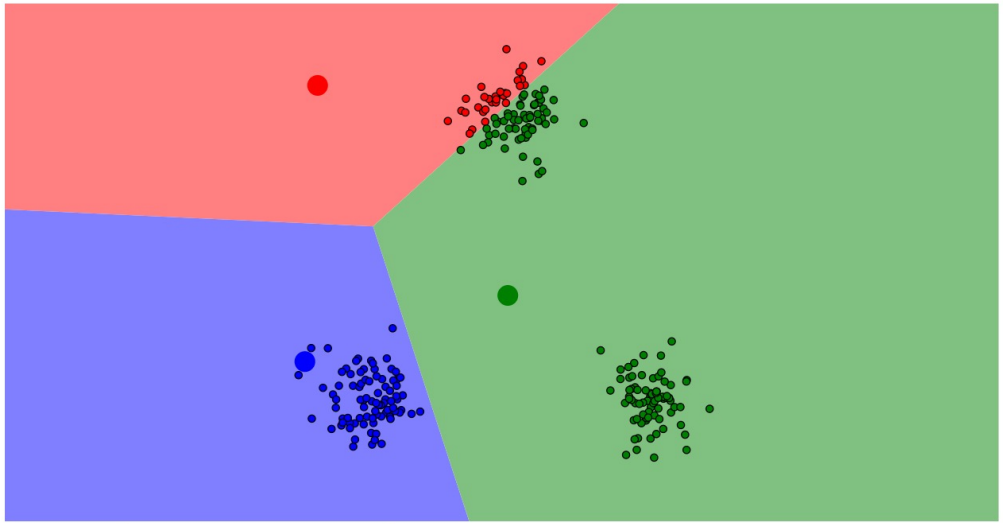
K-means Example: Step 1B



Each center is then updated…

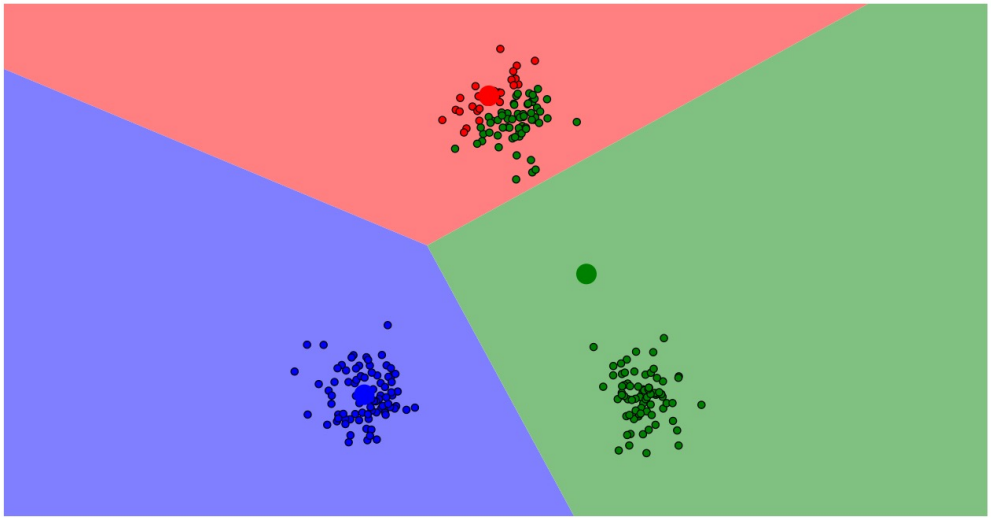
… using the mean of all points assigned to that cluster.

K-means Example: Step 2A



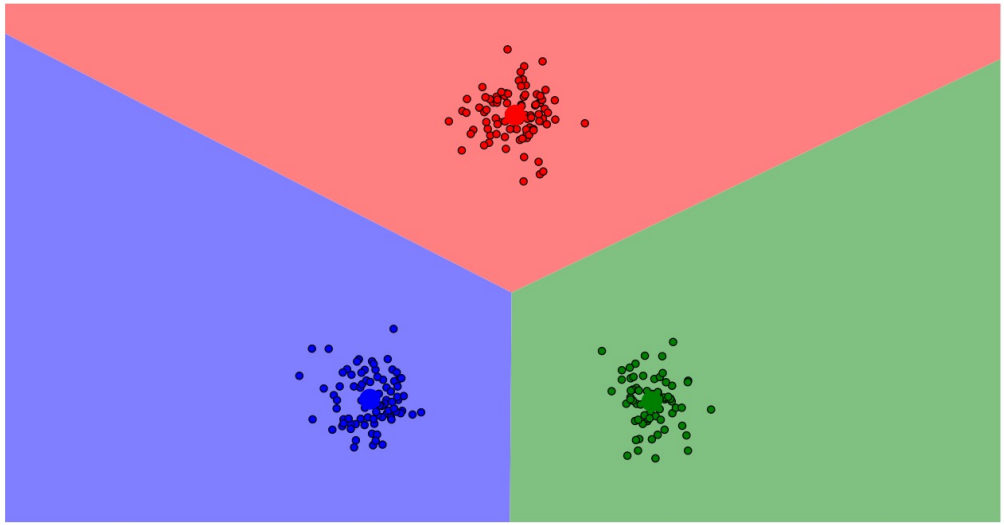
Data points are colored (again) according to the closest center.

K-means Example: Step 2B



Re-calculate all cluster centers.

K-means Example: Converged



After repeating these steps for several more iterations…

The centers converge to a stable solution!

These centers define the final clusters.

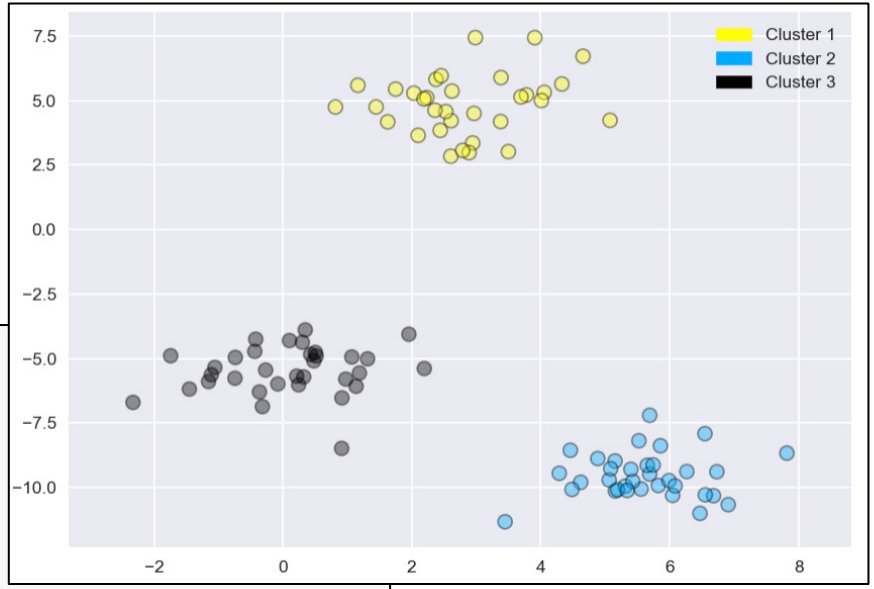
k-means Example in Scikit-Learn

```
from sklearn.datasets import make_blobs
from sklearn.cluster import KMeans
from adspy_shared_utilities import plot_labelled_scatter

X, y = make_blobs(random_state = 10)

kmeans = KMeans(n_clusters = 3)
kmeans.fit(X)

plot_labelled_scatter(X, kmeans.labels_, ['Cluster 1', 'Cluster 2', 'Cluster 3'])
```



k-means output on the Fruits Dataset

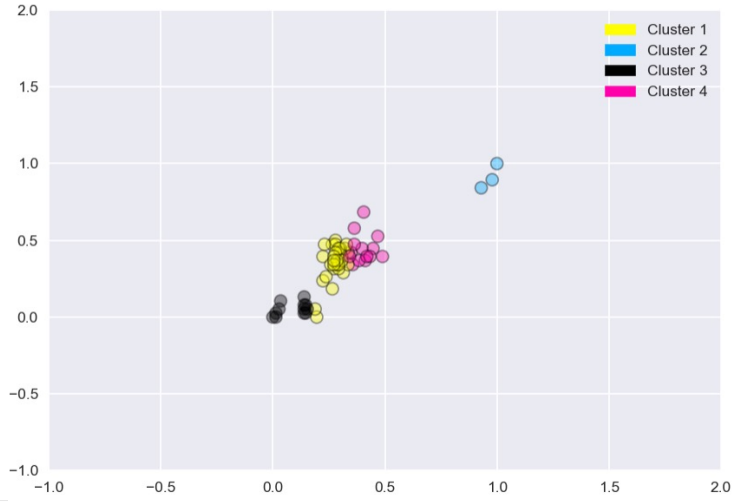
```
from sklearn.datasets import make_blobs
from sklearn.cluster import KMeans
from adspy_shared_utilities import plot_labelled_scatter
from sklearn.preprocessing import MinMaxScaler

fruits = pd.read_table('fruit_data_with_colors.txt')
X_fruits = fruits[['mass','width','height', 'color_score']].as_matrix()
y_fruits = fruits[['fruit_label']] - 1

X_fruits_normalized = MinMaxScaler().fit(X_fruits).transform(X_fruits)

kmeans = KMeans(n_clusters = 4, random_state = 0)
kmeans.fit(X_fruits)

plot_labelled_scatter(X_fruits_normalized, kmeans.labels_,
                      ['Cluster 1', 'Cluster 2', 'Cluster 3', 'Cluster 4'])
```



Can you interpret how these clusters correspond with the true fruit labels?

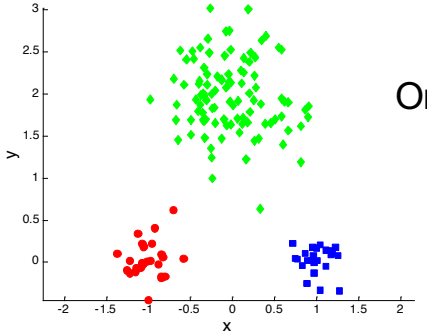
K-means clustering in practice

- Different initializations can result in different solutions.
  - Initial centroids are often chosen randomly.
  - Clusters produced vary from one run to another.
  - So multiple runs are sometimes done.
- Centroid is typically the mean of the points in the cluster.
  - K-medoid: center must be an actual datapoint. Useful when mean of a feature is not defined or available.
- 'Closeness' of points is measured by Euclidean distance, cosine similarity, correlation, etc.

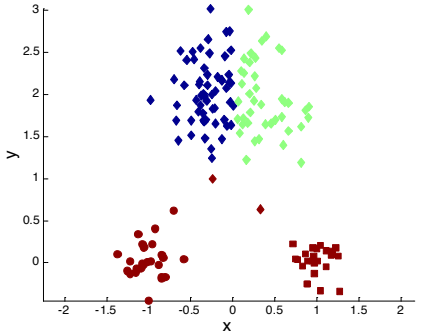
The convergence and computational complexity of k-means

- K-means will converge for commonly-used distance measures (e.g. the ones discussed here).
- Typically it will converge in a small number of iterations.
  - Results often improve only slightly after the first 10-12 iterations
  - Sometimes the stopping criterion used is: 'Until relatively few points change clusters'
- Complexity is $O(n \cdot K \cdot T \cdot d)$
  - n = number of points, K = number of clusters, T = number of iterations, d = number of attributes
- There are many variants that use heuristics to improve efficiency

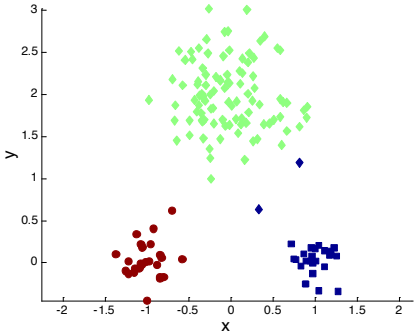
Two different K-means Clusterings



Original Points

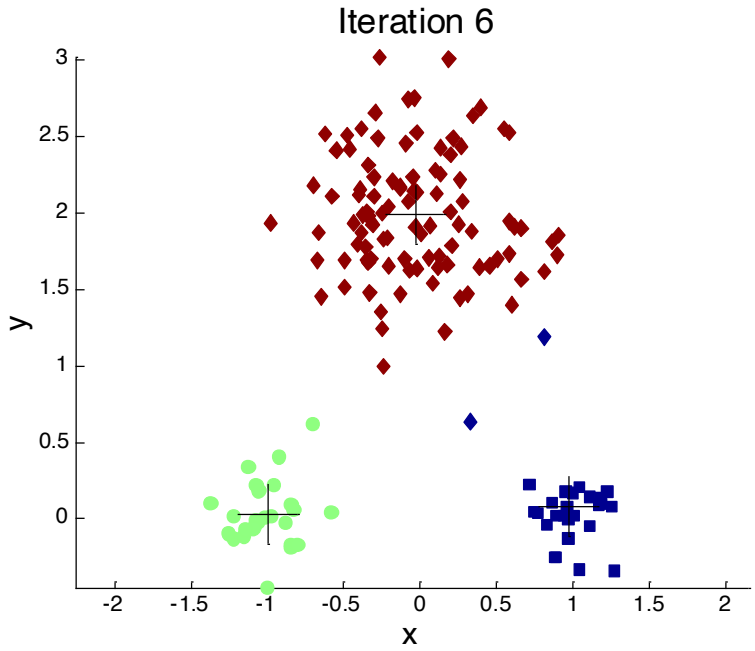


Optimal Clustering? No.

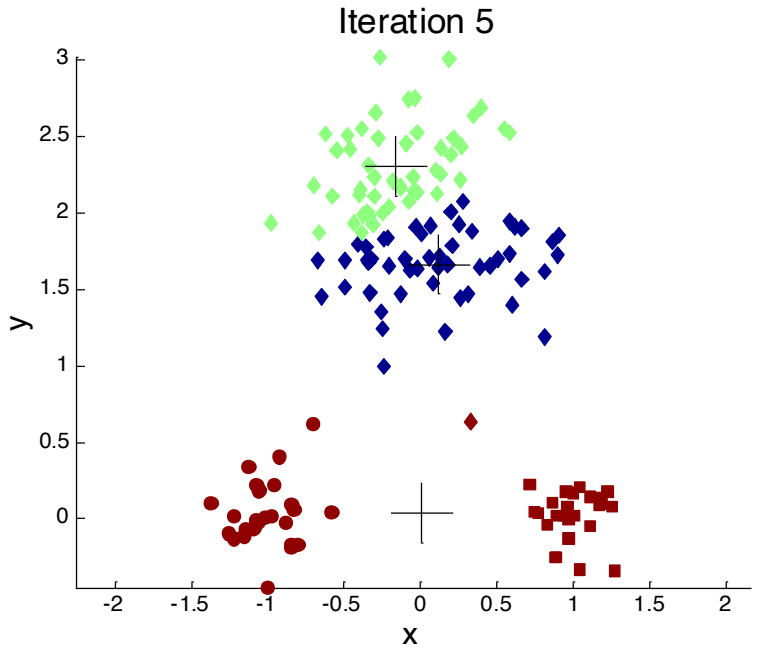


Optimal Clustering

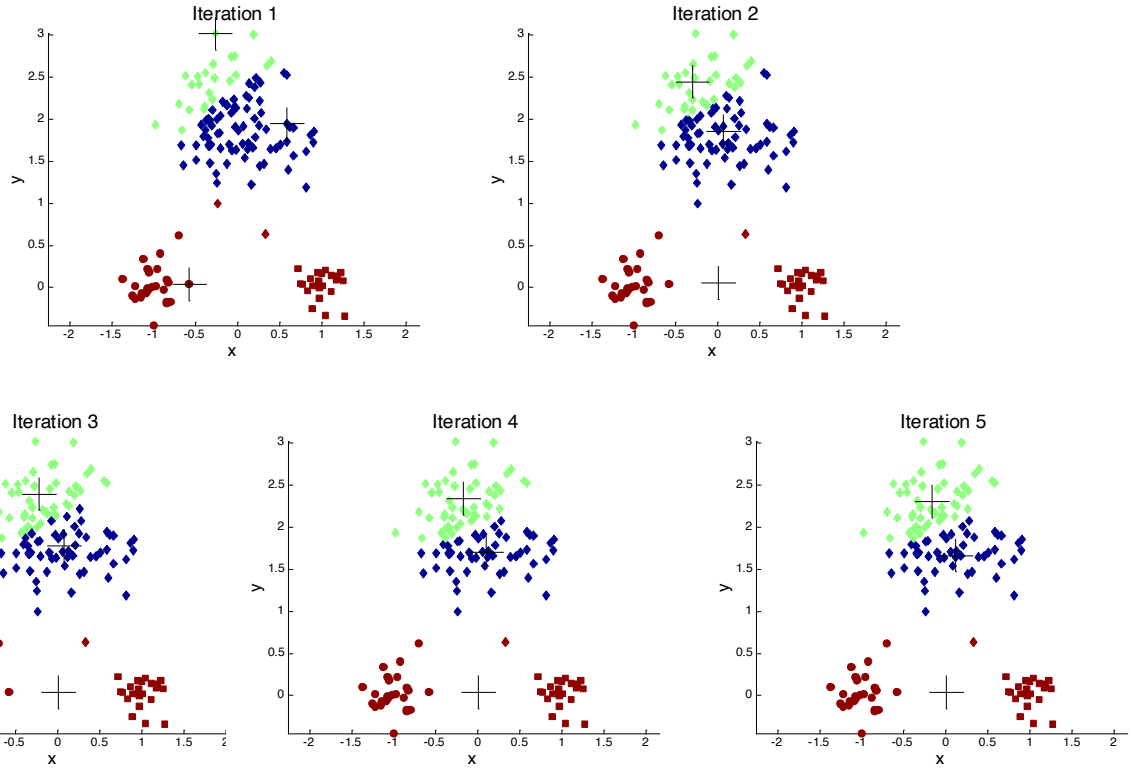
Example showing how picking the initial centroids can affect results



Example showing how picking the initial centroids can affect results



Example showing how picking the initial centroids can affect results



K-means is a special case of model-based clustering

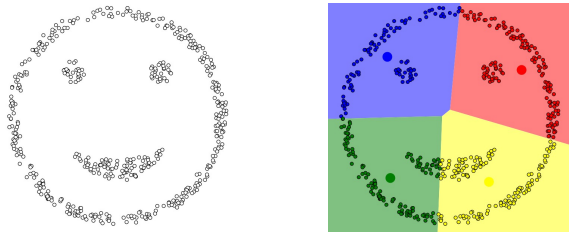
- Assume data generated from ***k*** probability distributions
- ***Goal:*** find the distribution parameters
  - K cluster centers (simple k-means) + covariances (Gaussian mixture models)
  - But the assignment of points to clusters is unknown (incomplete data)
- ***Algorithm:*** Expectation Maximization (EM)
  - We'll look at EM in detail later.
- ***Output:***
  - Distribution parameters
  - **Soft** assignment of points to clusters

Trying to find good optimal k-means clusterings

- Idea 1: Be careful about where you start:
  - Place first center on randomly chosen datapoint
  - Place second centroid on datapoint as far as possible from first center (or soft probabilistic version thereof)
  - Place *j*-th center on datapoint that's as far as possible from centers 1 thru *j* - 1
- Idea 2: Do many runs of k-means:
  - Each from a different random start configuration
  - Pick the highest-quality clustering solution that results from this set
- Many heuristics for applying k-means

Limitations of k-means

- Works well for simple clusters that are same size, well-separated, globular shapes.
- Does not do well with irregular, complex clusters with:
  - Different sizes
  - Different densities
  - Non-globular shapes
  - Outliers
- Variants of k-means like k-medoids can work with categorical features.



K-means typically performs poorly with data having complex, irregular clusters.

k-means vs hierarchical clustering

- Hierarchical clustering is appropriate when there is an inherent "tree" structure to the data.
- K-means clustering does not assume a tree structure.
- Apply dimensionality reduction and examine the cluster shapes: are you looking for similar spherical clusters (k-means), or are long chains (hierarchical) more suitable?
- k-means prefers solutions where the clusters are of similar size:
  - Very different cluster sizes, shapes, densities can confuse it
  - Complex cluster geometry, or outliers can also confuse it
  - Need to specify and test for good *k* choice

k-means vs hierarchical clustering

- Do you need to easily interpret the clusters?
- Do you know the right *k* ahead of time?
- Both methods can be used together:
  1. Try several hierarchical methods and see which gives the most interpretable clusters.
  2. Use k-means (with the hierarchical cluster centroids as starting points) to clean up the hierarchical cluster.

Kevyn Collins-Thompson

kevynct@umich.edu

School of Information

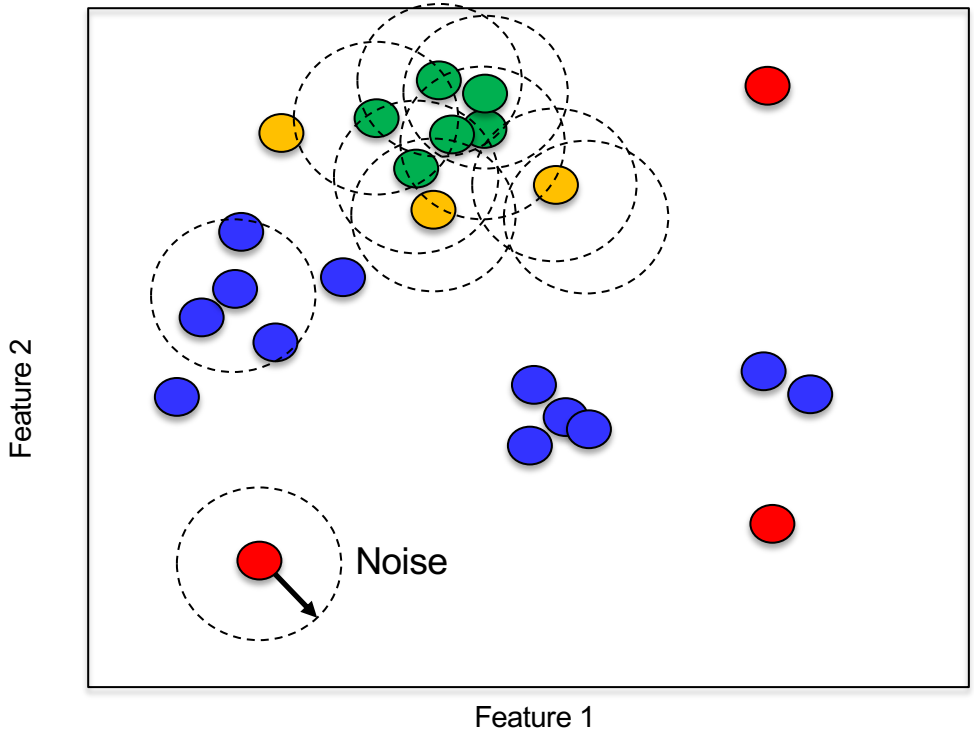
DBSCAN Clustering

Kevyn Collins-Thompson

Associate Professor of Information and Computer Science
School of Information, University of Michigan

DBSCAN Clustering

- Unlike k-means, you don't need to specify # of clusters
- Relatively efficient – can be used with large datasets
- Identifies likely noise points

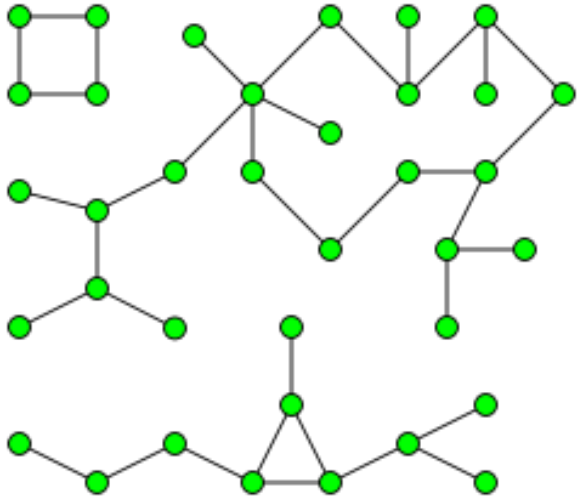


DBSCAN uses the idea of finding connected components on the graph representing neighboring points

- Connected component:

Any two vertices are connected
to each other by paths, and
is connected to no additional vertices in
the "containing" graph.

- Can be computed efficiently in linear time (vertices/edges). Breadth-first or depth-first search.

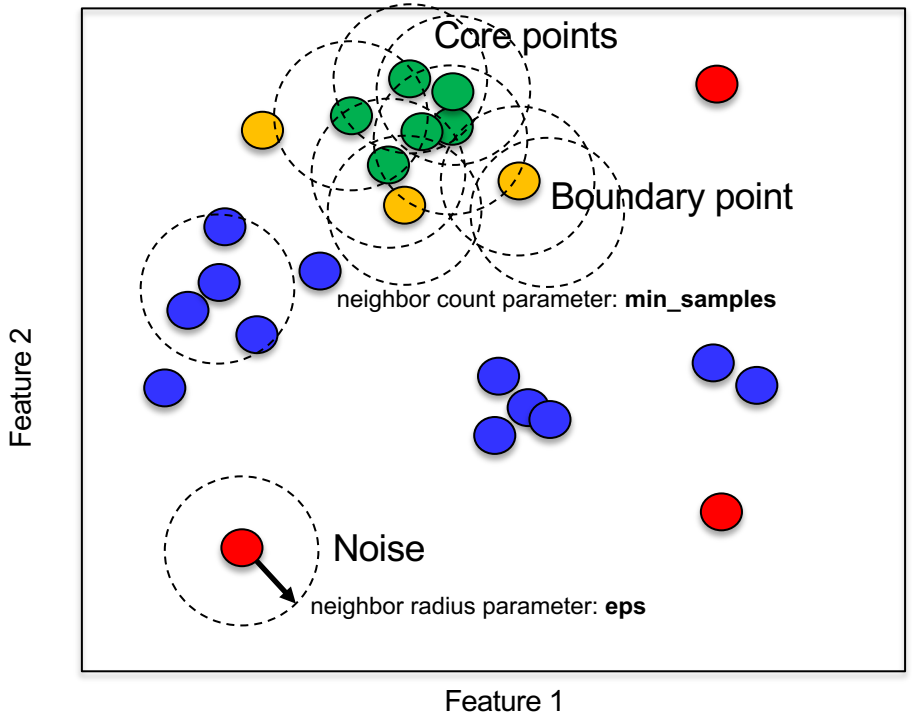


DBSCAN Clustering

1. Find the points in the ε (eps) neighborhood of every point, and identify the core points with more than min_samples neighbors.

2. Find the connected components of *core* points on the neighbor graph, ignoring all non-core points.

3. Assign each non-core point to a nearby cluster if the cluster is an ε (eps) neighbor (boundary point), otherwise assign it to noise.



DBSCAN Example in scikit-learn


```
from sklearn.cluster import DBSCAN
from sklearn.datasets import make_blobs

X, y = make_blobs(random_state = 9, n_samples = 25)

dbscan = DBSCAN(eps = 2, min_samples = 2)

cls = dbscan.fit_predict(X)
print("Cluster membership values:\n{}", format(cls))

plot_labelled_scatter(X, cls + 1,
        ['Noise', 'Cluster 0', 'Cluster 1', 'Cluster 2'])
```

```
Cluster membership values:
{} [ 0  1  0  2  0  0  0  2  2 -1  1  2  0  0 -1  0  0  1 -1  1  1  2  2  2  1]
```

How should I set DBSCAN parameters?

- Having the right settings of eps and min_samples can be critical to getting reasonable clusters.
- Clues that these parameters need further adjustment:
  - All points in your dataset are labeled -1 (noise/outlier points).
  - All points are labeled as being in the same single cluster.
- You should look at the distribution of distance values present between points in your dataset as a guide to setting “reasonable” parameters
  - Sample a subset of points (say, 10%).
  - Compute the K-nearest-neighbor distance for each point using small K (e.g. k = 1).
  - Set “eps” to some small multiple (e.g. 3x) of the median of this distance distribution.
  - Set “min_samples” to a small value (e.g. 5).
  - Try increasing multiples of “eps”, and adjusting “min_samples” to see the effect on cluster structure.

Pros and Cons of DBSCAN clustering

- Advantages:
  - Don’t need to specify the number of clusters in advance (unlike k-means)
  - Works well with datasets that have more complex cluster shapes.
  - Can also find noise points that are *outliers* that shouldn’t reasonably be assigned to any cluster.
  - DBScan is relatively efficient and can be used for large datasets.

- Disadvantages:
  - A bit slower than k-means or agglomerative clustering
  - Clustering not completely deterministic: membership of boundary points can vary between runs (usually not a big problem)
  - Quality of the clustering depends heavily on the distance measure used, and ability to choose a meaningful distance threshold eps.
  - Does not work well with data that has large differences in density: hard to find (min_samples, eps) that work well for all clusters.

Kevyn Collins-Thompson

kevynct@umich.edu

School of Information

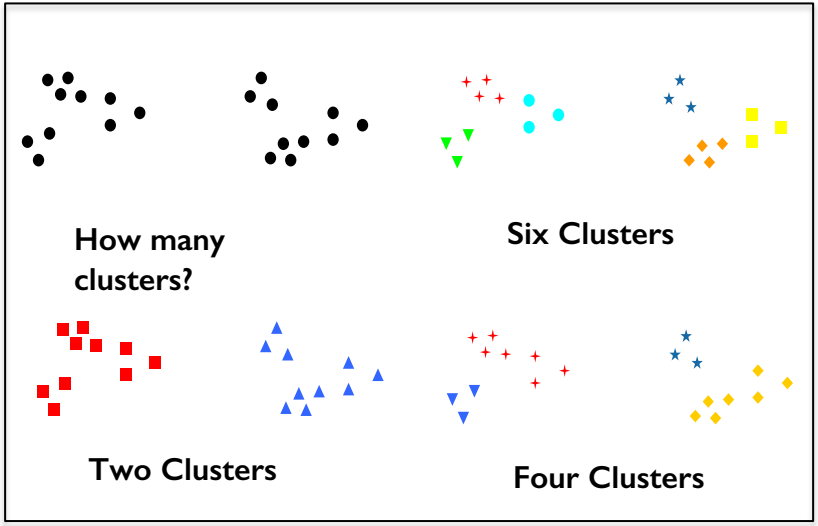
Evaluating and labeling clusters

Kevyn Collins-Thompson

Associate Professor of Information and Computer Science
School of Information, University of Michigan

Clustering Evaluation

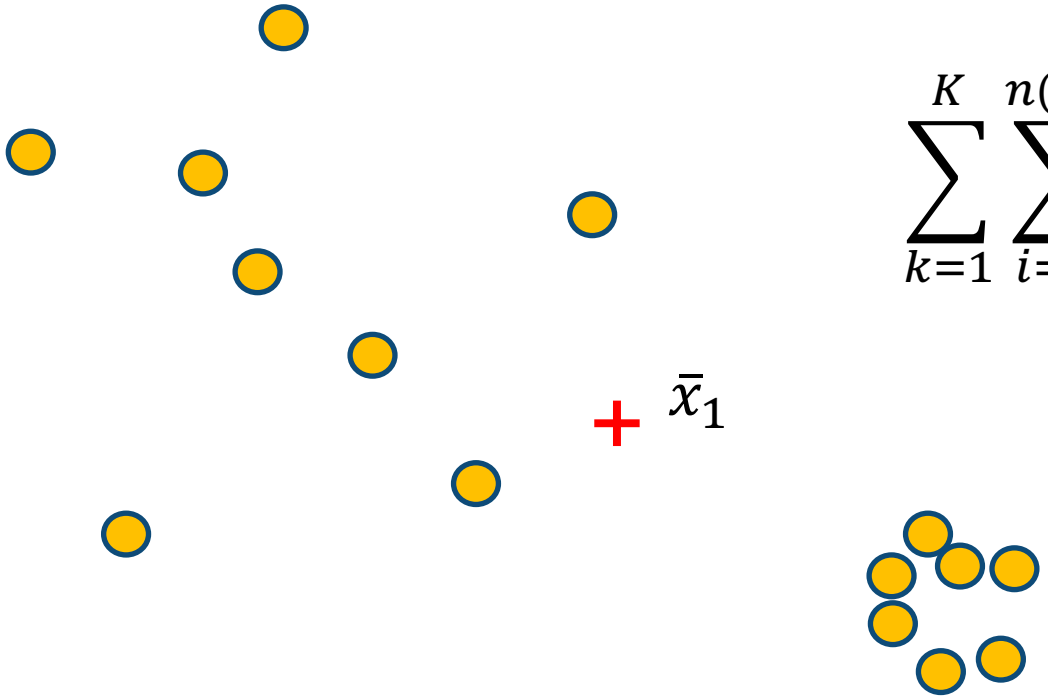
- With ground truth, existing labels can be used to evaluate cluster quality.
- Without ground truth, evaluation can difficult: multiple clusterings may be plausible for a dataset.
- Task-based evaluation: Evaluate clustering according to performance on a task that does have an objective basis for comparison.
- Example: the effectiveness of clustering-based features for text classification.



What is the optimal # of clusters?

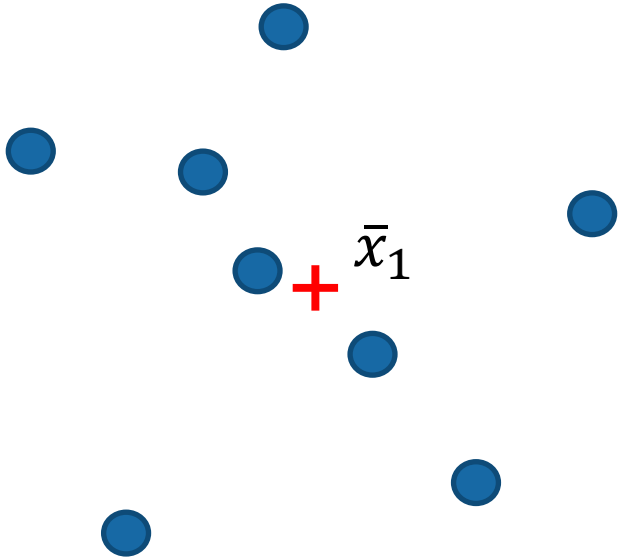
- Theoretical, conceptual or practical issues may guide the choice of number of clusters *a priori*.
- Hierarchical clustering:
  - Distance threshold at which clusters are combined.
- K-means and other non-hierarchical methods:
  - Ratio of total within-groups variance to between-group variance, vs # of clusters.
  - Special case: within-groups sum of squares vs # of clusters.
  - Elbow/sharp bend shows point at which adding more clusters helps reduce distortion measure less and less.

Example: Within-groups sum of squares: sum over squared distances of members to centroid

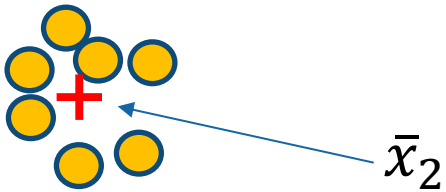


$$\sum_{k=1}^{K} \sum_{i=1}^{n(k)} (x_{ik} - \bar{x}_k)^2$$

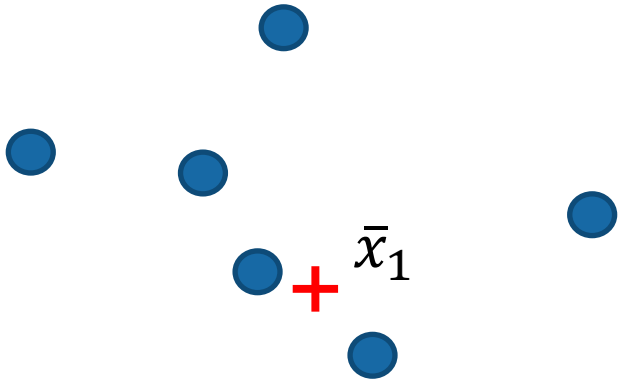
Increasing # of clusters generally reduces within-groups sum of squares



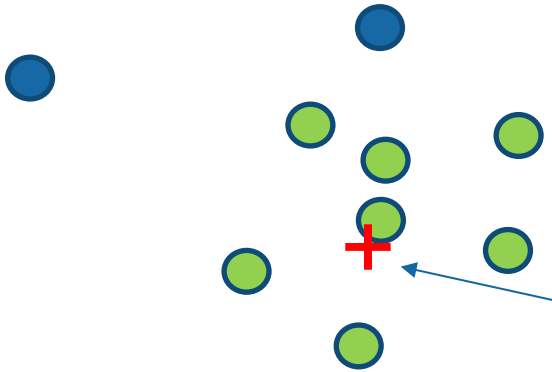
$$\sum_{k=1}^{K} \sum_{i=1}^{n(k)} (x_{ik} - \bar{x}_k)^2$$



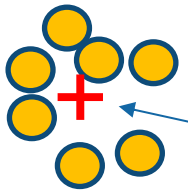
Between-group variance: sum over squared distances between centroids



$\bar{x}_1$



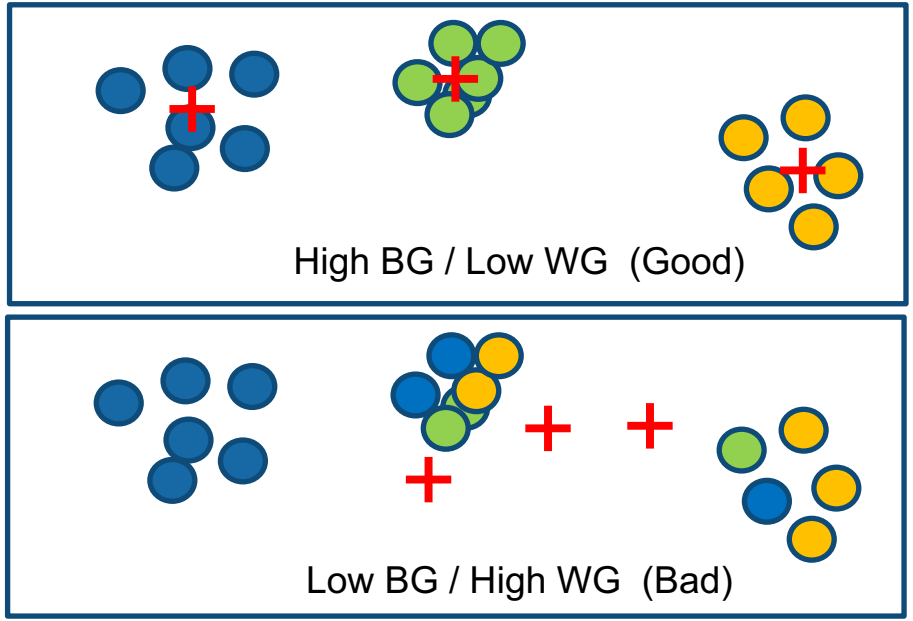
$\bar{x}_3$



$\bar{x}_2$

$$\sum_{j=1}^{K}\sum_{k=1}^{K}\left(\bar{x}_j - \bar{x}_k\right)^2$$

Ratio of between-groups to within-group variance

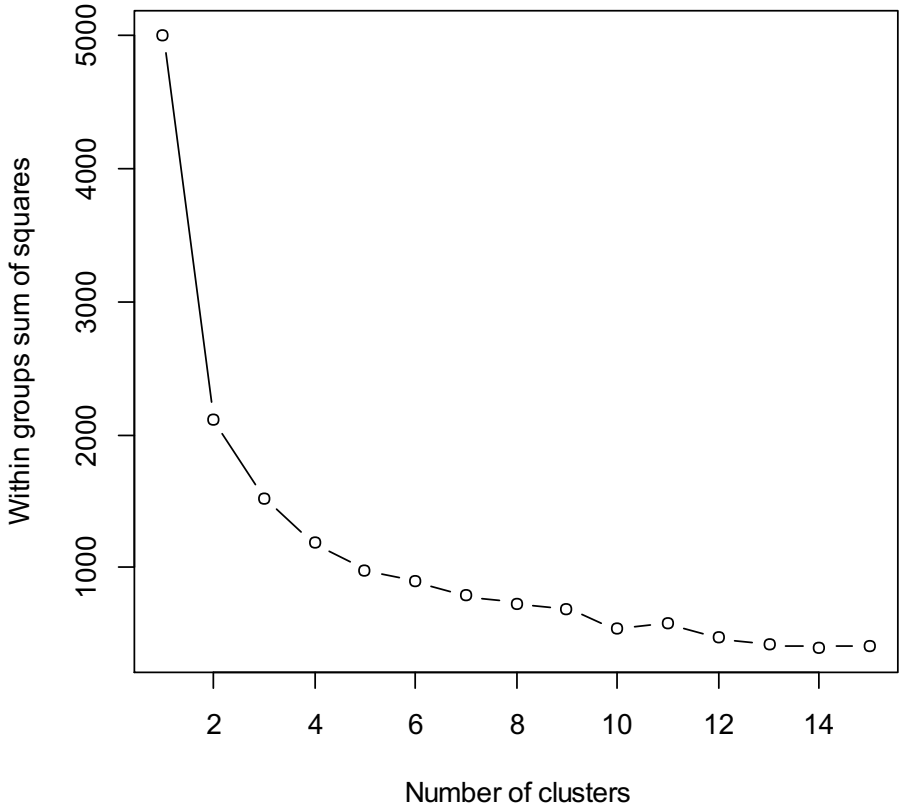


Between-group variance

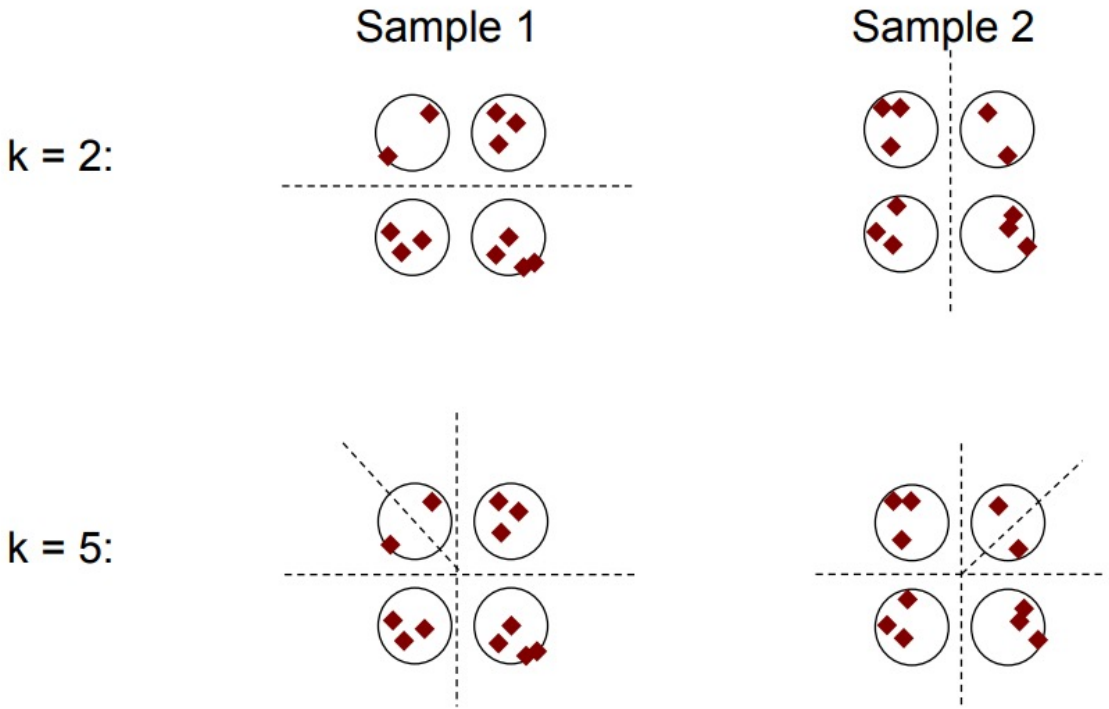
$$\frac{\sum_{j=1}^{K}\sum_{k=1}^{K}\left(\bar{x}_j - \bar{x}_k\right)^2}{\sum_{k=1}^{K}\sum_{i=1}^{n(k)}\left(x_{ik} - \bar{x}_k\right)^2}$$

Within-group variance

What is the optimal # of clusters?



Choosing the number of clusters based on stability



Source: von Luxburg. Clustering stability: an overview
https://people.eecs.berkeley.edu/~jordan/sail/readings/luxburg_ftml.pdf

How do we know if we've found *good quality* clusters?

1. Compare cluster stability across:
   - Different distance measures
   - Different clustering methods
   - Different 50/50 random data splits
   - Different variable/features deletions
   - Different data orderings (non-hierarchical)

   "Good" clusterings (if they exist) are generally stable and robust to perturbations in methods or data.
2. Use cluster quality metrics.
   - Do you have ground truth labels for cluster membership?
   - Let's look at methods (1) with and (2) without ground truth.

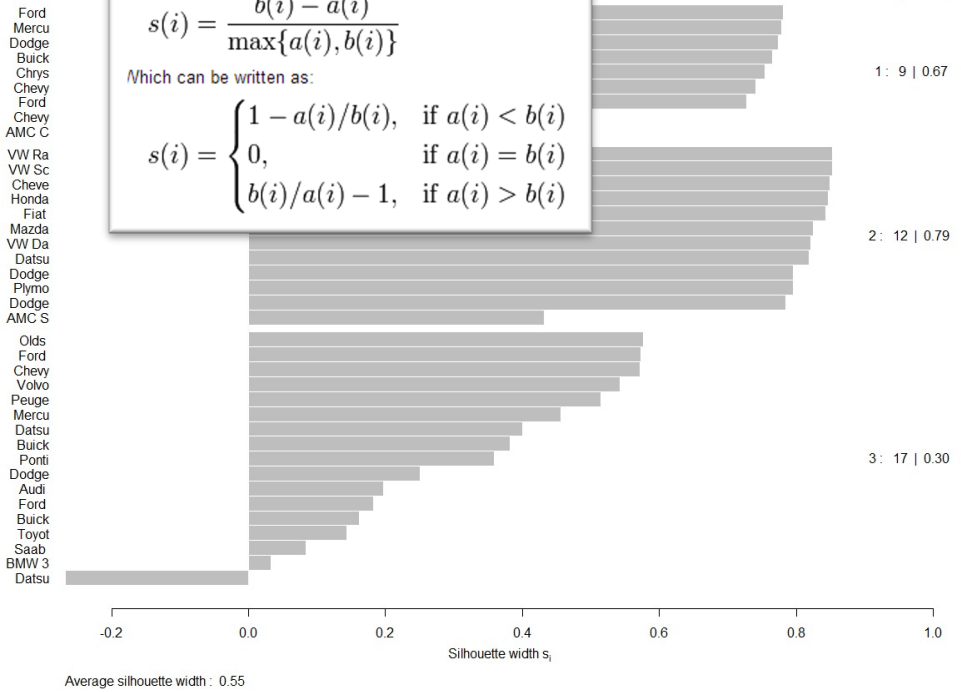
Silhouette scores (no ground truth needed)

- A graphical aid for interpretation and validation of cluster analysis.
- *a*(*i*) : average dissimilarity of instance *i* with others in same cluster.
- *b*(*i*): lowest average dissimilarity for other clusters (neighboring cluster).
- Gives the degree of confidence in cluster assignment.
  - Well-clustered elements: score near 1.
  - Near zero: Close to decision boundary between two neighboring clusters
  - Poorly-clustered elements: score near -1 (probably in wrong cluster)

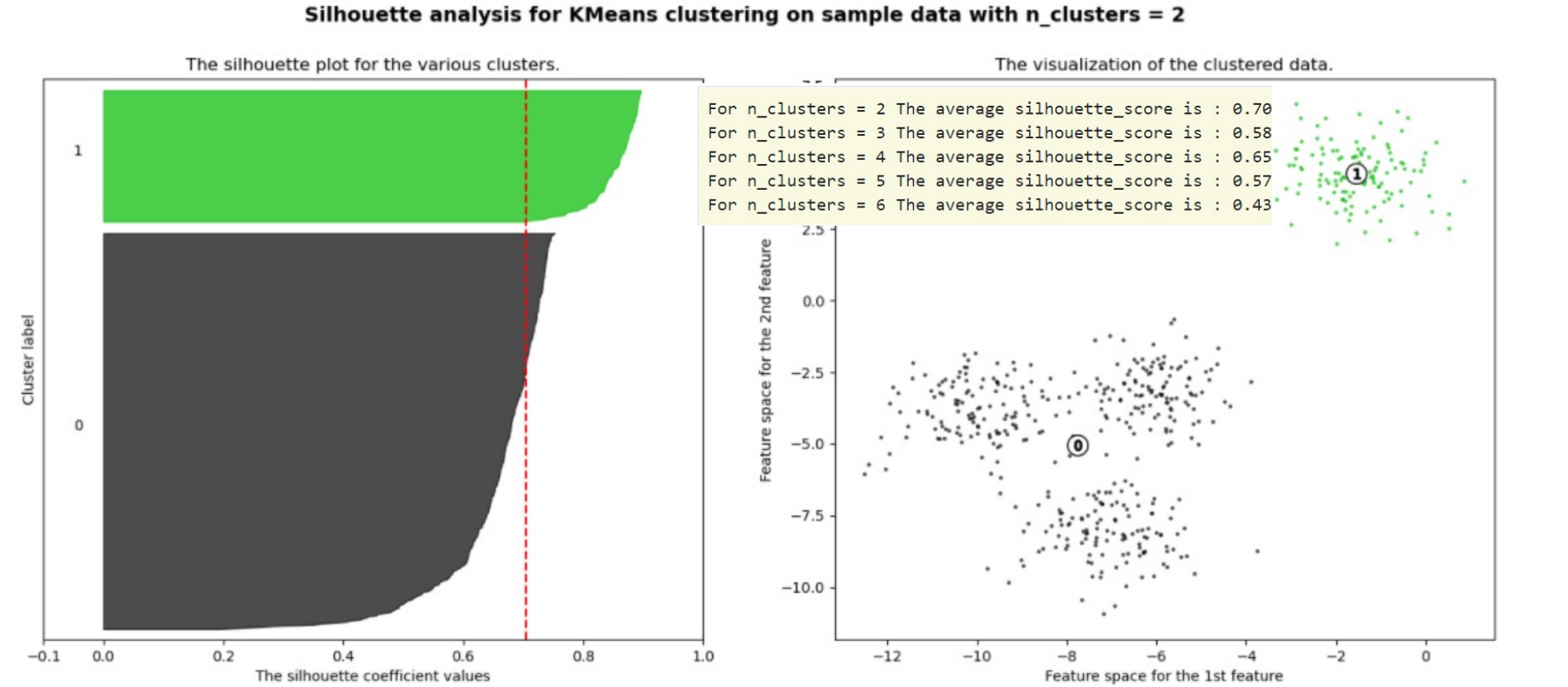
$$s(i) = \frac{b(i) - a(i)}{\max\{a(i), b(i)\}}$$

Which can be written as:

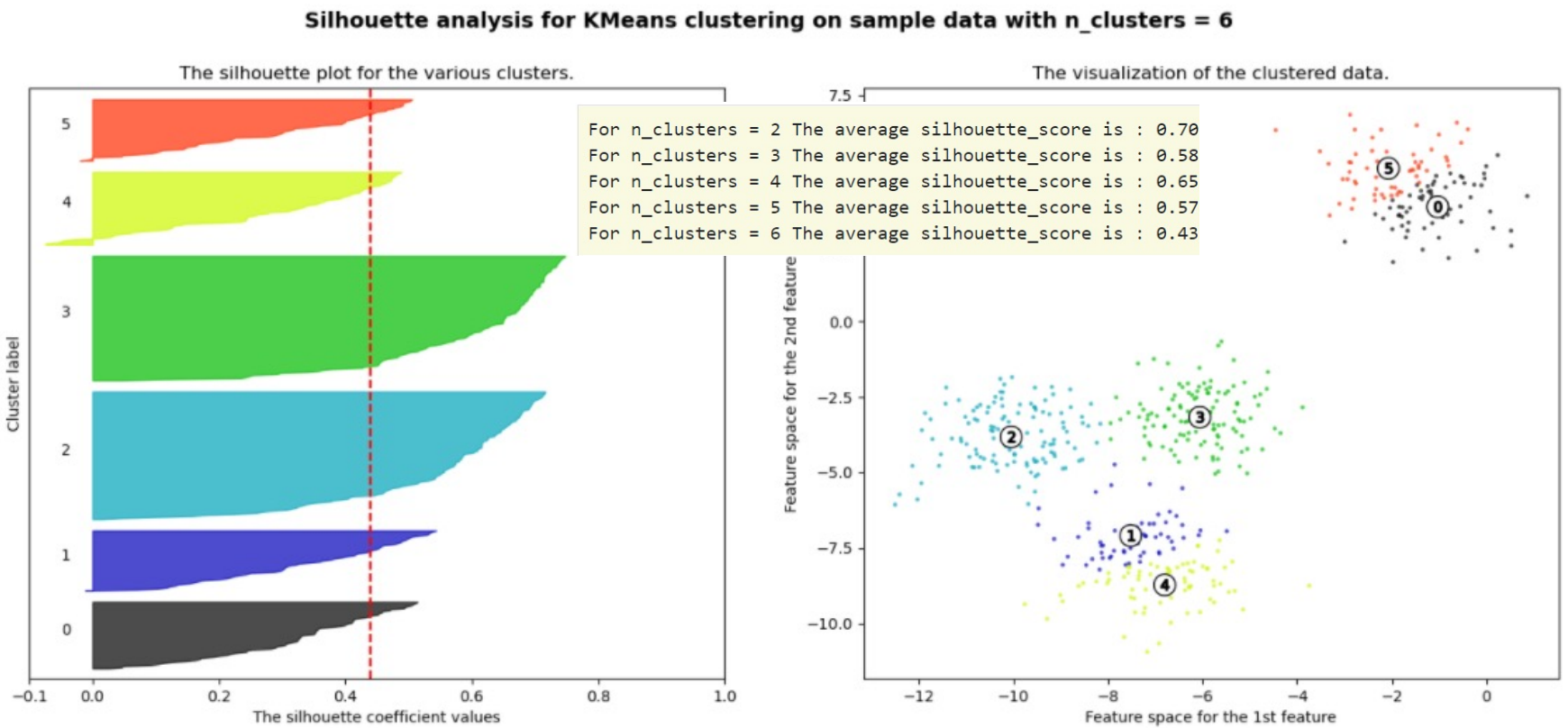
$$s(i) = \begin{cases} 1 - a(i)/b(i), & \text{if } a(i) < b(i) \\ 0, & \text{if } a(i) = b(i) \\ b(i)/a(i) - 1, & \text{if } a(i) > b(i) \end{cases}$$



Source: Peter J. Rousseeuw (1987). "Silhouettes: a Graphical Aid to the Interpretation and Validation of Cluster Analysis". *Computational and Applied Mathematics* **20**: 53–65.



Source: https://scikit-learn.org/stable/auto_examples/cluster/plot_kmeans_silhouette_analysis.html

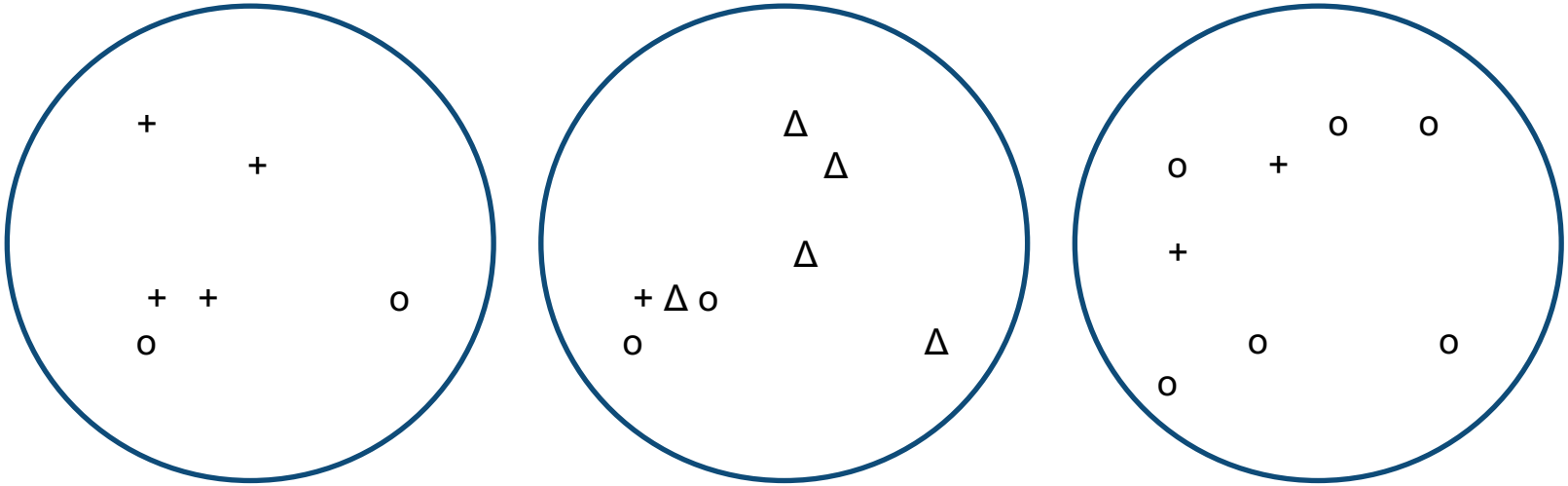


Source: https://scikit-learn.org/stable/auto_examples/cluster/plot_kmeans_silhouette_analysis.html

Cluster homogeneity & completeness (needs ground truth)

Homogeneity / purity : each cluster only has instances in same class
Completeness: all members of a class are assigned to the same cluster.

Both scores are between 0.0 and 1.0 Requires ground truth labels



Cluster 1

Majority class: +

Cluster 2

Majority class: Δ

Cluster 3

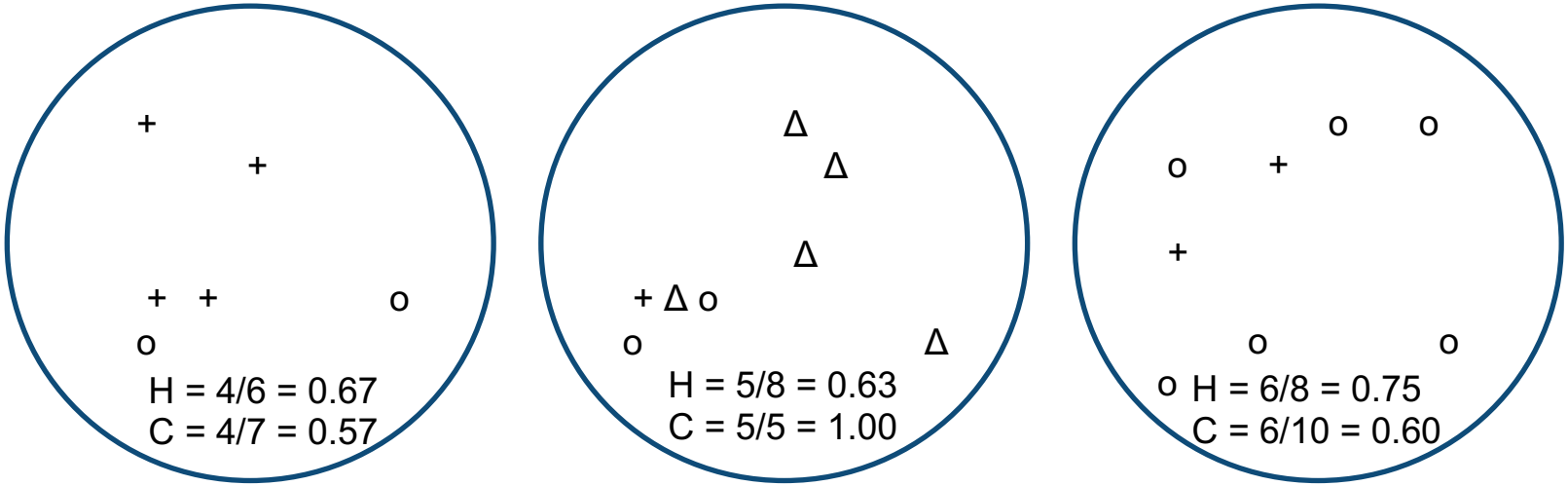
Majority class: o

Cluster homogeneity & completeness (needs ground truth)

Homogeneity / purity : each cluster only has instances in same class
Completeness: all members of a class are assigned to the same cluster.

Both scores are between 0.0 and 1.0

Requires ground truth labels



Cluster 1

Majority class: +

Cluster 2

Majority class: Δ

Cluster 3

Majority class: o

Adjusted Rand Index (needs ground truth)

- a = # pairs that are in same ground truth class AND same clustering class (1 credit)
- b = # pairs that are in different ground truth classes AND different clustering classes (1 credit)
- N_{PAIRS} = C choose 2 = total # of possible pairs

$$\mathbf{RI = a + b / N_{PAIRS}}$$

- In regular usage, we normalize by E[RI] : expected RI of random labelings
- $$\mathbf{ARI = RI – E[RI] \ / max(RI) – E[RI]}$$
- Perfect labeling scores 1.0
- Bad/random labeling has zero or negative scores (down to -1.0)
- No assumptions about cluster structure.
  - So we can use it to compare clusterings from very different methods.

scikit-learn supports a variety of cluster quality metrics that don’t need ground-truth labels.

- Davies-Bouldin score:
  - No ground truth labels needed.
  - *Lower* is better for the Davies-Bouldin score.
  - This index signifies the average ‘similarity’ between clusters, where the similarity is a measure that compares the distance between clusters with the size of the clusters themselves.
  - Zero is the lowest possible score. Values closer to zero indicate a better partition.

scikit-learn supports a variety of cluster quality metrics that don't need ground-truth labels.

- Calinski-Harabasz index (a.k.a. the Variance Ratio Criterion)
  - No ground truth labels needed.
  - *Higher* is better for the Calinski-Harabasz index: means better-defined clusters.
  - The index is the ratio of:
    - The mean between-clusters dispersion and
    - The inter-cluster dispersion for all clusters (where dispersion is defined as the sum of distances squared).
  - The score is higher when clusters are dense & well separated.
  - Fast to compute.

scikit-learn example: metrics without ground truth

```
from sklearn.cluster import KMeans
from sklearn import metrics
```

```
km = KMeans(n_clusters=n_trial, init='k-means++', max_iter=100, n_init=1,
        verbose=False, random_state =42)

km.fit(X)

print("Silhouette Coefficient: %0.3f"
  % metrics.silhouette_score(X, km.labels_, sample_size=1000))
print("Davies-Bouldin Score: %0.3f"
  % metrics.davies_bouldin_score(X.toarray(), km.labels_))
print("Calinski-Harbasz Score: %0.3f"
  % metrics.calinski_harabasz_score(X.toarray(), km.labels_))
```

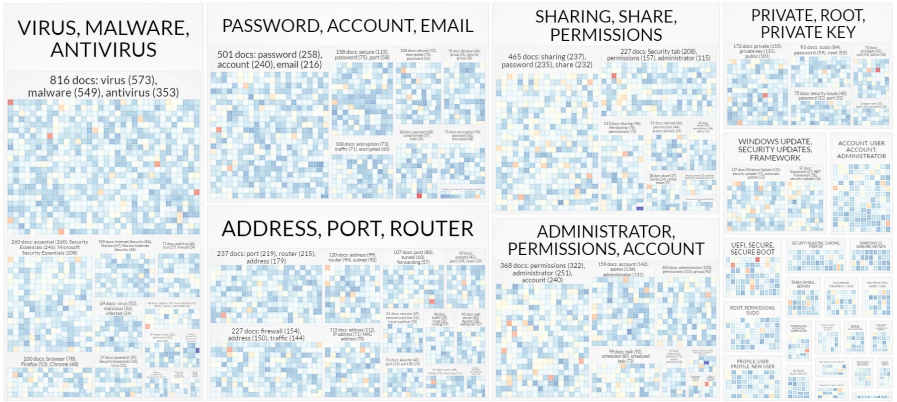
scikit-learn example: metrics with ground truth

```
from sklearn.cluster import KMeans
from sklearn import metrics
```

```
print("Homogeneity: %0.3f" % metrics.homogeneity_score(labels, km.labels_))
print("Completeness: %0.3f" % metrics.completeness_score(labels, km.labels_))
print("V-measure: %0.3f" % metrics.v_measure_score(labels, km.labels_))
print("Adjusted Rand-Index: %.3f" % metrics.adjusted_rand_score(labels, km.labels_))
```

Cluster labeling

- For user interface or data exploration tasks, humans need to interact with clusters.
- So clusters need human-friendly titles/labels to summarize what they're about.
- How could we get 'useful' labels automatically?



Document clusters returned for the query "security" with Carrot Search Lingo4G clustering engine.
Colors indicate popular documents.
Labels are based on high-frequency terms in each cluster.

Source: https://get.carrotsearch.com/lingo4g/latest/doc/

Differential vs internal labeling

- Differential cluster labeling:
  - Compare distribution of terms in one cluster with those of other clusters.
  - Mutual information, chi-squared.
  - Other feature selection methods.

		labeling method		
	# docs	centroid	mutual information	title
4	622	oil plant mexico production crude **power 000 refinery gas** bpd	plant oil production **barrels** crude bpd mexico **dolly capacity petroleum**	MEXICO: Hurricane Dolly heads for Mexico coast
9	1017	police security **russian** people military peace killed told **grozny court**	police killed military security peace told **troops forces rebels** people	RUSSIA: Russia's Lebed meets rebel chief in Chechnya
10	1259	00 000 tonnes traders futures wheat prices **cents september** tonne	**delivery** traders futures tonne tonnes **desk** wheat prices 000 00	USA: Export Business - Grain/oilseeds complex

K-means clustering of news articles: three different cluster labeling methods. Bold indicates selected terms unique to that method.

Source: https://nlp.stanford.edu/IR-book/html/htmledition/cluster-labeling-1.html

Differential vs internal labeling

- Differential cluster labeling:
  - Compare distribution of terms in one cluster with those of other clusters.
  - Mutual information, chi-squared.
  - Other feature selection methods.
- Cluster internal labeling:
  - Use centroid high-weight terms.
  - Use a text summary of a representative central instance (e.g. article title, if available!)

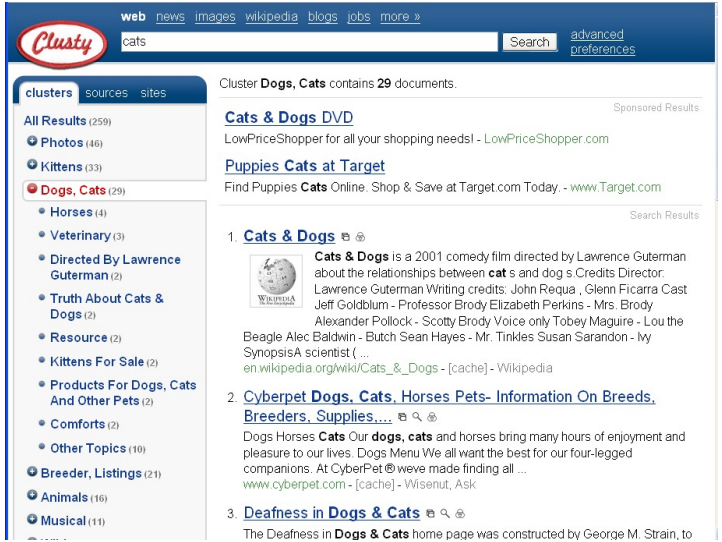
		labeling method		
	# docs	centroid	mutual information	title
4	622	oil plant mexico production crude **power 000 refinery gas** bpd	plant oil production **barrels** crude bpd mexico **dolly capacity petroleum**	MEXICO: Hurricane Dolly heads for Mexico coast
9	1017	police security **russian** people military peace killed told **grozny court**	police killed military security peace told **troops forces rebels** people	RUSSIA: Russia's Lebed meets rebel chief in Chechnya
10	1259	00 000 tonnes traders futures wheat prices **cents september** tonne	**delivery** traders futures tonne tonnes **desk** wheat prices 000 00	USA: Export Business - Grain/oilseeds complex

K-means clustering of news articles: three different cluster labeling methods. Bold indicates selected terms unique to that method.

Source: https://nlp.stanford.edu/IR-book/html/htmledition/cluster-labeling-1.html

How to automatically name or label clusters?

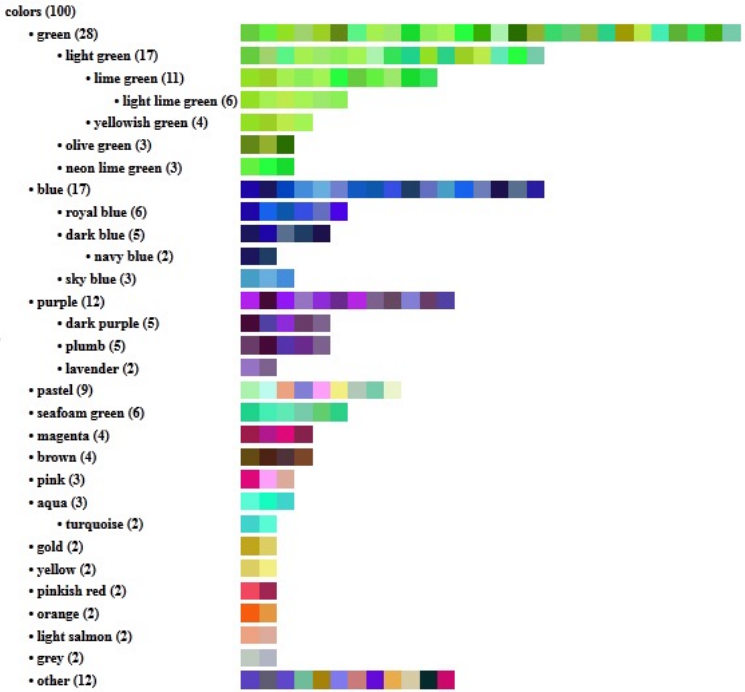
- Classify with existing hierarchy.
  e.g. Wikipedia topics or Open Directory.
- What are ‘good’ vs ‘bad’ cluster names anyway?
  - Usually based on a given task.
    e.g. search interface effect on user satisfaction/effectiveness, or time saved.
- Or maybe you don’t actually need cluster labels.
  - Benefits may be unclear, depending on scenario.



See: http://searchuserinterfaces.com/book/sui_ch8_navigation_and_search.html

How to automatically name or label clusters?

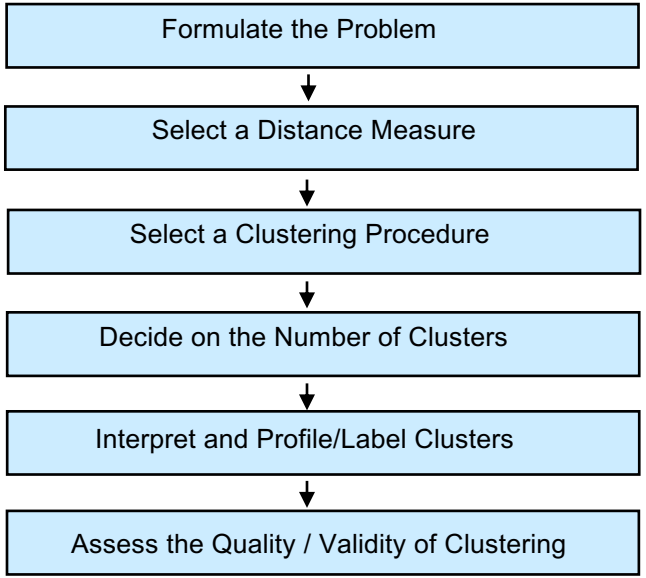
- Use crowd-powered methods to create and label object hierarchies/taxonomies
  e.g. [Chilton et al, CHI 2013]



- Easy, quick crowd tasks:
  - Generate categories
  - Pick the best categories for an item
  - Item-category membership decisions
- 80-90% hierarchy quality vs experts

Source: http://www.cs.columbia.edu/~chilton/web/my_publications/ChiltonCascadeCHI2013.pdf

Summary: Conducting Cluster Analysis



Kevyn Collins-Thompson

kevynct@umich.edu

School of Information