CLUSTERING

Unsupervised Learning and Clustering

- In unsupervised learning you are given a data set with no output classifications (labels)
- Clustering is an important type of unsupervised learning
  - PCA was another type of unsupervised learning
- The goal in clustering is to find "natural" clusters (classes) into which the data can be divided – a particular breakdown into clusters is a *clustering* (aka grouping, partition)
- How many clusters should there be (k)? – Either user-defined, discovered by trial and error, or automatically derived
- Example: Taxonomy of the species – one correct answer?
- Generalization – After clustering, when given a novel instance, we just assign it to the most similar cluster

Clustering

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

Outlier Handling

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

Distances Between Clusters

- Easy to measure distance between instances (elements, points), but how about the distance of an instance to another cluster or the distance between 2 clusters

- Can represent a cluster with
  - Centroid – cluster mean
    - Then just measure distance to the centroid
  - Medoid – an actual instance which is most typical of the cluster (e.g. medoid is point which would make the average distance from it to the other points the smallest)

- Other common distances between two Clusters *A* and *B*
  - Single link – Smallest distance between any 2 points in *A* and *B*
  - Complete link – Largest distance between any 2 points in *A* and *B*
  - Average link – Average distance between points in *A* and points in *B*

K-MEANS

k-Means

- Perhaps the most well-known clustering algorithm
  - Partitioning algorithm
  - Must choose a *k* beforehand
  - Thus, typically try a spread of different *k*'s (e.g. 2-10) and then compare results to see which made the best clustering
    - Could use cluster validity metrics (e.g. Silhouette) to help in the decision
- Time complexity is O(*mkn*) where *m* is # of iterations and space is O(*n*), both much better than HAC time and space (n^3 and n^2)

k-Means Continued

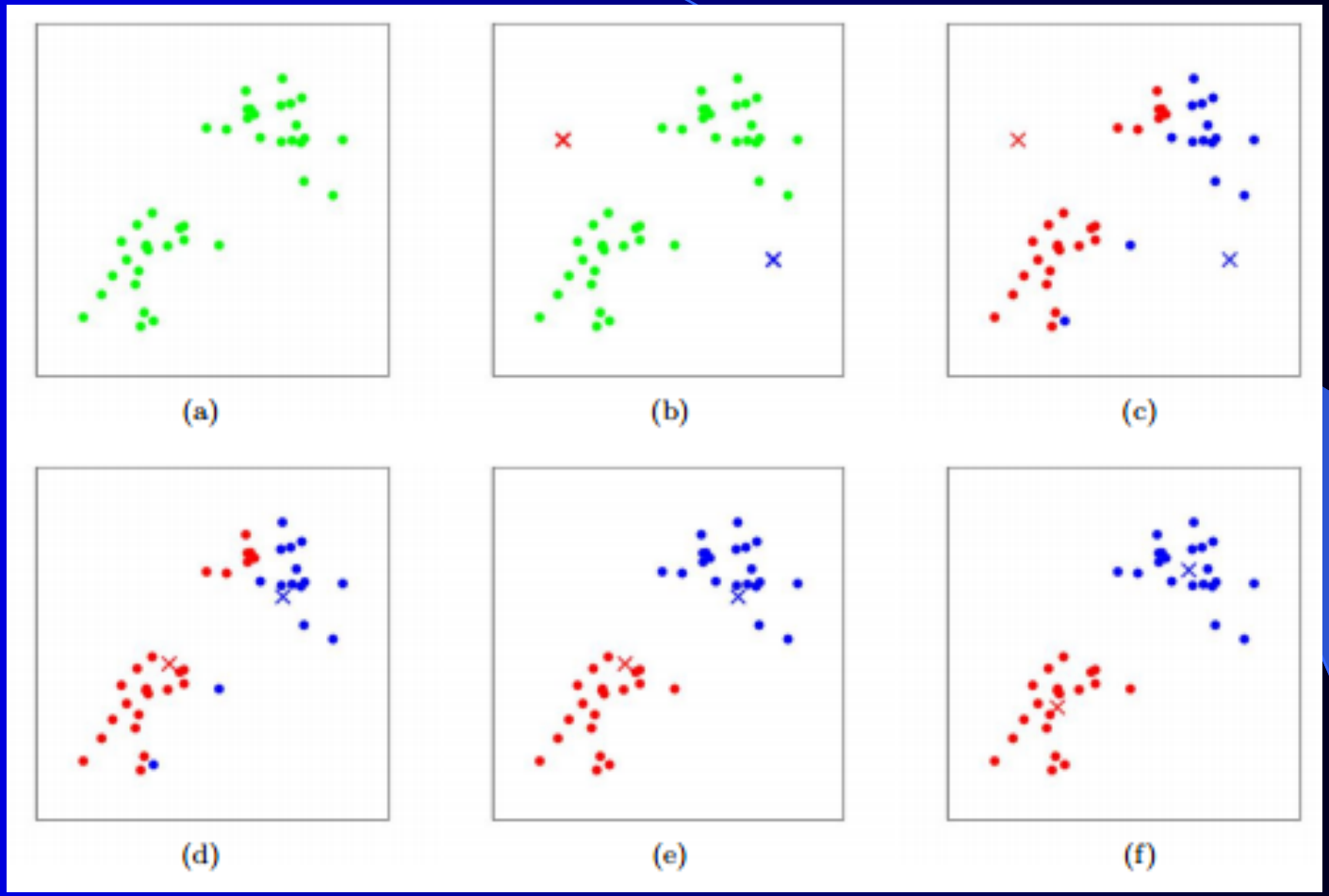
- Type of EM (Expectation-Maximization) algorithm, Gradient descent
  - Can struggle with local minima, unlucky random initial centroids, and outliers
    - k-medoids finds medoid (median) centers rather than average centers and is thus less effected by outliers
  - Local minima, empty clusters: Can just re-run with different initial centroids
    - Could compare different solutions *for a specific k value* by seeing which clusterings minimize the overall SSE to the cluster centers (i.e. compactness), or use silhouette, etc.
    - And test solutions with different *k* values using Silhouette or other metric

k-Means Algorithm

1. Choose k
2. Randomly initialize k centroids $\{C_1, C_2, \ldots, C_k\}$
3. While k-centroids keep changing:
   - Assign each point in the data to the cluster S_i whose centroid C_i is closest
   - Recalculate $\{C_1, C_2, \ldots, C_k\}$ based on the points in the clusters $\{S_1, S_2, \ldots, S_k\}$

$$\mathbf{C}_i = \frac{1}{|S_i|} \sum_{\mathbf{x} \in S_i} \mathbf{x}$$

k-Means Example



** k-Means Challenge Question **

- For the data below, show the centroid values and which instances are closest to each centroid *after* centroid calculation for two iterations of k-means using Manhattan distance
- By 2 iterations I mean 2 centroid changes after the initial centroids
- Assume $k = 2$ and that the first two instances are the initial centroids

Instance	*x*	*y*
a	3	1
b	4	0
c	0	0
d	0	1

Iteration	*Centroid 1 and instances*	*Centroid 2 and instances*
0		
1		
2		

1. Repeat until no more changes occur
   a) Group each instance with its closest centroid
   b) Recalculate the centroid based on its new cluster

** k-Means Challenge Question **

- For the data below, show the centroid values and which instances are closest to each centroid *after* centroid calculation for two iterations of k-means using Manhattan distance
- By 2 iterations I mean 2 centroid changes after the initial centroids
- Assume $k = 2$ and that the first two instances are the initial centroids

Instance	*x*	*y*
a	3	1
b	4	0
c	0	0
d	0	1

Iteration	*Centroid 1 and instances*	*Centroid 2 and instances*
0	(3, 1), {*a, c, d*}	(4, 0), {*b*}
1	(1, .66), {*c, d*}	(4, 0), {*a, b*}
2	(0, .5), {*c, d*}	(3.5, .5), {*a, b*}
Final	(0, .5), {*c, d*}	(3.5, .5), {*a, b*}

k-Means Homework

- For the data below, show the centroid values and which instances are closest to each centroid *after* centroid calculation for two iterations of k-means using Manhattan distance
- By 2 iterations I mean 2 centroid changes after the initial centroids
- Assume $k = 2$ and that the first two instances are the initial centroids

Instance	*x*	*y*
a	.9	.8
b	.2	.2
c	.7	.6
d	-.1	-.6
e	.5	.5

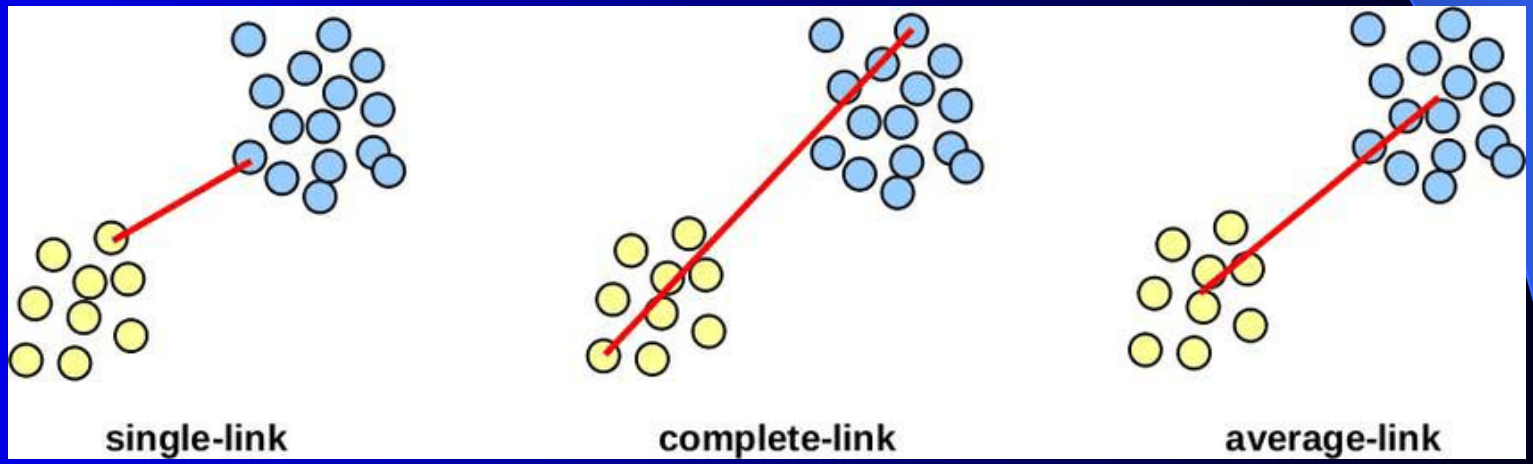
HIERARCHICAL CLUSTERING

Hierarchical Clustering

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

Hierarchical Agglomerative Clustering (HAC)

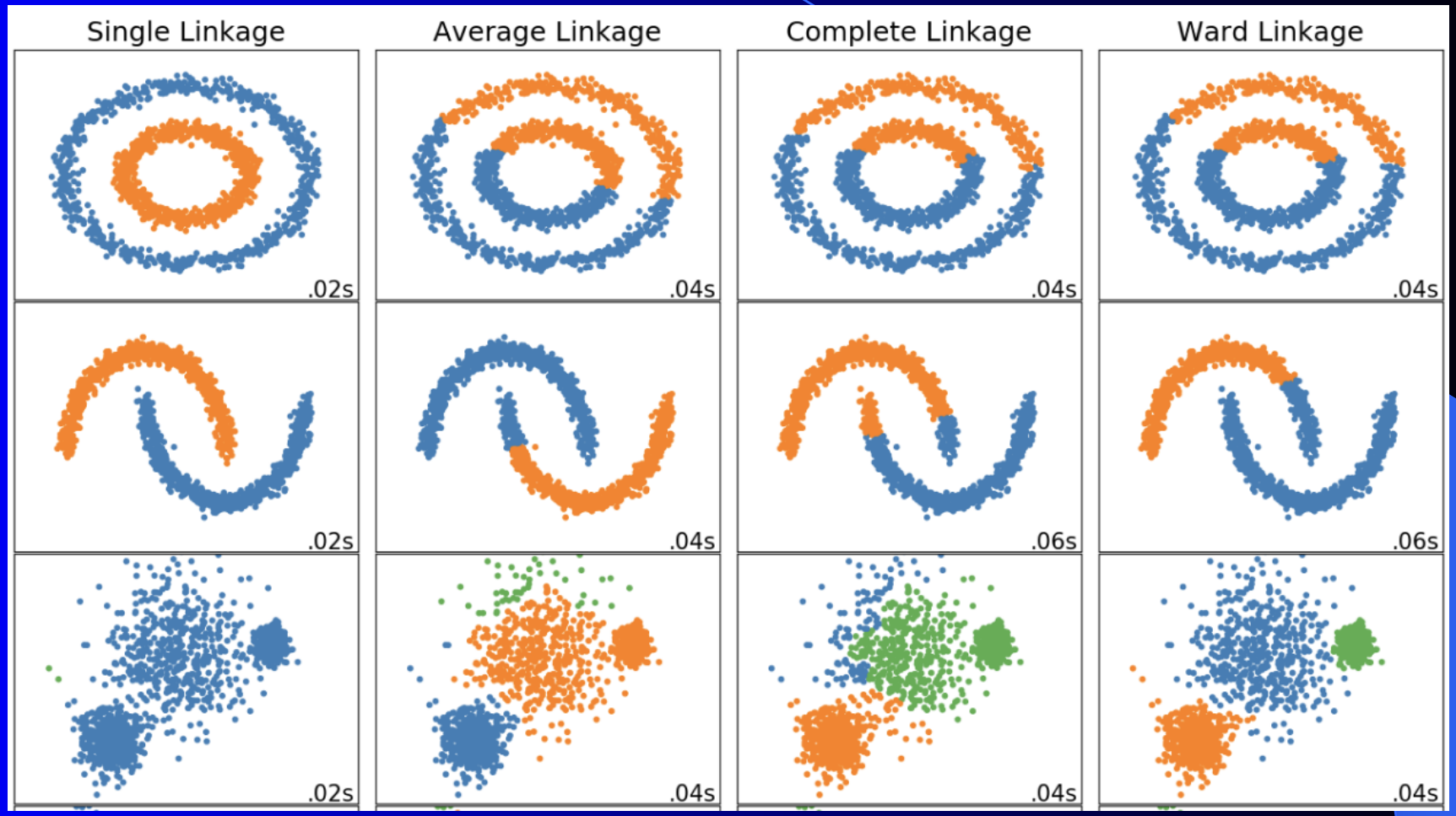
- Input is an $n \times n$ adjacency matrix giving the distance between each pair of instances
- Initialize each instance to be its own cluster
- Repeat until there is just one cluster containing all instances
  - Merge the two "closest" remaining clusters into one cluster
- HAC varies based on "Closeness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



HAC Linkages

- Single link – (nearest neighbor) can lead to long chained clusters where some points are quite far from each other
  - $D(C_I, C_J) = \min_{\mathbf{x} \in C_I, \mathbf{y} \in C_J} \{\text{dist}(\mathbf{x}, \mathbf{y})\}$
- Complete link – (farthest neighbor) finds more compact clusters
  - $D(C_I, C_J) = \max_{\mathbf{x} \in C_I, \mathbf{y} \in C_J} \{\text{dist}(\mathbf{x}, \mathbf{y})\}$
- Average link – Used less because have to re-compute the average each time
- Ward linkage – the distance between two clusters is the amount the SSE changes by when two clusters would be merged
- Once you have distances between clusters, always merge the closest clusters in terms of the distance

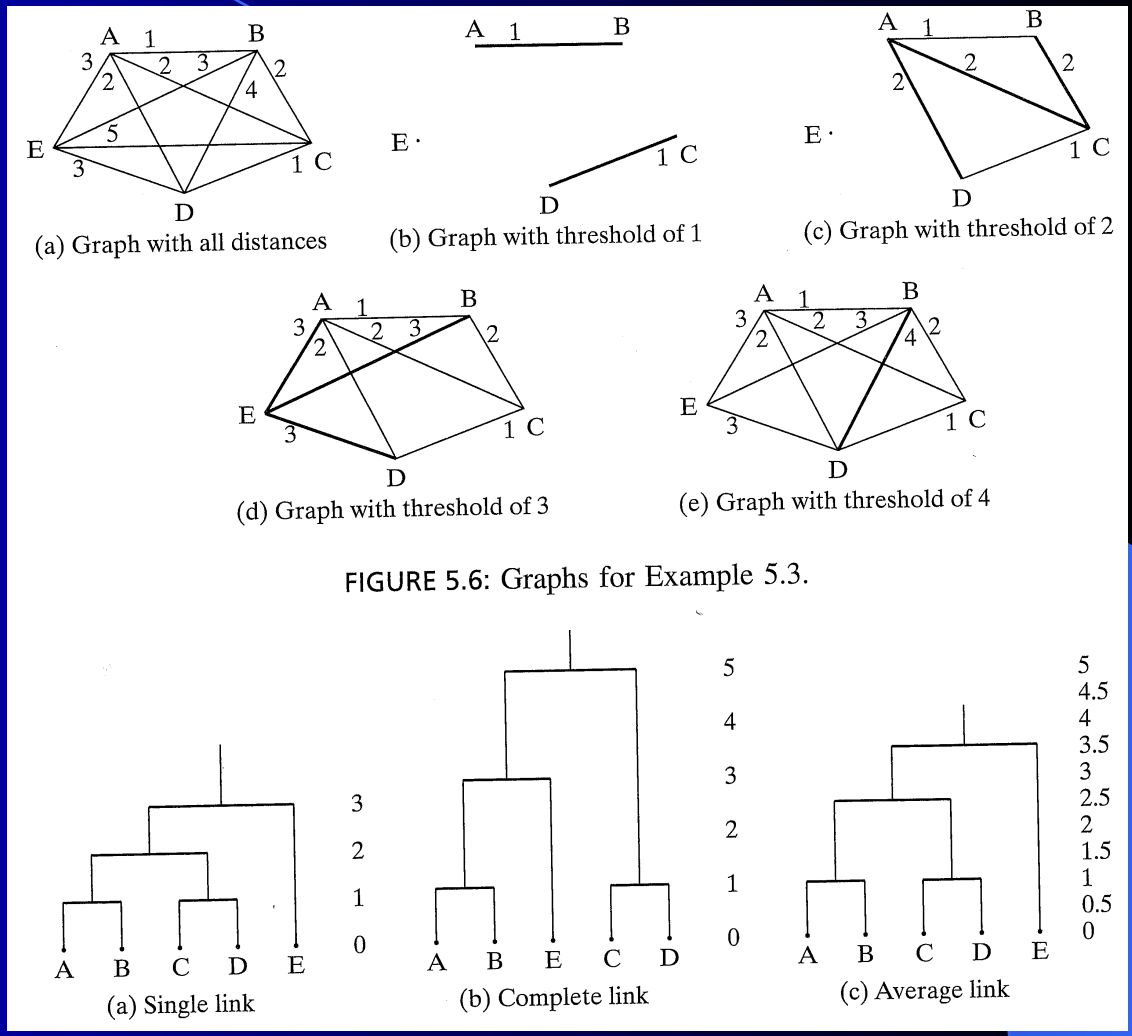
Linkage Methods



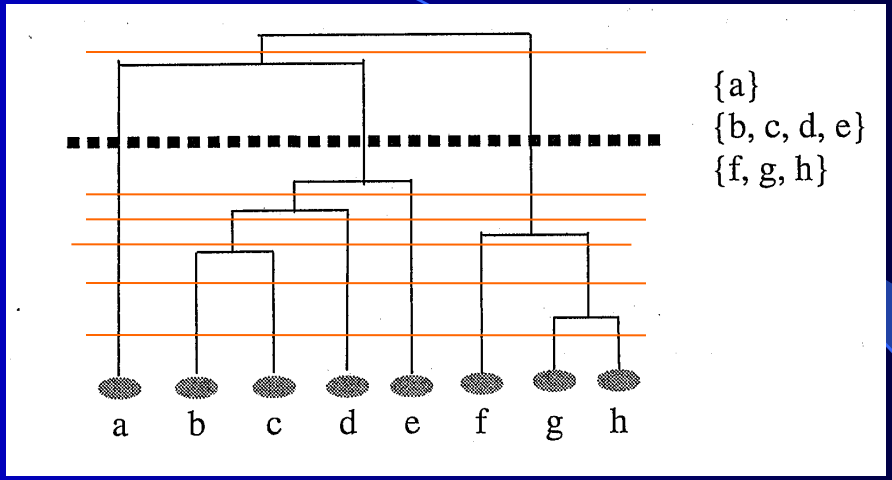
Dendrogram Representation

Item	A	B	C	D	E
A	0	1	2	2	3
B	1	0	2	4	3
C	2	2	0	1	5
D	2	4	1	0	3
E	3	3	5	3	0

- Standard HAC
  - Input is an adjacency matrix
  - Output can include a *dendrogram* which visually shows clusters and merge distance



Which cluster level to choose?

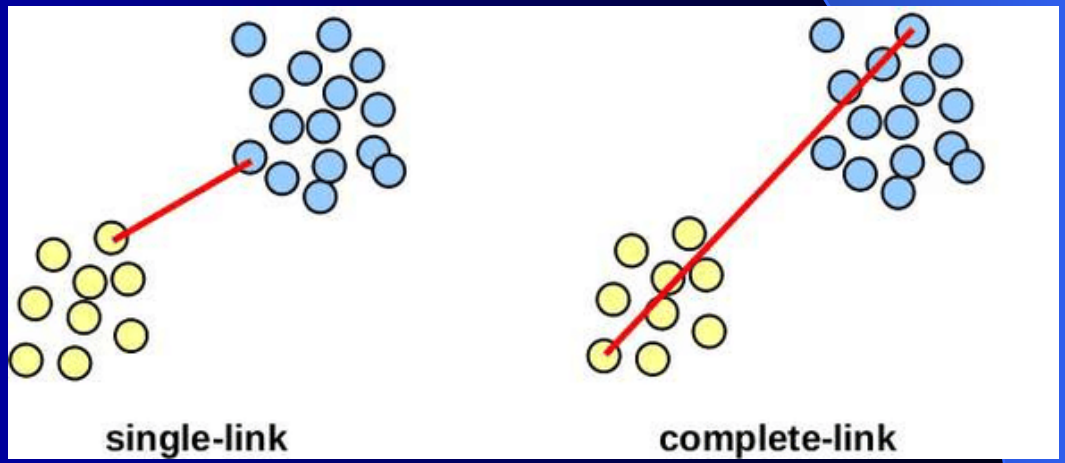


- Depends on goals
  - May know beforehand how many clusters you want - or at least a range (e.g. 2-10)
  - Could analyze the dendrogram and data after the full clustering to decide which sub-clustering level is most appropriate for the task at hand
  - Could use automated *cluster validity* metrics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

HAC *Challenge Question*

- For the data set below show 2 iterations (from 4 clusters until 2 clusters remain) for HAC complete link (furthest).
  - Repeat: Merge the 2 nearest clusters (using complete link distance)
  - Use Manhattan distance
  - Show the dendrogram, including properly labeled distances on the vertical-axis of the dendrogram

Instance	*x*	*y*
a	.8	.7
b	0	0
c	1	1
d	4	4



HAC Homework

- For the data set below show all iterations (from 5 clusters until 1 cluster remaining) for HAC single link.
  - Show work
  - Use Manhattan distance
  - In case of ties go with the cluster containing the least alphabetical instance.
  - Show the dendrogram, including properly labeled distances on the vertical-axis of the dendrogram.

Instance	*x*	*y*
a	.8	.7
b	-.1	.2
c	.9	.8
d	0	.2
e	.2	.1

HAC Summary

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

- Divisive – Starts with all the data in one cluster
  - One approach is to compute the MST (minimum spanning tree - n^2 time since it's a fully connected graph) and then divide the cluster at the tree edge with the largest distance – similar time complexity as HAC, different clusterings obtained
  - Could be more efficient than HAC if we want just a few clusters

CLUSTERING SCORES

Cluster Validity Metrics - Compactness

- One good goal is *compactness* – members of a cluster are all similar and close together
  - One measure of compactness of a cluster is the sum of squares distance from each point to its cluster centroid (aka SSE)
  - where **c** is the centroid of a cluster C, made up of instances X_c. Lower is better.
  - The overall compactness of a particular clustering is the sum of the compactness of the individual clusters
  - Gives us a numeric way to compare different clusterings by seeking clusterings which minimize the compactness metric
- However, for compactness, what clustering is always best?

Cluster Validity Metrics - Separability

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

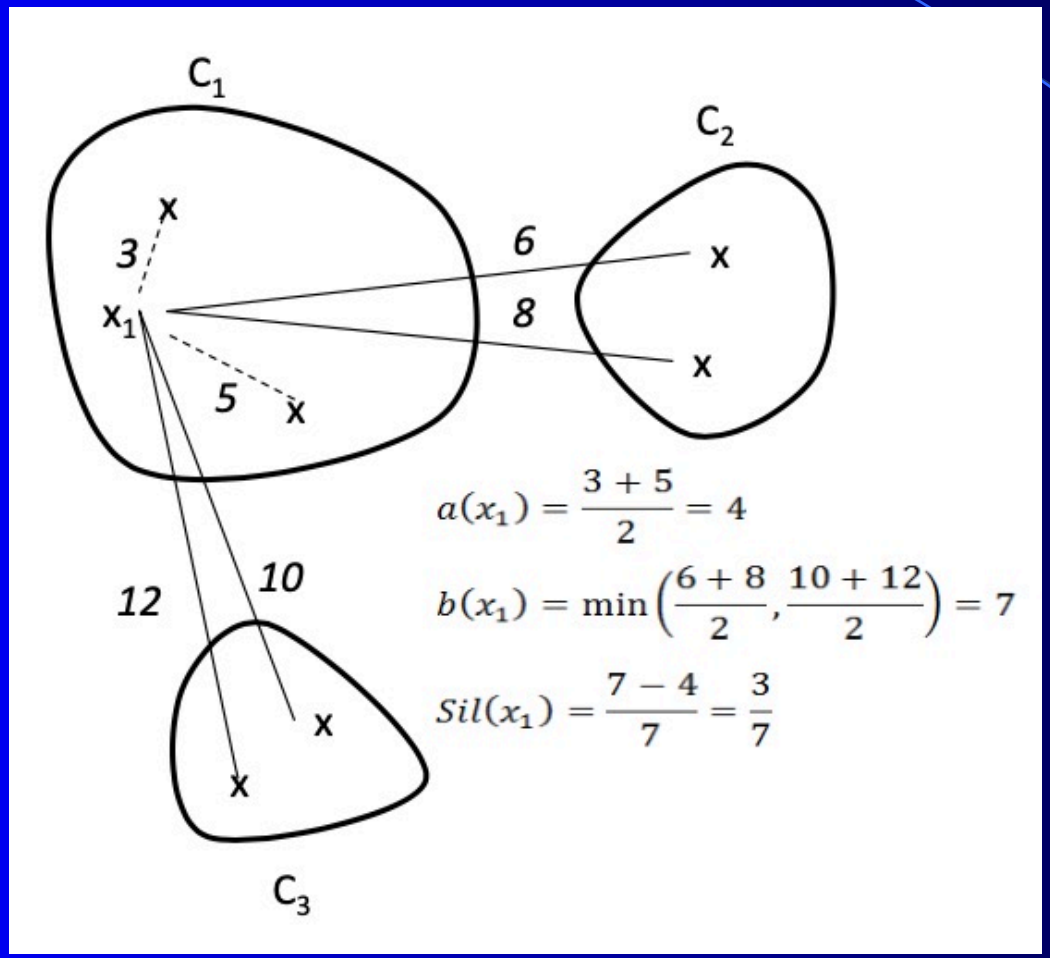
Cluster Validity Metrics - Separability

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

Silhouette

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

Silhouette



$$a(\mathbf{x}_i) = \frac{1}{|C_I| - 1} \sum_{\mathbf{x}_j \in C_I, j \neq i} \text{dist}(\mathbf{x}_i, \mathbf{x}_j)$$

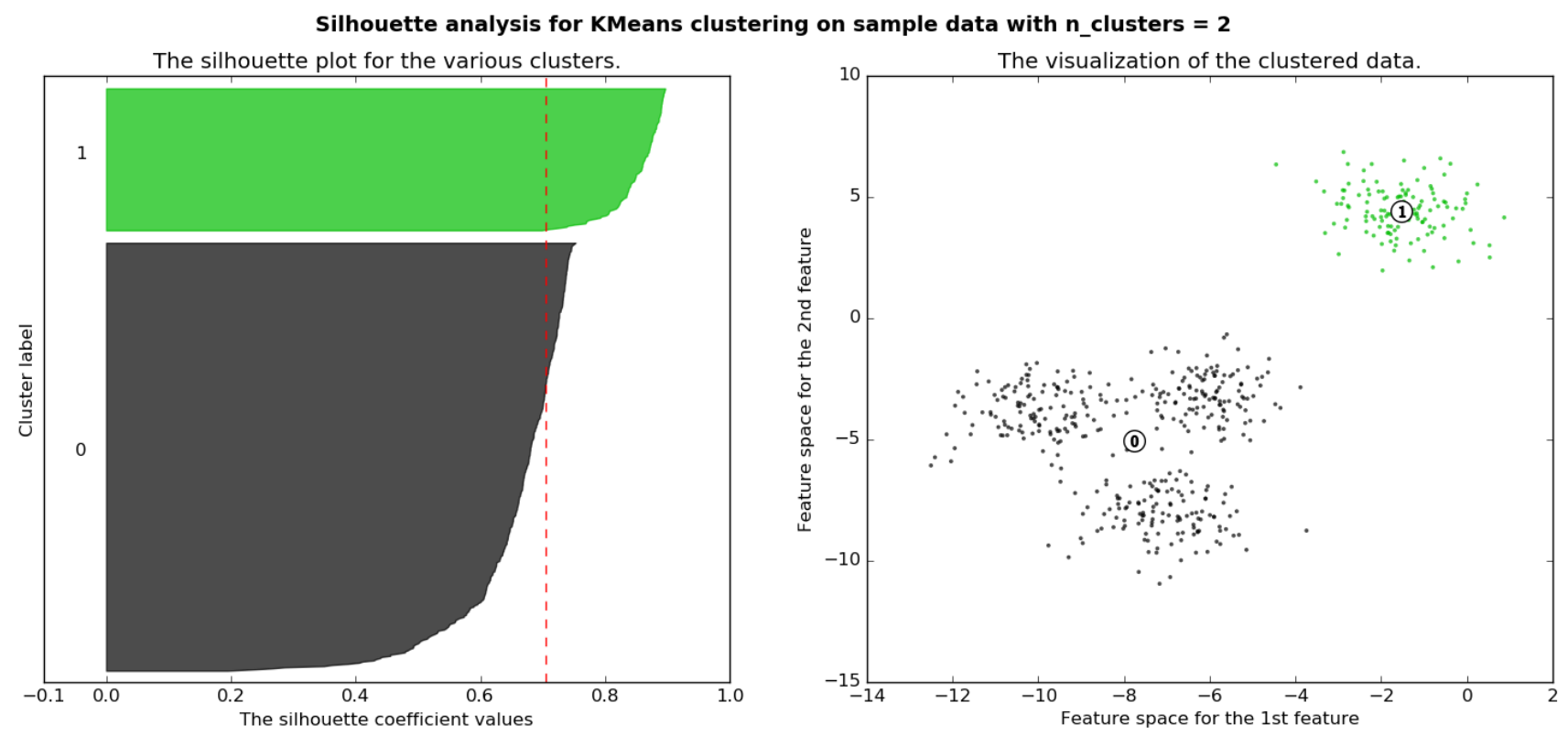
$$b(\mathbf{x}_i) = \min_{J \neq I} \left\{ \frac{1}{|C_J|} \sum_{\mathbf{x}_j \in C_J} \text{dist}(\mathbf{x}_i, \mathbf{x}_j) \right\}$$

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

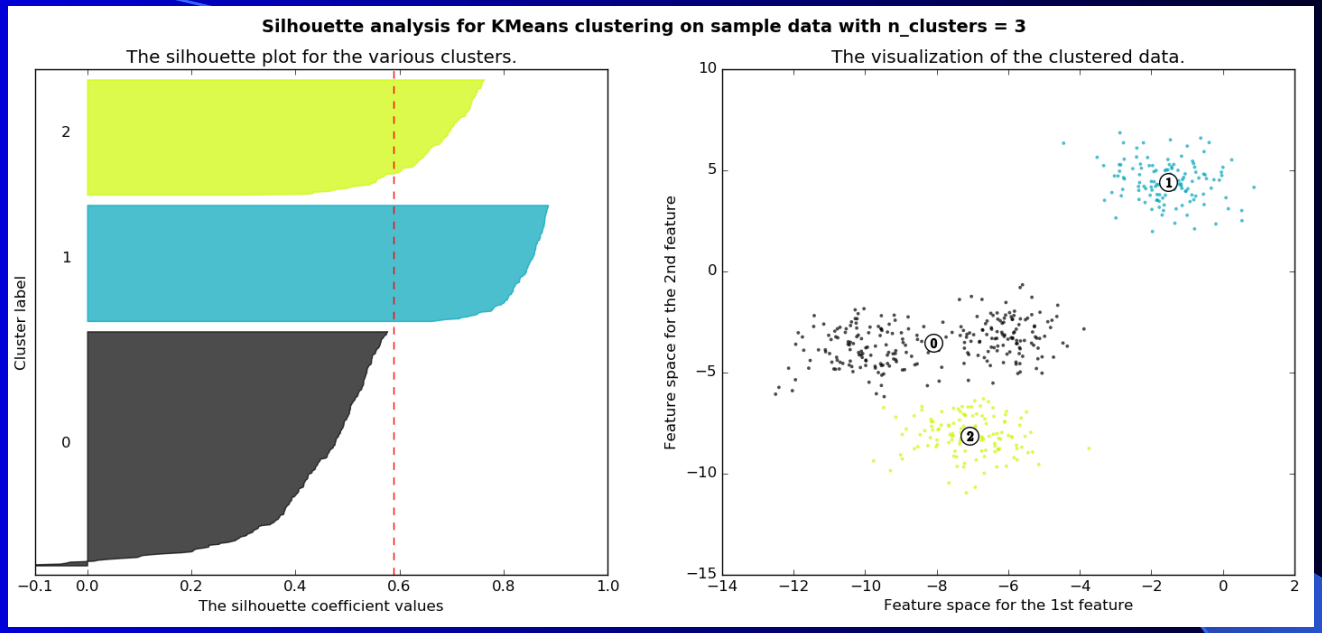
Silhouette

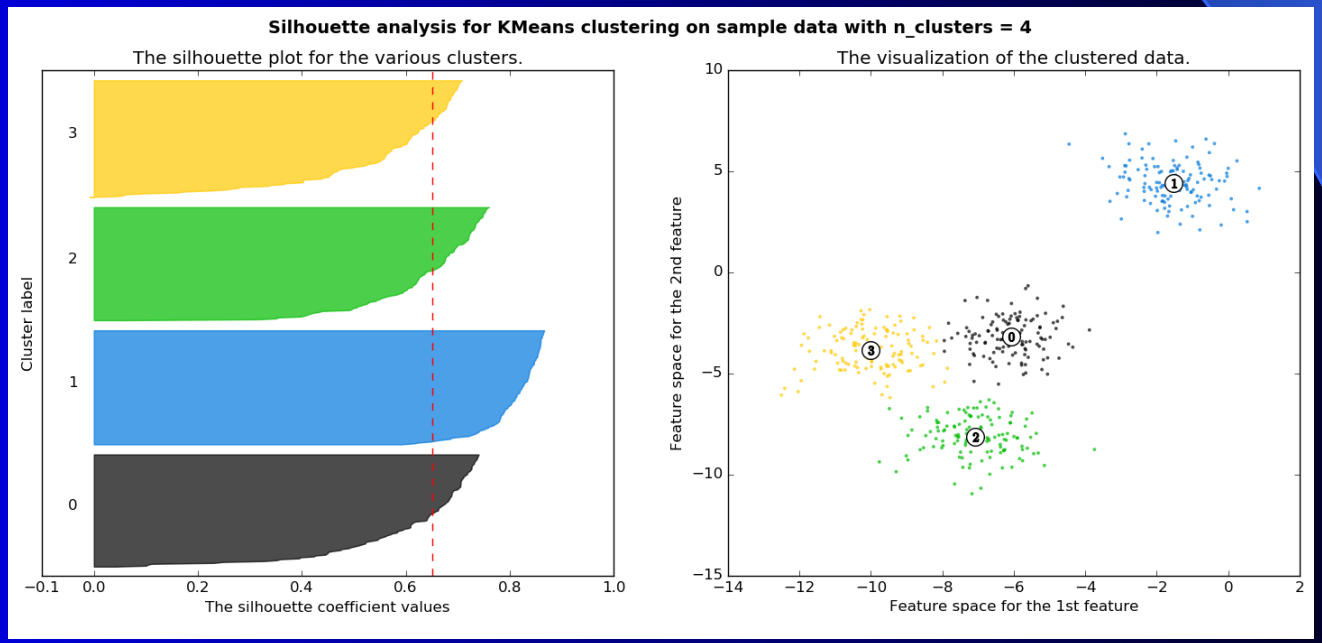
- $s(i)$ is close to one when "within" distance is much smaller than smallest "between" distance
- $s(i)$ is 0 when i is right on the border between two clusters
- $s(i)$ is negative when i probably belongs in another cluster
- By definition, $s(i) = 0$ if it is the only node in the cluster
- The quality of a single cluster can be measured by the average silhouette score of its members, (close to 1 is best)
- The quality of a total clustering can be measured by the average silhouette score of all the instances
- To find best clustering, compare total silhouette scores across clusterings with different numbers of clusters and choose the one with highest silhouette

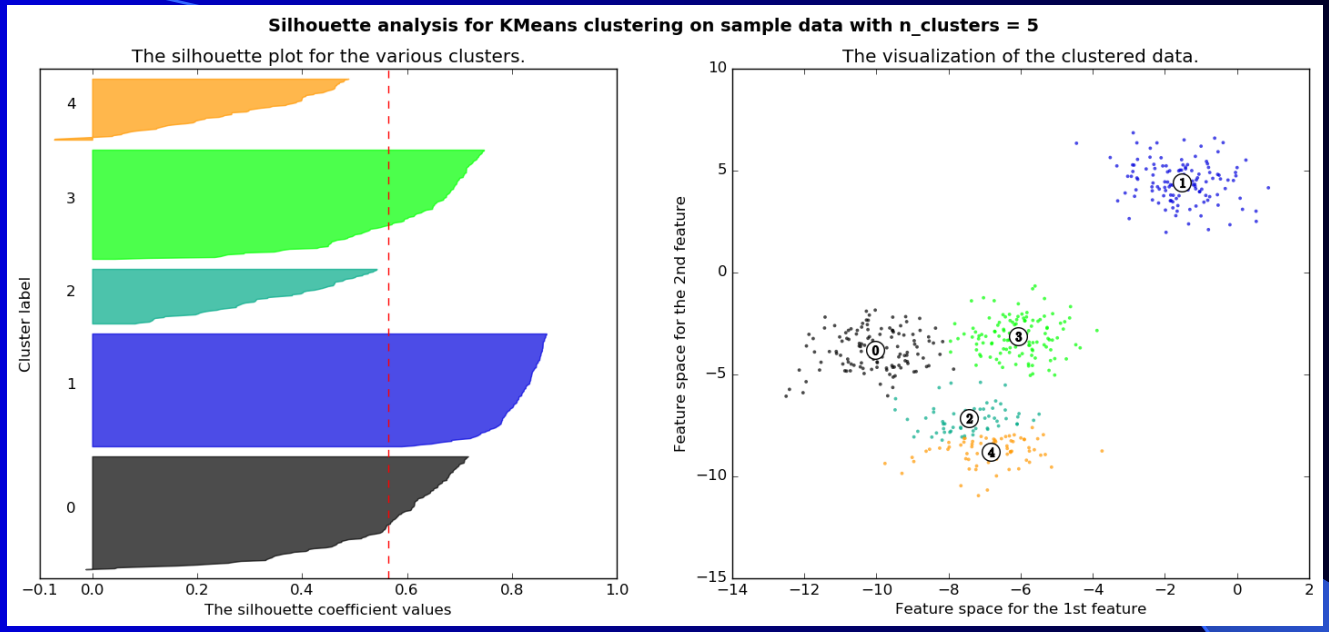
Visualizing Silhouette

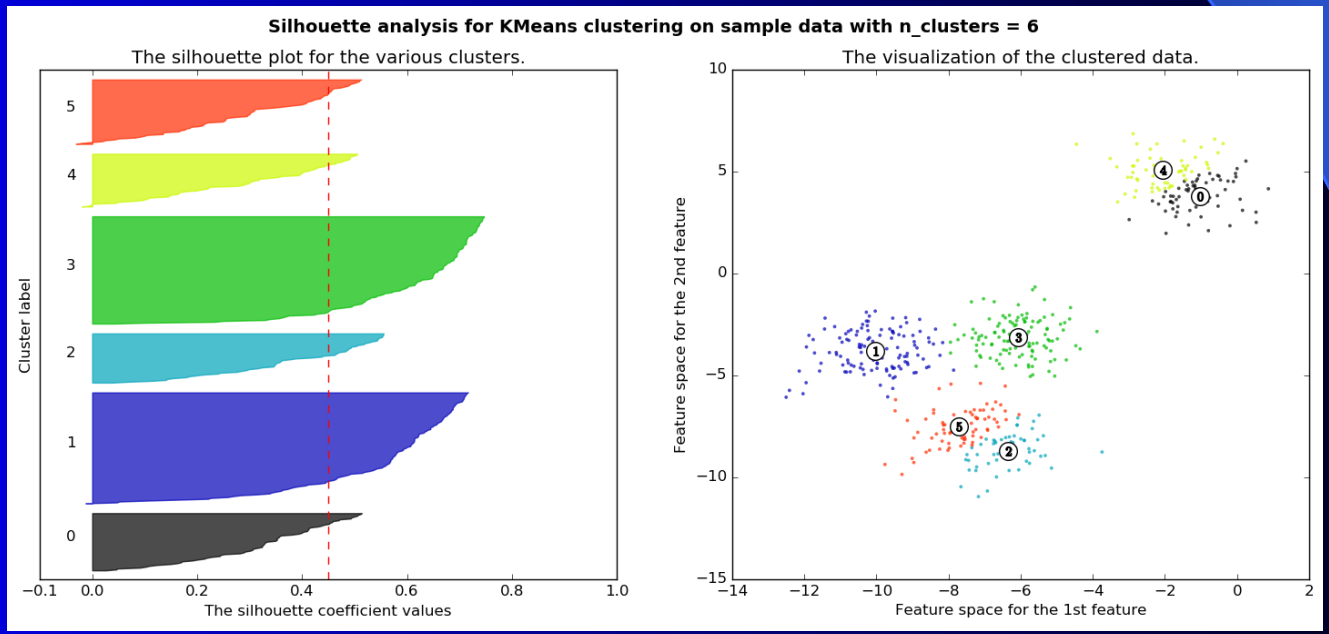


Length is quality of cluster, height is size of cluster
Dashed line is average silhouette score for clustering


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


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


Silhouette analysis for KMeans clustering on sample data with n_clusters = 5
The silhouette plot for the various clusters.
The visualization of the clustered data.
Cluster label
The silhouette coefficient values
Feature space for the 2nd feature
Feature space for the 1st feature


Silhouette analysis for KMeans clustering on sample data with n_clusters = 6
The silhouette plot for the various clusters.
The visualization of the clustered data.
Cluster label
The silhouette coefficient values
Feature space for the 2nd feature
Feature space for the 1st feature

Silhouette

- Best case graph for silhouette?

Silhouette

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

Silhouette

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

Silhouette Homework

- Assume a clustering with {a,b} in cluster 1 and {c,d,e} in cluster 2. What would the Silhouette score be for a) each instance, b) each cluster, and c) the entire clustering. d) Sketch the Silhouette visualization for this clustering. Use Manhattan distance for your distance calculations.

Instance	*x*	*y*
a	.8	.7
b	.9	.8
c	.6	.6
d	0	.2
e	.2	.1

Summary

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