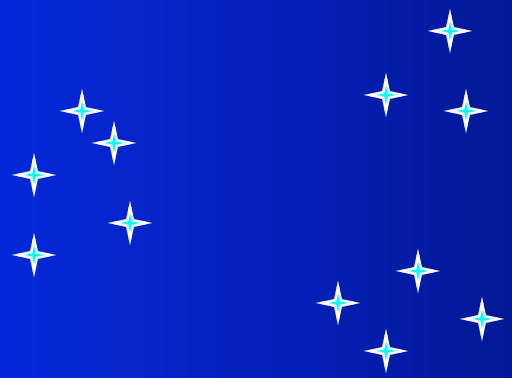
Unsupervised Learning and Clustering

- 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 application
  - Important to normalize the input data

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?

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*

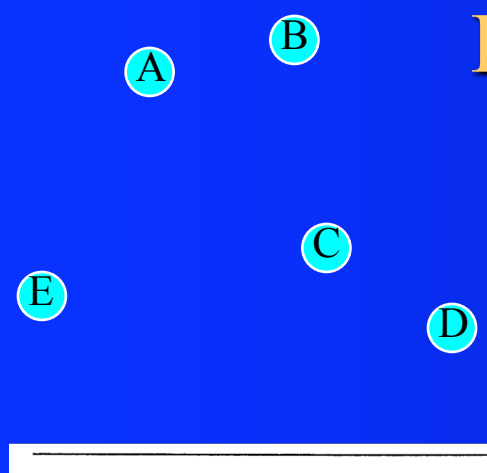
Partitional and Hierarchical Clustering

- Two most common high level approaches
- 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 set/grouping of clusters.
  - Divisive: Start with all instances as one cluster and progressively divide until all instances are their own cluster. You can then decide what level of granularity you want to output.
- With partitional clustering the algorithm creates one clustering (with multiple clusters, usually > 2), typically by minimizing some objective function
  - Note that you could run the partitional algorithm again in a recursive fashion on any or all of the new clusters if you want to build a hierarchy

Hierarchical Agglomerative Clustering (HAC)

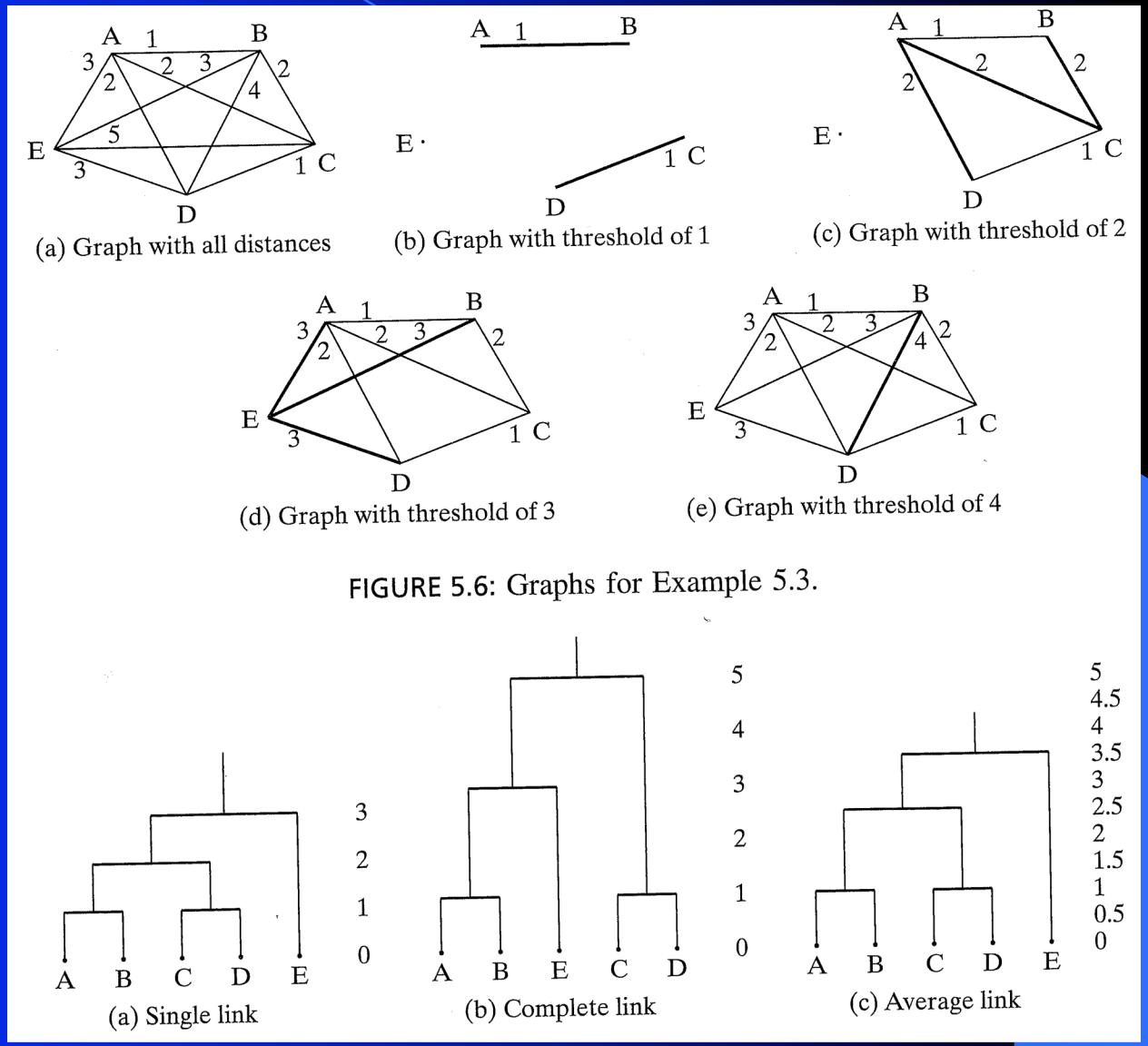
- Input is an $n \times n$ adjacency matrix giving the distance between each pair of instances
- Initialize each instance to be its own cluster
- Repeat until there is just one cluster containing all instances
  - Merge the two "closest" remaining clusters into one cluster
- HAC algorithms vary based on:
  - "Closeness definition", single, complete, or average link common
  - Which clusters to merge if there are distance ties
  - Just do one merge at each iteration, or do all merges that have a similarity value within a threshold which increases at each iteration

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 be a *dendrogram* which visually shows clusters and merge distance



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.
- Single link – (nearest neighbor) can lead to long chained clusters where some points are quite far from each other
- Complete link – (farthest neighbor) finds more compact clusters
- Average link – Used less because have to re-compute the average each time
- 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

HAC *Challenge Question*

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

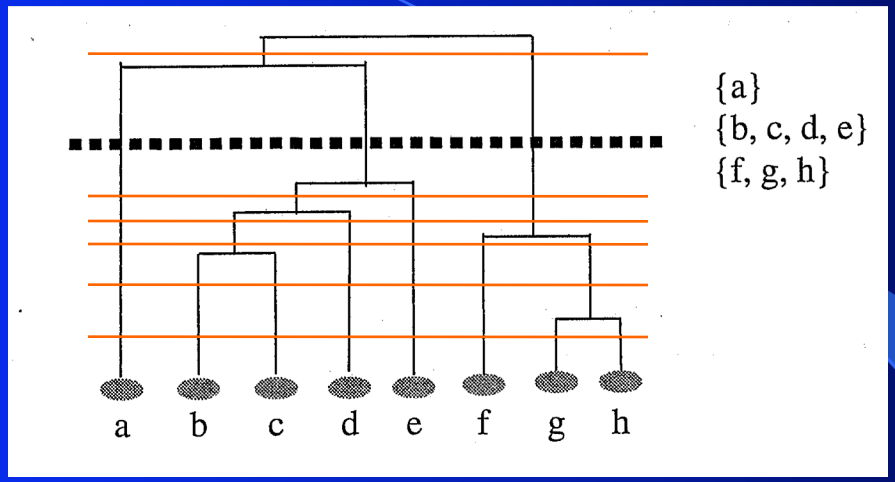
Pattern	*x*	*y*
a	.8	.7
b	0	0
c	1	1
d	4	4

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.

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

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 subclustering level is most appropriate for the task at hand
  - Could use automated *cluster validity* metrics to help
- Could do stopping criteria during clustering

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 SSE of the cluster instances compared to the cluster centroid

$$Comp(C) = \sum_{i=1}^{|X_c|} (\mathbf{c} - \mathbf{x}_i)^2$$

  - where **c** is the centroid of a cluster C, made up of instances X_c. Lower is better.
  - Thus, the overall compactness of a particular clustering is just 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 this metric, 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

$$Separability = \sum_{i=1}^{|C|} \min_j dist_{ij}(\mathbf{c}_i, \mathbf{c}_j)$$

  - We would like to find clusterings where separability is maximized
- However, separability is usually maximized when there are are very few clusters
  - squared distance amplifies larger distances

Silhouette

- Want techniques that find a balance between inter-cluster similarity and intra-cluster dissimilarity
- Silhouette is one good popular approach
- Start with a clustering, using any clustering algorithm, which has k unique clusters
- $a(i)$ = 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 (comparing each different cluster) average dissimilarity of instance i to all instances in that cluster – 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

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

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

$$-1 \le s(i) \le 1$$

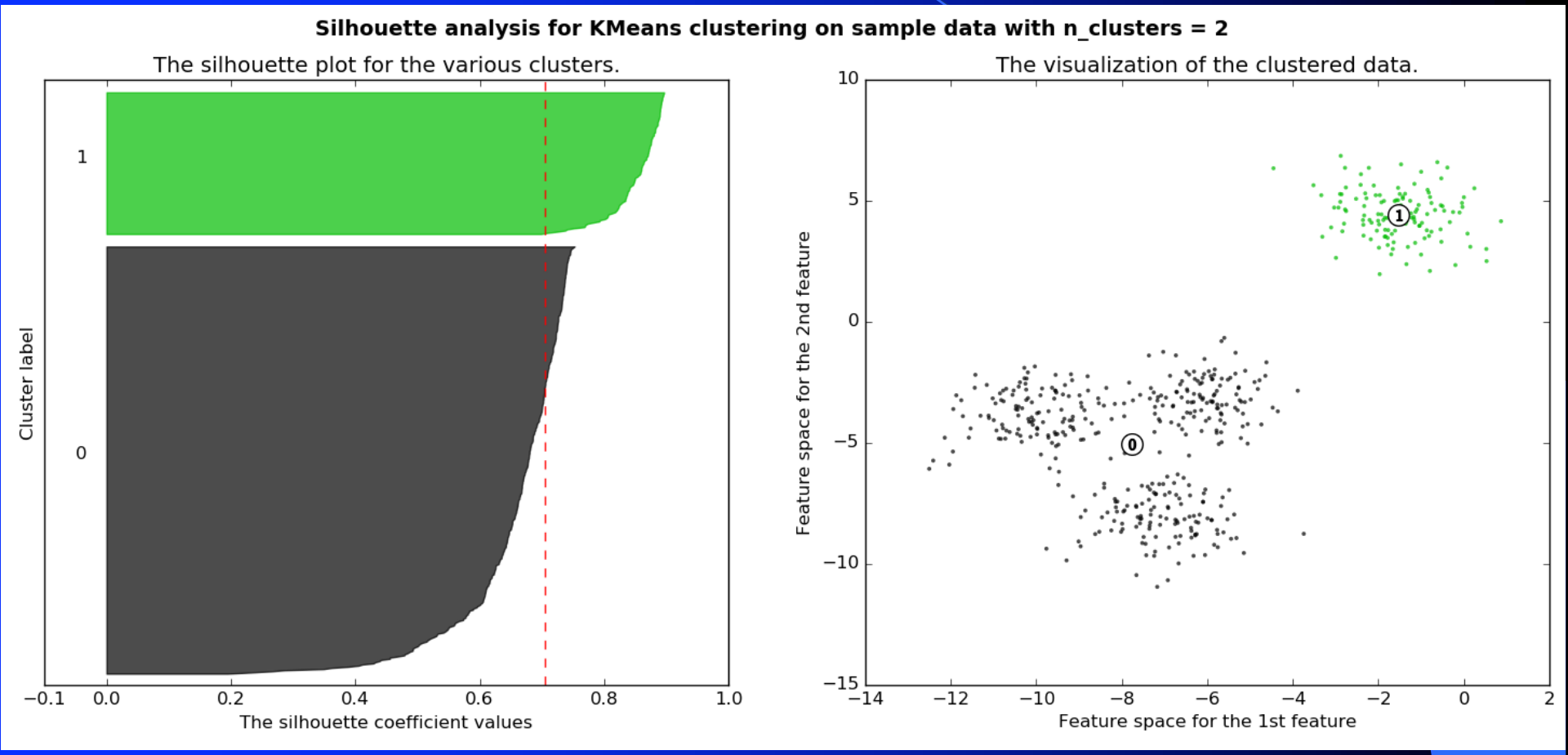
- $s(i)$ is close to one when "within" similarity is much smaller than smallest "between" similarity
- $s(i)$ is 0 when i is right on the border between two clusters
- $s(i)$ is negative when i really belongs in another cluster
- By definition, $s(i) = 0$ if it is the only node in the cluster
- The quality 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 k values and choose the highest

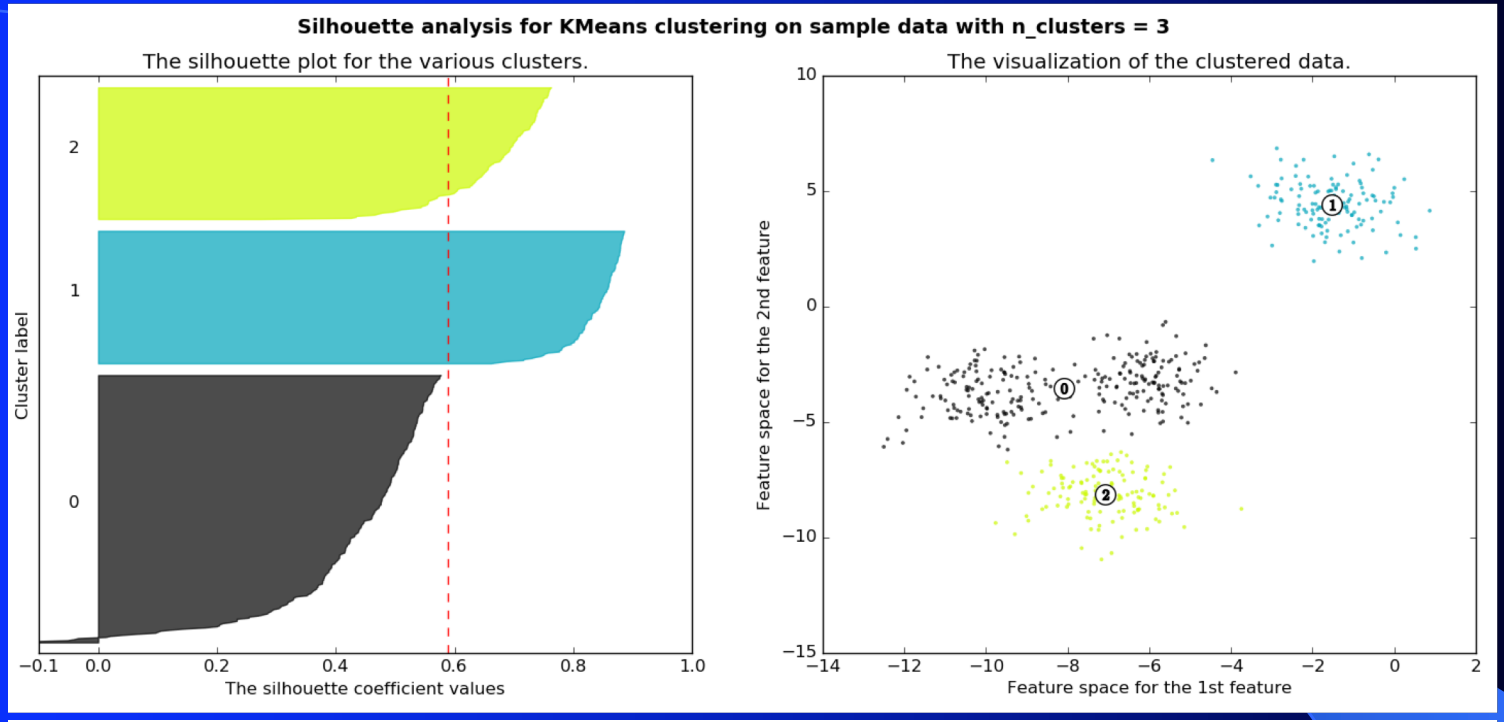
Silhouette Homework

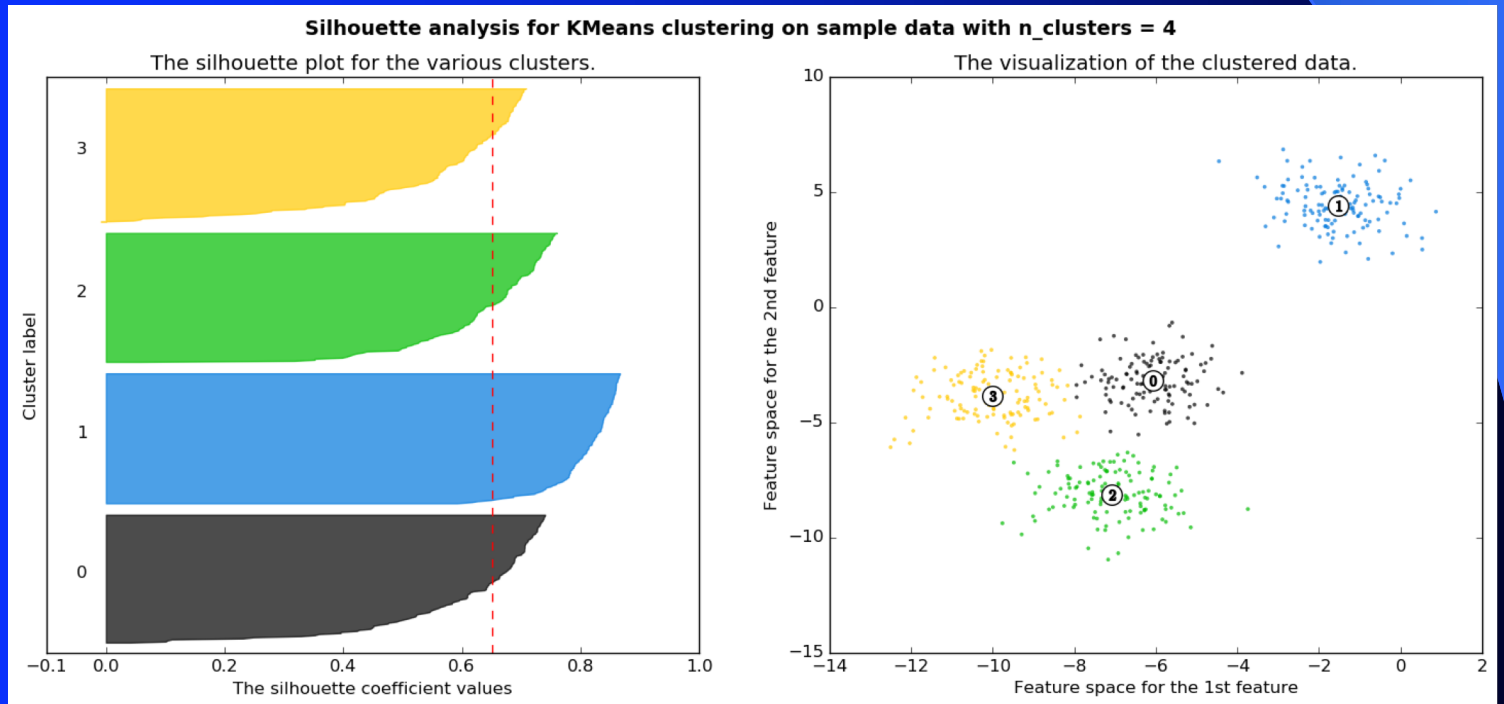
- Assume a clustering with {a,b} in cluster 1 and {c,d,e} in cluster 2. What would the Silhouette score be for a) each instance, b) each cluster, and c) the entire clustering. d) Sketch the Silhouette visualization for this clustering.

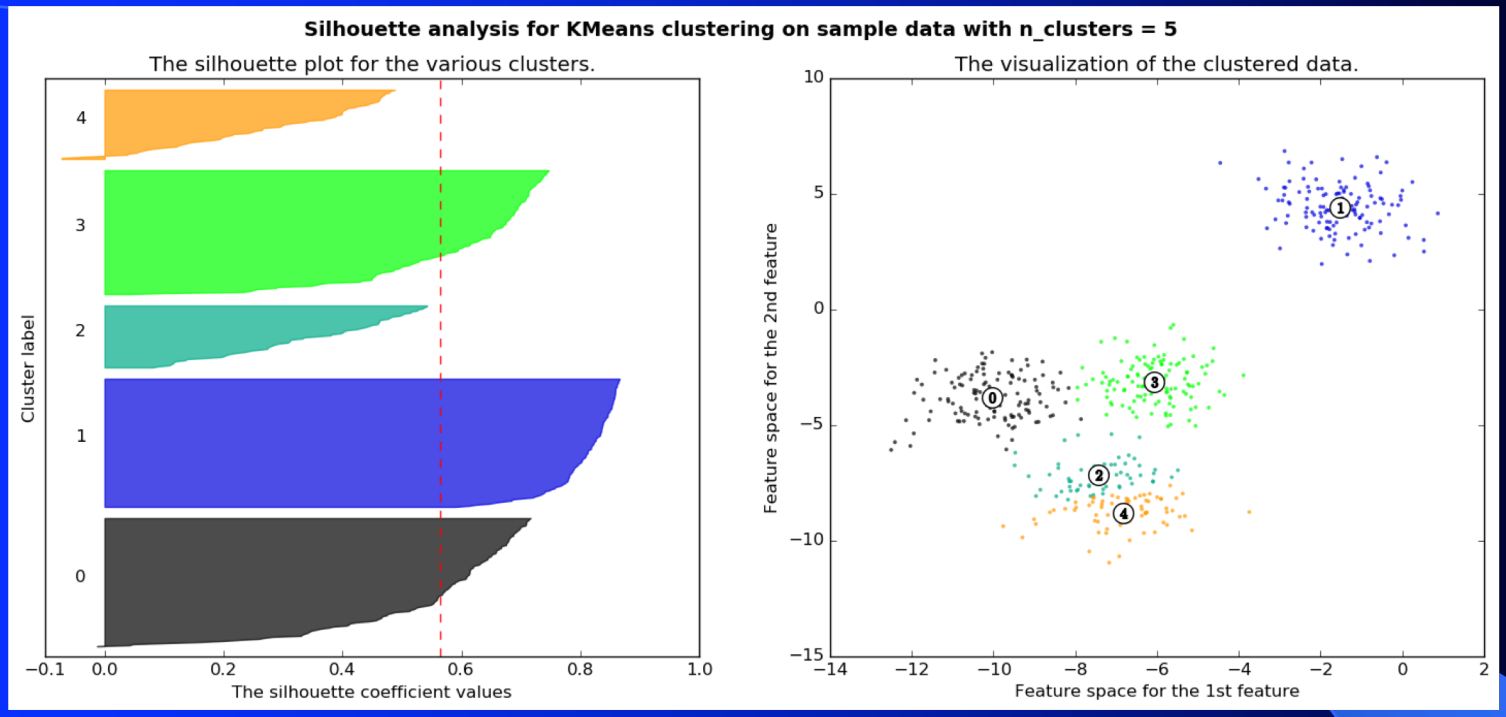
Pattern	*x*	*y*
a	.8	.7
b	.9	.8
c	.6	.6
d	0	.2
e	.2	.1

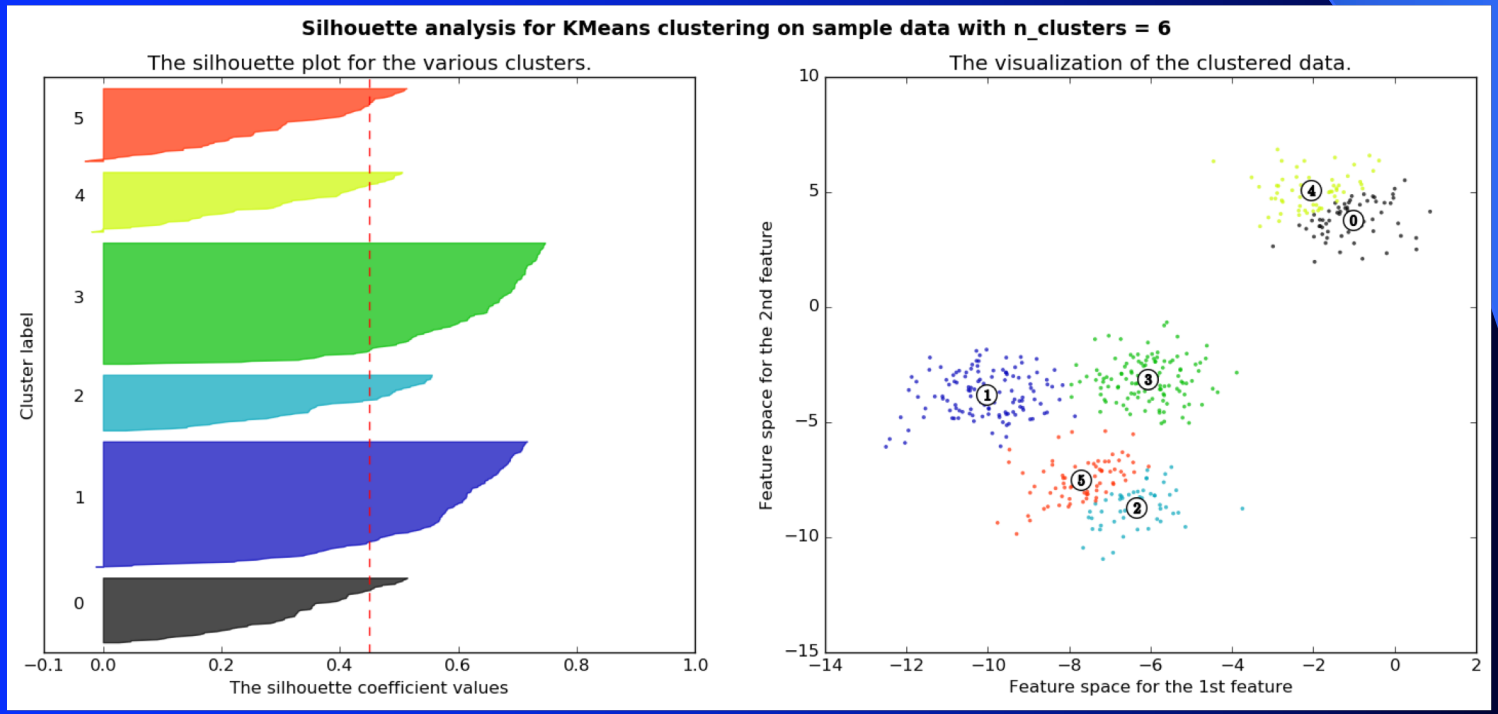
Visualizing Silhouette




Silhouette analysis for KMeans clustering on sample data with n_clusters = 3
The silhouette plot for the various clusters.
Cluster label
The silhouette coefficient values
The visualization of the clustered data.
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.
Cluster label
The silhouette coefficient values
The visualization of the clustered data.
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.
Cluster label
The silhouette coefficient values
The visualization of the clustered data.
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.
Cluster label
The silhouette coefficient values
The visualization of the clustered data.
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 wide – 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

- Can just use total silhouette average to decide best clustering but best to do silhouette analysis with a visualization tool and use average along with other aspects of the clustering
  - Cluster sizes
  - Number of clusters
  - Shape of clusters
  - Etc.
- Note when task dimensions 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"

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 to help in the decision

1. Randomly choose *k* instances from the data set to be the initial *k* centroids
2. Repeat until no (or negligible) more changes occur
   a) Group each instance with its closest centroid
   b) Recalculate the centroid based on its new cluster

- 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

- Could use Silhouette or other metric to choose best clustering
- 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.
- Can do further refinement of HAC results using any *k* centroids from HAC as starting centroids for *k*-means

k-means Homework

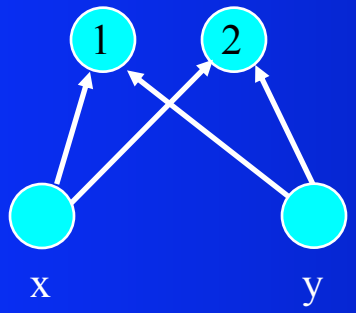
- For the data below, show the centroid values and which instances are closest to each centroid *after* centroid calculation for two iterations of *k*-means using Manhattan distance
- By 2 iterations I mean 2 centroid changes after the initial centroids
- Assume $k = 2$ and that the first two instances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

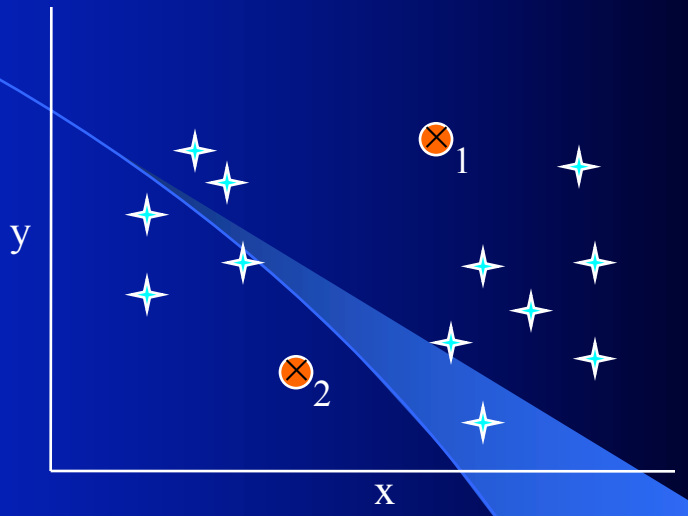
Pattern	*x*	*y*
a	.9	.8
b	.2	.2
c	.7	.6
d	-.1	-.6
e	.5	.5

Clustering Project

- Last individual project
- See Content section
- 15 points extra credit available if implement both HAC and *k*-means

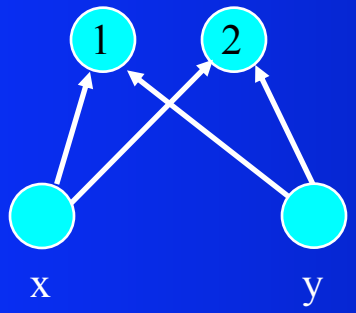
Neural Network Clustering

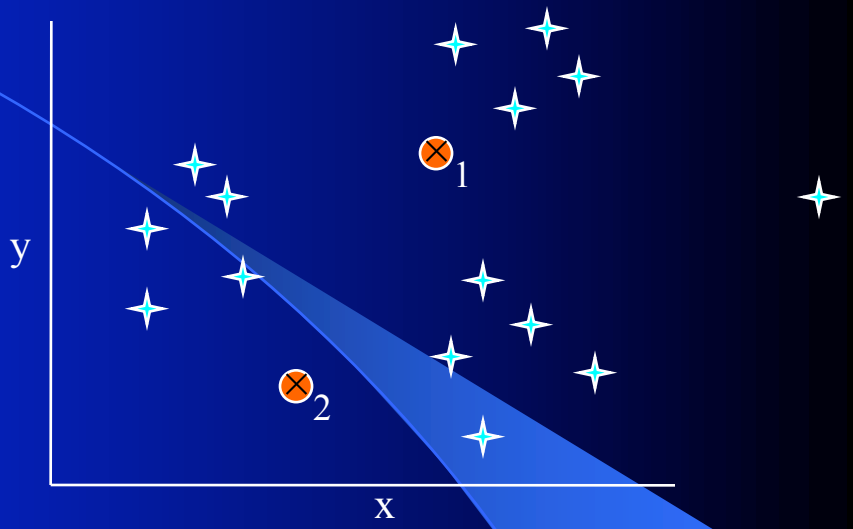




- Single layer network
  - Bit like a chopped off RBF, where prototypes become adaptive output nodes
- Arbitrary number of output nodes (cluster prototypes) – User defined
- Locations of output nodes (prototypes) can be initialized randomly
  - Could set them at locations of random instances, etc.
- Each node computes distance to the current instance
- Competitive Learning style – winner takes all – closest node decides the cluster during execution
- Closest node is also the node which usually adjusts during learning
- Node adjusts slightly (learning rate) towards the current example

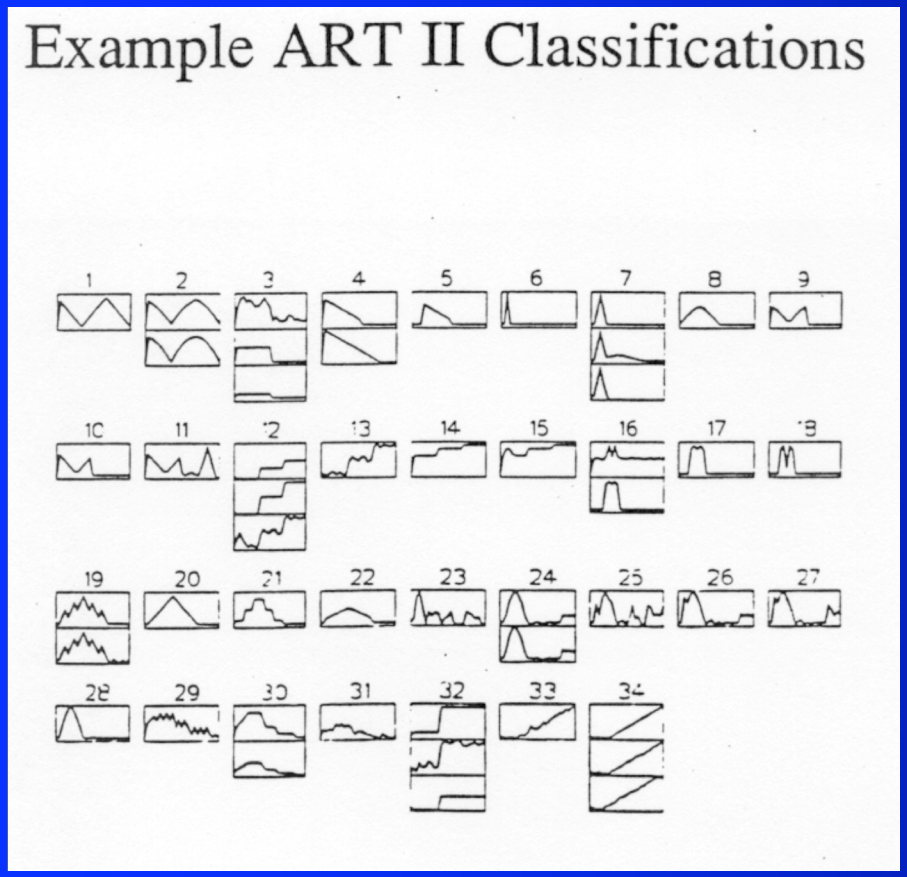
Neural Network Clustering

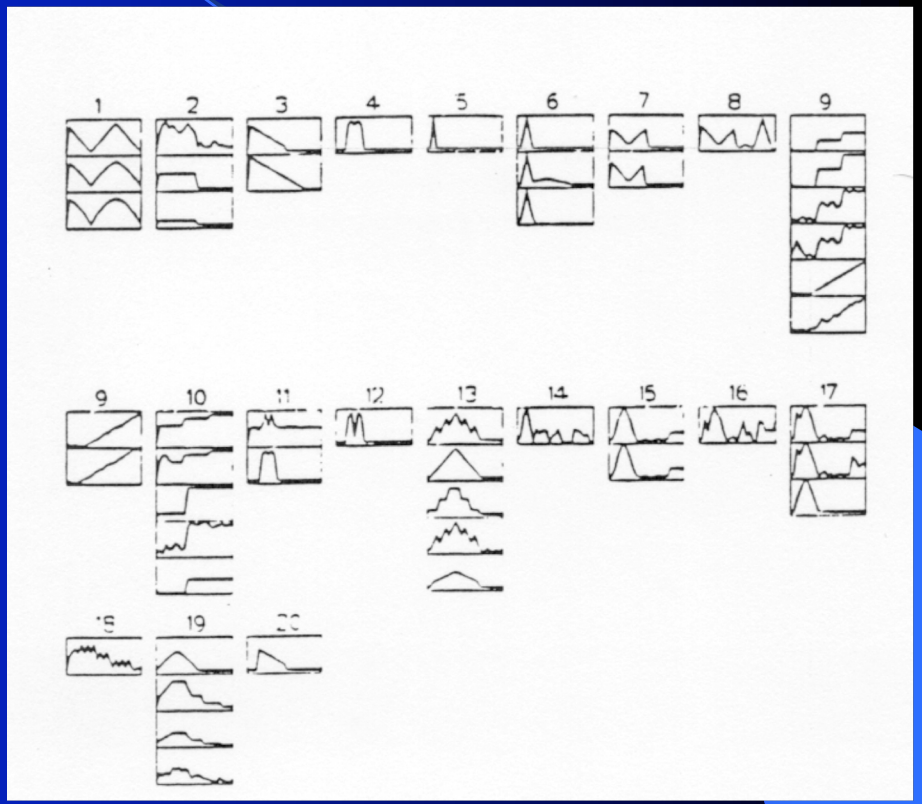




- What would happen in this situation?
- Could start with more nodes than probably needed and drop those that end up representing none or few instances
  - Could start them all in one spot – However…
- Could dynamically add/delete nodes
  - Local vigilance threshold
  - Global vs local vigilance
  - Outliers

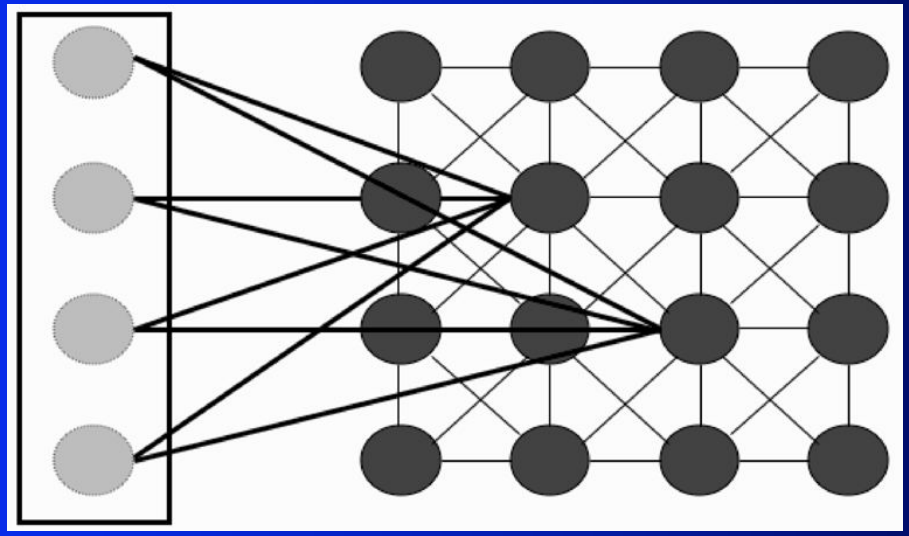
Example Clusterings with Vigilance





Self-Organizing Maps

- Output nodes which are close to each other represent similar classes – Biological plausibility
- Neighbors of winning node also update in the same direction (scaled by a learning rate), as the winner
- Self organizes to a topological class map (e.g. vowel sounds)
  - Can interpolate, *k* value less critical, different 2 or 3-dimensional topoligies



Other Unsupervised Models

- Vector Quantization – Discretize into codebooks
- K-medoids
- Conceptual Clustering (Symbolic AI) – Cobweb, Classit, etc.
  - Incremental vs Batch
- Density mixtures
- Interactive clustering
- Special models for large data bases – n^2 space?, disk I/O
  - Sampling – Bring in enough data to fill memory and then cluster
  - Once initial prototypes found, can iteratively bring in more data to adjust/fine-tune the prototypes as desired
  - Linear algorithms

Association Analysis – Link Analysis

- Used to discover relationships/rules in large databases
- Relationships represented as *association rules*
  - Unsupervised learning, can give significant business advantages, and also good for many other large data areas: astronomy, etc.
- One example is *market basket analysis* which seeks to understand more about what items are bought together
  - This can then lead to improved approaches for advertising, product placement, etc.
  - Example Association Rule: {Cereal} $\Rightarrow$ {Milk}

Transaction ID and Info	Items Bought
1 and (who, when, etc.)	{Ice cream, milk, eggs, cereal}
2	{Ice cream}
3	{milk, cereal, sugar}
4	{eggs, yogurt, sugar}
5	{Ice cream, milk, cereal}

Summary

- Can 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